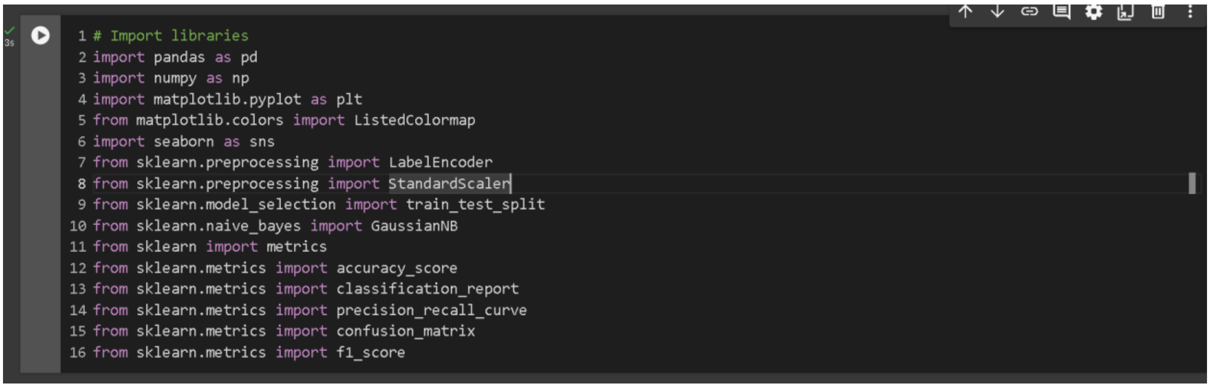
NAME: MARK LOPES

ROLL NO: 9913

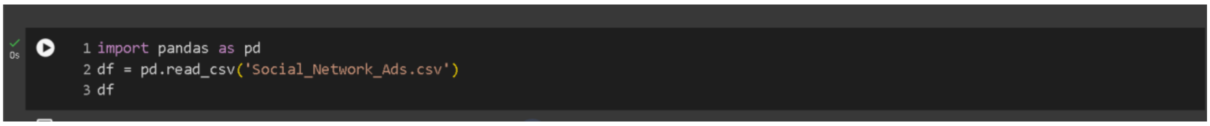
BRANCH: COMPUTER A

EXP 6: NAIVEBAYES CLASSIFICATION

Using the social network ads data concerning the Gender, Age, and Estimated Salary of several users and based on these data we would classify each user whether they would purchase the insurance or not.

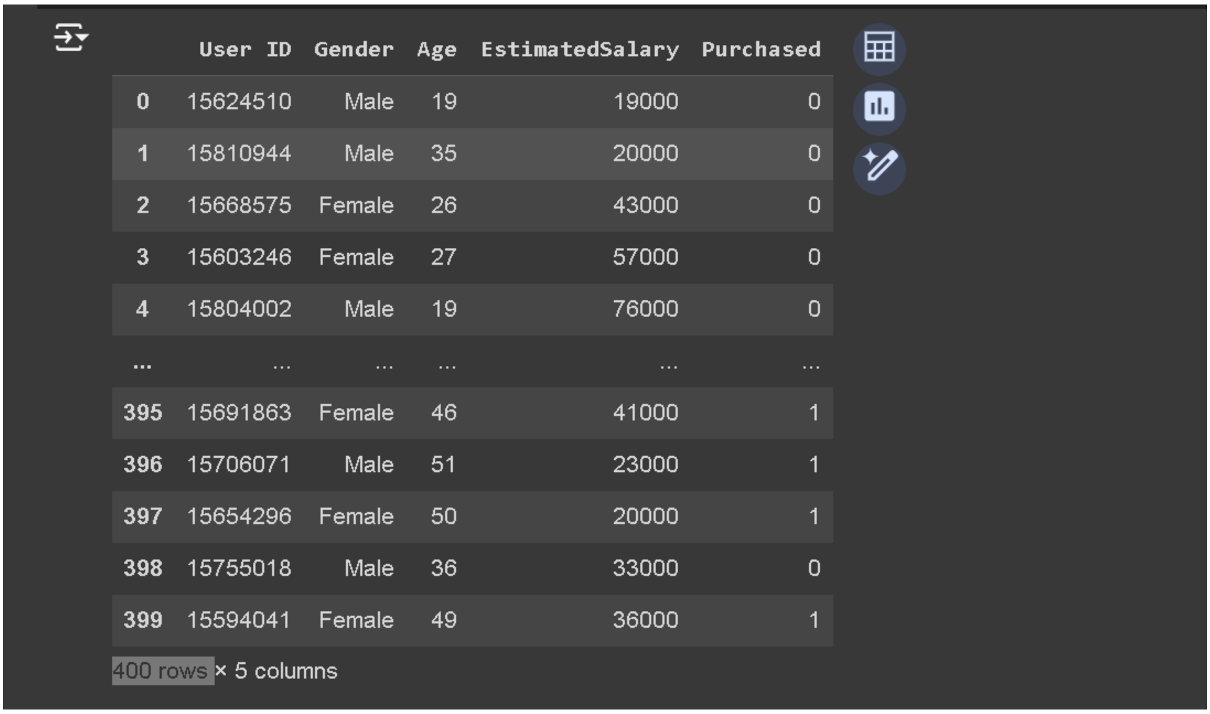


```python
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
```

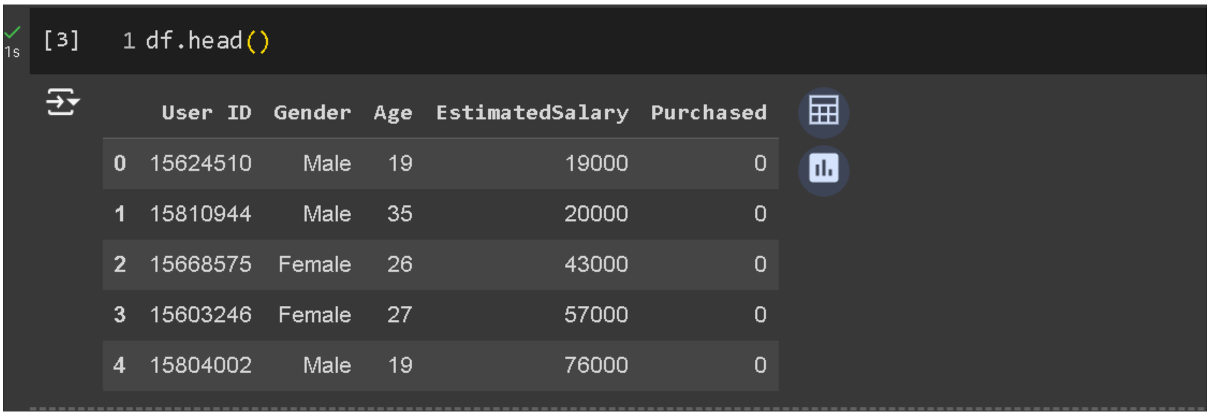


```python
import pandas as pd
df = pd.read_csv('Social_Network_Ads.csv')
df
```

Output:

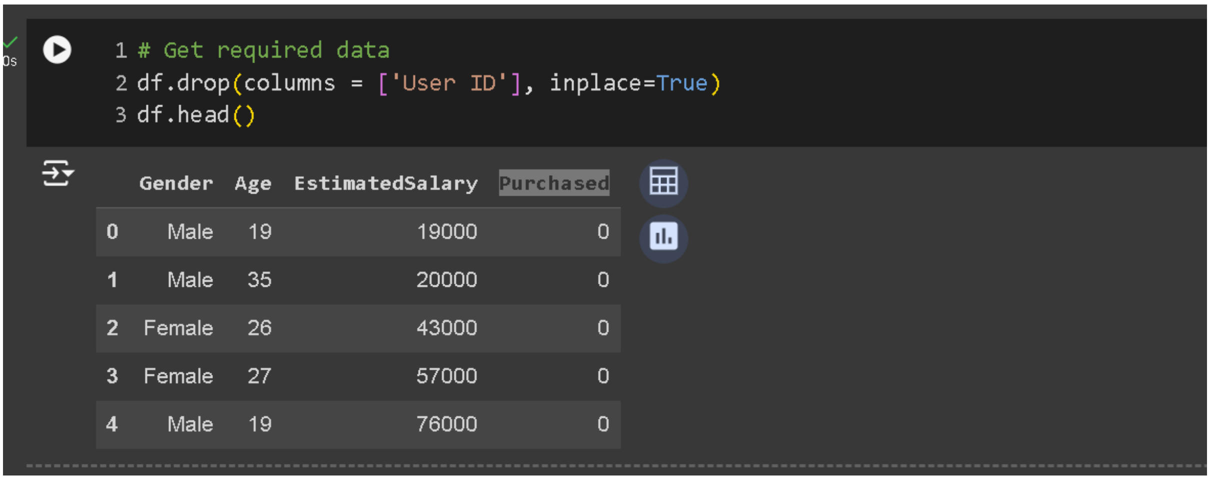


```
      User ID  Gender  Age  EstimatedSalary  Purchased
0    15624510    Male   19            19000          0
1    15810944    Male   35            20000          0
2    15668575  Female   26            43000          0
3    15603246  Female   27            57000          0
4    15804002    Male   19            76000          0
...       ...     ...  ...              ...        ...
395  15691863  Female   46            41000          1
396  15706071    Male   51            23000          1
397  15654296  Female   50            20000          1
398  15755018    Male   36            33000          0
399  15594041  Female   49            36000          1

400 rows × 5 columns
```



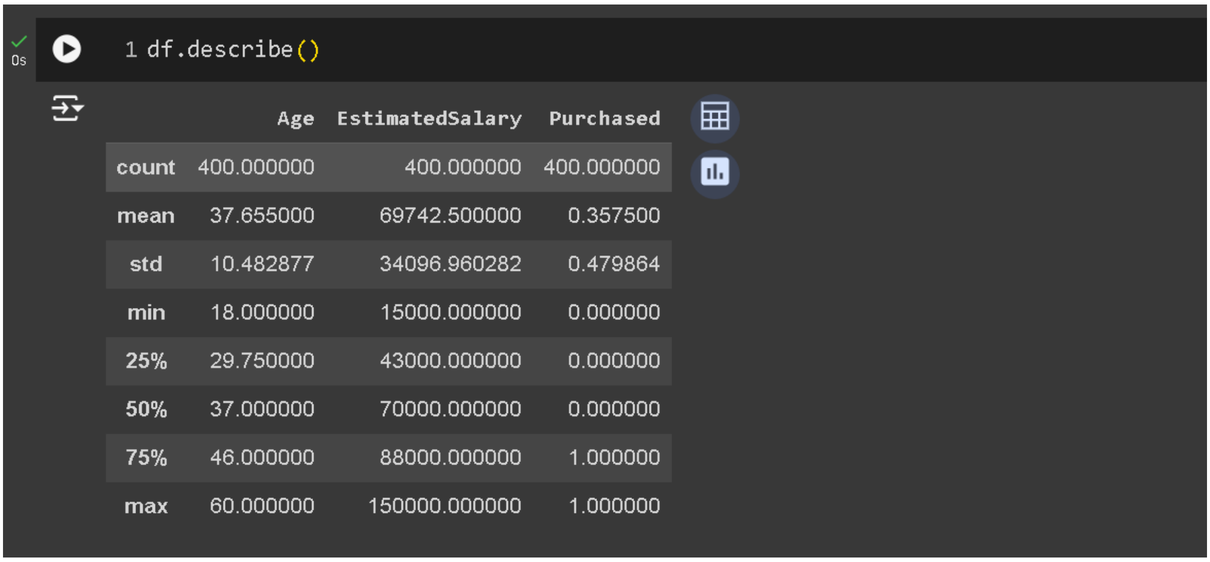
```
df.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0



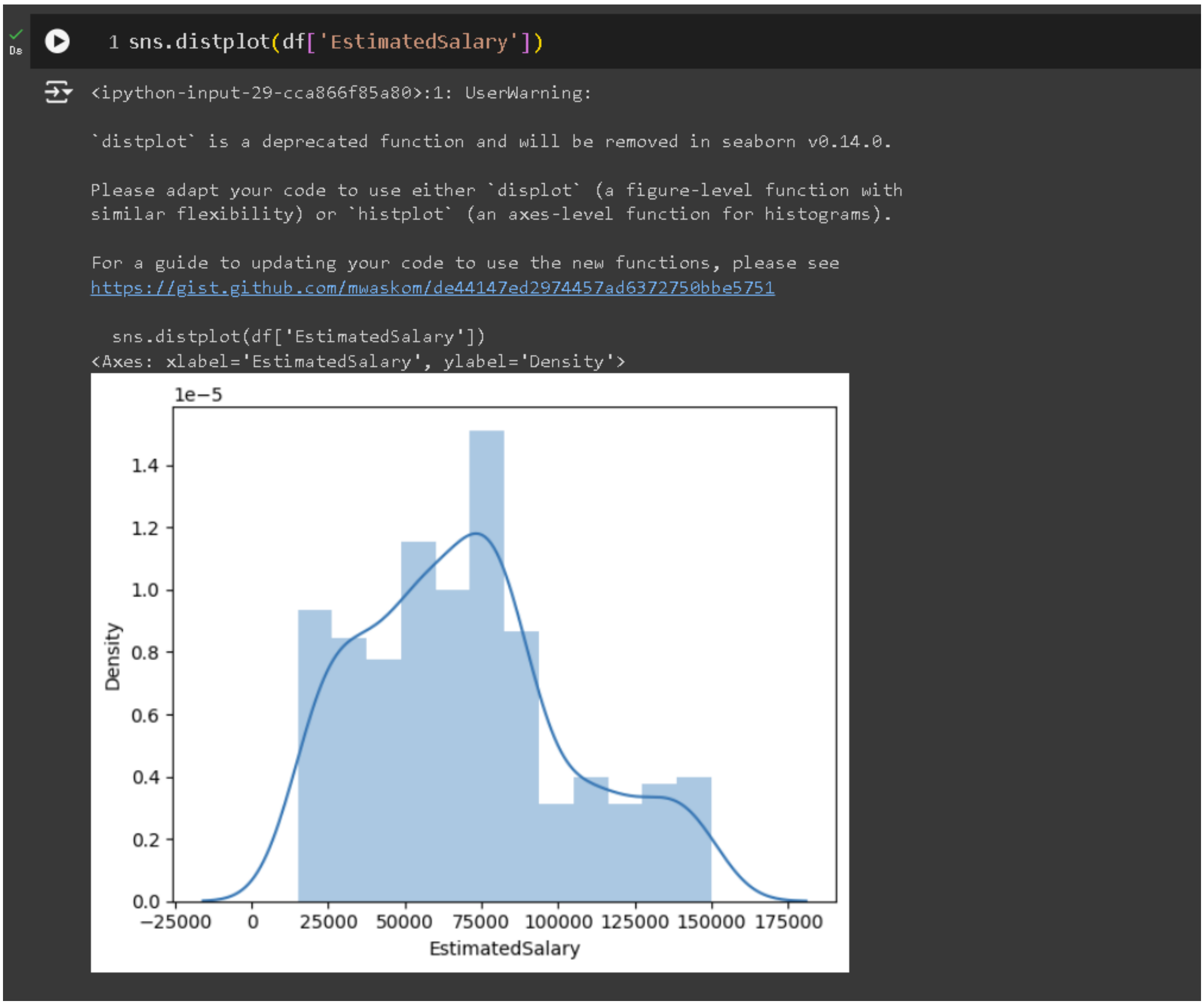
```
# Get required data
df.drop(columns = ['User ID'], inplace=True)
df.head()
```

	Gender	Age	EstimatedSalary	Purchased
0	Male	19	19000	0
1	Male	35	20000	0
2	Female	26	43000	0
3	Female	27	57000	0
4	Male	19	76000	0



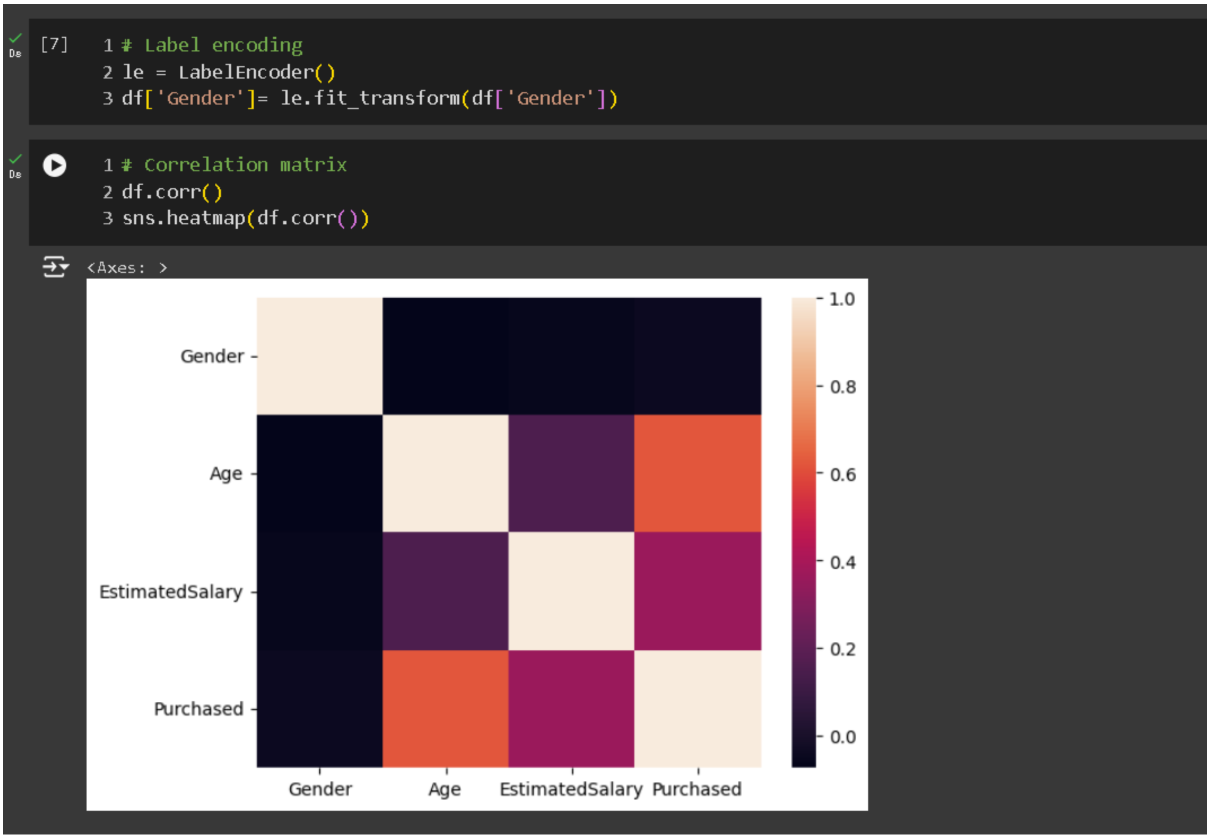
```
df.describe()
```

	Age	EstimatedSalary	Purchased
count	400.000000	400.000000	400.000000
mean	37.655000	69742.500000	0.357500
std	10.482877	34096.960282	0.479864
min	18.000000	15000.000000	0.000000
25%	29.750000	43000.000000	0.000000
50%	37.000000	70000.000000	0.000000
75%	46.000000	88000.000000	1.000000
max	60.000000	150000.000000	1.000000



```
sns.distplot(df['EstimatedSalary'])
```

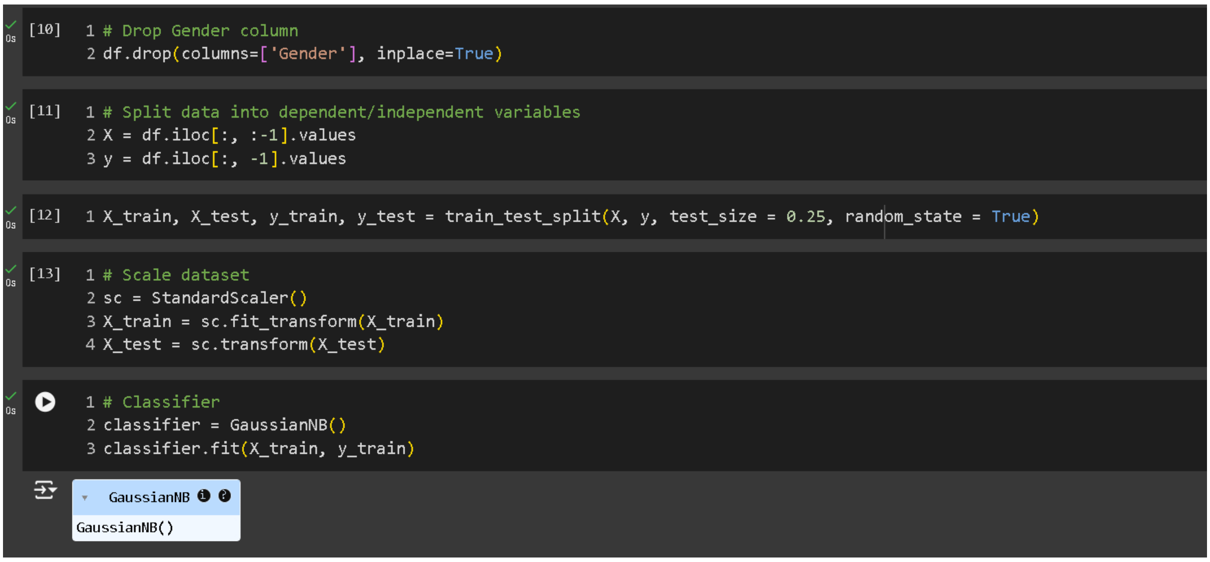
```
<ipython-input-29-cca866f85a80>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['EstimatedSalary'])
<Axes: xlabel='EstimatedSalary', ylabel='Density'>
```



```
[7]
# Label encoding
le = LabelEncoder()
df['Gender']= le.fit_transform(df['Gender'])
```

```
# Correlation matrix
df.corr()
sns.heatmap(df.corr())
```

```
<Axes: >
```



```python
# Drop Gender column
df.drop(columns=['Gender'], inplace=True)
```

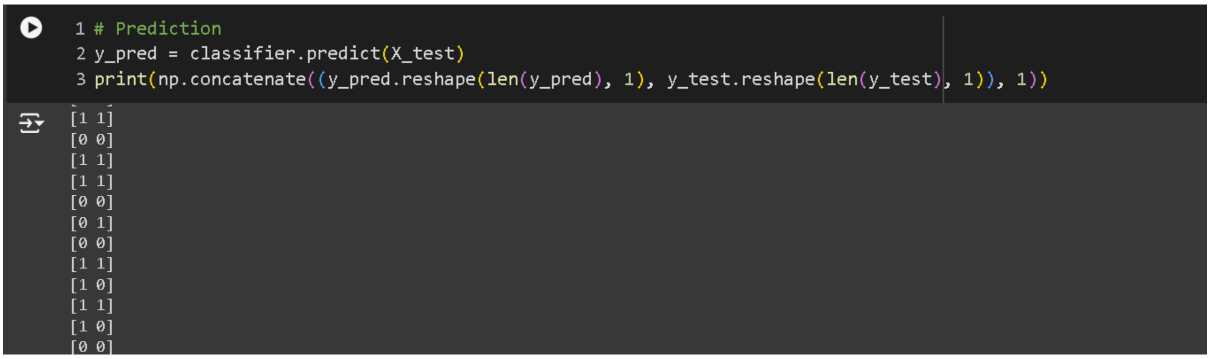
```python
# Split data into dependent/independent variables
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = True)
```

```python
# Scale dataset
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

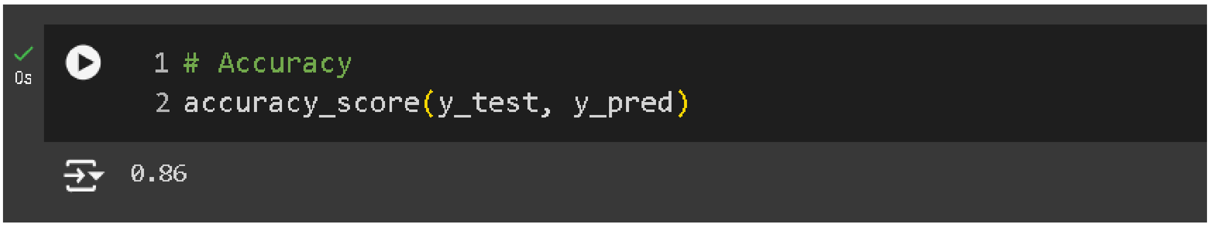
```python
# Classifier
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

```
GaussianNB
GaussianNB()
```



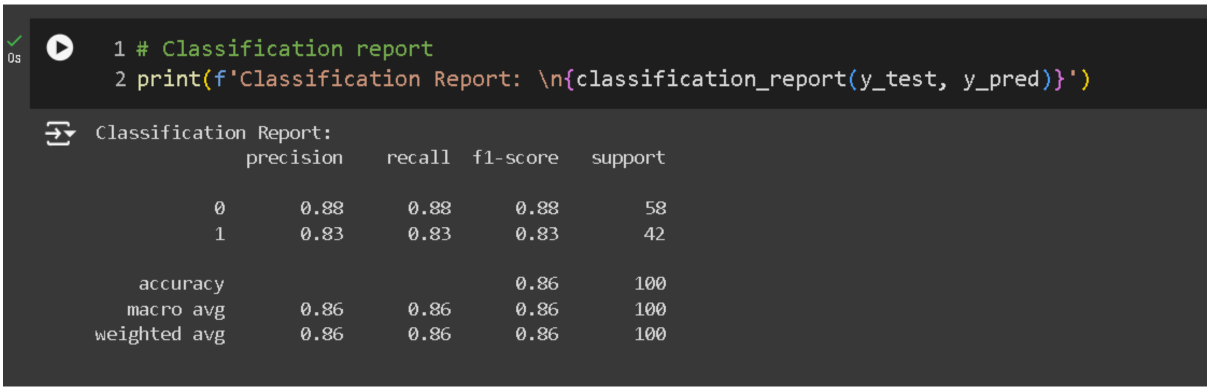
```python
# Prediction
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred), 1), y_test.reshape(len(y_test), 1)), 1))
```

```
[1 1]
[0 0]
[1 1]
[1 1]
[0 0]
[0 1]
[0 0]
[1 1]
[1 0]
[1 1]
[1 0]
[0 0]
```



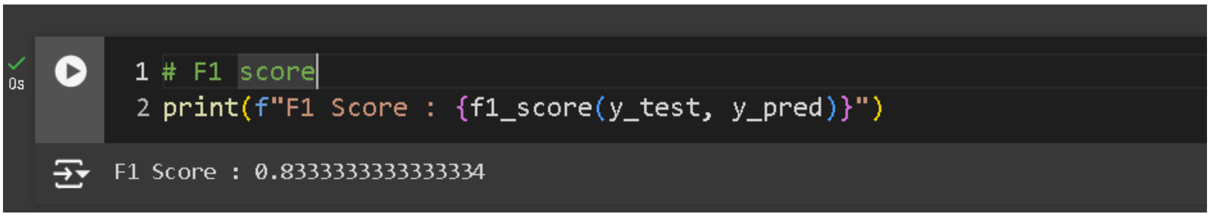
```python
# Accuracy
accuracy_score(y_test, y_pred)
```

```
0.86
```



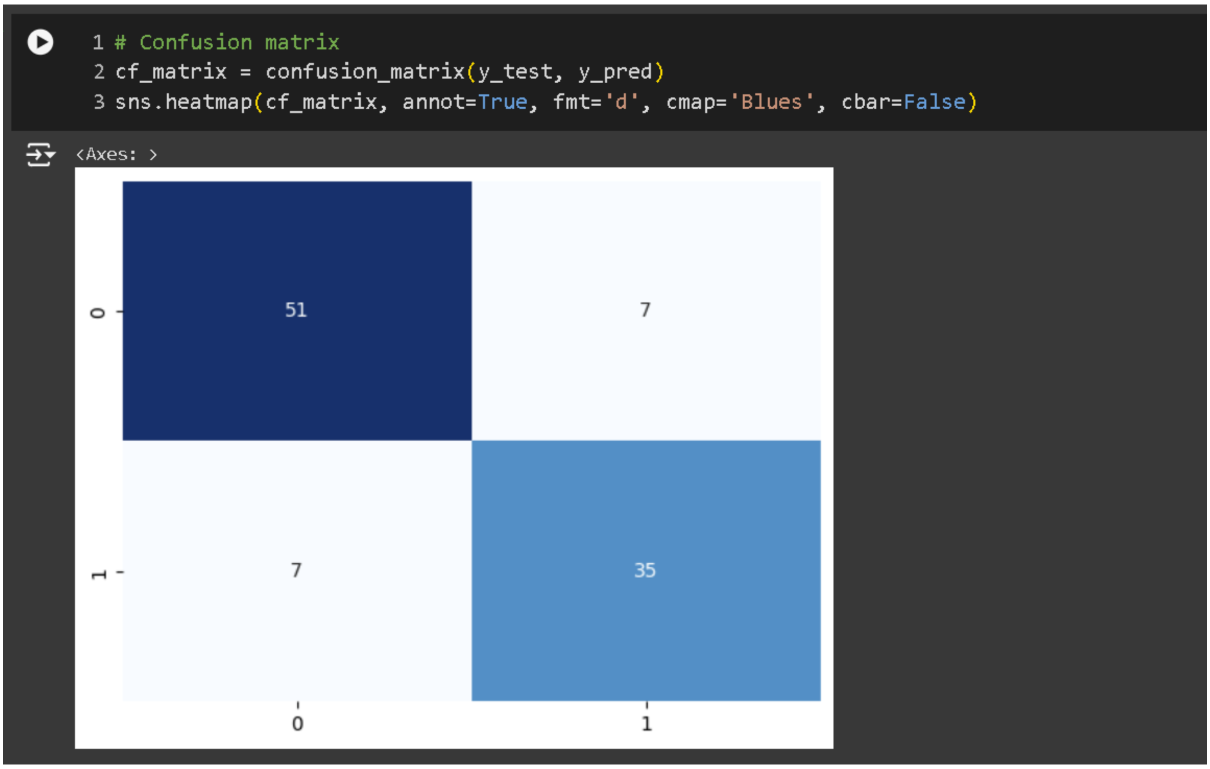
```python
# Classification report
print(f'Classification Report: \n{classification_report(y_test, y_pred)}')
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.88      0.88        58
           1       0.83      0.83      0.83        42

    accuracy                           0.86       100
   macro avg       0.86      0.86      0.86       100
weighted avg       0.86      0.86      0.86       100
```



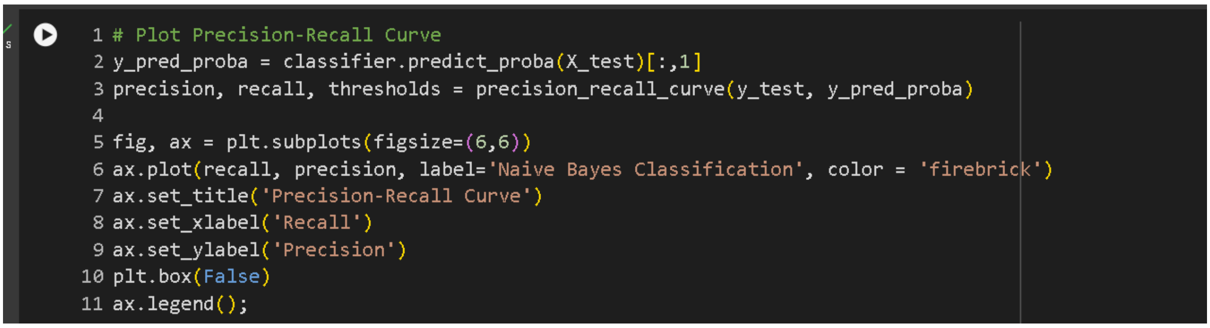
```
# F1 score
print(f"F1 Score : {f1_score(y_test, y_pred)}")
```

```
F1 Score : 0.8333333333333334
```



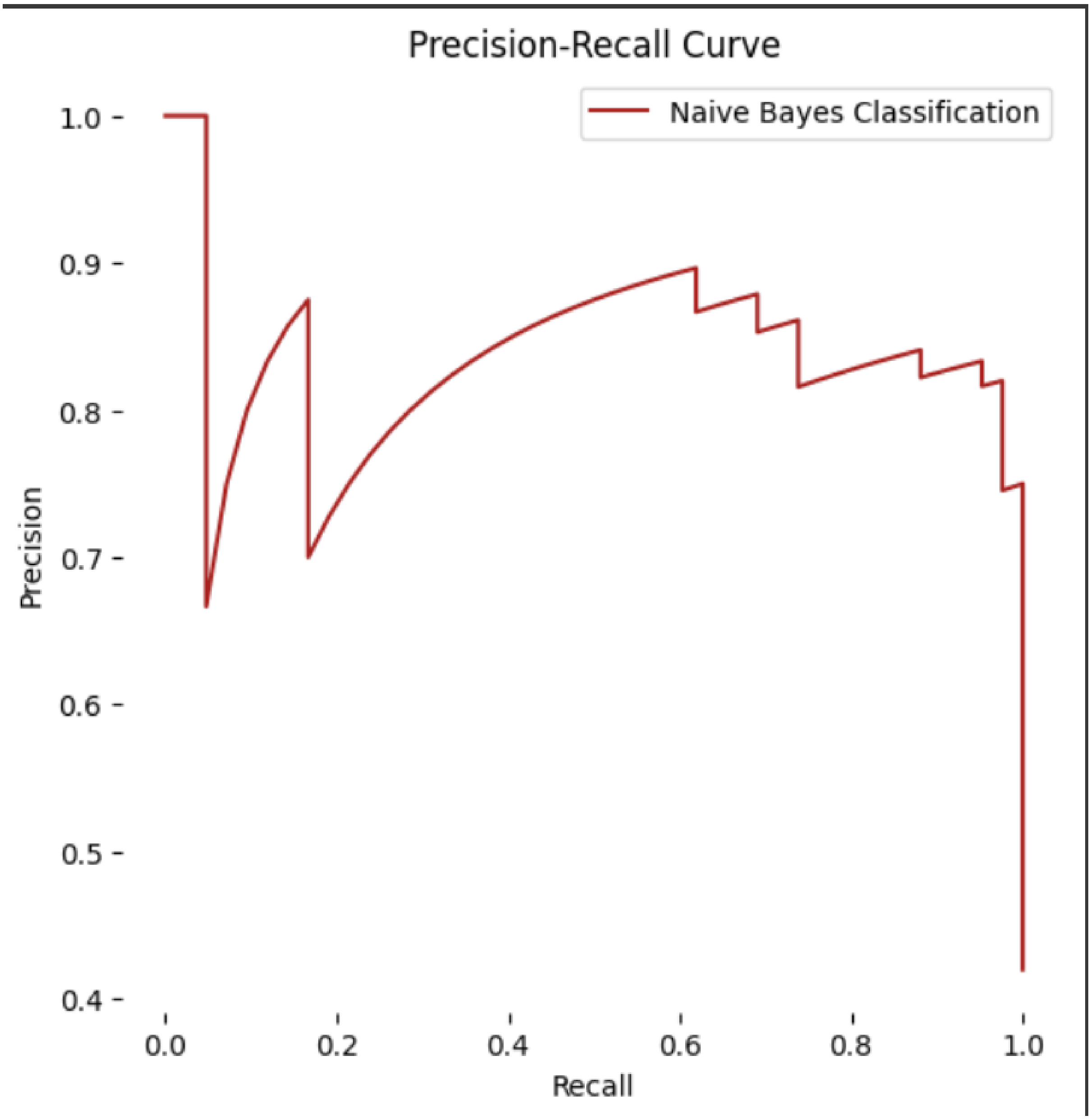
```
# Confusion matrix
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
```

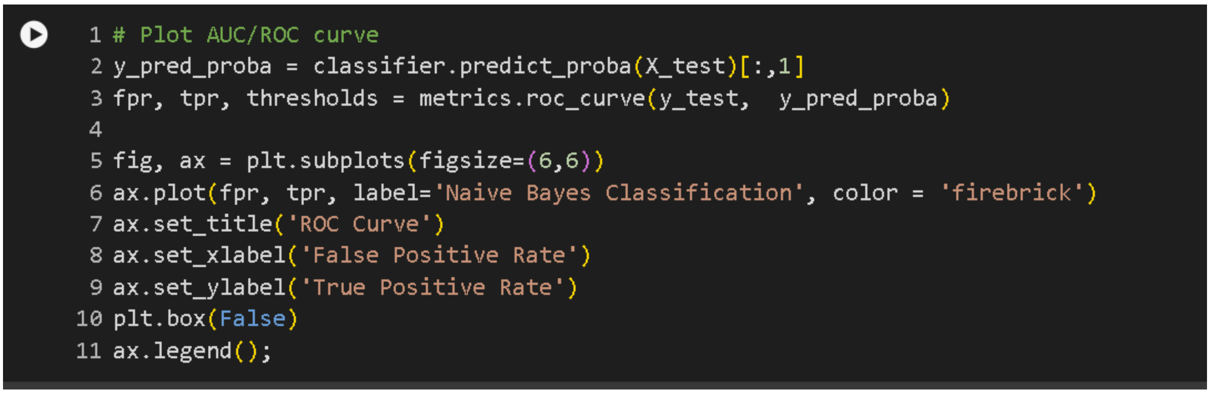
```
<Axes: >
```



```
# Plot Precision-Recall Curve
y_pred_proba = classifier.predict_proba(X_test)[:,1]
precision, recall, thresholds = precision_recall_curve(y_test, y_pred_proba)

fig, ax = plt.subplots(figsize=(6,6))
ax.plot(recall, precision, label='Naive Bayes Classification', color = 'firebrick')
ax.set_title('Precision-Recall Curve')
ax.set_xlabel('Recall')
ax.set_ylabel('Precision')
plt.box(False)
ax.legend();
```

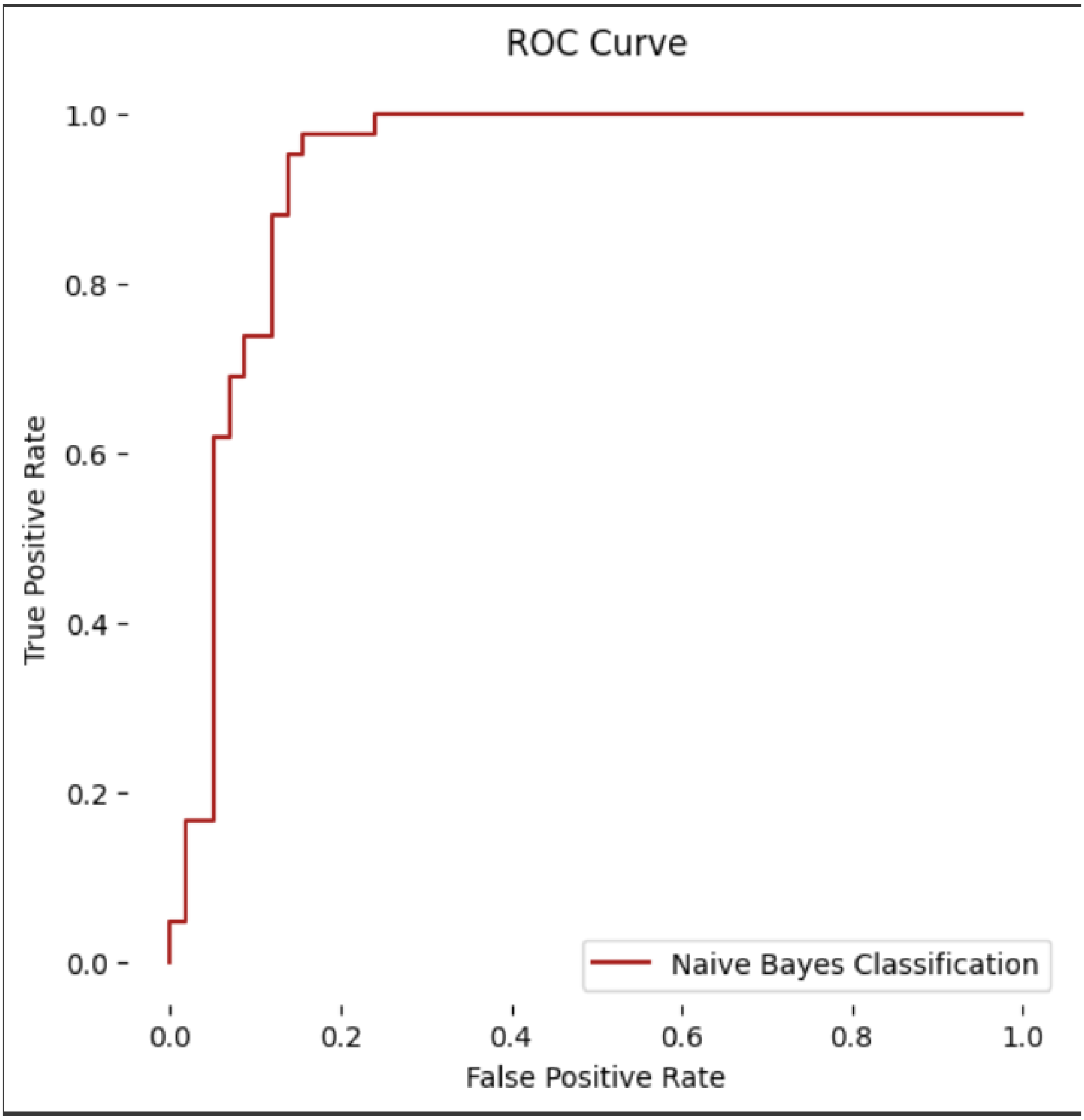
Output:



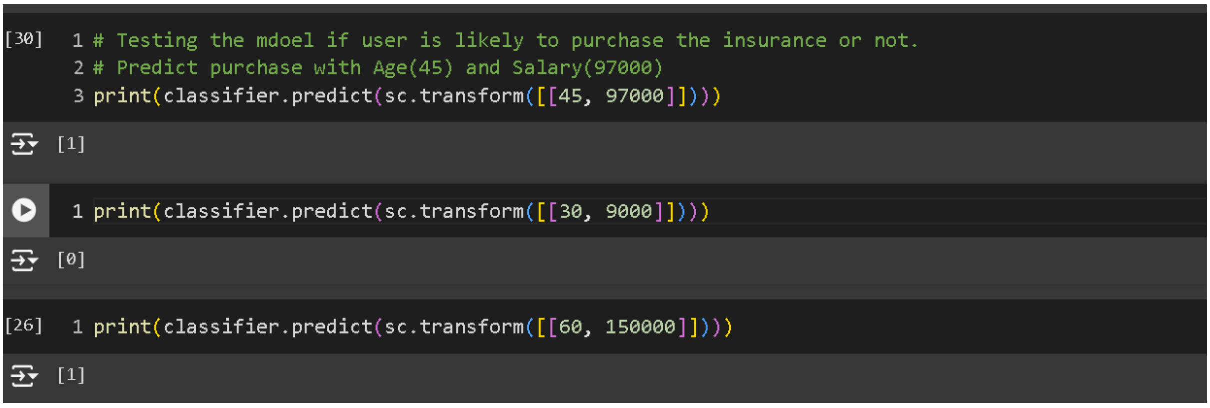


```
# Plot AUC/ROC curve
y_pred_proba = classifier.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = metrics.roc_curve(y_test,  y_pred_proba)

fig, ax = plt.subplots(figsize=(6,6))
ax.plot(fpr, tpr, label='Naive Bayes Classification', color = 'firebrick')
ax.set_title('ROC Curve')
ax.set_xlabel('False Positive Rate')
ax.set_ylabel('True Positive Rate')
plt.box(False)
ax.legend();
```

Output:



FINAL OUTPUT SHOWING THE MODEL PREDICTING BASED ON AGE AND SALARY



```
# Testing the mdoel if user is likely to purchase the insurance or not.
# Predict purchase with Age(45) and Salary(97000)
print(classifier.predict(sc.transform([[45, 97000]])))
```

```
[1]
```

```
print(classifier.predict(sc.transform([[30, 9000]])))
```

```
[0]
```

```
print(classifier.predict(sc.transform([[60, 150000]])))
```

```
[1]
```