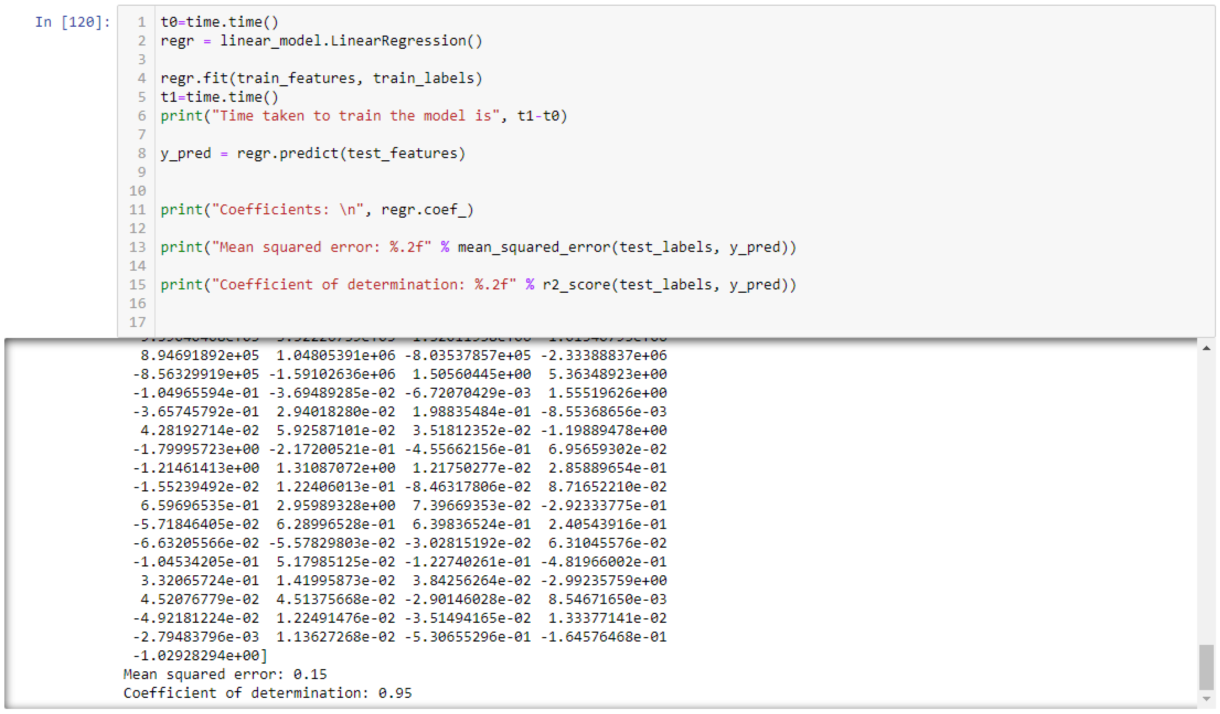
ML Project Report

Team-13

Pusapati Rama Krishna Raju

CSE-B [Company address]

Linear Regression Model:



```
In [120]:
t0=time.time()
regr = linear_model.LinearRegression()

regr.fit(train_features, train_labels)
t1=time.time()
print("Time taken to train the model is", t1-t0)

y_pred = regr.predict(test_features)


print("Coefficients: \n", regr.coef_)

print("Mean squared error: %.2f" % mean_squared_error(test_labels, y_pred))

print("Coefficient of determination: %.2f" % r2_score(test_labels, y_pred))


```

```
  8.94691892e+05  1.04805391e+06 -8.03537857e+05 -2.33388837e+06
 -8.56329919e+05 -1.59102636e+06  1.50560445e+00  5.36348923e+00
 -1.04965594e-01 -3.69489285e-02 -6.72070429e-03  1.55519626e+00
 -3.65745792e-01  2.94018280e-02  1.98835484e-01 -8.55368656e-03
  4.28192714e-02  5.92587101e-02  3.51812352e-02 -1.19889478e+00
 -1.79995723e+00 -2.17200521e-01 -4.55662156e-01  6.95659302e-02
 -1.21461413e+00  1.31087072e+00  1.21750277e-02  2.85889654e-01
 -1.55239492e-02  1.22406013e-01 -8.46317806e-02  8.71652210e-02
  6.59696535e-01  2.95989328e+00  7.39669353e-02 -2.92333775e-01
 -5.71846405e-02  6.28996528e-01  6.39836524e-01  2.40543916e-01
 -6.63205566e-02 -5.57829803e-02 -3.02815192e-02  6.31045576e-02
 -1.04534205e-01  5.17985125e-02 -1.22740261e-01 -4.81966002e-01
  3.32065724e-01  1.41995873e-02  3.84256264e-02 -2.99235759e+00
  4.52076779e-02  4.51375668e-02 -2.90146028e-02  8.54671650e-03
 -4.92181224e-02  1.22491476e-02 -3.51494165e-02  1.33377141e-02
 -2.79483796e-03  1.13627268e-02 -5.30655296e-01 -1.64576468e-01
 -1.02928294e+00]
Mean squared error: 0.15
Coefficient of determination: 0.95
```

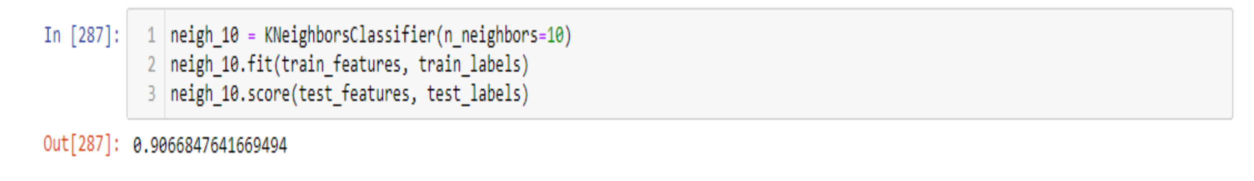
For our dataset the linear regression model showed an coefficient of determination of 0.95

where coefficient determines the measure of fit

if coefficient of determination is 0 then model is unfit and if coefficient of determination then it means an ideal fit.

Mean Square error(MSE) is used to fit the regression line to set of data points, if MSE is 0 then all values have been fit to line. Our model has given a minute MSE of 0.15

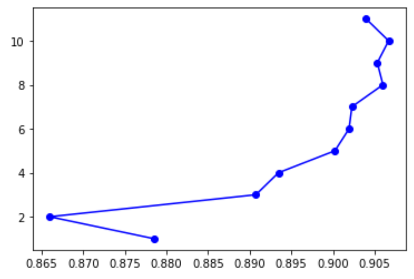
k-NN



```
In [287]: neigh_10 = KNeighborsClassifier(n_neighbors=10)
          neigh_10.fit(train_features, train_labels)
          neigh_10.score(test_features, test_labels)
Out[287]: 0.9066847641669494
```

k-NN neighbors predicts output depending on number of neighbors and their majority class features, the testing accuracy depends on k-value, for our dataset ideal value of k=10 has given max score of 90.3%

```
In [292]: plt.plot(score,k_index,'b-o')
Out[292]: [<matplotlib.lines.Line2D at 0x150067a67c0>]
```



Regression-Tree

```
In [121]: from sklearn.tree import DecisionTreeRegressor
          t0 = time.time()
          regressor = DecisionTreeRegressor(random_state = 0)


          regressor.fit(train_features, train_labels)
          t1=time.time()
          print("Time taken to train the model is", t1-t0)

          Time taken to train the model is 2.2450268268585205
```

```
In [122]: cross_val_score(regressor, test_features, test_labels, cv=10)
Out[122]: array([0.90165574, 0.81943461, 0.79811539, 0.93113585, 0.94853413,
                 0.93463494, 0.94036109, 0.93358962, 0.91960685, 0.89947585])
```

```
In [123]: fc=train_features.columns
```

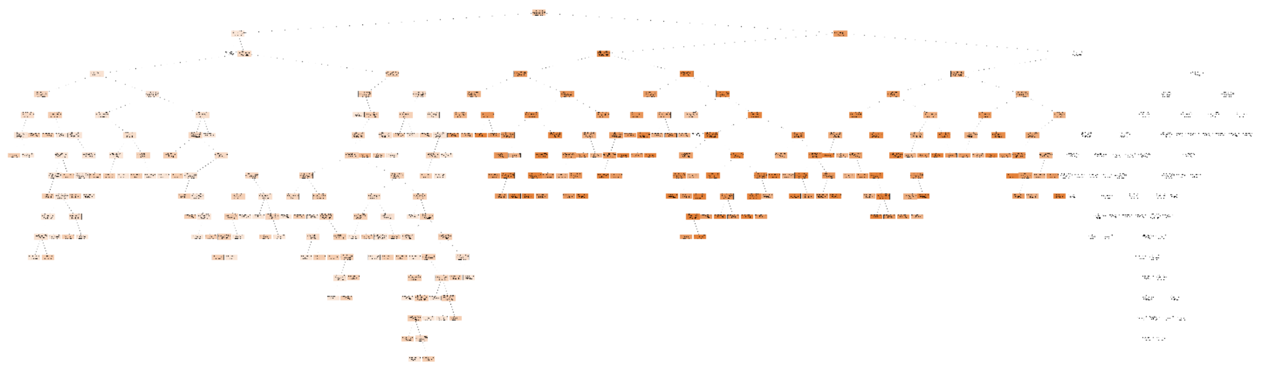
```
In [124]: from sklearn.tree import export_graphviz

          # export the decision tree to a tree.dot file
          # for visualizing the plot easily anywhere
          export_graphviz(regressor, out_file ='tree.dot',
                          feature_names =fc)
```

```
In [127]: plt.figure(figsize=(700,200))
          tree.plot_tree(regressor, filled=True)
          plt.show()
```

```
In [126]: regressor.score(test_features,test_labels)
Out[126]: 0.927141833307081
```

For our dataset the regression tree has given an accuracy of 92%

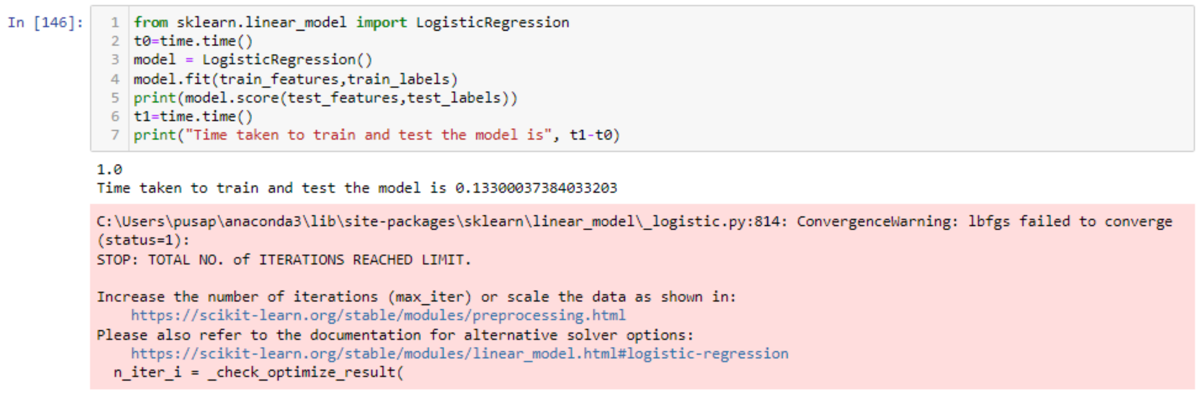
With tree given below



Although linear regression gives a maximum test score of 95% it is prone to noise when new data points are added.

Regression tree seems to be a better regression model even when overfit we can prune it and is easy to be interpreted with a good score of 92%.

Results from bi-class classification from Logistic Regression:



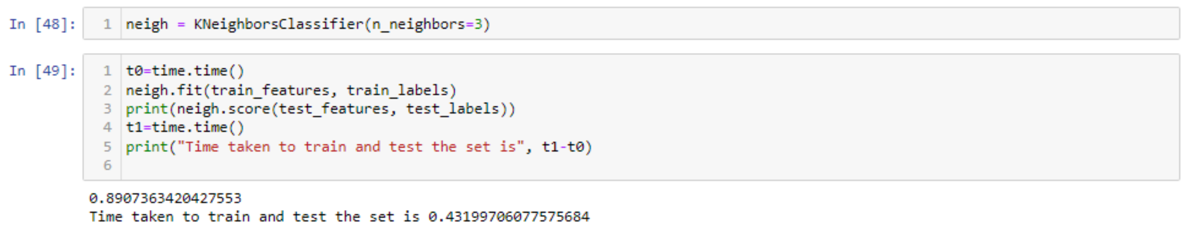
```
In [146]:
from sklearn.linear_model import LogisticRegression
t0=time.time()
model = LogisticRegression()
model.fit(train_features,train_labels)
print(model.score(test_features,test_labels))
t1=time.time()
print("Time taken to train and test the model is", t1-t0)
```

```
1.0
Time taken to train and test the model is 0.13300037384033203
```

```
C:\Users\pusap\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

It gave us a 100% accuracy with time taken to train and test 0.133 seconds.

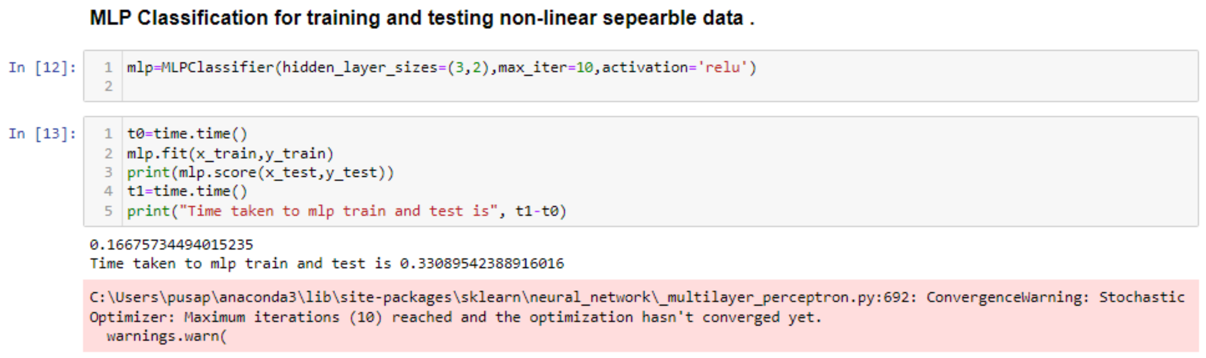
Results from k-NN classification:



```
In [48]: neigh = KNeighborsClassifier(n_neighbors=3)

In [49]: t0=time.time()
         neigh.fit(train_features, train_labels)
         print(neigh.score(test_features, test_labels))
         t1=time.time()
         print("Time taken to train and test the set is", t1-t0)

0.8907363420427553
Time taken to train and test the set is 0.43199706077575684
```

It gave us a accuracy of 89% with time taken to train and test 0.43 seconds.

Results from MLP classification:

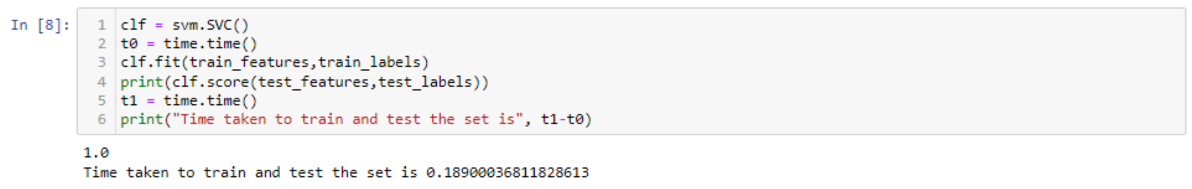


MLP Classification for training and testing non-linear seeparble data .

```
In [12]: mlp=MLPClassifier(hidden_layer_sizes=(3,2),max_iter=10,activation='relu')

In [13]: t0=time.time()
         mlp.fit(x_train,y_train)
         print(mlp.score(x_test,y_test))
         t1=time.time()
         print("Time taken to mlp train and test is", t1-t0)

0.16675734494015235
Time taken to mlp train and test is 0.33089542388916016
C:\Users\pusap\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (10) reached and the optimization hasn't converged yet.
  warnings.warn(
```

It gave us an accuracy of 16% and time of 0.33 seconds.

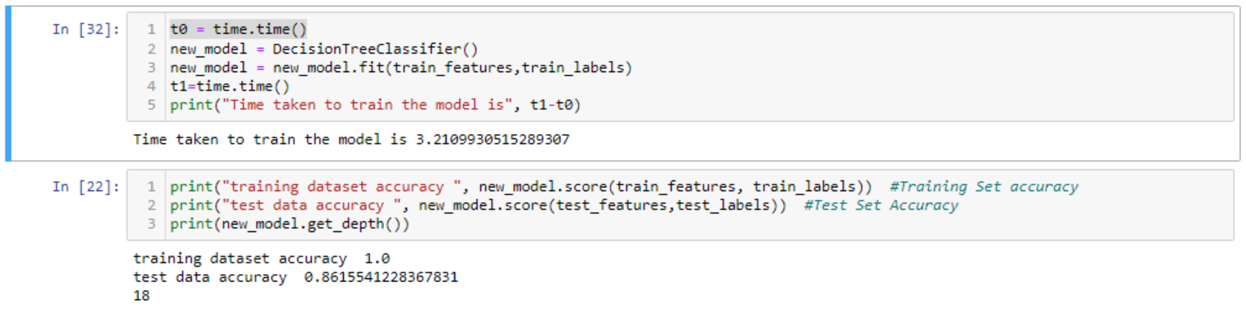
Result from SVM:



```
In [8]: clf = svm.SVC()
        t0 = time.time()
        clf.fit(train_features,train_labels)
        print(clf.score(test_features,test_labels))
        t1 = time.time()
        print("Time taken to train and test the set is", t1-t0)

1.0
Time taken to train and test the set is 0.18900036811828613
```

Accuracy is 100% and time taken is 0.189 seconds

Results from Decision Tree:

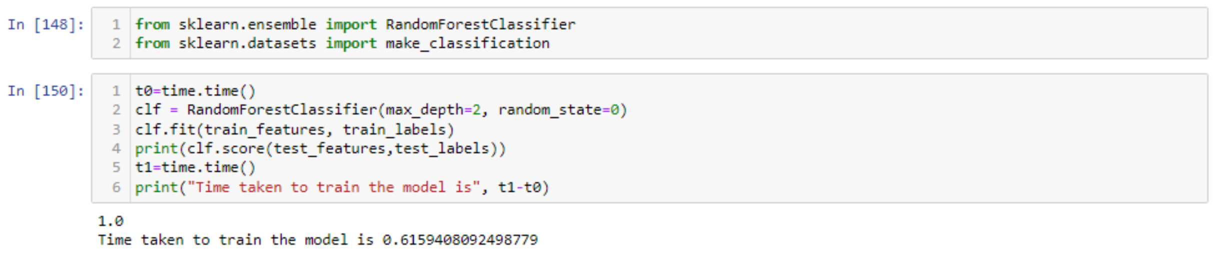


```
In [32]: t0 = time.time()
         new_model = DecisionTreeClassifier()
         new_model = new_model.fit(train_features,train_labels)
         t1=time.time()
         print("Time taken to train the model is", t1-t0)

Time taken to train the model is 3.2109930515289307

In [22]: print("training dataset accuracy ", new_model.score(train_features, train_labels))  #Training Set accuracy
         print("test data accuracy ", new_model.score(test_features,test_labels))  #Test Set Accuracy
         print(new_model.get_depth())

training dataset accuracy  1.0
test data accuracy  0.8615541228367831
18
```

Test data accuracy is 86%

And time taken is 3.2 seconds.

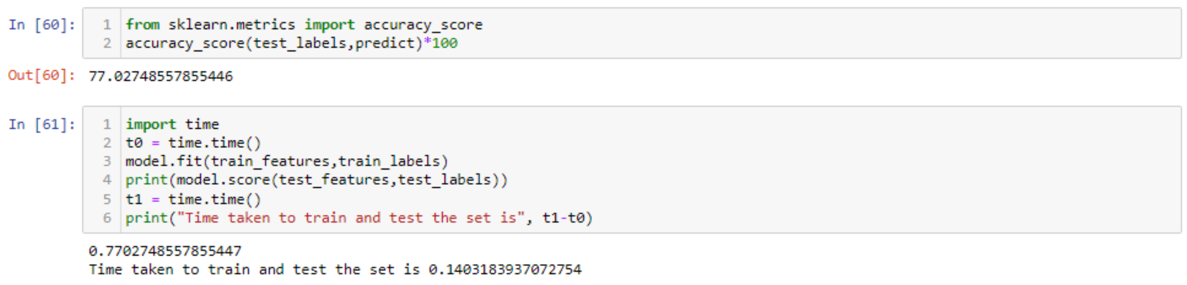
Results from Random Forest:



```
In [148]: from sklearn.ensemble import RandomForestClassifier
          from sklearn.datasets import make_classification

In [150]: t0=time.time()
          clf = RandomForestClassifier(max_depth=2, random_state=0)
          clf.fit(train_features, train_labels)
          print(clf.score(test_features,test_labels))
          t1=time.time()
          print("Time taken to train the model is", t1-t0)

1.0
Time taken to train the model is 0.6159408092498779
```

Accuracy is 100% and time taken is 0.61 seconds.

Result from NB Classifier:



```
In [60]:  from sklearn.metrics import accuracy_score
          accuracy_score(test_labels,predict)*100
Out[60]:  77.02748557855446

In [61]:  import time
          t0 = time.time()
          model.fit(train_features,train_labels)
          print(model.score(test_features,test_labels))
          t1 = time.time()
          print("Time taken to train and test the set is", t1-t0)

          0.7702748557855447
          Time taken to train and test the set is 0.1403183937072754
```

Accuracy is 77% and time taken is 0.14 seconds

Finally comparing accuracies all logistic, random forest and SVM classifier give an accuracy of 100% but least time taken to train and test is 0.13 seconds by Logistic Regression classifier.

So ideal is Logistic Regression Classifier for bi-class classification.