FINANCE & RISK ANALYTICS

By: Kavin Bharathi V

Contents

List of Tables

List Of Figures

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.

1.1 Data Overview:

The below table shows the first five rows of the data table.

	Co_Code	Co_Name	_Operating_Expense_Rate	_Research_and_development_expense_rate	_Cash_flow_rate	_Interest_bearing_debt_interest_rate
0	16974	Hind.Cables	8.820000e+09	0.000000e+00	0.462045	0.000352
1	21214	Tata Tele. Mah.	9.380000e+09	4.230000e+09	0.460116	0.000716
2	14852	ABG Shipyard	3.800000e+09	8.150000e+08	0.449893	0.000496
3	2439	GTL	6.440000e+09	0.000000e+00	0.462731	0.000592
4	23505	Bharati Defence	3.680000e+09	0.000000e+00	0.463117	0.000782

Table 1 – Top five Rows

The below table shows the last five rows of the data table.

Out[9]:

	Co_Code	Co_Name	_Operating_Expense_Rate	_Research_and_development_expense_rate	_Cash_flow_rate	_Interest_bearing_debt_interest_rate
2053	2743	Kothari Ferment.	3.021580e-04	6.490000e+09	0.477066	0.000000
2054	21216	Firstobj.Tech.	1.371450e-04	0.000000e+00	0.465211	0.000658
2055	142	Diamines & Chem.	2.114990e-04	8.370000e+09	0.480248	0.000502
2056	18014	IL&FS Engg.	3.750000e+09	0.000000e+00	0.474670	0.000578
2057	43229	Channel Nine	2.981110e-04	0.000000e+00	0.467203	0.000826

Table 2 - Last five Rows

Data Information:

```
0   Co_Code                                           2058 non-null   int64
1   Co_Name                                           2058 non-null   object
2   _Operating_Expense_Rate                           2058 non-null   float64
3   _Research_and_development_expense_rate            2058 non-null   float64
4   _Cash_flow_rate                                   2058 non-null   float64
5   _Interest_bearing_debt_interest_rate              2058 non-null   float64
6   _Tax_rate_A                                       2058 non-null   float64
7   _Cash_Flow_Per_Share                              1891 non-null   float64
8   _Per_Share_Net_profit_before_tax_Yuan_            2058 non-null   float64
9   _Realized_Sales_Gross_Profit_Growth_Rate          2058 non-null   float64
10  _Operating_Profit_Growth_Rate                     2058 non-null   float64
11  _Continuous_Net_Profit_Growth_Rate                2058 non-null   float64
12  _Total_Asset_Growth_Rate                          2058 non-null   float64
13  _Net_Value_Growth_Rate                            2058 non-null   float64
14  _Total_Asset_Return_Growth_Rate_Ratio             2058 non-null   float64
15  _Cash_Reinvestment_perc                           2058 non-null   float64
16  _Current_Ratio                                    2058 non-null   float64
17  _Quick_Ratio                                      2058 non-null   float64
18  _Interest_Expense_Ratio                           2058 non-null   float64
19  _Total_debt_to_Total_net_worth                    2037 non-null   float64
20  _Long_term_fund_suitability_ratio_A               2058 non-null   float64
21  _Net_profit_before_tax_to_Paid_in_capital         2058 non-null   float64
22  _Total_Asset_Turnover                             2058 non-null   float64
23  _Accounts_Receivable_Turnover                     2058 non-null   float64
24  _Average_Collection_Days                          2058 non-null   float64
25  _Inventory_Turnover_Rate_times                    2058 non-null   float64
26  _Fixed_Assets_Turnover_Frequency                  2058 non-null   float64
27  _Net_Worth_Turnover_Rate_times                    2058 non-null   float64
28  _Operating_profit_per_person                      2058 non-null   float64
29  _Allocation_rate_per_person                       2058 non-null   float64
30  _Quick_Assets_to_Total_Assets                     2058 non-null   float64
31  _Cash_to_Total_Assets                             1962 non-null   float64
32  _Quick_Assets_to_Current_Liability                2058 non-null   float64
33  _Cash_to_Current_Liability                        2058 non-null   float64
34  _Operating_Funds_to_Liability                     2058 non-null   float64
35  _Inventory_to_Working_Capital                     2058 non-null   float64
36  _Inventory_to_Current_Liability                   2058 non-null   float64
37  _Long_term_Liability_to_Current_Assets            2058 non-null   float64
38  _Retained_Earnings_to_Total_Assets                2058 non-null   float64
39   Total income to Total expense                    2058 non-null   float64
```

40	_Total_expense_to_Assets	2058 non-null	float64
41	_Current_Asset_Turnover_Rate	2058 non-null	float64
42	_Quick_Asset_Turnover_Rate	2058 non-null	float64
43	_Cash_Turnover_Rate	2058 non-null	float64
44	_Fixed_Assets_to_Assets	2058 non-null	float64
45	_Cash_Flow_to_Total_Assets	2058 non-null	float64
46	_Cash_Flow_to_Liability	2058 non-null	float64
47	_CFO_to_Assets	2058 non-null	float64
48	_Cash_Flow_to_Equity	2058 non-null	float64
49	_Current_Liability_to_Current_Assets	2044 non-null	float64
50	_Liability_Assets_Flag	2058 non-null	int64
51	_Total_assets_to_GNP_price	2058 non-null	float64
52	_No_credit_Interval	2058 non-null	float64
53	_Degree_of_Financial_Leverage_DFL	2058 non-null	float64
54	_Interest_Coverage_Ratio_Interest_expense_to_EBIT	2058 non-null	float64
55	_Net_Income_Flag	2058 non-null	int64
56	_Equity_to_Liability	2058 non-null	float64
57	Default	2058 non-null	int64

Table 3 - Data Information

- The dataset has **2,058 rows** with **58 features**, showing different financial and performance indicators of various companies.
- Among these features, there are **57 numerical data and 1 categorical data**, with some missing values.
- One of the columns in the dataset is named "**Default**" which shows whether a company has defaulted (1) or not (0). This could be useful for predicting defaults or further analysis related to company defaults.

Data Summary:

	count	mean	std	min	25%	50%	75%	max
Co_Code	2058.0	1.757211e+04	2.189289e+04	4.000000	3.674000e+03	6.240000e+03	2.428075e+04	7.249300e+04
_Operating_Expense_Rate	2058.0	2.052389e+09	3.252624e+09	0.000100	1.578727e-04	3.330330e-04	4.110000e+09	9.980000e+09
_Research_and_development_expense_rate	2058.0	1.208634e+09	2.144568e+09	0.000000	0.000000e+00	1.994130e-04	1.550000e+09	9.980000e+09
_Cash_flow_rate	2058.0	4.652426e-01	2.266269e-02	0.000000	4.600991e-01	4.634450e-01	4.680691e-01	1.000000e+00
_Interest_bearing_debt_interest_rate	2058.0	1.113022e+07	9.042595e+07	0.000000	2.760280e-04	4.540450e-04	6.630660e-04	9.900000e+08
_Tax_rate_A	2058.0	1.147770e-01	1.524457e-01	0.000000	0.000000e+00	3.709890e-02	2.161909e-01	9.996963e-01
_Cash_Flow_Per_Share	1891.0	3.199856e-01	1.529979e-02	0.169449	3.149890e-01	3.206479e-01	3.259178e-01	4.622268e-01
_Per_Share_Net_profit_before_tax_Yuan_	2058.0	1.769673e-01	3.015730e-02	0.000000	1.666039e-01	1.756421e-01	1.858854e-01	7.923477e-01
_Realized_Sales_Gross_Profit_Growth_Rate	2058.0	2.276117e-02	2.170104e-02	0.004282	2.205831e-02	2.210001e-02	2.215200e-02	1.000000e+00
_Operating_Profit_Growth_Rate	2058.0	8.481083e-01	4.589093e-03	0.736430	8.479740e-01	8.480386e-01	8.481147e-01	1.000000e+00
_Continuous_Net_Profit_Growth_Rate	2058.0	2.173915e-01	5.678779e-03	0.000000	2.175741e-01	2.175961e-01	2.176198e-01	2.332046e-01
_Total_Asset_Growth_Rate	2058.0	5.287663e+09	2.912615e+09	0.000000	4.315000e+09	6.225000e+09	7.220000e+09	9.980000e+09
_Net_Value_Growth_Rate	2058.0	5.189504e+06	2.077918e+08	0.000000	4.362833e-04	4.554170e-04	4.883758e-04	9.330000e+09
_Total_Asset_Return_Growth_Rate_Ratio	2058.0	2.641004e-01	2.415661e-03	0.251620	2.637383e-01	2.640161e-01	2.643097e-01	3.586288e-01

Table 4 - Data Summary

- The above table shows the summary of the descriptive statistics of the columns in the dataset.
- It also shows the mean, median, min, max and lower and upper quartile values.

Duplicate Values:

- There are no duplicate values found in the dataset.

1.2 Outlier Treatment:

Checking for outliers:

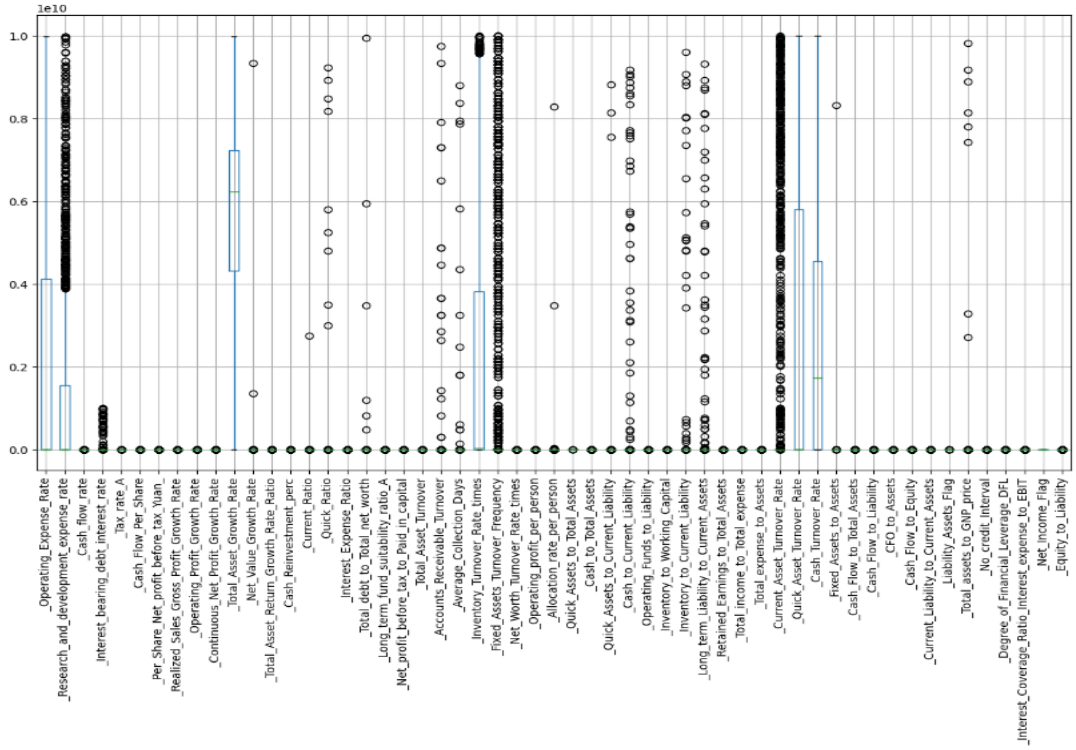

Figure 1 – Outliers

Outlier Removal:

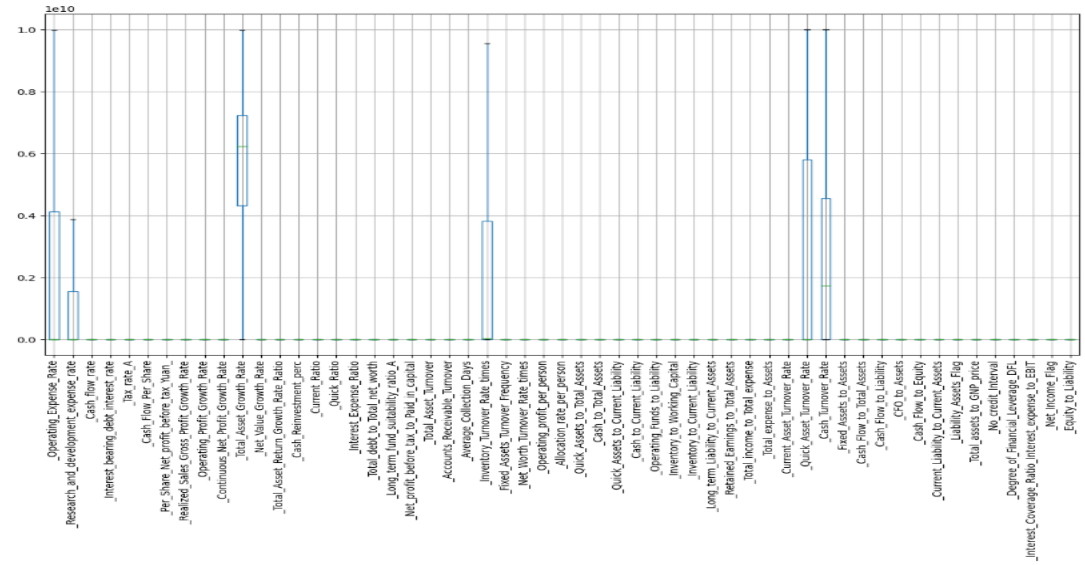

Figure 2 - Boxplot after Outlier Removal

Outlier removal is a crucial step in preprocessing the data. It brings the accurate analysis in the model building. Here, **IQR** method has been used and the outliers has been removed as seen above.

1.3 Missing Value Treatment:

Out[12]:

Co_Code	0
Co_Name	0
_Operating_Expense_Rate	0
_Research_and_development_expense_rate	0
_Cash_flow_rate	0
_Interest_bearing_debt_interest_rate	0
_Tax_rate_A	0
_Cash_Flow_Per_Share	167
_Per_Share_Net_profit_before_tax_Yuan_	0
_Realized_Sales_Gross_Profit_Growth_Rate	0
_Operating_Profit_Growth_Rate	0
_Continuous_Net_Profit_Growth_Rate	0
_Total_Asset_Growth_Rate	0
_Net_Value_Growth_Rate	0
_Total_Asset_Return_Growth_Rate_Ratio	0
_Cash_Reinvestment_perc	0
_Current_Ratio	0
_Quick_Ratio	0
_Interest_Expense_Ratio	0
_Total_debt_to_Total_net_worth	21
_Long_term_fund_suitability_ratio_A	0
_Net_profit_before_tax_to_Paid_in_capital	0
_Total_Asset_Turnover	0
_Accounts_Receivable_Turnover	0
_Average_Collection_Days	0
_Inventory_Turnover_Rate_times	0
_Fixed_Assets_Turnover_Frequency	0
_Net_Worth_Turnover_Rate_times	0
_Operating_profit_per_person	0
_Allocation_rate_per_person	0
_Quick_Assets_to_Total_Assets	0
_Cash_to_Total_Assets	96
_Quick_Assets_to_Current_Liability	0
_Cash_to_Current_Liability	0
_Operating_Funds_to_Liability	0
_Inventory_to_Working_Capital	0
_Inventory_to_Current_Liability	0
_Long_term_Liability_to_Current_Assets	0
_Retained_Earnings_to_Total_Assets	0
_Total_income_to_Total_expense	0
_Total_expense_to_Assets	0
_Current_Asset_Turnover_Rate	0
_Quick_Asset_Turnover_Rate	0
_Cash_Turnover_Rate	0
_Fixed_Assets_to_Assets	0
_Cash_Flow_to_Total_Assets	0
_Cash_Flow_to_Liability	0
_CFO_to_Assets	0
_Cash_Flow_to_Equity	0
_Current_Liability_to_Current_Assets	14
_Liability_Assets_Flag	0
_Total_assets_to_GNP_price	0
_No_credit_Interval	0
_Degree_of_Financial_Leverage_DFL	0
_Interest_Coverage_Ratio_Interest_expense_to_EBIT	0
_Net_Income_Flag	0
_Equity_to_Liability	0
Default	0

Table 5 - Missing Values

Missing Value After Treatment:

Out[29]:

_Operating_Expense_Rate	0
_Research_and_development_expense_rate	0
_Cash_flow_rate	0
_Interest_bearing_debt_interest_rate	0
_Tax_rate_A	0
_Cash_Flow_Per_Share	0
_Per_Share_Net_profit_before_tax_Yuan_	0
_Realized_Sales_Gross_Profit_Growth_Rate	0
_Operating_Profit_Growth_Rate	0
_Continuous_Net_Profit_Growth_Rate	0
_Total_Asset_Growth_Rate	0
_Net_Value_Growth_Rate	0
_Total_Asset_Return_Growth_Rate_Ratio	0
_Cash_Reinvestment_perc	0
_Current_Ratio	0
_Quick_Ratio	0
_Interest_Expense_Ratio	0
_Total_debt_to_Total_net_worth	0
_Long_term_fund_suitability_ratio_A	0
_Net_profit_before_tax_to_Paid_in_capital	0
_Total_Asset_Turnover	0
_Accounts_Receivable_Turnover	0
_Average_Collection_Days	0
_Inventory_Turnover_Rate_times	0
_Fixed_Assets_Turnover_Frequency	0
_Net_Worth_Turnover_Rate_times	0
_Operating_profit_per_person	0
_Allocation_rate_per_person	0
_Quick_Assets_to_Total_Assets	0
_Cash_to_Total_Assets	0
_Quick_Assets_to_Current_Liability	0
_Cash_to_Current_Liability	0
_Operating_Funds_to_Liability	0
_Inventory_to_Working_Capital	0
_Inventory_to_Current_Liability	0
_Long_term_Liability_to_Current_Assets	0
_Retained_Earnings_to_Total_Assets	0
_Total_income_to_Total_expense	0
_Total_expense_to_Assets	0
_Current_Asset_Turnover_Rate	0
_Quick_Asset_Turnover_Rate	0
_Cash_Turnover_Rate	0
_Fixed_Assets_to_Assets	0
_Cash_Flow_to_Total_Assets	0
_Cash_Flow_to_Liability	0
_CFO_to_Assets	0
_Cash_Flow_to_Equity	0
_Current_Liability_to_Current_Assets	0
_Liability_Assets_Flag	0
_Total_assets_to_GNP_price	0
_No_credit_Interval	0
_Degree_of_Financial_Leverage_DFL	0
_Interest_Coverage_Ratio_Interest_expense_to_EBIT	0
_Net_Income_Flag	0
_Equity_to_Liability	0

dtype: int64

Table 6 - Null Value Treatment

- We've identified a total of 298 missing values in our dataset. While this might seem small compared to the dataset's size, it's crucial to handle these missing values properly. To address this, we're using KNN (K-Nearest Neighbors) imputation with a parameter value of n-Neighbor = 5.

- KNN imputation is a technique used to fill in missing values by considering the values of neighboring data points. Each missing value is replaced with the average value of its five nearest neighbors. This method aims to make our dataset more complete and robust for analysis, while also minimizing bias in the imputed values.

1.4 Univariate & Bivariate Analysis:

- We've chosen important variables for our univariate and bivariate analyses based on their statistical significance. These variables have a p-value below 0.005, indicating a strong relationship with the target variable. Additionally, we've checked the Variance Inflation Factor (VIF) of each variable to ensure it's below 5, preventing multicollinearity issues.

Univariate Analysis:

Count of Defaulters:

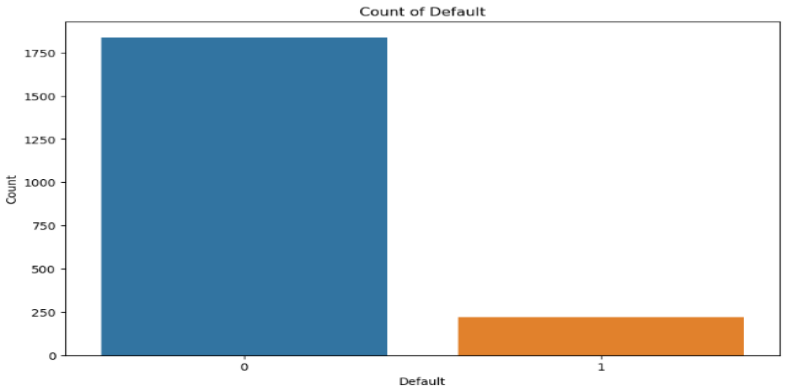

Figure 3 - No of Defaulters

- From the above plot we can conclude that large number of companies falls into non default category.
- Only around 220 companies comes under defaulters category.

Histogram of Cash flow per share:

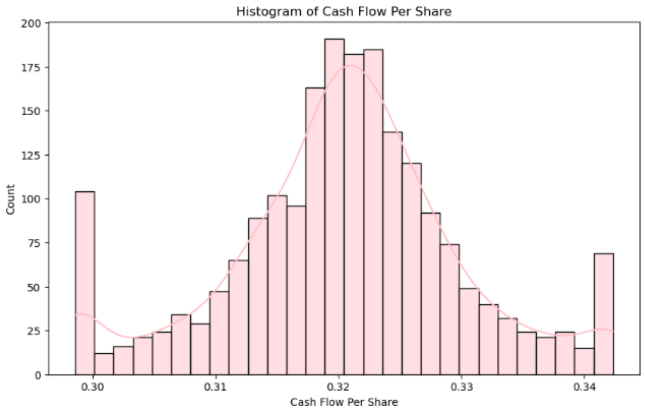

Figure 4 - Histogram of Cash Flow per share

- The histogram shows a clear peak at a cash flow per share value of 0.32, suggesting that a considerable portion of the dataset is concentrated around this particular value.

Boxplot of Research and Development Expense Rate:

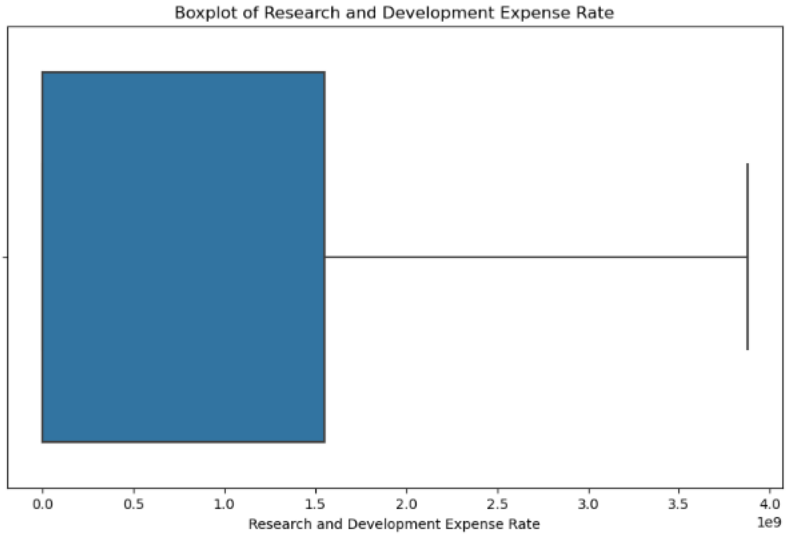

Figure 5 - Research and Development Expense Rate

- The boxplot demonstrates that the middle 50% of the data falls within the range of 0.0 to 1.5 for the research and development expense rate. Additionally, the maximum value for this variable is around 4.0, suggesting the presence of values beyond the upper range depicted by the boxplot.

Boxplot of Average Collection Days:

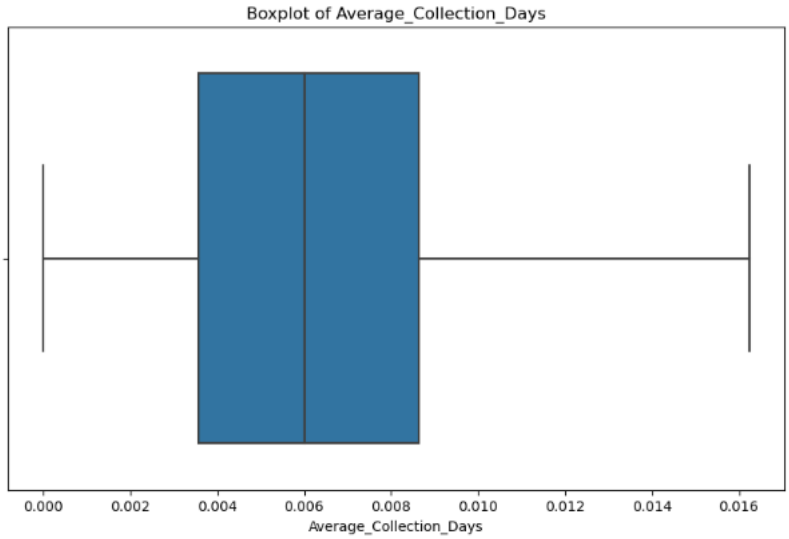

Figure 6 - Boxplot of Average Collection Days

- The boxplot of "Average Collection Days" shows a distribution ranging from a minimum value of 0.000 to a maximum value of 0.016. The median (50th percentile) is positioned at 0.006, representing the central tendency of the data.

Bi-Variate Analysis:

Boxplot of Quick Assets to Total Assets by Default:

- The boxplot analysis of "Quick Assets to Total Assets" based on default status shows distinct distributions between defaulted (1) and non-defaulted (0) companies.
- Although the overall patterns may seem similar, defaulted companies tend to have slightly lower values for "Quick Assets to Total Assets" compared to non-defaulted ones.
- This observation implies that the ratio of quick assets to total assets could be a significant indicator of default risk, with lower values potentially indicating a higher probability of default.

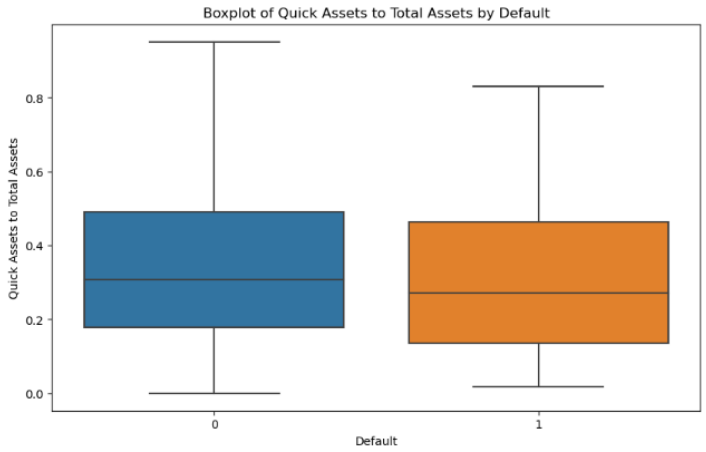

Figure 7 - Boxplot of Quick Assets to Total Assets by Default

Boxplot of Total Asset Growth Rate by Default:

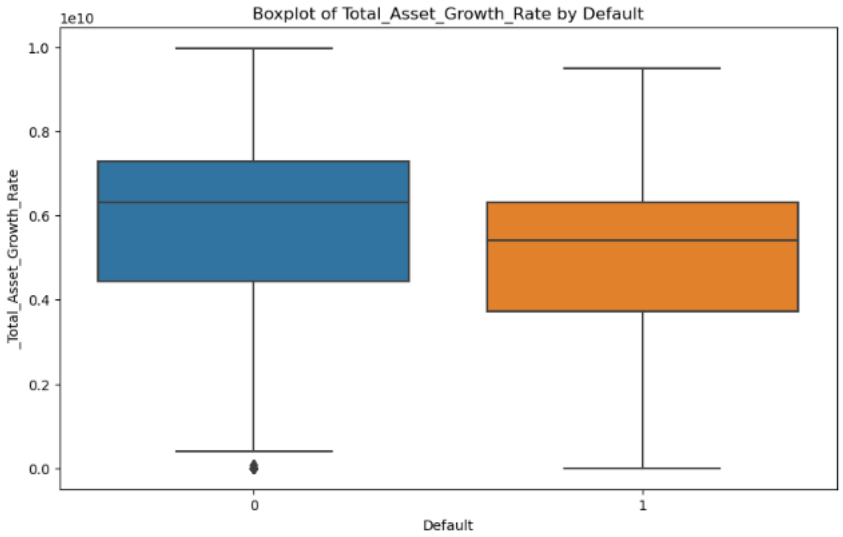

Figure 8 - Boxplot of Total Asset Growth Rate by Default

- The boxplot analysis of "Total Asset Growth Rate" based on default status (0 – Not Defaulted, 1 - Defaulted) reveals similar distributions for both defaulted and non-defaulted companies.
- Both groups exhibit comparable median values, ranging from 0.5 to 0.6.
- This indicates that the growth rate of total assets may not be the sole distinguishing factor for default risk in this dataset.

Boxplot of Research and Development Expense Rate by Default:

- The boxplot analysis of "Total Asset Growth Rate" by default status (0 - Not Defaulted, 1 - Defaulted) reveals intriguing observations.
- Non-defaulted companies (0) demonstrate limited growth, with most values clustered around 0.0 and a few outliers reaching up to 3.75. In contrast, defaulted companies (1) display a wider range of growth rates, with the median at 0.25 and some values extending up to 2.75.
- This indicates that defaulted companies tend to have more diverse asset growth rates compared to non-defaulted ones.

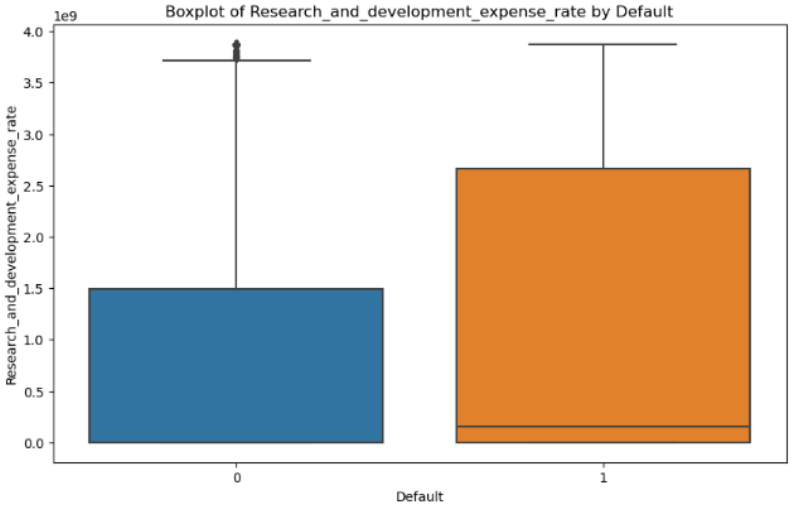

Figure 9 - Boxplot of Research and Development Expense Rate by Default

Boxplot of Operating Profit per Person by Default:

- The boxplot analysis of "Operating Profit per Person" by default status (0 - Not Defaulted, 1 - Defaulted) uncovers intriguing trends.
- Non-defaulted companies (0) exhibit a relatively narrow range of operating profit per person, with values concentrated between 0.393 and 0.400. Conversely, defaulted companies (1) display a slightly wider range, with the median at 0.390 and values extending up to 0.415.
- This suggests that there might be some variability in operating profit per person among defaulted companies compared to non-defaulted ones, although the overall difference is relatively minor.

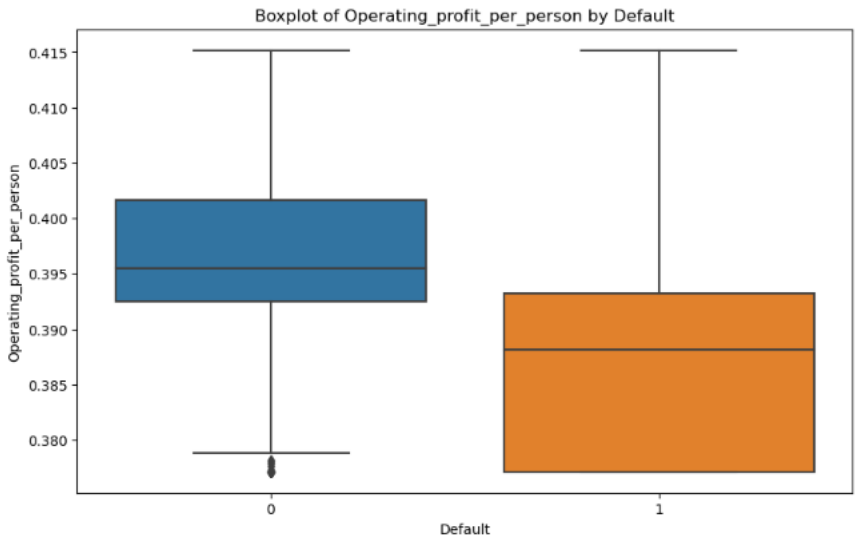

Figure 7 - Boxplot of Operating Profit per Person by Default

Heatmap

In our analysis, we found and got rid of some variables because they stayed the same throughout the dataset. These included things like

- _Realized_Sales_Gross_Profit_Growth_Rate,
- Operating_Profit_Growth_Rate,
- and Continuous_Net_Profit_Growth_Rate.

Then, we looked at the relationships between the remaining variables using a heatmap. We noticed that some variables were strongly connected to each other. This helps us tidy up the dataset by getting rid of useless variables and shows us how the remaining ones are connected, so we can focus on more important analysis.

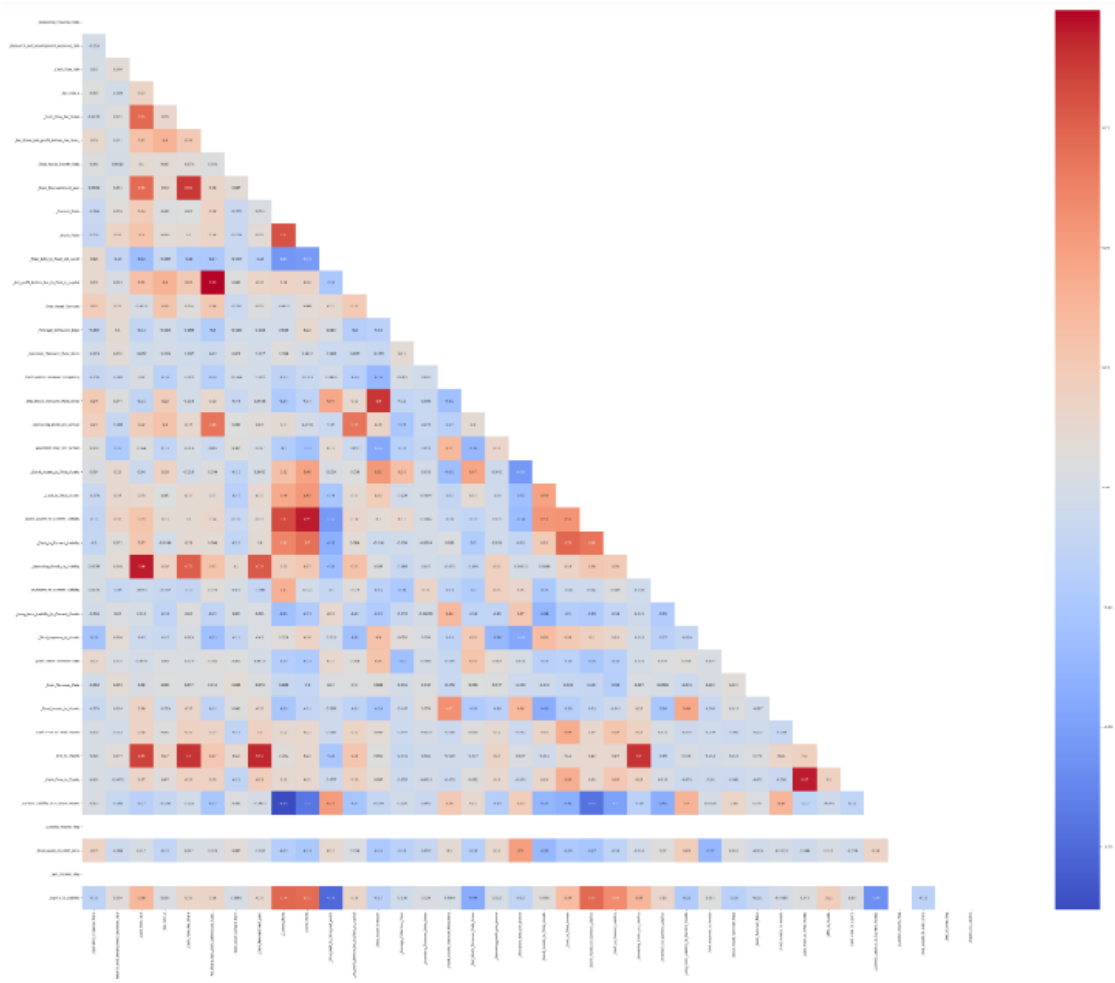

Figure 8 - Heatmap of the Variables

VIF (Variance Inflation Factor):

- VIF (Variance Inflation Factor) is a measure used to check for multicollinearity in a regression model. Multicollinearity happens when predictor variables are highly correlated with each other.

- To assess multicollinearity in the dataset, we used the statsmodels library to calculate VIF. VIF shows how much predictor variables are correlated with each other. Higher VIF values mean stronger correlation, which can make the regression model's coefficients unstable. By calculating VIF values for each variable, we can spot any multicollinearity issues in the dataset.

Out[57]:

	variables	VIF
14	_Quick_Assets_to_Total_Assets	4.098079
26	_Equity_to_Liability	3.785433
24	_Current_Liability_to_Current_Assets	3.628274
16	_Cash_to_Current_Liability	3.602009
15	_Cash_to_Total_Assets	3.550621
8	_Total_Asset_Turnover	2.985223
7	_Total_debt_to_Total_net_worth	2.911499
2	_Cash_flow_rate	2.878504
5	_Per_Share_Net_profit_before_tax_Yuan_	2.628898
22	_Fixed_Assets_to_Assets	2.505774
4	_Cash_Flow_Per_Share	2.428450
13	_Allocation_rate_per_person	2.410502
12	_Operating_profit_per_person	2.335970
17	_Inventory_to_Current_Liability	2.222968
19	_Total_expense_to_Assets	1.979909
11	_Fixed_Assets_Turnover_Frequency	1.729218
9	_Average_Collection_Days	1.693310
25	_Total_assets_to_GNP_price	1.662438
18	_Long_term_Liability_to_Current_Assets	1.582406
23	_Cash_Flow_to_Equity	1.362376
20	_Quick_Asset_Turnover_Rate	1.351777
3	_Tax_rate_A	1.341390
0	_Operating_Expense_Rate	1.297634
1	_Research_and_development_expense_rate	1.148972
10	_Inventory_Turnover_Rate_times	1.140767
6	_Total_Asset_Growth_Rate	1.093039
21	_Cash_Turnover_Rate	1.088528

Table 7 - VIF with values less than 5

- We used VIF (Variance Inflation Factor) values to find variables that are strongly correlated with each other. Higher VIF values mean stronger correlation.
- To keep the regression model stable, we removed variables with VIF values above 5 because they might cause multicollinearity issues. This helped us select a set of variables with minimal correlation, making our analysis more reliable.

1.5 Train Test Split:

Before splitting the data into training and testing sets, we will perform data scaling. This step is crucial because we noticed that some variables in the dataset have large data values. By scaling the data, we can ensure that all features are on a similar scale, preventing any single variable from dominating the learning process and ensuring compatibility across different algorithms. This method will help maintain the integrity of our analysis and enhance the accuracy and efficiency of our models.

To standardize the data, we used the StandardScaler method. This was necessary to bring all variables to a similar scale, eliminating any potential bias caused by variables with larger data values.

To assess the performance of machine learning models, we split the dataset into training and testing sets using the train_test_split function. This division ensures that the models are trained on a portion of the data and then tested on unseen data. The test size parameter was set to 0.33, indicating that 33% of the data is reserved for testing. The random_state parameter was set to 42. This procedure enables us to evaluate how well the model performs on new and unseen data.

```
Train dataset shape: (1378, 27) (1378,)
Test dataset shape: (680, 27) (680,)
```

Figure 9 - Train & Test Split

Head of the Train Data:

Out[61]:

	_Operating_Expense_Rate	_Research_and_development_expense_rate	_Cash_flow_rate	_Tax_rate_A	_Cash_Flow_Per_Share	_Per_Share_Net_profit_be
2011	-0.631148	-0.645151	2.093814	-0.819217	1.665836	
697	-0.631148	0.566343	0.256393	1.520875	0.860006	
160	-0.631148	-0.645151	-0.718630	-0.819217	-0.581192	
1273	1.201664	1.341698	-0.250583	-0.819217	0.046057	
541	-0.631148	-0.186860	-0.516785	-0.819217	-0.426224	

Table 8 - Head of the Train Data

Head of the Test Data:

Out[62]:

	_Operating_Expense_Rate	_Research_and_development_expense_rate	_Cash_flow_rate	_Tax_rate_A	_Cash_Flow_Per_Share	_Per_Share_Net_profit_be
974	-0.631148	0.178665	1.297347	1.127607	1.510869	
134	-0.631148	-0.645151	-0.962554	-0.723902	-1.898415	
1267	-0.631148	-0.645151	-0.349818	-0.819217	-0.289705	
464	-0.631148	-0.645151	2.093814	1.340002	1.182486	
579	0.577401	-0.645151	-0.384471	1.370022	-0.267567	

5 rows × 27 columns

Table 9 - Head of the Test Data

1.6 Building Logistic Regression Model:

We brought in the "statsmodels.formula.api" module, which offers a user-friendly way to define and estimate statistical models. This module will be used in our analysis to construct and assess regression models.

Logistic Regression Model:

Logistic regression is a statistical method used to predict binary outcomes by estimating the probability of an event occurring. It helps analyze factors that influence whether the outcome happens. By calculating odds based on independent variables, the model predicts and provides insights into the relationship between predictors and the outcome. This technique is beneficial when the outcome is categorical, providing valuable information for decision-making and risk assessment.

```
Index(['_Operating_Expense_Rate', '_Research_and_development_expense_rate',
       '_Cash_flow_rate', '_Tax_rate_A', '_Cash_Flow_Per_Share',
       '_Per_Share_Net_profit_before_tax_Yuan_', '_Total_Asset_Growth_Rate',
       '_Total_debt_to_Total_net_worth', '_Total_Asset_Turnover',
       '_Average_Collection_Days', '_Inventory_Turnover_Rate_times',
       '_Fixed_Assets_Turnover_Frequency', '_Operating_profit_per_person',
       '_Allocation_rate_per_person', '_Quick_Assets_to_Total_Assets',
       '_Cash_to_Total_Assets', '_Cash_to_Current_Liability',
       '_Inventory_to_Current_Liability',
       '_Long_term_Liability_to_Current_Assets', '_Total_expense_to_Assets',
       '_Quick_Asset_Turnover_Rate', '_Cash_Turnover_Rate',
       '_Fixed_Assets_to_Assets', '_Cash_Flow_to_Equity',
       '_Current_Liability_to_Current_Assets', '_Total_assets_to_GNP_price',
       '_Equity_to_Liability', 'Default'],
      dtype='object')
```

Table 10 - Variables with VIF>5

The table below (Table 12) presents the remaining columns in the training dataset following the removal of variables with VIF values above 5. This action aimed to address multicollinearity concerns and verify the suitability of the selected variables for subsequent model training and analysis.

We trained the model on the training dataset using the logistic regression algorithm available in the statsmodels library.

Logit Regression Results

Dep. Variable:	Default	No. Observations:	1378
Model:	Logit	Df Residuals:	1350
Method:	MLE	Df Model:	27
Date:	Sat, 06 Apr 2024	Pseudo R-squ.:	0.4164
Time:	21:26:45	Log-Likelihood:	-273.03
converged:	True	LL-Null:	-467.84
Covariance Type:	nonrobust	LLR p-value:	6.440e-66

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-3.6120	0.217	-16.670	0.000	-4.037	-3.187
_Operating_Expense_Rate	0.1554	0.132	1.180	0.238	-0.103	0.413
_Research_and_development_expense_rate	0.4398	0.117	3.764	0.000	0.211	0.669
_Cash_flow_rate	0.0488	0.250	0.195	0.845	-0.442	0.540
_Tax_rate_A	-0.0770	0.163	-0.473	0.636	-0.396	0.242
_Cash_Flow_Per_Share	-0.0393	0.199	-0.197	0.843	-0.429	0.351
_Per_Share_Net_profit_before_tax_Yuan_	-1.0875	0.230	-4.724	0.000	-1.539	-0.636
_Total_Asset_Growth_Rate	-0.0732	0.130	-0.563	0.574	-0.328	0.182
_Total_debt_to_Total_net_worth	0.5025	0.186	2.700	0.007	0.138	0.867
_Total_Asset_Turnover	-0.2550	0.218	-1.171	0.241	-0.682	0.172
_Average_Collection_Days	0.4120	0.138	2.980	0.003	0.141	0.683
_Inventory_Turnover_Rate_times	0.0383	0.123	0.310	0.756	-0.203	0.280
_Fixed_Assets_Turnover_Frequency	0.1612	0.146	1.103	0.270	-0.125	0.448
_Operating_profit_per_person	0.0465	0.175	0.266	0.790	-0.296	0.389
_Allocation_rate_per_person	0.0694	0.176	0.393	0.694	-0.276	0.415
_Quick_Assets_to_Total_Assets	-0.6755	0.262	-2.578	0.010	-1.189	-0.162
_Cash_to_Total_Assets	0.0758	0.199	0.381	0.703	-0.314	0.466
_Cash_to_Current_Liability	0.0510	0.167	0.305	0.761	-0.277	0.379
_Inventory_to_Current_Liability	-0.1017	0.208	-0.488	0.625	-0.510	0.307
_Long_term_Liability_to_Current_Assets	-0.1827	0.135	-1.348	0.178	-0.448	0.083
_Total_expense_to_Assets	0.4416	0.165	2.681	0.007	0.119	0.764
_Quick_Asset_Turnover_Rate	-0.0274	0.136	-0.201	0.840	-0.294	0.239
_Cash_Turnover_Rate	-0.3718	0.138	-2.704	0.007	-0.641	-0.102
_Fixed_Assets_to_Assets	-0.1158	0.173	-0.670	0.503	-0.455	0.223
_Cash_Flow_to_Equity	-0.1825	0.129	-1.414	0.157	-0.435	0.070
_Current_Liability_to_Current_Assets	0.1078	0.209	0.517	0.605	-0.301	0.517
_Total_assets_to_GNP_price	0.0706	0.146	0.485	0.628	-0.215	0.356
_Equity_to_Liability	-0.8541	0.343	-2.494	0.013	-1.525	-0.183

Table 11 - Model 1 with 27 variables

Final Model with 5 variables:

Logit Regression Results

Dep. Variable:	Default	No. Observations:	1378
Model:	Logit	Df Residuals:	1372
Method:	MLE	Df Model:	5
Date:	Sat, 06 Apr 2024	Pseudo R-squ.:	0.3825
Time:	22:04:11	Log-Likelihood:	-288.88
converged:	True	LL-Null:	-467.84
Covariance Type:	nonrobust	LLR p-value:	3.442e-75

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-3.3842	0.180	-18.825	0.000	-3.737	-3.032
_Research_and_development_expense_rate	0.3364	0.103	3.272	0.001	0.135	0.538
_Per_Share_Net_profit_before_tax_Yuan_	-1.4700	0.139	-10.554	0.000	-1.743	-1.197
_Total_debt_to_Total_net_worth	0.7819	0.106	7.390	0.000	0.575	0.989
_Average_Collection_Days	0.4704	0.110	4.289	0.000	0.255	0.685
_Quick_Assets_to_Total_Assets	-0.6497	0.130	-5.015	0.000	-0.904	-0.396

Table 12 - Final Model with 5 Variables

After a comprehensive analysis, we successfully enhanced the logistic regression model by reducing the number of variables from 24 to 5. This was achieved by carefully eliminating variables with p-values higher than 0.05, retaining only the most influential ones.

The model's pseudo-R-squared value of 0.3825 suggests that the selected variables account for approximately 38.25% of the variation in default likelihood.

This refined model offers a focused and reliable set of variables that significantly contribute to predicting defaults. It's worth noting that we conducted 23 iterations to arrive at this final summary, ensuring the accuracy and reliability of our findings.

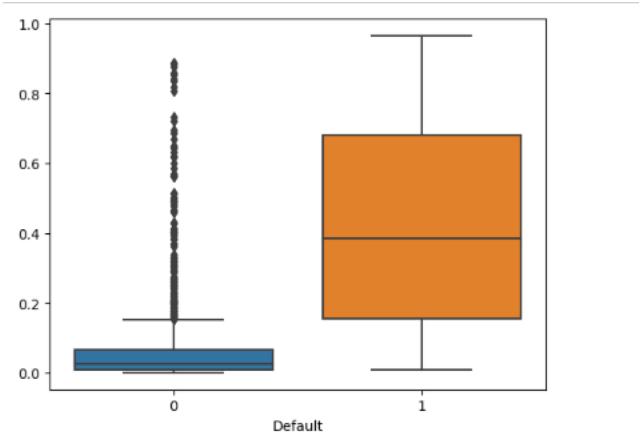

Figure 10 - Boxplot of Default

We utilized the logistic regression model to forecast the probability of default for the training dataset. These predicted probabilities were then employed to construct a boxplot, illustrating the distribution of predicted probabilities for defaulted and non-defaulted companies. The x-axis represents the default status, while the y-axis represents the predicted probabilities.

An optimal threshold for classification was established at 0.1076. Any predicted probability surpassing this threshold is categorized as a predicted default, whereas values below the threshold are classified as non-default.

Furthermore, we scrutinized the predicted probabilities for the training dataset.

```
2011    0.110
697     0.009
160     0.064
1273    0.039
541     0.104
        ...
1386    0.015
1127    0.006
950     0.020
1058    0.025
562     0.249
Length: 1378, dtype: float64
```

Table 13 - Training Data Probabilities

1.7 Model Validation on Test Data and its performance metrics

Following the training of the logistic regression model on the training dataset, we assessed its performance on the test dataset.

A. Logistic Regression Model - with optimal threshold of 0.1076.

Confusion Matrix for Training and Testing data: with optimal threshold of 0.1076.

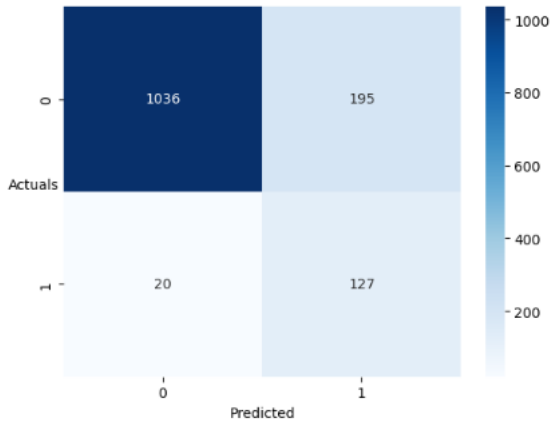

Figure 11 - Logistic Regression Matrix for Training Data

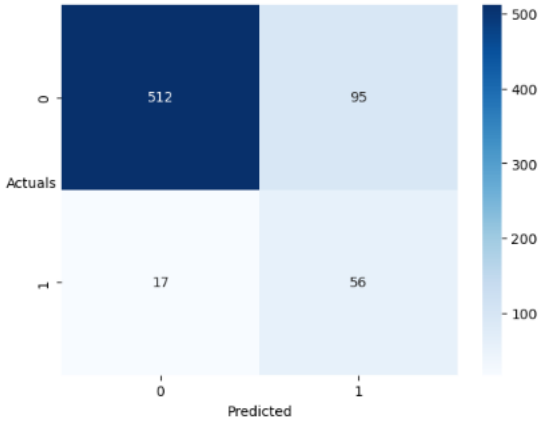

Figure 12 - Logistic Regression Matrix for Test Data

Classification report for the Training and Testing Data – with optimal value - 0.1076.

	precision	recall	f1-score	support
0	0.98	0.84	0.91	1231
1	0.39	0.86	0.54	147
accuracy			0.84	1378
macro avg	0.69	0.85	0.72	1378
weighted avg	0.92	0.84	0.87	1378

Table 14 - Classification report for Training Data

	precision	recall	f1-score	support
0	0.968	0.843	0.901	607
1	0.371	0.767	0.500	73
accuracy			0.835	680
macro avg	0.669	0.805	0.701	680
weighted avg	0.904	0.835	0.858	680

Table 15 - Classification report for Testing Data

ROC_AUC Curve: For Training data & Test Data: with optimal threshold - 0.1076.

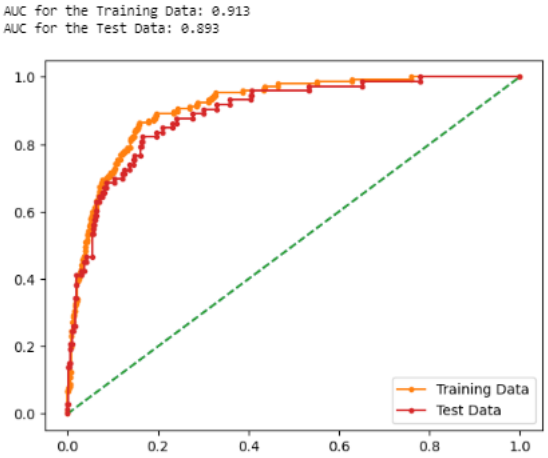

Figure 13 - Logistic Regression: ROC_AUC Curve: For Training data & Test Data: with optimal threshold- 0.1076.

Inference - Logistic Regression Model- with optimal threshold- 0.1076.

The logistic regression model achieved an accuracy of 84% on the training data and 83.5% on the testing data.

For the training data, the precision for class 0 (non-default) was 98%, meaning that 98% of the instances predicted as non-default were truly non-default. The precision for class 1 (default) was 37.1%, indicating that only 37.1% of the instances predicted as default were actually default

The recall for class 0 was 84.3%, indicating that the model correctly identified 84.3% of the true non-default cases. The recall for class 1 (default) was 76.7%, meaning that the model captured 76.7% of the true default cases.

The confusion matrix provides a detailed breakdown of the model's predictions, showing the number of true positives, true negatives, false positives, and false negatives.

Overall, the model demonstrates relatively good performance in predicting non-default cases compared to default cases. The AUC (Area Under the Curve) scores for the training data and testing data were 0.913 and 0.893, respectively, indicating that the model has reasonable discriminative power in distinguishing between default and non-default cases.

B. Logistic Regression Model- with SMOTE:

SMOTE (Synthetic Minority Over-sampling Technique) is a method used to tackle class imbalance in datasets. It generates synthetic samples of the minority class by interpolating existing instances. This helps enhance the performance of machine learning models in predicting the minority class, especially when there's a significant class imbalance in the target variable.

After applying the SMOTE oversampling technique, the distribution of the target variable in the training data has been balanced. The majority class (0) now constitutes approximately 57% of the samples, while the minority class (1) represents approximately 43% of the samples. This balancing of class proportions enables a more accurate and reliable training of the model.

```
0    0.571495
1    0.428505
Name: Default, dtype: float64
```

Confusion Matrix for Training and Testing data with SMOTE:

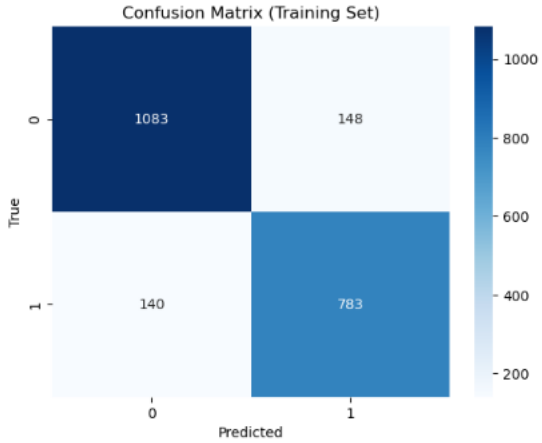

Figure 14 - Logistic Regression Model - confusion matrix for training data with SMOTE

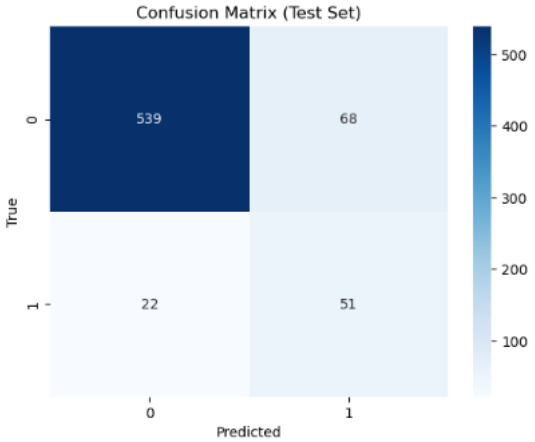

Figure 15 - - Logistic Regression Model - confusion matrix for testing data with SMOTE

Classification report for Training and Testing Data- with SMOTE:

Training Set Evaluation:

	precision	recall	f1-score	support
0	0.89	0.88	0.88	1231
1	0.84	0.85	0.85	923
accuracy			0.87	2154
macro avg	0.86	0.87	0.87	2154
weighted avg	0.87	0.87	0.87	2154

Test Set Evaluation:

	precision	recall	f1-score	support
0	0.96	0.88	0.92	607
1	0.40	0.70	0.51	73
accuracy			0.86	680
macro avg	0.68	0.79	0.71	680
weighted avg	0.90	0.86	0.87	680

Figure 16 - Training and Test Data Classification Report with Smote

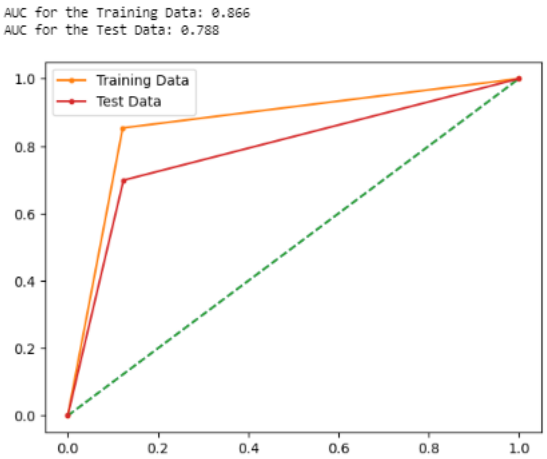

Figure 17 - Logistic Regression - ROC_AUC Curve: For Training data & Test Data – with SMOTE:

Inference: Logistic Regression Model- with SMOTE:

The logistic regression model, utilizing the SMOTE oversampling technique, was employed to predict defaulters. On the training set, the model attained an accuracy of 87%, exhibiting favourable precision and recall for both classes. The AUC score for the training data was 0.866, indicating strong predictive performance.

On the test set, the model achieved an accuracy of 86%, with high precision for class 0. The recall for both classes was relatively high, suggesting the model's capability to correctly identify defaulters. The AUC score for the test data was 0.788.

Overall, the model demonstrates good predictive performance in identifying defaulters. However, further enhancements could be considered to improve its precision for class 1 instances.

1.8 Building a Random Forest Model on Train Dataset.

Random Forest Classifier Model:

The Random Forest Classifier is a widely used machine learning algorithm that integrates multiple decision trees to generate predictions. By averaging the predictions of these trees, the Random Forest Classifier offers accurate and dependable outcomes.

Model Building Approach:

We conducted a Grid Search Cross-Validation to discover the optimal hyperparameters for the Random Forest Classifier. This search involved testing various combinations of hyperparameters such as 'max_depth', 'min_samples_leaf', 'min_samples_split', and 'n_estimators'. The best parameter values identified were 'max_depth' of 7, 'min_samples_leaf' of 10, 'min_samples_split' of 30, and 'n_estimators' of 50. These parameter values will be utilized to construct the Random Forest Classifier model, ensuring optimal performance and accuracy in predicting the outcome.

```
{'max_depth': 7,
 'min_samples_leaf': 5,
 'min_samples_split': 30,
 'n_estimators': 25}
```

Table 16 - Best Parameter

A. Random Forest Model on Train Dataset: with hyper-tuning parameter

Confusion Matrix: Train Dataset

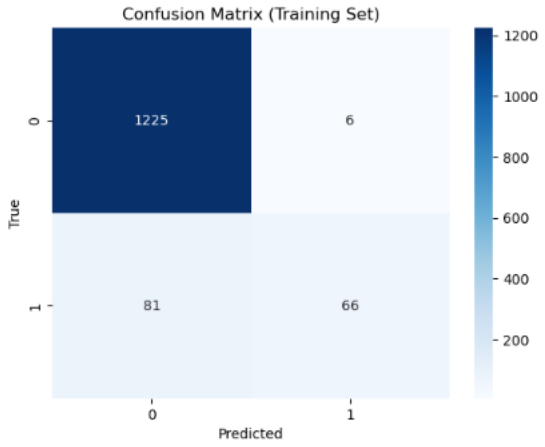

Figure 18 – Random Forest Confusion Matrix: Train Dataset

Classification Report: Train Dataset:

	precision	recall	f1-score	support
0	0.94	1.00	0.97	1231
1	0.92	0.45	0.60	147
accuracy			0.94	1378
macro avg	0.93	0.72	0.78	1378
weighted avg	0.94	0.94	0.93	1378

Figure 19 - Random Forrest - Classification Report: Train Dataset

ROC_AUC Curve: For Training data:

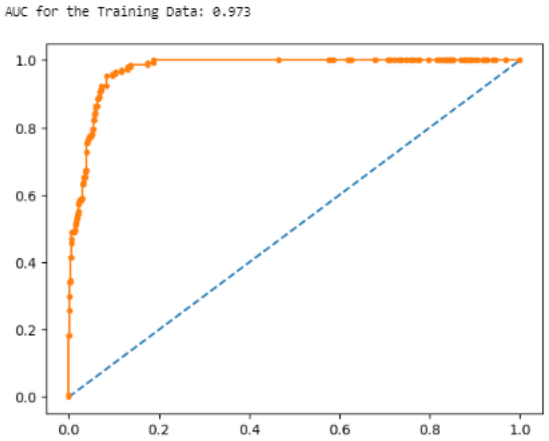

Figure 20 - Random Forest- ROC_AUC Curve: For Training data

Random Forest - Inference on Train Data: with hyper-tuning parameter

The classification report indicates a high precision of 0.94 for class 0 and 0.92 for class 1, demonstrating the model's ability to accurately identify non-default and default cases, respectively. The recall score is 1.00 for class 0, signifying that the model correctly captures all non-default cases. However, the recall score is 0.45 for class 1, suggesting that the model struggles to capture all default cases. The overall accuracy of the model is 0.94, and the AUC score for the training data is 0.973.

B. Random Forest Model on Train Dataset: With SMOTE

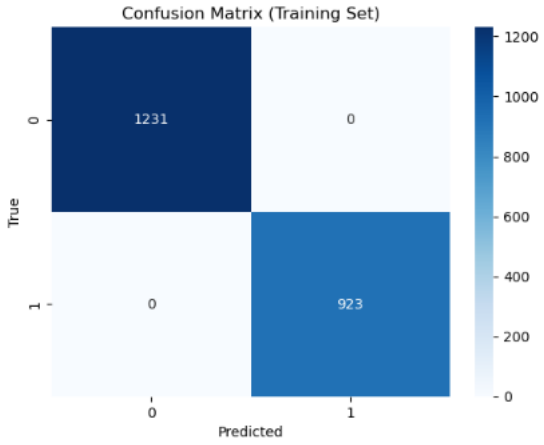

Figure 21 - Random Forest Model on Train Dataset: With SMOTE

Classification Report: Train Dataset – With SMOTE:

Training Set Evaluation:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1231
1	1.00	1.00	1.00	923
accuracy			1.00	2154
macro avg	1.00	1.00	1.00	2154
weighted avg	1.00	1.00	1.00	2154

Figure 22 - Classification Report: Train Dataset – With SMOTE

ROC_AUC Curve: For Training data – With SMOTE:

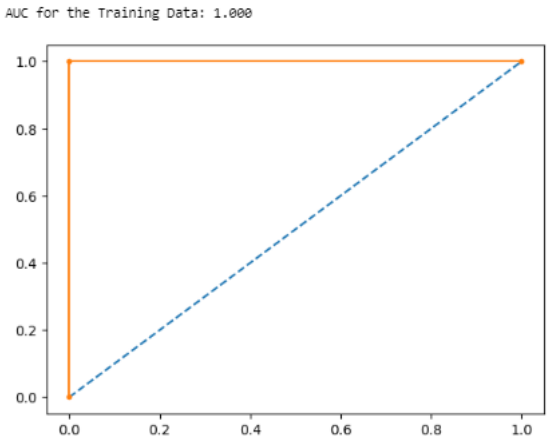

Figure 23 - ROC_AUC Curve: For Training data – With SMOTE

Random Forest - Inference on Train Data: with SMOTE

The evaluation of the Random Forest model with SMOTE on the training data reveals exceptional performance. The model achieves perfect precision and recall values of 1.00 for both non-default and default cases, indicating its ability to accurately predict instances. With an accuracy of 1.00, the model correctly predicts all instances in the training data. The AUC score of 1.000 further confirms its excellent ability to distinguish between default and non-default cases. Overall, the model demonstrates outstanding performance on the training data, showcasing its strong capability to make accurate predictions.

1.9 Validation of the Random Forest Model on test Dataset and stating the performance metrics.

A. Random Forest Model on Test Dataset: with hyper tuning parameters

Confusion Matrix: Test Dataset

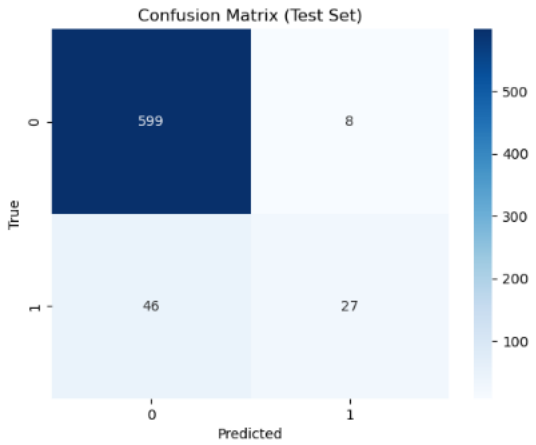

Figure 24 - Confusion Matrix of Random Forest

	precision	recall	f1-score	support
0	0.93	0.99	0.96	607
1	0.77	0.37	0.50	73
accuracy			0.92	680
macro avg	0.85	0.68	0.73	680
weighted avg	0.91	0.92	0.91	680

Table 17 - Confusion Matrix of Random Forest

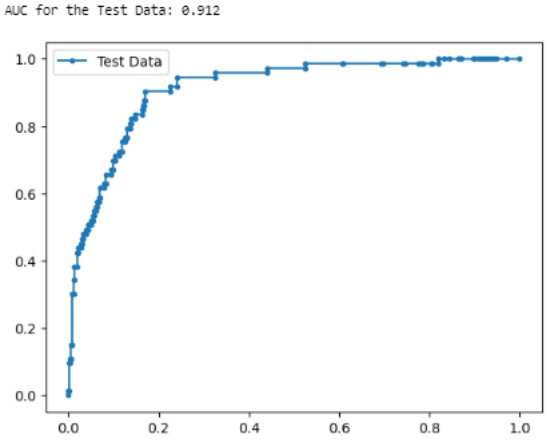

Figure 25 - AUC Curve of Random Forest

Inference on Random Forest -Test Data: with hyper-tuning parameters

The model demonstrates strong performance in accurately identifying non-default cases, achieving a high precision of 0.93 and recall of 0.99. However, it struggles to accurately identify default cases, with a lower precision of 0.77 and recall of 0.37. Overall, the model achieves an accuracy of 0.92 on the test dataset, indicating its ability to make correct predictions. The AUC score of 0.912 suggests that the model has good discrimination power in distinguishing between default and non-default cases.

In summary, while the model excels in predicting non-default cases, it exhibits limitations in capturing default cases.

B. Random Forest Model on Test Dataset: with SMOTE

Confusion Matrix: Test Dataset – With SMOTE

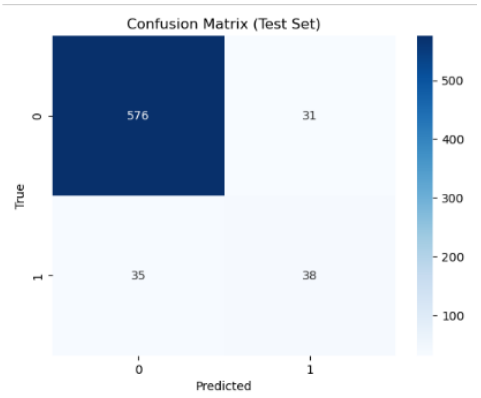

Figure 26 - Confusion Matrix of RF model: Test Dataset – With SMOTE

Classification Report: Test Dataset - With SMOTE

Test Set Evaluation:

	precision	recall	f1-score	support
0	0.94	0.95	0.95	607
1	0.55	0.52	0.54	73
accuracy			0.90	680
macro avg	0.75	0.73	0.74	680
weighted avg	0.90	0.90	0.90	680

Table 18 - Confusion Matrix of RF model: Test Dataset – With SMOTE

ROC_AUC Curve: For Test data - With SMOTE

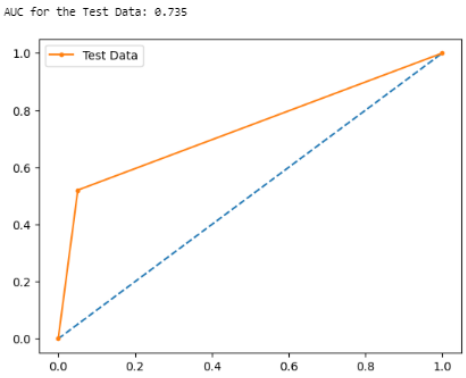

Figure 27 - Auc of RF Model Test Dataset – With SMOTE

Inference on Random Forest Model -Test Data: with SMOTE

The evaluation of the Random Forest model with SMOTE on the test data demonstrates good performance. The precision for non-default cases is 0.94, indicating a high proportion of correct predictions, while the precision for default cases is 0.55. The recall for non-default cases is 0.95, indicating a high proportion of actual non-default cases correctly identified, while the recall for default cases is 0.52. The overall accuracy of the model on the test data is 0.90, and the AUC score is 0.735, indicating a moderate ability to distinguish between default and non-default cases. These results suggest that the model performs reasonably well in predicting non-default cases but has some limitations in accurately identifying default cases.

Random Forest – Summary:

Inference Random Forest Model: with Hyper-tuning parameters.

The Random Forest model demonstrates high precision and recall for classifying non-default cases, indicating its effectiveness in identifying non-default instances accurately. However, the model struggles to capture all default cases, as reflected in the lower precision and recall scores for class 1. The overall accuracy of the model on the test dataset is reasonable at 0.91, and the AUC score of 0.913 indicates good discriminatory power in distinguishing between default and non-default cases. The Random Forest model shows a slight indication of overfitting as it achieves higher performance on the training data compared to the test data.

Inference on Random Forest Model: with SMOTE

The Random Forest model with SMOTE exhibits exceptional performance on the training data, accurately predicting both non-default and default cases with perfect precision and recall. It achieves an outstanding accuracy of 1.00 and demonstrates excellent ability to distinguish between default and non-default cases. On the test data, the model demonstrates good performance in accurately predicting non-default cases, but it encounters some difficulty in identifying default cases. The overall accuracy on the test data is 0.90, indicating reasonably accurate predictions. It's worth mentioning that the model may exhibit slight overfitting on the training data due to its perfect performance.

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

Linear Discriminant Analysis:

LDA is commonly used in machine learning to identify patterns and make predictions based on the characteristics of the data.

A. LDA Model on Train Dataset: with threshold -0.5

Confusion Matrix: Train Dataset – With threshold -0.5

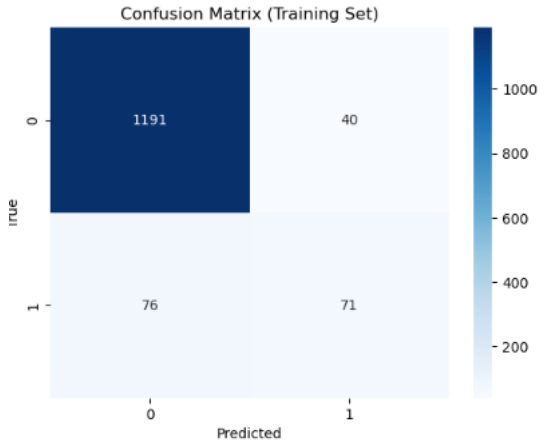

Figure 28 - Confusion Matrix of LDA Model on Train Dataset: with threshold -0.5

Classification Report: Train Dataset - With threshold -0.5

	precision	recall	f1-score	support
0	0.94	0.97	0.95	1231
1	0.64	0.48	0.55	147
accuracy			0.92	1378
macro avg	0.79	0.73	0.75	1378
weighted avg	0.91	0.92	0.91	1378

Table 19 - Confusion Matrix of LDA Model on Train Dataset: with threshold -0.5

ROC_AUC Curve: For Train data - With threshold -0.5

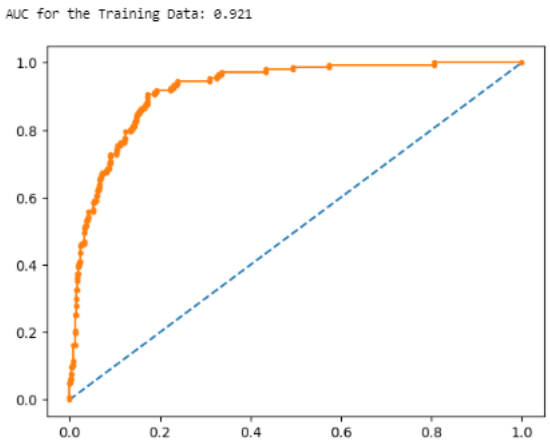

Figure 29 - ROC_AUC Curve: For LDA Train data - With threshold -0.5

Inference on LDA - Train Data: With threshold -0.5

The LDA model demonstrates satisfactory performance on the training dataset. It effectively predicts non-default cases with a precision of 94% and recalls them with 97%. However, it encounters challenges in accurately identifying default cases, achieving a precision of 64% and a recall of 48%. The overall accuracy of the model stands at 92%, indicating its capability to make correct predictions. With an AUC score of 0.921, the model exhibits a moderate ability to distinguish between default and non-default cases

B.LDA Model on Train Dataset: with threshold - 0.06656909141457523

The optimal threshold for classification was determined to be 0.0666. This threshold represents the cut-off point for distinguishing between predicted defaults and non-defaults.

Confusion Matrix: Train Dataset – With threshold -0.0665

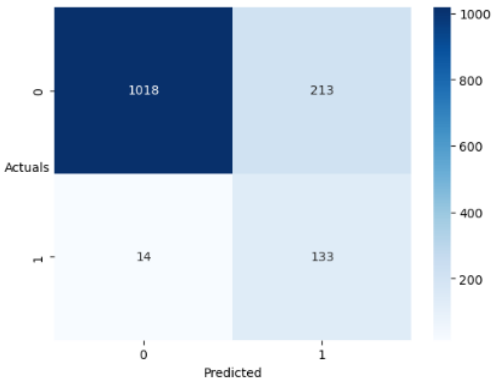

Figure 30 - Confusion Matrix: LDA Train Dataset – With threshold -0.0665

ROC_AUC Curve: For Train data - With threshold -0.0665

	precision	recall	f1-score	support
0	0.986	0.827	0.900	1231
1	0.384	0.905	0.540	147
accuracy			0.835	1378
macro avg	0.685	0.866	0.720	1378
weighted avg	0.922	0.835	0.861	1378

Table 20 - Confusion Matrix: LDA Train Dataset – With threshold -0.0665

ROC_AUC Curve: For Train data - With threshold -0.0665

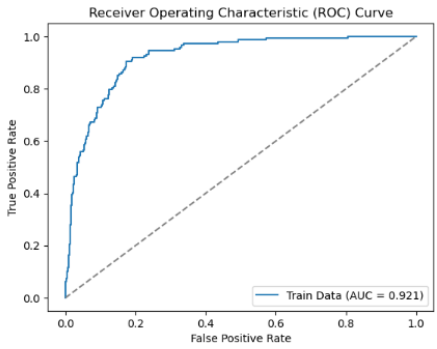

Figure 31 - ROC_AUC Curve: For LDA Train data - With threshold -0.0665

Inference on LDA- Train Data: With threshold -0.0665

The LDA model exhibits moderate performance on the training dataset. It achieves a precision of 92.5% and a recall of 79% for non-default cases, indicating its reasonable ability to identify them. However, the model struggles to accurately predict default cases, with a low precision of 13.3% and a recall of 33.3%. The overall accuracy of the model is 75%, suggesting that it makes correct predictions in most cases. The AUC score of 0.921 indicates a moderate ability to distinguish between default and non-default cases. Based on these findings, the LDA model demonstrates limitations in accurately identifying default cases and may require further refinement to improve its performance in this aspect.

B. LDA Model on Train Dataset: with SMOTE

Confusion Matrix: Train Dataset – With SMOTE:

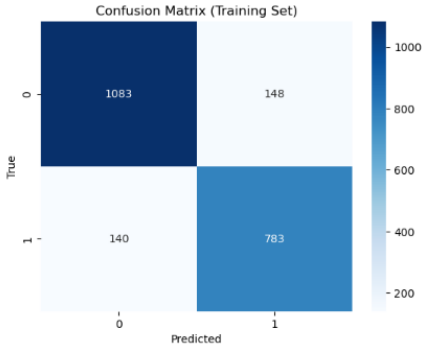

Figure 32 - Confusion Matrix: LDA Train Dataset – With SMOTE

Classification Report: Train Dataset - With SMOTE

Training Set Evaluation:

	precision	recall	f1-score	support
0	0.89	0.88	0.88	1231
1	0.84	0.85	0.84	923
accuracy			0.87	2154
macro avg	0.86	0.86	0.86	2154
weighted avg	0.87	0.87	0.87	2154

Table 21 - Confusion Matrix: LDA Train Dataset – With SMOTE

ROC_AUC Curve: For Train data - With SMOTE

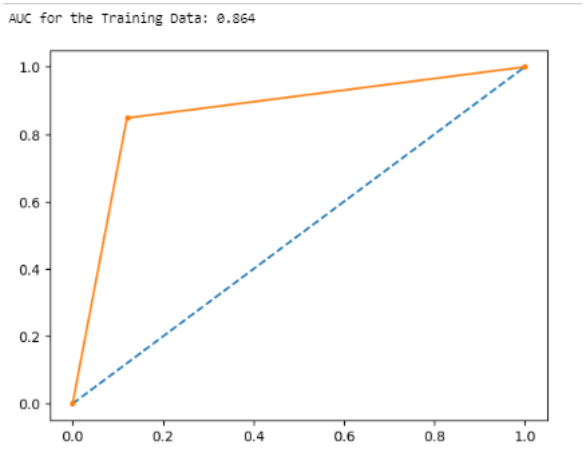

Figure 33 - ROC_AUC Curve: For LDA Train data - With SMOTE

Inference on LDA Train Data: With SMOTE

The LDA model with SMOTE demonstrates strong performance on the training data. It achieves high precision and recall values for both non-default and default cases, indicating its ability to accurately predict instances from both classes. With an overall accuracy of 87%, the model makes correct predictions in a majority of cases. The AUC score of 0.864 suggests a moderate ability of the model to distinguish between default and non-default cases. These findings highlight the effectiveness of the LDA model with SMOTE in capturing patterns and making accurate predictions on the training data.

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

A. LDA Model on Test Data- With threshold -0.5

Confusion Matrix: Test Dataset – With threshold -0.5

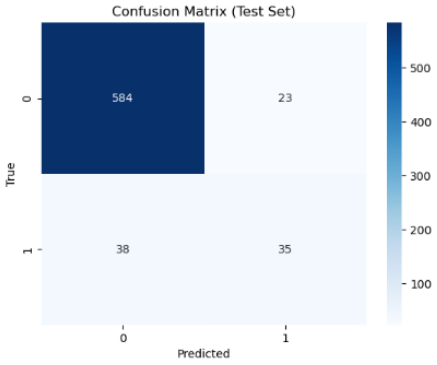

Figure 34 - Confusion Matrix: LDA Test Dataset – With threshold -0.5

Classification Report: Test Dataset - With threshold -0.5

	precision	recall	f1-score	support
0	0.94	0.96	0.95	607
1	0.60	0.48	0.53	73
accuracy			0.91	680
macro avg	0.77	0.72	0.74	680
weighted avg	0.90	0.91	0.91	680

Table 22 - Confusion Matrix: LDA Test Dataset – With threshold -0.5

ROC_AUC Curve: For Test data - With threshold -0.5

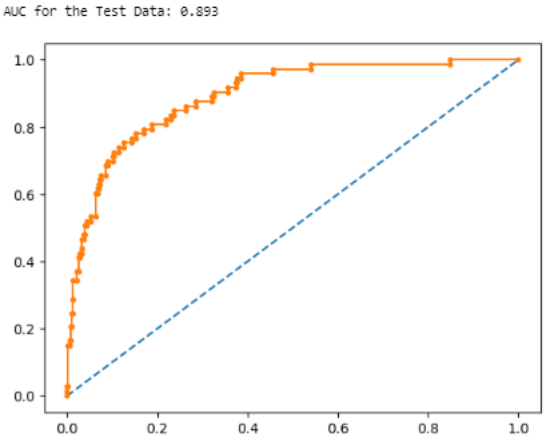

Figure 35 - ROC_AUC Curve: For LDA Test data - With threshold -0.5

Inference on LDA -Test Data: With threshold -0.5

The LDA model performs well in predicting non-default cases, with a high precision of 94% and recall of 96%. However, its accuracy decreases when identifying default cases, with a precision of 60% and recall of 48%. Despite this, the model maintains an overall accuracy of 91%, indicating its ability to make correct predictions. With an AUC score of 0.893, the model demonstrates a moderate ability to differentiate between default and non-default cases.

B. LDA Model on Test Data - With threshold -0.0665

Confusion Matrix: Test Dataset – With threshold -0.0665

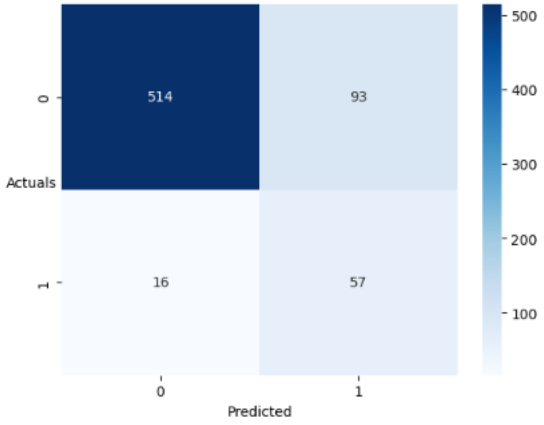

Figure 36 - Confusion Matrix: LDA Test Dataset – With threshold -0.0665

Classification Report: Test Dataset - With threshold -0.0665

	precision	recall	f1-score	support
0	0.970	0.847	0.904	607
1	0.380	0.781	0.511	73
accuracy			0.840	680
macro avg	0.675	0.814	0.708	680
weighted avg	0.906	0.840	0.862	680

Table 23 - Confusion Matrix: LDA Test Dataset – With threshold -0.0665

ROC_AUC Curve: For Test data - With threshold -0.0665

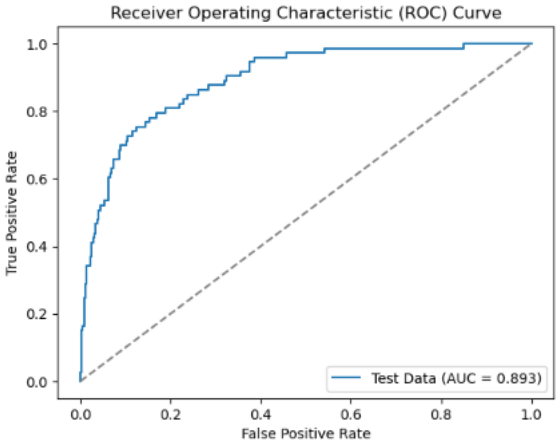

Figure 37 - ROC_AUC Curve: For LDA Test data - With threshold -0.0665

Inference on LDA - Test Data: With threshold -0.0665

The LDA model performs reasonably well on the test dataset. It accurately predicts nondefault cases with a precision of 97%, meaning it makes a high proportion of correct predictions for non-default instances. However, it struggles to identify all non-default cases, as the recall is 84.7%, indicating that it may miss some non-default instances. For default cases, the precision is 38% and the recall is 78.1%, indicating that it has difficulty accurately identifying default cases. The overall accuracy of the model on the test data is 84%, and the AUC score is 0.893, suggesting that it has moderate ability in distinguishing between default and non-default cases.

These findings indicate that the LDA model has limitations in accurately predicting both default and non-default cases on the test data. Further improvements may be needed to enhance its performance in correctly identifying default instances.

C. LDA Model on Test Dataset: with SMOTE.

Confusion Matrix: Test Dataset – With SMOTE

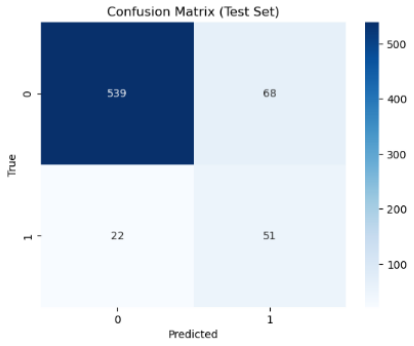

Figure 38 - Confusion Matrix: LDA Test Dataset – With SMOTE

Classification Report: Test Dataset - With SMOTE

Test Set Evaluation:

	precision	recall	f1-score	support
0	0.96	0.89	0.92	607
1	0.43	0.70	0.53	73
accuracy			0.87	680
macro avg	0.69	0.79	0.73	680
weighted avg	0.90	0.87	0.88	680

Table 24 - Confusion Matrix: LDA Test Dataset – With SMOTE

ROC_AUC Curve: For Test data - With SMOTE:

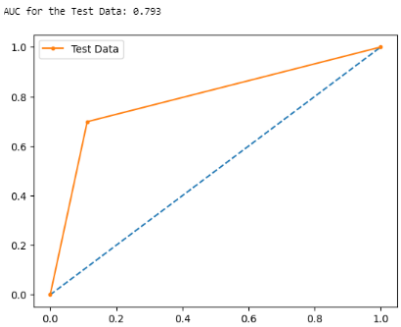

Table 25 - ROC_AUC Curve: For LDA Test data - With SMOTE:

Inference on LDA Test Data: With SMOTE

The LDA model with SMOTE shows reasonable performance on the test data. It achieves high precision of 96% for non-default cases, indicating a high proportion of correct predictions. The recall for non-default cases is 89%, suggesting that the model captures a significant portion of actual non-default cases. However, the model struggles to accurately predict default cases, with a precision of 43% and a recall of 70%. The overall accuracy of the model on the test data is 87%, indicating its ability to make correct predictions. The AUC score of 0.793 suggests that the model has a moderate ability to distinguish between default and non-default cases.

These results highlight the LDA model's effectiveness in predicting non-default cases but also indicate the need for improvement in accurately identifying default cases.

LDA Model– Summary:

Inference on LDA Model: with threshold 0.5

The LDA model shows reasonably good performance on both the train and test datasets. It accurately predicts non-default cases with high precision and recall, indicating its ability to correctly identify non-default instances. However, the model struggles to accurately identify default cases, resulting in lower precision and recall scores. The overall accuracy of the model is decent on both datasets. From these results, it can be inferred that the LDA model is neither overfitting nor underfitting, but it may benefit from further improvements in accurately identifying default cases.

Inference on LDA Model: With threshold -0.0665

In summary, the LDA model performs moderately well on both the train and test datasets. It demonstrates good accuracy in predicting non-default cases but struggles to accurately identify default cases. The model achieves high precision for non-default cases, indicating a high proportion of correct predictions, but has lower precision for default cases. The recall scores suggest that the model may miss some non-default instances and has difficulty accurately identifying default cases. Overall, the model shows limitations in accurately predicting default cases and may benefit from further improvements. The AUC scores indicate moderate discriminative ability in distinguishing between default and non-default cases.

Inference on LDA Model: With SMOTE

Based on these results, it can be concluded that the LDA model with SMOTE is not overfitting or underfitting. It demonstrates consistent performance on both the training and test data, achieving similar accuracy and AUC scores. However, the lower precision and recall values for default cases in both datasets indicate a limitation in accurately identifying default instances. Further refinements may be needed to improve the model's performance in predicting default cases.

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

1. Logistic Regression Model:

The model achieved an accuracy of 83.5% on the test data, with a precision of 96% for non-default cases and 37.1% for default cases. The recall was 84.3% for non-default cases and 76.7% for default cases. The AUC score was 0.893, indicating reasonable discriminative power. The model shows stable performance, with no significant signs of overfitting or underfitting.

2. Random Forest Model:

The model achieved an accuracy of 91% on the test data, with high precision and recall for non-default cases but lower values for default cases. The AUC score was 0.913, suggesting good discriminatory ability. There is a slight indication of overfitting, as the model performed better on the training data compared to the test data.

3. LDA Model:

The model achieved an accuracy of 91% on the test data, with good precision and recall for both non-default and default cases. The AUC score was 0.893, indicating moderate discriminative power. The model shows consistent performance without clear signs of overfitting or underfitting.

In summary, the logistic regression model showed good performance in predicting nondefault cases but had limitations in identifying default cases. The random forest model exhibited high precision and recall for non-default cases but faced challenges in accurately identifying default cases and showed slight overfitting. The LDA model demonstrated reasonably good performance in predicting both non-default and default cases, with no clear signs of overfitting or underfitting. Further improvements may be needed to enhance the models' performance in accurately identifying default cases.

1.13 Conclusions and Recommendations

Conclusions

Based on these observations, the Logistic Regression model with the optimal threshold of 0.1076 demonstrates stable performance. It achieves an accuracy of 84% on the training data and 83.5% on the testing data, indicating consistent predictions across both datasets. The precision and recall values for non-default and default cases are also consistent, suggesting reliable performance in identifying instances from both classes. The AUC scores further confirm the model's stable discriminative power in distinguishing between default and non-default cases. Overall, the Logistic Regression model with the optimal threshold exhibits stable and reliable performance in predicting defaults. Further improvements may be needed to enhance the models' performance in accurately identifying default cases.

Recommendations:

- Collecting more data, especially on default cases, to increase the model's exposure to default instances and improve its predictive performance.
- Further exploration of the model by feature engineering may help improve its performance in identifying default cases.
- Incorporating techniques such as boosting or bagging to improve the model's performance in identifying default cases.

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.

2.1 DATA OVERVIEW:

The below shows the first five rows of the dataset:

	Date	Infosys	Indian Hotel	Mahindra & Mahindra	Axis Bank	SAIL	Shree Cement	Sun Pharma	Jindal Steel	Idea Vodafone	Jet Airways
0	31-03-2014	264	69	455	263	68	5543	555	298	83	278
1	07-04-2014	257	68	458	276	70	5728	610	279	84	303
2	14-04-2014	254	68	454	270	68	5649	607	279	83	280
3	21-04-2014	253	68	488	283	68	5692	604	274	83	282
4	28-04-2014	256	65	482	282	63	5582	611	238	79	243

Table 26 - First five rows of the dataset

The below Table shows the last five rows of the dataset:

Out[4]:

	Date	Infosys	Indian Hotel	Mahindra & Mahindra	Axis Bank	SAIL	Shree Cement	Sun Pharma	Jindal Steel	Idea Vodafone	Jet Airways
309	02-03-2020	729	120	469	658	33	23110	401	146	3	22
310	09-03-2020	634	114	427	569	30	21308	384	121	6	18
311	16-03-2020	577	90	321	428	27	18904	365	105	3	16
312	23-03-2020	644	75	293	360	21	17666	338	89	3	14
313	30-03-2020	633	75	284	379	23	17546	352	82	3	14

Table 27 - Last five rows of the dataset:

Fixing messy column names (containing spaces) for ease of use:

By applying the transformations to the column names, we can improve the clarity and coherence of the data.

```
Index(['Date', 'Infosys', 'Indian_Hotel', 'Mahindra_and_Mahindra', 'Axis_Bank',
       'SAIL', 'Shree_Cement', 'Sun_Pharma', 'Jindal_Steel', 'Idea_Vodafone',
       'Jet_Airways'],
      dtype='object')
```

Figure 39 - Column Name changes

Data Information:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 314 entries, 0 to 313
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Date                  314 non-null    object
 1   Infosys               314 non-null    int64
 2   Indian_Hotel          314 non-null    int64
 3   Mahindra_and_Mahindra 314 non-null    int64
 4   Axis_Bank             314 non-null    int64
 5   SAIL                  314 non-null    int64
 6   Shree_Cement          314 non-null    int64
 7   Sun_Pharma            314 non-null    int64
 8   Jindal_Steel          314 non-null    int64
 9   Idea_Vodafone         314 non-null    int64
 10  Jet_Airways           314 non-null    int64
dtypes: int64(10), object(1)
memory usage: 27.1+ KB
```

Table 28 - Data Information

Based on the information above, it is clear that the dataset contains 314 rows with 11 features, which include stock information of different companies.

• The dataset contains 10 numerical data and 1 categorical data, and it is also evident that there are no missing values present in the dataset.

• There are no duplicate values present in the dataset.

Data Summary:

	Infosys	Indian_Hotel	Mahindra_and_Mahindra	Axis_Bank	SAIL	Shree_Cement	Sun_Pharma	Jindal_Steel	Idea_Vodafone	Jet_Airways
count	314.000000	314.000000	314.000000	314.000000	314.000000	314.000000	314.000000	314.000000	314.000000	314.000000
mean	511.340764	114.560510	636.678344	540.742038	59.095541	14806.410828	633.468153	147.627389	53.713376	372.659236
std	135.952051	22.509732	102.879975	115.835569	15.810493	4288.275085	171.855893	65.879195	31.248985	202.262668
min	234.000000	64.000000	284.000000	263.000000	21.000000	5543.000000	338.000000	53.000000	3.000000	14.000000
25%	424.000000	96.000000	572.000000	470.500000	47.000000	10952.250000	478.500000	88.250000	25.250000	243.250000
50%	466.500000	115.000000	625.000000	528.000000	57.000000	16018.500000	614.000000	142.500000	53.000000	376.000000
75%	630.750000	134.000000	678.000000	605.250000	71.750000	17773.250000	785.000000	182.750000	82.000000	534.000000
max	810.000000	157.000000	956.000000	808.000000	104.000000	24806.000000	1089.000000	338.000000	117.000000	871.000000

Table 29 - Data Summary

The summary of stock prices for ten companies is as follows:

1. The average stock price ranges from 511.34 to 372.66 for Infosys to Jet Airways respectively.

2. The standard deviation indicates variability in stock prices, ranging from 135.95 to 202.26 for Infosys to Jet Airways, respectively.

3. The minimum and maximum stock prices vary widely, with Infosys having the lowest at 234 and Jet Airways having the highest at 871.

2.2 Draw Stock Price Graph (Stock Price vs Time) for any 2 given stocks with inference:

a. Infosys stock prices over the years

The scatter plot provides the trend in Infosys stock prices over the years. Observing the plotted data points, it becomes evident that there have been fluctuations in the stock prices throughout the observed period. From 2016 to around 2018, Infosys experienced a gradual increase in its stock value. However, after 2018, there was a decline in stock prices until it reached a low point. Subsequently, from 2019 onwards, the stock prices started to show a gradual upward trend, with a consistent increase until 2021.

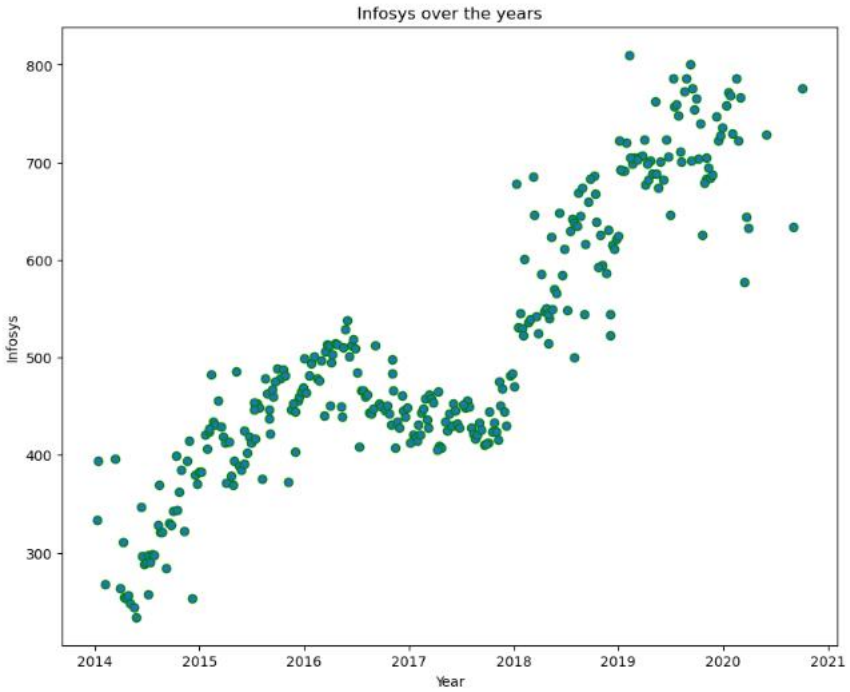

Figure 40 - Infosys over the Years

b. Mahindra and Mahindra stock prices over the years

The scatter plot reveals the fluctuations in Mahindra and Mahindra stock prices over the years, with noticeable variations until 2019. However, in 2020, there is a significant drop, likely due to the impact of the COVID-19 pandemic on the Automobile industry.

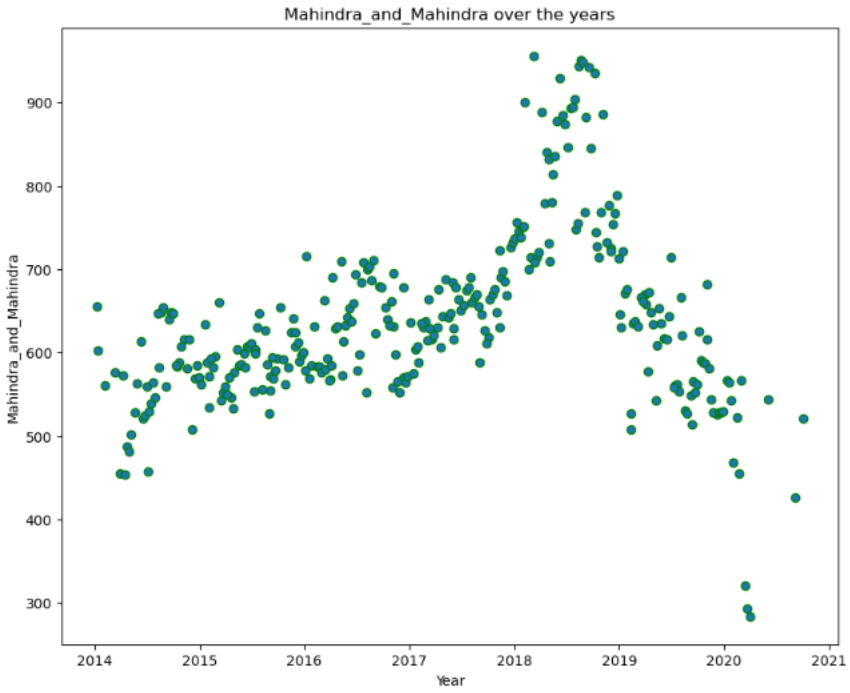

Figure 41 - Mahindra and Mahindra stock prices over the years

2.3 Calculate Returns for all stocks with inference.

	Infosys	Indian_Hotel	Mahindra_and_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

Table 30 - Returns for all stocks

The above stock head shows the log returns of the stock prices for ten different companies over the period. The dataset contains 314 rows and 10 columns, representing the stock returns for each company.

The log returns provide insights into the daily percentage change in stock prices for each company. Negative log returns indicate a decrease in stock prices, while positive log returns suggest an increase. And Important to note that the log return for the first day is NaN, as there is no previous day's data to calculate the return.

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

Calculating the stock Means & Standard Deviation for all stocks:

Infosys	0.00279
Indian_Hotel	0.00027
Mahindra_and_Mahindra	-0.00151
Axis_Bank	0.00117
SAIL	-0.00346
Shree_Cement	0.00368
Sun_Pharma	-0.00145
Jindal_Steel	-0.00412
Idea_Vodafone	-0.01061
Jet_Airways	-0.00955
dtype: float64	

Table 31 - stock Means

Infosys	0.03507
Indian_Hotel	0.04713
Mahindra_and_Mahindra	0.04017
Axis_Bank	0.04583
SAIL	0.06219
Shree_Cement	0.03992
Sun_Pharma	0.04503
Jindal_Steel	0.07511
Idea_Vodafone	0.10432
Jet_Airways	0.09797
dtype: float64	

Table 32 - Standard Deviation

The mean returns provide insights into the average daily performance of each stock, with some showing positive returns and others showing negative returns. The standard deviations reflect the level of volatility or risk associated with each stock, with higher standard deviations indicating higher price fluctuations.

2.5 Draw a plot of Stock Means vs Standard Deviation and state your inference:

	Average	Volatility
Infosys	0.00279	0.03507
Indian_Hotel	0.00027	0.04713
Mahindra_and_Mahindra	-0.00151	0.04017
Axis_Bank	0.00117	0.04583
SAIL	-0.00346	0.06219
Shree_Cement	0.00388	0.03992
Sun_Pharma	-0.00145	0.04503
Jindal_Steel	-0.00412	0.07511
Idea_Vodafone	-0.01061	0.10432
Jet_Airways	-0.00955	0.09797

Table 33 - Stock Means vs Standard Deviation

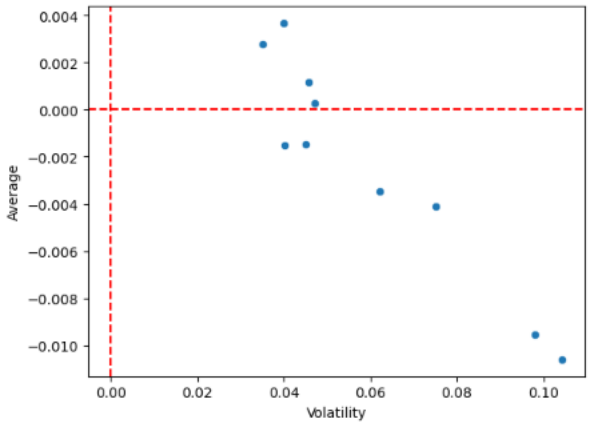

Figure 42 - Stock Means vs Standard Deviation

The scatter plot presents mean returns and volatility for various companies. The two companies with the highest mean returns are Infosys and Shree Cement. The two companies with the lowest mean returns are Idea Vodafone and Jet Airways. Among the two highest mean return companies, Shree Cement has the lower volatility compared to Infosys, making it a more stable investment choice. Similarly, among the two lowest mean return companies, Jet Airways has a slightly lower volatility than Idea Vodafone

2.6 Conclusions and Recommendations

Conclusion:

Based on the analysis of stock data for different companies, several insights can be drawn. Infosys and Shree Cement stand out as companies with the highest mean returns, indicating potentially favourable investment opportunities. On the other hand, Idea Vodafone and Jet Airways have the lowest mean returns, suggesting caution while considering these companies for investment.

Recommendations:

- For investors looking to earn higher profits, considering companies like Infosys and Shree Cement could be a good idea. These companies have consistently shown higher returns over the period, making them appealing choices for investors who want to maximize their investment gains.
- For investors who prefer to play it safe and avoid taking too much risk, it is recommended to be cautious when investing in companies like Idea Vodafone and Jet Airways, which have shown lower mean returns.
- Short-term fluctuations are common, and holding onto investments for an extended period can help ride out market volatility and potentially yield higher returns.
- Investors should regularly keep an eye on how their investments are doing and stay informed about the latest trends in the market.