Report

4a. Accuray that I got from decision tree itself is: (Accuracy: 0.9174511223750905)

Accuray that I got from Random Forest tree itself is: (Accuracy: 0.9377262853005068)

Total runtime of the Random Forest from scratch is 1828.573861837387 seconds

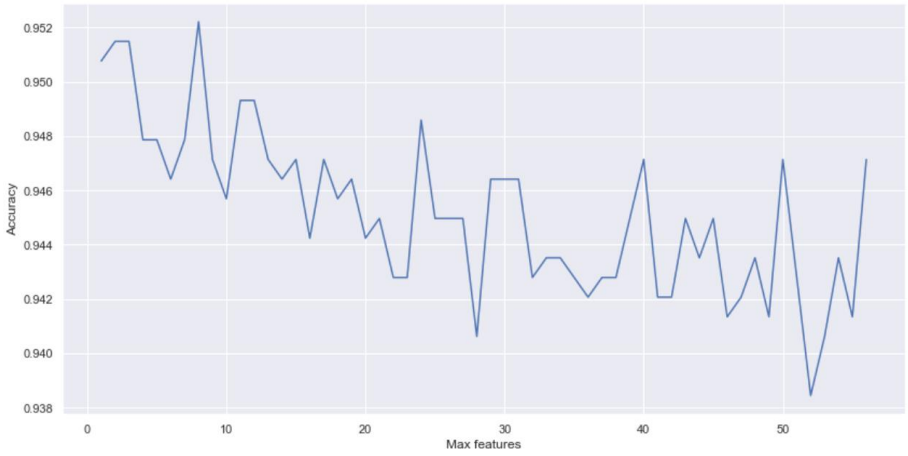
Accuray that I got from Random Forest using sklearn itself is:

(Accuracy:0.9500362056480811)

Total runtime of the Random Forest sklearn is at most 4 seconds

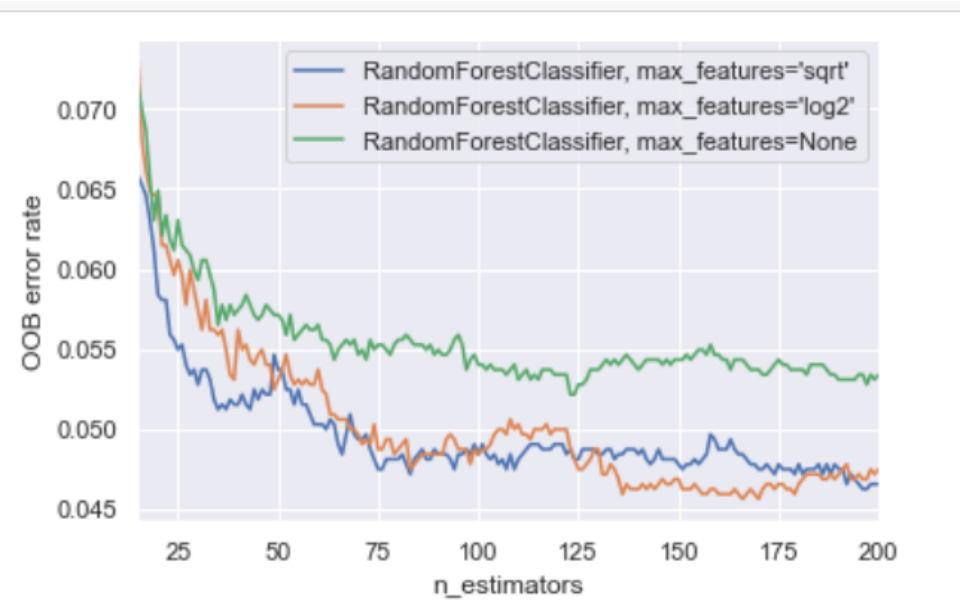
From the above results My random forest performed better while I implemented using sklearn as my accuracy is more and runtime is less for 100 trees using information gain.

4b.

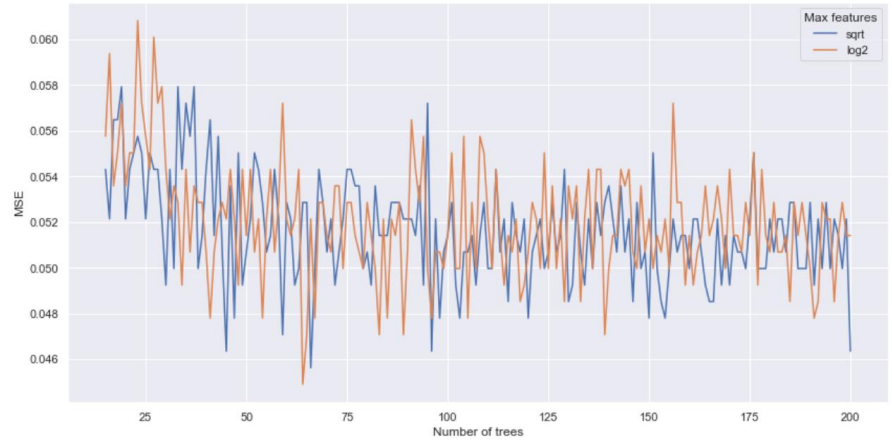


I drew the graph between maximum features and accuracy by implementing for loop. One take-away is Random Forest works better just like what Breiman suggested. When m << number of predictor(M) variables. Breiman suggests three possible values for m: ½√M, √M, and 2√M. Highest is observed at √M which is 'sqrt' aka "auto" for max_features in sklearn.

4c. I have performed analysis for OOB and error and drew a graph for 3 m values which are “auto”, “log2” and “None” I.e., 3 inbuilt max_features of sklearn. The plot is something like this:



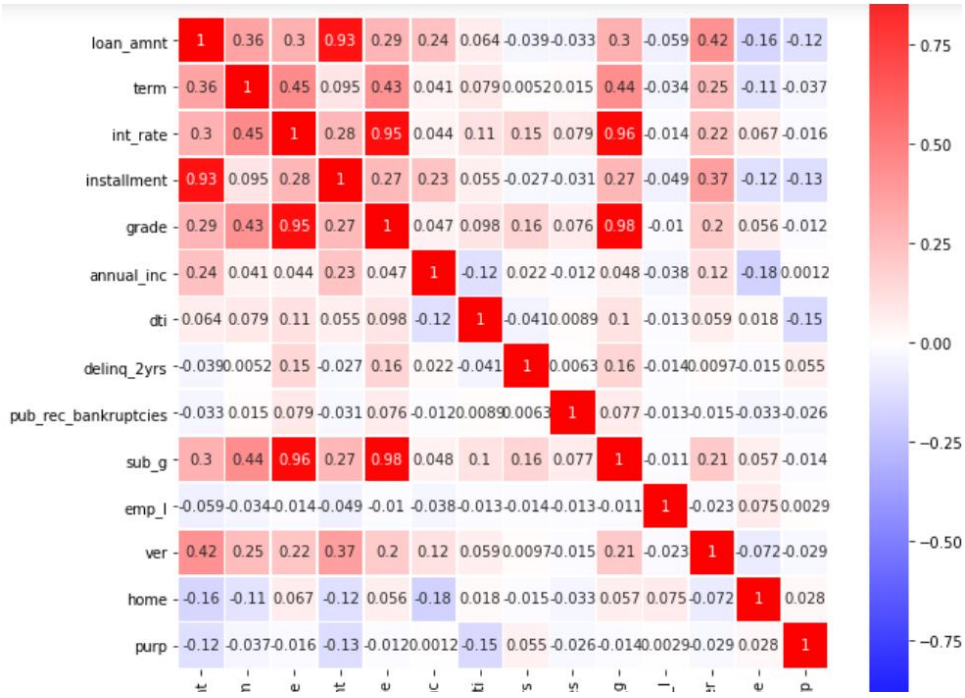
Plot between MSE and N_estimators for selected m features:



5a. My first step in Data-cleaning is removing the columns that 24999 null values. I have removed the columns "tax_liens","chargeoff_within_12_mths","collections_12_mths_ex_med" as it has only one unique value which is 0 and it wouldn't be useful, Also dropping the columns which aren't useful for analysis 'url', 'desc', 'policy_code','earliest_cr_line', 'emp_title','id','next_pymnt_d', 'title', 'total_rec_int', 'total_rec_late_fee', 'total_rec_prncp', 'zip_code','issue_d'.

-> 'loan_amnt','term','int_rate','installment','grade','sub_grade','emp_length','home_ownership', annual_inc','verification_status','purpose','dti','delinq_2yrs','pub_rec_bankruptcies','loan_status', These are the columns that I felt plays the crucial role for predicting whether the loan is Default or not.

I plotted heatmap for these features by changing the object type columns to category type columns



I found grade and sub-grade column have large correlation with int_rate so I have dropped them.

Null values that I found in my new data frame are:

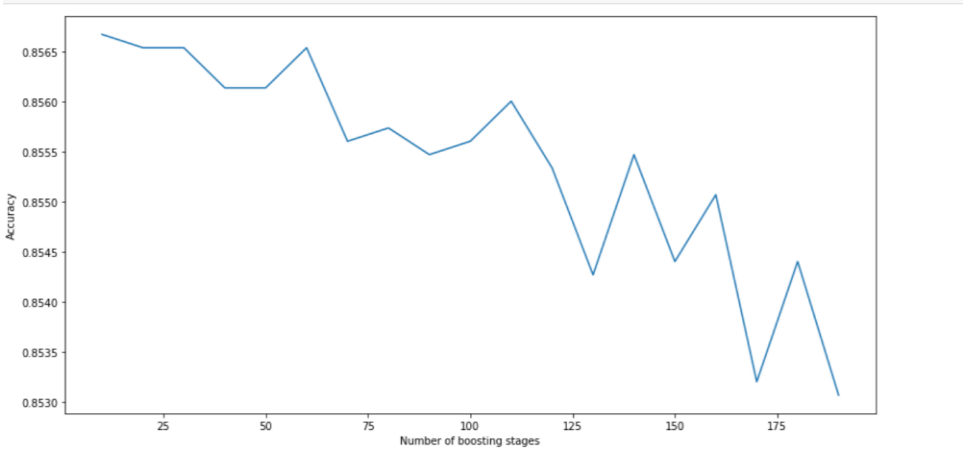
	Count	Percent
emp_length	41	0.164007
pub_rec_bankruptcies	417	1.668067

In emp_length ,I filled the null values with 0 assuming that the borrower hasn't worked many years for his data to be recorded. For Pub_rec_bankruptices I filled the Nan values with 0.0 Which are 94.3% values.

I have 'Charged Off', 'Fully Paid', 'Current values' in my loan_status , I gave default to charged-off ' and rest all I assumed it to be not-Default.

By hyperparameter tuning I got best results when learning_rate = 0.05, n_estimators = 10, subsample = 0.5, max_depth=5 which `0.862964`

I found accuracy decreased as the no. Of trees increasing. Here's the graph for it.



Classification result for the best model:

	precision	recall	f1-score	support
0	0.86	1.00	0.92	6425
1	0.00	0.00	0.00	1075
accuracy			0.86	7500
macro avg	0.43	0.50	0.46	7500
weighted avg	0.73	0.86	0.79	7500

Classification accuracy using simple Decision Tree

	precision	recall	f1-score	support
0	0.87	0.86	0.86	6425
1	0.19	0.20	0.20	1075
accuracy			0.76	7500
macro avg	0.53	0.53	0.53	7500
weighted avg	0.77	0.76	0.77	7500

Gradient Boosting has an accuracy of 86 percent while Simple decision tree gave 76 percent accuracy this show how powerful Gboosting is