```
#Authors: Achanta Sai Krishna,Kuralanbu,Vimal Dharshan
#Objective: To find the optimal k value
#Input: Dataset
#Output: Accuracy and Confusion Matrix
import pandas as pd #data analysis toolkit
import matplotlib.pyplot as plt #for plotting graphs
import numpy as np #high level computations
%matplotlib inline
```

```
from sklearn.preprocessing import StandardScaler #standardization of values
from sklearn.preprocessing import MinMaxScaler #normalization of values
from sklearn.model_selection import train_test_split #to split data
from sklearn.neighbors import KNeighborsClassifier #KNN classifier
from sklearn.metrics import confusion_matrix,accuracy_score #to get confusion matrix and a
from sklearn.model_selection import cross_val_score #to perform evaluation and cross-valid
```

```
data_set = pd.read_csv("/content/framingham (1).csv") #dataset_input
```

```
data_set=data_set.fillna(data_set.mean()) #mean for missing data
```

```
data_set = np.round(data_set, decimals=2) #rounding all values in dataset to 2 decimal pla
data_set.head() #first 5 values in dataset
```

| | Sex | age | Chest Pain Type | currentSmoker | cigsPerDay | BPMeds | prevalentStroke | prevalentHyp |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 39 | 4 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 46 | 2 | 0 | 0 | 0 | 0 | 0 |
| | | | | 1 | 20 | 0 | 0 | 0 |
| | | | | 1 | 30 | 0 | 0 | 1 |

Saved successfully!

```
data_set.tail() #It prints the last 5 values in dataset
```

| | Sex | age | Chest Pain Type | currentSmoker | cigsPerDay | BPMeds | prevalentStroke | prevalent |
|---|---|---|---|---|---|---|---|---|
| **4235** | 0 | 48 | 2 | 1 | 20 | 0 | 0 | |
| **4236** | 0 | 44 | 1 | 1 | 15 | 0 | 0 | |
| **4237** | 0 | 52 | 2 | 0 | 0 | 0 | 0 | |
| **4238** | 1 | 40 | 3 | 0 | 0 | 0 | 0 | |

```
# no of rows and columns in the data set
data_set.shape
```

```
(4240, 16)
```

```
#checking for missing values
data_set.isnull().sum  #False means no missing data
```

```
<bound method NDFrame._add_numeric_operations.<locals>.sum of         Sex    age  
Chest Pain Type  currentSmoker  cigsPerDay  BPMeds  \
0     False  False            False          False       False   False
1     False  False            False          False       False   False
2     False  False            False          False       False   False
3     False  False            False          False       False   False
4     False  False            False          False       False   False
...     ...    ...              ...            ...         ...     ...
4235  False  False            False          False       False   False
4236  False  False            False          False       False   False
4237  False  False            False          False       False   False
4238  False  False            False          False       False   False
4239  False  False            False          False       False   False

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
0               False         False     False    False  False  False  False
1               False         False     False    False  False  False  False
2               False         False     False    False  False  False  False
3               False         False     False    False  False  False  False
4               False         False     False    False  False  False  False
...               ...           ...       ...      ...    ...    ...    ...
4235            False         False     False    False  False  False  False
4236            False         False     False    False  False  False  False
4237            False         False     False    False  False  False  False
4238            False         False     False    False  False  False  False
4239            False         False     False    False  False  False  False

                                 t
0         False    False   False
1         False    False   False
2         False    False   False
3         False    False   False
4         False    False   False
...         ...      ...     ...
4235      False    False   False
4236      False    False   False
4237      False    False   False
4238      False    False   False
4239      False    False   False

[4240 rows x 16 columns]>
```



```
#Statistical measure about the dataset
data_set.describe()
```

| | Sex | age | Chest Pain Type | currentSmoker | cigsPerDay | BPMeds |
|---|---|---|---|---|---|---|
| **count** | 4240.000000 | 4240.000000 | 4240.000000 | 4240.000000 | 4240.000000 | 4240.000000 |
| **mean** | 0.429245 | 49.580189 | 1.930425 | 0.494104 | 8.944340 | 0.029245 |
| **std** | 0.495027 | 8.572942 | 1.053026 | 0.500024 | 11.904777 | 0.168513 |
| **min** | 0.000000 | 32.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 0.000000 | 42.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 |
| **50%** | 0.000000 | 49.000000 | 2.000000 | 0.000000 | 0.000000 | 0.000000 |
| **75%** | 1.000000 | 56.000000 | 3.000000 | 1.000000 | 20.000000 | 0.000000 |
| **max** | 1.000000 | 70.000000 | 4.000000 | 1.000000 | 70.000000 | 1.000000 |

```
#counting the  no of people's having Heart Disease ('1') and not having Heart Disease
data_set['target'].value_counts()
```

```
0    3596
1     644
Name: target, dtype: int64
```

```
dset_modified = data_set.drop('target',axis=1) #dataset without class feature
```

```
data_set_feat = pd.DataFrame(dset_modified,columns=data_set.columns[:-1]) #dataset without
```

```
data_set_feat = np.round(data_set_feat, decimals=2) #rounding all values to 2 decimal plac
```

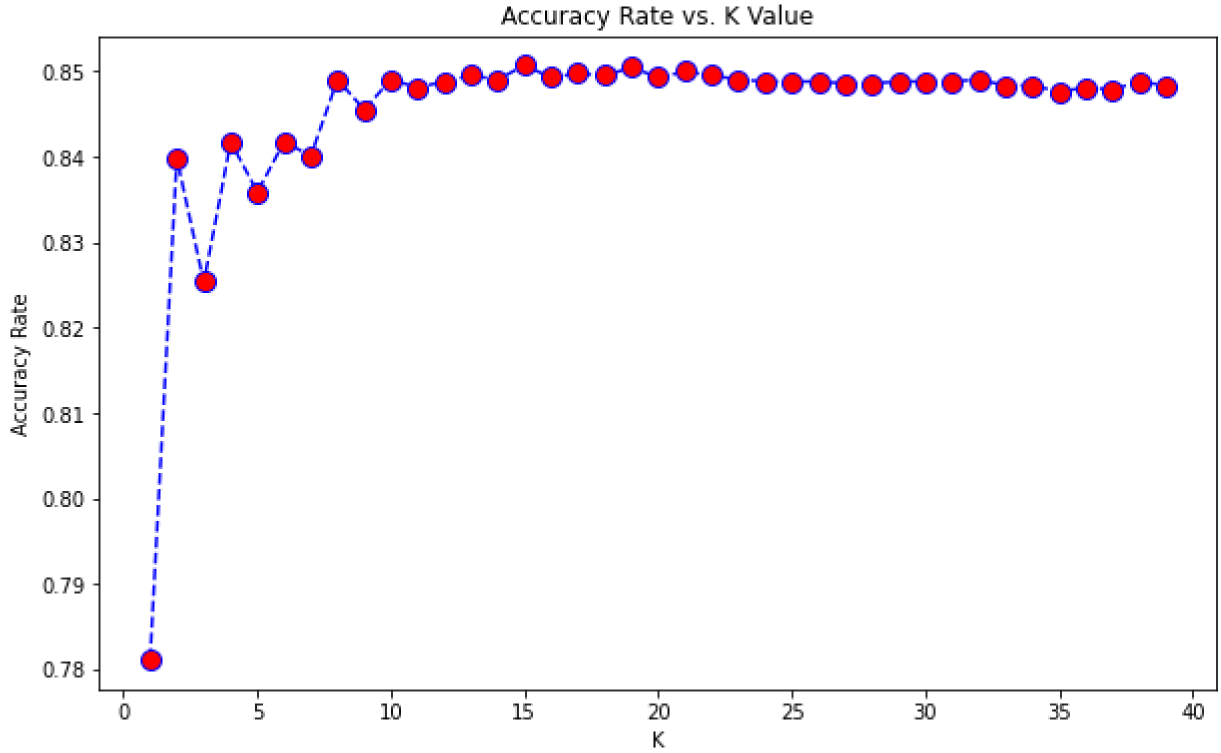
Saved successfully!

```
test = train_test_split(data_set_feat,data_set['target
```

```
#Computation of accuracy rates for various neighbor values
Accurate_rates = []

for i in range(1,40):

  k_nearest_neighbour = KNeighborsClassifier(n_neighbors=i)
  final_score=cross_val_score(k_nearest_neighbour,data_set_feat,data_set['target'], cv=5)
  Accurate_rates.append(final_score.mean())
```

```
#plot
plt.figure(figsize=(10,6))

plt.plot(range(1,40),Accurate_rates,color='blue',linestyle='dashed',marker='o',markerfacec
plt.title('Accuracy Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Accuracy Rate')
```

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




```
max_index = Accurate_rates.index(max(Accurate_rates)) #Best case identifier

k_nearest_neighbour = KNeighborsClassifier(n_neighbors=max_index)

k_nearest_neighbour.fit(one_train,two_train)
prediction = k_nearest_neighbour.predict(one_test)

print('For K=',max_index)
print('Confusion matrix:')
print('\n')
print(confusion_matrix(two_test,prediction)) #Confusion Matrix
print('\n')
cy_score(two_test,prediction),2)*100,'%')
```

Saved successfully!

```
For K= 20
Confusion matrix:


[[1066    1]
 [ 204    1]]


Accuracy rate:  84.0 %
```

```
t = 100 #Random K value

k_nearest_neighbour = KNeighborsClassifier(n_neighbors=t)

k_nearest_neighbour.fit(one_train,two_train)
prediction = k_nearest_neighbour.predict(one_test)
```

```
print('For K=' ,t)
print('Confusion Matrix:')
print('\n')
print(confusion_matrix(two_test,prediction)) #Confusion Matrix
print('\n')
print('Accuracy rate: ',round(accuracy_score(two_test,prediction),2)*100,'%')
#Accuracy rate
```

```
For K= 100
Confusion Matrix:


[[1067    0]
 [ 205    0]]


Accuracy rate:  84.0 %
```

```
scaled = MinMaxScaler() #function Minmax scaler for normalising values
```

```
scaled.fit(data_set.drop('target',axis=1)) #dropping class-feature
```

```
MinMaxScaler()
```

```
dset_modified = scaled.transform(data_set.drop('target',axis=1))#dropping class-feature
```

```
data_set_feat = pd.DataFrame(dset_modified,columns=data_set.columns[:-1]) #dropping class-
```

```
data_set_feat = np.round(data_set_feat, decimals=2) #rounding all values to 2 decimals
data_set_feat.head() #dataset_after_normalization
```



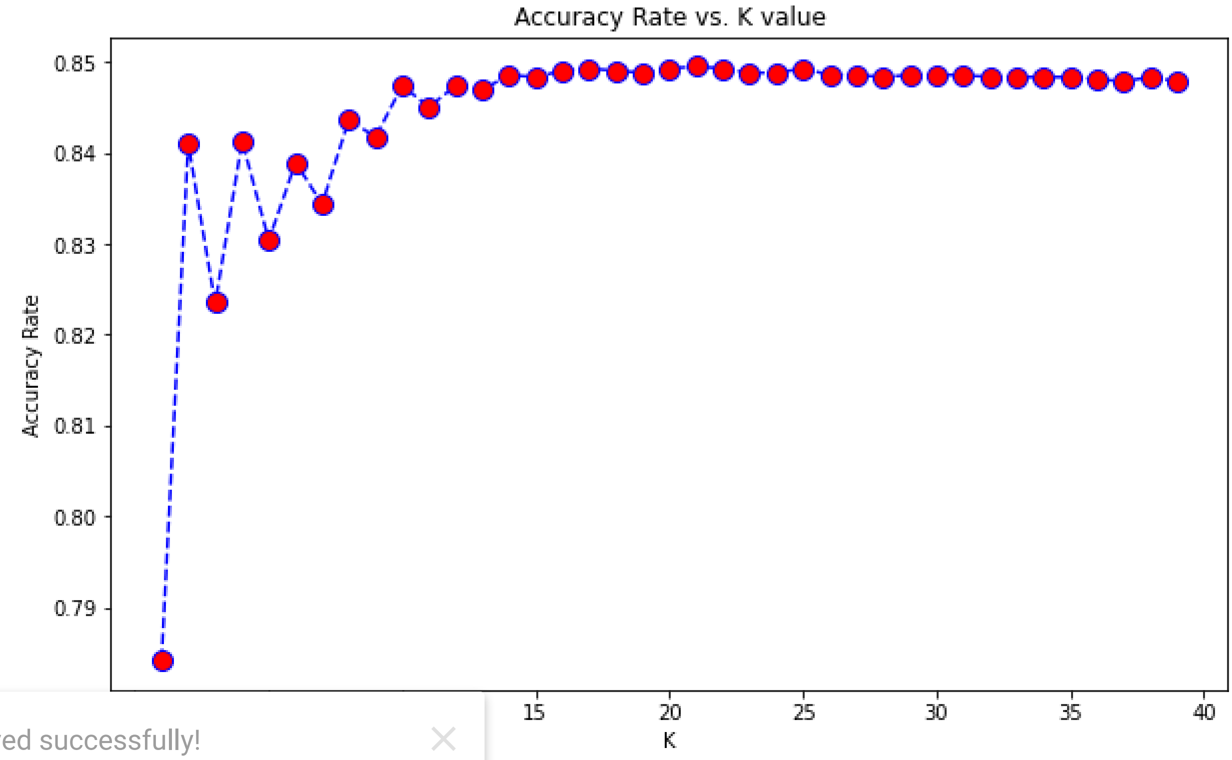
| | | | Chest | oker | cigsPerDay | BPMeds | prevalentStroke | prevalentHyp |
|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 0.18 | 1.00 | 0.0 | 0.00 | 0.0 | 0.0 | 0.0 |
| **1** | 0.0 | 0.37 | 0.50 | 0.0 | 0.00 | 0.0 | 0.0 | 0.0 |
| **2** | 1.0 | 0.42 | 0.25 | 1.0 | 0.29 | 0.0 | 0.0 | 0.0 |
| **3** | 0.0 | 0.76 | 0.75 | 1.0 | 0.43 | 0.0 | 0.0 | 1.0 |

```
#test_train split with test_size 30% and train_size 70%
one_train, one_test, two_train, two_test = train_test_split(data_set_feat,data_set['target
```

```
#Computation of accuracy rates for various neighbour values
```

```
Accurate_rates = []

for i in range(1,40):

  k_nearest_neighbour = KNeighborsClassifier(n_neighbors=i)
  final_score=cross_val_score(k_nearest_neighbour,data_set_feat,data_set['target'],cv=5)
  Accurate_rates.append(final_score.mean())

plt.figure(figsize=(10,6))

plt.plot(range(1,40),Accurate_rates,color='blue',linestyle='dashed',marker='o',markerfacec
plt.title('Accuracy Rate vs. K value')
plt.xlabel('K')
plt.ylabel('Accuracy Rate')
```

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




Saved successfully!

Saved successfully!