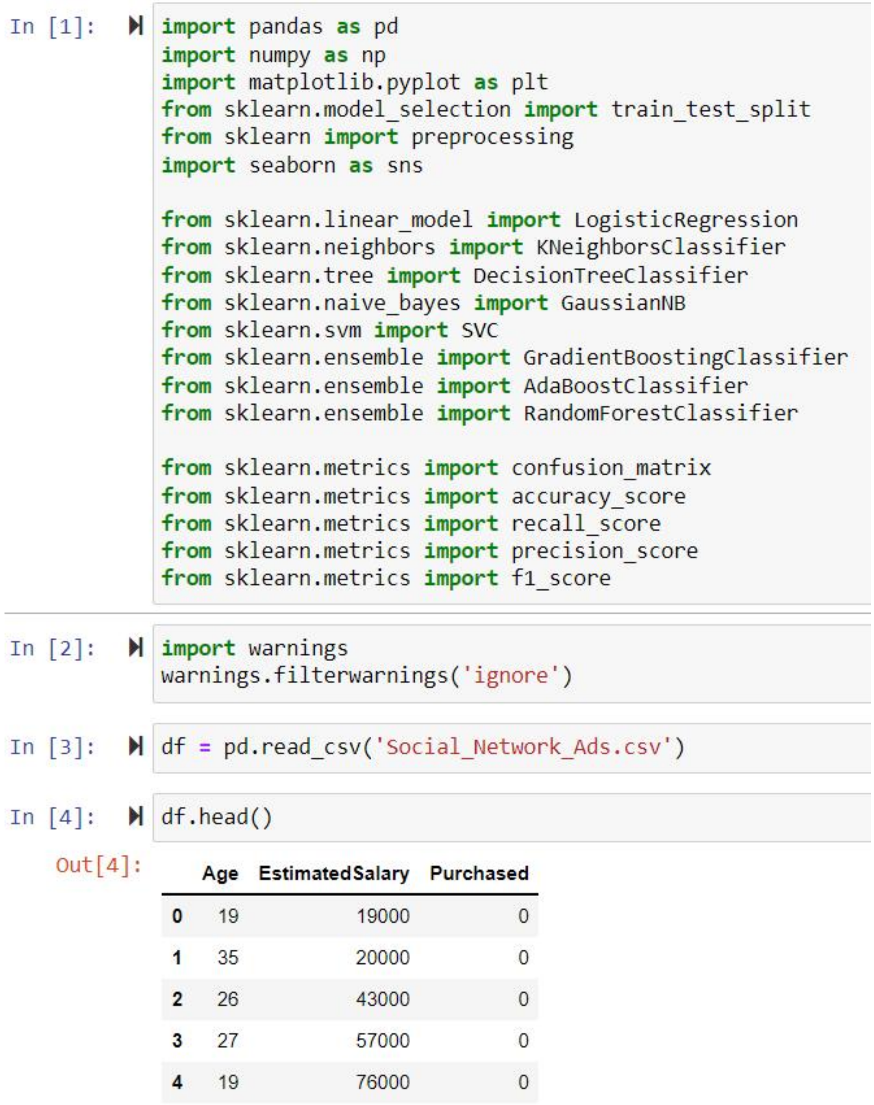
Pooja Agarwal
1905330

Lab 9

Q1: From dataset ‘social ad’:
Calculate performance metric
1. Accuracy
2. Sensitivity/Recall
3. Specificity
4. F-score
5. Precision
Use Algorithmk-NN
1. Logistic regression
2. SVM
3. Decision Tree
4. Naive Bayes

Plot a bar graph and compare the accuracy obtained in each case.

CODE:



In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import seaborn as sns

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

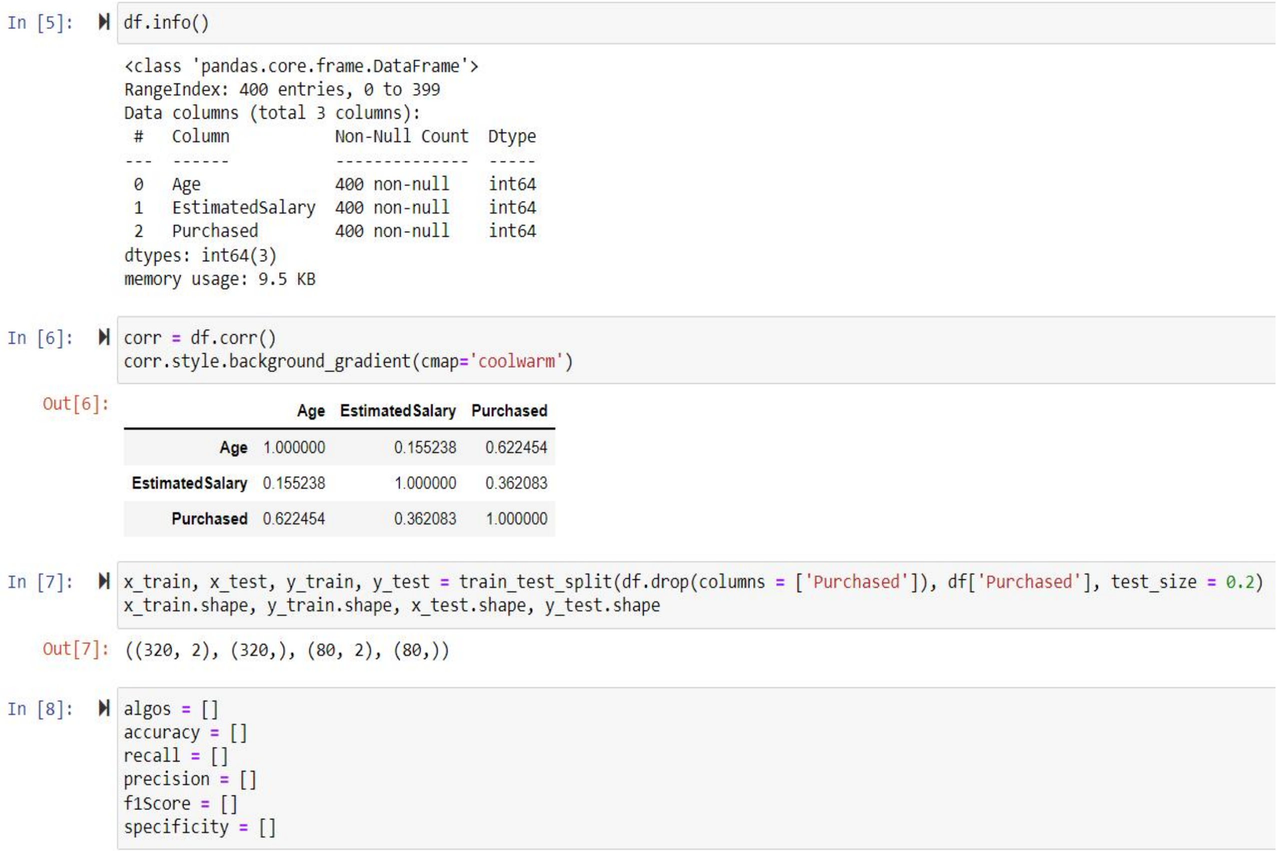
In [3]:
```
df = pd.read_csv('Social_Network_Ads.csv')
```

In [4]:
```
df.head()
```

Out[4]:

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

Pooja Agarwal
1905330



```
In [5]:  df.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 400 entries, 0 to 399
         Data columns (total 3 columns):
          #   Column           Non-Null Count  Dtype
         ---  ------           --------------  -----
          0   Age              400 non-null    int64
          1   EstimatedSalary  400 non-null    int64
          2   Purchased        400 non-null    int64
         dtypes: int64(3)
         memory usage: 9.5 KB
```

```
In [6]:  corr = df.corr()
         corr.style.background_gradient(cmap='coolwarm')
```

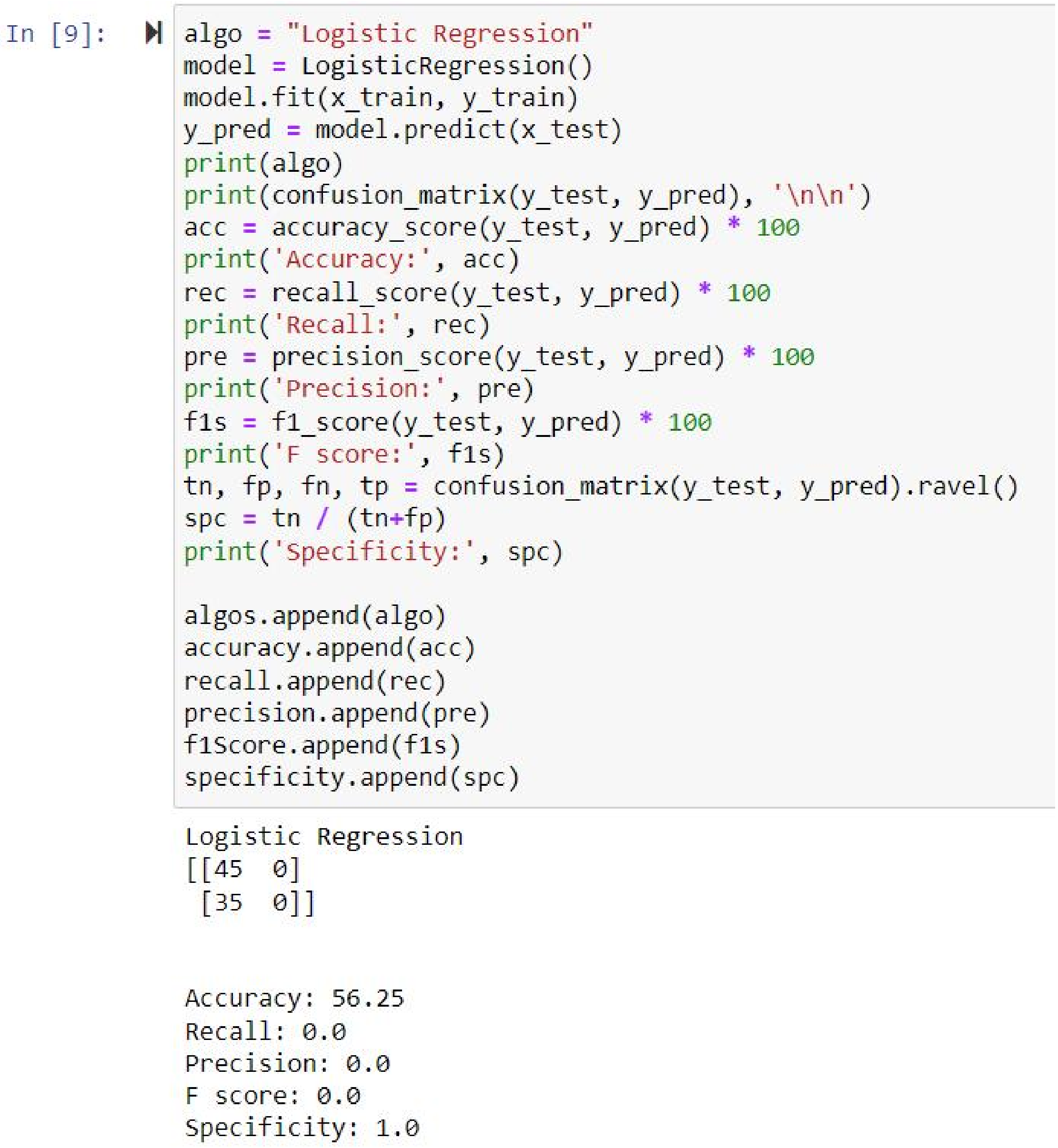
Out[6]:

	Age	EstimatedSalary	Purchased
Age	1.000000	0.155238	0.622454
EstimatedSalary	0.155238	1.000000	0.362083
Purchased	0.622454	0.362083	1.000000

```
In [7]:  x_train, x_test, y_train, y_test = train_test_split(df.drop(columns = ['Purchased']), df['Purchased'], test_size = 0.2)
         x_train.shape, y_train.shape, x_test.shape, y_test.shape

Out[7]:  ((320, 2), (320,), (80, 2), (80,))
```

```
In [8]:  algos = []
         accuracy = []
         recall = []
         precision = []
         f1Score = []
         specificity = []
```



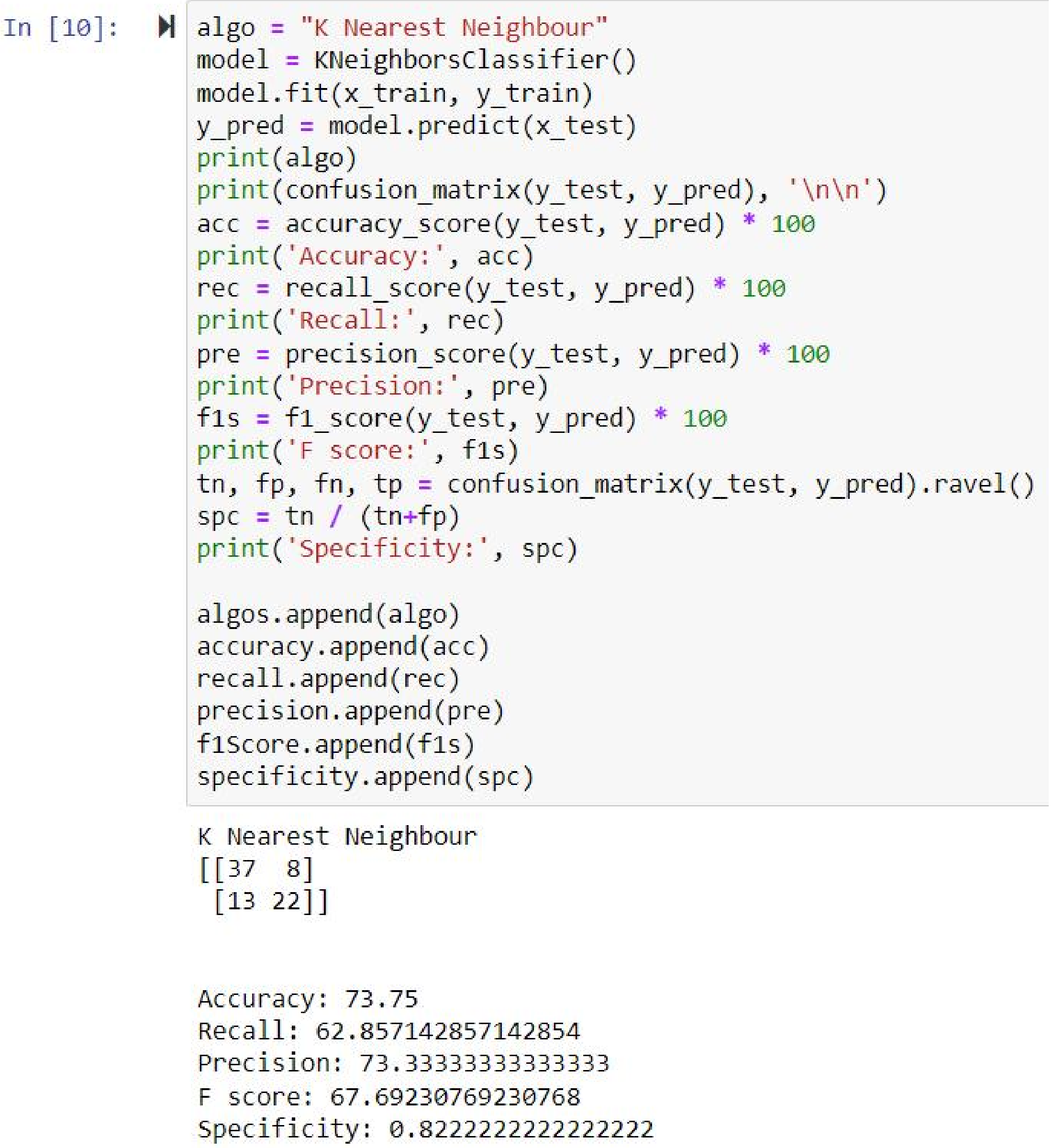
In [9]:

```
algo = "Logistic Regression"
model = LogisticRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print(algo)
print(confusion_matrix(y_test, y_pred), '\n\n')
acc = accuracy_score(y_test, y_pred) * 100
print('Accuracy:', acc)
rec = recall_score(y_test, y_pred) * 100
print('Recall:', rec)
pre = precision_score(y_test, y_pred) * 100
print('Precision:', pre)
f1s = f1_score(y_test, y_pred) * 100
print('F score:', f1s)
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
spc = tn / (tn+fp)
print('Specificity:', spc)

algos.append(algo)
accuracy.append(acc)
recall.append(rec)
precision.append(pre)
f1Score.append(f1s)
specificity.append(spc)
```

```
Logistic Regression
[[45  0]
 [35  0]]


Accuracy: 56.25
Recall: 0.0
Precision: 0.0
F score: 0.0
Specificity: 1.0
```



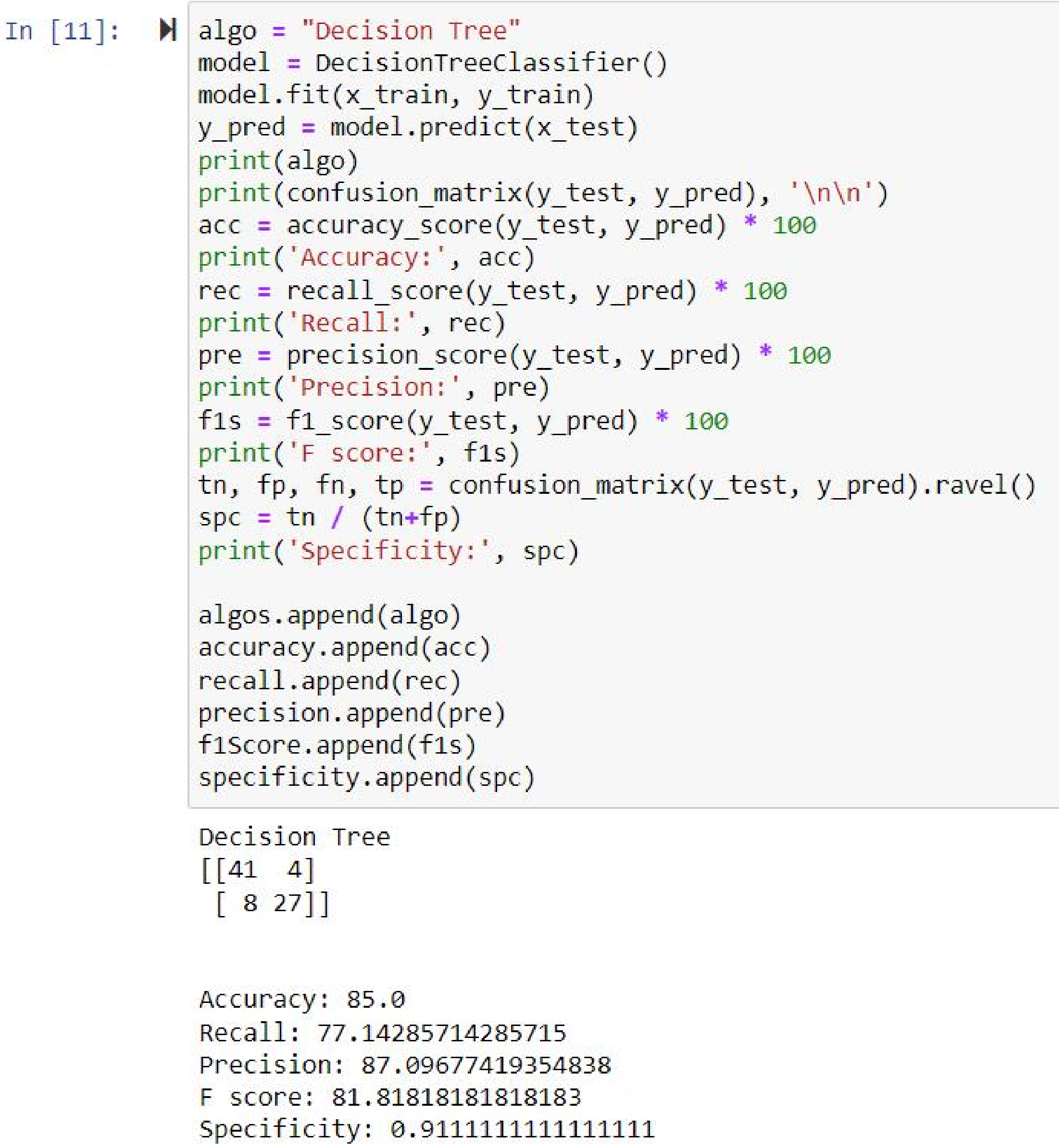
In [10]:

```
algo = "K Nearest Neighbour"
model = KNeighborsClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print(algo)
print(confusion_matrix(y_test, y_pred), '\n\n')
acc = accuracy_score(y_test, y_pred) * 100
print('Accuracy:', acc)
rec = recall_score(y_test, y_pred) * 100
print('Recall:', rec)
pre = precision_score(y_test, y_pred) * 100
print('Precision:', pre)
f1s = f1_score(y_test, y_pred) * 100
print('F score:', f1s)
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
spc = tn / (tn+fp)
print('Specificity:', spc)

algos.append(algo)
accuracy.append(acc)
recall.append(rec)
precision.append(pre)
f1Score.append(f1s)
specificity.append(spc)
```

```
K Nearest Neighbour
[[37  8]
 [13 22]]


Accuracy: 73.75
Recall: 62.857142857142854
Precision: 73.33333333333333
F score: 67.69230769230768
Specificity: 0.8222222222222222
```

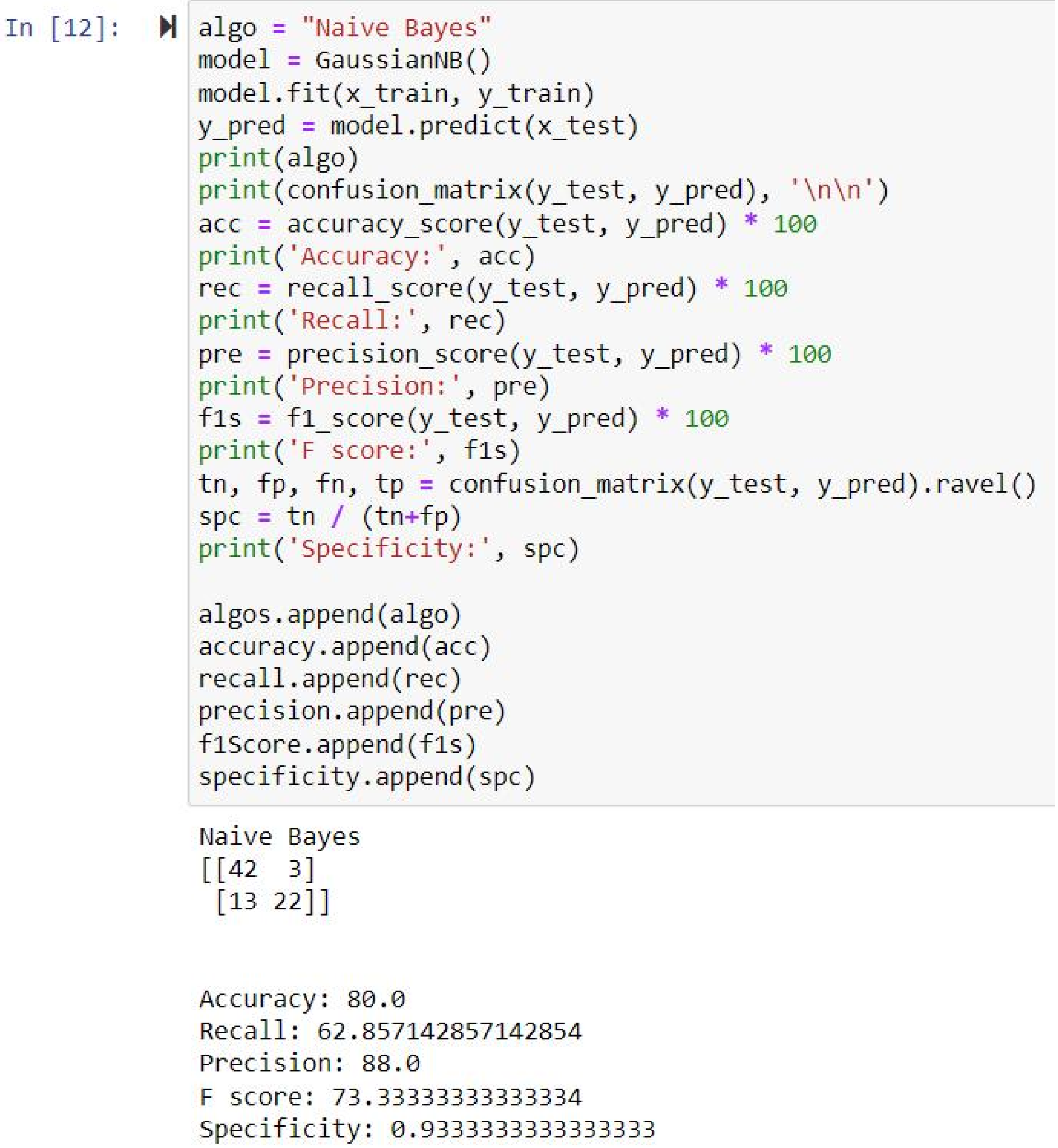


```
In [11]:  algo = "Decision Tree"
          model = DecisionTreeClassifier()
          model.fit(x_train, y_train)
          y_pred = model.predict(x_test)
          print(algo)
          print(confusion_matrix(y_test, y_pred), '\n\n')
          acc = accuracy_score(y_test, y_pred) * 100
          print('Accuracy:', acc)
          rec = recall_score(y_test, y_pred) * 100
          print('Recall:', rec)
          pre = precision_score(y_test, y_pred) * 100
          print('Precision:', pre)
          f1s = f1_score(y_test, y_pred) * 100
          print('F score:', f1s)
          tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
          spc = tn / (tn+fp)
          print('Specificity:', spc)

          algos.append(algo)
          accuracy.append(acc)
          recall.append(rec)
          precision.append(pre)
          f1Score.append(f1s)
          specificity.append(spc)
```

```
Decision Tree
[[41  4]
 [ 8 27]]


Accuracy: 85.0
Recall: 77.14285714285715
Precision: 87.09677419354838
F score: 81.81818181818183
Specificity: 0.9111111111111111
```



In [12]:

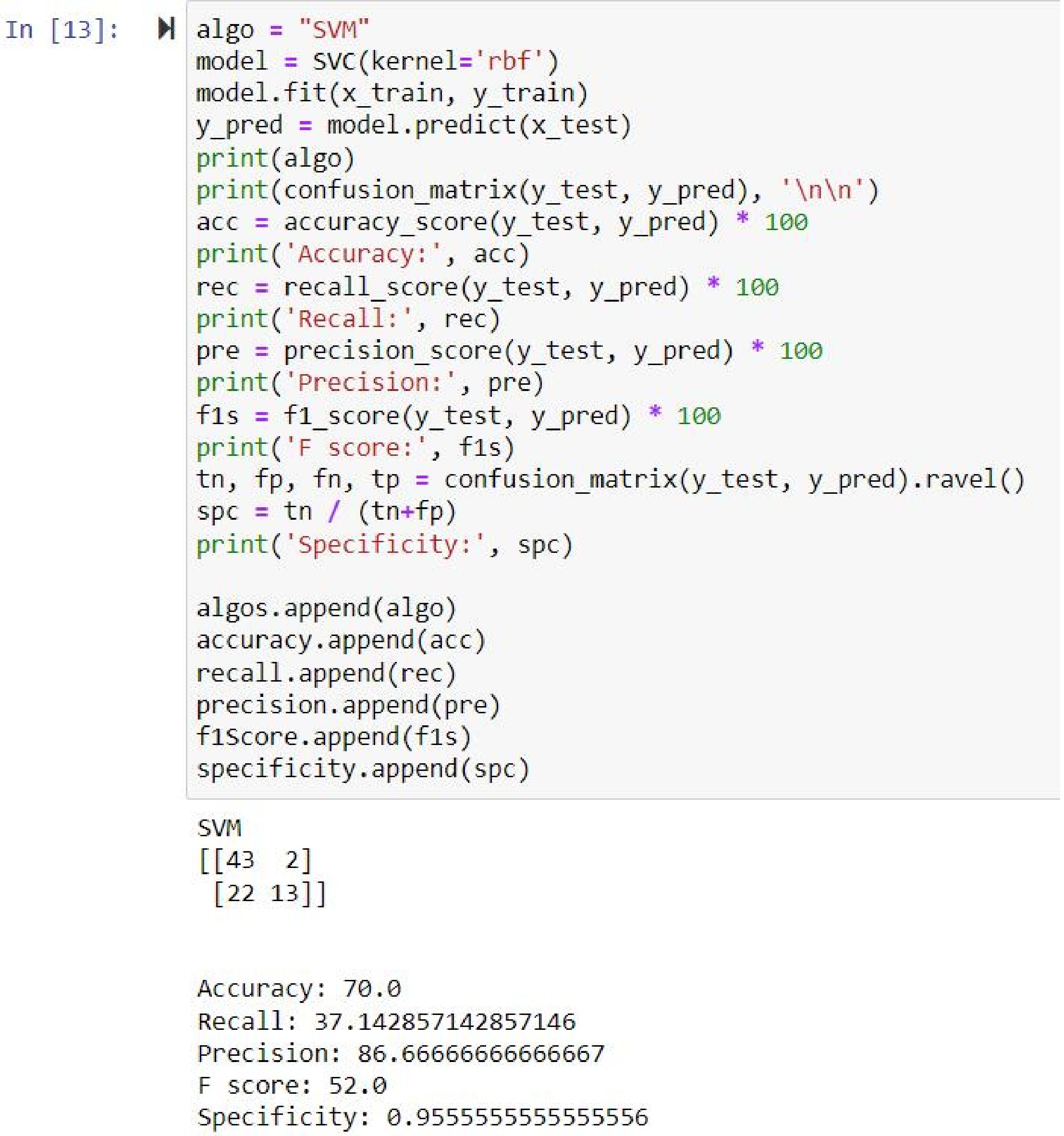
```
algo = "Naive Bayes"
model = GaussianNB()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print(algo)
print(confusion_matrix(y_test, y_pred), '\n\n')
acc = accuracy_score(y_test, y_pred) * 100
print('Accuracy:', acc)
rec = recall_score(y_test, y_pred) * 100
print('Recall:', rec)
pre = precision_score(y_test, y_pred) * 100
print('Precision:', pre)
f1s = f1_score(y_test, y_pred) * 100
print('F score:', f1s)
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
spc = tn / (tn+fp)
print('Specificity:', spc)

algos.append(algo)
accuracy.append(acc)
recall.append(rec)
precision.append(pre)
f1Score.append(f1s)
specificity.append(spc)
```

```
Naive Bayes
[[42  3]
 [13 22]]


Accuracy: 80.0
Recall: 62.857142857142854
Precision: 88.0
F score: 73.33333333333334
Specificity: 0.9333333333333333
```

Pooja Agarwal
1905330



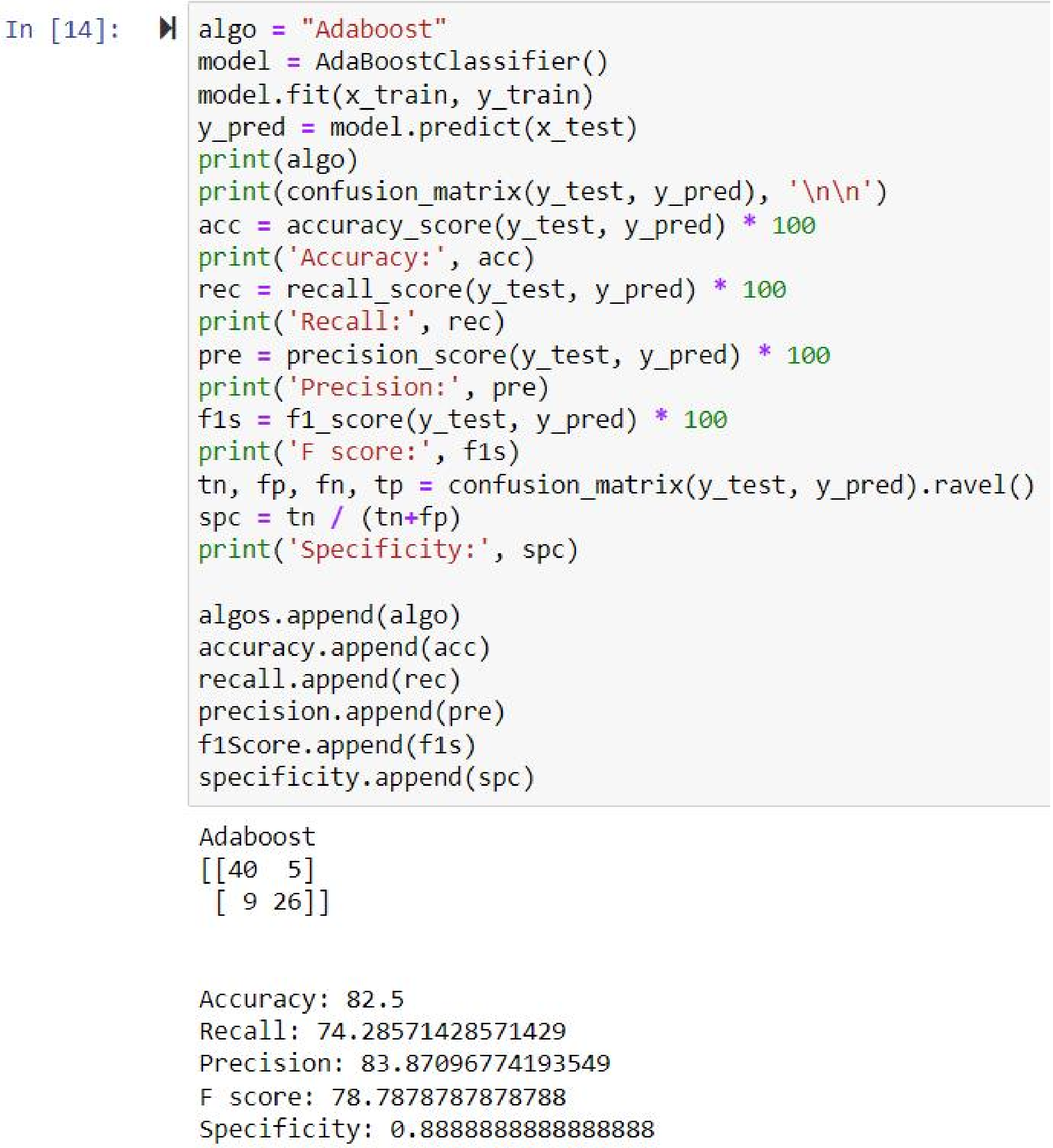
In [13]:

```
algo = "SVM"
model = SVC(kernel='rbf')
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print(algo)
print(confusion_matrix(y_test, y_pred), '\n\n')
acc = accuracy_score(y_test, y_pred) * 100
print('Accuracy:', acc)
rec = recall_score(y_test, y_pred) * 100
print('Recall:', rec)
pre = precision_score(y_test, y_pred) * 100
print('Precision:', pre)
f1s = f1_score(y_test, y_pred) * 100
print('F score:', f1s)
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
spc = tn / (tn+fp)
print('Specificity:', spc)

algos.append(algo)
accuracy.append(acc)
recall.append(rec)
precision.append(pre)
f1Score.append(f1s)
specificity.append(spc)
```

```
SVM
[[43  2]
 [22 13]]


Accuracy: 70.0
Recall: 37.142857142857146
Precision: 86.66666666666667
F score: 52.0
Specificity: 0.9555555555555556
```

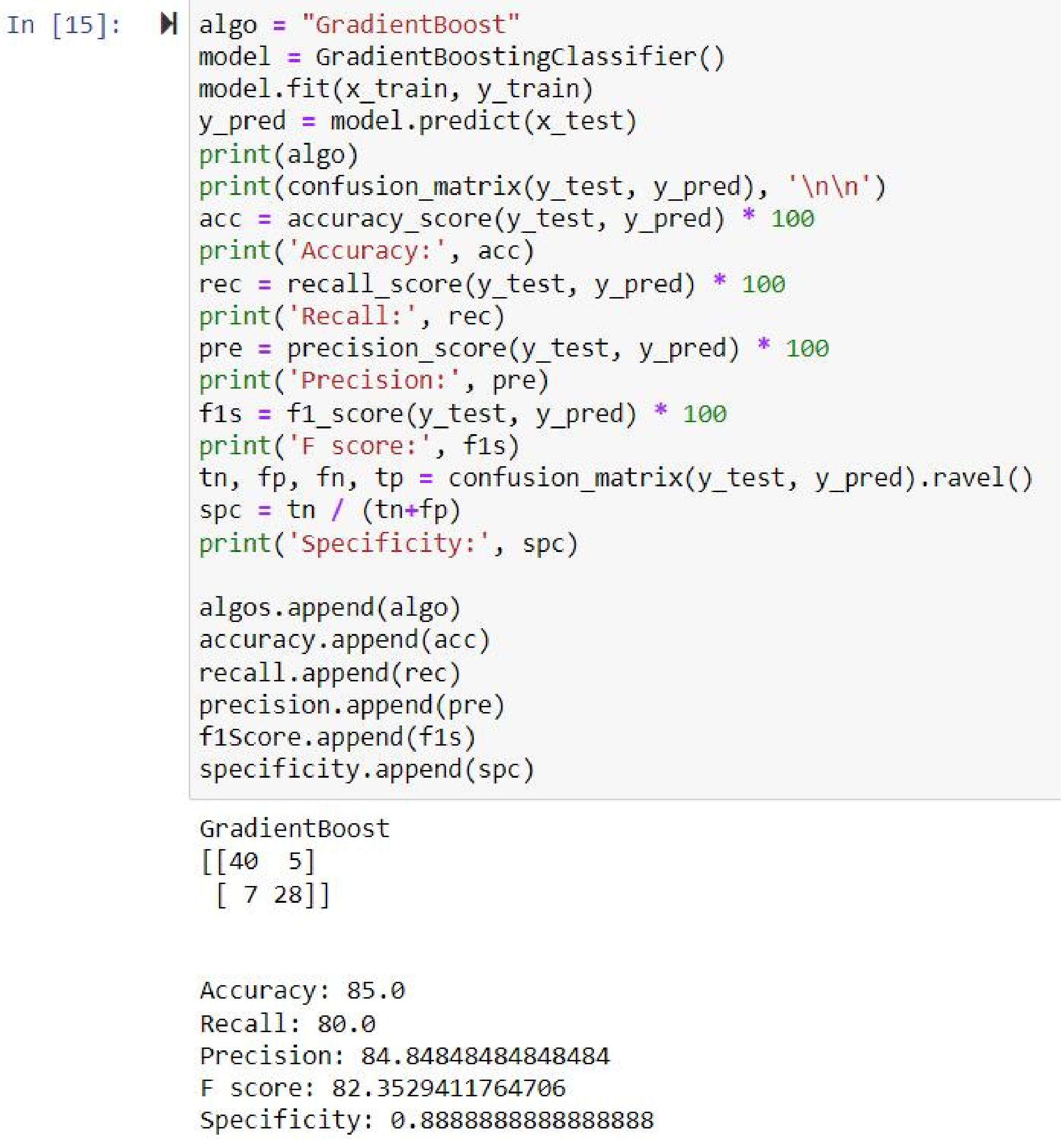


```
In [14]:
algo = "Adaboost"
model = AdaBoostClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print(algo)
print(confusion_matrix(y_test, y_pred), '\n\n')
acc = accuracy_score(y_test, y_pred) * 100
print('Accuracy:', acc)
rec = recall_score(y_test, y_pred) * 100
print('Recall:', rec)
pre = precision_score(y_test, y_pred) * 100
print('Precision:', pre)
f1s = f1_score(y_test, y_pred) * 100
print('F score:', f1s)
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
spc = tn / (tn+fp)
print('Specificity:', spc)

algos.append(algo)
accuracy.append(acc)
recall.append(rec)
precision.append(pre)
f1Score.append(f1s)
specificity.append(spc)
```

```
Adaboost
[[40  5]
 [ 9 26]]


Accuracy: 82.5
Recall: 74.28571428571429
Precision: 83.87096774193549
F score: 78.78787878787878
Specificity: 0.8888888888888888
```



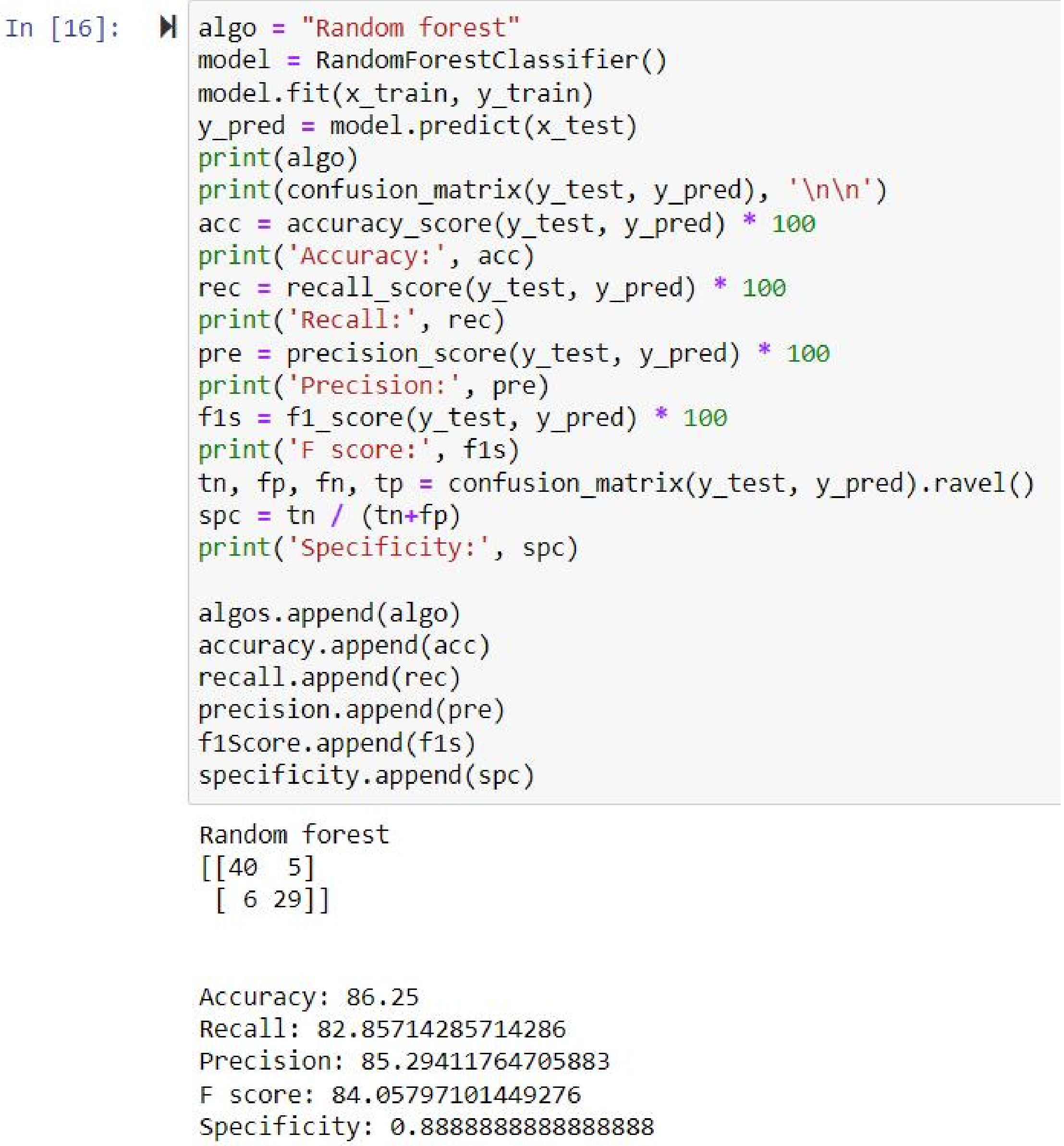
In [15]:

```
algo = "GradientBoost"
model = GradientBoostingClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print(algo)
print(confusion_matrix(y_test, y_pred), '\n\n')
acc = accuracy_score(y_test, y_pred) * 100
print('Accuracy:', acc)
rec = recall_score(y_test, y_pred) * 100
print('Recall:', rec)
pre = precision_score(y_test, y_pred) * 100
print('Precision:', pre)
f1s = f1_score(y_test, y_pred) * 100
print('F score:', f1s)
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
spc = tn / (tn+fp)
print('Specificity:', spc)

algos.append(algo)
accuracy.append(acc)
recall.append(rec)
precision.append(pre)
f1Score.append(f1s)
specificity.append(spc)
```

```
GradientBoost
[[40  5]
 [ 7 28]]


Accuracy: 85.0
Recall: 80.0
Precision: 84.84848484848484
F score: 82.3529411764706
Specificity: 0.8888888888888888
```



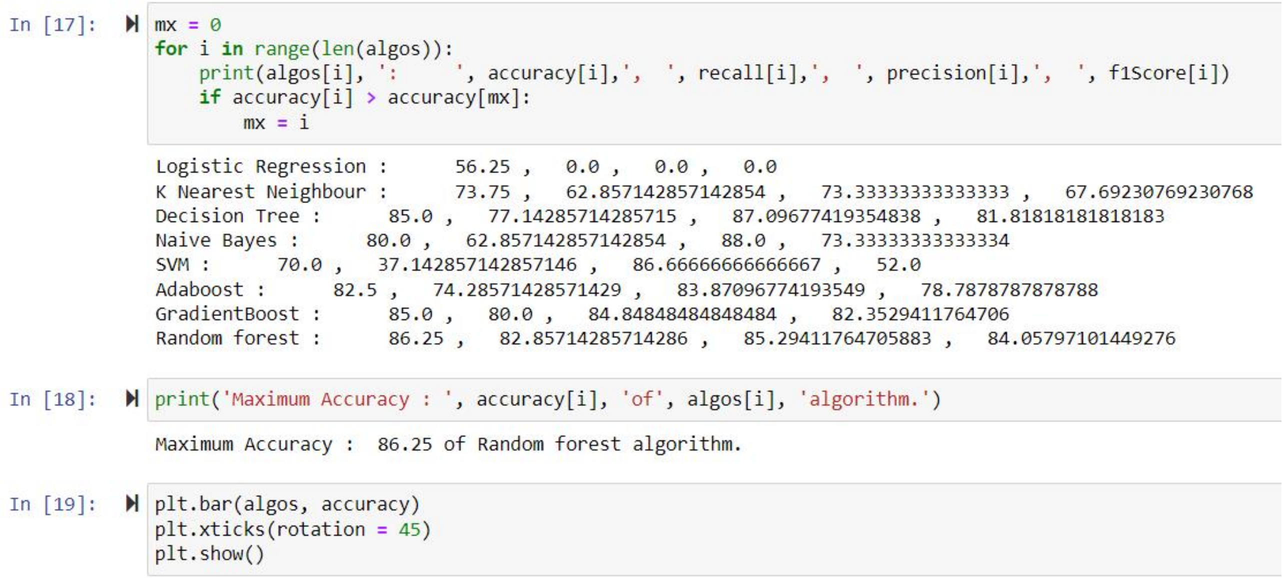
In [16]:

```
algo = "Random forest"
model = RandomForestClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print(algo)
print(confusion_matrix(y_test, y_pred), '\n\n')
acc = accuracy_score(y_test, y_pred) * 100
print('Accuracy:', acc)
rec = recall_score(y_test, y_pred) * 100
print('Recall:', rec)
pre = precision_score(y_test, y_pred) * 100
print('Precision:', pre)
f1s = f1_score(y_test, y_pred) * 100
print('F score:', f1s)
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
spc = tn / (tn+fp)
print('Specificity:', spc)

algos.append(algo)
accuracy.append(acc)
recall.append(rec)
precision.append(pre)
f1Score.append(f1s)
specificity.append(spc)
```

```
Random forest
[[40  5]
 [ 6 29]]


Accuracy: 86.25
Recall: 82.85714285714286
Precision: 85.29411764705883
F score: 84.05797101449276
Specificity: 0.8888888888888888
```



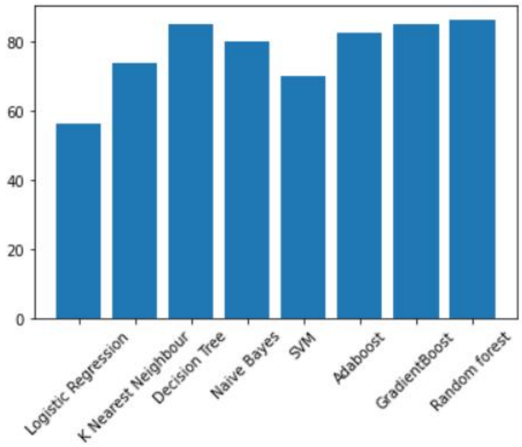
```
In [17]:  mx = 0
          for i in range(len(algos)):
              print(algos[i], ':      ', accuracy[i],',   ', recall[i],',   ', precision[i],',   ', f1Score[i])
              if accuracy[i] > accuracy[mx]:
                  mx = i
```

```
Logistic Regression :      56.25 ,    0.0 ,    0.0 ,    0.0
K Nearest Neighbour :      73.75 ,    62.857142857142854 ,    73.33333333333333 ,    67.69230769230768
Decision Tree :      85.0 ,    77.14285714285715 ,    87.09677419354838 ,    81.81818181818183
Naive Bayes :      80.0 ,    62.857142857142854 ,    88.0 ,    73.33333333333334
SVM :      70.0 ,    37.142857142857146 ,    86.66666666666667 ,    52.0
Adaboost :      82.5 ,    74.28571428571429 ,    83.87096774193549 ,    78.7878787878788
GradientBoost :      85.0 ,    80.0 ,    84.84848484848484 ,    82.3529411764706
Random forest :      86.25 ,    82.85714285714286 ,    85.29411764705883 ,    84.05797101449276
```

```
In [18]:  print('Maximum Accuracy : ', accuracy[i], 'of', algos[i], 'algorithm.')
```

```
Maximum Accuracy :  86.25 of Random forest algorithm.
```

```
In [19]:  plt.bar(algos, accuracy)
          plt.xticks(rotation = 45)
          plt.show()
```


Logistic Regression
K Nearest Neighbour
Decision Tree
Naive Bayes
SVM
Adaboost
GradientBoost
Random forest