Financial Risk Analytics
Business Report

By: Ambrish Verma

Date: 5-May-24

Contents

Part A:

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 interest 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.

Dependent variable - No need to create any new variable, as the 'Default' variable is already provided in the dataset, which can be considered as the dependent variable.

Test Train Split - Split the data into train and test datasets in the ratio of 67:33 and use a random state of 42 (*random_state=42*). Model building is to be done on the train dataset and model validation is to be done on the test dataset.

1. Outlier Treatment
2. Missing Value Treatment
3. Univariate (4 marks) & Bivariate (6 marks) analysis with proper interpretation. (You may choose to include only those variables which were significant in the model building)
4. Train Test Split
5. Build Logistic Regression Model (using statsmodels library) on most important variables on train dataset and choose the optimum cut-off. Also showcase your model building approach
6. Validate the Model on Test Dataset and state the performance metrics. Also state interpretation from the model
7. Build a Random Forest Model on Train Dataset. Also showcase your model building approach
8. Validate the Random Forest Model on test Dataset and state the performance metrics. Also state interpretation from the model
9. Build a LDA Model on Train Dataset. Also showcase your model building approach
10. Validate the LDA Model on test Dataset and state the performance metrics. Also state interpretation from the model
11. Compare the performances of Logistic Regression, Random Forest, and LDA models (include ROC curve)
12. Conclusions and Recommendations

Part B

Problem Statement:

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. You are expected to do the Market Risk Analysis using Python.

1. Draw Stock Price Graph(Stock Price vs Time) for any 2 given stocks with inference
2. Calculate Returns for all stocks with inference
3. Calculate Stock Means and Standard Deviation for all stocks with inference
4. Draw a plot of Stock Means vs Standard Deviation and state your inference
5. Conclusions and Recommendations

Part A

Before proceeding with the problems, following change was performed:

- In all the columns where the names were starting with '_', the under-score was removed.

1. Outlier Treatment

Solution for problem 1 and problem 2 are clubbed together and mentioned as part of Problem 2.

2. Missing Value Treatment

Box-plot of the columns with the original(unchanged) dataset:

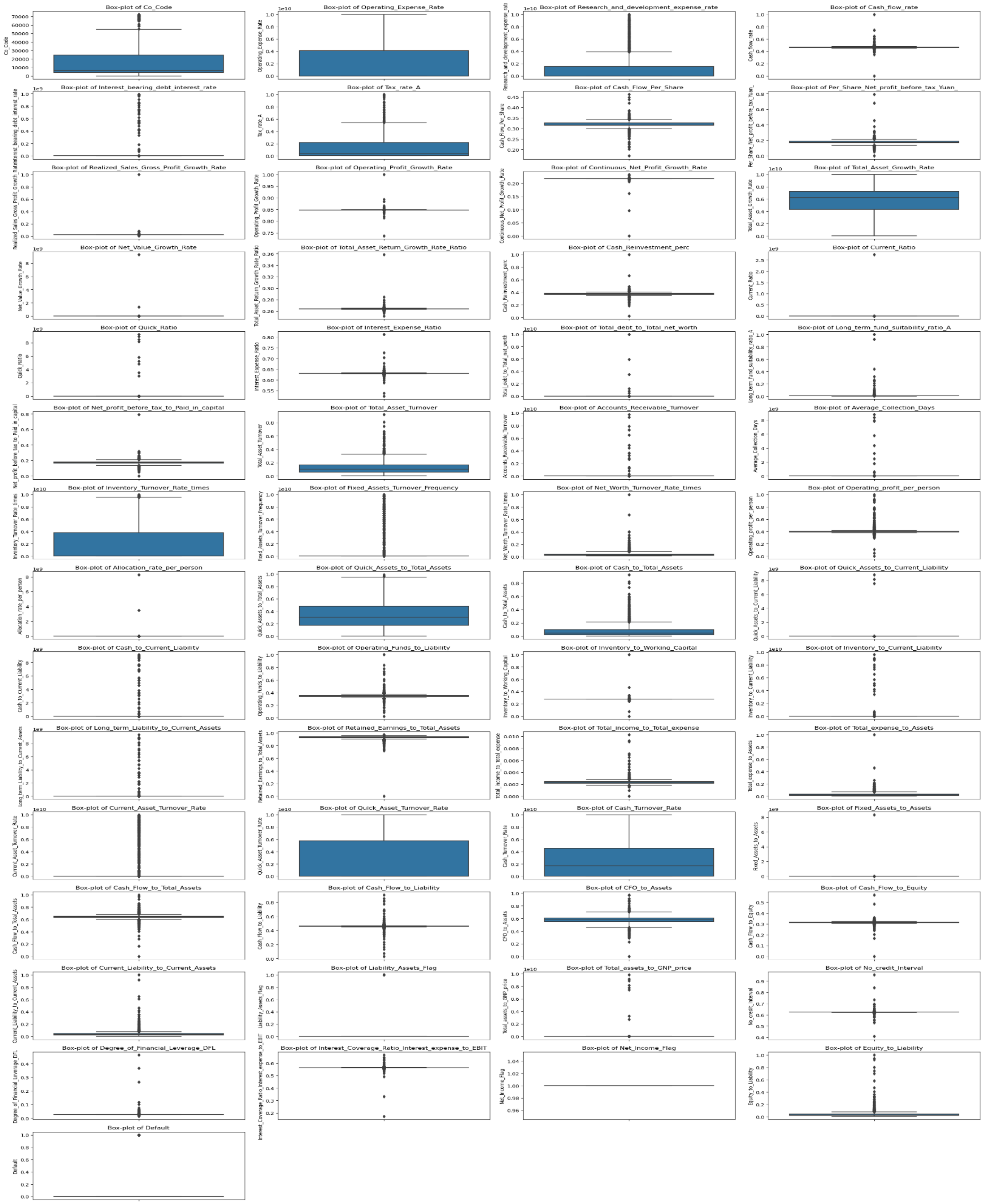


Figure 1 Box-plot before outlier treatment

- It is evident that there is a significant number of outliers present in the dataset. Moreover, the value of the outliers is very high compared to rest of the dataset. Considering the sensitivity of the data and presence of default for a small fraction of data, outlier treatment has been performed only on the attributes values which are beyond the IQR (Inter Quantile Range of 10-90). Performing outlier treatment using IQR of 25-75(normal range) might be counter-productive in this case. Hence, the steps for outlier treatment being employed are mentioned below:
    - Remove the columns that may not be useful, are mostly constant, indicating they may not be of much use.
    - Change the outliers to the NaN values.
    - Split the data into train and test.
    - Since KNN Imputer needs data to be scaled, the dataset is scaled using standard scalar on train data and then is applied onto the test data.
    - Perform the KNN imputer to impute all the null values. This serves both the Problems 1(Outlier treatment) and Problem 2(Null values treatment).
- Before outlier treatment and null values imputation,
    - total number of outliers = 2439
    - total number of nulls = 298
- After the conversion of outliers to NaN and performing KNN imputation,
    - Total number of nulls for train data = 0

- After outlier treatment and null values imputation, following are the boxplots for the attributes (that are not considered outright irrelevant):

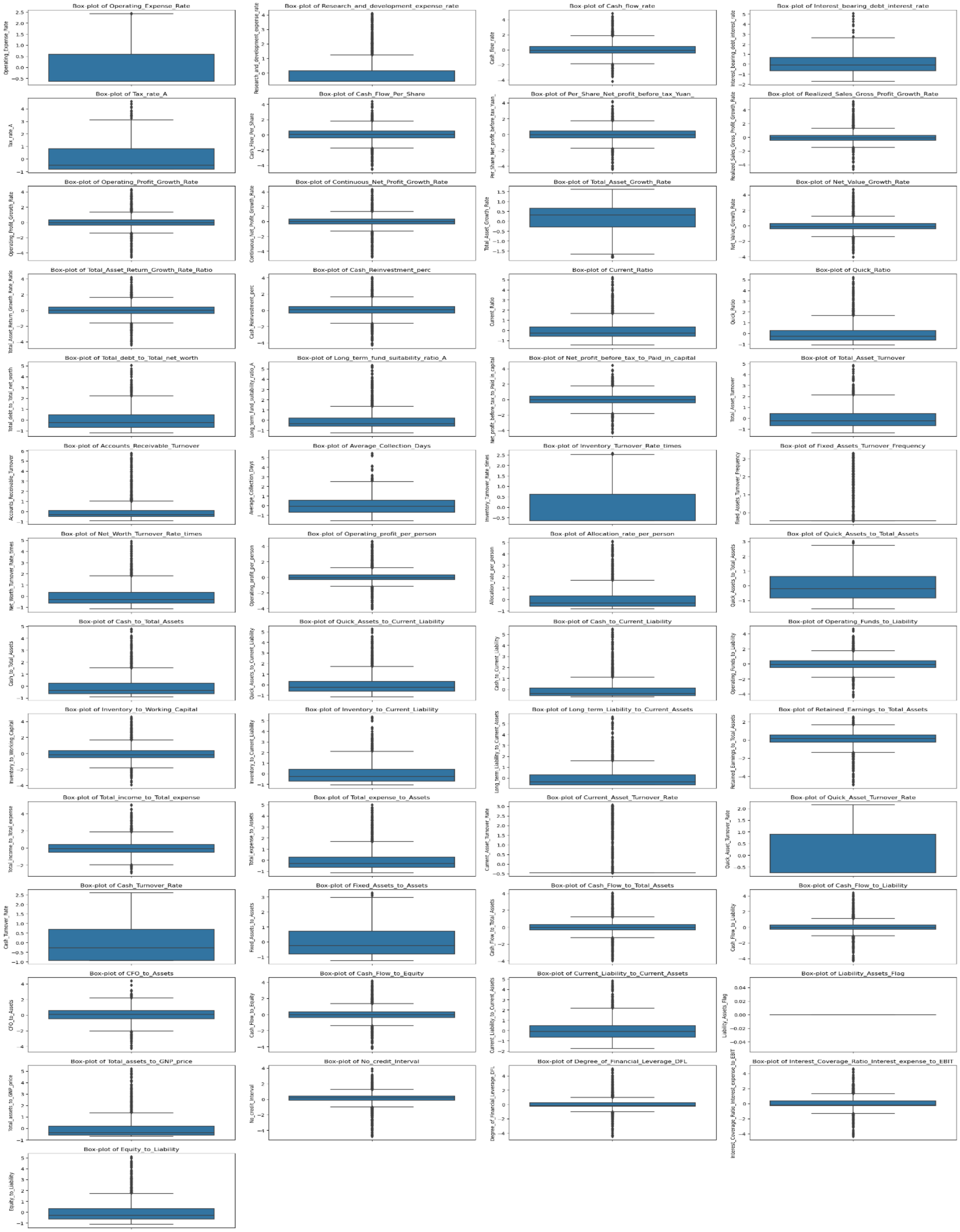


Figure 2 Box - plot after outlier treatment

3. Univariate (4 marks) & Bivariate (6 marks) analysis with proper interpretation. (You may choose to include only those variables which were significant in the model building)

- There is a total of 58 columns. Analyzing so many columns is difficult. Moreover, there is high likelihood of collinearity between attributes, some attributes being constant and not contributing in the decision making and non-numeric columns, the same are dropped using the below logics:
    - The attributes that are not numeric and are not categorical are dropped. Ex: Co_Name
    - ID attributes are dropped as they have not significance in the training the models. Ex: Co_Code
    - There is not much variation observed in the attributes. Ex: Net_Income_Flag, Liability_Assets_Flag, Interest_Expense_Ratio
    - Using the feature: Variance Inflation Factor, the attributes that have high degree of collinearity are identified iteratively and then dropped from the dataframe. Following is the list of attributes dropped and the VIF values calculated(with a VIF threshold = 5):
        - Per_Share_Net_profit_before_tax_Yuan_
        - Cash_Flow_to_Total_Assets
        - CFO_to_Assets
        - Quick_Assets_to_Current_Liability
        - Operating_Funds_to_Liability
        - Current_Ratio
        - Net_Worth_Turnover_Rate_times
        - Cash_Flow_Per_Share
    - Based on the VIF values, 6 attributes were identified based on the least VIF values. They are:
        - Interest_bearing_debt_interest_rate
        - Cash_Turnover_Rate
        - Research_and_development_expense_rate
        - Inventory_Turnover_Rate_times
        - Inventory_to_Working_Capital
        - Total_Asset_Growth_Rate

- Box-plot of significant attributes:

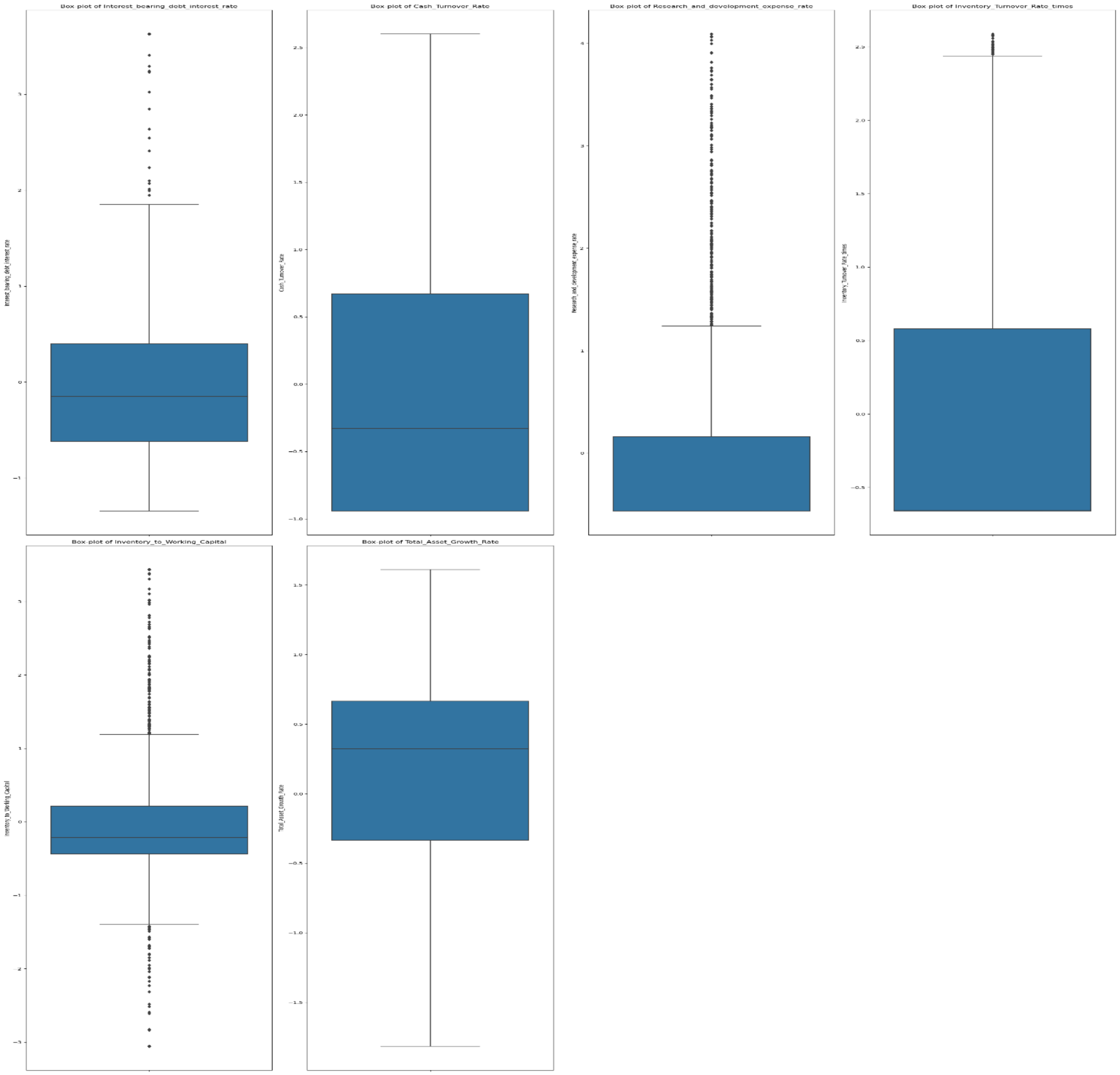


Figure 3 box-plot of significant attributes

Pairplot-plot of significant attributes:

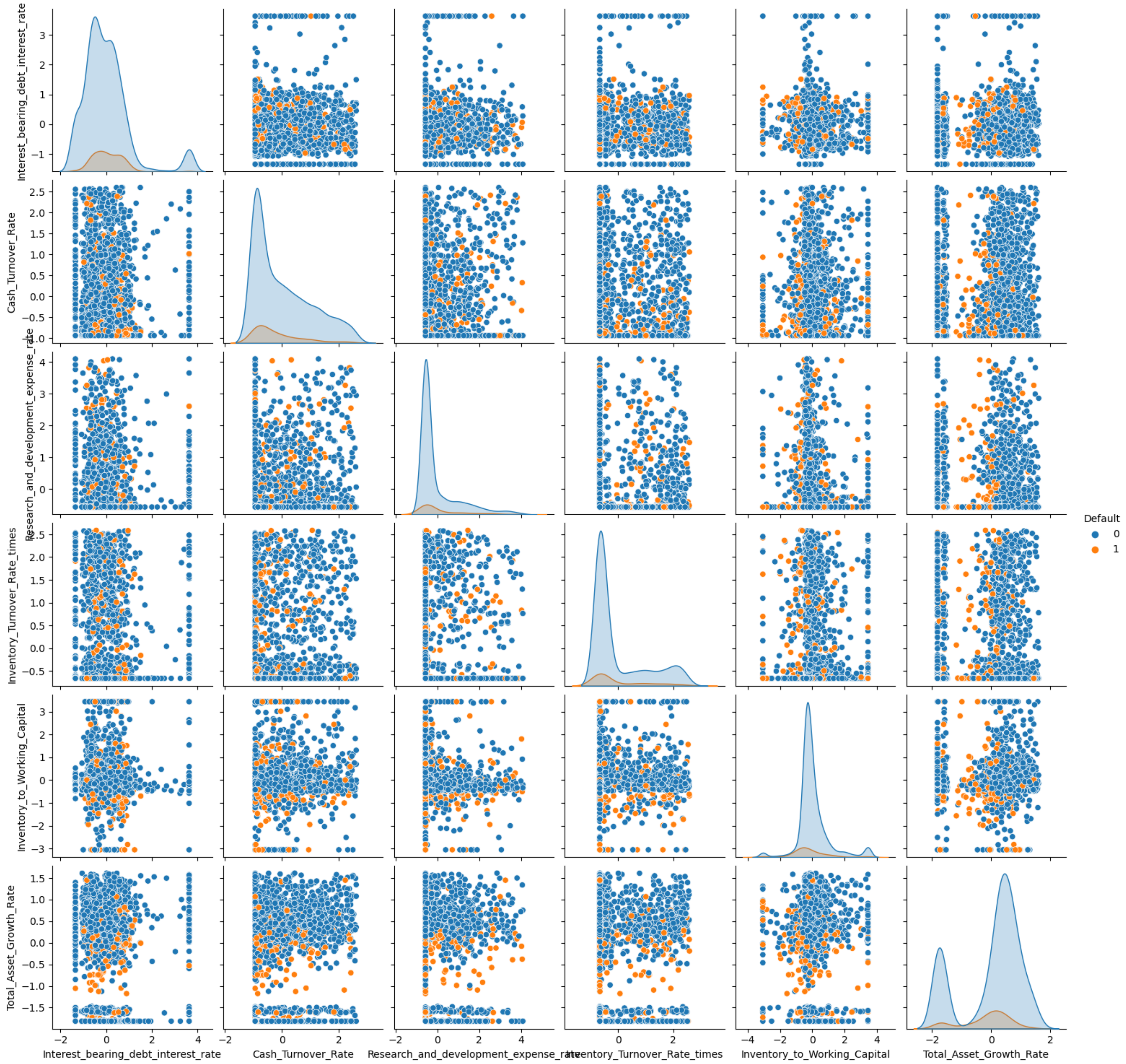


Figure 4 Pair-plot of significant attributes

- Correlation matrix:

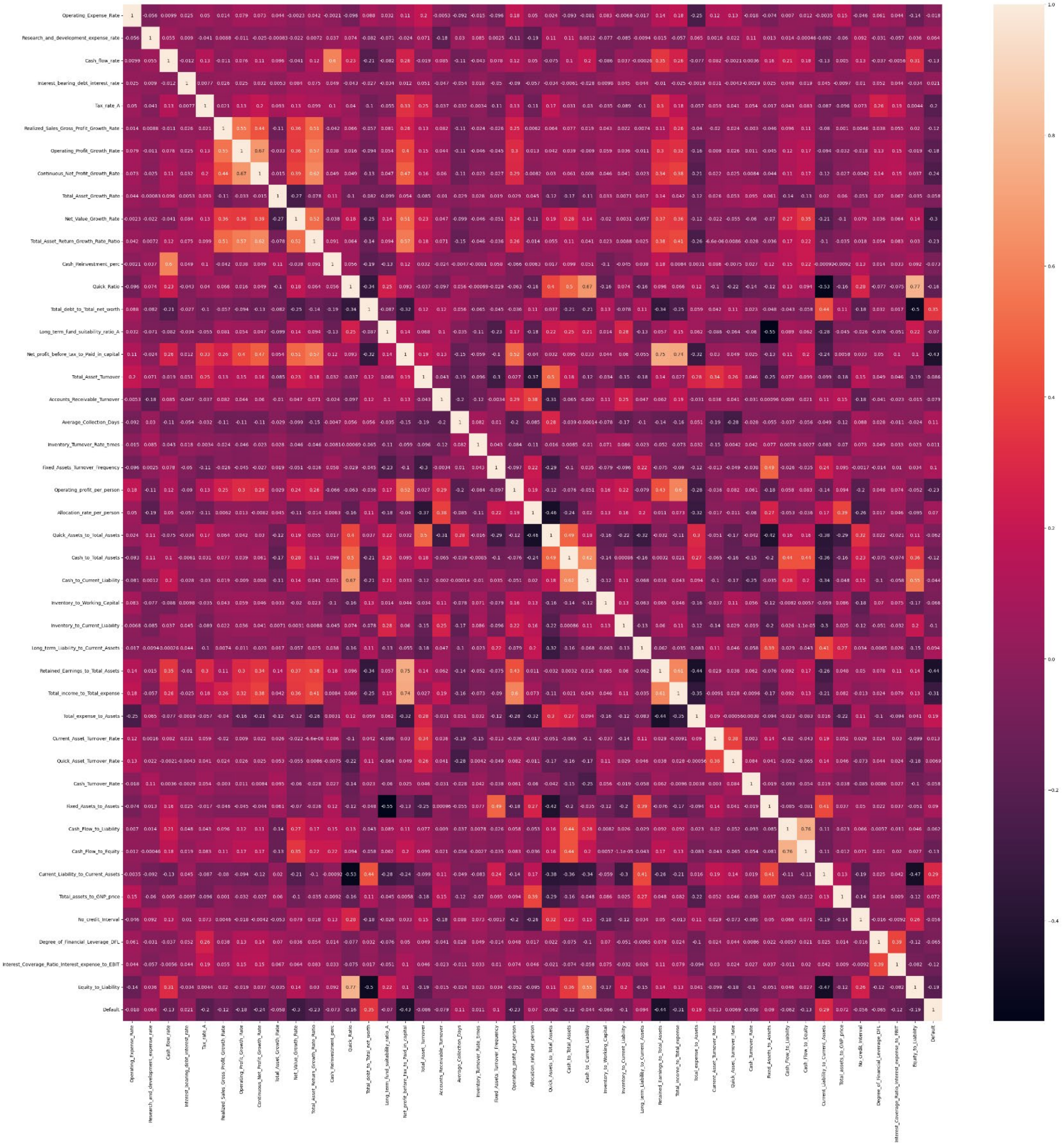


Figure 5 Correlation Matrix of attributes after dropping most insignificant attributes

- Interpretations from the above graphs:
  - The attributes ‘Retained_earnings_to_total_assets’, ‘Net_profit_before_tax_to_paid_in_capital’ are inversely proportional to Default. If either of these values, go low the probability of Default increases.
  - Same is applicable for the attributes to a lesser extent: Net_value_growth_rate, Total_income_to_total_expense. Hence, these attributes’s values also need to be checked to ensure the companies metrics on these do not fall below a threshold.
  - Total_debt_to_total_net_worth is directly proportional to the probability of Default. Hence, this values must be kept in check so tha total_debt as a proportion of total net worth does not exceed a limit set by the bank.

4. Train Test Split

- The dataset is split into Train and test data in a proportion such that the percentage of defaults is similar in both Train and test data sets.
- The ratio of Train and test data are 70:30.
- The Train Data is called ‘Train and Test data set is called ‘Test.
- The target variable: ‘Default’ has been taken out of the dataset and is stored in the variable: ‘y’ in the form of y_Train and y_Test

5. Build Logistic Regression Model (using statsmodels library) on most important variables on train dataset and choose the optimum cut-off. Also showcase your model building approach

Model Building using Logistic Regression for Probability of Default:

- The dataset provided had data quality enhanced by performing outlier treatment and imputation of null values. The data was scaled as well as the imputation technique utilized was KNN Imputer which requires data to be scaled.
- Before building the Logistic Regression model, the company data was split into Train and Test in the ratio of 70:30 and stratified split was performed to ensure the defaulter data is present in the same proportion in Test as well as Train dataset.
- The logistic regression model (called model_1) was built and its summary was evaluated in the summary. The P value of all the attributes was observed and the attributes for which the P value was < 0.05 were listed and used

to build another model(model_2) iteratively. Hence, to identify significant attributes, P<0.05 is taken up as the identifying parameter.

- The same approach was employed to build models successively till the P value for all the attributes was < 0.05.
- A total of 3 models were built using Logistic regression.
- In the first model, it was observed that multiple attributes had a P value of > 0.05, indicating that are insignificant for the model.
- Model 2 was built using only the following attributes: Research_and_development_expense_rate, Total_debt_to_Total_net_worth ,Long_term_fund_suitability_ratio_A ,Net_profit_before_tax_to_Paid_in_capital, Accounts_Receivable_Turnover,Retained_Earnings_to_Total_Assets ,Cash_Turnover_Rate ,Total_assets_to_GNP_price ,Degree_of_Financial_Leverage_DFL, Equity_to_Liability
- Even with Model 2, the attributes: Research_and_development_expense_rate, Long_term_fund_suitability_ratio_A, Degree_of_Financial_Leverage_DFL were found to have P >0.05. Hence Model 3 was built with the remaining columns: Total_debt_to_Total_net_worth, Net_profit_before_tax_to_Paid_in_capital, Accounts_Receivable_Turnover ,Retained_Earnings_to_Total_Assets ,Cash_Turnover_Rate, Total_assets_to_GNP_price, Equity_to_Liability
- In Model 3, there was no insignificant attribute identified. Hence proceeded with the Model 3.
- For model 3 training data, following was the summary:

• **Dep. Variable:**	Default	**No. Observations:**	1440
Model:	Logit	**Df Residuals:**	1432
Method:	MLE	**Df Model:**	7
Date:	Sun, 05 May 2024	**Pseudo R-squ.:**	0.3953
Time:	02:10:49	**Log-Likelihood:**	-296.15
converged:	True	**LL-Null:**	-489.71
Covariance Type:	nonrobust	**LLR p-value:**	1.378e-79

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	-3.4771	0.214	-16.214	0.000	-3.897	-3.057
Total_debt_to_Total_net_worth	0.3237	0.151	2.143	0.032	0.028	0.620
Net_profit_before_tax_to_Paid_in_capital	-1.1152	0.194	-5.753	0.000	-1.495	-0.735
Accounts_Receivable_Turnover	-0.5409	0.171	-3.171	0.002	-0.875	-0.207
Retained_Earnings_to_Total_Assets	-0.5452	0.130	-4.179	0.000	-0.801	-0.290

Cash_Turnover_Rate	-0.3027	0.128	-2.360	0.018	-0.554	-0.051
Total_assets_to_GNP_price	0.2393	0.090	2.671	0.008	0.064	0.415
Equity_to_Liability	-1.1641	0.377	-3.086	0.002	-1.903	-0.425

Table 1 Logistic Regression summary - train data

- To covert the result of Logistic regression from probability to a classification of 0 and 1, all the predicted probability values >=0.5 were treated as 1 and the rest were treated as 0.
- Following are the statistics for the probability threshold value of 0.5.
- Confusion matrix on the Training data:

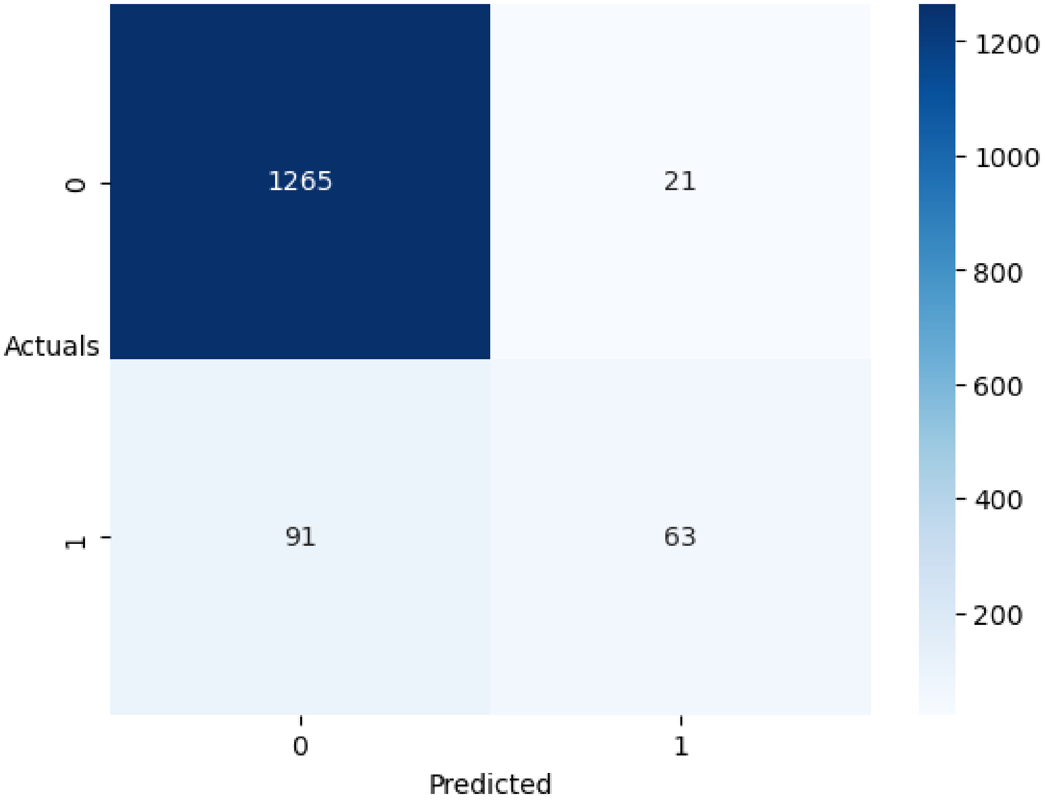


Figure 6 Confusion matric on train dataset

- Classification report:

```
              precision    recall  f1-score   support

           0       0.93      0.98      0.96      1286
           1       0.75      0.41      0.53       154

    accuracy                           0.92      1440
   macro avg       0.84      0.70      0.74      1440
weighted avg       0.91      0.92      0.91      1440
```

- By revising the probability threshold from 0.5 to 0.141, the recall value on Train data jumped to 78%.
- The confusion matrix after threshold revision:

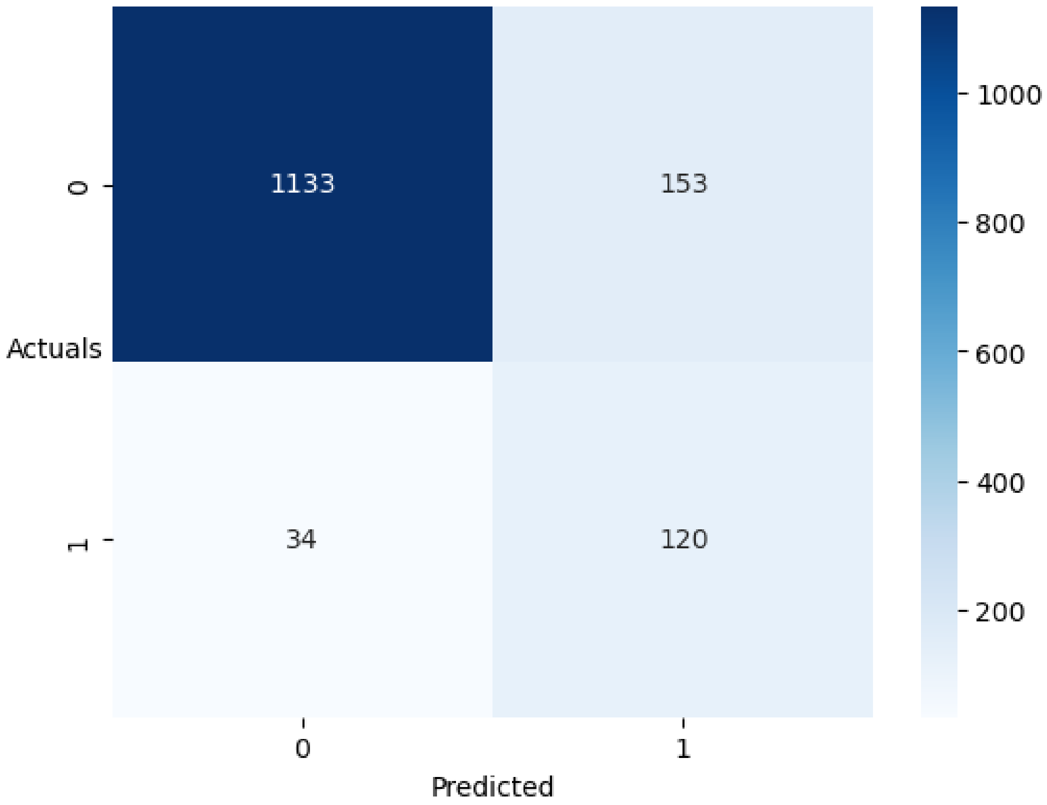


Figure 7 Confusion matric - Logistic Regression - Training - threshold = 0.141

- Classification report after threshold revision:

```
           0       0.97      0.88      0.92      1286
           1       0.44      0.78      0.56       154

    accuracy                           0.87      1440
   macro avg       0.71      0.83      0.74      1440
weighted avg       0.91      0.87      0.89      1440
```

6. Validate the Model on Test Dataset and state the performance metrics. Also state interpretation from the model

- For the current dataset, the precision of Default = 0 and recall of Default =1 is supposed to be high. The Model 3 was validated on the Test data set.
- The optimum probability threshold value in classifying dependent value of Default as 0 and 1, the optimum threshold value was calculated as 0.141. Before this threshold, the initial threshold was 0.5. Based on this threshold, following statistics were obtained:
- Confusion matrix(threshold = 0.5):

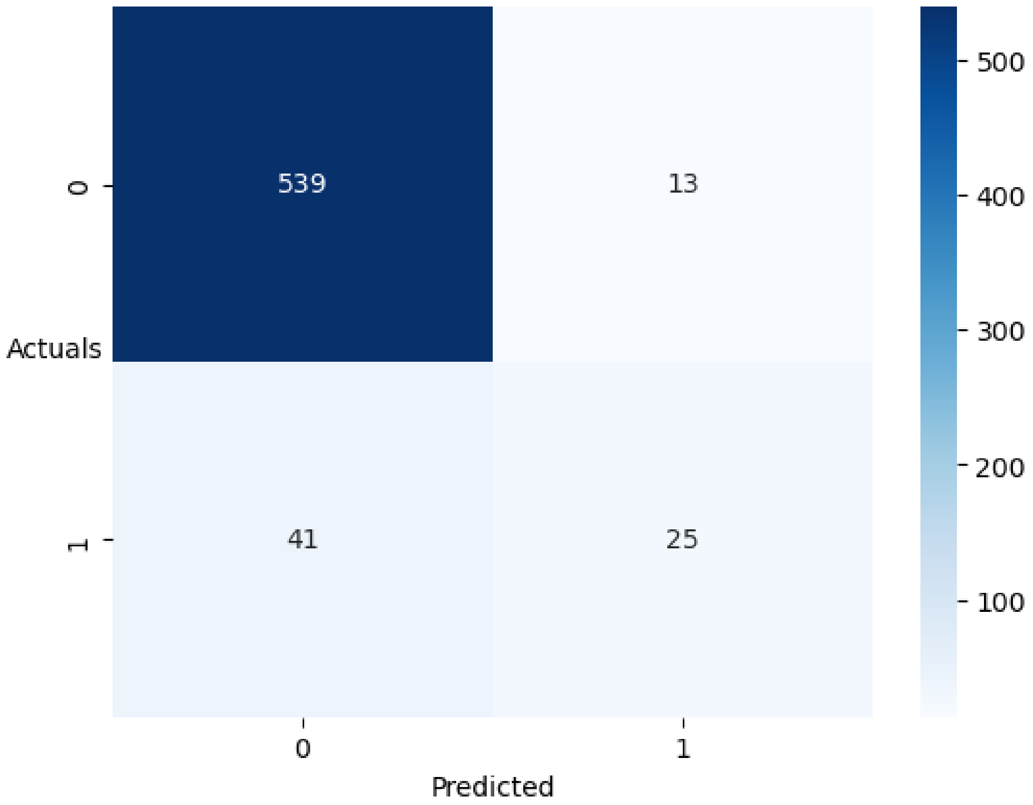


Figure 8 Confusion Matrix - Test dataset

- Correlation report (probability value threshold = 0.5):

```
              precision    recall  f1-score   support

           0       0.93      0.98      0.95       552
           1       0.66      0.38      0.48        66

    accuracy                           0.91       618
   macro avg       0.79      0.68      0.72       618
weighted avg       0.90      0.91      0.90       618
```

- The overall accuracy of the model 3 on Test data was found to be 91%.
- The precision of value, wherein the records were identified as actual default:total records predicted as default = 66%

- The recall value was found to be 38%. For less than half of the records, the model is predicting a default when the ration of defaults in the entire dataset = Total default / Total number of records = 220 / 2058 = 10.7%.
- With the revised threshold, following are the details:
- Confusion matrix(Threshold = 0.141):

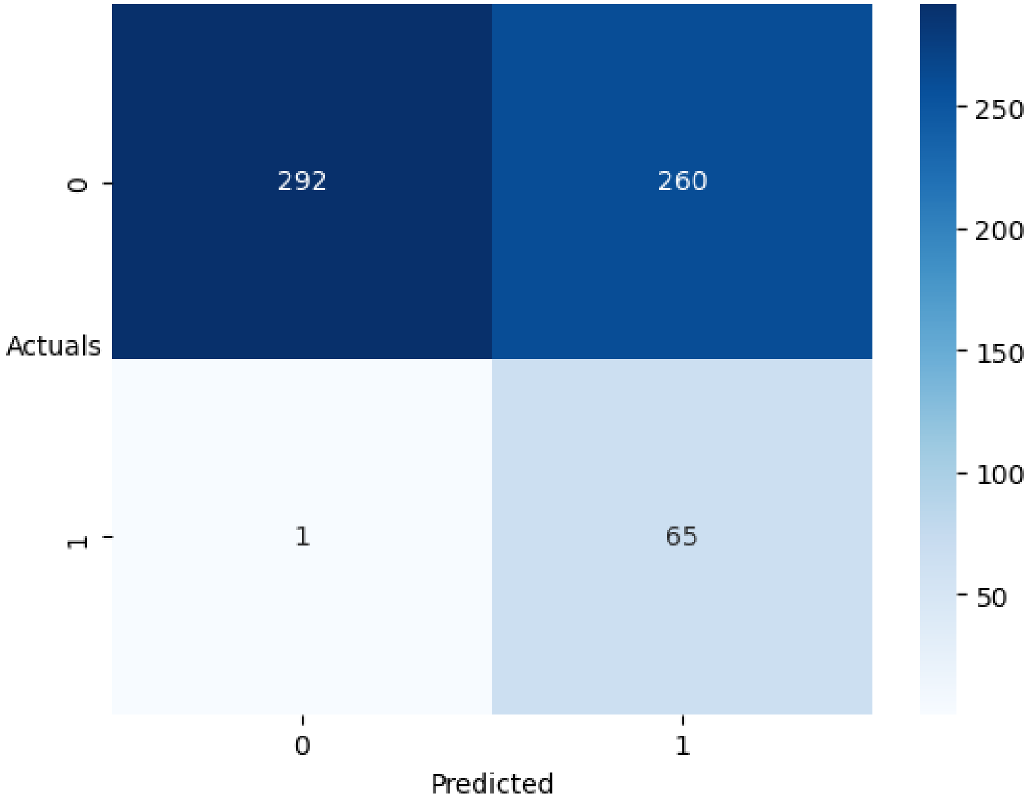


Figure 9 Confusion matrix - Logistic Regression - Test (threshold = 0.141)

- Classification report(Threshold = 0.141):

```
              precision    recall  f1-score   support

           0       1.00      0.53      0.69       552
           1       0.20      0.98      0.33        66

    accuracy                           0.58       618
   macro avg       0.60      0.76      0.51       618
weighted avg       0.91      0.58      0.65       618
```

- After revising the optimum threshold, the accuracy and precision reduced to 58% and 20% respectively for predicting the defaults.
- But the recall shot up to 98%. This indicates that the model is not overlooking the defaulters, even though it is bucketing some of the non-defaulters as defaulters.
- But for default =0, the precision is close to 100% and recall for default =1 is 98% which fit the criteria as mentioned above. Hence, it is a good model.

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

- The dataset provided had data quality enhanced by performing outlier treatment and imputation of null values. The data was scaled as well as the imputation technique utilized was KNN Imputer which requires data to be scaled.
- As part of outlier treatment, lower quantile was kept as 0.1 and upper quantile was set as 0.9.
- Before building the Random forest model, the company data was split into Train and Test in the ratio of 70:30 and stratified split was performed to ensure the defaulter data is present in the same proportion in Test as well as Train dataset.
- For the random forest classifier model, sklearn.ensemble technique has been used. As part of hyper-parameter tuning, following pruning have been applied:
    - N_estimators = 200
    - Max_depth of each tree = 15
    - Minimum samples split = 10
    - Minimum samples leaf = 5
    - Class weight: Balanced
- Random forest are not very sensitive to outlier, but the outliers present in the dataset are significant in number and the values also are far beyond the IQR ranges for the attributes, the dataframe that was employed for Logistic Regression has been used. Moreover, the imputation is not expected to reduce the accuracy of the model.
- The random forest is first trained on the independent variables within the train data set. The model is then used to predict the dependent variable on the train data and its performance is evaluated using the classification report and the confusion matrix.
- Confusion Matrix (Train data):

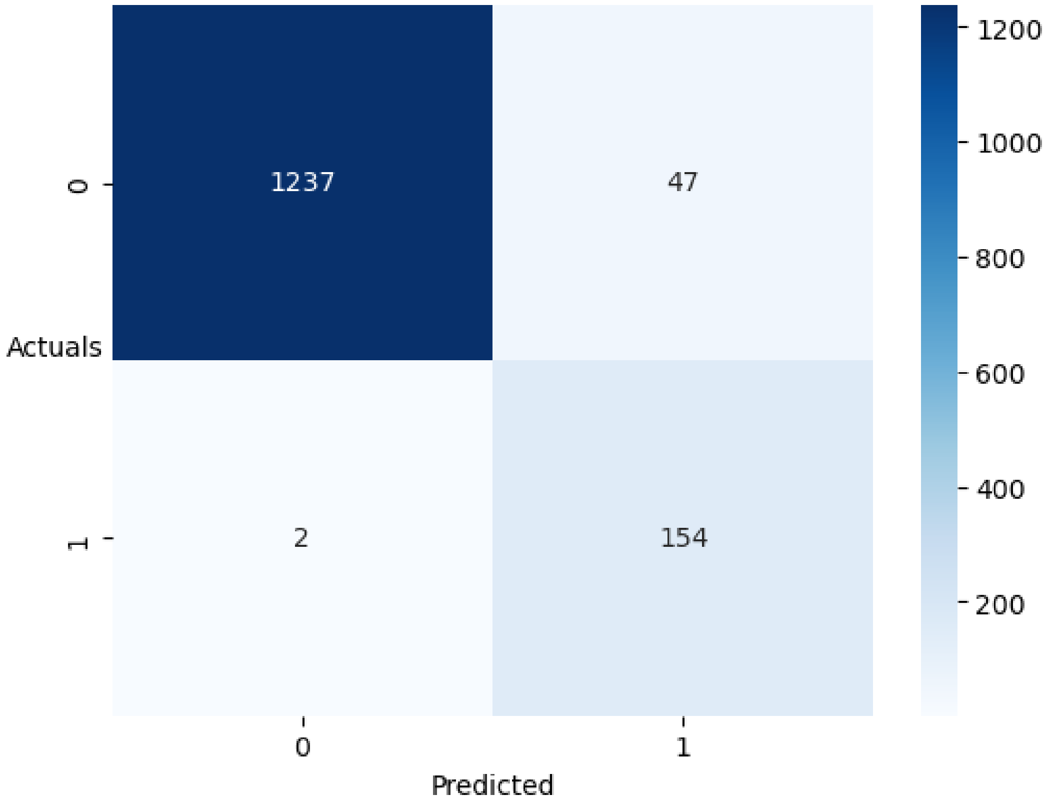


Figure 10 Confusion Matrix – Random Forest- Training

- Classification report on train data:

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98      1239
           1       0.99      0.77      0.86       201

    accuracy                           0.97      1440
   macro avg       0.98      0.88      0.92      1440
weighted avg       0.97      0.97      0.96      1440
```

- The precision of default = 0 for the random forest model is 96% and recall of default =1 is 77%.
- Overall accuracy of train data = 97%.
- In the train data, the model predicted 2 companies out of 1239 as non-defaulters who eventually defaulted.

8. Validate the Random Forest Model on test Dataset and state the performance metrics. Also state interpretation from the model

- The random forest classifier performed well on the train dataset.
- To ensure it does not overfit, pruning techniques were used. These are mentioned above.
- Despite the measures to restrict the model from overfitting. The precision and recall reduced in predicting the defaulters on Test dataset.
- Confusion Matrix on test data:

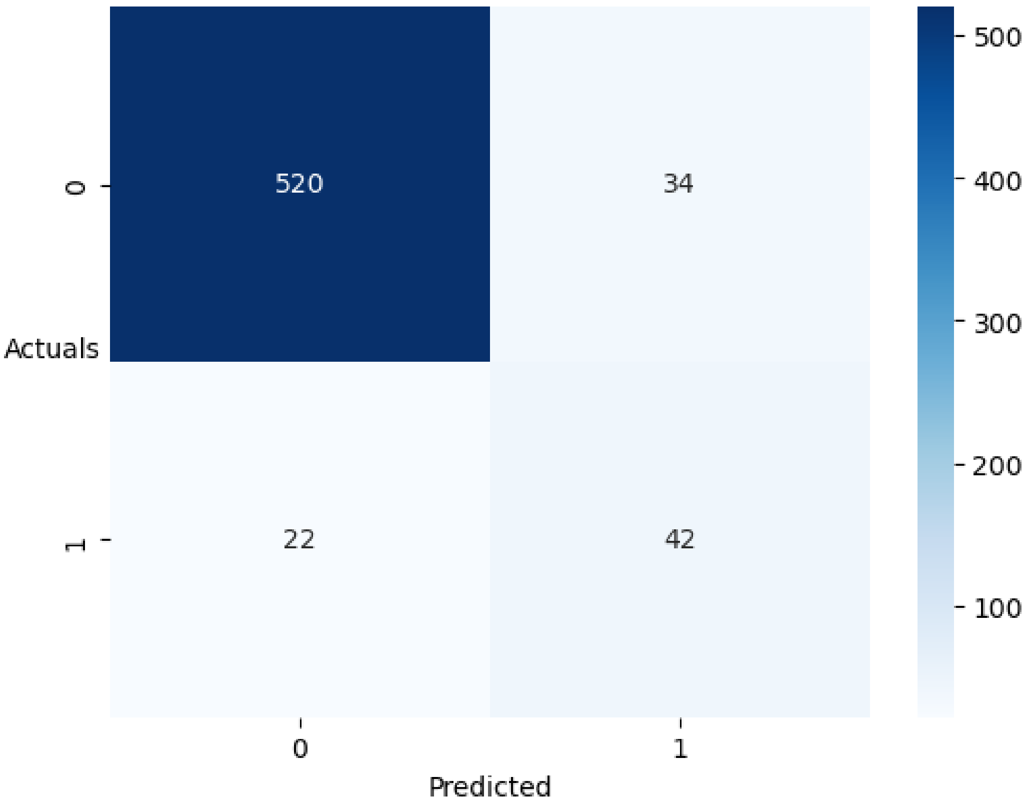


Figure 11 Confusion Matrix-Random Forest – test

- Classification report on test data:

```
              precision    recall  f1-score   support

           0       0.94      0.96      0.95       542
           1       0.66      0.55      0.60        76

    accuracy                           0.91       618
   macro avg       0.80      0.76      0.77       618
weighted avg       0.90      0.91      0.91       618
```

- The precision and recall values for default = 1 has reduced significantly when compared with the train data.
- There are 22 companies, where default was not predicted, and yet they defaulted.
- Out of 618 records in test dataset, 42 were correctly predicted as default and 22 wrongly predicted as non-defaulter.
- Random forest combines the predictions of multiple decision trees and to make a final prediction. The ensemble nature of the model improves the prediction accuracy compared to a single decision tree.

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

- The dataset provided had data quality enhanced by performing outlier treatment and imputation of null values. The data was scaled as well as the imputation technique utilized was KNN Imputer which requires data to be scaled.
- The dataset was then stratified split in the ratio of 70(Train):30 (Test) to ensure that the presence of dependent variable Default = 1 is present in the same proportion in both the datasets.
- Initially, the Linear Discriminant Analysis is performed on the train dataset. And the prediction is then evaluated on the train data set for dependent data(y_train).
- This model thus built is then used to predict the dependent variable(Default) on the test dataset. Its performance are then evaluated using the Confusion matrix and classification report.
- To enhance the performance of the LDA model, Grid Search with cross-validation was performed. Confusion matrix and classification report of the model both before applying Grid Search and after applying it have been shown below.
- Hyper parameter tuning as part of Grid Search provided the following hyper-parameters: Best Hyper-parameters: {'shrinkage': 'auto', 'solver': 'lsqr'}
- Confusion matrix of LDA on train data before Hyper parameter tuning:

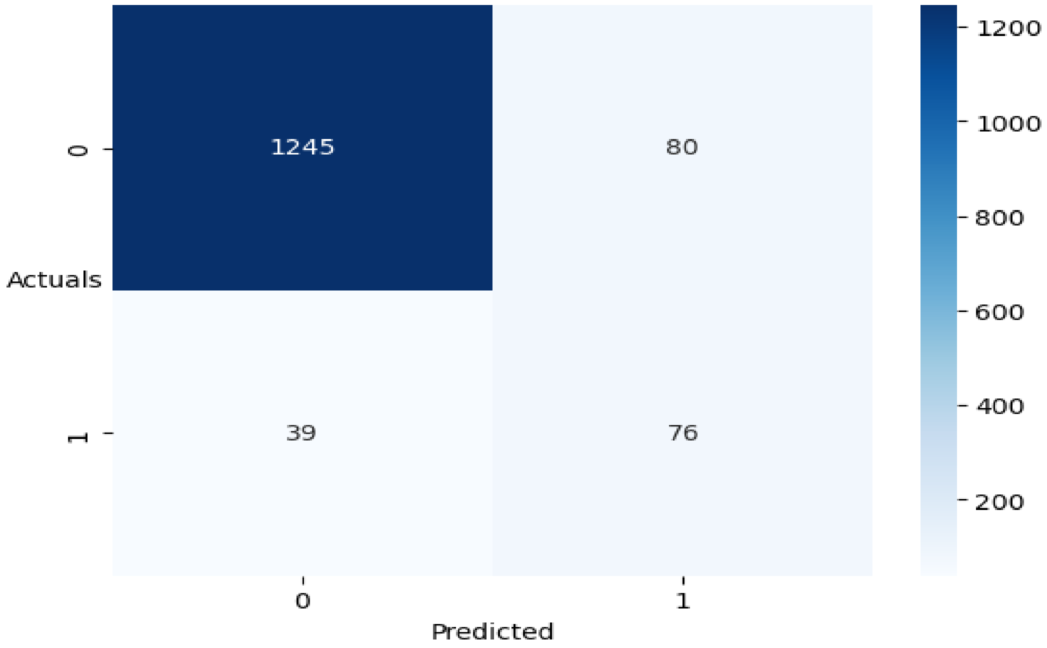


Figure 12 Confusion matrix - LDA – Training without hyper-parameter tuning

- Classification report of LDA on train data before Hyper parameter tuning:

```
              precision    recall  f1-score   support

           0       0.97      0.94      0.95      1325
           1       0.49      0.66      0.56       115

    accuracy                           0.92      1440
   macro avg       0.73      0.80      0.76      1440
weighted avg       0.93      0.92      0.92      1440
```

- Confusion matrix after hyper-parameter tuning:

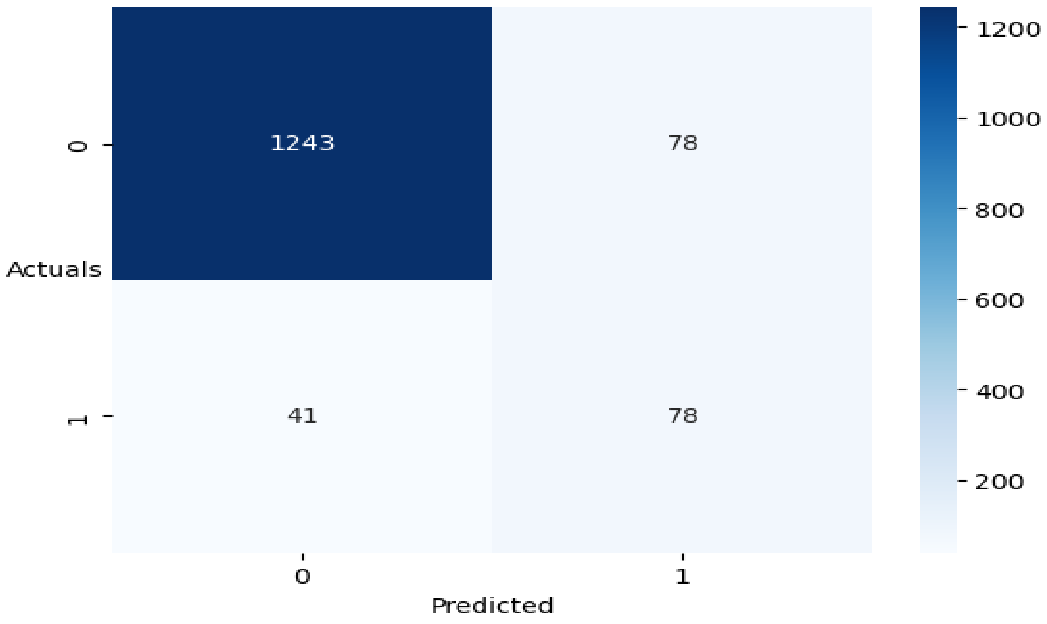


Figure 13 Confusion matrix - LDA – Train, with Hyper-parameter tuning (Grid Search)

- Classification report after hyper – parameter tuning:

```
              precision    recall  f1-score   support

           0       0.97      0.94      0.95      1321
           1       0.50      0.66      0.57       119

    accuracy                           0.92      1440
   macro avg       0.73      0.80      0.76      1440
weighted avg       0.93      0.92      0.92      1440
```

10. Validate the LDA Model on test Dataset and state the performance metrics. Also state interpretation from the model

- LDA model built on the train dataset was applied on test data to predict the dependent variable value(0 or 1).
- The predictions on independent variables in test data were evaluated using the confusion matrix and classification report on test data.
- Classification report on test data before Grid Search hyper-parameter tuning:

```
              precision    recall  f1-score   support

           0       0.95      0.96      0.95       554
           1       0.60      0.53      0.56        64

    accuracy                           0.91       618
   macro avg       0.77      0.74      0.76       618
weighted avg       0.91      0.91      0.91       618
```

Confusion matrix on test data before Grid Search hyper parameter tuning:

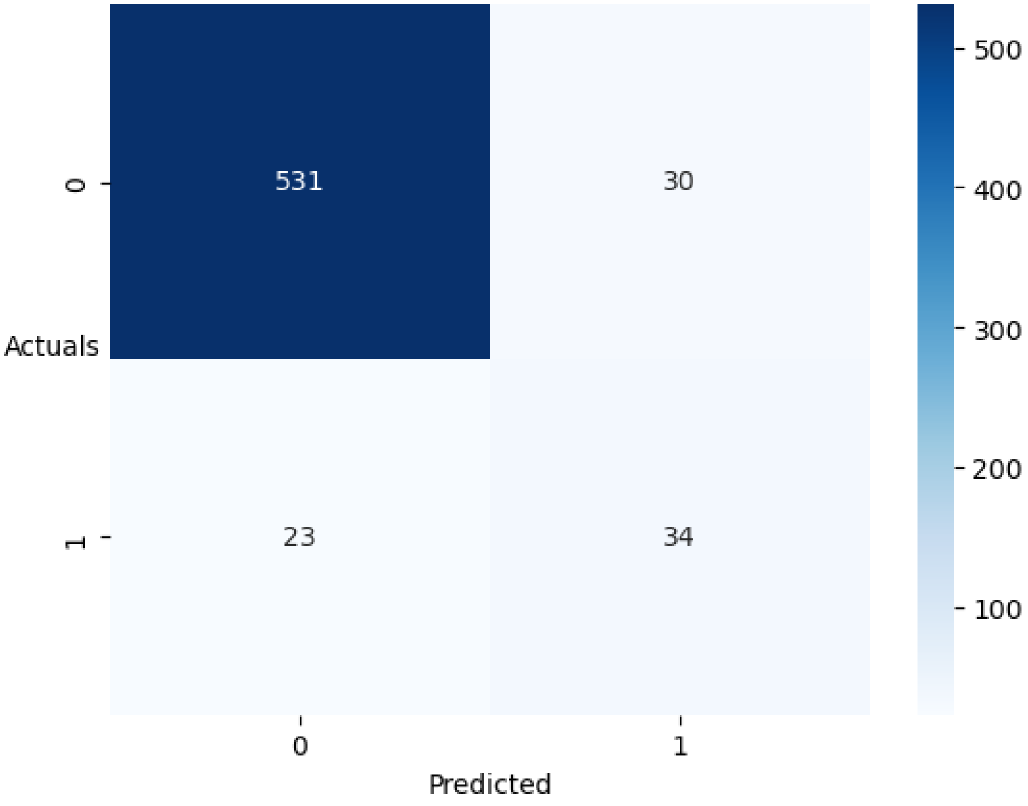


Figure 14 Confusion matrix - LDA - Test

- Classification report on the test data after Gris Search Hyper parameter tuning:

```
              precision    recall  f1-score   support

           0       0.96      0.95      0.96       558
           1       0.59      0.63      0.61        60

    accuracy                           0.92       618
   macro avg       0.78      0.79      0.78       618
weighted avg       0.92      0.92      0.92       618
```

- Confusion matrix on the test data after Gris Search Hyper parameter tuning:

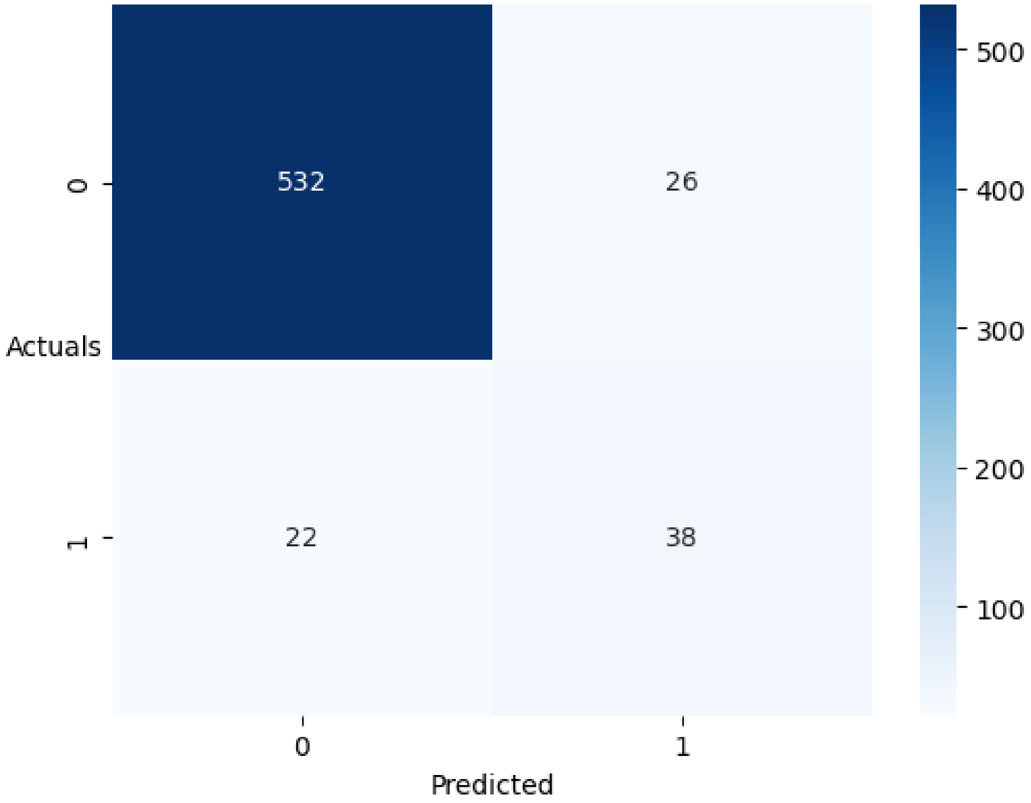


Figure 15 Confusion matrix - LDA - Test (Hyper parameter tuning using Grid Search)

- After hyper parameter tuning on test data, the recall of default = 1 is 63%.
- Out of a total size of 618 records, 22 companies were predicted as non-defaulters, but they defaulted. As a comparison, in Random forest, the number for same category was 22 whereas in Logistic Regression, it was 1.

11. Compare the performances of Logistic Regression, Random Forest, and LDA models (include ROC curve)

The models built using Logistic regression, Random Forest and Linear Discriminant Analysis are evaluated using the classification reports and confusion matrices earlier. The comparison of the models is done using ROC_AUC curve. They are all displayed in the below graph:

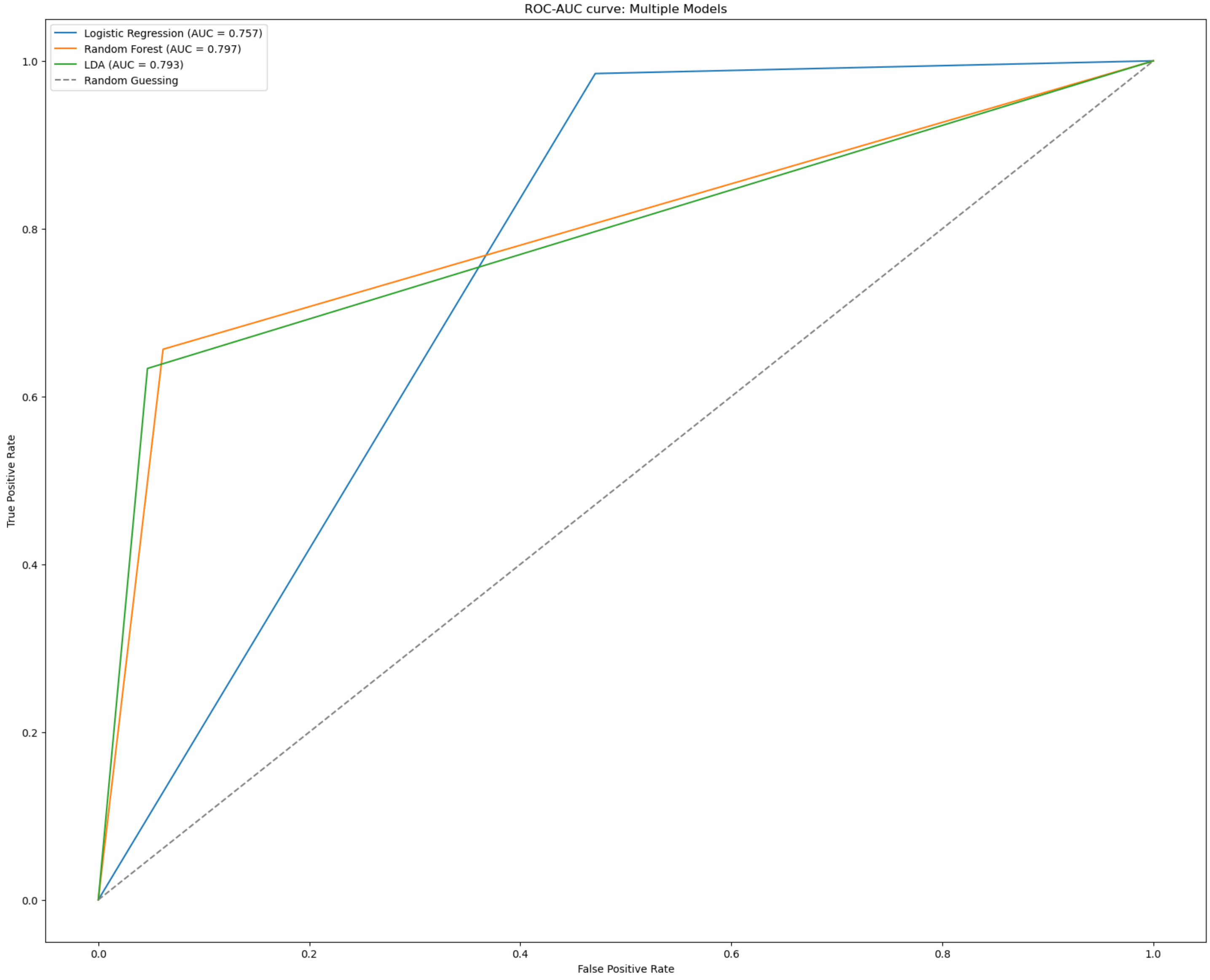


Figure 16 ROC - AUC Curve

- Based on the AUC values for the models, following is the result:
    - AUC for Logistic Regression = 0.757
    - AUC for Random Forest = 0.797
    - AUC for LDA = 0.793
- Only by AUC values, Random forest is the most accurate of the 3 models.

- But considering the most important prediction may be the accurately identifying the defaults, and ensuring that there are as less False Negatives(predicting Defaulters as non-defaulter) as possible, the model of choice in the current case study is Logistic Regression. This is because its False Negatives are minimum.

12. Conclusions and Recommendations

- To accurately predict the outcome of the credit, none of the models built as part of this case study are highly accurate in all the scenarios, i.e., accurately predicting the defaulters (No False Negatives and no False Positives).
- But the models can predict some of the scenarios better than the other models. Ex: To ensure minimum False Negatives (Predicting the Defaulters as non-Defaulters), The logistic regression is recommended.
- To Ensure minimum False Positives, LDA is recommended. For this scenario, Linear regression gives a poor result, so employing Linear Regression model for predicting False Positives is not recommended.

Part B

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

Plot of Stock price of Infosys over years:

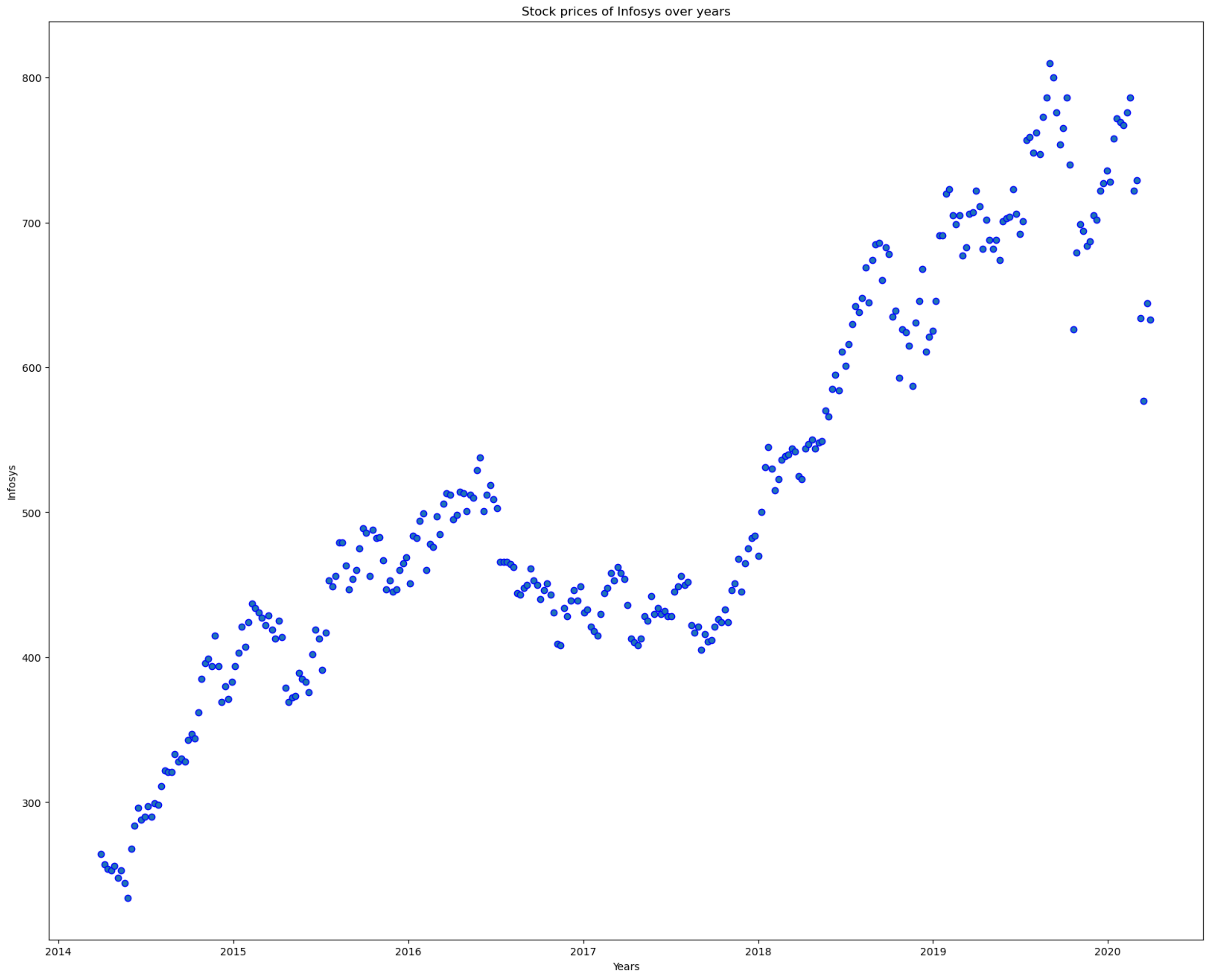


Figure 17 Stock price over the years - Infosys

- The stock prices of Infosys have been increasing overall from 2014 till 2021.
- There was a downward movement of the stock during the mid of 2016 till the start of 2018.
- After 2018, there has neem a steady increase in its stock prices, with the prices increasing faster in 2018-2019 timeframe.
- In the year 2020, there have been multiple spikes (both upward and downward), indicating sudden influences.

Plot of Stock price of SAIL over years:

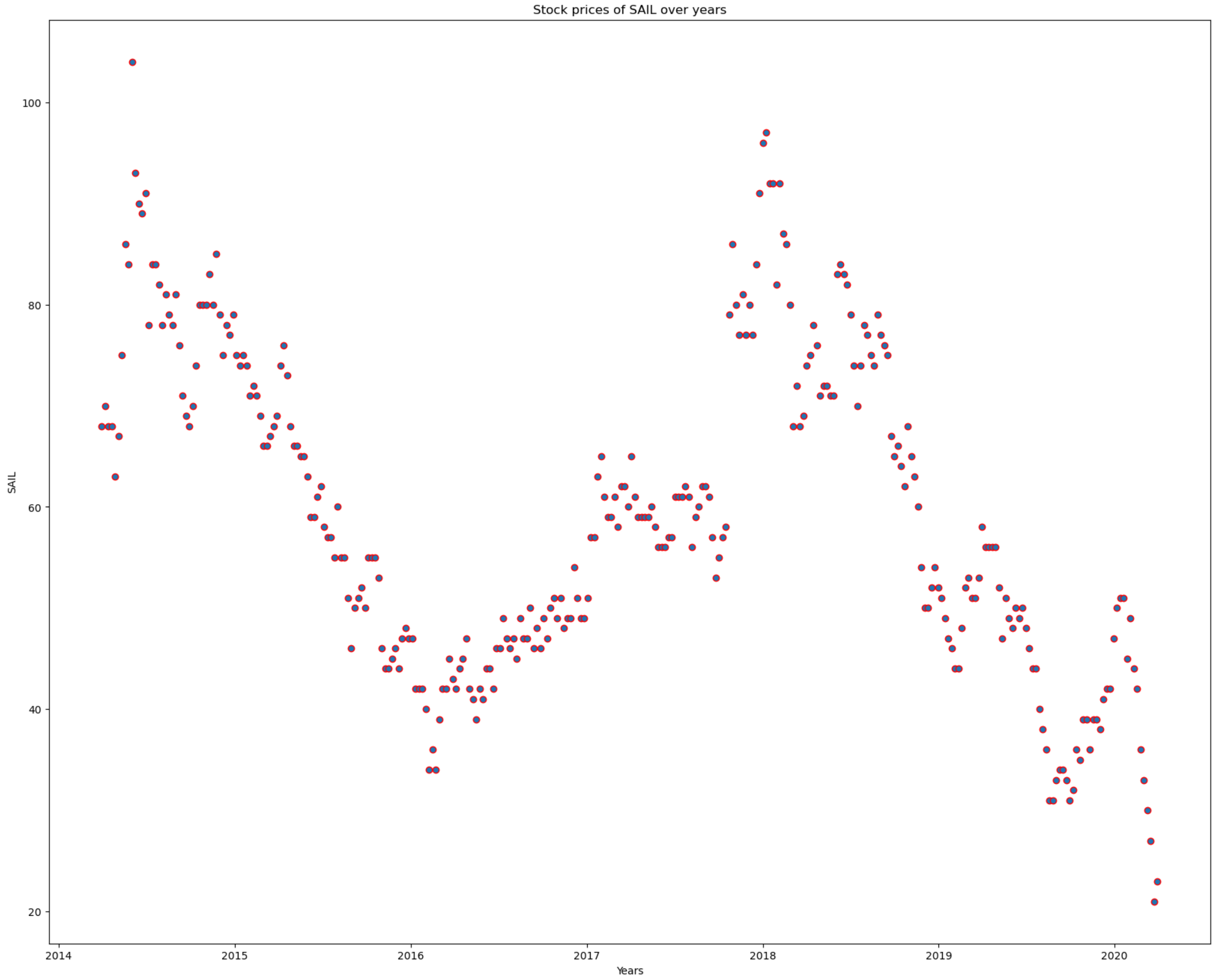
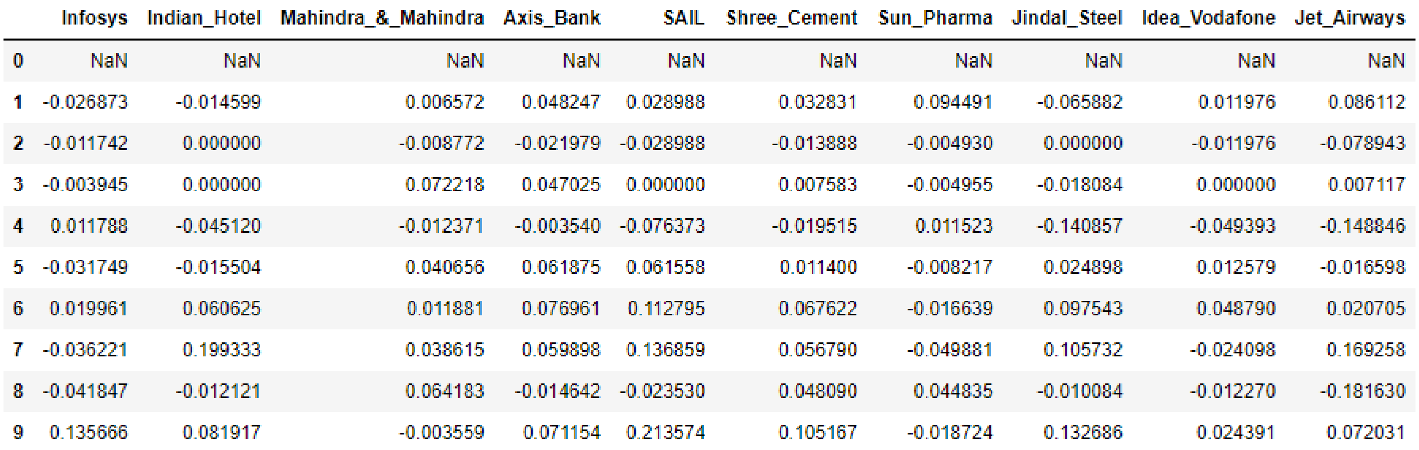


Figure 18 Stock price over the years - SAIL

- The stock prices of SAIL have seen decline overt multiple time-durations, in the 2014- 2016 and 2018-2020(mid).
- It recovered its 2014 price level in the year 2018, but has been in a steady decline after that, till the time we have stock data available for this company.

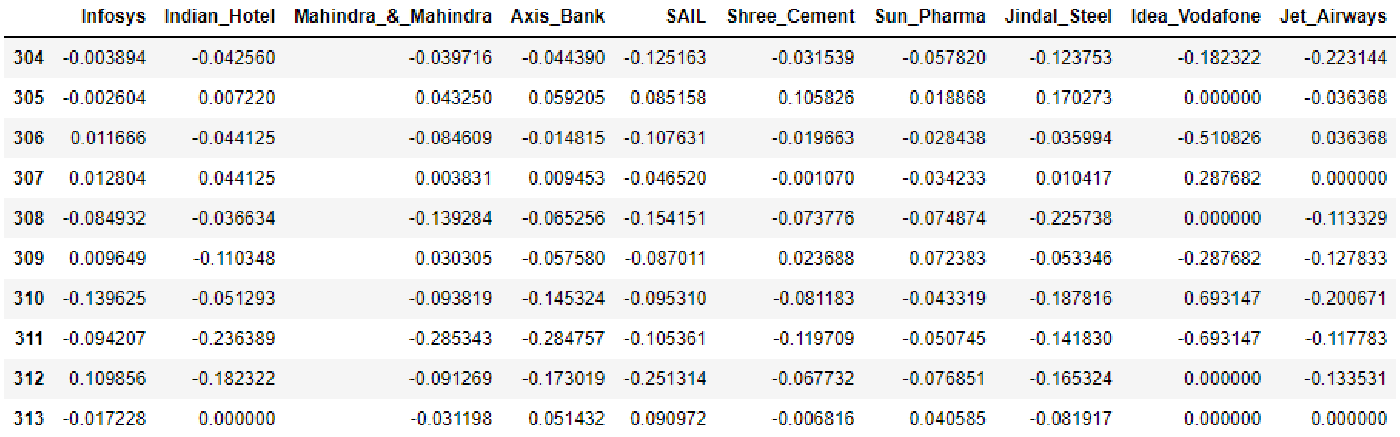
2. Calculate Returns for all stocks with inference

The stock return for all the stocks is shown in the below snapshots for the top 10 and bottom 10 instances:



	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	-0.026873	-0.014599	0.006572	0.048247	0.028988	0.032831	0.094491	-0.065882	0.011976	0.086112
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	-0.076373	-0.019515	0.011523	-0.140857	-0.049393	-0.148846
5	-0.031749	-0.015504	0.040656	0.061875	0.061558	0.011400	-0.008217	0.024898	0.012579	-0.016598
6	0.019961	0.060625	0.011881	0.076961	0.112795	0.067622	-0.016639	0.097543	0.048790	0.020705
7	-0.036221	0.199333	0.038615	0.059898	0.136859	0.056790	-0.049881	0.105732	-0.024098	0.169258
8	-0.041847	-0.012121	0.064183	-0.014642	-0.023530	0.048090	0.044835	-0.010084	-0.012270	-0.181630
9	0.135666	0.081917	-0.003559	0.071154	0.213574	0.105167	-0.018724	0.132686	0.024391	0.072031

Table 2 Stock returns - part 1



	Infosys	Indian_Hotel	Mahindra_&_Mahindra	Axis_Bank	SAIL	Shree_Cement	Sun_Pharma	Jindal_Steel	Idea_Vodafone	Jet_Airways
304	-0.003894	-0.042560	-0.039716	-0.044390	-0.125163	-0.031539	-0.057820	-0.123753	-0.182322	-0.223144
305	-0.002604	0.007220	0.043250	0.059205	0.085158	0.105826	0.018868	0.170273	0.000000	-0.036368
306	0.011666	-0.044125	-0.084609	-0.014815	-0.107631	-0.019663	-0.028438	-0.035994	-0.510826	0.036368
307	0.012804	0.044125	0.003831	0.009453	-0.046520	-0.001070	-0.034233	0.010417	0.287682	0.000000
308	-0.084932	-0.036634	-0.139284	-0.065256	-0.154151	-0.073776	-0.074874	-0.225738	0.000000	-0.113329
309	0.009649	-0.110348	0.030305	-0.057580	-0.087011	0.023688	0.072383	-0.053346	-0.287682	-0.127833
310	-0.139625	-0.051293	-0.093819	-0.145324	-0.095310	-0.081183	-0.043319	-0.187816	0.693147	-0.200671
311	-0.094207	-0.236389	-0.285343	-0.284757	-0.105361	-0.119709	-0.050745	-0.141830	-0.693147	-0.117783
312	0.109856	-0.182322	-0.091269	-0.173019	-0.251314	-0.067732	-0.076851	-0.165324	0.000000	-0.133531
313	-0.017228	0.000000	-0.031198	0.051432	0.090972	-0.006816	0.040585	-0.081917	0.000000	0.000000

Table 3 Stock returns - part 2

- To calculate the returns on the weekly stock prices of the 10 stocks provided, following have been performed:
  - The date column was converted into index(datetime)
  - Difference between 2 weeks was calculated for the returns for each stock(week-wise).
  - The first-row data is all NaN. This is because the difference for first row cannot be compared as there is no data present before it.
  - Wherever we have negative value in any of the cell, it is because the stock price in the subsequent week reduced. Hence the difference is shown as negative.

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

The stock means and volatility data for all the stocks:

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

Table 4 Stock returns versus volatility

- Idea Vodafone has lowest mean stock price returns and highest volatility.
- Infosys has the highest mean stock price return and lowest volatility.
- Axis Bank has second highest mean stock price return, but its volatility is also high.

4. Draw a plot of Stock Means vs Standard Deviation and state your inference

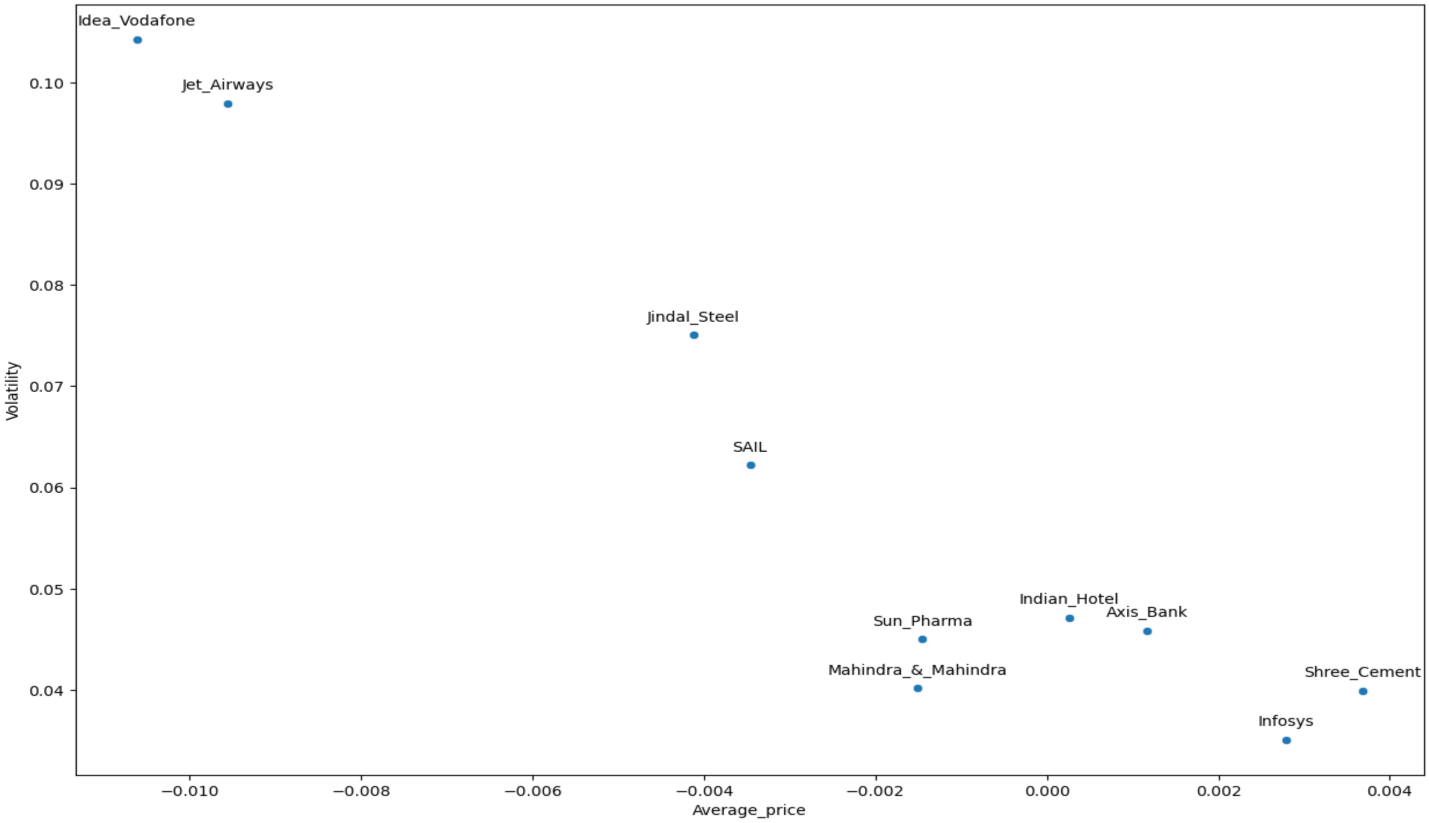


Figure 19 Average stock returns versus volatility

- The scatterplot of Average returns versus volatility is shown above.
- The lower dots show lower volatility. The dots on the right-hand side display higher price return and on the left side show the loss-making return (negative value).

5. Conclusions and Recommendations

- In the scatter-plot, the stocks return values versus volatility can serve as a primary provider of information on the performance of stock prices and aid in deciding the stock to pick.
- Infosys is the lowest in the graph, indicating it is the least volatile while it has the second highest mean value of returns.
- Shree Cement has higher return than Infosys but has higher volatility than it.
- Sun Pharma and Mahindra & Mahindra have similar stock returns, but Sun Pharma has higher volatility. So, considering other factors as constant (like ignoring the sector etc.), Mahindra & Mahindra is a better stock than Sun Pharma.
- Indian Hotel and Axis Bank have similar level of volatility but Axis bank has better average return than Indian Hotel. Hence, investing n Axis Bank may be safer than investing in Indian Hotel.