MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL TECHNICAL UNIVERSITY OF UKRAINE
" IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE"

Artem Volokyta

Software Security

Lab Work 2

Machine Learning Heart Disease Prediction

kulubecioglu Mehmet

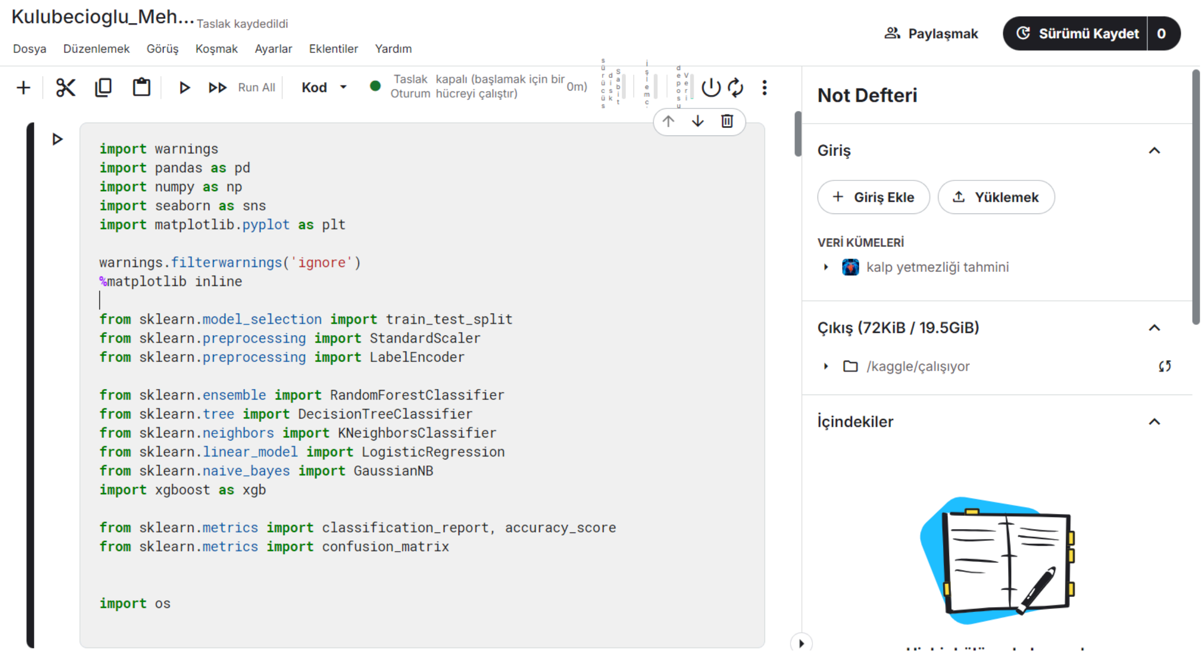
Class Number 12

IM-14 FIOT

Kyiv
IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE
2024

Machine Learning Heart Disease Prediction Report

1. Importing Libraries

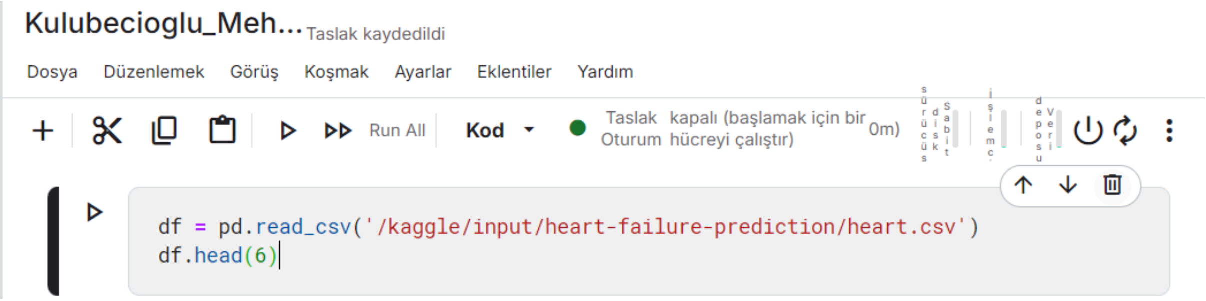


```python
import warnings
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

warnings.filterwarnings('ignore')
%matplotlib inline

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
import xgboost as xgb

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


import os
```

Explanation:

In this section, I imported all the necessary libraries for data analysis, preprocessing, and machine learning. The libraries include `pandas` for data manipulation, `seaborn` and `matplotlib` for data visualization, and several classification algorithms from `sklearn`. Additionally, I imported metrics like `accuracy_score` and `confusion_matrix` to evaluate the models.

2. Reading the Dataset

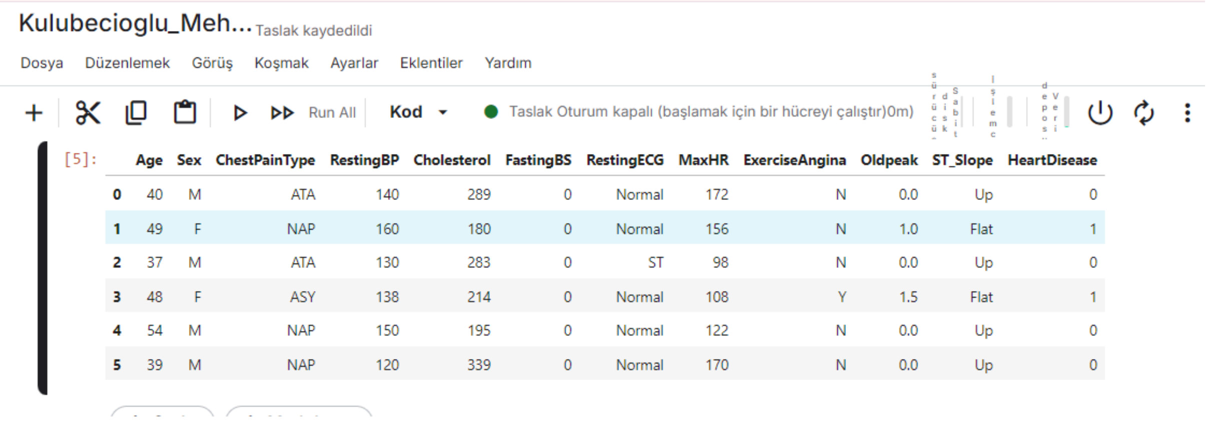

Kulubecioglu_Meh... Taslak kaydedildi
Dosya
Düzenlemek
Görüş
Koşmak
Ayarlar
Eklentiler
Yardım
Run All
Kod
Taslak Oturum kapalı (başlamak için bir hücreyi çalıştır) 0m)

```
df = pd.read_csv('/kaggle/input/heart-failure-prediction/heart.csv')
df.head(6)
```

Explanation:

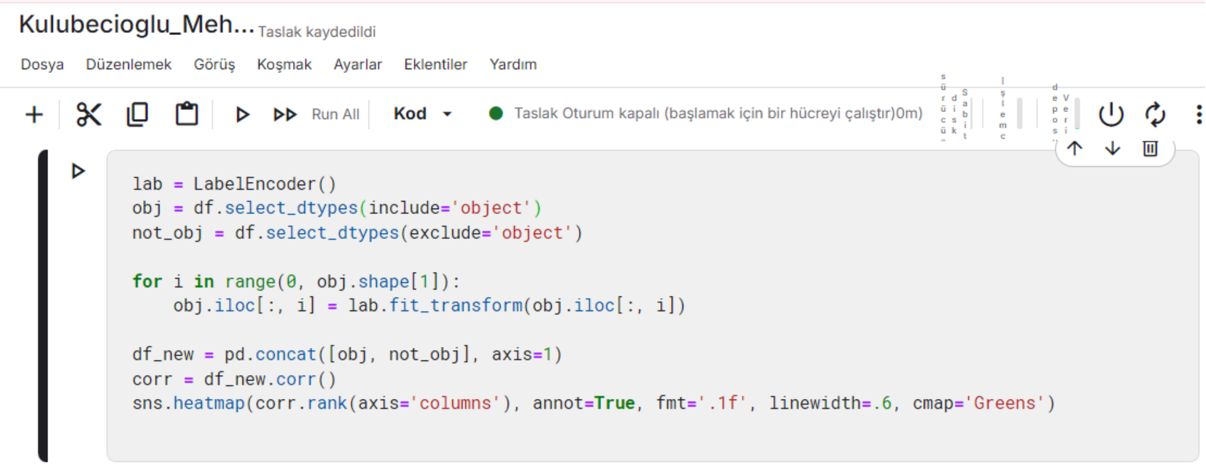
In this step, I loaded the heart disease dataset using `pandas`. The `df.head(6)` function displays the first 6 rows of the dataset, which allows me to inspect the data.

Output:


Kulubecioglu_Meh... Taslak kaydedildi
Dosya
Düzenlemek
Görüş
Koşmak
Ayarlar
Eklentiler
Yardım
Run All
Kod
Taslak Oturum kapalı (başlamak için bir hücreyi çalıştır)0m)
[5]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0
5	39	M	NAP	120	339	0	Normal	170	N	0.0	Up	0

3. Label Encoding and Correlation Analysis

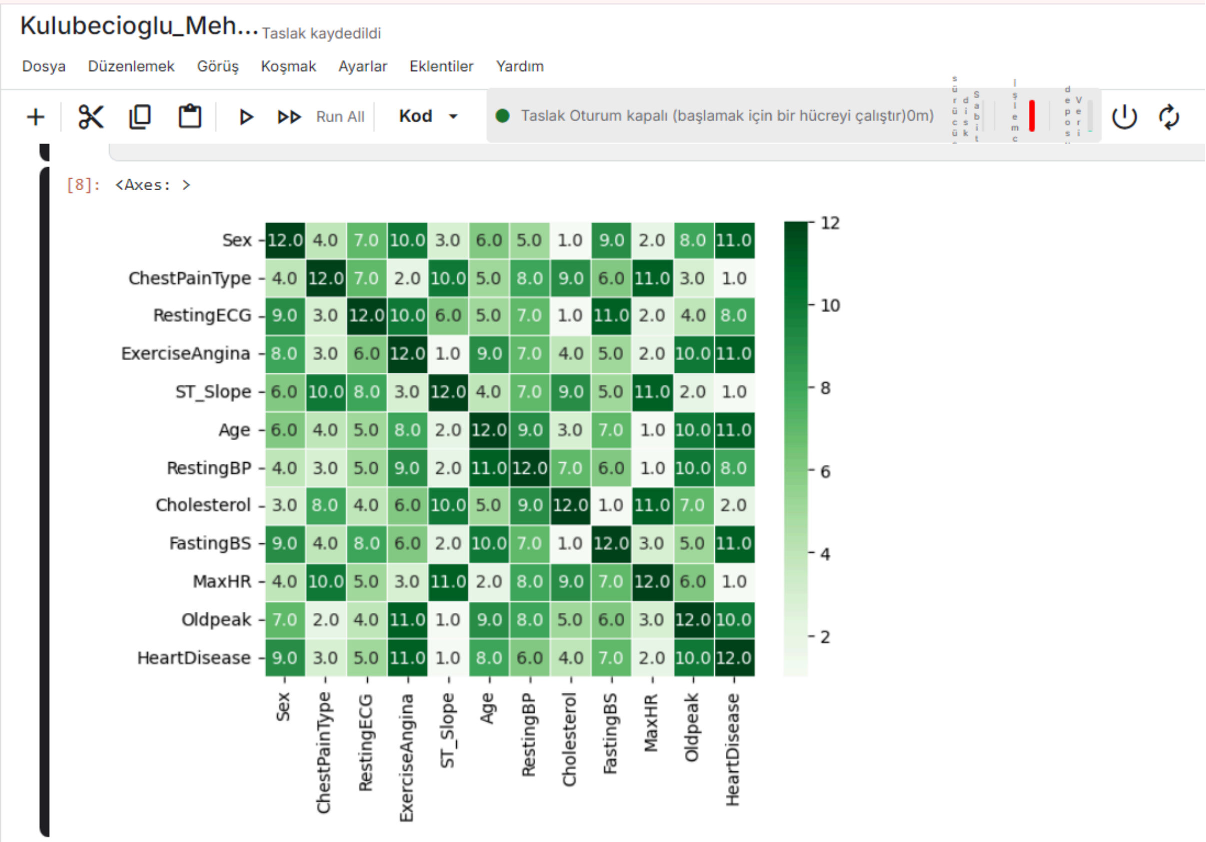

Kulubecioglu_Meh... Taslak kaydedildi
Dosya
Düzenlemek
Görüş
Koşmak
Ayarlar
Eklentiler
Yardım
Run All
Kod
Taslak Oturum kapalı (başlamak için bir hücreyi çalıştır)0m)

```
lab = LabelEncoder()
obj = df.select_dtypes(include='object')
not_obj = df.select_dtypes(exclude='object')

for i in range(0, obj.shape[1]):
    obj.iloc[:, i] = lab.fit_transform(obj.iloc[:, i])

df_new = pd.concat([obj, not_obj], axis=1)
corr = df_new.corr()
sns.heatmap(corr.rank(axis='columns'), annot=True, fmt='.1f', linewidth=.6, cmap='Greens')
```

Explanation:

In this part, I used `LabelEncoder` to convert categorical variables into numeric form. After that, I concatenated the encoded categorical data with the numerical columns to create a new dataset (`df_new`). Then, I visualized the correlation matrix of the dataset using a heatmap to understand the relationships between the features.

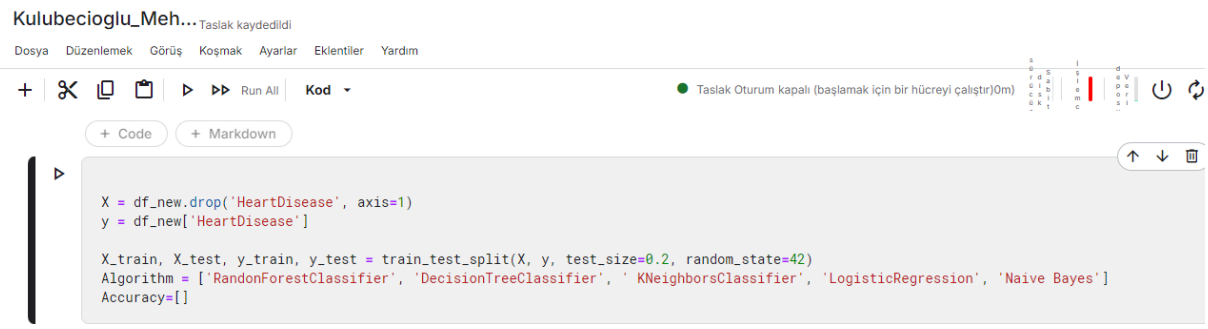
Output:


Kulubecioglu_Meh... Taslak kaydedildi
Dosya
Düzenlemek
Görüş
Koşmak
Ayarlar
Eklentiler
Yardım
Run All
Kod
Taslak Oturum kapalı (başlamak için bir hücreyi çalıştır)0m)
12
10
8
6
4
2

```
[8]: <Axes: >
```

	Sex	ChestPainType	RestingECG	ExerciseAngina	ST_Slope	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
Sex	12.0	4.0	7.0	10.0	3.0	6.0	5.0	1.0	9.0	2.0	8.0	11.0
ChestPainType	4.0	12.0	7.0	2.0	10.0	5.0	8.0	9.0	6.0	11.0	3.0	1.0
RestingECG	9.0	3.0	12.0	10.0	6.0	5.0	7.0	1.0	11.0	2.0	4.0	8.0
ExerciseAngina	8.0	3.0	6.0	12.0	1.0	9.0	7.0	4.0	5.0	2.0	10.0	11.0
ST_Slope	6.0	10.0	8.0	3.0	12.0	4.0	7.0	9.0	5.0	11.0	2.0	1.0
Age	6.0	4.0	5.0	8.0	2.0	12.0	9.0	3.0	7.0	1.0	10.0	11.0
RestingBP	4.0	3.0	5.0	9.0	2.0	11.0	12.0	7.0	6.0	1.0	10.0	8.0
Cholesterol	3.0	8.0	4.0	6.0	10.0	5.0	9.0	12.0	1.0	11.0	7.0	2.0
FastingBS	9.0	4.0	8.0	6.0	2.0	10.0	7.0	1.0	12.0	3.0	5.0	11.0
MaxHR	4.0	10.0	5.0	3.0	11.0	2.0	8.0	9.0	7.0	12.0	6.0	1.0
Oldpeak	7.0	2.0	4.0	11.0	1.0	9.0	8.0	5.0	6.0	3.0	12.0	10.0
HeartDisease	9.0	3.0	5.0	11.0	1.0	8.0	6.0	4.0	7.0	2.0	10.0	12.0

4. Splitting Data into Train and Test Sets

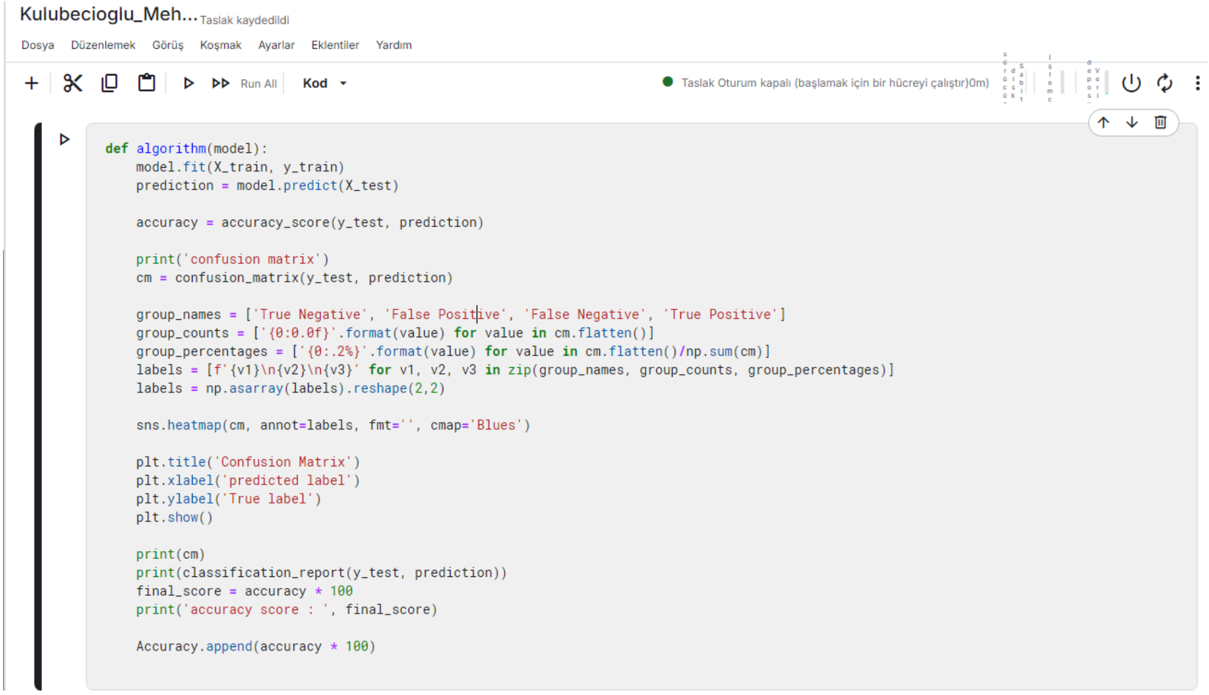

Kulubecioglu_Meh... Taslak kaydedildi
Dosya
Düzenlemek
Görüş
Koşmak
Ayarlar
Eklentiler
Yardım
Run All
Kod
Taslak Oturum kapalı (başlamak için bir hücreyi çalıştır)0m)
+ Code
+ Markdown

```
X = df_new.drop('HeartDisease', axis=1)
y = df_new['HeartDisease']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
Algorithm = ['RandonForestClassifier', 'DecisionTreeClassifier', ' KNeighborsClassifier', 'LogisticRegression', 'Naive Bayes']
Accuracy=[]
```

Explanation:

Here, I split the dataset into features (`X`) and target variable (`y`). The dataset is then split into training and testing sets with an 80-20 ratio using `train_test_split`. I also initialized an empty list (`Accuracy`) to store the accuracy of each model.

5. Function for Model Training and Evaluation


Kulubecioglu_Meh... Taslak kaydedildi
Dosya
Düzenlemek
Görüş
Koşmak
Ayarlar
Eklentiler
Yardım
Run All
Kod
Taslak Oturum kapalı (başlamak için bir hücreyi çalıştır)0m)

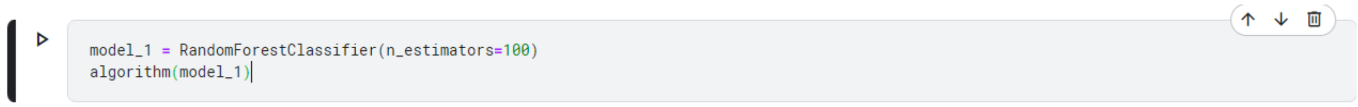
```python
def algorithm(model):
    model.fit(X_train, y_train)
    prediction = model.predict(X_test)

    accuracy = accuracy_score(y_test, prediction)

    print('confusion matrix')
    cm = confusion_matrix(y_test, prediction)

    group_names = ['True Negative', 'False Positive', 'False Negative', 'True Positive']
    group_counts = ['{0:0.0f}'.format(value) for value in cm.flatten()]
    group_percentages = ['{0:.2%}'.format(value) for value in cm.flatten()/np.sum(cm)]
    labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in zip(group_names, group_counts, group_percentages)]
    labels = np.asarray(labels).reshape(2,2)

    sns.heatmap(cm, annot=labels, fmt='', cmap='Blues')

    plt.title('Confusion Matrix')
    plt.xlabel('predicted label')
    plt.ylabel('True label')
    plt.show()

    print(cm)
    print(classification_report(y_test, prediction))
    final_score = accuracy * 100
    print('accuracy score : ', final_score)

    Accuracy.append(accuracy * 100)
```

Explanation:

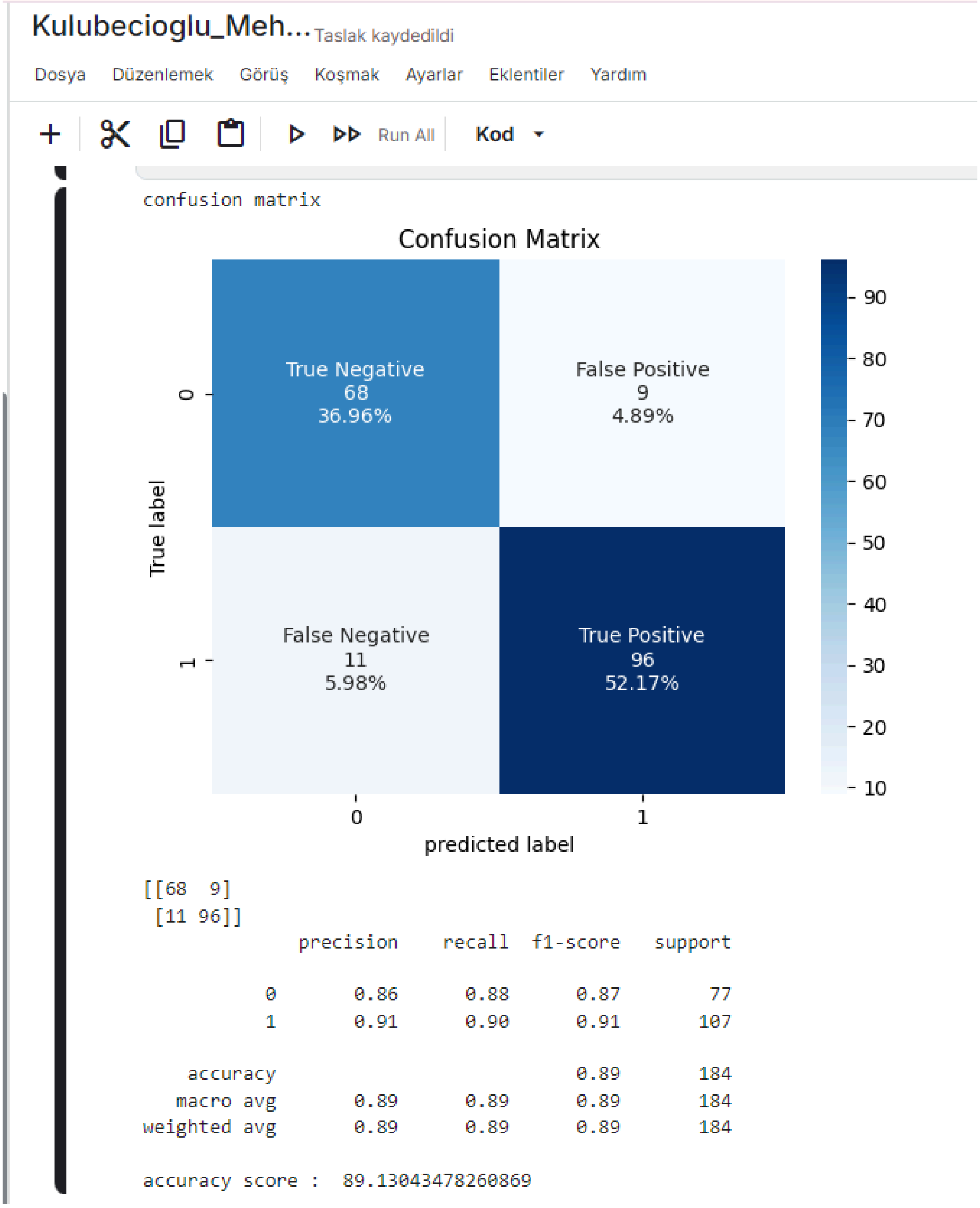
This function, `algorithm()`, is designed to train a given model, make predictions on the test data, and evaluate the model using a confusion matrix and a classification report. The function also plots the confusion matrix and calculates the accuracy score.

6-10. Training and Evaluating Models

Model1:



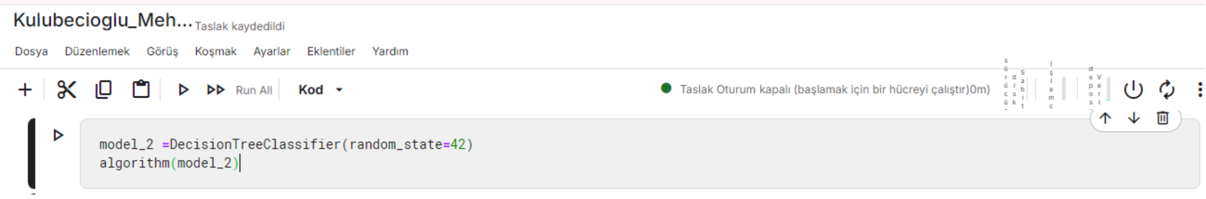
```
model_1 = RandomForestClassifier(n_estimators=100)
algorithm(model_1)
```



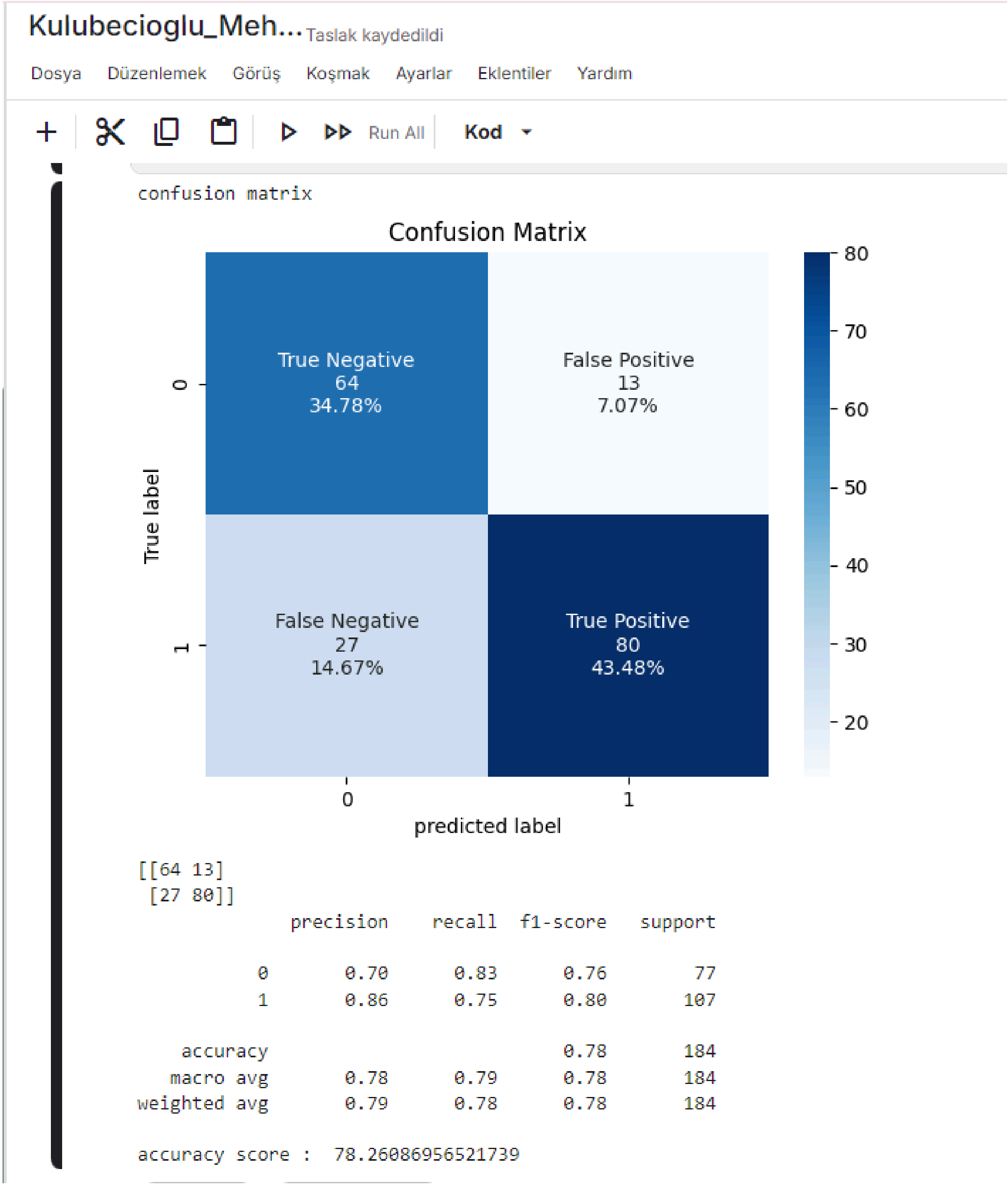
```
confusion matrix
```

```
[[68  9]
 [11 96]]
              precision    recall  f1-score   support

           0       0.86      0.88      0.87        77
           1       0.91      0.90      0.91       107

    accuracy                           0.89       184
   macro avg       0.89      0.89      0.89       184
weighted avg       0.89      0.89      0.89       184

accuracy score :  89.13043478260869
```

Model2:


Kulubecioglu_Meh... Taslak kaydedildi
Dosya
Düzenlemek
Görüş
Koşmak
Ayarlar
Eklentiler
Yardım
Run All
Kod
Taslak Oturum kapalı (başlamak için bir hücreyi çalıştır)0m)

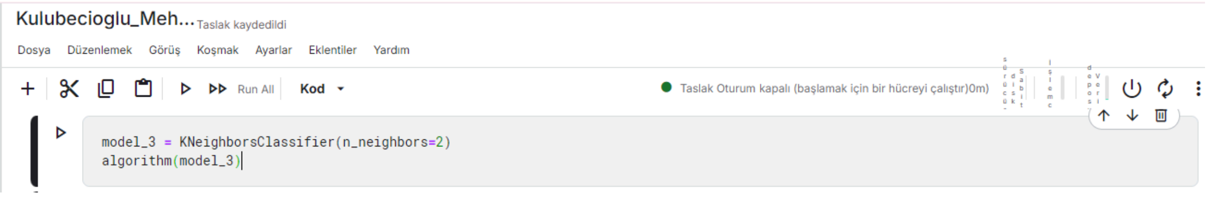
```
model_2 =DecisionTreeClassifier(random_state=42)
algorithm(model_2)
```



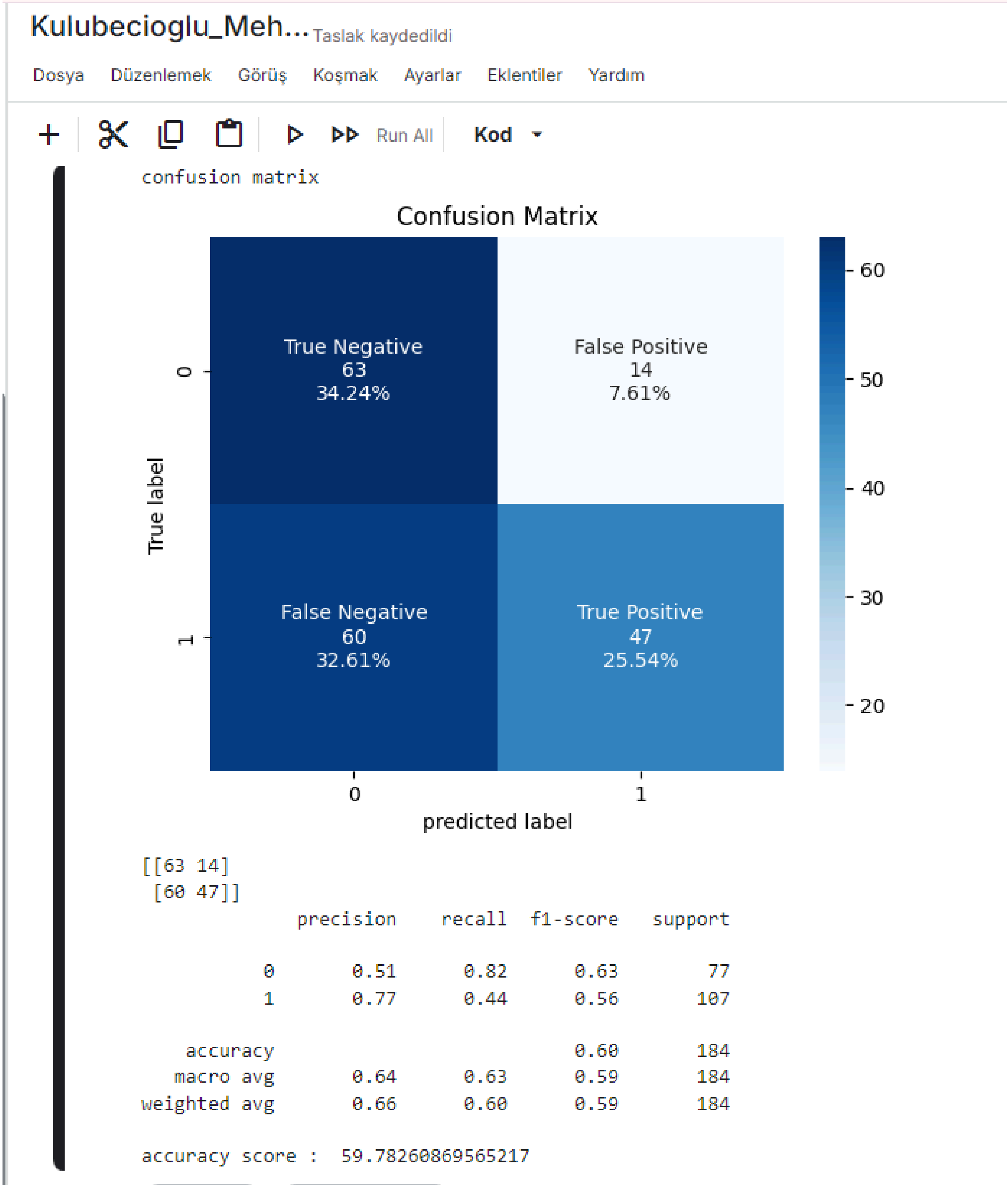
```
confusion matrix
```

```
[[64 13]
 [27 80]]
              precision    recall  f1-score   support

           0       0.70      0.83      0.76        77
           1       0.86      0.75      0.80       107

    accuracy                           0.78       184
   macro avg       0.78      0.79      0.78       184
weighted avg       0.79      0.78      0.78       184

accuracy score :  78.26086956521739
```

Model3:


Kulubecioglu_Meh... Taslak kaydedildi
Dosya Düzenlemek Görüş Koşmak Ayarlar Eklentiler Yardım
Run All
Kod
Taslak Oturum kapalı (başlamak için bir hücreyi çalıştır)0m)

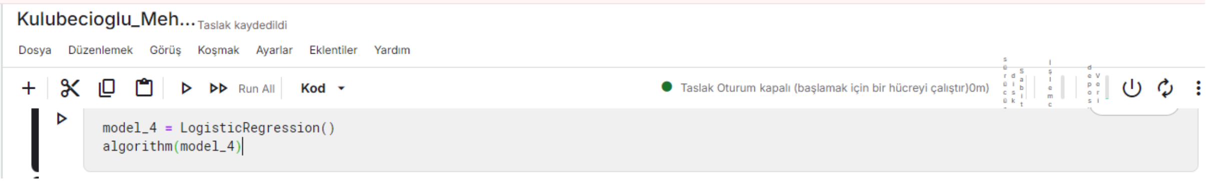
```
model_3 = KNeighborsClassifier(n_neighbors=2)
algorithm(model_3)
```



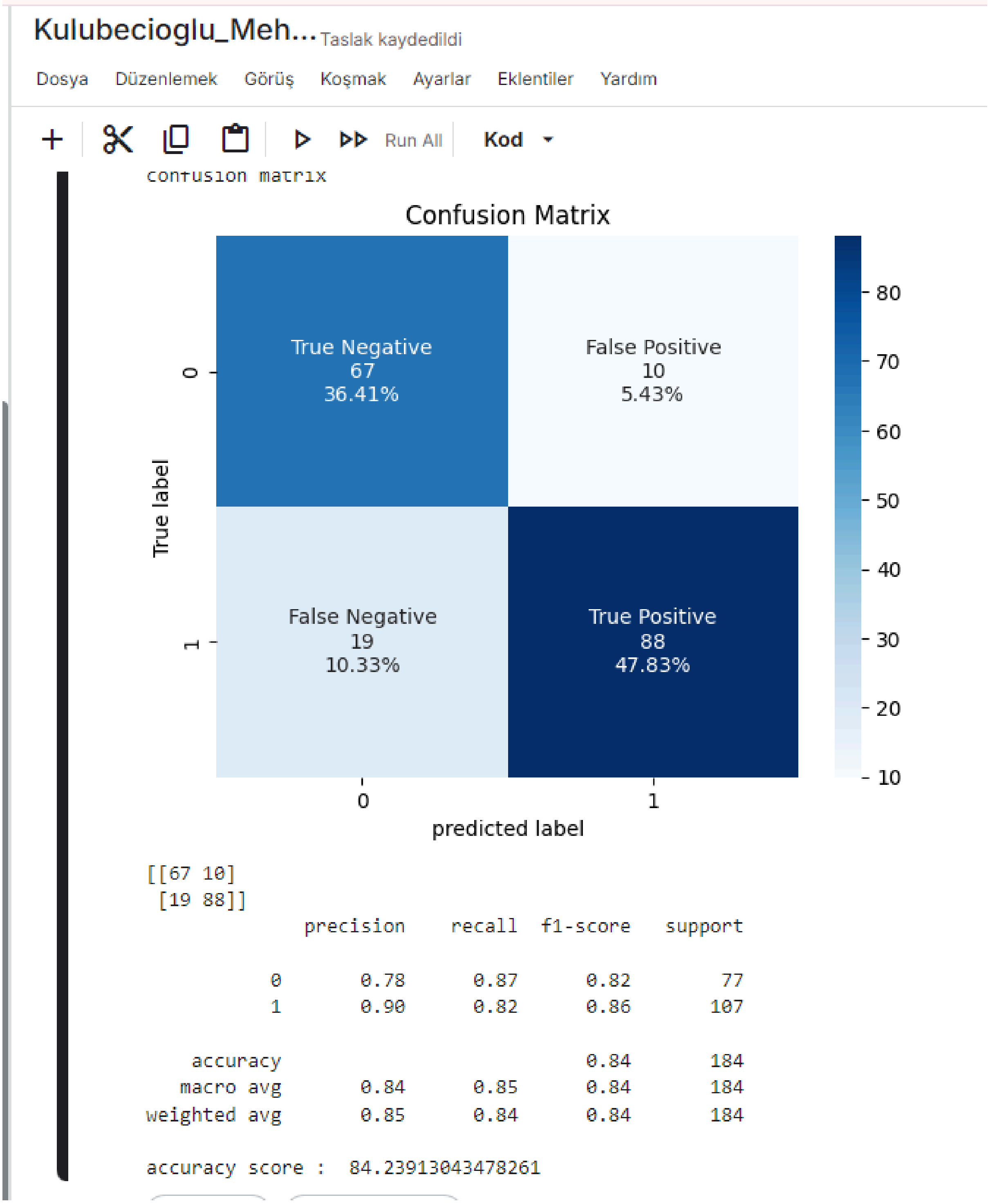
```
confusion matrix
```

```
[[63 14]
 [60 47]]
              precision    recall  f1-score   support

           0       0.51      0.82      0.63        77
           1       0.77      0.44      0.56       107

    accuracy                           0.60       184
   macro avg       0.64      0.63      0.59       184
weighted avg       0.66      0.60      0.59       184

accuracy score :  59.78260869565217
```

Model4:


Kulubecioglu_Meh... Taslak kaydedildi
Dosya
Düzenlemek
Görüş
Koşmak
Ayarlar
Eklentiler
Yardım
Run All
Kod
Taslak Oturum kapalı (başlamak için bir hücreyi çalıştır)0m)

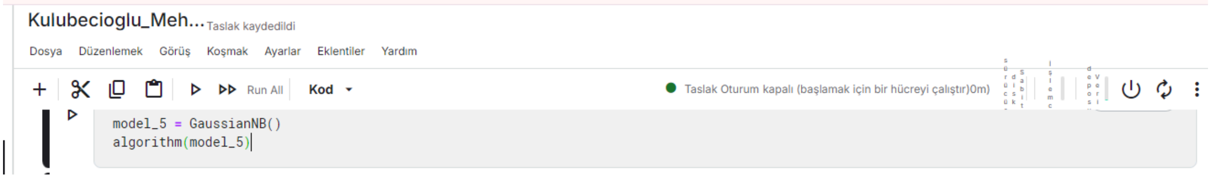
```
model_4 = LogisticRegression()
algorithm(model_4)
```



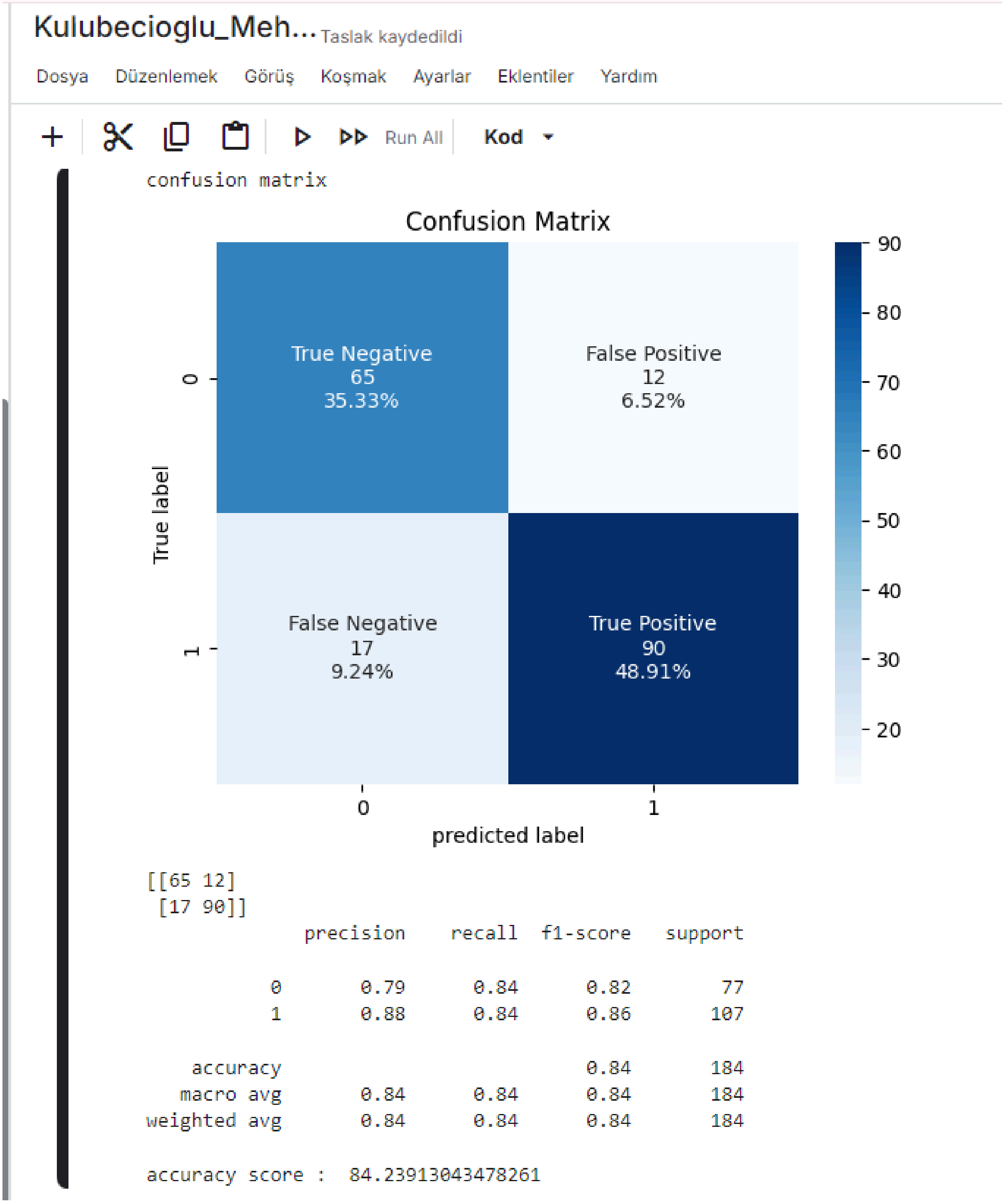
```
confusion matrix
```

```
[[67 10]
 [19 88]]
              precision    recall  f1-score   support

           0       0.78      0.87      0.82        77
           1       0.90      0.82      0.86       107

    accuracy                           0.84       184
   macro avg       0.84      0.85      0.84       184
weighted avg       0.85      0.84      0.84       184

accuracy score :  84.23913043478261
```

Model5:


Kulubecioglu_Meh... Taslak kaydedildi
Dosya Düzenlemek Görüş Koşmak Ayarlar Eklentiler Yardım
Run All
Kod
Taslak Oturum kapalı (başlamak için bir hücreyi çalıştır)0m)

```
model_5 = GaussianNB()
algorithm(model_5)
```



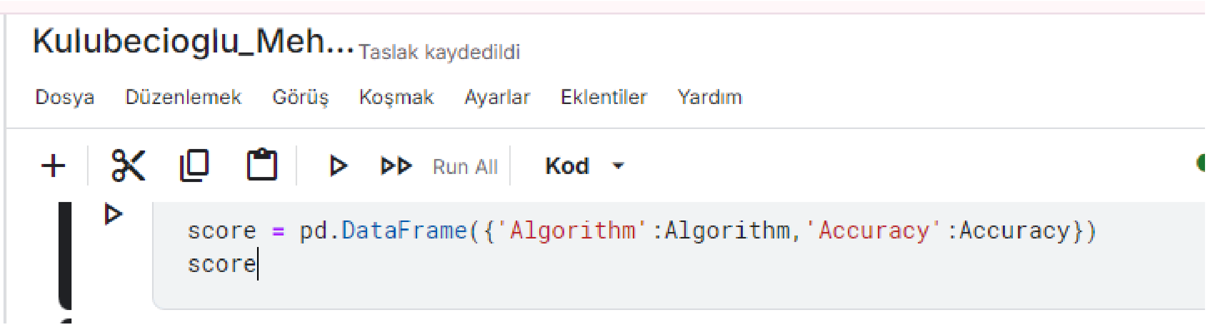
```
confusion matrix
```

```
[[65 12]
 [17 90]]
              precision    recall  f1-score   support

           0       0.79      0.84      0.82        77
           1       0.88      0.84      0.86       107

    accuracy                           0.84       184
   macro avg       0.84      0.84      0.84       184
weighted avg       0.84      0.84      0.84       184

accuracy score :  84.23913043478261
```

Explanation:

In these steps, I applied the `algorithm()` function to five different models: `RandomForestClassifier`, `DecisionTreeClassifier`, `KNeighborsClassifier`, `LogisticRegression`, and `GaussianNB`. For each model, the function trained the model, evaluated it, and appended the accuracy to the `Accuracy` list.

11. Creating a DataFrame for Accuracy

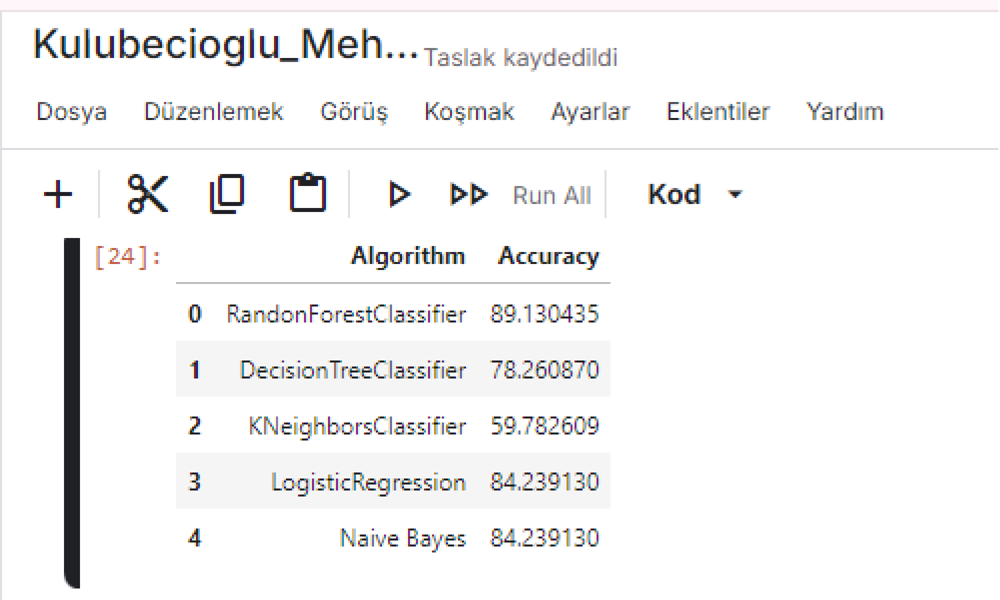

Kulubecioglu_Meh... Taslak kaydedildi
Dosya Düzenlemek Görüş Koşmak Ayarlar Eklentiler Yardım
Run All
Kod

```
score = pd.DataFrame({'Algorithm':Algorithm,'Accuracy':Accuracy})
score
```

Explanation:

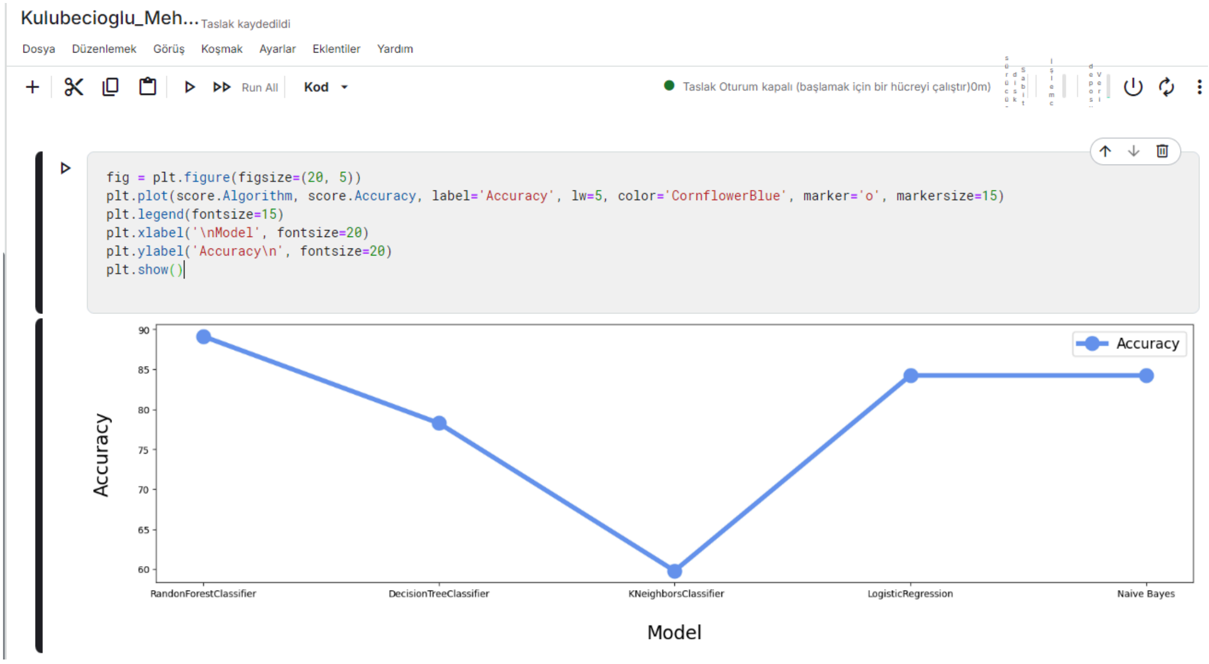
In this step, I created a pandas DataFrame called `score` that contains the names of the algorithms and their respective accuracies.

Output:


Kulubecioglu_Meh... Taslak kaydedildi
Dosya Düzenlemek Görüş Koşmak Ayarlar Eklentiler Yardım
Run All
Kod
[24]:

	Algorithm	Accuracy
0	RandonForestClassifier	89.130435
1	DecisionTreeClassifier	78.260870
2	KNeighborsClassifier	59.782609
3	LogisticRegression	84.239130
4	Naive Bayes	84.239130

12. Plotting Model Accuracy



```
fig = plt.figure(figsize=(20, 5))
plt.plot(score.Algorithm, score.Accuracy, label='Accuracy', lw=5, color='CornflowerBlue', marker='o', markersize=15)
plt.legend(fontsize=15)
plt.xlabel('\nModel', fontsize=20)
plt.ylabel('Accuracy\n', fontsize=20)
plt.show()
```

Explanation:

Finally, I plotted the accuracies of all models using a line plot. This visualization helps to compare the performance of different algorithms in terms of accuracy.