No. Experiment : 02

Name of the Experiment : Classification using K-Nearest-Neighbor algorithm for Nominal 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 ‘Student Survey ’, provided in CSV format.

Screenshot of the task Result:

```
%matplotlib inline
import pandas as pd
df = pd.read_csv('xAPI-Edu-Data.csv')
# df.shape

# df.isnull().sum()
columns = df.select_dtypes(include=['object']).columns.to_list()
# columns
```

```
from sklearn.preprocessing import StandardScaler,LabelEncoder
label=LabelEncoder()
```

```
def encode_labels(df,labels_to_encode):
    for column in labels_to_encode:
        df[column] = label.fit_transform(df[column])
    return df
```

```
print('Shape of dataframe before encoding : ', df.shape)
print('Shape of dataframe after encoding (using dummy encoder) : ',
          pd.get_dummies(df).shape) # this is HUGE!

df_labelled = encode_labels(df,columns)
print('Shape of dataframe after encoding (using label encoder) : ',
          df_labelled.shape)
```

```
Shape of dataframe before encoding :  (480, 17)
Shape of dataframe after encoding (using dummy encoder) :  (480, 75)
Shape of dataframe after encoding (using label encoder) :  (480, 17)
```

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()

X = pd.DataFrame(
sc_X.fit_transform(df_labelled.drop(['Class'], axis = 1))
)
# X.head()
y=df_labelled.Class
```

```
from sklearn.model_selection import train_test_split, cross_val_score

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3)

#Import knearest neighbors Classifier model
from sklearn.neighbors import KNeighborsClassifier
#Create KNN Classifier with k = 5
knn = KNeighborsClassifier(n_neighbors=5)
```

```
#Train the model using the training sets
knn.fit(X_train, y_train)
#Predict the response for test dataset
y_pred = knn.predict(X_test)
```

```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

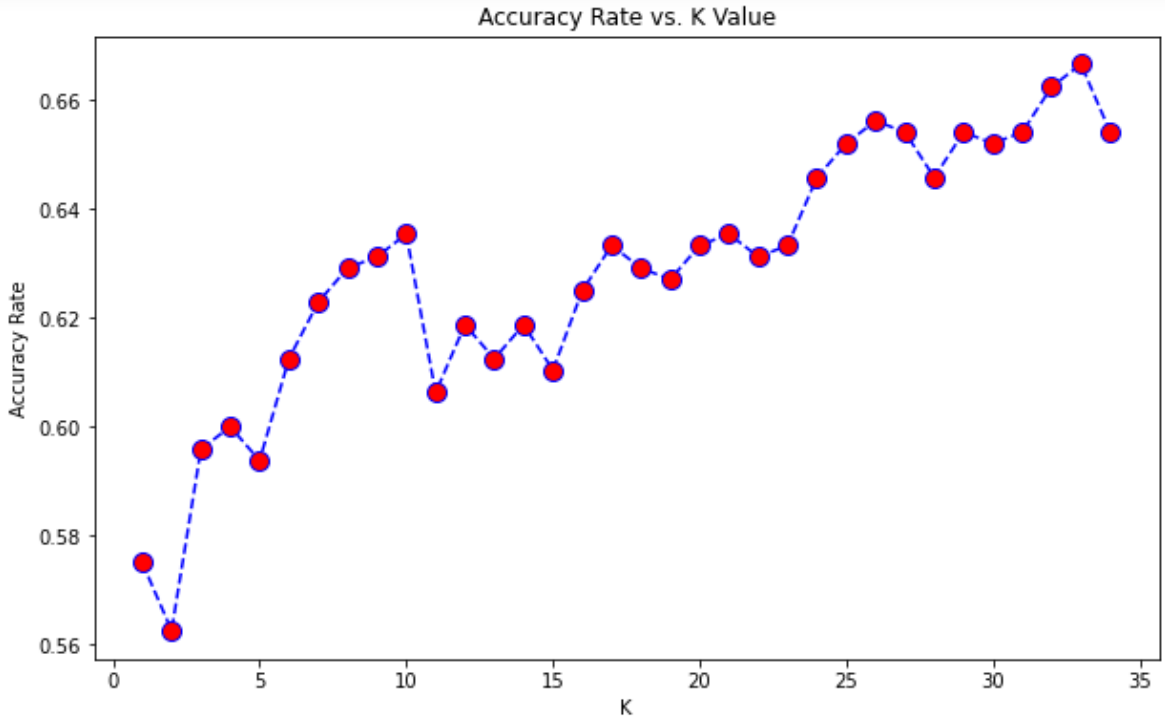
```
Accuracy: 0.7013888888888888
```

```
accuracy_rate = []
K_MAX = 35
# Will take some time
for i in range(1,K_MAX):

    knn = KNeighborsClassifier(n_neighbors=i)
    score=cross_val_score(knn,X,y,cv=5)
    accuracy_rate.append(score.mean())
```

```
import matplotlib.pyplot as plt
accuracy_rate
plt.figure(figsize=(10,6))
plt.plot(range(1,K_MAX),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')
```




```
knn = KNeighborsClassifier(n_neighbors=32)
#Predict the response for test dataset
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7291666666666666
```

Conclusion: Best accuracy found when 32 < K <= 34

Contribution by Members:

Abdullah Al Hasib (1105081)

Md Abdullah Al Mamun Mazumder (1105054)