K Nearest Neighbors Project

Import Libraries

In [27]:

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

Get the Data

In [5]:

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

In [6]:

```
df.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 [23]:

```

```

Out[23]:

	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

EDA

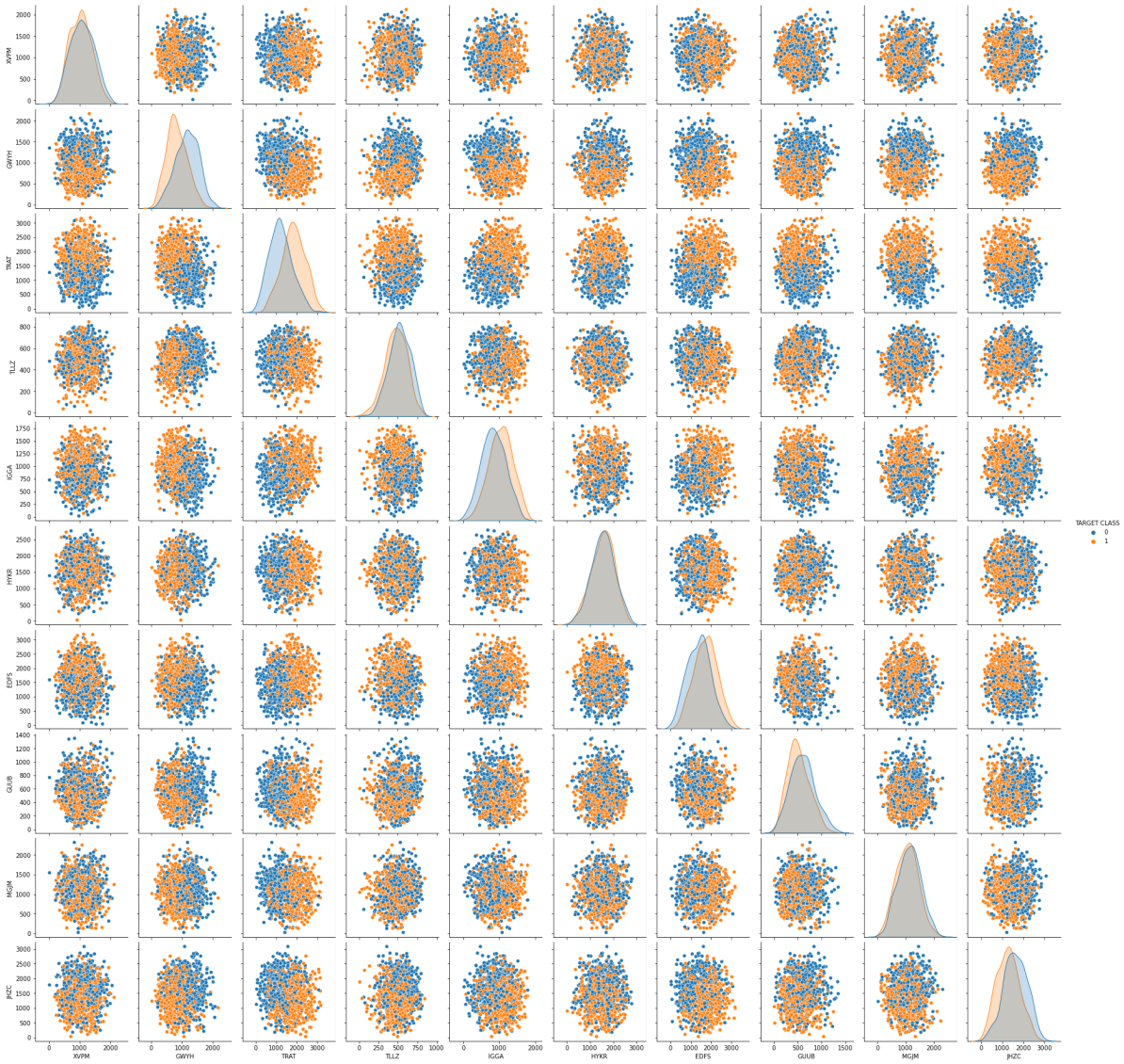
Since this data is artificial, we'll just do a large pairplot with seaborn.

In [8]:

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

Out[8]:

```
<seaborn.axisgrid.PairGrid at 0x7fd5bc896190>
```

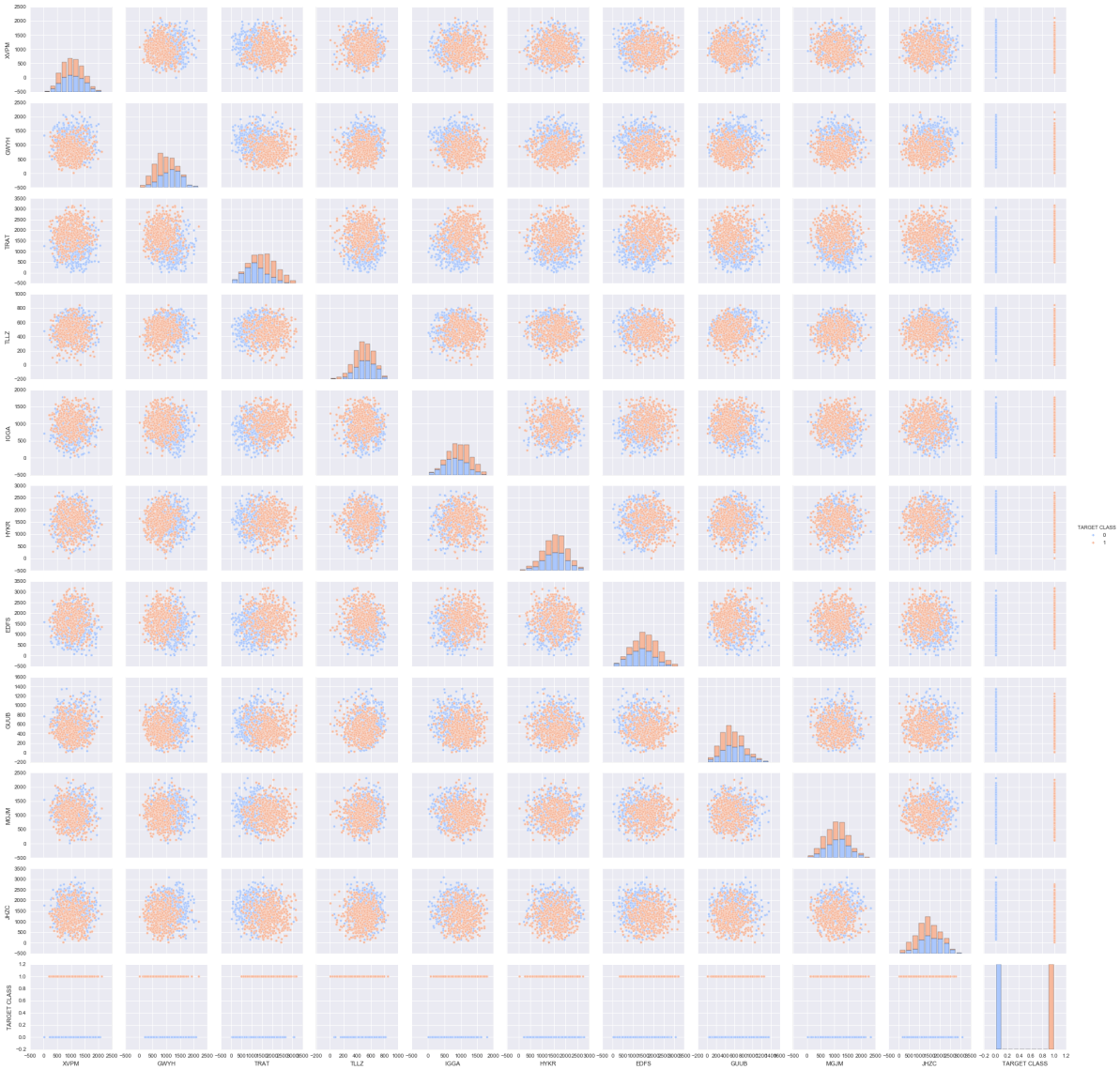


In [4]:

```
```

Out[4]:

```
<seaborn.axisgrid.PairGrid at 0x1197505f8>
```



Standardize the Variables

In [9]:

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

In [10]:

```
scaler = StandardScaler()
```

In [12]:

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

Out[12]:

```
StandardScaler()
```

In [13]:

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

In [16]:

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

Out[16]:

	XVPM	GWYH	TRAT	TLLZ	IGGA	HYKR	EDFS	GUUB	MG.
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

In [9]:

```

```

Out[9]:

	XVPM	GWYH	TRAT	TLLZ	IGGA	HYKR	EDFS	GUUB	MG.
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

Train Test Split

Use train_test_split to split your data into a training set and a testing set.

In [25]:

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

In [44]:

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

Using KNN

Import KNeighborsClassifier from scikit learn.

In [36]:

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

Create a KNN model instance with n_neighbors=1

In [59]:

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

Fit this KNN model to the training data.

In [49]:

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

Out[49]:

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

In [14]:

```

```

Out[14]:

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

Predictions and Evaluations

In [50]:

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

** Create a confusion matrix and classification report.**

In [51]:

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

In [52]:

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

```
[[114  41]
 [ 41 104]]
```

In [17]:

```

```

```
[[112  40]
 [ 34 114]]
```

In [57]:

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

```
              precision    recall  f1-score   support

           0       0.74      0.74      0.74       155
           1       0.72      0.72      0.72       145

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

In [18]:

```

```

```
             precision    recall  f1-score   support

          0       0.77      0.74      0.75       152
          1       0.74      0.77      0.75       148

avg / total       0.75      0.75      0.75       300
```

Choosing a K Value

In [72]:

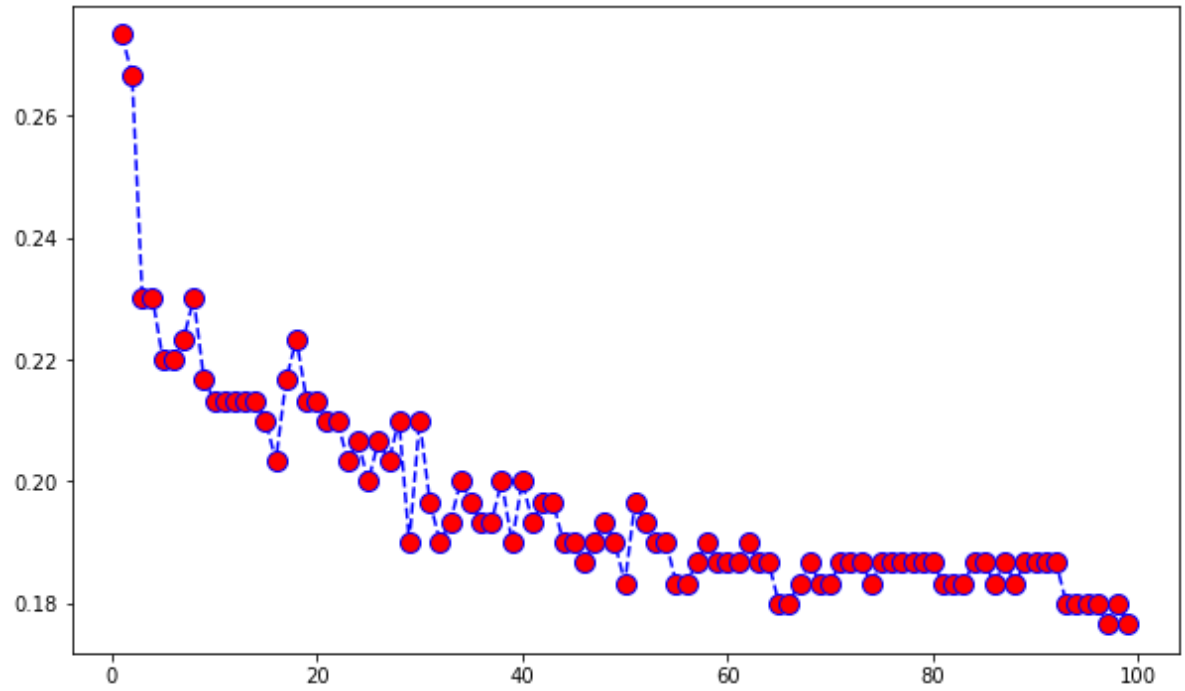
```
error_rate = []
for i in range(1,100):
    knn=KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i=knn.predict(X_test)
    error_rate.append(np.mean(pred_i!=y_test))
```

In [73]:

```
plt.figure(figsize=(10,6))
plt.plot(range(1,1),error_rate,color='blue',linestyle='--',marker='o',markerfacecolc
```

Out[73]:

```
[<matplotlib.lines.Line2D at 0x7fd5a3856760>]
```



In [20]:

Out[20]:

```
<matplotlib.text.Text at 0x11cbdb710>
```