mlflow-extend

Release []

harupy

May 24, 2020

CONTENTS:

1 Installation **1**

2 API Reference **3**
- 2.1 Logging . . . 3
- 2.2 Plotting . . . 8

3 Examples **15**
- 3.1 Quick Start . . . 15
- 3.2 LightGBM Binary Classification . . . 16

Python Module Index **19**

Index **21**

CHAPTER

ONE

INSTALLATION

From PyPI

```
pip install mlflow-extend
```

From GitHub (development version)

```
pip install git+https://github.com/harupy/mlflow-extend.git
```

CHAPTER

TWO

API REFERENCE

2.1 Logging

mlflow_extend.logging.**log_params_flatten**(*params*, *parent_key=''*, *sep='.'*)

Log a batch of params after flattening.

Parameters

- **params** (*dict*) – Dictionary of parameters to log.
- **parent_key** (*str, default ""*) – Parent key.
- **sep** (*str, default "."*) – Key separator.

Returns None

Return type None

Examples

```
>>> with mlflow.start_run() as run:
...     params = {"a": {"b": 0}}
...     mlflow.log_params_flatten(params)
...     mlflow.log_params_flatten(params, parent_key="d")
...     mlflow.log_params_flatten(params, sep="_")
>>> r = mlflow.get_run(run.info.run_id)
>>> sorted(r.data.params.items())
[('a.b', '0'), ('a_b', '0'), ('d.a.b', '0')]
```

mlflow_extend.logging.**log_metrics_flatten**(*metrics*, *step=None*, *parent_key=''*, *sep='.'*)

Log a batch of metrics after flattening.

Parameters

- **metrics** (*dict*) – Dictionary of metrics to log.
- **step** (*int, default None*) – Metric step. Defaults to zero if unspecified.
- **parent_key** (*str, default ""*) – Parent key.
- **sep** (*str, default "."*) – Key separator.

Returns None

Return type None

Examples

```
>>> with mlflow.start_run() as run:
...     metrics = {"a": {"b": 0.0}}
...     mlflow.log_metrics_flatten(metrics)
...     mlflow.log_metrics_flatten(metrics, parent_key="d")
...     mlflow.log_metrics_flatten(metrics, sep="_")
>>> r = mlflow.get_run(run.info.run_id)
>>> sorted(r.data.metrics.items())
[('a.b', 0.0), ('a_b', 0.0), ('d.a.b', 0.0)]
```

mlflow_extend.logging.**log_plt_figure**(*fig*, *path*)

Log a matplotlib figure as an artifact.

Parameters

- **fig** (*matplotlib.pyplot.Figure*) – Figure to log.
- **path** (*str*) – Path in the artifact store.

Returns None

Return type None

Examples

```
>>> with mlflow.start_run() as run:
...     fig, ax = plt.subplots()
...     _ = ax.plot([0, 1], [0, 1])
...     mlflow.log_figure(fig, 'plt_figure.png')
>>> list_artifacts(run.info.run_id)
['plt_figure.png']
```

mlflow_extend.logging.**log_plotly_figure**(*fig*, *path*)

Log a plotly figure as an artifact.

Parameters

- **fig** (*go.Figure*) – Figure to log.
- **path** (*str*) – Path in the artifact store.

Returns None

Return type None

Examples

```
>>> with mlflow.start_run() as run:
...     fig = go.Figure(data=[go.Bar(x=[1, 2, 3], y=[1, 3, 2])])
...     mlflow.log_figure(fig, 'plotly_figure.html')  # Must be an HTML file.
>>> list_artifacts(run.info.run_id)
['plotly_figure.html']
```

mlflow_extend.logging.**log_figure**(*fig*, *path*)

Log a matplotlib figure as an artifact.

Parameters

- **fig** (*matplotlib.pyplot.Figure or plotly.graph_objects.Figure*) – Figure to log.
- **path** (*str*) – Path in the artifact store.

Returns None

Return type None

Examples

Matplotlib

```
>>> with mlflow.start_run() as run:
...     fig, ax = plt.subplots()
...     _ = ax.plot([0, 1], [0, 1])
...     mlflow.log_figure(fig, 'plt_figure.png')
>>> list_artifacts(run.info.run_id)
['plt_figure.png']
```

Plotly

```
>>> with mlflow.start_run() as run:
...     fig = go.Figure(data=[go.Bar(x=[1, 2, 3], y=[1, 3, 2])])
...     mlflow.log_figure(fig, 'plotly_figure.html')  # Must be an HTML file.
>>> list_artifacts(run.info.run_id)
['plotly_figure.html']
```

mlflow_extend.logging.**log_dict**(*dct*, *path*, *fmt=None*)

Log a dictionary as an artifact.

Parameters

- **dct** (*dict*) – Dictionary to log.
- **path** (*str*) – Path in the artifact store.
- **fmt** (*str, default None*) – File format to save dict in. If None, file format is inferred from *path*.

Returns None

Return type None

Examples

```
>>> with mlflow.start_run() as run:
...     d = {'a': 0}
...     mlflow.log_dict(d, 'dict.json')
...     mlflow.log_dict(d, 'dict.yaml')
...     mlflow.log_dict(d, 'dict.yml')
>>> list_artifacts(run.info.run_id)
['dict.json', 'dict.yaml', 'dict.yml']
```

mlflow_extend.logging.**log_df**(*df*, *path*, *fmt='csv'*)

Log a dataframe as an artifact.

Parameters

- **df** (*pandas.DataFrame*) – Dataframe to log.

- **path** (*str*) – Path in the artifact store.
- **fmt** (*str, default "csv"*) – File format to save the dataframe in.

Returns None

Return type None

Examples

```
>>> with mlflow.start_run() as run:
...     mlflow.log_df(pd.DataFrame({'a': [0]}), 'df.csv')
>>> list_artifacts(run.info.run_id)
['df.csv']
```

mlflow_extend.logging.**log_text**(*text*, *path*)

Log a text as an artifact.

Parameters

- **text** (*str*) – Text to log.
- **path** (*str*) – Path in the artifact store.

Returns None

Return type None

Examples

```
>>> with mlflow.start_run() as run:
...     mlflow.log_text('text', 'text.txt')
>>> list_artifacts(run.info.run_id)
['text.txt']
```

mlflow_extend.logging.**log_numpy**(*arr*, *path*)

Log a numpy array as an artifact.

Parameters

- **arr** (*numpy.ndarray*) – Numpy array to log.
- **path** (*str*) – Path in the artifact store.

Returns None

Return type None

Examples

```
>>> with mlflow.start_run() as run:
...     mlflow.log_numpy(np.array([0]), 'array.npy')
>>> list_artifacts(run.info.run_id)
['array.npy']
```

mlflow_extend.logging.**log_confusion_matrix**(*cm*, *path='confusion_matrix.png'*)

Log a confusion matrix as an artifact.

Parameters

- **cm** (*array-like*) – Confusion matrix to log.
- **path** (*str, default "confusion_matrix.png"*) – Path in the artifact store.

Returns None

Return type None

Examples

```
>>> with mlflow.start_run() as run:
...     mlflow.log_confusion_matrix([[1, 2], [3, 4]])
>>> list_artifacts(run.info.run_id)
['confusion_matrix.png']
```

mlflow_extend.logging.**log_feature_importance**(*features*, *importances*, *importance_type*, *limit=None*, *normalize=False*, *path='feature_importance.png'*)

Log feature importance as an artifact.

Parameters

- **features** (*array-like*) – Feature names.
- **importances** (*array-like*) – Importance of each feature.
- **importance_type** (*str*) – Importance type (e.g. "gain").
- **path** (*str, default "feature_importance.png"*) – Path in the artifact store.
- ****kwargs** (*dict*) – Keyword arguments passed to mlflow.plotting.feature_importance.

Returns None

Return type None

Examples

```
>>> with mlflow.start_run() as run:
...     features = ['a', 'b', 'c']
...     importances = [1, 2, 3]
...     mlflow.log_feature_importance(features, importances, 'gain')
>>> list_artifacts(run.info.run_id)
['feature_importance.png']
```

mlflow_extend.logging.**log_roc_curve**(*fpr*, *tpr*, *auc=None*, *path='roc_curve.png'*)

Log ROC curve as an artifact.

Parameters

- **fpr** (*array-like*) – False positive rate.
- **tpr** (*array-like*) – True positive rate.
- **auc** (*float, default None*) – Area under the curve.
- **path** (*str, default "roc_curve.png"*) – Path in the artifact store.

Returns None

Return type None

Examples

```
>>> with mlflow.start_run() as run:
...     mlflow.log_roc_curve([0, 1], [0, 1])
>>> list_artifacts(run.info.run_id)
['roc_curve.png']
```

mlflow_extend.logging.**log_pr_curve**(*pre*, *rec*, *auc=None*, *path='pr_curve.png'*)

Log precision-recall curve as an artifact.

Parameters

- **pre** (*array-like*) – Precision.
- **rec** (*array-like*) – Recall.
- **auc** (*float, default None*) – Area under the curve.
- **path** (*str, default "pr_curve.png"*) – Path in the artifact store.

Returns None

Return type None

Examples

```
>>> with mlflow.start_run() as run:
...     mlflow.log_pr_curve([1, 0], [1, 0])
>>> list_artifacts(run.info.run_id)
['pr_curve.png']
```

2.2 Plotting

mlflow_extend.plotting.**corr_matrix**(*corr*)

Plot correlation matrix.

Parameters **corr** (*array-like*) – Correlation matrix.

Returns Figure object.

Return type matplotlib.pyplot.Figure

Examples

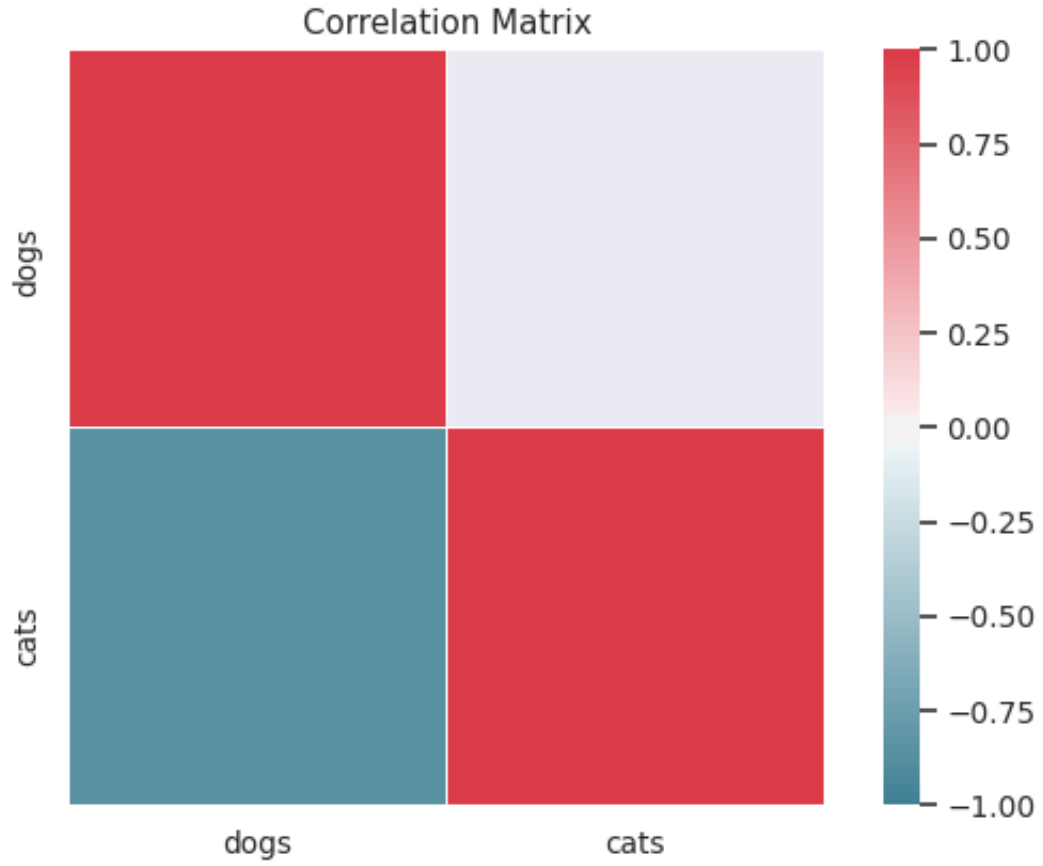
```
>>> df = pd.DataFrame([(0.2, 0.3), (0.0, 0.6), (0.6, 0.0), (0.2, 0.1)],
...                   columns=['dogs', 'cats'])
>>> corr_matrix(df.corr())
<Figure ... with 2 Axes>
```

mlflow_extend.plotting.**confusion_matrix**(*cm*, *labels=None*, *normalize=True*)

Plot confusion matrix.

Parameters

- **cm** (*array-like*) – Confusion matrix.
- **labels** (*list of str, default None*) – Label names.


Correlation Matrix
dogs
cats
dogs
cats
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

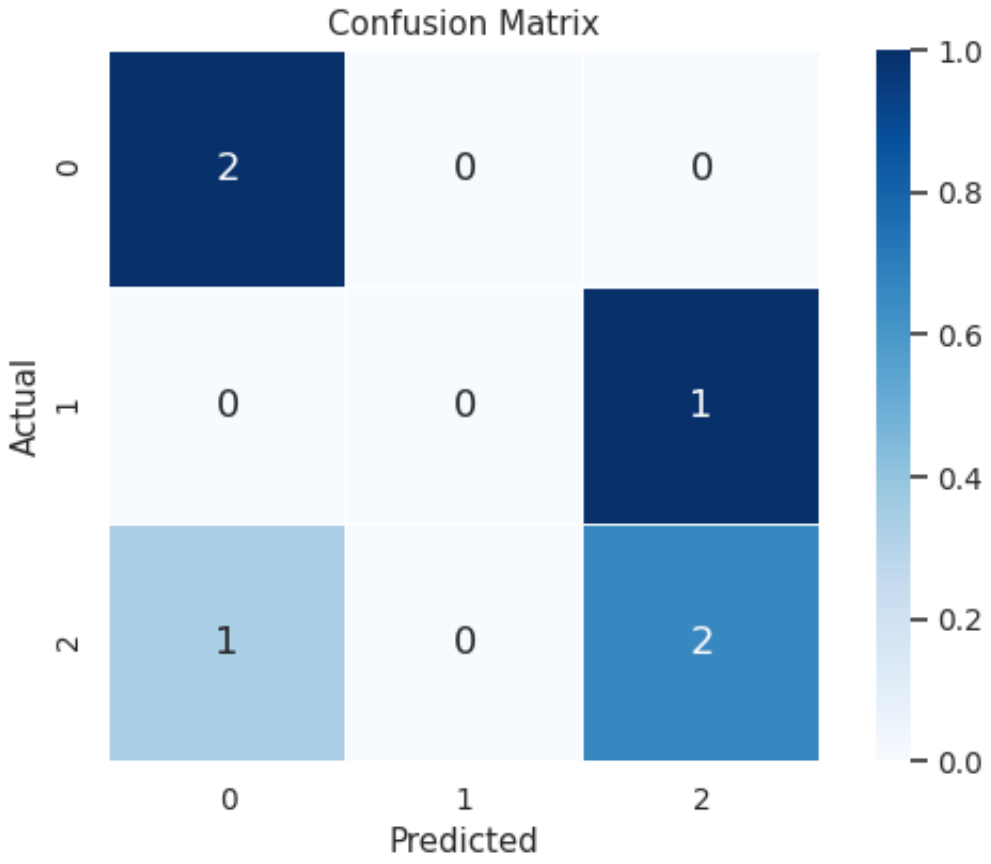
- **normalize** (*bool, default True*) – Divide each row by its sum.

Returns Figure object.

Return type matplotlib.pyplot.Figure

Examples

```
>>> cm = [[2, 0, 0],
...       [0, 0, 1],
...       [1, 0, 2]]
>>> confusion_matrix(cm)
<Figure ... with 2 Axes>
```



mlflow_extend.plotting.**feature_importance**(*features*, *importances*, *importance_type*, *limit=None*, *normalize=False*)

Plot feature importance.

Parameters

- **features** (*list of str*) – Feature names.
- **importances** (*array-like*) – Importance of each feature.
- **importance_type** (*str*) – Feature importance type (e.g. "gain").

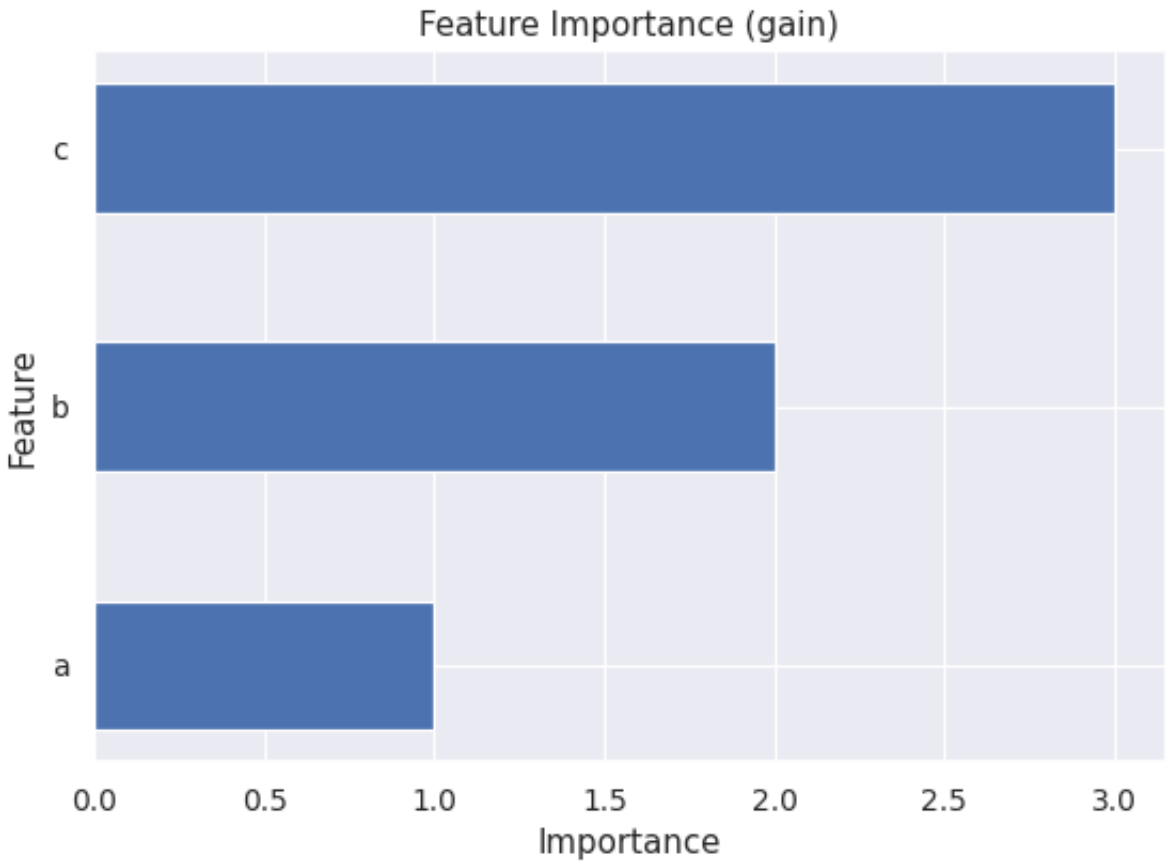
- **limit** (*int, default None*) – Number of features to plot. If `None`, all features will be plotted.
- **normalize** (*bool, default False*) – Divide importance by the sum.

Returns Figure object.

Return type matplotlib.pyplot.Figure

Examples

```
>>> features = ["a", "b", "c"]
>>> importances = [1, 2, 3]
>>> importance_type = "gain"
>>> feature_importance(features, importances, importance_type)
<Figure ... with 1 Axes>
```



mlflow_extend.plotting.**roc_curve**(*fpr*, *tpr*, *auc=None*)

Plot ROC curve.

Parameters

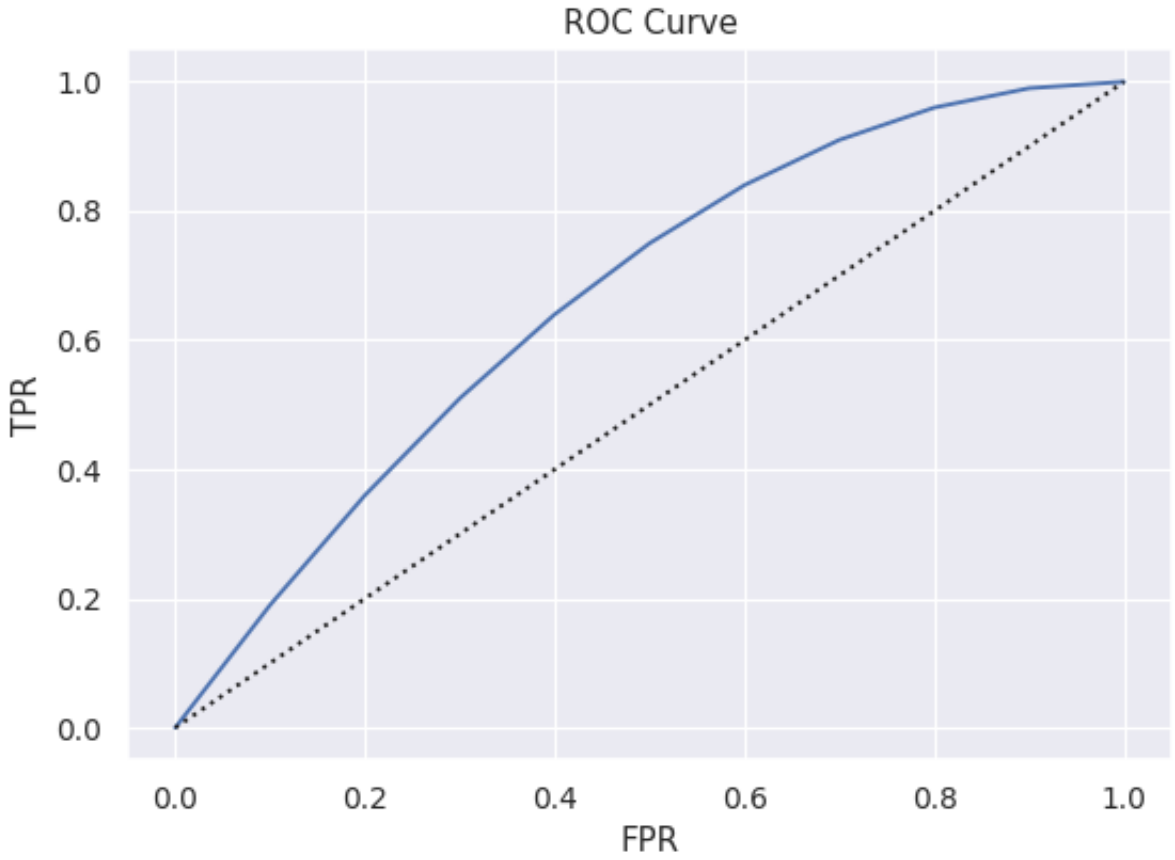
- **fpr** (*array-like*) – False positive rate.
- **tpr** (*array-like*) – True positive rate.
- **auc** (*float, default None*) – Area under the curve.

Returns Figure object.

Return type matplotlib.pyplot.Figure

Examples

```
>>> fpr = np.linspace(0, 1, 11)
>>> tpr = -((fpr - 1) ** 2) + 1
>>> roc_curve(fpr, tpr)
<Figure ... with 1 Axes>
```



mlflow_extend.plotting.**pr_curve**(*pre*, *rec*, *auc=None*)

Plot precision-recall curve.

Parameters

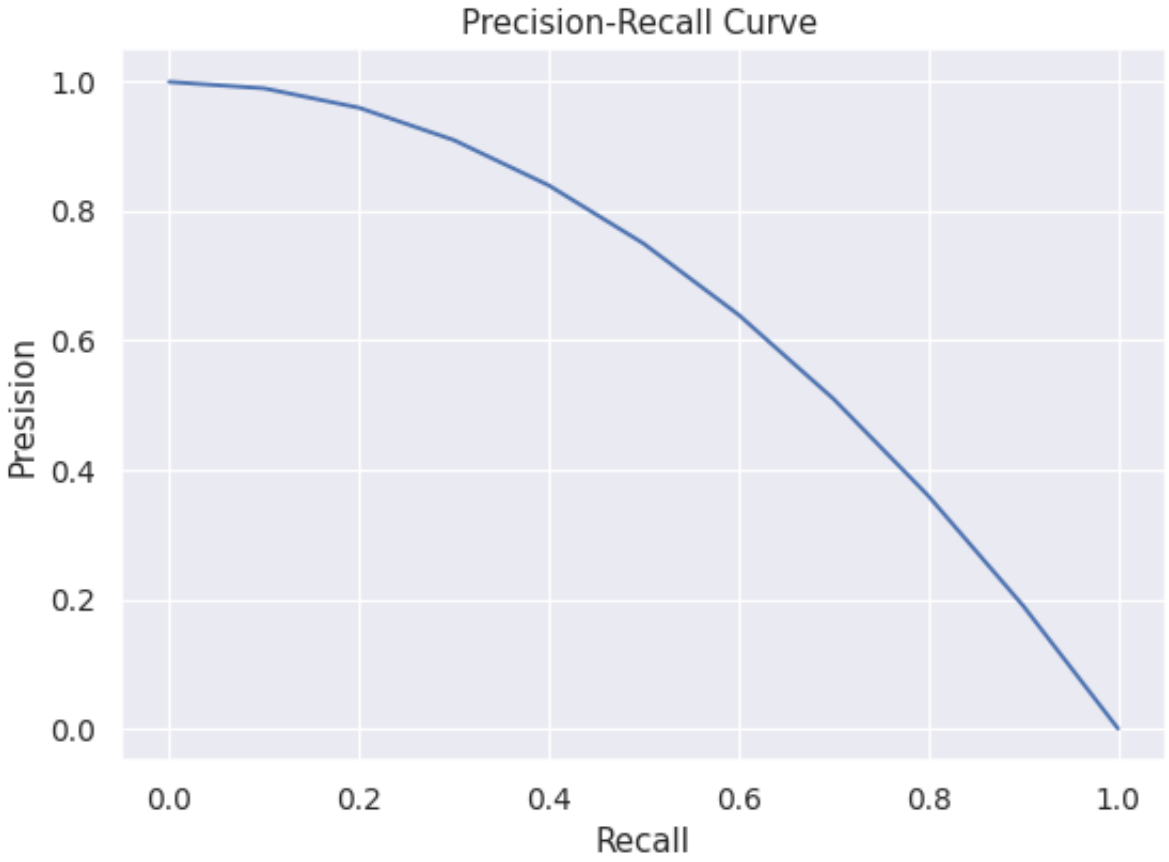
- **pre** (*array-like*) – Precision.
- **rec** (*array-like*) – Recall.
- **auc** (*float, default None*) – Area under the curve.

Returns Figure object.

Return type matplotlib.pyplot.Figure

Examples

```
>>> rec = np.linspace(0, 1, 11)
>>> pre = -(rec ** 2) + 1
>>> pr_curve(pre, rec)
<Figure ... with 1 Axes>
```



CHAPTER

THREE

EXAMPLES

3.1 Quick Start

How to run:

```
python examples/quickstart.py
```

Source code:

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

from mlflow_extend import mlflow


def main():
    with mlflow.start_run():
        # mlflow native APIs
        mlflow.log_param("param", 0)
        mlflow.log_metric("metric", 1.0)

        # flatten dict
        mlflow.log_params_flatten({"a": {"b": 0}})
        mlflow.log_metrics_flatten({"a": {"b": 0.0}})

        # dict
        mlflow.log_dict({"a": 0}, "dict.json")

        # numpy array
        mlflow.log_numpy(np.array([0]), "array.npy")

        # pandas dataframe
        mlflow.log_df(pd.DataFrame({"a": [0]}), "df.csv")

        # matplotlib figure
        fig, ax = plt.subplots()
        ax.plot([0, 1], [0, 1])
        mlflow.log_figure(fig, "figure.png")

        # confusion matrix
        mlflow.log_confusion_matrix([[1, 2], [3, 4]])
```

(continues on next page)

(continued from previous page)

```
if __name__ == "__main__":
    main()
```

3.2 LightGBM Binary Classification

How to run:

```
python examples/lightgbm_binary.py
```

Source code:

```
"""
An example script to train a LightGBM classifier on the breast cancer dataset.

The lines that call mlflow_extend APIs are marked with "EX".
"""
import lightgbm as lgb
import pandas as pd
from sklearn import datasets
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split

from mlflow_extend import mlflow


def breast_cancer():
    data = datasets.load_breast_cancer()
    columns = list(filter(lambda c: c.replace(" ", "_"), data.feature_names))
    X = pd.DataFrame(data.data, columns=columns)
    y = pd.Series(data.target, name="target")
    return X, y


def main():
    config = {
        "split": {"test_size": 0.2, "random_state": 42},
        "model": {"objective": "binary", "metric": "auc", "seed": 42},
        "fit": {"num_boost_round": 10, "early_stopping_rounds": 3},
    }
    # Prepare training data.
    X, y = breast_cancer()
    X_train, X_test, y_train, y_test = train_test_split(X, y, **config["split"])
    train_set = lgb.Dataset(X_train, label=y_train)

    # Set experiment.
    expr_name = "lightgbm"
    mlflow.get_or_create_experiment(expr_name)  # EX
    mlflow.set_experiment(expr_name)

    with mlflow.start_run():
        # Log training configuration.
        mlflow.log_params_flatten(config)  # EX
        mlflow.log_dict(config, "config.json")  # EX
```

(continues on next page)

(continued from previous page)

```
        # Train model.
        model = lgb.train(
            config["model"],
            train_set,
            valid_sets=[train_set],
            valid_names=["train"],
            **config["fit"]
        )

        # Log feature importance.
        importance_type = "gain"
        features = model.feature_name()
        importances = model.feature_importance(importance_type)
        mlflow.log_feature_importance(features, importances, importance_type)  # EX

        # Log confusion metrics.
        mlflow.log_metrics_flatten(model.best_score)

        # Log confusion matrix.
        y_proba = model.predict(X_test)
        cm = confusion_matrix(y_test, y_proba > 0.5)
        mlflow.log_confusion_matrix(cm)  # EX


if __name__ == "__main__":
    main()
```

PYTHON MODULE INDEX

m

mlflow_extend.logging, 3
mlflow_extend.plotting, 8

INDEX

C

confusion_matrix() (*in module mlflow_extend.plotting*), 8
corr_matrix() (*in module mlflow_extend.plotting*), 8

F

feature_importance() (*in module mlflow_extend.plotting*), 10

L

log_confusion_matrix() (*in module mlflow_extend.logging*), 6
log_df() (*in module mlflow_extend.logging*), 5
log_dict() (*in module mlflow_extend.logging*), 5
log_feature_importance() (*in module mlflow_extend.logging*), 7
log_figure() (*in module mlflow_extend.logging*), 4
log_metrics_flatten() (*in module mlflow_extend.logging*), 3
log_numpy() (*in module mlflow_extend.logging*), 6
log_params_flatten() (*in module mlflow_extend.logging*), 3
log_plotly_figure() (*in module mlflow_extend.logging*), 4
log_plt_figure() (*in module mlflow_extend.logging*), 4
log_pr_curve() (*in module mlflow_extend.logging*), 8
log_roc_curve() (*in module mlflow_extend.logging*), 7
log_text() (*in module mlflow_extend.logging*), 6

M

mlflow_extend.logging (*module*), 3
mlflow_extend.plotting (*module*), 8

P

pr_curve() (*in module mlflow_extend.plotting*), 12

R

roc_curve() (*in module mlflow_extend.plotting*), 11