No. Experiment : 01
Name of the Experiment : Classification using K-Nearest-Neighbor algorithm for Number Data .

Theory:
The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

In the classification setting, the K-nearest neighbor algorithm essentially boils down to forming a majority vote between the K most similar instances to a given “unseen” observation. Similarity is defined according to a distance metric between two data points. A popular choice is the Euclidean distance given by

$$d(x, x') = \sqrt{(x_1 - x'_1)^2 + \ldots + (x_n - x'_n)^2}$$

Advantages and Disadvantages:
Advantage:

1. The K-NN algorithm is very easy to implement.

2. KNN is called Lazy Learner (Instance based learning). It does not learn anything in the training period.

3. Since the KNN algorithm requires no training before making predictions, new data can be added seamlessly which will not impact the accuracy of the algorithm.

4. KNN is very easy to implement. There are only two parameters required to implement KNN i.e. the value of K and the distance function.

Disadvantage:
1. It does struggle when Dataset is large. In large datasets, performance reduce.
2. We need to do feature scaling (standardization and normalization) before applying KNN algorithm to any dataset.
3. KNN is sensitive to noise in the dataset. We need to manually impute missing values and remove outliers.

Algorithm:

1 START
2 Define X, Y
3 Define Test-Train Split
4 Import KNeighboursClassifier and Initialize
5 Fit X and Y in our classifier Model
6 Call the Classifier Model
7 Calculate ‘y_pred’ and ‘pred_train’ and their confusion matrix and compare them
8 END

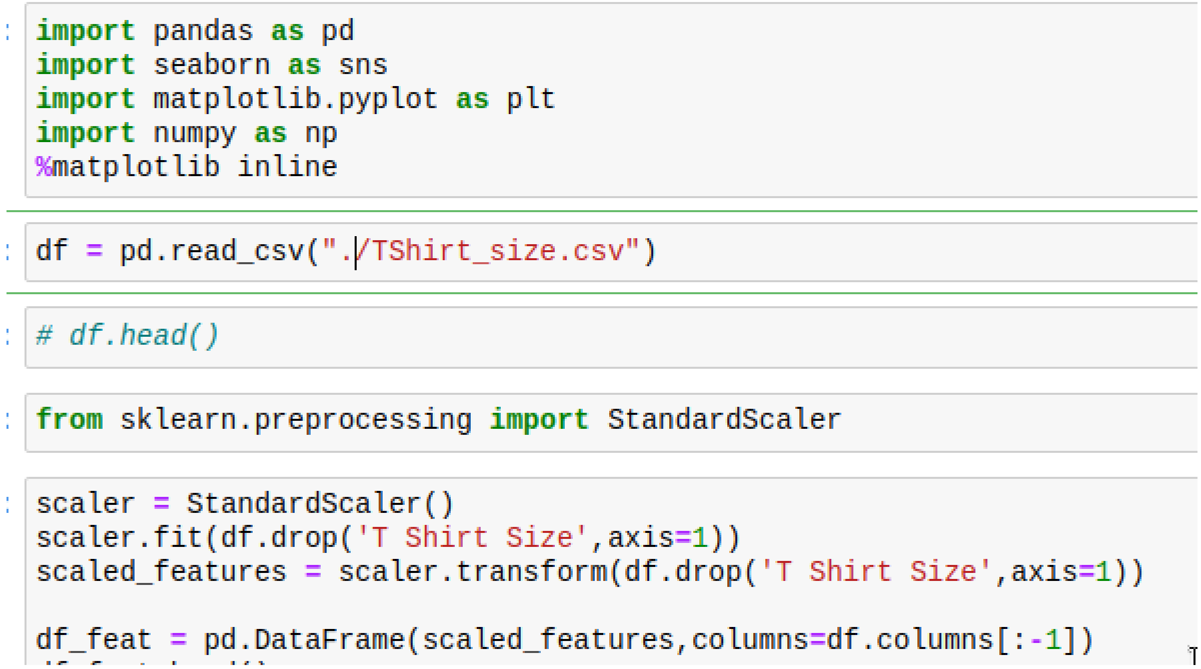
Pseudocode:

```
kNN (dataset, sample){
   1. Go through each item in the dataset, and calculate the "distance"
   from that data item to specific sample.
   2. Classify the sample as the majority class between K samples in
   the dataset having minimum distance to the sample.
}
```

Dataset:

Used a dataset that was based on ‘Tshirt_Size’, provided in CSV format.

Screenshot of the task:



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

```python
df = pd.read_csv("./TShirt_size.csv")
```

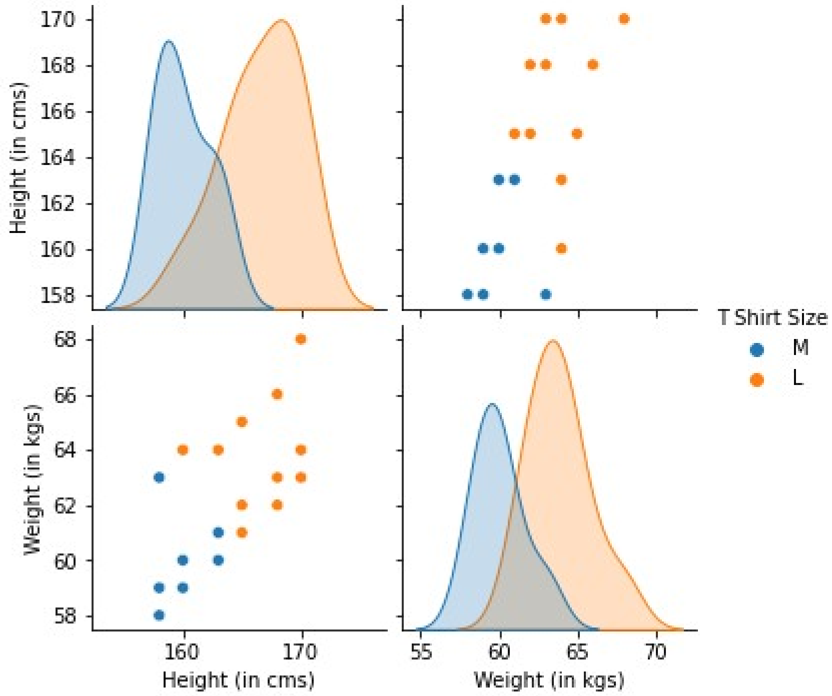
```python
# df.head()
```

```python
from sklearn.preprocessing import StandardScaler
```

```python
scaler = StandardScaler()
scaler.fit(df.drop('T Shirt Size',axis=1))
scaled_features = scaler.transform(df.drop('T Shirt Size',axis=1))

df_feat = pd.DataFrame(scaled_features,columns=df.columns[:-1])
```

```
sns.pairplot(df,hue='T Shirt Size')
```

```
<seaborn.axisgrid.PairGrid at 0x7fa2327e2ac0>
```



```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
        scaled_features,
        df['T Shirt Size'],
        test_size=0.25
)
```

```
from sklearn.neighbors import KNeighborsClassifier
```

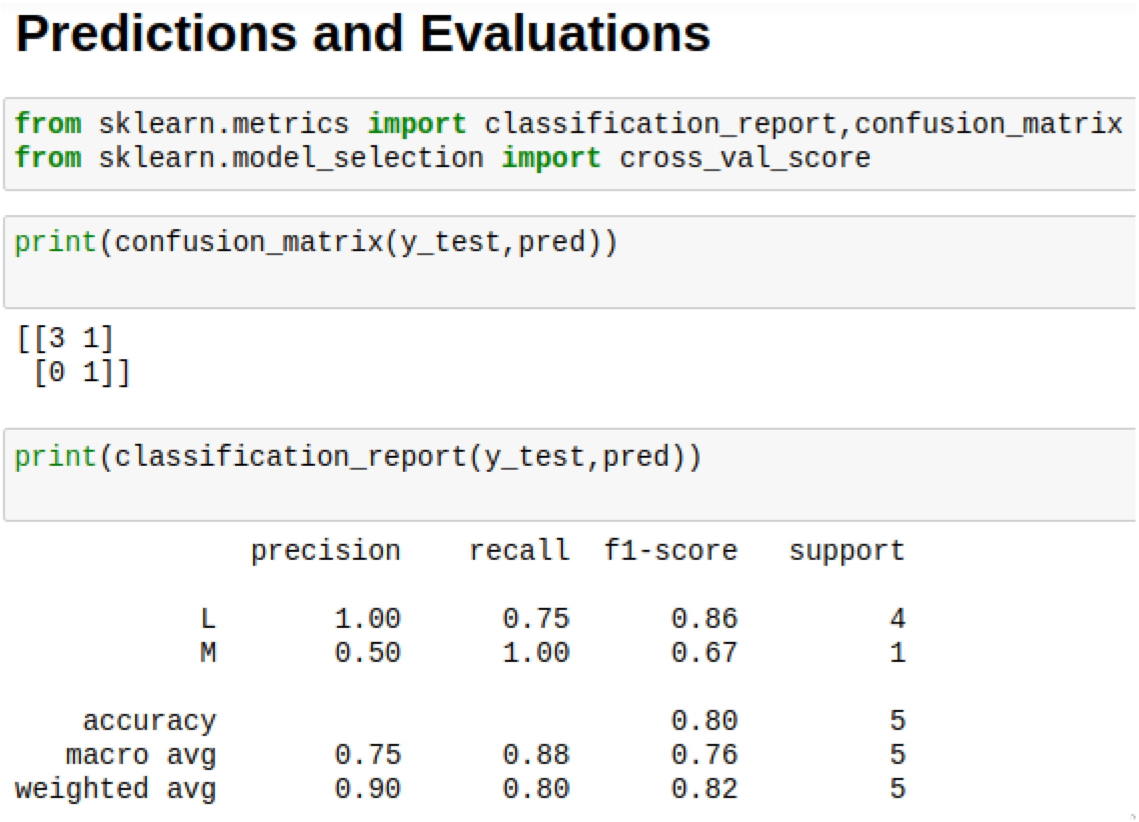
Using KNN

We'll start with k=1.

```
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train,y_train)
```

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

```
pred = knn.predict(X_test)
```



Predictions and Evaluations

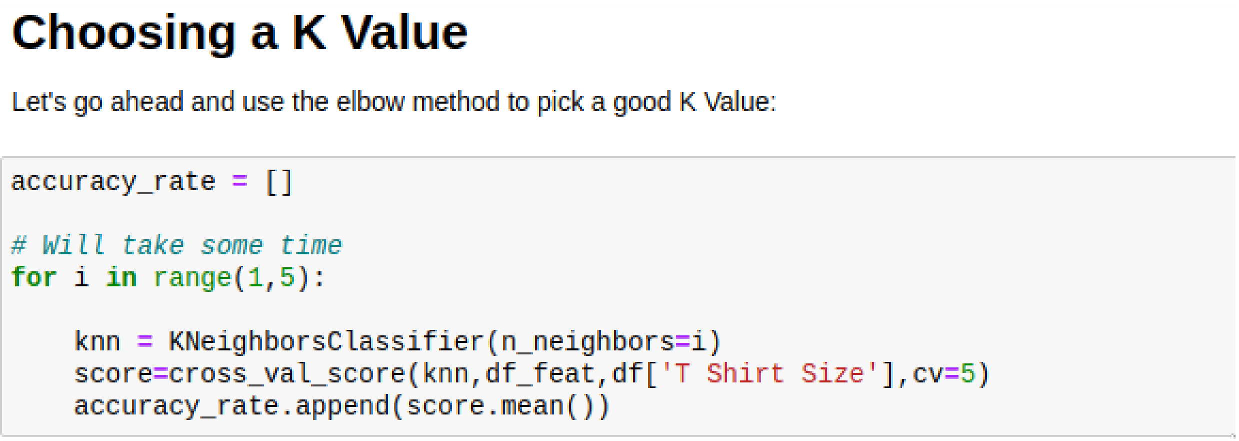
```
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.model_selection import cross_val_score
```

```
print(confusion_matrix(y_test,pred))
```

```
[[3 1]
 [0 1]]
```

```
print(classification_report(y_test,pred))
```

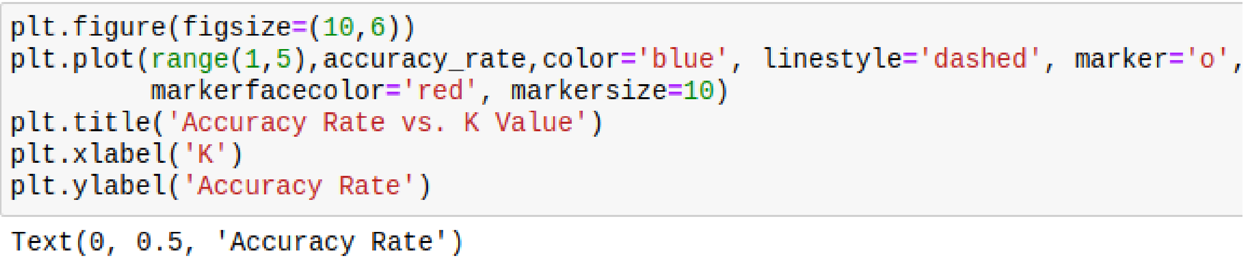
```
              precision    recall  f1-score   support

           L       1.00      0.75      0.86         4
           M       0.50      1.00      0.67         1

    accuracy                           0.80         5
   macro avg       0.75      0.88      0.76         5
weighted avg       0.90      0.80      0.82         5
```



Choosing a K Value

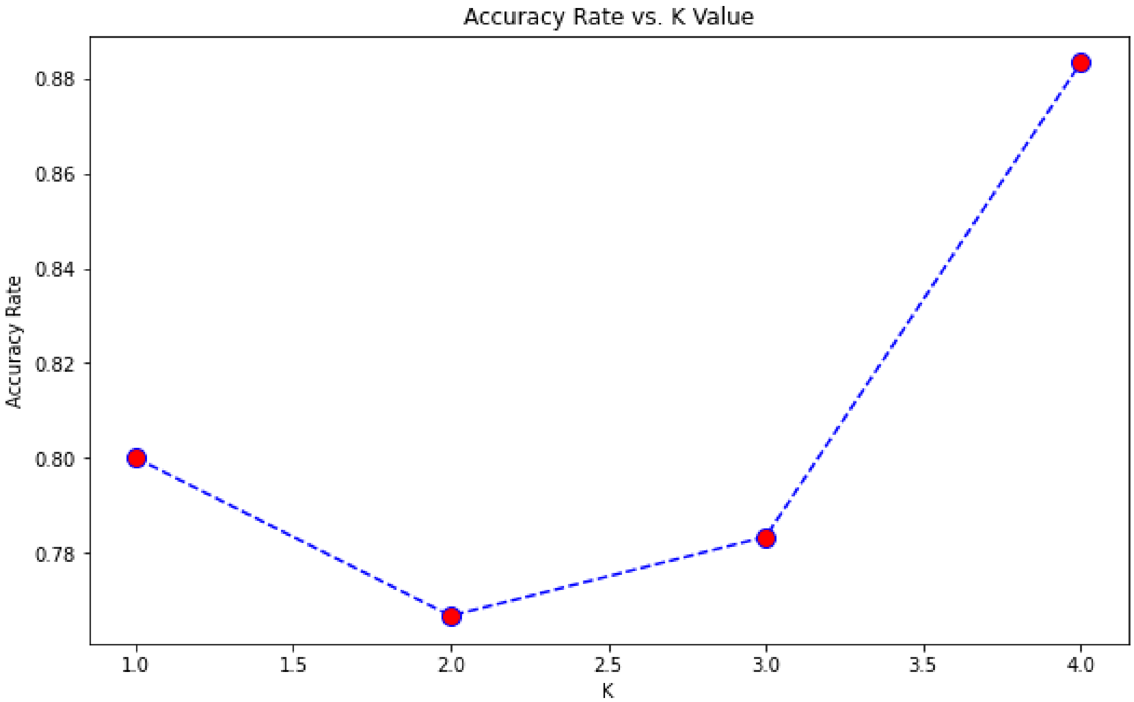
Let's go ahead and use the elbow method to pick a good K Value:

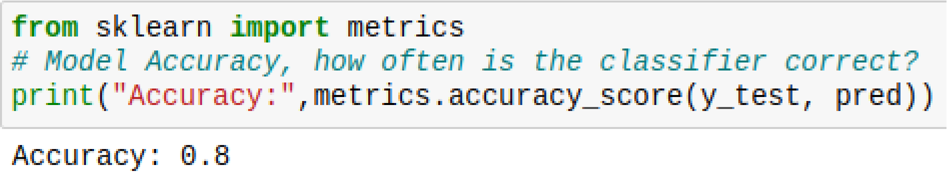
```
accuracy_rate = []

# Will take some time
for i in range(1,5):

    knn = KNeighborsClassifier(n_neighbors=i)
    score=cross_val_score(knn,df_feat,df['T Shirt Size'],cv=5)
    accuracy_rate.append(score.mean())
```



```
plt.figure(figsize=(10,6))
plt.plot(range(1,5),accuracy_rate,color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Accuracy Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Accuracy Rate')
```

```
Text(0, 0.5, 'Accuracy Rate')
```





```
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, pred))
```

```
Accuracy: 0.8
```

Result:

Accuracy for KNN (with K = 3)= 0.8

Conclusion:

The dataset accuracy expected more. Increasing the test data can improve the result.

Contribution by Members:

Abdullah Al Hasib (1105081)
Md Mamun Mazumdar (1105054)