In [1]:

```
import pandas as pd
import numpy as np
```

In [3]:

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [4]:

```
df=pd.read_csv('KNN_Project_Data')
```

In [5]:

```
df.head()
```

Out[5]:

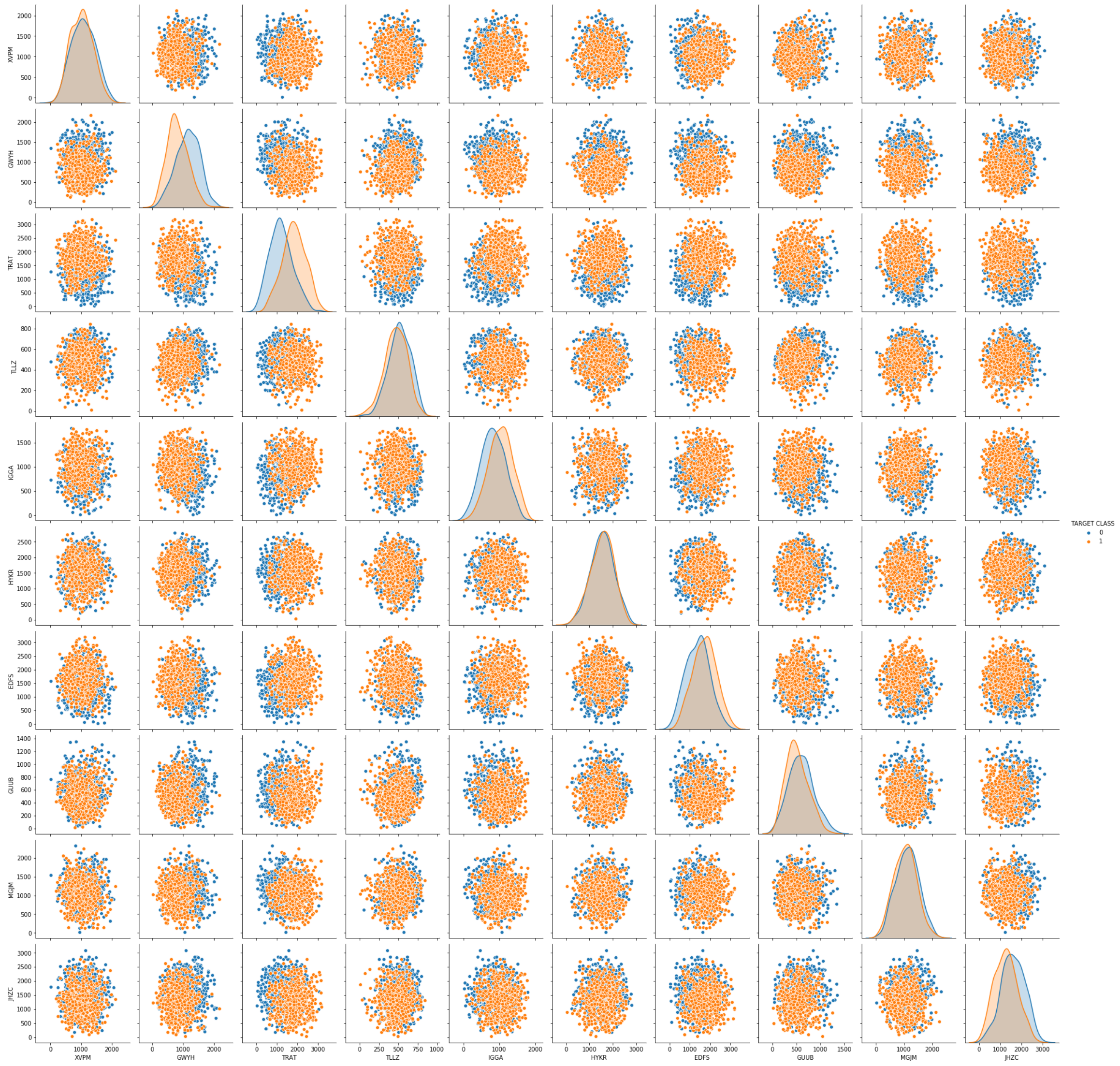
	XVPM	GWYH	TRAT	TLLZ	IGGA	HYKR	EDFS
0	1636.670614	817.988525	2565.995189	358.347163	550.417491	1618.870897	2147.641254
1	1013.402760	577.587332	2644.141273	280.428203	1161.873391	2084.107872	853.404981
2	1300.035501	820.518697	2025.854469	525.562292	922.206261	2552.355407	818.676686
3	1059.347542	1066.866418	612.000041	480.827789	419.467495	685.666983	852.867810
4	1018.340526	1313.679056	950.622661	724.742174	843.065903	1370.554164	905.469453

In [7]:

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

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x1ee9a9ed948>
```



In [8]:

```
#Time to standardize our scale
from sklearn.preprocessing import StandardScaler
```

In [9]:

```
scaler=StandardScaler()
```

In [10]:

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

Out[10]:

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

In [11]:

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

In [13]:

```
df_scaled=pd.DataFrame(scaled_feat,columns=df.columns[:-1])
```

In [14]:

```
df_scaled.head()
```

Out[14]:

	XVPM	GWYH	TRAT	TLLZ	IGGA	HYKR	EDFS	GUUB	I
0	1.568522	-0.443435	1.619808	-0.958255	-1.128481	0.138336	0.980493	-0.932794	1.0
1	-0.112376	-1.056574	1.741918	-1.504220	0.640009	1.081552	-1.182663	-0.461864	0.2
2	0.660647	-0.436981	0.775793	0.213394	-0.053171	2.030872	-1.240707	1.149298	2.1
3	0.011533	0.191324	-1.433473	-0.100053	-1.507223	-1.753632	-1.183561	-0.888557	0.1
4	-0.099059	0.820815	-0.904346	1.609015	-0.282065	-0.365099	-1.095644	0.391419	-1.3

In [15]:

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

In [18]:

```
X=df_scaled
y=df['TARGET CLASS']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=4
2)
```

In [19]:

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

In [20]:

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

In [21]:

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

Out[21]:

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

In [22]:

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

In [23]:

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

In [24]:

```
pred
```

Out[24]:

```
array([0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1,
       1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1,
       1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0,
       1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1,
       1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1,
       0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0,
       1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1], dtype=int64)
```

In [25]:

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

```
[[110  36]
 [ 47 107]]
```

In [26]:

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

```
              precision    recall  f1-score   support

           0       0.70      0.75      0.73       146
           1       0.75      0.69      0.72       154

    accuracy                           0.72       300
   macro avg       0.72      0.72      0.72       300
weighted avg       0.73      0.72      0.72       300
```

In [27]:

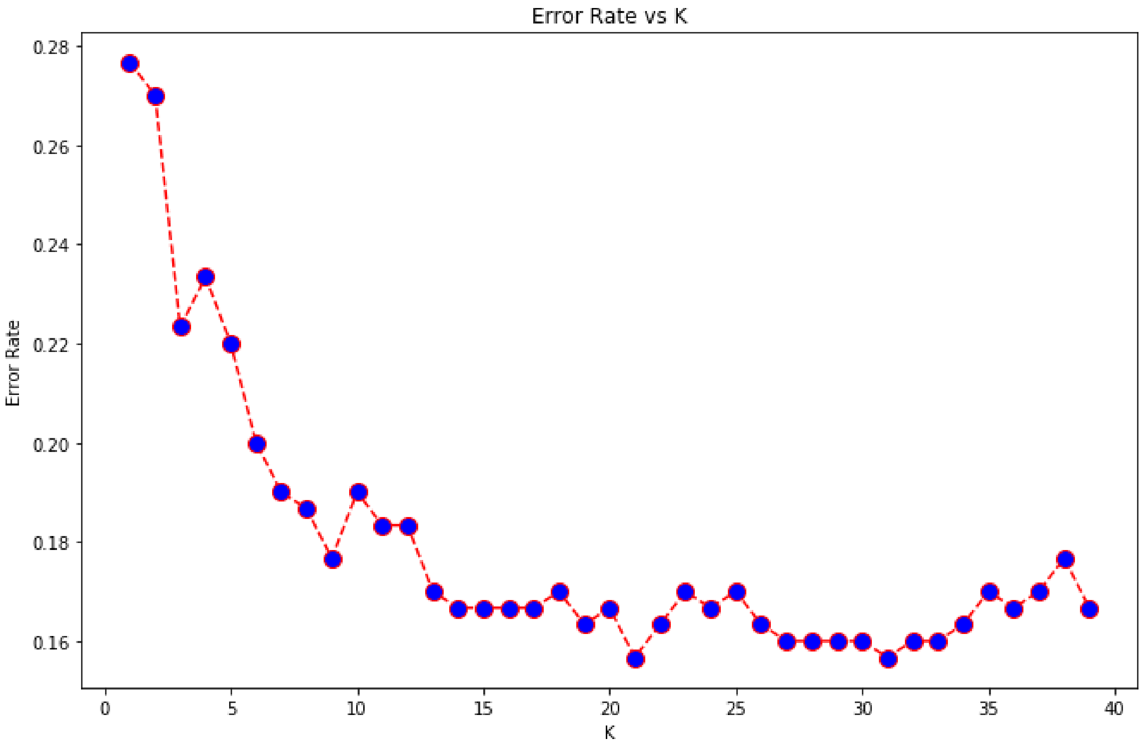
```
#WE can select a better value of k for more accuracy ELBOW METHOD
error_rate=[]
for i in range(1,40):
    knn=KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i=knn.predict(X_test)
    error_rate.append(np.mean(pred_i!=y_test))
```

In [28]:

```
plt.figure(figsize=(11,7))
plt.plot(range(1,40),error_rate,color='red',linestyle='--',marker='o',markerfacecolor=
'blue',markersize=10)
plt.title('Error Rate vs K')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

Out[28]:

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



In [31]:

```
#lets choose  a k with less error say 21
knn=KNeighborsClassifier(n_neighbors=21)
knn.fit(X_train,y_train)
predics=knn.predict(X_test)
print(confusion_matrix(y_test,predics))
print('\n')
print(classification_report(y_test,predics))
```

```
[[125  21]
 [ 26 128]]


              precision    recall  f1-score   support

           0       0.83      0.86      0.84       146
           1       0.86      0.83      0.84       154

    accuracy                           0.84       300
   macro avg       0.84      0.84      0.84       300
weighted avg       0.84      0.84      0.84       300
```

In [33]:

```
 #BETTER we are up to 84% from 73%
```

In [ ]:

```

```