K-Nearest Neighbour

Importing Necessary Packages

In [94]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score , classification_report , confusion_matrix
import matplotlib.pyplot as plt
```

In [55]:
```
data = pd.read_csv('KNN_Project_Data', sep = ',')
```

Exploratory Data Analysis

In [57]:
```
data.head()
```

Out[57]:

	XVPM	GWYH	TRAT	TLLZ	IGGA	HYKR	EDFS	GUUB	MGJM	
0	1636.670614	817.988525	2565.995189	358.347163	550.417491	1618.870897	2147.641254	330.727893	1494.878631	
1	1013.402760	577.587332	2644.141273	280.428203	1161.873391	2084.107872	853.404981	447.157619	1193.032521	
2	1300.035501	820.518697	2025.854469	525.562292	922.206261	2552.355407	818.676686	845.491492	1968.367513	1
3	1059.347542	1066.866418	612.000041	480.827789	419.467495	685.666983	852.867810	341.664784	1154.391368	1
4	1018.340526	1313.679056	950.622661	724.742174	843.065903	1370.554164	905.469453	658.118202	539.459350	1

In [58]:
```
data.dtypes
```

Out[58]:
```
XVPM            float64
GWYH            float64
TRAT            float64
TLLZ            float64
IGGA            float64
HYKR            float64
EDFS            float64
GUUB            float64
MGJM            float64
JHZC            float64
TARGET CLASS      int64
dtype: object
```

In [80]:
```
print(len(data['TARGET CLASS'] == 0))
print(len(data['TARGET CLASS'] == 1))
```

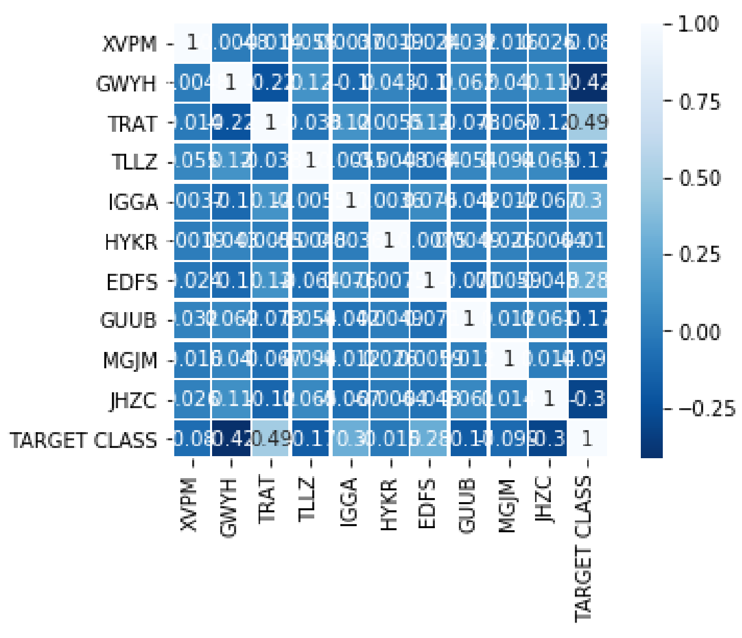
```
1000
1000
```

In [59]:
```
data.isnull().sum()
```

Out[59]:
```
XVPM            0
GWYH            0
TRAT            0
TLLZ            0
IGGA            0
HYKR            0
EDFS            0
GUUB            0
MGJM            0
JHZC            0
TARGET CLASS    0
dtype: int64
```

In [62]:
```
corr = data.corr()
sns.heatmap (corr , annot = True , square = True , linewidths = 0.5 , cmap = 'Blues_r')
```

Out[62]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1f9a42890f0>
```



Feature Scaling

In [64]:
```
scaler = StandardScaler()
```

In [67]:
```
scaled = scaler.fit_transform(data.drop('TARGET CLASS', axis = 1))
```

In [68]:
```
data_feat = pd.DataFrame(scaled , columns = data.columns[:-1])
```

In [70]:
```
data_feat.head()
```

Out[70]:

	XVPM	GWYH	TRAT	TLLZ	IGGA	HYKR	EDFS	GUUB	MGJM	JHZC
0	1.568522	-0.443435	1.619808	-0.958255	-1.128481	0.138336	0.980493	-0.932794	1.008313	-1.069627
1	-0.112376	-1.056574	1.741918	-1.504220	0.640009	1.081552	-1.182663	-0.461864	0.258321	-1.041546
2	0.660647	-0.436981	0.775793	0.213394	-0.053171	2.030872	-1.240707	1.149298	2.184784	0.342811
3	0.011533	0.191324	-1.433473	-0.100053	-1.507223	-1.753632	-1.183561	-0.888557	0.162310	-0.002793
4	-0.099059	0.820815	-0.904346	1.609015	-0.282065	-0.365099	-1.095644	0.391419	-1.365603	0.787762

Train Test Split

In [72]:
```
x_train, x_test , y_train , y_test = train_test_split (data_feat ,  data['TARGET CLASS'] , test_size
= 0.3)
```

Implementing KNN

In [90]:
```
knn = KNeighborsClassifier (n_neighbors = 1)
```

In [91]:
```
knn.fit(x_train , y_train)
```

Out[91]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='uniform')
```

In [81]:
```
pred = knn.predict(x_test)
```

Classification Report

In [85]:
```
confusion_matrix(pred, y_test)
```

Out[85]:
```
array([[110,  47],
       [ 40, 103]], dtype=int64)
```

In [86]:
```
classification_report(pred, y_test)
```

Out[86]:
```
'              precision    recall  f1-score   support\n\n           0       0.73      0.70
0.72       157\n           1       0.69      0.72      0.70       143\n\n    accuracy
0.71       300\n   macro avg       0.71      0.71      0.71       300\nweighted avg       0.71
0.71      0.71       300\n'
```

In [87]:
```
accuracy_score (pred, y_test)
```

Out[87]:
```
0.71
```

Choosing a K Value

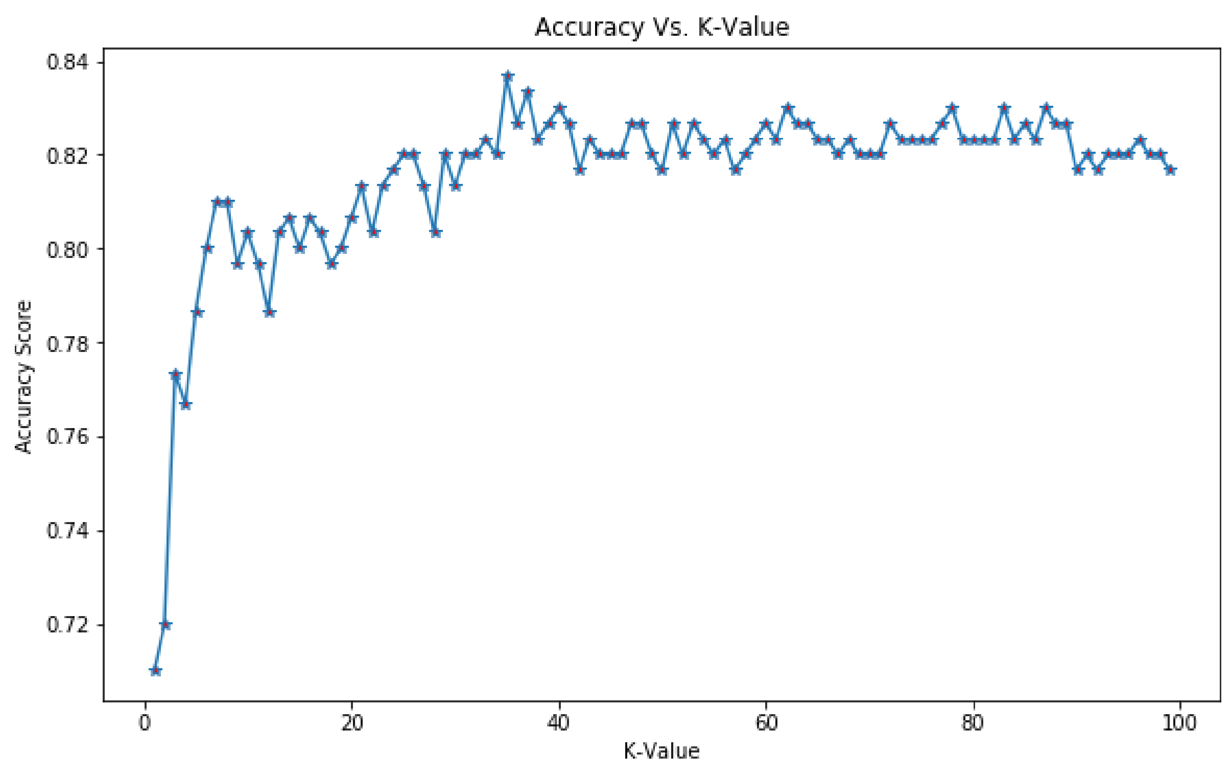
In [107]:
```
acrate = []

for i in range(1,100) :

    knn = KNeighborsClassifier(n_neighbors = i)
    knn.fit(x_train , y_train)
    pred = knn.predict(x_test)
    acc = accuracy_score(pred , y_test)
    acrate.append(acc.mean())
```

In [108]:
```
plt.figure(figsize= (10,6))
plt.plot (range(1,100) , acrate , marker = '*' , markerfacecolor = 'red')
plt.title('Accuracy Vs. K-Value')
plt.xlabel('K-Value')
plt.ylabel('Accuracy Score')
```

Out[108]:
```
Text(0, 0.5, 'Accuracy Score')
```



From the above plot we can see that the value of k was stable somewhere in the range of (60,85) Hence choosing k as 70

In [110]:
```
knn = KNeighborsClassifier (n_neighbors = 70)
knn.fit(x_train,y_train)
prd = knn.predict(x_test)
confusion_matrix (prd, y_test)
```

Out[110]:
```
array([[118,  22],
       [ 32, 128]], dtype=int64)
```

In [111]:
```
accuracy_score (prd, y_test)
```