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

```
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Classified Data",index_col=0)
```

```
df.head()
```

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

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

```
scaler = StandardScaler()
```

```
scaler.fit(df.drop('TARGET CLASS',axis=1))
```

```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

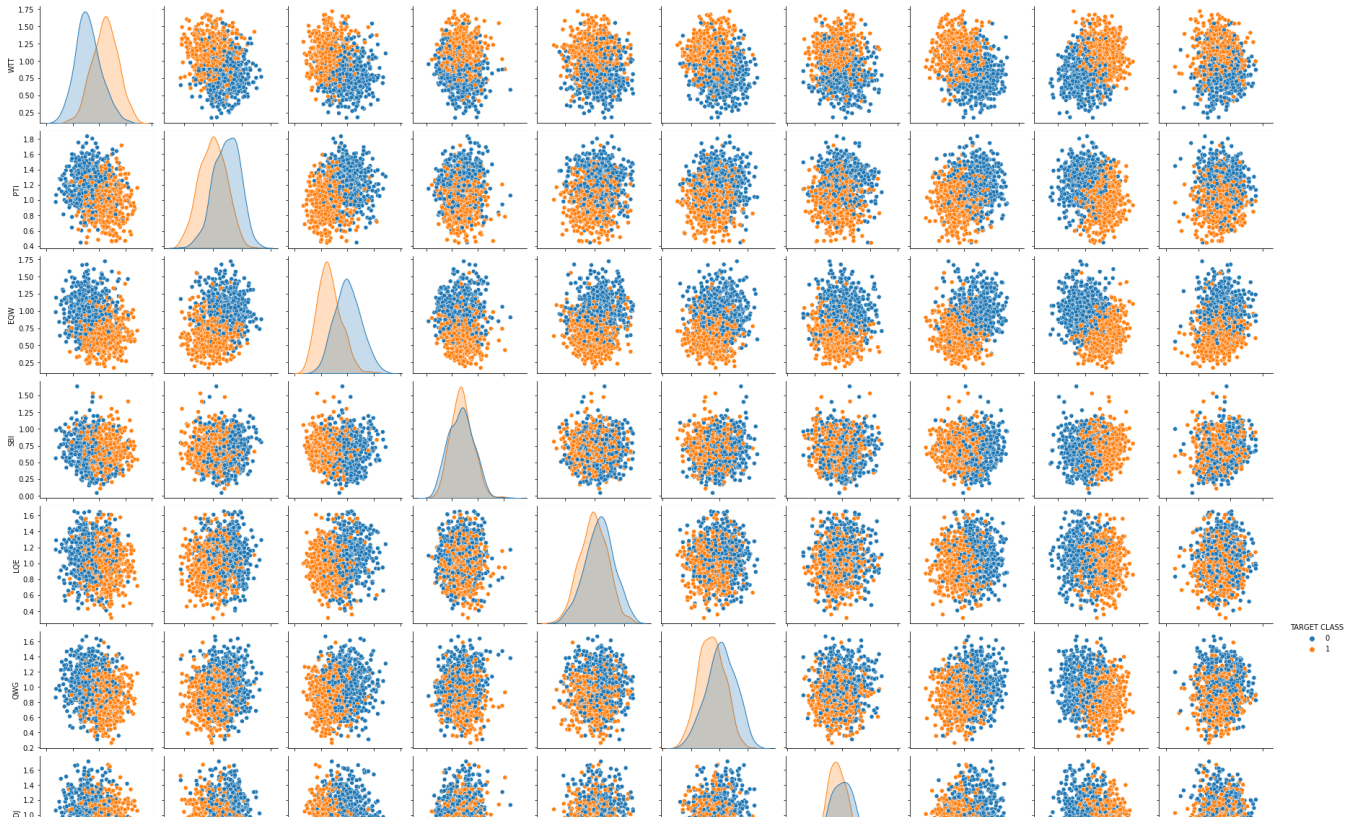
```
scaled_features = scaler.transform(df.drop('TARGET CLASS',axis=1))
```

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

	WTT	PTI	EQW	SBI	LQE	QWG	FDJ	PJF	
0	-0.123542	0.185907	-0.913431	0.319629	-1.033637	-2.308375	-0.798951	-1.482368	-0.9
1	-1.084836	-0.430348	-1.025313	0.625388	-0.444847	-1.152706	-1.129797	-0.202240	-1.8
2	-0.788702	0.339318	0.301511	0.755873	2.031693	-0.870156	2.599818	0.285707	-0.6
3	0.982841	1.060193	-0.621399	0.625299	0.452820	-0.267220	1.750208	1.066491	1.2
4	1.139275	-0.640392	-0.709819	-0.057175	0.822886	-0.936773	0.596782	-1.472352	1.0

```
import seaborn as sns
sns.pairplot(df,hue='TARGET CLASS')
```

```
<seaborn.axisgrid.PairGrid at 0x7fa3581b4990>
```




```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(scaled_features,df['TARGET CLASS'],
 test_size=0.30)
```

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

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

```
knn.fit(X_train,y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='uniform')
```

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

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

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

```
[[140  18]
```

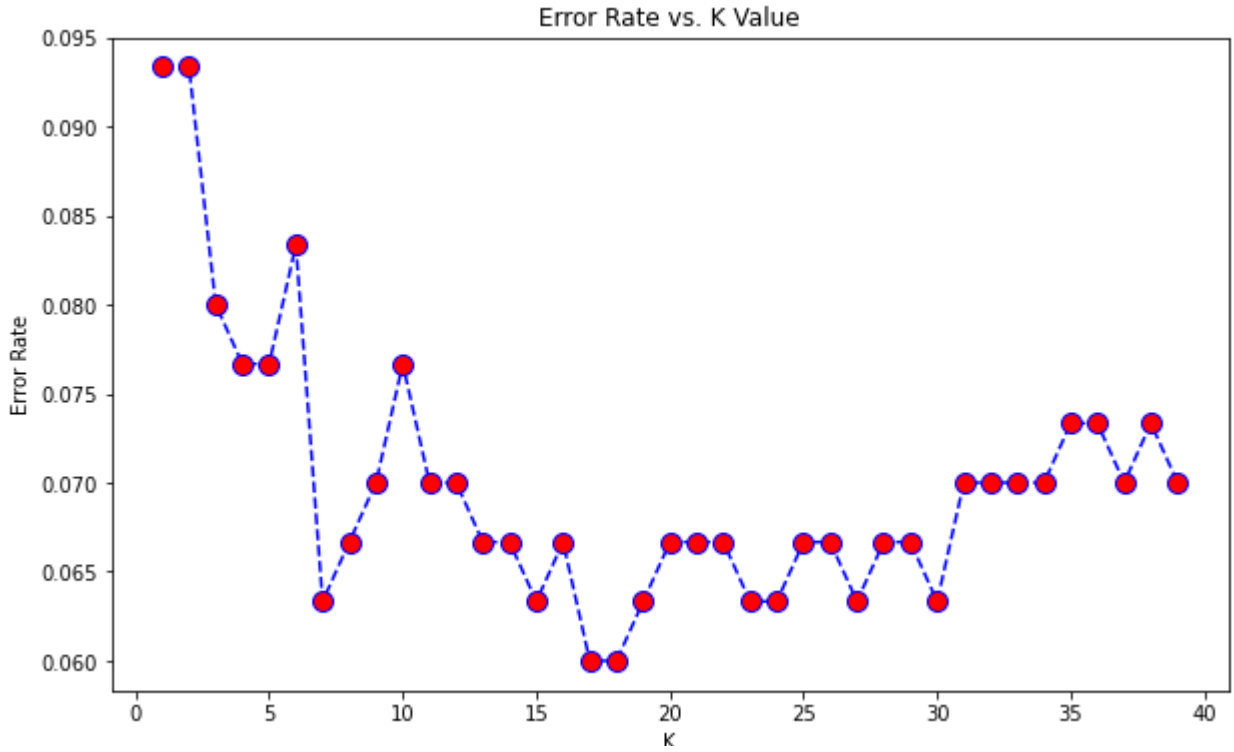
```
       [ 10 132]]
```

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

```
                   precision    recall  f1-score   support

                0       0.93      0.89      0.91       158
                1       0.88      0.93      0.90       142

         accuracy                           0.91       300
        macro avg       0.91      0.91      0.91       300
     weighted avg       0.91      0.91      0.91       300
```

```
accuracy_rate = []
# Will take some time
for i in range(1,40):

 knn = KNeighborsClassifier(n_neighbors=i)
 score=cross_val_score(knn,df_feat,df['TARGET CLASS'],cv=10)
 accuracy_rate.append(score.mean())


error_rate = []
# Will take some time
for i in range(1,40):

 knn = KNeighborsClassifier(n_neighbors=i)
 score=cross_val_score(knn,df_feat,df['TARGET CLASS'],cv=10)
 error_rate.append(1-score.mean())


error_rate = []
# Will take some time
for i in range(1,40):

 knn = KNeighborsClassifier(n_neighbors=i)
 knn.fit(X_train,y_train)
 pred_i = knn.predict(X_test)
 error_rate.append(np.mean(pred_i != y_test))


plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue', linestyle='dashed', marker='o',
 markerfacecolor='red', markersize=10)
# plt.plot(range(1,40),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')
```




```
# FIRST A QUICK COMPARISON TO OUR ORIGINAL K=1
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train,y_train)
pred = knn.predict(X_test)
print('WITH K=1')
print('\n')
print(confusion_matrix(y_test,pred))
print('\n')
print(classification_report(y_test,pred))
```

```
WITH K=1


[[140  18]
 [ 10 132]]


              precision    recall  f1-score   support

           0       0.93      0.89      0.91       158
           1       0.88      0.93      0.90       142

    accuracy                           0.91       300
   macro avg       0.91      0.91      0.91       300
weighted avg       0.91      0.91      0.91       300
```

```
# NOW WITH K=23
knn = KNeighborsClassifier(n_neighbors=37)
knn.fit(X_train,y_train)
pred = knn.predict(X_test)
```

```
print('WITH K=23')
print('\n')
print(confusion_matrix(y_test,pred))
print('\n')
print(classification_report(y_test,pred))
```

```
WITH K=23


[[143  15]
 [  6 136]]


              precision    recall  f1-score   support

           0       0.96      0.91      0.93       158
           1       0.90      0.96      0.93       142

    accuracy                           0.93       300
   macro avg       0.93      0.93      0.93       300
weighted avg       0.93      0.93      0.93       300
```

```
accuracy_rate
```

```
[0.9109999999999999,
 0.909,
 0.9280000000000002,
 0.9339999999999999,
 0.9289999999999999,
 0.929,
 0.9310000000000003,
 0.9340000000000002,
 0.9289999999999999,
 0.9350000000000002,
 0.9329999999999998,
 0.9350000000000002,
 0.937,
 0.9410000000000001,
 0.9390000000000001,
 0.9390000000000001,
 0.9349999999999999,
 0.9360000000000002,
 0.9360000000000002,
 0.9360000000000002,
 0.9339999999999999,
 0.9359999999999999,
 0.93,
 0.933,
 0.9349999999999999,
 0.937,
 0.9380000000000001,
 0.938,
 0.937,
```

```
0.9390000000000001,
0.9360000000000002,
0.943,
0.937,
0.9390000000000001,
0.9390000000000001,
0.9400000000000001,
0.9369999999999999,
0.943,
0.9410000000000001]
```

✓ 0s completed at 15:02