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

```
dataset=pd.read_csv('Classified Data.csv')
```

```
dataset.head()
```

	Unnamed: 0	WTT	PTI	EQW	SBI	LQE	QWG	FDJ	PJF	HQE	
0	0	0.913917	1.162073	0.567946	0.755464	0.780862	0.352608	0.759697	0.643798	0.879422	1.23
1	1	0.635632	1.003722	0.535342	0.825645	0.924109	0.648450	0.675334	1.013546	0.621552	1.49
2	2	0.721360	1.201493	0.921990	0.855595	1.526629	0.720781	1.626351	1.154483	0.957877	1.28
3	3	1.234204	1.386726	0.653046	0.825624	1.142504	0.875128	1.409708	1.380003	1.522692	1.15
4	4	1.279491	0.949750	0.627280	0.668976	1.232537	0.703727	1.115596	0.646691	1.463812	1.41

```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

```
print (X)
```

```
[[0.00000000e+00 9.13917327e-01 1.16207271e+00 ... 6.43797564e-01
  8.79422091e-01 1.23140944e+00]
 [1.00000000e+00 6.35631904e-01 1.00372163e+00 ... 1.01354599e+00
  6.21552215e-01 1.49270160e+00]
 [2.00000000e+00 7.21359808e-01 1.20149262e+00 ... 1.15448315e+00
  9.57877023e-01 1.28559679e+00]
 ...
 [9.97000000e+02 1.13546983e+00 9.82462329e-01 ... 3.89584420e-01
  9.19191428e-01 1.38550400e+00]
 [9.98000000e+02 1.08489449e+00 8.61769167e-01 ... 1.06133794e+00
  1.27745578e+00 1.18806277e+00]
 [9.99000000e+02 8.37459538e-01 9.61183523e-01 ... 9.07961870e-01
  1.25718998e+00 1.36483726e+00]]
```

```
dataset.shape
```

```
(1000, 12)
```

```
dataset.columns
```

```
Index(['Unnamed: 0', 'WTT', 'PTI', 'EQW', 'SBI', 'LQE', 'QWG', 'FDJ', 'PJF',
       'HQE', 'NXJ', 'TARGET CLASS'],
      dtype='object')
```

```
dataset.describe()
```

	Unnamed: 0	WTT	PTI	EQW	SBI	LQE	QWG	
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000
mean	499.500000	0.949682	1.114303	0.834127	0.682099	1.032336	0.943534	0.963
std	288.819436	0.289635	0.257085	0.291554	0.229645	0.243413	0.256121	0.255
min	0.000000	0.174412	0.441398	0.170924	0.045027	0.315307	0.262389	0.295
25%	249.750000	0.742358	0.942071	0.615451	0.515010	0.870855	0.761064	0.784
50%	499.500000	0.940475	1.118486	0.813264	0.676835	1.035824	0.941502	0.945
75%	749.250000	1.163295	1.307904	1.028340	0.834317	1.198270	1.123060	1.134
max	999.000000	1.721779	1.833757	1.722725	1.634884	1.650050	1.666902	1.713

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Unnamed: 0    1000 non-null   int64  
 1   WTT           1000 non-null   float64
 2   PTI           1000 non-null   float64
 3   EQW           1000 non-null   float64
 4   SBI           1000 non-null   float64
 5   LQE           1000 non-null   float64
 6   QWG           1000 non-null   float64
 7   FDJ           1000 non-null   float64
 8   PJF           1000 non-null   float64
 9   HQE           1000 non-null   float64
 10  NXJ           1000 non-null   float64
 11  TARGET CLASS  1000 non-null   int64  
dtypes: float64(10), int64(2)
memory usage: 93.9 KB
```

```
dataset.isnull().sum()
```

```
Unnamed: 0      0
WTT             0
PTI             0
EQW             0
SBI             0
LQE             0
QWG             0
FDJ             0
PJF             0
HQE             0
NXJ             0
TARGET CLASS    0
dtype: int64
```

dataset[dataset.isnull().any(axis=1)].head()

```
dataset[dataset.isnull().any(axis=1)].head()
```

Unnamed: 0	WTT	PTI	EQW	SBI	LQE	QWG	FDJ	PJF	HQE	NXJ	TARGET CLASS

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

```
print(X_train)
```

```
[[2.53000000e+02 1.65050251e+00 9.62292220e-01 ... 9.32753905e-01
  1.49851161e+00 1.04786370e+00]
 [6.67000000e+02 6.63910552e-01 1.35608617e+00 ... 1.30971776e+00
  9.70731053e-01 1.39914854e+00]
 [8.50000000e+01 7.62772508e-01 1.50036690e+00 ... 9.96190855e-01
  7.68383871e-01 1.60486991e+00]
 ...
 [6.29000000e+02 1.02748185e+00 8.36071268e-01 ... 1.30250402e+00
  1.22254394e+00 1.41020414e+00]
 [5.59000000e+02 9.69641571e-01 1.41466959e+00 ... 1.43953284e+00
  8.32868956e-01 1.46516688e+00]
 [6.84000000e+02 6.20549749e-01 1.19029710e+00 ... 1.51171562e+00
  1.00557762e+00 1.46433357e+00]]
```

```
print(y_train)
```

```
[1 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0 1 1 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0 1 0 0 1
 1 0 0 0 0 1 0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 0 1 1 1 0 0 1 0 1 0 1 0 1 0 0
 0 0 0 1 0 1 0 0 1 1 1 0 0 1 1 0 1 1 1 1 0 0 1 0 0 1 1 1 0 0 1 0 0 0 1 0 1 0 1
 0 0 0 1 0 1 0 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 1
 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 0 0 0 0 0 0 0 1 1 1 1
 0 0 0 1 0 0 1 0 0 0 0 1 0 1 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 1
 1 1 0 0 0 0 1 1 0 0 1 0 1 0 1 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 1 0
 0 0 0 0 0 0 1 0 0 1 1 0 1 1 0 1 0 0 1 1 0 0 0 1 0 1 0 0 0 1 1 0 1 1 1 0 1 1 1
 1 1 0 1 0 1 1 1 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 1 0 1 1 0 1 1 1 1
 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 1 1 0 1 1 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 1 0 1 0
 0 1 0 0 1 1 0 1 0 1 1 0 0 1 1 0 0 0 1 0 1 0 0 0 1 0 1 1 0 1 1 0 1 1 0 0 0 0
 1 1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 1 1 0 1 1 0 1 0 1 1 1 0 0 0 1 1 1 0 1 1 0 0
```

```
 0 0 0 1 1 0 0 1 1 0 1 0 1 1 0 0 1 0 0 0 0 1 1 0 1 1 0 1 0 0 1 1 1 0 1 0 1
 0 1 1 1 0 0 0 1 1 1 1 0 0 1 1 0 1 0 1 1 1 1 0 0 0 1 1 1 0 0 1 0 0 0 0 0 1 0
 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 0 0 0 1 1 1 0 0 1 1 0 1 0 0 0 0 1 1 0 1 1
 1 0 0 0 1 1 0 1 0 1 1 0 1 1 1 0 1 1 1 0 0 0 1 1 0 1 0 0 0 1 1 1 1 1 1 0 1 0
 0 0 1 0 0 0 1 0 1 0 1 1 1 0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 0 1 0 0 1 1 0 1 0 1
 1 0 0 1 1 0 1 0 0 1 0 1 1 1 1 1 1 1 0 1 0 0 0 1 1 1 0 0 1 0 0 1 1 1 1 1 0 1 0
 0 1 1 0 0 1 1 1 1 0 0 1 1 0 1 0 0 1 1 1 1 1 0 0 1 0 1 1 0 0 1 0 0 1 0 0 0 0 0
 0 1 0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 1 0 1 1 1 1 1 1 1 0 0 1 0 0 1 1 0 1 1 1 1 0
 0 1 1 1 1 0 0 1 0 0]
```

```
print(X_test)
```

```
[[9.93000000e+02 7.33687006e-01 1.04963599e+00 ... 8.62135005e-01
  1.46480167e+00 1.08875905e+00]
 [8.59000000e+02 7.65895286e-01 1.18660135e+00 ... 1.41710214e+00
  8.89595397e-01 1.67495338e+00]
 [2.98000000e+02 8.27248868e-01 1.17570238e+00 ... 1.44783698e+00
  1.25319080e+00 1.41241025e+00]
 ...
 [2.00000000e+00 7.21359808e-01 1.20149262e+00 ... 1.15448315e+00
  9.57877023e-01 1.28559679e+00]
 [4.78000000e+02 1.17982647e+00 5.68011960e-01 ... 8.54942856e-01
  1.22429534e+00 1.23361938e+00]
 [6.95000000e+02 1.06820331e+00 9.20753335e-01 ... 9.40780687e-01
  1.20931545e+00 1.37666073e+00]]
```

```
print(y_test)
```

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

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
print(X_train)
```

```
[[-0.82755578  2.42811996 -0.61707298 ... -0.53660395  1.14761884
  -1.54259267]
 [ 0.6002397  -0.95466712  0.94102448 ...  0.79885871 -0.61361735
   0.15818677]
 [-1.40695104 -0.61569319  1.51189015 ... -0.31186708 -1.28886235
   1.15420666]
 ...
 [ 0.46918601  0.29193162 -1.11648275 ...  0.77330271  0.22669773
   0.21171356]
 [ 0.22777132  0.09361119  1.17281747 ...  1.2587521  -1.07367165
   0.47782096]
 [ 0.65886898 -1.10334091  0.28505828 ...  1.51447265 -0.49733221
   0.47378639]]
```

```
print(X_test)
```

```
[[ 1.72454242 -0.7154204  -0.27148588 ... -0.78678418  1.0351267
  -1.34459387]
 [ 1.26240572 -0.60498594  0.27043554 ...  1.17928729 -0.88437203
   1.49352252]
 [-0.67236062 -0.39461922  0.22731236 ...  1.28817102  0.3289682
   0.22239463]
 ...
 [-1.6931999  -0.7576874   0.32935483 ...  0.248912   -0.65651204
  -0.39158494]
 [-0.05157997  0.81428477 -2.17709459 ... -0.81226367  0.23254228
  -0.64323859]
 [ 0.69680558  0.43155574 -0.78142703 ... -0.50816762  0.18255346
   0.0493099 ]]
```

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)
```

```
KNeighborsClassifier()
```

```
print(classifier.predict(X_test))
```

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

```
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

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

```
from sklearn.metrics import confusion_matrix, accuracy_score,classification_report
from sklearn .model_selection import cross_val_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[ 99  14]
 [  8 129]]
0.912
```

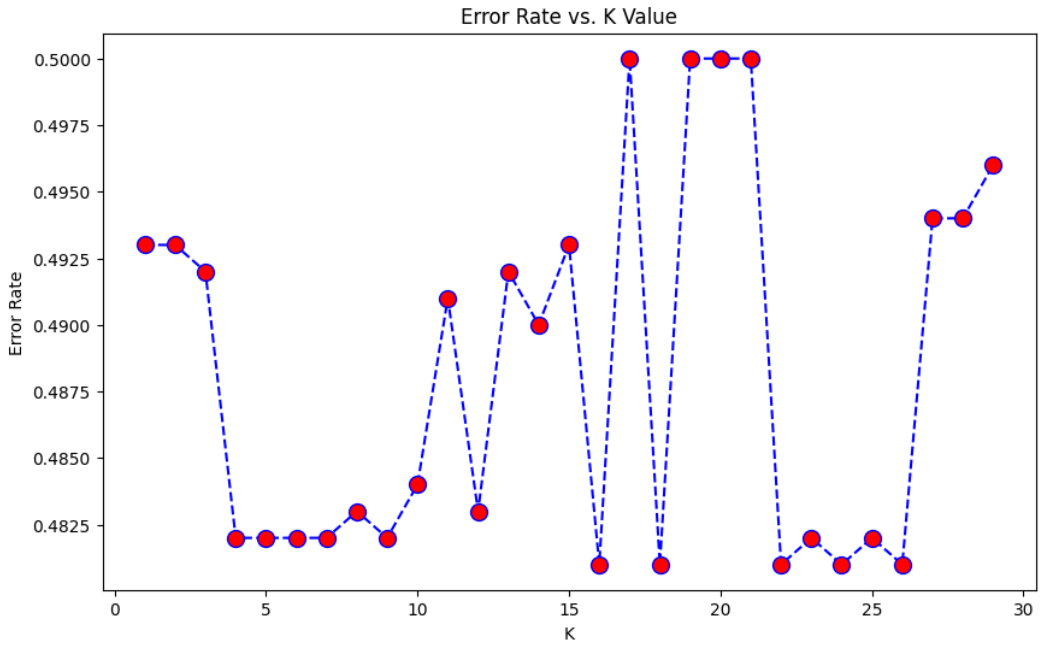
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.93      0.88      0.90       113
           1       0.90      0.94      0.92       137

    accuracy                           0.91       250
   macro avg       0.91      0.91      0.91       250
weighted avg       0.91      0.91      0.91       250
```

```
accuracy_rate=[]
for i in range( 1 , 30):
  knn = KNeighborsClassifier(n_neighbors=i)
  score=cross_val_score(knn, X, dataset[ 'TARGET CLASS' ],cv=5)
  accuracy_rate. append(score.mean())
```

```
plt.figure(figsize=(10, 6))
plt.plot(range(1,30),accuracy_rate,color='blue',linestyle='dashed', marker='o',markerfacecolor='red',markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel ('Error Rate')
```

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




```
knn = KNeighborsClassifier(n_neighbors=23)
knn.fit(X_train,y_train)
pred=knn.predict(X_test)
print('WITH K=23')
print('\n')
print(confusion_matrix(y_test ,y_pred))
print('\n')
print(classification_report(y_test, y_pred))
```

```
WITH K=23


[[ 99  14]
 [  8 129]]


              precision    recall  f1-score   support
```

```
              0       0.93      0.88      0.90       113
              1       0.90      0.94      0.92       137

       accuracy                           0.91       250
      macro avg       0.91      0.91      0.91       250
   weighted avg       0.91      0.91      0.91       250
```

```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train,y_train)
pred=knn.predict(X_test)
print('WITH K=3')
print('\n')
print(confusion_matrix(y_test ,y_pred))
print('\n')
print(classification_report(y_test, y_pred))
```

```
WITH K=3


[[ 99  14]
 [  8 129]]


              precision    recall  f1-score   support

           0       0.93      0.88      0.90       113
           1       0.90      0.94      0.92       137

    accuracy                           0.91       250
   macro avg       0.91      0.91      0.91       250
weighted avg       0.91      0.91      0.91       250
```

✓ 0s completed at 12:56 AM