Ex. 3

31-7-25

Study of Classifiers with respect to Statistical Parameters

Aim:-

To compare the perfomance of different classifiers using statistical parameters such as F1_score, recall, confusion matrix.

Objective:-

* To implement and evaluate three classifiers
    * KNN
    * SVM
    * Navie Bayes
* To compare their performance
* To identify the most sutable model for the Iris dataset.

Pseudo code:

```
Start
    1. Import and load the Iris dataset
-> Extract featurs (x) and target labels (y)
-> split the dataset into traing and testing
-> Standardise the features using Standard-
   -Scalar
-> Initialize the classifiers
   model 1 = KNeighborClassifier(n_neighbors=5)
   model 2 = Svc (kernel='linear')
```

model 3 = GaussianNB ()

→ Train all models

- Model 1 fit (x-train, y-train)
- Model 2 . fit(x-train, y-train)
- model 3. fit(x-train, y-train)

7. Predict the test data using each classifier

8. Evaluate each model

Observation :-

SVM :-

Classification Report

	Precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	0.89	0.94	9
2	0.92	1.00	0.96	11
accuracy			0.97	30
macro avg	0.97	0.96	0.97	30
weighted avg	0.97	0.97	0.97	30

Navie Bayes

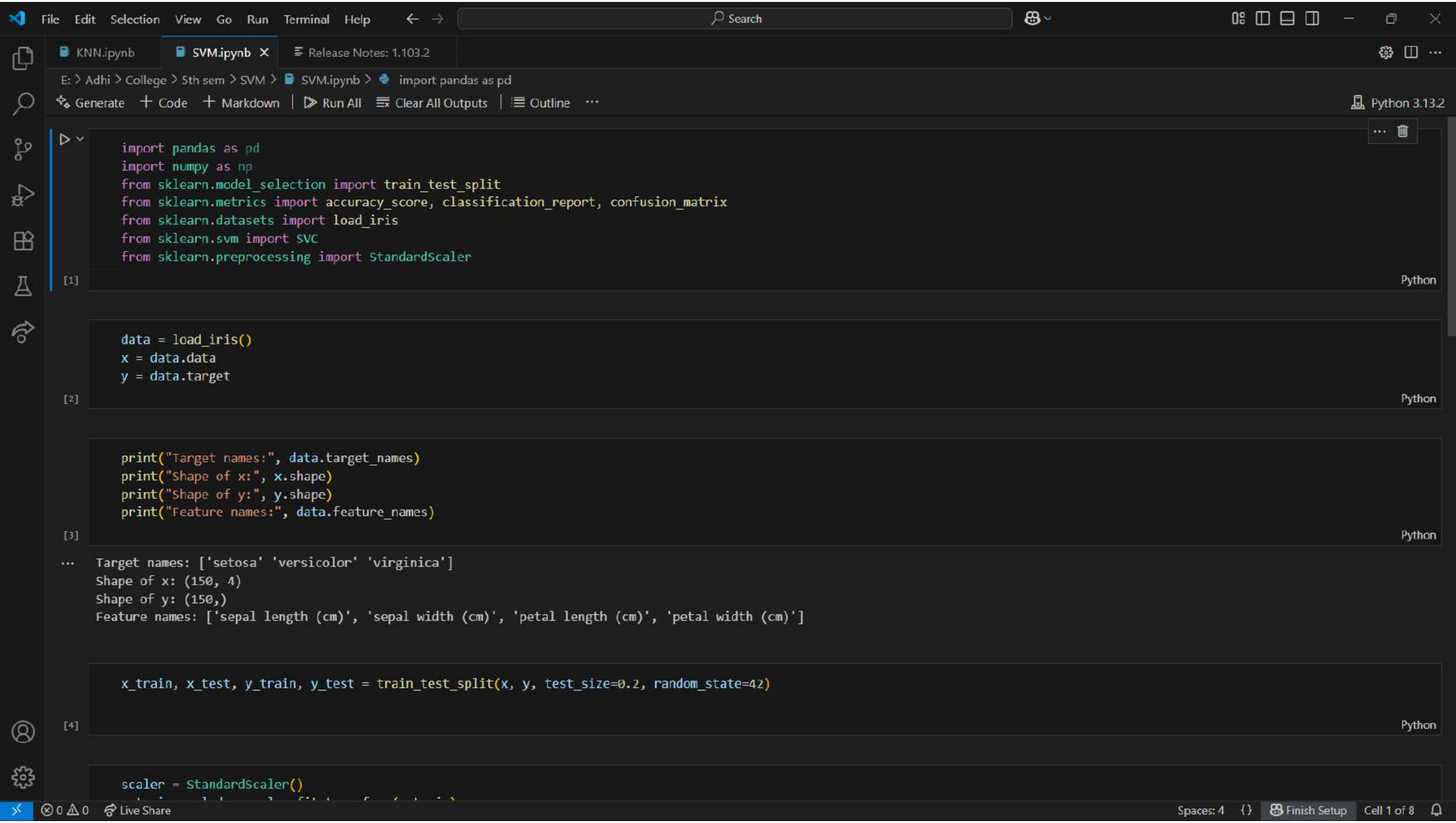
Classification Report:

	precision	recall	f1-score	Support
0	1.00	1.00	1.00	19
1	1.00	0.92	0.96	13
2	0.93	1.00	0.96	13
acuracy			0.98	45
macro avg	0.98	0.97	0.97	45
weighted avg	0.98	0.98	0.98	45

11/7/25

Result:-

The code has executed sucessfully and Navie Bayes performed well on Iris dataset.


File Edit Selection View Go Run Terminal Help
Search
KNN.ipynb
SVM.ipynb
Release Notes: 1.103.2
E: > Adhi > College > 5th sem > SVM > SVM.ipynb > import pandas as pd
Generate + Code + Markdown | Run All Clear All Outputs | Outline ···
Python 3.13.2

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.datasets import load_iris
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
```

[1] Python

```python
data = load_iris()
x = data.data
y = data.target
```

[2] Python

```python
print("Target names:", data.target_names)
print("Shape of x:", x.shape)
print("Shape of y:", y.shape)
print("Feature names:", data.feature_names)
```

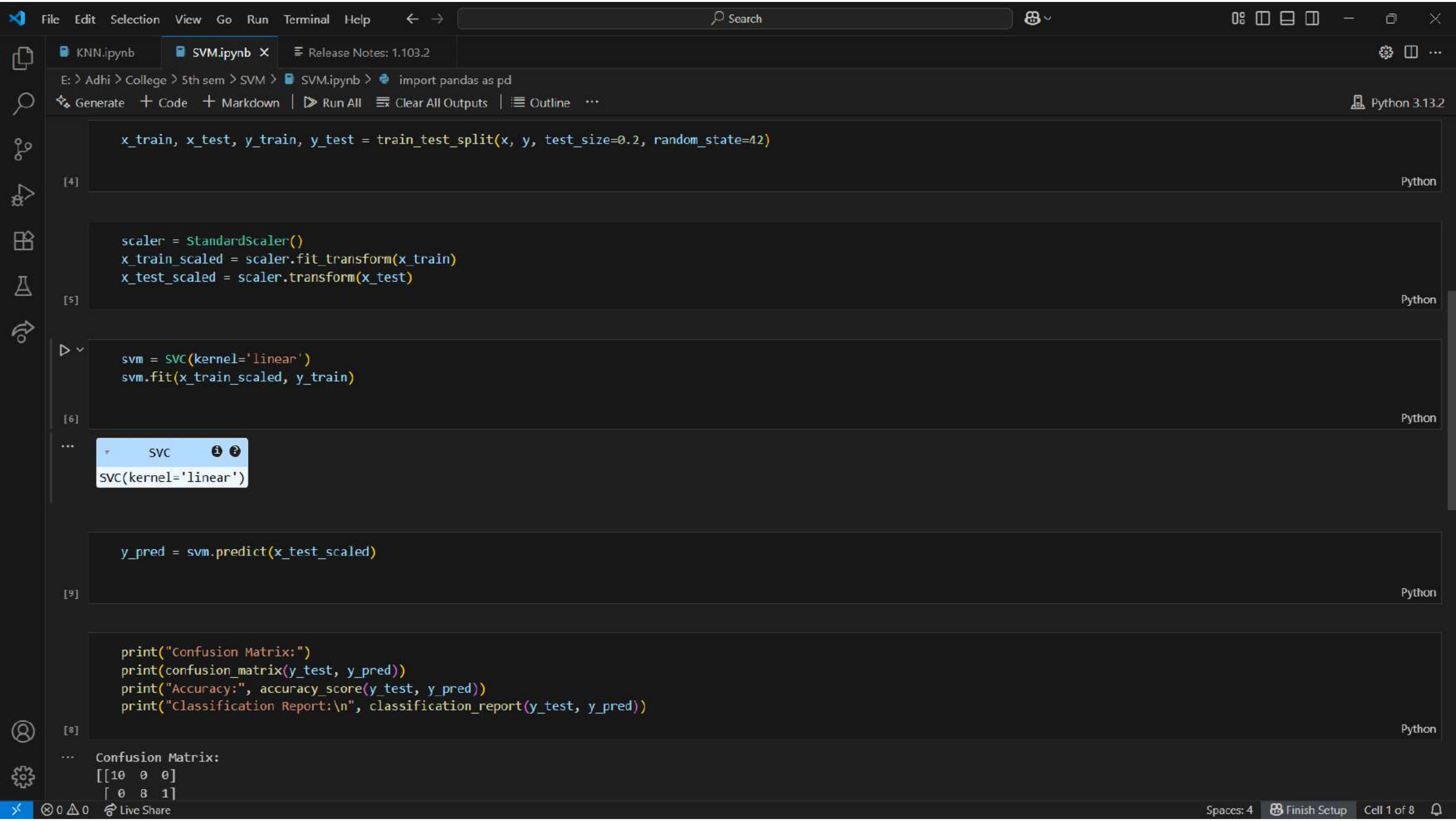
[3] Python

```
Target names: ['setosa' 'versicolor' 'virginica']
Shape of x: (150, 4)
Shape of y: (150,)
Feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

```python
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

[4] Python

```python
scaler = StandardScaler()
```



```
[4]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

```
[5]
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

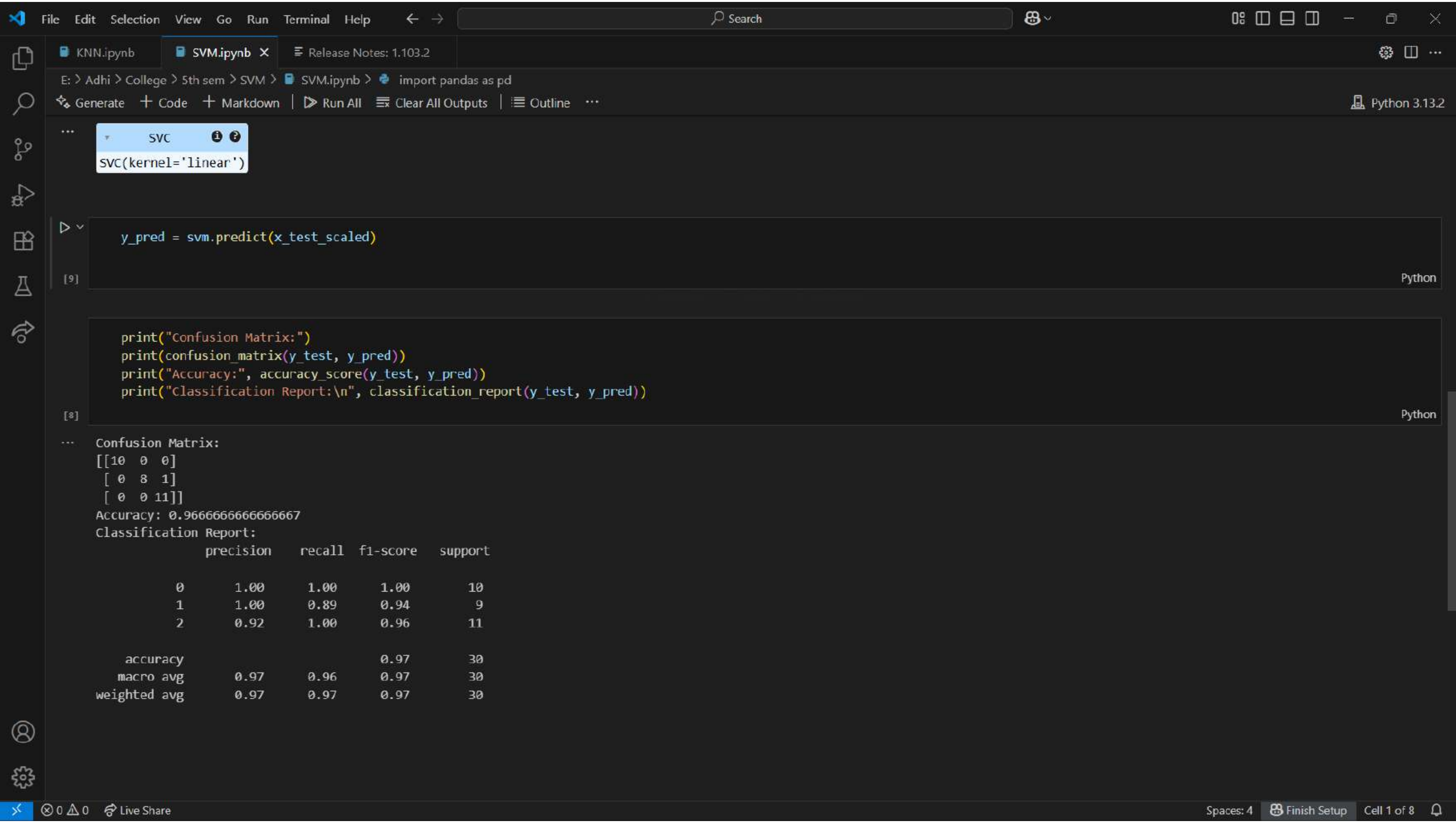
```
[6]
svm = SVC(kernel='linear')
svm.fit(x_train_scaled, y_train)
```

```
SVC
SVC(kernel='linear')
```

```
[9]
y_pred = svm.predict(x_test_scaled)
```

```
[8]
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Confusion Matrix:
[[10  0  0]
 [ 0  8  1]
```



```
SVC(kernel='linear')
```

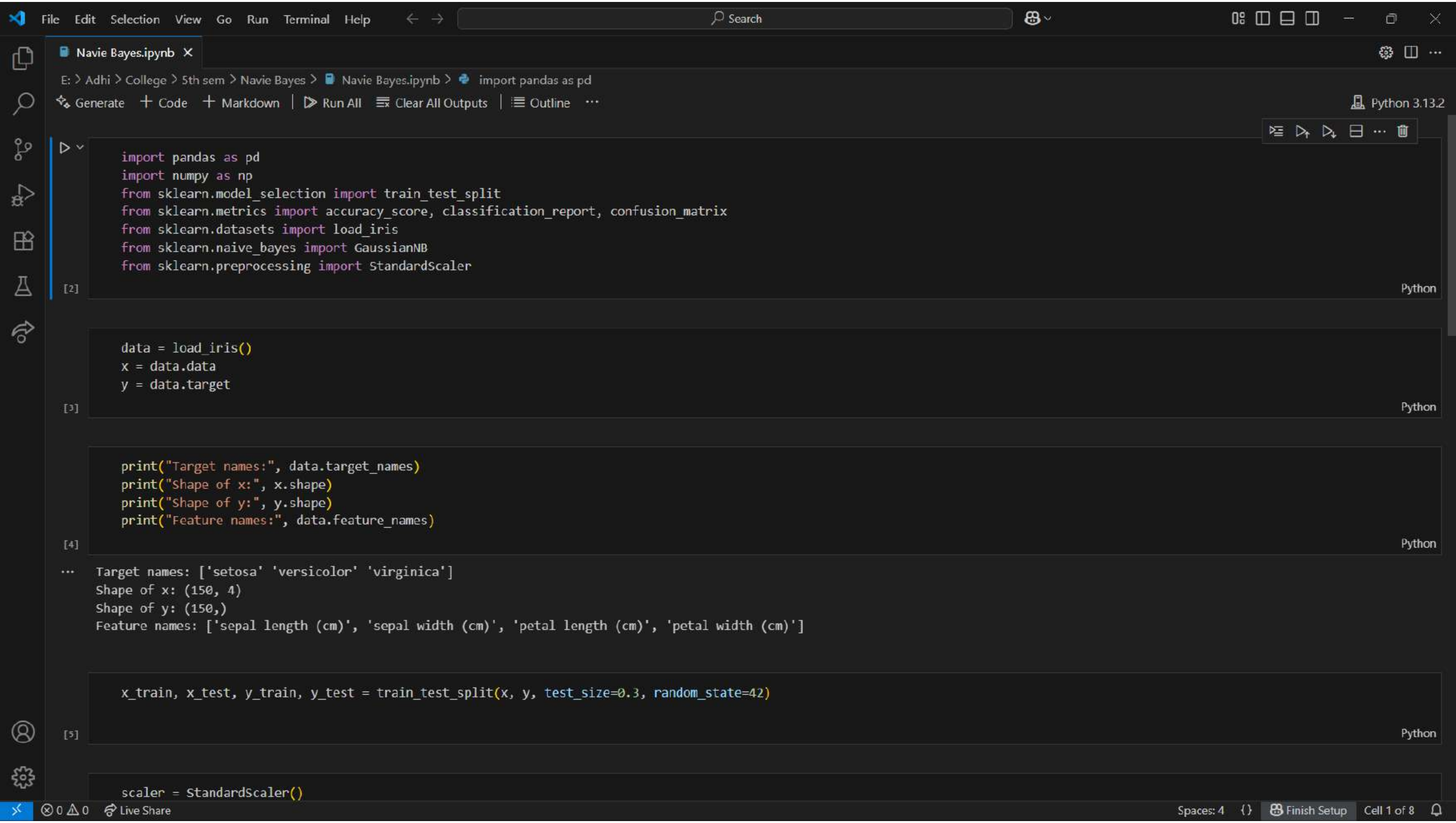
```
[9]
y_pred = svm.predict(x_test_scaled)
```

```
[8]
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Confusion Matrix:
[[10  0  0]
 [ 0  8  1]
 [ 0  0 11]]
Accuracy: 0.9666666666666667
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.89      0.94         9
           2       0.92      1.00      0.96        11

    accuracy                           0.97        30
   macro avg       0.97      0.96      0.97        30
weighted avg       0.97      0.97      0.97        30
```


File Edit Selection View Go Run Terminal Help
Search
Navie Bayes.ipynb
E: > Adhi > College > 5th sem > Navie Bayes > Navie Bayes.ipynb > import pandas as pd
Generate + Code + Markdown Run All Clear All Outputs Outline
Python 3.13.2

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.datasets import load_iris
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import StandardScaler
```

[2] Python

```
data = load_iris()
x = data.data
y = data.target
```

[3] Python

```
print("Target names:", data.target_names)
print("Shape of x:", x.shape)
print("Shape of y:", y.shape)
print("Feature names:", data.feature_names)
```

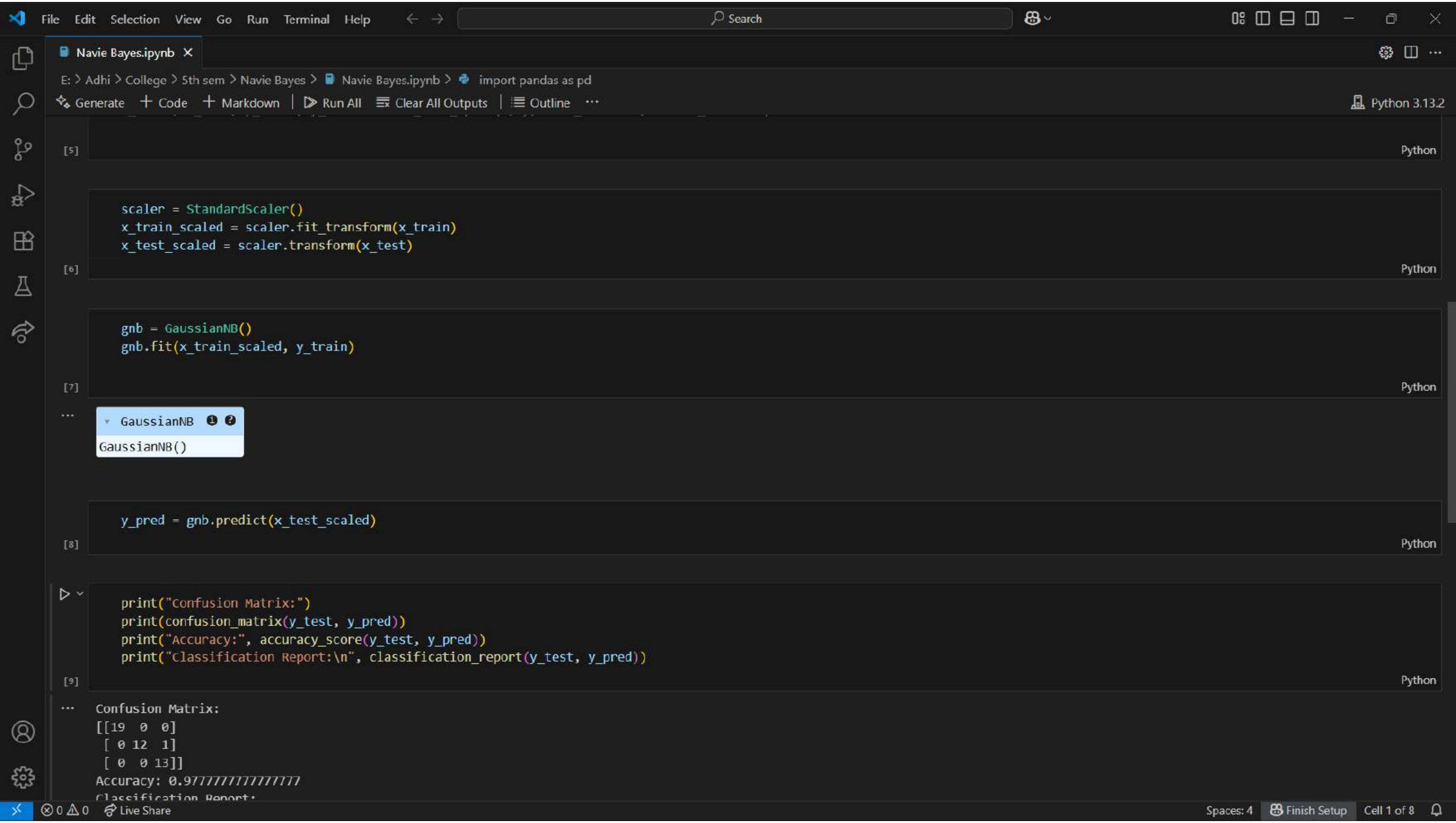
[4] Python

```
Target names: ['setosa' 'versicolor' 'virginica']
Shape of x: (150, 4)
Shape of y: (150,)
Feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

[5] Python

```
scaler = StandardScaler()
```



```python
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

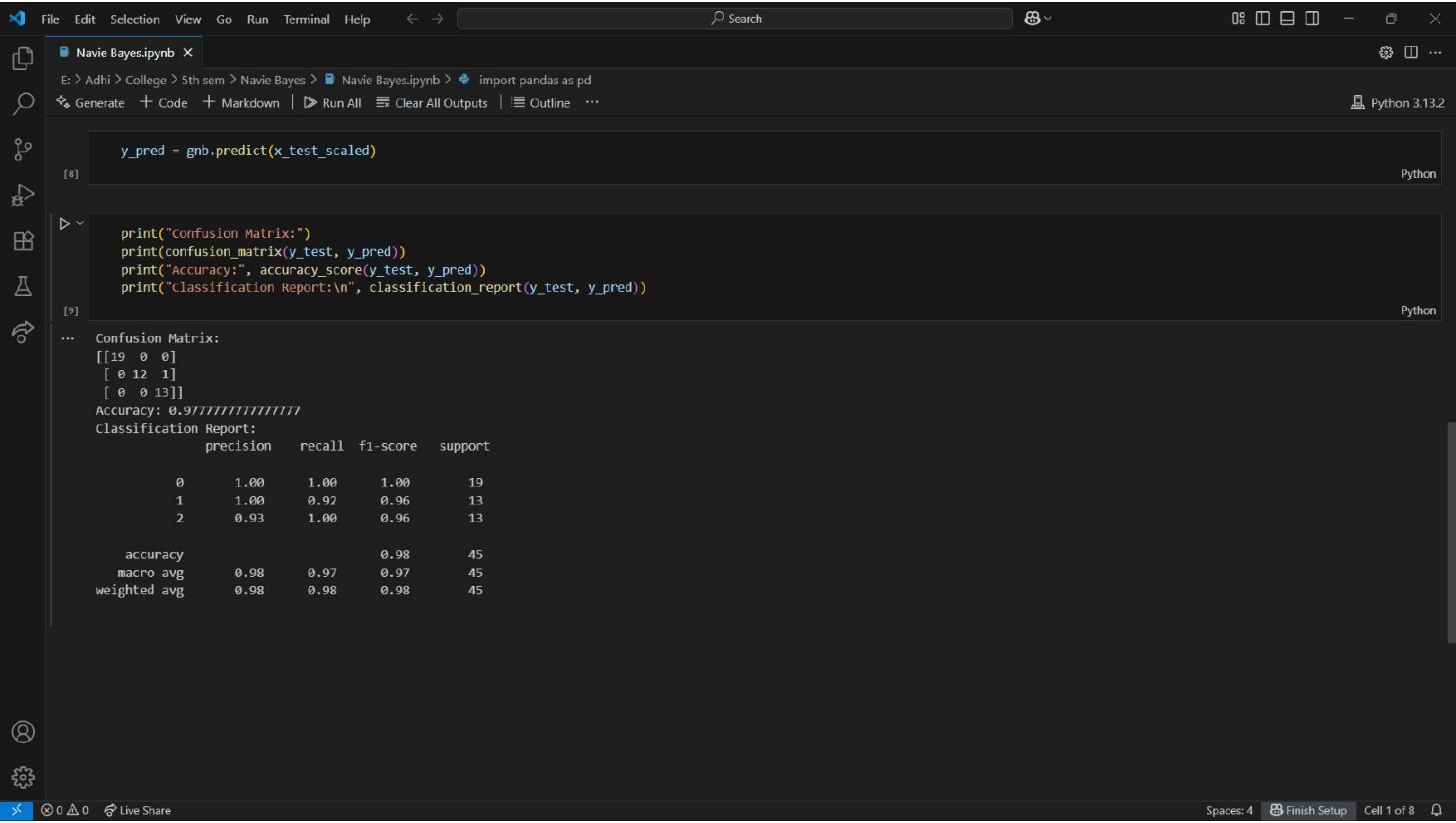
```python
gnb = GaussianNB()
gnb.fit(x_train_scaled, y_train)
```

```
GaussianNB
GaussianNB()
```

```python
y_pred = gnb.predict(x_test_scaled)
```

```python
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Confusion Matrix:
[[19  0  0]
 [ 0 12  1]
 [ 0  0 13]]
Accuracy: 0.9777777777777777
Classification Report:
```



```
y_pred = gnb.predict(x_test_scaled)
```

[8]

```
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

[9]

```
Confusion Matrix:
[[19  0  0]
 [ 0 12  1]
 [ 0  0 13]]
Accuracy: 0.9777777777777777
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      0.92      0.96        13
           2       0.93      1.00      0.96        13

    accuracy                           0.98        45
   macro avg       0.98      0.97      0.97        45
weighted avg       0.98      0.98      0.98        45
```