Machine Learning

Lab 7

In [1]:

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

In [2]:

```
df = pd.read_csv("Classified Data", index_col=0)
df.head()
```

Out[2]:

	WTT	PTI	EQW	SBI	LQE	QWG	FDJ	PJF	HQE	NXJ	TARGET CLASS
0	0.913917	1.162073	0.567946	0.755464	0.780862	0.352608	0.759697	0.643798	0.879422	1.231409	1
1	0.635632	1.003722	0.535342	0.825645	0.924109	0.648450	0.675334	1.013546	0.621552	1.492702	0
2	0.721360	1.201493	0.921990	0.855595	1.526629	0.720781	1.626351	1.154483	0.957877	1.285597	0
3	1.234204	1.386726	0.653046	0.825624	1.142504	0.875128	1.409708	1.380003	1.522692	1.153093	1
4	1.279491	0.949750	0.627280	0.668976	1.232537	0.703727	1.115596	0.646691	1.463812	1.419167	1

Scaling the data

In [3]:
```
from sklearn.preprocessing import StandardScaler

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

In [4]:
```
df_feat = pd.DataFrame(scaled_features, columns=df.columns[:-1])
df_feat.head()
```

Out[4]:

	WTT	PTI	EQW	SBI	LQE	QWG	FDJ	PJF	HQE	NXJ
0	-0.123542	0.185907	-0.913431	0.319629	-1.033637	-2.308375	-0.798951	-1.482368	-0.949719	-0.643314
1	-1.084836	-0.430348	-1.025313	0.625388	-0.444847	-1.152706	-1.129797	-0.202240	-1.828051	0.636759
2	-0.788702	0.339318	0.301511	0.755873	2.031693	-0.870156	2.599818	0.285707	-0.682494	-0.377850
3	0.982841	1.060193	-0.621399	0.625299	0.452820	-0.267220	1.750208	1.066491	1.241325	-1.026987
4	1.139275	-0.640392	-0.709819	-0.057175	0.822886	-0.936773	0.596782	-1.472352	1.040772	0.276510

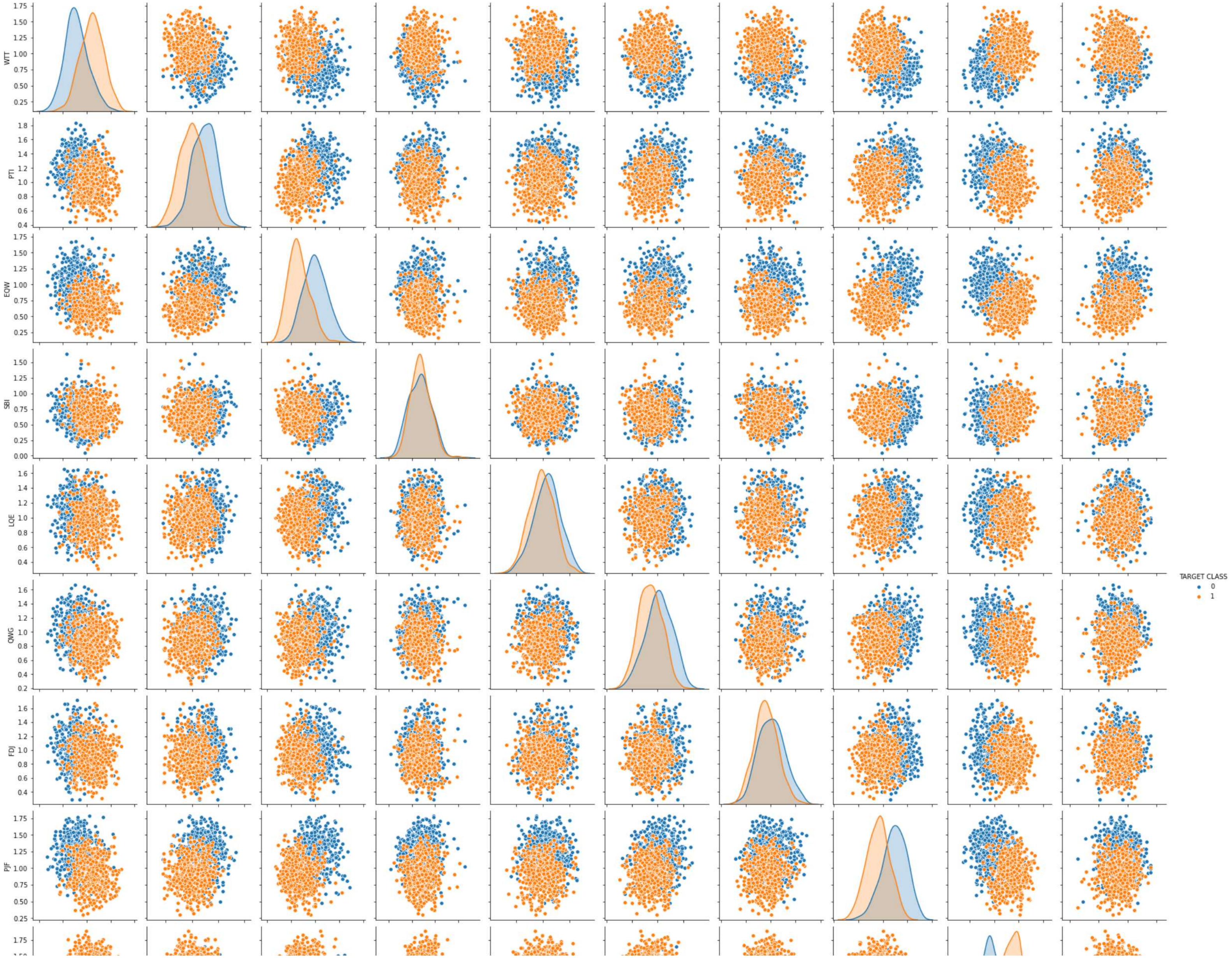
Visualize

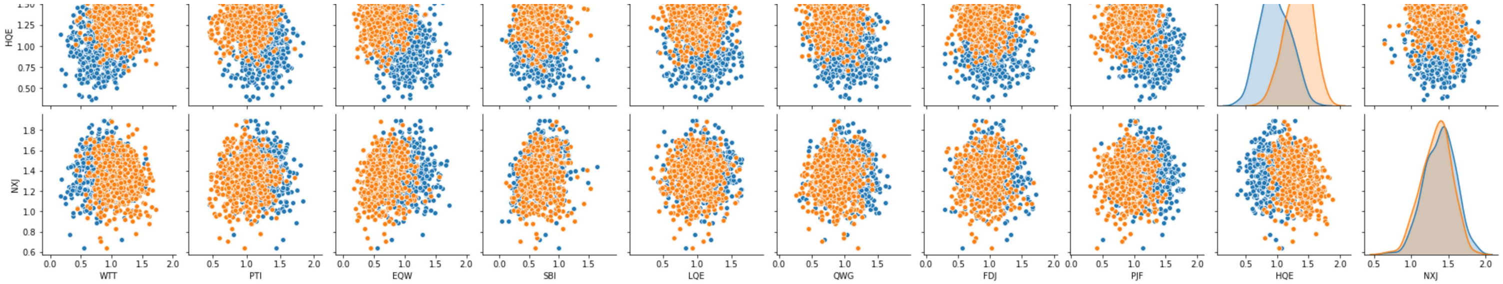
In [5]:

```
import seaborn as sns

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

Out[5]:
```
<seaborn.axisgrid.PairGrid at 0x1e612eca9a0>
```


WTT
1.75
1.50
1.25
1.00
0.75
0.50
0.25
PTI
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
EQW
1.75
1.50
1.25
1.00
0.75
0.50
0.25
SBI
1.50
1.25
1.00
0.75
0.50
0.25
0.00
LOE
1.6
1.4
1.2
1.0
0.8
0.6
0.4
QWG
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
FDJ
1.6
1.4
1.2
1.0
0.8
0.6
0.4
PJF
1.75
1.50
1.25
1.00
0.75
0.50
0.25
1.75
TARGET CLASS
0
1



Here, data points are overlapped on eachother. So, We can't use Linear Regression.

In [6]:
```
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=0.3)
```

Check KNN with neighbour 1

In [7]:
```
from sklearn.neighbors import KNeighborsClassifier

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

In [8]:
```
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import cross_val_score

confusion_matrix(y_test, y_pred)
```

Out[8]:
```
array([[135,  19],
       [ 12, 134]], dtype=int64)
```

Choosing K value

In [9]:

```
accuracy_rate = []

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())
```

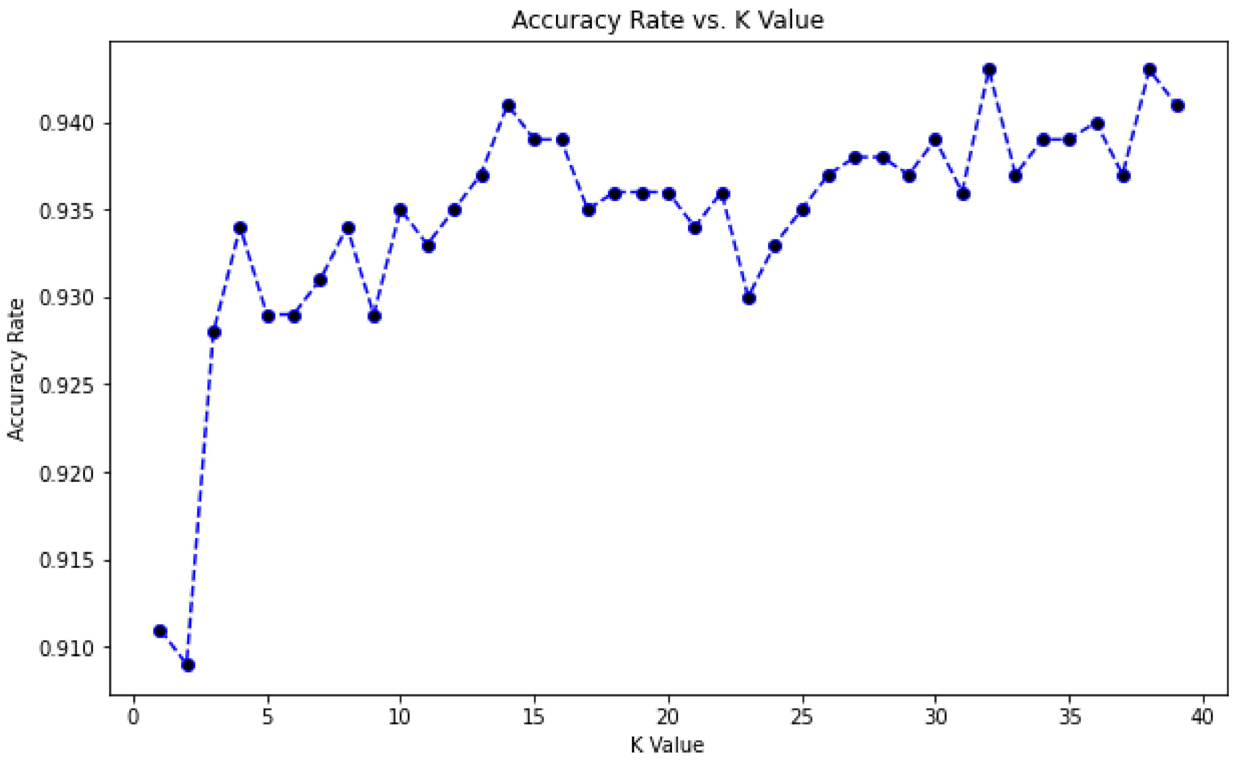
In [10]:

```
error_rate = []

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())
```

In [11]:
```
plt.figure(figsize=(10,6))
plt.plot(range(1,40), accuracy_rate, markerfacecolor='black', marker='o', color='blue', linestyle='dashed')
plt.title('Accuracy Rate vs. K Value')
plt.xlabel('K Value')
plt.ylabel('Accuracy Rate')
```

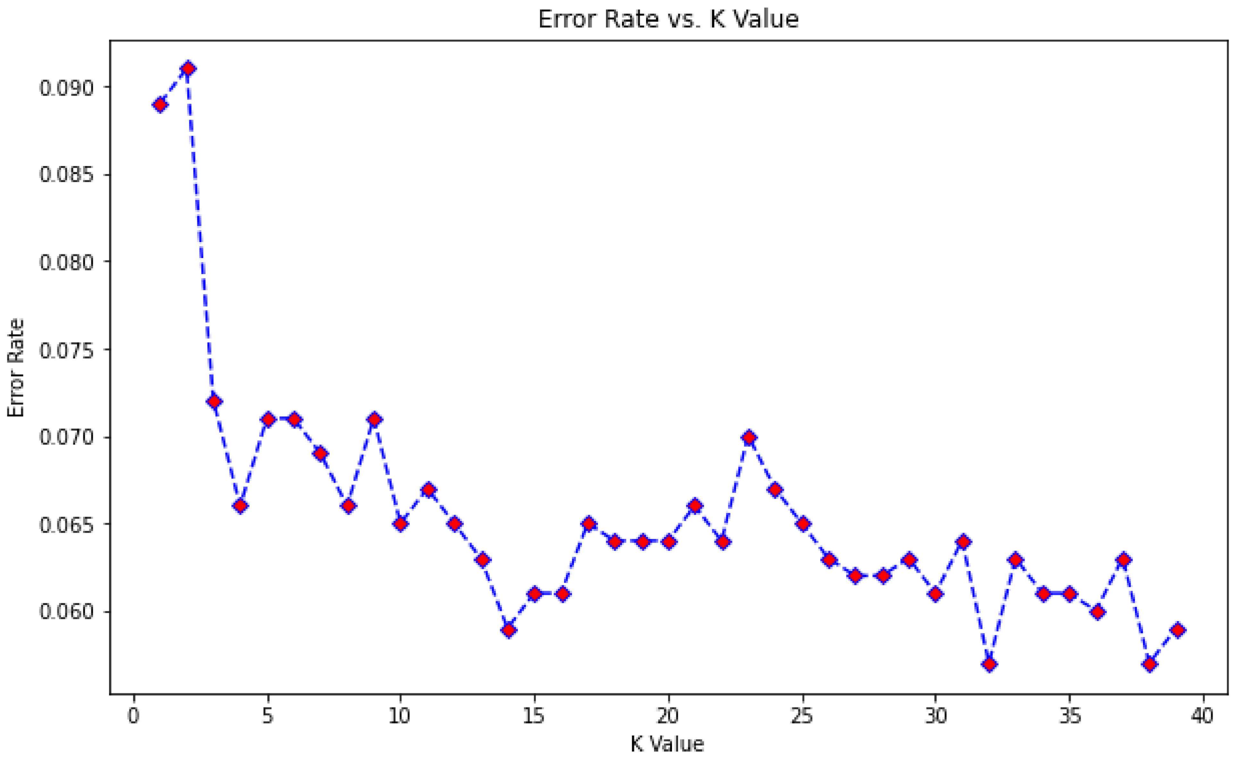
Out[11]:
```
Text(0, 0.5, 'Accuracy Rate')
```



Plotting Accuracy rate and Error rate

In [12]:
```
plt.figure(figsize=(10,6))
plt.plot(range(1,40), error_rate, color='blue', marker='D', markerfacecolor='red', linestyle='dashed')
plt.title('Error Rate vs. K Value')
plt.xlabel('K Value')
plt.ylabel('Error Rate')
```

Out[12]:
```
Text(0, 0.5, 'Error Rate')
```



By observing above plots, the accuracy increases after the K=24.

In [13]:

```
knn = KNeighborsClassifier(n_neighbors=24)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

In [14]:

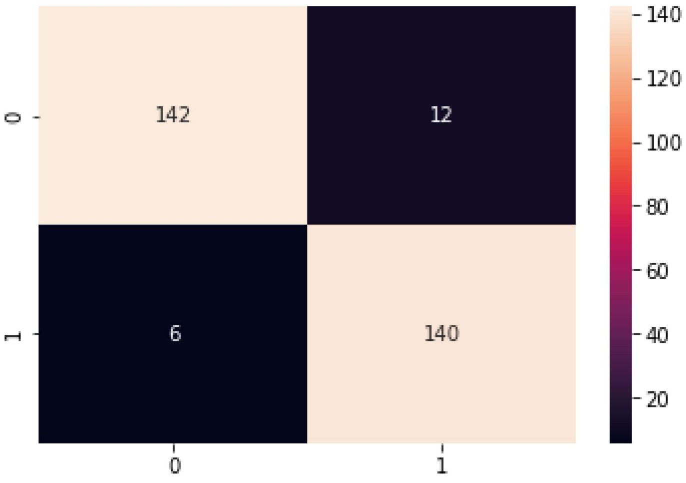
```
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[14]:

```
array([[142,  12],
       [  6, 140]], dtype=int64)
```

In [15]:

```
sns.heatmap(cm, annot=True, fmt='d')
plt.show()
```



In [16]:

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

```
              precision    recall  f1-score   support

           0       0.96      0.92      0.94       154
           1       0.92      0.96      0.94       146

    accuracy                           0.94       300
   macro avg       0.94      0.94      0.94       300
weighted avg       0.94      0.94      0.94       300
```

In [ ]: