



**FRA Project (Milestone-2)**

# Credit Risk

## Problem Statement

Businesses or companies can fall prey to default if they are not able to keep up their debt obligations. Defaults will lead to a lower credit rating for the company which in turn reduces its chances of getting credit in the future and may have to pay higher interests on existing debts as well as any new obligations. From an investor's point of view, he would want to invest in a company if it is capable of handling its financial obligations, can grow quickly, and is able to manage the growth scale.

A balance sheet is a financial statement of a company that provides a snapshot of what a company owns, owes, and the amount invested by the shareholders. Thus, it is an important tool that helps evaluate the performance of a business.

Data that is available includes information from the financial statement of the companies for the previous year (2015). Also, information about the Networth of the company in the following year (2016) is provided which can be used to drive the labeled field.

Explanation of data fields available in Data Dictionary, 'Credit Default Data Dictionary.xlsx'

## 1.8 Build a Random Forest Model on Train Dataset. Also showcase your model building approach.

we have created two models using Random Forest one without SMOTE data and one with SMOTE data. We did not scale the data for Random Forest as tree based models are not distance based models and can handle varying ranges of features. We also used GridSearchCV for hyper parameter tuning. PFB the grid which was used. Same grid was used for both the models i.e. with and without smote to maintain uniformity.

A custom function named apply\_evl which takes below arguments was created. This function is able to handle modelling for different type of models and returns the performance metrics as the output. The same function has been used to build and evaluate the different models here.

```
RandomForest
-----Best Parameters-----
{'max_depth': 10, 'max_features': 6, 'min_samples_leaf': 50, 'min_samples_split': 50, 'n_estimators': 100}
```

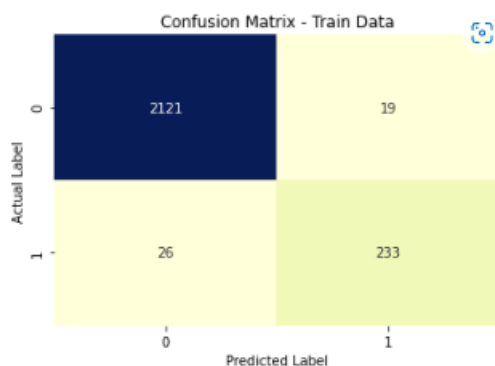
```
-----Best Model Params-----
RandomForestClassifier(max_depth=10, max_features=6, min_samples_leaf=50,
                        min_samples_split=50)
```

Train Accuracy Score for model RandomForestClassifier() is 0.9812421842434348

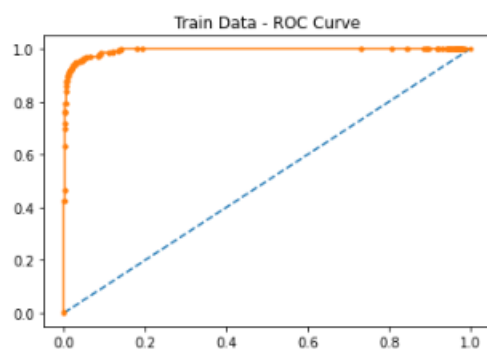
```
-----Classification Report - Train Data-----
              precision    recall  f1-score   support

     0           0.99       0.99       0.99         2140
     1           0.92       0.90       0.91          259

 accuracy          0.98          0.98          0.98         2399
 macro avg         0.96          0.95          0.95         2399
 weighted avg      0.98          0.98          0.98         2399
```



AUC: 0.994



Below are the performance metrics on train data for random forest model with smote data.

```
RF_With_Smote
-----Best Parameters-----
{'max_depth': 30, 'max_features': 3, 'min_samples_leaf': 25, 'min_samples_split': 25, 'n_estimators': 150}

-----Best Model Params-----
RandomForestClassifier(max_depth=30, max_features=3, min_samples_leaf=25,
                        min_samples_split=25, n_estimators=150)

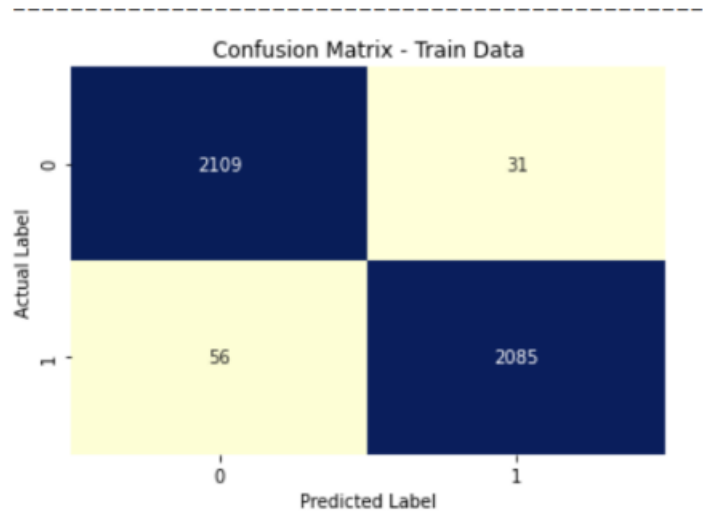
Train Accuracy Score for model RandomForestClassifier() is 0.979677645409951

-----Classification Report - Train Data-----
              precision    recall  f1-score   support

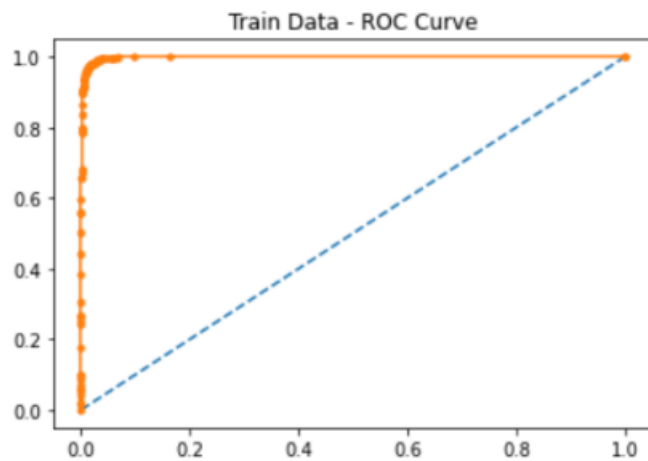
     0       0.97         0.99         0.98         2140
     1       0.99         0.97         0.98         2141

 accuracy          0.98
 macro avg         0.98         0.98         0.98         4281
 weighted avg      0.98         0.98         0.98         4281

-----
```



AUC: 0.998



**1.9 Validate the Random Forest Model on test Dataset and state the performance matrices. Also state interpretation from the model.**

Both the random forest models were then evaluated on the test data. Below are the results for test data for random forest model without smote data.

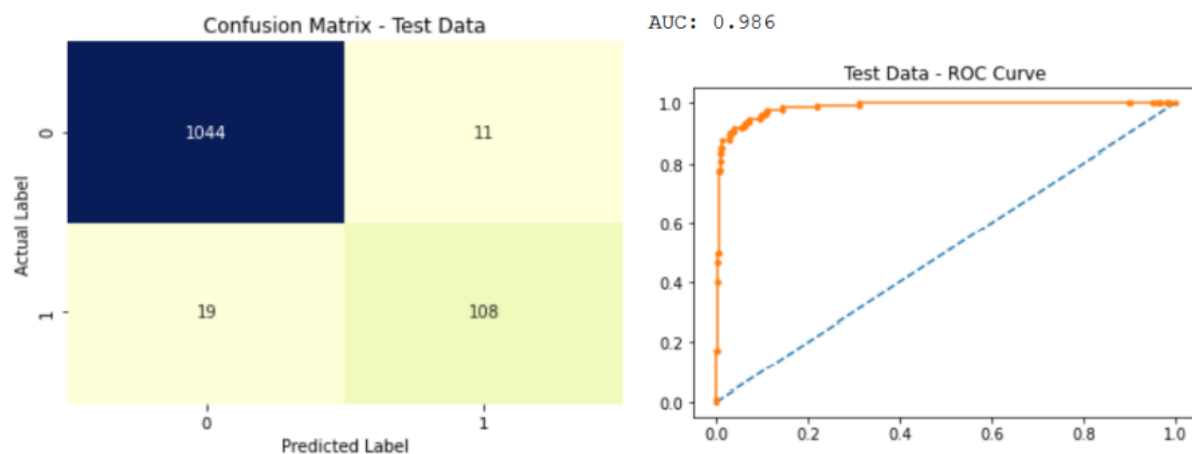
Test Accuracy Score for model RandomForestClassifier() is 0.9746192893401016

```
-----Classification Report - Test Data-----
              precision    recall  f1-score   support

     0       0.98         0.99         0.99        1055
     1       0.91         0.85         0.88         127

 accuracy          0.97         0.97         0.97        1182
 macro avg         0.94         0.92         0.93        1182
 weighted avg      0.97         0.97         0.97        1182

-----
```



Below are the performance metrics on test data for random forest model with smote data.

Test Accuracy Score for model RandomForestClassifier() is 0.9701280227596017

```
-----Classification Report - Test Data-----
              precision    recall  f1-score   support

     0       0.97         0.97         0.97        1055
     1       0.97         0.97         0.97        1054

 accuracy          0.97         0.97         0.97        2109
 macro avg         0.97         0.97         0.97        2109
 weighted avg      0.97         0.97         0.97        2109

-----
```

Below are the comparison of various evaluation metrics in tabular format for both model.

With and without smote.

	Accuracy	Precision	Recall	F1	AUC
RandomForest_Train	0.981242	0.924603	0.899614	0.911937	0.99399
RandomForest_Test	0.974619	0.907563	0.850394	0.878049	0.985857

	Accuracy	Precision	Recall	F1	AUC
RF_With_Smote_Train	0.979678	0.98535	0.973844	0.979563	0.998354
RF_With_Smote_Test	0.970128	0.97056	0.969639	0.9701	0.996038

We can see in the above comparison that even though Random Forest without SMOTE has higher accuracy, but Random Forest with SMOTE surpasses the other model significantly in terms of Precision, Recall, F1 score as well as AUC score.

Also there is not a significant difference in the accuracies of both the models.

Hence we can say Random Forest with SMOTE data is the better of the two models with 97.01% accuracy, 97.05% precision, 96.96% recall and a recall and F1 score of 0.9701 and 0.996038 respectively.

## 1.10 Build a LDA Model on Train Dataset. Also showcase your model building approach.

we have created two models using LDA, one without SMOTE data and one with SMOTE data. We did not scale the data for LDA as it finds its coefficients using the variation between the classes, hence scaling doesn't matter.

We also used GridSearchCV for hyper parameter tuning. PFB the grid which was used. Same grid was used for both the models i.e. with and without smote to maintain uniformity.

```
param_grid_lda = {'solver': ['svd', 'lsqr', 'eigen'],
                  'shrinkage': ['auto', 'none', 'default'],
                  'tol': [0.000001, 0.0000001, 0.000000001]}
```

A custom function named `apply_evl` which takes below arguments was created. This function is able to handle modelling for different type of models and returns the performance metrics as the output. The same function has been used to build and evaluate the different models here.

Below are the performance metrics on train data for LDA model without smote data.

```
LDA
-----Best Parameters-----
{'shrinkage': 'auto', 'solver': 'lsqr', 'tol': 1e-06}

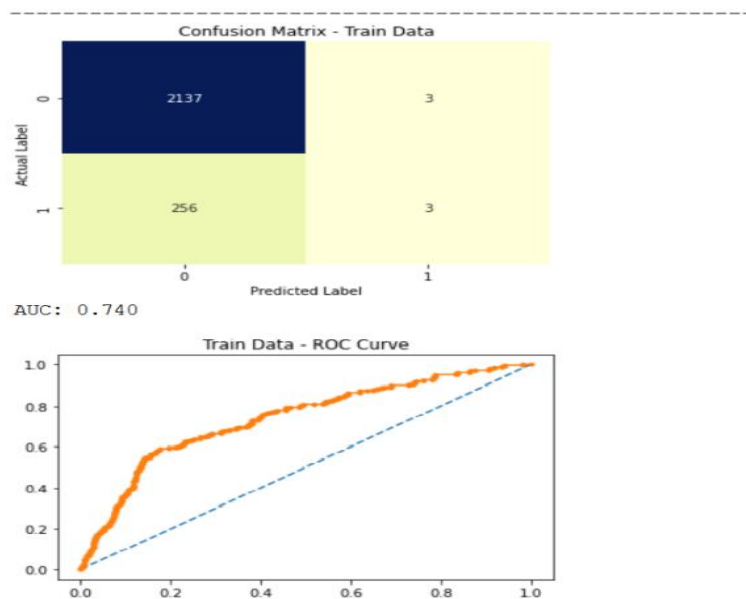
-----Best Model Params-----
LinearDiscriminantAnalysis(shrinkage='auto', solver='lsqr', tol=1e-06)

Train Accuracy Score for model LinearDiscriminantAnalysis() is 0.0920303493122134
-----Classification Report - Train Data-----
      precision    recall  f1-score   support

     0       0.89       1.00       0.94       2140
     1       0.50       0.01       0.02        259

 accuracy          0.89          2399
  macro avg          0.70          2399
 weighted avg          0.85          2399
```





Below are the performance metrics on train data for LDA model with smote data.

LDA\_With\_Smote

-----Best Parameters-----  
{'shrinkage': 'auto', 'solver': 'lsqr', 'tol': 1e-06}

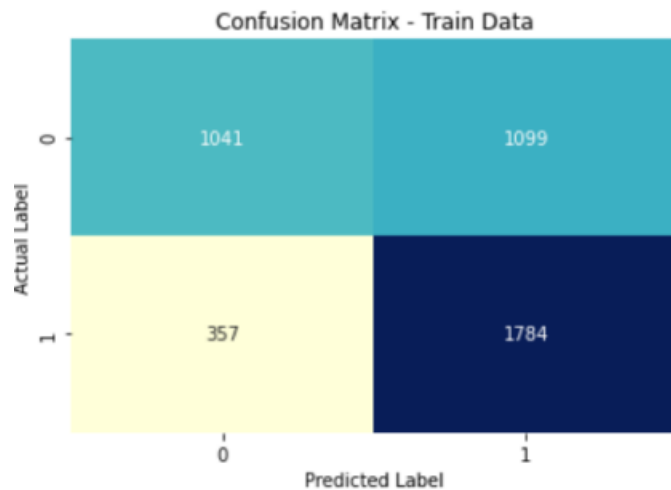
-----Best Model Params-----  
LinearDiscriminantAnalysis(shrinkage='auto', solver='lsqr', tol=1e-06)

Train Accuracy Score for model LinearDiscriminantAnalysis() is 0.6598925484699837

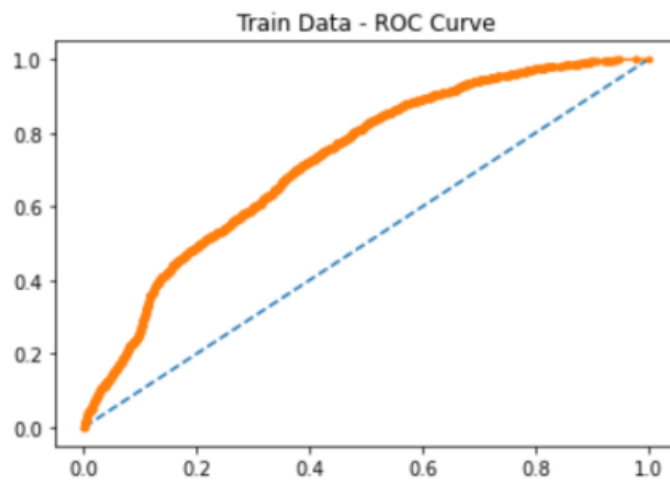
-----Classification Report - Train Data-----

	precision	recall	f1-score	support
0	0.74	0.49	0.59	2140
1	0.62	0.83	0.71	2141
accuracy			0.66	4281
macro avg	0.68	0.66	0.65	4281
weighted avg	0.68	0.66	0.65	4281

-----



AUC: 0.725

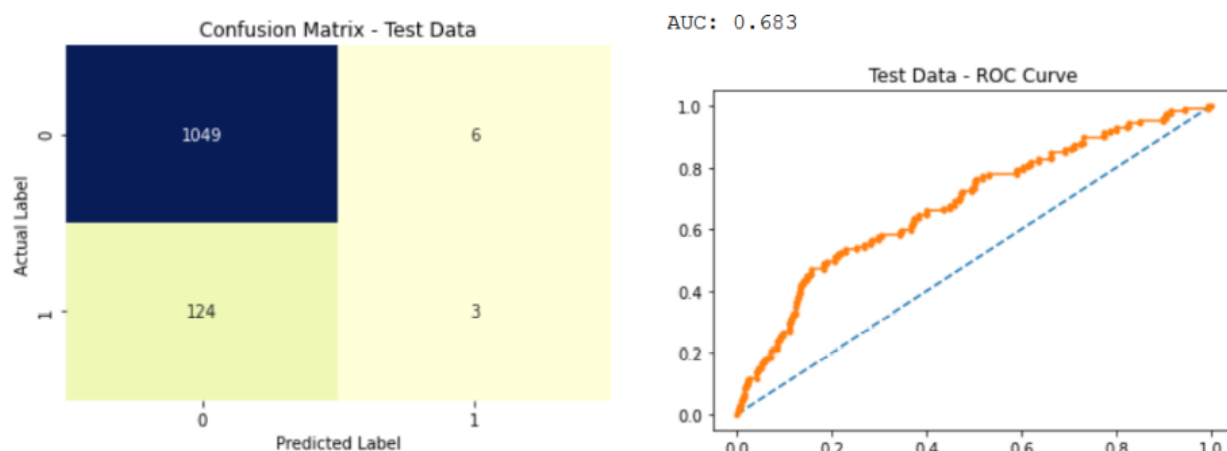


## 1.11 Validate the LDA Model on test Dataset and state the performance matrices. Also state interpretation from the model

Both the LDA models were then evaluated on the test data. Below are the results for test data for LDA model without smote data

Test Accuracy Score for model LinearDiscriminantAnalysis() is 0.8900169204737732

-----Classification Report - Test Data-----				
	precision	recall	f1-score	support
0	0.89	0.99	0.94	1055
1	0.33	0.02	0.04	127
accuracy			0.89	1182
macro avg	0.61	0.51	0.49	1182
weighted avg	0.83	0.89	0.85	1182

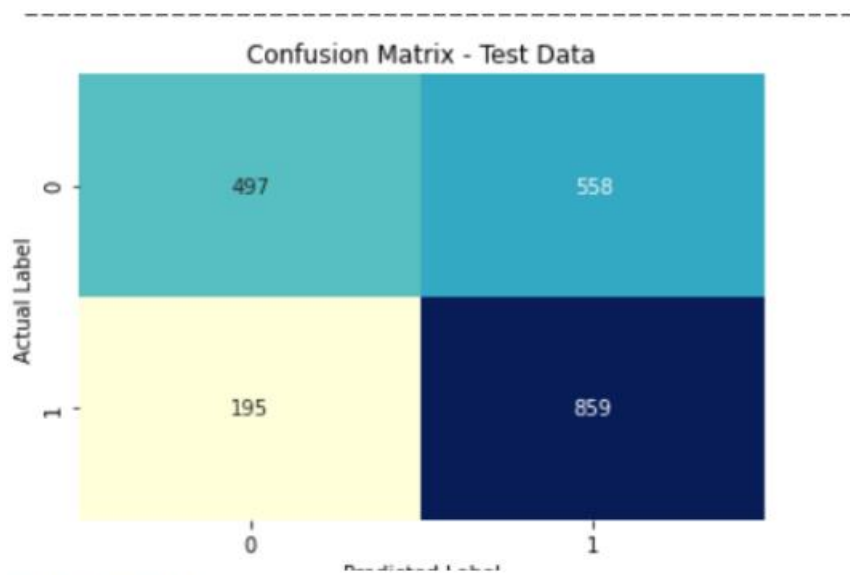


Below are the performance metrics on test data for LDA model with smote data.

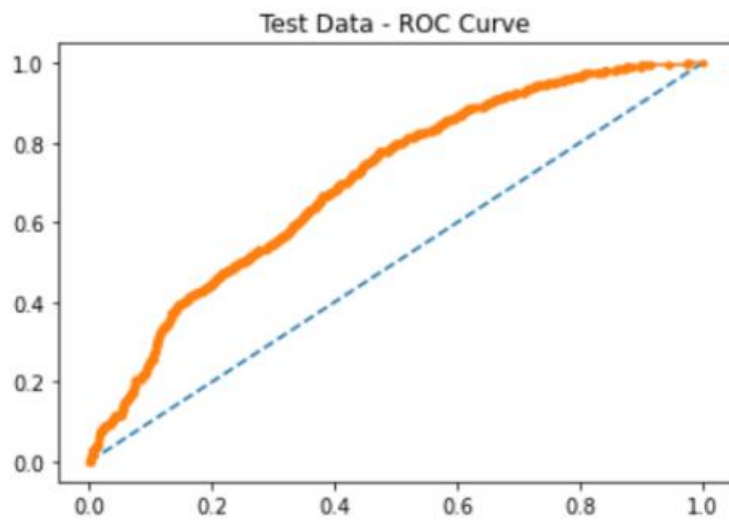
Test Accuracy Score for model LinearDiscriminantAnalysis() is 0.6429587482219061

-----Classification Report - Test Data-----				
	precision	recall	f1-score	support
0	0.72	0.47	0.57	1055
1	0.61	0.81	0.70	1054
accuracy			0.64	2109
macro avg	0.66	0.64	0.63	2109
weighted avg	0.66	0.64	0.63	2109

-----



AUC: 0.703



Below are the comparison of various evaluation metrics in tabular format for both the models i.e. with and without smote.

	Accuracy	Precision	Recall	F1	AUC
LDA_Train	0.892038	0.5	0.011583	0.0226415	0.740082
LDA_Test	0.890017	0.333333	0.023622	0.0441176	0.683203

	Accuracy	Precision	Recall	F1	AUC
LDA_With_Smote_Train	0.659893	0.6188	0.833255	0.710191	0.724862
LDA_With_Smote_Test	0.642959	0.60621	0.814991	0.695265	0.702944

Comparing the two models, we can see LDA model without SMOTE data has higher accuracy on test data. However it has really bad precision, recall and F1 scores compared to the other model.

On purely accuracy perspective model without smote data performs better. But in real world scenario model with smote will perform better given better recall and precision values.

## 1.12 Compare the performances of Logistics, Radom Forest and LDA models (include ROC Curve).

We made following models as part of this exercise. For Random Forest and LDA scaling was not taken into account as these models are not impacted by varying magnitudes of the variables.

Logistic regression using statsModel without SMOTE

Logistic regression using statsModel with SMOTE

Logistic Regression using sklearn With Unscaled Variables Without SMOTE

Logistic Regression using sklearn With Unscaled Variables Without SMOTE.

Logistic Regression using sklearn With Unscaled Variables With SMOTE

Logistic Regression using sklearn With Scaled Variables With SMOTE

LDA using sklearn without SMOTE

LDA using sklearn with SMOTE

Random Forest using sklearn without SMOTE

Random Forest using sklearn with SMOTE.

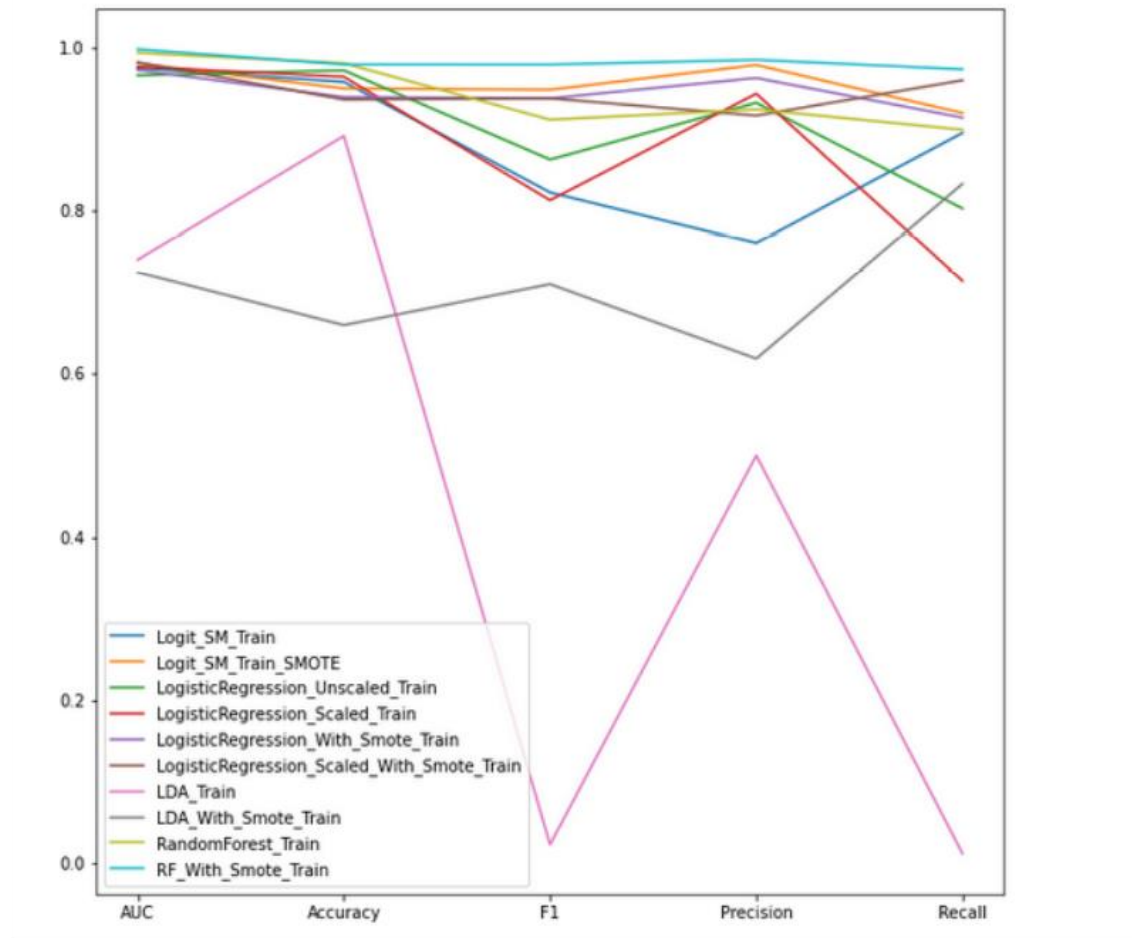
Below is the comparison chart for all the models.

	Accuracy	Precision	Recall	F1	AUC
RandomForest_Train	0.981242	0.924603	0.899614	0.911937	0.99399
RF_With_Smote_Train	0.979678	0.98535	0.973844	0.979563	0.998354
RandomForest_Test	0.974619	0.907563	0.850394	0.878049	0.985857
LogisticRegression_Unscaled_Train	0.972489	0.932735	0.803089	0.863071	0.966442
RF_With_Smote_Test	0.970128	0.97056	0.969639	0.9701	0.996038
LogisticRegression_Scaled_Train	0.964569	0.943878	0.714286	0.813187	0.97694
LogisticRegression_Unscaled_Test	0.959391	0.869159	0.732283	0.794872	0.945725
Logit_SM_Train	0.958316	0.760656	0.895753	0.822695	0.975275
Logit_SM_Test	0.952623	0.726115	0.897638	0.802817	0.956256
LogisticRegression_Scaled_Test	0.951777	0.830189	0.692913	0.755365	0.945098
Logit_SM_Train_SMOTE	0.950245	0.979125	0.920131	0.948712	0.981177
Logit_SM_Test_SMOTE	0.945472	0.962562	0.926945	0.944418	0.974799
LogisticRegression_With_Smote_Train	0.939734	0.963109	0.914526	0.938189	0.972807
LogisticRegression_Scaled_With_Smote_Train	0.936697	0.917038	0.960299	0.93817	0.982349
LogisticRegression_With_Smote_Test	0.935514	0.94305	0.926945	0.934928	0.968668
LogisticRegression_Scaled_With_Smote_Test	0.928876	0.902135	0.962049	0.931129	0.97516
LDA_Train	0.892038	0.5	0.011583	0.0226415	0.740082
LDA_Test	0.890017	0.333333	0.023622	0.0441176	0.683203
LDA_With_Smote_Train	0.659893	0.6188	0.833255	0.710191	0.724862
LDA_With_Smote_Test	0.642959	0.60621	0.814991	0.695265	0.702944

Looking at the above comparison chart we can see Random Forest model has surpassed all other models in terms of accuracy. Both Random Forest with and without smote feature in the top 2 models.

While Random Forest was the best performing model, LDA models were the worst performers of all. LDA with or without were both really bad in terms of accuracies, LDA with smote being the worst of the lot.

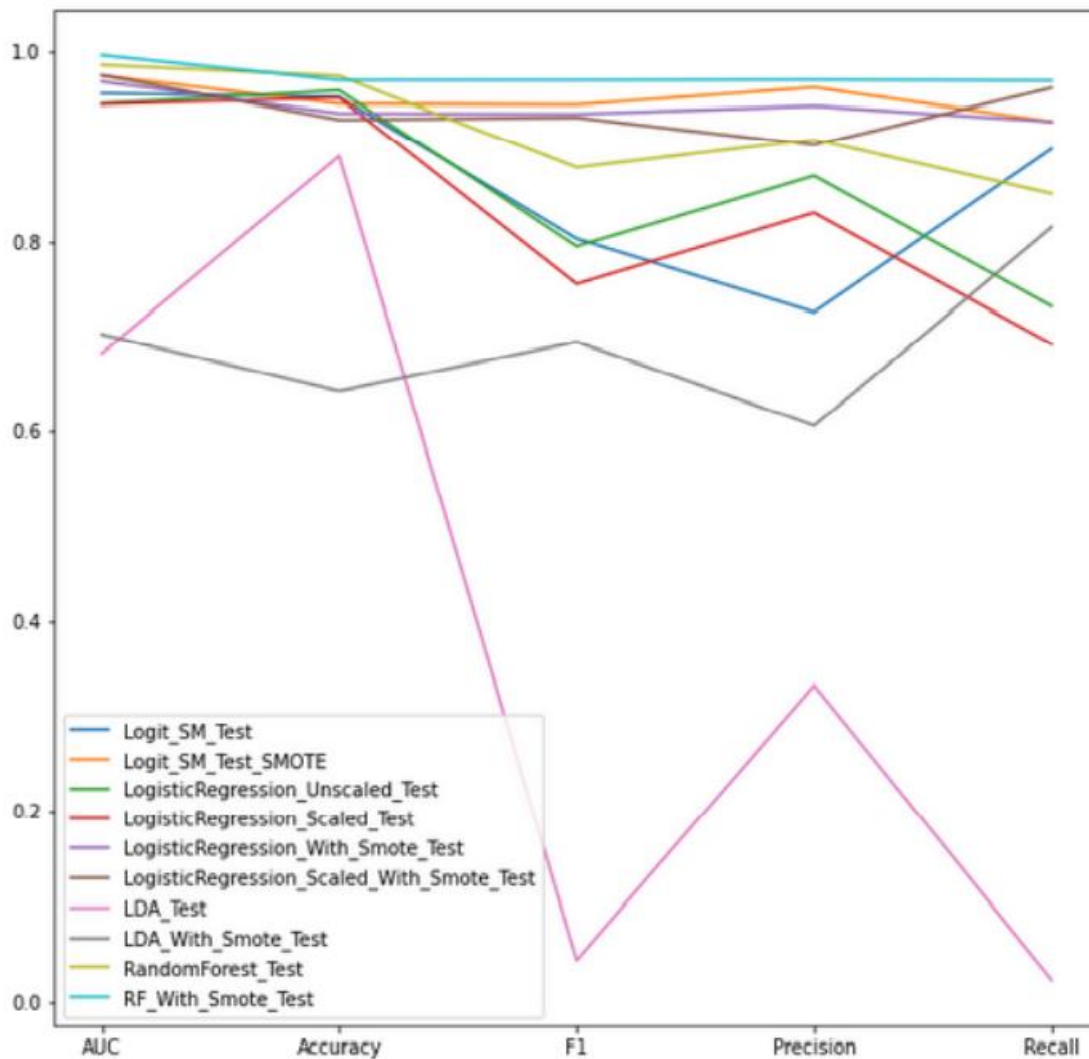
Below is a comparison of different evaluation metrics for all the models on the training data set.



Looking at the above chart we clearly see the blue line indicating "Random Forest With SMOTE" being on top of all other models, consistently for all the evaluation parameters.

While the pink line for LDA without smote is slightly higher than LDA with smote in terms of accuracy and auc score, it slips down significantly on all other parameters, making it the worst performing model of the lot.

Below is a comparison of different evaluation metrics for all the models on the test data set.



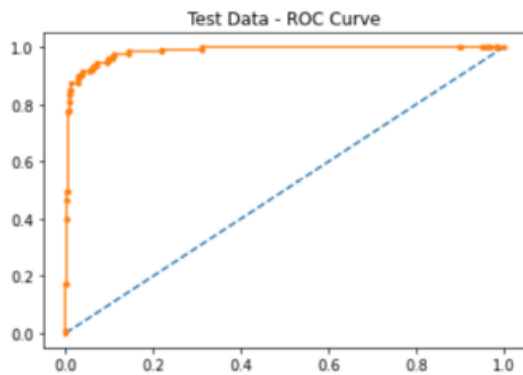
On the test data set, which is the deciding factor, here too we see Random Forest with SMOTE indicated by blue line performing the best on almost all the fronts. We can see the blue line dip only very slightly on accuracy compared to Random Forest without smote, on all other fronts the blue line stays on the top and is the clear winner amongst all the models.

Random Forest model with smote dataset is the best model amongst all the models with 97.01% accuracy, 97.05% precision, 96.96% recall and a recall and F1 score of 0.9701 and 0.996038 respectively.



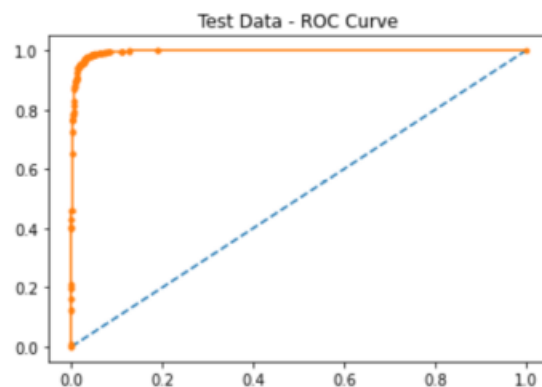
Also the ROC curves for Logit, LDA and random forest are plotted below. ROC curves for the best performing models on test data set have been plotted for comparison.

Comparing AUC scores too we see Random forest with smote data clearly wins compared to the others.



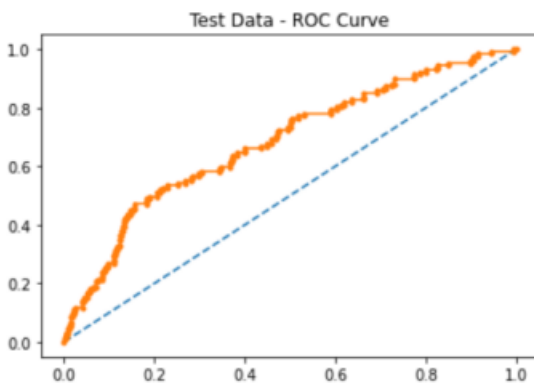
RF without SMOTE

AUC: 0.683

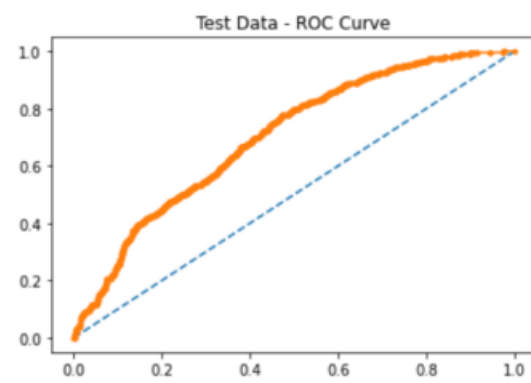


RF with SMOTE

AUC: 0.703

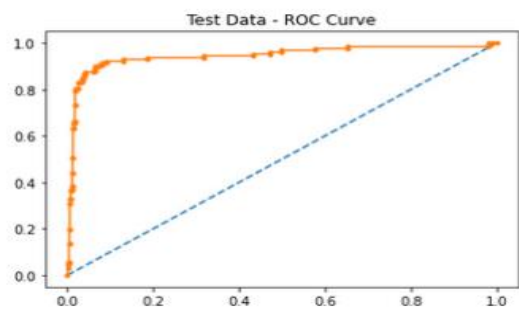


LDA Without SMOTE



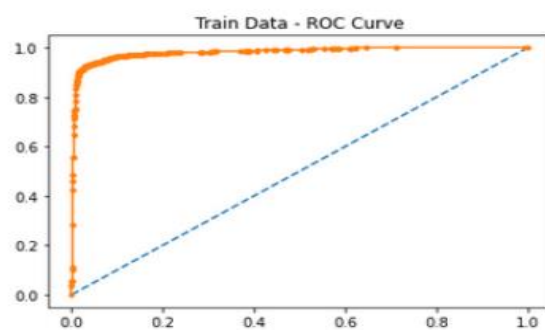
LDA with SMOTE

AUC: 0.945



Logit without SMOTE (Scaled)

AUC: 0.982



Logit with SMOTE (Scaled)

## 1.13 State Recommendations from the above models.

Using the information gained from above exercise, we can say Random Forest with smote data is the best model. We also looked at the coefficients derived from the best Logit model built using Stats model to derive some more insights.

	coef	std err	z	P> z	[0.025	0.975]
Intercept	-0.3898	0.148	-2.641	0.008	-0.679	-0.100
Book_Value_Unit_Curr	-0.1514	0.012	-12.174	0.000	-0.176	-0.127
CEPS_annualised_Unit_Curr	-0.0972	0.013	-7.649	0.000	-0.122	-0.072
Curr_Ratio_Latest	-0.4580	0.097	-4.733	0.000	-0.648	-0.268
Interest_Cover_Ratio_Latest	-0.0024	0.001	-2.999	0.003	-0.004	-0.001

From the above analysis, we can infer below business insights. Following things should be kept in mind while investing in these companies.

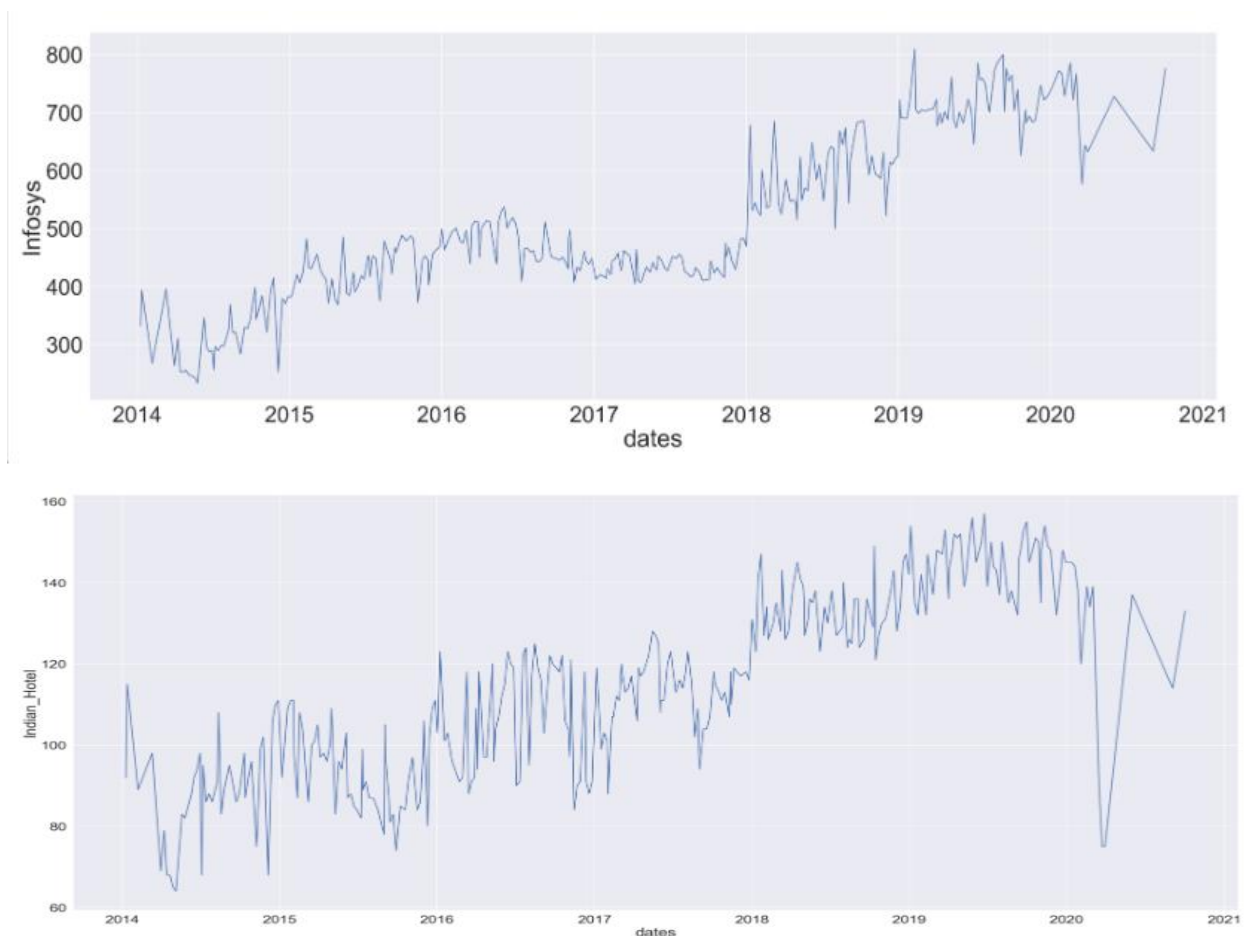
- 1) Lower the Book\_value\_unit\_curr i.e. Net assets, higher is the chance of a default, which would mean the net worth next year for this company is expected to be negative.
- 2) Lower the CEPS\_annualised\_Unit\_Curr i.e. Cash earning per share, higher is the change of a default.
- 3) Higher the Curr\_Ratio\_Latest, i.e. the companies ability to pay short term dues, lower are its chances of defaulting or having a negative net worth in the next year.
- 4) Higher the Interest\_Cover\_Ratio\_Latest lower the chances of default. Which means easier the company is able to pay the interest on its outstanding debt, lower are its chances to default.

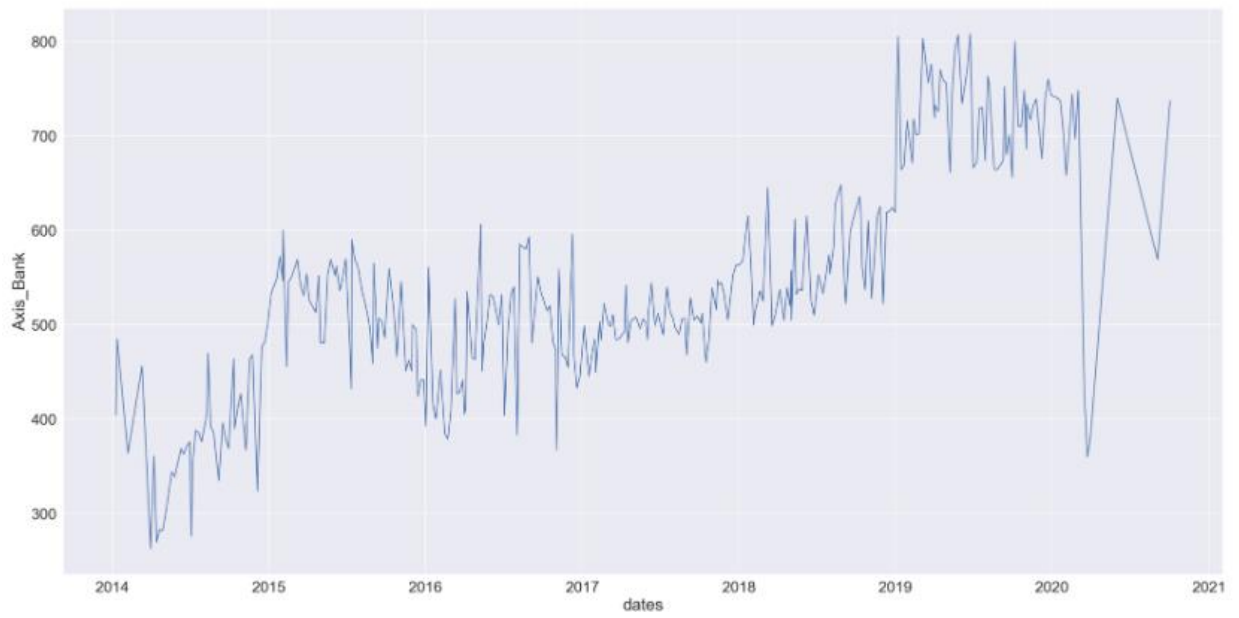
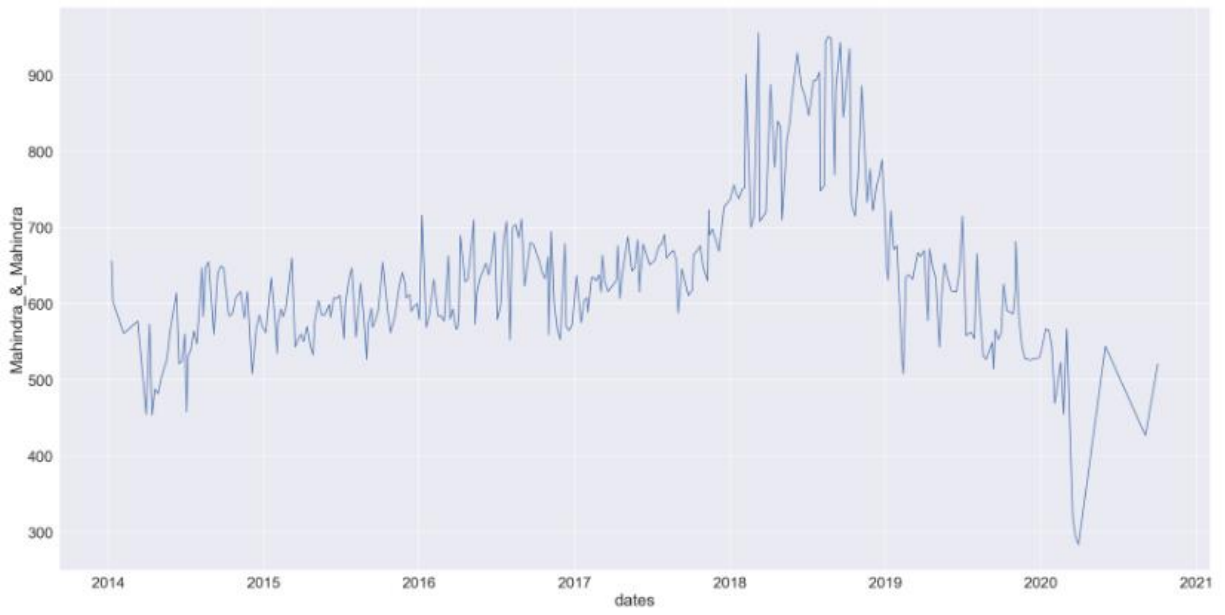
Curr\_Ratio\_Latest is most important criteria amongst the above parameters, while Interest\_Cover\_Ratio\_Latest is the least important when considering only these 4 parameters. However all these 4 parameters remain important compared to the other variables in the data set.

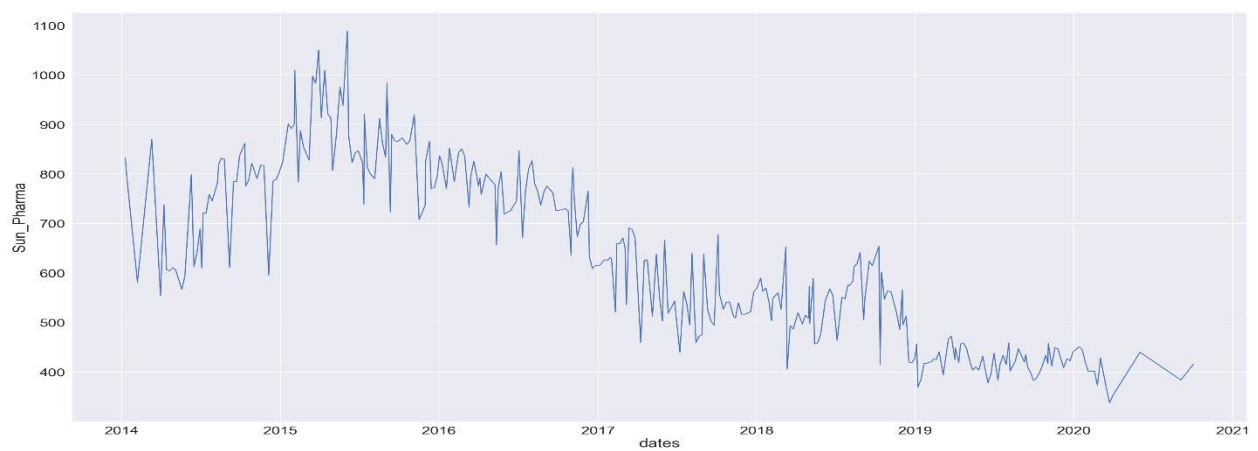
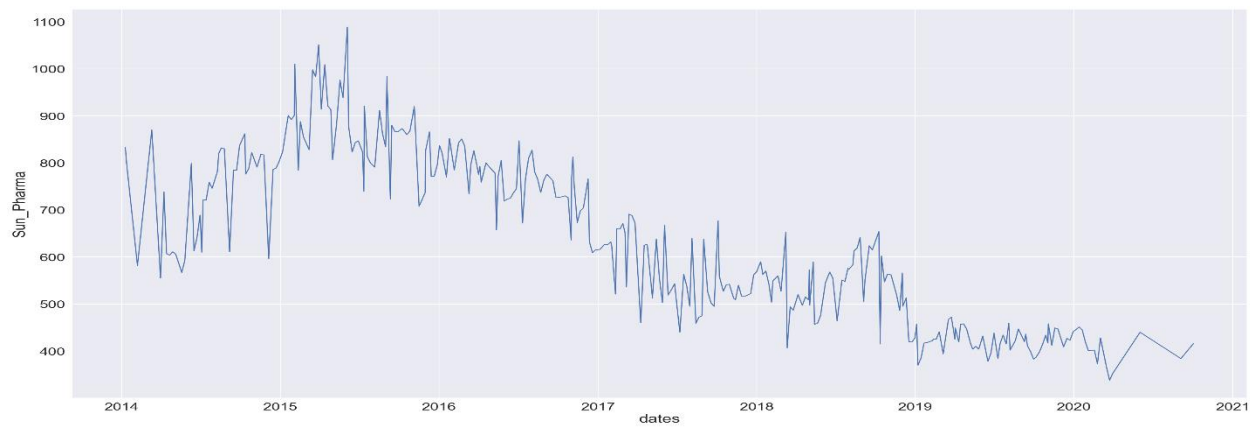
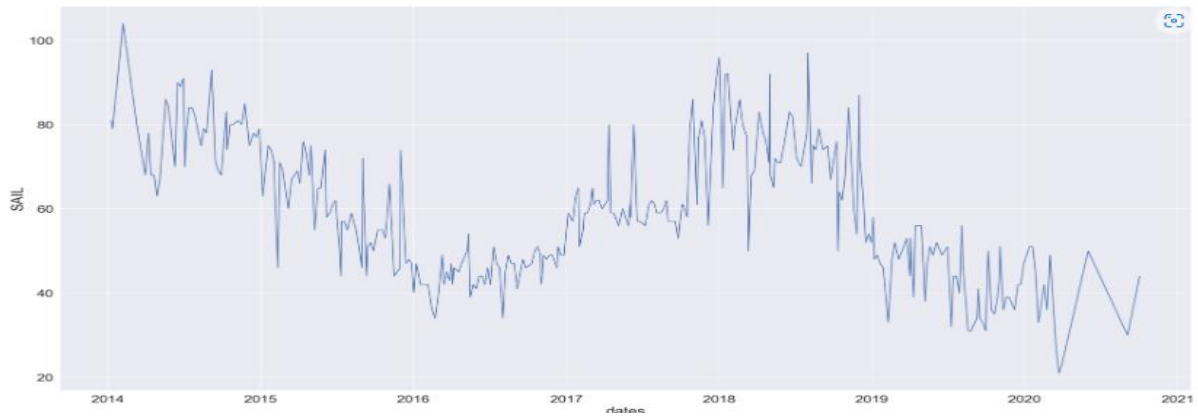
## PART – 2

The dataset contains 6 years of information(weekly stock information) on the stock prices of 10 different Indian Stocks. Calculate the mean and standard deviation on the stock returns and share insights. Please find attached the files to be referred.

### 2.1 Draw Stock Price Graph(Stock Price vs Time) for any 2 given stocks with inference.







## 2.2 Calculate Returns for all stocks with inference.

Returns for all the stocks i.e. difference of log of price at t and the log of price at t-1 are shown below.

	Infosys	Indian_Hotel	Mahindra_&_Mahindra	Axis_Bank	SAIL	Shree_Cement	Sun_Pharma	Jindal_Steel	Idea_Vodafone	Jet_Airways
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	-0.028873	-0.014599	0.008572	0.048247	0.028988	0.032831	0.094491	-0.085882	0.011976	0.088112
2	-0.011742	0.000000	-0.008772	-0.021979	-0.028988	-0.013888	-0.004930	0.000000	-0.011976	-0.078943
3	-0.003945	0.000000	0.072218	0.047025	0.000000	0.007583	-0.004955	-0.018084	0.000000	0.007117
4	0.011788	-0.045120	-0.012371	-0.003540	-0.078373	-0.019515	0.011523	-0.140857	-0.049393	-0.148848

Week over week returns for all the stocks has been given in the above figure. We have given only the top 5 rows due to space constraints here. For complete data refer jupyter notebook

## 2.3 Calculate Stock Means and Standard Deviation for all stocks with inference.

Stock means and standard deviations have been calculated below. The values have been sorted in descending order

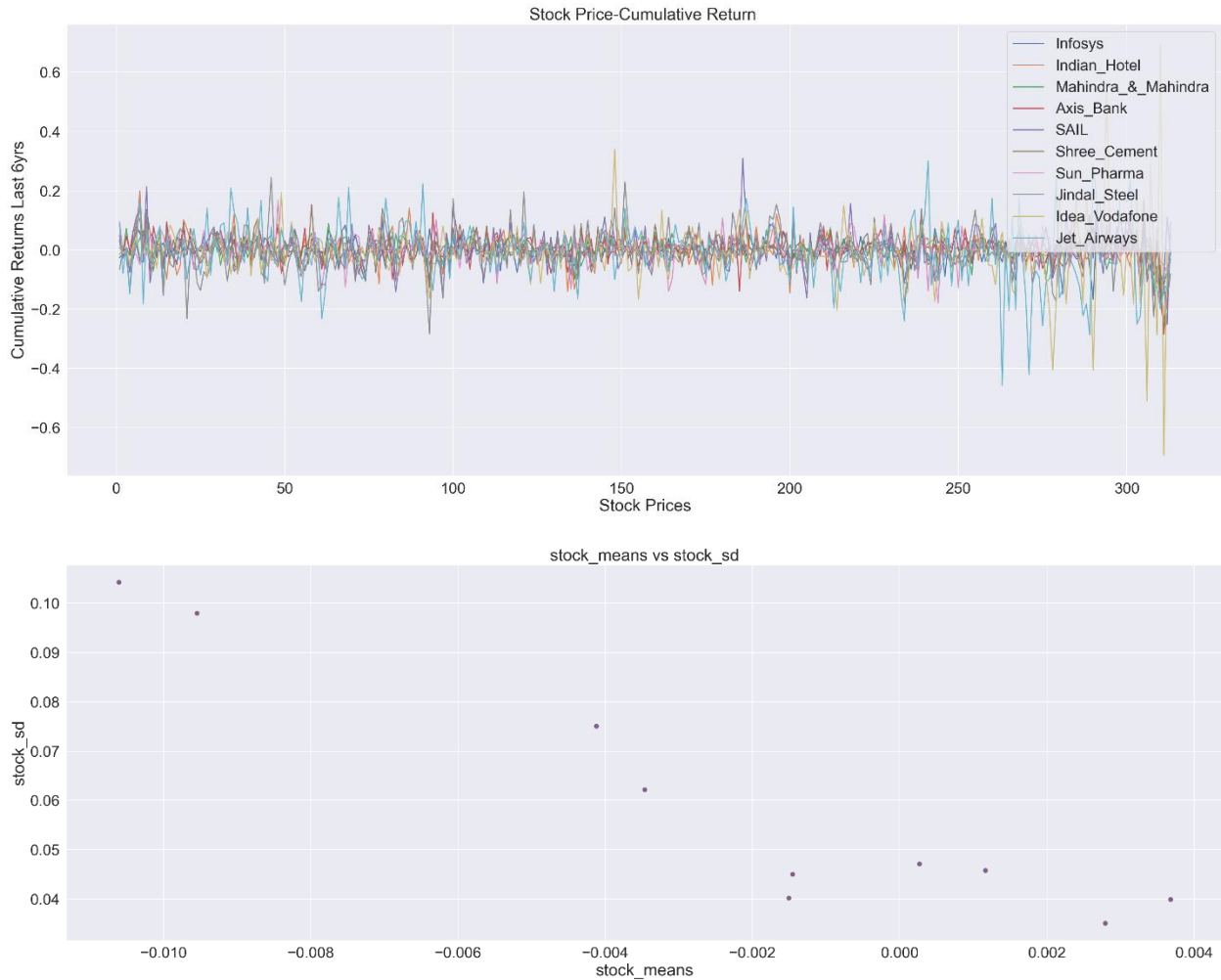
```
Shree_Cement      0.003681
Infosys           0.002794
Axis_Bank         0.001167
Indian_Hotel      0.000266
Sun_Pharma        -0.001455
Mahindra_&_Mahindra -0.001506
SAIL              -0.003463
Jindal_Steel      -0.004123
Jet_Airways       -0.009548
Idea_Vodafone     -0.010608
dtype: float64
```

Idea\_Vodafone has the lowest returns, while shree cements have the highest returns.

```
Idea_Vodafone      0.104315
Jet_Airways        0.097972
Jindal_Steel       0.075108
SAIL               0.062188
Indian_Hotel       0.047131
Axis_Bank          0.045828
Sun_Pharma         0.045033
Mahindra_&_Mahindra 0.040169
Shree_Cement       0.039917
Infosys            0.035070
dtype: float64
```

## Q.2.4 Draw a plot of Stock Means vs Standard Deviation

Below is the plot of stock means vs standard deviation.



Stocks higher up but on the far left indicate high volatility and low returns, while the stocks on the bottom right indicate low volatility and high returns.

This is a useful graph to find a balance between risk and reward when it comes to investing in different companies.



## Q2.5 Conclusion and Recommendations.

	Average	Volatility
Shree_Cement	0.003681	0.039917
Infosys	0.002794	0.035070
Axis_Bank	0.001167	0.045828
Indian_Hotel	0.000266	0.047131
Sun_Pharma	-0.001455	0.045033
Mahindra_&_Mahindra	-0.001506	0.040169
SAIL	-0.003463	0.062188
Jindal_Steel	-0.004123	0.075108
Jet_Airways	-0.009548	0.097972
Idea_Vodafone	-0.010608	0.104315

	Average	Volatility
Idea_Vodafone	-0.010608	0.104315
Jet_Airways	-0.009548	0.097972
Jindal_Steel	-0.004123	0.075108
SAIL	-0.003463	0.062188
Indian_Hotel	0.000266	0.047131
Axis_Bank	0.001167	0.045828
Sun_Pharma	-0.001455	0.045033
Mahindra_&_Mahindra	-0.001506	0.040169
Shree_Cement	0.003681	0.039917
Infosys	0.002794	0.035070

We can conclude by saying the below.

Stock with a lower mean & higher standard deviation do not play a role in a portfolio that has competing stock with more returns & less risk.

Thus for the data we have here, we are only left few stocks:

One with highest return and lowest risk &

- One with lowest risk and highest return

Therefore from pure Returns perspective, Shree\_Cement followed by Infosys & Axis\_Bank looks good in this dataset.

From pure Risk perspective (as measured by standard deviation), Infosys followed by Shree\_Cement & Mahindra\_&\_Mahindra looks good in this dataset.

We would recommend using the stock means vs standard deviation plot to assess the risk to reward ratio. More volatile stock might give short term gains but might not be a good investment in long term. Whereas a low volatile stock might not be a good investment in short term, but might give a good return in long term.

Hence based on the type of investment that one is looking for, an inference should be made from the above mentioned plot.