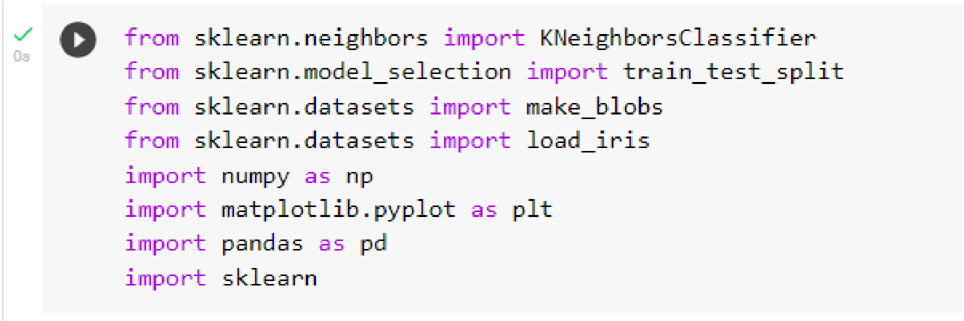
INTRODUCTION TO MACHINE LEARNING

LAB ASSIGNMENT – 7

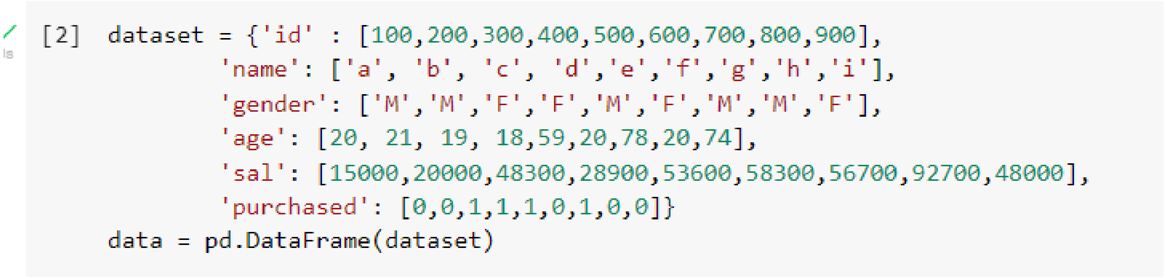
NAME : PRATHAPANI SATWIKA

REG.NO.: 20BCD7160

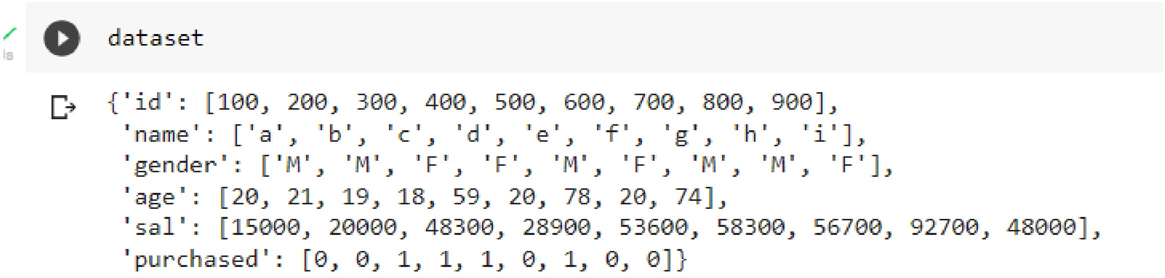
1.Create an array of x and y along with classes of your own size along with class as 0 and 1.



```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_blobs
from sklearn.datasets import load_iris
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn
```

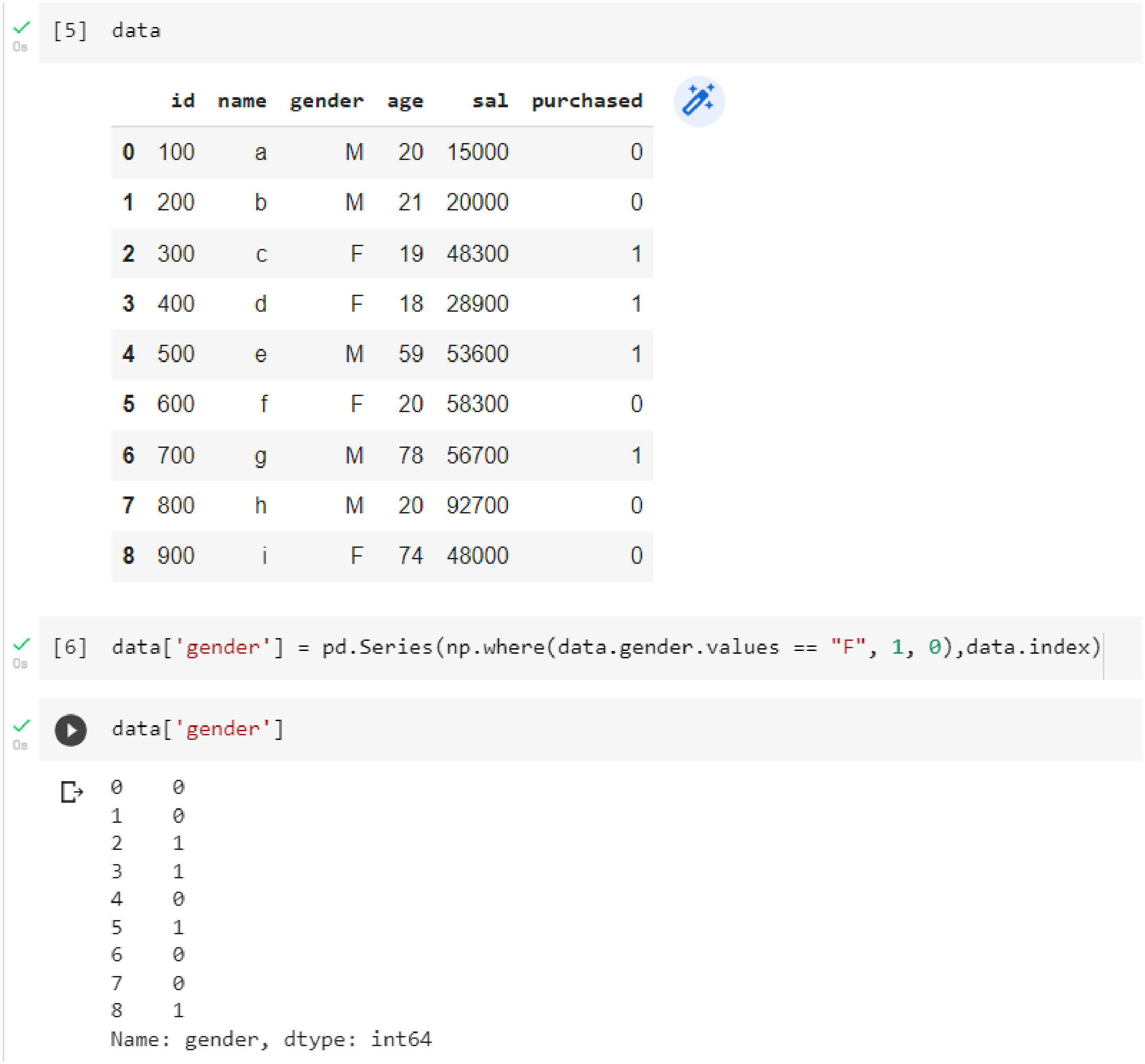


```
[2] dataset = {'id' : [100,200,300,400,500,600,700,800,900],
              'name': ['a', 'b', 'c', 'd','e','f','g','h','i'],
              'gender': ['M','M','F','F','M','F','M','M','F'],
              'age': [20, 21, 19, 18,59,20,78,20,74],
              'sal': [15000,20000,48300,28900,53600,58300,56700,92700,48000],
              'purchased': [0,0,1,1,1,0,1,0,0]}
    data = pd.DataFrame(dataset)
```



```
dataset
```

```
{'id': [100, 200, 300, 400, 500, 600, 700, 800, 900],
 'name': ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i'],
 'gender': ['M', 'M', 'F', 'F', 'M', 'F', 'M', 'M', 'F'],
 'age': [20, 21, 19, 18, 59, 20, 78, 20, 74],
 'sal': [15000, 20000, 48300, 28900, 53600, 58300, 56700, 92700, 48000],
 'purchased': [0, 0, 1, 1, 1, 0, 1, 0, 0]}
```



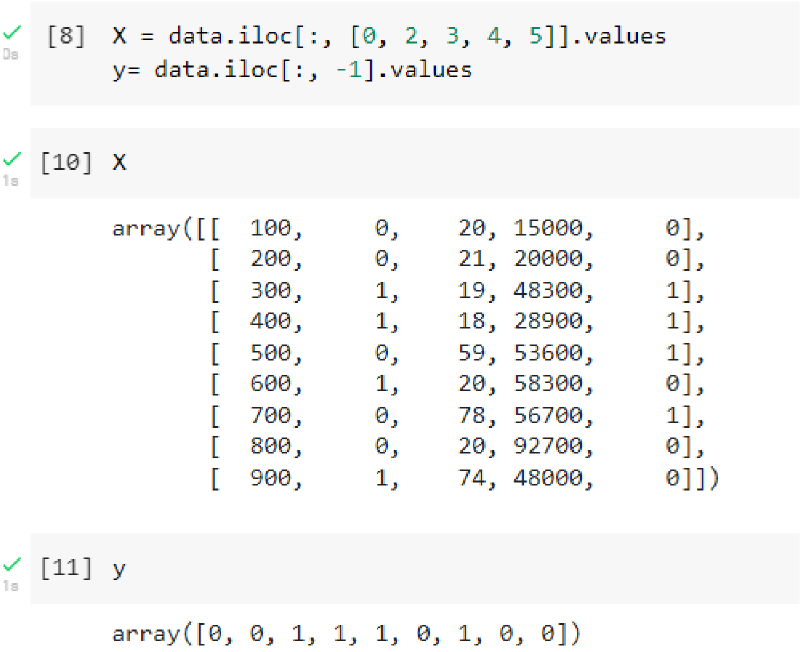
```
data
```

	id	name	gender	age	sal	purchased
0	100	a	M	20	15000	0
1	200	b	M	21	20000	0
2	300	c	F	19	48300	1
3	400	d	F	18	28900	1
4	500	e	M	59	53600	1
5	600	f	F	20	58300	0
6	700	g	M	78	56700	1
7	800	h	M	20	92700	0
8	900	i	F	74	48000	0

```
data['gender'] = pd.Series(np.where(data.gender.values == "F", 1, 0),data.index)
```

```
data['gender']
```

```
0    0
1    0
2    1
3    1
4    0
5    1
6    0
7    0
8    1
Name: gender, dtype: int64
```



```
[8]  X = data.iloc[:, [0, 2, 3, 4, 5]].values
     y= data.iloc[:, -1].values
```

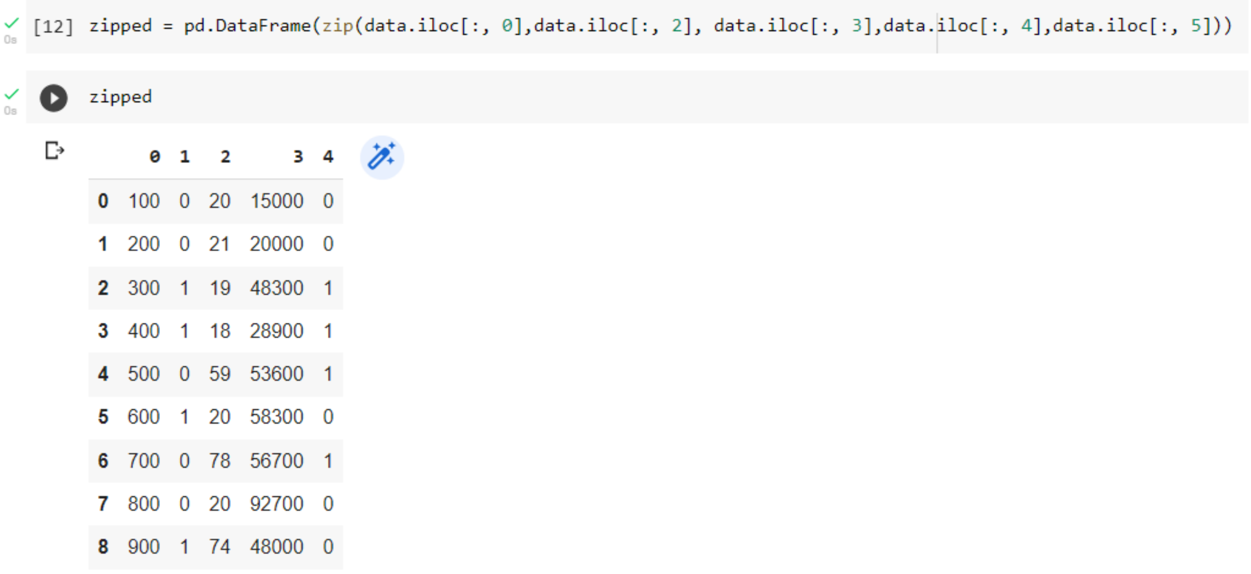
```
[10] X

     array([[  100,      0,     20, 15000,      0],
            [  200,      0,     21, 20000,      0],
            [  300,      1,     19, 48300,      1],
            [  400,      1,     18, 28900,      1],
            [  500,      0,     59, 53600,      1],
            [  600,      1,     20, 58300,      0],
            [  700,      0,     78, 56700,      1],
            [  800,      0,     20, 92700,      0],
            [  900,      1,     74, 48000,      0]])
```

```
[11] y

     array([0, 0, 1, 1, 1, 0, 1, 0, 0])
```

2.Turn the input features into a set of points with the zip command

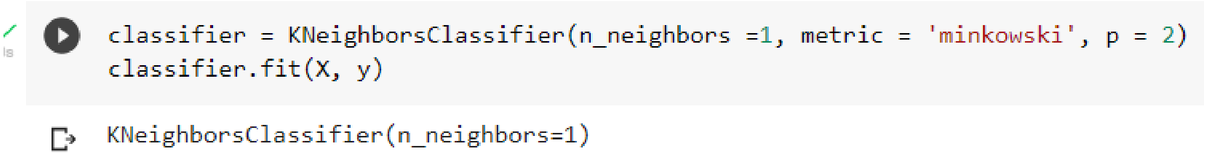


```
[12] zipped = pd.DataFrame(zip(data.iloc[:, 0],data.iloc[:, 2], data.iloc[:, 3],data.iloc[:, 4],data.iloc[:, 5]))
```

```
zipped
```

	0	1	2	3	4
0	100	0	20	15000	0
1	200	0	21	20000	0
2	300	1	19	48300	1
3	400	1	18	28900	1
4	500	0	59	53600	1
5	600	1	20	58300	0
6	700	0	78	56700	1
7	800	0	20	92700	0
8	900	1	74	48000	0

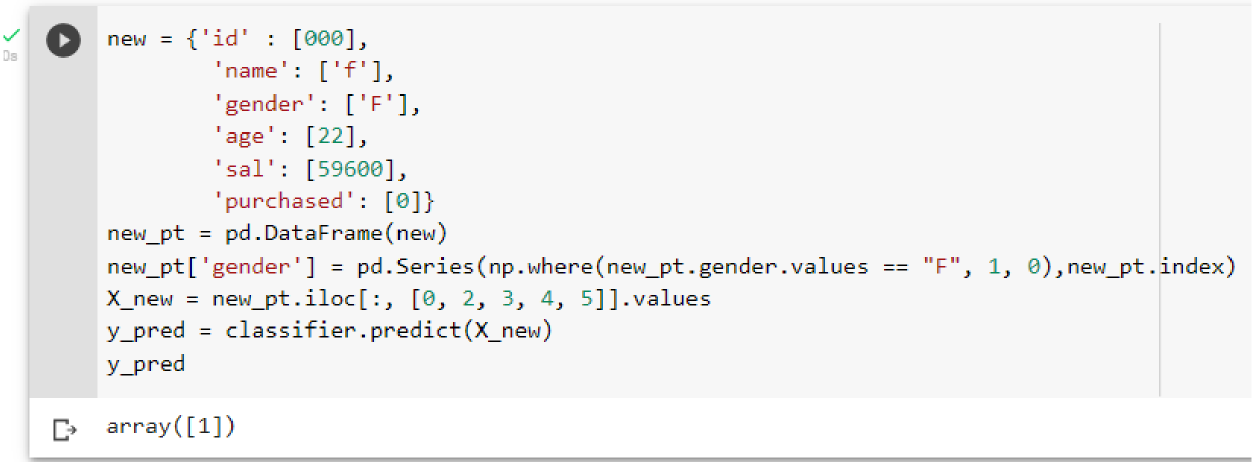
3. fit a KNN model on the model using 1 nearest neighbour



```
classifier = KNeighborsClassifier(n_neighbors =1, metric = 'minkowski', p = 2)
classifier.fit(X, y)
```

```
KNeighborsClassifier(n_neighbors=1)
```

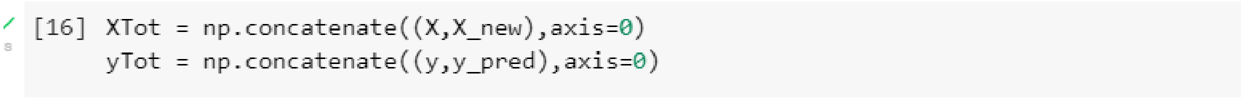
4.predict the class for a new data point.



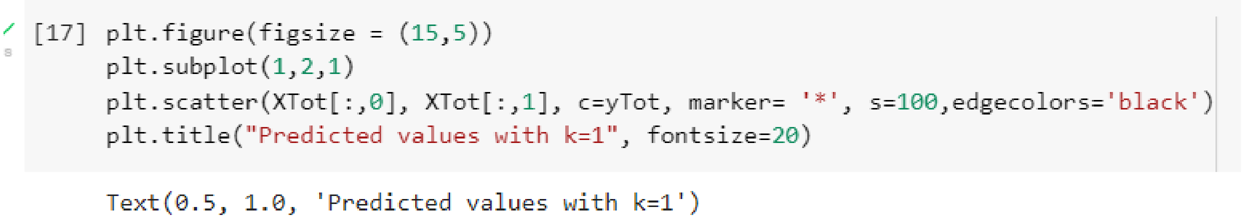
```
new = {'id' : [000],
        'name': ['f'],
        'gender': ['F'],
        'age': [22],
        'sal': [59600],
        'purchased': [0]}
new_pt = pd.DataFrame(new)
new_pt['gender'] = pd.Series(np.where(new_pt.gender.values == "F", 1, 0),new_pt.index)
X_new = new_pt.iloc[:, [0, 2, 3, 4, 5]].values
y_pred = classifier.predict(X_new)
y_pred
```

```
array([1])
```

5.When we plot all the data along with the new point and class, check for its label.

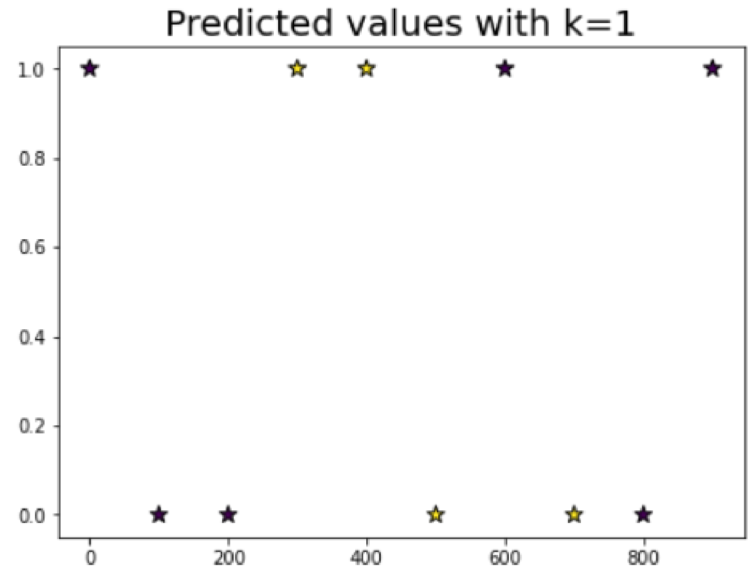


```
[16] XTot = np.concatenate((X,X_new),axis=0)
     yTot = np.concatenate((y,y_pred),axis=0)
```

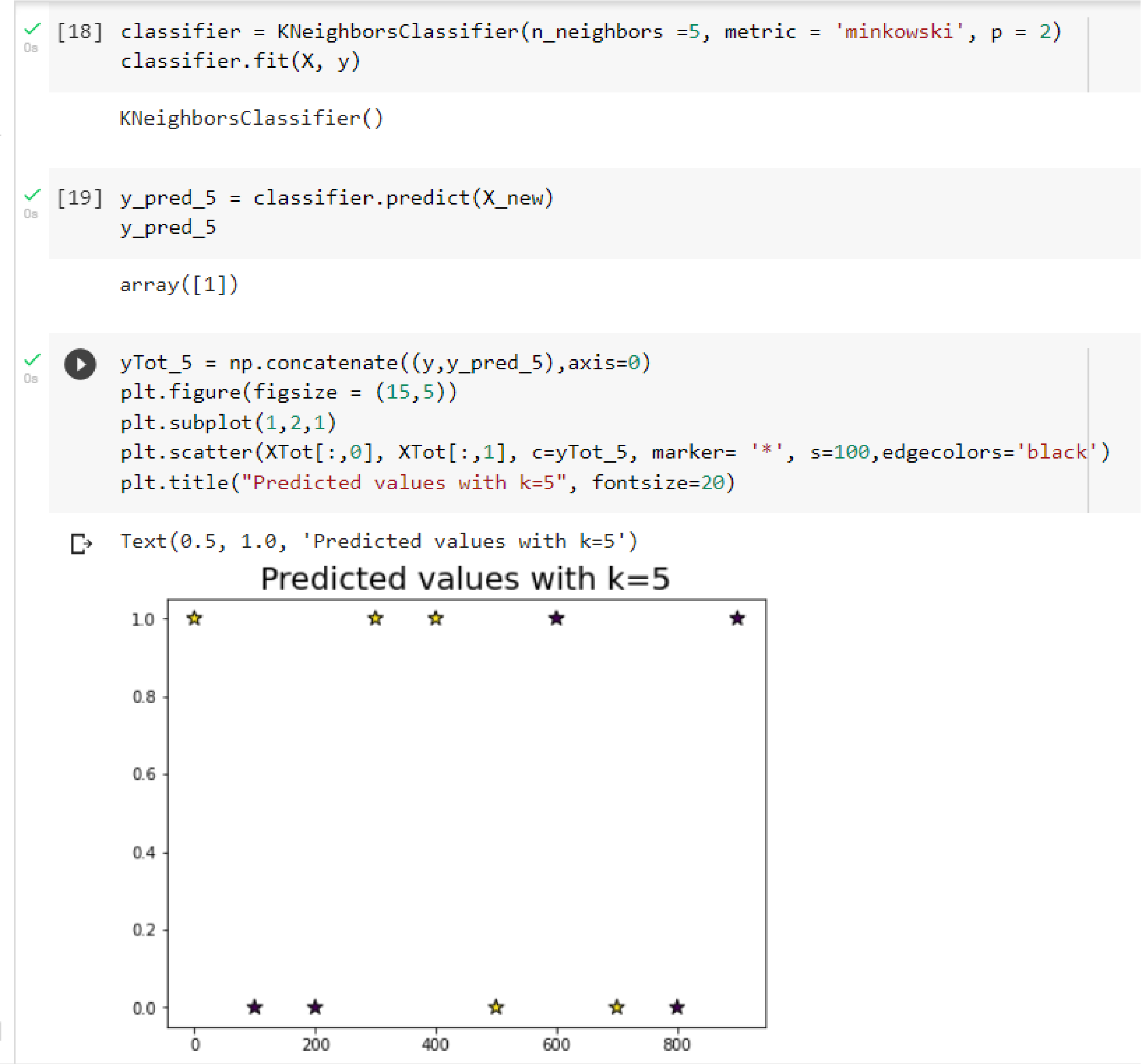


```
[17] plt.figure(figsize = (15,5))
     plt.subplot(1,2,1)
     plt.scatter(XTot[:,0], XTot[:,1], c=yTot, marker= '*', s=100,edgecolors='black')
     plt.title("Predicted values with k=1", fontsize=20)
```

```
Text(0.5, 1.0, 'Predicted values with k=1')
```



6.When k value =5, how it is predicted.



```
[18] classifier = KNeighborsClassifier(n_neighbors =5, metric = 'minkowski', p = 2)
     classifier.fit(X, y)

     KNeighborsClassifier()
```

```
[19] y_pred_5 = classifier.predict(X_new)
     y_pred_5

     array([1])
```

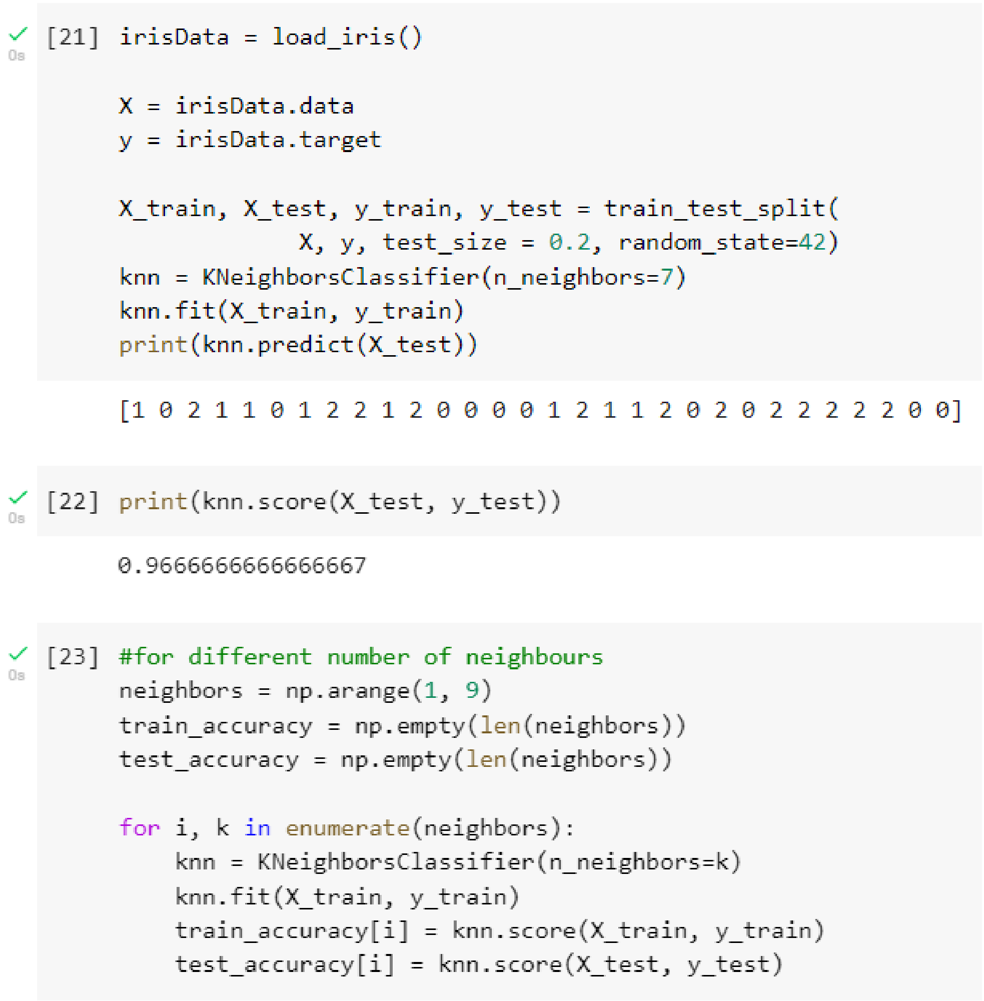
```
yTot_5 = np.concatenate((y,y_pred_5),axis=0)
plt.figure(figsize = (15,5))
plt.subplot(1,2,1)
plt.scatter(XTot[:,0], XTot[:,1], c=yTot_5, marker= '*', s=100,edgecolors='black')
plt.title("Predicted values with k=5", fontsize=20)

Text(0.5, 1.0, 'Predicted values with k=5')
```

7. Write down your inferences when k value varies.

Inference: When the k value changes, the grouping of different clusters together also changes. A point which was previously grouped with some points is now grouped with different points. The prediction, however does not vary that much.

8. Implement KNN on iris dataset and observe the inferences.



```
[21] irisData = load_iris()

     X = irisData.data
     y = irisData.target

     X_train, X_test, y_train, y_test = train_test_split(
                  X, y, test_size = 0.2, random_state=42)
     knn = KNeighborsClassifier(n_neighbors=7)
     knn.fit(X_train, y_train)
     print(knn.predict(X_test))
```

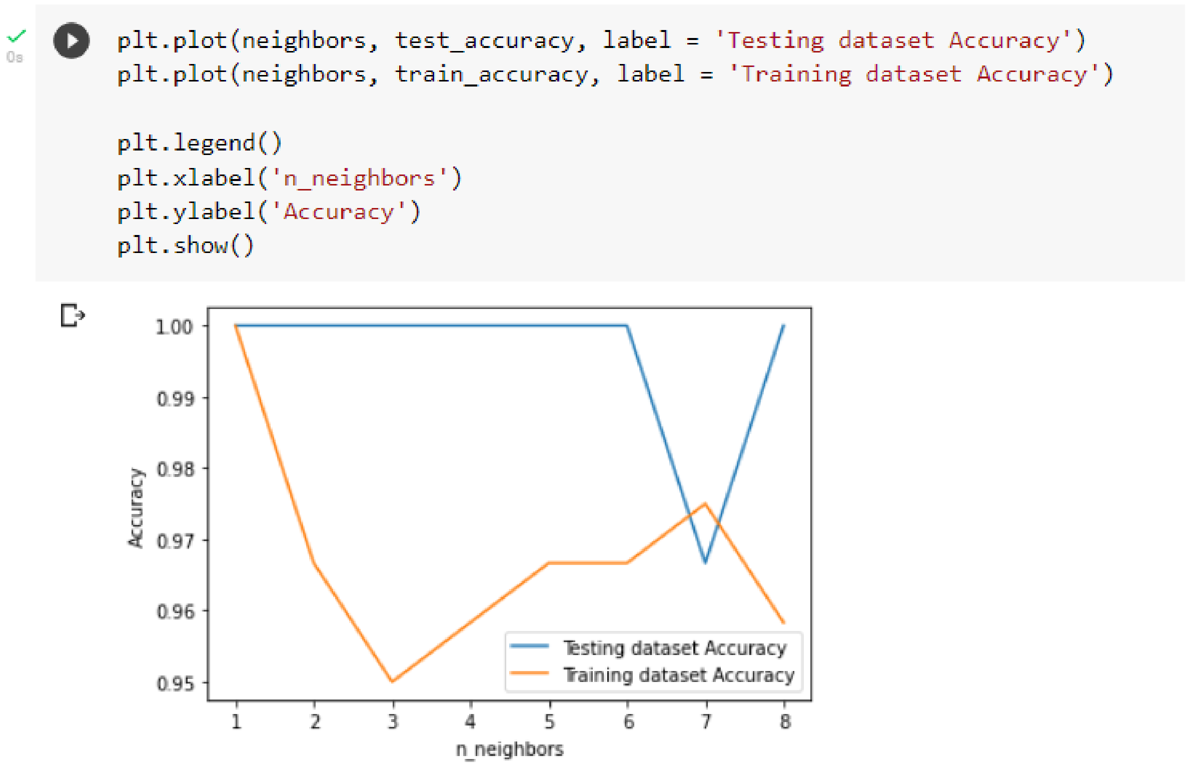
```
[1 0 2 1 1 0 1 2 2 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
```

```
[22] print(knn.score(X_test, y_test))
```

```
0.9666666666666667
```

```
[23] #for different number of neighbours
     neighbors = np.arange(1, 9)
     train_accuracy = np.empty(len(neighbors))
     test_accuracy = np.empty(len(neighbors))

     for i, k in enumerate(neighbors):
         knn = KNeighborsClassifier(n_neighbors=k)
         knn.fit(X_train, y_train)
         train_accuracy[i] = knn.score(X_train, y_train)
         test_accuracy[i] = knn.score(X_test, y_test)
```



```
plt.plot(neighbors, test_accuracy, label = 'Testing dataset Accuracy')
plt.plot(neighbors, train_accuracy, label = 'Training dataset Accuracy')

plt.legend()
plt.xlabel('n_neighbors')
plt.ylabel('Accuracy')
plt.show()
```