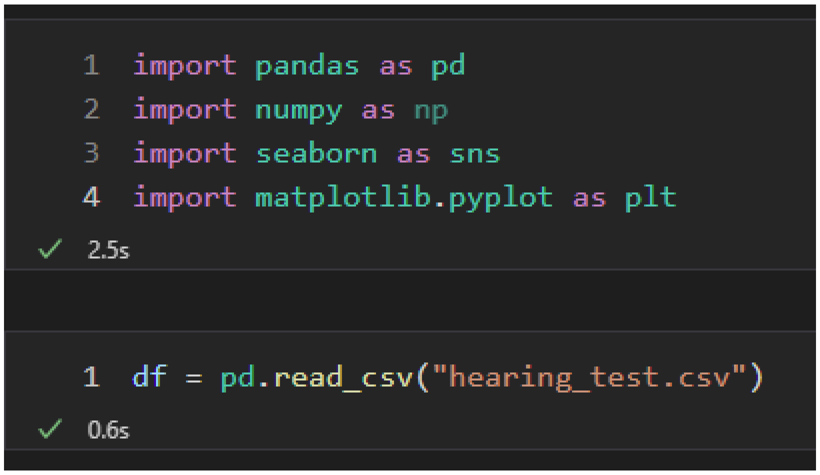
Logistic Regr



```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

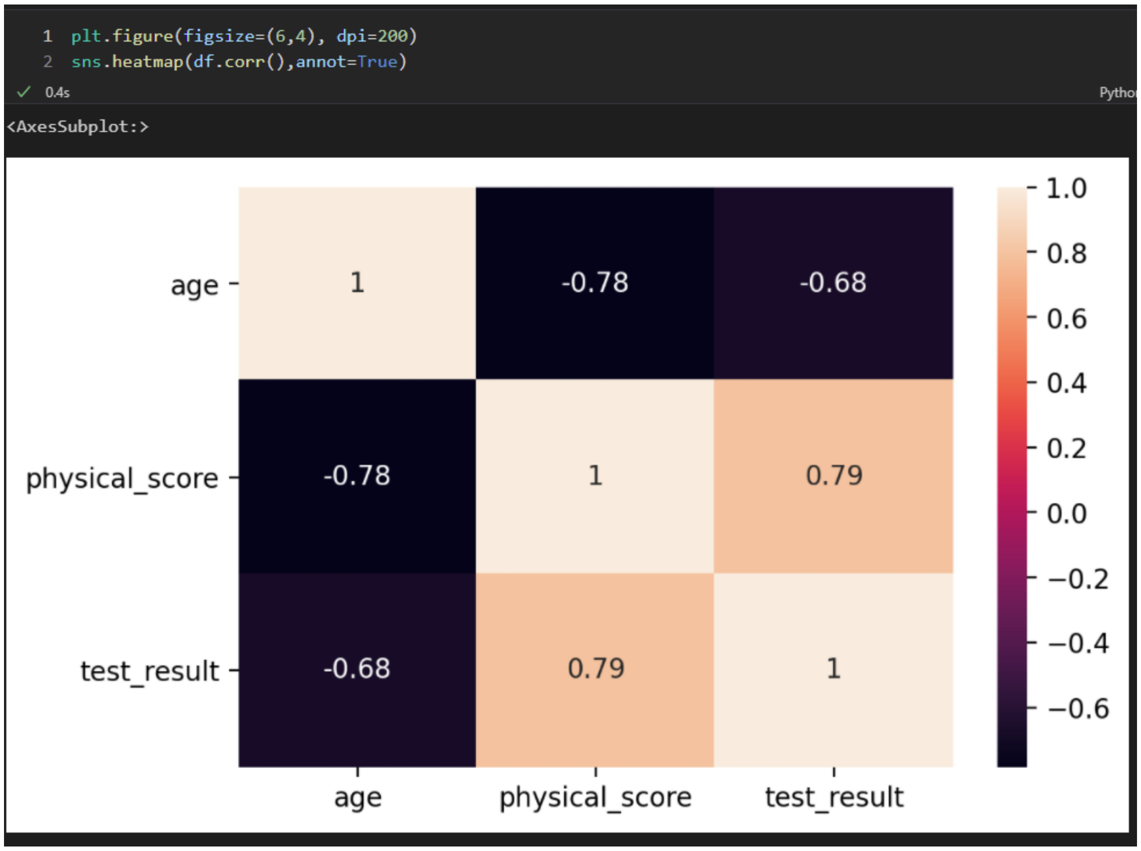
✓ 2.5s

```python
df = pd.read_csv("hearing_test.csv")
```

✓ 0.6s

▼ Ekler

- sns.heatmap(df.corr(),annot=True) :
  annot=True : Corelasyon değerlerini kutularda gösterir



```
plt.figure(figsize=(6,4), dpi=200)
sns.heatmap(df.corr(),annot=True)
```

✓ 0.4s

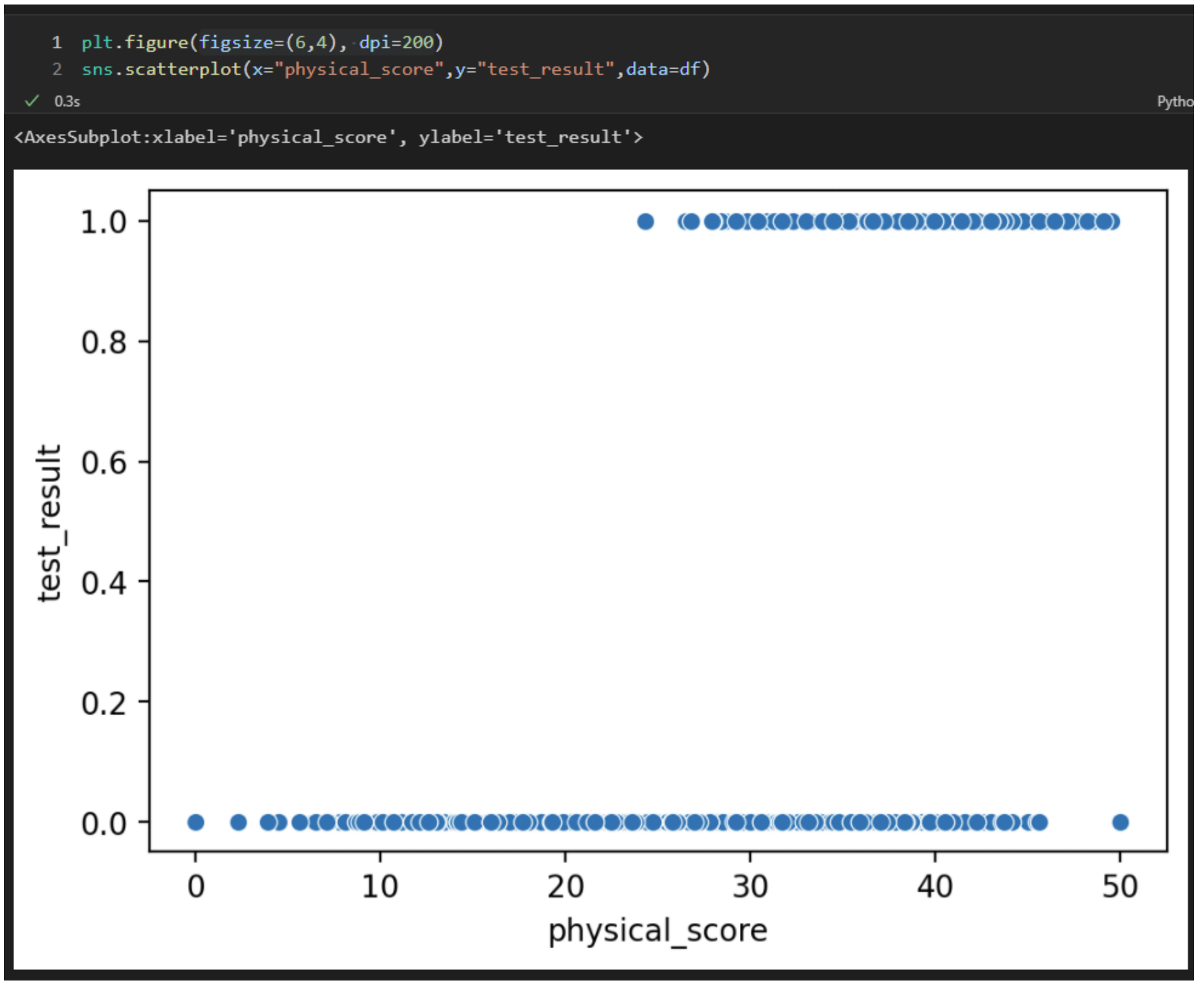
```
<AxesSubplot:>
```

	age	physical_score	test_result
age	1	-0.78	-0.68
physical_score	-0.78	1	0.79
test_result	-0.68	0.79	1

-

▼ Introduction

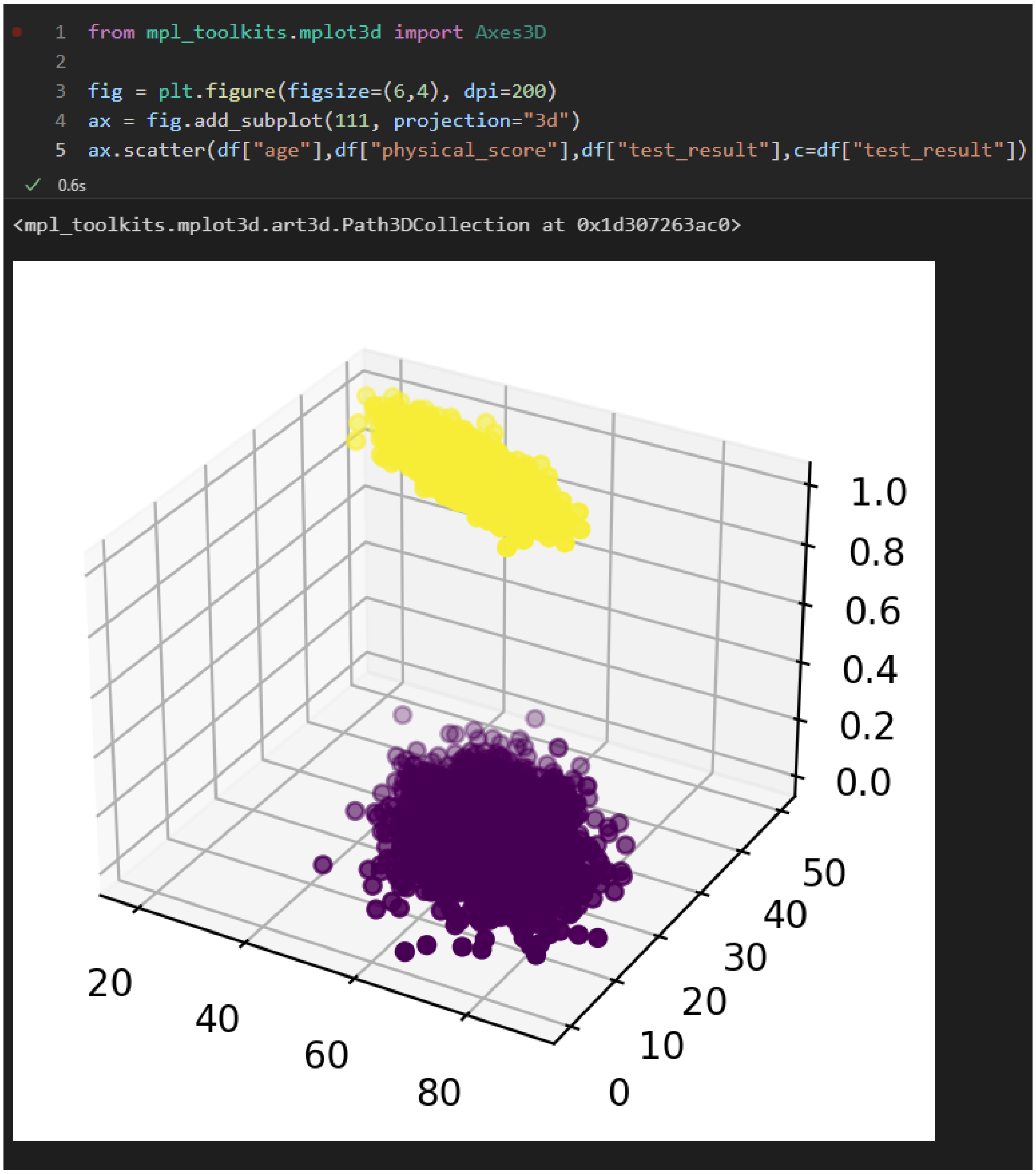
- scatter plot



```
plt.figure(figsize=(6,4), dpi=200)
sns.scatterplot(x="physical_score",y="test_result",data=df)
```

```
<AxesSubplot:xlabel='physical_score', ylabel='test_result'>
```

- 3D Scatter plot



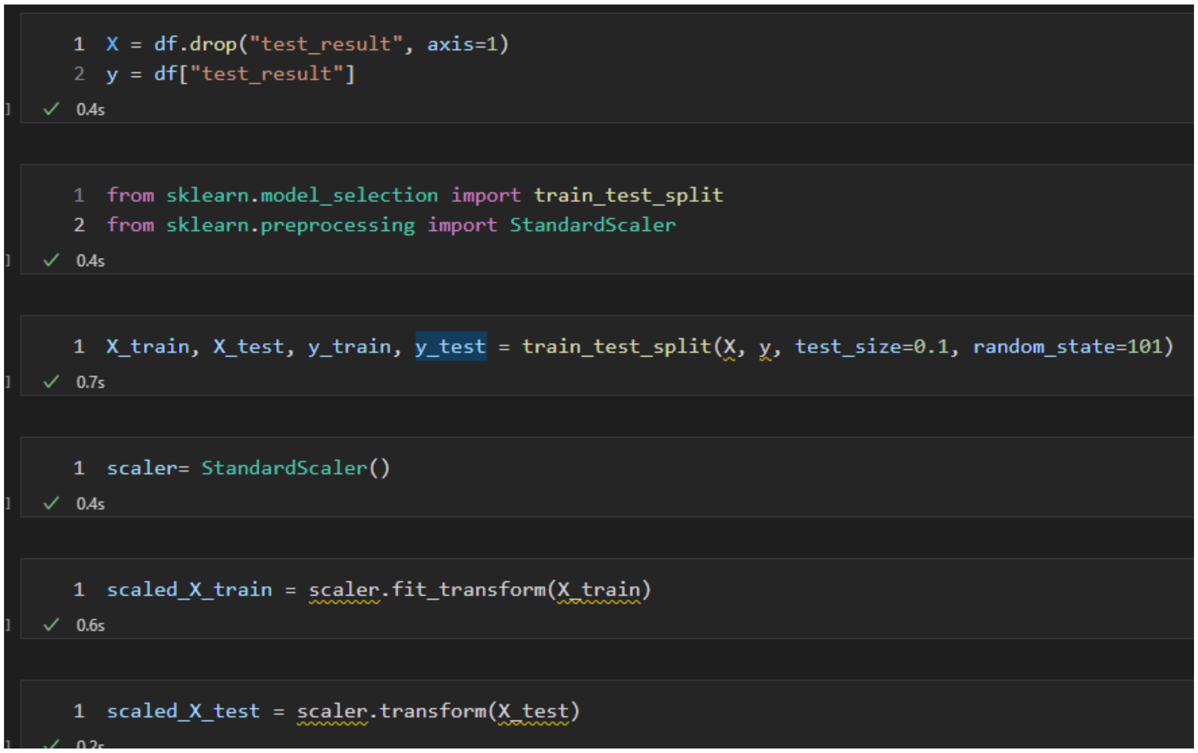
```
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(6,4), dpi=200)
ax = fig.add_subplot(111, projection="3d")
ax.scatter(df["age"],df["physical_score"],df["test_result"],c=df["test_result"])
```

0.6s

```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x1d307263ac0>
```

▼ Model Training

- Data preparing, train test split and scalar transform



```
X = df.drop("test_result", axis=1)
y = df["test_result"]
```
✓ 0.4s

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```
✓ 0.4s

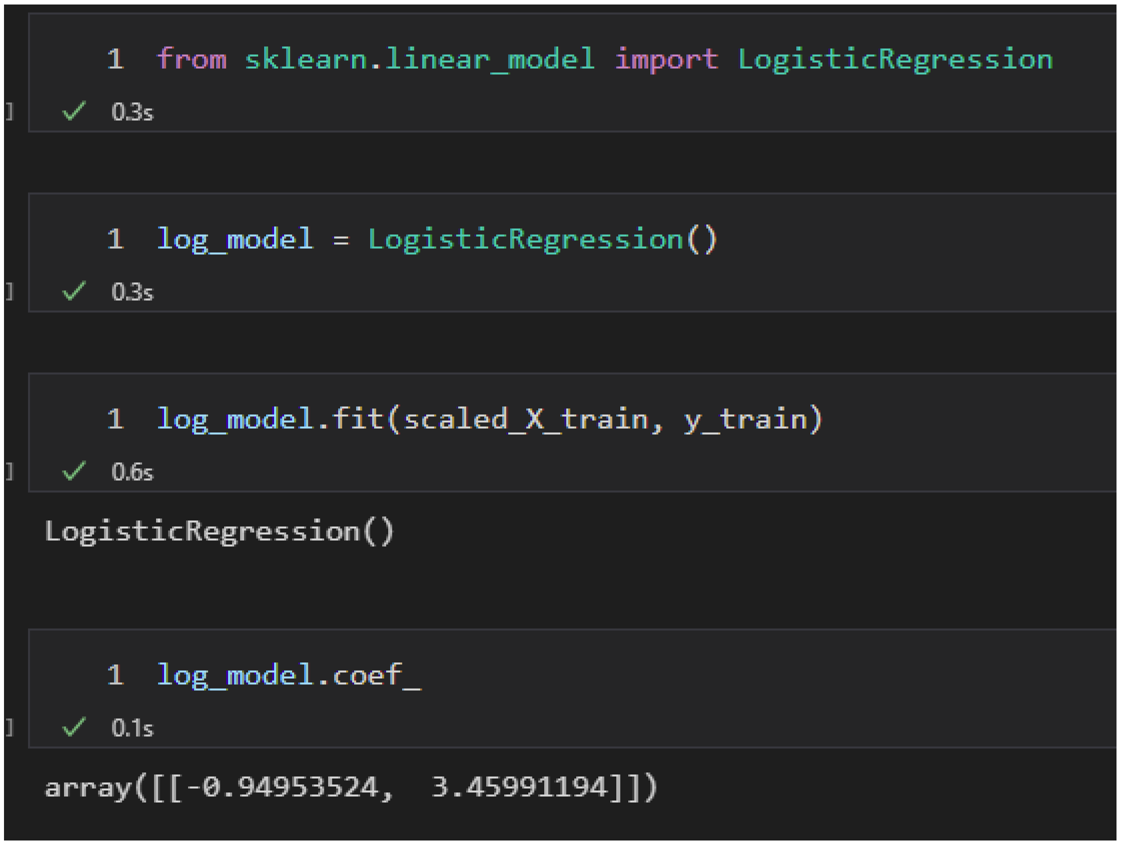
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=101)
```
✓ 0.7s

```
scaler= StandardScaler()
```
✓ 0.4s

```
scaled_X_train = scaler.fit_transform(X_train)
```
✓ 0.6s

```
scaled_X_test = scaler.transform(X_test)
```

- Logistic regression



```
from sklearn.linear_model import LogisticRegression
```
✓ 0.3s

```
log_model = LogisticRegression()
```
✓ 0.3s

```
log_model.fit(scaled_X_train, y_train)
```
✓ 0.6s

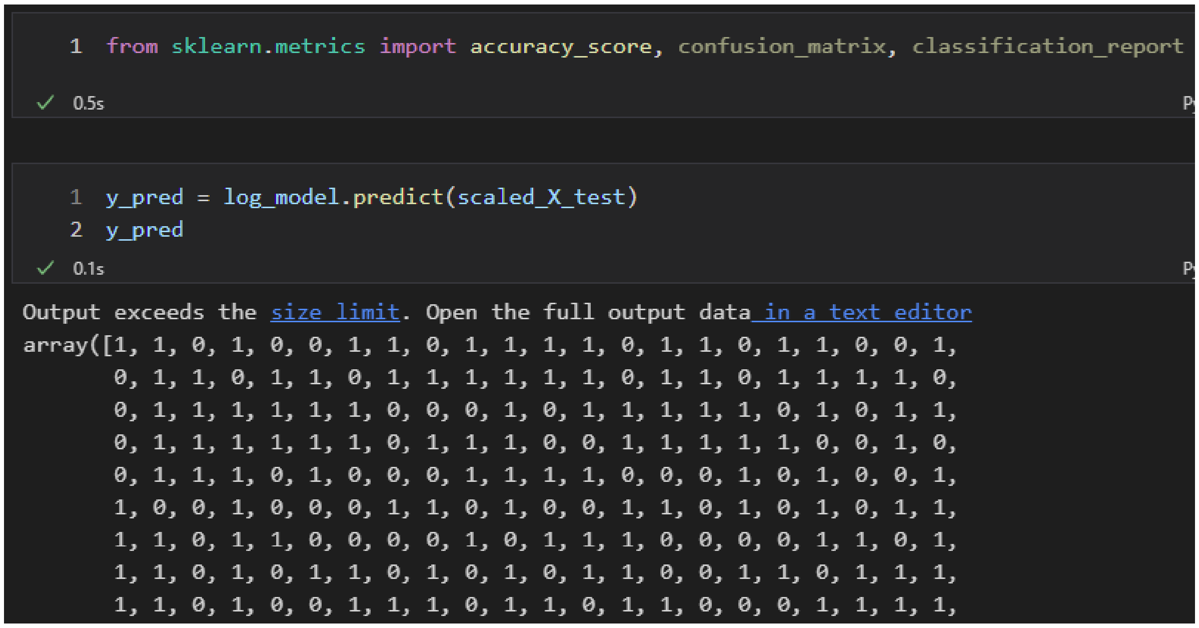
```
LogisticRegression()
```

```
log_model.coef_
```
✓ 0.1s

```
array([[-0.94953524,  3.45991194]])
```

▼ Performance Evaluation

- import and Prediction



```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

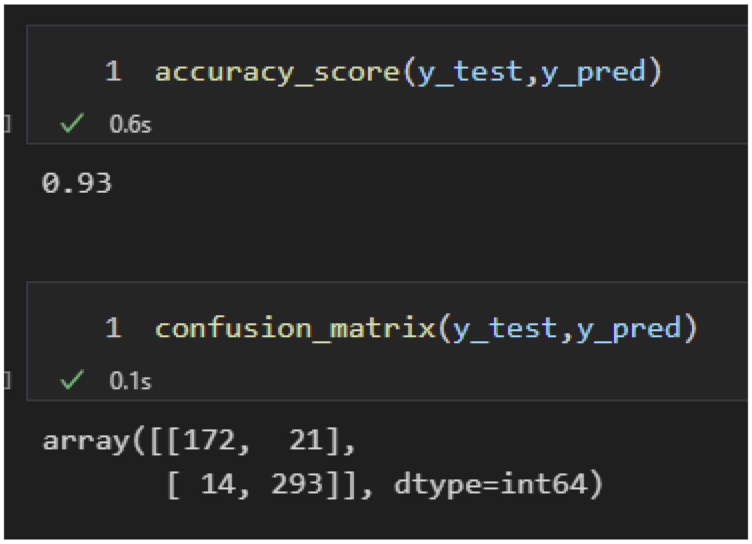
✓ 0.5s

```
y_pred = log_model.predict(scaled_X_test)
y_pred
```

✓ 0.1s

```
Output exceeds the size limit. Open the full output data in a text editor
array([1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1,
       0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0,
       0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0,
       0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1,
       1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1,
       1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1,
```

- accuracy score & confusion metrix(True false neg pos)



```
accuracy_score(y_test,y_pred)
```

✓ 0.6s

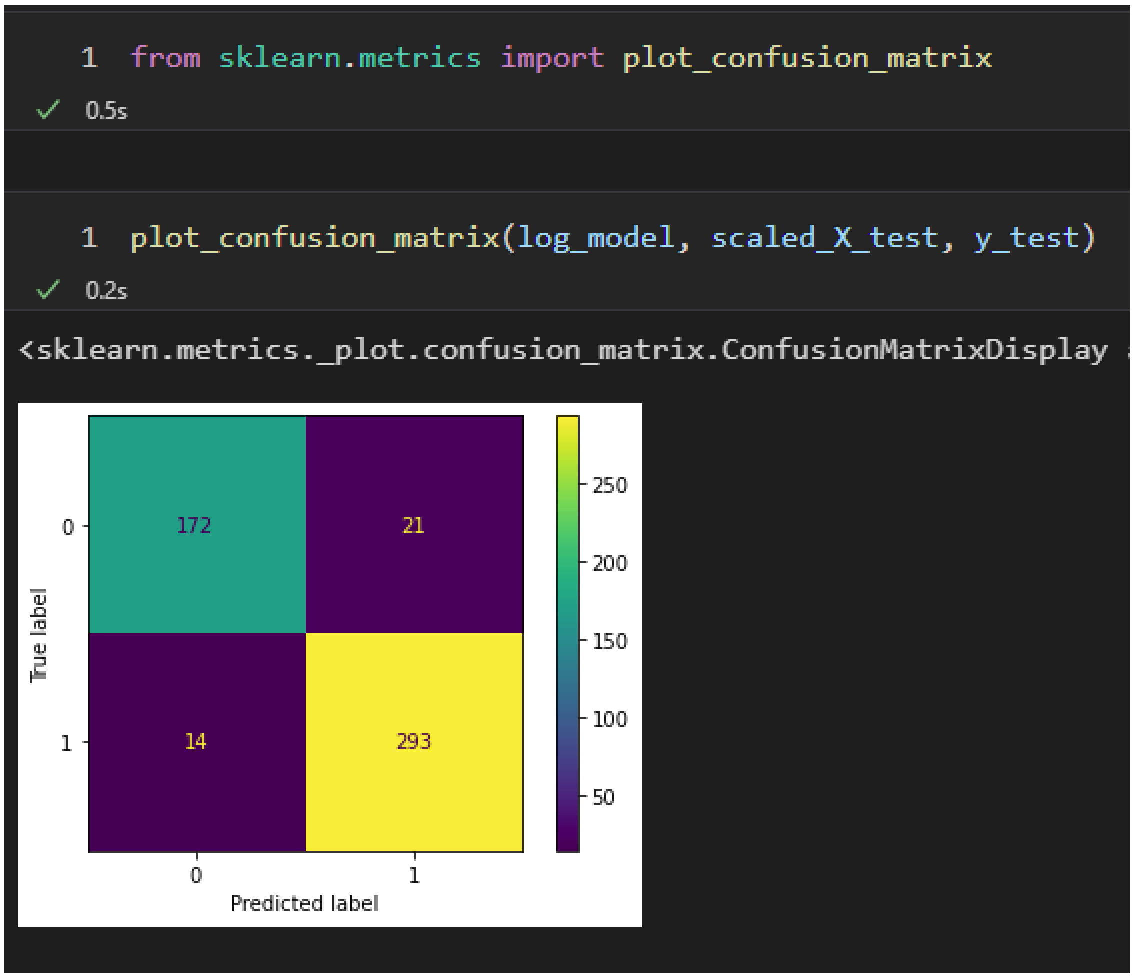
```
0.93
```

```
confusion_matrix(y_test,y_pred)
```

✓ 0.1s

```
array([[172,  21],
       [ 14, 293]], dtype=int64)
```

- Confusion Matrix plot



```
from sklearn.metrics import plot_confusion_matrix
```

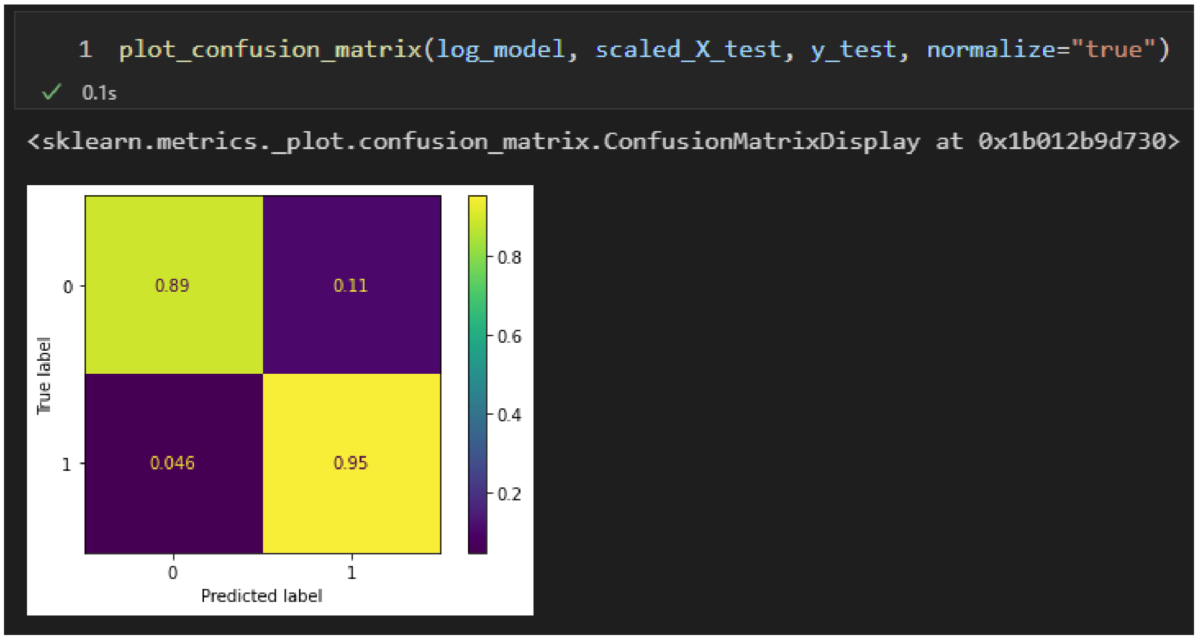
✓ 0.5s

```
plot_confusion_matrix(log_model, scaled_X_test, y_test)
```

✓ 0.2s

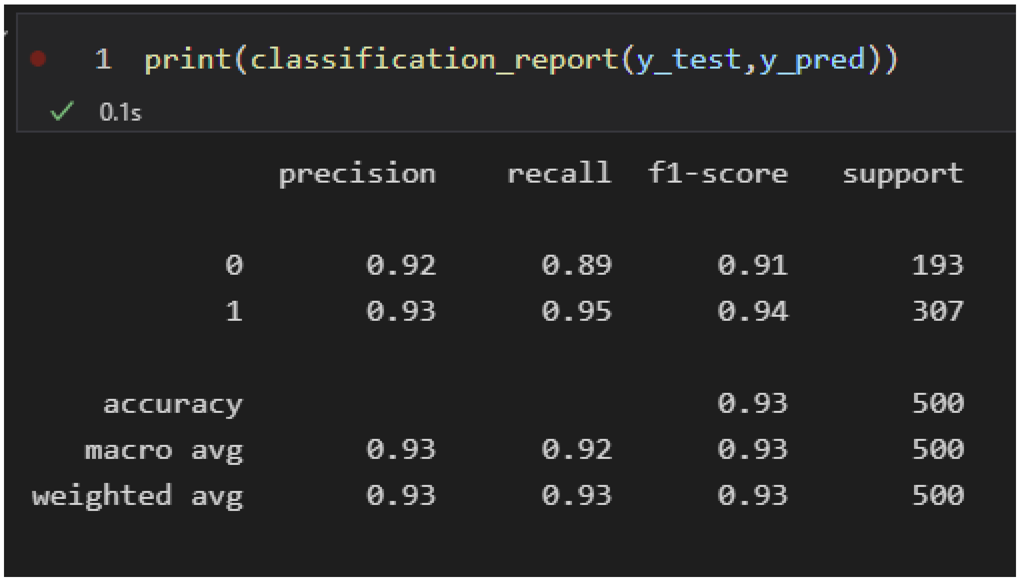
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay
```

- plot_confusion_matrix(log_model, scaled_X_test, y_test, normalize="true") : yüzdelik olarak değer verir



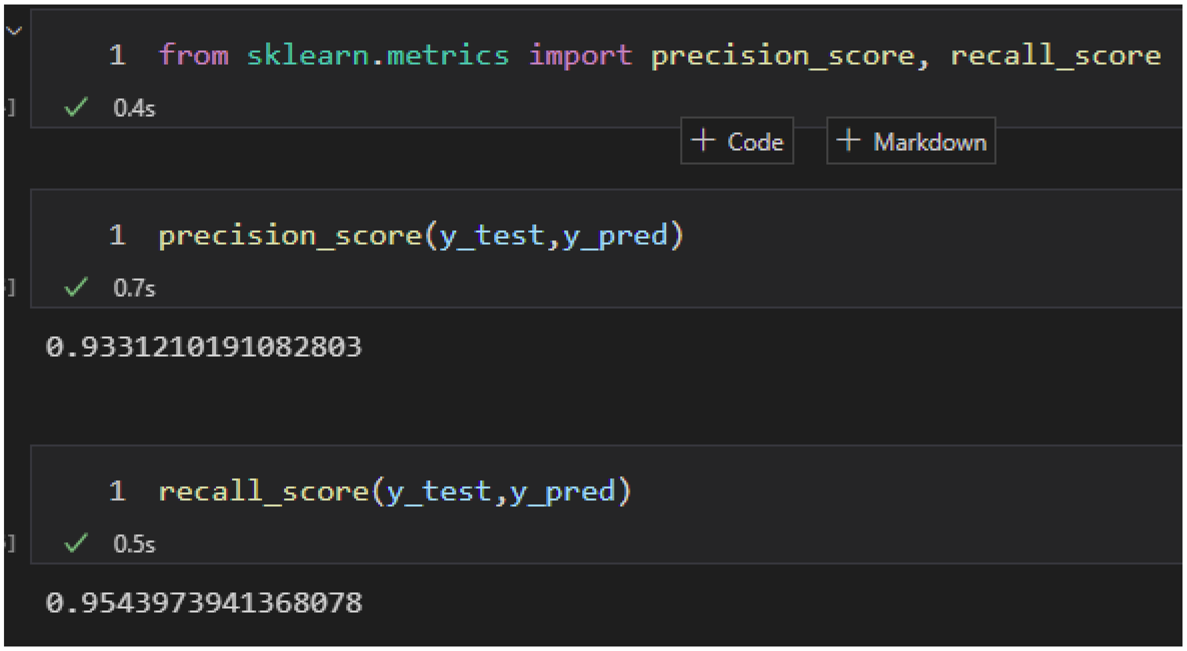
```
plot_confusion_matrix(log_model, scaled_X_test, y_test, normalize="true")
✓ 0.1s
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1b012b9d730>
```

- print(classification_report(y_test,y_pred)) : skorlar.
  Print ile yazdırınca daha düzgün görünüyor



```
print(classification_report(y_test,y_pred))
✓ 0.1s
              precision    recall  f1-score   support

           0       0.92      0.89      0.91       193
           1       0.93      0.95      0.94       307

    accuracy                           0.93       500
   macro avg       0.93      0.92      0.93       500
weighted avg       0.93      0.93      0.93       500
```

- precission ve recall score



```
from sklearn.metrics import precision_score, recall_score
```

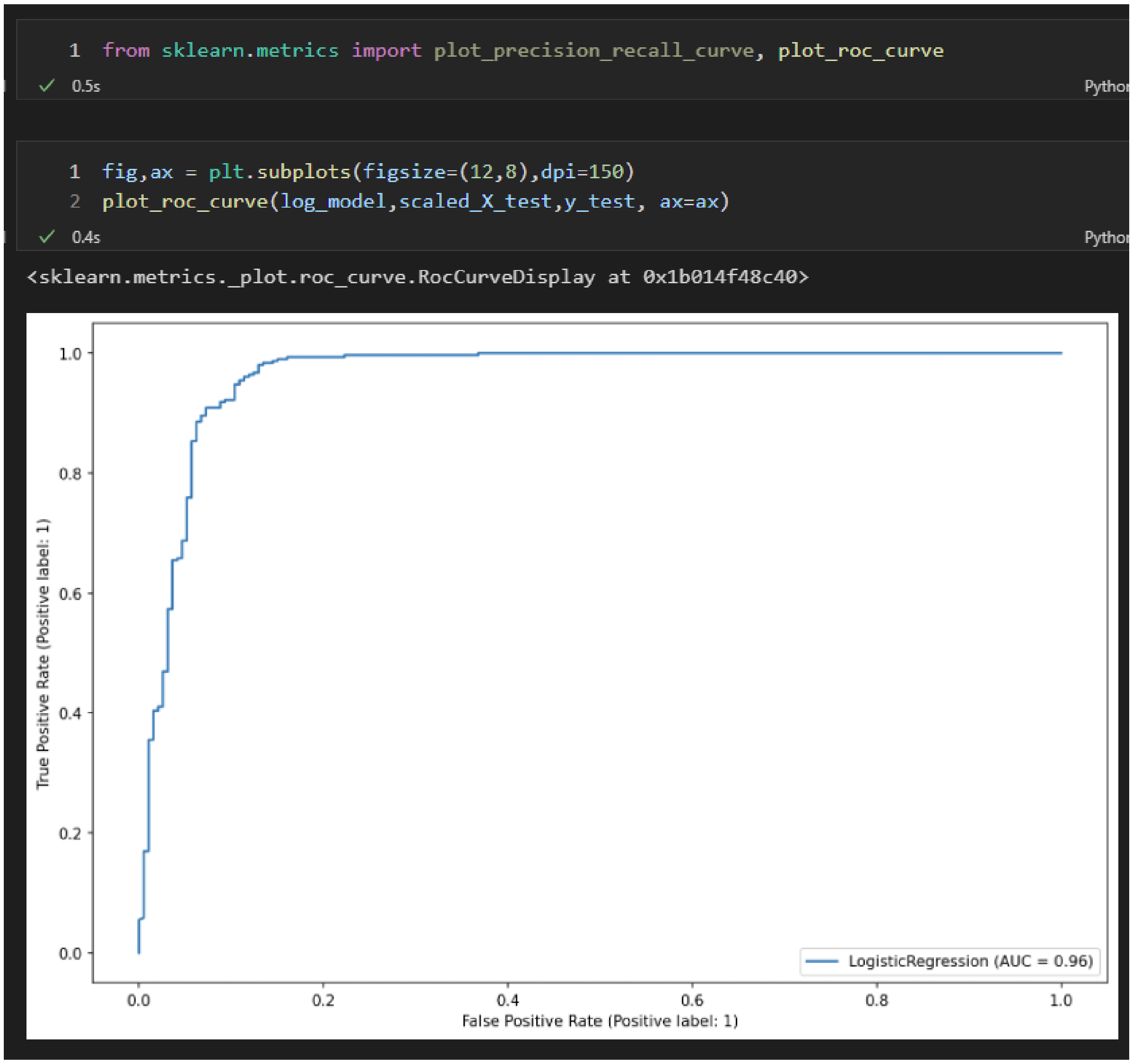
```
precision_score(y_test,y_pred)
```

```
0.9331210191082803
```

```
recall_score(y_test,y_pred)
```

```
0.9543973941368078
```

- ROC curve



```python
from sklearn.metrics import plot_precision_recall_curve, plot_roc_curve
```

✓ 0.5s Pytho

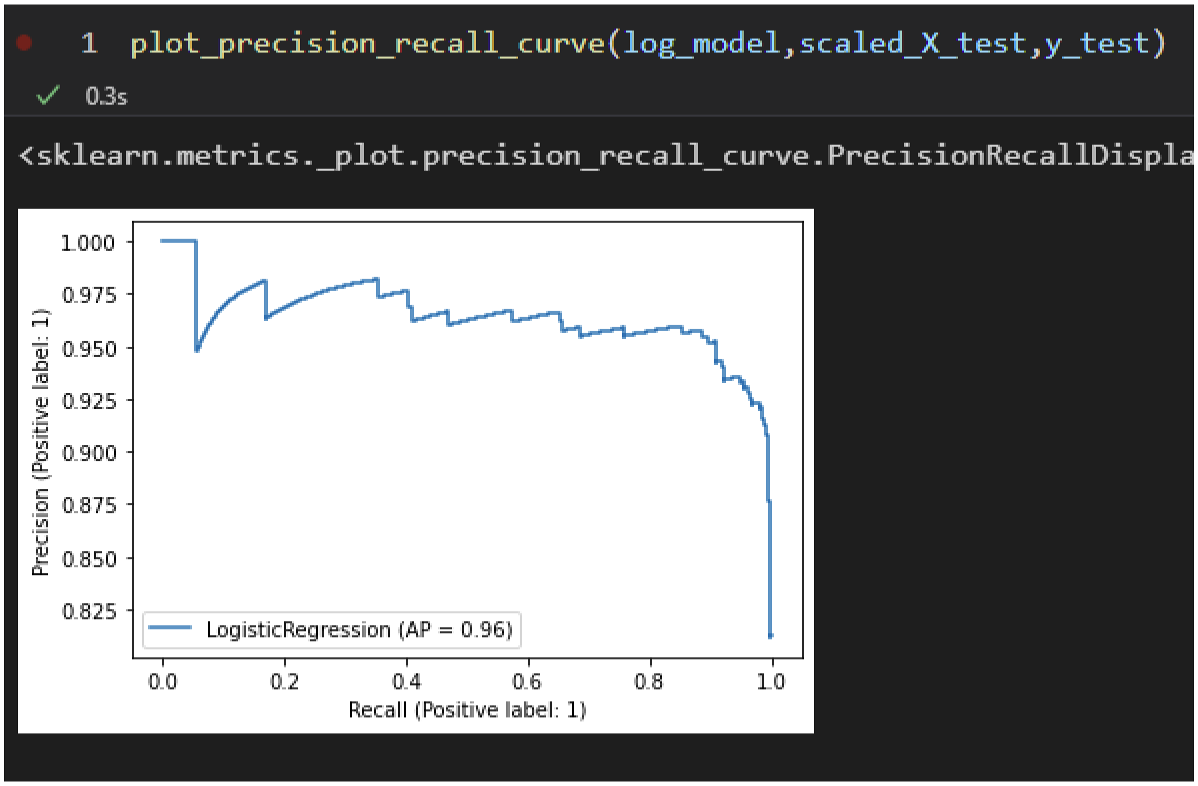
```python
fig,ax = plt.subplots(figsize=(12,8),dpi=150)
plot_roc_curve(log_model,scaled_X_test,y_test, ax=ax)
```

✓ 0.4s Pytho

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x1b014f48c40>
```

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
LogisticRegression (AUC = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0

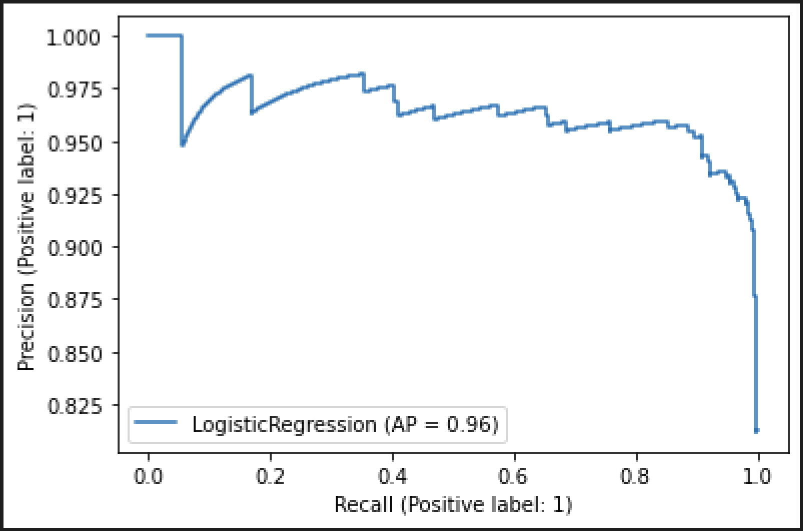
- precision recall curve



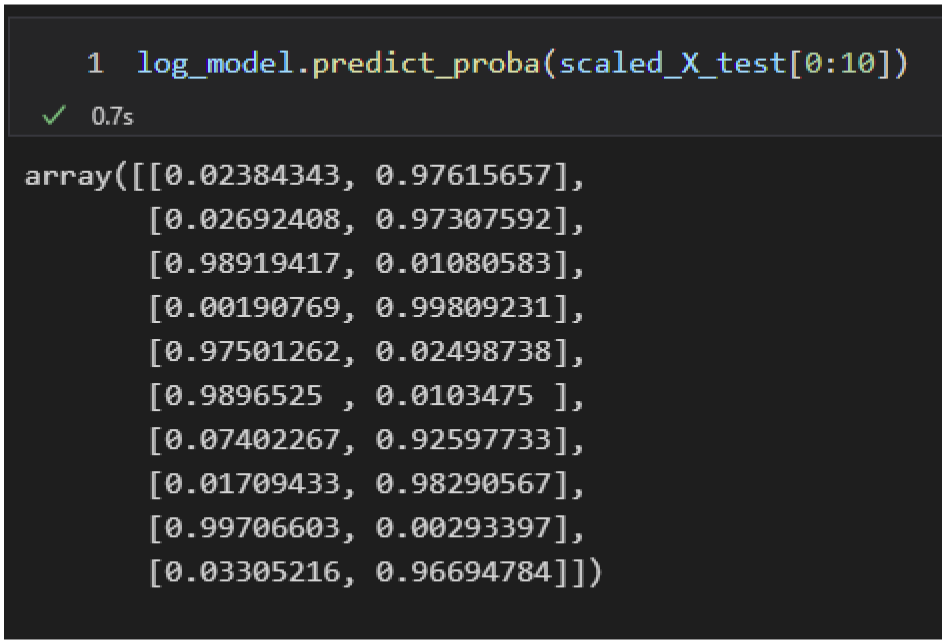
```
plot_precision_recall_curve(log_model,scaled_X_test,y_test)
```

0.3s

```
<sklearn.metrics._plot.precision_recall_curve.PrecisionRecallDispla
```



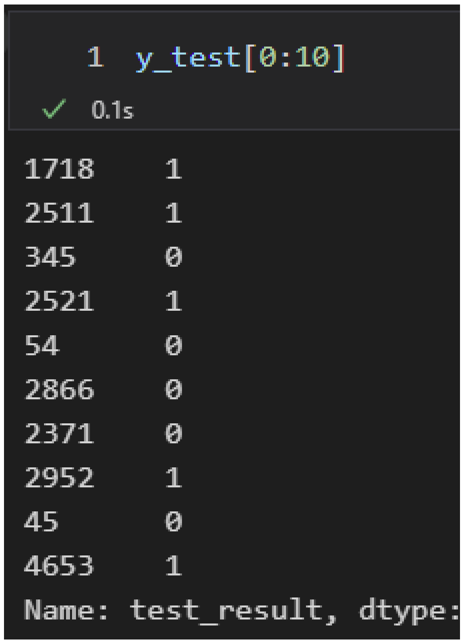
- Probabilities of model's first 10



```
log_model.predict_proba(scaled_X_test[0:10])
```

0.7s

```
array([[0.02384343, 0.97615657],
       [0.02692408, 0.97307592],
       [0.98919417, 0.01080583],
       [0.00190769, 0.99809231],
       [0.97501262, 0.02498738],
       [0.9896525 , 0.0103475 ],
       [0.07402267, 0.92597733],
       [0.01709433, 0.98290567],
       [0.99706603, 0.00293397],
       [0.03305216, 0.96694784]])
```



```
y_test[0:10]
```

✓ 0.1s

```
1718    1
2511    1
345     0
2521    1
54      0
2866    0
2371    0
2952    1
45      0
4653    1
Name: test_result, dtype:
```