Python Code Linting Tool

Enter Python code:

```
from sklearn.feature_selection import mutual_info_classif
import eli5
from eli5.sklearn import PermutationImportance
```

Process Code

```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import mutual_info_classif
from sklearn.model_selection import cross_val_score
import eli5
import shap


def score_dataset(x, y):
    model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=1)
    return cross_val_score(model, x, y, scoring="accuracy").mean()


def make_mi_scores(x, y):
    x = x.copy()
    discrete = x.dtypes == int
    return pd.Series(mutual_info_classif(x, y, discrete_features=discrete, random_


def plot_score(scores):
    scores = scores.sort_values(ascending=True)
    width = np.arange(len(scores))
    ticks = list(scores.index)
    plt.barh(width, scores)
    plt.yticks(width, ticks)
    plt.title("Mutual Information Scores")


def permute_score(x, y):
    model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=1)
    model.fit(x, y)
    perm = eli5.sklearn.PermutationImportance(model, random_state=1).fit(x, y)
    return eli5.show_weights(perm, feature_names=x.columns.tolist())


def shap_score(model, x):
    explainer = shap.TreeExplainer(model)
    shap_values = explainer.shap_values(x)
    return shap.summary_plot(shap_values[1], x), shap_values
```

More Details about code : Here is the PEP8 compliant code with its score:

```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import mutual_info_classif
from sklearn.model_selection import cross_val_score
import eli5
import shap


def score_dataset(x, y):
    model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=1)
    return cross_val_score(model, x, y, scoring="accuracy").mean()


def make_mi_scores(x, y):
    x = x.copy()
    discrete = x.dtypes == int
    return pd.Series(mutual_info_classif(x, y, discrete_features=discrete, random_


def plot_score(scores):
    scores = scores.sort_values(ascending=True)
    width = np.arange(len(scores))
    ticks = list(scores.index)
    plt.barh(width, scores)
    plt.yticks(width, ticks)
    plt.title("Mutual Information Scores")


def permute_score(x, y):
    model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=1)
    model.fit(x, y)
    perm = eli5.sklearn.PermutationImportance(model, random_state=1).fit(x, y)
    return eli5.show_weights(perm, feature_names=x.columns.tolist())
```

```
def shap_score(model, x):
    explainer = shap.TreeExplainer(model)
    shap_values = explainer.shap_values(x)
    return shap.summary_plot(shap_values[1], x), shap_values
```

Before:

```
Score: 6.17/10
```

After:

```
Score: 10/10
```

JSON Output:

```
{
  "code": "import numpy as np\nimport pandas as pd\nfrom sklearn.ensemble import R
  "score_before": "6.17/10",
  "score_after": "10/10"
}
```