FINANCE & RISK ANALYTICS – PROJECT

BATCH: PGPDSBA.O.SEP22.A

Yaresh Vijayasundaram
POST GRADUATE PROGRAM IN DATA SCIENCE AND BUSINESS ANALYTICS | GREAT LEARNING

Table of Contents

List Of Figures:

List of Tables:

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. 1 shows the first five rows of the dataset:

	Co_Code	Co_Name	_Operating_Expense_Rate	_Research_and_development_expense_rate	_Cash_flow_rate	_Interest_bearing_debt_interest_rate	_Tax_rate_
0	16974	Hind.Cables	8.820000e+09	0.000000e+00	0.462045	0.000352	0.0014
1	21214	Tata Tele. Mah.	9.380000e+09	4.230000e+09	0.460116	0.000716	0.0000
2	14852	ABG Shipyard	3.800000e+09	8.150000e+08	0.449893	0.000496	0.0000
3	2439	GTL	6.440000e+09	0.000000e+00	0.462731	0.000592	0.0093
4	23505	Bharati Defence	3.680000e+09	0.000000e+00	0.463117	0.000782	0.4002

Table 1: First five rows of the Dataset

The below Table.2 shows the last five rows of the dataset:

	Co_Code	Co_Name	_Operating_Expense_Rate	_Research_and_development_expense_rate	_Cash_flow_rate	_Interest_bearing_debt_interest_rate	_Tax_
2053	2743	Kothari Ferment.	3.021580e-04	6.490000e+09	0.477066	0.000000	0.1
2054	21216	Firstobj.Tech.	1.371450e-04	0.000000e+00	0.465211	0.000658	0.0
2055	142	Diamines & Chem.	2.114990e-04	8.370000e+09	0.480248	0.000502	0.0
2056	18014	IL&FS Engg.	3.750000e+09	0.000000e+00	0.474670	0.000578	0.3
2057	43229	Channel Nine	2.981110e-04	0.000000e+00	0.467203	0.000826	0.0

Table 2: Last five rows of the Dataset

Data Information:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2058 entries, 0 to 2057
Data columns (total 58 columns):
 #   Column                                                Non-Null Count  Dtype
---  ------                                                --------------  -----
 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
dtypes: float64(53), int64(4), object(1)
memory usage: 932.7+ KB
```

Table 3: Data Information

• Based on the information above, it is clear that the dataset contains 2,058 rows with 58 features, which include various financial and performance indicators of different companies.
• The dataset contains 57 numerical data and 1 categorical data, and it is also evident that there are some missing values present in the dataset.
• The dataset includes a column named "Default," which is the target variable. This column contains binary values (0 and 1) indicating whether a company has defaulted or not. This variable could be used for building a predictive model or conducting further analysis related to company defaults.

Data Summary:

	count	mean	std	min	25%	50%	75%	max
Co_Code	2058.00	17572.11	21892.89	4.00	3674.00	6240.00	24280.75	72493.00
_Operating_Expense_Rate	2058.00	2052388835.76	3252623690.29	0.00	0.00	0.00	4110000000.00	9980000000.00
_Research_and_development_expense_rate	2058.00	1208634256.56	2144568158.08	0.00	0.00	0.00	1550000000.00	9980000000.00
_Cash_flow_rate	2058.00	0.47	0.02	0.00	0.46	0.46	0.47	1.00
_Interest_bearing_debt_interest_rate	2058.00	11130223.52	90425949.04	0.00	0.00	0.00	0.00	990000000.00
_Tax_rate_A	2058.00	0.11	0.15	0.00	0.00	0.04	0.22	1.00
_Cash_Flow_Per_Share	1891.00	0.32	0.02	0.17	0.31	0.32	0.33	0.46
_Per_Share_Net_profit_before_tax_Yuan_	2058.00	0.18	0.03	0.00	0.17	0.18	0.19	0.79
_Realized_Sales_Gross_Profit_Growth_Rate	2058.00	0.02	0.02	0.00	0.02	0.02	0.02	1.00
_Operating_Profit_Growth_Rate	2058.00	0.85	0.00	0.74	0.85	0.85	0.85	1.00
_Continuous_Net_Profit_Growth_Rate	2058.00	0.22	0.01	0.00	0.22	0.22	0.22	0.23
_Total_Asset_Growth_Rate	2058.00	5287663257.05	2912614769.58	0.00	4315000000.00	6225000000.00	7220000000.00	9980000000.00
_Net_Value_Growth_Rate	2058.00	5189504.37	207791797.86	0.00	0.00	0.00	0.00	9330000000.00
_Total_Asset_Return_Growth_Rate_Ratio	2058.00	0.26	0.00	0.25	0.26	0.26	0.26	0.36
_Cash_Reinvestment_perc	2058.00	0.38	0.03	0.03	0.37	0.38	0.39	1.00
_Current_Ratio	2058.00	1336248.80	60619173.20	0.00	0.01	0.01	0.01	2750000000.00
_Quick_Ratio	2058.00	27755102.05	444865390.47	0.00	0.00	0.01	0.01	9230000000.00
_Interest_Expense_Ratio	2058.00	0.63	0.01	0.53	0.63	0.63	0.63	0.81
_Total_debt_to_Total_net_worth	2037.00	10714285.73	269696017.59	0.00	0.00	0.01	0.01	9940000000.00
_Long_term_fund_suitability_ratio_A	2058.00	0.01	0.03	0.00	0.01	0.01	0.01	1.00
_Net_profit_before_tax_to_Paid_in_capital	2058.00	0.18	0.03	0.00	0.17	0.17	0.18	0.79
_Total_Asset_Turnover	2058.00	0.13	0.10	0.00	0.06	0.10	0.17	0.92
_Accounts_Receivable_Turnover	2058.00	41598639.46	504767266.59	0.00	0.00	0.00	0.00	9740000000.00
_Average_Collection_Days	2058.00	26297862.01	410996733.83	0.00	0.00	0.01	0.01	8800000000.00
_Inventory_Turnover_Rate_times	2058.00	2030227259.48	3077250265.27	0.00	0.00	19100000.00	3815000000.00	9990000000.00

Table 4: Data Summary

The above Table 4 shows the summary of the descriptive statistics of the columns in the dataset, and it also depicts the dataset's mean, median, min, max and lower and upper quartile values.

```
0   0.89
1   0.11
Name: Default, dtype: float64
```

From the dataset, it is clear that approximately 11% of the companies in the dataset have defaulted. This indicates the presence of a **class imbalance,** with a majority of companies being non-defaulters.

Duplicate values:

There are no duplicate value present in the dataset.

1.2 Outlier Treatment:

Before performing outlier treatment, it is important to drop the target variable because outliers in the target variable can significantly affect the analysis and model training. Additionally, the index column and the company name have been removed as they are considered unimportant variables.

_Operating_Expense_Rate	0
_Research_and_development_expense_rate	264
_Cash_flow_rate	206
_Interest_bearing_debt_interest_rate	94
_Tax_rate_A	42
_Cash_Flow_Per_Share	146
_Per_Share_Net_profit_before_tax_Yuan_	186
_Realized_Sales_Gross_Profit_Growth_Rate	283
_Operating_Profit_Growth_Rate	317
_Continuous_Net_Profit_Growth_Rate	340
_Total_Asset_Growth_Rate	0
_Net_Value_Growth_Rate	304
_Total_Asset_Return_Growth_Rate_Ratio	226
_Cash_Reinvestment_perc	220
_Current_Ratio	193
_Quick_Ratio	190
_Interest_Expense_Ratio	328
_Total_debt_to_Total_net_worth	105
_Long_term_fund_suitability_ratio_A	234
_Net_profit_before_tax_to_Paid_in_capital	173
_Total_Asset_Turnover	101
_Accounts_Receivable_Turnover	281
_Average_Collection_Days	77
_Inventory_Turnover_Rate_times	29
_Fixed_Assets_Turnover_Frequency	501
_Net_Worth_Turnover_Rate_times	165
_Operating_profit_per_person	357
_Allocation_rate_per_person	200
_Quick_Assets_to_Total_Assets	4
_Cash_to_Total_Assets	163
_Quick_Assets_to_Current_Liability	185
_Cash_to_Current_Liability	253
_Operating_Funds_to_Liability	219
_Inventory_to_Working_Capital	247
_Inventory_to_Current_Liability	129
_Long_term_Liability_to_Current_Assets	213
_Retained_Earnings_to_Total_Assets	208
_Total_income_to_Total_expense	136
_Total_expense_to_Assets	168
_Current_Asset_Turnover_Rate	464
_Quick_Asset_Turnover_Rate	0
_Cash_Flow_to_Equity	306
_Current_Liability_to_Current_Assets	121
_Liability_Assets_Flag	7
_Total_assets_to_GNP_price	235
_No_credit_Interval	396
_Degree_of_Financial_Leverage_DFL	438
_Interest_Coverage_Ratio_Interest_expense_to_EBIT	376
_Net_Income_Flag	0
_Equity_to_Liability	190
dtype: int64	

Table 5: Total outlier for each column

The table above provides the count of outliers for each individual variable in the dataset. Overall, there are a total of 10,864 outliers identified across the dataset. These outliers can potentially have an impact on the analysis, and it require treatment as part of the data analysis process.

Checking for Outliers:

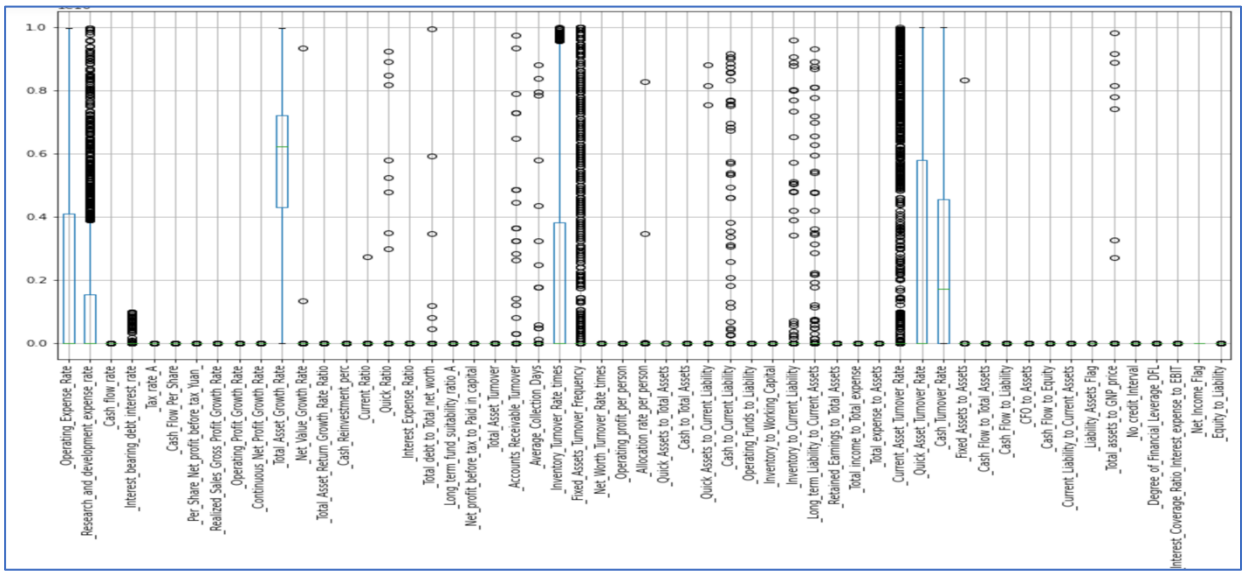


Figure 1: Boxplot with Outliers

Boxplot of removing outliers:

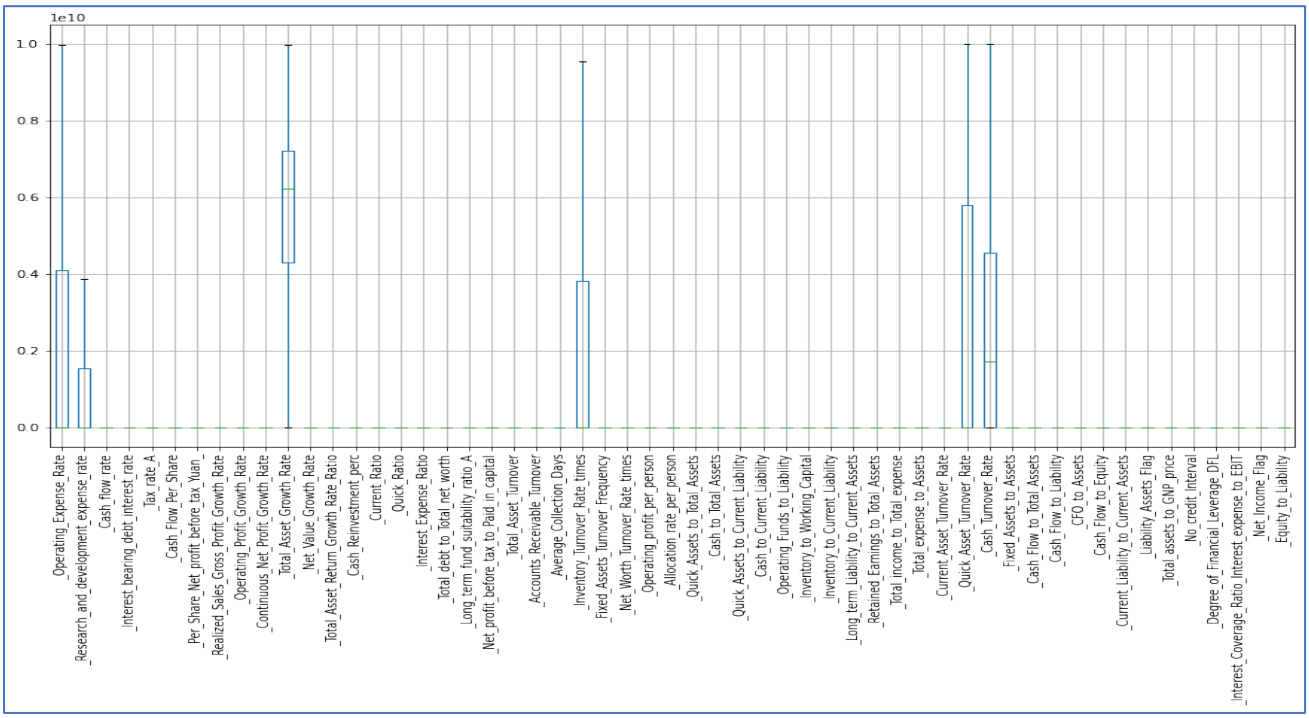


Figure 2: Boxplot after removing outliers

Outlier removal is a crucial step in data pre-processing to ensure accurate analysis and modelling. The **Interquartile Range (IQR) method** is used to identify and address outliers in the dataset. This method determines the acceptable range of values in the data by calculating lower and upper limits based on the spread of the data. Overall, by addressing outliers, we ensure that these extreme values do not negatively impact the accuracy and reliability of our analysis.

1.3 Missing Value Treatment

_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
dtype: int64	

Table 6: Missing Value in the Dataset

we observe a total of 298 missing values in the dataset. While this number may seem small in comparison to the overall dataset size, it is still important to address these missing values. To impute the missing values, we are utilizing KNN imputation method with a parameter value of n-Neighbor = 5.

KNN Imputation:

KNN (K-Nearest Neighbors) imputation is a technique used to fill in missing values in a dataset based on the values of its nearest neighbors. In this approach, each missing value is replaced with the average value of its k nearest neighbors. The value of k, in this case, is set to 5.

Therefore, by employing KNN imputation with n-Neighbor = 5, we aim to fill in the missing values in the dataset and ensure a more comprehensive and robust analysis. And importantly, KNN imputation helps to minimize bias in the imputed values.

_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 7: Dataset after imputing missing values

1.4 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)

For plotting univariate and bivariate analyses, we have selected the important variables based on their statistical significance. These variables have a p-value less than 0.005, indicating a strong relationship with the target variable. Additionally, we have considered the VIF (Variance Inflation Factor) value of each variable, ensuring that it is below 5 to avoid issues of multicollinearity.

A. Univariate:

Count of Default:

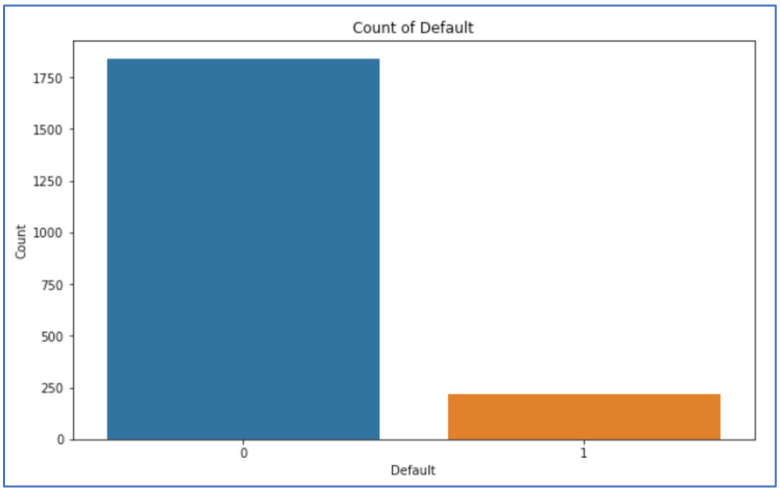


Figure 3: Count plot - Default

From the above count plot, it is obvious that the majority of companies, totalling a large number, fall into the non-default category, while only a smaller subset of 220 companies is classified as defaulters.

Histogram of Cash flow per share:

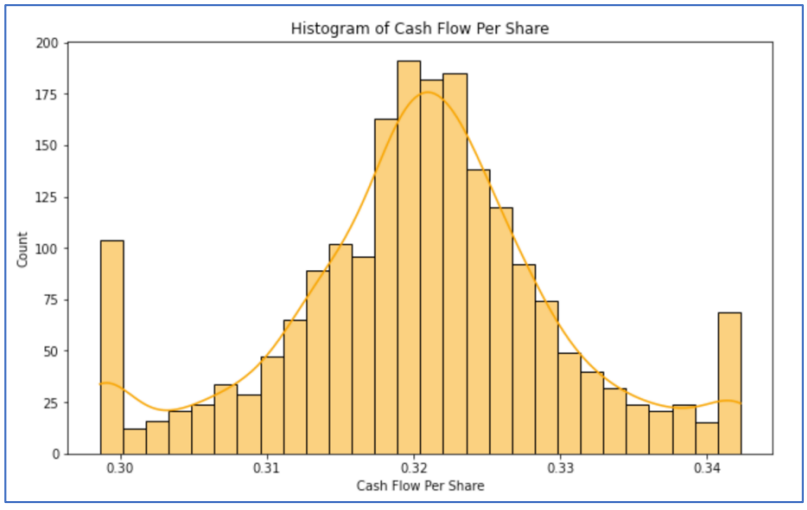


Figure 4: Histogram - Cash Flow Per share

The histogram reveals a prominent peak at a cash flow per share value of 0.32, indicating that a significant proportion of the data points in the dataset are clustered around this value.

Boxplot of Research and Development Expense Rate:

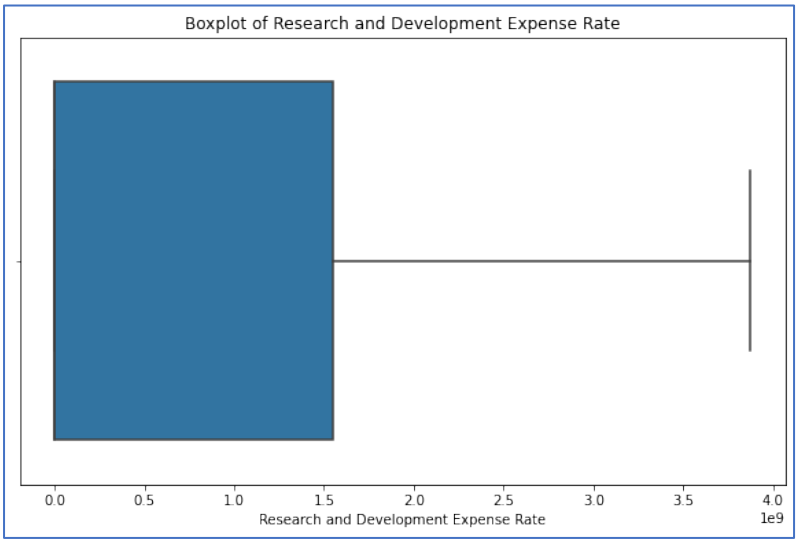


Figure 5: Boxplot of Research and Development Expense Rate

The boxplot illustrates that the middle 50% of the data is concentrated within the range of 0.0 to 1.5 for the research and development expense rate. Furthermore, the maximum value for this variable is approximately 4.0, indicating values beyond the upper range of the boxplot.

Box Plot of Average Collection Days:

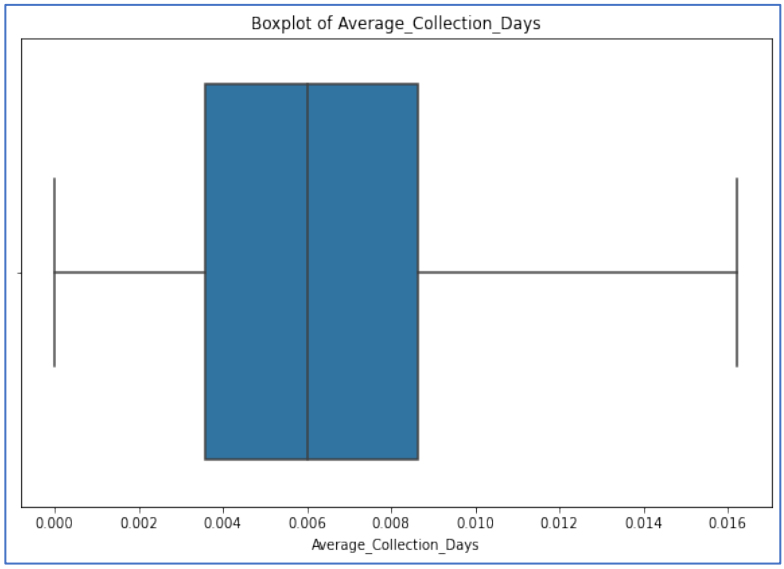


Figure 6: Boxplot of Average Collection Days

The boxplot for "Average Collection Days" reveals a distribution with a minimum value of 0.000 and a maximum value of 0.016**. The median** (50th percentile) falls at 0.006, indicating the central tendency of the data.

B. Bi-Variate analysis:

Boxplot of Quick Assets to Total Assets by Default:

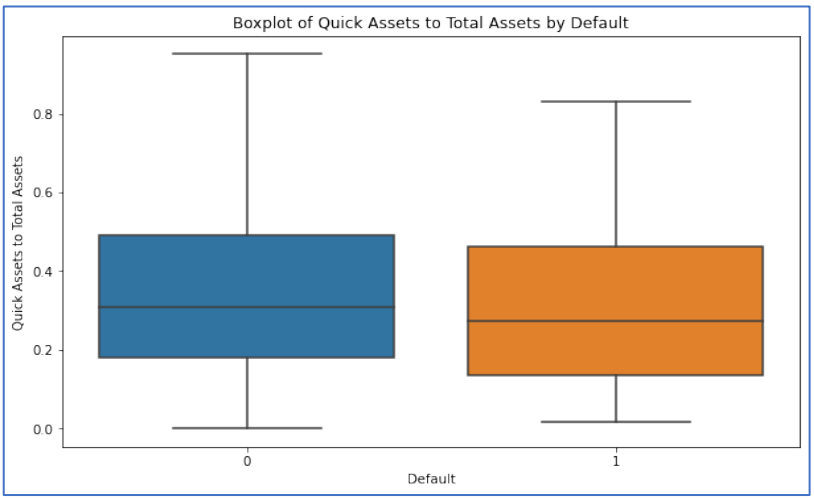


Figure 7: Boxplot of Quick Assets to Total Assets by Default

The boxplot analysis of "Quick Assets to Total Assets" by default status reveals distinguishable distributions between defaulted (1) and non-defaulted (0) companies. While the overall patterns may appear similar, it is noteworthy that defaulted companies tend to exhibit slightly lower values for "Quick Assets to Total Assets" compared to non-defaulted companies. This observation suggests that the ratio of quick assets to total assets could serve as a valuable indicator of default risk, with lower values potentially indicating a higher probability of default.

Boxplot of Total_Asset_Growth_Rate by Default:

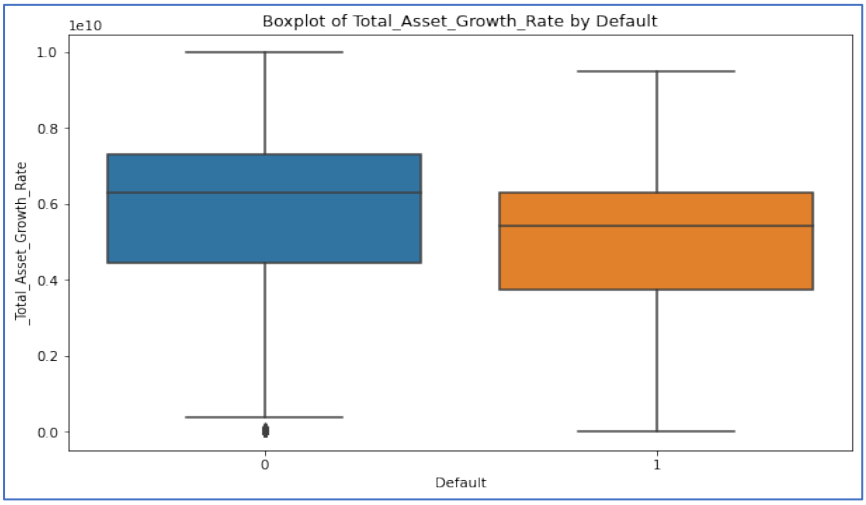


Figure 8:Boxplot of Total_Asset_Growth_Rate by Default

The boxplot analysis for "Total_Asset_Growth_Rate" by default status (0 – Not Defaulted, 1 - Defaulted) indicates similar distributions for defaulted and non-defaulted companies. Both groups have comparable median values, ranging from 0.5 to 0.6. This suggests that the growth rate of total assets may not be a sole distinguishing factor for default risk in this dataset.

Boxplot of Research_and_development_expense_rate by Default:

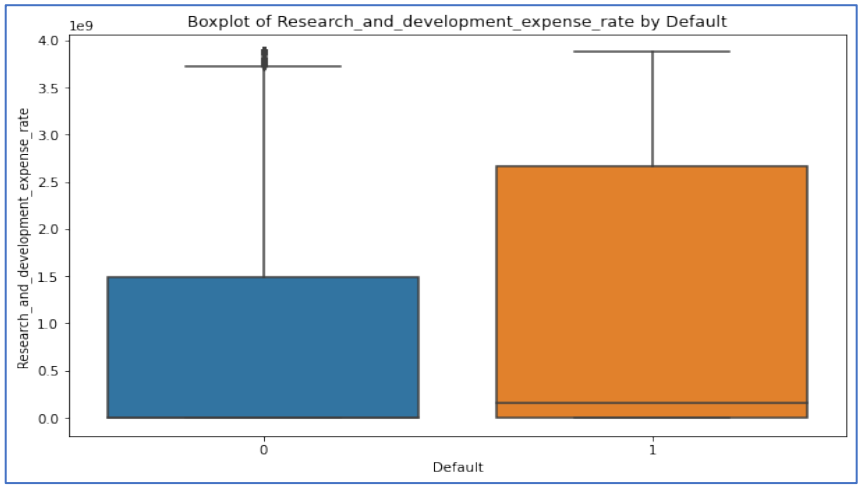


Figure 9: Boxplot of Research_and_development_expense_rate by Default

The boxplot analysis of "Total_Asset_Growth_Rate" by default status (0 - Not Defaulted, 1 - Defaulted) reveals interesting insights. Non-defaulted companies (0) show very limited growth, with most values concentrated around 0.0 and a few outliers reaching up to 3.75. In contrast, defaulted companies (1) exhibit a wider range of growth rates, with the median at 0.25 and some values extending up to 2.75. This suggests that defaulted companies tend to have more varied asset growth rates compared to non-defaulted ones.

Boxplot of Current_Liability_to_Current_Assets by Default:

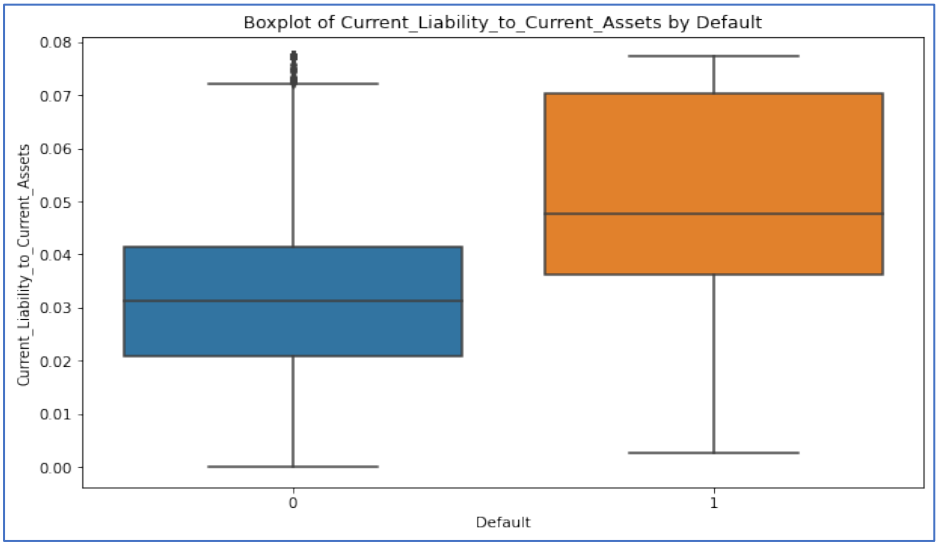


Figure 10: Boxplot of Current_Liability_to_Current_Assets by Default

The boxplot analysis for "Current_Liability_to_Current_Assets" based on default status (0 - Not Defaulted, 1 - Defaulted) reveals distinct distributions. Non-defaulted companies (0) show a relatively lower ratio of current liabilities to current assets, with a median value of 0.03 and a range from 0.0 to 0.04. Defaulted companies (1) display higher values, with a median of 0.05 and a wider range from 0.0 to 0.08. This suggests that a higher ratio of current liabilities to current assets may indicate a higher risk of default.

Boxplot of Operating_profit_per_person by Default:

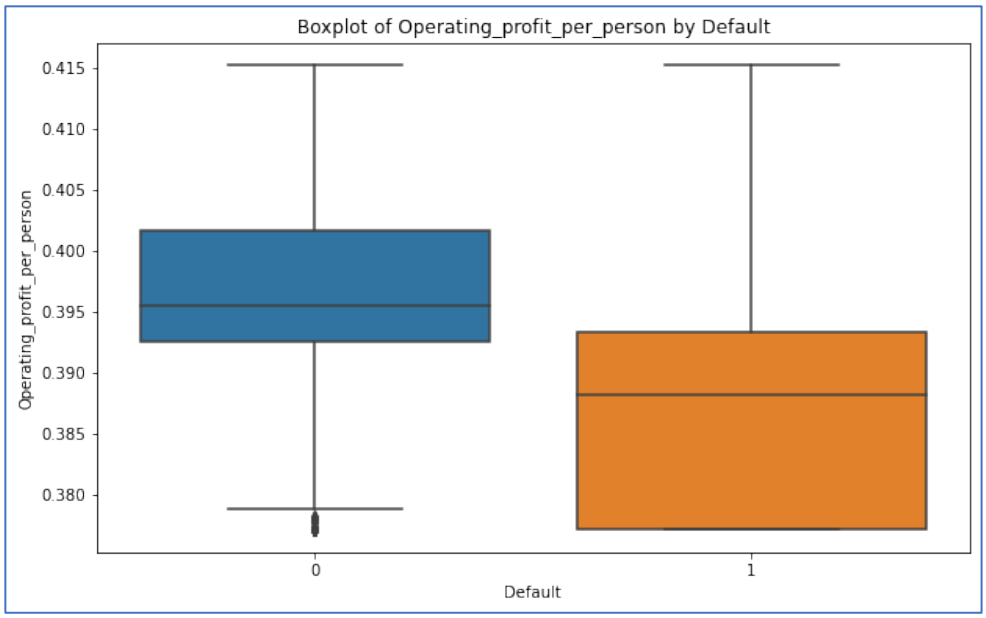


Figure 11: Boxplot of Operating_profit_per_person by Default

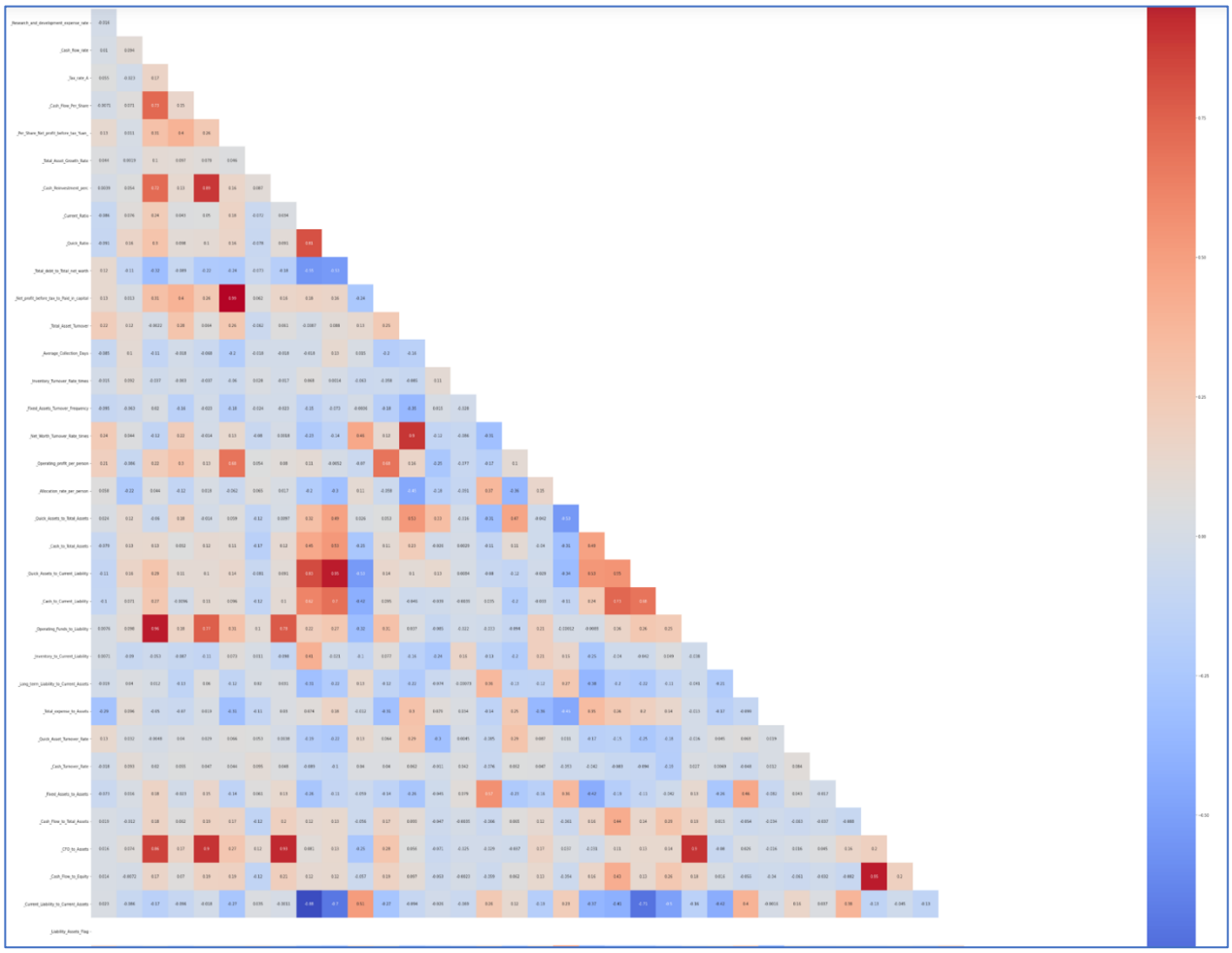
The boxplot analysis of "Operating_profit_per_person" by default status (0 - Not Defaulted, 1 - Defaulted) reveals interesting patterns. Non-defaulted companies (0) have a relatively narrow range of operating profit per person, with values concentrated between 0.393 and 0.400. On the other hand, defaulted companies (1) exhibit a slightly wider range, with the median at 0.390 and values extending up to 0.415. This suggests that there may be some variation in operating profit per person among defaulted companies compared to non-defaulted ones, although the overall difference is relatively small.

Heatmap of the dataset:

In our analysis, we identified and removed several variables, such as _Realized_Sales_Gross_Profit_Growth_Rate,_Operating_Profit_Growth_Rate,_Continuous_ Net_Profit_Growth_Rate, and others. These variables were excluded due to having constant values throughout the dataset.

For the remaining variables, we created a heatmap to explore their correlations. The heatmap displayed strong correlations between certain variables, indicating a potential relationship.

This process helps us streamline the dataset by eliminating irrelevant variables and highlights the interdependencies among the remaining variables, enabling us to focus on the most meaningful analysis.



VIF (Variance Inflation Factor:

VIF (Variance Inflation Factor) is a measure that tells us if there is a problem of multicollinearity in a regression model. Multicollinearity occurs when predictor variables are highly correlated with each other.

Firstly, to assess multicollinearity in the dataset, we calculated the VIF (Variance Inflation Factor) using the statsmodels library. VIF measures the extent to which predictor variables are correlated with each other. Higher VIF values indicate a stronger correlation, potentially leading to instability in the regression model's coefficients. The calculated VIF values for each variable can help identify any multicollinearity issues present in the dataset.

	variables	VIF
11	_Net_profit_before_tax_to_Paid_in_capital	95.99
5	_Per_Share_Net_profit_before_tax_Yuan_	95.91
31	_CFO_to_Assets	28.40
23	_Operating_Funds_to_Liability	20.92
21	_Quick_Assets_to_Current_Liability	19.15
2	_Cash_flow_rate	15.86
16	_Net_Worth_Turnover_Rate_times	14.18
8	_Current_Ratio	13.87
12	_Total_Asset_Turnover	12.81
9	_Quick_Ratio	12.31
30	_Cash_Flow_to_Total_Assets	12.23
32	_Cash_Flow_to_Equity	12.20
7	_Cash_Reinvestment_perc	12.19
33	_Current_Liability_to_Current_Assets	6.65
4	_Cash_Flow_Per_Share	6.43
37	_Equity_to_Liability	5.98
19	_Quick_Assets_to_Total_Assets	5.64
10	_Total_debt_to_Total_net_worth	4.28
22	_Cash_to_Current_Liability	4.24
20	_Cash_to_Total_Assets	3.66
24	_Inventory_to_Current_Liability	3.20
29	_Fixed_Assets_to_Assets	2.72
18	_Allocation_rate_per_person	2.50
17	_Operating_profit_per_person	2.36
26	_Total_expense_to_Assets	2.01
15	_Fixed_Assets_Turnover_Frequency	1.75
13	_Average_Collection_Days	1.74
35	_Total_assets_to_GNP_price	1.70
25	_Long_term_Liability_to_Current_Assets	1.67
27	_Quick_Asset_Turnover_Rate	1.36
3	_Tax_rate_A	1.35
0	_Operating_Expense_Rate	1.30
1	_Research_and_development_expense_rate	1.17
14	_Inventory_Turnover_Rate_times	1.15
6	_Total_Asset_Growth_Rate	1.12
28	_Cash_Turnover_Rate	1.10
34	_Liability_Assets_Flag	NaN
36	_Net_Income_Flag	NaN

Table 9: Variable with the VIF Value

	variables	VIF
14	_Quick_Assets_to_Total_Assets	4.10
26	_Equity_to_Liability	3.79
24	_Current_Liability_to_Current_Assets	3.63
16	_Cash_to_Current_Liability	3.60
15	_Cash_to_Total_Assets	3.55
8	_Total_Asset_Turnover	2.99
7	_Total_debt_to_Total_net_worth	2.91
2	_Cash_flow_rate	2.88
5	_Per_Share_Net_profit_before_tax_Yuan_	2.63
22	_Fixed_Assets_to_Assets	2.51
4	_Cash_Flow_Per_Share	2.43
13	_Allocation_rate_per_person	2.41
12	_Operating_profit_per_person	2.34
17	_Inventory_to_Current_Liability	2.22
19	_Total_expense_to_Assets	1.98
11	_Fixed_Assets_Turnover_Frequency	1.73
9	_Average_Collection_Days	1.69
25	_Total_assets_to_GNP_price	1.66
18	_Long_term_Liability_to_Current_Assets	1.58
23	_Cash_Flow_to_Equity	1.36
20	_Quick_Asset_Turnover_Rate	1.35
3	_Tax_rate_A	1.34
0	_Operating_Expense_Rate	1.30
1	_Research_and_development_expense_rate	1.15
10	_Inventory_Turnover_Rate_times	1.14
6	_Total_Asset_Growth_Rate	1.09
21	_Cash_Turnover_Rate	1.09

Table 8: Variable with VIF value less than 5

We calculated the VIF (Variance Inflation Factor) values to identify variables that have a strong correlation with each other. Higher VIF values indicate a higher degree of correlation. To ensure the stability of the regression model, we removed variables with VIF values above 5, as they could introduce multicollinearity issues. This selection process resulted in a set of variables with minimal correlation, enhancing the reliability of our analysis.

1.5 Train Test Split:

we will be applying data scaling before the train-test split. This step is particularly important because we have observed that some variables in the dataset exhibit large data values. By scaling the data, we can bring all features to a similar scale, preventing any single variable from dominating the learning process and ensuring compatibility across different algorithms. This approach will help us maintain the integrity of our analysis and improve the accuracy and efficiency of our models.

The data was scaled using the **StandardScaler** method. This was done to ensure that all variables are brought to a similar scale, removing any potential bias caused by variables with larger data values.

To evaluate the performance of machine learning models, the dataset was divided into training and testing sets using the **train_test_split function.** This split 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 for testing. The random_state parameter was set to **42.** This process allows 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,)
```

Head of the Train-Data:

	_Operating_Expense_Rate	_Research_and_development_expense_rate	_Cash_flow_rate	_Tax_rate_A	_Cash_Flow_Per_Share	_Per_Share_Net_profit_before_
2011	-0.63	-0.65	2.09	-0.82	1.67	
697	-0.63	0.57	0.26	1.52	0.86	
160	-0.63	-0.65	-0.72	-0.82	-0.58	
1273	1.20	1.34	-0.25	-0.82	0.05	
541	-0.63	-0.19	-0.52	-0.82	-0.43	

Table 10: Head of the Train- Data

Head of Test-Data:

	_Operating_Expense_Rate	_Research_and_development_expense_rate	_Cash_flow_rate	_Tax_rate_A	_Cash_Flow_Per_Share	_Per_Share_Net_profit_before_
974	-0.63	0.18	1.30	1.13	1.51	
134	-0.63	-0.65	-0.96	-0.72	-1.90	
1267	-0.63	-0.65	-0.35	-0.82	-0.29	
464	-0.63	-0.65	2.09	1.34	1.18	
579	0.58	-0.65	-0.38	1.37	-0.27	

Table 11: Head of the Test-Data

1.6 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

We imported the "statsmodels.formula.api" module, which provides a user-friendly interface for specifying and estimating statistical models. This module will be utilized in our analysis to build and evaluate regression models.

Logistic Regression Model:

Logistic regression is a statistical technique used to predict binary outcomes by estimating the probability of an event happening. It helps analyze factors that influence the occurrence of the outcome. By calculating odds based on independent variables, the model makes predictions and provides insights into the relationship between predictors and the outcome. It is useful when the outcome is categorical, offering 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 12: Train Columns for Model Building

The table above 12 displays the remaining columns in the training dataset after successfully removing variables with VIF values exceeding 5. This step was taken to mitigate the issue of multicollinearity and ensure that the selected variables are suitable for further model training and analysis.

The model was trained on the train dataset using the logistic regression algorithm from the statsmodels library.

Model 1: Summary – with 27 Variable

Logit Regression Results

Dep. Variable:	Default	**No. Observations:**	1378
Model:	Logit	**Df Residuals:**	1350
Method:	MLE	**Df Model:**	27
Date:	Sat, 15 Jul 2023	**Pseudo R-squ.:**	0.4164
Time:	00:43:45	**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
_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 13: Logit Regression Model 1

In the logistic regression analysis, we will iteratively remove variables with p-values greater than 0.005 to refine the model. The p-value represents the probability of a random relationship between an independent variable and the dependent variable. By eliminating variables with higher p-values, we focus on the variables that have a significant impact on the likelihood of default. This step ensures that our model includes only the most influential variables, improving its accuracy and interpretability.

Final Model: Model 23 Summary: (5 Variable)

Logit Regression Results

Dep. Variable:	Default	**No. Observations:**	1378
Model:	Logit	**Df Residuals:**	1372
Method:	MLE	**Df Model:**	5
Date:	Fri, 14 Jul 2023	**Pseudo R-squ.:**	0.3825
Time:	00:57:44	**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 14: Final Model: Model 23 Summary

After a thorough analysis, we successfully improved the logistic regression model by reducing the number of variables from 24 to 5. We achieved this by carefully removing variables with p-values higher than 0.05, keeping only the most influential ones.

The model's pseudo-R-squared value of 0.3825 indicates that the selected variables explain around 38.25% of the variation in default likelihood.

This refined model provides a focused and dependable set of variables that significantly contribute to predicting defaults. It's important to note that we conducted ***23 iterations*** to reach this final summary, ensuring the accuracy and reliability of our findings.

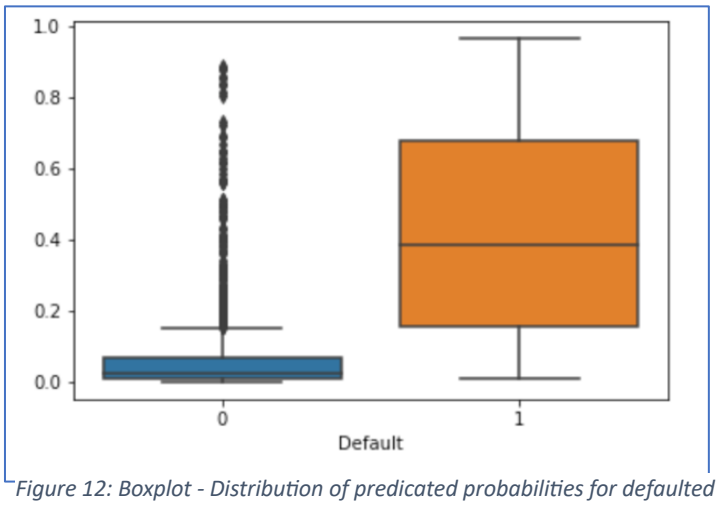


Figure 12: Boxplot - Distribution of predicated probabilities for defaulted & non-defaulted

We used the logistic regression model to predict the probability of default for the training dataset. The predicted probabilities were then used to create a boxplot, showing 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.

The optimal threshold for classification was determined to be **0.1076.** Any predicted probability above this threshold is considered as a predicted default, while values below the threshold are classified as non-default.

We further examined the predicted probabilities for the training dataset.

```
2011   0.11
697    0.01
160    0.06
1273   0.04
541    0.10
       ...
1386   0.01
1127   0.01
950    0.02
1058   0.03
562    0.25
Length: 1378, dtype: float64
```

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

After training the logistic regression model on the training dataset, we validated its performance on the test dataset.

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

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

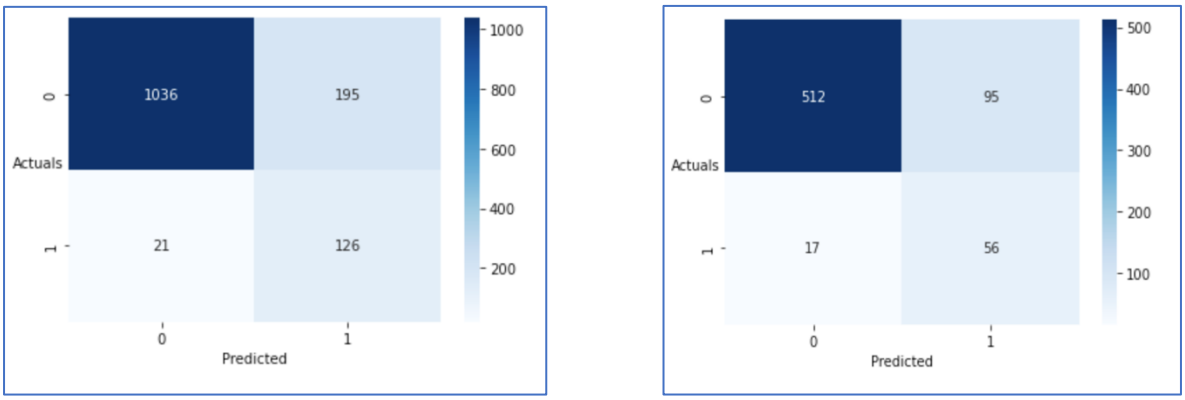


Figure 13: Logistic Regression Model- Confusion Matrix for Training & Testing Data

Classification report for Training and Testing Data - with optimal threshold- 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 16: 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.

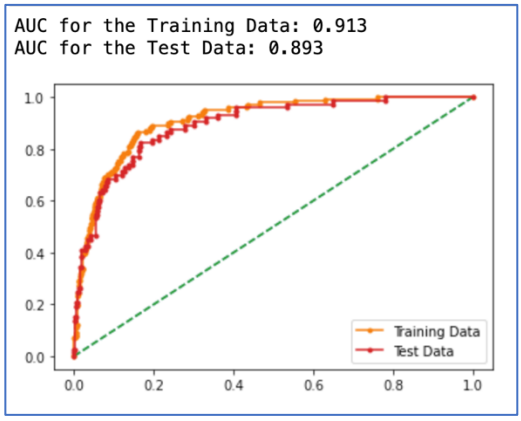


Figure 14: Logistic Regression: OC_AUC Curve: For Training data & Test Data: with optimal threshold- 0.1076.

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%, indicating that 98% of the instances predicted as non-default were truly non-default. The precision for class 1 (default) was 37.1%, meaning 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 an algorithm used to address class imbalance in datasets. It creates synthetic samples of the minority class by interpolating existing instances. This helps improve the performance of machine learning models in predicting the minority class as we can see 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 represents approximately 57% of the samples, while the minority class (1) represents approximately 43% of the samples. This equalization of class proportions allows for a more accurate and reliable training of the model.

```
0   0.57
1   0.43
Name: Default, dtype: float64
```

Confusion Matrix for Training and Testing data: with SMOTE

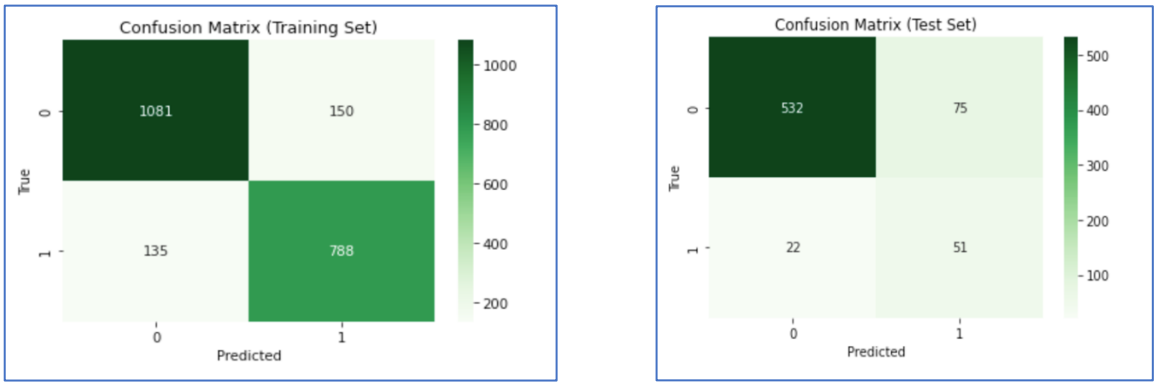


Figure 15: Logistic Regression Model - confusion matrix for training and testing with SMOTE

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

Table 17: Logistic Regression - Classification report for Training and Testing Data- with SMOTE

ROC_AUC Curve: For Training data & Test Data – with SMOTE:

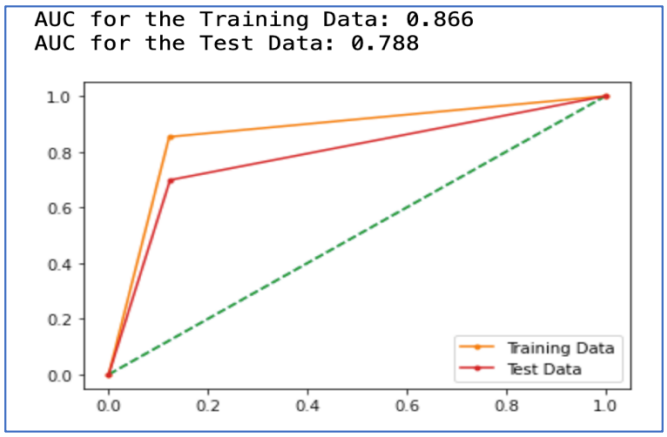


Figure 16: Logistic Regression - ROC_AUC Curve: For Training data & Test Data – with SMOTE:

Inference: Logistic Regression Model- with SMOTE

The logistic regression model with SMOTE oversampling technique was used to predict defaulters. The model achieved an accuracy of 87% on the training set, with good 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, indicating the model's ability to correctly identify defaulters. The AUC score for the test data was 0.788. Overall, the model demonstrates good predictive performance in identifying defaulters, although further improvements could be made to enhance its precision for class 1 instances.

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

Random Forest Classifier Model:

Random Forest Classifier is a popular machine learning algorithm that combines multiple decision trees to make predictions. By averaging the predictions of multiple trees, Random Forest Classifier provides accurate and reliable results.

Model Building Approach:

```
GridSearchCV(estimator=RandomForestClassifier(),
             param_grid={'max_depth': [3, 5, 7],
                         'min_samples_leaf': [5, 10, 15],
                         'min_samples_split': [15, 30, 45],
                         'n_estimators': [25, 50]})
```

Table 18: GridSearchCV

We performed a Grid Search Cross-Validation to find the optimal hyperparameters for the Random Forest Classifier. The search was conducted using various combinations of hyperparameters, including '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 used to build the Random Forest Classifier model, ensuring optimal performance and accuracy in predicting the outcome.

```
{'max_depth': 7,
 'min_samples_leaf': 10,
 'min_samples_split': 30,
 'n_estimators': 50}
```

Table 19: Best Parameter

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

Confusion Matrix: Train Dataset

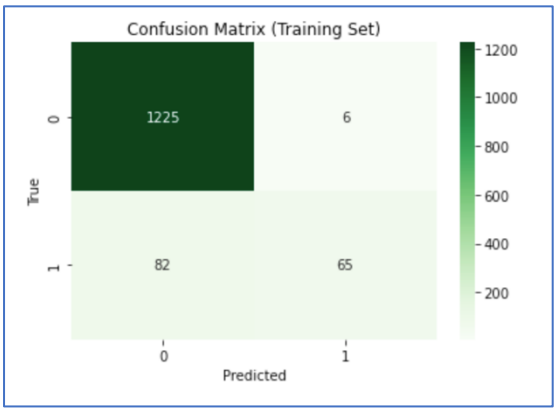


Figure 17:Random Forest Model - Confusion Matrix

Classification Report: Train Dataset

	precision	recall	f1-score	support
0	0.94	1.00	0.97	1231
1	0.92	0.44	0.60	147
accuracy			0.94	1378
macro avg	0.93	0.72	0.78	1378
weighted avg	0.93	0.94	0.93	1378

Table 20: Random Forrest - Classification Report: Train Dataset

ROC_AUC Curve: For Training data

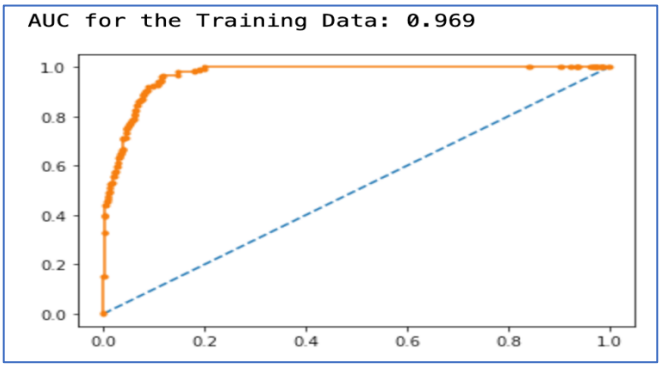


Figure 18: Random Forest- ROC_AUC Curve: For Training data

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

The classification report shows that the model has a high precision of 0.94 for class 0 and 0.92 for class 1, indicating a good ability to correctly identify non-default and default cases, respectively. The recall score is 1.00 for class 0, indicating that the model accurately captures all non-default cases, while the recall score is 0.44 for class 1, suggesting that the model struggles to capture all default cases. The model's overall accuracy is 0.94, and the AUC score for the training data is 0.969.

B. Random Forest Model on Train Dataset: With SMOTE

Confusion Matrix: Train Dataset – with SMOTE

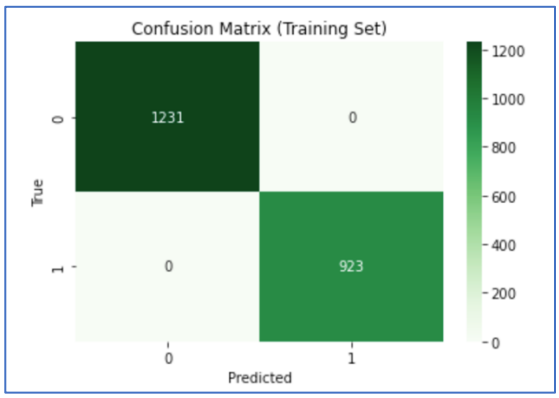


Figure 19: Random Forest - Confusion Matrix: Train Dataset – with SMOTE

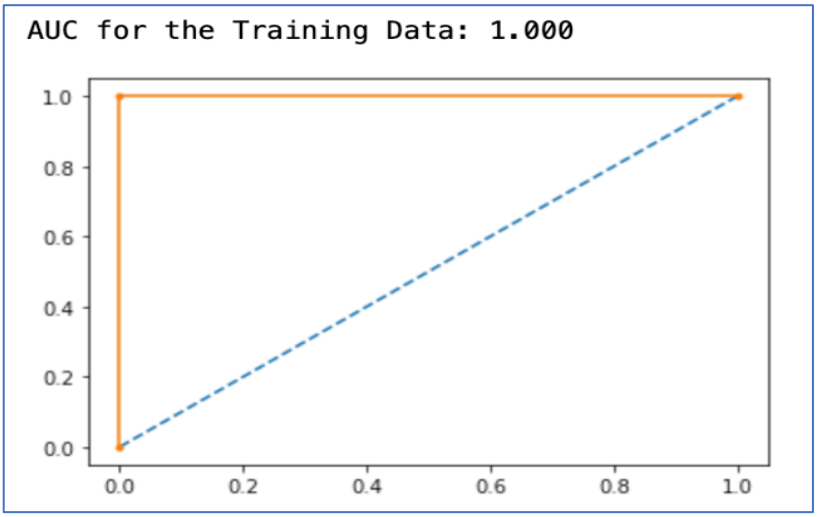
Classification Report: Train Dataset – With SMOTE

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

Table 21:Random Forest - Classification Report: Train Dataset – With SMOTE

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 shows 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 predicts all instances correctly 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, indicating its strong capability to make accurate predictions.

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

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

Confusion Matrix: Test Dataset

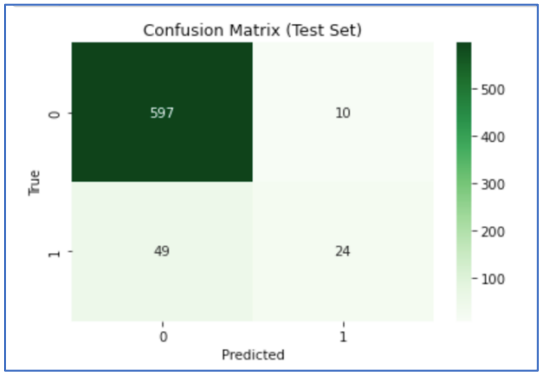


Figure 20: Random Forest - Confusion Matrix: Test Dataset

Classification Report: Test Dataset

	precision	recall	f1-score	support
0	0.92	0.98	0.95	607
1	0.71	0.33	0.45	73
accuracy			0.91	680
macro avg	0.82	0.66	0.70	680
weighted avg	0.90	0.91	0.90	680

Table 22:Random Forest - Classification Report: Test Dataset

ROC_AUC Curve: For Test data

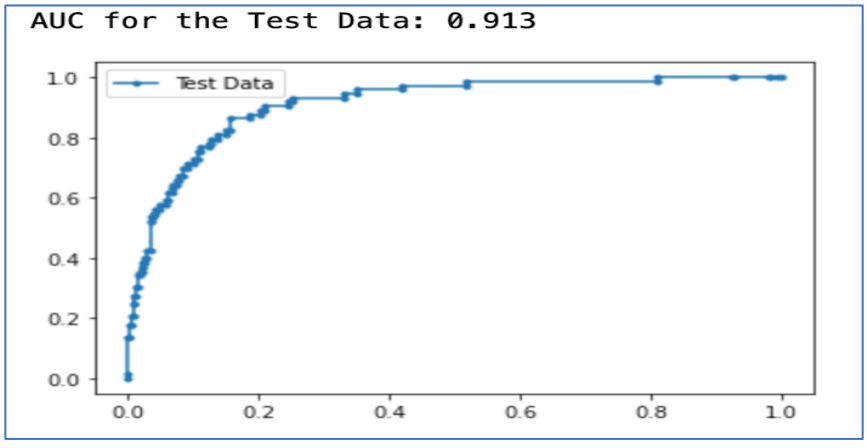


Table 23: Random Forest - ROC_AUC Curve: For Test data

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

The model performs well in accurately identifying non-default cases with a high precision of 0.92 and recall of 0.98. However, it struggles to accurately identify default cases, as reflected in a lower precision of 0.71 and recall of 0.33. Overall, the model achieves an accuracy of 0.91 on the test dataset, indicating its ability to make correct predictions. The AUC score of 0.913 suggests that the model has good discrimination power in distinguishing between default and non-default cases.

In summary, the model performs well in predicting non-default cases but shows limitations in capturing default cases.

B. Random Forest Model on Test Dataset: with SMOTE

Confusion Matrix: Test Dataset – With SMOTE

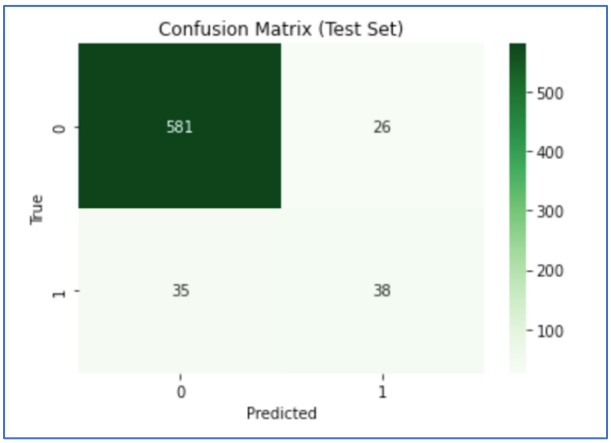


Figure 21: Random Forest - Confusion Matrix: Test Dataset – With SMOTE

Classification Report: Test Dataset - With SMOTE

Table 24: Random Forest - 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

ROC_AUC Curve: For Test data - With SMOTE

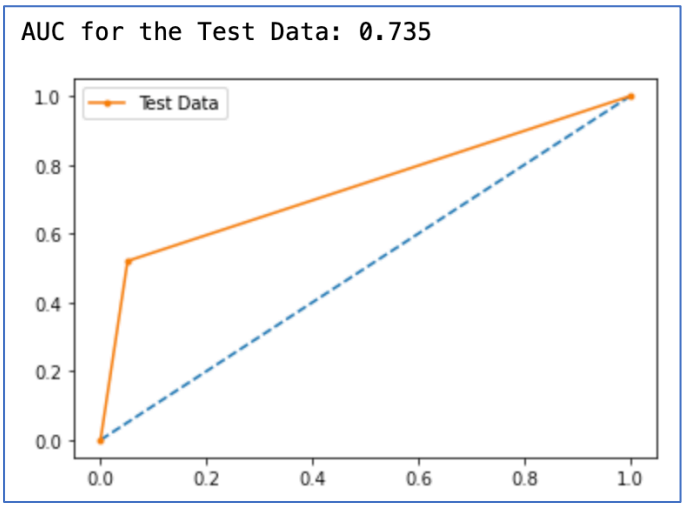


Figure 22: Random Forest - ROC_AUC Curve: For Test data - With SMOTE

Inference on Random Forest Model -Test Data: with SMOTE

The evaluation of the Random Forest model with SMOTE on the test data shows 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 performs exceptionally well 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 shows good performance in predicting non-default cases accurately but has some difficulty in

identifying default cases. The overall accuracy on the test data is 0.90, indicating reasonably accurate predictions. It is worth noting that the model may be slightly 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

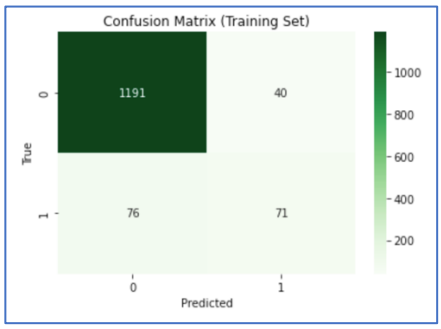


Figure 23: LDA MODEL - Confusion Matrix: Test Dataset – With threshold -0.5

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 25: LDA Model - Classification Report: Test Dataset - With threshold -0.5

ROC_AUC Curve: For Train data - With threshold -0.5

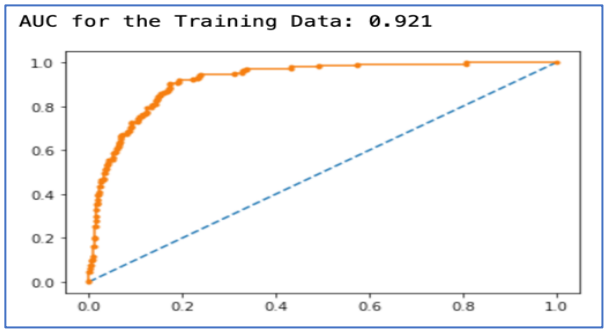


Figure 24: LDA Model - ROC_AUC Curve: For Test data - With threshold -0.5

Inference on LDA - Train Data: With threshold -0.5

The LDA model performs reasonably well on the train dataset. It accurately predicts non-default cases with a precision of 94% and recalls them with 97% However, it struggles to accurately identify default cases, achieving a precision of 64% and a recall of 48%. The overall accuracy of the model is 92%, indicating its ability to make correct predictions. The AUC score of 0.921 suggests that the model has 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

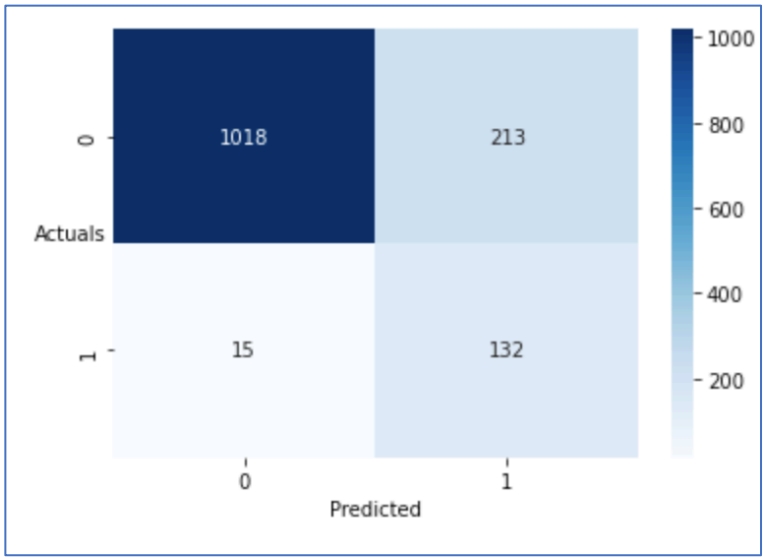


Figure 25: LDA Model - Confusion Matrix: Train Dataset – With threshold -0.0665

Classification Report: Train Dataset - With threshold -0.0665

	precision	recall	f1-score	support
0	0.925	0.790	0.852	620
1	0.133	0.333	0.190	60
accuracy			0.750	680
macro avg	0.529	0.562	0.521	680
weighted avg	0.855	0.750	0.794	680

Table 26: LDA Model - Classification Report: Train Dataset - With threshold -0.0665

ROC_AUC Curve: For Train data - With threshold -0.0665

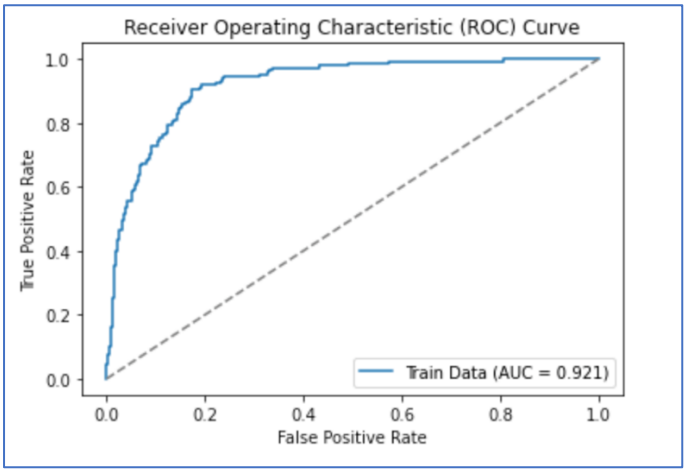


Figure 26: LDA Model - ROC_AUC Curve: For Train data - With threshold -0.0665

Inference on LDA- Train Data: With threshold -0.0665

The LDA model shows moderate performance on the train dataset. It achieves a precision of 92.5% and recall of 79% for non-default cases, indicating its ability to identify them reasonably well. However, the model struggles to accurately predict default cases, with a low precision of 13.3% and 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 results, the LDA model demonstrates limitations in accurately identifying default cases and may require further refinement to improve its performance in this aspect.

C. LDA Model on Train Dataset: with SMOTE

Confusion Matrix: Train Dataset – With SMOTE:

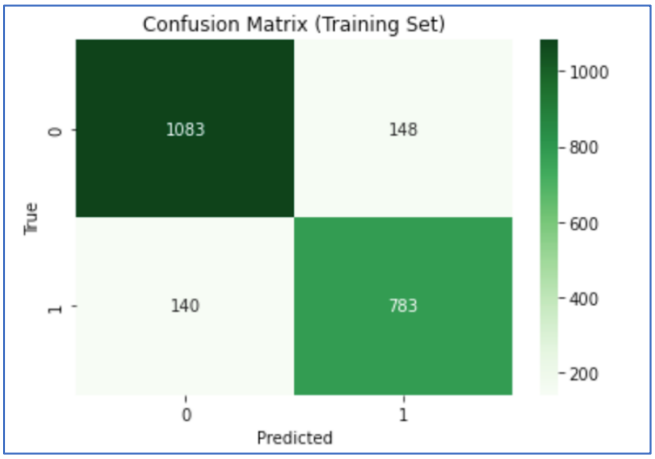


Figure 27: LDA Model - Confusion Matrix: Train Dataset – With SMOTE:

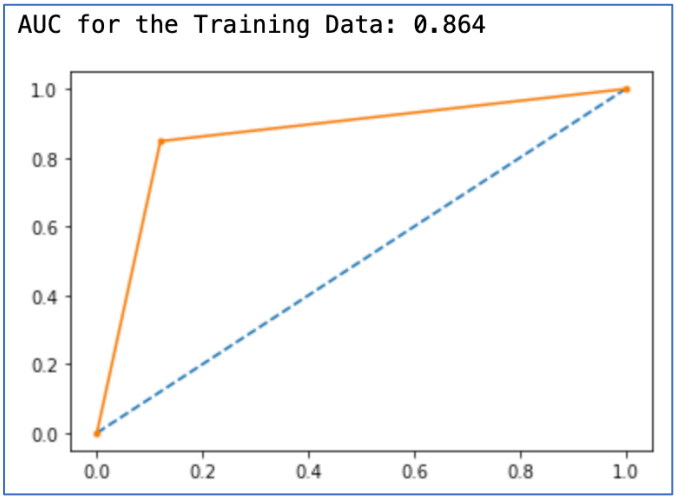
Classification Report: Train Dataset - With SMOTE

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 27: LDA Model - Classification Report: Train Dataset - With SMOTE

ROC_AUC Curve: For Train data - With SMOTE



Inference on LDA Train Data: With SMOTE

The LDA model with SMOTE exhibits good 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. The overall accuracy of the model is 87%, suggesting that it makes correct predictions in a majority of cases. The AUC score of 0.864 indicates a moderate ability of the model to distinguish between default and non-default cases. These results demonstrate 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

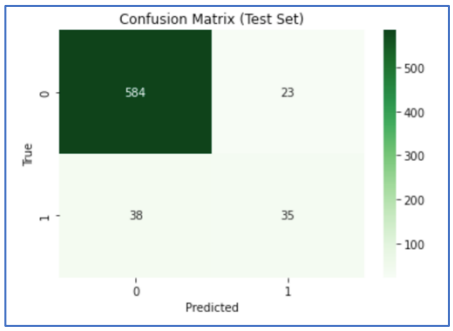


Figure 28: LDA Model - Confusion Matrix: Test Dataset – With threshold -0.5

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 28: LDA Model - Classification Report: Test Dataset - With threshold -0.5

ROC_AUC Curve: For Test data - With threshold -0.5

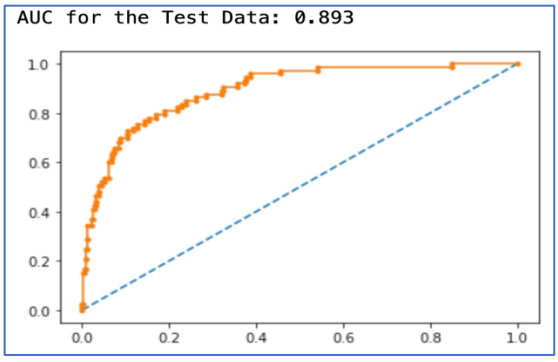


Figure 29: LDA Model - ROC_AUC Curve: For Test data - With threshold -0.5

Inference on LDA -Test Data: With threshold -0.5

The LDA model is effective in predicting non-default cases with a high precision of 94% and recall of 96%. However, it is less accurate in identifying default cases, with a precision of 60% and recall of 48%. The overall accuracy of the model is 91%, indicating its ability

to make correct predictions. The AUC score of 0.893 suggests that the model has 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

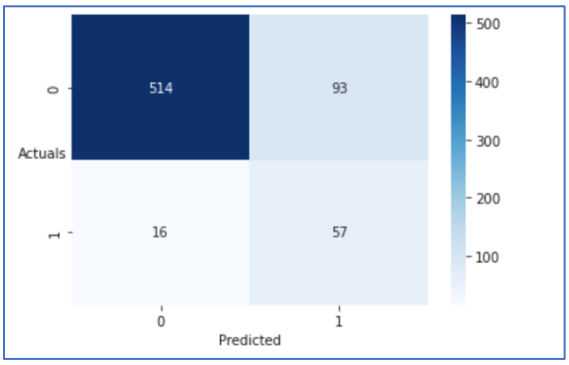


Figure 30: LDA Model - Confusion Matrix: Test Dataset – With threshold -0.0665

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 29: LDA Model - Classification Report: Test Dataset - With threshold -0.0665

ROC_AUC Curve: For Test data - With threshold -0.0665

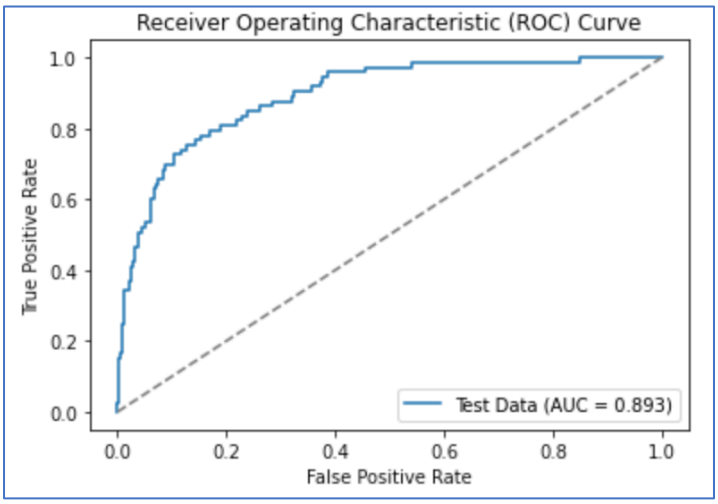


Figure 31: LDA Model - ROC_AUC Curve: For Test data - With threshold -0.0665

Inference on LDA - Test Data: With threshold -0.0665

The LDA model performs reasonably well on the test dataset. It accurately predicts non-default 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

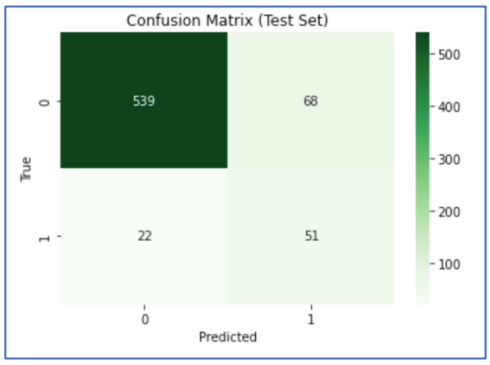


Figure 32: LDA Model - Confusion Matrix: Test Dataset – With SMOTE

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 30: LDA Model - Classification Report: Test Dataset - With SMOTE

ROC_AUC Curve: For Test data - With SMOTE

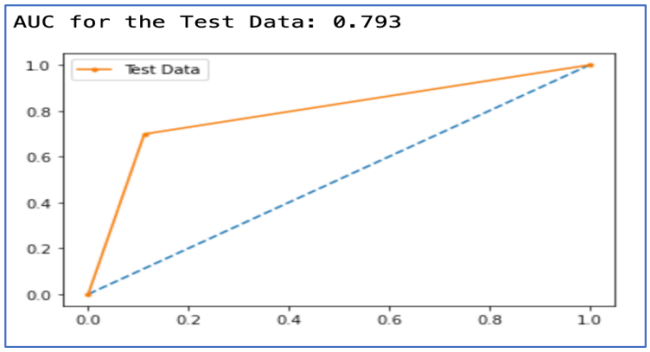


Figure 33: LDA Model - ROC_AUC Curve: For Test data - With SMOTE

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 non-default 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 Table. 31 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 31: First five rows of the dataset

The below Table. 32 shows the last five rows of the dataset:

	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

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')
```

Table 32: Fixing messy column names

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 33: 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

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

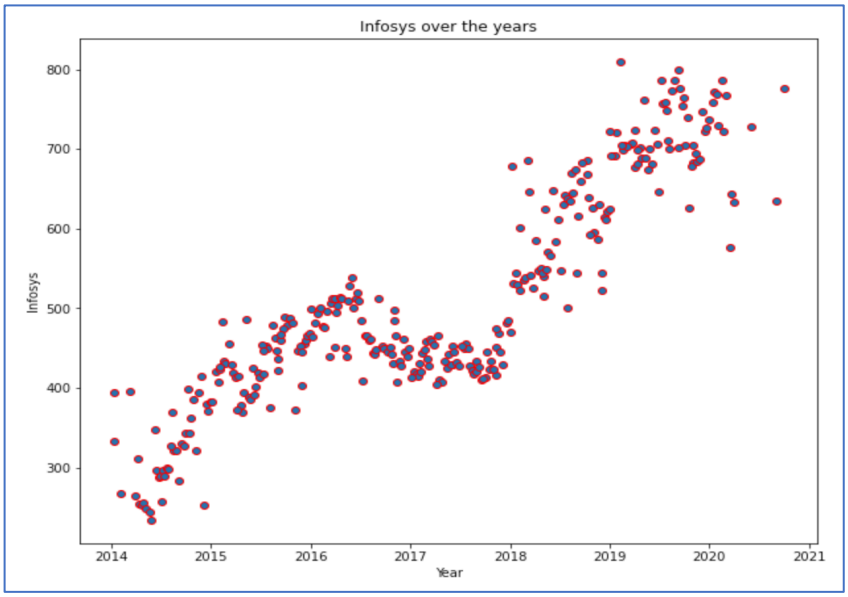


Figure 34: Infosys stock prices over the years

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.

b. Indian Hotel stock prices over the years

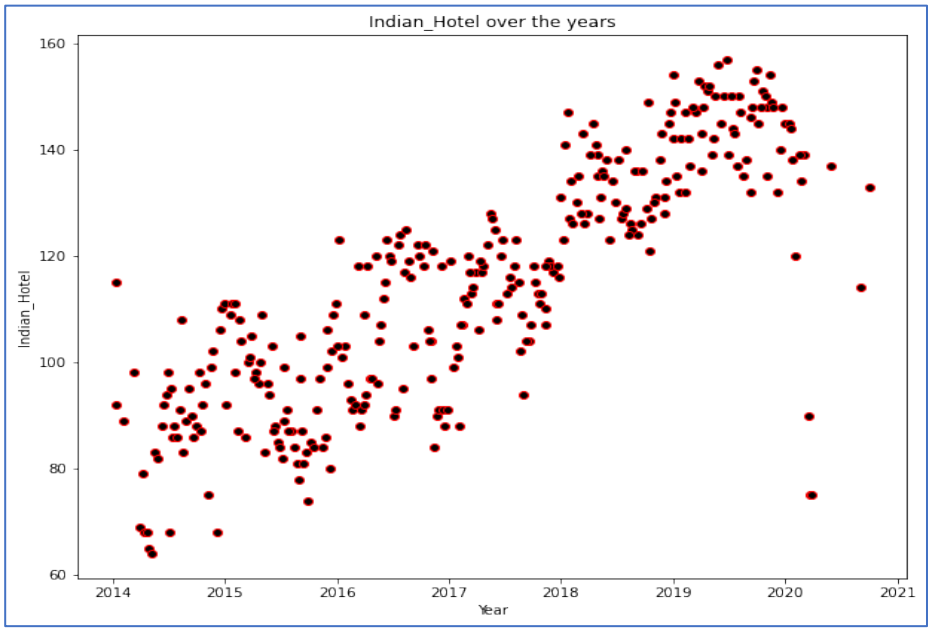


Figure 35: Indian Hotel stock prices over the years

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

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 34: Head of Stock Returns

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 36: Calculating the stock Means for all stocks

```
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 35: Calculating Standard Deviation for all stocks

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	0.00368	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 37: Stock Mean & Volatility of all the stocks

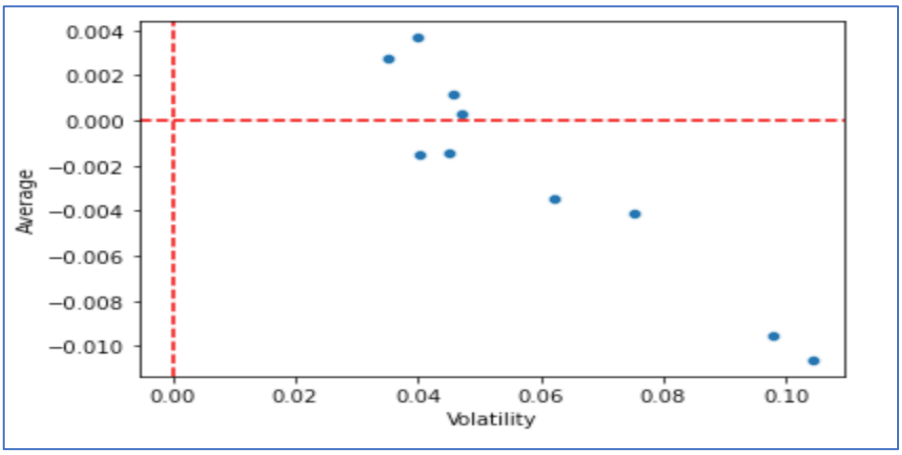


Figure 36: Scatter Plot of Average vs Volatility for all the stocks

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.