ExpyBox

Jakub Štercl

Jan 10, 2020

CONTENTS:

ExpyBox is a Jupyter toolbox for model interpretability/explainability. It lets you create interactive Jupyter notebooks to explain your model. Because of alibi package ExpyBox requires **64-bit** Python 3.7 or higher.

It is also recommended to create separate virtual enviroment - you can use venv.

Otherwise the installation process is the same as for other packages, just use pip:

```
pip install expybox
```

CHAPTER

ONE

API REFERENCE

class expybox.**ExpyBox**(*train_data*, *predict_function*, *kernel_globals=None*, *categorical_names=None*, *mode='classification'*, *class_names=None*, *feature_names=None*)

Main class to instantiate and hold global data like predict_function, train_data and so on.

Parameters

- **train_data** – numpy array or pandas dataframe of shape (#instances, #features) Data used to explain models (for methods that require data) and to offer instances from (when building instance). Categorical features need to be label_encoded into integers starting from 0. One-hot encoding is not currently supported.
- **predict_function** – Callable Predict function of a model. In case of classification it should return probabilities of classes for each class. **Needs** to be able to deal with numpy array with shape (#instances, #features). If that's not the case for your model, I recommend writing a wrapper function and pass that one.
- **kernel_globals** – dict A dictionary with name_of_variable: variable (NOT its value!) from your kernel. Used to enable passing variable declared in your Jupyter notebook as a value to certain fields. You can either create this dictionary yourself (if you know what you're doing) or just use globals() (this is the recommended usage as it updates when you declare new variable and you don't need to care about it - i.e. just do ... kernel_globals=globals(), ...)
- **categorical_names** – dict A dictionary with index_of_feature: list of the names for categorical features. For example if n-th feature of train_data (train_data[:, n]) is categorical the dict entry should look like this: {n: [name_if_0, name_if_1, name_if_2, ...]} Used for better readability and determining which features are categorical, which improves some methods.
- **mode** – *'classification' or 'regression'*
- **class_names** – list Names for output classes in order in which the predict_function returns probabilities for them. Used just to make input and output more human-friendly
- **feature_names** – list Names for features in order of appearing in train_data. If not filled and train_data is an instance of pandas.DataFrame, train_data.columns is used. Used to make input and output more human-friendly

1.1 Partial Dependence plot

ExpyBox.**pdplot**()
Create dialog for partial dependence plot

Returns None

1.1.1 Method parameters

- **Feature to plot** Which feature should be plotted
- **Number of grid points** Number of grid points to calculate partial dependence value in (for numeric feature)
- **Grid type** Type of grid points for numeric feature (percentile or equal)
- **Custom grid range**
  - **Custom range minimum/maximum** Percentile (when `grid_type = percentile`) or value (when `grid_type = equal`) lower/upper bound of range to investigate (for numeric feature)

    (Enabled only when `Custom grid range` is checked and `Variable with grid points` is `None`)
  - **Variable with grid points** Name of variable (or None) from `kernel_globals` with customized list of grid points (list of integers) for numeric feature
- **Center the plot** Whether the plot should be centered (starting at [0,0])
- **Plot data points distribution** Whether to display data points distribution below the plot
- **Show precentile buckets** Whether to show percentile buckets below the plot
- **Plot lines - ICE plot** Whether to plot also lines for single instances (which makes the plot an ICE plot)
  - **Lines to plot** How many lines to plot, can be a integer or a float
    - integer values higher than 1 are interpreted as absolute amount
    - floats are interpreted as fraction (e.g. 0.5 means half of all possible lines)
  - **Cluster lines** Whether to cluster the lines
    - Number of cluster centers - Number of cluster centers for lines
    - Cluster method - Method to use for clustering of lines (KMeans or MiniBatchKMeans)

1.1.2 Resources

- Documentation (pdp_isolate)
- Documentation (pdp_plot)
- Partial Dependence plot
- Individual Conditional Expectation
- PDPbox on GitHub

1.2 LIME

`ExpyBox.lime()`
Create dialog for lime

Returns None

1.2.1 Method parameters

- **Number of features** Maximum number of features present in explanation
- **Number of samples** Size of the neighborhood to learn the surrogate linear model
- **Kernel width** Kernel width for the exponential kernel. Actual value used will be the `inputted value * sqrt(train_data.shape[1])`.
- **Feature selection** Feature selection method for choosing the best features for surrogate model. There are following options:
  - forward_selection: iteratively add features to the model (costly when num_features is high)
  - highest_weights: selects the features that have the highest product of absolute weight * original data point when learning with all the features
  - lasso_path: choose features based on the lasso regularization path
  - none: use all features, ignore `Number of features` option
  - auto: use forward_selection if `Number of features` <= 6, and highest_weights otherwise
- **Discretize continuous** Whether to discretize all non-categorical features
- **Discretizer** Which discretizer to use when discretizing continuous features. Only matters if discretize continuous is True.
- **Distance metric** What distance metric to use for calculating weights of perturbed instances.

  It's used as an `distance_metric` argument for `sklearn.metrics.pairwise_distances()`. Documentation of the function (with options for metric): sklearn.metrics.pairwise_distances
- **Variable with model regressor** If you want to use a different regressor than Ridge regressor you can specify a variable in provided `kernel_globals` dictionary with the regressor.

  It must have `model_regressor.coef_` and "sample_weight" as a parameter to `model_regressor.fit()`

1.2.2 Resources

- LIME (lime-ml/lime_tabular) documentation,
- LIME in IML book
- LIME paper
- lime-ml package on GitHub

1.3 Anchors

`ExpyBox.anchors()`
Create dialog for Anchors

Returns None

1.3.1 Method parameters

- **threshold** Minimum precision threshold for anchor rules.
- **delta** Used to compute beta. The lower the value, the higher the beta parameter for beam search (wider beam) is. You can see how exactly beta is computed here.
- **tau** Margin between lower confidence bound and minimum precision or upper bound. Here tau is used in place of delta from IML book or the Anchros paper.
- **batch size** Batch size used for sampling
- **discretizer percentiles** Iterable with percentiles (ints) used for discretization of ordinal features. If None, `[25, 50, 75]` will be used. Here you need specify the name of the variable with list of percentiles that is in the kernel_globals parameter that you passed to ExpyBox.

1.3.2 Resources

- Anchors paper
- Anchors (alibi package) documentation
- Alibi API reference
- compute_beta method in alibi (how B parameter for beam search is calculated from delta)

1.4 SHAP

`ExpyBox.shap()`
Create dialog for shap, providing force and decision plots

Returns None

1.4.1 Method parameters

- **Plot** Which plot to draw decision, force or both
- **Background data** What background data will be used to sample from, when simulating "missing" feature
  - KMeans: use KMeans to sample from provided dataset
  - Custom variable: provide variable (that is in `kernel_globals`) with instances to use
- **Count of KMeans centers** Number of means to use when creating background data. Only used if `Background data` is set to `KMeans`
- **Background data variable** Variable with background data from which the "missing" features will be sampled. Only used if `Background data` is set to `Custom variable`

- **Link** A generalized linear model link to connect the feature importance values to the model output

  Since the feature importance values, phi, sum up to the model output, it often makes sense to connect them to the ouput with a link function where link(outout) = sum(phi).

  If the model output is a probability then the LogitLink link function makes the feature importance values have log-odds units.

- **Model sample size** Number of times to re-evaluate the model when explaining each prediction.

  More samples lead to lower variance estimates of the SHAP values.

- **Auto choose model sample size** The `auto` setting uses `Model sample size` = 2 * train_data.shape[1] + 2048.

- **L1 regularization** The l1 regularization to use for feature selection (the estimation procedure is based on a debiased lasso).

  You can choose following inputs:

  - The `auto` option currently uses "aic" when less that 20% of the possible sample space is enumerated, otherwise it uses no regularization.
  - The `aic` or `bic` options use the AIC and BIC rules for regularization
  - `Integer` selects a fix number of top features
  - `float` directly sets the "alpha" parameter of the sklearn.linear_model.Lasso model used for feature selection

- **Class to plot** If explaining classification problem, select which class prediction to explain

1.4.2 Resources

- SHAP paper
- SHAP in IML book
- Documentation of shap
- Implementation (currently documentation replacement) of force plot
- Implementation (currently documentation replacement) of decision plot

1.5 SHAP Feature Importance

`ExpyBox.`**`shap_feature_importance`**`()`

Create dialog for shap summary plot, i.e. feature importance based on Shapley values

Returns None

1.5.1 Method parameters

See shap parameters.

Additional parameter:

- **Sample size** Number of instances from train data to use for calculating mean shap value for each feature.

1.5.2 Resources

- SHAP paper
- SHAP in IML book
- Documentation of shap
- Implementation (currently documentation replacement) of summary plot

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

INDEX