K Nearest Neighbors

Import libraries

In [3]:

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

Load Data set

In [5]:

```
Data_org = pd.read_csv('Data/KNN_Project_Data')
```

In [6]:

```
Data_org.head()
```

Out[6]:

	XVPM	GWYH	TRAT	TLLZ	IGGA	HYKR	EDFS	
0	1636.670614	817.988525	2565.995189	358.347163	550.417491	1618.870897	2147.641254	3
1	1013.402760	577.587332	2644.141273	280.428203	1161.873391	2084.107872	853.404981	4
2	1300.035501	820.518697	2025.854469	525.562292	922.206261	2552.355407	818.676686	8
3	1059.347542	1066.866418	612.000041	480.827789	419.467495	685.666983	852.867810	3
4	1018.340526	1313.679056	950.622661	724.742174	843.065903	1370.554164	905.469453	6

In [7]:

```
print(Data_org.shape)
```

```
(1000, 11)
```

In [9]:

```
print(Data_org.isnull().sum().any())
```

```
False
```

In [14]:

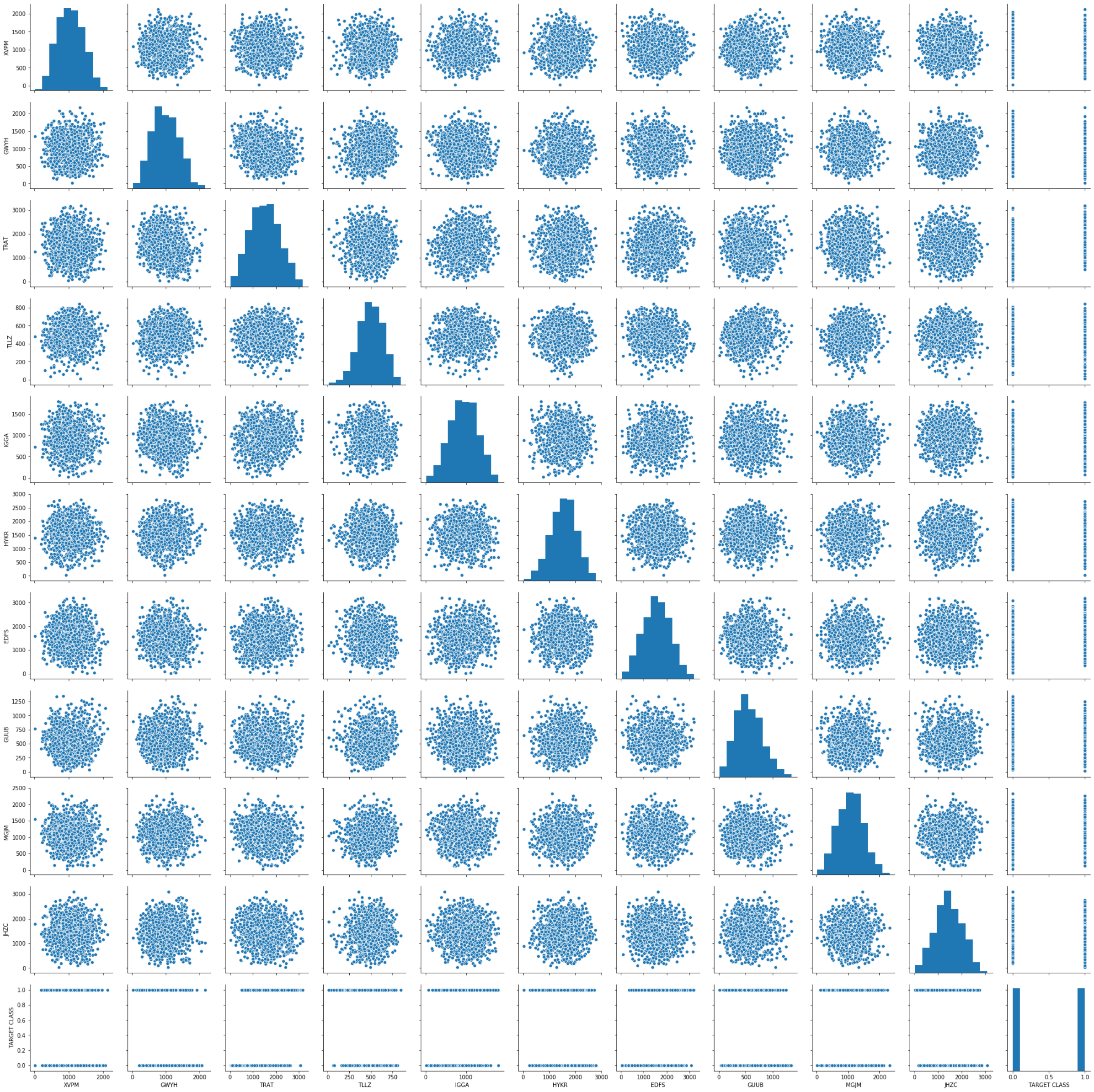
```
(Data_org.dtypes == 'object').any()
```

Out[14]:

```
False
```

In [15]:
```
sns.pairplot(Data_org)
```

Out[15]:
```
<seaborn.axisgrid.PairGrid at 0x1af7a0766d8>
```



Make copy of original data

In [18]:
```
Data_org.columns
```

Out[18]:
```
Index(['XVPM', 'GWYH', 'TRAT', 'TLLZ', 'IGGA', 'HYKR', 'EDFS', 'GUUB', 'MGJ
M',
       'JHZC', 'TARGET CLASS'],
      dtype='object')
```

In [19]:
```
X = Data_org.drop(columns=['TARGET CLASS'], axis=1)
y = Data_org['TARGET CLASS']
```

Standardize the Variables¶

Because the KNN classifier predicts the class of a given test observation by identifying the observations that are nearest to it, the scale of the variables matters. Any variables that are on a large scale will have a much larger effect on the distance between the observations, and hence on the KNN classifier, than variables that are on a small scale.

In [16]:
```
from sklearn.preprocessing import StandardScaler
```

In [21]:
```
scaler = StandardScaler()
scaler.fit(X)
```

Out[21]:
```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [23]:
```
feature_scaled = scaler.transform(X)
type(feature_scaled)
```

Out[23]:
```
numpy.ndarray
```

In [27]:
```
X_scaled = pd.DataFrame(feature_scaled, columns=X.columns)
X_scaled.head()
```

Out[27]:

	XVPM	GWYH	TRAT	TLLZ	IGGA	HYKR	EDFS	GUUB	MGJ
0	1.568522	-0.443435	1.619808	-0.958255	-1.128481	0.138336	0.980493	-0.932794	1.0083
1	-0.112376	-1.056574	1.741918	-1.504220	0.640009	1.081552	-1.182663	-0.461864	0.2583
2	0.660647	-0.436981	0.775793	0.213394	-0.053171	2.030872	-1.240707	1.149298	2.1847
3	0.011533	0.191324	-1.433473	-0.100053	-1.507223	-1.753632	-1.183561	-0.888557	0.1623
4	-0.099059	0.820815	-0.904346	1.609015	-0.282065	-0.365099	-1.095644	0.391419	-1.3656

KNN Model

In [91]:
```
from sklearn.model_selection import train_test_split
```

In [92]:
```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3
0, random_state=100)
```

In [93]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [94]:

```
def getKNN(n_neighbors):
    model = KNeighborsClassifier(n_neighbors = n_neighbors)
    model.fit(X_train, y_train)
    pred = model.predict(X_test)
    err = np.mean(y_test != pred)
    acc_train = model.score(X_train, y_train)
    acc_test =  model.score(X_test, y_test)
    return err, acc_train, acc_test
```

In [ ]:

Choosing a K Value

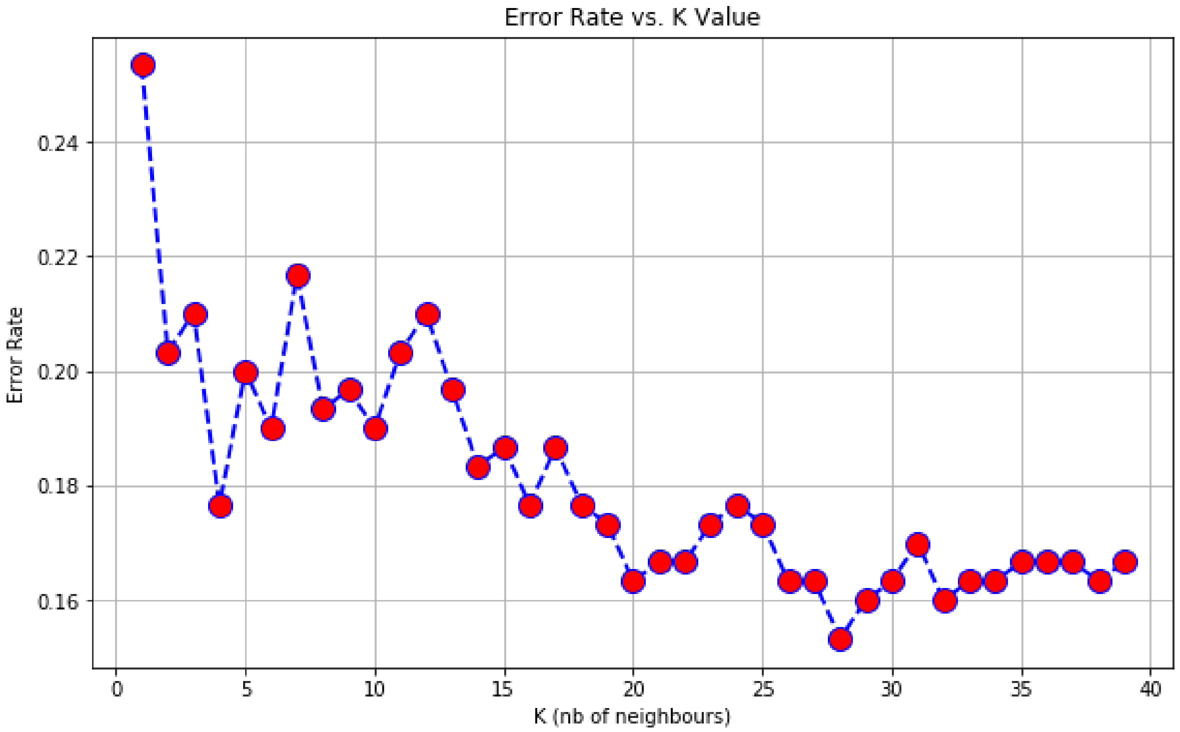
using elbow method to pick a good K Value!

In [95]:

```
train_acc = []
test_acc = []
Err = []
k_val = range(1,40)

for k in  k_val:
    res = getKNN(k)
    Err.append(res[0])
    train_acc.append(res[1])
    test_acc.append(res[2])
```

In [96]:
```
plt.figure(figsize=(10,6))
plt.plot(k_val, Err, color='blue', marker='o', linestyle='dashed',linewidth=2,
markersize=12, markerfacecolor='red')
plt.xlabel('K (nb of neighbours)')
plt.ylabel('Error Rate')
plt.title('Error Rate vs. K Value')
plt.grid(True)
```

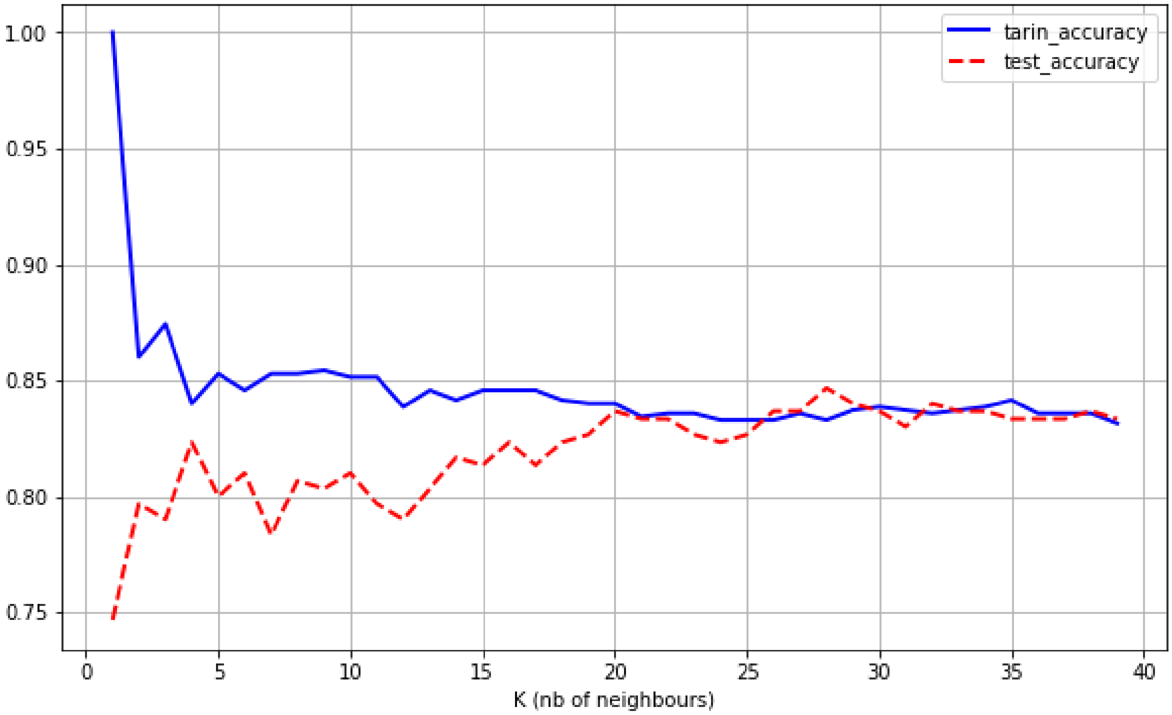


In [97]:

```
plt.figure(figsize=(10,6))
plt.plot(k_val, train_acc, color = 'blue', linewidth=2)
plt.plot(k_val, test_acc, color = 'red', linewidth=2, linestyle ='dashed')

plt.xlabel('K (nb of neighbours)')
plt.legend(['tarin_accuracy', 'test_accuracy'])

plt.grid(True)
```



Best K value

In [98]:

```
best_k = k_val[np.argmin(Err)]
best_k
```

Out[98]:

```
28
```

In [99]:

```
np.min(Err)
```

Out[99]:

```
0.15333333333333332
```

model with K= 20, K=28

In [105]:

```
k1=20
k2=28

model1 = KNeighborsClassifier(n_neighbors=k1)
model1.fit(X_train, y_train)
pred1= model1.predict(X_test)

model2 = KNeighborsClassifier(n_neighbors=k2)
model2.fit(X_train, y_train)
pred2 = model2.predict(X_test)
```

In [103]:

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

In [111]:

```
print('K=%d' %k1)
print(classification_report(y_test, pred1))
print(confusion_matrix(y_test, pred1))
print('\n accuracy:%2.2f' %accuracy_score(y_test, pred1))
```

```
K=20
              precision    recall  f1-score   support

           0       0.90      0.80      0.84       167
           1       0.78      0.89      0.83       133

   micro avg       0.84      0.84      0.84       300
   macro avg       0.84      0.84      0.84       300
weighted avg       0.84      0.84      0.84       300

[[133  34]
 [ 15 118]]

 accuracy:0.84
```

In [112]:

```
print('K=%d' %k2)
print(classification_report(y_test, pred2))
print(confusion_matrix(y_test, pred2))
print('\n accuracy:%2.2f' %accuracy_score(y_test, pred2))
```

```
K=28
              precision    recall  f1-score   support

           0       0.91      0.81      0.85       167
           1       0.79      0.89      0.84       133

   micro avg       0.85      0.85      0.85       300
   macro avg       0.85      0.85      0.85       300
weighted avg       0.85      0.85      0.85       300

[[135  32]
 [ 14 119]]

 accuracy:0.85
```

In [ ]: