explainerdashboard Documentation

Release 0.2

Oege Dijk

Aug 01, 2023

CONTENTS

CHAPTER

ONE

SUMMARY

`explainerdashboard` is a library for quickly building interactive dashboards for analyzing and explaining the predictions and workings of (scikit-learn compatible) machine learning models, including xgboost, catboost and lightgbm. This makes your model transparant and explainable with just two lines of code.

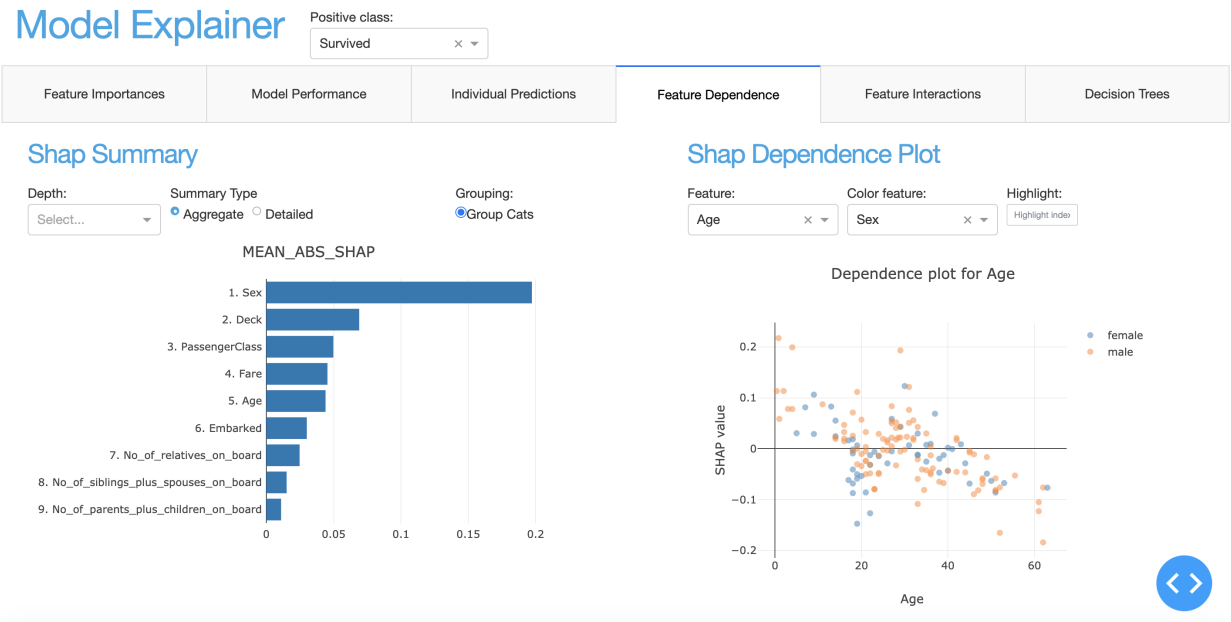
It allows you to investigate SHAP values, permutation importances, interaction effects, partial dependence plots, all kinds of performance plots, and even individual decision trees inside a random forest. With `explainerdashboard` any data scientist can create an interactive explainable AI web app in minutes, without having to know anything about web development or deployment.

You first construct an `explainer` object out of your model and the test data:

```
from explainerdashboard import ClassifierExplainer, ExplainerDashboard
explainer = ClassifierExplainer(model, X_test, y_test)
```

You then pass this `explainer` object to an `ExplainerDashboard` and run it:

```
ExplainerDashboard(explainer).run()
```



You can host multiple ExplainerDashboard in an `ExplainerHub` by passing in a list of dashboards:

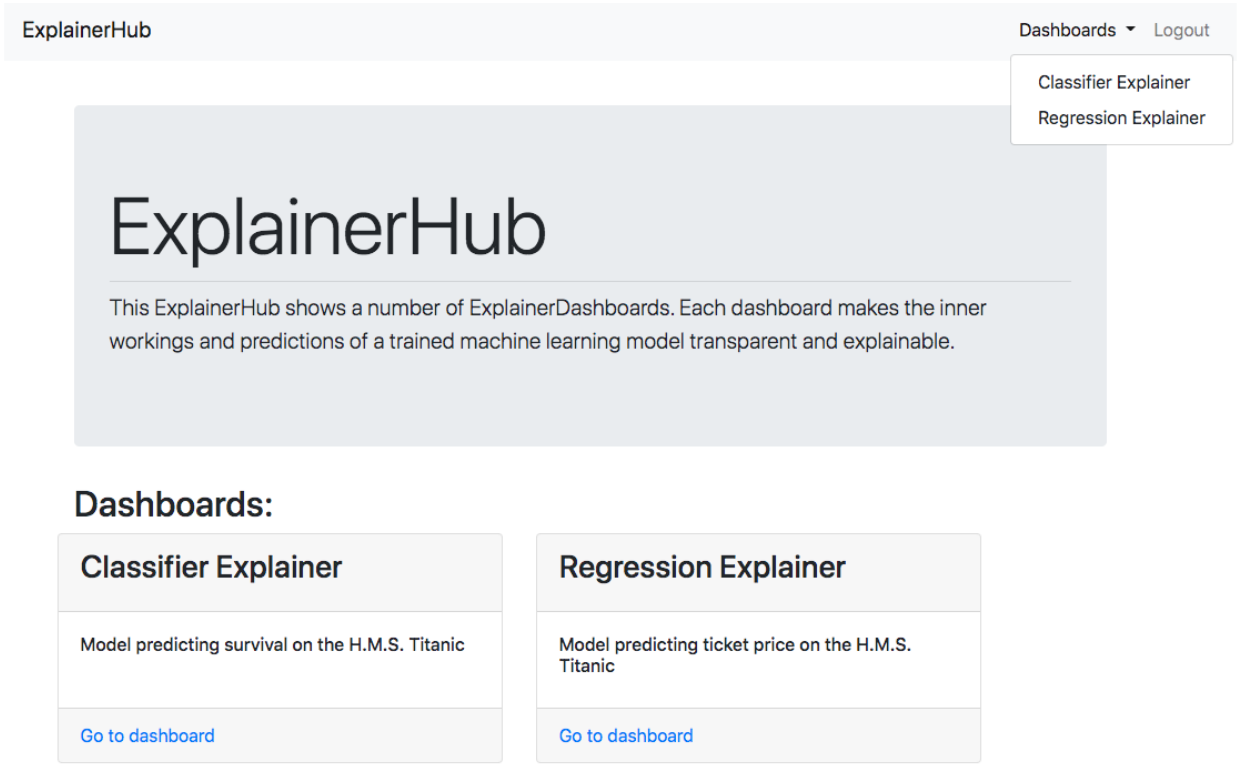
```
db1 = ExplainerDashboard(explainer1)
db2 = ExplainerDashboard(explainer2)
```

(continues on next page)

(continued from previous page)

```
hub = ExplainerHub([db1, db2])
hub.run()
```

Each dashboard is hosted on it's own url path (e.g. `localhost:8050/dashboard1`), and a front-end dashboard with links and descriptions for every dashboard is hosted at e.g. `localhost:8050`:



See *ExplainerHub documentation*

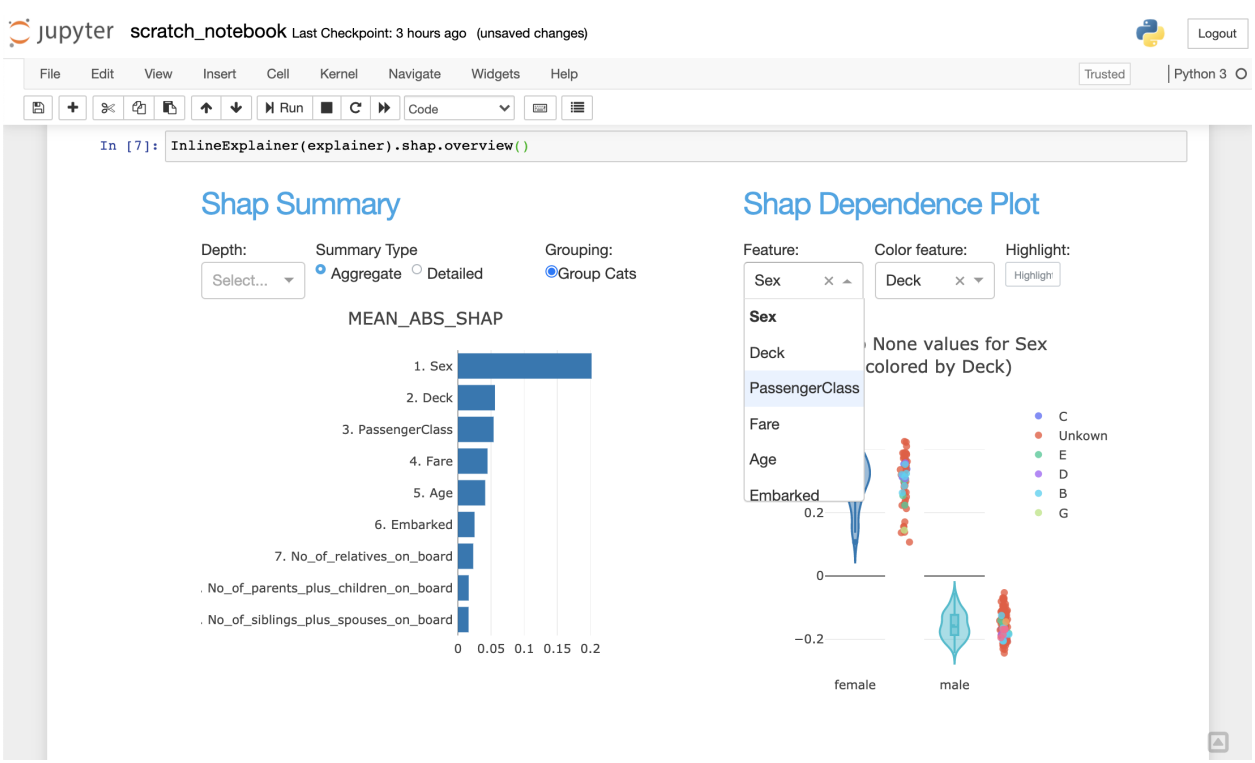
CHAPTER

TWO

INLINEEXPLAINER

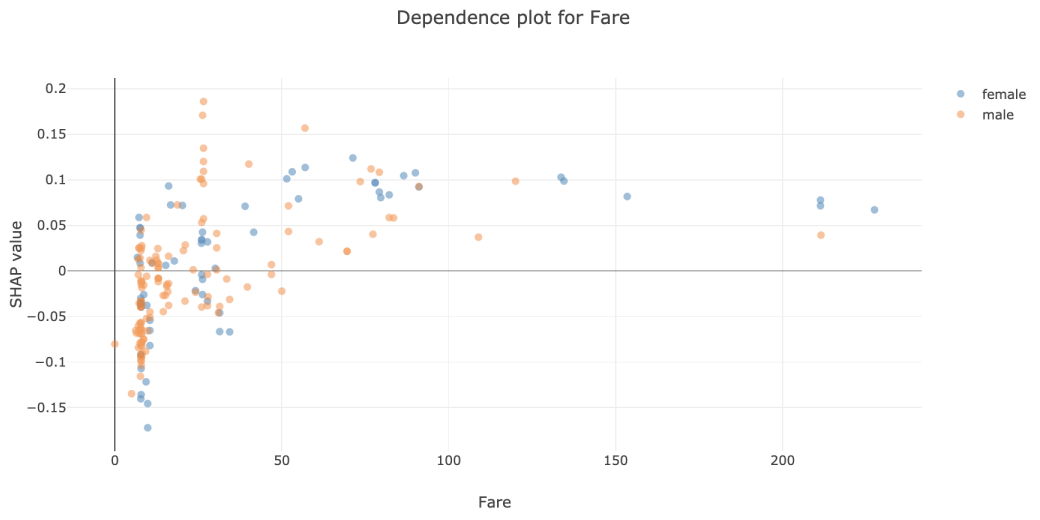
For viewing and customizing individual components or tabs directly inside your notebook you use the `InlineExplainer`:

```
from explainerdashboard import InlineExplainer

InlineExplainer(explainer).shap.dependence()
InlineExplainer(explainer).shap.dependence(hide_cats=True, hide_index=True, col="Fare")
InlineExplainer(explainer).shap.overview()
InlineExplainer(explainer).tab.importances()
```



The `explainer` object itself is also a plot factory that you can use to directly make plots inline in your notebook:

```
In [29]: explainer.plot_shap_dependence("Fare", color_col="Sex")
```



CHAPTER

THREE

A MORE EXTENDED EXAMPLE

Some example code, where we load some data, fit a model, construct an explainer, pass it on to an ExplainerDashboard and run the dashboard:

```
from sklearn.ensemble import RandomForestClassifier

from explainerdashboard import ClassifierExplainer, ExplainerDashboard
from explainerdashboard.datasets import titanic_survive

X_train, y_train, X_test, y_test = titanic_survive()

model = RandomForestClassifier(n_estimators=50, max_depth=5)
model.fit(X_train, y_train)

explainer = ClassifierExplainer(
                model, X_test, y_test,
                # optional:
                cats=['Sex', 'Deck', 'Embarked'],
                labels=['Not survived', 'Survived'])

db = ExplainerDashboard(explainer, title="Titanic Explainer",
                    whatif=False, # you can switch off tabs with bools
                    shap_interaction=False,
                    decision_trees=False)
db.run(port=8051)
```

Or, as a one-liner:

```
ExplainerDashboard(
    ClassifierExplainer(
        RandomForestClassifier().fit(X_train, y_train),
        X_test, y_test
    )
).run()
```

The result of the lines above can be seen in the screenshot above or can be viewed on this example dashboard deployed to heroku.

For a more simple single tab dashboard try:

```
ExplainerDashboard(explainer, simple=True).run()
```

CHAPTER

FOUR

CUSTOM DASHBOARDS

You can easily *remix and customize* `ExplainerComponent` primitives into your own custom layouts for a dashboard that is specifically tailored to your own model and project. For example a dashboard with a single SHAP dependence component:

```
from explainerdashboard.custom import *

class CustomDashboard(ExplainerComponent):
    def __init__(self, explainer, name=None):
        super().__init__(explainer, name=name)
        self.dependence = ShapDependenceComponent(explainer, name=self.name+"dep",
                hide_selector=True, hide_cats=True, hide_index=True, col="Fare")

    def layout(self):
        return html.Div([self.dependence.layout()])

ExplainerDashboard(explainer, CustomDashboard).run()
```

A more elaborate example of *a custom dashboard* (example deployed here):

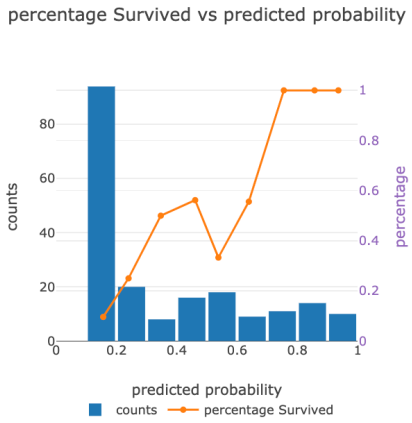
Titanic Explainer

Model | Predictions

Model Performance

As you can see on the right, the model performs quite well.
The higher the predicted probability of survival predicted by the model on the basis of learning from examples in the training set, the higher is the actual percentage for a person surviving in the test set

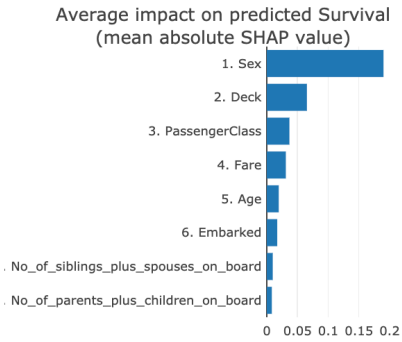
Model Precision Plot



Feature Importances Plot

Summary Type

Aggregate | Detailed



Feature importances

On the left you can check out for yourself which parameters were the most important.
Sex was the most important, followed by Deck and PassengerClass.
If you select 'detailed' you can see the impact of that variable on each individual prediction.
With 'aggregate' you see the average impact size of that variable on the finale prediction.
With the detailed view you can clearly see that the the large impact from Sex stems both from males having a much lower chance of survival and females a much higher chance.

Relations between features and model output

In the plot to the right you can see that the higher the priaceof the Fare that people paid, the higher the chance of survival.
Probably the people with more expensive tickets were in higher up cabins, and were more likely to make it to a lifeboat.
When you color the impacts by the PassengerClass, you can clearly see that the more expensive tickets were mostly 1st class, and the cheaper tickets mostly 3rd class.
On the right you can check out for yourself how different features impact the model output.

Feature impact plot

Feature: Fare | Color feature: Passenger...

More examples of how to start dashboards for different types of models and with different parameters can be found in the dashboard_examples notebook in the github repo.

For examples on how to interact with and get plots and dataframes out of the explainer object check out explainer_examples notebook in the github repo.

4.1 Explainers

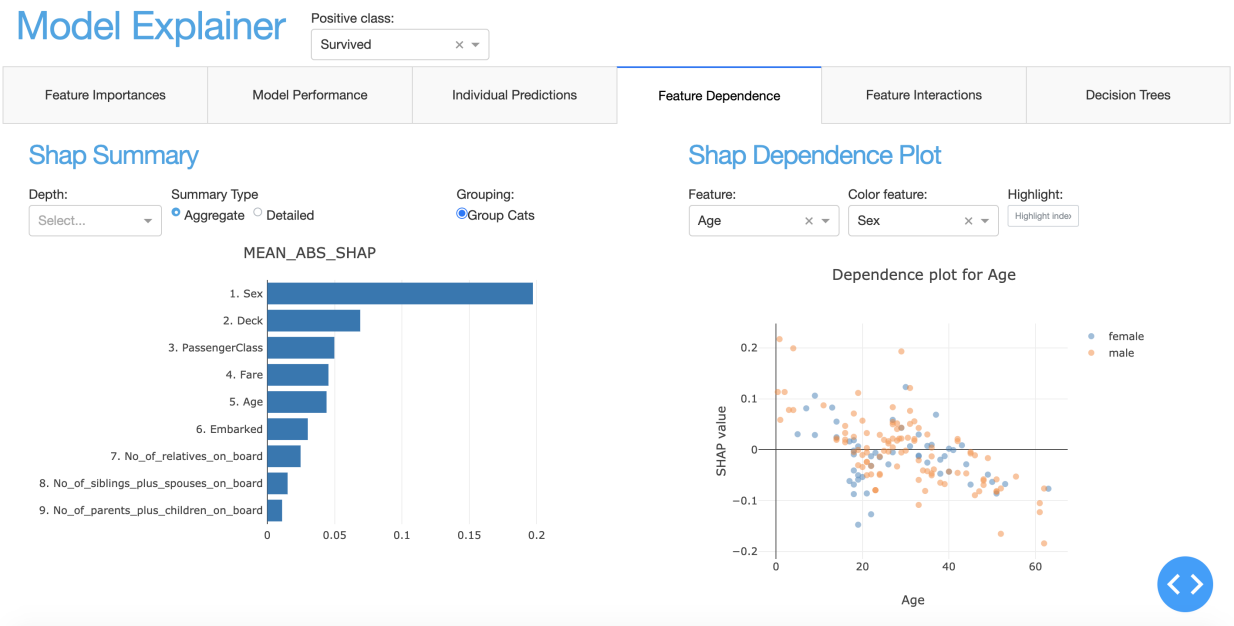
4.1.1 Simple example

In order to start an `ExplainerDashboard` you first need to construct an `Explainer` instance. They come in two flavours and at its most basic they only need a model, and a test set X and y:

```
from explainerdashboard import ClassifierExplainer, RegressionExplainer

explainer = ClassifierExplainer(model, X_test, y_test)
explainer = RegressionExplainer(model, X_test, y_test)
```

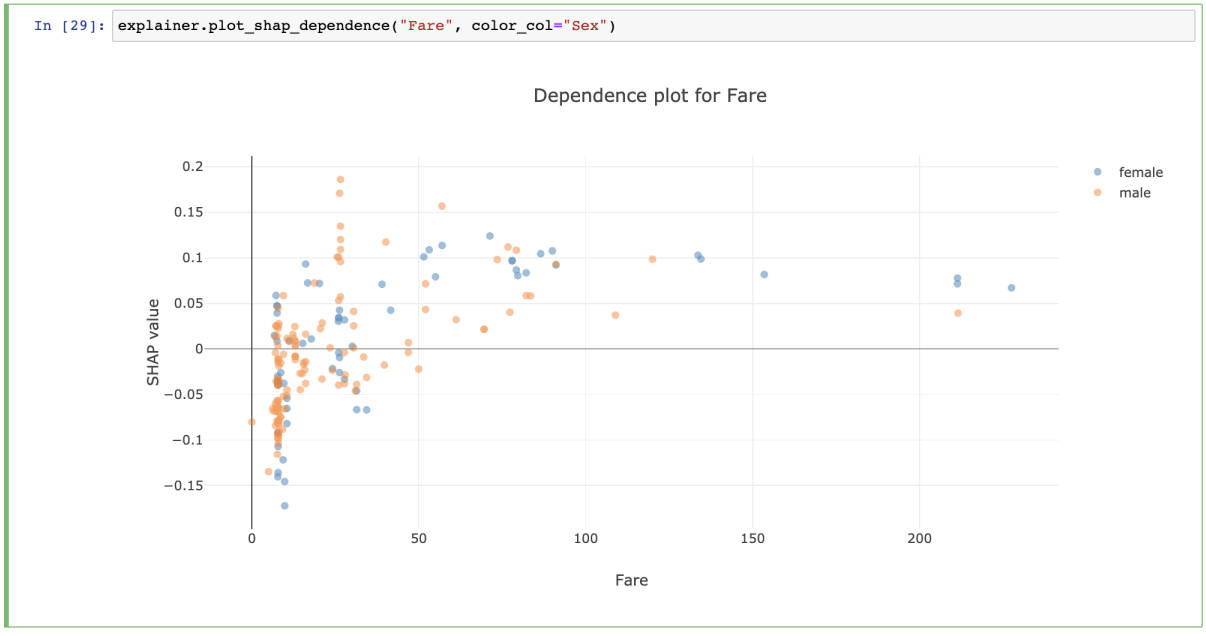
This is enough to launch an ExplainerDashboard:

```
from explainerdashboard import ExplainerDashboard
ExplainerDashboard(explainer).run()
```



Or you can use it interactively in a notebook to inspect your model using the built-in plotting methods, e.g.:

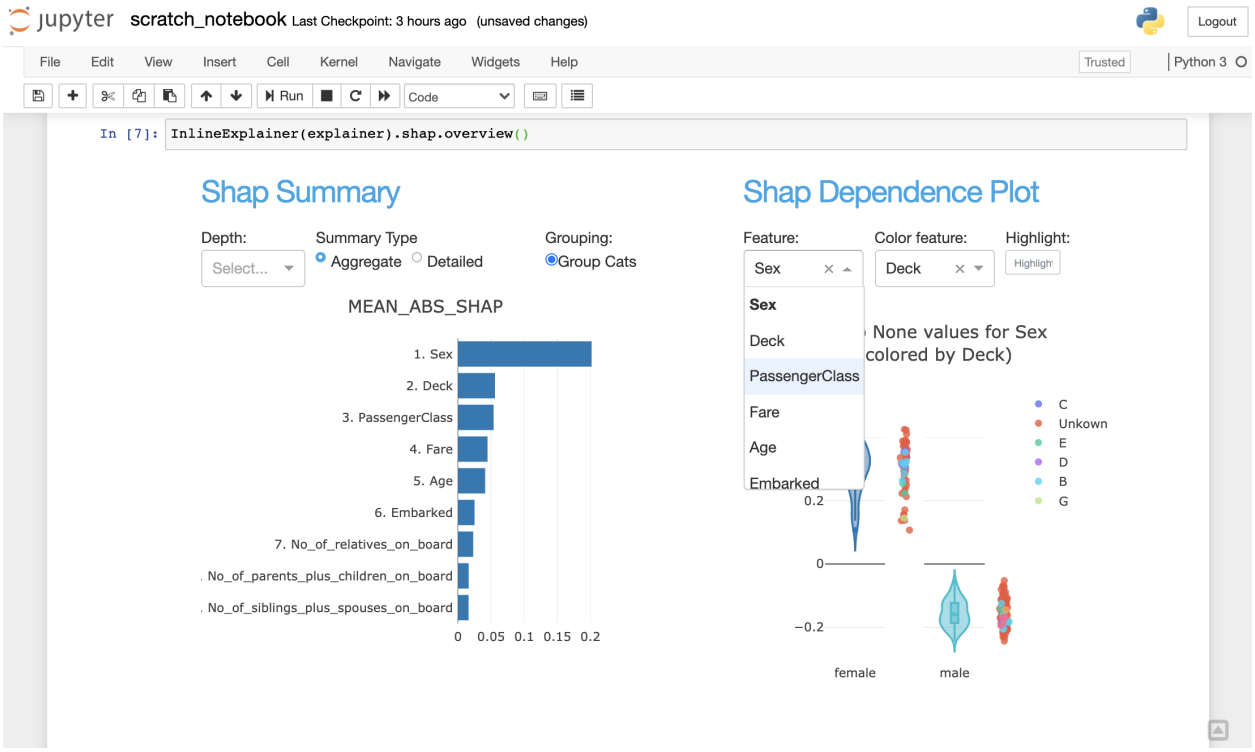
```
explainer.plot_confusion_matrix()
explainer.plot_contributions(index=0)
explainer.plot_dependence("Fare", color_col="Sex")
```



For the full lists of plots available see *Plots*.

Or you can start an interactive *ExplainerComponent* in your notebook using *InlineExplainer*, e.g.:

```
from explainerdashboard import InlineExplainer

InlineExplainer(explainer).tab.importances()
InlineExplainer(explainer).classifier.roc_auc()
InlineExplainer(explainer).regression.residuals_vs_col()
InlineExplainer(explainer).shap.overview()
```



4.1.2 Parameters

There are a number of optional parameters that can either make sure that SHAP values get calculated in the appropriate way, or that make the explainer give a bit nicer and more convenient output:

```
ClassifierExplainer(model, X_test, y_test,
        shap='linear', # manually set shap type, overrides default 'guess'
        X_background=X_train, # set background dataset for shap calculations
        model_output='logodds', # set model_output to logodds (vs probability)
        cats=['Sex', 'Deck', 'Embarked'], # makes it easy to group onehotencoded vars
        idxs=test_names, # index with str identifier
        index_name="Passenger", # description of index
        descriptions=feature_descriptions, # show long feature descriptions in hovers
        target='Survival', # the name of the target variable (y)
        precision='float32', # save memory by setting lower precision. Default is 'float64
↪'
        labels=['Not survived', 'Survived']) # show target labels instead of ['0', '1']
```

cats

If you have onehot-encoded your categorical variables, they will show up as a lot of independent features. This clutters your feature space, and often makes it hard to interpret the effect of the underlying categorical feature.

You can pass a `dict` to the parameter `cats` specifying which are the onehotencoded columns, and what the grouped feature name should be:

```
ClassifierExplainer(model, X, y, cats={'Gender': ['Sex_male', 'Sex_female']})
```

However if you encoded your feature with `pd.get_dummies(df, prefix=['Name'])`, then the resulting onehot encoded columns should be named 'Name_John', 'Name_Mary', Name_Bob', etc. (or in general CategoricalFeature_Category), then you can simply pass a list of the prefixes to cats:

```
ClassifierExplainer(model, X, y, cats=['Sex', 'Deck', 'Embarked'])
```

And you can also combine the two methods:

```
ClassifierExplainer(model, X, y,
    cats=[{'Gender': ['Sex_male', 'Sex_female']}, 'Deck', 'Embarked'])
```

You can now use these categorical features directly as input for plotting methods, e.g. `explainer.plot_dependence("Deck")`, which will now generate violin plots instead of the default scatter plots.

cats_notencoded

When you have onehotencoded a categorical feature, you may have dropped some columns during feature selection. Or there are new categories in the test set that were not encoded as columns in the training set. In that cases all columns in your onehot encoding may be equal to `0` for some rows. By default the value assigned to the aggregated feature for such cases is `'NOT_ENCODED'`, but this can be overriden with the `cats_notencoded` parameter:

```
ClassifierExplainer(model, X, y,
    cats=[{'Gender': ['Sex_male', 'Sex_female']}, 'Deck', 'Embarked'],
    cats_notencoded={'Gender': 'Gender Other', 'Deck': 'Unknown Deck', 'Embarked':
↪'Stowaway'})
```

idxs

You may have specific identifiers (names, customer id's, etc) for each row in your dataset. By default `X.index` will get used to identify individual rows/records in the dashboard. And you can index using both the numerical index, e.g. `explainer.get_contrib_df(0)` for the first row, or using the identifier, e.g. `explainer.get_contrib_df("Braund, Mr. Owen Harris")`.

You can override using `X.index` by passing a list/array/Series `idxs` to the explainer:

```
from explainerdashboard.datasets import titanic_names

test_names = titanic_names(test_only=True)
ClassifierExplainer(model, X_test, y_test, idxs=test_names)
```

index_name

By default `X.index.name` or `idxs.name` is used as the description of the index, but you can also pass it explicitly, e.g.: `index_name="Passenger"`.

descriptions

`descriptions` can be passed as a dictionary of descriptions for each feature. In order to be explanatory, you often have to explain the meaning of the features themselves (especially if the naming is not obvious). Passing the dict along to descriptions will show hover-over tooltips for the various features in the dashboard. If you grouped onehotencoded features with the `cats` parameter, you can also give descriptions of these groups, e.g:

```
ClassifierExplainer(model, X, y,
    cats=[{'Gender': ['Sex_male', 'Sex_female']}, 'Deck', 'Embarked'],
    descriptions={
        'Gender': 'Gender of the passenger',
        'Fare': 'The price of the ticket paid for by the passenger',
        'Deck': 'The deck of the cabin of the passenger',
        'Age': 'Age of the passenger in year'
    })
```

target

Name of the target variable. By default the name of the `y` (`y.name`) is used if `y` is a `pd.Series`, else it defaults to `'target'`, bu this can be overriden:

```
ClassifierExplainer(model, X, y, target="Survival")
```

labels

For `ClassifierExplainer` only: The outcome variables for a classification `y` are assumed to be encoded `0, 1 (, 2, 3, ...)` You can assign string labels by passing e.g. `labels=['Not survived', 'Survived']`:

```
ClassifierExplainer(model, X, y, labels=['Not survived', 'Survived'])
```

units

For `RegressionExplainer` only: the units of the `y` variable. E.g. if the model is predicting house prices in dollars you can set `units='$'`. If it is predicting maintenance time you can set `units='hours'`, etc. This will then be displayed along the axis of various plots:

```
RegressionExplainer(model, X, y, units="$")
```

X_background

Some models like sklearn `LogisticRegression` (as well as certain gradient boosting algorithms such as *xgboost* in probability space) need a background dataset to calculate shap values. These can be passed as `X_background`. If you don't pass an X_background, Explainer uses X instead but gives off a warning. (You want to limit the size of X_background in order to keep the SHAP calculations from getting too slow. Usually a representative background dataset of a couple of hunderd rows should be enough to get decent shap values.)

model_output

By default `model_output` for classifiers is set to `"probability"`, as this is more intuitively explainable to non data scientist stakeholders. However certain models (e.g. `XGBClassifier`, `LGBMCLassifier`, `CatBoostClassifier`), need a background dataset `X_background` to calculate SHAP values in probability space, and are not able to calculate shap interaction values in probability space at all. Therefore you can also pass model_output='logodds', in which case shap values get calculated faster and interaction effects can be studied. Now you just need to explain to your stakeholders what logodds are :)

shap

By default `shap='guess'`, which means that the Explainer will try to guess based on the model what kind of shap explainer it needs: e.g. `shap.TreeExplainer(...)`, `shap.LinearExplainer(...)`, etc.

In case the guess fails or you'd like to override it, you can set it manually: e.g. `shap='tree'` for `shap.TreeExplainer`, `shap='linear'` for `shap.LinearExplainer`, `shap='kernel'` for `shap.KernelExplainer`, `shap='deep'` for `shap.DeepExplainer`, etc.

model_output, X_background example

An example of using setting `X_background` and `model_output` with a `LogisticRegression`:

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X_train, y_train)

explainer = ClassifierExplainer(model, X_test, y_test,
                                    shap='linear',
                                    X_background=X_train,
                                    model_output='logodds')
ExplainerDashboard(explainer).run()
```

cv

Normally metrics and permutation importances get calculated over a single fold (assuming the data `X` is the test set). However if you pass the training set to the explainer, you may wish to cross-validate calculate the permutation importances and metrics. In that case pass the number of folds to `cv`. Note that custom metrics do not work with cross validation for now.

na_fill

If you fill missing values with some extreme value such as `-999` (typical for tree based methods), these can mess with the horizontal axis of your plots. In order to filter these out, you need to tell the explainer what the extreme value is that you used to fill. Defaults to `-999`.

precision

You can set the precision of the calculated shap values, predictions, etc, in order to save on memory usage. Default is `'float64'`, but `'float32'` is probably fine, maybe even `'float16'` for your application.

4.1.3 Pre-calculated shap values

Perhaps you already have calculated the shap values somewhere, or you can calculate them off on a giant cluster somewhere, or your model supports GPU generated shap values.

You can simply add these pre-calculated shap values to the explainer with `explainer.set_shap_values()` and `explainer.set_shap_interaction_values()` methods.

4.1.4 Plots

Shared Plots

The abstract base class `BaseExplainer` defines most of the functionality such as feature importances (both SHAP and permutation based), SHAP values, SHAP interaction values partial dependences, individual contributions, etc. Along with a number of convenient plotting methods. In practice you will use `ClassifierExplainer` or `RegressionExplainer`, however they both inherit all of these basic methods:

```
plot_importances(kind='shap', topx=None, round=3, pos_label=None)
plot_contributions(index, topx=None, cutoff=None, round=2, pos_label=None)
plot_importances_detailed(topx=None, pos_label=None)
plot_interactions_detailed(col, topx=None, pos_label=None)
plot_dependence(col, color_col=None, highlight_idx=None, pos_label=None)
plot_interaction(interact_col, highlight_idx=None, pos_label=None)
plot_pdp(col, index=None, drop_na=True, sample=100, num_grid_lines=100, num_grid_
↪points=10, pos_label=None)
```

example code:

```
explainer = ClassifierExplainer(model, X, y, cats=['Sex', 'Deck', 'Embarked'])
explainer.plot_importances()
explainer.plot_contributions(index=0, topx=5)
explainer.plot_dependence("Fare")
explainer.plot_interaction("Embarked", "PassengerClass")
explainer.plot_pdp("Sex", index=0)
```

plot_importances

BaseExplainer.**plot_importances**(*kind='shap'*, *topx=None*, *round=3*, *pos_label=None*)

plot barchart of importances in descending order.

Parameters

- **type** (*str, optional*) – shap' for mean absolute shap values, 'permutation' for permutation importances, defaults to 'shap'
- **topx** (*int, optional, optional*) – Only return topx features, defaults to None
- **kind** – (Default value = 'shap')
- **round** – (Default value = 3)
- **pos_label** – (Default value = None)

Returns

fig

Return type

plotly.fig

plot_importances_detailed

BaseExplainer.**plot_importances_detailed**(*highlight_index=None*, *topx=None*, *max_cat_colors=5*, *plot_sample=None*, *pos_label=None*)

Plot barchart of mean absolute shap value.

Displays all individual shap value for each feature in a horizontal scatter chart in descending order by mean absolute shap value.

Parameters

- **highlight_index** (*str or int*) – index to highlight
- **topx** (*int, optional*) – Only display topx most important features, defaults to None
- **max_cat_colors** (*int, optional*) – for categorical features, maximum number of categories to label with own color. Defaults to 5.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **pos_label** – positive class (Default value = None)

Returns

plotly.Fig

plot_contributions

BaseExplainer.**plot_contributions**(*index=None*, *X_row=None*, *topx=None*, *cutoff=None*, *sort='abs'*, *orientation='vertical'*, *higher_is_better=True*, *round=2*, *pos_label=None*)

plot waterfall plot of shap value contributions to the model prediction for index.

Parameters

- **index** (*int or str*) – index for which to display prediction
- **X_row** (*pd.DataFrame single row*) – a single row of a features to plot shap contributions for. Can use this instead of index for what-if scenarios.
- **topx** (*int, optional, optional*) – Only display topx features, defaults to None
- **cutoff** (*float, optional, optional*) – Only display features with at least cutoff contribution, defaults to None
- **sort** (*{'abs', 'high-to-low', 'low-to-high', 'importance'}, optional*) – sort by absolute shap value, or from high to low, or low to high, or by order of shap feature importance. Defaults to 'abs'.
- **orientation** (*{'vertical', 'horizontal'}*) – Horizontal or vertical bar chart. Horizontal may be better if you have lots of features. Defaults to 'vertical'.
- **higher_is_better** (*bool*) – if True, up=green, down=red. If false reversed. Defaults to True.
- **round** (*int, optional, optional*) – round contributions to round precision, defaults to 2
- **pos_label** – (Default value = None)

Returns

fig

Return type
plotly.Fig

plot_dependence

BaseExplainer.**plot_dependence**(*col*, *color_col=None*, *highlight_index=None*, *topx=None*, *sort='alphabet'*, *max_cat_colors=5*, *round=3*, *plot_sample=None*, *remove_outliers=False*, *pos_label=None*)

plot shap dependence

Plots a shap dependence plot:

- on the x axis the possible values of the feature *col*
- on the y axis the associated individual shap values

Parameters

- **col** (*str*) – feature to be displayed
- **color_col** (*str*) – if color_col provided then shap values colored (blue-red) according to feature color_col (Default value = None)
- **highlight_index** – individual observation to be highlighed in the plot. (Default value = None)
- **topx** (*int, optional*) – for categorical features only display topx categories.
- **sort** (*str*) – for categorical features, how to sort the categories: alphabetically 'alphabet', most frequent first 'freq', highest mean absolute value first 'shap'. Defaults to 'alphabet'.
- **max_cat_colors** (*int, optional*) – for categorical features, maximum number of categories to label with own color. Defaults to 5.
- **round** (*int, optional*) – rounding to apply to floats. Defaults to 3.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **remove_outliers** (*bool, optional*) – remove observations that are >1.5*IQR in either col or color_col. Defaults to False.
- **pos_label** – positive class (Default value = None)

Returns:

plot_interaction

BaseExplainer.**plot_interaction**(*col*, *interact_col*, *highlight_index=None*, *topx=10*, *sort='alphabet'*, *max_cat_colors=5*, *plot_sample=None*, *remove_outliers=False*, *pos_label=None*)

plots a dependence plot for shap interaction effects

Parameters

- **col** (*str*) – feature for which to find interaction values
- **interact_col** (*str*) – feature for which interaction value are displayed
- **highlight_index** (*str, optional*) – index that will be highlighted, defaults to None

- **topx** (*int, optional*) – number of categorical features to display in violin plots.
- **sort** (*str, optional*) – how to sort categorical features in violin plots. Should be in {'alphabet', 'freq', 'shap'}.
- **max_cat_colors** (*int, optional*) – for categorical features, maximum number of categories to label with own color. Defaults to 5.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **remove_outliers** (*bool, optional*) – remove observations that are >1.5*IQR in either col or color_col. Defaults to False.
- **pos_label** – (Default value = None)

Returns
Plotly Fig

Return type
plotly.Fig

plot_pdp

BaseExplainer.**plot_pdp**(*col*, *index=None*, *X_row=None*, *drop_na=True*, *sample=100*, *gridlines=100*, *gridpoints=10*, *sort='freq'*, *round=2*, *pos_label=None*)

plot partial dependence plot (pdp)

returns plotly fig for a partial dependence plot showing ice lines for num_grid_lines rows, average pdp based on sample of sample. If index is given, display pdp for this specific index.

Parameters

- **col** (*str*) – feature to display pdp graph for
- **index** (*int or str, optional, optional*) – index to highlight in pdp graph, defaults to None
- **X_row** (*pd.Dataframe, single row, optional*) – a row of features to highlight predictions for. Alternative to passing index.
- **drop_na** (*bool, optional, optional*) – if true drop samples with value equal to na_fill, defaults to True
- **sample** (*int, optional, optional*) – sample size on which the average pdp will be calculated, defaults to 100
- **gridlines** (*int, optional*) – number of ice lines to display, defaults to 100
- **gridpoints(ints** – int, optional): number of points on the x axis to calculate the pdp for, defaults to 10
- **sort** (*str, optional*) – For categorical features: how to sort: 'alphabet', 'freq', 'shap'. Defaults to 'freq'.
- **round** (*int, optional*) – round float prediction to number of digits. Defaults to 2.
- **pos_label** – (Default value = None)

Returns
fig

Return type
plotly.Fig

plot_interactions_importance

BaseExplainer.**plot_interactions_importance**(*col*, *topx=None*, *pos_label=None*)

plot mean absolute shap interaction value for col.

Parameters

- **col** – column for which to generate shap interaction value
- **topx** (*int, optional, optional*) – Only return topx features, defaults to None
- **pos_label** – (Default value = None)

Returns
fig

Return type
plotly.fig

plot_interactions_detailed

BaseExplainer.**plot_interactions_detailed**(*col*, *highlight_index=None*, *topx=None*, *max_cat_colors=5*, *plot_sample=None*, *pos_label=None*)

Plot barchart of mean absolute shap interaction values

Displays all individual shap interaction values for each feature in a horizontal scatter chart in descending order by mean absolute shap value.

Parameters

- **col** (*type]*) – feature for which to show interactions summary
- **highlight_index** (*str or int*) – index to highlight
- **topx** (*int, optional*) – only show topx most important features, defaults to None
- **max_cat_colors** (*int, optional*) – for categorical features, maximum number of categories to label with own color. Defaults to 5.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **pos_label** – positive class (Default value = None)

Returns
fig

Classifier Plots

ClassifierExplainer defines a number of additional plotting methods:

```
plot_precision(bin_size=None, quantiles=None, cutoff=None, multiclass=False, pos_
↪label=None)
plot_cumulative_precision(pos_label=None)
plot_classification(cutoff=0.5, percentage=True, pos_label=None)
plot_confusion_matrix(cutoff=0.5, normalized=False, binary=False, pos_label=None)
plot_lift_curve(cutoff=None, percentage=False, round=2, pos_label=None)
plot_roc_auc(cutoff=0.5, pos_label=None)
plot_pr_auc(cutoff=0.5, pos_label=None)
```

example code:

```
explainer = ClassifierExplainer(model, X, y, labels=['Not Survived', 'Survived'])
explainer.plot_confusion_matrix(cutoff=0.6)
explainer.plot_precision(quantiles=10, cutoff=0.6, multiclass=True)
explainer.plot_lift_curve(percentage=True)
explainer.plot_roc_auc(cutoff=0.7)
explainer.plot_pr_auc(cutoff=0.3)
```

More examples in the notebook on the github repo.

plot_precision

ClassifierExplainer.**plot_precision**(*bin_size=None*, *quantiles=None*, *cutoff=None*, *multiclass=False*, *pos_label=None*)

plot precision vs predicted probability

plots predicted probability on the x-axis and observed precision (fraction of actual positive cases) on the y-axis.

Should pass either bin_size fraction of number of quantiles, but not both.

Parameters

- **bin_size** (*float, optional*) – size of the bins on x-axis (e.g. 0.05 for 20 bins)
- **quantiles** (*int, optional*) – number of equal sized quantiles to split the predictions by e.g. 20, optional)
- **cutoff** – cutoff of model to include in the plot (Default value = None)
- **multiclass** – whether to display all classes or only positive class, defaults to False
- **pos_label** – positive label to display, defaults to self.pos_label

Returns

Plotly fig

plot_cumulative_precision

ClassifierExplainer.**plot_cumulative_precision**(*percentile=None*, *pos_label=None*)

plot cumulative precision

returns a cumulative precision plot, which is a slightly different representation of a lift curve.

Parameters
pos_label – positive label to display, defaults to self.pos_label

Returns
plotly fig

plot_classification

ClassifierExplainer.**plot_classification**(*cutoff=0.5*, *percentage=True*, *pos_label=None*)

plot showing a barchart of the classification result for cutoff

Parameters

- **cutoff** (*float, optional*) – cutoff of positive class to calculate lift (Default value = 0.5)
- **percentage** (*bool, optional*) – display percentages instead of counts, defaults to True
- **pos_label** – positive label to display, defaults to self.pos_label

Returns
plotly fig

plot_confusion_matrix

ClassifierExplainer.**plot_confusion_matrix**(*cutoff=0.5*, *percentage=False*, *normalize='all'*, *binary=False*, *pos_label=None*)

plot of a confusion matrix.

Parameters

- **cutoff** (*float, optional, optional*) – cutoff of positive class to calculate confusion matrix for, defaults to 0.5
- **percentage** (*bool, optional, optional*) – display percentages instead of counts , defaults to False
- **normalize** (*str['observed', 'pred', 'all']*) – normalizes confusion matrix over the observed (rows), predicted (columns) conditions or all the population. Defaults to all.
- **binary** (*bool, optional, optional*) – if multiclass display one-vs-rest instead, defaults to False
- **pos_label** – positive label to display, defaults to self.pos_label

Returns
plotly fig

plot_lift_curve

ClassifierExplainer.**plot_lift_curve**(*cutoff=None*, *percentage=False*, *add_wizard=True*, *round=2*, *pos_label=None*)

plot of a lift curve.

Parameters

- **cutoff** (*float, optional*) – cutoff of positive class to calculate lift (Default value = None)
- **percentage** (*bool, optional*) – display percentages instead of counts, defaults to False
- **add_wizard** (*bool, optional*) – Add a line indicating how a perfect model would perform ("the wizard"). Defaults to True.
- **round** – number of digits to round to (Default value = 2)
- **pos_label** – positive label to display, defaults to self.pos_label

Returns

plotly fig

plot_roc_auc

ClassifierExplainer.**plot_roc_auc**(*cutoff=0.5*, *pos_label=None*)

plots ROC_AUC curve.

The TPR and FPR of a particular cutoff is displayed in crosshairs.

Parameters

- **cutoff** – cutoff value to be included in plot (Default value = 0.5)
- **pos_label** – (Default value = None)

Returns:

plot_pr_auc

ClassifierExplainer.**plot_pr_auc**(*cutoff=0.5*, *pos_label=None*)

plots PR_AUC curve.

the precision and recall of particular cutoff is displayed in crosshairs.

Parameters

- **cutoff** – cutoff value to be included in plot (Default value = 0.5)
- **pos_label** – (Default value = None)

Returns:

Regression Plots

For the derived RegressionExplainer class again some additional plots:

```
explainer.plot_predicted_vs_actual(...)
explainer.plot_residuals(...)
explainer.plot_residuals_vs_feature(...)
```

plot_predicted_vs_actual

RegressionExplainer.**plot_predicted_vs_actual**(*round=2*, *logs=False*, *log_x=False*, *log_y=False*, *plot_sample=None*, ***kwargs*)

plot with predicted value on x-axis and actual value on y axis.

Parameters

- **round** (*int, optional*) – rounding to apply to outcome, defaults to 2
- **logs** (*bool, optional*) – log both x and y axis, defaults to False
- **log_y** (*bool, optional*) – only log x axis. Defaults to False.
- **log_x** (*bool, optional*) – only log y axis. Defaults to False.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- ****kwargs** –

Returns

Plotly fig

plot_residuals

RegressionExplainer.**plot_residuals**(*vs_actual=False*, *round=2*, *residuals='difference'*, *plot_sample=None*)

plot of residuals. x-axis is the predicted outcome by default

Parameters

- **vs_actual** (*bool, optional*) – use actual value for x-axis, defaults to False
- **round** (*int, optional*) – rounding to perform on values, defaults to 2
- **residuals** (*str, {'difference', 'ratio', 'log-ratio'} optional*) – How to cal-cualte residuals. Defaults to 'difference'.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)

Returns

Plotly fig

plot_residuals_vs_feature

RegressionExplainer.**plot_residuals_vs_feature**(*col*, *residuals='difference'*, *round=2*, *dropna=True*, *points=True*, *winsor=0*, *topx=None*, *sort='alphabet'*, *plot_sample=None*)

Plot residuals vs individual features

Parameters

- **col** (*str*) – Plot against feature col
- **residuals** (*str, {'difference', 'ratio', 'log-ratio'} optional*) – How to calcualte residuals. Defaults to 'difference'.
- **round** (*int, optional*) – rounding to perform on residuals, defaults to 2
- **dropna** (*bool, optional*) – drop missing values from plot, defaults to True.
- **points** (*bool, optional*) – display point cloud next to violin plot. Defaults to True.
- **winsor** (*int, 0-50, optional*) – percentage of outliers to winsor out of the y-axis. Defaults to 0.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)

Returns

plotly fig

DecisionTree Plots

There are additional mixin classes specifically for `sklearn` `RandomForests` and for xgboost models that define additional methods and plots to investigate and visualize individual decision trees within the ensemblke. These uses the `dtreeviz` library to visualize individual decision trees.

You can get a pd.DataFrame summary of the path that a specific index row took through a specific decision tree. You can also plot the individual predictions of each individual tree for specific row in your data indentified by `index`:

```
explainer.get_decisionpath_df(tree_idx, index)
explainer.get_decisionpath_summary_df(tree_idx, index)
explainer.plot_trees(index)
```

And for dtreeviz visualization of individual decision trees (svg format):

```
explainer.decisiontree(tree_idx, index)
explainer.decisiontree_file(tree_idx, index)
explainer.decisiontree_encoded(tree_idx, index)
```

These methods are part of the `RandomForestExplainer` and XGBExplainer`` mixin classes that get automatically loaded when you pass either a RandomForest or XGBoost model.

plot_trees

RandomForestExplainer.**plot_trees**(*index*, *highlight_tree=None*, *round=2*, *higher_is_better=True*, *pos_label=None*)

plot barchart predictions of each individual prediction tree

Parameters

- **index** – index to display predictions for
- **highlight_tree** – tree to highlight in plot (Default value = None)
- **round** – rounding of numbers in plot (Default value = 2)
- **higher_is_better** (*bool*) – flip red and green. Dummy bool for compatibility with gbm plot_trees().
- **pos_label** – positive class (Default value = None)

Returns:

decisiontree

RandomForestExplainer.**decisiontree**(*tree_idx*, *index*, *show_just_path=False*)

get a dtreeviz visualization of a particular tree in the random forest.

Parameters

- **tree_idx** – the n'th tree in the random forest
- **index** – row index
- **show_just_path** (*bool, optional*) – show only the path not rest of the tree. Defaults to False.

Returns

a IPython display SVG object for e.g. jupyter notebook.

decisiontree_file

RandomForestExplainer.**decisiontree_file**(*tree_idx*, *index*, *show_just_path=False*)

decisiontree_encoded

RandomForestExplainer.**decisiontree_encoded**(*tree_idx*, *index*, *show_just_path=False*)

get a dtreeviz visualization of a particular tree in the random forest.

Parameters

- **tree_idx** – the n'th tree in the random forest
- **index** – row index
- **show_just_path** (*bool, optional*) – show only the path not rest of the tree. Defaults to False.

Returns

a base64 encoded image, for inclusion in websites (e.g. dashboard)

4.1.5 Other explainer outputs

Base outputs

Some other useful tables and outputs you can get out of the explainer:

```
metrics()
get_mean_abs_shap_df(topx=None, cutoff=None, cats=False, pos_label=None)
get_permutation_importances_df(topx=None, cutoff=None, cats=False, pos_label=None)
get_importances_df(kind="shap", topx=None, cutoff=None, cats=False, pos_label=None)
get_contrib_df(index, cats=True, topx=None, cutoff=None, pos_label=None)
get_contrib_summary_df(index, cats=True, topx=None, cutoff=None, round=2, pos_label=None)
get_interactions_df(col, cats=False, topx=None, cutoff=None, pos_label=None)
```

metrics

BaseExplainer.**metrics**(**args*, ***kwargs*)

returns a dict of metrics.

Implemented by either ClassifierExplainer or RegressionExplainer

metrics_descriptions

ClassifierExplainer.**metrics_descriptions**(*cutoff=0.5*, *round=3*, *pos_label=None*)

Returns a metrics dict with the value replaced with a description/interpretation of the value

Parameters

- **cutoff** (*float, optional*) – Cutoff for calculating the metrics. Defaults to 0.5.
- **round** (*int, optional*) – Round to apply to floats. Defaults to 3.
- **pos_label** (*None, optional*) – positive label. Defaults to None.

Returns

dict

RegressionExplainer.**metrics_descriptions**(*round=2*)

Returns a metrics dict, with the metric values replaced by a descriptive string, explaining/interpreting the value of the metric

Returns

dict

get_mean_abs_shap_df

BaseExplainer.**get_mean_abs_shap_df**(*topx=None*, *cutoff=None*, *pos_label=None*)

sorted dataframe with mean_abs_shap

returns a pd.DataFrame with the mean absolute shap values per features, sorted rom highest to lowest.

Parameters

- **topx** (*int, optional, optional*) – Only return topx most importance features, defaults to None

- **cutoff** (*float, optional, optional*) – Only return features with mean abs shap of at least cutoff, defaults to None
- **pos_label** – (Default value = None)

Returns
shap_df

Return type
pd.DataFrame

get_permutation_importances_df

BaseExplainer.**get_permutation_importances_df**(*topx=None*, *cutoff=None*, *pos_label=None*)

dataframe with features ordered by permutation importance.

For more about permutation importances.

see https://explained.ai/rf-importance/index.html

Parameters

- **topx** (*int, optional, optional*) – only return topx most important features, defaults to None
- **cutoff** (*float, optional, optional*) – only return features with importance of at least cutoff, defaults to None
- **pos_label** – (Default value = None)

Returns
importance_df

Return type
pd.DataFrame

get_importances_df

BaseExplainer.**get_importances_df**(*kind='shap'*, *topx=None*, *cutoff=None*, *pos_label=None*)

wrapper function for get_mean_abs_shap_df() and get_permutation_importance_df()

Parameters

- **kind** (*str*) – 'shap' or 'permutations' (Default value = "shap")
- **topx** – only display topx highest features (Default value = None)
- **cutoff** – only display features above cutoff (Default value = None)
- **pos_label** – Positive class (Default value = None)

Returns
pd.DataFrame

get_contrib_df

BaseExplainer.**get_contrib_df**(*index=None*, *X_row=None*, *topx=None*, *cutoff=None*, *sort='abs'*, *pos_label=None*)

shap value contributions to the prediction for index.

Used as input for the plot_contributions() method.

Parameters

- **index** (*int or str*) – index for which to calculate contributions
- **X_row** (*pd.DataFrame, single row*) – single row of feature for which to calculate contrib_df. Can us this instead of index
- **topx** (*int, optional*) – Only return topx features, remainder called REST, defaults to None
- **cutoff** (*float, optional*) – only return features with at least cutoff contributions, defaults to None
- **sort** (*{'abs', 'high-to-low', 'low-to-high', 'importance'}, optional*) – sort by absolute shap value, or from high to low, low to high, or ordered by the global shap importances. Defaults to 'abs'.
- **pos_label** – (Default value = None)

Returns
contrib_df

Return type
pd.DataFrame

get_contrib_summary_df

BaseExplainer.**get_contrib_summary_df**(*index=None*, *X_row=None*, *topx=None*, *cutoff=None*, *round=2*, *sort='abs'*, *pos_label=None*)

Takes a contrib_df, and formats it to a more human readable format

Parameters

- **index** – index to show contrib_summary_df for
- **X_row** (*pd.DataFrame, single row*) – single row of feature for which to calculate contrib_df. Can us this instead of index
- **topx** – Only show topx highest features(Default value = None)
- **cutoff** – Only show features above cutoff (Default value = None)
- **round** – round figures (Default value = 2)
- **sort** (*{'abs', 'high-to-low', 'low-to-high', 'importance'}, optional*) – sort by absolute shap value, or from high to low, or low to high, or ordered by the global shap importances. Defaults to 'abs'.
- **pos_label** – Positive class (Default value = None)

Returns
pd.DataFrame

get_interactions_df

BaseExplainer.**get_interactions_df**(*col*, *topx=None*, *cutoff=None*, *pos_label=None*)

dataframe of mean absolute shap interaction values for col

Parameters

- **col** – Feature to get interactions_df for
- **topx** – Only display topx most important features (Default value = None)
- **cutoff** – Only display features with mean abs shap of at least cutoff (Default value = None)
- **pos_label** – Positive class (Default value = None)

Returns

pd.DataFrame

Classifier outputs

For ClassifierExplainer in addition:

```
random_index(y_values=None, return_str=False,pred_proba_min=None, pred_proba_max=None,
                pred_percentile_min=None, pred_percentile_max=None, pos_label=None)
prediction_result_df(index, pos_label=None)
cutoff_from_percentile(percentile, pos_label=None)
get_precision_df(bin_size=None, quantiles=None, multiclass=False, round=3, pos_
↪label=None)
get_liftcurve_df(pos_label=None)
```

random_index

ClassifierExplainer.**random_index**(*y_values=None*, *return_str=False*, *pred_proba_min=None*, *pred_proba_max=None*, *pred_percentile_min=None*, *pred_percentile_max=None*, *pos_label=None*)

random index satisfying various constraint

Parameters

- **y_values** – list of labels to include (Default value = None)
- **return_str** – return str from self.idxs (Default value = False)
- **pred_proba_min** – minimum pred_proba (Default value = None)
- **pred_proba_max** – maximum pred_proba (Default value = None)
- **pred_percentile_min** – minimum pred_proba percentile (Default value = None)
- **pred_percentile_max** – maximum pred_proba percentile (Default value = None)
- **pos_label** – positive class (Default value = None)

Returns

index

cutoff_from_percentile

ClassifierExplainer.**cutoff_from_percentile**(*percentile*, *pos_label=None*)

The cutoff equivalent to the percentile given

For example if you want the cutoff that splits the highest 20% pred_proba from the lowest 80%, you would set percentile=0.8 and get the correct cutoff.

Parameters

- **percentile** (*float*) – percentile to convert to cutoff
- **pos_label** – positive class (Default value = None)

Returns
cutoff

percentile_from_cutoff

ClassifierExplainer.**percentile_from_cutoff**(*cutoff*, *pos_label=None*)

The percentile equivalent to the cutoff given

For example if set the cutoff at 0.8, then what percentage of pred_proba is above this cutoff?

Parameters

- **cutoff** (*float*) – cutoff to convert to percentile
- **pos_label** – positive class (Default value = None)

Returns
percentile

get_precision_df

ClassifierExplainer.**get_precision_df**(*bin_size=None*, *quantiles=None*, *multiclass=False*, *round=3*, *pos_label=None*)

dataframe with predicted probabilities and precision

Parameters

- **bin_size** (*float, optional, optional*) – group predictions in bins of size bin_size, defaults to 0.1
- **quantiles** (*int, optional, optional*) – group predictions in evenly sized quantiles of size quantiles, defaults to None
- **multiclass** (*bool, optional, optional*) – whether to calculate precision for every class (Default value = False)
- **round** – (Default value = 3)
- **pos_label** – (Default value = None)

Returns
precision_df

Return type
pd.DataFrame

get_liftcurve_df

ClassifierExplainer.**get_liftcurve_df**(*pos_label=None*)

returns a pd.DataFrame with data needed to build a lift curve

Parameters
pos_label – (Default value = None)

Returns:

get_classification_df

ClassifierExplainer.**get_classification_df**(*cutoff=0.5*, *pos_label=None*)

Returns a dataframe with number of observations in each class above and below the cutoff.

Parameters

- **cutoff** (*float, optional*) – Cutoff to split on. Defaults to 0.5.
- **pos_label** (*int, optional*) – Pos label to generate dataframe for. Defaults to self.pos_label.

Returns
pd.DataFrame

roc_auc_curve

ClassifierExplainer.**roc_auc_curve**(*pos_label=None*)

Returns a dict with output from sklearn.metrics.roc_curve() for pos_label: fpr, tpr, thresholds, score

pr_auc_curve

ClassifierExplainer.**pr_auc_curve**(*pos_label=None*)

Returns a dict with output from sklearn.metrics.precision_recall_curve() for pos_label: fpr, tpr, thresholds, score

confusion_matrix

ClassifierExplainer.**confusion_matrix**(*cutoff=0.5*, *binary=True*, *pos_label=None*)

Regression outputs

For RegressionExplainer:

```
random_index(y_min=None, y_max=None, pred_min=None, pred_max=None,
                residuals_min=None, residuals_max=None,
                abs_residuals_min=None, abs_residuals_max=None,
                return_str=False)
```

random_index

RegressionExplainer.**random_index**(*y_min=None*, *y_max=None*, *pred_min=None*, *pred_max=None*, *residuals_min=None*, *residuals_max=None*, *abs_residuals_min=None*, *abs_residuals_max=None*, *return_str=False*, ***kwargs*)

random index following to various exclusion criteria

Parameters

- **y_min** – (Default value = None)
- **y_max** – (Default value = None)
- **pred_min** – (Default value = None)
- **pred_max** – (Default value = None)
- **residuals_min** – (Default value = None)
- **residuals_max** – (Default value = None)
- **abs_residuals_min** – (Default value = None)
- **abs_residuals_max** – (Default value = None)
- **return_str** – return the str index from self.idxs (Default value = False)
- ****kwargs** –

Returns

a random index that fits the exclusion criteria

RandomForest and XGBoost outputs

For RandomForest and XGBoost models mixin classes that visualize individual decision trees will be loaded: `RandomForestExplainer` and `XGBExplainer` with the following additional methods:

```
decisiontree_df(tree_idx, index, pos_label=None)
decisiontree_summary_df(tree_idx, index, round=2, pos_label=None)
decision_path_file(tree_idx, index)
decision_path_encoded(tree_idx, index)
decision_path(tree_idx, index)
```

get_decisionpath_df

RandomForestExplainer.**get_decisionpath_df**(*tree_idx*, *index*, *pos_label=None*)

dataframe with all decision nodes of a particular decision tree for a particular observation.

Parameters

- **tree_idx** – the n'th tree in the random forest
- **index** – row index
- **pos_label** – positive class (Default value = None)

Returns

dataframe with summary of the decision tree path

get_decisionpath_summary_df

RandomForestExplainer.**get_decisionpath_summary_df**(*tree_idx*, *index*, *round=2*, *pos_label=None*)

formats decisiontree_df in a slightly more human readable format.

Parameters

- **tree_idx** – the n'th tree in the random forest or boosted ensemble
- **index** – index
- **round** – rounding to apply to floats (Default value = 2)
- **pos_label** – positive class (Default value = None)

Returns

dataframe with summary of the decision tree path

decisiontree_file

RandomForestExplainer.**decisiontree_file**(*tree_idx*, *index*, *show_just_path=False*)

decisiontree_encoded

RandomForestExplainer.**decisiontree_encoded**(*tree_idx*, *index*, *show_just_path=False*)

get a dtreeviz visualization of a particular tree in the random forest.

Parameters

- **tree_idx** – the n'th tree in the random forest
- **index** – row index
- **show_just_path** (*bool, optional*) – show only the path not rest of the tree. Defaults to False.

Returns

a base64 encoded image, for inclusion in websites (e.g. dashboard)

decisiontree

RandomForestExplainer.**decisiontree**(*tree_idx*, *index*, *show_just_path=False*)

get a dtreeviz visualization of a particular tree in the random forest.

Parameters

- **tree_idx** – the n'th tree in the random forest
- **index** – row index
- **show_just_path** (*bool, optional*) – show only the path not rest of the tree. Defaults to False.

Returns

a IPython display SVG object for e.g. jupyter notebook.

4.1.6 Calculated Properties

In general `Explainers` don't calculate any properties of the model or the data until they are needed for an output, so-called lazy calculation. When the property is calculated once, it is stored for next time. So the first time you invoke a plot involving shap values may take a while to calculate. The next time will be basically instant.

You can access these properties directly from the explainer, e.g. `explainer.get_shap_values_df()`. For classifier models if you want values for a particular `pos_label` you can pass this label `explainer.get_shap_values_df(0)` would get the shap values for the 0'th class label.

In order to calculate all properties of the explainer at once, you can call `explainer.calculate_properties()`. (`ExplainerComponents` have a similar method `component.calculate_dependencies()` to calculate all properties that that specific component will need).

The various properties are:

```
explainer.preds
explainer.pred_percentiles
explainer.permutation_importances(pos_label)
explainer.mean_abs_shap_df(pos_label)
explainer.shap_base_value(pos_label)
explainer.get_shap_values_df(pos_label)
explainer.shap_interaction_values
```

For `ClassifierExplainer`:

```
explainer.y_binary
explainer.pred_probas_raw
explainer.pred_percentiles_raw
explainer.pred_probas(pos_label)
explainer.roc_auc_curve(pos_label)
explainer.pr_auc_curve(pos_label)
explainer.get_classification_df(cutoff, pos_label)
explainer.get_liftcurve_df(pos_label)
explainer.confusion_matrix(cutoff, binary, pos_label)
```

For `RegressionExplainer`:

```
explainer.residuals
explainer.abs_residuals
```

4.1.7 Setting pos_label

For `ClassifierExplainer` you can calculate most properties for multiple labels as the positive label. With a binary classification usually label '1' is the positive class, but in some cases you might also be interested in the '0' label.

For multiclass classification you may want to investigate shap dependences for the various classes.

You can pass a parameter `pos_label` to almost every property or method, to get the output for that specific positive label. If you don't pass a `pos_label` manually to a specific method, the global `self.pos_label` will be used. You can set this directly on the explainer (even us str labels if you have set these):

```
explainer.pos_label = 0
explainer.plot_dependence("Fare") # will show plot for pos_label=0
explainer.pos_label = 'Survived'
```

(continues on next page)

(continued from previous page)

```
explainer.plot_dependence("Fare") # will now show plot for pos_label=1
explainer.plot_dependence("Fare", pos_label=0) # show plot for label 0, without changing␣
↪explainer.pos_label
```

The `ExplainerDashboard` will show a dropdown menu in the header to choose a particular `pos_label`. Changing this will basically update every single plot in the dashboard.

4.1.8 BaseExplainer

class explainerdashboard.explainers.**BaseExplainer**(*model*, *X*, *y=None*, *permutation_metric=sklearn.metrics.r2_score*, *shap='guess'*, *X_background=None*, *model_output='raw'*, *cats=None*, *cats_notencoded=None*, *idxs=None*, *index_name=None*, *target=None*, *descriptions=None*, *n_jobs=None*, *permutation_cv=None*, *cv=None*, *na_fill=-999*, *precision='float64'*, *shap_kwargs=None*)

Defines the basic functionality that is shared by both ClassifierExplainer and RegressionExplainer.

Parameters

- **model** – a model with a scikit-learn compatible .fit and .predict methods
- **X** (*pd.DataFrame*) – a pd.DataFrame with your model features
- **y** (*pd.Series*) – Dependent variable of your model, defaults to None
- **permutation_metric** (*function or str*) – is a scikit-learn compatible metric function (or string). Defaults to r2_score
- **shap** (*str*) – type of shap_explainer to fit: 'tree', 'linear', 'kernel'. Defaults to 'guess'.
- **X_background** (*pd.DataFrame*) – background X to be used by shap explainers that need a background dataset (e.g. shap.KernelExplainer or shap.TreeExplainer with boosting models and model_output='probability').
- **model_output** (*str*) – model_output of shap values, either 'raw', 'logodds' or 'probability'. Defaults to 'raw' for regression and 'probability' for classification.
- **cats** (*{dict, list}*) – dict of features that have been onehotencoded. e.g. cats={'Sex':['Sex_male', 'Sex_female']}. If all encoded columns are underscore-seperated (as above), can simply pass a list of prefixes: cats=['Sex']. Allows to group onehot encoded categorical variables together in various plots. Defaults to None.
- **cats_notencoded** (*dict*) – value to display when all onehot encoded columns are equal to zero. Defaults to 'NOT_ENCODED' for each onehot col.
- **idxs** (*pd.Series*) – list of row identifiers. Can be names, id's, etc. Defaults to X.index.
- **index_name** (*str*) – identifier for row indexes. e.g. index_name='Passenger'. Defaults to X.index.name or idxs.name.
- **target** (Optional[str]) – name of the predicted target, e.g. "Survival", "Ticket price", etc. Defaults to y.name.
- **n_jobs** (*int*) – for jobs that can be parallelized using joblib, how many processes to split the job in. For now only used for calculating permutation importances. Defaults to None.

- **permutation_cv** (*int*) – Deprecated! Use parameter cv instead! (now also works for calculating metrics)
- **cv** (*int*) – If not None then permutation importances and metrics will get calculated using cross validation across X. Use this when you are passing the training set to the explainer. Defaults to None.
- **na_fill** (*int*) – The filler used for missing values, defaults to -999.
- **precision** (*str*) – precision with which to store values. Defaults to "float64".
- **shap_kwargs** (*dict*) – dictionary of keyword arguments to be passed to the shap explainer. most typically used to supress an additivity check e.g. *shap_kwargs=dict(check_additivity=False)*

get_permutation_importances_df(*topx=None*, *cutoff=None*, *pos_label=None*)

dataframe with features ordered by permutation importance.

For more about permutation importances.

see https://explained.ai/rf-importance/index.html

Parameters

- **topx** (*int, optional, optional*) – only return topx most important features, defaults to None
- **cutoff** (*float, optional, optional*) – only return features with importance of at least cutoff, defaults to None
- **pos_label** – (Default value = None)

Returns
importance_df

Return type
pd.DataFrame

get_shap_values_df(*pos_label=None*)

SHAP values calculated using the shap library

set_shap_values(*base_value*, *shap_values*)

Set shap values manually. This is useful if you already have shap values calculated, and do not want to calculate them again inside the explainer instance. Especially for large models and large datasets you may want to calculate shap values on specialized hardware, and then add them to the explainer manually.

Parameters

- **base_value** (*float*) – the shap intercept generated by e.g. base_value = shap.TreeExplainer(model).shap_values(X_test).expected_value
- **shap_values** (*np.ndarray]*) – Generated by e.g. shap_values = shap.TreeExplainer(model).shap_values(X_test)

set_shap_interaction_values(*shap_interaction_values*)

Manually set shap interaction values in case you have already pre-computed these elsewhere and do not want to re-calculate them again inside the explainer instance.

Parameters
shap_interaction_values (*np.ndarray*) – shap interactions values of shape (n, m, m)

get_mean_abs_shap_df(*topx=None*, *cutoff=None*, *pos_label=None*)

sorted dataframe with mean_abs_shap

returns a pd.DataFrame with the mean absolute shap values per features, sorted rom highest to lowest.

Parameters

- **topx** (*int, optional, optional*) – Only return topx most importance features, defaults to None
- **cutoff** (*float, optional, optional*) – Only return features with mean abs shap of at least cutoff, defaults to None
- **pos_label** – (Default value = None)

Returns
shap_df

Return type
pd.DataFrame

get_importances_df(*kind='shap'*, *topx=None*, *cutoff=None*, *pos_label=None*)

wrapper function for get_mean_abs_shap_df() and get_permutation_importance_df()

Parameters

- **kind** (*str*) – 'shap' or 'permutations' (Default value = "shap")
- **topx** – only display topx highest features (Default value = None)
- **cutoff** – only display features above cutoff (Default value = None)
- **pos_label** – Positive class (Default value = None)

Returns
pd.DataFrame

plot_importances(*kind='shap'*, *topx=None*, *round=3*, *pos_label=None*)

plot barchart of importances in descending order.

Parameters

- **type** (*str, optional*) – shap' for mean absolute shap values, 'permutation' for permutation importances, defaults to 'shap'
- **topx** (*int, optional, optional*) – Only return topx features, defaults to None
- **kind** – (Default value = 'shap')
- **round** – (Default value = 3)
- **pos_label** – (Default value = None)

Returns
fig

Return type
plotly.fig

plot_importances_detailed(*highlight_index=None*, *topx=None*, *max_cat_colors=5*, *plot_sample=None*, *pos_label=None*)

Plot barchart of mean absolute shap value.

Displays all individual shap value for each feature in a horizontal scatter chart in descending order by mean absolute shap value.

Parameters

- **highlight_index** (*str or int*) – index to highlight
- **topx** (*int, optional*) – Only display topx most important features, defaults to None
- **max_cat_colors** (*int, optional*) – for categorical features, maximum number of categories to label with own color. Defaults to 5.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **pos_label** – positive class (Default value = None)

Returns
plotly.Fig

plot_contributions(*index=None*, *X_row=None*, *topx=None*, *cutoff=None*, *sort='abs'*, *orientation='vertical'*, *higher_is_better=True*, *round=2*, *pos_label=None*)

plot waterfall plot of shap value contributions to the model prediction for index.

Parameters

- **index** (*int or str*) – index for which to display prediction
- **X_row** (*pd.DataFrame single row*) – a single row of a features to plot shap contributions for. Can use this instead of index for what-if scenarios.
- **topx** (*int, optional, optional*) – Only display topx features, defaults to None
- **cutoff** (*float, optional, optional*) – Only display features with at least cutoff contribution, defaults to None
- **sort** (*{'abs', 'high-to-low', 'low-to-high', 'importance'}, optional*) – sort by absolute shap value, or from high to low, or low to high, or by order of shap feature importance. Defaults to 'abs'.
- **orientation** (*{'vertical', 'horizontal'}*) – Horizontal or vertical bar chart. Horizontal may be better if you have lots of features. Defaults to 'vertical'.
- **higher_is_better** (*bool*) – if True, up=green, down=red. If false reversed. Defaults to True.
- **round** (*int, optional, optional*) – round contributions to round precision, defaults to 2
- **pos_label** – (Default value = None)

Returns
fig

Return type
plotly.Fig

plot_dependence(*col*, *color_col=None*, *highlight_index=None*, *topx=None*, *sort='alphabet'*, *max_cat_colors=5*, *round=3*, *plot_sample=None*, *remove_outliers=False*, *pos_label=None*)

plot shap dependence

Plots a shap dependence plot:

- on the x axis the possible values of the feature *col*
- on the y axis the associated individual shap values

Parameters

- **col** (*str*) – feature to be displayed
- **color_col** (*str*) – if color_col provided then shap values colored (blue-red) according to feature color_col (Default value = None)
- **highlight_index** – individual observation to be highlighed in the plot. (Default value = None)
- **topx** (*int, optional*) – for categorical features only display topx categories.
- **sort** (*str*) – for categorical features, how to sort the categories: alphabetically 'alphabet', most frequent first 'freq', highest mean absolute value first 'shap'. Defaults to 'alphabet'.
- **max_cat_colors** (*int, optional*) – for categorical features, maximum number of categories to label with own color. Defaults to 5.
- **round** (*int, optional*) – rounding to apply to floats. Defaults to 3.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **remove_outliers** (*bool, optional*) – remove observations that are >1.5*IQR in either col or color_col. Defaults to False.
- **pos_label** – positive class (Default value = None)

Returns:

plot_interaction(*col*, *interact_col*, *highlight_index=None*, *topx=10*, *sort='alphabet'*, *max_cat_colors=5*, *plot_sample=None*, *remove_outliers=False*, *pos_label=None*)

plots a dependence plot for shap interaction effects

Parameters

- **col** (*str*) – feature for which to find interaction values
- **interact_col** (*str*) – feature for which interaction value are displayed
- **highlight_index** (*str, optional*) – index that will be highlighted, defaults to None
- **topx** (*int, optional*) – number of categorical features to display in violin plots.
- **sort** (*str, optional*) – how to sort categorical features in violin plots. Should be in {'alphabet', 'freq', 'shap'}.
- **max_cat_colors** (*int, optional*) – for categorical features, maximum number of categories to label with own color. Defaults to 5.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **remove_outliers** (*bool, optional*) – remove observations that are >1.5*IQR in either col or color_col. Defaults to False.
- **pos_label** – (Default value = None)

Returns
Plotly Fig

Return type
plotly.Fig

plot_interactions_detailed(*col*, *highlight_index=None*, *topx=None*, *max_cat_colors=5*, *plot_sample=None*, *pos_label=None*)

Plot barchart of mean absolute shap interaction values

Displays all individual shap interaction values for each feature in a horizontal scatter chart in descending order by mean absolute shap value.

Parameters

- **col** (*type]*) – feature for which to show interactions summary
- **highlight_index** (*str or int*) – index to highlight
- **topx** (*int, optional*) – only show topx most important features, defaults to None
- **max_cat_colors** (*int, optional*) – for categorical features, maximum number of categories to label with own color. Defaults to 5.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **pos_label** – positive class (Default value = None)

Returns

fig

plot_pdp(*col*, *index=None*, *X_row=None*, *drop_na=True*, *sample=100*, *gridlines=100*, *gridpoints=10*, *sort='freq'*, *round=2*, *pos_label=None*)

plot partial dependence plot (pdp)

returns plotly fig for a partial dependence plot showing ice lines for num_grid_lines rows, average pdp based on sample of sample. If index is given, display pdp for this specific index.

Parameters

- **col** (*str*) – feature to display pdp graph for
- **index**(*int or str, optional, optional*) – index to highlight in pdp graph, defaults to None
- **X_row** (*pd.Dataframe, single row, optional*) – a row of features to highlight predictions for. Alternative to passing index.
- **drop_na** (*bool, optional, optional*) – if true drop samples with value equal to na_fill, defaults to True
- **sample** (*int, optional, optional*) – sample size on which the average pdp will be calculated, defaults to 100
- **gridlines** (*int, optional*) – number of ice lines to display, defaults to 100
- **gridpoints(ints** – int, optional): number of points on the x axis to calculate the pdp for, defaults to 10
- **sort** (*str, optional*) – For categorical features: how to sort: 'alphabet', 'freq', 'shap'. Defaults to 'freq'.
- **round** (*int, optional*) – round float prediction to number of digits. Defaults to 2.
- **pos_label** – (Default value = None)

Returns

fig

Return type
plotly.Fig

4.1.9 ClassifierExplainer

For classification (e.g. `RandomForestClassifier`) models you use `ClassifierExplainer`.

You can pass an additional parameter to `__init__()` with a list of label names. For multilabel classifier you can set the positive class with e.g. `explainer.pos_label=1`. This will make sure that for example `explainer.pred_probas` will return the probability of that label.

More examples in the notebook on the github repo.

class explainerdashboard.explainers.**ClassifierExplainer**(*model*, *X*, *y=None*, *permutation_metric=sklearn.metrics.roc_auc_score*, *shap='guess'*, *X_background=None*, *model_output='probability'*, *cats=None*, *cats_notencoded=None*, *idxs=None*, *index_name=None*, *target=None*, *descriptions=None*, *n_jobs=None*, *permutation_cv=None*, *cv=None*, *na_fill=-999*, *precision='float64'*, *shap_kwargs=None*, *labels=None*, *pos_label=1*)

Explainer for classification models. Defines the shap values for each possible class in the classification.

You assign the positive label class afterwards with e.g. explainer.pos_label=0

In addition defines a number of plots specific to classification problems such as a precision plot, confusion matrix, roc auc curve and pr auc curve.

Compared to BaseExplainer defines two additional parameters

Parameters

- **model** – a model with a scikit-learn compatible .fit and .predict methods
- **X** (*pd.DataFrame*) – a pd.DataFrame with your model features
- **y** (*pd.Series*) – Dependent variable of your model, defaults to None
- **permutation_metric** (*function or str*) – is a scikit-learn compatible metric function (or string). Defaults to r2_score
- **shap** (*str*) – type of shap_explainer to fit: 'tree', 'linear', 'kernel'. Defaults to 'guess'.
- **X_background** (*pd.DataFrame*) – background X to be used by shap explainers that need a background dataset (e.g. shap.KernelExplainer or shap.TreeExplainer with boosting models and model_output='probability').
- **model_output** (*str*) – model_output of shap values, either 'raw', 'logodds' or 'probability'. Defaults to 'raw' for regression and 'probability' for classification.
- **cats** (*{dict, list}*) – dict of features that have been onehotencoded. e.g. cats={'Sex':['Sex_male', 'Sex_female']}. If all encoded columns are underscore-seperated (as above), can simply pass a list of prefixes: cats=['Sex']. Allows to group onehot encoded categorical variables together in various plots. Defaults to None.
- **cats_notencoded** (*dict*) – value to display when all onehot encoded columns are equal to zero. Defaults to 'NOT_ENCODED' for each onehot col.

- **idxs** (*pd.Series*) – list of row identifiers. Can be names, id's, etc. Defaults to X.index.
- **index_name** (*str*) – identifier for row indexes. e.g. index_name='Passenger'. Defaults to X.index.name or idxs.name.
- **target** (Optional[str]) – name of the predicted target, e.g. "Survival", "Ticket price", etc. Defaults to y.name.
- **n_jobs** (*int*) – for jobs that can be parallelized using joblib, how many processes to split the job in. For now only used for calculating permutation importances. Defaults to None.
- **permutation_cv** (*int*) – Deprecated! Use parameter cv instead! (now also works for calculating metrics)
- **cv** (*int*) – If not None then permutation importances and metrics will get calculated using cross validation across X. Use this when you are passing the training set to the explainer. Defaults to None.
- **na_fill** (*int*) – The filler used for missing values, defaults to -999.
- **precision** (str) – precision with which to store values. Defaults to "float64".
- **shap_kwargs** (*dict*) – dictionary of keyword arguments to be passed to the shap explainer. most typically used to supress an additivity check e.g. *shap_kwargs=dict(check_additivity=False)*
- **labels** (*list*) – list of str labels for the different classes, defaults to e.g. ['0', '1'] for a binary classification
- **pos_label** (int) – class that should be used as the positive class, defaults to 1

set_shap_values(*base_value*, *shap_values*)

Set shap values manually. This is useful if you already have shap values calculated, and do not want to calculate them again inside the explainer instance. Especially for large models and large datasets you may want to calculate shap values on specialized hardware, and then add them to the explainer manually.

Parameters

- **base_value** (*list[float]*) – list of shap intercept generated by e.g. base_value = shap.TreeExplainer(model).shap_values(X_test).expected_value. Should be a list with a float for each class. For binary classification and some models shap only provides the base value for the positive class, in which case you need to provide [1-base_value, base_value] or [-base_value, base_value] depending on whether the shap values are for probabilities or logodds.
- **shap_values** (*list[np.ndarray]*) – Generated by e.g. shap_values = shap.TreeExplainer(model).shap_values(X_test) For binary classification and some models shap only provides the shap values for the positive class, in which case you need to provide [1-shap_values, shap_values] or [-shap_values, shap_values] depending on whether the shap values are for probabilities or logodds.

set_shap_interaction_values(*shap_interaction_values*)

Manually set shap interaction values in case you have already pre-computed these elsewhere and do not want to re-calculate them again inside the explainer instance.

Parameters

shap_interaction_values (*np.ndarray*) – shap interactions values of shape (n, m, m)

random_index(*y_values=None*, *return_str=False*, *pred_proba_min=None*, *pred_proba_max=None*, *pred_percentile_min=None*, *pred_percentile_max=None*, *pos_label=None*)

random index satisfying various constraint

Parameters

- **y_values** – list of labels to include (Default value = None)
- **return_str** – return str from self.idxs (Default value = False)
- **pred_proba_min** – minimum pred_proba (Default value = None)
- **pred_proba_max** – maximum pred_proba (Default value = None)
- **pred_percentile_min** – minimum pred_proba percentile (Default value = None)
- **pred_percentile_max** – maximum pred_proba percentile (Default value = None)
- **pos_label** – positive class (Default value = None)

Returns
index

get_precision_df(*bin_size=None*, *quantiles=None*, *multiclass=False*, *round=3*, *pos_label=None*)

dataframe with predicted probabilities and precision

Parameters

- **bin_size** (*float, optional, optional*) – group predictions in bins of size bin_size, defaults to 0.1
- **quantiles** (*int, optional, optional*) – group predictions in evenly sized quantiles of size quantiles, defaults to None
- **multiclass** (*bool, optional, optional*) – whether to calculate precision for every class (Default value = False)
- **round** – (Default value = 3)
- **pos_label** – (Default value = None)

Returns
precision_df

Return type
pd.DataFrame

get_liftcurve_df(*pos_label=None*)

returns a pd.DataFrame with data needed to build a lift curve

Parameters
pos_label – (Default value = None)

Returns:

get_classification_df(*cutoff=0.5*, *pos_label=None*)

Returns a dataframe with number of observations in each class above and below the cutoff.

Parameters

- **cutoff** (*float, optional*) – Cutoff to split on. Defaults to 0.5.
- **pos_label** (*int, optional*) – Pos label to generate dataframe for. Defaults to self.pos_label.

Returns
pd.DataFrame

plot_precision(*bin_size=None*, *quantiles=None*, *cutoff=None*, *multiclass=False*, *pos_label=None*)

plot precision vs predicted probability

plots predicted probability on the x-axis and observed precision (fraction of actual positive cases) on the y-axis.

Should pass either bin_size fraction of number of quantiles, but not both.

Parameters

- **bin_size** (*float, optional*) – size of the bins on x-axis (e.g. 0.05 for 20 bins)
- **quantiles** (*int, optional*) – number of equal sized quantiles to split the predictions by e.g. 20, optional)
- **cutoff** – cutoff of model to include in the plot (Default value = None)
- **multiclass** – whether to display all classes or only positive class, defaults to False
- **pos_label** – positive label to display, defaults to self.pos_label

Returns
Plotly fig

plot_cumulative_precision(*percentile=None*, *pos_label=None*)

plot cumulative precision

returns a cumulative precision plot, which is a slightly different representation of a lift curve.

Parameters
pos_label – positive label to display, defaults to self.pos_label

Returns
plotly fig

plot_confusion_matrix(*cutoff=0.5*, *percentage=False*, *normalize='all'*, *binary=False*, *pos_label=None*)

plot of a confusion matrix.

Parameters

- **cutoff** (*float, optional, optional*) – cutoff of positive class to calculate confusion matrix for, defaults to 0.5
- **percentage** (*bool, optional, optional*) – display percentages instead of counts , defaults to False
- **normalize** (*str['observed', 'pred', 'all']*) – normalizes confusion matrix over the observed (rows), predicted (columns) conditions or all the population. Defaults to all.
- **binary** (*bool, optional, optional*) – if multiclass display one-vs-rest instead, defaults to False
- **pos_label** – positive label to display, defaults to self.pos_label

Returns
plotly fig

plot_lift_curve(*cutoff=None*, *percentage=False*, *add_wizard=True*, *round=2*, *pos_label=None*)

plot of a lift curve.

Parameters

- **cutoff** (*float, optional*) – cutoff of positive class to calculate lift (Default value = None)

- **percentage** (*bool, optional*) – display percentages instead of counts, defaults to False
- **add_wizard** (*bool, optional*) – Add a line indicating how a perfect model would perform ("the wizard"). Defaults to True.
- **round** – number of digits to round to (Default value = 2)
- **pos_label** – positive label to display, defaults to self.pos_label

Returns
plotly fig

plot_classification(*cutoff=0.5*, *percentage=True*, *pos_label=None*)

plot showing a barchart of the classification result for cutoff

Parameters

- **cutoff** (*float, optional*) – cutoff of positive class to calculate lift (Default value = 0.5)
- **percentage** (*bool, optional*) – display percentages instead of counts, defaults to True
- **pos_label** – positive label to display, defaults to self.pos_label

Returns
plotly fig

plot_roc_auc(*cutoff=0.5*, *pos_label=None*)

plots ROC_AUC curve.

The TPR and FPR of a particular cutoff is displayed in crosshairs.

Parameters

- **cutoff** – cutoff value to be included in plot (Default value = 0.5)
- **pos_label** – (Default value = None)

Returns:

plot_pr_auc(*cutoff=0.5*, *pos_label=None*)

plots PR_AUC curve.

the precision and recall of particular cutoff is displayed in crosshairs.

Parameters

- **cutoff** – cutoff value to be included in plot (Default value = 0.5)
- **pos_label** – (Default value = None)

Returns:

4.1.10 RegressionExplainer

For regression models (e.g. `RandomForestRegressor`) models you use `RegressionExplainer`.

You can pass `units` as an additional parameter for the units of the target variable (e.g. `units="$"`).

More examples in the notebook on the github repo.

class explainerdashboard.explainers.**RegressionExplainer**(*model*, *X*, *y=None*, *permutation_metric=sklearn.metrics.r2_score*, *shap='guess'*, *X_background=None*, *model_output='raw'*, *cats=None*, *cats_notencoded=None*, *idxs=None*, *index_name=None*, *target=None*, *descriptions=None*, *n_jobs=None*, *permutation_cv=None*, *cv=None*, *na_fill=-999*, *precision='float64'*, *shap_kwargs=None*, *units=''*)

Explainer for regression models.

In addition to BaseExplainer defines a number of plots specific to regression problems such as a predicted vs actual and residual plots.

Combared to BaseExplainerBunch defines two additional parameters.

Parameters

- **model** – a model with a scikit-learn compatible .fit and .predict methods
- **X** (*pd.DataFrame*) – a pd.DataFrame with your model features
- **y** (*pd.Series*) – Dependent variable of your model, defaults to None
- **permutation_metric** (*function or str*) – is a scikit-learn compatible metric function (or string). Defaults to r2_score
- **shap** (*str*) – type of shap_explainer to fit: 'tree', 'linear', 'kernel'. Defaults to 'guess'.
- **X_background** (*pd.DataFrame*) – background X to be used by shap explainers that need a background dataset (e.g. shap.KernelExplainer or shap.TreeExplainer with boosting models and model_output='probability').
- **model_output** (*str*) – model_output of shap values, either 'raw', 'logodds' or 'probability'. Defaults to 'raw' for regression and 'probability' for classification.
- **cats** (*{dict, list}*) – dict of features that have been onehotencoded. e.g. cats={'Sex':['Sex_male', 'Sex_female']}. If all encoded columns are underscore-seperated (as above), can simply pass a list of prefixes: cats=['Sex']. Allows to group onehot encoded categorical variables together in various plots. Defaults to None.
- **cats_notencoded** (*dict*) – value to display when all onehot encoded columns are equal to zero. Defaults to 'NOT_ENCODED' for each onehot col.
- **idxs** (*pd.Series*) – list of row identifiers. Can be names, id's, etc. Defaults to X.index.
- **index_name** (*str*) – identifier for row indexes. e.g. index_name='Passenger'. Defaults to X.index.name or idxs.name.
- **target** (Optional[str]) – name of the predicted target, e.g. "Survival", "Ticket price", etc. Defaults to y.name.
- **n_jobs** (*int*) – for jobs that can be parallelized using joblib, how many processes to split the job in. For now only used for calculating permutation importances. Defaults to None.
- **permutation_cv** (*int*) – Deprecated! Use parameter cv instead! (now also works for calculating metrics)
- **cv** (*int*) – If not None then permutation importances and metrics will get calculated using cross validation across X. Use this when you are passing the training set to the explainer. Defaults to None.

- **na_fill** (*int*) – The filler used for missing values, defaults to -999.
- **precision** (str) – precision with which to store values. Defaults to "float64".
- **shap_kwargs** (*dict*) – dictionary of keyword arguments to be passed to the shap explainer. most typically used to supress an additivity check e.g. *shap_kwargs=dict(check_additivity=False)*
- **units** (*str*) – units to display for regression quantity

property residuals

y-preds

Type
residuals

random_index(*y_min=None*, *y_max=None*, *pred_min=None*, *pred_max=None*, *residuals_min=None*, *residuals_max=None*, *abs_residuals_min=None*, *abs_residuals_max=None*, *return_str=False*, ***kwargs*)

random index following to various exclusion criteria

Parameters

- **y_min** – (Default value = None)
- **y_max** – (Default value = None)
- **pred_min** – (Default value = None)
- **pred_max** – (Default value = None)
- **residuals_min** – (Default value = None)
- **residuals_max** – (Default value = None)
- **abs_residuals_min** – (Default value = None)
- **abs_residuals_max** – (Default value = None)
- **return_str** – return the str index from self.idxs (Default value = False)
- ****kwargs** –

Returns
a random index that fits the exclusion criteria

metrics(*show_metrics=None*)

dict of performance metrics: root_mean_squared_error, mean_absolute_error and R-squared

Parameters
show_metrics (*List*) – list of metrics to display in order. Defaults to None, displaying all metrics.

plot_predicted_vs_actual(*round=2*, *logs=False*, *log_x=False*, *log_y=False*, *plot_sample=None*, ***kwargs*)

plot with predicted value on x-axis and actual value on y axis.

Parameters

- **round** (*int, optional*) – rounding to apply to outcome, defaults to 2
- **logs** (*bool, optional*) – log both x and y axis, defaults to False
- **log_y** (*bool, optional*) – only log x axis. Defaults to False.
- **log_x** (*bool, optional*) – only log y axis. Defaults to False.

- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- ****kwargs** –

Returns
Plotly fig

plot_residuals(*vs_actual=False*, *round=2*, *residuals='difference'*, *plot_sample=None*)

plot of residuals. x-axis is the predicted outcome by default

Parameters

- **vs_actual** (*bool, optional*) – use actual value for x-axis, defaults to False
- **round** (*int, optional*) – rounding to perform on values, defaults to 2
- **residuals** (*str, {'difference', 'ratio', 'log-ratio'} optional*) – How to calcualte residuals. Defaults to 'difference'.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)

Returns
Plotly fig

plot_residuals_vs_feature(*col*, *residuals='difference'*, *round=2*, *dropna=True*, *points=True*, *winsor=0*, *topx=None*, *sort='alphabet'*, *plot_sample=None*)

Plot residuals vs individual features

Parameters

- **col** (*str*) – Plot against feature col
- **residuals** (*str, {'difference', 'ratio', 'log-ratio'} optional*) – How to calcualte residuals. Defaults to 'difference'.
- **round** (*int, optional*) – rounding to perform on residuals, defaults to 2
- **dropna** (*bool, optional*) – drop missing values from plot, defaults to True.
- **points** (*bool, optional*) – display point cloud next to violin plot. Defaults to True.
- **winsor** (*int, 0-50, optional*) – percentage of outliers to winsor out of the y-axis. Defaults to 0.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)

Returns
plotly fig

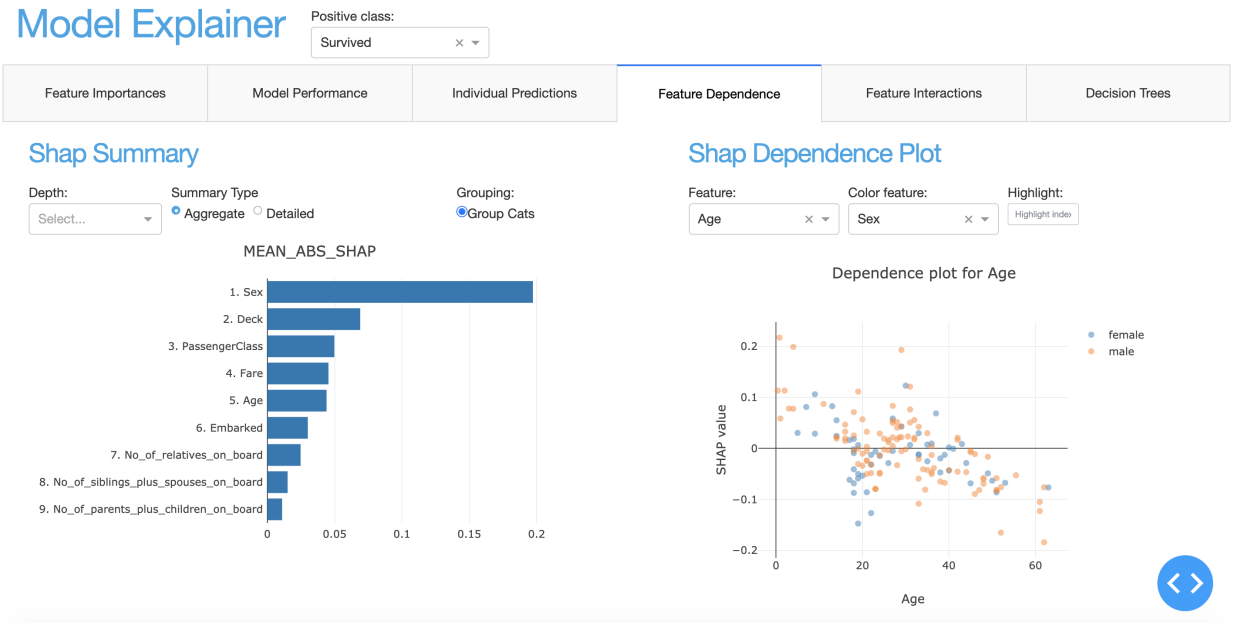
4.2 ExplainerDashboard

4.2.1 Starting the default dashboard

In order to start an `ExplainerDashboard` you first need to contruct an `Explainer` instance. On the basis of this explainer you can then quickly start an interactive dashboard.

The `ExplainerDashboard` API is quite flexible. By default it tries to display all the default tabs that are compatible with your `model` and `model_output` (i.e. interactions and decision_trees might be excluded):

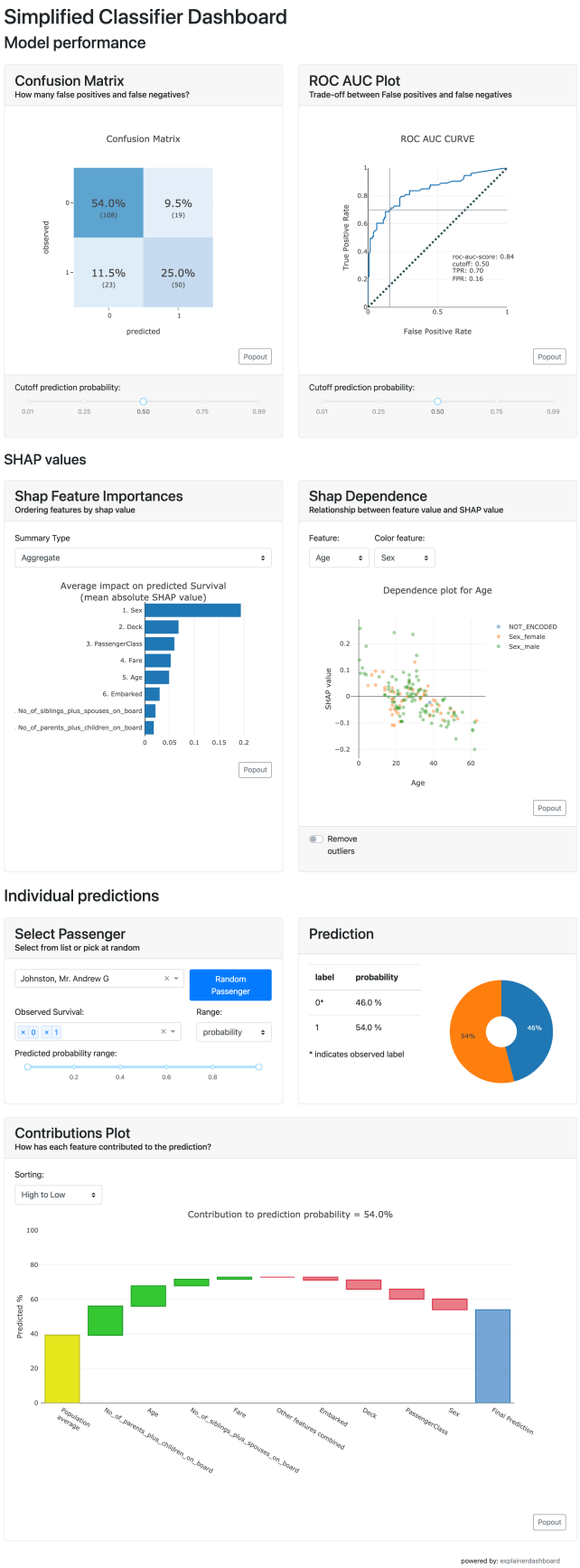
```
from explainerdashboard import ClassifierExplainer, ExplainerDashboard

explainer = ClassifierExplainer(model, X_test, y_test)
ExplainerDashboard(explainer).run()
```



4.2.2 Simplified single page dashboard

For a simplified single page dashboard, use:

```
ExplainerDashboard(explainer, simple=True).run()
```


Simplified Classifier Dashboard
Model performance
Confusion Matrix
How many false positives and false negatives?
Confusion Matrix
54.0%
(108)
9.5%
(19)
11.5%
(23)
25.0%
(50)
observed
predicted
Popout
Cutoff prediction probability:
0.01
0.25
0.50
0.75
0.99
ROC AUC Plot
Trade-off between False positives and false negatives
ROC AUC CURVE
True Positive Rate
False Positive Rate
roc-auc-score: 0.84
cutoff: 0.50
TPR: 0.70
FPR: 0.16
Popout
Cutoff prediction probability:
SHAP values
Shap Feature Importances
Ordering features by shap value
Summary Type
Aggregate
Average impact on predicted Survival
(mean absolute SHAP value)
1. Sex
2. Deck
3. PassengerClass
4. Fare
5. Age
6. Embarked
No_of_siblings_plus_spouses_on_board
No_of_parents_plus_children_on_board
Popout
Shap Dependence
Relationship between feature value and SHAP value
Feature:
Age
Color feature:
Sex
Dependence plot for Age
NOT_ENCODED
Sex_female
Sex_male
SHAP value
Age
Popout
Remove
outliers
Individual predictions
Select Passenger
Select from list or pick at random
Johnston, Mr. Andrew G
Random
Passenger
Observed Survival:
Range:
probability
Predicted probability range:
Prediction
label
probability
0*
46.0 %
1
54.0 %
* indicates observed label
46%
54%
Contributions Plot
How has each feature contributed to the prediction?
Sorting:
High to Low
Contribution to prediction probability = 54.0%
Predicted %
Population average
No_of_parents_plus_children_on_board
Age
No_of_siblings_plus_spouses_on_board
Fare
Other features combined
Embarked
Deck
PassengerClass
Sex
Final Prediction
Popout
powered by: explainerdashboard

4.2.3 Switching off tabs with booleans

If you'd like a little bit more control over which tabs are displayed, you can switch off individual tabs with their respective booleans (they all default to True):

```
ExplainerDashboard(explainer,
   importances=True,
   model_summary=False,
   contributions=True,
   whatif=True,
   shap_dependence=True,
   shap_interaction=False
   decision_trees=True
).run()
```

Note: The interactions tab can take quite some time to compute, so you would usually switch it off if you're not particularly interested in interaction effects between features.

4.2.4 Starting a single tab dashboard

If you pass a single ExplainerComponent class or instance or string identifier, ExplainerDashboard will display that component as a standalone page. The following three lines will all have the effect of launching an ImportancesTab as a single page:

```
from explainerdashboard.custom import ImportancesComposite

ExplainerDashboard(explainer, ImportancesComposite).run()

imp_tab = ImportancesTab(explainer)
ExplainerDashboard(explainer, imp_tab).run()

ExplainerDashboard(explainer, "importances").run()
```

4.2.5 Starting a multitab dashboard

Besided the single page dashboard above you can also pass a list of `ExplainerComponents` to construct multiple tabs. These can be a mix of the different types discussed above. E.g.:

```
ExplainerDashboard(explainer, [ImportancesCompoiste, imp_tab, "importances"]).run()
```

This would start a dashboard with three importances tabs. (not sure why you would do that, but hopefully you get the point :)

The tabs can be imported from `explainerdashboard.custom`, they include `ImportancesComposite`, `ModelSummaryComposite`, `IndividualPredictionsComposite`, `WhatIfComposite`, `ShapDependenceComposite`, `ShapInteractionsComposite` and `DecisionTreesComposite`.

You can also build your own custom tabs, see the *Custom Dashboards* section.

4.2.6 Using explainerdashboard inside Jupyter notebook or google colab

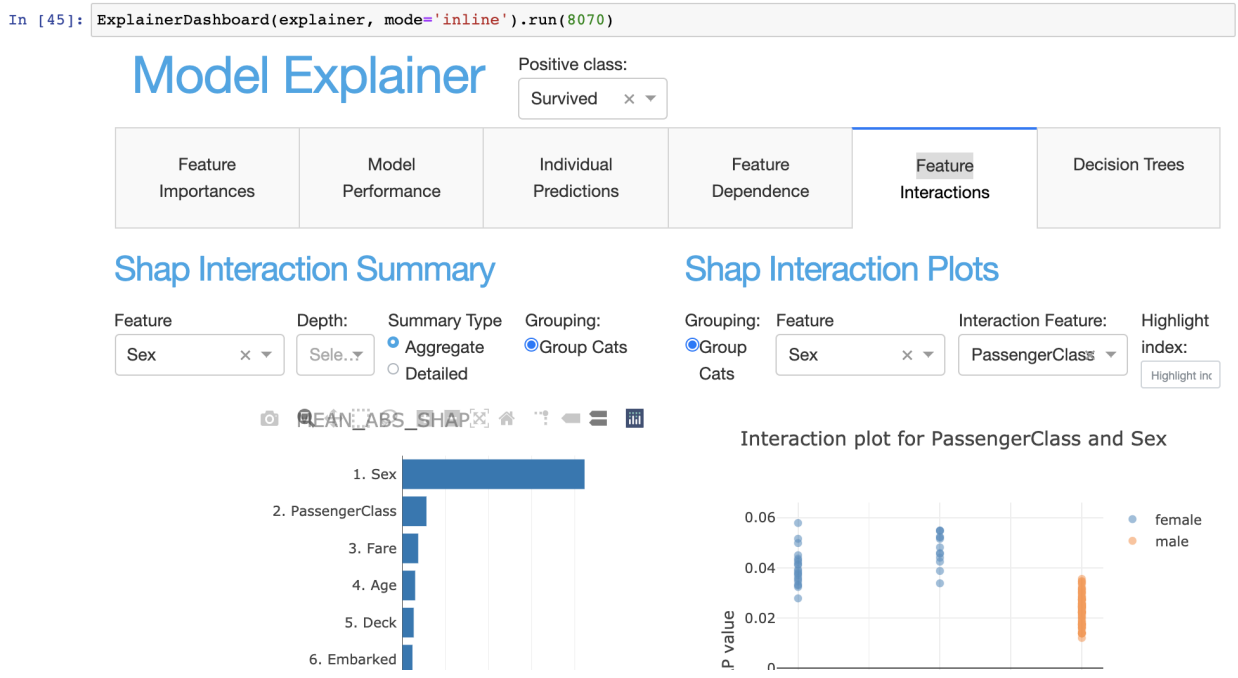
You can start the dashboard with the standard `dash.Dash()` server or with the new notebook friendly `JupyterDash` server. The latter will allow you to keep working interactively in your notebook while the dashboard is running. Also, this allows you to run an explainerdashboard from within Google Colab!

The default dash server is started with `mode='dash'`. (except in Google Colab, where the default is `mode='external'`) There are three notebook compatible options: `mode='inline'` for running the dashboard in an output cell in your notebook, `mode='jupyterlab'` for runnning the dashboard in jupyterlab pane, or `mode='external'` which runs the dashboard in a seperate tab:

```
ExplainerDashboard(explainer).run() # default is either 'dash' or 'external' in colab
ExplainerDashboard(explainer, mode='dash').run()
ExplainerDashboard(explainer, mode='inline').run(port=8051)
ExplainerDashboard(explainer, mode='jupyterlab').run(8052)
ExplainerDashboard(explainer, mode='external').run()
```

The parameters `width` and `height` determine the size of the output area in pixels. (default to `1000x800`). You can kill a JupyterDash based dashboard with the `classmethod .terminate(port)`:

```
ExplainerDashboard().terminate(8050)
```



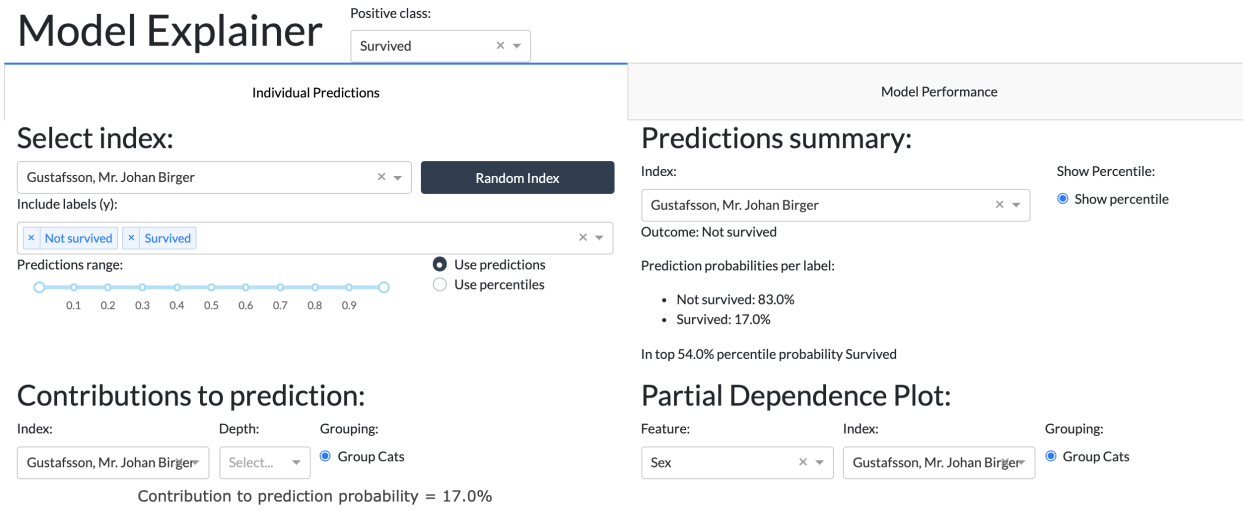
4.2.7 Adding a theme

`explainerdashboard` comes with the default bootstrap theme, but you can override it with the `bootstrap` parameter. Additional info on styling bootstrap layout can be found at: https://dash-bootstrap-components.opensource.faculty.ai/docs/themes/

You can add a theme passing a url of the `.css` stylesheet. `dash_bootstrap_components` provide a convenient `themes` module with urls to the most popular themes, e.g.:

```
from dash_bootstrap_components.themes import FLATLY

ExplainerDashboard(explainer, ["contributions", "model_summary"],
                   bootstrap=FLATLY, mode='external').run()
```

Example of a nice flat black and white theme called "FLATLY":



The full list of available themes can be found on the dbc documentation page.

4.2.8 Hiding title and label selector

For multiclass classification models it is convenient to be able to set the positive class for the entire dashboard with the dropdown in the header. However if you wish to this dropdown selector you can simply pass `header_hide_selector=True`.

In order to hide the title itself pass `header_hide_title=True`. Or to hide the entire header pass `hide_header=True`.

4.2.9 Choosing a port

By default `dash` apps run on port 8050, however you can choose any other port in the run method:

```
ExplainerDashboard(explainer).run(port=8051)
```

Or even shorter:

```
ExplainerDashboard(explainer).run(8051)
```

4.2.10 Exposing the flask server

When running a dashboard in production you probably want to run it with some heavier web server like `gunicorn` or `waitress`. For this you need to expose the flask server. This can be found in `self.app.server`, or with the `flask_server()` method.

If you define your dashboard in dashboard.py then you can expose your dashboard server like this:

```
db = ExplainerDashboard(explainer)
server = db.flask_server()
# equivalently: server = db.app.server
```

You then start the dashboard on the commandline with:

```
gunicorn dashboard:server
```

Or if you are on windows:

```
waitress-serve dashboard:server
```

See the deployment section for more info on using explainerdashboard in production.

4.2.11 ExplainerDashboard documentation

class explainerdashboard.dashboards.**ExplainerDashboard**(*explainer=None*, *tabs=None*, *title='Model Explainer'*, *name=None*, *description=None*, *simple=False*, *hide_header=False*, *header_hide_title=False*, *header_hide_selector=False*, *header_hide_download=False*, *hide_poweredby=False*, *block_selector_callbacks=False*, *pos_label=None*, *fluid=True*, *mode='dash'*, *width=1000*, *height=800*, *bootstrap=None*, *external_stylesheets=None*, *server=True*, *url_base_pathname=None*, *routes_pathname_prefix=None*, *requests_pathname_prefix=None*, *responsive=True*, *logins=None*, *port=8050*, *importances=True*, *model_summary=True*, *contributions=True*, *whatif=True*, *shap_dependence=True*, *shap_interaction=True*, *decision_trees=True*, ***kwargs*)

Creates an explainerdashboard out of an Explainer object.

single page dashboard:
If tabs is a single ExplainerComponent class or instance, display it as a standalone page without tabs.

Multi tab dashboard:
If tabs is a list of ExplainerComponent classes or instances, then construct a layout with a tab per component. Instead of components you can also pass the following strings: "importances", "model_summary", "contributions", "shap_dependence", "shap_interaction" or "decision_trees". You can mix and combine these different modularities, e.g.:

tabs=[ImportancesTab, "contributions", custom_tab]

If tabs is None, then construct tabs based on the boolean parameters:
importances, model_summary, contributions, shap_dependence, shap_interaction and decision_trees, which all default to True.

You can select four different modes:

- 'dash': standard dash.Dash() app
- 'inline': JupyterDash app inline in a notebook cell output
- 'jupyterlab': JupyterDash app in jupyterlab pane
- 'external': JupyterDash app in external tab

You can switch off the title and positive label selector
with header_hide_title=True and header_hide_selector=True.

You run the dashboard
with e.g. ExplainerDashboard(explainer).run(port=8050)

Parameters

- **explainer()** – explainer object
- **tabs()** – single component or list of components
- **title** (*str, optional*) – title of dashboard, defaults to 'Model Explainer'
- **name** (*str, optional*) – name of the dashboard. Used for assigning url in ExplainerHub.
- **description** (*str, optional*) – summary for dashboard. Gets used for title tooltip and in description for ExplainerHub.
- **simple** (*bool, optional*) – instead of full dashboard with all tabs display a single page SimplifiedClassifierDashboard or SimplifiedRegressionDashboard.
- **hide_header** (*bool, optional) hide the header (title+selector*) –
- **header_hide_title** (*bool, optional*) – hide the title, defaults to False
- **header_hide_selector** (*bool, optional*) – hide the positive class selector for classifier models, defaults, to False
- **header_hide_download** (*bool, optional*) – hide the download link in the header. Defaults to False.
- **hide_poweredby** (*bool, optional*) – hide the powered by footer
- **block_selector_callbacks** (*bool, optional*) – block the callback of the pos label selector. Useful to avoid clashes when you have your own PosLabelSelector in your layout. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **mode** (*str, {'dash', 'inline' , 'jupyterlab', 'external'}, optional*) – type of dash server to start. 'inline' runs in a jupyter notebook output cell. 'jupyterlab' runs in a jupyterlab pane. 'external' runs in an external tab while keeping the notebook interactive.
- **fluid** (*bool, optional*) – whether to stretch the layout to available space. Defaults to True.
- **width** (*int, optional*) – width of notebook output cell in pixels, defaults to 1000.
- **height** (*int, optional*) – height of notebookn output cell in pixels, defaults to 800.

- **bootstrap** (*str, optional*) – link to bootstrap url. Can use dbc.themese to generate the url, e.g. bootstrap=dbc.themes.FLATLY. Defaults to default bootstrap theme that is stored in the /assets folder so that it works even behind a firewall.
- **external_stylesheets** (*list, optional*) – additional external stylesheets to add. (for themes use the bootstrap parameter)
- **server** (*Flask instance or bool*) – either an instance of an existing Flask server to tie the dashboard to, or True in which case a new Flask server is created.
- **url_base_pathname** (*str*) – url_base_pathname for dashboard, e.g. "/dashboard". Defaults to None.
- **responsive** (*bool*) – make layout responsive to viewport size (i.e. reorganize bootstrap columns on small devices). Set to False when e.g. testing with a headless browser. Defaults to True.
- **logins** (*list of lists*) – list of (hardcoded) logins, e.g. [['login1', 'password1'], ['login2', 'password2']]. Defaults to None (no login required)
- **importances** (*bool, optional*) – include ImportancesTab, defaults to True.
- **model_summary** (*bool, optional*) – include ModelSummaryTab, defaults to True.
- **contributions** (*bool, optional*) – include ContributionsTab, defaults to True.
- **whatif** (*bool, optional*) – include WhatIfTab, defaults to True.
- **shap_dependence** (*bool, optional*) – include ShapDependenceTab, defaults to True.
- **shap_interaction** (*bool, optional*) – include InteractionsTab if model allows it, defaults to True.
- **decision_trees** (*bool, optional*) – include DecisionTreesTab if model allows it, defaults to True.

flask_server()

returns self.app.server so that it can be exposed to e.g. gunicorn

run(*port=None*, *host='0.0.0.0'*, *use_waitress=False*, *mode=None*, ***kwargs*)

Start ExplainerDashboard on port

Parameters

- **port** (*int, optional*) – port to run on. If None, then use self.port.
- **host** (*str, optional*) – host to run on. Defaults to '0.0.0.0'.
- **use_waitress** (*bool, optional*) – use the waitress python web server instead of the flask development server. Only works with mode='dash'. Defaults to False.
- **mode** (*str, {'dash', 'inline' , 'jupyterlab', 'external'}, optional*) – Type of dash server to start. 'inline' runs in a jupyter notebook output cell. 'jupyterlab' runs in a jupyterlab pane. 'external' runs in an external tab while keeping the notebook interactive. 'dash' is the default server. Overrides self.mode, in which case the dashboard will get rebuilt before running it with the right type of dash server. (dash.Dash or JupyterDash). Defaults to None (i.e. self.mode)
- **8050.** (*Defaults to None.self.port defaults to*) –

Raises

ValueError – if mode is unknown

classmethod terminate(*port*, *token=None*)

Classmethodd to terminate any JupyterDash dashboard (so started with mode='inline', mode='external' or mode='jupyterlab') from any ExplainerDashboard by specifying the right port.

Example

ExplainerDashboard(explainer, mode='external').run(port=8050)

ExplainerDashboard.terminate(8050)

Parameters

- **port** (*int*) – port on which the dashboard is running.
- **token** (*str, optional*) – JupyterDash._token class property. Defaults to the _token of the JupyterDash in the current namespace.

Raises

ValueError – if can't find the port to terminate.

4.3 ExplainerHub

If you are hosting multiple `ExplainerDashboards` it becomes convenient to host them at a single place. This is made easy with `ExplainerHub`.

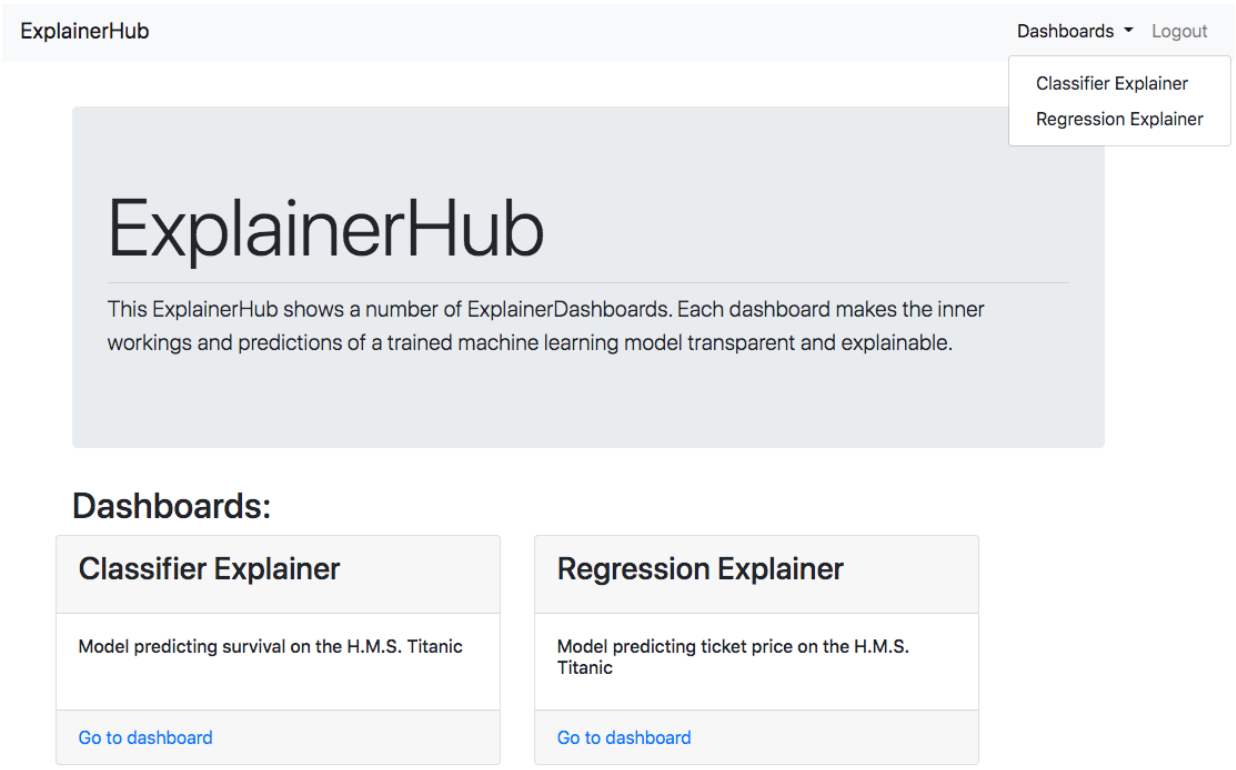
Possible use cases are:

1. Showcasing multiple models to compare and decide which one to put into production
2. Keeping ExplainerDashboards up and running for all models in production in a single place

You can initialize an `ExplainerHub` by passing in a list of `ExplainerDashboards`:

```
db1 = ExplainerDashboard(explainer1)
db2 = ExplainerDashboard(explainer2)
hub = ExplainerHub([db1, db2])
hub.run()
```

Each dashboard is hosted on it's own url path (e.g. `localhost:8050/dashboard1`), and a front end dashboard with links and descriptions for every dashboard is hosted at e.g. `localhost:8050`:



4.3.1 Adjusting title and descriptions

You can adjust the title of the `ExplainerHub` and the description in the jumbotron, by passing `title` and `description`. You can also adjust the `title` and `description` of each `ExplainerDashboard`, and set the url path with `name`:

```
db1 = ExplainerDashboard(explainer1, title="Model One", name="db1",
            description="This is model option one")
db2 = ExplainerDashboard(explainer2, title="Model Two", name="db2",
            description="This is model option two")
hub = ExplainerHub([db1, db2], title="Model Comparison",
            description="Showing dashboards for both model one and two")
hub.run()
```

4.3.2 Adding dashboards

You can add additional dashboards to a hub with:

```
hub.add_dashboard(db2)
```

And remove them by passing their name:

```
hub.remove_dashboard("db2")
```

Adding dashboards using url

You can even add dashboards to a running dashboard by navigating to the `/add_dashboard` route and specifying the path to a `.yaml` file:

```
db2.to_yaml("dashboards/dashboard2.yaml", dump_explainer=True)
ExplainerHub([db1], add_dashboard_route=True).run()
```

If you then navigate to e.g. `http://localhost:8050/add_dashboard/dashboards/dashboard2.yaml` then this dashboard will be added to the hub. By default you can specify any `.yaml` file in any sub directory in which the hub is running.

This can be useful when you for example store explainers and dashboards as part of an MLOps or CI/CD flow.

If you store dashboards in a particular location, you can also specify a pattern to add dashboards:

```
ExplainerHub([db1],
        add_dashboard_route=True,
        add_dashboard_pattern="dashboards/{}.yaml").run()
```

Now you can simply nagivate to `http://localhost:8050/add_dashboard/dashboard2` and it will find `dashboards/dashboard2.yaml` and add it.

You can also remove dashboards by navigating to e.g. `http://localhost:8050/remove_dashboard/db2`.

Note: Dashboards will be added to a particular instance of the hub that is running. So if you have a deployment with multiple workers/nodes, this method will not work for now.

4.3.3 Changing size, theme, etc

By default the hub fills the entire width of the browser, you can make it more slim by passing `fluid=False`. You can also pass other bootstrap themes: `bootstrap=dbc.themes.SKETCHY`. You can adjust the size of the iFrame with `min_height=2000`.

You can also build your own front end if you want. If you pass `no_index=True`, the index page and navbars routes will not get loaded, while the dashboards are still loaded on their respective routes. E.g.:

```
hub = ExplainerHub([db1, db2], no_index=True)
app = hub.flask_server()

@app.route("/")
def custom_index():
    return render_template("custom_index.html")
```

4.3.4 Managing users

You can manage logins and which usernames have access to particular dashboards by passing `logins` and `db_users`. Here we create two users (`user1` and `user2`) and only give `user1` access to the first dashboard, and only give `user2` access to the second dashboard:

```
hub = ExplainerHub([db1, db2],
        logins=[['user1', 'password1'],['user2','password2']],
        db_users=dict(db1=['user1'], db2=['user2'])
```

If you had defined users in your `ExplainerDashboard` then these get automatically carried over to the`` ExplainerHub``:

```
db1 = ExplainerDashboard(explainer1, logins=[['user1', 'password1']])
db2 = ExplainerDashboard(explainer2, logins=[['user2', 'password2']])
hub = ExplainerHub([db1, db2])
```

You can also add users from the hub itself:

```
hub.add_user("user3", "password3")
hub.add_user_to_dashboard("db2", "user3")
```

User/Password pairs can also be stored to file (with passwords hashed). The filename can be set with the `users_file` parameter and defaults to `users.yaml`. When you store a `hub.to_yaml("hub.yaml")` all logins will automatically be exported to this file. This `users_file` can also be managed with `explainerhub` CLI tool (see below).

By default, if you define any user logins, then the hub will only be accesible after logging in. However you can also pass the parameter `dbs_open_by_default=True`, in which case the hub index and any dashboards for which no `db_users` have been defined will not force logins. Only dashboards for which you passed a list of `db_users` will be password locked.

4.3.5 Storing to config

You can store an `ExplainerHub` to disk with `ExplainerHub.to_yaml()`. This will also dump all the explainers to disk, store the configuration of dashboards that make up the hub to individual .yaml files, and store logins to `users.yaml`. You reload a hub with the `from_config` classmethod:

```
hub.to_yaml("hub.yaml")
hub2 = ExplainerHub.from_config("hub.yaml")
```

The *hub.yaml* file looks something like this:

```
explainerhub:
    title: ExplainerHub
    description: Showing dashboards for both model one and two
    masonry: false
    n_dashboard_cols: 3
    users_file: users.yaml
    db_users: null
    port: 8050
    kwargs: {}
    dashboards:
    - db1_dashboard.yaml
    - db2_dashboard.yaml
```

If you pass `integrate_dashboard_yamls=True`, then the configuration of the dashboards get integrated into a single `hub.yaml` file instead of being stored in separate files.

4.3.6 Storing to static html

You can store the hub front-end and the underlying dashboards to static html with e.g. *hub.to_html("hub.html")*. This will also generate individual *.html* files for every dashboard e.g. *dashboard1.html*, *dashboard2.html*, etc, etc.

This might become a bit messy, so instead you can save straight to a zipfile with *hub.to_zip("hub.zip")*.

4.3.7 explainerhub CLI

You can also use the `explainerhub` CLI tool to start your `ExplainerHub` and manage your users straight from the commandline:

```
$ explainerhub run hub.yaml
$ explainerhub add-user
$ explainerhub delete-user
$ explainerhub add-dashboard-user
$ explainerhub delete-dashboard-user
```

4.3.8 SECRET_KEY

In order to make the user session (and so logins) persist when you reboot the server, you need to pass a `SECRET_KEY` to the hub. Like with any Flask app you should be very careful not to store this key somewhere easily findable. Ususally people store it as an environmental variable:

```
$ export SECRET_KEY='5f352379324c22463451387a0aec5d2f'
```

Then you load it with the *os* module and pass it to the hub:

```
ExplainerHub([db1, db2], secret_key=os.environ.get("SECRET_KEY"))
```

If you do not pass a secret key, a random uuid key is generated each time you initialize the hub (which means you'll have to log in every time).

class explainerdashboard.dashboards.**ExplainerHub**(*dashboards*, *title='ExplainerHub'*, *description=None*, *masonry=False*, *n_dashboard_cols=3*, *users_file='users.yaml'*, *user_json=None*, *logins=None*, *db_users=None*, *dbs_open_by_default=False*, *port=8050*, *min_height=3000*, *secret_key=None*, *no_index=False*, *bootstrap=None*, *fluid=True*, *base_route='dashboards'*, *index_to_base_route=False*, *static_to_base_route=False*, *max_dashboards=None*, *add_dashboard_route=False*, *add_dashboard_pattern=None*, ***kwargs*)

ExplainerHub is a way to host multiple dashboards in a single point, and manage access through adding user accounts.

Example

```
hub = ExplainerHub([db1, db2], logins=[['user', 'password']], secret_key="SECRET")
hub.run()
```

A frontend is hosted at e.g. `localhost:8050`, with summaries and links to each individual dashboard. Each ExplainerDashboard is hosted on its own url path, so that you can also find it directly, e.g.:

`localhost:8050/dashboards/dashboard1` and `localhost:8050/dashboards/dashboard2`.

You can store the hub configuration, dashboard configurations, explainers and user database with a single command: `hub.to_yaml('hub.yaml')`.

You can restore the hub with `hub2 = ExplainerHub.from_config('hub.yaml')`

You can start the hub from the command line using the `explainerhub` CLI command: `$ explainerhub run hub.yaml`. You can also use the CLI to add and delete users.

Note: Logins can be defined in multiple places: users.json, ExplainerHub.logins and ExplainerDashboard.logins for each dashboard in dashboards. When users with the same username are defined in multiple locations then passwords are looked up in the following order: hub.logins > dashboard.logins > user.json

---

Note: **kwargs will be forwarded to each dashboard in dashboards.

Parameters

- **dashboards** (*List[ExplainerDashboard]*) – list of ExplainerDashboard to include in ExplainerHub.
- **title** (*str, optional*) – title to display. Defaults to "ExplainerHub".
- **description** (*str, optional*) – Short description of ExplainerHub. Defaults to default text.
- **masonry** (*bool, optional*) – Lay out dashboard cards in fluid bootstrap masonry responsive style. Defaults to False.
- **n_dashboard_cols** (*int, optional*) – If masonry is False, organize cards in rows and columns. Defaults to 3 columns.
- **users_file** (*Path, optional*) – a .yaml or .json file used to store user and (hashed) password data. Defaults to 'users.yaml'.
- **user_json** (*Path, optional*) – Deprecated! A .json file used to store user and (hashed) password data. Defaults to 'users.json'. Was replaced by users_file which can also be a more readable .yaml.
- **logins** (*List[List[str, str]], optional*) – List of ['login', 'password'] pairs, e.g. logins = [['user1', 'password1'], ['user2', 'password2']]
- **db_users** (*dict, optional*) – dictionary limiting access to certain dashboards to a subset of users, e.g dict(dashboard1=['user1', 'user2'], dashboard2=['user3']).
- **dbs_open_by_default** (*bool, optional*) – Only force logins for dashboard with defined db_users. All other dashboards and index no login required. Default to False,
- **port** (*int, optional*) – Port to run hub on. Defaults to 8050.
- **min_height** (*int, optional*) – Defaults to 3000 pixels.

- **secret_key** (*str*) – Flask secret key to pass to dashboard in order to persist logins. Defaults to a new random uuid string every time you start the dashboard. (i.e. no persistence) You should store the secret key somewhere save, e.g. in a environmental variable.
- **no_index** (*bool, optional*) – do not add the "/" route and "dashboards/_dashboard1" etc routes, but only mount the dashboards on e.g. dashboards/dashboard1. This allows you to add your own custom front_end.
- **bootstrap** (*str, optional*) – url with custom bootstrap css, e.g. bootstrap=dbc.themes.FLATLY. Defaults to static bootstrap css.
- **fluid** (*bool, optional*) – Let the bootstrap container fill the entire width of the browser. Defaults to True.
- **base_route** (*str, optional*) – Base route for dashboard : /<base_route>/dashboard1. Defaults to "dashboards".
- **index_to_base_route** (*bool, optional*) – Dispatches Hub to "/base_route/index" instead of the default "/" and "/index". Useful when the host root is not reserved for the ExplainerHub
- **static_to_base_route** (*bool, optional*) – Dispatches Hub to "/base_route/static" instead of the default "/static". Useful when the host root is not reserved for the ExplainerHub
- **max_dashboards** (*int, optional*) – Max number of dashboards in the hub. Defaults to None (for no limitation). If set and you add an additional dashboard, the first dashboard in self.dashboards will be deleted!
- **add_dashboard_route** (*bool, optional*) – open a route /add_dashboard and /remove_dashboard If a user navigates to e.g. /add_dashboard/dashboards/dashboard4.yaml, the hub will check if there exists a folder dashboards which contains a dashboard4.yaml file. If so load this dashboard and add it to the hub. You can remove it with e.g. /remove_dashboard/dashboard4 Alternatively you can specify a path pattern with add_dashboard_pattern. Warning: this will only work if you run the hub on a single worker or node!
- **add_dashboard_pattern** (*str, optional*) – a str pattern with curly brackets in the place of where the dashboard.yaml file can be found. So e.g. if you keep your dashboards in a subdirectory dashboards with a subdirectory for the dashboard name and each yaml file called dashboard.yaml, you could set this to "dashboards/{}/dashboard.yaml", and then navigate to /add_dashboard/dashboard5 to add dashboards/dashboard5/dashboard.yaml.
- ****kwargs** – all kwargs will be forwarded to the constructors of each dashboard in dashboards dashboards.

classmethod from_config(*config*, ***update_params*)

Instantiate an ExplainerHub based on a config file.

Parameters

- **config** (*Union[dict, str, Path]*) – either a dict or a .yaml config file to load
- **update_params** – additional kwargs to override stored settings.

Returns

new instance of ExplainerHub according to the config.

Return type

ExplainerHub

to_yaml(*filepath=None*, *dump_explainers=True*, *return_dict=False*, *integrate_dashboard_yamls=False*, *pickle_type='joblib'*)

Store ExplainerHub to configuration .yaml, store the users to users.json and dump the underlying dashboard .yamls and explainers.

If filepath is None, does not store yaml config to file, but simply return config yaml string.

If filepath provided and dump_explainers=True, then store all underlying explainers to disk.

Parameters

- **filepath** (*Path, optional*) – .yaml file filepath. Defaults to None.
- **dump_explainers** (*bool, optional*) – Store the explainers to disk along with the .yaml file. Defaults to True.
- **return_dict** (*bool, optional*) – Instead of returning or storing yaml return a configuration dictionary. Returns a single dict as if separate_dashboard_yamls=True. Defaults to False.
- **integrate_dashboard_yamls** (*bool, optional*) – Do not generate an individual .yaml file for each dashboard, but integrate them in hub.yaml.
- **pickle_type** (*{'joblib', 'dill', 'pkl'}, optional*) – Defaults to "joblib". Alternatives are "dill" and "pkl".

Returns

{dict, yaml, None}

add_user(*username*, *password*, *add_to_users_file=False*)

add a user with username and password.

Parameters

- **username** (*str*) – username
- **password** (*str*) – password
- **add_to_users_file** (*bool, optional*) – Add the user to the .yaml file defined in self.users_file instead of to self.logins. Defaults to False.

add_user_to_dashboard(*dashboard*, *username*, *add_to_users_file=False*)

add a user to a specific dashboard. If

Parameters

- **dashboard** (*str*) – name of dashboard
- **username** (*str*) – user to add to dashboard
- **add_to_users_file** (*bool, optional*) – add the user to the .yaml or .json file defined in self.users_file instead of to self.db_users. Defaults to False.

get_dashboard_users(*dashboard*)

return all users that have been approved to use a specific dashboard

Parameters

dashboard (*str*) – dashboard

Returns

List

flask_server()

return the Flask server inside the class instance

run(*port=None*, *host='0.0.0.0'*, *use_waitress=False*, ***kwargs*)

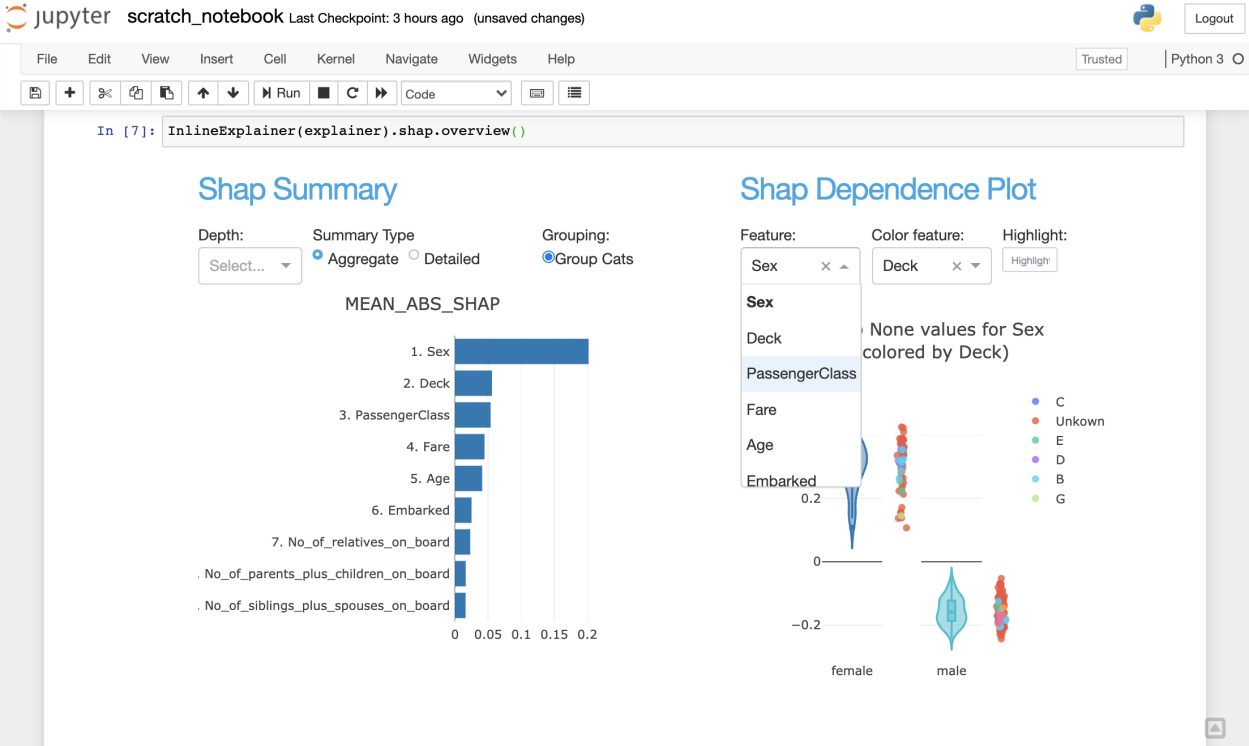
start the ExplainerHub.

Parameters

- **port** (*int, optional*) – Override default port. Defaults to None.
- **host** (*str, optional*) – host name to run dashboard. Defaults to '0.0.0.0'.
- **use_waitress** (*bool, optional*) – Use the waitress python web server instead of the Flask development server. Defaults to False.
- ****kwargs** – will be passed forward to either waitress.serve() or app.run()

4.4 InlineExplainer

As a data scientist you often work inside a notebook environment where you quickly interactively like to explore your data. The `InlineExplainer` allows you to do this by running `ExplainerComponents` (or whole tabs) inline inside your Jupyter notebook (also works in Google Colab!).



This allows you to quickly check model performance, look for shap importances, etc. The components are sorted into subcategories and work with tab-completion.

Example use:

```
from explainerdashboard import InlineExplainer
ie = InlineExplainer(explainer)
ie.importances()
ie.model_stats()
ie.prediction()
ie.random_index()
```

(continues on next page)

(continued from previous page)

```
ie.tab.importances()
ie.tab.modelsummary()
ie.tab.contributions()
ie.tab.dependence()
ie.tab.interactions()
ie.tab.decisiontrees()
ie.shap.overview()
ie.shap.summary()
ie.shap.dependence()
ie.shap.interaction_overview()
ie.shap.interaction_summary()
ie.shap.interaction_dependence()
ie.shap.contributions_graph()
ie.shap.contributions_table()
ie.classifier.model_stats()
ie.classifier.precision()
ie.classifier.confusion_matrix()
ie.classifier.lift_curve()
ie.classifier.classification()
ie.classifier.roc_auc()
ie.classifier.pr_auc()
ie.regression.model_stats()
ie.regression.pred_vs_actual()
ie.regression.residuals()
ie.regression.plots_vs_col()
ie.decisiontrees.overview()
ie.decisiontrees.decision_trees()
ie.decisiontrees.decisionpath_table()
ie.decisiontrees.decisionpath_graph()
```

You can also add options for the size of the output width, or to display the component in a separate tab ('external'), or running on a different port:

```
InlineExplainer(explainer, mode='external', port=8051, width=1000, height=800).
↪importances()
```

Note: You can run a component without instantiating the InlineExplainer first, like for example `InlineExplainer(explainer).importances()`, but then you cannot inspect the kwargs and docstring of that particular component. So to inspect kwargs and docstring you would run:

```
ie = InlineExplainer(explainer)
?ie.importances
```

(or alternatively hit shift-tab in jupyter of course)

You can kill an `InlineExplainer` running on a particular port with `ExplainerDashboard.terminate(port=8050)`.

4.4.1 InlineExplainer documentation

class explainerdashboard.dashboards.**InlineExplainer**(*explainer*, *mode='inline'*, *width=1000*, *height=800*, *port=8050*, ***kwargs*)

Run a single tab inline in a Jupyter notebook using specific method calls.

Parameters

- **explainer** (*BaseExplainer*) – an Explainer object
- **mode** (*str, optional*) – either 'inline', 'jupyterlab' or 'external'
- **width** (int) – width in pixels of inline iframe
- **height** (int) – height in pixels of inline iframe
- **port** (int) – port to run if mode='external'

tab

subclass with InlineExplainerTabs layouts, e.g. InlineExplainer(explainer).tab.modelsummary()

shap

subclass with InlineShapExplainer layouts, e.g. InlineExplainer(explainer).shap.dependence()

classifier

subclass with InlineClassifierExplainer plots, e.g. InlineExplainer(explainer).classifier.confusion_matrix()

regression

subclass with InlineRegressionExplainer plots, e.g. InlineExplainer(explainer).regression.residuals()

decisiontrees

subclass with InlineDecisionTreesExplainer plots, e.g. InlineExplainer(explainer).decisiontrees.decisiontrees()

importances(*title='Importances'*, *name=None*, *subtitle='Which features had the biggest impact?'*, *hide_type=False*, *hide_depth=False*, *hide_popout=False*, *hide_title=False*, *hide_subtitle=False*, *hide_selector=False*, *pos_label=None*, *importance_type='shap'*, *depth=None*, *no_permutations=False*, *description=None*)

Display features importances component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Feature Importances".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str, optional*) – Subtitle.
- **hide_type** (*bool, optional*) – Hide permutation/shap selector toggle. Defaults to False.
- **hide_depth** (*bool, optional*) – Hide number of features toggle. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button
- **hide_title** (*bool, optional*) – hide title. Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.

- **hide_selector** (*bool, optional*) – hide pos label selectors. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **importance_type** (*str, {'permutation', 'shap'} optional*) – initial importance type to display. Defaults to "shap".
- **depth** (*int, optional*) – Initial number of top features to display. Defaults to None (=show all).
- **no_permutations** (*bool, optional*) – Do not use the permutation importances for this component. Defaults to False.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

model_stats(*title='Models Stats'*, ***kwargs*)

Runs model_stats inline in notebook

prediction(*title='Prediction'*, *name=None*, *hide_index=False*, *hide_percentile=False*, *hide_title=False*, *hide_subtitle=False*, *hide_selector=False*, *pos_label=None*, *index=None*, *percentile=True*, *description=None*)

Shows a summary for a particular prediction

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Prediction Summary".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_index** (*bool, optional*) – hide index selector. Defaults to False.
- **hide_percentile** (*bool, optional*) – hide percentile toggle. Defaults to False.
- **hide_title** (*bool, optional*) – hide title. Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selectors. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **index** (*{int, str}, optional*) – Index to display prediction summary for. Defaults to None.
- **percentile** (*bool, optional*) – Whether to add the prediction percentile. Defaults to True.

random_index(*title='Random Index'*, ***kwargs*)

show random index selector inline in notebook

pdp(*title='Partial Dependence Plots'*, *name=None*, *subtitle='How does the prediction change if you change one feature?'*, *hide_col=False*, *hide_index=False*, *hide_title=False*, *hide_subtitle=False*, *hide_footer=False*, *hide_selector=False*, *hide_popout=False*, *hide_dropna=False*, *hide_sample=False*, *hide_gridlines=False*, *hide_gridpoints=False*, *hide_cats_sort=False*, *index_dropdown=True*, *feature_input_component=None*, *pos_label=None*, *col=None*, *index=None*, *dropna=True*, *sample=100*, *gridlines=50*, *gridpoints=10*, *cats_sort='freq'*, *description=None*)

Show Partial Dependence Plot component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Partial Dependence Plot".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_col** (*bool, optional*) – Hide feature selector. Defaults to False.
- **hide_index** (*bool, optional*) – Hide index selector. Defaults to False.
- **hide_title** (*bool, optional*) – Hide title, Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_selector** (*bool, optional*) – hide pos label selectors. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button
- **hide_dropna** (*bool, optional*) – Hide drop na's toggle Defaults to False.
- **hide_sample** (*bool, optional*) – Hide sample size input. Defaults to False.
- **hide_gridlines** (*bool, optional*) – Hide gridlines input. Defaults to False.
- **hide_gridpoints** (*bool, optional*) – Hide gridpounts input. Defaults to False.
- **hide_cats_sort** (*bool, optional*) – Hide the categorical sorting dropdown. Defaults to False.
- **index_dropdown** (*bool, optional*) – Use dropdown for index input instead of free text input. Defaults to True.
- **feature_input_component** (FeatureInputComponent) – A FeatureInputComponent that will give the input to the graph instead of the index selector. If not None, hide_index=True. Defaults to None.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **col** (*str, optional*) – Feature to display PDP for. Defaults to None.
- **index** (*{int, str}, optional*) – Index to add ice line to plot. Defaults to None.
- **dropna** (*bool, optional*) – Drop rows where values equal explainer.na_fill (usually -999). Defaults to True.
- **sample** (*int, optional*) – Sample size to calculate average partial dependence. Defaults to 100.
- **gridlines** (*int, optional*) – Number of ice lines to display in plot. Defaults to 50.
- **gridpoints** (*int, optional*) – Number of breakpoints on horizontal axis Defaults to 10.
- **cats_sort** (*str, optional*) – how to sort categories: 'alphabet', 'freq' or 'shap'. Defaults to 'freq'.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

class explainerdashboard.dashboards.**InlineExplainerTabs**(*inline_explainer*, *name*)

importances(*title='Importances'*, *name=None*, *hide_title=True*, *hide_importances=False*, *hide_descriptions=False*, *hide_selector=True*)

Overview tab of feature importances

Can show both permutation importances and mean absolute shap values.

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Feature Importances".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_title** (*bool, optional*) – hide the title
- **hide_importances** (*bool, optional*) – hide the ImportancesComponent
- **hide_descriptions** (*bool, optional*) – hide the FeatureDescriptionsComponent
- **hide_selector** (*bool, optional*) – hide the post label selector. Defaults to True.

modelsummary(*title='Model Summary'*, *name=None*, *hide_title=True*, *hide_modelsummary=False*, *hide_predsvsactual=False*, *hide_residuals=False*, *hide_regvscol=False*, *logs=False*, *pred_or_actual='vs_pred'*, *residuals='difference'*, *col=None*)

Composite for displaying multiple regression related graphs:

- predictions vs actual plot
- residual plot
- residuals vs feature

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Regression Stats".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_title** (*bool, optional*) – hide title. Defaults to True.
- **hide_modelsummary** (*bool, optional*) – hide RegressionModelSummaryComponent
- **hide_predsvsactual** (*bool, optional*) – hide PredictedVsActualComponent
- **hide_residuals** (*bool, optional*) – hide ResidualsComponent
- **hide_regvscol** (*bool, optional*) – hide RegressionVsColComponent
- **logs** (*bool, optional*) – Use log axis. Defaults to False.
- **pred_or_actual** (*str, optional*) – plot residuals vs predictions or vs y (actual). Defaults to "vs_pred".
- **residuals** (*str, {'difference', 'ratio', 'log-ratio'} optional*) – How to calcualte residuals. Defaults to 'difference'.
- **col** (*{str, int}, optional*) – Feature to use for residuals plot. Defaults to None.

contributions(*title='Contributions'*, *name=None*, *hide_predindexselector=False*, *hide_predictionsummary=False*, *hide_contributiongraph=False*, *hide_pdp=False*, *hide_contributiontable=False*, *hide_title=False*, *hide_selector=True*, *index_check=True*)

Composite for a number of component that deal with individual predictions:

- random index selector
- prediction summary
- shap contributions graph
- shap contribution table
- pdp graph

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Individual Predictions".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_predindexselector** (*bool, optional*) – hide ClassifierRandomIndexComponent or RegressionRandomIndexComponent
- **hide_predictionsummary** (*bool, optional*) – hide ClassifierPredictionSummaryComponent or RegressionPredictionSummaryComponent
- **hide_contributiongraph** (*bool, optional*) – hide ShapContributionsGraphComponent
- **hide_pdp** (*bool, optional*) – hide PdpComponent
- **hide_contributiontable** (*bool, optional*) – hide ShapContributionsTableComponent
- **hide_title** (*bool, optional*) – hide title. Defaults to False.
- **index_check** (*bool, optional*) – only pass valid indexes from random index selector to feature input. Defaults to True.
- **hide_selector** (*bool, optional*) – hide all pos label selectors. Defaults to True.

whatif(*title='What if...'*, *name=None*, *hide_whatifindexselector=False*, *hide_inputeditor=False*, *hide_whatifprediction=False*, *hide_whatifcontributiongraph=False*, *hide_whatifpdp=False*, *hide_whatifcontributiontable=False*, *hide_title=True*, *hide_selector=True*, *index_check=True*, *n_input_cols=4*, *sort='importance'*)

Composite for the whatif component:

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Individual Predictions".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_title** (*bool, optional*) – hide title. Defaults to True.

- **hide_selector** (*bool, optional*) – hide all pos label selectors. Defaults to True.
- **hide_whatifindexselector** (*bool, optional*) – hide ClassifierRandomIndexComponent or RegressionRandomIndexComponent
- **hide_inputeditor** (*bool, optional*) – hide FeatureInputComponent
- **hide_whatifprediction** (*bool, optional*) – hide PredictionSummaryComponent
- **hide_whatifcontributiongraph** (*bool, optional*) – hide ShapContributionsGraphComponent
- **hide_whatifcontributiontable** (*bool, optional*) – hide ShapContributionsTableComponent
- **hide_whatifpdp** (*bool, optional*) – hide PdpComponent
- **index_check** (*bool, optional*) – only pass valid indexes from random index selector to feature input. Defaults to True.
- **n_input_cols** (*int, optional*) – number of columns to divide the feature inputs into. Defaults to 4.
- **sort** (*{'abs', 'high-to-low', 'low-to-high', 'importance'}, optional*) – sorting of shap values. Defaults to 'importance'.

dependence(*title='Shap Dependence'*, *name=None*, *hide_selector=True*, *hide_shapsummary=False*, *hide_shapdependence=False*, *depth=None*)

Composite of ShapSummary and ShapDependence component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Feature Dependence".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_selector** (*bool, optional*) – hide all pos label selectors. Defaults to True.
- **hide_shapsummary** (*bool, optional*) – hide ShapSummaryComponent
- **hide_shapdependence** (*bool, optional*) – ShapDependenceComponent
- **depth** (*int, optional*) – Number of features to display. Defaults to None.

interactions(*title='Shap Interactions'*, *name=None*, *hide_selector=True*, *hide_interactionsummary=False*, *hide_interactiondependence=False*, *depth=None*)

Composite of InteractionSummaryComponent and InteractionDependenceComponent

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Feature Interactions".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_selector** (*bool, optional*) – hide all pos label selectors. Defaults to True.
- **hide_interactionsummary** (*bool, optional*) – hide InteractionSummaryComponent

- **hide_interactiondependence** (*bool, optional*) – hide InteractionDependenceComponent
- **depth** (*int, optional*) – Initial number of features to display. Defaults to None.

decisiontrees(*title='Decision Trees'*, *name=None*, *hide_treeindexselector=False*, *hide_treesgraph=False*, *hide_treepathtable=False*, *hide_treepathgraph=False*, *hide_selector=True*, *index_check=True*)

Composite of decision tree related components:

- index selector
- individual decision trees barchart
- decision path table
- deciion path graph

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either RandomForestClassifierExplainer() or RandomForestRegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Decision Trees".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_treeindexselector** (*bool, optional*) – hide ClassifierRandomIndexComponent or RegressionRandomIndexComponent
- **hide_treesgraph** (*bool, optional*) – hide DecisionTreesComponent
- **hide_treepathtable** (*bool, optional*) – hide DecisionPathTableComponent
- **hide_treepathgraph** (*bool, optional*) – DecisionPathGraphComponent
- **hide_selector** (*bool, optional*) – hide all pos label selectors. Defaults to True.
- **index_check** (*bool, optional*) – only pass valid indexes from random index selector to feature input. Defaults to True.

class explainerdashboard.dashboards.**InlineShapExplainer**(*inline_explainer*, *name*)

overview(*title='Shap Overview'*, *name=None*, *hide_selector=True*, *hide_shapsummary=False*, *hide_shapdependence=False*, *depth=None*)

Composite of ShapSummary and ShapDependence component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Feature Dependence".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_selector** (*bool, optional*) – hide all pos label selectors. Defaults to True.
- **hide_shapsummary** (*bool, optional*) – hide ShapSummaryComponent
- **hide_shapdependence** (*bool, optional*) – ShapDependenceComponent
- **depth** (*int, optional*) – Number of features to display. Defaults to None.

summary(*title='Shap Summary'*, *name=None*, *subtitle='Ordering features by shap value'*, *hide_title=False*, *hide_subtitle=False*, *hide_depth=False*, *hide_type=False*, *hide_index=False*, *hide_selector=False*, *hide_popout=False*, *pos_label=None*, *depth=None*, *summary_type='aggregate'*, *max_cat_colors=5*, *index=None*, *plot_sample=None*, *description=None*)

Shows shap summary component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Shap Dependence Summary".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide the title. Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_depth** (*bool, optional*) – hide the depth toggle. Defaults to False.
- **hide_type** (*bool, optional*) – hide the summary type toggle (aggregated, detailed). Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **depth** (*int, optional*) – initial number of features to show. Defaults to None.
- **summary_type** (*str, {'aggregate', 'detailed'}. optional*) – type of summary graph to show. Defaults to "aggregate".
- **max_cat_colors** (*int, optional*) – for categorical features, maximum number of categories to label with own color. Defaults to 5.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

dependence(*title='Shap Dependence'*, *name=None*, *subtitle='Relationship between feature value and SHAP value'*, *hide_title=False*, *hide_subtitle=False*, *hide_col=False*, *hide_color_col=False*, *hide_index=False*, *hide_selector=False*, *hide_outliers=False*, *hide_cats_topx=False*, *hide_cats_sort=False*, *hide_popout=False*, *hide_footer=False*, *pos_label=None*, *col=None*, *color_col=None*, *index=None*, *remove_outliers=False*, *cats_topx=10*, *cats_sort='freq'*, *max_cat_colors=5*, *plot_sample=None*, *description=None*)

Show shap dependence graph

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Shap Dependence".

- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide component title. Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_col** (*bool, optional*) – hide feature selector. Defaults to False.
- **hide_color_col** (*bool, optional*) – hide color feature selector Defaults to False.
- **hide_index** (*bool, optional*) – hide index selector Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **hide_cats_topx** (*bool, optional*) – hide the categories topx input. Defaults to False.
- **hide_cats_sort** (*bool, optional*) – hide the categories sort selector.Defaults to False.
- **hide_outliers** (*bool, optional*) – Hide remove outliers toggle input. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **col** (*str, optional*) – Feature to display. Defaults to None.
- **color_col** (*str, optional*) – Color plot by values of this Feature. Defaults to None.
- **index** (*int, optional*) – Highlight a particular index. Defaults to None.
- **remove_outliers** (*bool, optional*) – remove outliers in feature and color feature from the plot.
- **cats_topx** (*int, optional*) – maximum number of categories to display for categorical features. Defaults to 10.
- **cats_sort** (*str, optional*) – how to sort categories: 'alphabet', 'freq' or 'shap'. Defaults to 'freq'.
- **max_cat_colors** (*int, optional*) – for categorical features, maximum number of categories to label with own color. Defaults to 5.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

interaction_overview(*title='Interactions Overview'*, *name=None*, *hide_selector=True*, *hide_interactionsummary=False*, *hide_interactiondependence=False*, *depth=None*)

Composite of InteractionSummaryComponent and InteractionDependenceComponent

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Feature Interactions".

- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_selector** (*bool, optional*) – hide all pos label selectors. Defaults to True.
- **hide_interactionsummary** (*bool, optional*) – hide InteractionSummaryComponent
- **hide_interactiondependence** (*bool, optional*) – hide InteractionDependenceComponent
- **depth** (*int, optional*) – Initial number of features to display. Defaults to None.

interaction_summary(*title='Shap Interaction Summary'*, *name=None*, *subtitle='Ordering features by shap interaction value'*, *hide_title=False*, *hide_subtitle=False*, *hide_col=False*, *hide_depth=False*, *hide_type=False*, *hide_index=False*, *hide_popout=False*, *hide_selector=False*, *pos_label=None*, *col=None*, *depth=None*, *summary_type='aggregate'*, *max_cat_colors=5*, *index=None*, *plot_sample=None*, *description=None*)

Show SHAP Interaciton values summary component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Interactions Summary".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide the component title. Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_col** (*bool, optional*) – Hide the feature selector. Defaults to False.
- **hide_depth** (*bool, optional*) – Hide depth toggle. Defaults to False.
- **hide_type** (*bool, optional*) – Hide summary type toggle. Defaults to False.
- **hide_index** (*bool, optional*) – Hide the index selector. Defaults to False
- **hide_popout** (*bool, optional*) – hide popout button
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **col** (*str, optional*) – Feature to show interaction summary for. Defaults to None.
- **depth** (*int, optional*) – Number of interaction features to display. Defaults to None.
- **summary_type** (*str, {'aggregate', 'detailed'}, optional*) – type of summary graph to display. Defaults to "aggregate".
- **max_cat_colors** (*int, optional*) – for categorical features, maximum number of categories to label with own color. Defaults to 5.
- **index** (*str*) – Default index. Defaults to None.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)

- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

interaction_dependence(*title='Shap Interaction Dependence'*, *name=None*, *subtitle='Relation between feature value and shap interaction value'*, *hide_title=False*, *hide_subtitle=False*, *hide_col=False*, *hide_interact_col=False*, *hide_index=False*, *hide_popout=False*, *hide_selector=False*, *hide_outliers=False*, *hide_cats_topx=False*, *hide_cats_sort=False*, *hide_top=False*, *hide_bottom=False*, *pos_label=None*, *col=None*, *interact_col=None*, *remove_outliers=False*, *cats_topx=10*, *cats_sort='freq'*, *max_cat_colors=5*, *plot_sample=None*, *description=None*, *index=None*)

Interaction Dependence Component.

Shows two graphs:

top graph: col vs interact_col bottom graph: interact_col vs col

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Interactions Dependence".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – Hide component title. Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_col** (*bool, optional*) – Hide feature selector. Defaults to False.
- **hide_interact_col** (*bool, optional*) – Hide interaction feature selector. Defaults to False.
- **hide_highlight** (*bool, optional*) – Hide highlight index selector. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **hide_outliers** (*bool, optional*) – Hide remove outliers toggle input. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button
- **hide_cats_topx** (*bool, optional*) – hide the categories topx input. Defaults to False.
- **hide_cats_sort** (*bool, optional*) – hide the categories sort selector. Defaults to False.
- **hide_top** (*bool, optional*) – Hide the top interaction graph (col vs interact_col). Defaults to False.
- **hide_bottom** (*bool, optional*) – hide the bottom interaction graph (interact_col vs col). Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **col** (*str, optional*) – Feature to find interactions for. Defaults to None.
- **interact_col** (*str, optional*) – Feature to interact with. Defaults to None.
- **highlight** (*int, optional*) – Index row to highlight Defaults to None.

- **remove_outliers** (*bool, optional*) – remove outliers in feature and color feature from the plot.
- **cats_topx** (*int, optional*) – number of categories to display for categorical features.
- **cats_sort** (*str, optional*) – how to sort categories: 'alphabet', 'freq' or 'shap'. Defaults to 'freq'.
- **max_cat_colors** (*int, optional*) – for categorical features, maximum number of categories to label with own color. Defaults to 5.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

contributions_graph(*title='Contributions'*, *name=None*, *subtitle='How has each feature contributed to the prediction?'*, *hide_title=False*, *hide_subtitle=False*, *hide_index=False*, *hide_depth=False*, *hide_sort=False*, *hide_orientation=True*, *hide_selector=False*, *hide_popout=False*, *feature_input_component=None*, *index_dropdown=True*, *pos_label=None*, *index=None*, *depth=None*, *sort='high-to-low'*, *orientation='vertical'*, *higher_is_better=True*, *description=None*)

Display Shap contributions to prediction graph component

Parameters

- **explainer** (*Explainer*) – explainer object constructed , with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Contributions Plot".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – Hide component title. Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_index** (*bool, optional*) – Hide index selector. Defaults to False.
- **hide_depth** (*bool, optional*) – Hide depth toggle. Defaults to False.
- **hide_sort** (*bool, optional*) – Hide the sorting dropdown. Defaults to False.
- **hide_orientation** (*bool, optional*) – Hide the orientation dropdown. Defaults to True.
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button
- **feature_input_component** (FeatureInputComponent) – A FeatureInputComponent that will give the input to the graph instead of the index selector. If not None, hide_index=True. Defaults to None.
- **index_dropdown** (*bool, optional*) – Use dropdown for index input instead of free text input. Defaults to True.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **index** (*{int, bool}, optional*) – Initial index to display. Defaults to None.

- **depth** (*int, optional*) – Initial number of features to display. Defaults to None.
- **sort** (*{'abs', 'high-to-low', 'low-to-high', 'importance'}, optional*) – sorting of shap values. Defaults to 'high-to-low'.
- **orientation** (*{'vertical', 'horizontal'}, optional*) – orientation of bar chart. Defaults to 'vertical'.
- **higher_is_better** (*bool, optional*) – Color positive shap values green and negative shap values red, or the reverse.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

contributions_table(*title='Contributions'*, *name=None*, *subtitle='How has each feature contributed to the prediction?'*, *hide_title=False*, *hide_subtitle=False*, *hide_index=False*, *hide_depth=False*, *hide_sort=False*, *hide_selector=False*, *feature_input_component=None*, *index_dropdown=True*, *pos_label=None*, *index=None*, *depth=None*, *sort='abs'*, *description=None*)

Show SHAP values contributions to prediction in a table component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Contributions Table".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – Hide component title. Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_index** (*bool, optional*) – Hide index selector. Defaults to False.
- **hide_depth** (*bool, optional*) – Hide depth selector. Defaults to False.
- **hide_sort** (*bool, optional*) – Hide sorting dropdown. Default to False.
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **feature_input_component** (FeatureInputComponent) – A FeatureInputComponent that will give the input to the graph instead of the index selector. If not None, hide_index=True. Defaults to None.
- **index_dropdown** (*bool, optional*) – Use dropdown for index input instead of free text input. Defaults to True.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **index** (*[type], optional*) – Initial index to display. Defaults to None.
- **depth** (*[type], optional*) – Initial number of features to display. Defaults to None.
- **sort** (*{'abs', 'high-to-low', 'low-to-high', 'importance'}, optional*) – sorting of shap values. Defaults to 'high-to-low'.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

class explainerdashboard.dashboards.**InlineClassifierExplainer**(*inline_explainer*, *name*)

model_stats(*title='Models Stats'*, *name=None*, *hide_title=True*, *hide_selector=True*, *hide_globalcutoff=False*, *hide_modelsummary=False*, *hide_confusionmatrix=False*, *hide_precision=False*, *hide_classification=False*, *hide_rocauc=False*, *hide_prauc=False*, *hide_liftcurve=False*, *hide_cumprecision=False*, *pos_label=None*, *bin_size=0.1*, *quantiles=10*, *cutoff=0.5*)

Composite of multiple classifier related components:

- precision graph
- confusion matrix
- lift curve
- classification graph
- roc auc graph
- pr auc graph

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Decision Trees".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_title** (*bool, optional*) – hide title. Defaults to True.
- **hide_selector** (*bool, optional*) – hide all pos label selectors. Defaults to True.
- **hide_globalcutoff** (*bool, optional*) – hide CutoffPercentileComponent
- **hide_modelsummary** (*bool, optional*) – hide ClassifierModelSummaryComponent
- **hide_confusionmatrix** (*bool, optional*) – hide ConfusionMatrixComponent
- **hide_precision** (*bool, optional*) – hide PrecisionComponent
- **hide_classification** (*bool, optional*) – hide ClassificationComponent
- **hide_rocauc** (*bool, optional*) – hide RocAucComponent
- **hide_prauc** (*bool, optional*) – hide PrAucComponent
- **hide_liftcurve** (*bool, optional*) – hide LiftCurveComponent
- **hide_cumprecision** (*bool, optional*) – hide CumulativePrecisionComponent
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **bin_size** (*float, optional*) – bin_size for precision plot. Defaults to 0.1.
- **quantiles** (*int, optional*) – number of quantiles for precision plot. Defaults to 10.
- **cutoff** (*float, optional*) – initial cutoff. Defaults to 0.5.

precision(*title='Precision Plot'*, *name=None*, *subtitle='Does fraction positive increase with predicted probability?'*, *hide_title=False*, *hide_subtitle=False*, *hide_footer=False*, *hide_cutoff=False*, *hide_binsize=False*, *hide_binmethod=False*, *hide_multiclass=False*, *hide_selector=False*, *hide_popout=False*, *pos_label=None*, *bin_size=0.1*, *quantiles=10*, *cutoff=0.5*, *quantiles_or_binsize='bin_size'*, *multiclass=False*, *description=None*)

Shows a precision graph with toggles.

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Precision Plot".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide title
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_cutoff** (*bool, optional*) – Hide cutoff slider. Defaults to False.
- **hide_binsize** (*bool, optional*) – hide binsize/quantiles slider. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **hide_binmethod** (*bool, optional*) – Hide binsize/quantiles toggle. Defaults to False.
- **hide_multiclass** (*bool, optional*) – Hide multiclass toggle. Defaults to False.
- **hide_selector** – Hide pos label selector. Default to True.
- **hide_popout** (*bool, optional*) – hide popout button
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **bin_size** (*float, optional*) – Size of bins in probability space. Defaults to 0.1.
- **quantiles** (*int, optional*) – Number of quantiles to divide plot. Defaults to 10.
- **cutoff** (*float, optional*) – Cutoff to display in graph. Defaults to 0.5.
- **quantiles_or_binsize** (*str, {'quantiles', 'bin_size'}, optional*) – Default bin method. Defaults to 'bin_size'.
- **multiclass** (*bool, optional*) – Display all classes. Defaults to False.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

cumulative_precision(*title='Cumulative Precision Plot'*, *name=None*, *subtitle='Expected distribution for highest scores'*, *hide_title=False*, *hide_subtitle=False*, *hide_footer=False*, *hide_selector=False*, *hide_popout=False*, *pos_label=None*, *hide_cutoff=False*, *cutoff=None*, *hide_percentile=False*, *percentile=None*, *description=None*)

Show cumulative precision component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Cumulative Precision".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide the title.

- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label

confusion_matrix(*title='Confusion Matrix'*, *name=None*, *subtitle='How many false positives and false negatives?'*, *hide_title=False*, *hide_subtitle=False*, *hide_footer=False*, *hide_cutoff=False*, *hide_percentage=False*, *hide_binary=False*, *hide_selector=False*, *hide_popout=False*, *hide_normalize=False*, *normalize='all'*, *pos_label=None*, *cutoff=0.5*, *percentage=True*, *binary=True*, *description=None*)

Display confusion matrix component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Confusion Matrix".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide title.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_cutoff** (*bool, optional*) – Hide cutoff slider. Defaults to False.
- **hide_percentage** (*bool, optional*) – Hide percentage toggle. Defaults to False.
- **hide_binary** (*bool, optional*) – Hide binary toggle. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **cutoff** (*float, optional*) – Default cutoff. Defaults to 0.5.
- **percentage** (*bool, optional*) – Display percentages instead of counts. Defaults to True.
- **binary** (*bool, optional*) – Show binary instead of multiclass confusion matrix. Defaults to True.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.
- **normalize** (*str['true', 'pred', 'all']*) – normalizes confusion matrix over the true (rows), predicted (columns) conditions or all the population. Defaults to all

lift_curve(*title='Lift Curve'*, *name=None*, *subtitle='Performance how much better than random?'*, *hide_title=False*, *hide_subtitle=False*, *hide_footer=False*, *hide_cutoff=False*, *hide_percentage=False*, *hide_wizard=False*, *hide_selector=False*, *hide_popout=False*, *pos_label=None*, *cutoff=0.5*, *percentage=True*, *wizard=True*, *description=None*)

Show liftcurve component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Lift Curve".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide title.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_cutoff** (*bool, optional*) – Hide cutoff slider. Defaults to False.
- **hide_percentage** (*bool, optional*) – Hide percentage toggle. Defaults to False.
- **hide_wizard** (*bool, optional*) – hide the wizard toggle. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **cutoff** (*float, optional*) – Cutoff for lift curve. Defaults to 0.5.
- **percentage** (*bool, optional*) – Display percentages instead of counts. Defaults to True.
- **wizard** (*bool, optional*) – display the wizard in the graph.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

classification(*title='Classification'*, *name=None*, *subtitle='Distribution of labels above and below cutoff'*, *hide_title=False*, *hide_subtitle=False*, *hide_footer=False*, *hide_cutoff=False*, *hide_percentage=False*, *hide_selector=False*, *hide_popout=False*, *pos_label=None*, *cutoff=0.5*, *percentage=True*, *description=None*)

Shows a barchart of the number of classes above the cutoff and below the cutoff.

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Classification Plot".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide the title.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_cutoff** (*bool, optional*) – Hide cutoff slider. Defaults to False.
- **hide_percentage** (*bool, optional*) – Hide percentage toggle. Defaults to False.

- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **cutoff** (*float, optional*) – Cutoff for prediction. Defaults to 0.5.
- **percentage** (*bool, optional*) – Show percentage instead of counts. Defaults to True.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

roc_auc(*title='ROC AUC Curve'*, *name=None*, *subtitle='Trade-off between False positives and false negatives'*, *hide_title=False*, *hide_subtitle=False*, *hide_footer=False*, *hide_cutoff=False*, *hide_selector=False*, *hide_popout=False*, *pos_label=None*, *cutoff=0.5*, *description=None*)

Show ROC AUC curve component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "ROC AUC Plot".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide title.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_cutoff** (*bool, optional*) – Hide cutoff slider. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **cutoff** (*float, optional*) – default cutoff. Defaults to 0.5.

pr_auc(*title='PR AUC Curve'*, *name=None*, *subtitle='Trade-off between Precision and Recall'*, *hide_title=False*, *hide_subtitle=False*, *hide_footer=False*, *hide_cutoff=False*, *hide_selector=False*, *hide_popout=False*, *pos_label=None*, *cutoff=0.5*, *description=None*)

Display PR AUC plot component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "PR AUC Plot".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide title.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.

- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_cutoff** (*bool, optional*) – hide cutoff slider. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **cutoff** (*float, optional*) – default cutoff. Defaults to 0.5.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

class explainerdashboard.dashboards.**InlineRegressionExplainer**(*inline_explainer*, *name*)

model_stats(*title='Models Stats'*, *name=None*, *hide_title=True*, *hide_modelsummary=False*, *hide_predsvsactual=False*, *hide_residuals=False*, *hide_regvscol=False*, *logs=False*, *pred_or_actual='vs_pred'*, *residuals='difference'*, *col=None*)

Composite for displaying multiple regression related graphs:

- predictions vs actual plot
- residual plot
- residuals vs feature

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Regression Stats".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_title** (*bool, optional*) – hide title. Defaults to True.
- **hide_modelsummary** (*bool, optional*) – hide RegressionModelSummaryComponent
- **hide_predsvsactual** (*bool, optional*) – hide PredictedVsActualComponent
- **hide_residuals** (*bool, optional*) – hide ResidualsComponent
- **hide_regvscol** (*bool, optional*) – hide RegressionVsColComponent
- **logs** (*bool, optional*) – Use log axis. Defaults to False.
- **pred_or_actual** (*str, optional*) – plot residuals vs predictions or vs y (actual). Defaults to "vs_pred".
- **residuals** (*str, {'difference', 'ratio', 'log-ratio'} optional*) – How to calcualte residuals. Defaults to 'difference'.
- **col** (*{str, int}, optional*) – Feature to use for residuals plot. Defaults to None.

pred_vs_actual(*title='Predicted vs Actual'*, *name=None*, *subtitle='How close is the predicted value to the observed?'*, *hide_title=False*, *hide_subtitle=False*, *hide_log_x=False*, *hide_log_y=False*, *hide_popout=False*, *logs=False*, *log_x=False*, *log_y=False*, *round=3*, *plot_sample=None*, *description=None*)

Shows a plot of predictions vs y.

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Predicted vs Actual".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) –
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_log_x** (*bool, optional*) – Hide the log_x toggle. Defaults to False.
- **hide_log_y** (*bool, optional*) – Hide the log_y toggle. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **logs** (*bool, optional*) – Whether to use log axis. Defaults to False.
- **log_x** (*bool, optional*) – log only x axis. Defaults to False.
- **log_y** (*bool, optional*) – log only y axis. Defaults to False.
- **round** (*int, optional*) – rounding to apply to float predictions. Defaults to 3.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

residuals(*title='Residuals'*, *name=None*, *subtitle='How much is the model off?'*, *hide_title=False*, *hide_subtitle=False*, *hide_footer=False*, *hide_pred_or_actual=False*, *hide_ratio=False*, *hide_popout=False*, *pred_or_actual='vs_pred'*, *residuals='difference'*, *round=3*, *plot_sample=None*, *description=None*)

Residuals plot component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Residuals".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) –
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_pred_or_actual** (*bool, optional*) – hide vs predictions or vs actual for x-axis toggle. Defaults to False.
- **hide_ratio** (*bool, optional*) – hide residual type dropdown. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **pred_or_actual** (*str, {'vs_actual', 'vs_pred'}, optional*) – Whether to plot actual or predictions on the x-axis. Defaults to "vs_pred".

- **residuals** (*str, {'difference', 'ratio', 'log-ratio'} optional*) – How to calcualte residuals. Defaults to 'difference'.
- **round** (*int, optional*) – rounding to apply to float predictions. Defaults to 3.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

plots_vs_col(*title='Plots vs col'*, *name=None*, *subtitle='Are predictions and residuals correlated with features?'*, *hide_title=False*, *hide_subtitle=False*, *hide_footer=False*, *hide_col=False*, *hide_ratio=False*, *hide_points=False*, *hide_winsor=False*, *hide_cats_topx=False*, *hide_cats_sort=False*, *hide_popout=False*, *col=None*, *display='difference'*, *round=3*, *points=True*, *winsor=0*, *cats_topx=10*, *cats_sort='freq'*, *plot_sample=None*, *description=None*)

Show residuals, observed or preds vs a particular Feature component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Plot vs feature".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) –
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_col** (*bool, optional*) – Hide de column selector. Defaults to False.
- **hide_ratio** (*bool, optional*) – Hide the toggle. Defaults to False.
- **hide_points** (*bool, optional*) – Hide group points toggle. Defaults to False.
- **hide_winsor** (*bool, optional*) – Hide winsor input. Defaults to False.
- **hide_cats_topx** (*bool, optional*) – hide the categories topx input. Defaults to False.
- **hide_cats_sort** (*bool, optional*) – hide the categories sort selector.Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **col** (*[type], optional*) – Initial feature to display. Defaults to None.
- **display** (*str, {'observed', 'predicted', difference', 'ratio', 'log-ratio'} optional*) – What to display on y axis. Defaults to 'difference'.
- **round** (*int, optional*) – rounding to apply to float predictions. Defaults to 3.
- **points** (*bool, optional*) – display point cloud next to violin plot for categorical cols. Defaults to True
- **winsor** (*int, 0-50, optional*) – percentage of outliers to winsor out of the y-axis. Defaults to 0.
- **cats_topx** (*int, optional*) – maximum number of categories to display for categorical features. Defaults to 10.

- **cats_sort** (*str, optional*) – how to sort categories: 'alphabet', 'freq' or 'shap'. Defaults to 'freq'.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

class explainerdashboard.dashboards.**InlineDecisionTreesExplainer**(*inline_explainer*, *name*)

overview(*title='Decision Trees'*, *name=None*, *hide_treeindexselector=False*, *hide_treesgraph=False*, *hide_treepathtable=False*, *hide_treepathgraph=False*, *hide_selector=True*, *index_check=True*)

Composite of decision tree related components:

- index selector
- individual decision trees barchart
- decision path table
- deciion path graph

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either RandomForestClassifierExplainer() or RandomForestRegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Decision Trees".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_treeindexselector** (*bool, optional*) – hide ClassifierRandomIndexComponent or RegressionRandomIndexComponent
- **hide_treesgraph** (*bool, optional*) – hide DecisionTreesComponent
- **hide_treepathtable** (*bool, optional*) – hide DecisionPathTableComponent
- **hide_treepathgraph** (*bool, optional*) – DecisionPathGraphComponent
- **hide_selector** (*bool, optional*) – hide all pos label selectors. Defaults to True.
- **index_check** (*bool, optional*) – only pass valid indexes from random index selector to feature input. Defaults to True.

decisiontrees(*title='Decision Trees'*, *name=None*, *subtitle='Displaying individual decision trees'*, *hide_title=False*, *hide_subtitle=False*, *hide_index=False*, *hide_highlight=False*, *hide_selector=False*, *hide_popout=False*, *index_dropdown=True*, *pos_label=None*, *index=None*, *highlight=None*, *higher_is_better=True*, *description=None*)

Show prediction from individual decision trees inside RandomForest component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Decision Trees".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle

- **hide_title** (*bool, optional*) – hide title, Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_index** (*bool, optional*) – Hide index selector. Defaults to False.
- **hide_highlight** (*bool, optional*) – Hide tree highlight selector. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selectors. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button
- **index_dropdown** (*bool, optional*) – Use dropdown for index input instead of free text input. Defaults to True.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **index** (*{str, int}, optional*) – Initial index to display. Defaults to None.
- **highlight** (*int, optional*) – Initial tree to highlight. Defaults to None.
- **higher_is_better** (*bool, optional*) – up is green, down is red. If False flip the colors. (for gbm models only)
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

decisionpath_table(*title='Decision path'*, *name=None*, *subtitle='Decision path through decision tree'*, *hide_title=False*, *hide_subtitle=False*, *hide_index=False*, *hide_highlight=False*, *hide_selector=False*, *index_dropdown=True*, *pos_label=None*, *index=None*, *highlight=None*, *description=None*)

Display a table of the decision path through a particular decision tree

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Decision path table".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide title, Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_index** (*bool, optional*) – Hide index selector. Defaults to False.
- **hide_highlight** (*bool, optional*) – Hide tree index selector. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selectors. Defaults to False.
- **index_dropdown** (*bool, optional*) – Use dropdown for index input instead of free text input. Defaults to True.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **index** (*{str, int}, optional*) – Initial index to display decision path for. Defaults to None.
- **highlight** (*int, optional*) – Initial tree idx to display decision path for. Defaults to None.

- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

decisionpath_graph(*title='Decision path'*, *name=None*, *subtitle='Decision path through decision tree'*, *hide_title=False*, *hide_subtitle=False*, *hide_index=False*, *hide_highlight=False*, *hide_selector=False*, *index_dropdown=True*, *pos_label=None*, *index=None*, *highlight=None*, *description=None*)

Display a table of the decision path through a particular decision tree

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Decision path table".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide title, Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_index** (*bool, optional*) – Hide index selector. Defaults to False.
- **hide_highlight** (*bool, optional*) – Hide tree index selector. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selectors. Defaults to False.
- **index_dropdown** (*bool, optional*) – Use dropdown for index input instead of free text input. Defaults to True.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **index** (*{str, int}, optional*) – Initial index to display decision path for. Defaults to None.
- **highlight** (*int, optional*) – Initial tree idx to display decision path for. Defaults to None.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

4.5 explainerdashboard CLI

The library comes with a `explainerdashboard` command line tool (CLI) that you can use to build and run explainerdashboards from your terminal. This makes it easy to start a dashboard without having to run python code or start a notebook first. You can also use it to build explainer objects as part of a CI/CD flow.

4.5.1 Run dashboard from stored explainer

In order to run a dashboard from a stored explainer from the commandline, we first need to store an explainer to disk. You can do this with:

```
explainer = ClassifierExplainer(model, X, y)
explainer.dump("explainer.joblib")
```

And then you can run the default dashboard and launch a browser tab from the command line by running:

```
$ explainerdashboard run explainer.joblib
```

The CLI uses the `waitress` web server by default to run your dashboard. To run on a specific port, not launch a browser or show help:

```
$ explainerdashboard run explainer.joblib --port 8051
$ explainerdashboard run explainer.joblib --no-browser
$ explainerdashboard run --help
```

4.5.2 Run custom dashboard from dashboard.yaml

If you'd like to launch a custom dashboard with custom tabs and parameters, you can do so by storing the configuration to *.yaml*:

```
db = ExplainerDashboard(explainer, [ShapDependenceTab, "importances"],
        port=9000, title="Custom Dashboard", header_hide_title=True)
db.to_yaml("dashboard.yaml", explainerfile="explainer.joblib")
```

You can edit `dashboard.yaml` directly to make further configuration changes if you wish. Start the dashboard from the commandline with:

```
$ explainerdashboard run dashboard.yaml
```

4.5.3 Building explainers from explainer.yaml

You can build also explainers from the commandline by storing the model (e.g. `model.pkl`) and datafile (e.g. `data.csv`), indicating which column is `y` (e.g. `'Survival'`), and which is the index (e.g. `'Name'`), along with the other parameters of the explainer.

You can get this configuration by storing the configuration as before:

```
explainer = ClassifierExplainer(model, X, y,
                labels=['Not survived', 'Survived'])
pickle.dump(model, open("model.pkl", "wb))

explainer.to_yaml("explainer.yaml",
            explainerfile="explainer.joblib",
            modelfile="model.pkl",
            datafile="data.csv",
            target_col="Survival",
            index_col="Name",
            dashboard_yaml="dashboard.yaml")
```

You can then build the `explainer.joblib` file by running:

```
$ explainerdashboard build explainer.yaml
```

This will load the model and dataset, construct an explainer, construct the custom dashboard, calculate all properties needed for that specific dashboard, and store the explainer to disk. This can be useful when you for example would like to populate the dashboard with a new set of data: you can simply update `data.csv` and run `explainerdashboard build`. To start the dashboard you can then run:

```
$ explainerdashboard run dashboard.yaml
```

To build the explainer for a specific dashboard (other than the one specified in `dashboard_yaml`, pass it as a second argument:

```
$ explainerdashboard build explainer.yaml dashboard.yaml
```

Note: If you use the default naming scheme of `explainer.joblib`, `dashboard.yaml` and `explainer.yaml`, you can omit these arguments and simply run e.g.:

```
$ explainerdashboard build
$ explainerdashboard run
```

4.5.4 dump, from_file, to_yaml

Explainer.dump()

BaseExplainer.**dump**(*filepath*)

Dump the current Explainer to file. Depending on the suffix of the filepath will either dump with pickle ('.pkl'), dill ('.dill') or joblib ('joblib').

If no suffix given, will dump with joblib and add '.joblib'

Parameters

filepath (*str, Path*) – filepath where to save the Explainer.

Explainer.from_file()

classmethod BaseExplainer.**from_file**(*filepath*)

Load an Explainer from file. Depending on the suffix of the filepath will either load with pickle ('.pkl'), dill ('.dill') or joblib ('joblib').

If no suffix given, will try with joblib.

Parameters

- **{str** (*filepath*) –
- **Explainer** (*Path} the location of the stored*) –

Returns

Explainer object

Explainer.to_yaml()

BaseExplainer.**to_yaml**(*filepath=None*, *return_dict=False*, *modelfile='model.pkl'*, *datafile='data.csv'*, *index_col=None*, *target_col=None*, *explainerfile='explainer.joblib'*, *dashboard_yaml='dashboard.yaml'*)

Returns a yaml configuration for the current Explainer that can be used by the explainerdashboard CLI. Recommended filename is *explainer.yaml*.

Parameters

- **filepath** (*{str, Path}, optional*) – Filepath to dump yaml. If None returns the yaml as a string. Defaults to None.
- **return_dict** (*bool, optional*) – instead of yaml return dict with config.
- **modelfile** (*str, optional*) – filename of model dump. Defaults to *model.pkl*
- **datafile** (*str, optional*) – filename of datafile. Defaults to *data.csv*.
- **index_col** (*str, optional*) – column to be used for idxs. Defaults to self.idxs.name.
- **target_col** (*str, optional*) – column to be used for to split X and y from datafile. Defaults to self.target.
- **explainerfile** (*str, optional*) – filename of explainer dump. Defaults to *explainer.joblib*.
- **dashboard_yaml** (*str, optional*) – filename of the dashboard.yaml configuration file. This will be used to determine which properties to calculate before storing to disk. Defaults to *dashboard.yaml*.

ExplainerDashboard.to_yaml()

ExplainerDashboard.**to_yaml**(*filepath=None*, *return_dict=False*, *explainerfile='explainer.joblib'*, *dump_explainer=False*, *explainerfile_absolute_path=None*)

Returns a yaml configuration of the current ExplainerDashboard that can be used by the explainerdashboard CLI or to reinstate an identical dashboard from the (dumped) explainer and saved configuration. Recommended filename is *dashboard.yaml*.

Parameters

- **filepath** (*{str, Path}, optional*) – Filepath to dump yaml. If None returns the yaml as a string. Defaults to None.
- **return_dict** (*bool, optional*) – instead of yaml return dict with config.
- **explainerfile** (*str, optional*) – filename of explainer dump. Defaults to *explainer.joblib*. Should en in either .joblib .dill or .pkl
- **dump_explainer** (*bool, optional*) – dump the explainer along with the yaml. You must pass explainerfile parameter for the filename. Defaults to False.
- **explainerfile_absolute_path** (*str, Path, bool, optional*) – absolute path to save explainerfile if not in the same directory as the yaml file. You can also pass True which still saves the explainerfile to the filepath directory, but adds an absolute path to the yaml file.

ExplainerDashboard.from_config

classmethod ExplainerDashboard.**from_config**(*arg1*, *arg2=None*, ***update_params*)

Loading a dashboard from a configuration .yaml file. You can either pass both an explainer and a yaml file generated with ExplainerDashboard.to_yaml("dashboard.yaml"):

db = ExplainerDashboard.from_config(explainer, "dashboard.yaml")

When you specify an explainerfile in to_yaml with ExplainerDashboard.to_yaml("dashboard.yaml", explainerfile="explainer.joblib"), you can also pass just the .yaml:

db = ExplainerDashboard.from_config("dashboard.yaml")

You can also load the explainerfile seperately:

db = ExplainerDashboard.from_config("explainer.joblib", "dashboard.yaml")

Parameters

- **arg1** (*explainer or config*) – arg1 should either be a config (yaml or dict), or an explainer (instance or str/Path).
- **arg2** (*[type], optional*) – If arg1 is an explainer, arg2 should be config.
- **update_params** (*dict*) – You can override parameters in the the yaml config by passing additional kwargs to .from_config()

Returns

ExplainerDashboard

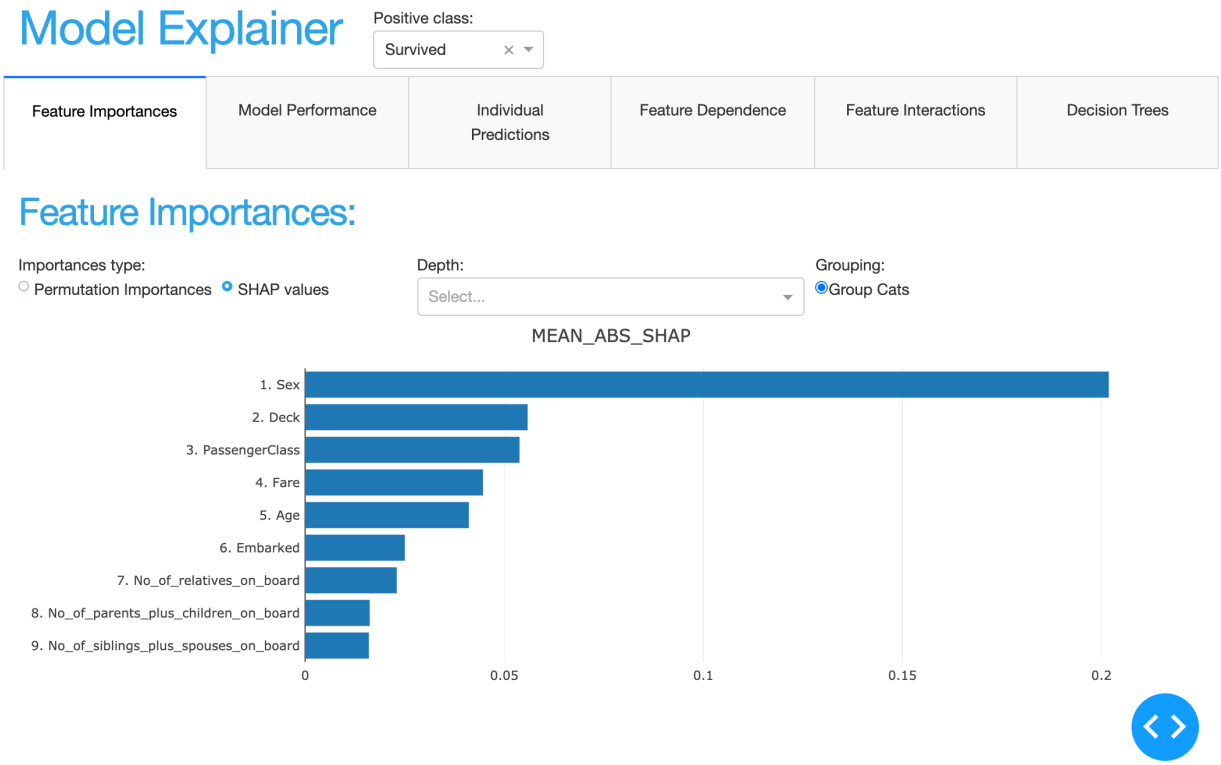
4.6 ExplainerTabs

There are seven tabs that make up the default``ExplainerDashboard``:

```
from explainerdashboard.custom import (ImportancesComposite,
                                       ModelSummaryComposite,
                                       IndividualPredictionsComposite,
                                       WhatIfComposite,
                                       ShapDependenceComposite,
                                       ShapInteractionsComposite,
                                       DecisionTreesComposite)
```

The definitions can be found in the github repo and can serve as a nice starting point for designing your own custom tabs.

4.6.1 ImportancesComposite



class explainerdashboard.dashboard_components.composites.**ImportancesComposite**(*explainer*, *title='Feature Importances'*, *name=None*, *hide_title=True*, *hide_importances=False*, *hide_descriptions=False*, *hide_selector=True*, ***kwargs*)

Overview tab of feature importances

Can show both permutation importances and mean absolute shap values.

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Feature Importances".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_title** (*bool, optional*) – hide the title
- **hide_importances** (*bool, optional*) – hide the ImportancesComponent
- **hide_descriptions** (*bool, optional*) – hide the FeatureDescriptionsComponent

- **hide_selector** (*bool, optional*) – hide the post label selector. Defaults to True.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

4.6.2 ClassifierModelStatsComposite

class explainerdashboard.dashboard_components.composites.**ClassifierModelStatsComposite**(*explainer*, *title='Classification Stats'*, *name=None*, *hide_title=True*, *hide_selector=True*, *hide_globalcutoff=Fa*
hide_modelsummary=
hide_confusionmatrix
hide_precision=False
hide_classification=F
hide_rocauc=False, *hide_prauc=False*, *hide_liftcurve=False*,
hide_cumprecision=F
pos_label=None, *bin_size=0.1*, *quantiles=10*, *cutoff=0.5*, ***kwargs*)

Composite of multiple classifier related components:

- precision graph
- confusion matrix
- lift curve
- classification graph
- roc auc graph
- pr auc graph

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Decision Trees".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_title** (*bool, optional*) – hide title. Defaults to True.
- **hide_selector** (*bool, optional*) – hide all pos label selectors. Defaults to True.
- **hide_globalcutoff** (*bool, optional*) – hide CutoffPercentileComponent
- **hide_modelsummary** (*bool, optional*) – hide ClassifierModelSummaryComponent
- **hide_confusionmatrix** (*bool, optional*) – hide ConfusionMatrixComponent
- **hide_precision** (*bool, optional*) – hide PrecisionComponent
- **hide_classification** (*bool, optional*) – hide ClassificationComponent
- **hide_rocauc** (*bool, optional*) – hide RocAucComponent

- **hide_prauc** (*bool, optional*) – hide PrAucComponent
- **hide_liftcurve** (*bool, optional*) – hide LiftCurveComponent
- **hide_cumprecision** (*bool, optional*) – hide CumulativePrecisionComponent
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **bin_size** (*float, optional*) – bin_size for precision plot. Defaults to 0.1.
- **quantiles** (*int, optional*) – number of quantiles for precision plot. Defaults to 10.
- **cutoff** (*float, optional*) – initial cutoff. Defaults to 0.5.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

4.6.3 RegressionModelStatsComposite

class explainerdashboard.dashboard_components.composites.**RegressionModelStatsComposite**(*explainer*, *title='Regression Stats'*, *name=None*, *hide_title=True*, *hide_modelsummary=*, *hide_predsvsactual=*, *hide_residuals=False*, *hide_regvscol=False*, *logs=False*, *pred_or_actual='vs_*, *residuals='difference'*, *col=None*, ***kwargs*)

Composite for displaying multiple regression related graphs:

- predictions vs actual plot
- residual plot
- residuals vs feature

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Regression Stats".

- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_title** (*bool, optional*) – hide title. Defaults to True.
- **hide_modelsummary** (*bool, optional*) – hide RegressionModelSummaryComponent
- **hide_predsvsactual** (*bool, optional*) – hide PredictedVsActualComponent
- **hide_residuals** (*bool, optional*) – hide ResidualsComponent
- **hide_regvscol** (*bool, optional*) – hide RegressionVsColComponent
- **logs** (*bool, optional*) – Use log axis. Defaults to False.
- **pred_or_actual** (*str, optional*) – plot residuals vs predictions or vs y (actual). Defaults to "vs_pred".
- **residuals** (*str, {'difference', 'ratio', 'log-ratio'} optional*) – How to calcualte residuals. Defaults to 'difference'.
- **col** (*{str, int}, optional*) – Feature to use for residuals plot. Defaults to None.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

4.6.4 IndividualPredictionsComposite

Model Explainer

Positive class: Survived

Feature Importances | Model Performance | Individual Predictions | Feature Dependence | Feature Interactions | Decision Trees

Select index:

Glynn, Miss. Mary Agatha

RANDOM INDEX

Include labels (y): Not survived, Survived

Predictions range: 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9

Use predictions / Use percentiles

Predictions summary:

Index: Glynn, Miss. Mary Agatha

Show Percentile: Show percentile

Outcome: Survived

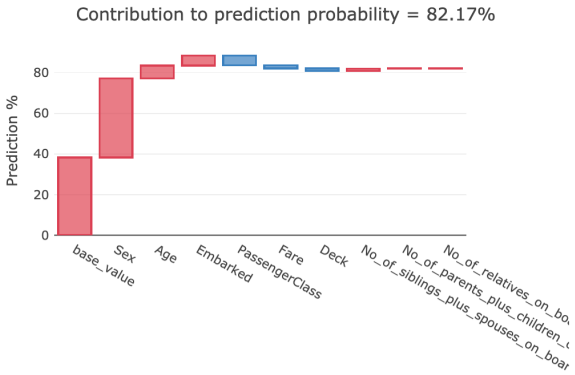
Prediction probabilities per label:

- Not survived: 17.83%
- Survived: 82.17%

In top 16.0% percentile probability Survived

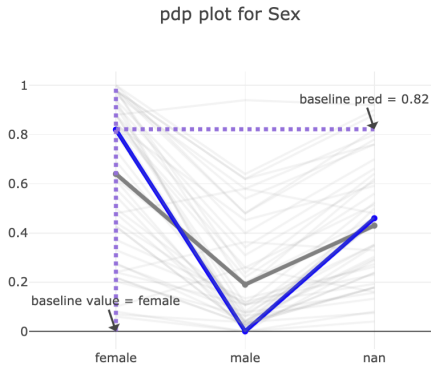
Contributions to prediction:

Index: Glynn, Miss. Mary A... Depth: Select... Grouping: Group Cats



Partial Dependence Plot:

Feature: Sex Index: Glynn, Miss. Mary A... Grouping: Group Cats



Drop na: Drop na's | pdp sample size: 100 | gridlines: 50 | gridpoints: 10

Contributions to prediction:

Display contributions for: Glynn, Miss. Mary A... Depth: Select... Grouping: Group Cats

Reason	Effect
Average of population	38.27%
Sex = female	+38.89%
Age = -999.0	+6.36%
Embarked = Queenstown	+4.88%
PassengerClass = 3	-4.81%
Fare = 7.75	-1.54%
Deck = Unkown	-1.26%
No_of_siblings_plus_spouses_on_board = 0	+1.18%
No_of_parents_plus_children_on_board = 0	+0.15%
No_of_relatives_on_board = 0	+0.05%
Final prediction	82.17%

class explainerdashboard.dashboard_components.composites.**IndividualPredictionsComposite**(*explainer*, *title='Individual Predictions'*, *name=None*, *hide_predindexselec*
hide_predictionsum
hide_contributiongr
hide_pdp=False, *hide_contributionta*
hide_title=False, *hide_selector=True*, *index_check=True*, ***kwargs*)

Composite for a number of component that deal with individual predictions:

- random index selector
- prediction summary
- shap contributions graph
- shap contribution table
- pdp graph

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Individual Predictions".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_predindexselector** (*bool, optional*) – hide ClassifierRandomIndexComponent or RegressionRandomIndexComponent
- **hide_predictionsummary** (*bool, optional*) – hide ClassifierPredictionSummaryComponent or RegressionPredictionSummaryComponent
- **hide_contributiongraph** (*bool, optional*) – hide ShapContributionsGraphComponent
- **hide_pdp** (*bool, optional*) – hide PdpComponent
- **hide_contributiontable** (*bool, optional*) – hide ShapContributionsTableComponent
- **hide_title** (*bool, optional*) – hide title. Defaults to False.
- **index_check** (*bool, optional*) – only pass valid indexes from random index selector to feature input. Defaults to True.
- **hide_selector** (*bool, optional*) – hide all pos label selectors. Defaults to True.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

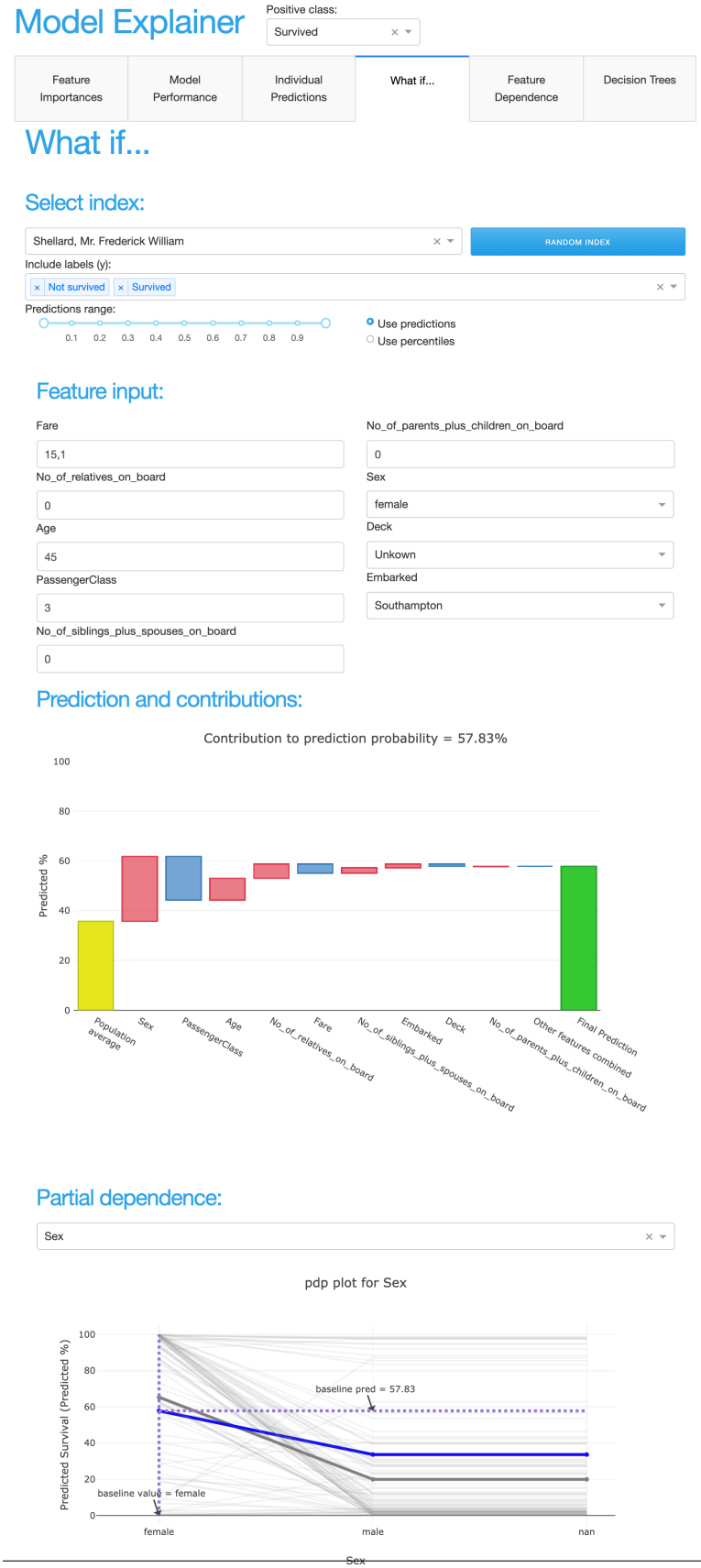
to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

4.6.5 WhatIfComposite



class explainerdashboard.dashboard_components.composites.**WhatIfComposite**(*explainer*, *title='What if...'*, *name=None*, *hide_whatifindexselector=False*, *hide_inputeditor=False*, *hide_whatifprediction=False*, *hide_whatifcontributiongraph=False*, *hide_whatifpdp=False*, *hide_whatifcontributiontable=False*, *hide_title=True*, *hide_selector=True*, *index_check=True*, *n_input_cols=4*, *sort='importance'*, ***kwargs*)

Composite for the whatif component:

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Individual Predictions".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_title** (*bool, optional*) – hide title. Defaults to True.
- **hide_selector** (*bool, optional*) – hide all pos label selectors. Defaults to True.
- **hide_whatifindexselector** (*bool, optional*) – hide ClassifierRandomIndexComponent or RegressionRandomIndexComponent
- **hide_inputeditor** (*bool, optional*) – hide FeatureInputComponent
- **hide_whatifprediction** (*bool, optional*) – hide PredictionSummaryComponent
- **hide_whatifcontributiongraph** (*bool, optional*) – hide ShapContributionsGraphComponent
- **hide_whatifcontributiontable** (*bool, optional*) – hide ShapContributionsTableComponent
- **hide_whatifpdp** (*bool, optional*) – hide PdpComponent
- **index_check** (*bool, optional*) – only pass valid indexes from random index selector to feature input. Defaults to True.
- **n_input_cols** (*int, optional*) – number of columns to divide the feature inputs into. Defaults to 4.
- **sort** (*{'abs', 'high-to-low', 'low-to-high', 'importance'}, optional*) – sorting of shap values. Defaults to 'importance'.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

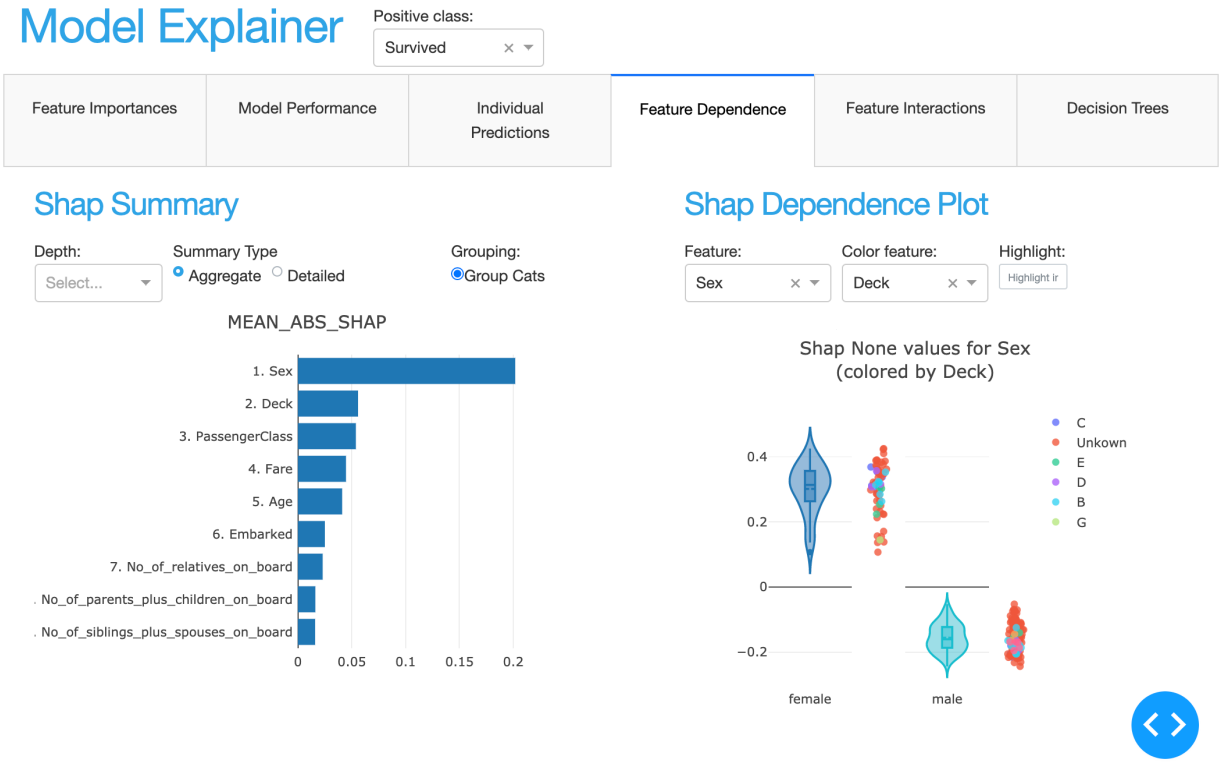
to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

4.6.6 ShapDependenceComposite



class explainerdashboard.dashboard_components.composites.**ShapDependenceComposite**(*explainer*, *title='Feature Dependence'*, *name=None*, *hide_selector=True*, *hide_shapsummary=False*, *hide_shapdependence=False*, *depth=None*, ***kwargs*)

Composite of ShapSummary and ShapDependence component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Feature Dependence".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.

- **hide_selector** (*bool, optional*) – hide all pos label selectors. Defaults to True.
- **hide_shapsummary** (*bool, optional*) – hide ShapSummaryComponent
- **hide_shapdependence** (*bool, optional*) – ShapDependenceComponent
- **depth** (*int, optional*) – Number of features to display. Defaults to None.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

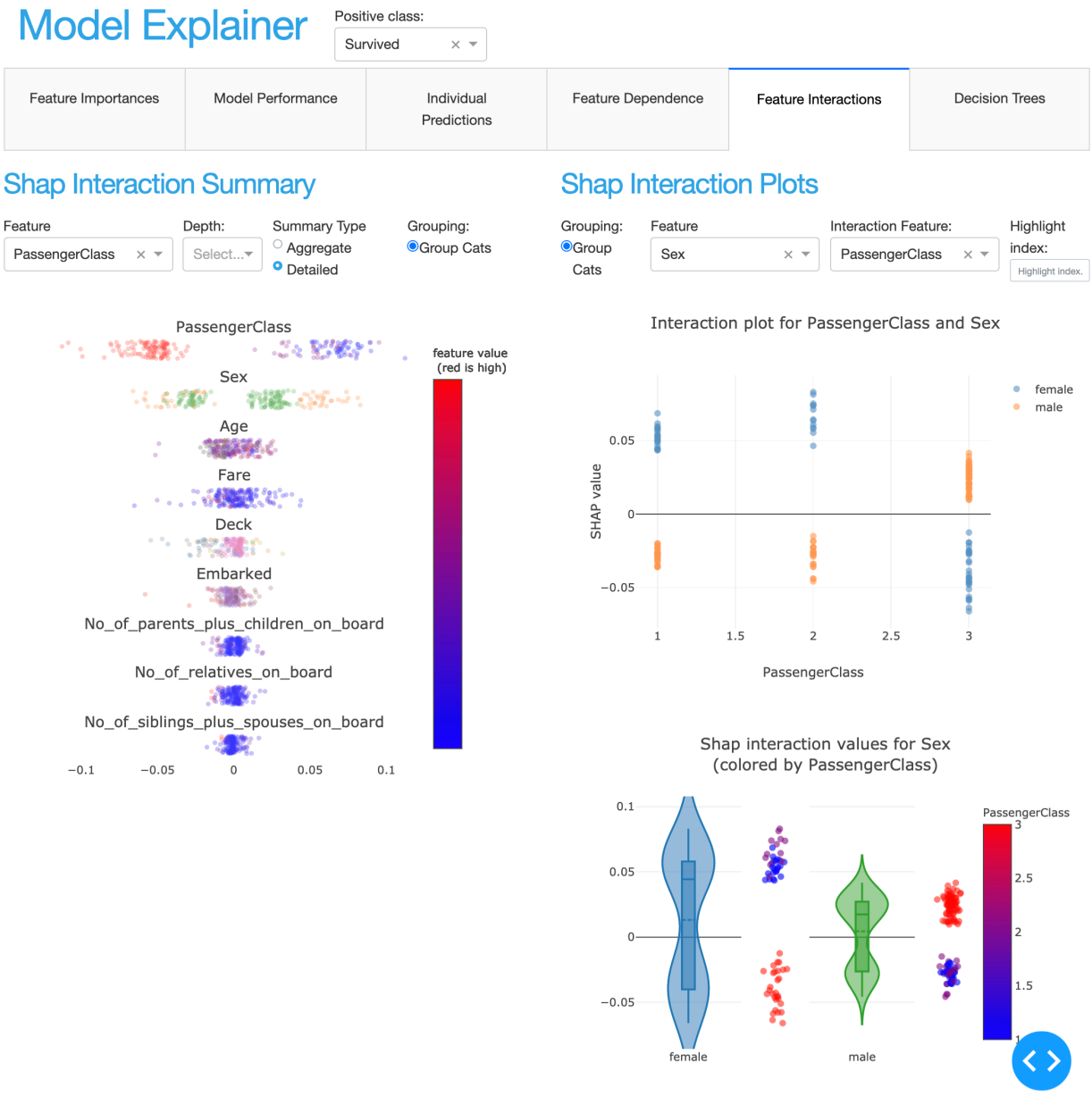
to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

4.6.7 ShapInteractionsComposite



class explainerdashboard.dashboard_components.composites.**ShapInteractionsComposite**(*explainer*, *title='Feature Interactions'*, *name=None*, *hide_selector=True*, *hide_interactionsummary=*
hide_interactiondependenc
depth=None, ***kwargs*)

Composite of InteractionSummaryComponent and InteractionDependenceComponent

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Feature Interactions".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_selector** (*bool, optional*) – hide all pos label selectors. Defaults to True.
- **hide_interactionsummary** (*bool, optional*) – hide InteractionSummaryComponent
- **hide_interactiondependence** (*bool, optional*) – hide InteractionDependenceComponent
- **depth** (*int, optional*) – Initial number of features to display. Defaults to None.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

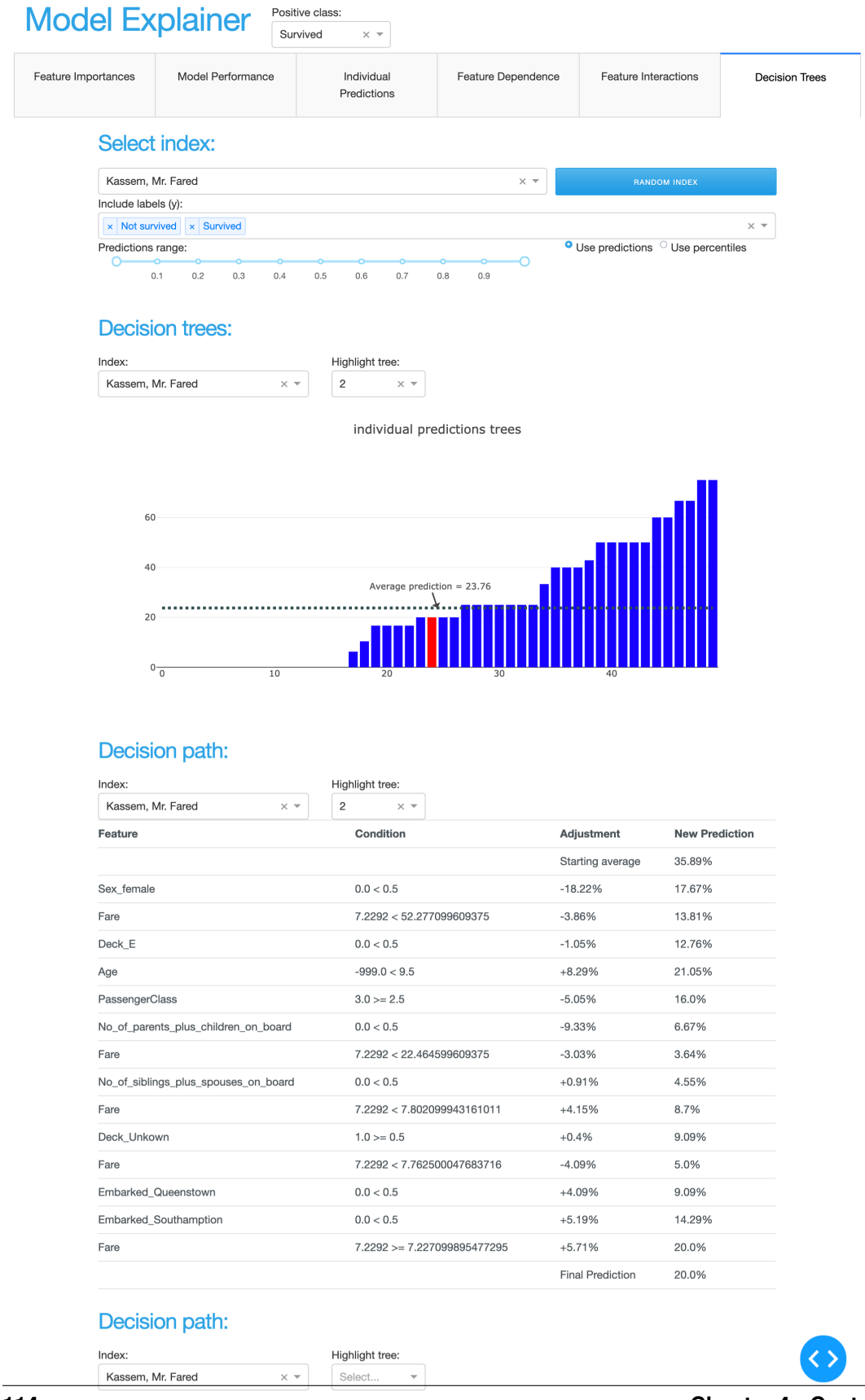
to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

4.6.8 DecisionTreesComposite



class explainerdashboard.dashboard_components.composites.**DecisionTreesComposite**(*explainer*, *title='Decision Trees'*, *name=None*, *hide_treeindexselector=False*, *hide_treesgraph=False*, *hide_treepathtable=False*, *hide_treepathgraph=False*, *hide_selector=True*, *index_check=True*, ***kwargs*)

Composite of decision tree related components:

- index selector
- individual decision trees barchart
- decision path table
- deciion path graph

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either RandomForestClassifierExplainer() or RandomForestRegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Decision Trees".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_treeindexselector** (*bool, optional*) – hide ClassifierRandomIndexComponent or RegressionRandomIndexComponent
- **hide_treesgraph** (*bool, optional*) – hide DecisionTreesComponent
- **hide_treepathtable** (*bool, optional*) – hide DecisionPathTableComponent
- **hide_treepathgraph** (*bool, optional*) – DecisionPathGraphComponent
- **hide_selector** (*bool, optional*) – hide all pos label selectors. Defaults to True.
- **index_check** (*bool, optional*) – only pass valid indexes from random index selector to feature input. Defaults to True.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

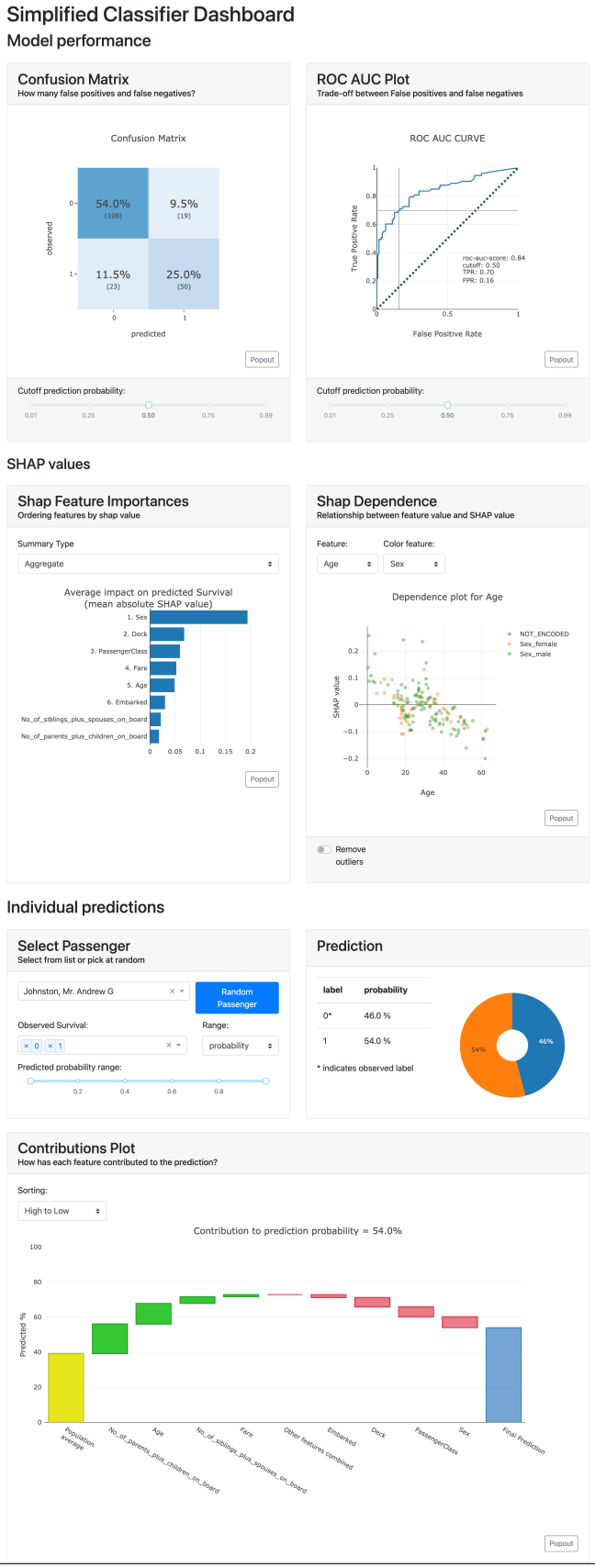
to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

4.6.9 SimplifiedClassifierComposite



You can also load this composite with:

```
explainer = ClassifierExplainer(model, X, y)
ExplainerDashboard(explainer, simple=True)
```

class explainerdashboard.dashboard_components.composites.**SimplifiedClassifierComposite**(*explainer*, *title='Simple Classifier Explainer'*, *name=None*, *hide_title=False*, *classifier_custom_compone*
hide_confusionmatrix
hide_classifier_custo
hide_shapsummary=
hide_shapdependenc
hide_predindexselect
hide_predictionsumm
hide_contributiongra
***kwargs*)

Composite of multiple classifier related components, on a single tab:

- confusion matrix
- **one other model quality indicator: choose from pr auc graph, precision graph,** lift curve, classification graph, or roc auc graph
- shap importance
- shap dependence
- index selector
- index prediction summary
- index shap contribution graph

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Simple Classification Stats".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_title** (*bool, optional*) – hide the title. Defaults to False.
- **classifier_custom_component** (*str, optional*) – custom classifier quality indicator supported by the ClassifierExplainer object. Valid values are: 'roc_auc', 'metrics', pr_auc', 'precision_graph', 'lift_curve', 'classification'. Defaults to 'roc_auc'.
- **hide_confusionmatrix** (*bool, optional*) – hide ConfusionMatrixComponent

- **hide_classifier_custom_component** (*bool, optional*) – hide the chosen classifier_custom_component
- **hide_shapsummary** (*bool, optional*) – hide ShapSummaryComponent
- **hide_shapdependence** (*bool, optional*) – hide ShapDependenceComponent
- **hide_predindexselector** (*bool, optional*) – hide ClassifierRandomIndexComponent or RegressionRandomIndexComponent
- **hide_predictionsummary** (*bool, optional*) – hide ClassifierPredictionSummaryComponent or RegressionPredictionSummaryComponent
- **hide_contributiongraph** (*bool, optional*) – hide ShapContributionsGraphComponent

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

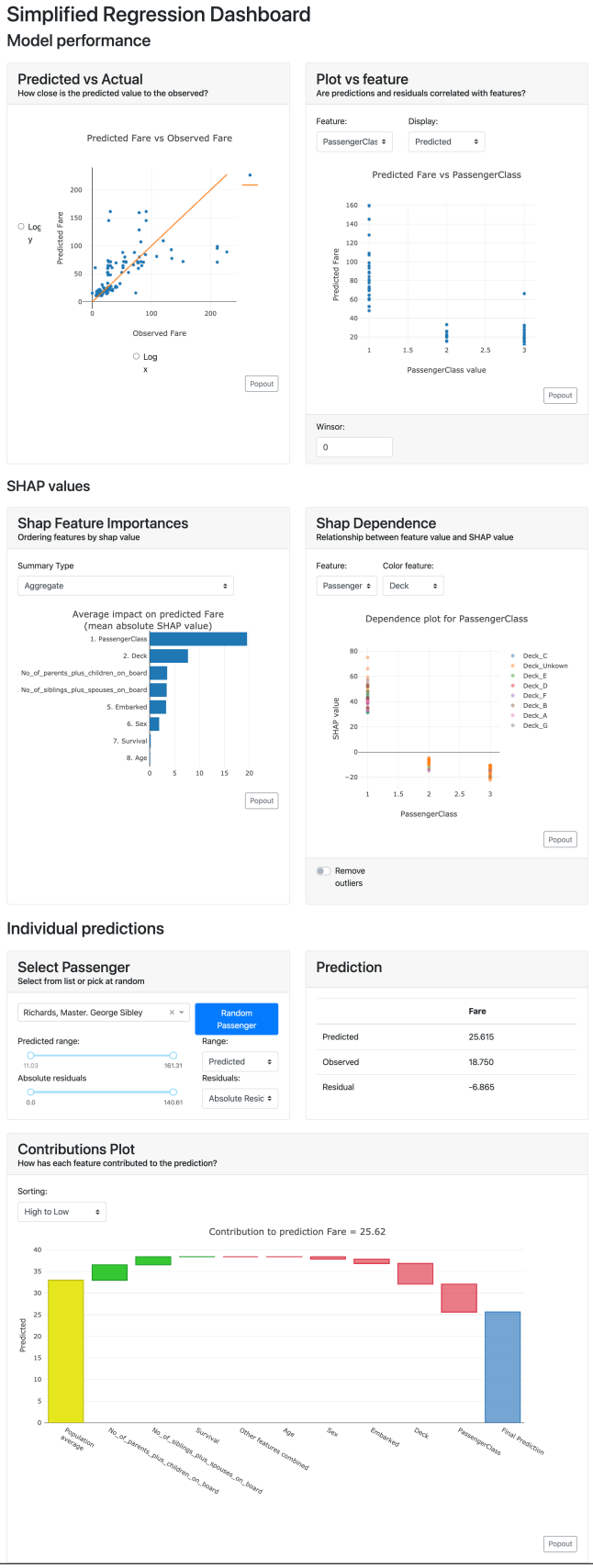
to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

4.6.10 SimplifiedRegressionComposite



You can also load this composite with:

```
explainer = RegressionExplainer(model, X, y)
ExplainerDashboard(explainer, simple=True)
```

class explainerdashboard.dashboard_components.composites.**SimplifiedRegressionComposite**(*explainer*, *title='Simple Regression Explainer'*, *name=None*, *hide_title=False*, *regression_custom_compon*
hide_goodness_of_fit
hide_regression_cust
hide_shapsummary=
hide_shapdependenc
hide_predindexselect
hide_predictionsumm
hide_contributiongra
***kwargs*)

Composite of multiple classifier related components, on a single tab:

- goodness of fit component
- one other model quality indicator: 'metrics', 'residuals' or'residuals_vs_col'
- shap importance
- shap dependence
- index selector
- index prediction summary
- index shap contribution graph

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Simple Classification Stats".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_title** (*bool, optional*) – hide the title. Defaults to False.
- **regression_custom_component** (*str, optional*) – custom classifier quality indicator supported by the ClassifierExplainer object. Valid values are: 'metrics', 'residuals' or'vs_col'
- **hide_goodness_of_fit** (*bool, optional*) – hide goodness of fit component

- **hide_regression_custom_component** (*bool, optional*) – hide the chosen regression_custom_component
- **hide_shapsummary** (*bool, optional*) – hide ShapSummaryComponent
- **hide_shapdependence** (*bool, optional*) – hide ShapDependenceComponent
- **hide_predindexselector** (*bool, optional*) – hide RegressionRandomIndexComponent or RegressionRandomIndexComponent
- **hide_predictionsummary** (*bool, optional*) – hide RegressionPredictionSummaryComponent or RegressionPredictionSummaryComponent
- **hide_contributiongraph** (*bool, optional*) – hide ShapContributionsGraphComponent

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

4.6.11 ExplainerTabsLayout

class explainerdashboard.dashboards.**ExplainerTabsLayout**(*explainer*, *tabs*, *title='Model Explainer'*, *name=None*, *description=None*, *header_hide_title=False*, *header_hide_selector=False*, *header_hide_download=False*, *hide_poweredby=False*, *block_selector_callbacks=False*, *pos_label=None*, *fluid=True*, ***kwargs*)

Generates a multi tab layout from a a list of ExplainerComponents. If the component is a class definition, it gets instantiated first. If the component is not derived from an ExplainerComponent, then attempt with duck typing to nevertheless instantiate a layout.

Parameters

- **explainer** (*[type]*) – explainer
- **tabs** (*list[ExplainerComponent class or instance]*) – list of ExplainerComponent class definitions or instances.
- **title** (*str, optional*) – [description]. Defaults to 'Model Explainer'.
- **description** (*str, optional*) – description tooltip to add to the title.
- **header_hide_title** (*bool, optional*) – Hide the title. Defaults to False.
- **header_hide_selector** (*bool, optional*) – Hide the positive label selector. Defaults to False.
- **header_hide_download** (*bool, optional*) – Hide the download link. Defaults to False.
- **hide_poweredby** (*bool, optional*) – hide the powered by footer

- **block_selector_callbacks** (*bool, optional*) – block the callback of the pos label selector. Useful to avoid clashes when you have your own PosLabelSelector in your layout. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **fluid** (*bool, optional*) – Stretch layout to fill space. Defaults to False.

layout()

returns a multitab layout plus ExplainerHeader

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

register_callbacks(*app*)

Registers callbacks for all tabs

calculate_dependencies()

Calculates dependencies for all tabs

4.6.12 ExplainerPageLayout

class explainerdashboard.dashboards.**ExplainerPageLayout**(*explainer*, *component*, *title='Model Explainer'*, *name=None*, *description=None*, *header_hide_title=False*, *header_hide_selector=False*, *header_hide_download=False*, *hide_poweredby=False*, *block_selector_callbacks=False*, *pos_label=None*, *fluid=False*, ***kwargs*)

Generates a single page layout from a single ExplainerComponent. If the component is a class definition, it gets instantiated.

If the component is not derived from an ExplainerComponent, then tries with duck typing to nevertheless instantiate a layout.

Parameters

- **explainer** (*[type]*) – explainer
- **component** (*ExplainerComponent class or instance*) – ExplainerComponent class definition or instance.
- **title** (*str, optional*) – Defaults to 'Model Explainer'.
- **description** (*str, optional*) – Will be displayed as title tooltip.
- **header_hide_title** (*bool, optional*) – Hide the title. Defaults to False.
- **header_hide_selector** (*bool, optional*) – Hide the positive label selector. Defaults to False.
- **header_hide_download** (*bool, optional*) – Hide the download link. Defaults to False.
- **hide_poweredby** (*bool, optional*) – hide the powered by footer

- **block_selector_callbacks** (*bool, optional*) – block the callback of the pos label selector. Useful to avoid clashes when you have your own PosLabelSelector in your layout. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **fluid** (*bool, optional*) – Stretch layout to fill space. Defaults to False.

layout()

returns single page layout with an ExplainerHeader

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

register_callbacks(*app*)

Register callbacks of page

calculate_dependencies()

Calculate dependencies of page

4.7 ExplainerComponents

The dashboard is constructed out of `ExplainerComponents`: self-contained reusable elements usually consisting of a plot or table and various dropdowns, sliders and toggles to manipulate that plot. Components can be connected with connectors, so that when you select an index in one component that automatically updates the index in another component for example.

When you run `ExplainerDashboard` you get the default dashboard with basically every component listed below with every toggle and slider visible.

The `ExplainerComponents` make it very easy to construct your own dashboard with your own layout, with specific explanations for the workings and results of your model. So you can select which components to use, where to put them in the layout, which toggles and sliders to display, and what the initial values for component should be. This way you can also control which interactive aspects your end users can and cannot control.

You import the components with `from explainerdashboard.custom import *`

A simple example, where you build a page with only a `ShapDependenceComponent`, but with no group cats or highlight toggle, and initial feature set to 'Fare':

```
from explainerdashboard.custom import *

class CustomDashboard(ExplainerComponent):
    def __init__(self, explainer, title="Custom Dashboard", name="None"):
        super().__init__(explainer, title, name=name)
        self.shap_dependence = ShapDependenceComponent(explainer, name=self.name+"dep",
                            hide_title=True, hide_cats=True, hide_highlight=True,
                            cats=True, col='Fare')

    def layout(self):
        return html.Div([
            self.shap_dependence.layout()
        ])
```

(continues on next page)

(continued from previous page)

```
ExplainerDashboard(explainer, CustomDashboard).run()
```

4.7.1 ExplainerComponent

Each component subclasses `ExplainerComponent` which provides the basic functionality of registering subcomponents, dependencies, registering callbacks of subcomponents, calculating dependencies, and providing a list of pos label selectors of all subcomponents.

ExplainerComponent

class explainerdashboard.dashboard_methods.**ExplainerComponent**(*explainer*, *title=None*, *name=None*)

ExplainerComponent is a bundle of a dash layout and callbacks that make use of an Explainer object.

An ExplainerComponent can have ExplainerComponent subcomponents, that you register with register_components(). If the component depends on certain lazily calculated Explainer properties, you can register these with register_dependencies().

ExplainerComponent makes sure that:

1. Callbacks of subcomponents are registered.
2. Lazily calculated dependencies (even of subcomponents) can be calculated.
3. Pos labels selector id's of all subcomponents can be calculated.

Each ExplainerComponent adds a unique uuid name string to all elements, so that there is never a name clash even with multiple ExplanerComponents of the same type in a layout.

Important: define your callbacks in component_callbacks() and ExplainerComponent will register callbacks of subcomponents in addition to component_callbacks() when calling register_callbacks()

---

initialize the ExplainerComponent

Parameters

- **explainer** (*Explainer*) – explainer object constructed with e.g. ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to None.
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.

exclude_callbacks(**components*)

exclude certain subcomponents from the register_components scan

register_components(**components*)

register subcomponents so that their callbacks will be registered and dependencies can be tracked

Parameters

- **scan_self** (*bool, optional*) – scan self.__dict__ and add all
- **True** (*ExplainerComponent attributes to _components. Defaults to*) –

register_dependencies(**dependencies*)

register dependencies: lazily calculated explainer properties that you want to calculate *before* starting the dashboard

property dependencies

returns a list of unique dependencies of the component and all subcomponents

property component_imports

returns a list of ComponentImport namedtuples("component", "module") all components and and subcomponents

get_state_tuples()

returns a list of State (id, property) tuples for the component and all subcomponents

get_state_args(*state_dict=None*)

returns _state_dict with correct self.name attached if state_dict is passed then replace the state_id_prop_tuples with their values from state_dict or else as a property.

property pos_labels

returns a list of unique pos label selector elements of the component and all subcomponents

calculate_dependencies()

calls all properties in self.dependencies so that they get calculated up front. This is useful to do before starting a dashboard, so you don't compute properties multiple times in parallel.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters
state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

save_html(*filename*)

Store output of to_html to a file

Parameters
filename (*str, Path*) – filename to store html

component_callbacks(*app*)

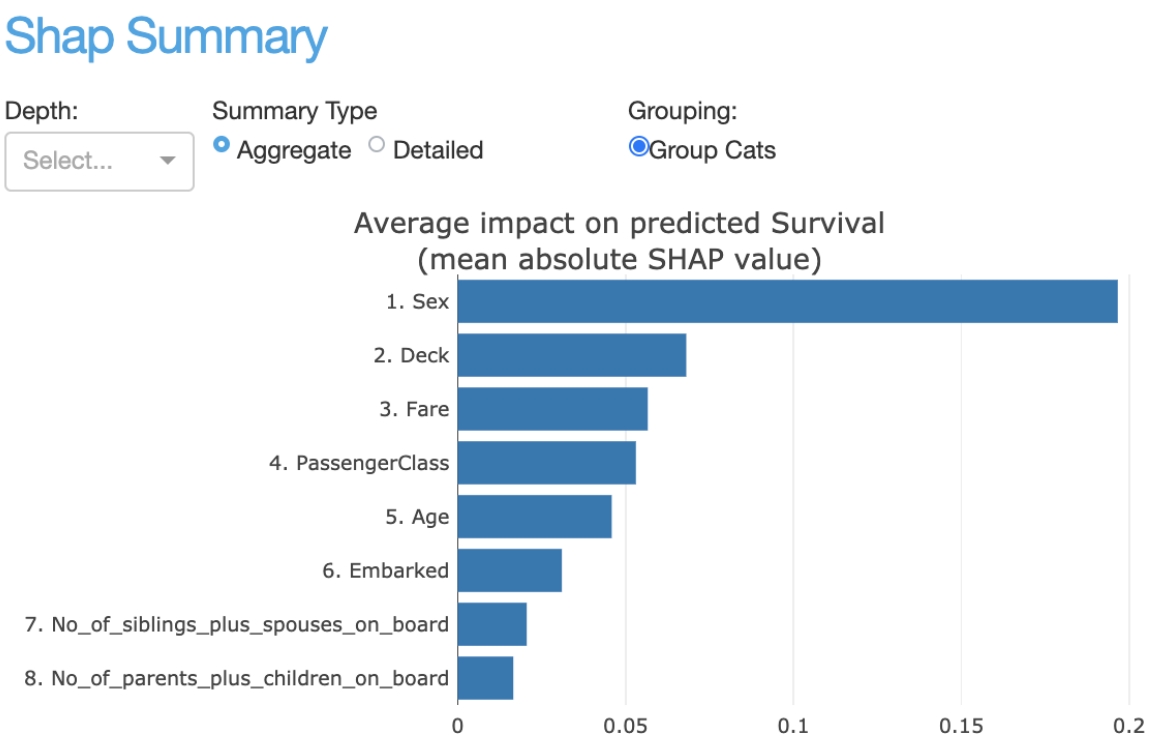
register callbacks specific to this ExplainerComponent.

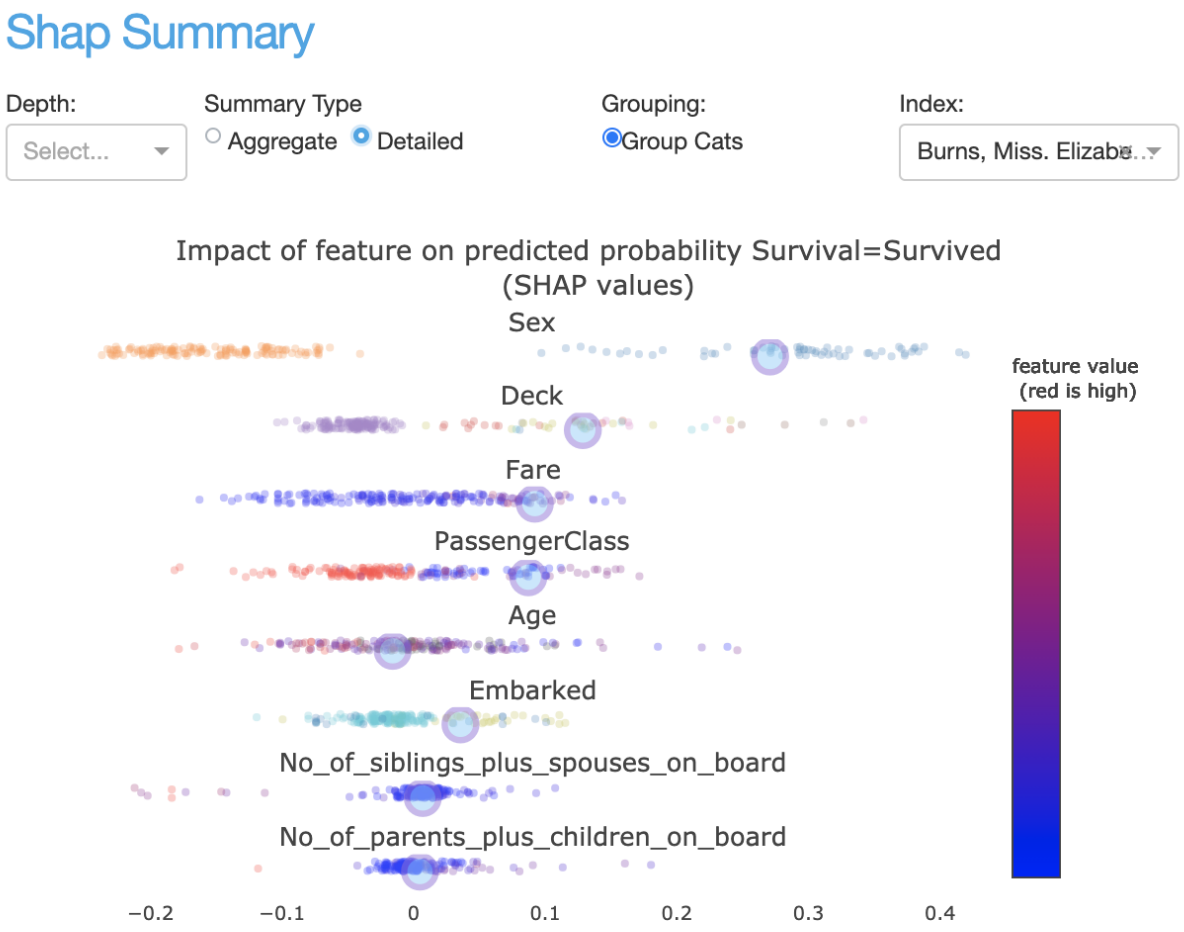
register_callbacks(*app*)

First register callbacks of all subcomponents, then call component_callbacks(app)

4.7.2 shap_components

ShapSummaryComponent




Shap Summary
Depth:
Select...
Summary Type
Aggregate
Detailed
Grouping:
Group Cats
Index:
Burns, Miss. Elizabe...
Impact of feature on predicted probability Survival=Survived
(SHAP values)
Sex
Deck
Fare
PassengerClass
Age
Embarked
No_of_siblings_plus_spouses_on_board
No_of_parents_plus_children_on_board
feature value
(red is high)
−0.2
−0.1
0
0.1
0.2
0.3
0.4

class explainerdashboard.dashboard_components.shap_components.**ShapSummaryComponent**(*explainer*, *title='Shap Summary'*, *name=None*, *subtitle='Ordering features by shap value'*, *hide_title=False*, *hide_subtitle=False*, *hide_depth=False*, *hide_type=False*, *hide_index=False*, *hide_selector=False*, *hide_popout=False*, *pos_label=None*, *depth=None*, *summary_type='aggregate'*, *max_cat_colors=5*, *index=None*, *plot_sample=None*, *description=None*, ***kwargs*)

Shows shap summary component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Shap Dependence Summary".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide the title. Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_depth** (*bool, optional*) – hide the depth toggle. Defaults to False.
- **hide_type** (*bool, optional*) – hide the summary type toggle (aggregated, detailed). Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **depth** (*int, optional*) – initial number of features to show. Defaults to None.

- **summary_type** (*str, {'aggregate', 'detailed'}. optional*) – type of summary graph to show. Defaults to "aggregate".
- **max_cat_colors** (*int, optional*) – for categorical features, maximum number of categories to label with own color. Defaults to 5.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

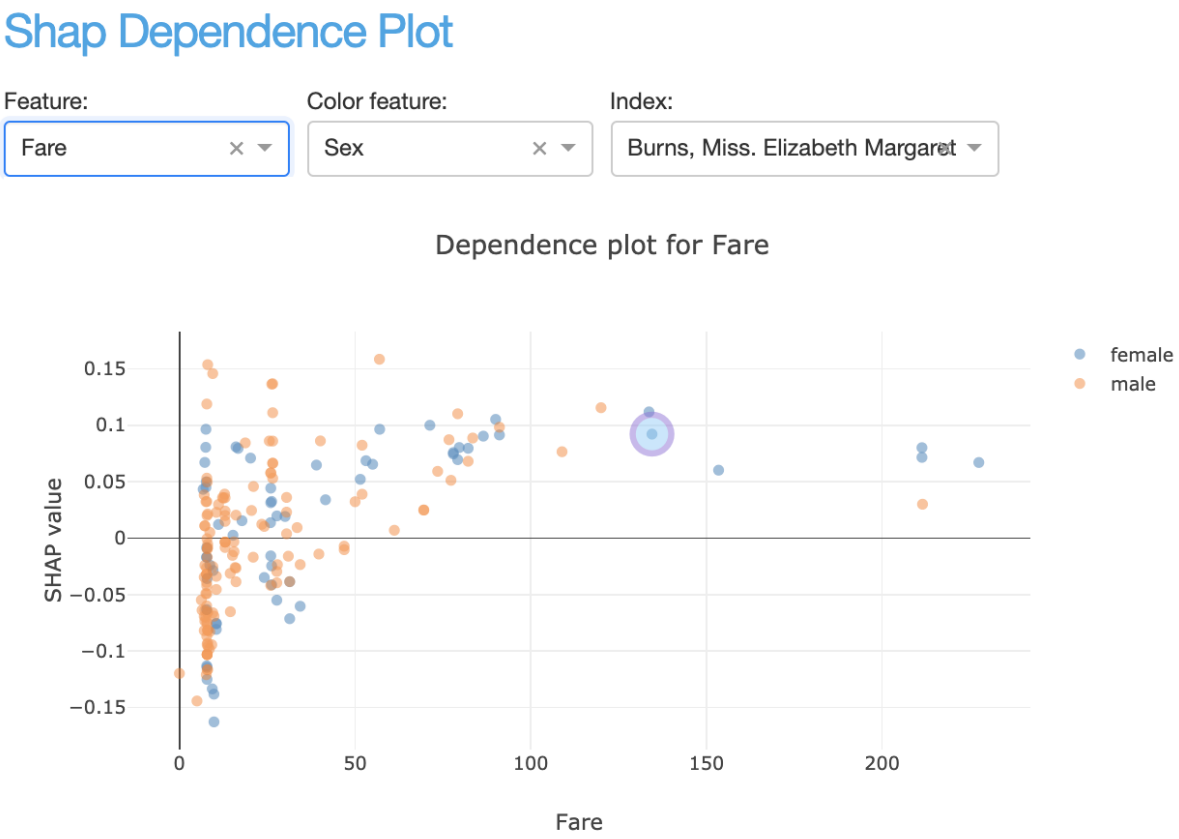
Parameters

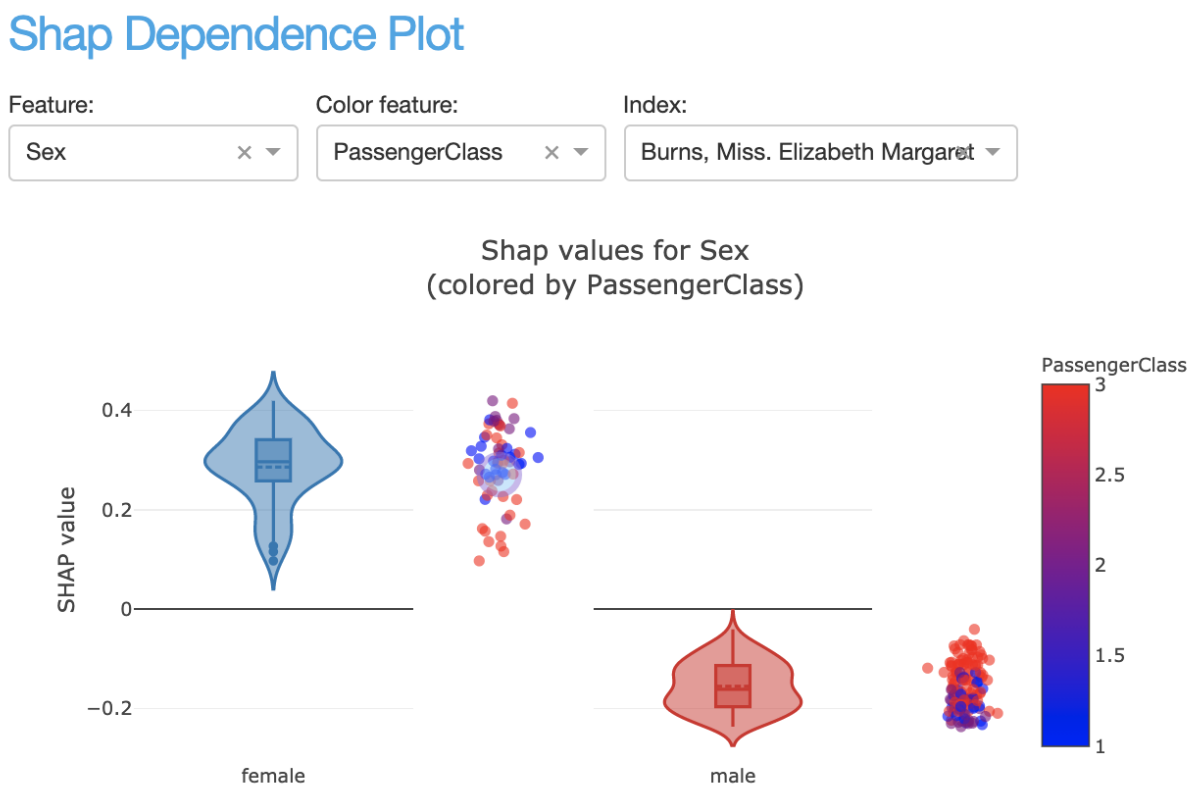
state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

ShapDependenceComponent




Shap Dependence Plot
Feature:
Sex
Color feature:
PassengerClass
Index:
Burns, Miss. Elizabeth Margaret
Shap values for Sex
(colored by PassengerClass)
SHAP value
0.4
0.2
0
−0.2
female
male
PassengerClass
3
2.5
2
1.5
1

class explainerdashboard.dashboard_components.shap_components.**ShapDependenceComponent**(*explainer*, *title='Shap Dependence'*, *name=None*, *subtitle='Relationship between feature value and SHAP value'*, *hide_title=False*, *hide_subtitle=False*, *hide_col=False*, *hide_color_col=False*, *hide_index=False*, *hide_selector=False*, *hide_outliers=False*, *hide_cats_topx=False*, *hide_cats_sort=False*, *hide_popout=False*, *hide_footer=False*, *pos_label=None*, *col=None*, *color_col=None*, *index=None*, *remove_outliers=False*, *cats_topx=10*, *cats_sort='freq'*, *max_cat_colors=5*, *plot_sample=None*, *description=None*, ***kwargs*)

Show shap dependence graph

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Shap Dependence".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.

- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide component title. Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_col** (*bool, optional*) – hide feature selector. Defaults to False.
- **hide_color_col** (*bool, optional*) – hide color feature selector Defaults to False.
- **hide_index** (*bool, optional*) – hide index selector Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **hide_cats_topx** (*bool, optional*) – hide the categories topx input. Defaults to False.
- **hide_cats_sort** (*bool, optional*) – hide the categories sort selector.Defaults to False.
- **hide_outliers** (*bool, optional*) – Hide remove outliers toggle input. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **col** (*str, optional*) – Feature to display. Defaults to None.
- **color_col** (*str, optional*) – Color plot by values of this Feature. Defaults to None.
- **index** (*int, optional*) – Highlight a particular index. Defaults to None.
- **remove_outliers** (*bool, optional*) – remove outliers in feature and color feature from the plot.
- **cats_topx** (*int, optional*) – maximum number of categories to display for categorical features. Defaults to 10.
- **cats_sort** (*str, optional*) – how to sort categories: 'alphabet', 'freq' or 'shap'. Defaults to 'freq'.
- **max_cat_colors** (*int, optional*) – for categorical features, maximum number of categories to label with own color. Defaults to 5.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

ShapSummaryDependenceConnector

class explainerdashboard.dashboard_components.shap_components.**ShapSummaryDependenceConnector**(*shap_summa*
shap_depend

Connects a ShapSummaryComponent with a ShapDependence Component:

- When clicking on feature in ShapSummary, then select that feature in ShapDependence

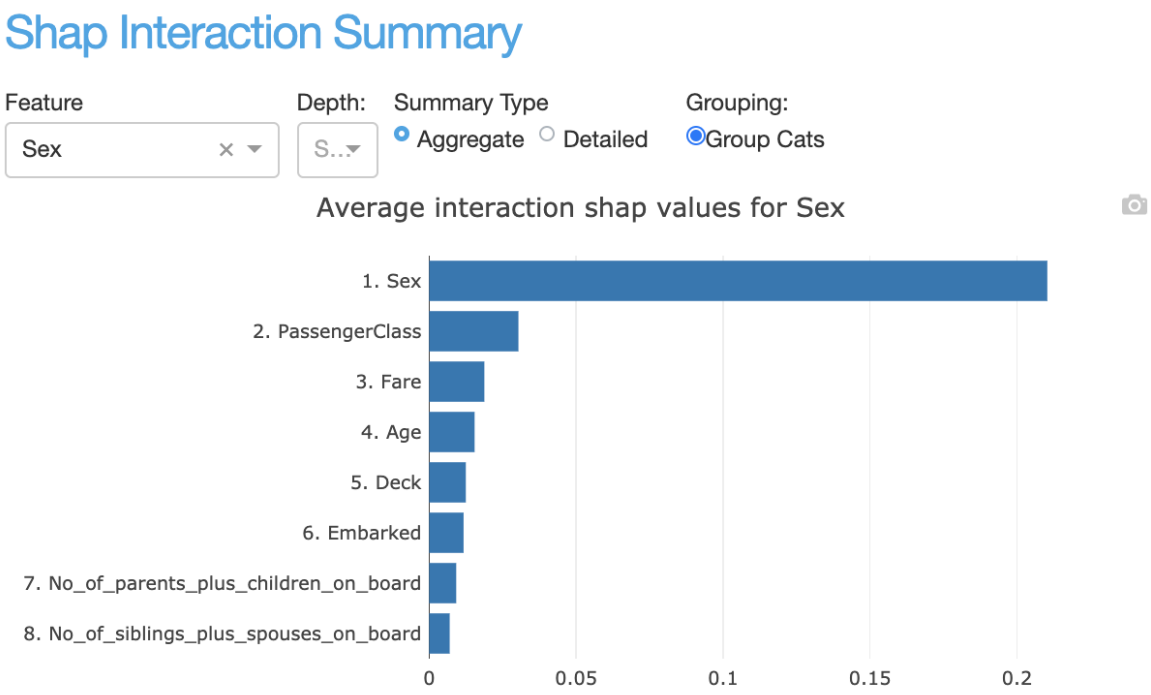
Parameters

- **shap_summary_component** (ShapSummaryComponent) – ShapSummaryComponent
- **shap_dependence_component** (ShapDependenceComponent) – ShapDependenceComponent

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

InteractionSummaryComponent



class explainerdashboard.dashboard_components.shap_components.**InteractionSummaryComponent**(*explainer*, *title='Interactions Summary'*, *name=None*, *subtitle='Ordering features by shap interaction value'*, *hide_title=False*, *hide_subtitle=False*, *hide_col=False*, *hide_depth=False*, *hide_type=False*, *hide_index=False*, *hide_popout=False*, *hide_selector=False*, *pos_label=None*, *col=None*, *depth=None*, *summary_type='aggr'*, *max_cat_colors=5*, *index=None*, *plot_sample=None*, *description=None*, ***kwargs*)

Show SHAP Interaciton values summary component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Interactions Summary".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide the component title. Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_col** (*bool, optional*) – Hide the feature selector. Defaults to False.

- **hide_depth** (*bool, optional*) – Hide depth toggle. Defaults to False.
- **hide_type** (*bool, optional*) – Hide summary type toggle. Defaults to False.
- **hide_index** (*bool, optional*) – Hide the index selector. Defaults to False
- **hide_popout** (*bool, optional*) – hide popout button
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **col** (*str, optional*) – Feature to show interaction summary for. Defaults to None.
- **depth** (*int, optional*) – Number of interaction features to display. Defaults to None.
- **summary_type** (*str, {'aggregate', 'detailed'}, optional*) – type of summary graph to display. Defaults to "aggregate".
- **max_cat_colors** (*int, optional*) – for categorical features, maximum number of categories to label with own color. Defaults to 5.
- **index** (*str*) – Default index. Defaults to None.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

InteractionDependenceComponent

class explainerdashboard.dashboard_components.shap_components.**InteractionDependenceComponent**(*explainer*, *title='Interacti De- pen- dence'*, *name=None*, *sub- ti- tle='Relation be- tween fea- ture value and shap in- ter- ac- tion value'*, *hide_title=Fa hide_subtitle= hide_col=Fal hide_interact hide_index=F hide_popout= hide_selector hide_outliers= hide_cats_top hide_cats_so hide_top=Fal hide_bottom= pos_label=No col=None*, *in- ter- act_col=Non re- move_outliers cats_topx=10 cats_sort='fr max_cat_colo plot_sample= de- scrip- tion=None*, *in- dex=None*, ***kwargs*)

Interaction Dependence Component.

Shows two graphs:

top graph: col vs interact_col bottom graph: interact_col vs col

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Interactions Dependence".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – Hide component title. Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_col** (*bool, optional*) – Hide feature selector. Defaults to False.
- **hide_interact_col** (*bool, optional*) – Hide interaction feature selector. Defaults to False.
- **hide_highlight** (*bool, optional*) – Hide highlight index selector. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **hide_outliers** (*bool, optional*) – Hide remove outliers toggle input. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button
- **hide_cats_topx** (*bool, optional*) – hide the categories topx input. Defaults to False.
- **hide_cats_sort** (*bool, optional*) – hide the categories sort selector. Defaults to False.
- **hide_top** (*bool, optional*) – Hide the top interaction graph (col vs interact_col). Defaults to False.
- **hide_bottom** (*bool, optional*) – hide the bottom interaction graph (interact_col vs col). Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **col** (*str, optional*) – Feature to find interactions for. Defaults to None.
- **interact_col** (*str, optional*) – Feature to interact with. Defaults to None.
- **highlight** (*int, optional*) – Index row to highlight Defaults to None.
- **remove_outliers** (*bool, optional*) – remove outliers in feature and color feature from the plot.
- **cats_topx** (*int, optional*) – number of categories to display for categorical features.
- **cats_sort** (*str, optional*) – how to sort categories: 'alphabet', 'freq' or 'shap'. Defaults to 'freq'.
- **max_cat_colors** (*int, optional*) – for categorical features, maximum number of categories to label with own color. Defaults to 5.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

InteractionSummaryDependenceConnector

class explainerdashboard.dashboard_components.shap_components.**InteractionSummaryDependenceConnector**(*inter*
in-
ter-
ac-
*tion_*

Connects a InteractionSummaryComponent with an InteractionDependenceComponent:

- When select feature in summary, then select col in Dependence
- **When clicking on interaction feature in Summary, then select that interaction** feature in Dependence.

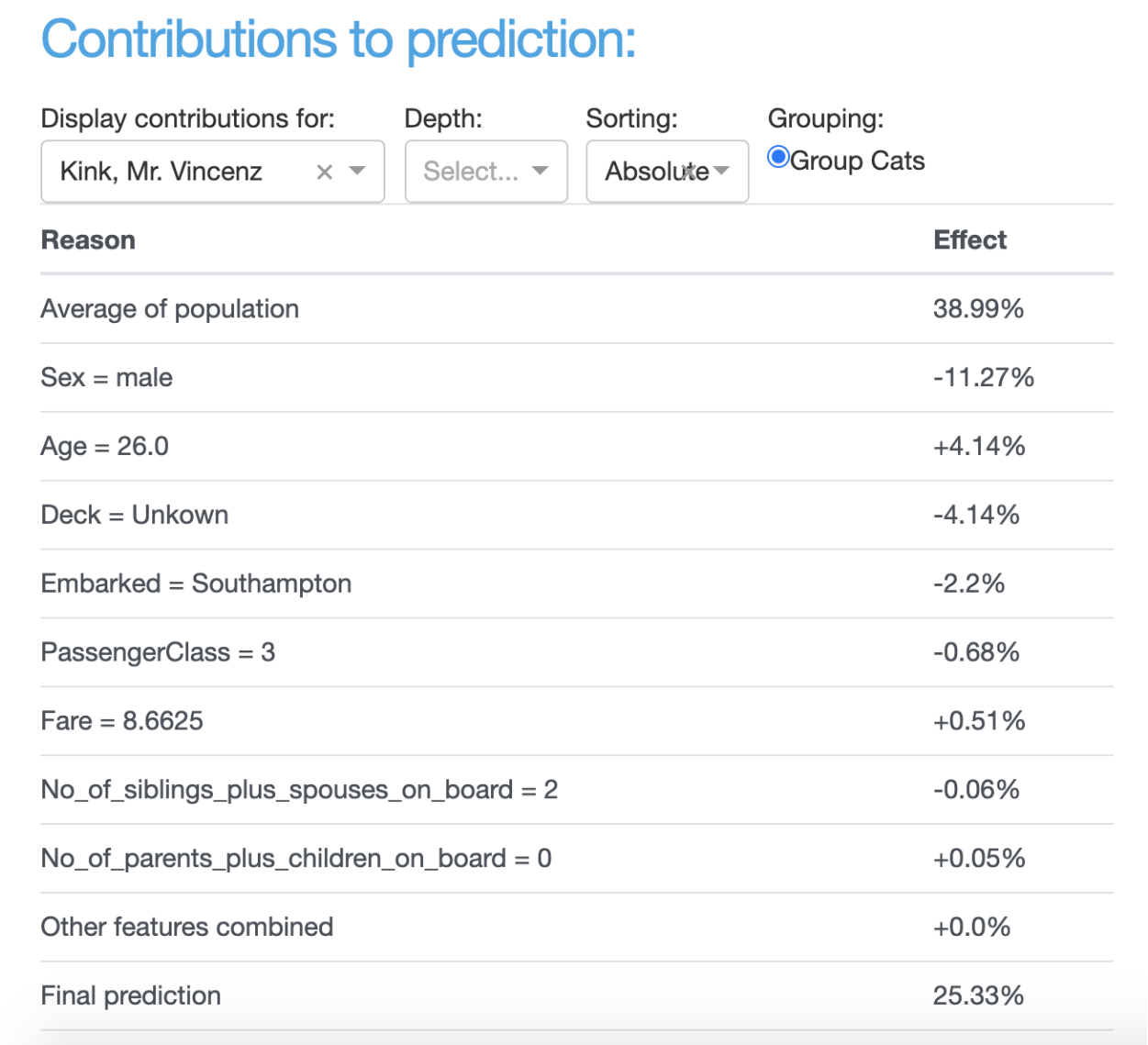
Parameters

- **shap_summary_component** (ShapSummaryComponent) – ShapSummaryComponent
- **shap_dependence_component** (ShapDependenceComponent) – ShapDependenceComponent

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

ShapContributionsTableComponent



class explainerdashboard.dashboard_components.shap_components.**ShapContributionsTableComponent**(*explainer*, *title='Contrib Table'*, *name=None*, *subtitle='How has each feature contributed to the prediction?'*, *hide_title=F hide_subtitl hide_index= hide_depth= hide_sort=F hide_selecto feature_input_c index_dropdo pos_label=N index=None*, *depth=None*, *sort='abs'*, *description=None*, ***kwargs*)

Show SHAP values contributions to prediction in a table component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Contributions Table".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – Hide component title. Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_index** (*bool, optional*) – Hide index selector. Defaults to False.

- **hide_depth** (*bool, optional*) – Hide depth selector. Defaults to False.
- **hide_sort** (*bool, optional*) – Hide sorting dropdown. Default to False.
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **feature_input_component** (FeatureInputComponent) – A FeatureInputComponent that will give the input to the graph instead of the index selector. If not None, hide_index=True. Defaults to None.
- **index_dropdown** (*bool, optional*) – Use dropdown for index input instead of free text input. Defaults to True.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **index** (*[type], optional*) – Initial index to display. Defaults to None.
- **depth** (*[type], optional*) – Initial number of features to display. Defaults to None.
- **sort** (*{'abs', 'high-to-low', 'low-to-high', 'importance'}, optional*) – sorting of shap values. Defaults to 'high-to-low'.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

get_state_tuples()

returns a list of State (id, property) tuples for the component and all subcomponents

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

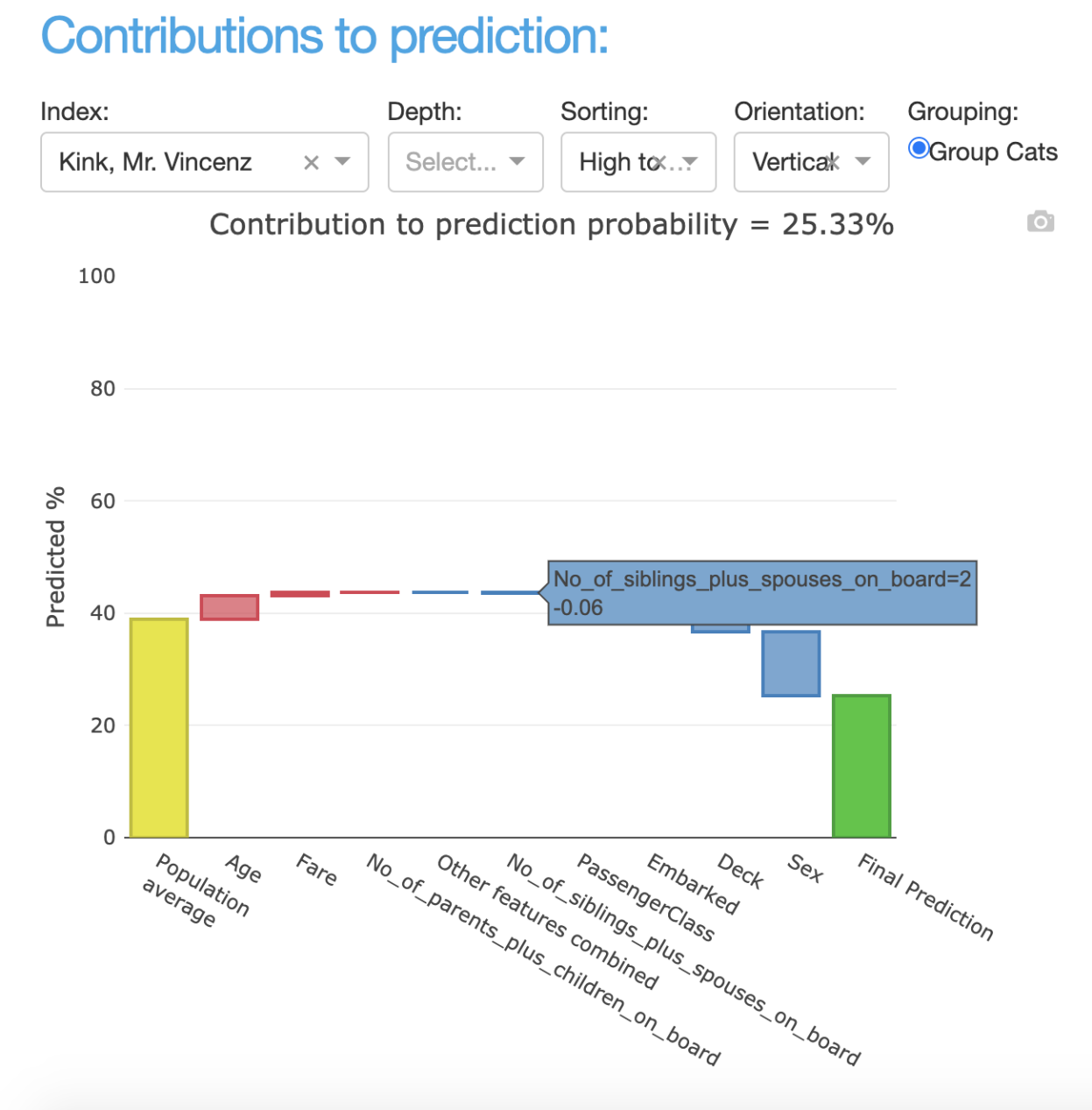
Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

ShapContributionsGraphComponent



class explainerdashboard.dashboard_components.shap_components.**ShapContributionsGraphComponent**(*explainer*, *title='Contrib Plot'*, *name=None*, *subtitle='How has each feature contributed to the prediction?'*, *hide_title=F hide_subtitl hide_index= hide_depth= hide_sort=F hide_orienta hide_selecto hide_popou feature_input_c index_dropdo pos_label=N index=None*, *depth=None*, *sort='high-to-low'*, *orientation='vertica higher_is_b description=None*, ***kwargs*)

Display Shap contributions to prediction graph component

Parameters

- **explainer** (*Explainer*) – explainer object constructed , with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Contributions Plot".

- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – Hide component title. Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_index** (*bool, optional*) – Hide index selector. Defaults to False.
- **hide_depth** (*bool, optional*) – Hide depth toggle. Defaults to False.
- **hide_sort** (*bool, optional*) – Hide the sorting dropdown. Defaults to False.
- **hide_orientation** (*bool, optional*) – Hide the orientation dropdown. Defaults to True.
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button
- **feature_input_component** (FeatureInputComponent) – A FeatureInputComponent that will give the input to the graph instead of the index selector. If not None, hide_index=True. Defaults to None.
- **index_dropdown** (*bool, optional*) – Use dropdown for index input instead of free text input. Defaults to True.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **index** (*{int, bool}, optional*) – Initial index to display. Defaults to None.
- **depth** (*int, optional*) – Initial number of features to display. Defaults to None.
- **sort** (*{'abs', 'high-to-low', 'low-to-high', 'importance'}, optional*) – sorting of shap values. Defaults to 'high-to-low'.
- **orientation** (*{'vertical', 'horizontal'}, optional*) – orientation of bar chart. Defaults to 'vertical'.
- **higher_is_better** (*bool, optional*) – Color positive shap values green and negative shap values red, or the reverse.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

get_state_tuples()

returns a list of State (id, property) tuples for the component and all subcomponents

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

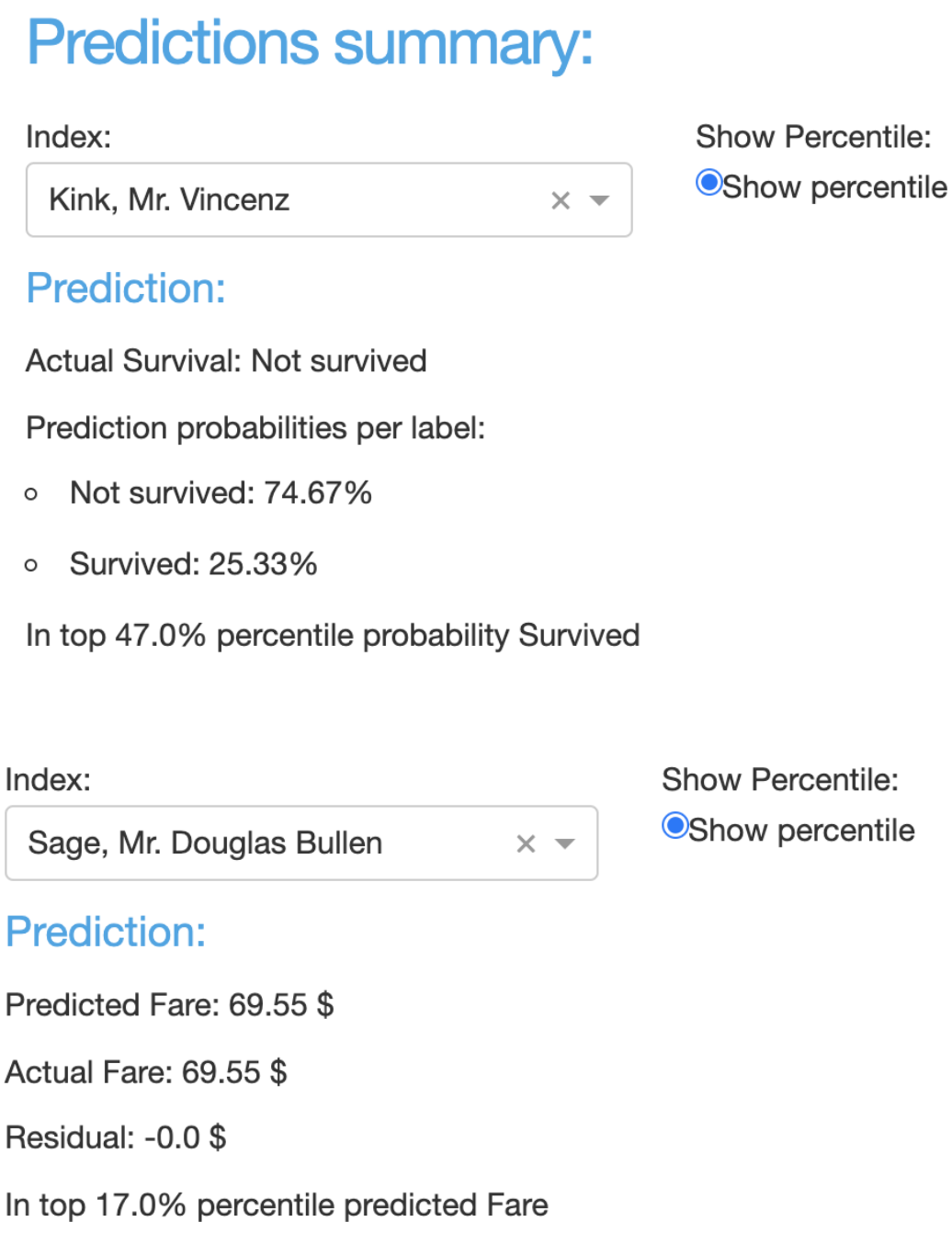
state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

4.7.3 overview_components

PredictionSummaryComponent



class explainerdashboard.dashboard_components.overview_components.**PredictionSummaryComponent**(*explainer*, *title='Prediction Summary'*, *name=None*, *hide_index=False*, *hide_percentile=False*, *hide_title=False*, *hide_subtitle=False*, *hide_selector=False*, *pos_label=None*, *index=None*, *percentile=True*, *description=None*, ***kwargs*)

Shows a summary for a particular prediction

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Prediction Summary".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_index** (*bool, optional*) – hide index selector. Defaults to False.
- **hide_percentile** (*bool, optional*) – hide percentile toggle. Defaults to False.
- **hide_title** (*bool, optional*) – hide title. Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selectors. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **index** (*{int, str}, optional*) – Index to display prediction summary for. Defaults to None.
- **percentile** (*bool, optional*) – Whether to add the prediction percentile. Defaults to True.

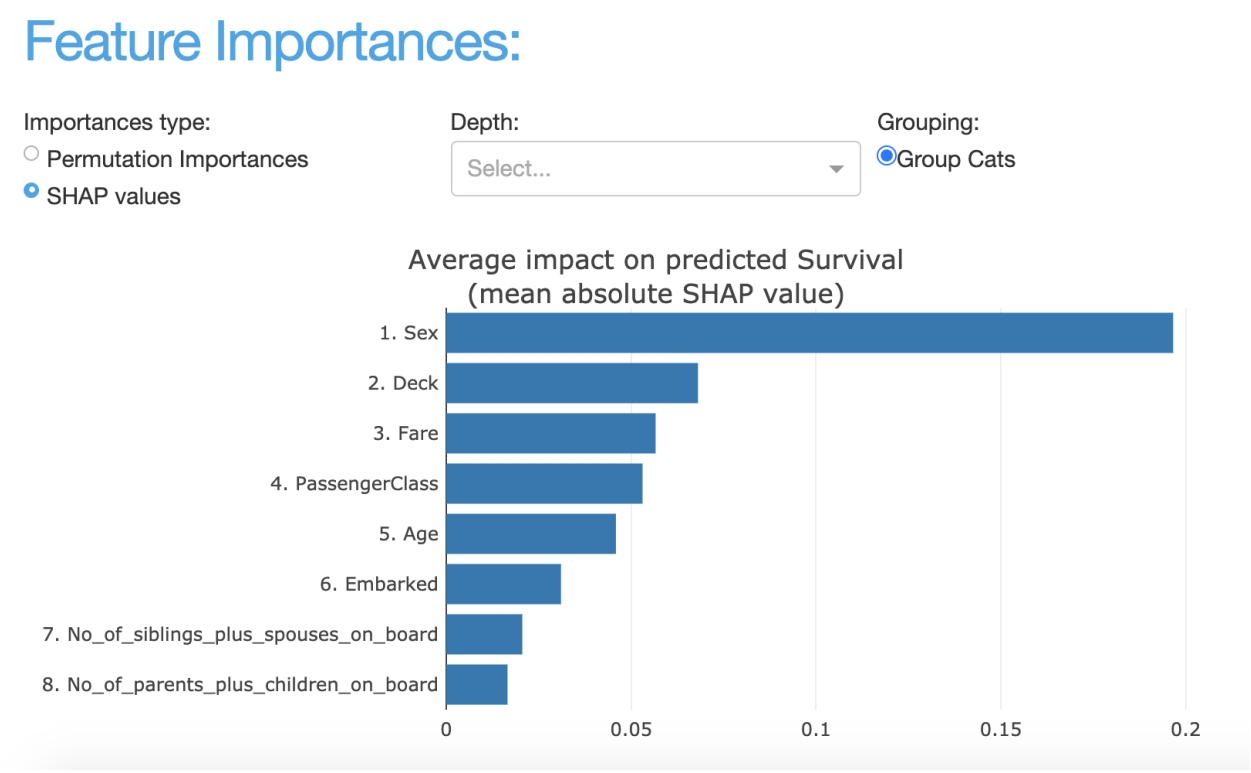
layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

ImportancesComponent



class explainerdashboard.dashboard_components.overview_components.**ImportancesComponent**(*explainer*, *title='Feature Importances'*, *name=None*, *subtitle='Which features had the biggest impact?'*, *hide_type=False*, *hide_depth=False*, *hide_popout=False*, *hide_title=False*, *hide_subtitle=False*, *hide_selector=False*, *pos_label=None*, *importance_type='shap'*, *depth=None*, *no_permutations=Fa*, *description=None*, ***kwargs*)

Display features importances component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Feature Importances".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str, optional*) – Subtitle.
- **hide_type** (*bool, optional*) – Hide permutation/shap selector toggle. Defaults to False.
- **hide_depth** (*bool, optional*) – Hide number of features toggle. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button
- **hide_title** (*bool, optional*) – hide title. Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selectors. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label

- **importance_type** (*str, {'permutation', 'shap'} optional*) – initial importance type to display. Defaults to "shap".
- **depth** (*int, optional*) – Initial number of top features to display. Defaults to None (=show all).
- **no_permutations** (*bool, optional*) – Do not use the permutation importances for this component. Defaults to False.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

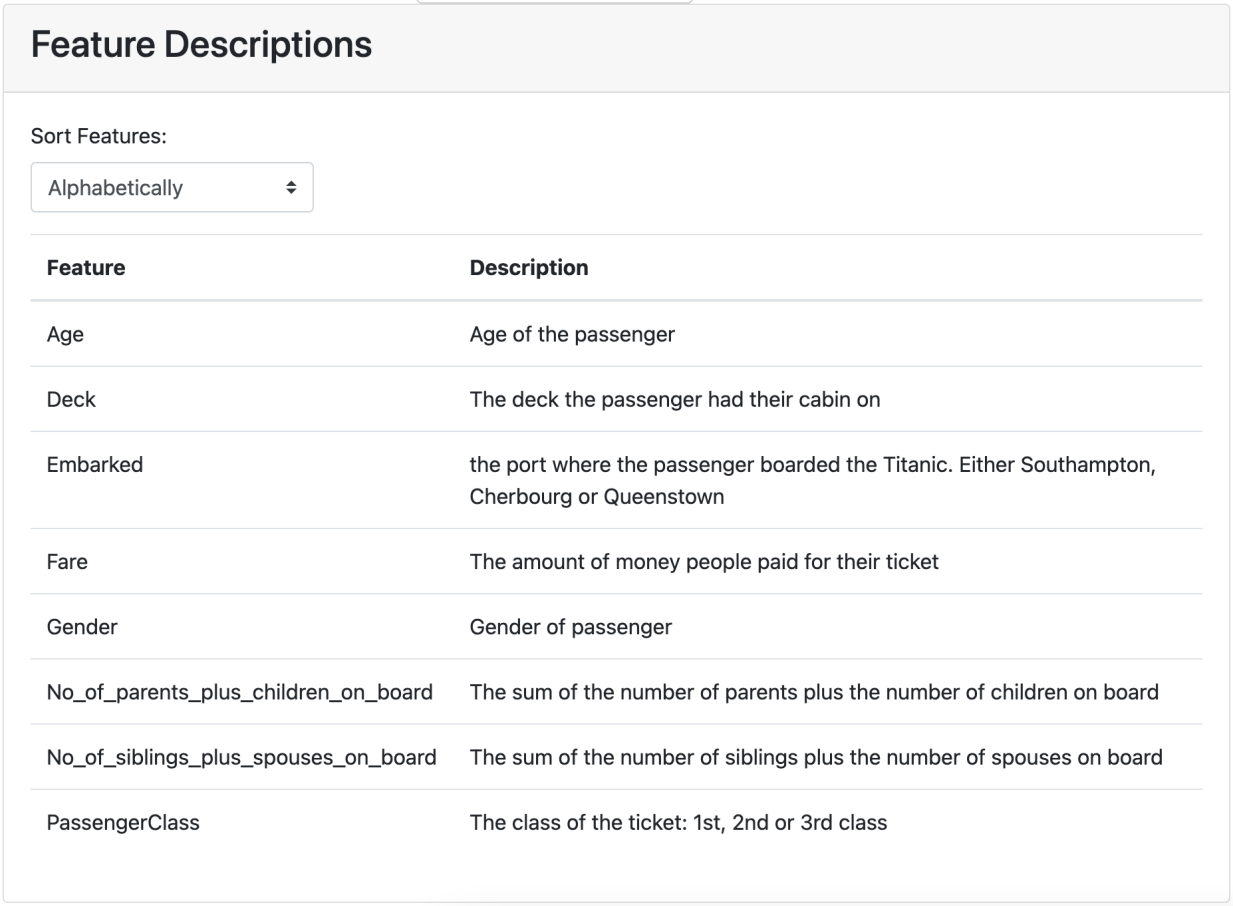
Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*, ***kwargs*)

register callbacks specific to this ExplainerComponent.

FeatureDescriptionsComponent



class explainerdashboard.dashboard_components.overview_components.**FeatureDescriptionsComponent**(*explainer*, *title='Feature Descriptions'*, *name=None*, *hide_title=False*, *hide_sort=False*, *sort='alphabet'*, ***kwargs*)

Display Feature Descriptions table.

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Feature Importances".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_title** (*bool, optional*) – hide the title

- **hide_sort** (*bool, optional*) – hide the sort
- **sort** (*str, optional*) – how to sort the features, either 'alphabet' or by mean abs shap ('shap')

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

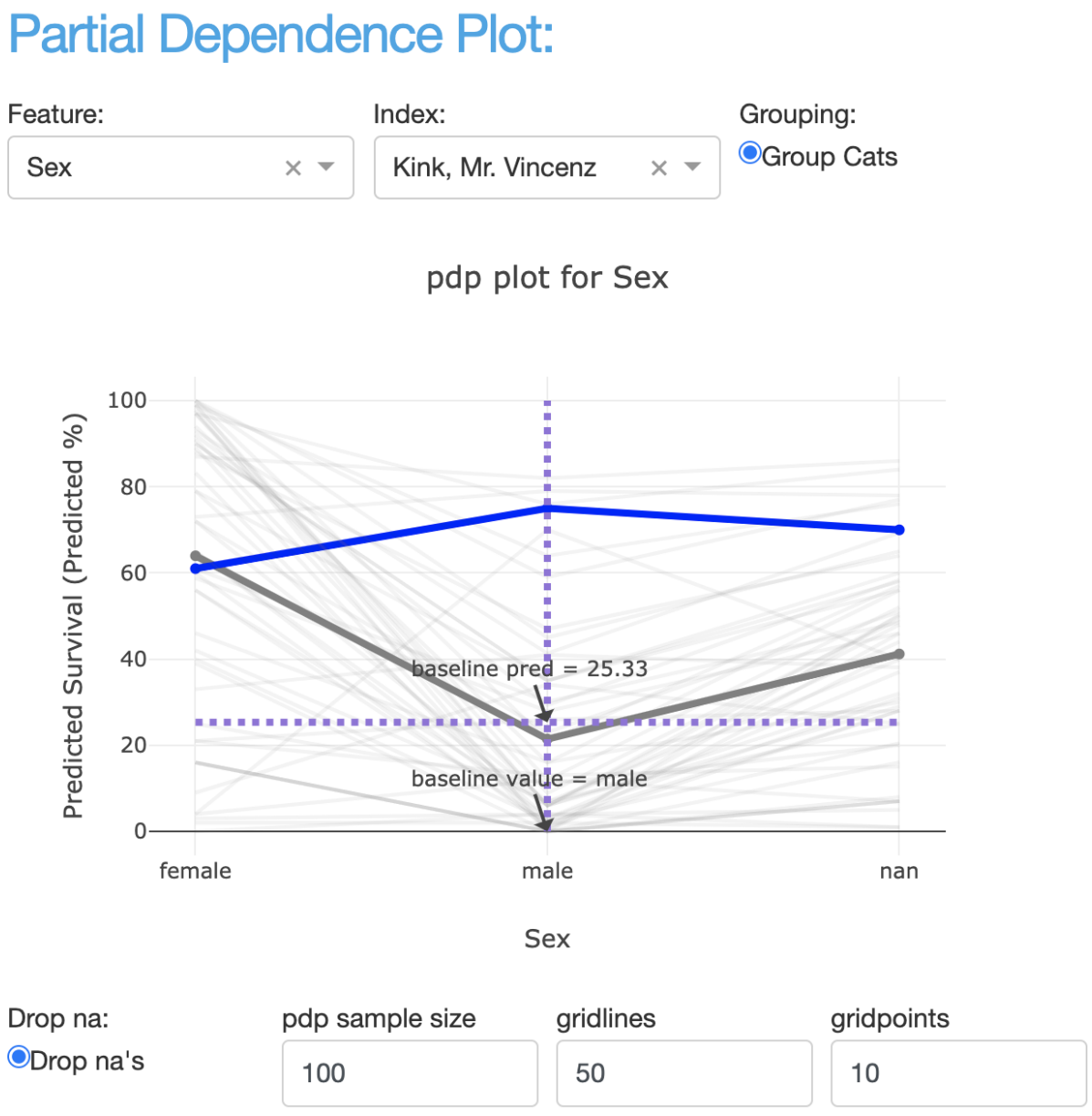
return static html for this component and all subcomponents.

Parameters
state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

PdpComponent



class explainerdashboard.dashboard_components.overview_components.**PdpComponent**(*explainer*, *title='Partial Dependence Plot'*, *name=None*, *subtitle='How does the prediction change if you change one feature?'*, *hide_col=False*, *hide_index=False*, *hide_title=False*, *hide_subtitle=False*, *hide_footer=False*, *hide_selector=False*, *hide_popout=False*, *hide_dropna=False*, *hide_sample=False*, *hide_gridlines=False*, *hide_gridpoints=False*, *hide_cats_sort=False*, *index_dropdown=True*, *feature_input_component=None*, *pos_label=None*, *col=None*, *index=None*, *dropna=True*, *sample=100*, *gridlines=50*, *gridpoints=10*, *cats_sort='freq'*, *description=None*, ***kwargs*)

Show Partial Dependence Plot component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Partial Dependence Plot".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_col** (*bool, optional*) – Hide feature selector. Defaults to False.
- **hide_index** (*bool, optional*) – Hide index selector. Defaults to False.

- **hide_title** (*bool, optional*) – Hide title, Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_selector** (*bool, optional*) – hide pos label selectors. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button
- **hide_dropna** (*bool, optional*) – Hide drop na's toggle Defaults to False.
- **hide_sample** (*bool, optional*) – Hide sample size input. Defaults to False.
- **hide_gridlines** (*bool, optional*) – Hide gridlines input. Defaults to False.
- **hide_gridpoints** (*bool, optional*) – Hide gridpounts input. Defaults to False.
- **hide_cats_sort** (*bool, optional*) – Hide the categorical sorting dropdown. Defaults to False.
- **index_dropdown** (*bool, optional*) – Use dropdown for index input instead of free text input. Defaults to True.
- **feature_input_component** (FeatureInputComponent) – A FeatureInputComponent that will give the input to the graph instead of the index selector. If not None, hide_index=True. Defaults to None.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **col** (*str, optional*) – Feature to display PDP for. Defaults to None.
- **index** (*{int, str}, optional*) – Index to add ice line to plot. Defaults to None.
- **dropna** (*bool, optional*) – Drop rows where values equal explainer.na_fill (usually -999). Defaults to True.
- **sample** (*int, optional*) – Sample size to calculate average partial dependence. Defaults to 100.
- **gridlines** (*int, optional*) – Number of ice lines to display in plot. Defaults to 50.
- **gridpoints** (*int, optional*) – Number of breakpoints on horizontal axis Defaults to 10.
- **cats_sort** (*str, optional*) – how to sort categories: 'alphabet', 'freq' or 'shap'. Defaults to 'freq'.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

get_state_tuples()

returns a list of State (id, property) tuples for the component and all subcomponents

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

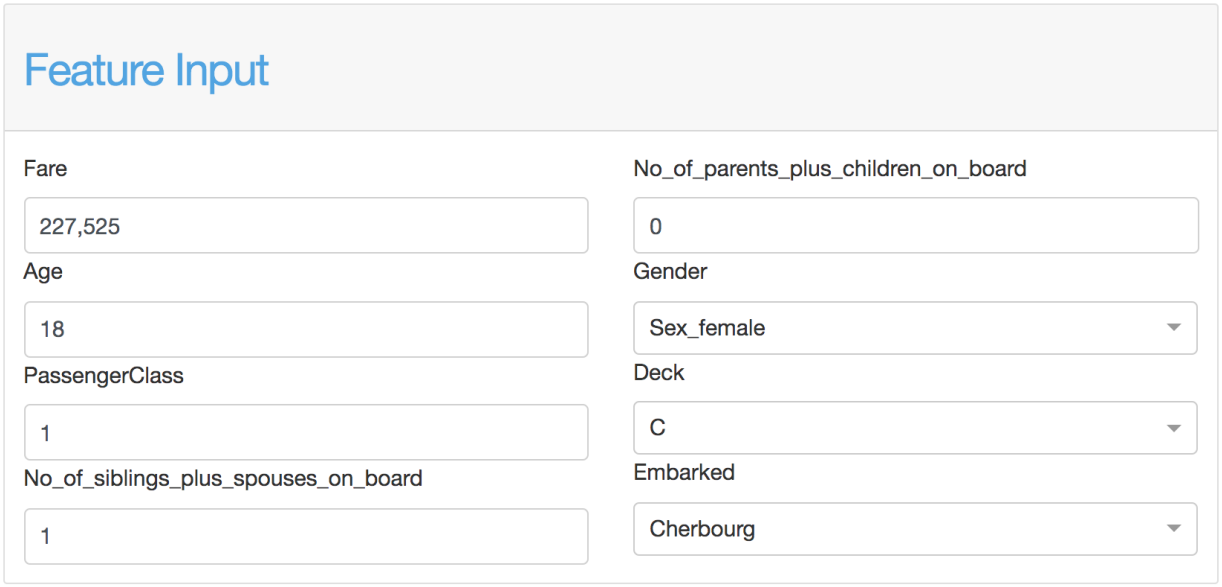
Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

FeatureInputComponent



Using the feature input component you can edit the features for a particular observation in order to check what would be the change in prediction if you change one or more features.

You can connect the `FeatureInputComponent` to the `ShapContributionsGraphComponent` and the `PdpComponent` using the``feature_input_component`` parameter:

```
class WhatIfComposite(ExplainerComponent):
   def __init__(self, explainer, title="What if..."):
      super().__init__(explainer, title, name)

      self.input = FeatureInputComponent(explainer)
      self.contrib = ShapContributionsGraphComponent(explainer, feature_input_
↪component=self.input)
      self.pdp = PdpComponent(explainer, feature_input_component=self.input)
```

class explainerdashboard.dashboard_components.overview_components.**FeatureInputComponent**(*explainer*, *title='Feature Input'*, *name=None*, *subtitle='Adjust the feature values to change the prediction'*, *hide_title=False*, *hide_subtitle=False*, *hide_index=False*, *hide_range=False*, *index=None*, *n_input_cols=4*, *sort_features='shap'*, *fill_row_first=True*, *description=None*, ***kwargs*)

Interaction Dependence Component.

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "What if…".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide the title
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_index** (*bool, optional*) – hide the index selector
- **hide_range** (*bool, optional*) – hide the range label under the inputs
- **index** (*str, int, optional*) – default index
- **n_input_cols** (*int, optional*) – number of columns to split features inputs in. Defaults to 4.

- **sort_features** (*str, optional*) – how to sort the features. For now only options is 'shap' to sort by mean absolute shap value.
- **fill_row_first** (*bool, optional*) – if True most important features will be on top row, if False they will be in most left column.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

get_slices_cols_first(*n_inputs*, *n_cols=2*)

returns a list of slices to divide n inputs into n_cols columns, filling columns first

get_slices_rows_first(*n_inputs*, *n_cols=3*)

returns a list of slices to divide n inputs into n_cols columns, filling columns first

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

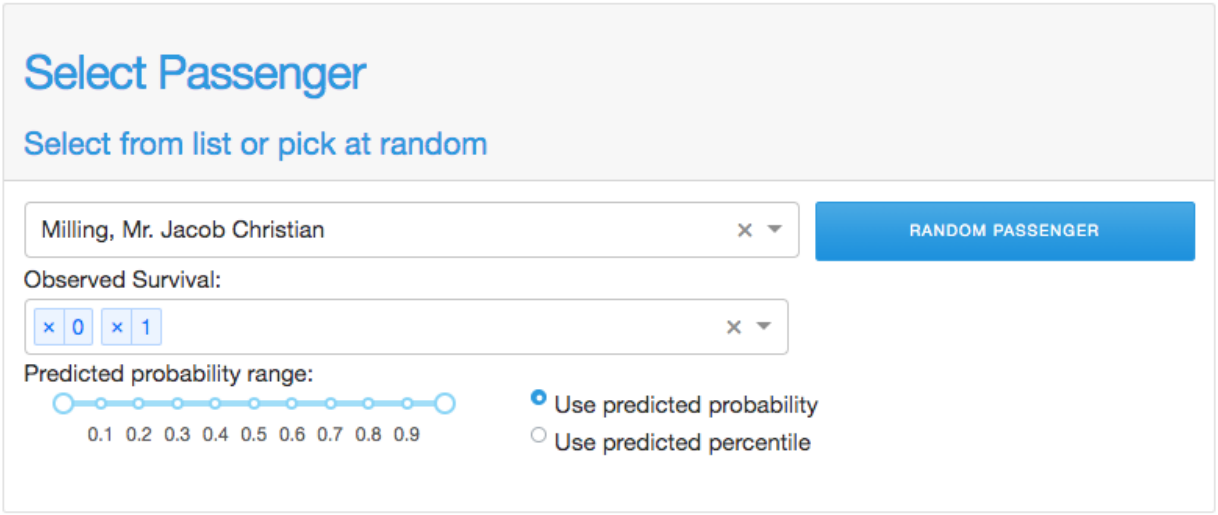
state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

4.7.4 classifier_components

ClassifierRandomIndexComponent



class explainerdashboard.dashboard_components.classifier_components.**ClassifierRandomIndexComponent**(*explainer*, *title='Select Random Index'*, *name=None*, *subtitle='Select from list or pick at random'*, *hide_title=False*, *hide_subtitle=False*, *hide_index=False*, *hide_predindex=False*, *hide_predslider=False*, *hide_target=False*, *hide_selector=False*, *index_dropdown=True*, *pos_label=None*, *index=None*, *slider=None*, *labels=None*, *pred_only=False*, *description=None*, ***kwargs*)

Select a random index subject to constraints component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Select Random Index".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – Hide title. Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_index** (*bool, optional*) – Hide index selector. Defaults to False.

- **hide_slider** (*bool, optional*) – Hide prediction/percentile slider. Defaults to False.
- **hide_labels** (*bool, optional*) – Hide label selector Defaults to False.
- **hide_pred_or_perc** (*bool, optional*) – Hide prediction/percentiles toggle. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selectors. Defaults to False.
- **hide_button** (*bool, optional*) – Hide button. Defaults to False.
- **index_dropdown** (*bool, optional*) – Use dropdown for index input instead of free text input. Defaults to True.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **index** (*{str, int}, optional*) – Initial index to display. Defaults to None.
- **slider** (*[float,float], optional*) – initial slider position [lower bound, upper bound]. Defaults to None.
- **labels** (*[str], optional*) – list of initial labels(str) to include. Defaults to None.
- **pred_or_perc** (*str, optional*) – Whether to use prediction or percentiles slider. Defaults to 'predictions'.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

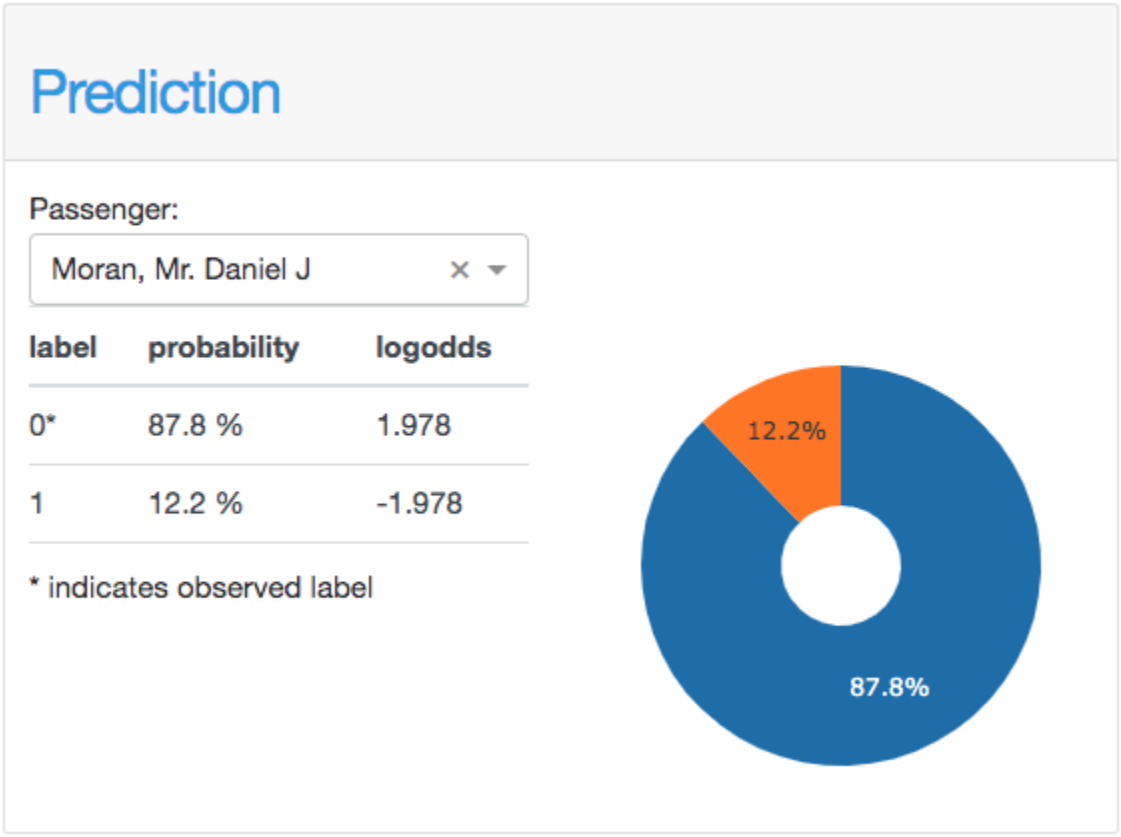
Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

ClassifierPredictionSummaryComponent



class explainerdashboard.dashboard_components.classifier_components.**ClassifierPredictionSummaryComponen**

Shows a summary for a particular prediction

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Prediction Summary".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_index** (*bool, optional*) – hide index selector. Defaults to False.
- **hide_title** (*bool, optional*) – hide title. Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_table** (*bool, optional*) – hide the results table
- **hide_piechart** (*bool, optional*) – hide the results piechart
- **hide_star_explanation** (*bool, optional*) – hide the * *indicates.*. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selectors. Defaults to False.
- **index_dropdown** (*bool, optional*) – Use dropdown for index input instead of free text input. Defaults to True.
- **feature_input_component** (FeatureInputComponent) – A FeatureInputComponent that will give the input to the graph instead of the index selector. If not None, hide_index=True. Defaults to None.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **index** (*{int, str}, optional*) – Index to display prediction summary for. Defaults to None.
- **round** (*int, optional*) – rounding to apply to pred_proba float. Defaults to 3.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

get_state_tuples()

returns a list of State (id, property) tuples for the component and all subcomponents

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

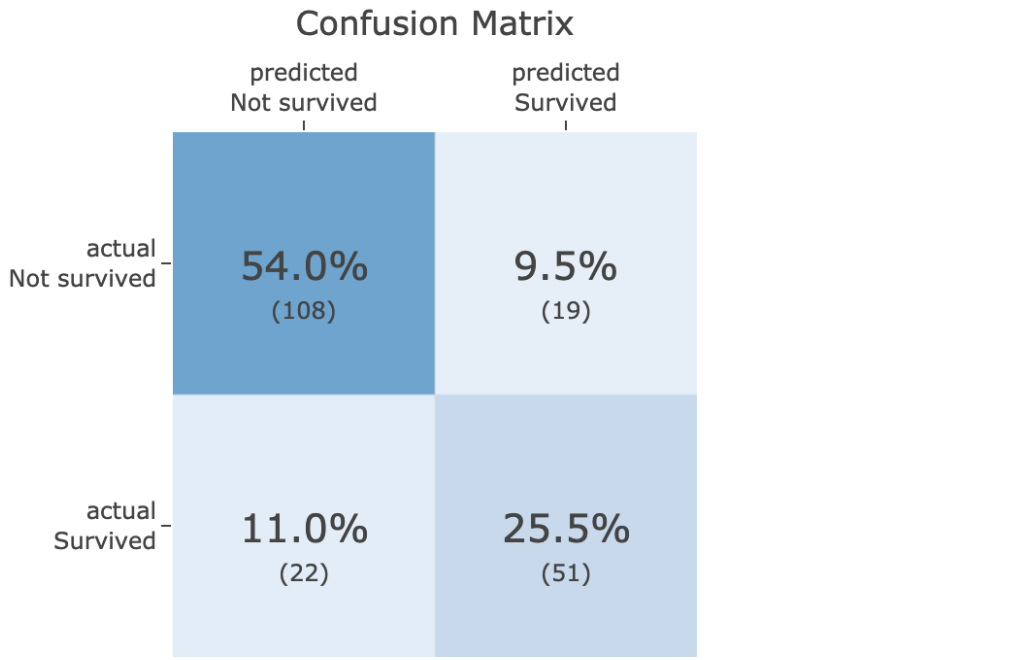
Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

ConfusionMatrixComponent



Cutoff prediction probability:

0.01 0.25 0.50 0.75 0.99

◉Display percentages

◉Binary (use cutoff for positive vs not positive)

class explainerdashboard.dashboard_components.classifier_components.**ConfusionMatrixComponent**(*explainer*, *title='Confusio Ma- trix'*, *name=None*, *sub- ti- tle='How many false pos- i- tives and false neg- a- tives?'*, *hide_title=Fa hide_subtitle= hide_footer= hide_cutoff= hide_percenta hide_binary= hide_selector hide_popout= hide_normali nor- mal- ize='all'*, *pos_label=No cut- off=0.5*, *per- cent- age=True*, *bi- nary=True*, *de- scrip- tion=None*, ***kwargs*)

Display confusion matrix component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Confusion Matrix".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle

- **hide_title** (*bool, optional*) – hide title.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_cutoff** (*bool, optional*) – Hide cutoff slider. Defaults to False.
- **hide_percentage** (*bool, optional*) – Hide percentage toggle. Defaults to False.
- **hide_binary** (*bool, optional*) – Hide binary toggle. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **cutoff** (*float, optional*) – Default cutoff. Defaults to 0.5.
- **percentage** (*bool, optional*) – Display percentages instead of counts. Defaults to True.
- **binary** (*bool, optional*) – Show binary instead of multiclass confusion matrix. Defaults to True.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.
- **normalize** (*str['true', 'pred', 'all']*) – normalizes confusion matrix over the true (rows), predicted (columns) conditions or all the population. Defaults to all

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

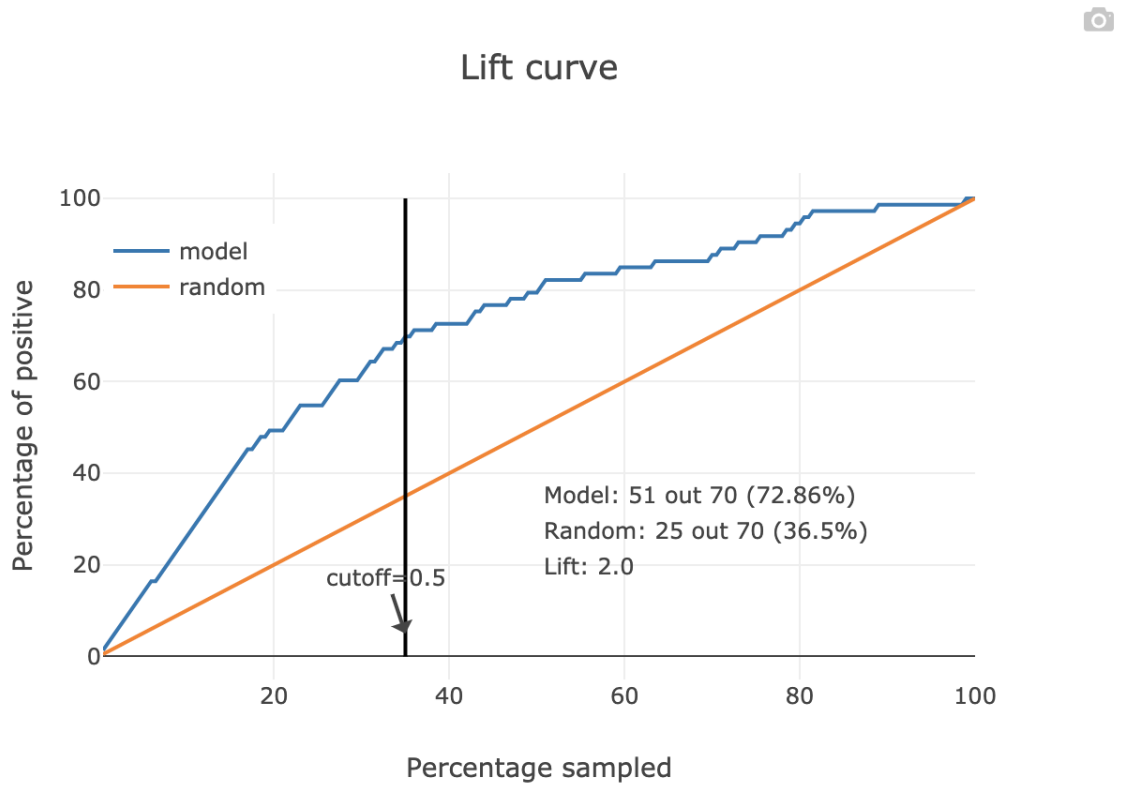
Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

LiftCurveComponent



Cutoff prediction probability:

0.01 0.25 0.50 0.75 0.99

Display percentages

class explainerdashboard.dashboard_components.classifier_components.**LiftCurveComponent**(*explainer*, *title='Lift Curve'*, *name=None*, *subtitle='Performance how much better than random?'*, *hide_title=False*, *hide_subtitle=False*, *hide_footer=False*, *hide_cutoff=False*, *hide_percentage=Fal*
hide_wizard=False, *hide_selector=False*, *hide_popout=False*, *pos_label=None*, *cutoff=0.5*, *percentage=True*, *wizard=True*, *description=None*, ***kwargs*)

Show liftcurve component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Lift Curve".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide title.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_cutoff** (*bool, optional*) – Hide cutoff slider. Defaults to False.
- **hide_percentage** (*bool, optional*) – Hide percentage toggle. Defaults to False.

- **hide_wizard** (*bool, optional*) – hide the wizard toggle. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **cutoff** (*float, optional*) – Cutoff for lift curve. Defaults to 0.5.
- **percentage** (*bool, optional*) – Display percentages instead of counts. Defaults to True.
- **wizard** (*bool, optional*) – display the wizard in the graph.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

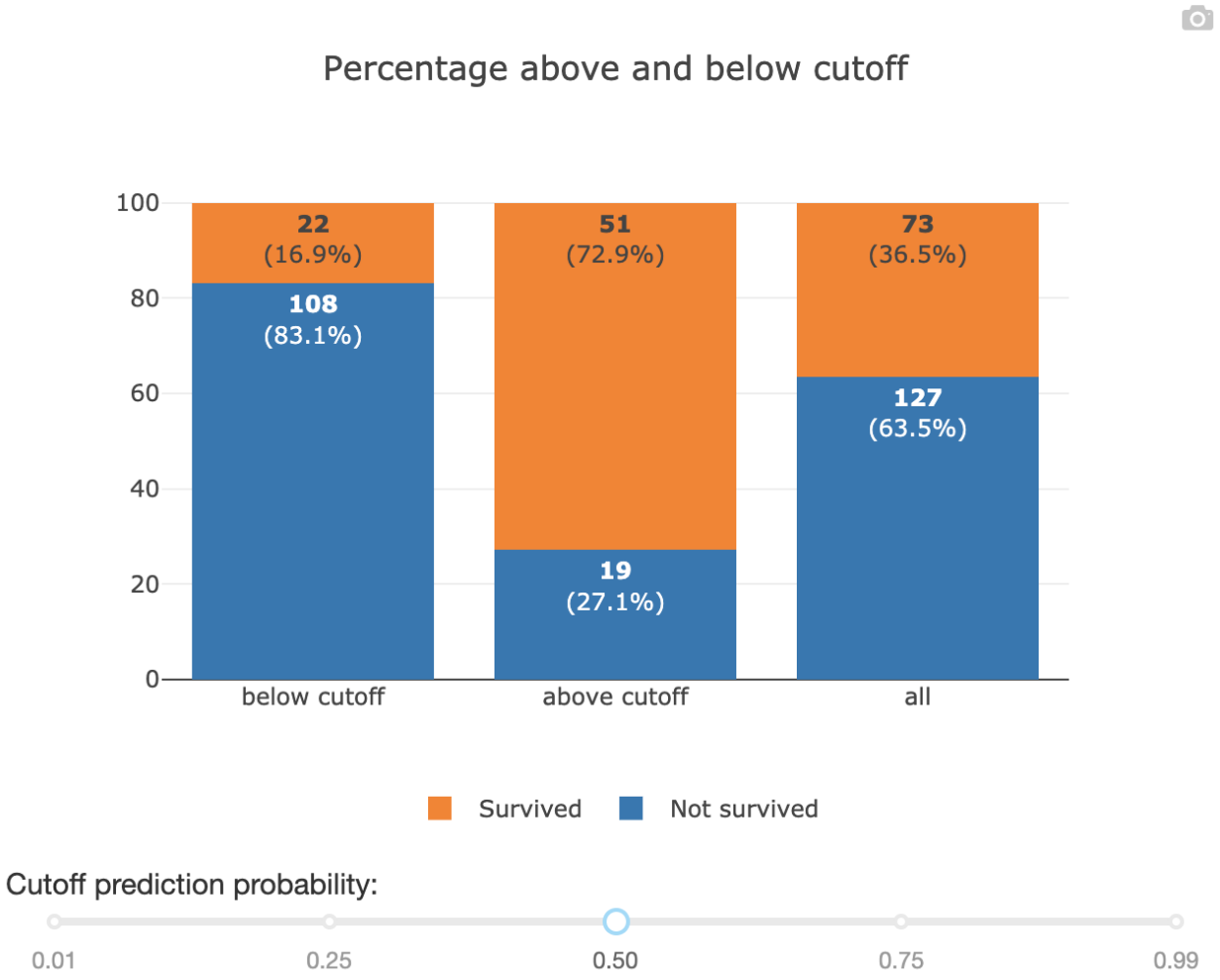
Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

ClassificationComponent



Cutoff prediction probability:

0.01 0.25 0.50 0.75 0.99

Display percentages

class explainerdashboard.dashboard_components.classifier_components.**ClassificationComponent**(*explainer*, *title='Classifica Plot'*, *name=None*, *subtitle='Distributi of labels above and below cutoff'*, *hide_title=Fal hide_subtitle= hide_footer=F hide_cutoff=F hide_percentag hide_selector= hide_popout= pos_label=No cutoff=0.5*, *percentage=True*, *description=None*, ***kwargs*)

Shows a barchart of the number of classes above the cutoff and below the cutoff.

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Classification Plot".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide the title.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_cutoff** (*bool, optional*) – Hide cutoff slider. Defaults to False.
- **hide_percentage** (*bool, optional*) – Hide percentage toggle. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.

- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **cutoff** (*float, optional*) – Cutoff for prediction. Defaults to 0.5.
- **percentage** (*bool, optional*) – Show percentage instead of counts. Defaults to True.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

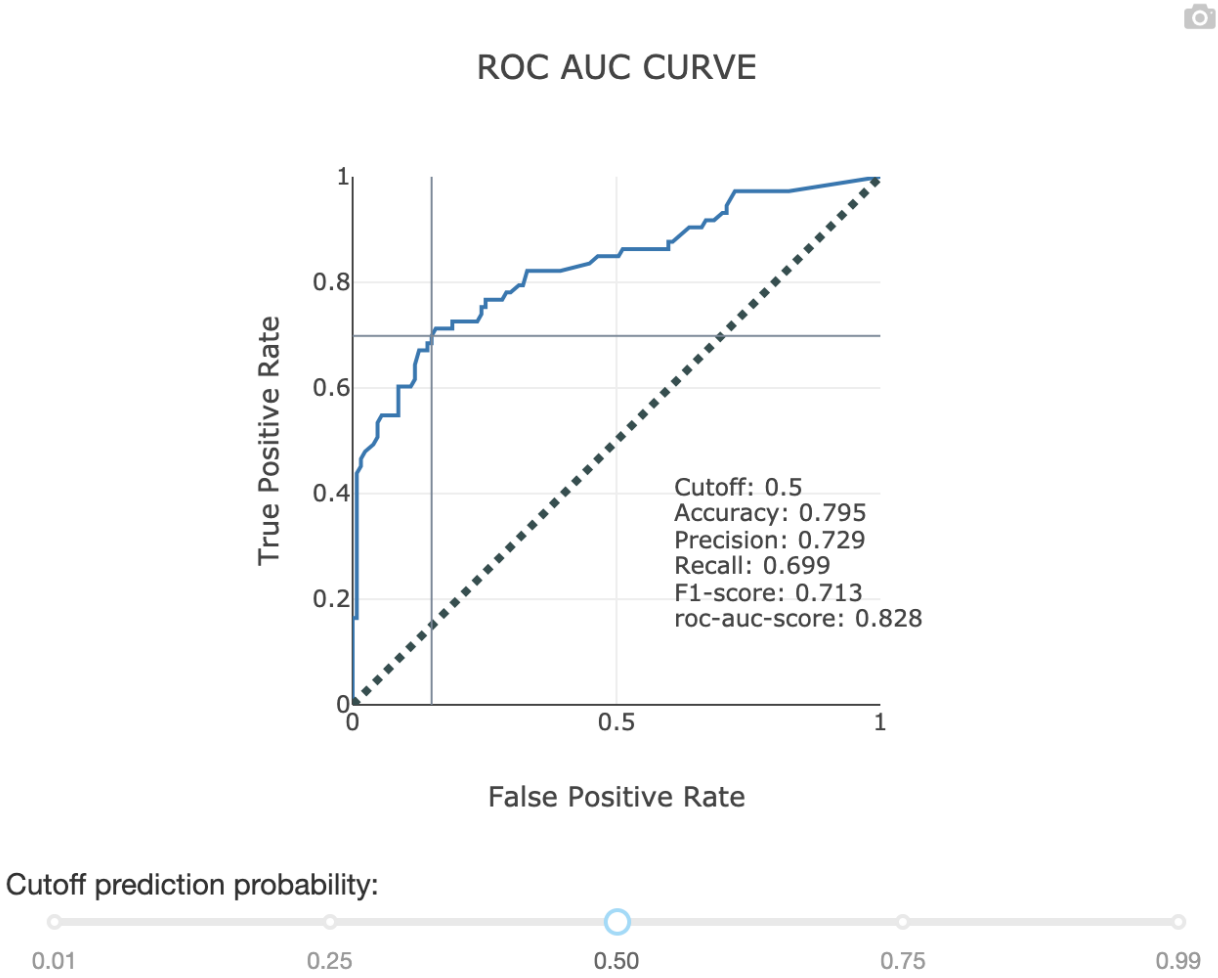
Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

RocAucComponent



class explainerdashboard.dashboard_components.classifier_components.**RocAucComponent**(*explainer*, *title='ROC AUC Plot'*, *name=None*, *subtitle='Trade-off between False positives and false negatives'*, *hide_title=False*, *hide_subtitle=False*, *hide_footer=False*, *hide_cutoff=False*, *hide_selector=False*, *hide_popout=False*, *pos_label=None*, *cutoff=0.5*, *description=None*, ***kwargs*)

Show ROC AUC curve component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "ROC AUC Plot".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide title.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_cutoff** (*bool, optional*) – Hide cutoff slider. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **cutoff** (*float, optional*) – default cutoff. Defaults to 0.5.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

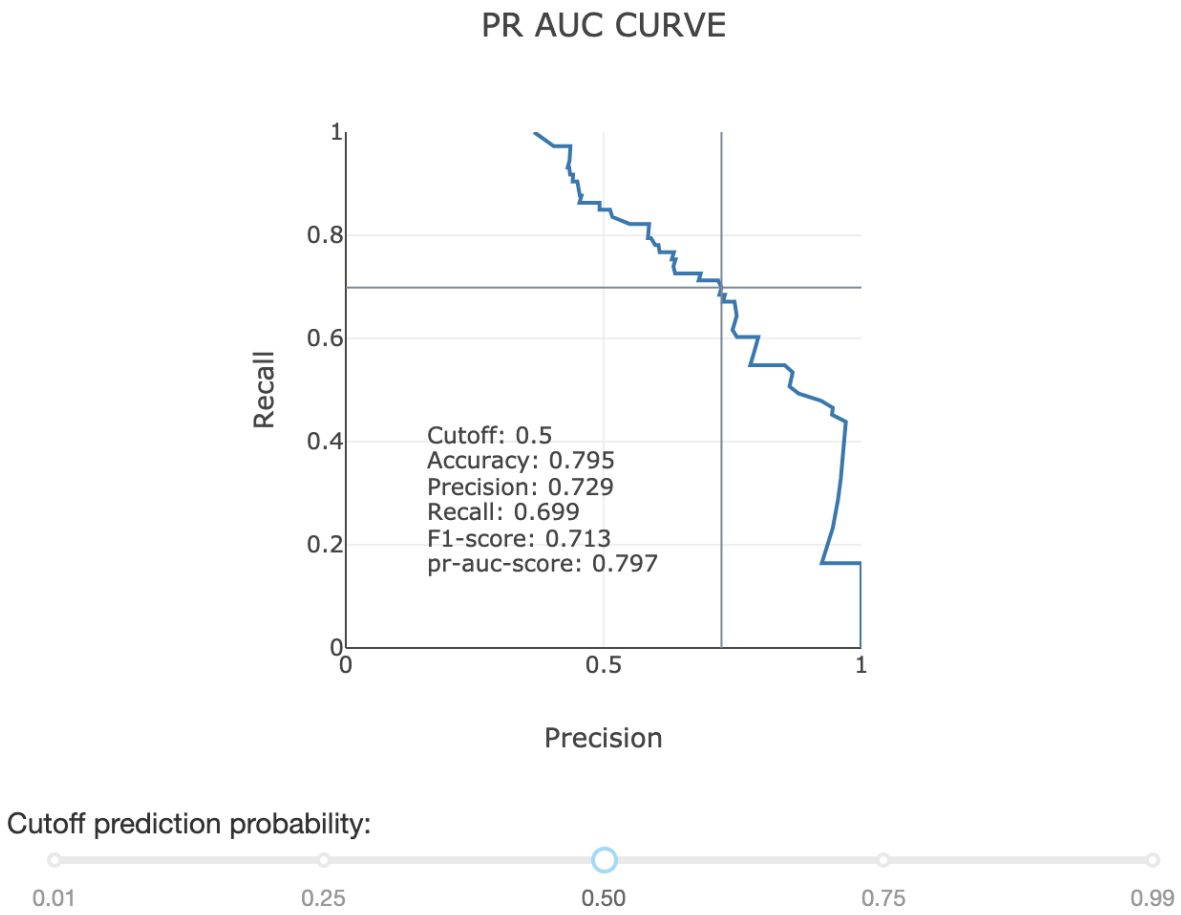
Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

PrAucComponent



class explainerdashboard.dashboard_components.classifier_components.**PrAucComponent**(*explainer*, *title='PR AUC Plot'*, *name=None*, *subtitle='Trade-off between Precision and Recall'*, *hide_title=False*, *hide_subtitle=False*, *hide_footer=False*, *hide_cutoff=False*, *hide_selector=False*, *hide_popout=False*, *pos_label=None*, *cutoff=0.5*, *description=None*, ***kwargs*)

Display PR AUC plot component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "PR AUC Plot".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide title.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_cutoff** (*bool, optional*) – hide cutoff slider. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **cutoff** (*float, optional*) – default cutoff. Defaults to 0.5.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

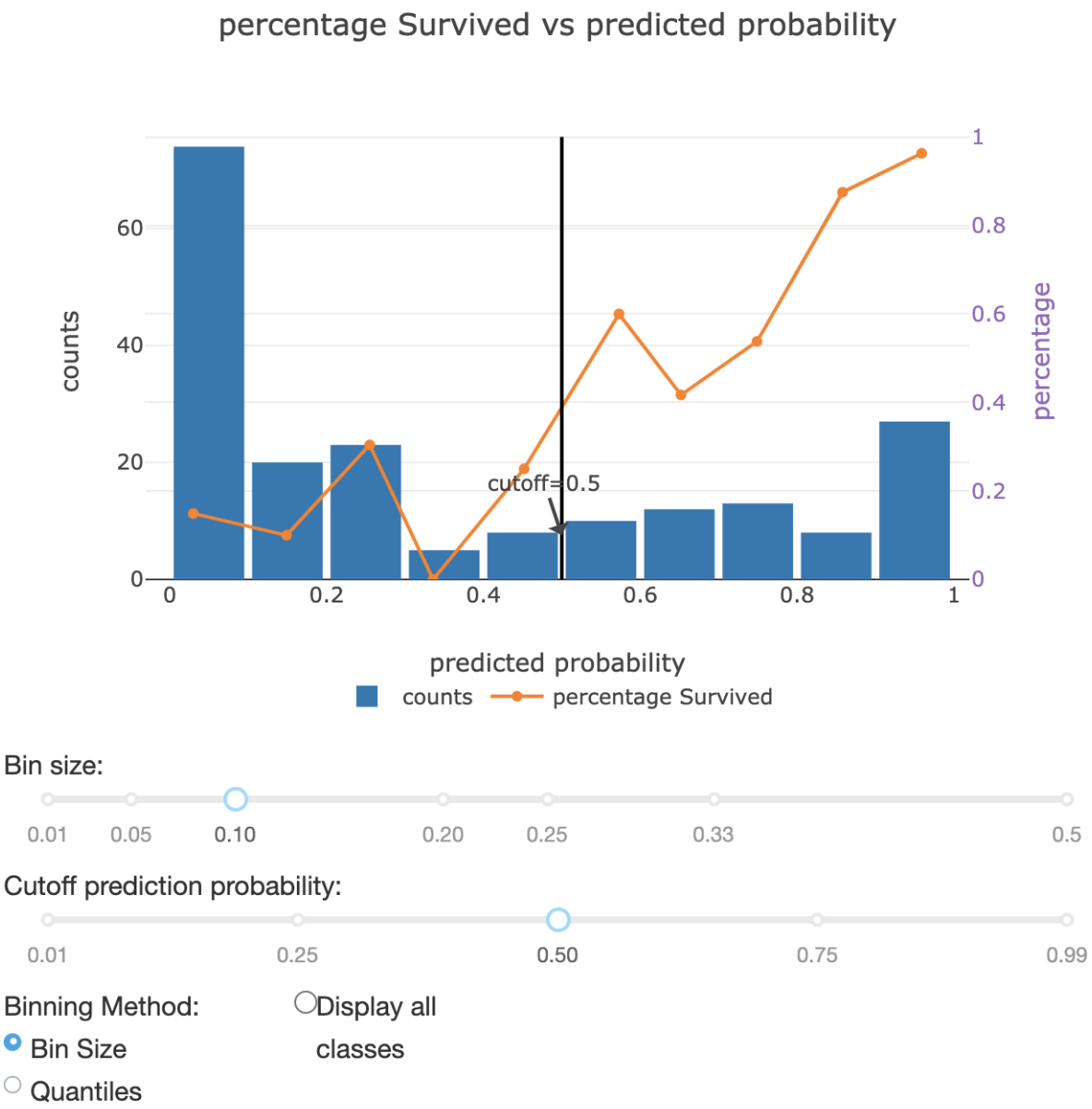
Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

PrecisionComponent



class explainerdashboard.dashboard_components.classifier_components.**PrecisionComponent**(*explainer*, *title='Precision Plot'*, *name=None*, *subtitle='Does fraction positive increase with predicted probability?'*, *hide_title=False*, *hide_subtitle=False*, *hide_footer=False*, *hide_cutoff=False*, *hide_binsize=False*, *hide_binmethod=Fal*
hide_multiclass=Fals
hide_selector=False, *hide_popout=False*, *pos_label=None*, *bin_size=0.1*, *quantiles=10*, *cutoff=0.5*, *quantiles_or_binsize='bin*
multiclass=False, *description=None*, ***kwargs*)

Shows a precision graph with toggles.

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Precision Plot".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random

uuid is generated to make sure it's unique. Defaults to None.

- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide title
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_cutoff** (*bool, optional*) – Hide cutoff slider. Defaults to False.
- **hide_binsize** (*bool, optional*) – hide binsize/quantiles slider. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **hide_binmethod** (*bool, optional*) – Hide binsize/quantiles toggle. Defaults to False.
- **hide_multiclass** (*bool, optional*) – Hide multiclass toggle. Defaults to False.
- **hide_selector** – Hide pos label selector. Default to True.
- **hide_popout** (*bool, optional*) – hide popout button
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **bin_size** (*float, optional*) – Size of bins in probability space. Defaults to 0.1.
- **quantiles** (*int, optional*) – Number of quantiles to divide plot. Defaults to 10.
- **cutoff** (*float, optional*) – Cutoff to display in graph. Defaults to 0.5.
- **quantiles_or_binsize** (*str, {'quantiles', 'bin_size'}, optional*) – Default bin method. Defaults to 'bin_size'.
- **multiclass** (*bool, optional*) – Display all classes. Defaults to False.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

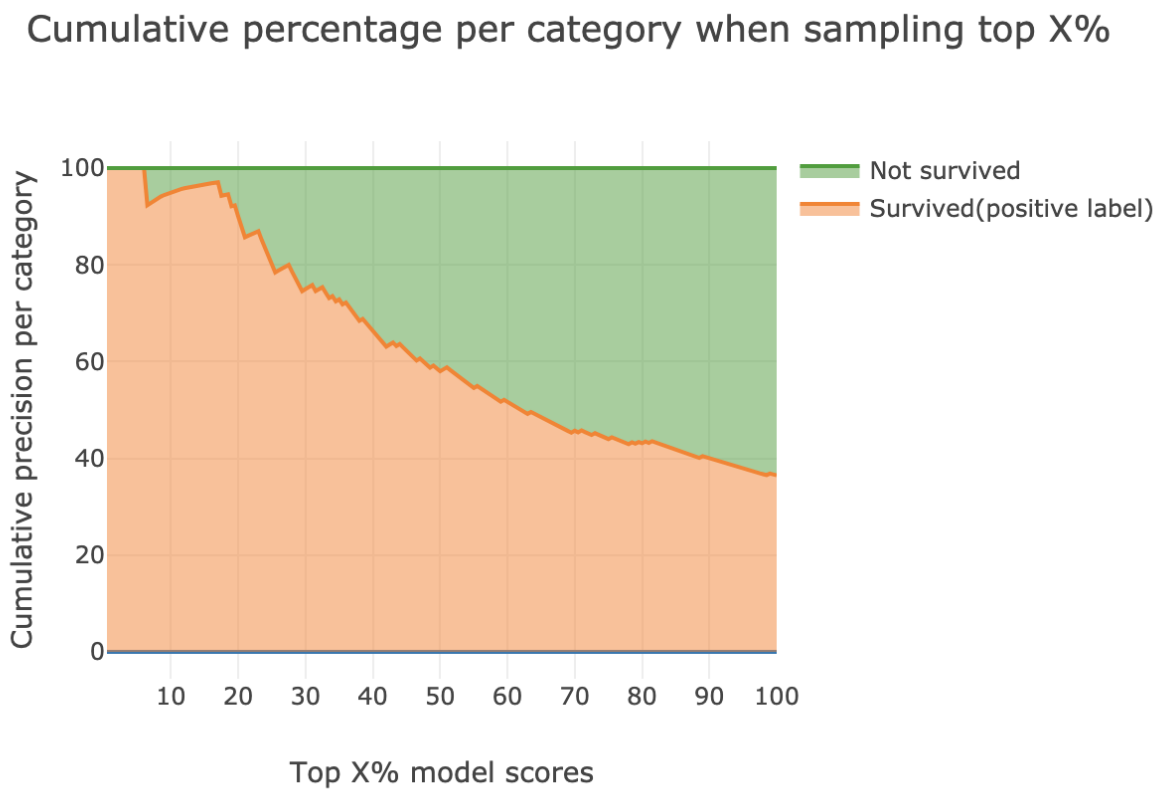
Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

CumulativePrecisionComponent



class explainerdashboard.dashboard_components.classifier_components.**CumulativePrecisionComponent**(*explaine*
ti-
tle='Cu
Pre-
ci-
sion',
name=N
sub-
ti-
tle='Exp
dis-
tri-
bu-
tion
for
high-
est
scores',
hide_titl
hide_su
hide_foo
hide_sel
hide_po
pos_lab
hide_cu
cut-
off=Non
hide_pe
per-
centile=
de-
scrip-
tion=No
***kwarg*

Show cumulative precision component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Cumulative Precision".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide the title.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

ClassifierModelSummaryComponent

class explainerdashboard.dashboard_components.classifier_components.**ClassifierModelSummaryComponent**(*expl*
ti-
tle=
per-
for-
man
met-
rics'
nam
hide
hide
hide
hide
hide
*pos_*
cut-
off=
roun
show
de-
scrip
tion=
***kw*

Show model summary statistics (accuracy, precision, recall,
f1, roc_auc, pr_auc, log_loss) component.

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Model performance metrics".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_title** (*bool, optional*) – hide title.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_cutoff** (*bool, optional*) – hide cutoff slider. Defaults to False.

- **hide_selector** (*bool, optional*) – hide pos label selector. Defaults to False.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **cutoff** (*float, optional*) – default cutoff. Defaults to 0.5.
- **round** (*int*) – round floats. Defaults to 3.
- **show_metrics** (*List*) – list of metrics to display in order. Defaults to None, displaying all metrics.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

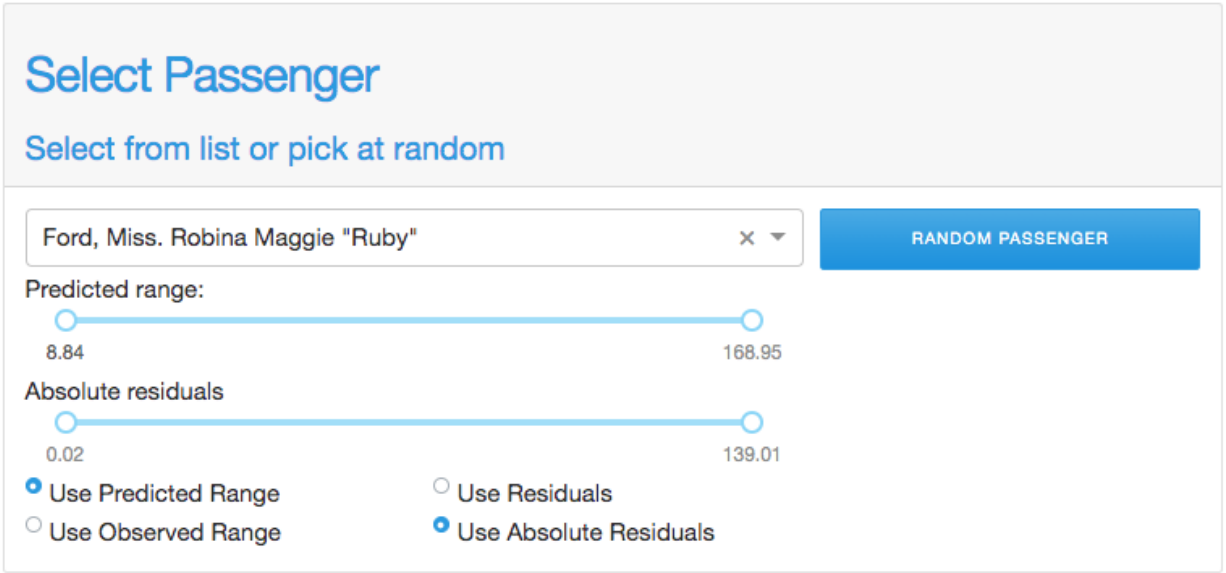
state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

4.7.5 regression_components

RegressionRandomIndexComponent



class explainerdashboard.dashboard_components.regression_components.**RegressionRandomIndexComponent**(*explainer*, *title='Random Index'*, *name=None*, *subtitle='Select from list or pick at random'*, *hide_title=False*, *hide_subtitle=False*, *hide_index=False*, *hide_predindex=False*, *hide_predictionslider=False*, *hide_residualslider=False*, *hide_absresidualslider=False*, *hide_predabserrorslider=False*, *hide_rounding=False*, *index_dropdown=True*, *index=None*, *pred_or_y_slider='pred'*, *residual_slider='abs'*, *abs_residual_slider=None*, *pred_abserror_slider=None*, *rounding=3*, *description=None*, ***kwargs*)

Select a random index subject to constraints component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Select Random Index".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide title
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_index** (*bool, optional*) – Hide index selector. Defaults to False.

- **hide_pred_slider** (*bool, optional*) – Hide prediction slider. Defaults to False.
- **hide_residual_slider** (*bool, optional*) – hide residuals slider. Defaults to False.
- **hide_pred_or_y** (*bool, optional*) – hide prediction or actual toggle. Defaults to False.
- **hide_abs_residuals** (*bool, optional*) – hide absolute residuals toggle. Defaults to False.
- **hide_button** (*bool, optional*) – hide button. Defaults to False.
- **index_dropdown** (*bool, optional*) – Use dropdown for index input instead of free text input. Defaults to True.
- **index** (*{str, int}, optional*) – Initial index to display. Defaults to None.
- **pred_slider** (*[lb, ub], optional*) – Initial values for prediction values slider [lower-bound, upperbound]. Defaults to None.
- **y_slider** (*[lb, ub], optional*) – Initial values for y slider [lower bound, upper bound]. Defaults to None.
- **residual_slider** (*[lb, ub], optional*) – Initial values for residual slider [lower bound, upper bound]. Defaults to None.
- **abs_residual_slider** (*[lb, ub], optional*) – Initial values for absolute residuals slider [lower bound, upper bound] Defaults to None.
- **pred_or_y** (*str, {'preds', 'y'}, optional*) – Initial use predictions or y slider. Defaults to "preds".
- **abs_residuals** (*bool, optional*) – Initial use residuals or absolute residuals. Defaults to True.
- **round** (*int, optional*) – rounding used for slider spacing. Defaults to 2.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

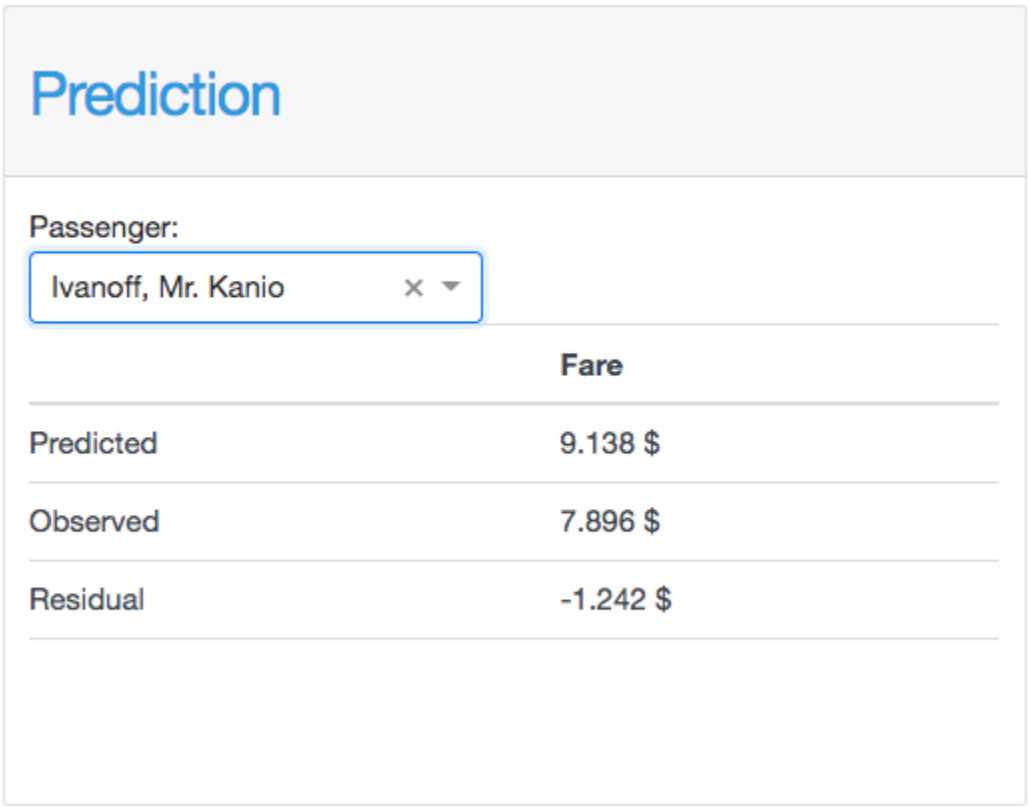
Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

RegressionPredictionSummaryComponent



class explainerdashboard.dashboard_components.regression_components.**RegressionPredictionSummaryComponen**

Shows a summary for a particular prediction

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()

- **title** (*str, optional*) – Title of tab or page. Defaults to "Prediction".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_index** (*bool, optional*) – hide index selector. Defaults to False.
- **hide_title** (*bool, optional*) – hide title. Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_table** (*bool, optional*) – hide the results table
- **index_dropdown** (*bool, optional*) – Use dropdown for index input instead of free text input. Defaults to True.
- **feature_input_component** (FeatureInputComponent) – A FeatureInputComponent that will give the input to the graph instead of the index selector. If not None, hide_index=True. Defaults to None.
- **index** (*{int, str}, optional*) – Index to display prediction summary for. Defaults to None.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

get_state_tuples()

returns a list of State (id, property) tuples for the component and all subcomponents

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

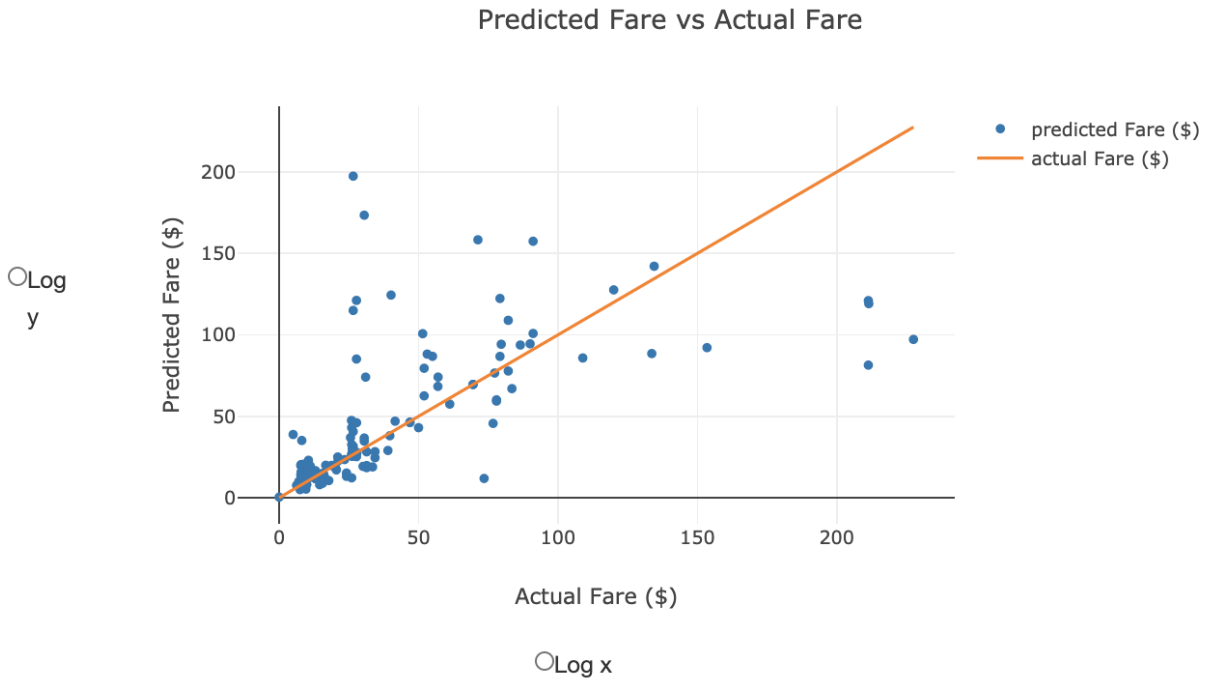
Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

PredictedVsActualComponent



class explainerdashboard.dashboard_components.regression_components.**PredictedVsActualComponent**(*explainer*, *title='Predicted vs Actual'*, *name=None*, *subtitle='How close is the predicted value to the observed?'*, *hide_title=False*, *hide_subtitle=False*, *hide_log_x=False*, *hide_log_y=False*, *hide_popout=False*, *logs=False*, *log_x=False*, *log_y=False*, *round=3*, *plot_sample=None*, *description=None*, ***kwargs*)

Shows a plot of predictions vs y.

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Predicted vs Actual".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) –
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_log_x** (*bool, optional*) – Hide the log_x toggle. Defaults to False.
- **hide_log_y** (*bool, optional*) – Hide the log_y toggle. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **logs** (*bool, optional*) – Whether to use log axis. Defaults to False.

- **log_x** (*bool, optional*) – log only x axis. Defaults to False.
- **log_y** (*bool, optional*) – log only y axis. Defaults to False.
- **round** (*int, optional*) – rounding to apply to float predictions. Defaults to 3.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

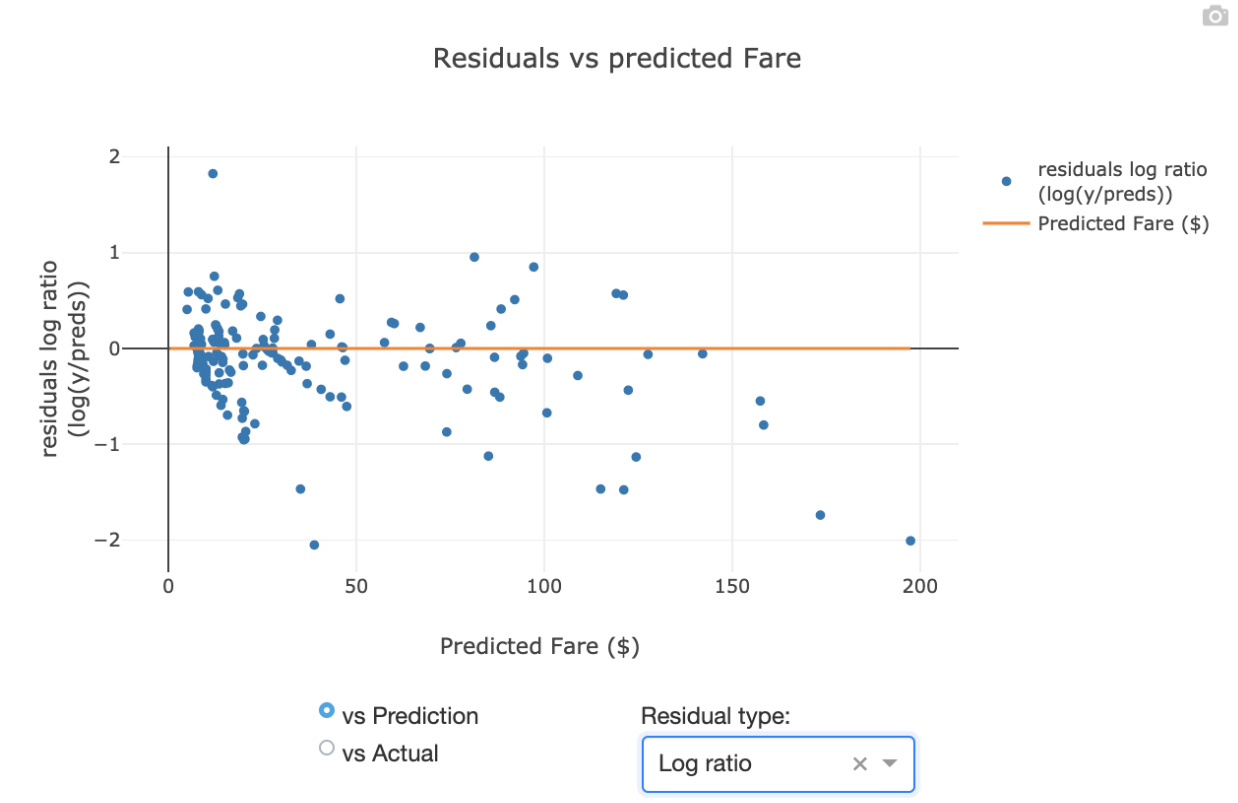
Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

ResidualsComponent



class explainerdashboard.dashboard_components.regression_components.**ResidualsComponent**(*explainer*, *title='Residuals'*, *name=None*, *subtitle='How much is the model off?'*, *hide_title=False*, *hide_subtitle=False*, *hide_footer=False*, *hide_pred_or_actual=*
hide_ratio=False, *hide_popout=False*, *pred_or_actual='vs_*
residuals='difference', *round=3*, *plot_sample=None*, *description=None*, ***kwargs*)

Residuals plot component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Residuals".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) –
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_pred_or_actual** (*bool, optional*) – hide vs predictions or vs actual for x-axis toggle. Defaults to False.
- **hide_ratio** (*bool, optional*) – hide residual type dropdown. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **pred_or_actual** (*str, {'vs_actual', 'vs_pred'}, optional*) – Whether to plot actual or predictions on the x-axis. Defaults to "vs_pred".
- **residuals** (*str, {'difference', 'ratio', 'log-ratio'} optional*) – How to calcualte residuals. Defaults to 'difference'.

- **round** (*int, optional*) – rounding to apply to float predictions. Defaults to 3.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

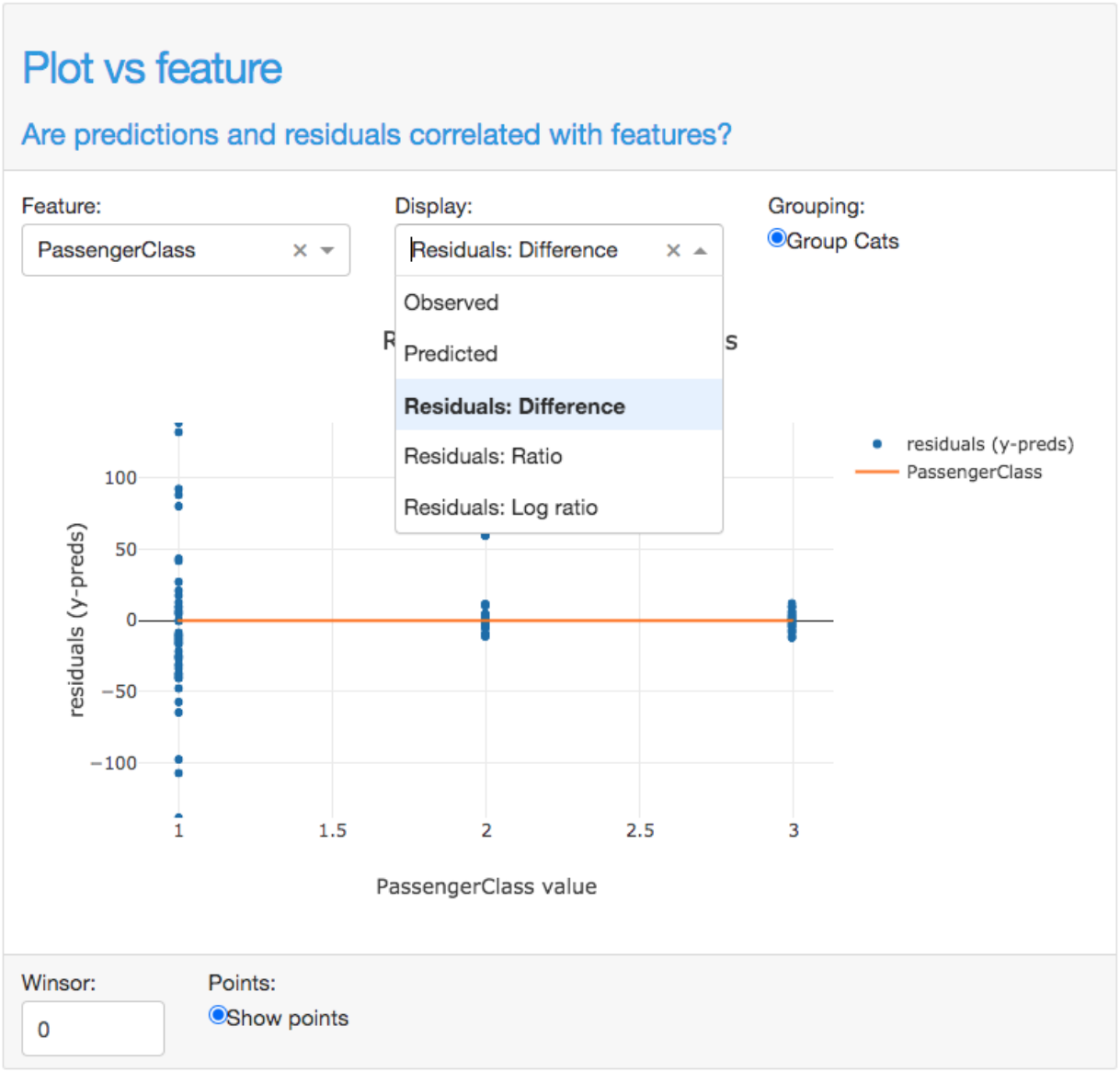
return static html for this component and all subcomponents.

Parameters
state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

RegressionVsColComponent



class explainerdashboard.dashboard_components.regression_components.**RegressionVsColComponent**(*explainer*, *title='Plot vs feature'*, *name=None*, *subtitle='Are predictions and residuals correlated with features?'*, *hide_title=Fa*
hide_subtitle=
hide_footer=
hide_col=Fal
hide_ratio=F
hide_points=
hide_winsor=
hide_cats_top
hide_cats_so
hide_popout=
col=None, *display='differen*
round=3, *points=True*, *winsor=0*, *cats_topx=10*
cats_sort='fr
plot_sample=
description=None, ***kwargs*)

Show residuals, observed or preds vs a particular Feature component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Plot vs feature".

- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) –
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_footer** (*bool, optional*) – hide the footer at the bottom of the component
- **hide_col** (*bool, optional*) – Hide de column selector. Defaults to False.
- **hide_ratio** (*bool, optional*) – Hide the toggle. Defaults to False.
- **hide_points** (*bool, optional*) – Hide group points toggle. Defaults to False.
- **hide_winsor** (*bool, optional*) – Hide winsor input. Defaults to False.
- **hide_cats_topx** (*bool, optional*) – hide the categories topx input. Defaults to False.
- **hide_cats_sort** (*bool, optional*) – hide the categories sort selector.Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button. Defaults to False.
- **col** (*[type], optional*) – Initial feature to display. Defaults to None.
- **display** (*str, {'observed', 'predicted', difference', 'ratio', 'log-ratio'} optional*) – What to display on y axis. Defaults to 'difference'.
- **round** (*int, optional*) – rounding to apply to float predictions. Defaults to 3.
- **points** (*bool, optional*) – display point cloud next to violin plot for categorical cols. Defaults to True
- **winsor** (*int, 0-50, optional*) – percentage of outliers to winsor out of the y-axis. Defaults to 0.
- **cats_topx** (*int, optional*) – maximum number of categories to display for categorical features. Defaults to 10.
- **cats_sort** (*str, optional*) – how to sort categories: 'alphabet', 'freq' or 'shap'. Defaults to 'freq'.
- **plot_sample** (*int, optional*) – Instead of all points only plot a random sample of points. Defaults to None (=all points)
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

RegressionModelSummaryComponent

class explainerdashboard.dashboard_components.regression_components.**RegressionModelSummaryComponent**(*expl*
ti-
tle=
Sum
mar
nam
sub-
ti-
tle=
met-
rics
for
mod
per-
for-
man
hide
hide
rou
sho
de-
scri
tion=
***kv*

Show model summary statistics (RMSE, MAE, R2) component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Model Summary".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide title
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **round** (*int*) – rounding to perform to metric floats.
- **show_metrics** (*List*) – list of metrics to display in order. Defaults to None, displaying all metrics.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

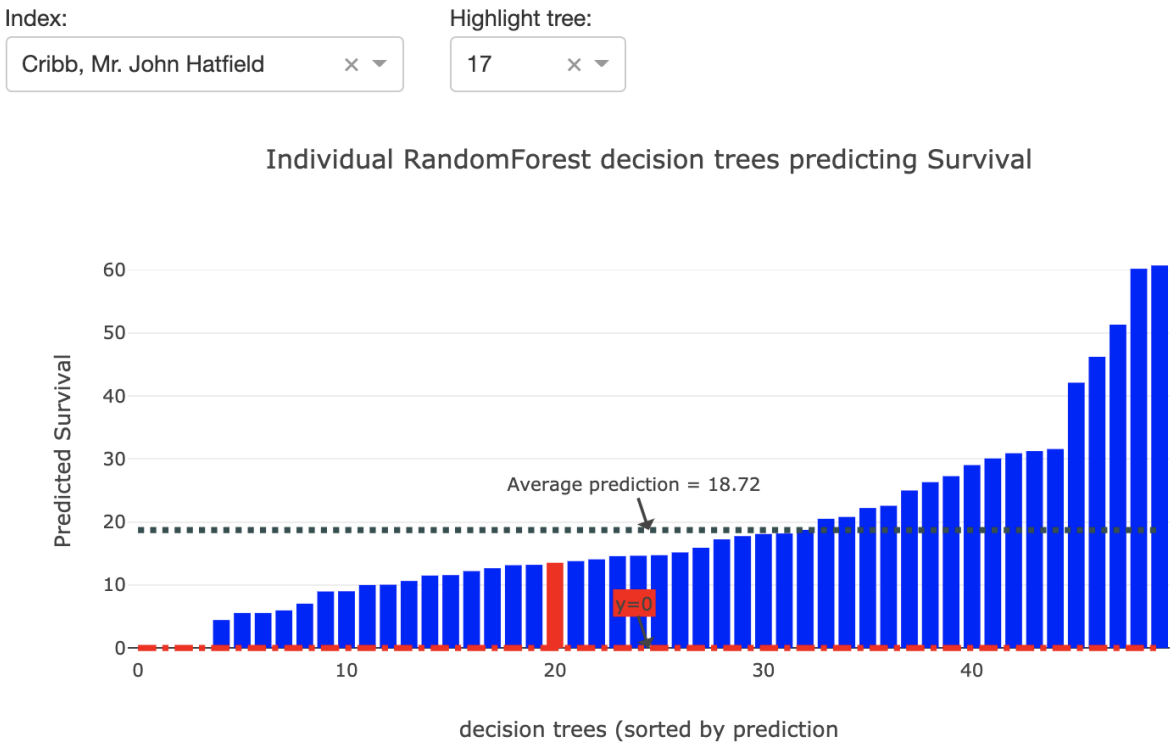
to_html(*state_dict=None*, *add_header=True*)

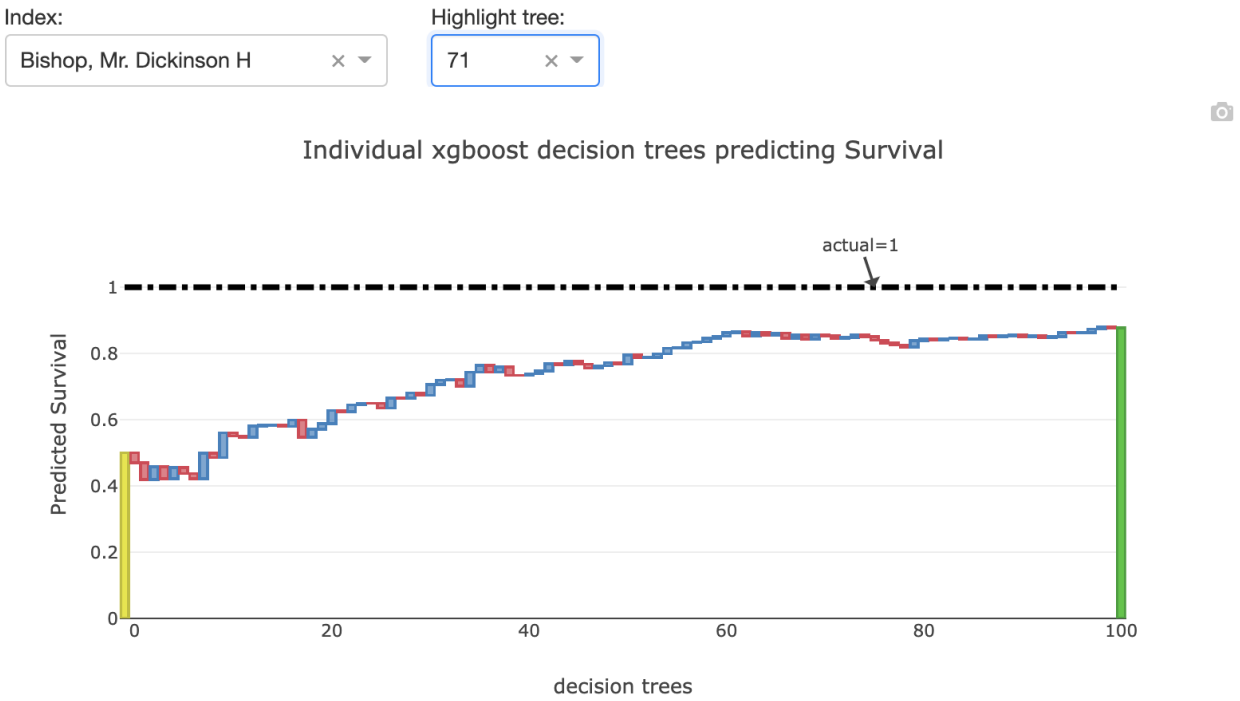
return static html for this component and all subcomponents.

Parameters
state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

4.7.6 decisiontree_components

DecisionTreesComponent




Index:
Bishop, Mr. Dickinson H
Highlight tree:
71
Individual xgboost decision trees predicting Survival
actual=1
Predicted Survival
1
0.8
0.6
0.4
0.2
0
0
20
40
60
80
100
decision trees

class explainerdashboard.dashboard_components.decisiontree_components.**DecisionTreesComponent**(*explainer*, *title='Decision Trees'*, *name=None*, *subtitle='Displayi individual decision trees'*, *hide_title=Fa hide_subtitle= hide_index=F hide_highligh hide_selector hide_popout= index_dropdow pos_label=No index=None*, *highlight=None*, *higher_is_be description=None*, ***kwargs*)

Show prediction from individual decision trees inside RandomForest component

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Decision Trees".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide title, Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_index** (*bool, optional*) – Hide index selector. Defaults to False.
- **hide_highlight** (*bool, optional*) – Hide tree highlight selector. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selectors. Defaults to False.
- **hide_popout** (*bool, optional*) – hide popout button

- **index_dropdown** (*bool, optional*) – Use dropdown for index input instead of free text input. Defaults to True.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **index** (*{str, int}, optional*) – Initial index to display. Defaults to None.
- **highlight** (*int, optional*) – Initial tree to highlight. Defaults to None.
- **higher_is_better** (*bool, optional*) – up is green, down is red. If False flip the colors. (for gbm models only)
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

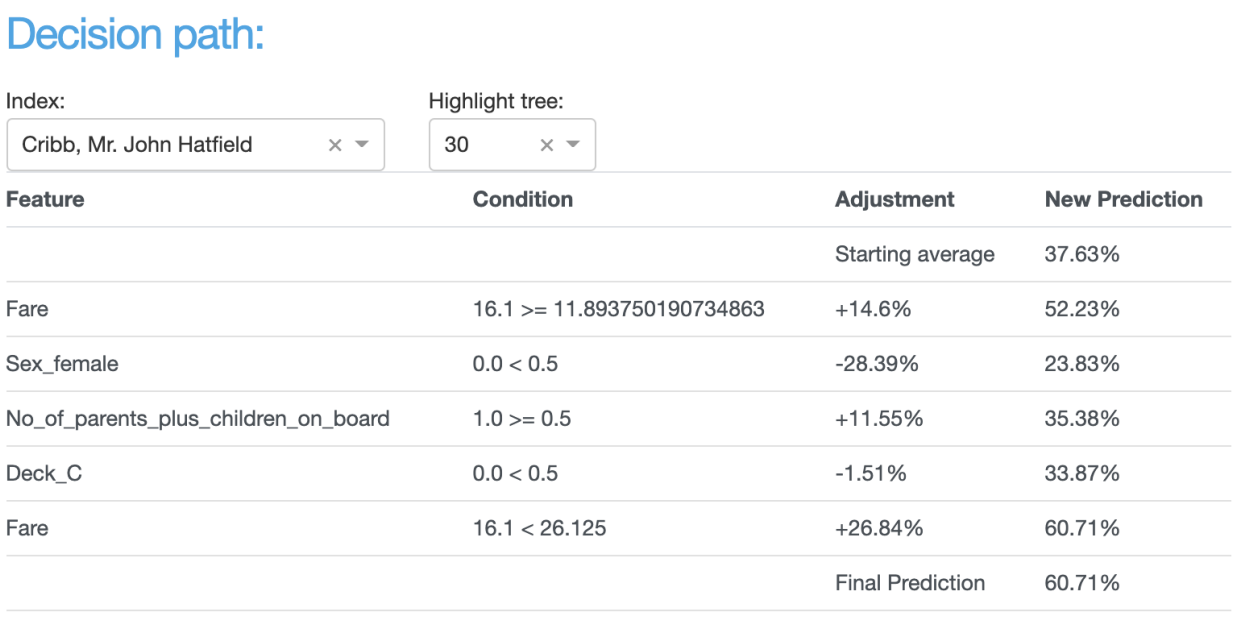
Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

DecisionPathTableComponent



Feature	Condition	Adjustment	New Prediction
		Starting average	37.63%
Fare	16.1 >= 11.893750190734863	+14.6%	52.23%
Sex_female	0.0 < 0.5	-28.39%	23.83%
No_of_parents_plus_children_on_board	1.0 >= 0.5	+11.55%	35.38%
Deck_C	0.0 < 0.5	-1.51%	33.87%
Fare	16.1 < 26.125	+26.84%	60.71%
		Final Prediction	60.71%

class explainerdashboard.dashboard_components.decisiontree_components.**DecisionPathTableComponent**(*explaine*
ti-
tle='Dec
path
ta-
ble',
name=N
sub-
ti-
tle='Dec
path
through
de-
ci-
sion
tree',
hide_titl
hide_sul
hide_ind
hide_hig
hide_sel
in-
dex_dro
pos_labe
in-
dex=No
high-
light=N
de-
scrip-
tion=No
***kwarg*

Display a table of the decision path through a particular decision tree

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Decision path table".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide title, Defaults to False.
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_index** (*bool, optional*) – Hide index selector. Defaults to False.
- **hide_highlight** (*bool, optional*) – Hide tree index selector. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selectors. Defaults to False.
- **index_dropdown** (*bool, optional*) – Use dropdown for index input instead of free text input. Defaults to True.
- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label

- **index** (*{str, int}, optional*) – Initial index to display decision path for. Defaults to None.
- **highlight** (*int, optional*) – Initial tree idx to display decision path for. Defaults to None.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

to_html(*state_dict=None*, *add_header=True*)

return static html for this component and all subcomponents.

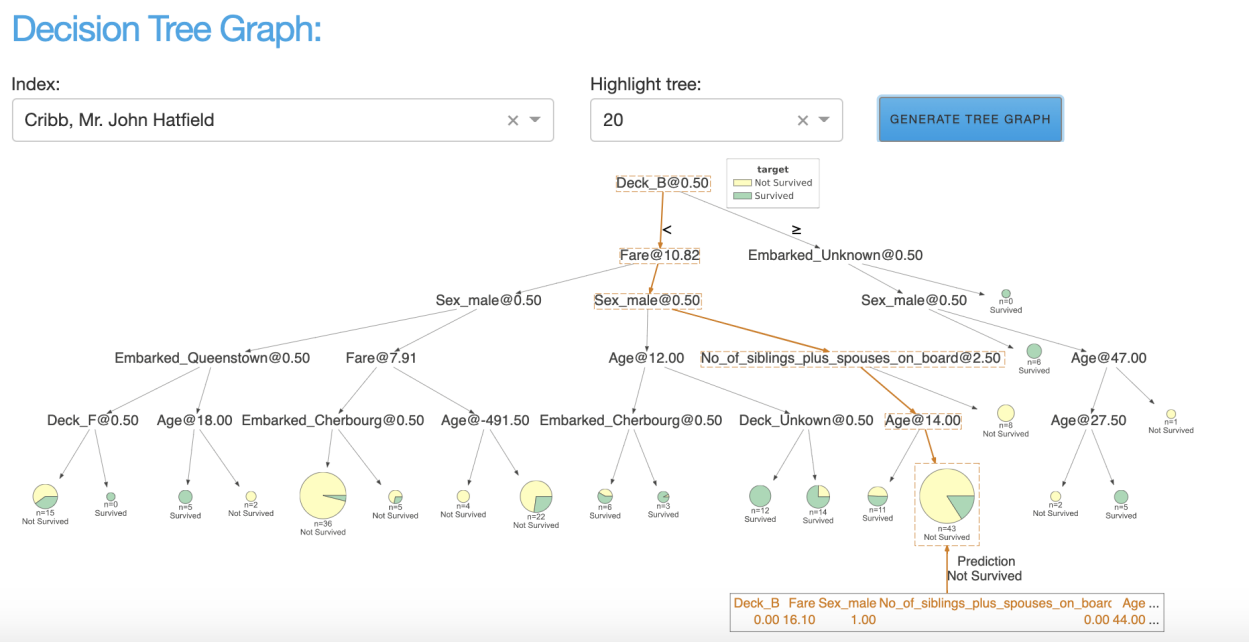
Parameters

state_dict (*dict*) – dictionary with id_prop_tuple as keys and state as value.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

DecisionPathGraphComponent



class explainerdashboard.dashboard_components.decisiontree_components.**DecisionPathGraphComponent**(*explainer*, *title='Decision path graph'*, *name=None*, *subtitle='Visualize the entire decision tree'*, *hide_title=False*, *hide_subtitle=False*, *hide_index=False*, *hide_highlight=False*, *hide_button=False*, *hide_selector=False*, *index_dropdown=True*, *pos_label=None*, *index=None*, *highlight=None*, *description=None*, ***kwargs*)

Display dtreeviz decision path

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Decision path graph".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **subtitle** (*str*) – subtitle
- **hide_title** (*bool, optional*) – hide title
- **hide_subtitle** (*bool, optional*) – Hide subtitle. Defaults to False.
- **hide_index** (*bool, optional*) – hide index selector. Defaults to False.
- **hide_highlight** (*bool, optional*) – hide tree idx selector. Defaults to False.
- **hide_button** (*bool, optional*) – hide the button, Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selectors. Defaults to False.
- **index_dropdown** (*bool, optional*) – Use dropdown for index input instead of free text input. Defaults to True.

- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **index** (*{str, int}, optional*) – Initial index to display. Defaults to None.
- **highlight** (*[type], optional*) – Initial tree idx to display. Defaults to None.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

4.7.7 Connectors

CutoffPercentileComponent

class explainerdashboard.dashboard_components.connectors.**CutoffPercentileComponent**(*explainer*, *title='Global cutoff'*, *name=None*, *hide_title=False*, *hide_cutoff=False*, *hide_percentile=False*, *hide_selector=False*, *pos_label=None*, *cutoff=0.5*, *percentile=None*, *description=None*, ***kwargs*)

Slider to set a cutoff for Classifier components, based on setting the cutoff at a certain percentile of predictions, e.g.: percentile=0.8 means "mark the 20% highest scores as positive".

This cutoff can then be conencted with other components like e.g. RocAucComponent with a CutoffConnector.

Parameters

- **explainer** (*Explainer*) – explainer object constructed with either ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to "Global Cutoff".
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.
- **hide_title** (*bool, optional*) – Hide title.
- **hide_cutoff** (*bool, optional*) – Hide the cutoff slider. Defaults to False.
- **hide_percentile** (*bool, optional*) – Hide percentile slider. Defaults to False.
- **hide_selector** (*bool, optional*) – hide pos label selectors. Defaults to False.

- **pos_label** (*{int, str}, optional*) – initial pos label. Defaults to explainer.pos_label
- **cutoff** (*float, optional*) – Initial cutoff. Defaults to 0.5.
- **percentile** (*[type], optional*) – Initial percentile. Defaults to None.
- **description** (*str, optional*) – Tooltip to display when hover over component title. When None default text is shown.

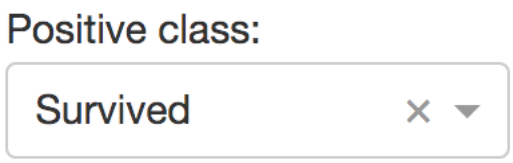
layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

PosLabelSelector



class explainerdashboard.dashboard_methods.**PosLabelSelector**(*explainer*, *title='Pos Label Selector'*, *name=None*, *pos_label=None*)

For classifier models displays a drop down menu with labels to be selected as the positive class.

Generates a positive label selector with element id 'pos_label-'+self.name

Parameters

- **explainer** (*Explainer*) – explainer object constructed with e.g. ClassifierExplainer() or RegressionExplainer()
- **title** (*str, optional*) – Title of tab or page. Defaults to None.
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to 'Pos Label Selector'.
- **pos_label** (*int, optional*) – Initial pos label. Defaults to explainer.pos_label.

layout()

layout to be defined by the particular ExplainerComponent instance. All element id's should append +self.name to make sure they are unique.

PosLabelConnector

class explainerdashboard.dashboard_components.connectors.**PosLabelConnector**(*input_pos_label*, *output_pos_labels*)

initialize the ExplainerComponent

Parameters

- **explainer** (*Explainer*) – explainer object constructed with e.g. ClassifierExplainer() or RegressionExplainer()

- **title** (*str, optional*) – Title of tab or page. Defaults to None.
- **name** (*str, optional*) – unique name to add to Component elements. If None then random uuid is generated to make sure it's unique. Defaults to None.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

CutoffConnector

class explainerdashboard.dashboard_components.connectors.**CutoffConnector**(*input_cutoff*, *output_cutoffs*)

Connect the cutoff selector of input_cutoff with those of output_cutoffs.

You can use this to connect a CutoffPercentileComponent with a RocAucComponent for example,

When you change the cutoff in input_cutoff, all the cutoffs in output_cutoffs will automatically be updated.

Parameters

- **input_cutoff** (*[{str, ExplainerComponent}]*) – Either a str or an ExplainerComponent. If str should be equal to the name of the cutoff property. If ExplainerComponent then should have a .cutoff_name property.
- **output_cutoffs** (*list(str, ExplainerComponent)*) – list of str of ExplainerComponents.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

IndexConnector

class explainerdashboard.dashboard_components.connectors.**IndexConnector**(*input_index*, *output_indexes*, *explainer=None*)

Connect the index selector of input_index with those of output_indexes.

You can use this to connect a RandomIndexComponent with a PredictionSummaryComponent for example.

When you change the index in input_index, all the indexes in output_indexes will automatically be updated.

Parameters

- **input_index** (*[{str, ExplainerComponent}]*) – Either a str or an ExplainerComponent. If str should be equal to the name of the index property. If ExplainerComponent then should have a .index_name property.
- **output_indexes** (*list(str, ExplainerComponent)*) – list of str of ExplainerComponents.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

HighlightConnector

class explainerdashboard.dashboard_components.connectors.**HighlightConnector**(*input_highlight*, *output_highlights*)

Connect the highlight selector of input_highlight with those of output_highlights.

You can use this to connect a DecisionTreesComponent component to a DecisionPathGraphComponent for example.

When you change the highlight in input_highlight, all the highlights in output_highlights will automatically be updated.

Parameters

- **input_highlight** (*[{str, ExplainerComponent}]*) – Either a str or an ExplainerComponent. If str should be equal to the name of the highlight property. If ExplainerComponent then should have a .highlight_name property.
- **output_highlights** (*list(str, ExplainerComponent)*) – list of str of ExplainerComponents.

component_callbacks(*app*)

register callbacks specific to this ExplainerComponent.

4.8 Customizing your dashboard

The dashboard is highly modular and customizable so that you can adjust it your own needs and project. You can switch off tabs, control the features of invidual components in the dashboard, or even build your entire custom dashboard layout like the example below:

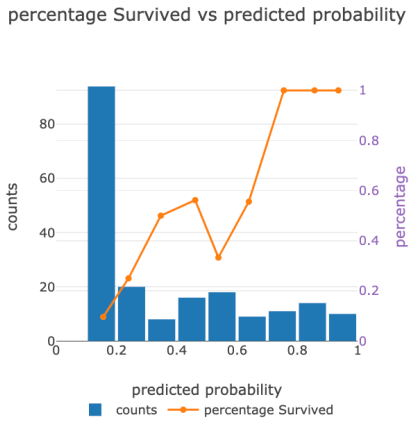
Titanic Explainer

Model | Predictions

Model Performance

As you can see on the right, the model performs quite well.
The higher the predicted probability of survival predicted by the model on the basis of learning from examples in the training set, the higher is the actual percentage for a person surviving in the test set

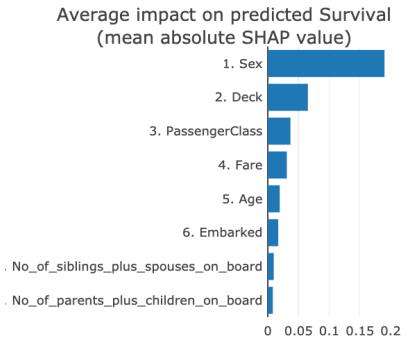
Model Precision Plot



Feature Importances Plot

Summary Type

Aggregate | Detailed



Feature importances

On the left you can check out for yourself which parameters were the most important.
Sex was the most important, followed by Deck and PassengerClass.
If you select 'detailed' you can see the impact of that variable on each individual prediction.
With 'aggregate' you see the average impact size of that variable on the finale prediction.
With the detailed view you can clearly see that the the large impact from Sex stems both from males having a much lower chance of survival and females a much higher chance.

Relations between features and model output

In the plot to the right you can see that the higher the priaceof the Fare that people paid, the higher the chance of survival.
Probably the people with more expensive tickets were in higher up cabins, and were more likely to make it to a lifeboat.
When you color the impacts by the PassengerClass, you can clearly see that the more expensive tickets were mostly 1st class, and the cheaper tickets mostly 3rd class.
On the right you can check out for yourself how different features impact the model output.

Feature impact plot

Feature: Fare | Color feature: Passenger...

4.8.1 Changing bootstrap theme

You can change the bootstrap theme by passing a link to the appropriate css file. You can use the convenient themes module of dash_bootstrap_components to generate the css url for you:

```
import dash_bootstrap_components as dbc

ExplainerDashboard(explainer, bootstrap=dbc.themes.FLATLY).run()
```

See the dbc themes documentation for the different themes that are supported.

4.8.2 Switching off tabs

You can switch off individual tabs using boolean flags, e.g.:

```
ExplainerDashboard(explainer,
    importances=False,
    model_summary=True,
    contributions=True,
    whatif=True,
    shap_dependence=True,
    shap_interaction=False,
    decision_trees=True)
```

4.8.3 Hiding components

You can also hide individual components on the various tabs:

```
ExplainerDashboard(explainer,
    # importances tab:
    hide_importances=True,
    # classification stats tab:
    hide_globalcutoff=True, hide_modelsummary=True,
    hide_confusionmatrix=True, hide_precision=True,
    hide_classification=True, hide_rocauc=True,
    hide_prauc=True, hide_liftcurve=True, hide_cumprecision=True,
    # regression stats tab:
    # hide_modelsummary=True,
    hide_predsvsactual=True, hide_residuals=True,
    hide_regvscol=True,
    # individual predictions tab:
    hide_predindexselector=True, hide_predictionsummary=True,
    hide_contributiongraph=True, hide_pdp=True,
    hide_contributiontable=True,
    # whatif tab:
    hide_whatifindexselector=True, hide_inputeditor=True,
    hide_whatifcontribution=True, hide_whatifpdp=True,
    # shap dependence tab:
    hide_shapsummary=True, hide_shapdependence=True,
    # shap interactions tab:
    hide_interactionsummary=True, hide_interactiondependence=True,
```

(continues on next page)

(continued from previous page)

```
    # decisiontrees tab:
    hide_treeindexselector=True, hide_treesgraph=True,
    hide_treepathtable=True, hide_treepathgraph=True,
    ).run()
```

4.8.4 Hiding toggles and dropdowns inside components

You can also hide individual toggles and dropdowns using ***kwargs*. However they are not individually targeted, so if you pass *hide_cats=True* then the group cats toggle will be hidden on every component that has one:

```
ExplainerDashboard(explainer,
    no_permutations=True, # do not show or calculate permutation importances
    hide_popout=True, # hide the 'popout' button for each graph
    hide_poweredby=True, # hide the 'powerered by: explainerdashboard' footer
    hide_popout=True, # hide the 'popout' button from each graph
    hide_depth=True, # hide the depth (no of features) dropdown
    hide_sort=True, # hide sort type dropdown in contributions graph/table
    hide_orientation=True, # hide orientation dropdown in contributions graph/table
    hide_type=True, # hide shap/permutation toggle on ImportancesComponent
    hide_dropna=True, # hide dropna toggle on pdp component
    hide_sample=True, # hide sample size input on pdp component
    hide_gridlines=True, # hide gridlines on pdp component
    hide_gridpoints=True, # hide gridpoints input on pdp component
    hide_cutoff=True, # hide cutoff selector on classification components
    hide_percentage=True, # hide percentage toggle on classificaiton components
    hide_log_x=True, # hide x-axis logs toggle on regression plots
    hide_log_y=True, # hide y-axis logs toggle on regression plots
    hide_ratio=True, # hide the residuals type dropdown
    hide_points=True, # hide the show violin scatter markers toggle
    hide_winsor=True, # hide the winsorize input
    hide_wizard=True, # hide the wizard from the lift curve
)
```

4.8.5 Setting default values

You can also set default values for the various dropdowns and toggles. All the components with their parameters can be found *in the componentsdocumentation*. Some examples of useful parameters to pass:

```
ExplainerDashboard(explainer,
    index='Rugg, Miss. Emily', # initial index to display
    col='Fare', # initial feature in shap graphs
    color_col='Age', # color feature in shap dependence graph
    interact_col='Age', # interaction feature in shap interaction
    higher_is_better=False, # flip green and red in contributions graph
    depth=5, # only show top 5 features
    sort = 'low-to-high', # sort features from lowest shap to highest in contributions␣
↪graph/table
    orientation='horizontal', # horizontal bars in contributions graph
    cats_topx = 3, # show only the top 3 categories
    cats_sort = 'shap', # sort categories by mean abs shap instead of 'freq' or 'alphabet'
```

(continues on next page)

(continued from previous page)

```
    pdp_col='Fare', # initial pdp feature
    cutoff=0.8, # cutoff for classification plots
    round=2 # round floats to 2 digits
    show_metrics=['accuracy', 'f1', custom_metric] # only show certain metrics
    plot_sample=1000, # only display a 1000 random markers in scatter plots
    )
```

4.8.6 Using custom metrics

By default the dashboard shows a number of metrics for classifiers (accuracy, etc) and regression models (R-squared, etc). You can control which metrics are shown and in what order by passing `show_metrics`:

```
ExplainerDashboard(explainer, show_metrics=['accuracy', 'f1', 'recall']).run()
```

However you can also define custom metrics functions yourself as long as they take `y_true` and `y_pred` as parameters:

```
def custom_metric(y_true, y_pred):
    return np.mean(y_true)-np.mean(y_pred)

ExplainerDashboard(explainer, show_metrics=['accuracy', custom_metric]).run()
```

For `ClassifierExplainer`, `y_true` and `y_pred` will have already been calculated as an array of `1` and `0` depending on the `pos_label` and `cutoff` that was passed to `explainer.metrics()`. However, if you take `pos_label` and `cutoff` as parameters to the custom metric function, then you will get the unprocessed raw labels and *pred_probas*. So for example you could calculate a sum of cost function over the confusion matrix as a custom metric. Then the following metrics would all work and have the equivalent result:

```
from sklearn.metrics import confusion_matrix

def cost_metric(y_true, y_pred):
    cost_matrix = np.array([[10, -50], [-20, 10]])
    cm = confusion_matrix(y_true, y_pred)
    return (cost_matrix * cm).sum()

def cost_metric2(y_true, y_pred, cutoff):
    return cost_metric(y_true, np.where(y_pred>cutoff, 1, 0))

def cost_metric3(y_true, y_pred, pos_label):
    return cost_metric(np.where(y_true==pos_label, 1, 0), y_pred[:, pos_label])

def cost_metric4(y_true, y_pred, cutoff, pos_label):
    return cost_metric(np.where(y_true==pos_label, 1, 0),
                       np.where(y_pred[:, pos_label] > cutoff, 1, 0))

explainer.metrics(show_metrics=[cost_metric, cost_metric2, cost_metric3, cost_metric4]).
↪run()
```

Note: When storing an `ExplainerDashboard.to_yaml()` the custom metric functions will be stored to the `.yaml` file with a reference to their name and module. So when loading the dashboard `from_config()` you have to make sure the metric function can be found by the same name in the same module (which could be `__main__`), otherwise

the dashboard will fail to load.

4.9 Building custom layout

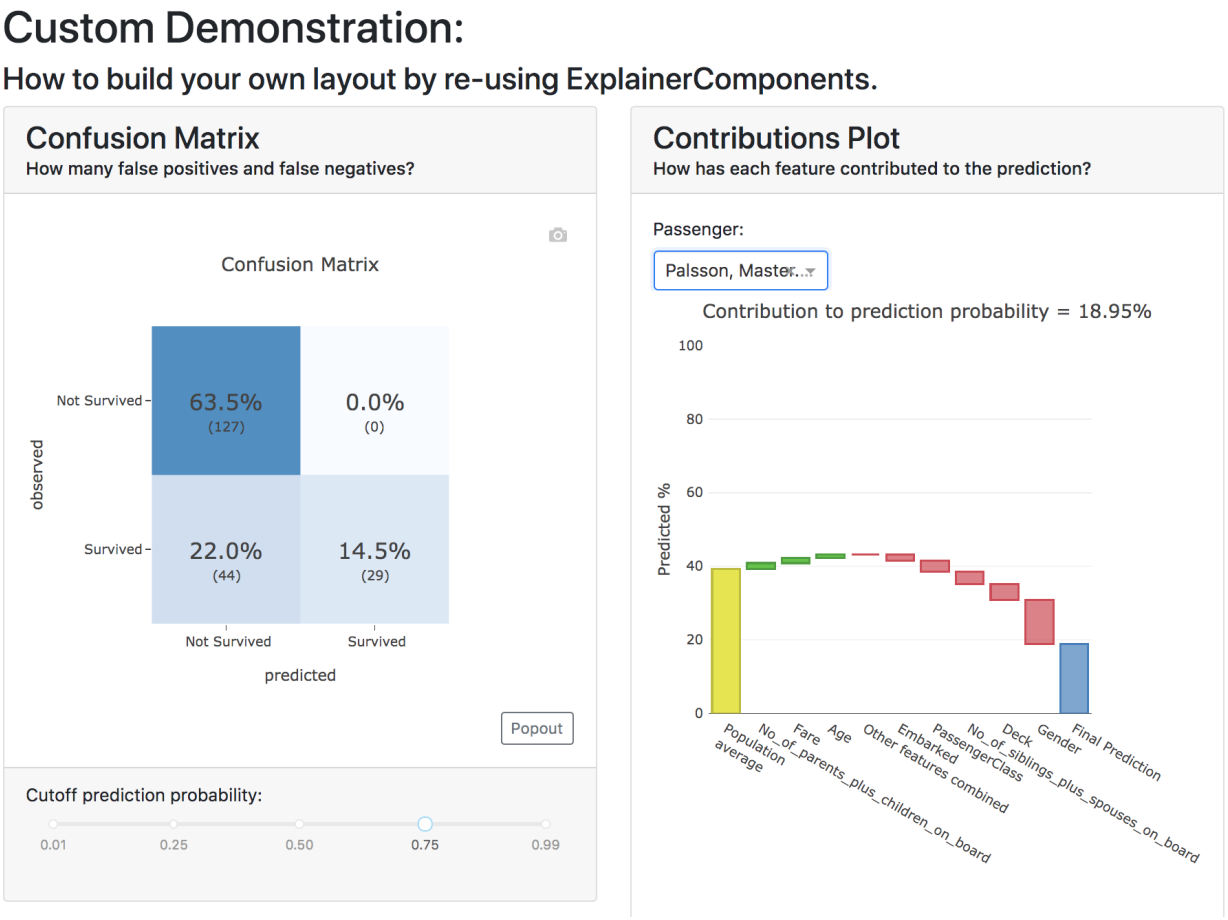
You can build your own custom dashboard layout by re-using the modular *ExplainerComponents and connectors* without needing to know much about web development or even much about plotly dash, which is the underlying technology that `explainerdashboard` is built on.

You can get some inspiration from the explainerdashboard composites that build the layout of the default dashboard tabs. You can copy that code move some of the components around and add some text to make it specific to your own project.

4.9.1 Simple Example

For example if you only wanted to build a custom dashboard that only contains a `ConfusionMatrixComponent` and a `ShapContributionsGraphComponent`, but you want to hide a few toggles:

```
from explainerdashboard.custom import *

class CustomDashboard(ExplainerComponent):
    def __init__(self, explainer, name=None, **kwargs):
        super().__init__(explainer, title="Custom Dashboard")
        self.confusion = ConfusionMatrixComponent(explainer,
                            hide_selector=True, hide_percentage=True,
                            cutoff=0.75, **kwargs)
        self.contrib = ShapContributionsGraphComponent(explainer,
                            hide_selector=True, hide_cats=True,
                            hide_depth=True, hide_sort=True,
                            index='Rugg, Miss. Emily', **kwargs)

    def layout(self):
        return dbc.Container([
            dbc.Row([
                dbc.Col([
                    html.H1("Custom Demonstration:"),
                    html.H3("How to build your own layout using ExplainerComponents.")
                ])
            ]),
            dbc.Row([
                dbc.Col([
                    self.confusion.layout(),
                ]),
                dbc.Col([
                    self.contrib.layout(),
                ])
            ])
        ])

db = ExplainerDashboard(explainer, CustomDashboard, hide_header=True)
db.run()
```



So you need to

1. Import `ExplainerComponents` with `from explainerdashboard.custom import *`. (this also imports `dash_html_components as html`, `dash_core_components as dcc` and `dash_bootstrap_components as dbc` for you.
2. Derive a child class from `ExplainerComponent`.
3. Include `explainer, name=None` in your `__init__()`.
4. Call the init of the parent class with `super().__init__(explainer, title)`.
5. Instantiate the components that you wish to include as attributes in your `__init__`: `self.confusion = ConfusionMatrixComponent(explainer)` and `self.contrib = ShapContributionsGraphComponent(explainer)`
6. Define a `layout()` method that returns a custom layout.
7. Build your layout using `html` and bootstrap (`dbc`) elements and include your components' layout in this overall layout with `self.confusion.layout()` and `self.contrib.layout()`.
8. Pass the class to an `ExplainerDashboard` and `run()` it.

You can find the list of all `ExplainerComponents` in the *documentation*.

Note: To save on boilerplate code, parameters in the `__init__` will automagically be stored to attributes by `super().__init__(explainer, title)`. So in the example below you do not have to explicitly call `self.a = a` in the init:

```
class CustomDashboard(ExplainerComponent):
    def __init__(self, explainer, name=None, a=1):
        super().__init__(explainer)

custom = CustomDashboard(explainer)
assert custom.a == 1
```

This includes the naming of the component itself, by setting `name=None`, in the `__init__`. `ExplainerDashboard` will then assign a unique name of your component to make sure that component *id*'s will not clash, but will be consistent with multi worker or multi node deployments.

4.9.2 Including ExplainerComponents in regular `dash` app

An `ExplainerComponent` can easily be included in regular dash code:

```
import dash

custom = CustomDashboard(explainer)

app = dash.Dash(__name__)
app.title = "Dash demo"
app.layout = html.Div([
    custom.layout()
    ])
custom.register_callbacks(app)
app.run_server()
```

4.9.3 Constructing the layout

You construct the layout using `dash_bootstrap_components` and `dash_html_components`:

dash_bootstrap_components

Using the `dash_bootstrap_components` library it is very easy to construct a modern looking responsive web interface with just a few lines of python code.

The basis of any layout is that you divide your layout into `dbc.Rows` and then divide each row into a number of `dbc.Cols` where the total column widths should add up to 12. (e.g. two columns of width 6 each)

Then `dash_bootstrap_components` offer a lot of other modern web design elements such as cards, modals, etc that you can find more information on in their documentation: https://dash-bootstrap-components.opensource.faculty.ai/

dash_html_components

If you know a little bit of html then using `import dash_html_components as html` you can add further elements to your design. For example in order to insert a header add `html.H1("This is my header!")`, etc.

4.9.4 Elaborate Example

CustomModelTab

A more elaborate example is below where we include three components: the precision graph, the shap summary and the shap dependence component, and add explanatory text on either side of each component. The `ShapSummaryDependenceConnector` connects a ShapSummaryComponent and a ShapDependenceComponent so that when you select a feature in the summary, it automatically gets selected in the dependence plot. You can find other connectors such *IndexConnector*, *PosLabelConnector*, *CutoffConnector* and *HighlightConnector* in the *Connector documentation*:

```
import dash_html_components as html
import dash_bootstrap_components as dbc

from explainerdashboard.custom import *
from explainerdashboard import ExplainerDashboard

class CustomModelTab(ExplainerComponent):
    def __init__(self, explainer, name=None):
        super().__init__(explainer, title="Titanic Explainer")
        self.precision = PrecisionComponent(explainer,
                                hide_cutoff=True, hide_binsize=True,
                                hide_binmethod=True, hide_multiclass=True,
                                hide_selector=True,
                                cutoff=None)
        self.shap_summary = ShapSummaryComponent(explainer,
                                hide_title=True, hide_selector=True,
                                hide_depth=True, depth=8,
                                hide_cats=True, cats=True)
        self.shap_dependence = ShapDependenceComponent(explainer,
                                hide_title=True, hide_selector=True,
                                hide_cats=True, cats=True,
                                hide_index=True,
                                col='Fare', color_col="PassengerClass")
        self.connector = ShapSummaryDependenceConnector(
                self.shap_summary, self.shap_dependence)

    def layout(self):
        return dbc.Container([
            html.H1("Titanic Explainer"),
            dbc.Row([
                dbc.Col([
                    html.H3("Model Performance"),
                    html.Div("As you can see on the right, the model performs quite well.
↪"),
                    html.Div("The higher the predicted probability of survival predicted␣
↪by"
```

(continues on next page)

(continued from previous page)

```
                            "the model on the basis of learning from examples in the␣
↪training set"
                            ", the higher is the actual percentage for a person␣
↪surviving in "
                            "the test set"),
                ], width=4),
                dbc.Col([
                    html.H3("Model Precision Plot"),
                    self.precision.layout()
                ])
            ]),
            dbc.Row([
                dbc.Col([
                    html.H3("Feature Importances Plot"),
                    self.shap_summary.layout()
                ]),
                dbc.Col([
                    html.H3("Feature importances"),
                    html.Div("On the left you can check out for yourself which␣
↪parameters were the most important."),
                    html.Div(f"{self.explainer.columns_ranked_by_shap(cats=True)[0]} was␣
↪the most important"
                            f", followed by {self.explainer.columns_ranked_by_
↪shap(cats=True)[1]}"
                            f" and {self.explainer.columns_ranked_by_shap(cats=True)[2]}.
↪"),
                    html.Div("If you select 'detailed' you can see the impact of that␣
↪variable on "
                            "each individual prediction. With 'aggregate' you see the␣
↪average impact size "
                            "of that variable on the finale prediction."),
                    html.Div("With the detailed view you can clearly see that the the␣
↪large impact from Sex "
                            "stems both from males having a much lower chance of␣
↪survival and females a much "
                            "higher chance.")
                ], width=4)
            ]),
            dbc.Row([
                dbc.Col([
                    html.H3("Relations between features and model output"),
                    html.Div("In the plot to the right you can see that the higher the␣
↪priace"
                            "of the Fare that people paid, the higher the chance of␣
↪survival. "
                            "Probably the people with more expensive tickets were in␣
↪higher up cabins, "
                            "and were more likely to make it to a lifeboat."),
                    html.Div("When you color the impacts by the PassengerClass, you can␣
↪clearly see that "
                            "the more expensive tickets were mostly 1st class, and the␣
↪cheaper tickets "
```

(continues on next page)

(continued from previous page)

```
                            "mostly 3rd class."),
                    html.Div("On the right you can check out for yourself how different␣
↪features impact "
                            "the model output."),
                ], width=4),
                dbc.Col([
                    html.H3("Feature impact plot"),
                    self.shap_dependence.layout()
                ]),
            ])
        ])

ExplainerDashboard(explainer, CustomModelTab, hide_header=True).run()
```

Note: All subcomponents that are defined as attibutes in the `__init__`, either explicitly or automagically through the `super().__init__`, and hence are added to `self.__dict__` also automatically get their callbacks registered when you call `.register_callbacks(app)` on the parent component. If you would like to exclude that (for example because the subcomponent has already been initialized elsewhere and you just need to store the reference), then you can exclude it with `exclude_callbacks(components)`:

```
class CustomDashboard(ExplainerComponent):
    def __init__(self, explainer, name=None, feature_input_component):
        super().__init__(explainer)
        self.exclude_callbacks(self.feature_input_component)
```

CustomPredictionsTab

We can also add another tab to investigate individual predictions, that includes an index selector, a SHAP contributions graph and a Random Forest individual trees graph. The `IndexConnector` connects the index selected in `ClassifierRandomIndexComponent` with the index dropdown in the contributions graph and trees components. We also pass a custom dbc theme called FLATLY as a custom css file:

```
class CustomPredictionsTab(ExplainerComponent):
    def __init__(self, explainer, name=None):
        super().__init__(explainer, title="Predictions")

        self.index = ClassifierRandomIndexComponent(explainer,
                                                    hide_title=True, hide_index=False,
                                                    hide_slider=True, hide_labels=True,
                                                    hide_pred_or_perc=True,
                                                    hide_selector=True, hide_
↪button=False)

        self.contributions = ShapContributionsGraphComponent(explainer,
                                                             hide_title=True, hide_
↪index=True,
                                                             hide_depth=True, hide_
↪sort=True,
                                                             hide_orientation=True, hide_
```

(continues on next page)

(continued from previous page)

```
↪cats=True,
                                                        hide_selector=True,
                                                        sort='importance')

        self.trees = DecisionTreesComponent(explainer,
                                            hide_title=True, hide_index=True,
                                            hide_highlight=True, hide_selector=True)

        self.connector = IndexConnector(self.index, [self.contributions, self.trees])

    def layout(self):
        return dbc.Container([
            dbc.Row([
                dbc.Col([
                    html.H3("Enter name:"),
                    self.index.layout()
                ])
            ]),
            dbc.Row([
                dbc.Col([
                    html.H3("Contributions to prediction:"),
                    self.contributions.layout()
                ]),

            ]),
            dbc.Row([
                dbc.Col([
                    html.H3("Every tree in the Random Forest:"),
                    self.trees.layout()
                ]),
            ])
        ])

ExplainerDashboard(explainer, [CustomModelTab, CustomPredictionsTab],
               title='Titanic Explainer',
               header_hide_selector=True,
               bootstrap=dbc.themes.FLATLY).run()
```

Below you can see the result. (also note how the component title shows up as the tab title). This dashboard has also been deployed at http://titanicexplainer.herokuapp.com/custom:

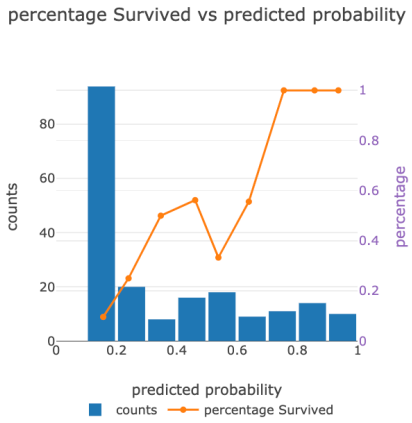
Titanic Explainer

Model | Predictions

Model Performance

As you can see on the right, the model performs quite well.
The higher the predicted probability of survival predicted by the model on the basis of learning from examples in the training set, the higher is the actual percentage for a person surviving in the test set

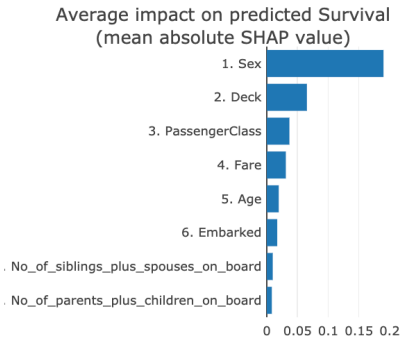
Model Precision Plot



Feature Importances Plot

Summary Type

Aggregate | Detailed



Feature importances

On the left you can check out for yourself which parameters were the most important.
Sex was the most important, followed by Deck and PassengerClass.
If you select 'detailed' you can see the impact of that variable on each individual prediction.
With 'aggregate' you see the average impact size of that variable on the finale prediction.
With the detailed view you can clearly see that the the large impact from Sex stems both from males having a much lower chance of survival and females a much higher chance.

Relations between features and model output

In the plot to the right you can see that the higher the priaceof the Fare that people paid, the higher the chance of survival.
Probably the people with more expensive tickets were in higher up cabins, and were more likely to make it to a lifeboat.
When you color the impacts by the PassengerClass, you can clearly see that the more expensive tickets were mostly 1st class, and the cheaper tickets mostly 3rd class.
On the right you can check out for yourself how different features impact the model output.

Feature impact plot

Feature: Fare | Color feature: Passenger...

4.9.5 Comparing multiple models

You can also compare multiple models in the same dashboard, or inside the same tab. In this case it is important to already instantiate the component/tab before passing it on to the `ExplainerDashboard`:

```
from explainerdashboard import *
from explainerdashboard.datasets import *
from explainerdashboard.custom import *

from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

X_train, y_train, X_test, y_test = titanic_survive()

model1 = RandomForestClassifier(n_estimators=50, max_depth=4).fit(X_train, y_train)
model2 = XGBClassifier(n_estimators=10, max_depth=5).fit(X_train, y_train)

explainer1 = ClassifierExplainer(model1, X_test, y_test)
explainer2 = ClassifierExplainer(model2, X_test, y_test)

class ConfusionComparison(ExplainerComponent):
    def __init__(self, explainer1, explainer2):
        super().__init__(explainer1)
        self.confmat1 = ConfusionMatrixComponent(explainer1, cutoff=0.6,
                            hide_selector=True, hide_percentage=True)
        self.confmat2 = ConfusionMatrixComponent(explainer2, cutoff=0.6,
                            hide_selector=True, hide_percentage=True)

    def layout(self):
        return dbc.Container([
            dbc.Row([
                dbc.Col([
                    self.confmat1.layout()
                ]),
                dbc.Col([
                    self.confmat2.layout()
                ])
            ])
        ])

tab = ConfusionComparison(explainer1, explainer2)

ExplainerDashboard(explainer1, tab).run()
```

4.9.6 Custom static html export

To enable your custom dashboard to be exported to html you need to define a `to_html` method that returns an html layout. There are helper functions in `explainerdashboard.to_html` to help you construct this html using python code, which get automatically loaded when you `from explainerdashboard.custom import *`:

```
from explainerdashboard.custom import *

class CustomDashboard(ExplainerComponent):
    def __init__(self, explainer, name=None, **kwargs):
        super().__init__(explainer, title="Custom Dashboard")
        self.confusion = ConfusionMatrixComponent(explainer, **kwargs)
        self.contrib = ShapContributionsGraphComponent(explainer, **kwargs)

    def layout(self):
        return dbc.Container([
            dbc.Row([
                dbc.Col([
                    html.H1("Custom Demonstration:"),
                    html.H3("How to build your own layout using ExplainerComponents.")
                ])
            ]),
            dbc.Row([
                dbc.Col([
                    self.confusion.layout(),
                ]),
                dbc.Col([
                    self.contrib.layout(),
                ])
            ])
        ])

    def to_html(self, state_dict=None, add_header=True):
        html = to_html.title(self.title)
        html += to_html.card_row(
            self.confusion.to_html(state_dict, add_header=False),
            self.contrib.to_html(state_dict, add_header=False)
        )
        if add_header:
            return to_html.add_header(html)
        return html
```

So the arguments of `to_html()` should be `(self, state_dict=None, add_header=True)`, and you should pass `(state_dict, add_header=False)` to all sub-components. Only the top-level component should add a html header (which links to bootstrap css and javascript and triggers a window resize in order to fix plotly figure overflow issues).

If you build your own components whose display depend on the state of various toggles that you put in the dash layout, you need to define input component id's as a `_state_props` class attribute. This should be a dict with the keys equal to `__init__` parameters, and the values a tuple identifying the dash component id and property. The id's should be without *+self.name* as this will get automatically added behind the scenes. (and since it's a class attribute it doesn't have access to `self` in any case).

You can then call `args = self.get_state_args(state_dict)` to get the current values of the parameters inside your `to_html` method. When you call the method directly from the component (e.g. inside a jupyter notebook), it will gather the arguments from the instance properties (so wil e.g. return `{'cutoff': self.cutoff}`), so it will

output html based on the initial values. When clicking the download button on a running dashboard, it will substitute the current values of the dashboard state. These values get collected in `state_dict`, which is why it is important to pass `state_dict` down to subcomponents.

An example is the `ConfusionMatrixComponent`:

```
class ConfusionMatrixComponent(ExplainerComponent):
    _state_props = dict(
        cutoff=('confusionmatrix-cutoff-', 'value'),
        percentage=('confusionmatrix-percentage-', 'value'),
        normalize=('confusionmatrix-normalize-', 'value'),
        binary=('confusionmatrix-binary-', 'value'),
        pos_label=('pos-label-', 'value')
    )

    def __init__(self, explainer, title="Confusion Matrix", name=None,
                    subtitle="How many false positives and false negatives?",
                    hide_title=False, hide_subtitle=False, hide_footer=False,
                    hide_cutoff=False, hide_percentage=False, hide_binary=False,
                    hide_selector=False, hide_popout=False, hide_normalize=False,
                    normalize='all', pos_label=None, cutoff=0.5,
                    percentage=True, binary=True, description=None,
                    **kwargs):
        pass # removed the init and layout function here for brevity


    def to_html(self, state_dict=None, add_header=True):
        args = self.get_state_args(state_dict)
        fig = self.explainer.plot_confusion_matrix(cutoff=args['cutoff'],
                    percentage=bool(args['percentage']), binary=bool(args['binary']),
                    normalize=args['normalize'], pos_label=args['pos_label'])

        html = to_html.card(to_html.fig(fig), title=self.title, subtitle=self.subtitle)
        if add_header:
            return to_html.add_header(html)
        return html

    def component_callbacks(self, app):
        @app.callback(
            Output('confusionmatrix-graph-'+self.name, 'figure'),
            [Input('confusionmatrix-cutoff-'+self.name, 'value'),
            Input('confusionmatrix-percentage-'+self.name, 'value'),
            Input('confusionmatrix-normalize-'+self.name, 'value')],
            Input('confusionmatrix-binary-'+self.name, 'value'),
            Input('pos-label-'+self.name, 'value'),
        )
        def update_confusionmatrix_graph(cutoff, percentage, normalize, binary, pos_
↪label):
            return self.explainer.plot_confusion_matrix(
                        cutoff=cutoff, percentage=bool(percentage), normalize=normalize,
                        binary=bool(binary), pos_label=pos_label)
```

When exporting to html you probably want to set a default `index` value for your `ExplainerDashboard`. This is the index that will be displayed in the plots with individual rows of data when you call `ExplainerDashboard.to_html()` directly, e.g. `ExplainerDashboard(explainer, index=0).to_html('dashboard.html')`.

4.9.7 `to_html` helper functions

The `explainerdashboard.to_html` module contains a number of useful functions that you can use inside your custom `to_html()` methods:

Helper module to define static html outputs

`explainerdashboard.to_html.`**`add_header`**(*html*, *title='explainerdashboard'*, *resize=True*)

Turns a html snippet into a full html layout by adding <html>, <head> and <body> tags.

Loads bootstrap css and javascript and triggers a resize event in order to prevent plotly figs from overflowing their div containers.

resize adds a javascript snippet that simulates a window resize in order to properly size the plotly figs. (bit of a hack, but it works :)

Return type
`str`

`explainerdashboard.to_html.`**`row`**(**cols*)

Turns a series of html snippets into a bootstrap row with equally sized columns for each snippet.

Return type
`str`

Example

to_html.row(" first snippet ", " second snippet ")

`explainerdashboard.to_html.`**`rows`**(**col_lists*)

Turns a list of lists of html snippets into a series of bootstrap rows with equally sized columns for each snippet.

Return type
`str`

Example

to_html.row(
[" first snippet ", " second snippet "], [" second row snippet snippet ", " second row snippet two "]

)

`explainerdashboard.to_html.`**`fig`**(*fig*, *include_plotlyjs='cdn'*, *full_html=False*)

Returns html for a plotly figure. By default the plotly javascript is not included but imported from the plotly cdn, and the full html wrapper is not included.

Parameters

- **`include_plotlyjs`** (*`bool, str`*) – how to import the necessary javascript for the plotly fig. Defaults to 'cdn', which means the figure just links to javascript file hosted by plotly. If set to True then a 3MB javascript snippet is included. For other options check https://plotly.com/python-api-reference/generated/plotly.io.to_html.html
- **`full_html`** (*`bool`*) – include <html>, <head> and <body> tags. Defaults to False.

Return type
`str`

explainerdashboard.to_html.**card**(*html*, *title=None*, *subtitle=None*, *border=True*)

Wrap to html snippet in a bootstrap card. You can optionally add a title and subtitle to the card.

Return type
str

explainerdashboard.to_html.**dashboard_card**(*title=None*, *description=None*, *url=None*)

Generate a dashboard description card for ExplainerHub. Consists of title, description and url.

Return type
str

explainerdashboard.to_html.**card_row**(**cards*)

Turns a series of bootstrap into a row with equally sized columns for each card.

Return type
str

Example

to_html.card_row(' first card ", ' second snippet ')

explainerdashboard.to_html.**card_rows**(**cardrows_list*)

Turn a list of lists of bootstrap cards into a series of bootstrap rows with cards.

Return type
str

Example

to_html.card_rows(
[to_html.card("card1"), to_html.card("card2")], [to_html.card("card3"), to_html.card("card4")],

)

explainerdashboard.to_html.**title**(*title*)

wrap a title string in div and <H1></H1>

Return type
str

explainerdashboard.to_html.**div**(*html*)

wrap an html snippet in a

Return type
str

explainerdashboard.to_html.**table_from_df**(*df*)

Generate a html table from a pandas DataFrame

Return type
str

explainerdashboard.to_html.**hide**(*html*, *hide=False*)

optionally hide an html snippet (return empty div) if parameter hide=True

Return type
str

explainerdashboard.to_html.**tabs**(*tabs_dict*)

Generate a series of bootstrap tabs for a dictionary tabs_dict with the name of each tab as the dict key and the html contents of the tab as the dict value.

Return type
str

explainerdashboard.to_html.**input**(*feature*, *value*, *disabled=False*)

Return a html feature input with a feature name and default value.

Parameters

- **feature** (*str*) – name of feature
- **value** (*str*) – default value
- **disabled** (*bool*) – disable the input. Defaults to False.

Return type
str

explainerdashboard.to_html.**jumbotron**(*title*, *description*)

display a bootstrap jumbotron with title and description

Return type
str

4.10 Deployment

When deploying your dashboard it is better not to use the built-in flask development server but use a more robust production server like `gunicorn` or `waitress`. Probably gunicorn is a bit more fully featured and faster but only works on unix/linux/osx, whereas waitress also works on Windows and has very minimal dependencies.

Install with either `pip install gunicorn` or `pip install waitress`.

4.10.1 Storing explainer and running default dashboard with gunicorn

Before you start a dashboard with gunicorn you need to store both the explainer instance and and a configuration for the dashboard:

```
from explainerdashboard import ClassifierExplainer, ExplainerDashboard

explainer = ClassifierExplainer(model, X, y)
db = ExplainerDashboard(explainer, title="Cool Title", shap_interaction=False)
db.to_yaml("dashboard.yaml", explainerfile="explainer.joblib", dump_explainer=True)
```

Now you re-load your dashboard and expose a flask server as `app` in `dashboard.py`:

```
from explainerdashboard import ExplainerDashboard

db = ExplainerDashboard.from_config("dashboard.yaml")
app = db.flask_server()
```

If you named the file above `dashboard.py`, you can now start the gunicorn server with:

```
$ gunicorn dashboard:app
```

If you want to run the server server with for example three workers, binding to port `8050` you launch gunicorn with:

```
$ gunicorn -w 3 -b localhost:8050 dashboard:app
```

If you now point your browser to `http://localhost:8050` you should see your dashboard. Next step is finding a nice url in your organization's domain, and forwarding it to your dashboard server.

With waitress you would call:

```
$ waitress-serve --port=8050 dashboard:app
```

Although you can all use the `waitress` directly from the dashboard by passing the `use_waitress=True` flag to `.run()`:

```
ExplainerDashboard(explainer).run(use_waitress=True)
```

4.10.2 Deploying dashboard as part of Flask app on specific route

Another way to deploy the dashboard is to first start a `Flask` app, and then use this app as the backend of the Dashboard, and host the dashboard on a specific route. This way you can for example host multiple dashboard under different urls. You need to pass the Flask `server` instance and the `url_base_pathname` to the `ExplainerDashboard` constructor, and then the dashboard itself can be found under `db.app.index`:

```
from flask import Flask

app = Flask(__name__)

[...]

db = ExplainerDashboard(explainer, server=app, url_base_pathname="/dashboard/")

@app.route('/dashboard')
def return_dashboard():
    return db.app.index()
```

Now you can start the dashboard by:

```
$ gunicorn -b localhost:8050 dashboard:app
```

And you can visit the dashboard on `http://localhost:8050/dashboard`.

4.10.3 Deploying to heroku

In case you would like to deploy to heroku (which is normally the simplest option for dash apps, see dash instructions here). The demonstration dashboard is also hosted on heroku at titanicexplainer.herokuapp.com.

In order to deploy the heroku there are a few things to keep in mind. First of all you need to add `explainerdashboard` and `gunicorn` to `requirements.txt` (pinning is recommended to force a new build of your environment whenever you upgrade versions):

```
explainerdashboard==0.3.1
gunicorn
```

Select a python runtime compatible with the version that you used to pickle your explainer in `runtime.txt`:

```
python-3.8.6
```

(supported versions as of this writing are `python-3.9.0`, `python-3.8.6`, `python-3.7.9` and `python-3.6.12`, but check the heroku documentation for the latest)

And you need to tell heroku how to start your server in `Procfile`:

```
web: gunicorn dashboard:app
```

Graphviz buildpack

If you want to visualize individual trees inside your `RandomForest` or `xgboost` model using the `dtreeviz` package you will need to make sure that `graphviz` is installed on your `heroku` dyno by adding the following buildstack (as well as the `python` buildpack): `https://github.com/weibeld/heroku-buildpack-graphviz.git`

(you can add buildpacks through the "settings" page of your heroku project)

4.10.4 Docker deployment

You can also deploy a dashboard using docker. You can build the dashboard and store it inside the container to make sure it is compatible with the container environment. E.g. **generate_dashboard.py**:

```
from sklearn.ensemble import RandomForestClassifier

from explainerdashboard import *
from explainerdashboard.datasets import *

X_train, y_train, X_test, y_test = titanic_survive()
model = RandomForestClassifier(n_estimators=50, max_depth=5).fit(X_train, y_train)

explainer = ClassifierExplainer(model, X_test, y_test,
                                cats=["Sex", 'Deck', 'Embarked'],
                                labels=['Not Survived', 'Survived'],
                                descriptions=feature_descriptions)

db = ExplainerDashboard(explainer)
db.to_yaml("dashboard.yaml", explainerfile="explainer.joblib", dump_explainer=True)
```

run_dashboard.py:

```
from explainerdashboard import ExplainerDashboard

db = ExplainerDashboard.from_config("dashboard.yaml")
db.run(host='0.0.0.0', port=9050, use_waitress=True)
```

Dockerfile:

```
FROM python:3.8

RUN pip install explainerdashboard

COPY generate_dashboard.py ./
COPY run_dashboard.py ./

RUN python generate_dashboard.py

EXPOSE 9050
CMD ["python", "./run_dashboard.py"]
```

And build and run the container exposing port `9050`:

```
$ docker build -t explainerdashboard .
$ docker run -p 9050:9050 explainerdashboard
```

4.10.5 Reducing memory usage

If you deploy the dashboard with a large dataset with a large number of rows (`n`) and a large number of columns (`m`), it can use up quite a bit of memory: the dataset itself, shap values, shap interaction values and any other calculated properties are alle kept in memory in order to make the dashboard responsive. You can check the (approximate) memory usage with `explainer.memory_usage()`. In order to reduce the memory footprint there are a number of things you can do:

1. **Not including shap interaction tab.**
   Shap interaction values are shape `n*m*m`, so can take a subtantial amount of memory, especially if you have a significant amount of columns `m`.
2. **Setting a lower precision.**
   By default shap values are stored as `'float64'`, but you can store them as `'float32'` instead and save half the space: `` `ClassifierExplainer(model, X_test, y_test, precision='float32')` ``. You can also set a lower precision on your `X_test` dataset yourself ofcourse.
3. **Drop non-positive class shap values.**
   For multi class classifiers, by default `ClassifierExplainer` calculates shap values for all classes. If you are only interested in a single class you can drop the other shap values with `explainer.keep_shap_pos_label_only(pos_label)`
4. **Storing row data externally and loading on the fly.**
   You can for example only store a subset of `10.000` rows in the `explainer` itself (enough to generate representative importance and dependence plots), and store the rest of your millions of rows of input data in an external file or database that get loaded one by one with the following functions:
   - with `explainer.set_X_row_func()` you can set a function that takes an *index* as argument and returns a single row dataframe with model compatible input data for that index. This function can include a query to a database or fileread.
   - with `explainer.set_y_func()` you can set a function that takes and *index* as argument and returns the observed outcome `y` for that index.
   - with `explainer.set_index_list_func()` you can set a function that returns a list of available indexes that can be queried.

   If the number of indexes is too long to fit in a dropdown you can pass `index_dropdown=False` which

turns the dropdowns into free text fields. Instead of an `index_list_func` you can also set an `explainer.set_index_check_func(func)` which should return a bool whether the `index` exists or not.

Important: these function can be called multiple times by multiple independent components, so probably best to implement some kind of caching functionality. The functions you pass can be also methods, so you have access to all of the internals of the explainer.

4.10.6 Setting logins and password

`ExplainerDashboard` supports dash basic auth functionality. `ExplainerHub` uses `flask_simple_login` for its user authentication.

You can simply add a list of logins to the `ExplainerDashboard` to force a login and prevent random users from accessing the details of your model dashboard:

```
ExplainerDashboard(explainer, logins=[['login1', 'password1'], ['login2', 'password2']]).
↪run()
```

Whereas *ExplainerHub* has somewhat more intricate user management using `FlaskLogin`, but the basic syntax is the same. See the *ExplainerHub documetation* for more details:

```
hub = ExplainerHub([db1, db2], logins=[['login1', 'password1'], ['login2', 'password2']])
```

Make sure not to check these login/password pairs into version control though, but store them somewhere safe! `ExplainerHub` stores passwords into a hashed format by default.

4.10.7 Automatically restart gunicorn server upon changes

We can use the `explainerdashboard` CLI tools to automatically rebuild our explainer whenever there is a change to the underlying model, dataset or explainer configuration. And we we can use `kill -HUP gunicorn.pid` to force the gunicorn to restart and reload whenever a new `explainer.joblib` is generated or the dashboard configuration `dashboard.yaml` changes. These two processes together ensure that the dashboard automatically updates whenever there are underlying changes.

First we store the explainer config in `explainer.yaml` and the dashboard config in `dashboard.yaml`. We also indicate which modelfiles and datafiles the explainer depends on, and which columns in the datafile should be used as a target and which as index:

```
explainer = ClassifierExplainer(model, X, y, labels=['Not Survived', 'Survived'])
explainer.dump("explainer.joblib")
explainer.to_yaml("explainer.yaml",
                  modelfile="model.pkl",
                  datafile="data.csv",
                  index_col="Name",
                  target_col="Survival",
                  explainerfile="explainer.joblib",
                  dashboard_yaml="dashboard.yaml")

db = ExplainerDashboard(explainer, [ShapDependenceTab, ImportancesTab], title="Custom␣
↪Title")
db.to_yaml("dashboard.yaml", explainerfile="explainer.joblib")
```

The `dashboard.py` is the same as before and simply loads an `ExplainerDashboard` directly from the config file:

```
from explainerdashboard import ExplainerDashboard

db = ExplainerDashboard.from_config("dashboard.yaml")
app = db.app.server
```

Now we would like to rebuild the `explainer.joblib` file whenever there is a change to `model.pkl`, `data.csv` or `explainer.yaml` by running `explainerdashboard build`. And we restart the `gunicorn` server whenever there is a change in `explainer.joblib` or `dashboard.yaml` by killing the gunicorn server with `kill -HUP pid` To do that we need to install the python package `watchdog` (`pip install watchdog[watchmedo]`). This package can keep track of filechanges and execute shell-scripts upon file changes.

So we can start the gunicorn server and the two watchdog filechange trackers from a shell script `start_server.sh`:

```
trap "kill 0" EXIT  # ensures that all three process are killed upon exit

source venv/bin/activate # activate virtual environment first

gunicorn --pid gunicorn.pid gunicorn_dashboard:app &
watchmedo shell-command  -p "./model.pkl;./data.csv;./explainer.yaml" -c
↪"explainerdashboard build explainer.yaml" &
watchmedo shell-command -p "./explainer.joblib;./dashboard.yaml" -c 'kill -HUP $(cat␣
↪gunicorn.pid)' &

wait # wait till user hits ctrl-c to exit and kill all three processes
```

Now we can simply run `chmod +x start_server.sh` and `./start_server.sh` to get our server up and running.

Whenever we now make a change to either one of the source files (`model.pkl`, `data.csv` or `explainer.yaml`), this produces a fresh `explainer.joblib`. And whenever there is a change to either `explainer.joblib` or `dashboard.yaml` gunicorns restarts and rebuild the dashboard.

So you can keep an explainerdashboard running without interuption and simply an updated `model.pkl` or a fresh dataset `data.csv` into the directory and the dashboard will automatically update.

4.11 License

4.12 Contact

Question? Please contact oegedijk@gmail.com

4.13 Help

Question? Please contact oegedijk@gmail.com or open an issue on github: https://github.com/oegedijk/explainerdashboard/issues

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

D

E

F

G

H

I

J

L

R

S

T

W