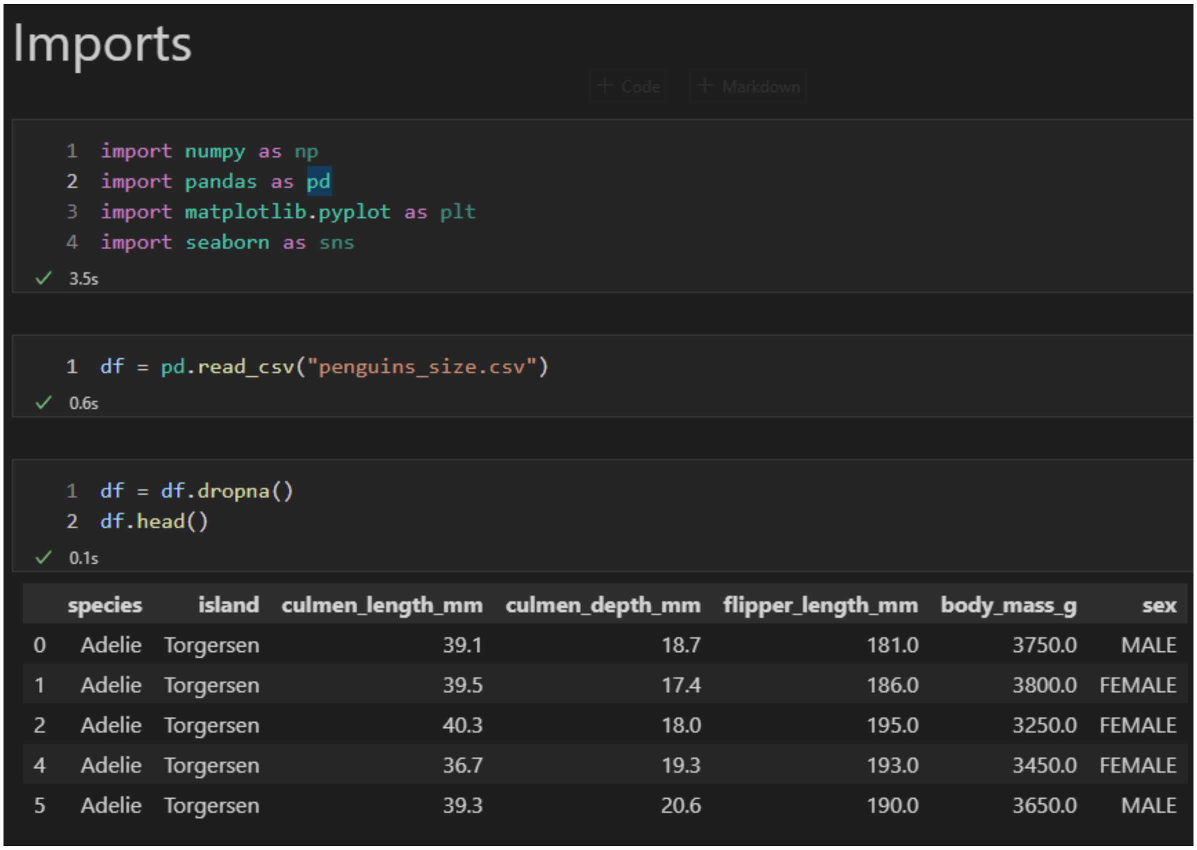
🌳

Random Forest

▼ PENGUIN SPECIES

- INTRO



Imports

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

✓ 3.5s

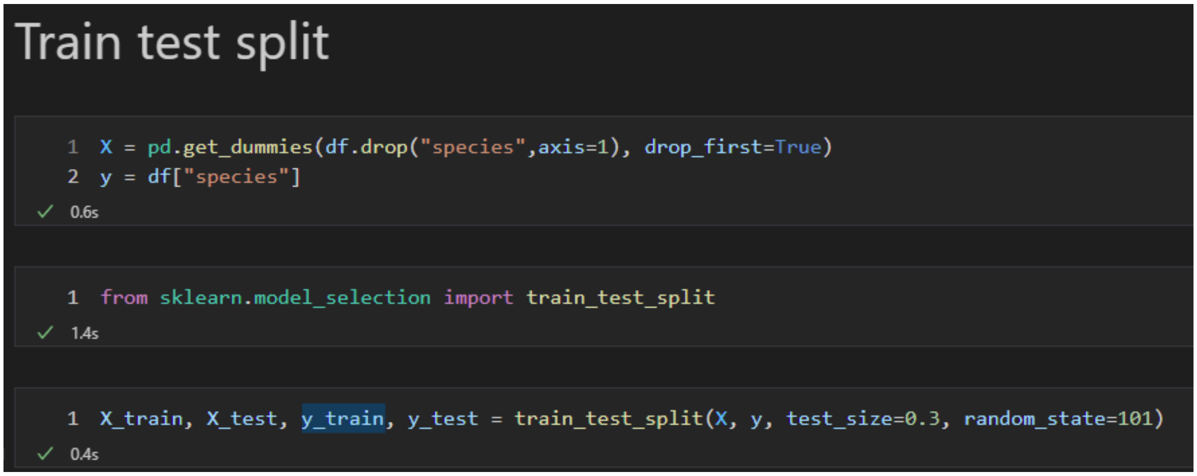
```
df = pd.read_csv("penguins_size.csv")
```

✓ 0.6s

```
df = df.dropna()
df.head()
```

✓ 0.1s

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	MALE

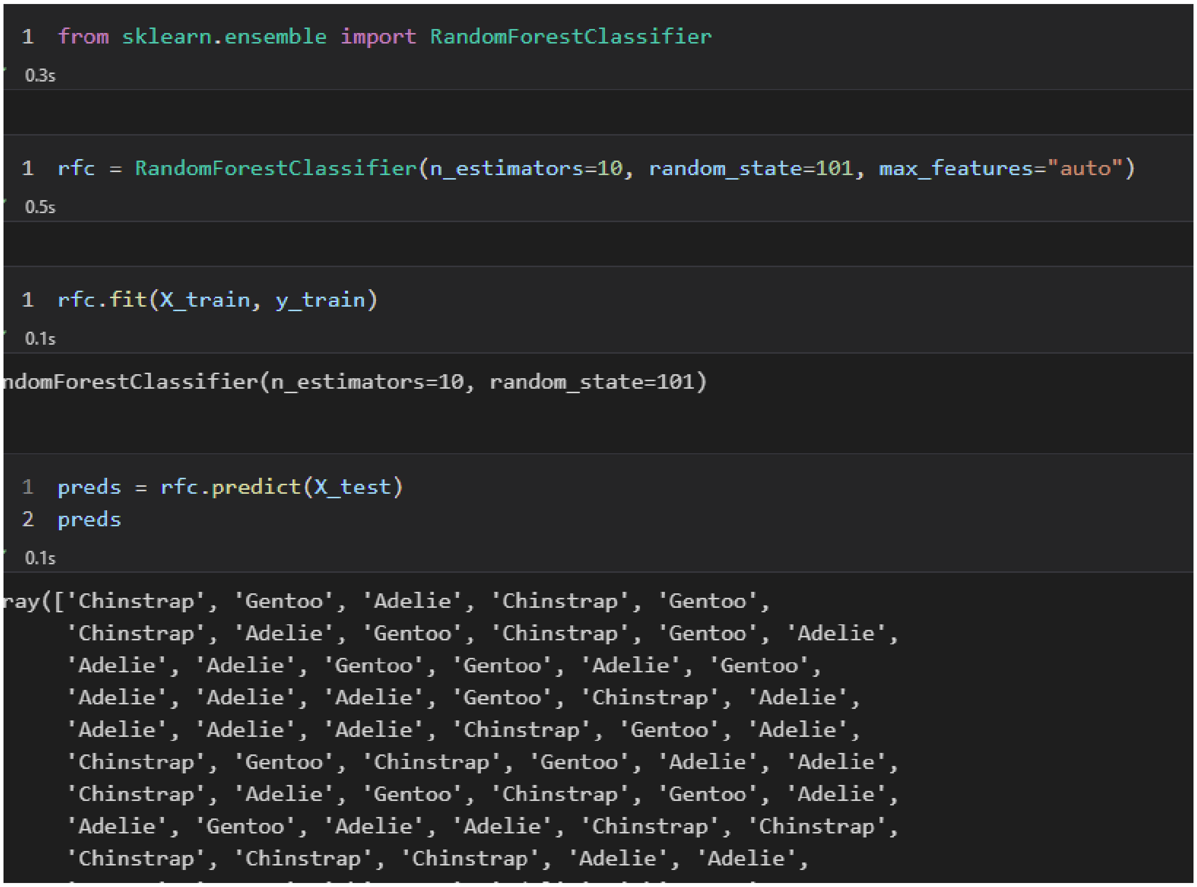


Train test split

```
X = pd.get_dummies(df.drop("species",axis=1), drop_first=True)
y = df["species"]
```
✓ 0.6s

```
from sklearn.model_selection import train_test_split
```
✓ 1.4s

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```
✓ 0.4s



```
from sklearn.ensemble import RandomForestClassifier
```
0.3s

```
rfc = RandomForestClassifier(n_estimators=10, random_state=101, max_features="auto")
```
0.5s

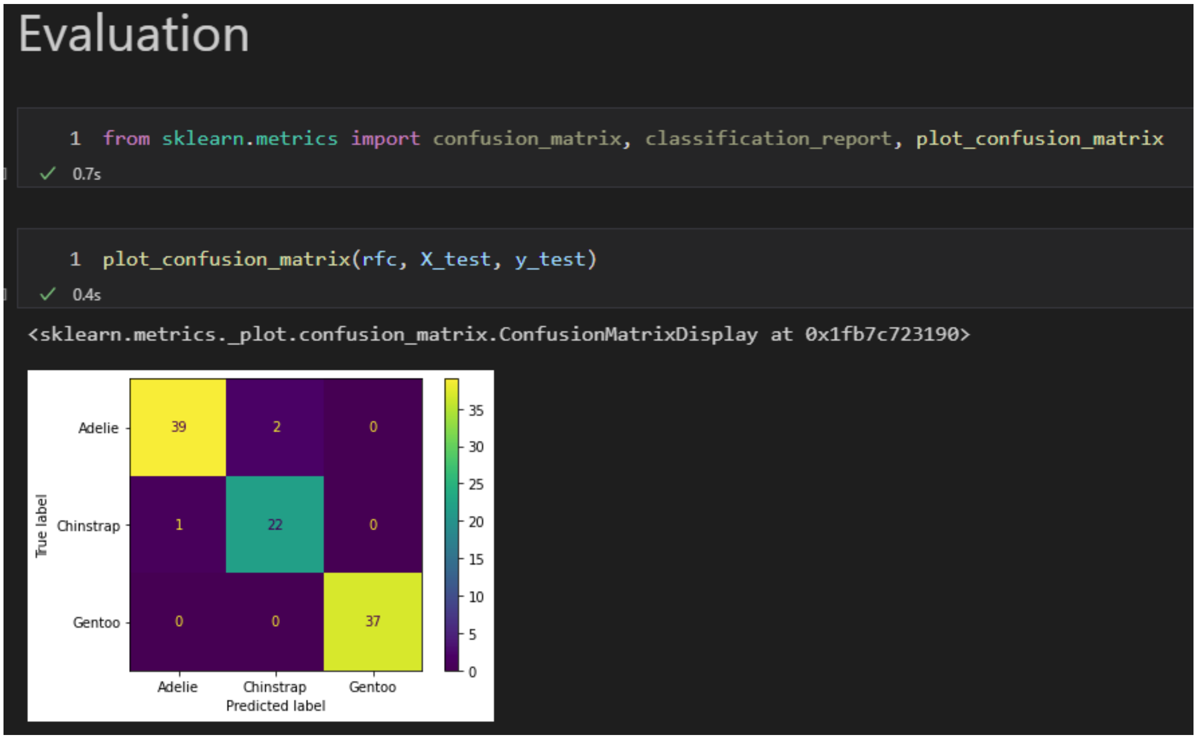
```
rfc.fit(X_train, y_train)
```
0.1s

```
ndomForestClassifier(n_estimators=10, random_state=101)
```

```
preds = rfc.predict(X_test)
preds
```
0.1s

```
ray(['Chinstrap', 'Gentoo', 'Adelie', 'Chinstrap', 'Gentoo',
     'Chinstrap', 'Adelie', 'Gentoo', 'Chinstrap', 'Gentoo', 'Adelie',
     'Adelie', 'Adelie', 'Gentoo', 'Gentoo', 'Adelie', 'Gentoo',
     'Adelie', 'Adelie', 'Adelie', 'Gentoo', 'Chinstrap', 'Adelie',
     'Adelie', 'Adelie', 'Adelie', 'Chinstrap', 'Gentoo', 'Adelie',
     'Chinstrap', 'Gentoo', 'Chinstrap', 'Gentoo', 'Adelie', 'Adelie',
     'Chinstrap', 'Adelie', 'Gentoo', 'Chinstrap', 'Gentoo', 'Adelie',
     'Adelie', 'Gentoo', 'Adelie', 'Adelie', 'Chinstrap', 'Chinstrap',
     'Chinstrap', 'Chinstrap', 'Chinstrap', 'Adelie', 'Adelie',
```

- Evaluation



Evaluation

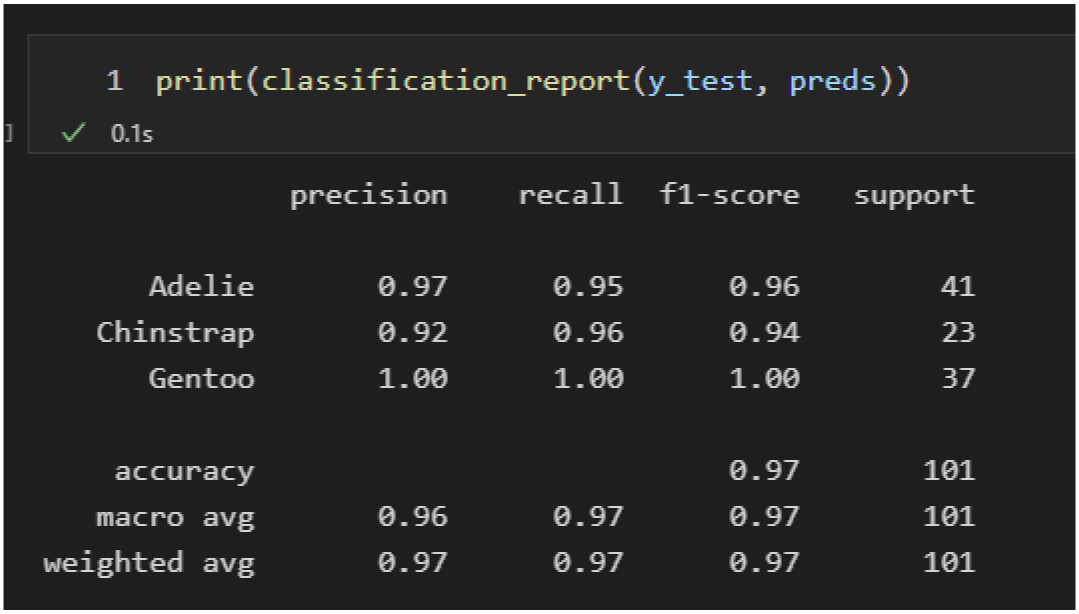
```
from sklearn.metrics import confusion_matrix, classification_report, plot_confusion_matrix
```

✓ 0.7s

```
plot_confusion_matrix(rfc, X_test, y_test)
```

✓ 0.4s

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1fb7c723190>
```

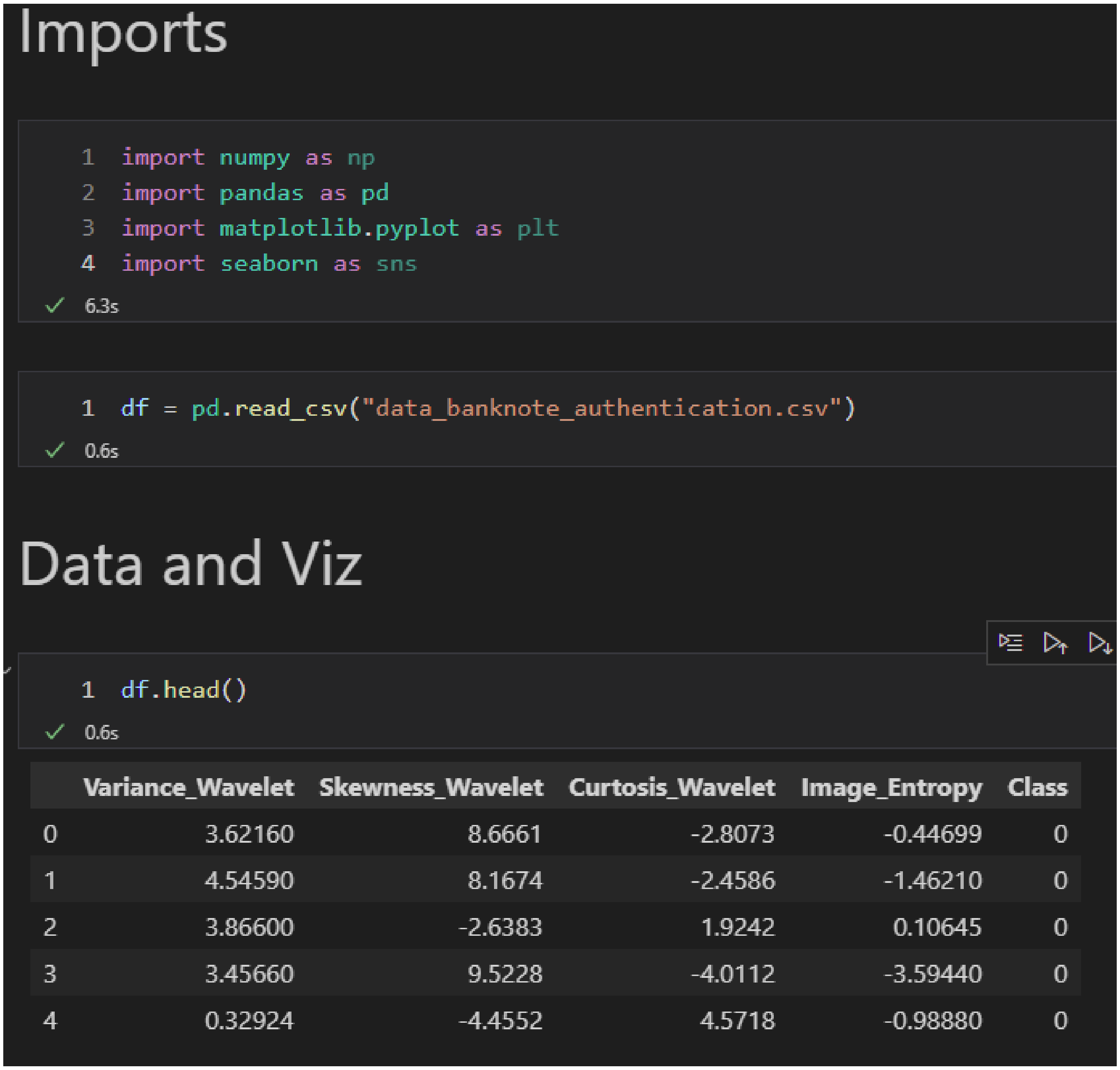


```
print(classification_report(y_test, preds))
```

✓ 0.1s

```
              precision    recall  f1-score   support

      Adelie       0.97      0.95      0.96        41
   Chinstrap       0.92      0.96      0.94        23
      Gentoo       1.00      1.00      1.00        37

    accuracy                           0.97       101
   macro avg       0.96      0.97      0.97       101
weighted avg       0.97      0.97      0.97       101
```

▼ Banknote Authentication



Imports

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

✓ 6.3s

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

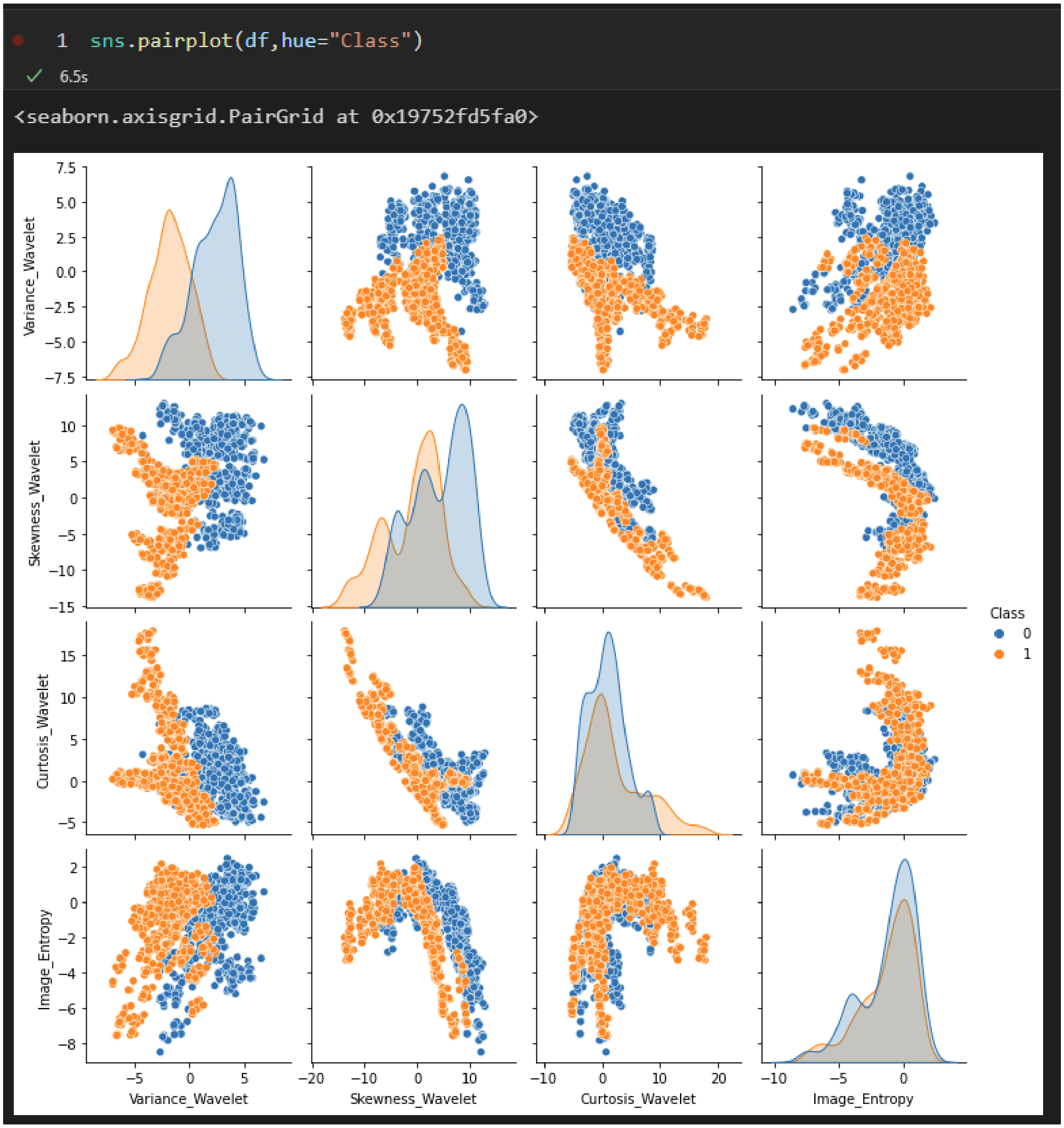
✓ 0.6s

Data and Viz

```python
df.head()
```

✓ 0.6s

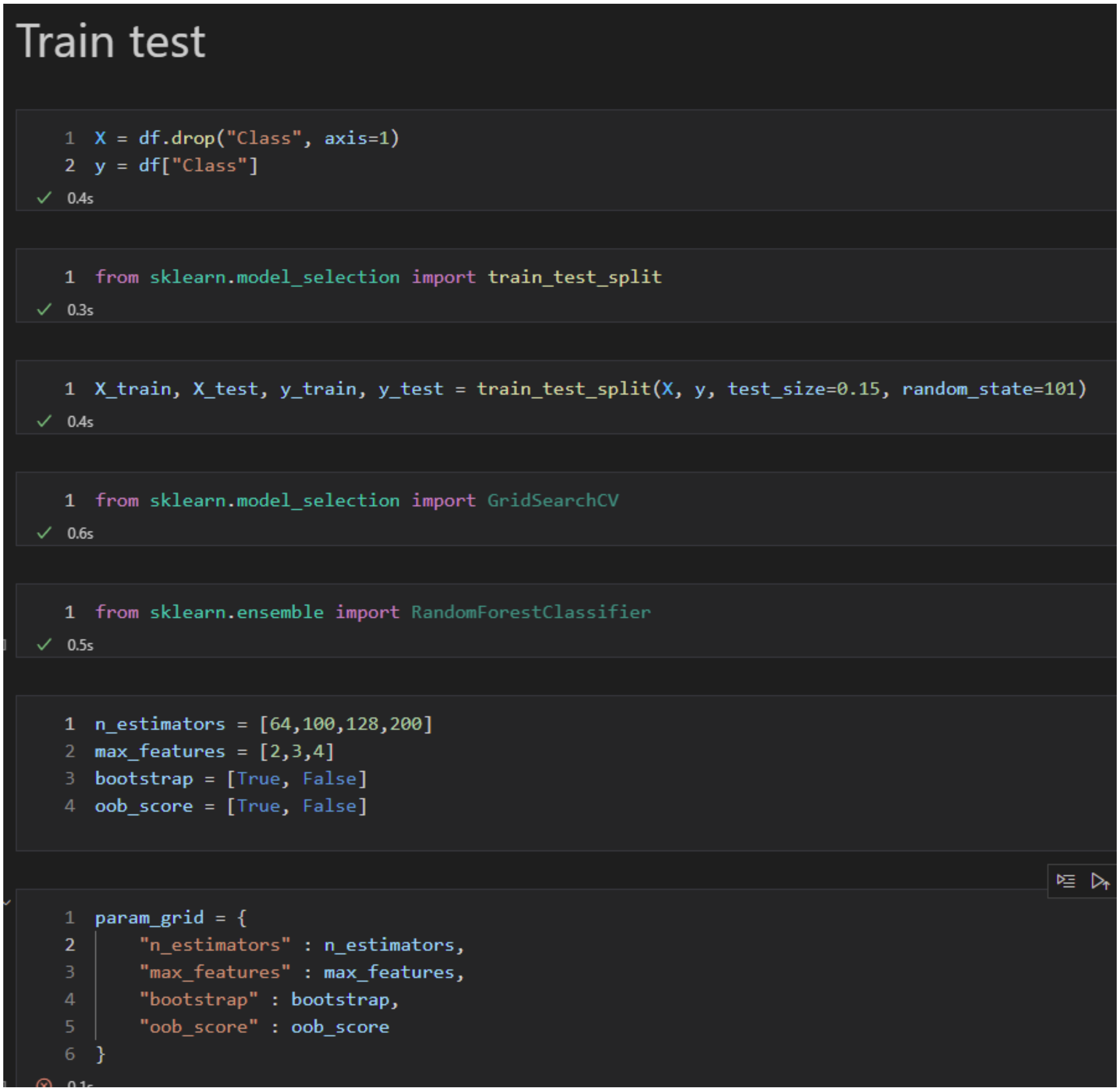
	Variance_Wavelet	Skewness_Wavelet	Curtosis_Wavelet	Image_Entropy	Class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0



```
sns.pairplot(df,hue="Class")
```

✓ 6.5s

```
<seaborn.axisgrid.PairGrid at 0x19752fd5fa0>
```



Train test

```python
X = df.drop("Class", axis=1)
y = df["Class"]
```

✓ 0.4s

```python
from sklearn.model_selection import train_test_split
```

✓ 0.3s

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_state=101)
```

✓ 0.4s

```python
from sklearn.model_selection import GridSearchCV
```

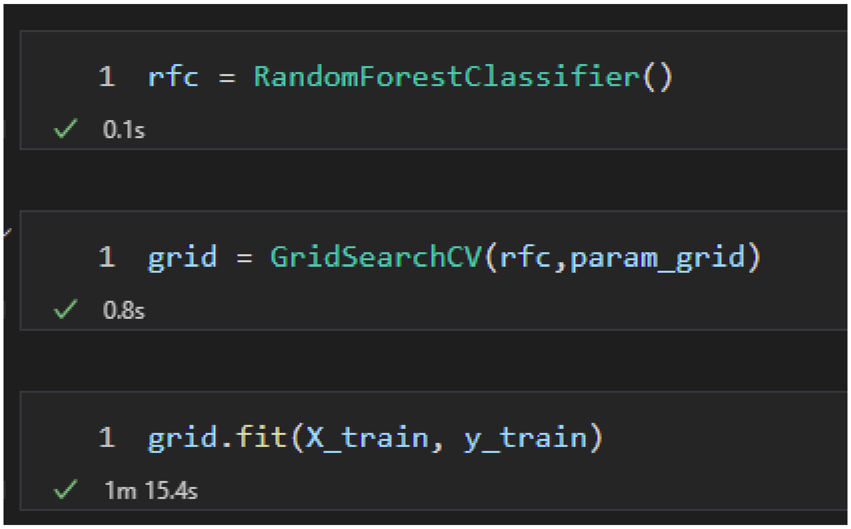
✓ 0.6s

```python
from sklearn.ensemble import RandomForestClassifier
```

✓ 0.5s

```python
n_estimators = [64,100,128,200]
max_features = [2,3,4]
bootstrap = [True, False]
oob_score = [True, False]
```

```python
param_grid = {
    "n_estimators" : n_estimators,
    "max_features" : max_features,
    "bootstrap" : bootstrap,
    "oob_score" : oob_score
}
```



```python
rfc = RandomForestClassifier()
```

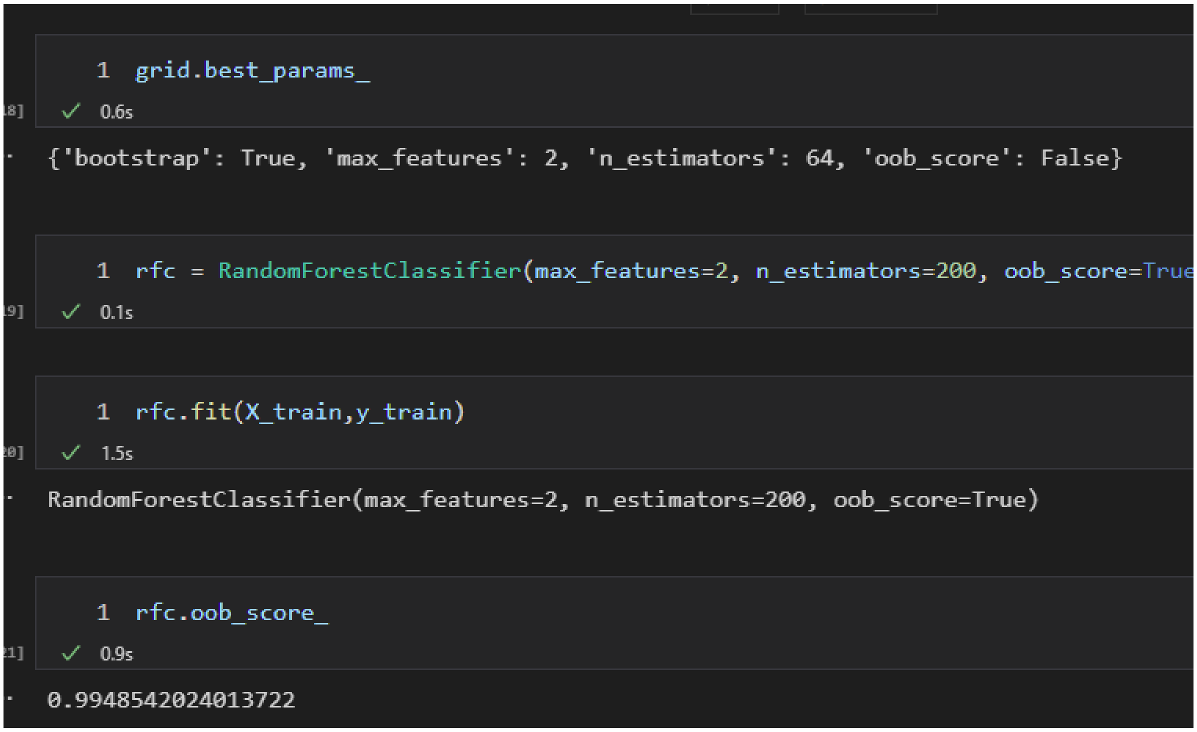
✓ 0.1s

```python
grid = GridSearchCV(rfc,param_grid)
```

✓ 0.8s

```python
grid.fit(X_train, y_train)
```

✓ 1m 15.4s



```
grid.best_params_
```

✓ 0.6s

```
{'bootstrap': True, 'max_features': 2, 'n_estimators': 64, 'oob_score': False}
```

```
rfc = RandomForestClassifier(max_features=2, n_estimators=200, oob_score=True
```

✓ 0.1s

```
rfc.fit(X_train,y_train)
```

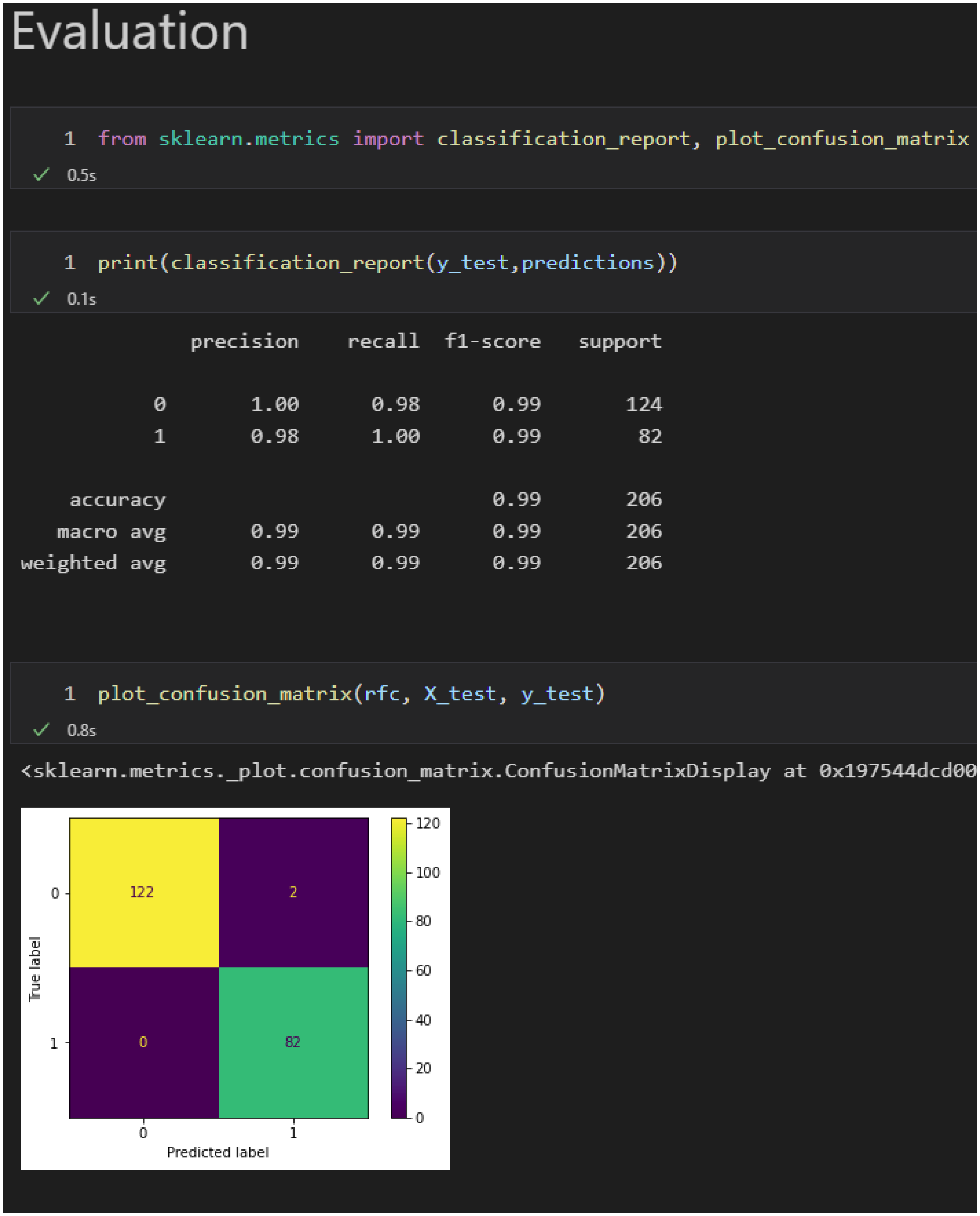
✓ 1.5s

```
RandomForestClassifier(max_features=2, n_estimators=200, oob_score=True)
```

```
rfc.oob_score_
```

✓ 0.9s

```
0.9948542024013722
```



Evaluation

```
from sklearn.metrics import classification_report, plot_confusion_matrix
```

✓ 0.5s

```
print(classification_report(y_test,predictions))
```

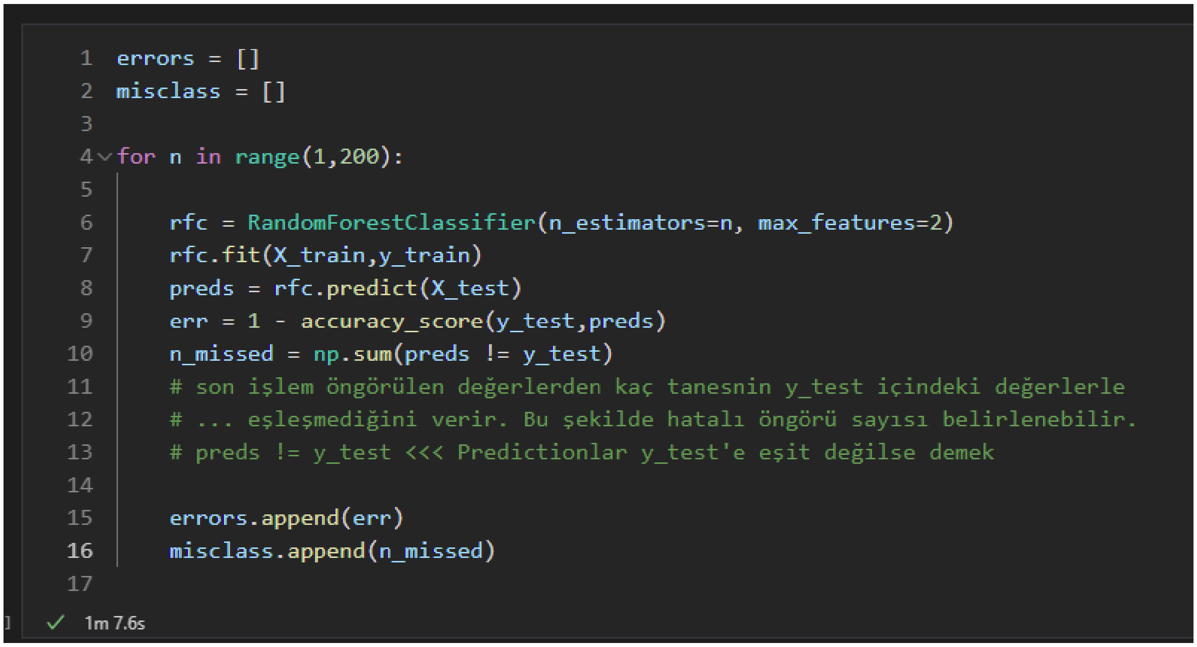
✓ 0.1s

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       124
           1       0.98      1.00      0.99        82

    accuracy                           0.99       206
   macro avg       0.99      0.99      0.99       206
weighted avg       0.99      0.99      0.99       206
```

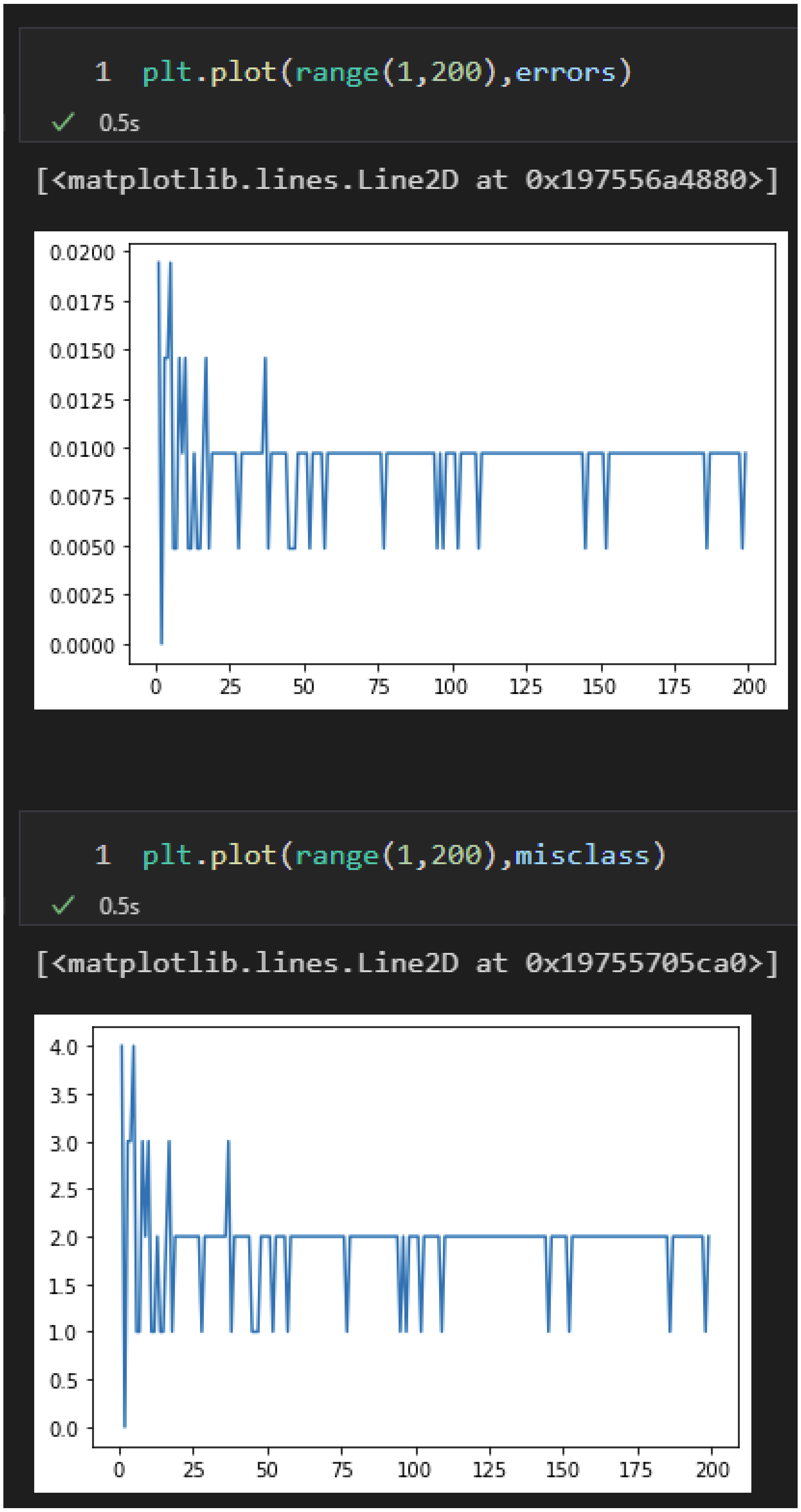
```
plot_confusion_matrix(rfc, X_test, y_test)
```

✓ 0.8s

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x197544dcd00
```



```python
errors = []
misclass = []

for n in range(1,200):

    rfc = RandomForestClassifier(n_estimators=n, max_features=2)
    rfc.fit(X_train,y_train)
    preds = rfc.predict(X_test)
    err = 1 - accuracy_score(y_test,preds)
    n_missed = np.sum(preds != y_test)
    # son işlem öngörülen değerlerden kaç tanesnin y_test içindeki değerlerle
    # ... eşleşmediğini verir. Bu şekilde hatalı öngörü sayısı belirlenebilir.
    # preds != y_test <<< Predictionlar y_test'e eşit değilse demek

    errors.append(err)
    misclass.append(n_missed)

```



```
plt.plot(range(1,200),errors)
```

✓ 0.5s

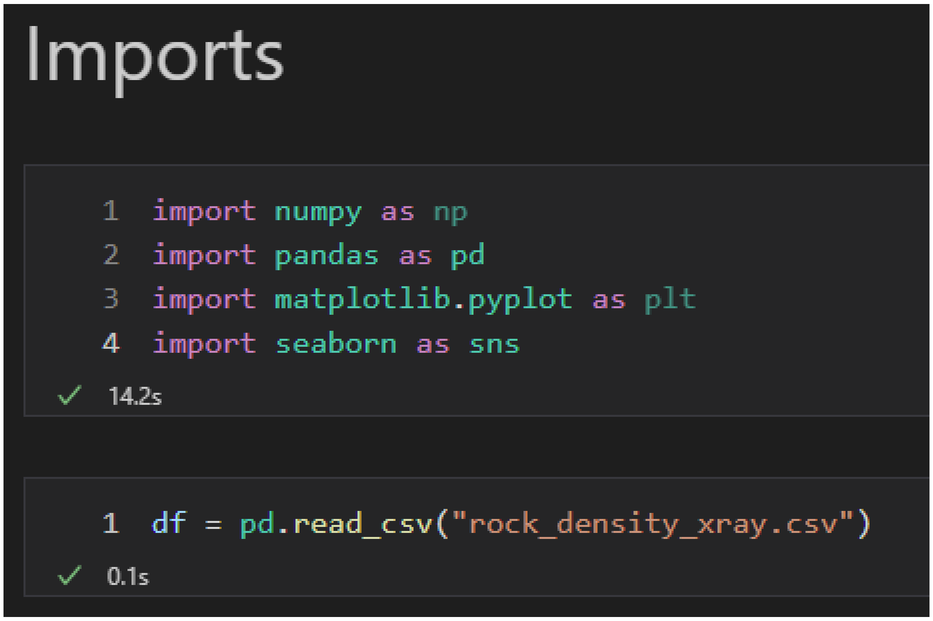
```
[<matplotlib.lines.Line2D at 0x197556a4880>]
```

```
plt.plot(range(1,200),misclass)
```

✓ 0.5s

```
[<matplotlib.lines.Line2D at 0x19755705ca0>]
```

▼ Rock Tunnelling



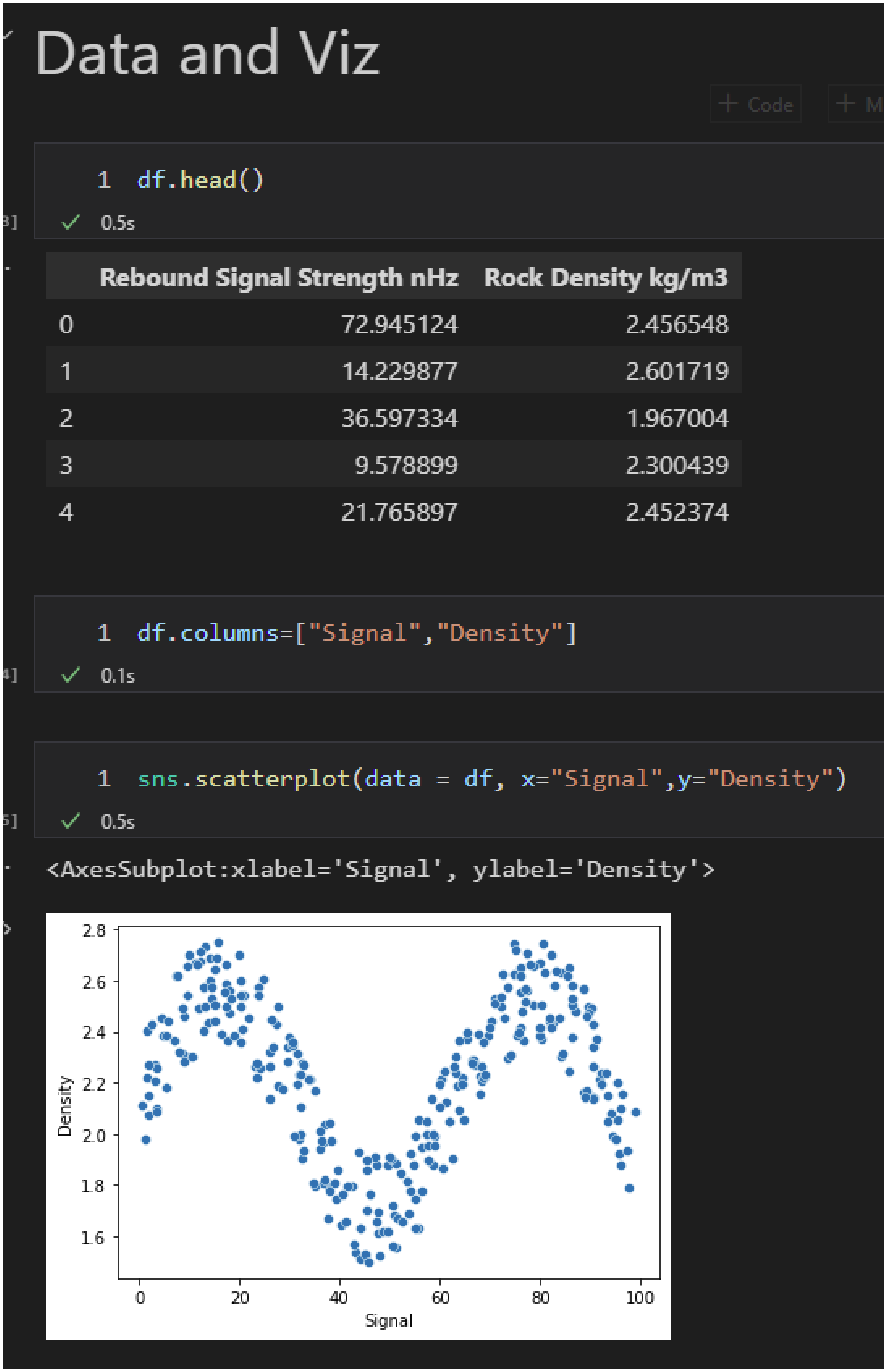
Imports

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

✓ 14.2s

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

✓ 0.1s



Data and Viz

```
df.head()
```

✓ 0.5s

	Rebound Signal Strength nHz	Rock Density kg/m3
0	72.945124	2.456548
1	14.229877	2.601719
2	36.597334	1.967004
3	9.578899	2.300439
4	21.765897	2.452374

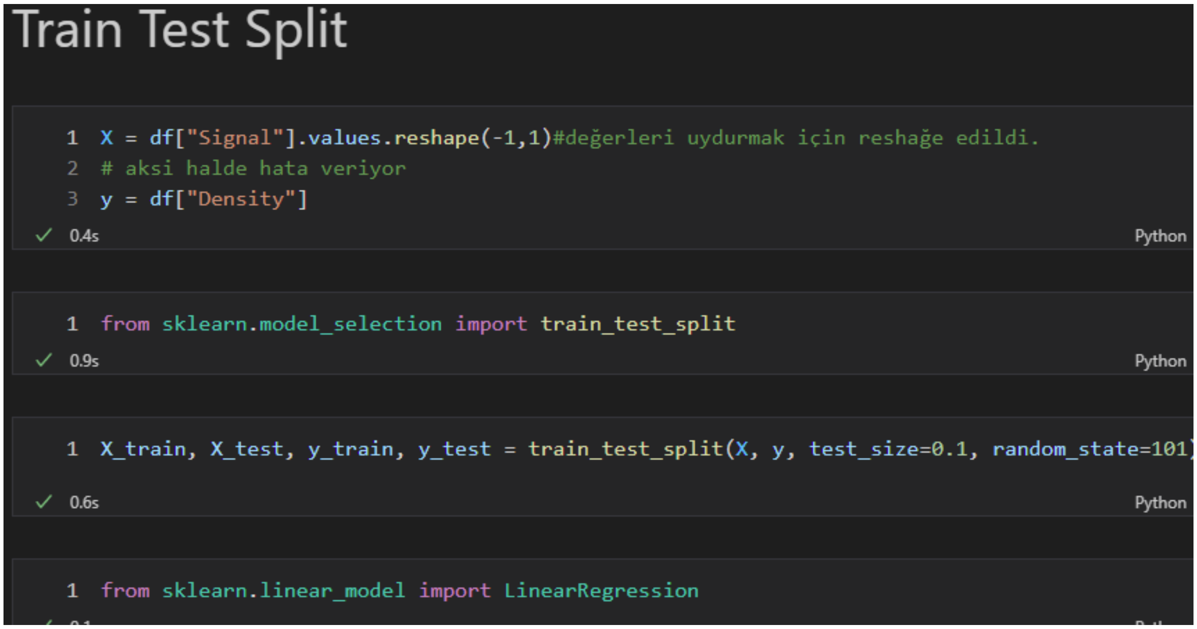
```
df.columns=["Signal","Density"]
```

✓ 0.1s

```
sns.scatterplot(data = df, x="Signal",y="Density")
```

✓ 0.5s

```
<AxesSubplot:xlabel='Signal', ylabel='Density'>
```



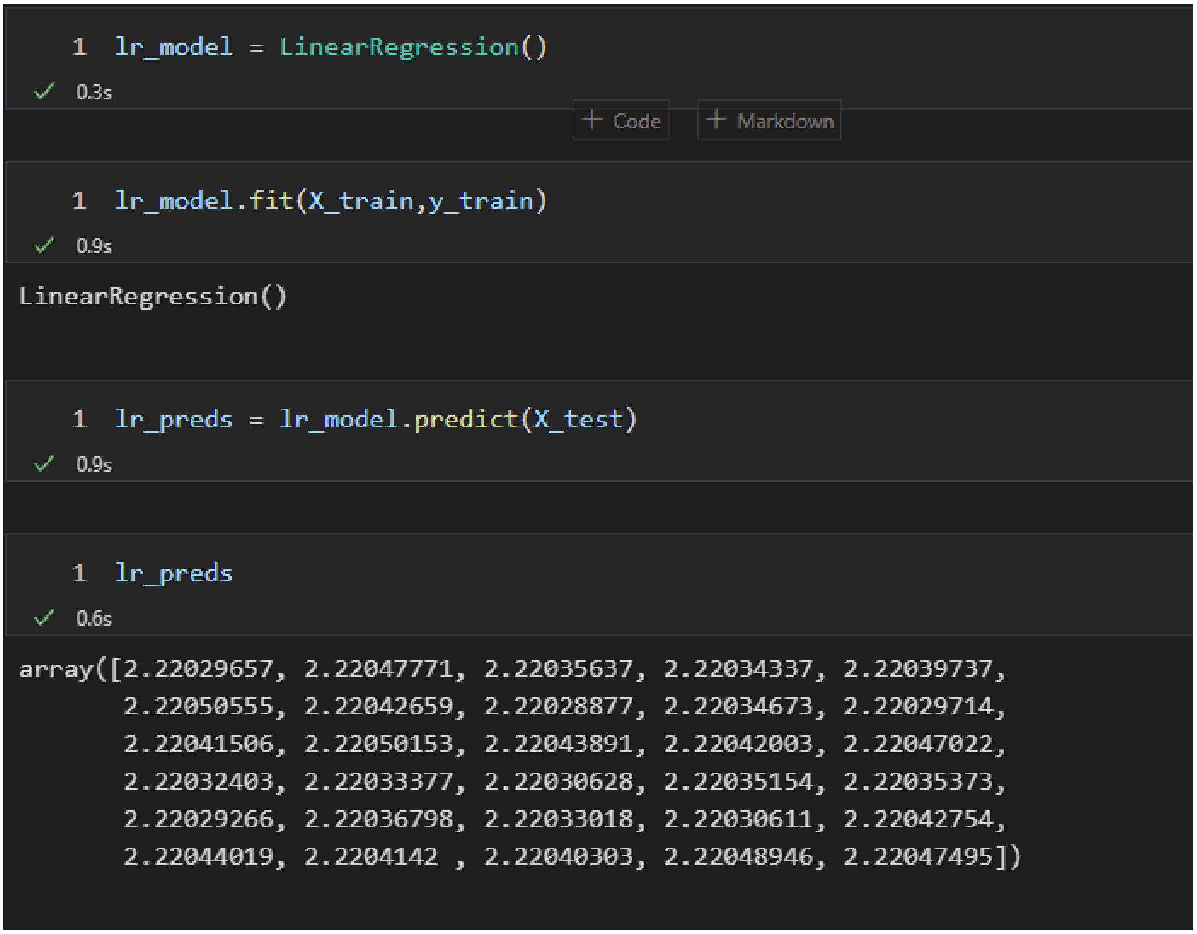
Train Test Split

```python
X = df["Signal"].values.reshape(-1,1)#değerleri uydurmak için reshağe edildi.
# aksi halde hata veriyor
y = df["Density"]
```

```python
from sklearn.model_selection import train_test_split
```

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

```python
from sklearn.linear_model import LinearRegression
```



```python
lr_model = LinearRegression()
```

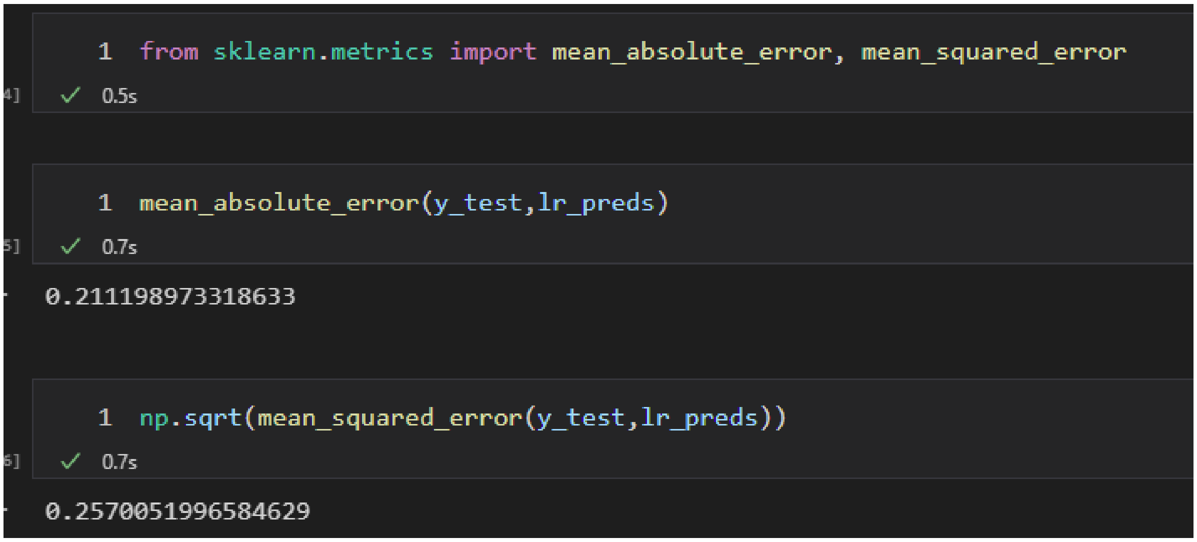
```python
lr_model.fit(X_train,y_train)
```

```
LinearRegression()
```

```python
lr_preds = lr_model.predict(X_test)
```

```python
lr_preds
```

```
array([2.22029657, 2.22047771, 2.22035637, 2.22034337, 2.22039737,
       2.22050555, 2.22042659, 2.22028877, 2.22034673, 2.22029714,
       2.22041506, 2.22050153, 2.22043891, 2.22042003, 2.22047022,
       2.22032403, 2.22033377, 2.22030628, 2.22035154, 2.22035373,
       2.22029266, 2.22036798, 2.22033018, 2.22030611, 2.22042754,
       2.22044019, 2.2204142 , 2.22040303, 2.22048946, 2.22047495])
```



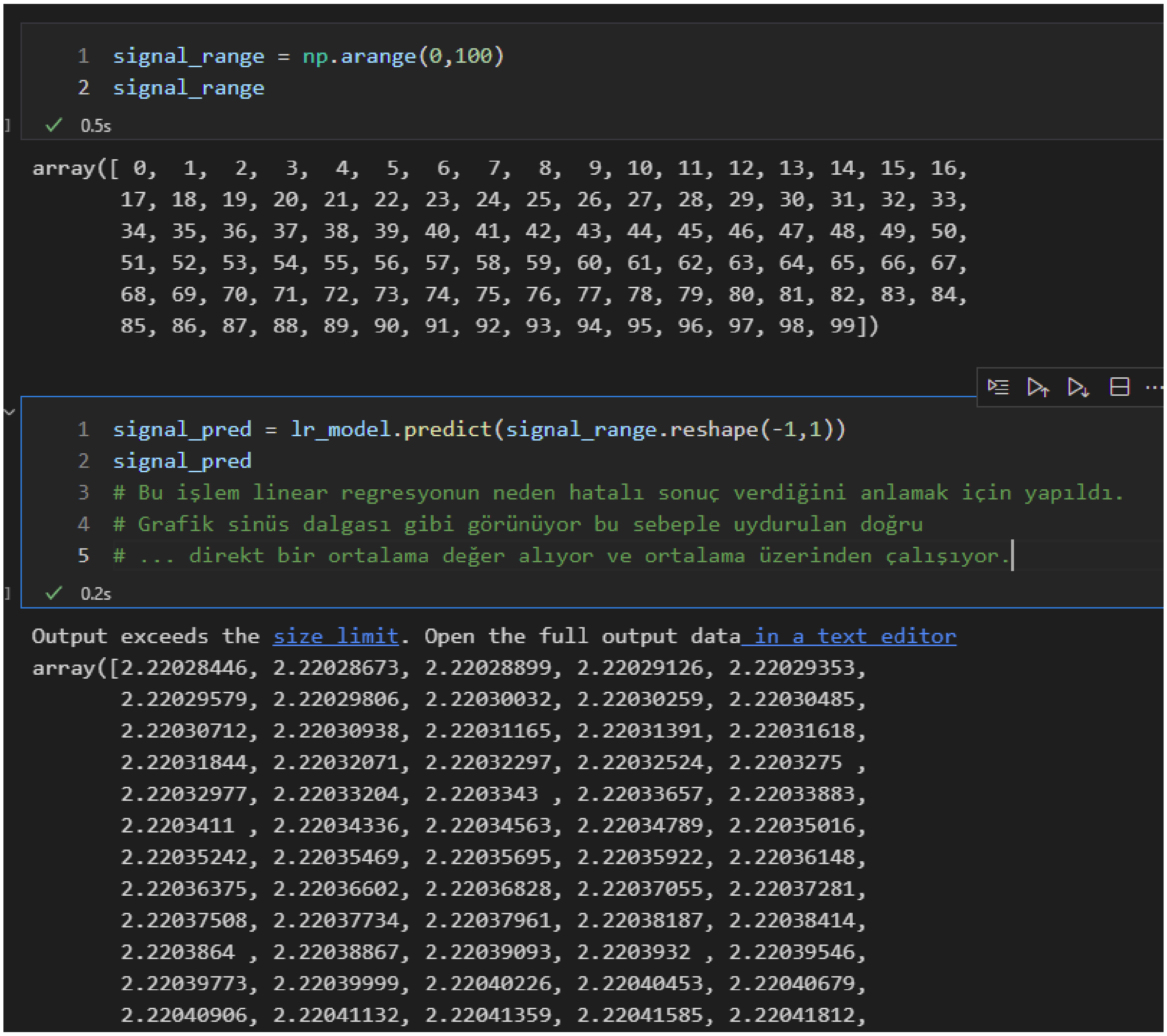
```python
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

```python
mean_absolute_error(y_test,lr_preds)
```

```
0.211198973318633
```

```python
np.sqrt(mean_squared_error(y_test,lr_preds))
```

```
0.2570051996584629
```

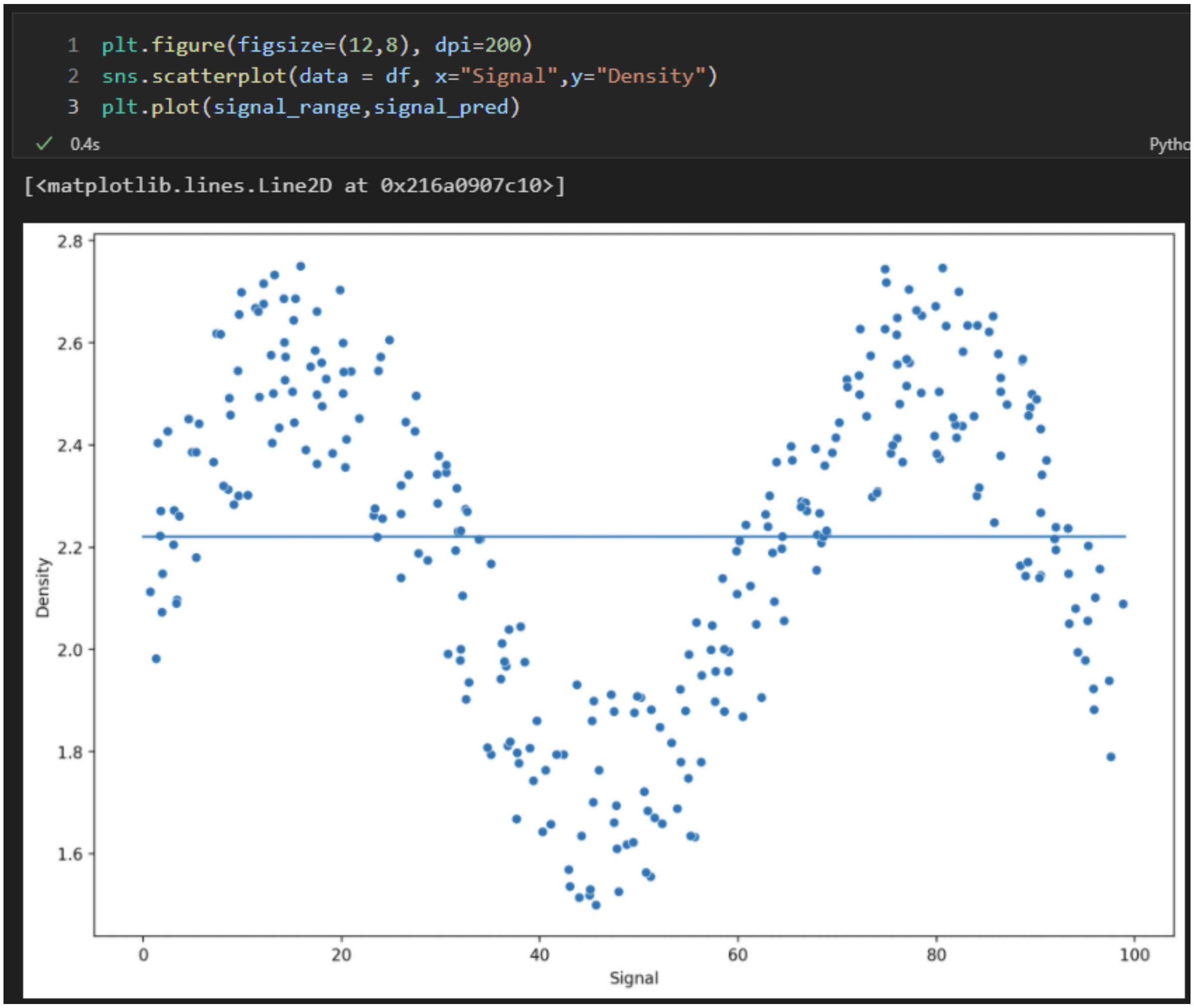


```python
signal_range = np.arange(0,100)
signal_range
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99])
```

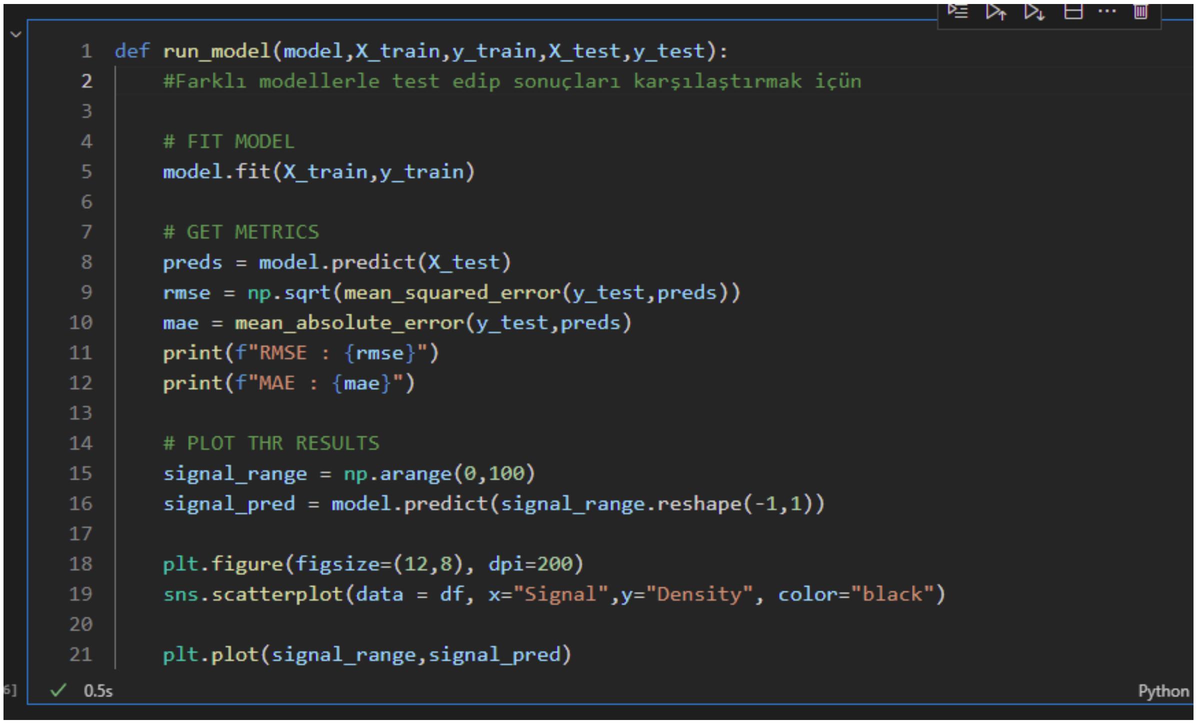
```python
signal_pred = lr_model.predict(signal_range.reshape(-1,1))
signal_pred
# Bu işlem linear regresyonun neden hatalı sonuç verdiğini anlamak için yapıldı.
# Grafik sinüs dalgası gibi görünüyor bu sebeple uydurulan doğru
# ... direkt bir ortalama değer alıyor ve ortalama üzerinden çalışıyor.
```

```
Output exceeds the size limit. Open the full output data in a text editor
array([2.22028446, 2.22028673, 2.22028899, 2.22029126, 2.22029353,
       2.22029579, 2.22029806, 2.22030032, 2.22030259, 2.22030485,
       2.22030712, 2.22030938, 2.22031165, 2.22031391, 2.22031618,
       2.22031844, 2.22032071, 2.22032297, 2.22032524, 2.2203275 ,
       2.22032977, 2.22033204, 2.2203343 , 2.22033657, 2.22033883,
       2.2203411 , 2.22034336, 2.22034563, 2.22034789, 2.22035016,
       2.22035242, 2.22035469, 2.22035695, 2.22035922, 2.22036148,
       2.22036375, 2.22036602, 2.22036828, 2.22037055, 2.22037281,
       2.22037508, 2.22037734, 2.22037961, 2.22038187, 2.22038414,
       2.2203864 , 2.22038867, 2.22039093, 2.2203932 , 2.22039546,
       2.22039773, 2.22039999, 2.22040226, 2.22040453, 2.22040679,
       2.22040906, 2.22041132, 2.22041359, 2.22041585, 2.22041812,
```

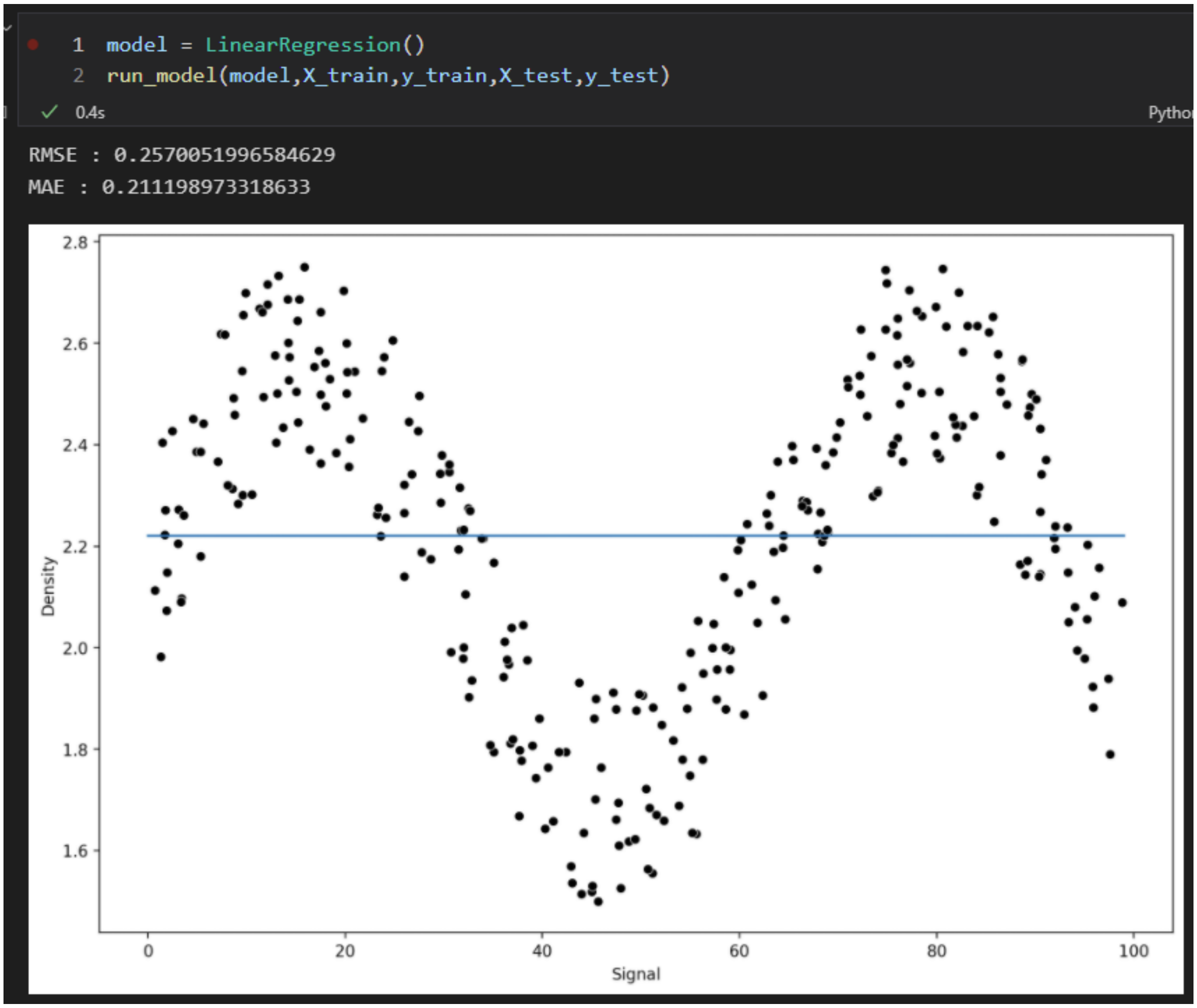


```
plt.figure(figsize=(12,8), dpi=200)
sns.scatterplot(data = df, x="Signal",y="Density")
plt.plot(signal_range,signal_pred)
```

```
[<matplotlib.lines.Line2D at 0x216a0907c10>]
```

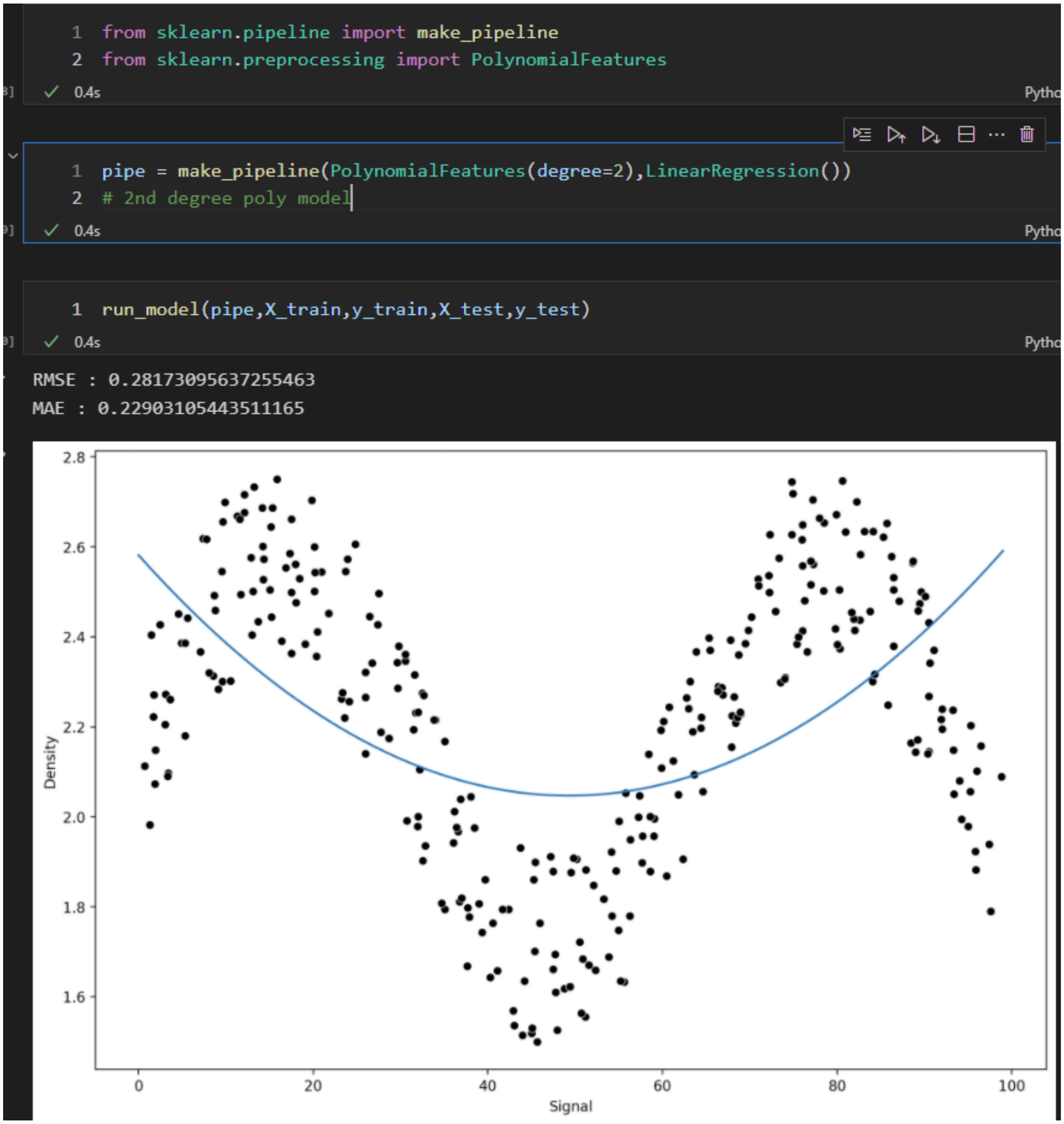


```python
def run_model(model,X_train,y_train,X_test,y_test):
    #Farklı modellerle test edip sonuçları karşılaştırmak için

    # FIT MODEL
    model.fit(X_train,y_train)

    # GET METRICS
    preds = model.predict(X_test)
    rmse = np.sqrt(mean_squared_error(y_test,preds))
    mae = mean_absolute_error(y_test,preds)
    print(f"RMSE : {rmse}")
    print(f"MAE : {mae}")

    # PLOT THR RESULTS
    signal_range = np.arange(0,100)
    signal_pred = model.predict(signal_range.reshape(-1,1))

    plt.figure(figsize=(12,8), dpi=200)
    sns.scatterplot(data = df, x="Signal",y="Density", color="black")

    plt.plot(signal_range,signal_pred)
```



```
model = LinearRegression()
run_model(model,X_train,y_train,X_test,y_test)
```

```
RMSE : 0.2570051996584629
MAE : 0.211198973318633
```

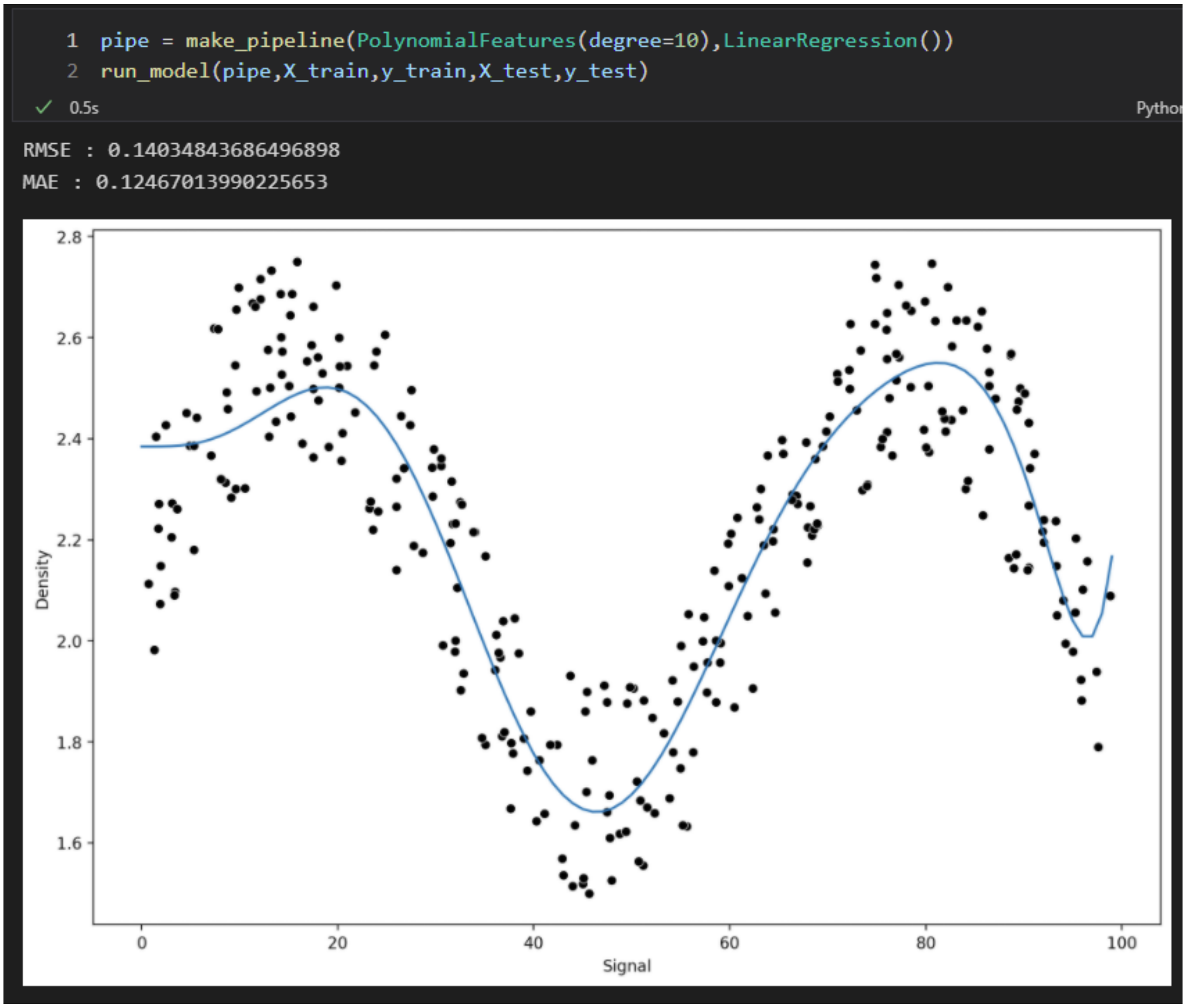


```python
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import PolynomialFeatures
```

```python
pipe = make_pipeline(PolynomialFeatures(degree=2),LinearRegression())
# 2nd degree poly model
```

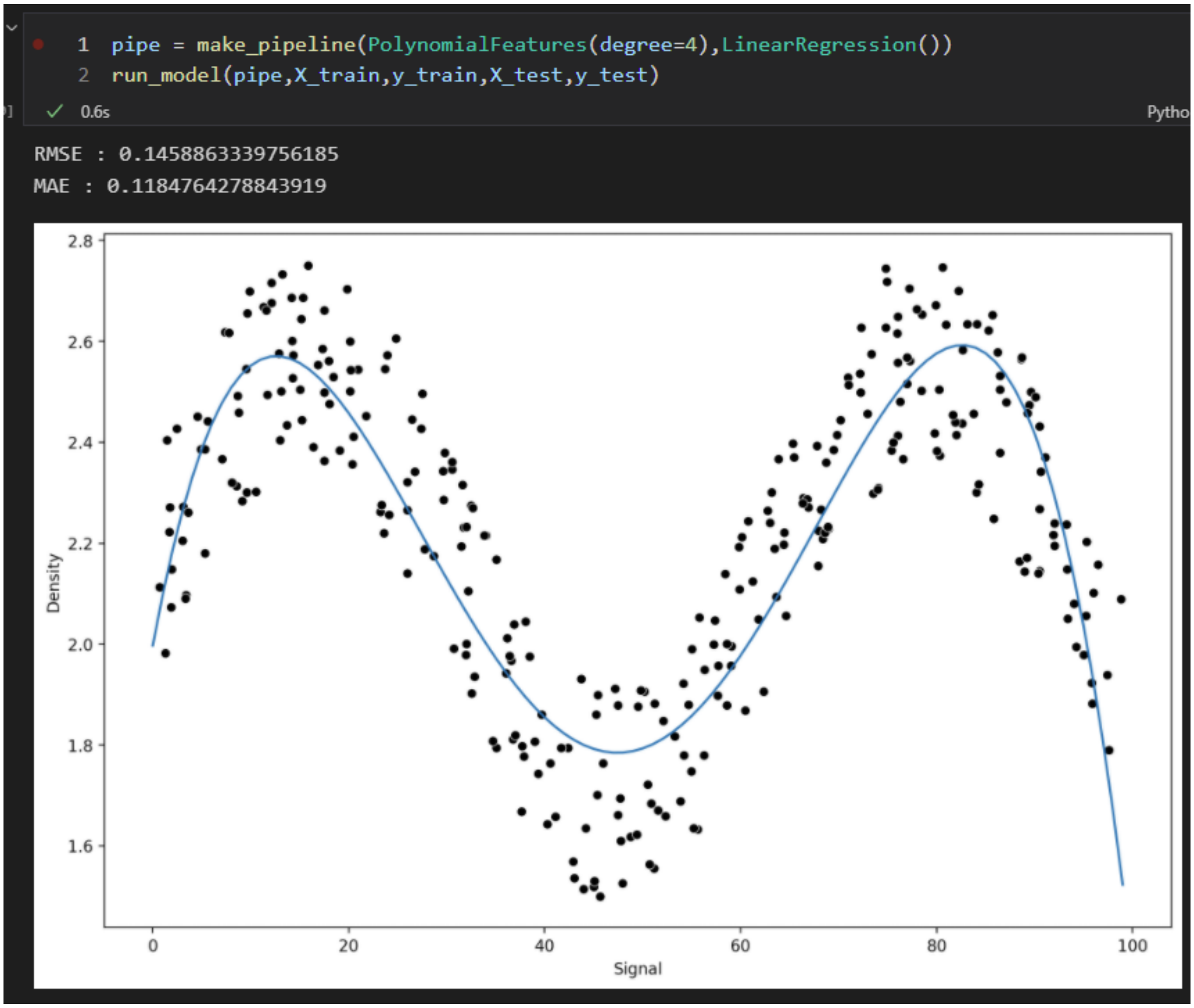
```python
run_model(pipe,X_train,y_train,X_test,y_test)
```

```
RMSE : 0.28173095637255463
MAE : 0.22903105443511165
```



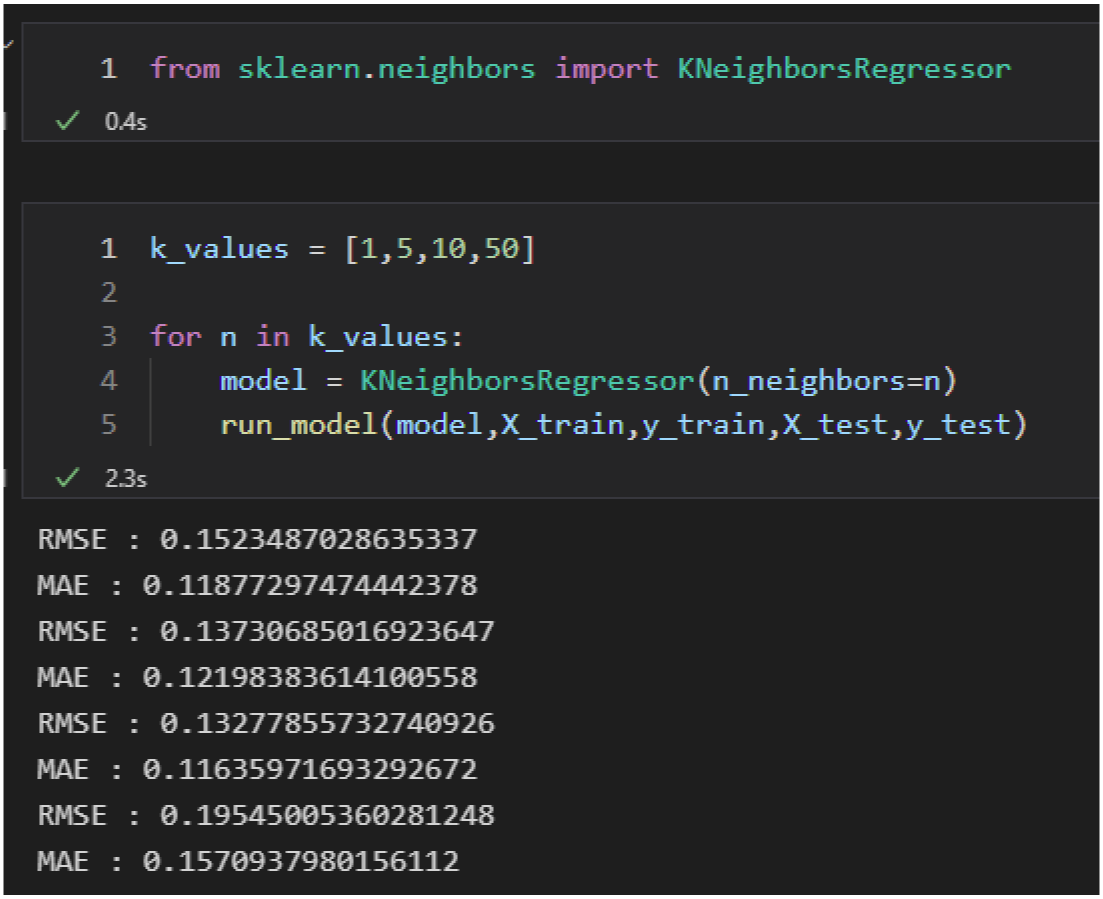
```
pipe = make_pipeline(PolynomialFeatures(degree=10),LinearRegression())
run_model(pipe,X_train,y_train,X_test,y_test)
```

```
RMSE : 0.14034843686496898
MAE : 0.12467013990225653
```



```
pipe = make_pipeline(PolynomialFeatures(degree=4),LinearRegression())
run_model(pipe,X_train,y_train,X_test,y_test)
```

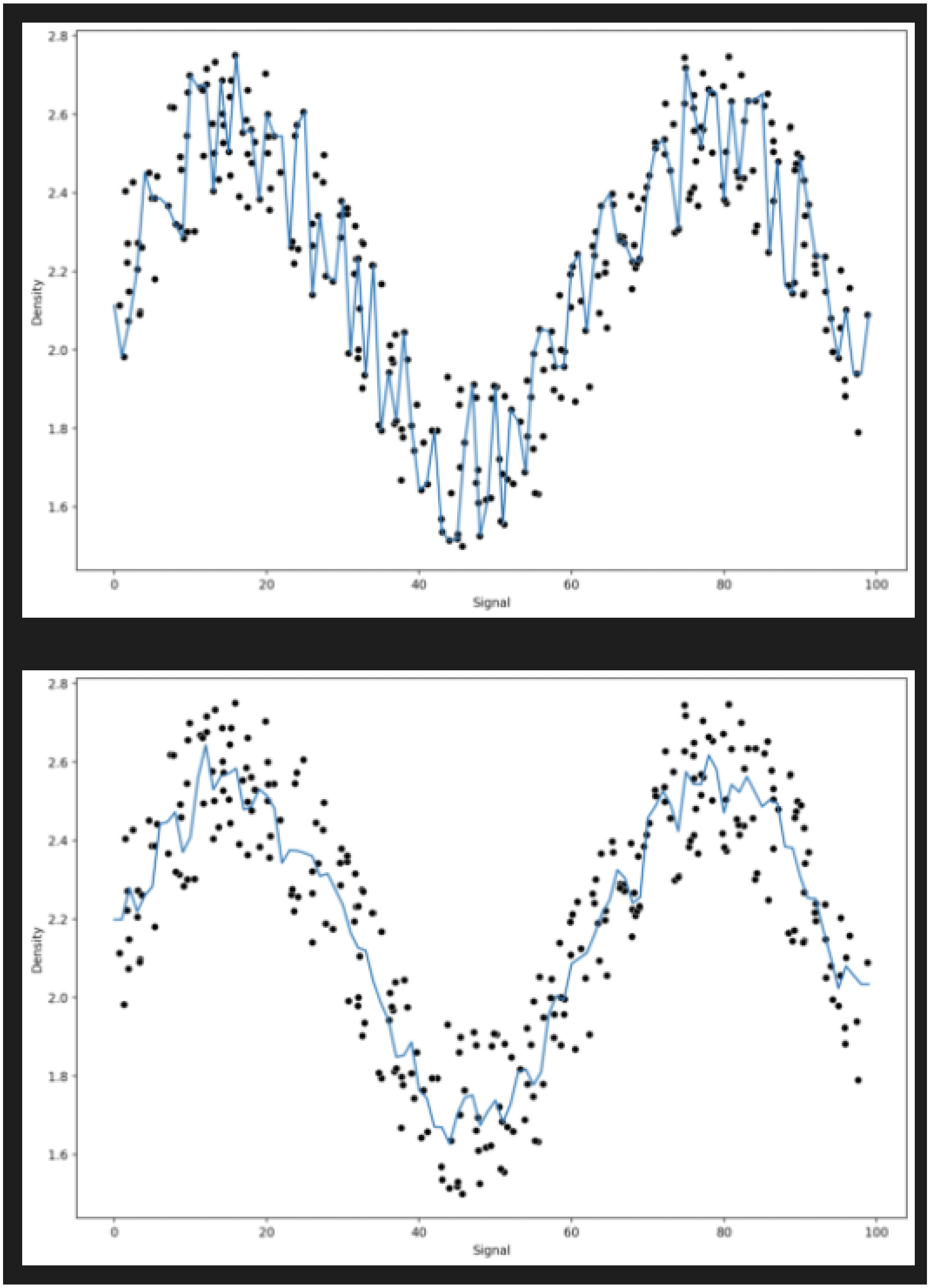
```
RMSE : 0.1458863339756185
MAE : 0.1184764278843919
```

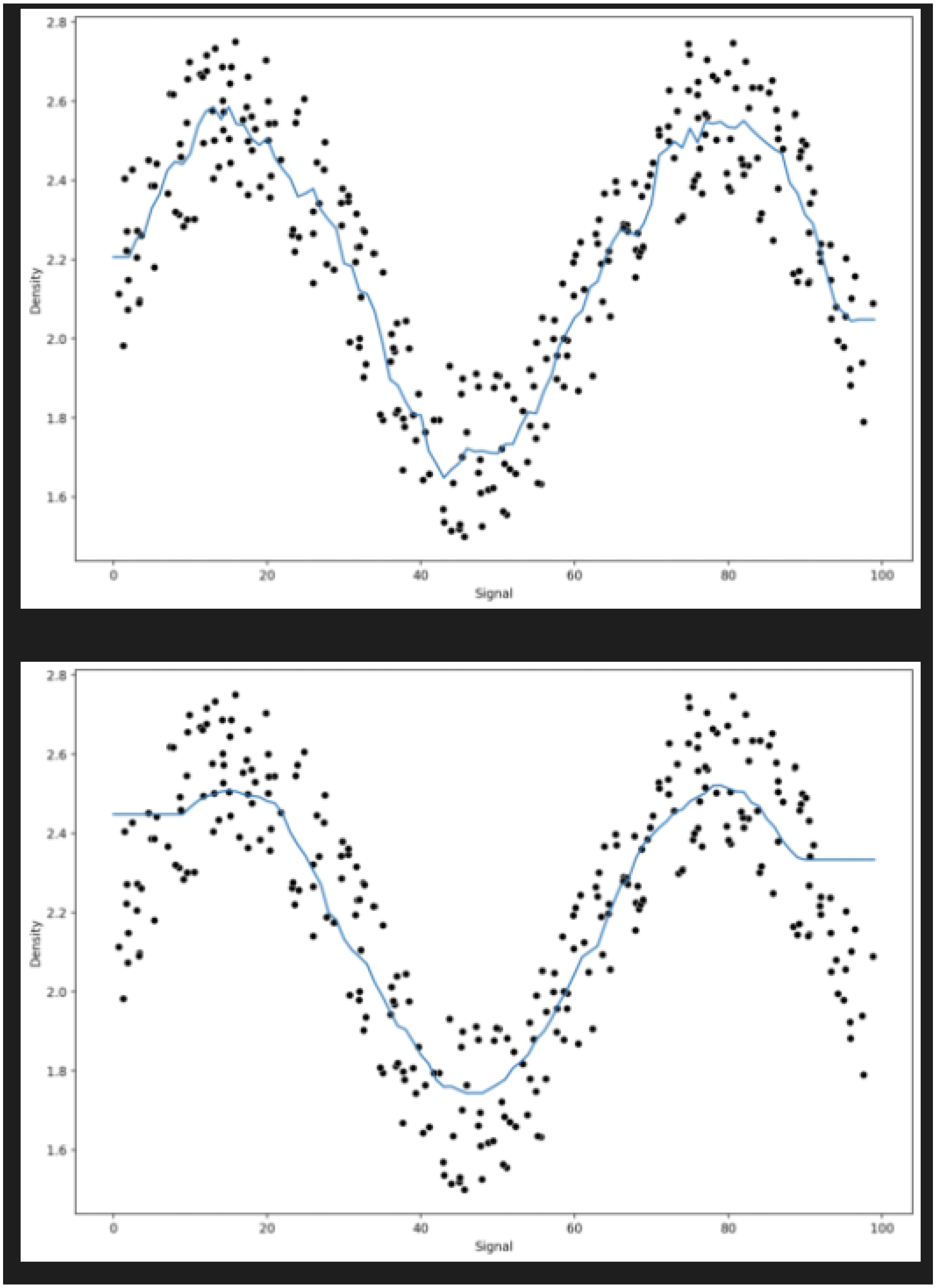


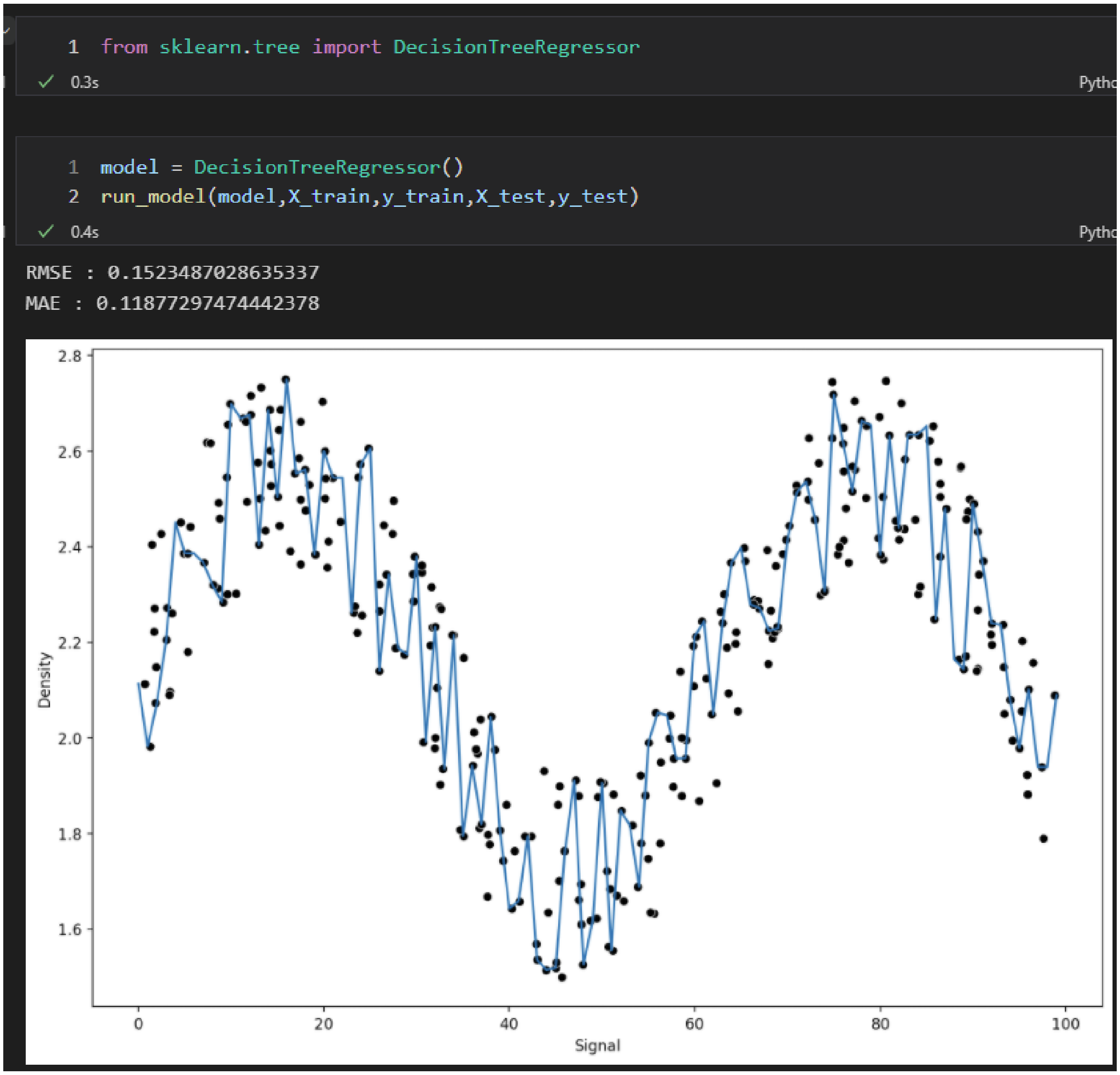
```python
from sklearn.neighbors import KNeighborsRegressor
```

```python
k_values = [1,5,10,50]

for n in k_values:
    model = KNeighborsRegressor(n_neighbors=n)
    run_model(model,X_train,y_train,X_test,y_test)
```

```
RMSE : 0.1523487028635337
MAE : 0.11877297474442378
RMSE : 0.13730685016923647
MAE : 0.12198383614100558
RMSE : 0.13277855732740926
MAE : 0.11635971693292672
RMSE : 0.19545005360281248
MAE : 0.1570937980156112
```


2.8
2.6
2.4
2.2
2.0
1.8
1.6
Density
0
20
40
60
80
100
Signal
2.8
2.6
2.4
2.2
2.0
1.8
1.6
Density
0
20
40
60
80
100
Signal

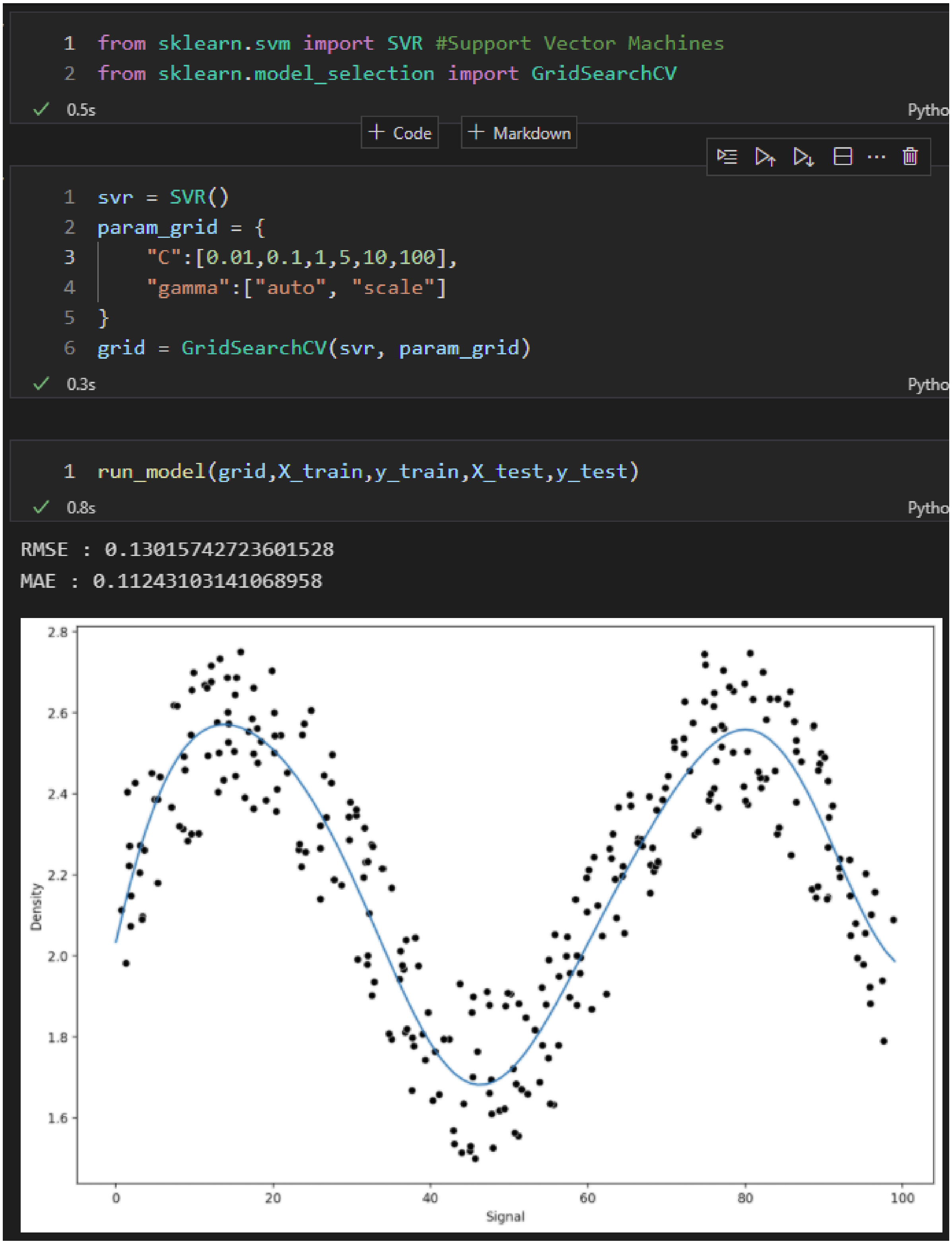

2.8
2.6
2.4
2.2
2.0
1.8
1.6
Density
0
20
40
60
80
100
Signal
2.8
2.6
2.4
2.2
2.0
1.8
1.6
Density
0
20
40
60
80
100
Signal



```
from sklearn.tree import DecisionTreeRegressor
```

```
model = DecisionTreeRegressor()
run_model(model,X_train,y_train,X_test,y_test)
```

```
RMSE : 0.1523487028635337
MAE : 0.11877297474442378
```

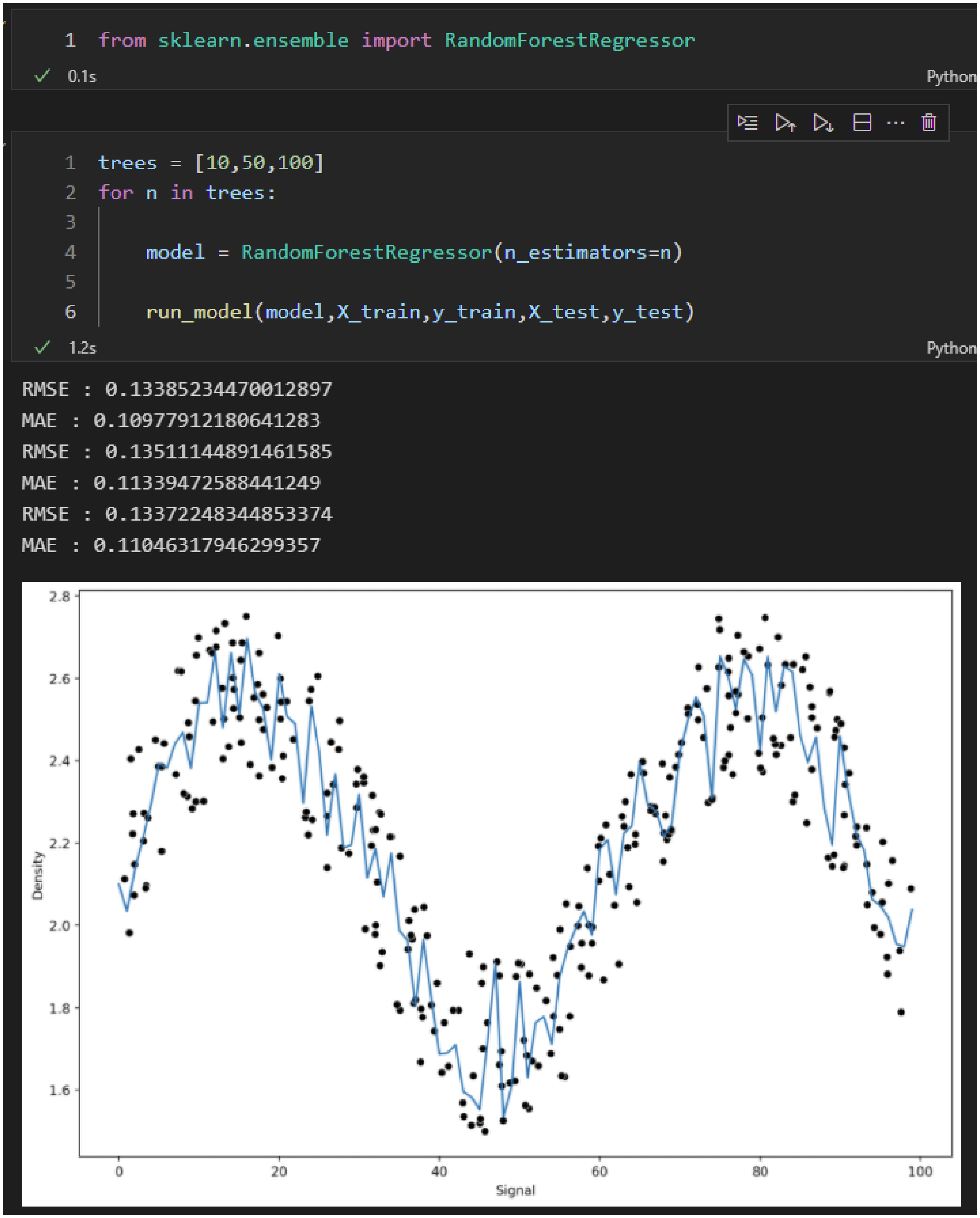


```
from sklearn.svm import SVR #Support Vector Machines
from sklearn.model_selection import GridSearchCV
```

```
svr = SVR()
param_grid = {
    "C":[0.01,0.1,1,5,10,100],
    "gamma":["auto", "scale"]
}
grid = GridSearchCV(svr, param_grid)
```

```
run_model(grid,X_train,y_train,X_test,y_test)
```

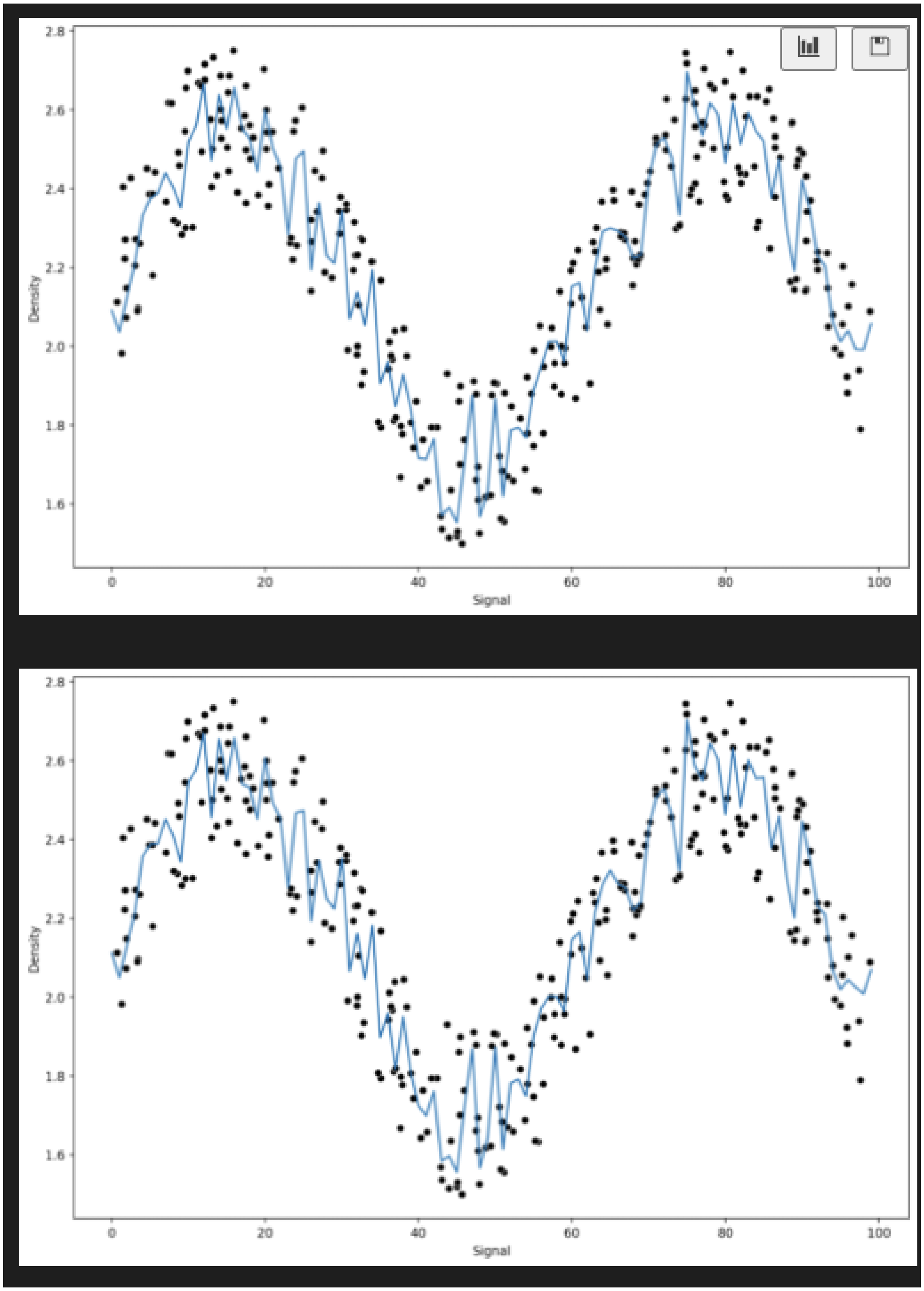
```
RMSE : 0.13015742723601528
MAE : 0.11243103141068958
```

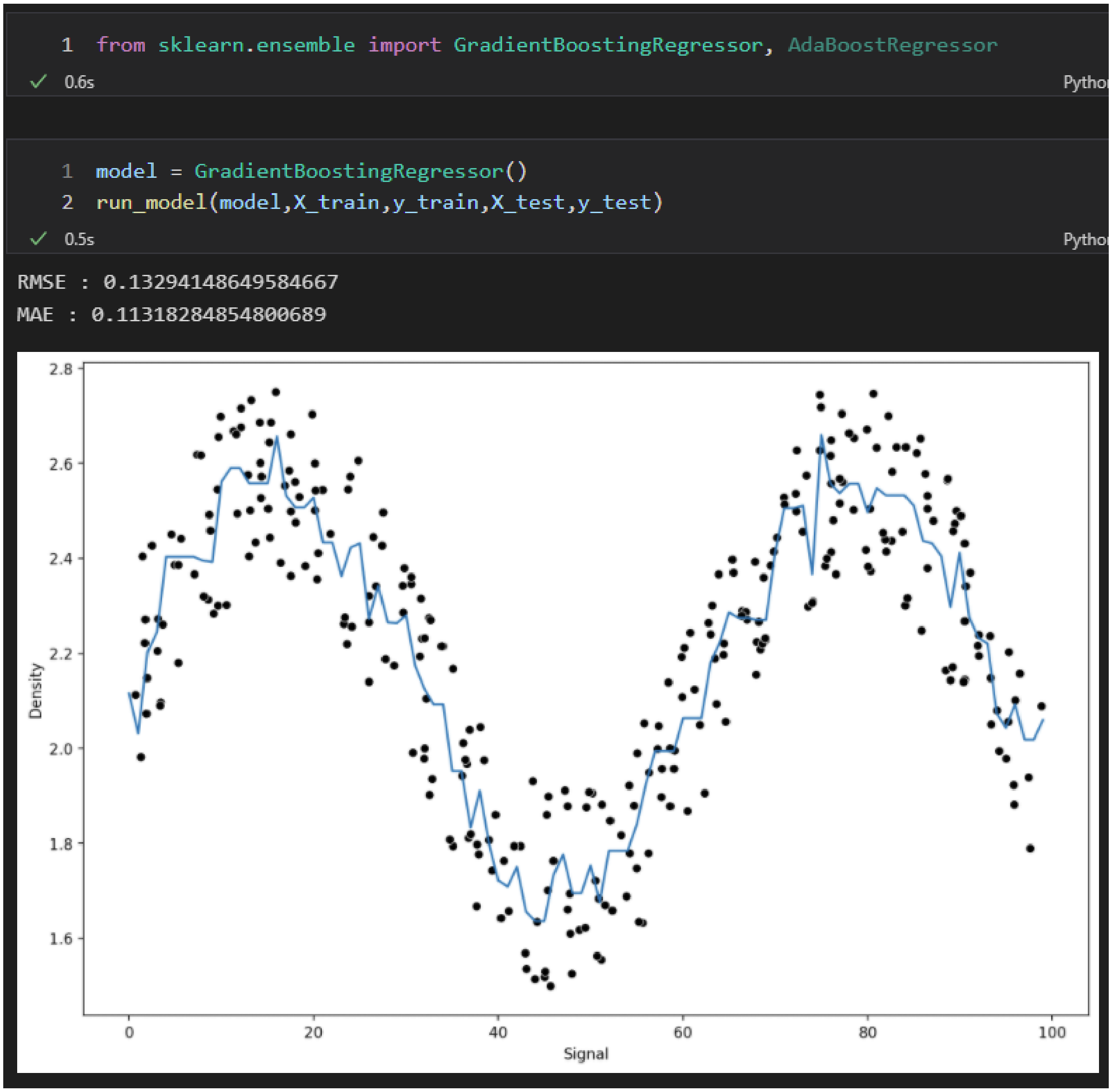


```python
from sklearn.ensemble import RandomForestRegressor
```

```python
trees = [10,50,100]
for n in trees:

    model = RandomForestRegressor(n_estimators=n)

    run_model(model,X_train,y_train,X_test,y_test)
```

```
RMSE : 0.13385234470012897
MAE  : 0.10977912180641283
RMSE : 0.13511144891461585
MAE  : 0.11339472588441249
RMSE : 0.13372248344853374
MAE  : 0.11046317946299357
```

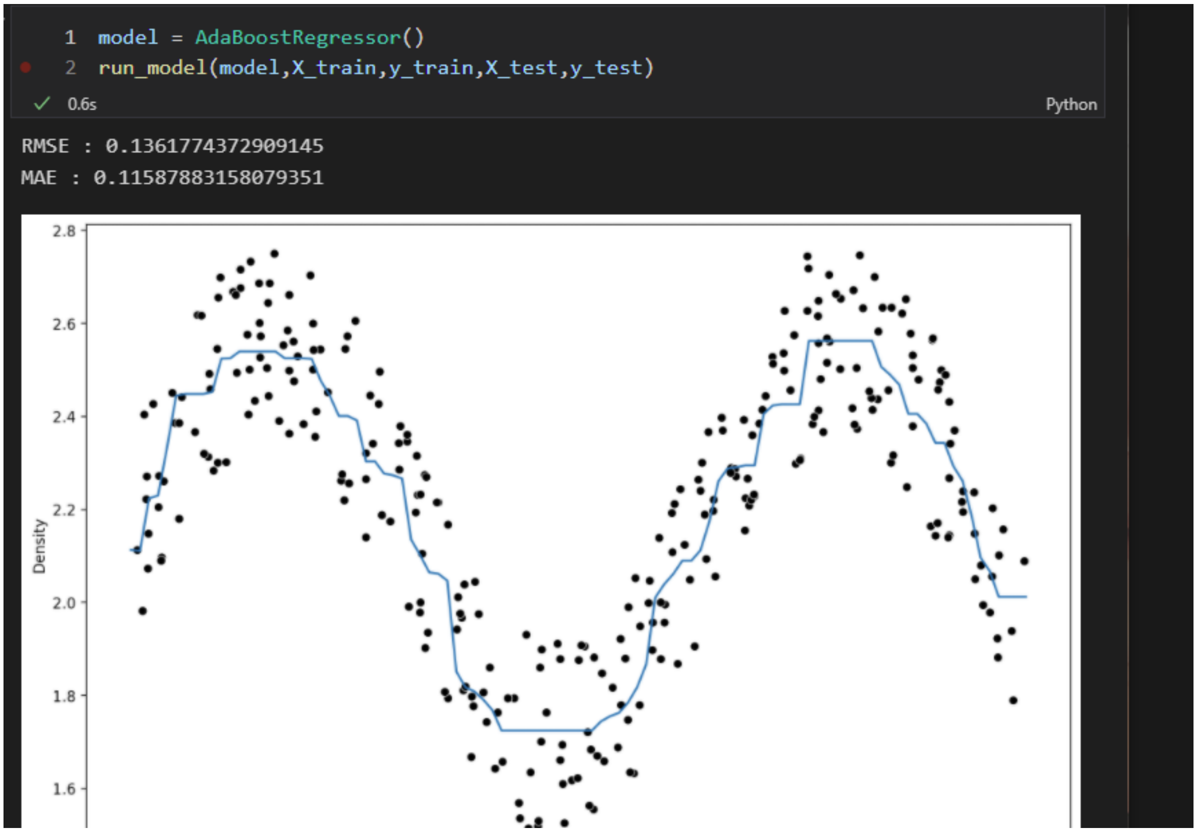

2.8
2.6
2.4
2.2
2.0
1.8
1.6
Density
0
20
40
60
80
100
Signal
2.8
2.6
2.4
2.2
2.0
1.8
1.6
Density
0
20
40
60
80
100
Signal



```
from sklearn.ensemble import GradientBoostingRegressor, AdaBoostRegressor
```

```
model = GradientBoostingRegressor()
run_model(model,X_train,y_train,X_test,y_test)
```

```
RMSE : 0.13294148649584667
MAE : 0.11318284854800689
```



```
model = AdaBoostRegressor()
run_model(model,X_train,y_train,X_test,y_test)
```

```
RMSE : 0.1361774372909145
MAE : 0.11587883158079351
```