Assignment 3

4.

a.

Time taken to run Random Forest from scratch around 18min

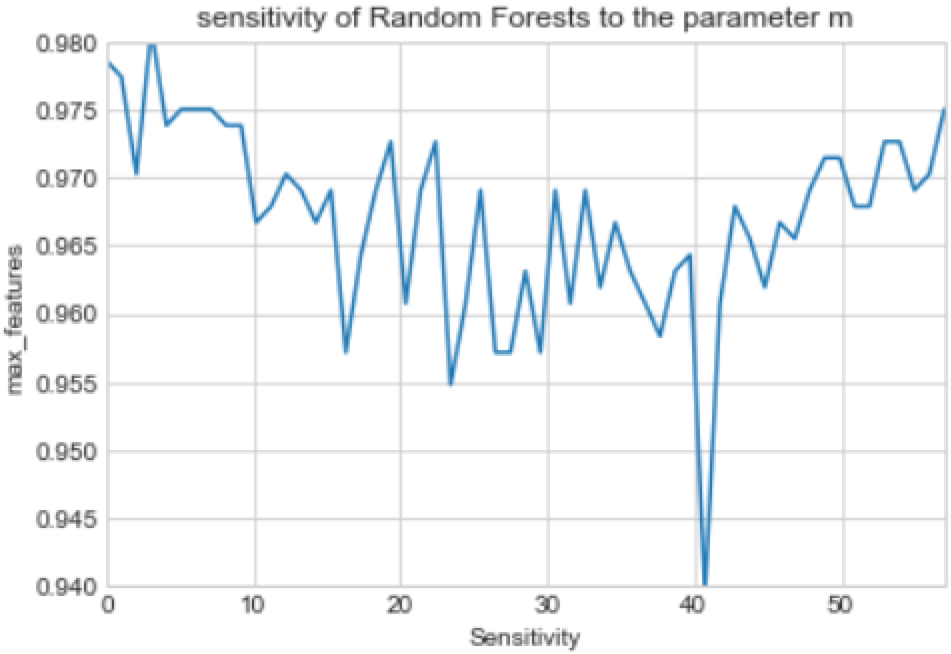
Time taken to Random Forest using inbuilt sklearn around few seconds

Accuracy from scratch code 1 (actually rounded off)

Accuracy for sklearn built model 0.9485879797248371

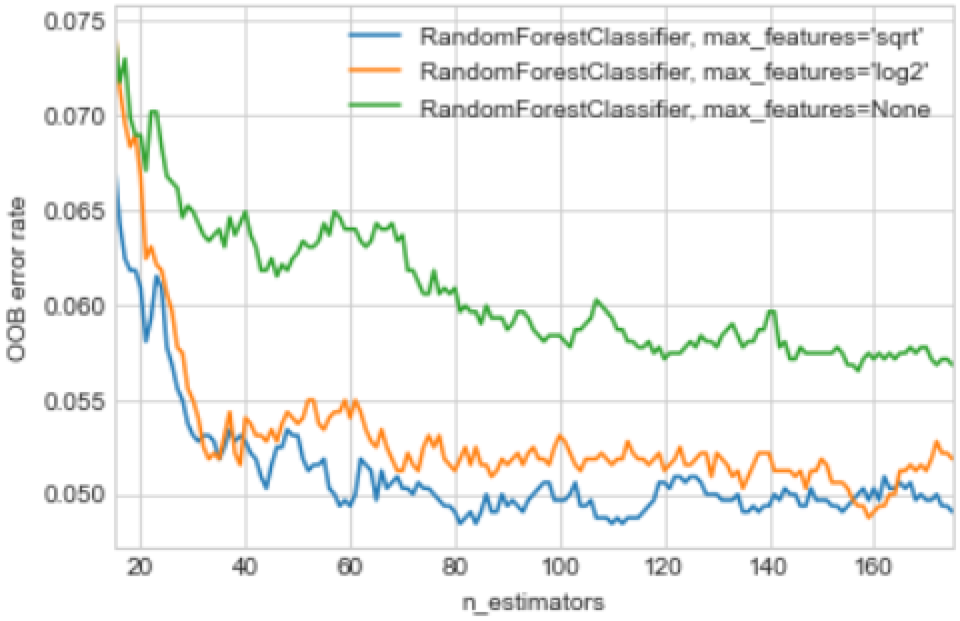
b.

Sensitivity vs max_features graph was plotted and variations were observed

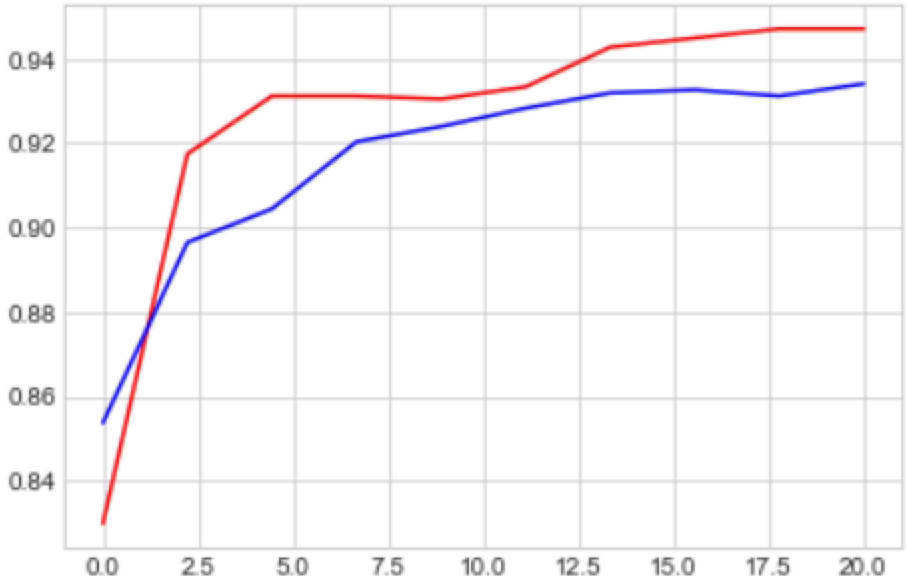


c.

Out of bag error vs n_estimators



Accuracy vs n_estimators



Red : When max_features is sqrt Blue : When max_features is log2

5

a.

Data set is loaded into jupyter notebooks.
Steps followed in preprocessing are:

1. Test dataset is of size 24999*111. Labels (loan_status) are converted into numerical and separated initially.
2. Converted interest column percentages into float values.
3. Then I started eyeballing all those columns of object datatype
4. Took the term column as numerical values.
5. Removed Grade and sub_grade columns.
6. Calculated the mean of the acc_now_delinq column and found it to be zero,so removed it.
7. Removed those columns which had all 0 values.
8. Removed all those columns which are filled with NaN,Nas. This drastically reduced the size of the dataset.
9. Removed those columns which have the same values throughout the columns (uniqueness).
10. Removed those which contained only 0s and Nan type throughout the columns.
11. Removed those columns which I think are not required for the model to build up.
12. Converted revol_util column percentages into float values.
13. Removed emp_title column because of its skewness (majority of data points belong to the same class).
14. Now changed those object data type columns to integer by assigning appropriate equals. (Understood domain and assigned values not randomly
    Suppose in home_ownership columnI assigned Rent : 1, Own : 2, Mortgage : -1, others : 0 as Mortgage holders already have a loan on it so gave -1, own house owners given high weight.
    Every column was understood and done.
15. Filled all null values with the previous one. Some of nulls are assigned with their mean values of respective columns.

With all these preprocessing steps I felt the training dataset is ready to work on.

Followed the same steps with test data set.

b.

Hyper parameters used:
n_estimators : No.of trees
learning_rate : weights assigned to predictions from each tree (low learning rate takes time but gets perfect fit)
subsample : Takes random subset of inputs for each tree which reduce overfitting
max_depth : max depth allowed for each tree to grow
max_feautures increases speed of training reduce overfitting
verbose is used to control the print of iterations

Accuracy relates how exactly we predicted,
high precision relates to a low false positive rate and
high recall relates to a low false negative rate.

Best accuracy, Best precision and recall values : 0.9883815735833673 0.9887756495964876 0.9883815735833673

Learning rate(weights assigned to predictions from each tree), No.of trees has a huge impact on accuracy, precision and recall values is observed.

accuracy,precision and recall values for single decision tree :
0.9943606468270145 0.9943496942824094 0.9943606468270145

I went to 28 columns finally. I haven seen the feature importance matrix too . But the values will be continuously changing .
But I knew in industry standards generally people go with 10 features max.
One issue I faced here was lack of proper domain knowledge. So with proper domain higher accuracies can be achieved even with low no.of features.