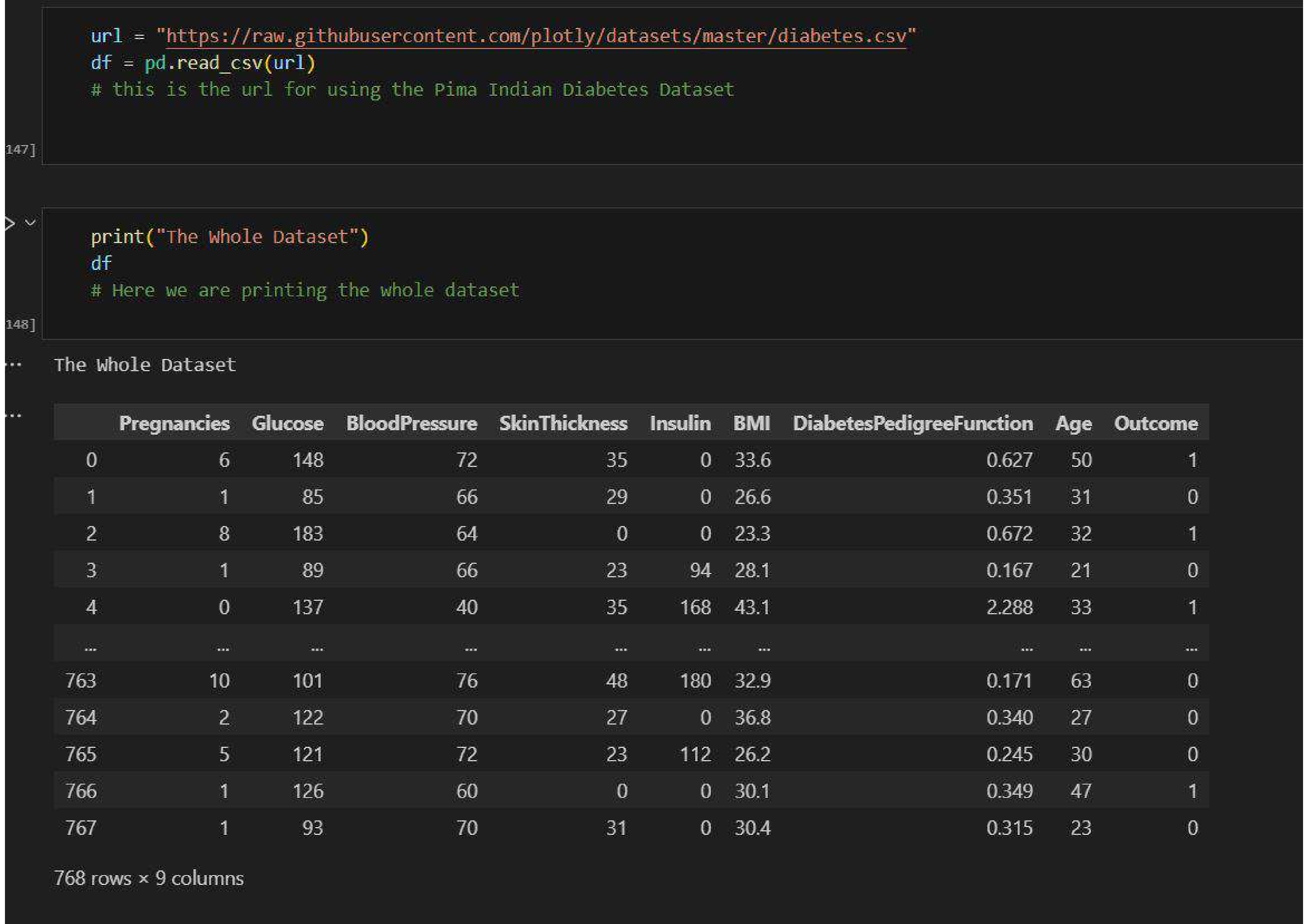


```
url = "https://raw.githubusercontent.com/plotly/datasets/master/diabetes.csv"
df = pd.read_csv(url)
# this is the url for using the Pima Indian Diabetes Dataset
```

147]

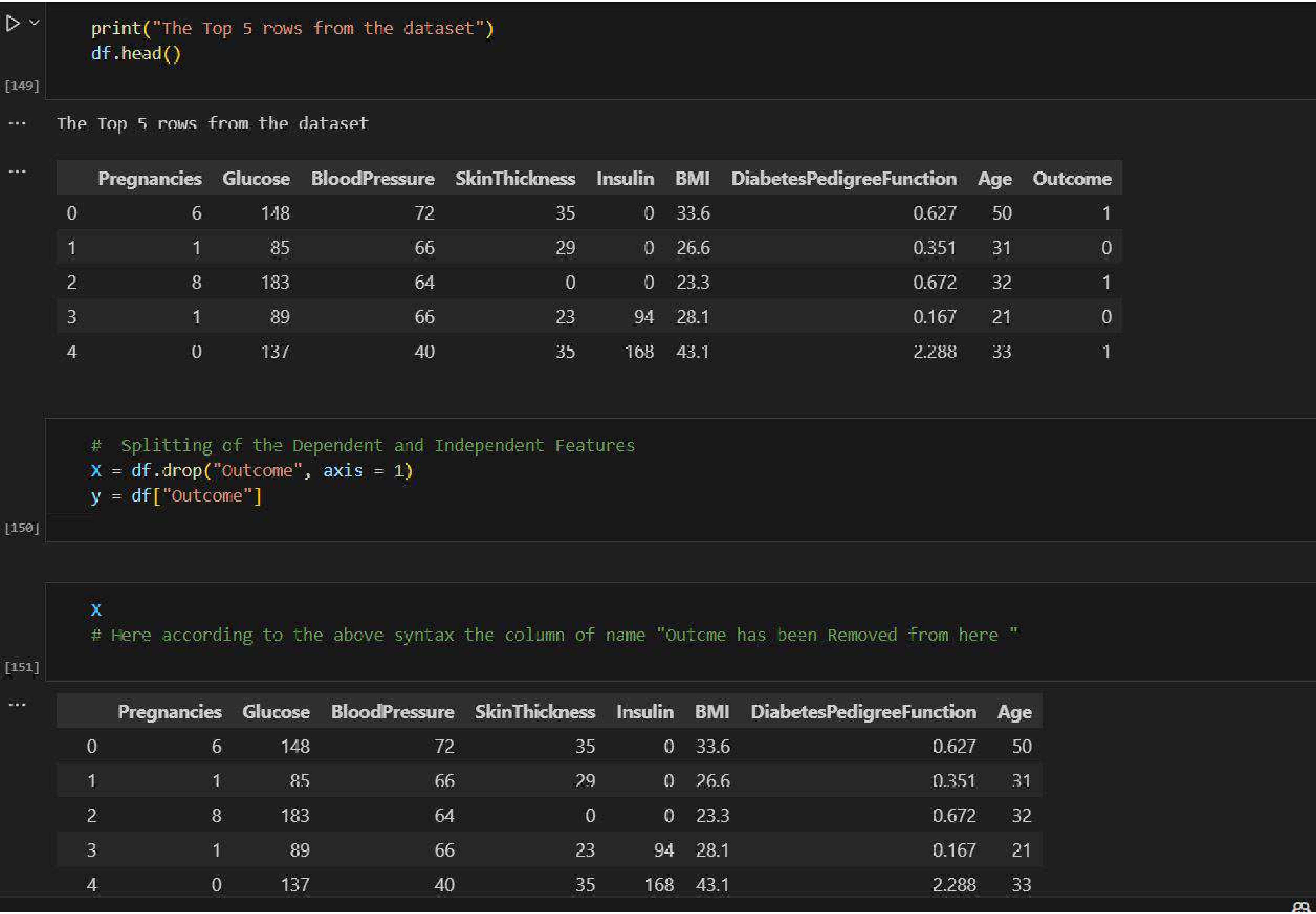
```
print("The Whole Dataset")
df
# Here we are printing the whole dataset
```

148]

```
The Whole Dataset
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns



```
print("The Top 5 rows from the dataset")
df.head()
```

[149]

```
The Top 5 rows from the dataset
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

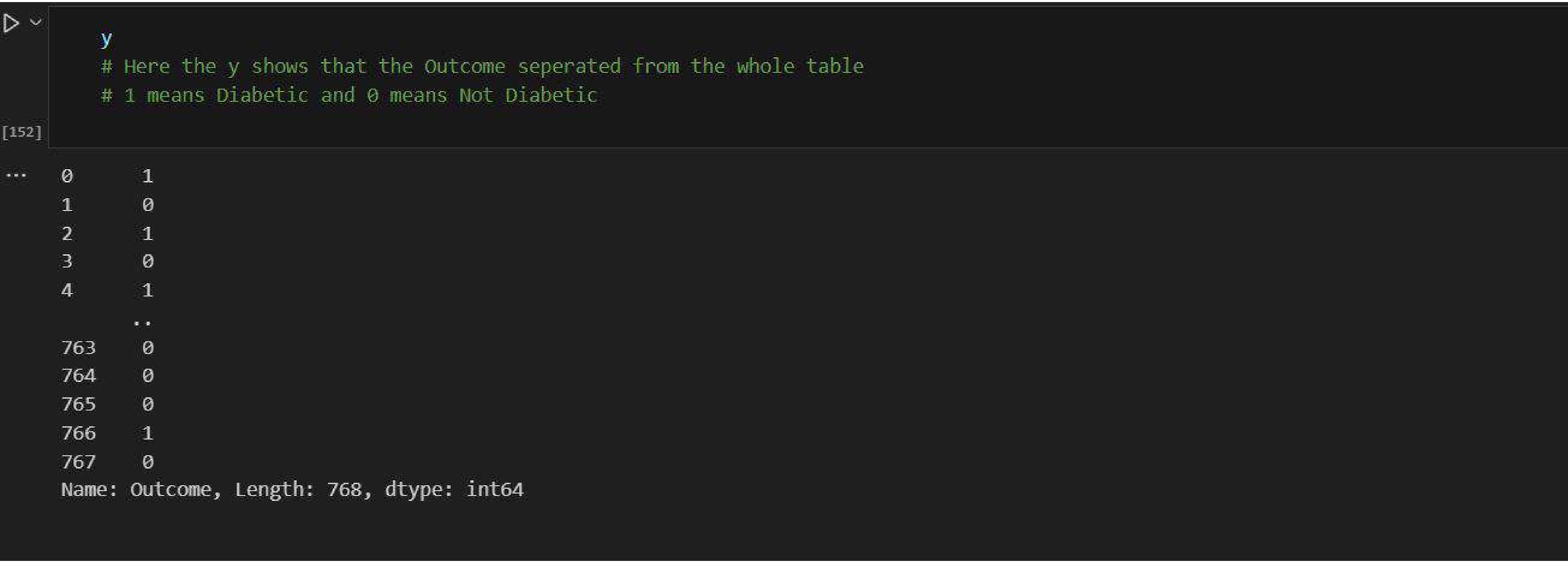
```
#  Splitting of the Dependent and Independent Features
X = df.drop("Outcome", axis = 1)
y = df["Outcome"]
```

[150]

```
X
# Here according to the above syntax the column of name "Outcme has been Removed from here "
```

[151]

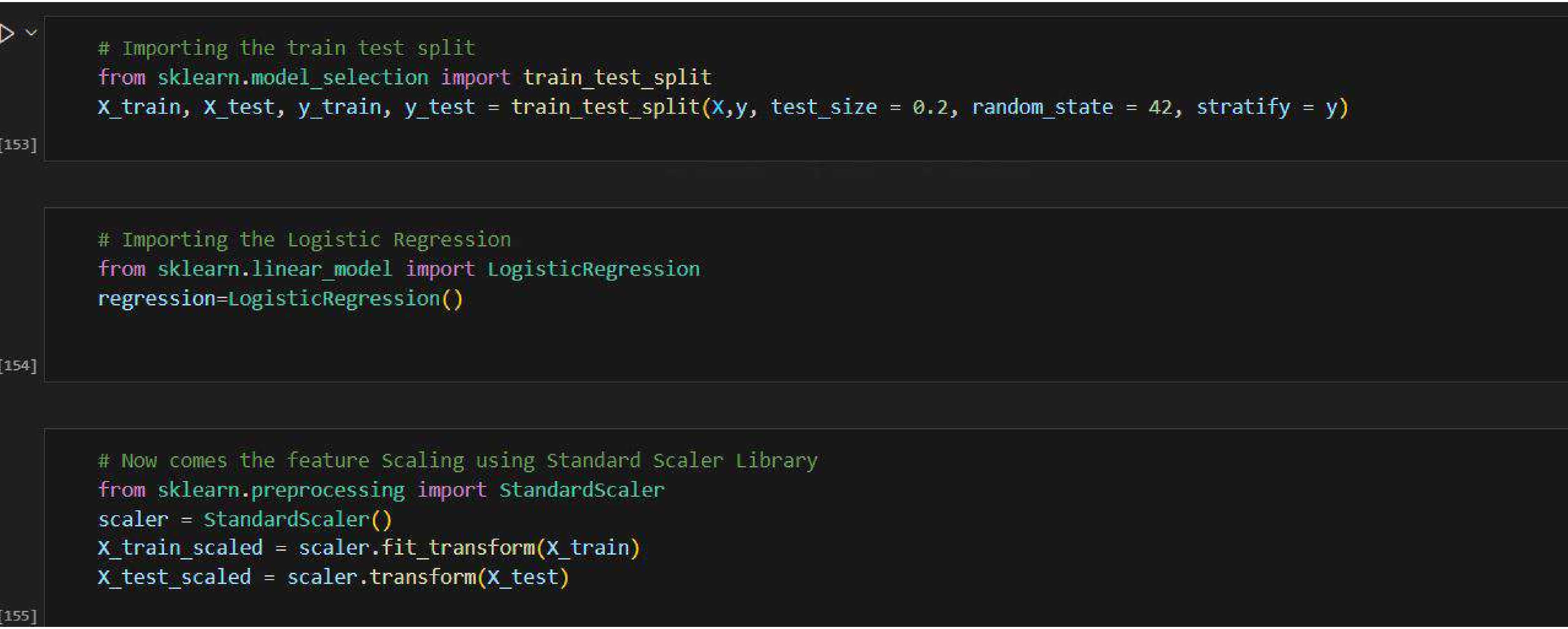
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33



[152]

```
y
# Here the y shows that the Outcome seperated from the whole table
# 1 means Diabetic and 0 means Not Diabetic
```

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```



```
# Importing the train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, random_state = 42, stratify = y)
```

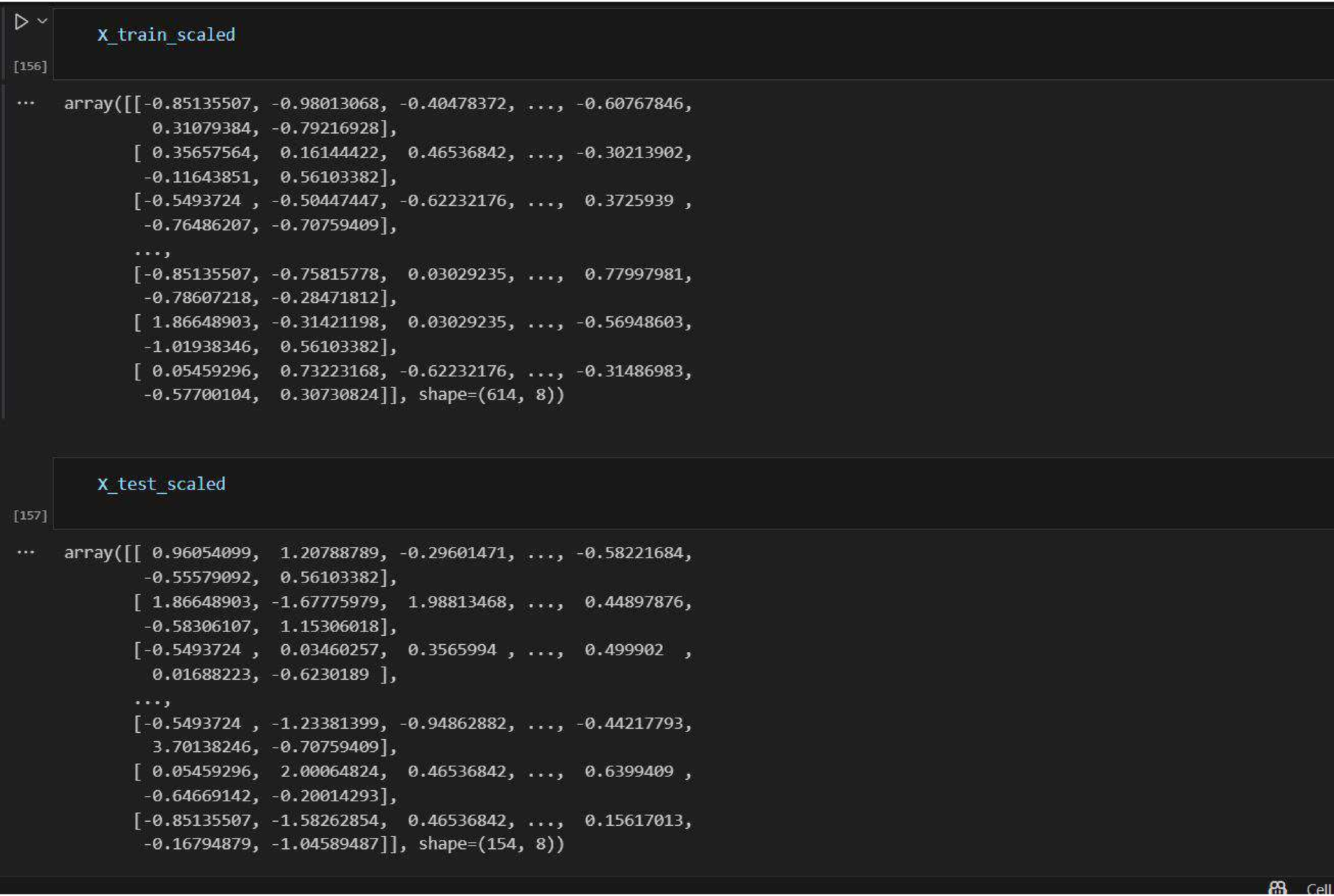
[153]

```
# Importing the Logistic Regression
from sklearn.linear_model import LogisticRegression
regression=LogisticRegression()
```

[154]

```
# Now comes the feature Scaling using Standard Scaler Library
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[155]

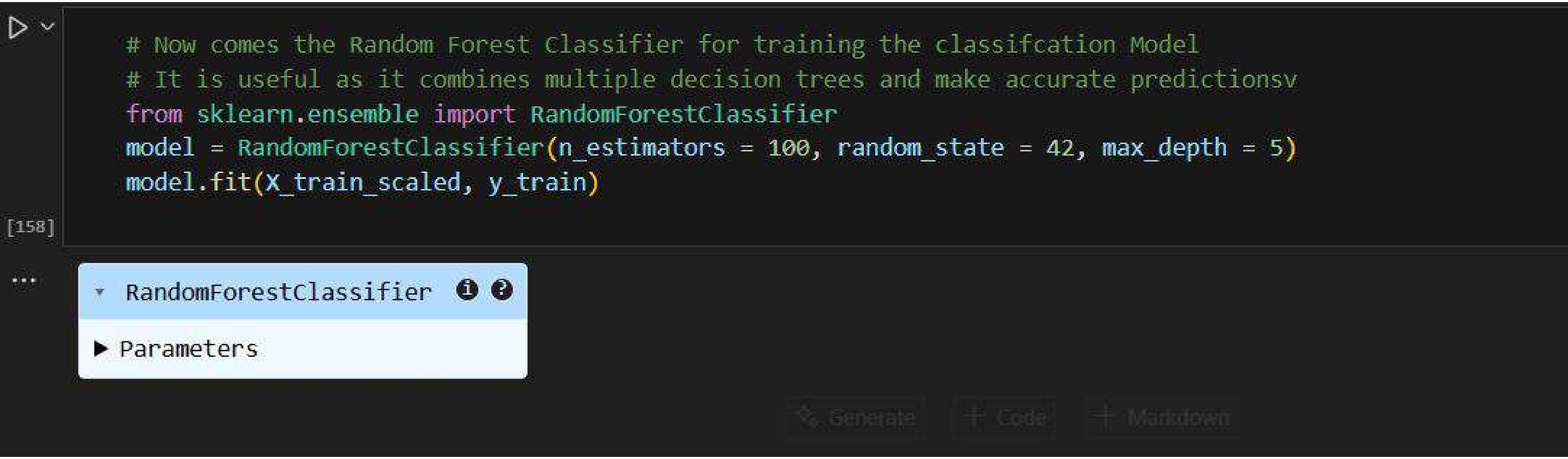


```
X_train_scaled
```

```
array([[-0.85135507, -0.98013068, -0.40478372, ..., -0.60767846,
         0.31079384, -0.79216928],
       [ 0.35657564,  0.16144422,  0.46536842, ..., -0.30213902,
        -0.11643851,  0.56103382],
       [-0.5493724 , -0.50447447, -0.62232176, ...,  0.3725939 ,
        -0.76486207, -0.70759409],
       ...,
       [-0.85135507, -0.75815778,  0.03029235, ...,  0.77997981,
        -0.78607218, -0.28471812],
       [ 1.86648903, -0.31421198,  0.03029235, ..., -0.56948603,
        -1.01938346,  0.56103382],
       [ 0.05459296,  0.73223168, -0.62232176, ..., -0.31486983,
        -0.57700104,  0.30730824]], shape=(614, 8))
```

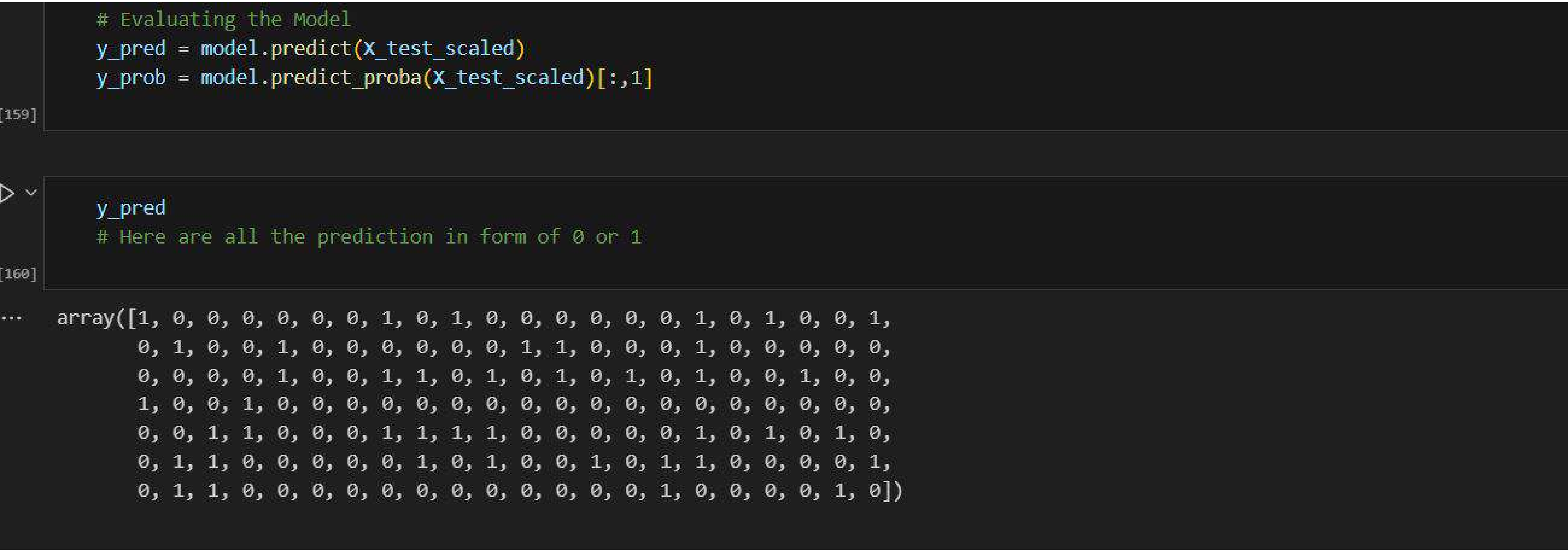
```
X_test_scaled
```

```
array([[ 0.96054099,  1.20788789, -0.29601471, ..., -0.58221684,
        -0.55579092,  0.56103382],
       [ 1.86648903, -1.67775979,  1.98813468, ...,  0.44897876,
        -0.58306107,  1.15306018],
       [-0.5493724 ,  0.03460257,  0.3565994 , ...,  0.499902  ,
         0.01688223, -0.6230189 ],
       ...,
       [-0.5493724 , -1.23381399, -0.94862882, ..., -0.44217793,
         3.70138246, -0.70759409],
       [ 0.05459296,  2.00064824,  0.46536842, ...,  0.6399409 ,
        -0.64669142, -0.20014293],
       [-0.85135507, -1.58262854,  0.46536842, ...,  0.15617013,
        -0.16794879, -1.04589487]], shape=(154, 8))
```



```
# Now comes the Random Forest Classifier for training the classifcation Model
# It is useful as it combines multiple decision trees and make accurate predictionsv
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators = 100, random_state = 42, max_depth = 5)
model.fit(X_train_scaled, y_train)
```

```
▾ RandomForestClassifier
▶ Parameters
```



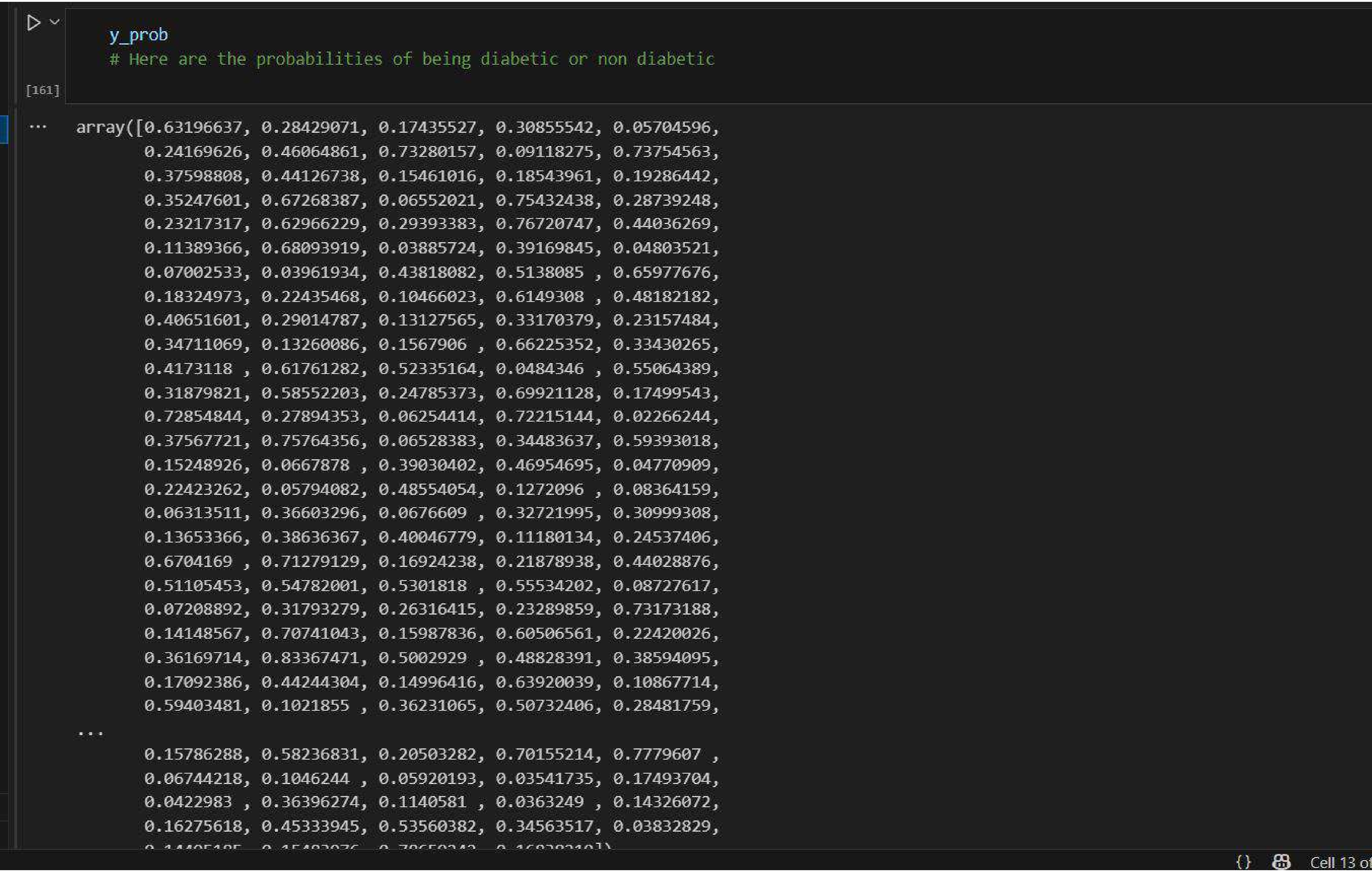
```
# Evaluating the Model
y_pred = model.predict(X_test_scaled)
y_prob = model.predict_proba(X_test_scaled)[:,1]
```

[159]

```
y_pred
# Here are all the prediction in form of 0 or 1
```

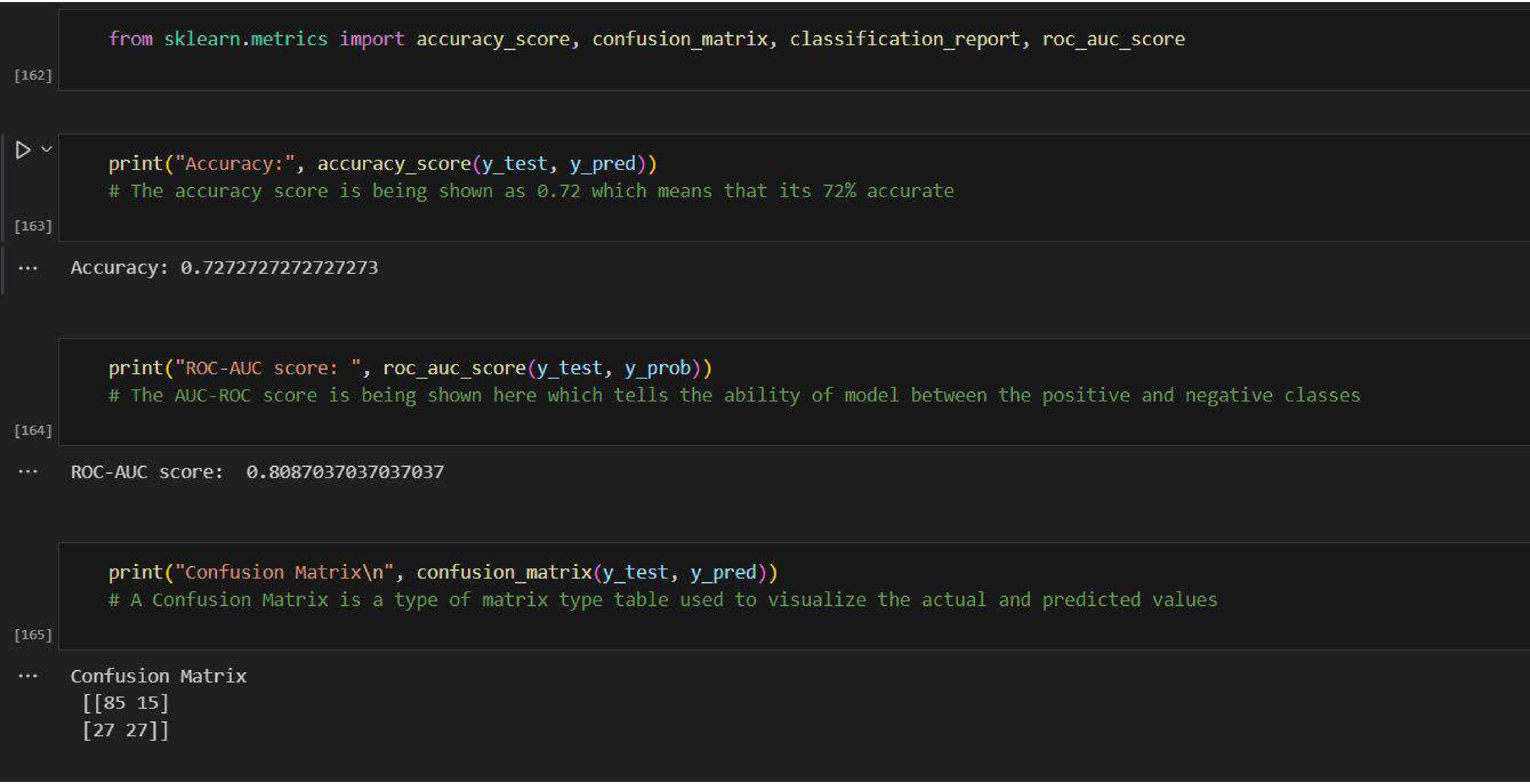
[160]

```
array([1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0])
```



```
y_prob
# Here are the probabilities of being diabetic or non diabetic
```

```
array([0.63196637, 0.28429071, 0.17435527, 0.30855542, 0.05704596,
       0.24169626, 0.46064861, 0.73280157, 0.09118275, 0.73754563,
       0.37598808, 0.44126738, 0.15461016, 0.18543961, 0.19286442,
       0.35247601, 0.67268387, 0.06552021, 0.75432438, 0.28739248,
       0.23217317, 0.62966229, 0.29393383, 0.76720747, 0.44036269,
       0.11389366, 0.68093919, 0.03885724, 0.39169845, 0.04803521,
       0.07002533, 0.03961934, 0.43818082, 0.5138085 , 0.65977676,
       0.18324973, 0.22435468, 0.10466023, 0.6149308 , 0.48182182,
       0.40651601, 0.29014787, 0.13127565, 0.33170379, 0.23157484,
       0.34711069, 0.13260086, 0.1567906 , 0.66225352, 0.33430265,
       0.4173118 , 0.61761282, 0.52335164, 0.0484346 , 0.55064389,
       0.31879821, 0.58552203, 0.24785373, 0.69921128, 0.17499543,
       0.72854844, 0.27894353, 0.06254414, 0.72215144, 0.02266244,
       0.37567721, 0.75764356, 0.06528383, 0.34483637, 0.59393018,
       0.15248926, 0.0667878 , 0.39030402, 0.46954695, 0.04770909,
       0.22423262, 0.05794082, 0.48554054, 0.1272096 , 0.08364159,
       0.06313511, 0.36603296, 0.0676609 , 0.32721995, 0.30999308,
       0.13653366, 0.38636367, 0.40046779, 0.11180134, 0.24537406,
       0.6704169 , 0.71279129, 0.16924238, 0.21878938, 0.44028876,
       0.51105453, 0.54782001, 0.5301818 , 0.55534202, 0.08727617,
       0.07208892, 0.31793279, 0.26316415, 0.23289859, 0.73173188,
       0.14148567, 0.70741043, 0.15987836, 0.60506561, 0.22420026,
       0.36169714, 0.83367471, 0.5002929 , 0.48828391, 0.38594095,
       0.17092386, 0.44244304, 0.14996416, 0.63920039, 0.10867714,
       0.59403481, 0.1021855 , 0.36231065, 0.50732406, 0.28481759,
...
       0.15786288, 0.58236831, 0.20503282, 0.70155214, 0.7779607 ,
       0.06744218, 0.1046244 , 0.05920193, 0.03541735, 0.17493704,
       0.0422983 , 0.36396274, 0.1140581 , 0.0363249 , 0.14326072,
       0.16275618, 0.45333945, 0.53560382, 0.34563517, 0.03832829,
       0.14495185, 0.45482076, 0.78650242, 0.16828210])
```



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

[162]

```
print("Accuracy:", accuracy_score(y_test, y_pred))
# The accuracy score is being shown as 0.72 which means that its 72% accurate
```

[163]

```
Accuracy: 0.7272727272727273
```

```
print("ROC-AUC score: ", roc_auc_score(y_test, y_prob))
# The AUC-ROC score is being shown here which tells the ability of model between the positive and negative classes
```

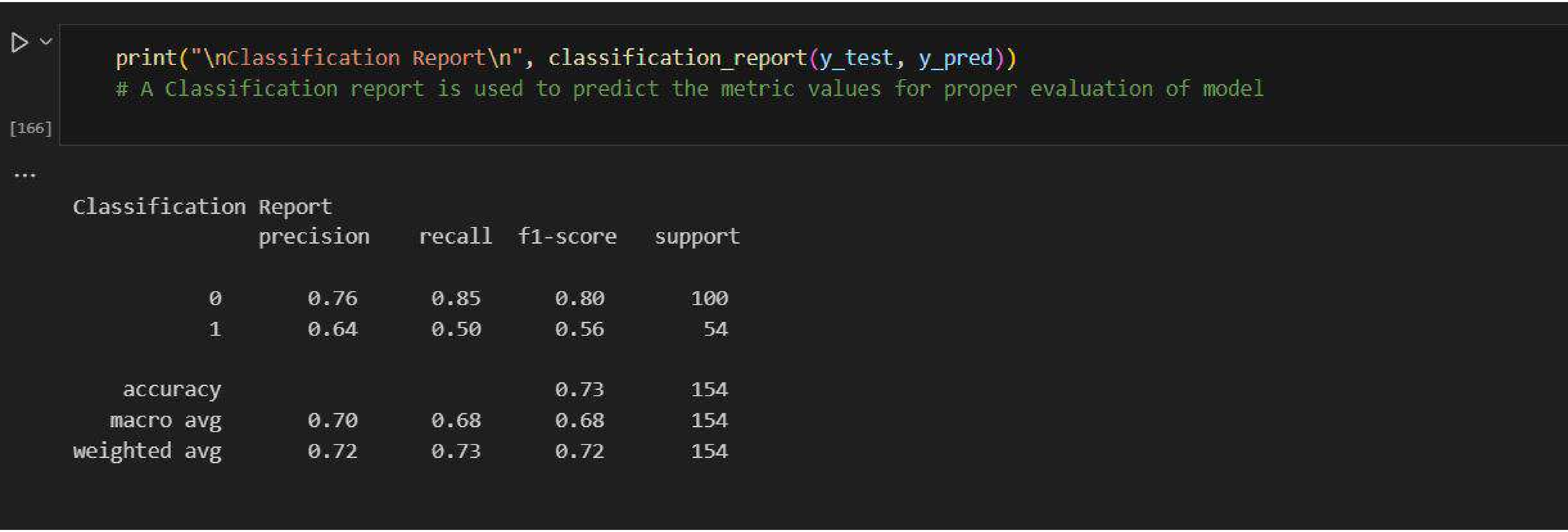
[164]

```
ROC-AUC score:  0.8087037037037037
```

```
print("Confusion Matrix\n", confusion_matrix(y_test, y_pred))
# A Confusion Matrix is a type of matrix type table used to visualize the actual and predicted values
```

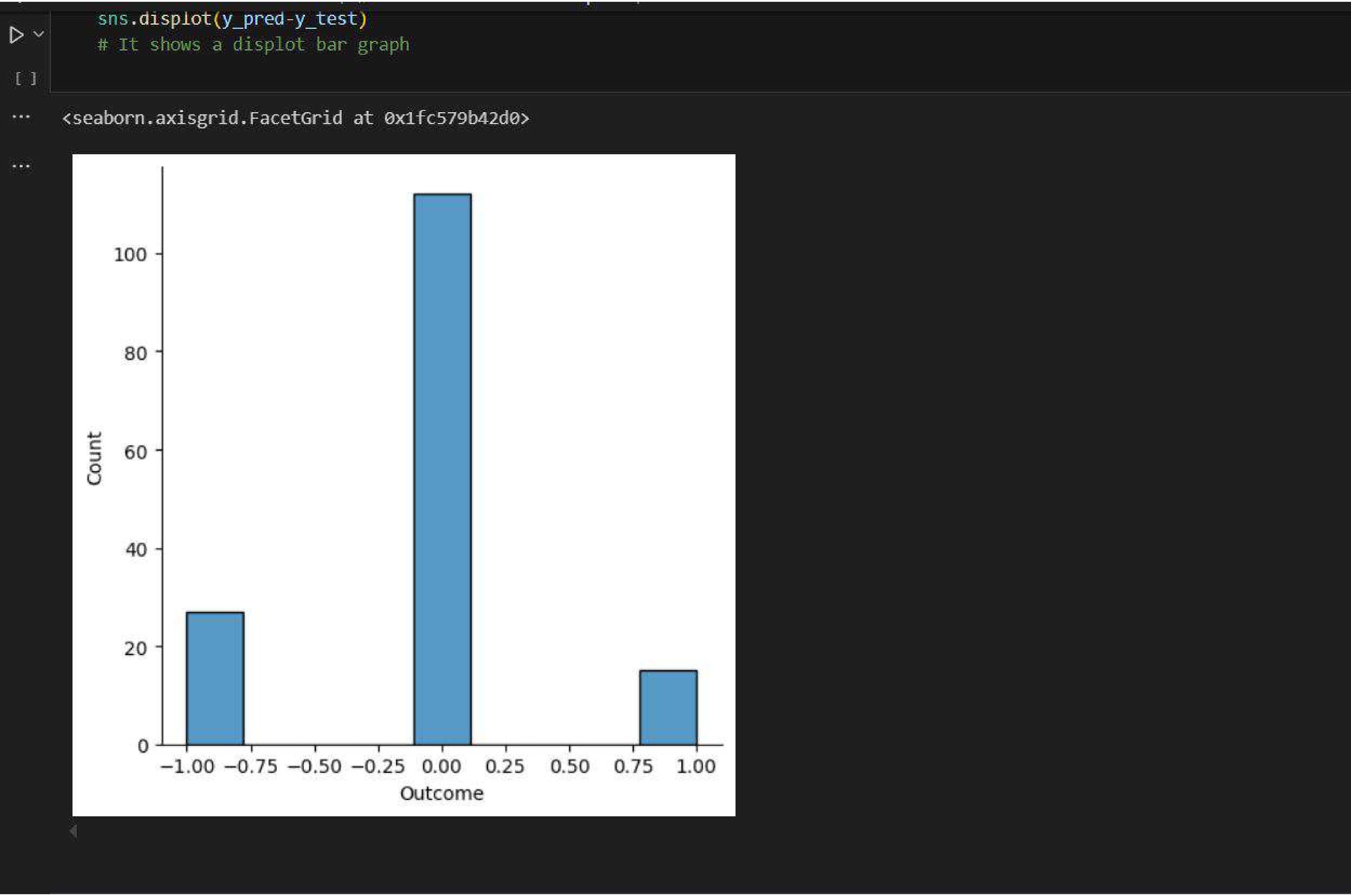
[165]

```
Confusion Matrix
 [[85 15]
 [27 27]]
```



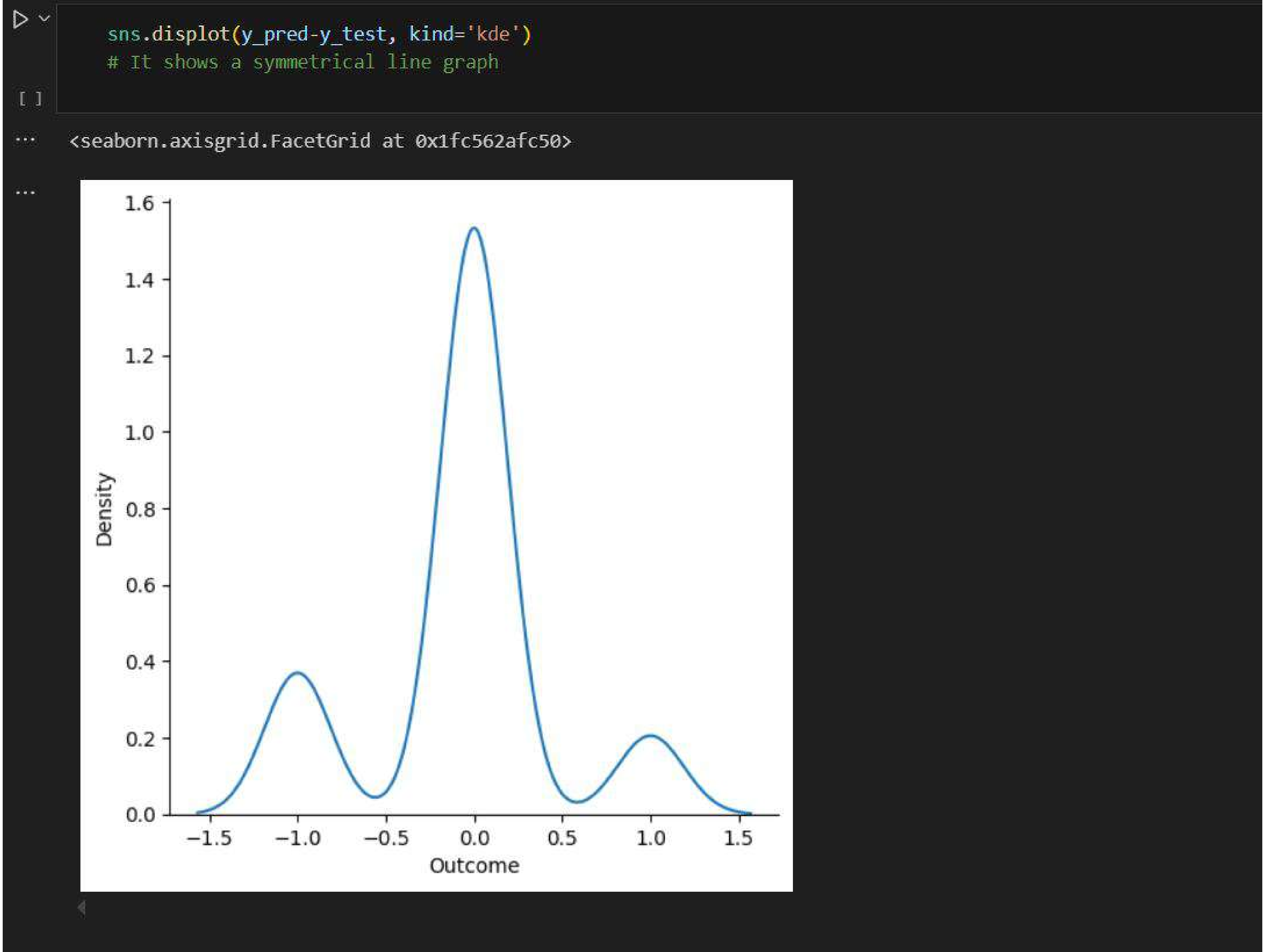
```
print("\nClassification Report\n", classification_report(y_test, y_pred))
# A Classification report is used to predict the metric values for proper evaluation of model
```

```
Classification Report
               precision    recall  f1-score   support

           0       0.76      0.85      0.80       100
           1       0.64      0.50      0.56        54

    accuracy                           0.73       154
   macro avg       0.70      0.68      0.68       154
weighted avg       0.72      0.73      0.72       154
```



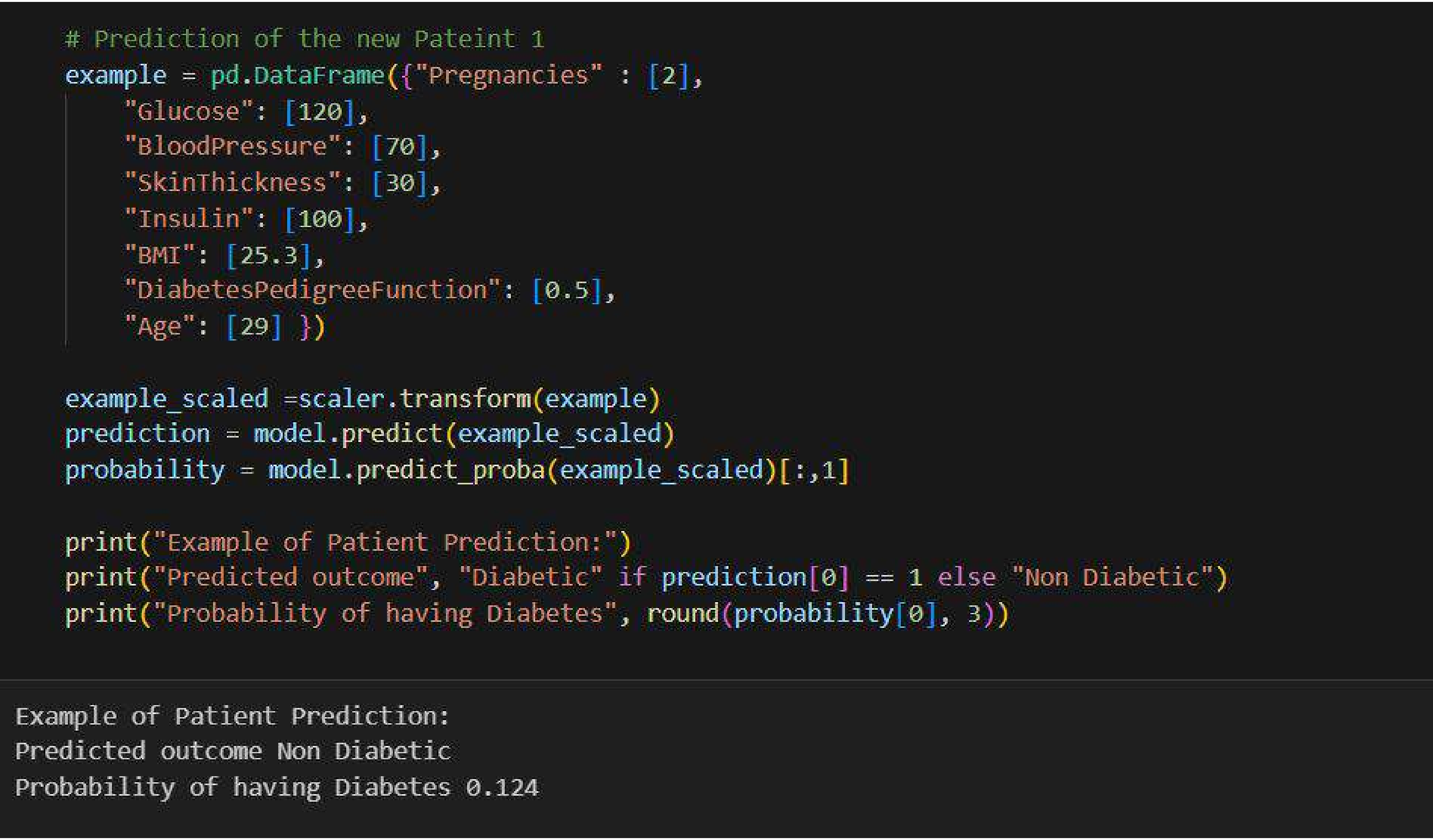
```
sns.displot(y_pred-y_test)
# It shows a displot bar graph
```

```
<seaborn.axisgrid.FacetGrid at 0x1fc579b42d0>
```



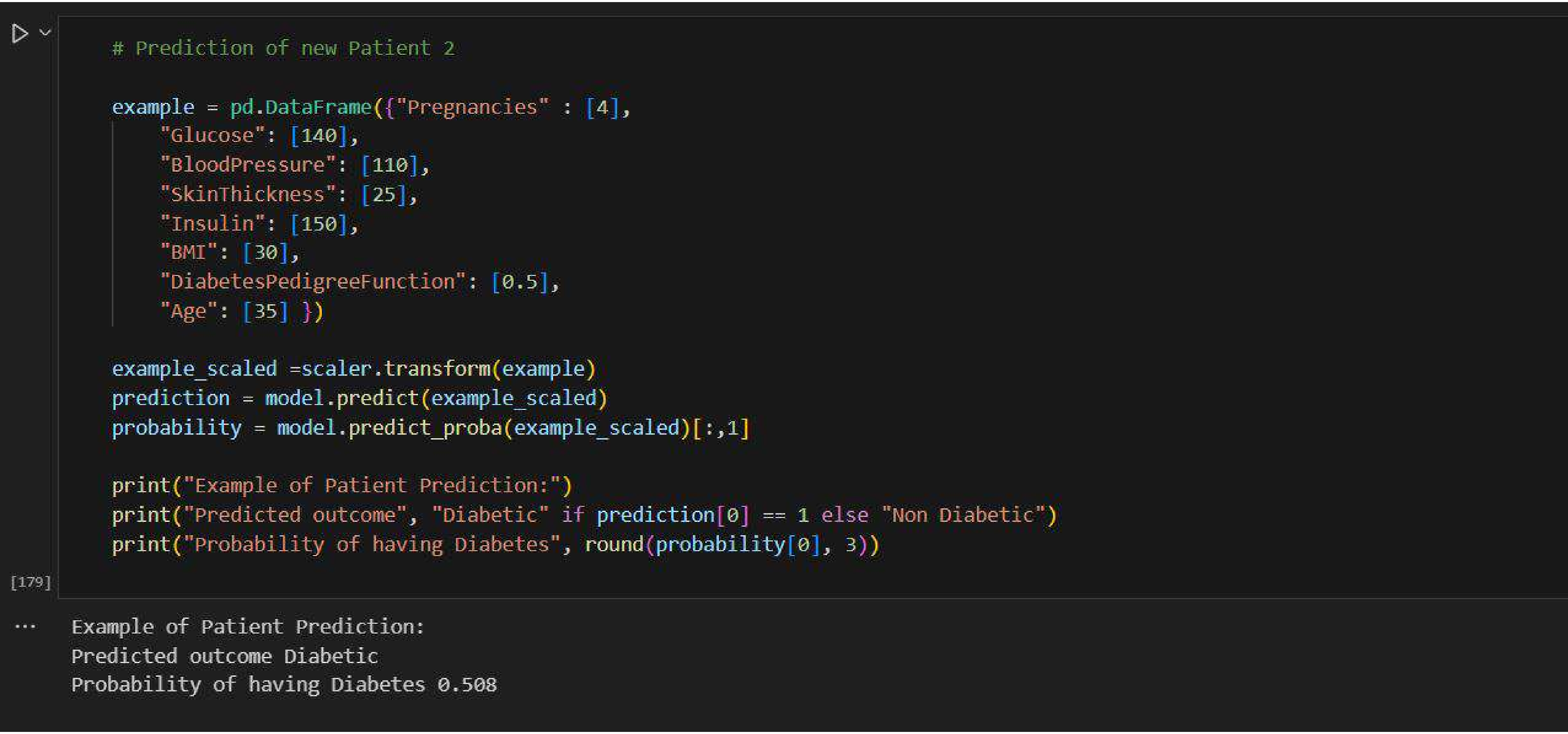
```
sns.displot(y_pred-y_test, kind='kde')
# It shows a symmetrical line graph
```

```
<seaborn.axisgrid.FacetGrid at 0x1fc562afc50>
```



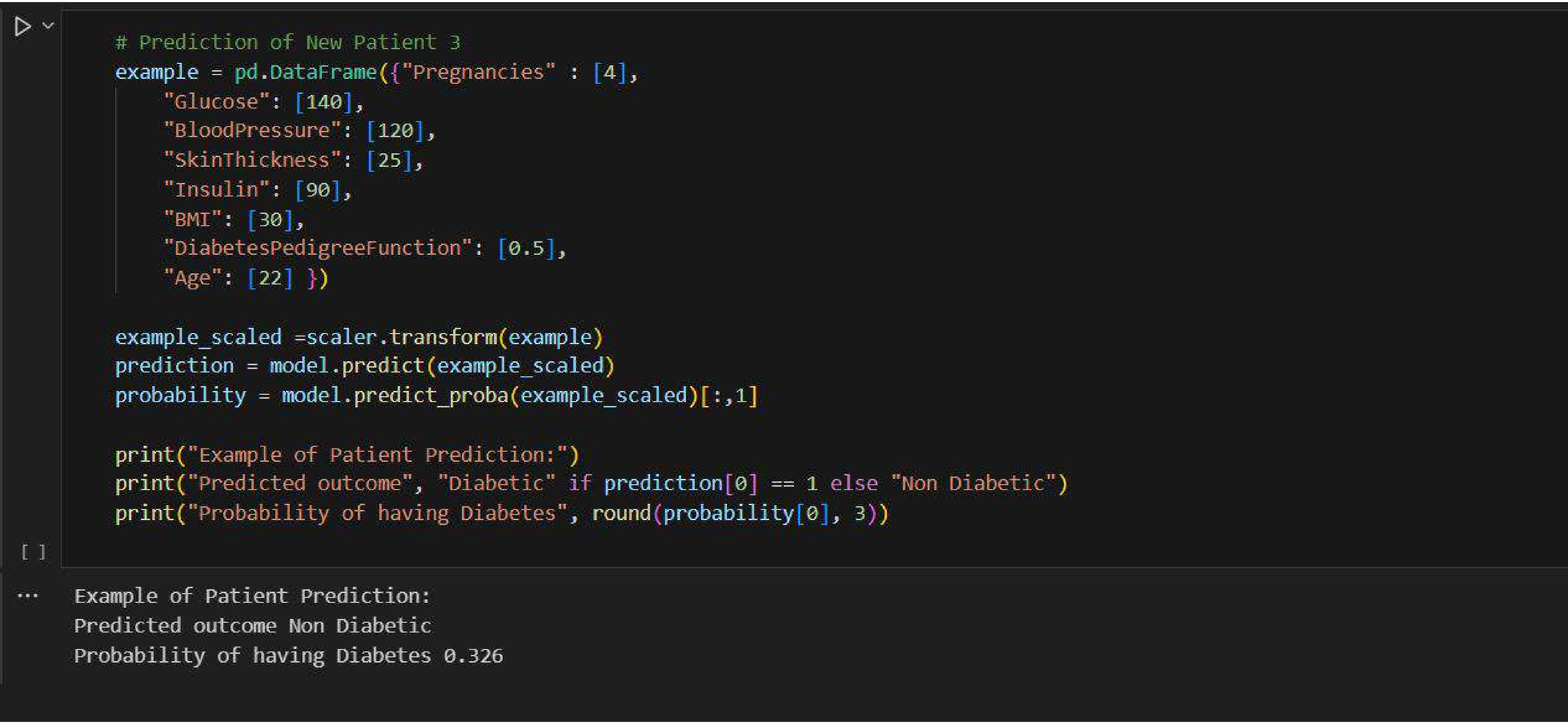
```
# Prediction of the new Pateint 1
example = pd.DataFrame({"Pregnancies" : [2],
    "Glucose": [120],
    "BloodPressure": [70],
    "SkinThickness": [30],
    "Insulin": [100],
    "BMI": [25.3],
    "DiabetesPedigreeFunction": [0.5],
    "Age": [29] })

example_scaled =scaler.transform(example)
prediction = model.predict(example_scaled)
probability = model.predict_proba(example_scaled)[:,1]

print("Example of Patient Prediction:")
print("Predicted outcome", "Diabetic" if prediction[0] == 1 else "Non Diabetic")
print("Probability of having Diabetes", round(probability[0], 3))
```

```
Example of Patient Prediction:
Predicted outcome Non Diabetic
Probability of having Diabetes 0.124
```



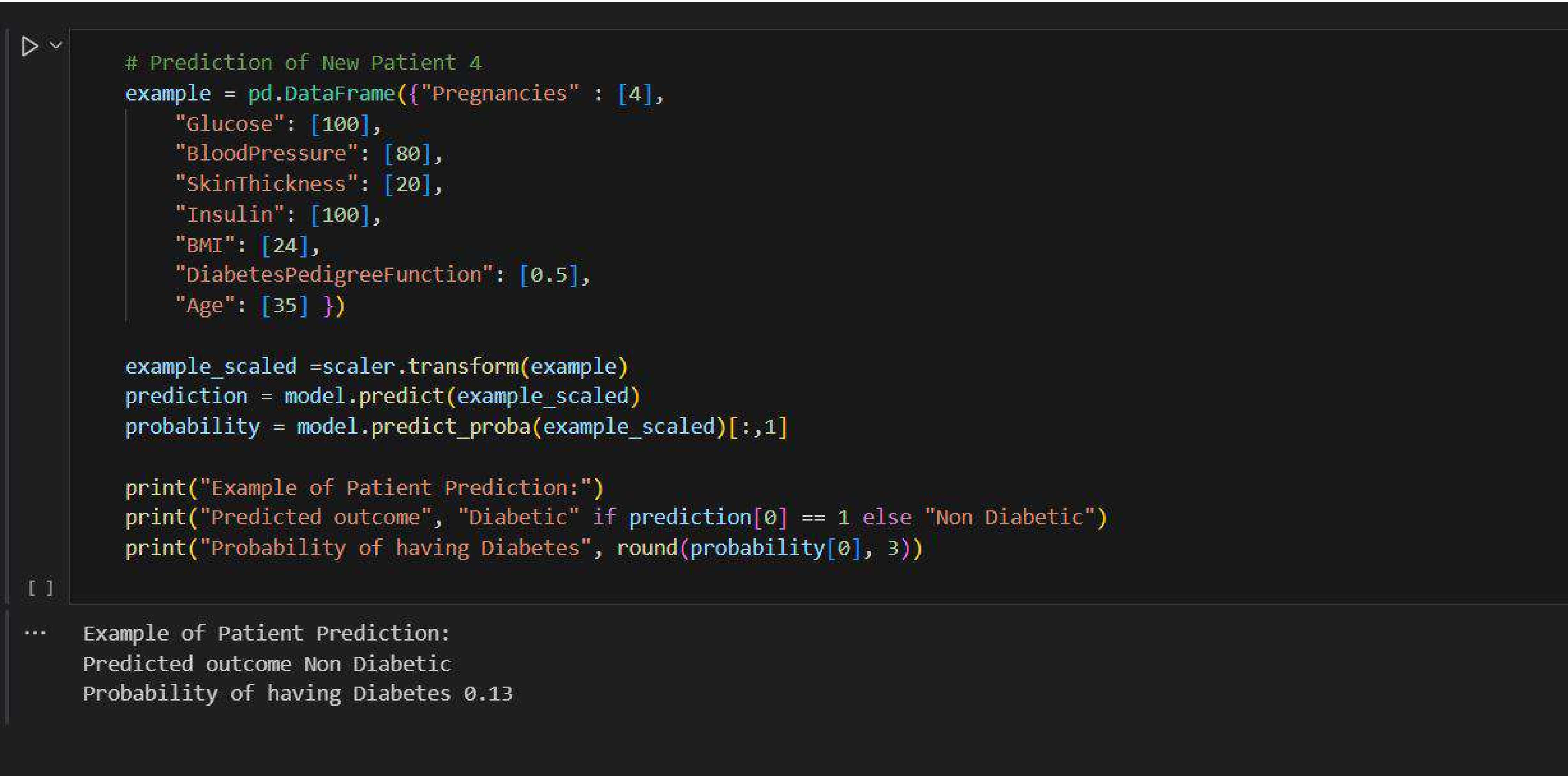
```python
# Prediction of new Patient 2

example = pd.DataFrame({"Pregnancies" : [4],
    "Glucose": [140],
    "BloodPressure": [110],
    "SkinThickness": [25],
    "Insulin": [150],
    "BMI": [30],
    "DiabetesPedigreeFunction": [0.5],
    "Age": [35] })

example_scaled =scaler.transform(example)
prediction = model.predict(example_scaled)
probability = model.predict_proba(example_scaled)[:,1]

print("Example of Patient Prediction:")
print("Predicted outcome", "Diabetic" if prediction[0] == 1 else "Non Diabetic")
print("Probability of having Diabetes", round(probability[0], 3))
```

```
Example of Patient Prediction:
Predicted outcome Diabetic
Probability of having Diabetes 0.508
```



```python
# Prediction of New Patient 3
example = pd.DataFrame({"Pregnancies" : [4],
    "Glucose": [140],
    "BloodPressure": [120],
    "SkinThickness": [25],
    "Insulin": [90],
    "BMI": [30],
    "DiabetesPedigreeFunction": [0.5],
    "Age": [22] })

example_scaled =scaler.transform(example)
prediction = model.predict(example_scaled)
probability = model.predict_proba(example_scaled)[:,1]

print("Example of Patient Prediction:")
print("Predicted outcome", "Diabetic" if prediction[0] == 1 else "Non Diabetic")
print("Probability of having Diabetes", round(probability[0], 3))
```

```
Example of Patient Prediction:
Predicted outcome Non Diabetic
Probability of having Diabetes 0.326
```



```python
# Prediction of New Patient 4
example = pd.DataFrame({"Pregnancies" : [4],
    "Glucose": [100],
    "BloodPressure": [80],
    "SkinThickness": [20],
    "Insulin": [100],
    "BMI": [24],
    "DiabetesPedigreeFunction": [0.5],
    "Age": [35] })

example_scaled =scaler.transform(example)
prediction = model.predict(example_scaled)
probability = model.predict_proba(example_scaled)[:,1]

print("Example of Patient Prediction:")
print("Predicted outcome", "Diabetic" if prediction[0] == 1 else "Non Diabetic")
print("Probability of having Diabetes", round(probability[0], 3))
```

```
Example of Patient Prediction:
Predicted outcome Non Diabetic
Probability of having Diabetes 0.13
```