Part (a):

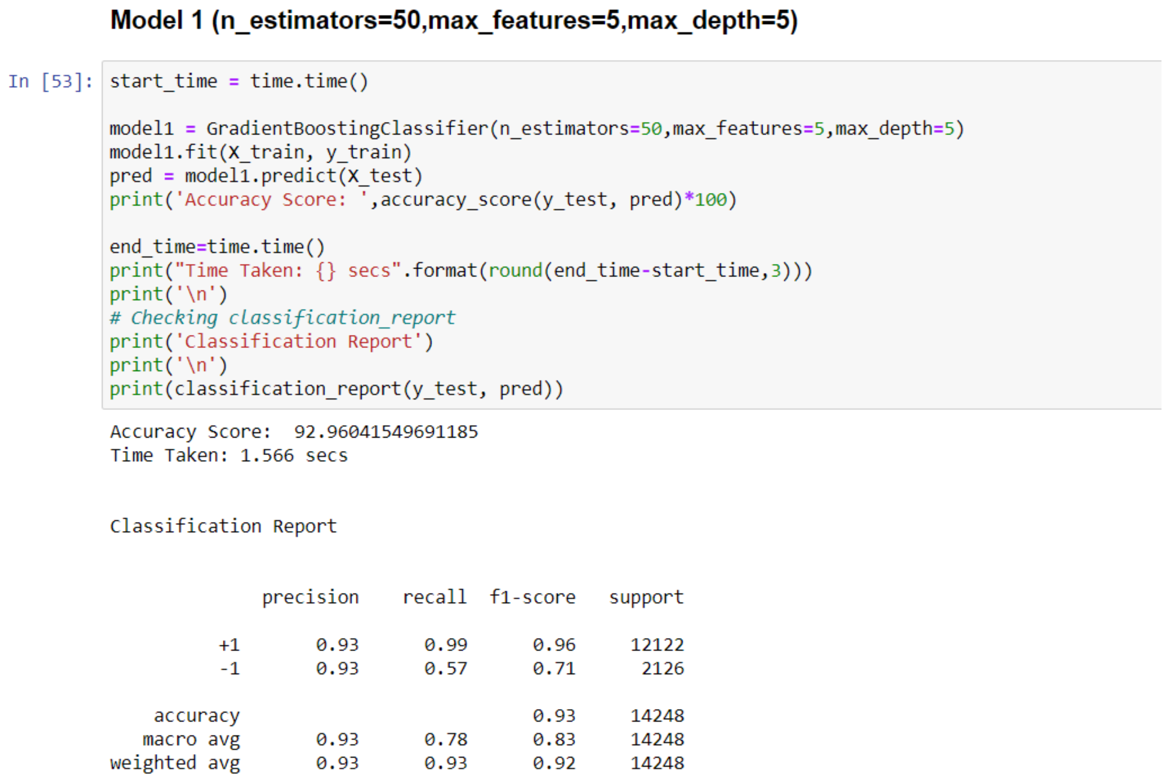
After pre-processing the data, checking and handling all the missing values, taking care of categorical and ordinal features and null values I ended up with 46 features that according to me should be important for the model prediction.

So, after pre-processing the data from 111 features I have shrinked my dataset down to 46 features only.

Part (b):

Accuracy,Precision and Recall for each of the model

Model 1:



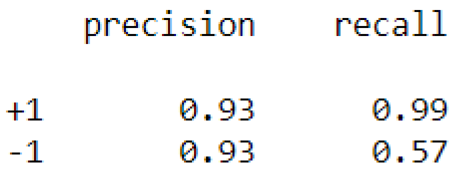
Model 1 (n_estimators=50,max_features=5,max_depth=5)

```
In [53]: start_time = time.time()

         model1 = GradientBoostingClassifier(n_estimators=50,max_features=5,max_depth=5)
         model1.fit(X_train, y_train)
         pred = model1.predict(X_test)
         print('Accuracy Score: ',accuracy_score(y_test, pred)*100)

         end_time=time.time()
         print("Time Taken: {} secs".format(round(end_time-start_time,3)))
         print('\n')
         # Checking classification_report
         print('Classification Report')
         print('\n')
         print(classification_report(y_test, pred))
```

```
Accuracy Score:  92.96041549691185
Time Taken: 1.566 secs


Classification Report


              precision    recall  f1-score   support

          +1       0.93      0.99      0.96     12122
          -1       0.93      0.57      0.71      2126

    accuracy                           0.93     14248
   macro avg       0.93      0.78      0.83     14248
weighted avg       0.93      0.93      0.92     14248
```

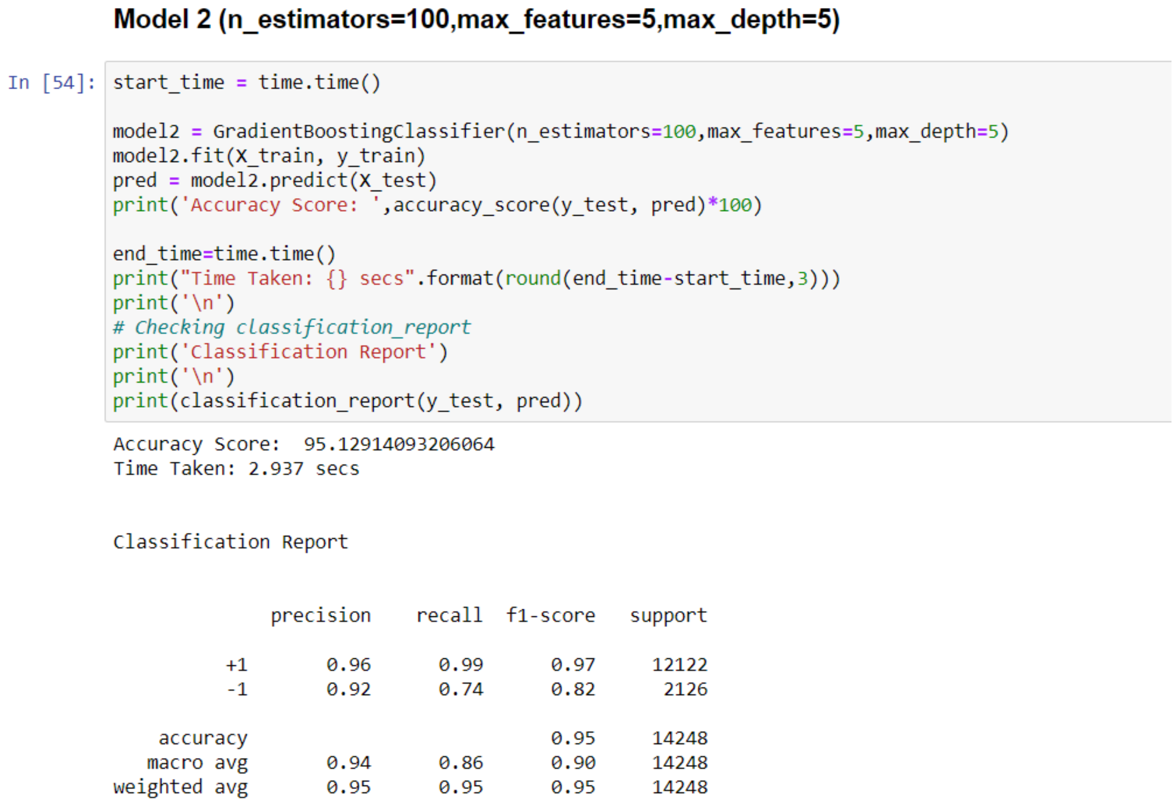
Accuracy : 92.96 %

Precision and Recall :



```
      precision    recall

+1         0.93      0.99
-1         0.93      0.57
```

Model 2:



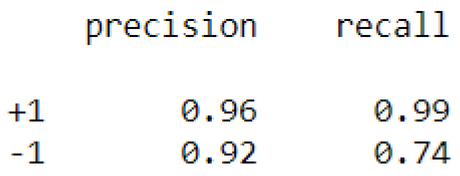
Model 2 (n_estimators=100,max_features=5,max_depth=5)

```
In [54]: start_time = time.time()

         model2 = GradientBoostingClassifier(n_estimators=100,max_features=5,max_depth=5)
         model2.fit(X_train, y_train)
         pred = model2.predict(X_test)
         print('Accuracy Score:  ',accuracy_score(y_test, pred)*100)

         end_time=time.time()
         print("Time Taken: {} secs".format(round(end_time-start_time,3)))
         print('\n')
         # Checking classification_report
         print('Classification Report')
         print('\n')
         print(classification_report(y_test, pred))
```

```
Accuracy Score:  95.12914093206064
Time Taken: 2.937 secs


Classification Report


              precision    recall  f1-score   support

          +1       0.96      0.99      0.97     12122
          -1       0.92      0.74      0.82      2126

    accuracy                           0.95     14248
   macro avg       0.94      0.86      0.90     14248
weighted avg       0.95      0.95      0.95     14248
```

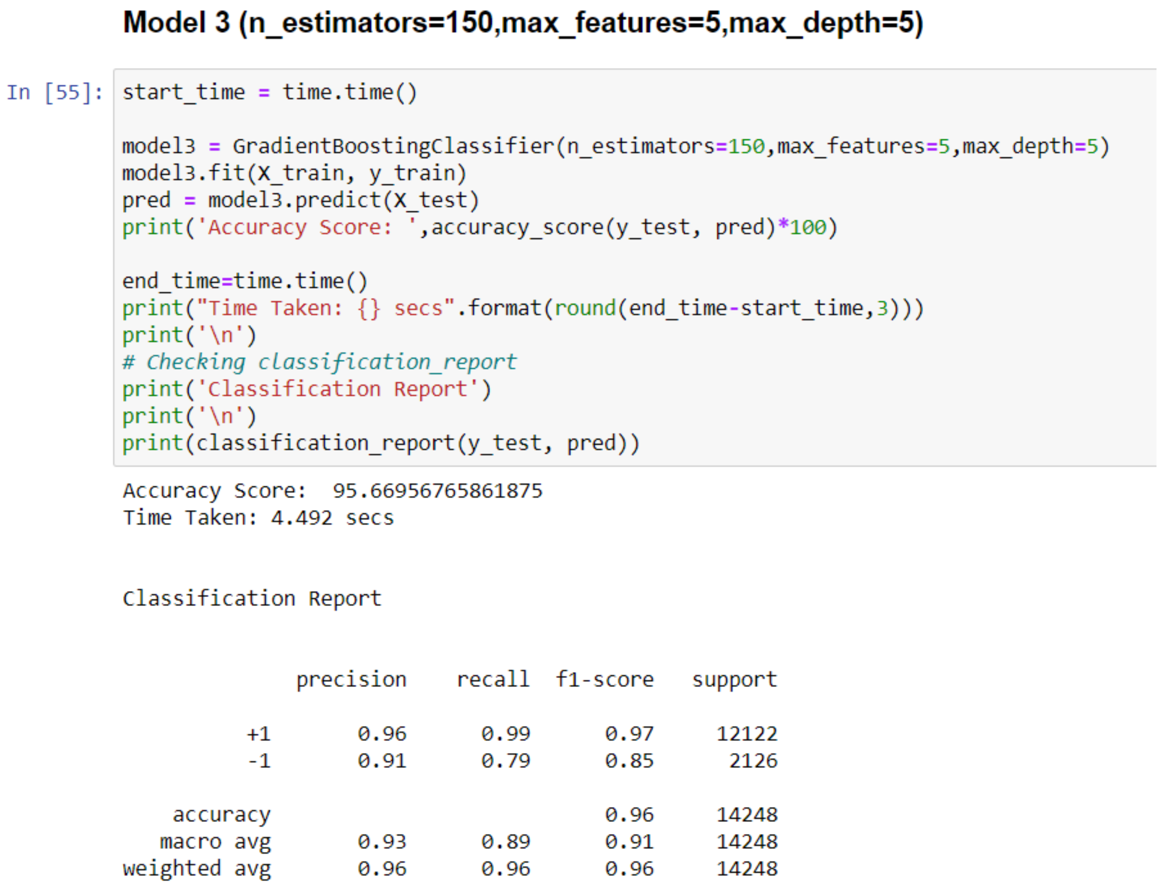
Accuracy : 95.12 %

Precision and Recall :



```
      precision    recall

+1         0.96      0.99
-1         0.92      0.74
```

Model 3:



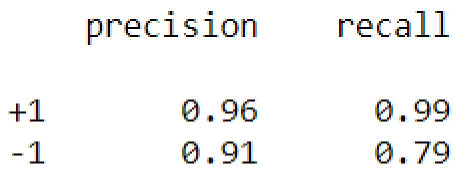
Model 3 (n_estimators=150,max_features=5,max_depth=5)

```
In [55]: start_time = time.time()

         model3 = GradientBoostingClassifier(n_estimators=150,max_features=5,max_depth=5)
         model3.fit(X_train, y_train)
         pred = model3.predict(X_test)
         print('Accuracy Score: ',accuracy_score(y_test, pred)*100)

         end_time=time.time()
         print("Time Taken: {} secs".format(round(end_time-start_time,3)))
         print('\n')
         # Checking classification_report
         print('Classification Report')
         print('\n')
         print(classification_report(y_test, pred))
```

```
Accuracy Score:  95.66956765861875
Time Taken: 4.492 secs


Classification Report


              precision    recall  f1-score   support

          +1       0.96      0.99      0.97     12122
          -1       0.91      0.79      0.85      2126

    accuracy                           0.96     14248
   macro avg       0.93      0.89      0.91     14248
weighted avg       0.96      0.96      0.96     14248
```

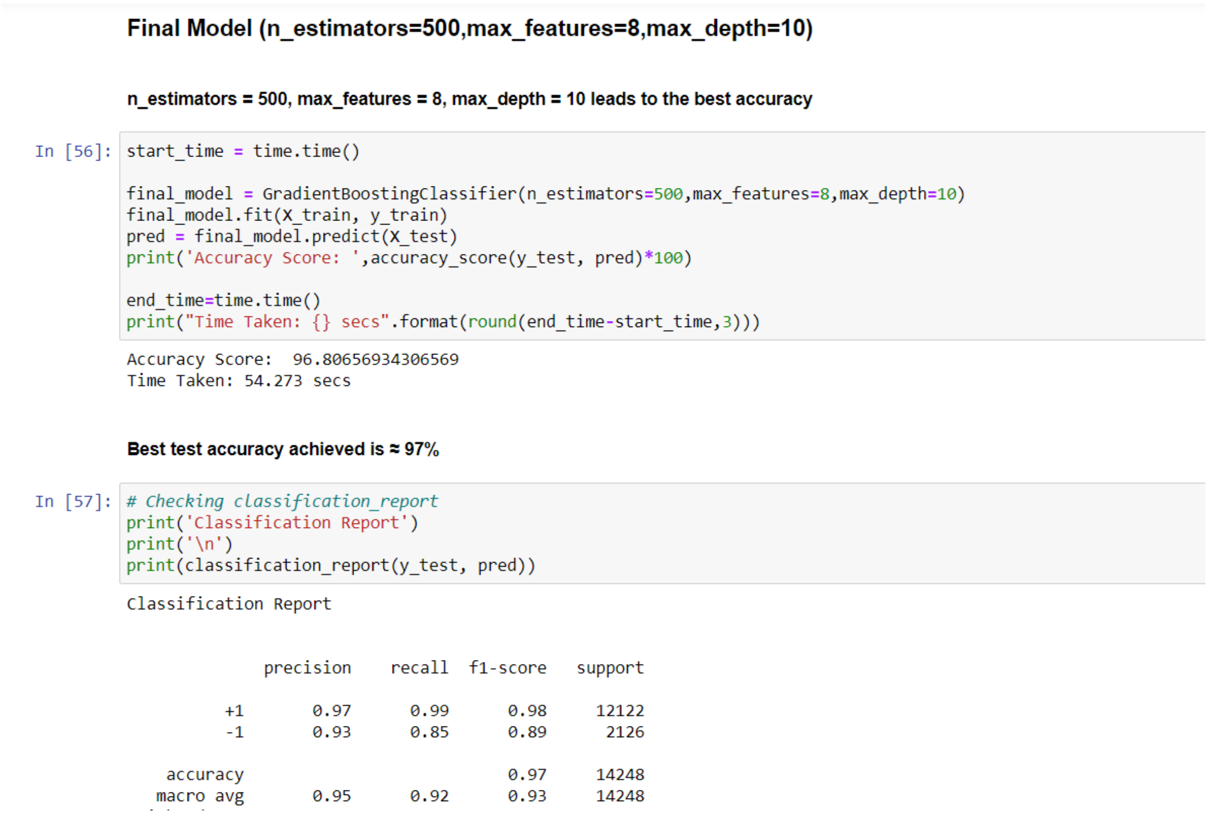
Accuracy : 95.66 %

Precision and Recall :



```
      precision    recall

+1         0.96      0.99
-1         0.91      0.79
```

Final Model:



Final Model (n_estimators=500,max_features=8,max_depth=10)

n_estimators = 500, max_features = 8, max_depth = 10 leads to the best accuracy

```
In [56]: start_time = time.time()

         final_model = GradientBoostingClassifier(n_estimators=500,max_features=8,max_depth=10)
         final_model.fit(X_train, y_train)
         pred = final_model.predict(X_test)
         print('Accuracy Score: ',accuracy_score(y_test, pred)*100)

         end_time=time.time()
         print("Time Taken: {} secs".format(round(end_time-start_time,3)))
```

```
Accuracy Score:  96.80656934306569
Time Taken: 54.273 secs
```

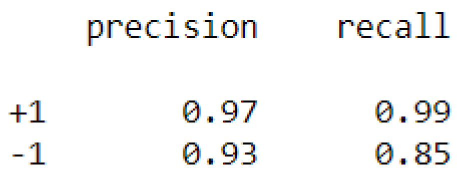
Best test accuracy achieved is ≈ 97%

```
In [57]: # Checking classification_report
         print('Classification Report')
         print('\n')
         print(classification_report(y_test, pred))
```

```
Classification Report


              precision    recall  f1-score   support

          +1       0.97      0.99      0.98     12122
          -1       0.93      0.85      0.89      2126

    accuracy                           0.97     14248
   macro avg       0.95      0.92      0.93     14248
```

Accuracy : 96.80 %

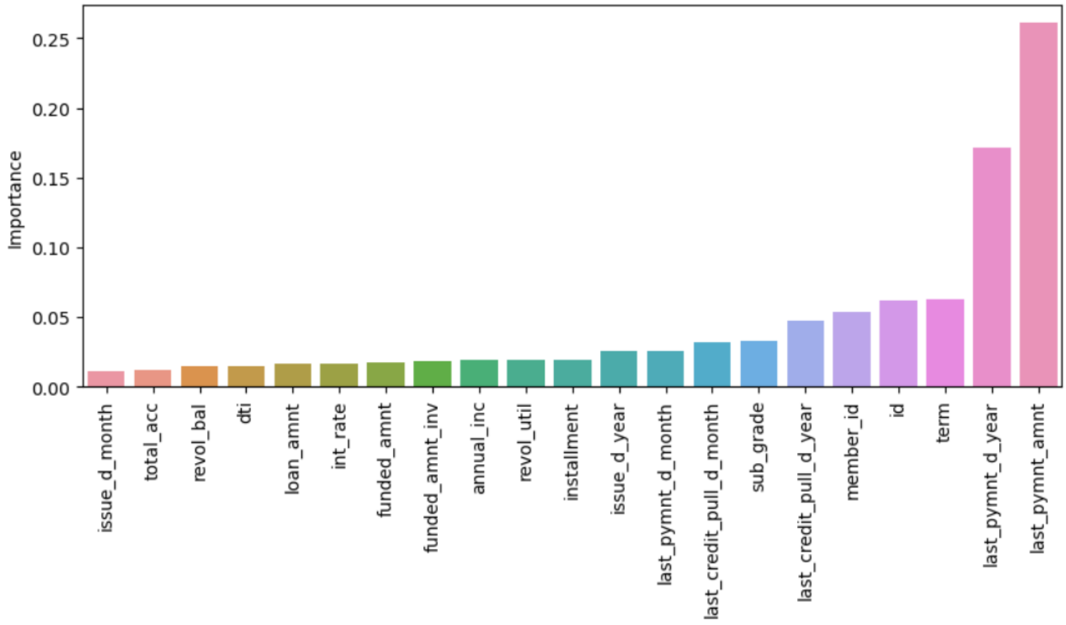
Precision and Recall :



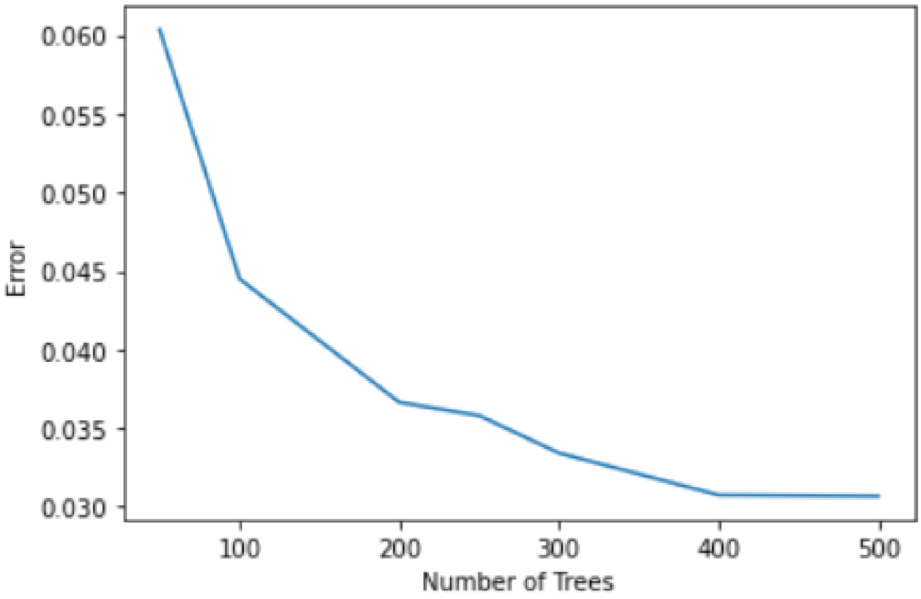
```
         precision    recall

  +1          0.97      0.99
  -1          0.93      0.85
```

So, Best Accuracy achieved is : 96.80%

Checking for the Important features having atleast 1% corr with our class label as per our model



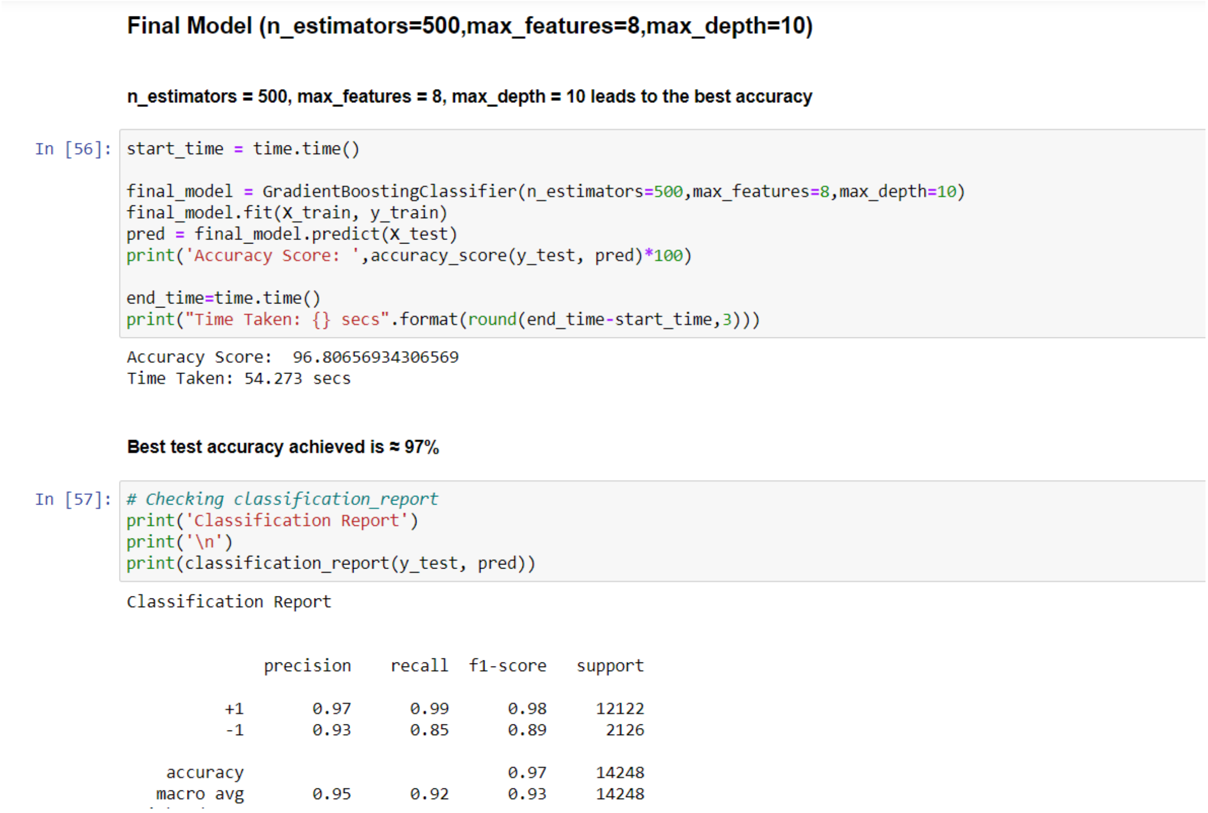
Effect of increasing the number of trees on the test error:



As we increase the number of trees the test error starts to decrease and it will come to a point where increasing the number of trees any further will not affect the test error(like from 400-500 trees in the diagram above).

Comparing the best performance (accuracy, precision, recall) of our model against a simple decision tree built using information gain:

Using GradientBoosting Classifier(Best One):



Final Model (n_estimators=500,max_features=8,max_depth=10)

n_estimators = 500, max_features = 8, max_depth = 10 leads to the best accuracy

```
In [56]: start_time = time.time()

         final_model = GradientBoostingClassifier(n_estimators=500,max_features=8,max_depth=10)
         final_model.fit(X_train, y_train)
         pred = final_model.predict(X_test)
         print('Accuracy Score: ',accuracy_score(y_test, pred)*100)

         end_time=time.time()
         print("Time Taken: {} secs".format(round(end_time-start_time,3)))
```

```
Accuracy Score:  96.80656934306569
Time Taken: 54.273 secs
```

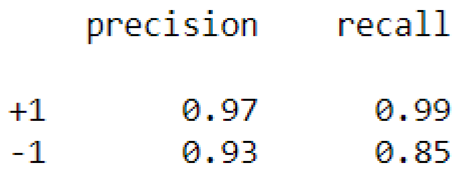
Best test accuracy achieved is ≈ 97%

```
In [57]: # Checking classification_report
         print('Classification Report')
         print('\n')
         print(classification_report(y_test, pred))
```

```
Classification Report


              precision    recall  f1-score   support

          +1       0.97      0.99      0.98     12122
          -1       0.93      0.85      0.89      2126

    accuracy                           0.97     14248
   macro avg       0.95      0.92      0.93     14248
```

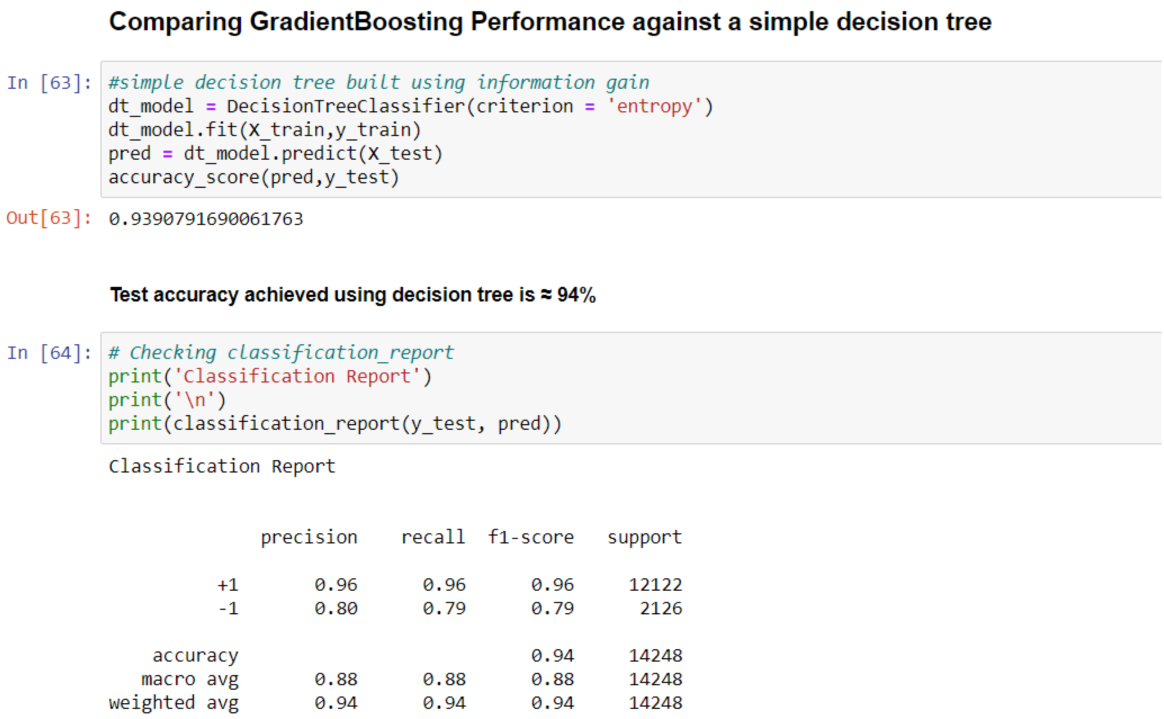
Accuracy: ~ 97 %

Precision and Relcall:



```
      precision    recall

+1         0.97      0.99
-1         0.93      0.85
```

Simple Decision Tree built using Information Gain:



Comparing GradientBoosting Performance against a simple decision tree

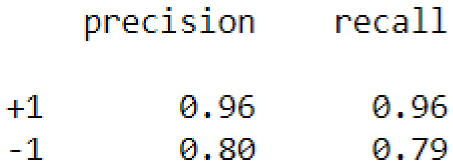
```
In [63]: #simple decision tree built using information gain
         dt_model = DecisionTreeClassifier(criterion = 'entropy')
         dt_model.fit(X_train,y_train)
         pred = dt_model.predict(X_test)
         accuracy_score(pred,y_test)

Out[63]: 0.9390791690061763
```

Test accuracy achieved using decision tree is ≈ 94%

```
In [64]: # Checking classification_report
         print('Classification Report')
         print('\n')
         print(classification_report(y_test, pred))

         Classification Report


                       precision    recall  f1-score   support

                  +1       0.96      0.96      0.96     12122
                  -1       0.80      0.79      0.79      2126

            accuracy                           0.94     14248
           macro avg       0.88      0.88      0.88     14248
        weighted avg       0.94      0.94      0.94     14248
```

Accuracy: ~ 94 %

Precision and Recall:



```
      precision    recall

+1         0.96      0.96
-1         0.80      0.79
```