TE2Rules

Release latest

G Roshan Lal

Apr 22, 2024

CONTENTS

TE2Rules (Tree Ensemble to Rules) is a Python library to explain Tree Ensemble Models by extracting a rule list from the trained model. It is an implementation of the TE2Rules algorithm introduced in the paper TE2Rules: Explaining Tree Ensembles using Rules and offers a *simple* and *intuitive* API.

Check out the *Usage* section for further information, including *Installation* of the project.

Note: This project is under active development.

CHAPTER

ONE

CONTENTS

1.1 Usage

1.1.1 Installation

TE2Rules package is available on PyPI and can be installed with pip:

```
$ pip install te2rules
```

1.1.2 Explaining Tree Ensemble Models

To use TE2Rules, start with instantiating a `te2rules.explainer.ModelExplainer` with the tree ensemble model to be explained.

class te2rules.explainer.**ModelExplainer**(*model: sklearn.ensemble | XGBClassifier*, *feature_names: List[str]*, *verbose: bool = False*)

The *te2rules.explainer.ModelExplainer* module explains Tree Ensemble models (TE) like XGBoost, Random Forest, trained on a binary classification task, using a rule list. The algorithm used by TE2Rules is based on Apriori Rule Mining. For more details on the algorithm, please check out our paper TE2Rules: Explaining Tree Ensembles using Rules.

te2rules.explainer.ModelExplainer.**__init__**(*self*, *model: sklearn.ensemble | XGBClassifier*, *feature_names: List[str]*, *verbose: bool = False*)

Initialize the explainer with the trained tree ensemble model and feature names used by the model.

Returns a ModelExplainer object

Parameters

- **model** (*sklearn.ensemble.GradientBoostingClassifier or sklearn.ensemble.RandomForestClassifier or xgboost.XGBClassifier*) – The trained Tree Ensemble model to be explained. The model is expected to be a binary classifier.
- **feature_name** (*List[str]*) – List of feature names used by the *model*. Only alphanumeric characters and underscores are allowed in feature names.
- **verbose** (*bool, optional*) – Optional boolean value to give more insights on the running of the explanation algorithm. Default = False

Returns

self – A ModelExplainer object initialized with the model to be explained.

Return type

te2rules.explainer.ModelExplainer

Raises

- `ValueError:` – when *model* is not a supported Tree Ensemble Model. Currently, only scikit-learn's GradientBoostingClassifier, RandomForestClassifier and xgboost's XGBClassifier are supported.
- `ValueError:` – when *feature_name* list contains a name that has any character other than alphanumeric characters or underscore.

To explain, the tree ensemble model globally with a list of rules:

`te2rules.explainer.ModelExplainer.`**`explain`**(*self*, *X: List[List[float]]*, *y: List[int]*, *num_stages: int* | *None = None*, *min_precision: float = 0.95*, *jaccard_threshold: float = 0.2*) → List[str]

A method to extract rule list from the tree ensemble model. This method takes in input features used by the model and predicted class output by the model.

Returns a List of rule strings.

Parameters

- **X** (*2d numpy.array*) – 2 dimensional input data used by the *model*
- **y** (*1d numpy.array*) – 1 dimensional model class predictions (0 or 1) from the *model*
- **num_stages** (*int, optional*) – The algorithm runs in stages starting from stage 1, stage 2 to all the way till stage n where n is the number of trees in the ensemble. Stopping the algorithm at an early stage results in a few short rules (with quicker run time, but less coverage in data). By default, the algorithm explores all stages before terminating.
- **min_precision** (*float, optional*) – This parameter controls the minimum precision of extracted rules. Setting it to a smaller threshold, allows extracting shorter (more interpretable, but less faithful) rules. By default, the algorithm uses a minimum precision threshold of 0.95.
- **jaccard_threshold** (*float, optional*) – This parameter (between 0 and 1) controls how rules from different node combinations are combined. Setting it to a smaller threshold, ensures that only rules that are very different from one another are combined, speeding up the algorithm at the cost of missing some rules that can potentially explain the model. Combining similar rules that cover very similar set of instances do not provide any new information for the explainer algorithm. By default, the algorithm uses a jaccard threshold of 0.20.

Returns

rules – A List of human readable rules.

Return type

List[str]

Raises

- `ValueError:` – when *X* and *y* are of different length.
- `ValueError:` – when entries in *y* are other than 0 and 1. Only binary classification is supported.

Notes

The data is used for extracting rules with relevant combination of input features. Without data, explainer would need to extract rules for all possible combinations of input features, including those combinations which are extremely rare in the data.

Examples

```
>>> from te2rules.explainer import ModelExplainer
>>> model_explainer = ModelExplainer(model=model, feature_names=feature_names)
>>> rules = model_explainer.explain(X=x_train, y=y_train_pred)
```

To explain, the tree ensemble model's positive class prediction locally, for some specific input:

`te2rules.explainer.ModelExplainer.`**`explain_instance_with_rules`**`(`*self*, *X: List[List[float]]*, *explore_all_rules: bool = True*`)` → List[List[str]]

A method to explain the model output for a list of inputs using rules. For each instance in the list, if the model output is positive, this method returns a corresponding list of rules that explain that instance. For any instance in the list, for which the model output is negative, this method returns an empty list corresponding to that instance.

Returns a list of explanations corresponding to each input. Each explanation is a list of possible rules that can explain the corresponding instance.

Parameters

- **X** (*2d numpy.array*) – 2 dimensional data with feature values that can be sent to the model for predicting outcome.
- **explore_all_rules** (*boolean, optional*) – optional boolean variable guiding the algorithm's behavior. When set to True, the algorithm considers all possible rules (longer_rules) extracted by the explain() function before employing set cover to select a condensed subset of rules. When set to False, the algorithm considers only the condensed subset of rules returned by explain().

  By default, the function utilizes all possible rules (longer_rules) obtained through explain().

Returns

- *list of explaining rules for each instance, with each explanation presented*
- *as a list of rules each of which can independently explain the instance.*

To evaluate the extracted rule list:

`te2rules.explainer.ModelExplainer.`**`get_fidelity`**`(`*self*, *X: List[List[float]] | None = None*, *y: List[int] | None = None*`)` → Tuple[float, float, float]

A method to evaluate the rule list extracted by the *explain* method

Returns a fidelity on positives, negative, overall

Parameters

- **X** (*2d numpy.array, optional*) – 2 dimensional data with feature values used for calculating fidelity. Defaults to data used by the model for rule extraction.
- **y** (*1d numpy.array, optional*) – 1 dimensional model class predictions (0 or 1) from the *model* on X. Defaults to model class predictions on the data used by the model for rule extraction.

Returns
fidelity – Fidelity is the fraction of data for which the rule list agrees with the tree ensemble. Returns the fidelity on overall data, positive predictions and negative predictions by the model.

Return type
[float, float, float]

Examples

```
>>> (fidelity, fidelity_pos, fidelity_neg) = model_explainer.get_fidelity()
```

1.2 API

`te2rules.explainer.ModelExplainer`(model, ...)	The `te2rules.explainer.ModelExplainer` module explains Tree Ensemble models (TE) like XGBoost, Random Forest, trained on a binary classification task, using a rule list.

1.2.1 te2rules.explainer.ModelExplainer

class `te2rules.explainer.`**`ModelExplainer`**(*model: sklearn.ensemble | XGBClassifier*, *feature_names: List[str]*, *verbose: bool = False*)

The `te2rules.explainer.ModelExplainer` module explains Tree Ensemble models (TE) like XGBoost, Random Forest, trained on a binary classification task, using a rule list. The algorithm used by TE2Rules is based on Apriori Rule Mining. For more details on the algorithm, please check out our paper TE2Rules: Explaining Tree Ensembles using Rules.

`__init__`(*model: sklearn.ensemble | XGBClassifier*, *feature_names: List[str]*, *verbose: bool = False*)

Initialize the explainer with the trained tree ensemble model and feature names used by the model.

Returns a ModelExplainer object

Parameters

- **model** (*sklearn.ensemble.GradientBoostingClassifier or sklearn.ensemble.RandomForestClassifier or xgboost.XGBClassifier*) – The trained Tree Ensemble model to be explained. The model is expected to be a binary classifier.
- **feature_name** (*List[str]*) – List of feature names used by the *model*. Only alphanumeric characters and underscores are allowed in feature names.
- **verbose** (*bool, optional*) – Optional boolean value to give more insights on the running of the explanation algorithm. Default = False

Returns
self – A ModelExplainer object initialized with the model to be explained.

Return type
te2rules.explainer.ModelExplainer

Raises

- **ValueError:** – when *model* is not a supported Tree Ensemble Model. Currently, only scikit-learn's GradientBoostingClassifier, RandomForestClassifier and xgboost's XGBClassifier are supported.
- **ValueError:** – when *feature_name* list contains a name that has any character other than alphanumeric characters or underscore.

Methods

__init__(model, feature_names[, verbose])	Initialize the explainer with the trained tree ensemble model and feature names used by the model.
explain(X, y[, num_stages, min_precision, ...])	A method to extract rule list from the tree ensemble model.
explain_instance_with_rules(X[, ...])	A method to explain the model output for a list of inputs using rules.
get_fidelity([X, y])	A method to evaluate the rule list extracted by the *explain* method
predict(X)	A method to apply rules found by the explain() method on a given input data.

INDEX