REPORT FOR INTERNSHIP PROJECT

- **SAI GOWTHAM BABU AMBURI**

1) Before Loading the data in Jupiter, I filled the empty cells in excel. I found that there are almost 5 empty cells in X1. So, instead of removing or aggregating those rows, I found their relative terms in cell X2 and Filled it with that number using VLOOKUP function in excel, this led me take upon nearest possible values that can be filled for missing values. I repeated the same process for X2 missing variables.

X2 Values	X1 filling Values
116.138522	109.6240088
36.90540167	31.44072054
-116.385719	-121.3936378
34.71432764	31.78245726
53.18219122	45.74886637

X1 Values	X2 Filling Values
-99.6275223	67.21389845
55.16225775	65
-74.126054	41.80193067

I also Found that there are some outliers in the data. So, modified them also based on the possible numbers. Ex: - 99009998 is changed to 99. Since, all the remaining values also, fall between that range only.

	A	B	C	D
1	x1	x2	y	
543	66.63333	99009998	0	
228	109.624	99999999	0	
229				

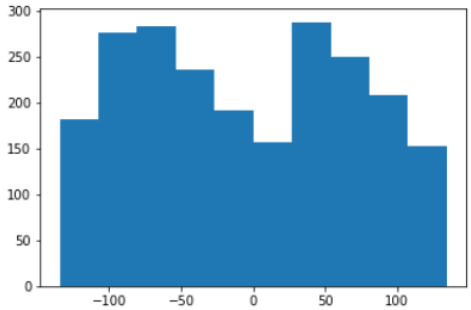
2) There is a negative correlation between Y (the predicted feature) and the independent features (X1, X2). But the accuracy may not be affected since, there is not more dependent variables present.

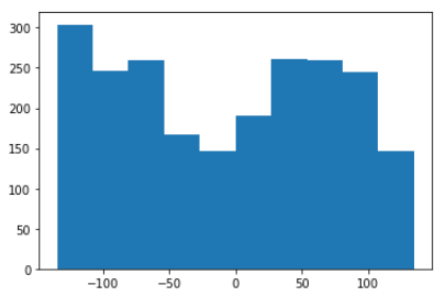
```
In [3]: df.corr()
```

Out[3]:

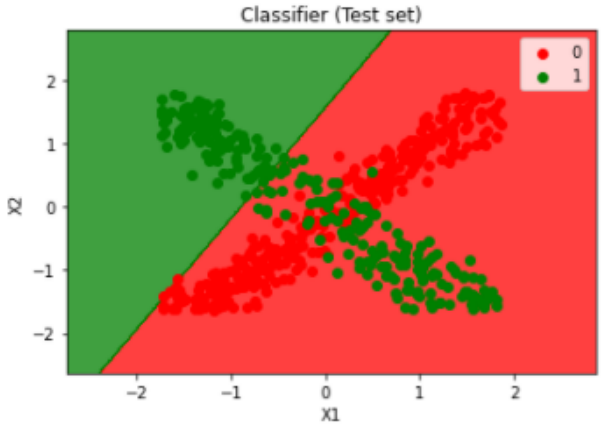
	x1	x2	y
x1	1.000000	0.112683	-0.121294
x2	0.112683	1.000000	0.061546
y	-0.121294	0.061546	1.000000

3) The values both the independent variables seem to be in same range. But better we can normalize them for more accuracy.





4) Logistic Regression with best model accuracy.

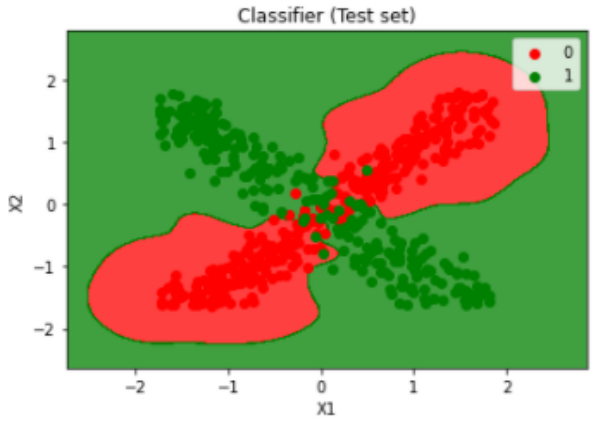


In [13]:
```
y_train_pred = classifier.predict(X_train)
y_test_pred = classifier.predict(X_test)

train_test_full_error = pd.concat([measure_error(y_train, y_train_pred, 'train'),
                                   measure_error(y_test, y_test_pred, 'test')],
                                   axis=1)

train_test_full_error
```

Out[13]:

	train	test
accuracy	0.742515	0.725314
precision	0.980769	0.981132
recall	0.419178	0.407843
f1	0.587332	0.576177

5) SVM with best model (rbf) and accuracy



In [20]:
```
y_train_pred = classifier.predict(X_train)
y_test_pred = classifier.predict(X_test)

train_test_full_error = pd.concat([measure_error(y_train, y_train_pred, 'train'),
                                   measure_error(y_test, y_test_pred, 'test')],
                                   axis=1)

train_test_full_error
```

Out[20]:

	train	test
accuracy	0.956886	0.955117
precision	0.948229	0.956349
recall	0.953425	0.945098
f1	0.950820	0.950690

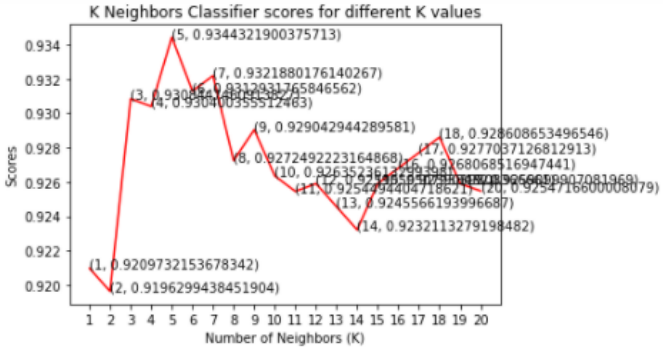
6) Decision Tree with best model accuracy


```
In [33]: y_train_pred = dt.predict(X_train)
         y_test_pred = dt.predict(X_test)

         train_test_full_error = pd.concat([measure_error(y_train, y_train_pred, 'train'),
                                            measure_error(y_test, y_test_pred, 'test')],
                                           axis=1)

         train_test_full_error
```

Out[33]:

	train	test
accuracy	1.0	0.922801
precision	1.0	0.917323
recall	1.0	0.913725
f1	1.0	0.915521

7) KNN best model and accuracy




```
In [29]: classifier = KNeighborsClassifier(n_neighbors = 5)
         classifier.fit(X_train, y_train)

         y_train_pred = classifier.predict(X_train)
         y_test_pred = classifier.predict(X_test)

         train_test_full_error = pd.concat([measure_error(y_train, y_train_pred, 'train'),
                                            measure_error(y_test, y_test_pred, 'test')],
                                           axis=1)

         train_test_full_error
```

Out[29]:

	train	test
accuracy	0.956287	0.949731
precision	0.955617	0.952191
recall	0.943836	0.937255
f1	0.949690	0.944664

Observation

We can see that we have different accuracies for different models. I would like to take SVM rbf kernel for my model building because this model is not overfitting like decision tree and also giving a max accuracy on the test set compared to remaining models.