KNN MODEL

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('Classified Data')
df.head(2)
```

Out[2]:

	Unnamed: 0	WTT	PTI	EQW	SBI	LQE	QWG	FDJ	PJF	HQE	NXJ	TARGET CLASS
0	0	0.913917	1.162073	0.567946	0.755464	0.780862	0.352608	0.759697	0.643798	0.879422	1.231409	1
1	1	0.635632	1.003722	0.535342	0.825645	0.924109	0.648450	0.675334	1.013546	0.621552	1.492702	0

In [3]:
```
#STANDARDIZE ALL FEATURES TO THE SAME SCALE
from sklearn.preprocessing import StandardScaler
```

In [4]:
```
scaler = StandardScaler()
```

In [5]:
```
scaler.fit(df.drop('TARGET CLASS', axis=1))
```

Out[5]:
```
StandardScaler()
```

In [6]:
```
scaled_feat = scaler.transform(df.drop('TARGET CLASS', axis=1))
```

In [7]:
```
df_feat = pd.DataFrame(scaled_feat , columns = df.columns[:-1])
```

In [8]:
```
df_feat.head(2)
```

Out[8]:

	Unnamed: 0	WTT	PTI	EQW	SBI	LQE	QWG	FDJ	PJF	HQE	NXJ
0	-1.730320	-0.123542	0.185907	-0.913431	0.319629	-1.033637	-2.308375	-0.798951	-1.482368	-0.949719	-0.643314
1	-1.726856	-1.084836	-0.430348	-1.025313	0.625388	-0.444847	-1.152706	-1.129797	-0.202240	-1.828051	0.636759

In [9]:
```
#Dividing data
X = df_feat
y = df['TARGET CLASS']
```

In [10]:
```
#TRAIN TEST SPLIT
from sklearn.model_selection import train_test_split
X_train , X_test , y_train , y_test = train_test_split(X,y,test_size=0.3)
```

In [11]:
```
#IMPORT MODEL
from sklearn.neighbors import KNeighborsClassifier
```

In [12]:
```
#INITIALIZING MODEL
knn = KNeighborsClassifier(n_neighbors = 1)
```

In [13]:
```
#FITTING MODEL OVER TRAINING DATA
knn.fit(X_train , y_train)
```

Out[13]:
```
KNeighborsClassifier(n_neighbors=1)
```

In [14]:
```
#PREDICTING TARGET VALUES
pred = knn.predict(X_test)
```

In [15]:
```
pred
```

Out[15]:
```
array([0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0,
       1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1,
       0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0,
       0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1,
       0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0], dtype=int64)
```

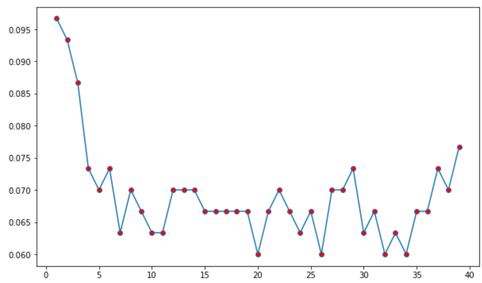
In [16]:
```
#EVALUATING OUR MODEL
from sklearn.metrics import classification_report , confusion_matrix
```

In [18]:
```
print(classification_report(y_test,pred))
print('\n')
print(confusion_matrix(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.89      0.90       152
           1       0.89      0.92      0.90       148

    accuracy                           0.90       300
   macro avg       0.90      0.90      0.90       300
weighted avg       0.90      0.90      0.90       300



[[135  17]
 [ 12 136]]
```

In [23]:
```
#ELBOW METHOD TO CHOOSE CORRECT K VALUE

#INITIALIZING EMPTY ERROR LIST
error_rate = []

#Looping through k values
for i in range(1,40):
    knn = KNeighborsClassifier(n_neighbors=i)   #Initializing the model for i value of k

    knn.fit(X_train , y_train) #Fitting the model over training data

    pred_i = knn.predict(X_test) #Predicting values over test data

    error_rate.append(np.mean(pred_i != y_test)) #Appending the values into list if predicted value of model is not equal to actual value


plt.figure(figsize=(10,6))
plt.plot(range(1,40) , error_rate , marker='o', markerfacecolor='red')
```

Out[23]:
```
[<matplotlib.lines.Line2D at 0x27d197cb670>]
```



In [24]:
```
#THIS SHOWS THAT MODEL GIVES LESS ERROR AT K = 34
```

In [32]:
```
#REFITTING THE MODEL OVER NEW K VALUE
knn_new = KNeighborsClassifier(n_neighbors = 34)
```

In [33]:
```
knn_new.fit(X_train , y_train)
```

Out[33]:
```
KNeighborsClassifier(n_neighbors=34)
```

In [34]:
```
pred_new = knn_new.predict(X_test)
```

In [35]:
```
pred_new
```

Out[35]:
```
array([0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1,
       1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1,
       0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0,
       0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0], dtype=int64)
```

In [36]:
```
print(classification_report(y_test,pred_new))
print('\n')
print(confusion_matrix(y_test,pred_new))
```

```
              precision    recall  f1-score   support

           0       0.95      0.93      0.94       152
           1       0.93      0.95      0.94       148

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



[[141  11]
 [  7 141]]
```

In [ ]:
```
```