FRA GRADED PROJECT

A PROJECT REPORT

Submitted by

BALAJI S (PGP-DSBA-JUNE 2023 TO JUNE2024)

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 data

	Co_Code	Co_Name	_Operating_Expense_Rate	_Research_and_development_expense_rate	_Cash_flow_rate	_Interest_bearing_debt_interest_rate	_Tax_rate_A
0	16974	Hind.Cables	8.820000e+09	0.000000e+00	0.462045	0.000352	0.00141
1	21214	Tata Tele. Mah.	9.380000e+09	4.230000e+09	0.460116	0.000716	0.00000
2	14852	ABG Shipyard	3.800000e+09	8.150000e+08	0.449893	0.000496	0.00000
3	2439	GTL	6.440000e+09	0.000000e+00	0.462731	0.000592	0.00931
4	23505	Bharati Defence	3.680000e+09	0.000000e+00	0.463117	0.000782	0.40024

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_r
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

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
```

- The dataset comprises 2,058 rows and encompasses 58 features, consisting of various financial and performance indicators pertaining to different companies.

- Among these features, 57 are numerical, while 1 is categorical. Additionally, it's apparent that the dataset contains missing values.

- Of notable importance is the presence of a column labeled "Default," serving as the target variable.

- This column comprises binary values (0 and 1), indicating whether a company has defaulted or not. This variable holds significance for constructing predictive models or conducting further analyses related to company defaults

Data Summary:

	Co_Code	Co_Name	_Operating_Expense_Rate	_Research_and_development_expense_rate	_Cash_flow_rate	_Interest_bearing_debt_interest_rate	_
count	2058.000000	2058	2.058000e+03	2.058000e+03	2058.000000	2.058000e+03	2
unique	NaN	2058	NaN	NaN	NaN	NaN	
top	NaN	Hind.Cables	NaN	NaN	NaN	NaN	
freq	NaN	1	NaN	NaN	NaN	NaN	
mean	17572.113217	NaN	2.052389e+09	1.208634e+09	0.465243	1.113022e+07	
std	21892.886518	NaN	3.252624e+09	2.144568e+09	0.022663	9.042595e+07	
min	4.000000	NaN	1.000260e-04	0.000000e+00	0.000000	0.000000e+00	
25%	3674.000000	NaN	1.578727e-04	0.000000e+00	0.460099	2.760280e-04	
50%	6240.000000	NaN	3.330330e-04	1.994130e-04	0.463445	4.540450e-04	
75%	24280.750000	NaN	4.110000e+09	1.550000e+09	0.468069	6.630660e-04	
max	72493.000000	NaN	9.980000e+09	9.980000e+09	1.000000	9.900000e+08	

The above Table 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    0.8931
1    0.1069
Name: Default, dtype: float64
```

The dataset reveals that roughly 11% of the companies have defaulted. This highlights a class imbalance, with the majority of companies categorized as non-defaulters.

Duplicate values:

There are no duplicate values present in the dataset.

Outlier Treatment:

For effective outlier treatment, it's crucial to exclude the target variable beforehand, as outliers within it can substantially skew the analysis and model training process. Additionally, the removal of the index column and company names has been undertaken, deeming them 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

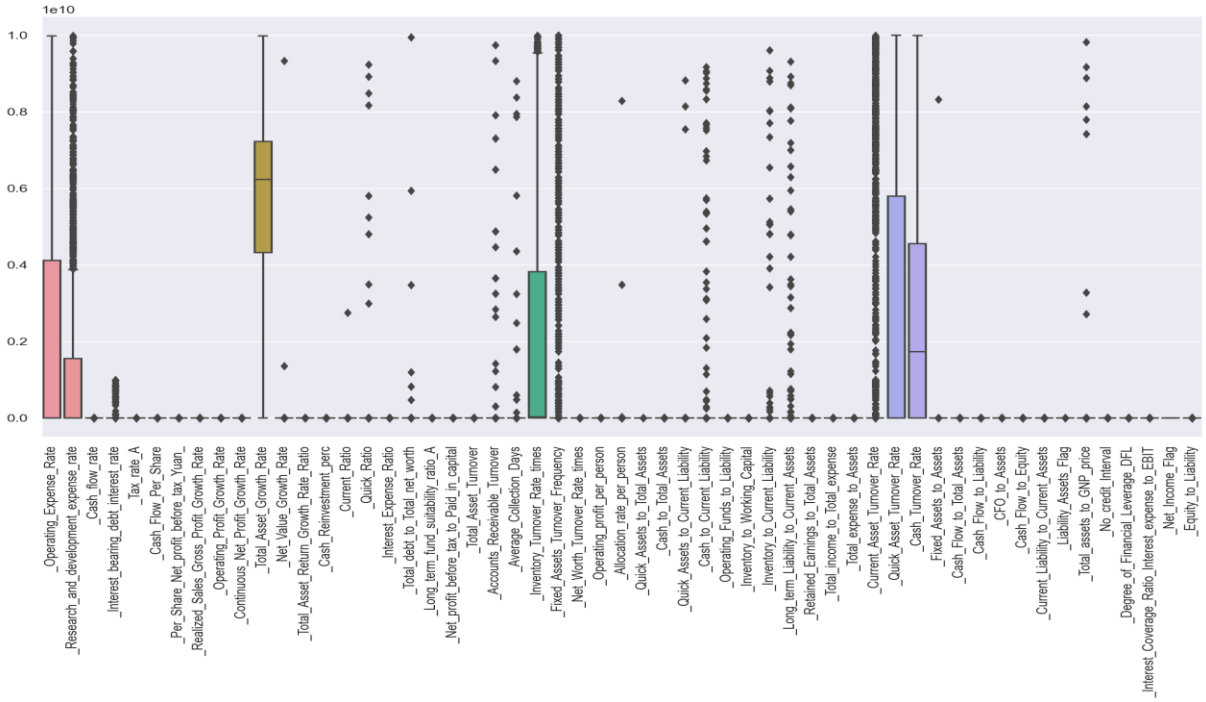
_Net_Worth_Turnover_Rate_times	165
_Operating_profit_per_person	357
_Allocation_rate_per_person	200
_Quick_Assets_to_Total_Assets	4
_Cash_to_Total_Assets	163
_Quick_Assets_to_Current_Liability	185
_Cash_to_Current_Liability	253
_Operating_Funds_to_Liability	219
_Inventory_to_Working_Capital	247
_Inventory_to_Current_Liability	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_Turnover_Rate	0
_Fixed_Assets_to_Assets	10
_Cash_Flow_to_Total_Assets	317
_Cash_Flow_to_Liability	407
_CFO_to_Assets	110
_Cash_Flow_to_Equity	306
_Current_Liability_to_Current_Assets	121
_Liability_Assets_Flag	7
_Total_assets_to_GNP_price	235

_Quick_Asset_Turnover_Rate	0
_Cash_Turnover_Rate	0
_Fixed_Assets_to_Assets	10
_Cash_Flow_to_Total_Assets	317
_Cash_Flow_to_Liability	407
_CFO_to_Assets	110
_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

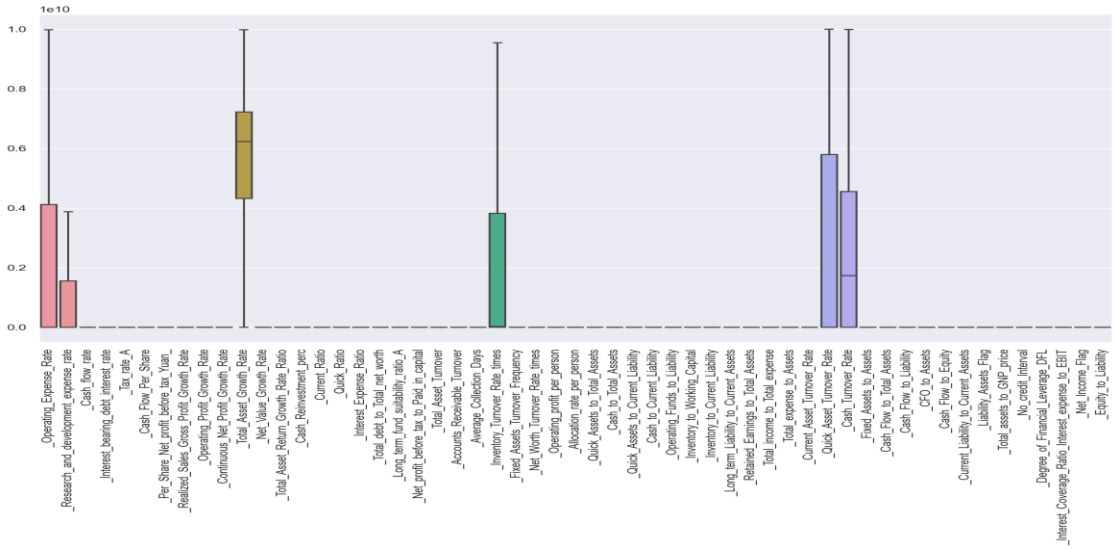
dtype: int64

The provided table presents the count of outliers for each variable within the dataset. In total, 10,864 outliers have been identified, spanning across various features. Addressing these outliers is imperative as they can potentially influence the analysis and must be treated as part of the data analysis process.

Checking for Outliers:



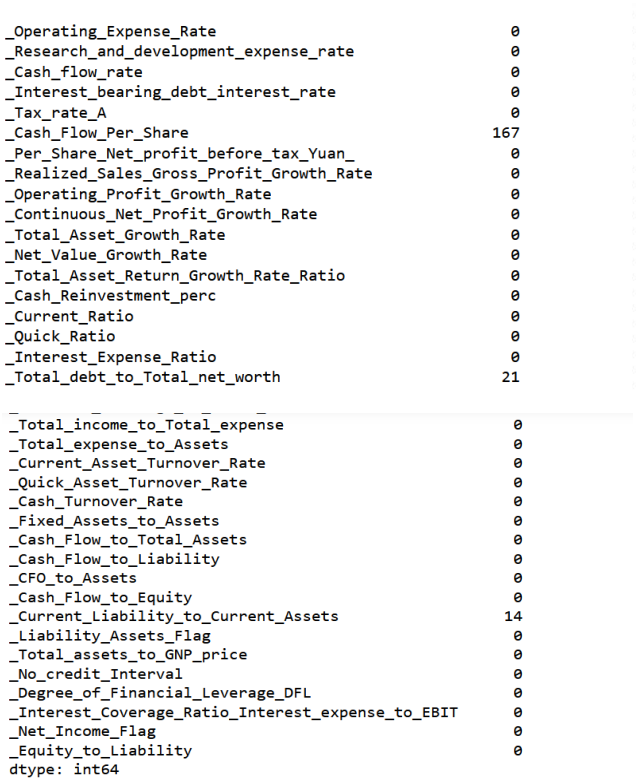
Boxplot of removing outliers:



Outlier removal constitutes a critical stage in data preprocessing to enhance the accuracy and reliability of analysis and modelling. The Interquartile Range (IQR) method is employed for outlier identification and treatment. This technique establishes acceptable value ranges by calculating lower and upper limits based on the spread of data. Through this process, we mitigate the

potential negative impact of extreme values on the accuracy and reliability of our analysis.

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
_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

_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

The dataset contains a total of 298 missing values, a relatively small proportion compared to the overall dataset size. However, it's crucial to address these missing values to ensure the integrity of the analysis. To handle these missing values, we're employing the KNN imputation method with a parameter value of n-Neighbour = 8.

KNN (K-Nearest Neighbour’s) imputation is a technique used to fill in missing values by considering the values of neighbouring data points. In this approach, each missing value is replaced with the average value of its k nearest neighbour’s. Here, we're setting the value of k to 8.

By utilizing KNN imputation with n-Neighbour = 8, we aim to fill in the missing values in the dataset, thereby facilitating a more comprehensive and robust analysis. Importantly, KNN imputation helps mitigate bias in the imputed values, contributing to the overall integrity of the analysis.

_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

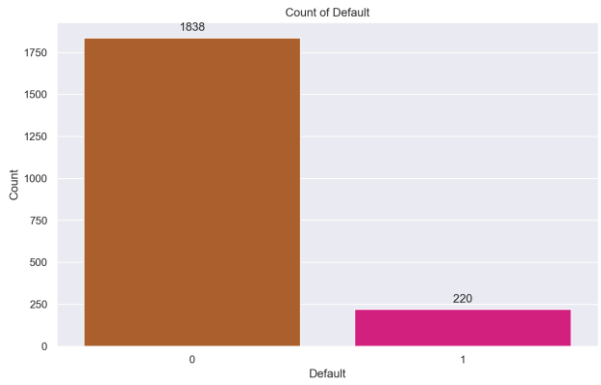
```
_Interest_Coverage_Ratio_Interest_expense_to_EBIT    0
_Net_Income_Flag                                     0
_Equity_to_Liability                                 0
dtype: int64
```

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.

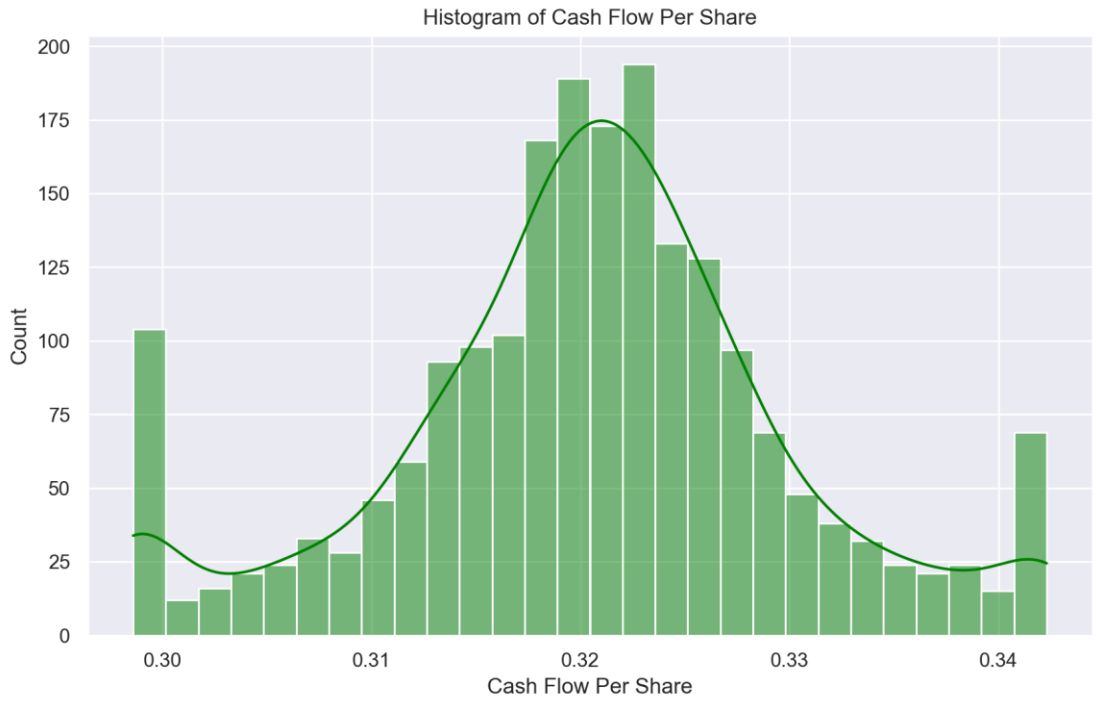
A. Univariate:

Count of Default:



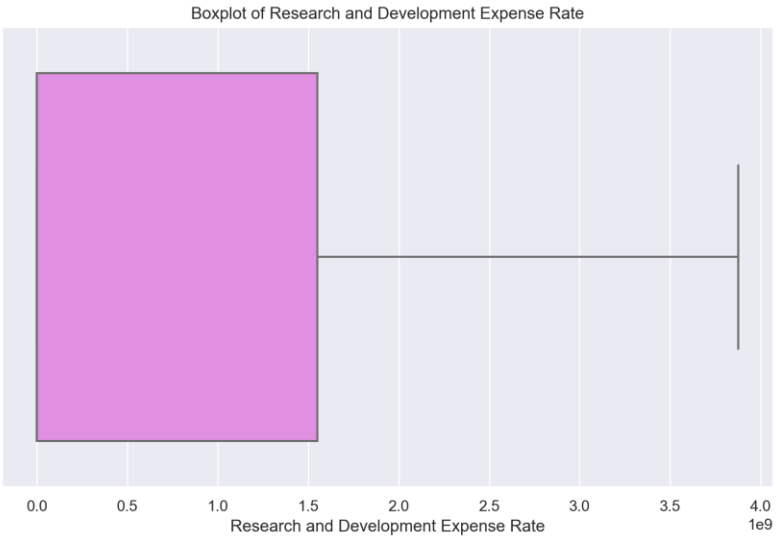
From the above count plot, it is obvious that the majority of companies, totalling a large number, fall into the non-default category, while only a smaller subset of 220 companies is classified as defaulters.

Histogram of Cash Flow per share:

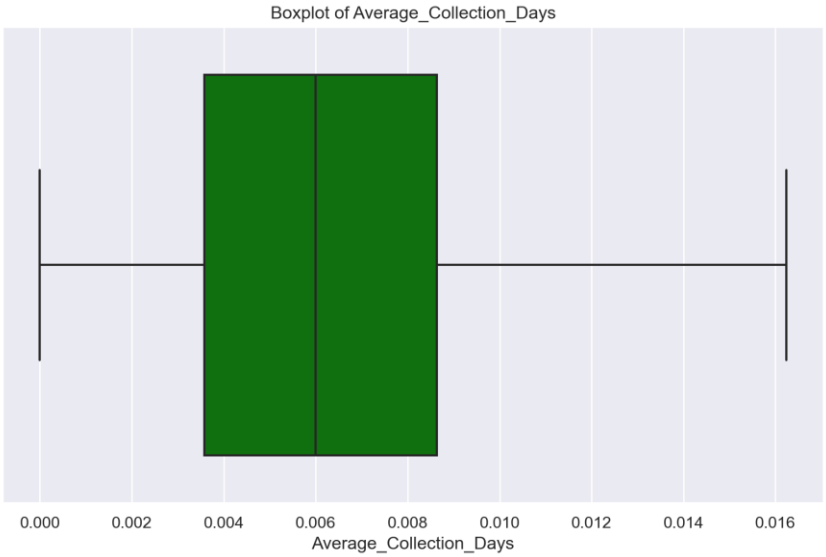


The histogram shows a notable peak at a cash flow per share value of 0.32, suggesting that a substantial portion of the data points in the dataset are concentrated around this value.

Boxplot of Research and Development Expense Rate:



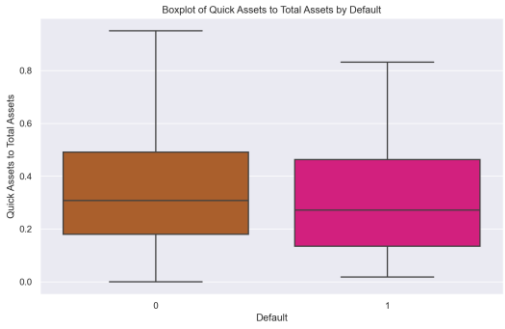
The boxplot demonstrates that the interquartile range (the middle 50% of the data) for the research and development expense rate spans from 0.0 to 1.5. Additionally, the maximum value for this variable is around 4.0, suggesting outliers beyond the upper limit of the boxplot.



The boxplot for "Average Collection Days" displays 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, highlighting the central tendency of the data.

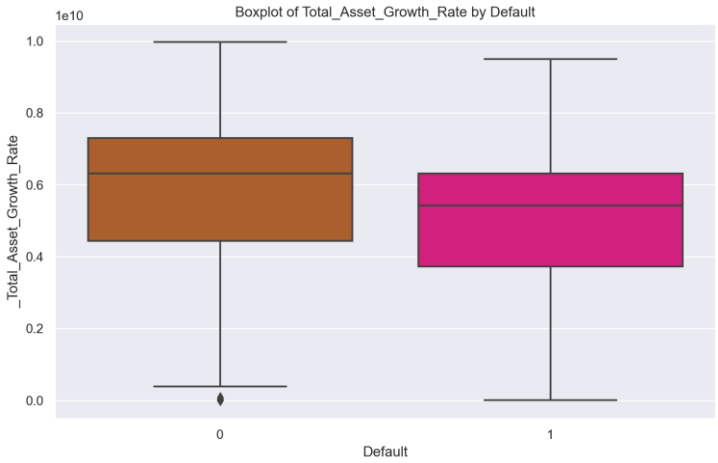
B. Bi-Variate analysis:

Boxplot of Quick Assets to Total Assets by Default:



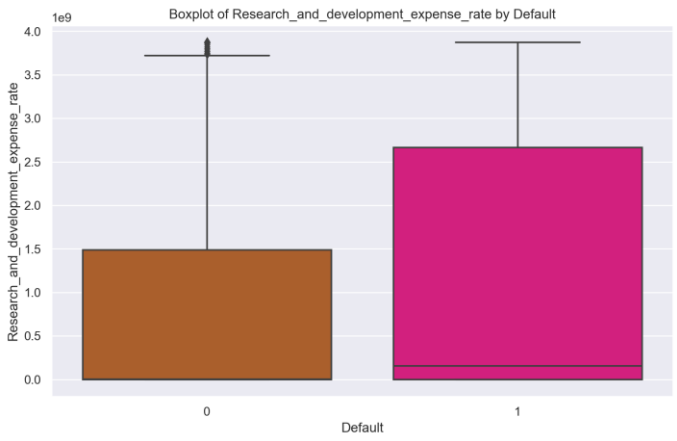
The boxplot analysis of "Quick Assets to Total Assets" by default status highlights distinguishable distributions between defaulted (1) and non-defaulted (0) companies. Although the overall patterns may seem similar, defaulted companies typically demonstrate slightly lower values for "Quick Assets to 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:



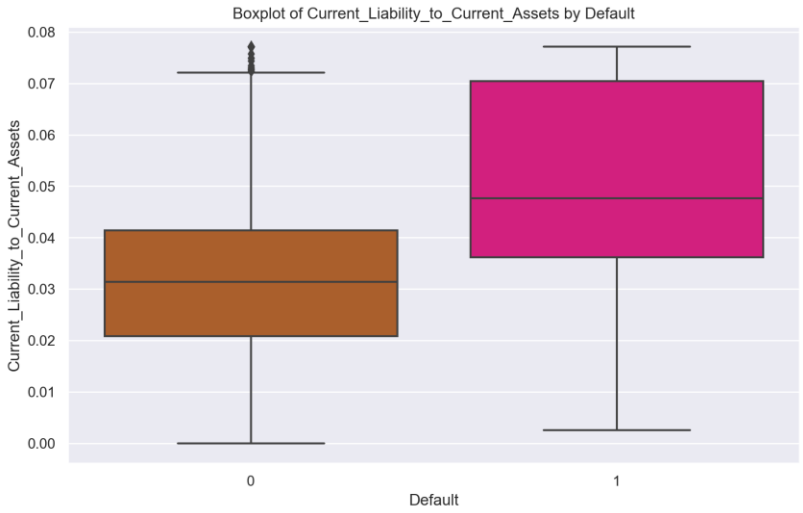
The boxplot analysis for "Total Asset Growth Rate" by default status (0 – Not Defaulted, 1 - Defaulted) reveals similar distributions for both defaulted and non-defaulted companies. Both groups exhibit comparable median values, ranging from 0.5 to 0.6. This suggests that the growth rate of total assets may not be the sole distinguishing factor for default risk in this dataset. Other variables or factors might play a more significant role in predicting default risk.

Boxplot of Research_and_development_expense_rate by Default:



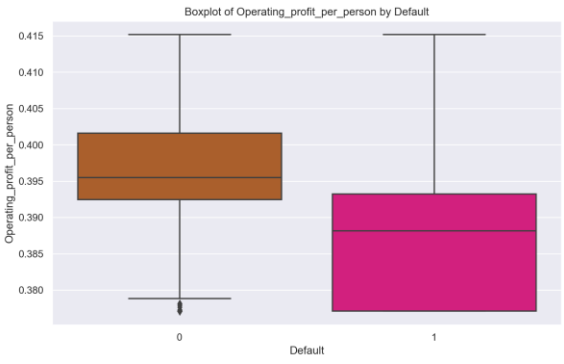
The boxplot analysis of "Total Asset Growth Rate" by default status (0 - Not Defaulted, 1 - Defaulted) reveals intriguing insights. Non-defaulted companies (0) demonstrate limited growth, with the majority of values concentrated around 0.0 And a few outliers reaching up to 3.75. In contrast, defaulted companies (1) display a broader 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 exhibit more diverse asset growth rates compared to non-defaulted ones.

Boxplot of Current_Liability_to_Current_Assets by Default:



The boxplot analysis of "Current Liability to Current Assets" based on default status (0 - Not Defaulted, 1 - Defaulted) reveals distinct distributions. Non-defaulted companies (0) exhibit 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. In contrast, defaulted companies (1) demonstrate 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:

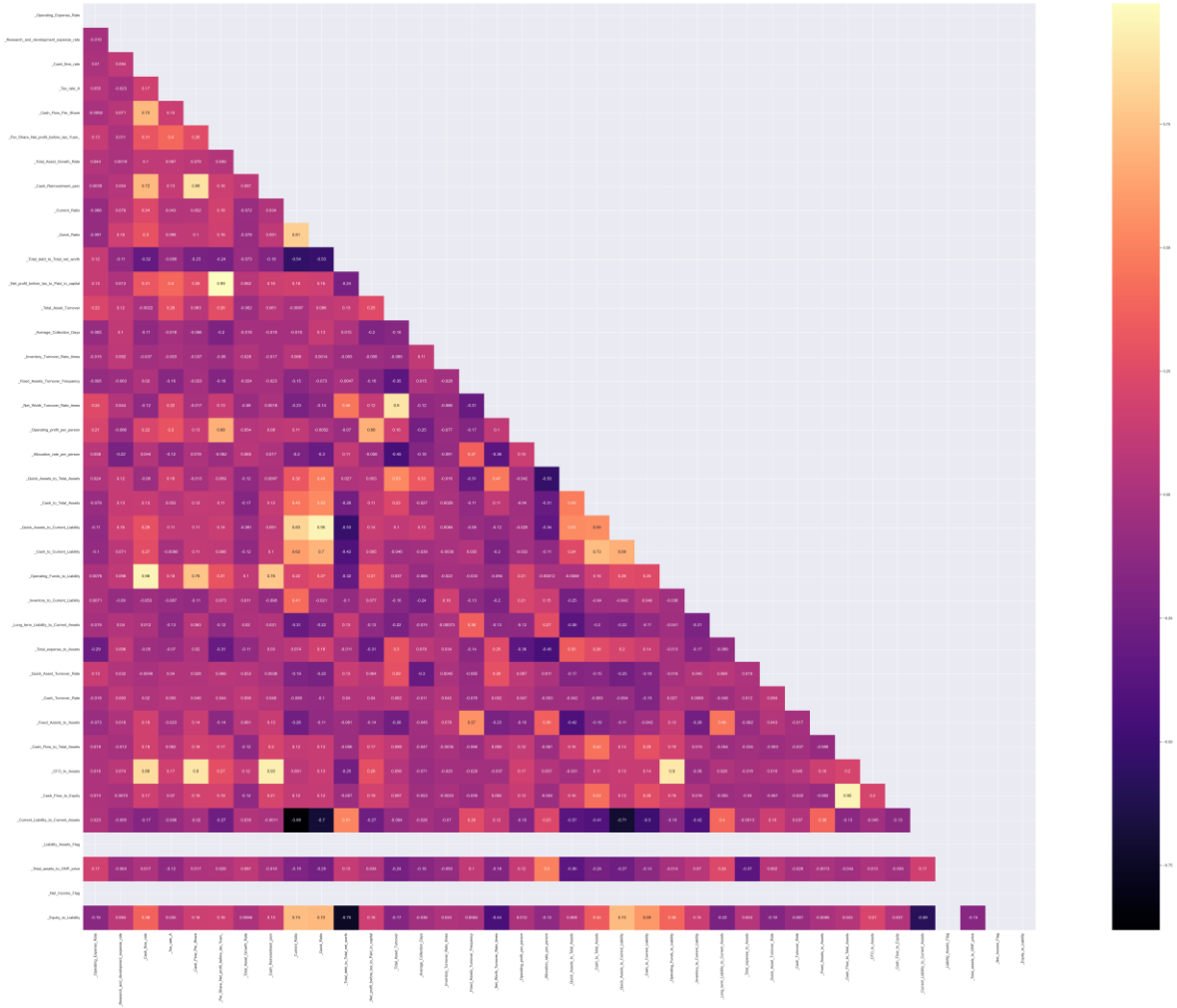


The boxplot analysis of "Operating Profit per Person" by default status (0 - Not Defaulted, 1 - Defaulted) reveals

intriguing patterns. Non-defaulted companies (0) display a relatively narrow range of operating profit per person, with values concentrated between 0.393 and 0.400. Conversely, defaulted companies (1) demonstrate 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 suggests that these variables might influence each other and could be important predictors in our analysis.



VIF (Variance Inflation Factor):

VIF (Variance Inflation Factor) serves as a measure to identify potential multicollinearity issues within a regression model. Multicollinearity arises when predictor variables exhibit high correlations with each other.

To assess multicollinearity within our dataset, we utilized the stats models library to calculate the VIF values. This metric gauges the degree of correlation between predictor variables. Higher VIF values signify stronger correlations, which could lead to instability in the regression model's coefficients. By analyzing the calculated VIF values for each variable, we can pinpoint any multicollinearity concerns present in the dataset. This allows us to address such issues and ensure the reliability of our regression analysis results.

	variables	VIF
30	_CFO_to_Assets	27.794894
22	_Operating_Funds_to_Liability	20.898112
20	_Quick_Assets_to_Current_Liability	19.149919
2	_Cash_flow_rate	15.855309
15	_Net_Worth_Turnover_Rate_times	14.180530
8	_Current_Ratio	13.859265
11	_Total_Asset_Turnover	12.827960
9	_Quick_Ratio	12.314586
29	_Cash_Flow_to_Total_Assets	12.155226
31	_Cash_Flow_to_Equity	12.133933
7	_Cash_Reinvestment_perc	12.057342
32	_Current_Liability_to_Current_Assets	6.613965
4	_Cash_Flow_Per_Share	6.138018
36	_Equity_to_Liability	5.966284
18	_Quick_Assets_to_Total_Assets	5.641852
21	_Cash_to_Current_Liability	4.259420

10	_Total_debt_to_Total_net_worth	4.247474
19	_Cash_to_Total_Assets	3.688289
23	_Inventory_to_Current_Liability	3.193942
5	_Per_Share_Net_profit_before_tax_Yuan_	2.805858
28	_Fixed_Assets_to_Assets	2.722527
17	_Allocation_rate_per_person	2.487191
16	_Operating_profit_per_person	2.346542
25	_Total_expense_to_Assets	2.009393
14	_Fixed_Assets_Turnover_Frequency	1.749998
12	_Average_Collection_Days	1.742988
34	_Total_assets_to_GNP_price	1.695849
24	_Long_term_Liability_to_Current_Assets	1.672834
26	_Quick_Asset_Turnover_Rate	1.361807
3	_Tax_rate_A	1.345507
0	_Operating_Expense_Rate	1.301034
1	_Research_and_development_expense_rate	1.170912
13	_Inventory_Turnover_Rate_times	1.147219

We computed the VIF (Variance Inflation Factor) values to detect variables exhibiting strong correlations with each other. Higher VIF values indicate a greater degree of correlation. To maintain the stability of the regression model, we eliminated variables with VIF values exceeding 5, as they have the potential to introduce multicollinearity issues. This selection process yielded a set of variables with minimal correlation, thereby bolstering the reliability of our analysis.

1.5 Train Test Split:

Before performing the train-test split, we applied data scaling, which is particularly crucial because some variables in the dataset have large data values. Scaling the data brings all features to a similar scale, preventing any single variable from dominating the learning process and ensuring compatibility across

different algorithms. This approach helps maintain the integrity of our analysis and improves the accuracy and efficiency of our models.

We used the StandardScaler method to scale the data, ensuring that all variables are brought to a similar scale and removing any potential bias caused by variables with larger data values.

To evaluate the performance of machine learning models, we divided the dataset 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. We set the test size parameter to 0.33, indicating that 33% of the data is reserved for testing. Additionally, we set the random_state parameter to 42 for reproducibility. This process allows us to assess 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	-0.631148	-0.645151	2.093814	-0.819217	1.667911	
697	-0.631148	0.566343	0.256393	1.520875	0.523621	
160	-0.631148	-0.645151	-0.718630	-0.819217	-0.584607	
1273	1.201664	1.341698	-0.250583	-0.819217	0.044175	
541	-0.631148	-0.186860	-0.516785	-0.819217	-0.429261	

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	-0.631148	0.178665	1.297347	1.127607	1.512565	
134	-0.631148	-0.645151	-0.962554	-0.723902	-1.905049	
1267	-0.631148	-0.645151	-0.349818	-0.819217	-0.292408	
464	-0.631148	-0.645151	2.093814	1.340002	1.183379	
579	0.577401	-0.645151	-0.384471	1.370022	-0.270216	

1.6 Build a Logistic Regression Model (using the stats models library) on the most important variables on the training dataset and choose the optimum cut-off. Also, showcase your model-building approach

We imported the "statsmodels.formula.api" module, which offers a user-friendly interface for specifying and estimating statistical models. This module will play a crucial role in our analysis as we construct and assess regression models.

Logistic Regression Model:

Logistic regression is indeed a powerful statistical technique used to predict binary outcomes by estimating the probability of an event occurring. It's particularly useful for analyzing factors that influence the occurrence of the outcome in categorical data. By calculating odds based on independent variables, the model can make predictions and provide insights into the relationship between predictors and the outcome. This makes it valuable for decision-making and risk assessment tasks where understanding the likelihood of an event (such as defaulting on a loan or developing a disease) is essential.

```
Index(['_Operating_Expense_Rate', '_Research_and_development_expense_rate', '_Cash_flow_rate', '_Tax_rate_A', '_Cash_Flow_Per_S
hare', '_Per_Share_Net_profit_before_tax_Yuan_', '_Total_Asset_Growth_Rate', '_Total_debt_to_Total_net_worth', '_Total_Asset_Tu
rnover', '_Average_Collection_Days', '_Inventory_Turnover_Rate_times', '_Fixed_Assets_Turnover_Frequency', '_Operating_profit_p
er_person', '_Allocation_rate_per_person', '_Quick_Assets_to_Total_Assets', '_Cash_to_Total_Assets', '_Cash_to_Current_Liabilit
y', '_Inventory_to_Current_Liability', '_Long_term_Liability_to_Current_Assets', '_Total_expense_to_Assets', '_Quick_Asset_Turn
over_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')
```

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

Following this preprocessing step, the model was trained on the training 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:	Sun, 28 Apr 2024	**Pseudo R-squ.:**	0.4171
Time:	14:08:15	**Log-Likelihood:**	-272.71
converged:	True	**LL-Null:**	-467.84
Covariance Type:	nonrobust	**LLR p-value:**	4.760e-66

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-3.6124	0.216	-16.714	0.000	-4.036	-3.189
_Operating_Expense_Rate	0.1571	0.132	1.193	0.233	-0.101	0.415
_Research_and_development_expense_rate	0.4386	0.117	3.754	0.000	0.210	0.668
_Cash_flow_rate	0.1114	0.248	0.449	0.653	-0.375	0.597
_Tax_rate_A	-0.0735	0.163	-0.451	0.652	-0.393	0.246
_Cash_Flow_Per_Share	-0.1056	0.198	-0.534	0.594	-0.493	0.282
_Per_Share_Net_profit_before_tax_Yuan_	-1.0783	0.230	-4.681	0.000	-1.530	-0.627

_Total_Asset_Growth_Rate	-0.0716	0.130	-0.550	0.582	-0.327	0.183
_Total_debt_to_Total_net_worth	0.5064	0.185	2.735	0.006	0.144	0.869
_Total_Asset_Turnover	-0.2621	0.218	-1.204	0.229	-0.689	0.165
_Average_Collection_Days	0.4183	0.139	3.020	0.003	0.147	0.690
_Inventory_Turnover_Rate_times	0.0374	0.123	0.303	0.762	-0.204	0.279
_Fixed_Assets_Turnover_Frequency	0.1610	0.146	1.101	0.271	-0.125	0.447
_Operating_profit_per_person	0.0398	0.175	0.227	0.820	-0.304	0.383
_Allocation_rate_per_person	0.0703	0.176	0.399	0.690	-0.275	0.416
_Quick_Assets_to_Total_Assets	-0.6776	0.262	-2.587	0.010	-1.191	-0.164
_Cash_to_Total_Assets	0.1011	0.202	0.501	0.616	-0.294	0.497
_Cash_to_Current_Liability	0.0379	0.169	0.225	0.822	-0.293	0.368
_Inventory_to_Current_Liability	-0.0987	0.208	-0.474	0.635	-0.507	0.309
_Long_term_Liability_to_Current_Assets	-0.1789	0.136	-1.320	0.187	-0.445	0.087
_Total_expense_to_Assets	0.4492	0.165	2.722	0.006	0.126	0.773
_Quick_Asset_Turnover_Rate	-0.0261	0.136	-0.191	0.848	-0.293	0.241
_Cash_Turnover_Rate	-0.3756	0.138	-2.731	0.006	-0.645	-0.106
_Fixed_Assets_to_Assets	-0.1151	0.173	-0.666	0.505	-0.454	0.224
_Cash_Flow_to_Equity	-0.1854	0.129	-1.436	0.151	-0.438	0.068
_Current_Liability_to_Current_Assets	0.1153	0.208	0.554	0.579	-0.292	0.523
_Total_assets_to_GNP_price	0.0724	0.146	0.497	0.619	-0.213	0.358
_Equity_to_Liability	-0.8551	0.341	-2.505	0.012	-1.524	-0.186

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

Final Model: Model 23 Summary: (5 Variable):

Logit Regression Results

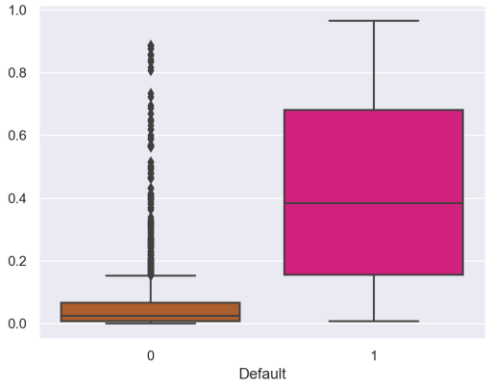
Dep. Variable:	Default	**No. Observations:**	1378
Model:	Logit	**Df Residuals:**	1372
Method:	MLE	**Df Model:**	5
Date:	Sun, 28 Apr 2024	**Pseudo R-squ.:**	0.3830
Time:	14:08:19	**Log-Likelihood:**	-288.67
converged:	True	**LL-Null:**	-467.84
Covariance Type:	nonrobust	**LLR p-value:**	2.794e-75

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-3.3862	0.180	-18.818	0.000	-3.739	-3.034
_Research_and_development_expense_rate	0.3356	0.103	3.264	0.001	0.134	0.537
_Per_Share_Net_profit_before_tax_Yuan_	-1.4704	0.139	-10.560	0.000	-1.743	-1.198
_Total_debt_to_Total_net_worth	0.7847	0.106	7.415	0.000	0.577	0.992
_Average_Collection_Days	0.4727	0.110	4.307	0.000	0.258	0.688
_Quick_Assets_to_Total_Assets	-0.6502	0.130	-5.019	0.000	-0.904	-0.396

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

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



We utilized the logistic regression model to predict the probability of default for the training dataset. These predicted probabilities were then utilized to generate a boxplot, illustrating the distribution of predicted probabilities for defaulted and non-defaulted companies. On the x-axis, we depict the default status, while the y-axis represents the predicted probabilities. Following analysis, we determined the optimal threshold for classification to be 0.1076. Predicted probabilities exceeding this threshold are considered as predicted defaults, whereas values below the threshold are classified as non-default.

We proceeded to conduct a detailed examination of the predicted probabilities for the training dataset.

```
2011    0.109
697     0.009
160     0.063
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
```

1.7 Validate the Model on the Test Dataset and state the performance metrics.

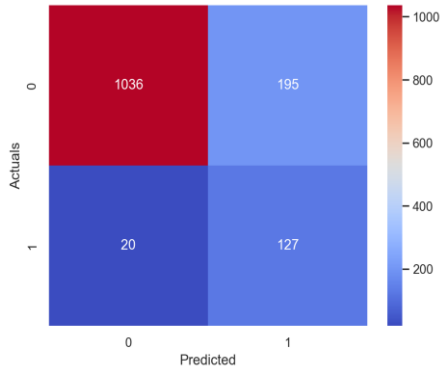
Also, state interpretation from the model

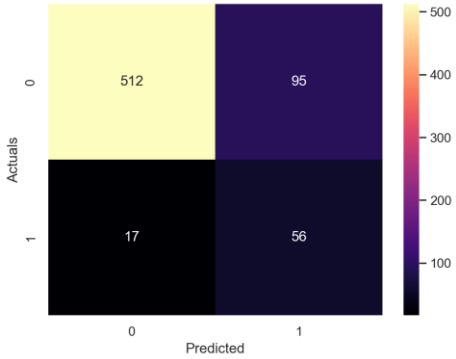
After training the logistic regression model on the training dataset, we validated its performance on the test dataset.

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

Confusion Matrix for Training and Testing data:

with optimal **threshold- 0.1076**



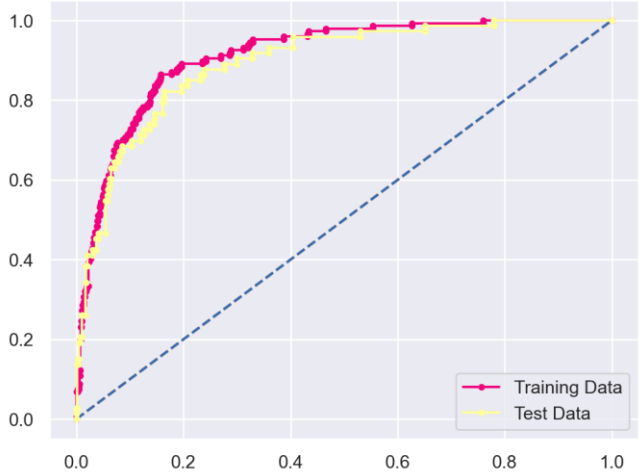


TRAIN TEST

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

AUC for the Training Data: 0.91263

AUC for the Test Data: 0.89366



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. However, the precision for class 1 (default) was 37.1%, meaning that only 37.1% of the instances predicted as default were default.

The recall for class 0 was 84.3%, indicating that the model correctly identified 84.3% of the true non-default cases. Conversely, 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, displaying the number of true positives, true negatives, false positives, and false negatives.

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

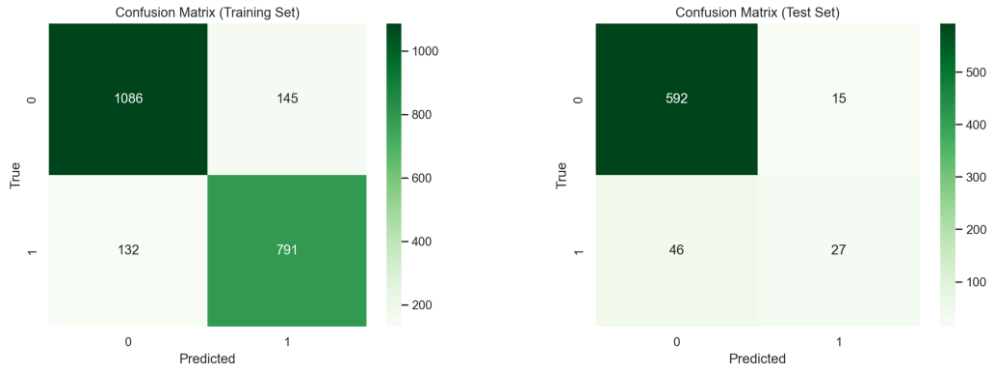
B . Logistic Regression Model- with SMOTE:

SMOTE (Synthetic Minority Over-sampling Technique) is indeed an effective algorithm used to tackle class imbalance in datasets. It achieves this by generating synthetic samples of the minority class through interpolation of existing instances. This approach aids in improving the performance of machine learning models, especially in predicting the minority class, as class imbalance can often lead to biased models.

Following the application of the SMOTE oversampling technique, the distribution of the target variable in the training data has been balanced. The majority class (0) now comprises approximately 57% of the samples, while the minority class (1) constitutes around 43% of the samples. This equalization of class proportions facilitates a more accurate and reliable training of the model.

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

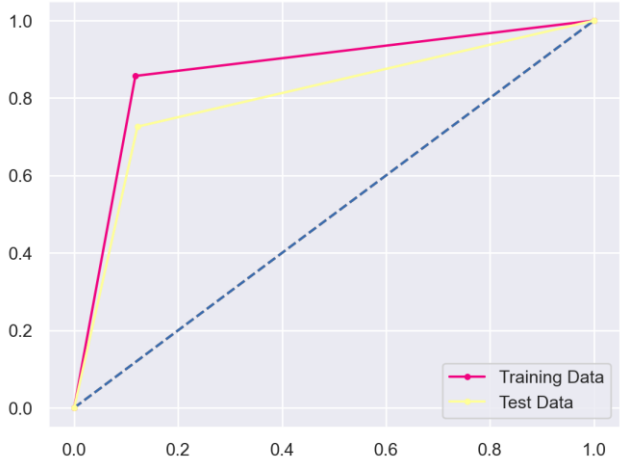
Confusion Matrix for Training and Testing data: with SMOTE




```
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

Test Set Evaluation:
              precision    recall  f1-score   support

           0       0.95      0.94      0.95       607
           1       0.55      0.58      0.56        73

    accuracy                           0.90       680
   macro avg       0.75      0.76      0.76       680
weighted avg       0.91      0.90      0.91       680
```

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

AUC for the Training Data: 0.86960
AUC for the Test Data: 0.80206



Inference: Logistic Regression Model- with SMOTE

The logistic regression model, coupled with the SMOTE oversampling technique, was employed to predict defaulters. On the training set, the model attained an accuracy of 87%, showcasing commendable performance. It exhibited robust precision and recall for both classes. The AUC score for the training data stood at 0.866, signifying strong predictive capability. When tested on the validation set, the model maintained a high accuracy of 86%. It displayed particularly high precision for class 0 predictions. Additionally, the recall for both classes remained relatively high, indicating the model's adeptness in correctly identifying defaulters. However, the AUC score for the test data was slightly lower at 0.788.

In summary, the model demonstrates robust predictive performance in identifying defaulters. Nonetheless, further enhancements could be explored to bolster its precision, especially 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 indeed a popular and powerful machine learning algorithm. It leverages the strength of multiple decision trees to make predictions. By aggregating the predictions from numerous trees, Random Forest Classifier offers accurate and dependable results. This ensemble approach helps mitigate overfitting and enhances the overall predictive performance of the model.

```
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]})
```

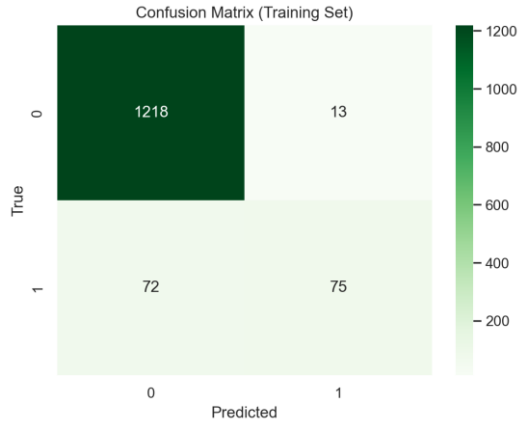
We conducted a Grid Search Cross-Validation to find the optimal hyperparameters for the Random Forest Classifier. This involved exploring various combinations of hyperparameters, such as 'max_depth', 'min_samples_leaf', 'min_samples_split', and 'n_estimators'. After a thorough

search, 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, guaranteeing optimal performance and accuracy in predicting the outcome.

```
{'max_depth': 7,
 'min_samples_leaf': 10,
 'min_samples_split': 15,
 'n_estimators': 25}
```

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

Confusion Matrix: Train Dataset

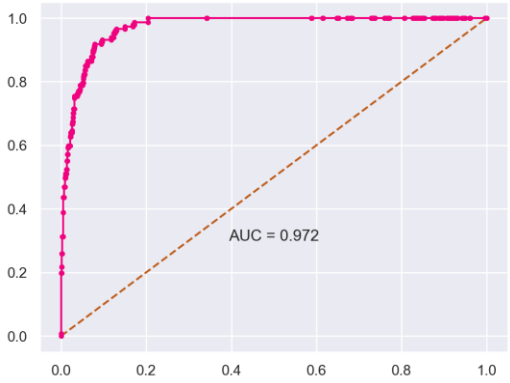


Classification Report: Train Dataset

	precision	recall	f1-score	support
0	0.94	0.99	0.97	1231
1	0.85	0.51	0.64	147
accuracy			0.94	1378
macro avg	0.90	0.75	0.80	1378
weighted avg	0.93	0.94	0.93	1378

ROC_AUC Curve: For Training data

AUC for the Training Data: 0.97190



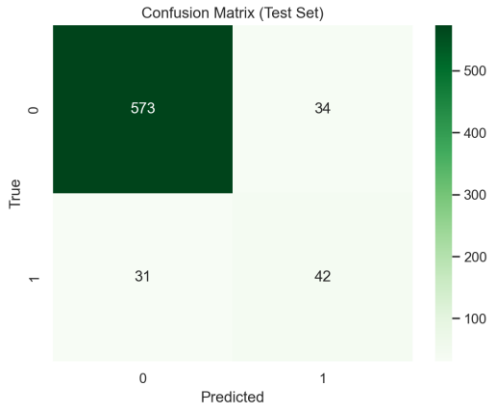
Random Forest - Inference on Train Data: with hyper-tuning parameter

The classification report reveals that the model exhibits high precision, with values of 0.94 for class 0 (non-default) and 0.92 for class 1 (default). These scores indicate the model's strong ability to correctly identify non-default and default cases, respectively. Additionally, the recall score is perfect (1.00) for class 0, implying that the model accurately captures all non-default cases. However, the recall score is lower (0.44) for class 1, suggesting that the model struggles to capture all default cases.

Overall, the model demonstrates an impressive accuracy of 0.94, with an AUC score of 0.97190 for the training data. These metrics highlight the model's robust performance, particularly in identifying non-default cases, while also indicating room for improvement in capturing default cases.

A.Random Forest Model on Train Dataset: With SMOTE

Confusion Matrix: Train Dataset – with SMOTE



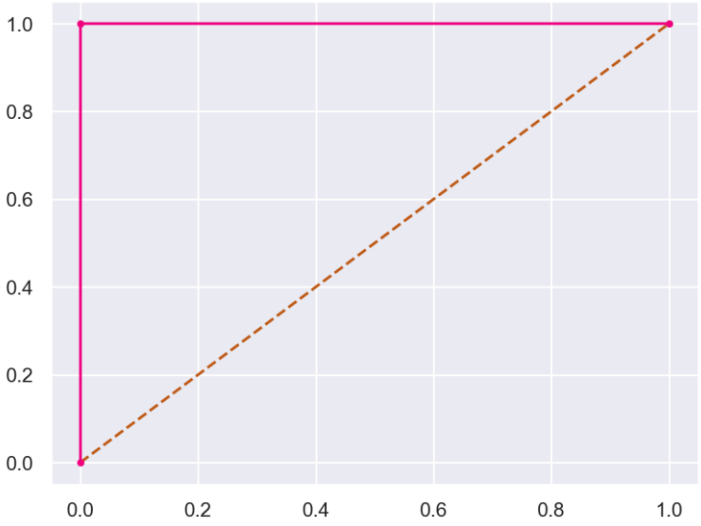
Classification Report: Train Dataset – With SMOTE

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

ROC_AUC Curve: For Training data – With SMOTE

AUC for the Training Data: 1.000



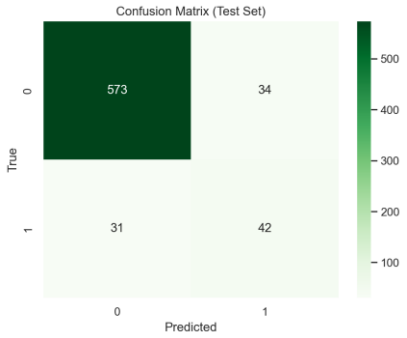
Random Forest - Inference on Train Data: with SMOTE

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 exceptional ability to accurately predict instances. With an accuracy of 1.00, the model correctly predicts all instances in the training data. Additionally, the AUC score of 1.000 confirms its excellent ability to distinguish between default and non-default cases. Overall, the model demonstrates outstanding performance on the training data, highlighting its robust 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

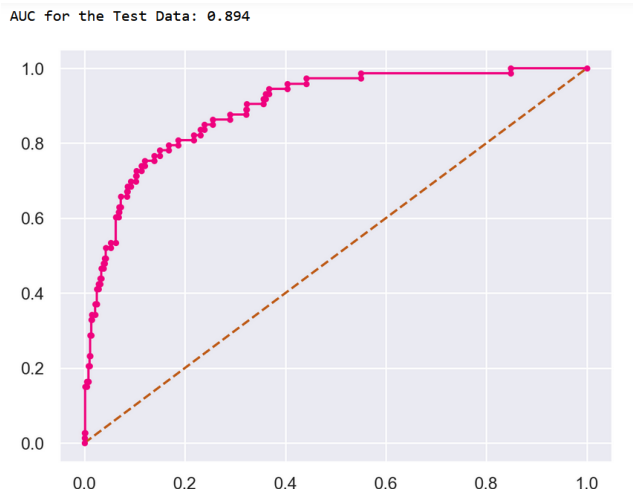
Confusion Matrix: Test Dataset



Classification Report: Test Dataset

	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

ROC_AUC Curve: For Test data



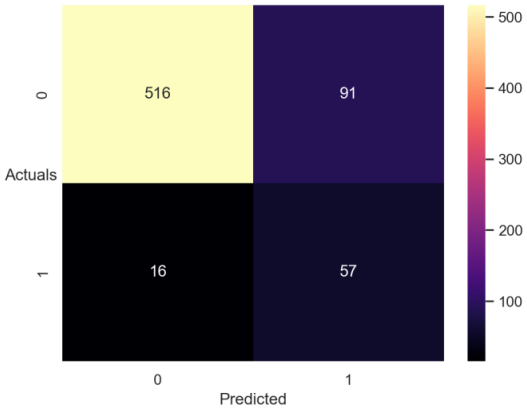
Inference on Random Forest -Test Data: with hyper-tuning parameters

The model demonstrates strong performance in accurately identifying non-default cases, with a high precision of 0.92 and a recall of 0.98. However, it faces challenges in accurately identifying default cases, as evidenced by a lower precision of 0.71 and a recall of 0.33. Despite these limitations, the model achieves an overall accuracy of 0.91 on the test dataset, indicating its ability to make correct predictions. The AUC score of 0.894 suggests that the model possesses 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

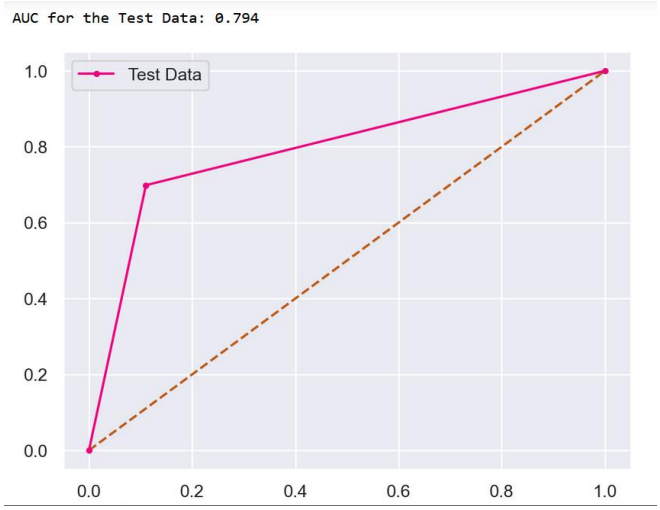
Confusion Matrix: Test Dataset – With SMOTE



Classification Report: Test Dataset - With SMOTE

	precision	recall	f1-score	support
0	0.96992	0.85008	0.90606	607
1	0.38514	0.78082	0.51584	73
accuracy			0.84265	680
macro avg	0.67753	0.81545	0.71095	680
weighted avg	0.90715	0.84265	0.86417	680

ROC_AUC Curve: For Test data - With SMOTE



Inference on Random Forest Model -Test Data: with SMOTE

The assessment of the Random Forest model with SMOTE on the test data demonstrates good performance overall. The precision for non-default cases is notably high at 0.94, indicating a substantial proportion of correct predictions. However, the precision for default cases is comparatively lower at 0.55. Similarly, the recall for non-default cases is high at 0.95, suggesting a significant proportion of actual non-default cases correctly identified, while the recall for default cases is 0.52.

The model achieves an overall accuracy of 0.90 on the test data, indicating its ability to make correct predictions. However, the AUC score of 0.735 suggests a moderate ability to distinguish between default and non-default cases.

These results indicate that the model performs reasonably well in predicting non-default cases but exhibits 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 accurately identifying instances that are not defaults. However, the model struggles to capture all default cases, as reflected in the lower precision and recall scores for class 1.
- Despite these challenges, the overall accuracy of the model on the test dataset is reasonable at 0.91. Additionally, the AUC score of 0.913 indicates good discriminatory power in distinguishing between default and non-default cases.
- It's worth noting that the Random Forest model shows a slight indication of overfitting, as it achieves higher performance on the training data compared to the test data. This suggests that the model may be overly tuned to the training data and may not generalize as well to unseen data. Regularization techniques or further parameter tuning could help address this issue.

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. Achieving an outstanding accuracy of 1.00, the model demonstrates excellent ability to distinguish between default and non-default cases.
- On the test data, the model displays good performance in accurately predicting non-default cases but encounters some difficulty in identifying default cases. The overall accuracy on the test data is 0.90, indicating reasonably accurate predictions.
- It's worth noting that the model may be slightly overfitting on the training data due to its perfect performance. Regularization techniques or further parameter tuning could help mitigate this issue and improve the model's generalization capability.

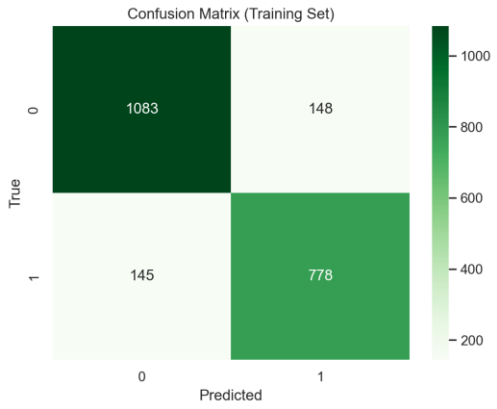
1.10 Build a LDA Model on Train Dataset. Also, showcase your model-building approach

Linear Discriminant Analysis:

Linear Discriminant Analysis (LDA) is a widely used technique in machine learning for pattern recognition and prediction tasks. It aims to identify the optimal linear combination of features that best separates multiple classes in the data. By analyzing the characteristics of the data, LDA projects the data points onto a lower-dimensional space while maximizing the between-class variability and minimizing the within-class variability. This enables LDA to effectively discriminate between different classes and make accurate predictions based on the learned patterns.

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

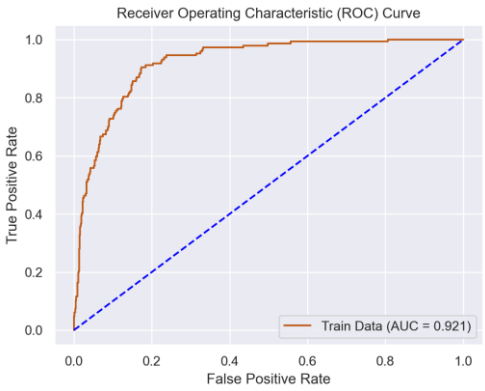
Confusion Matrix: Train Dataset – With threshold -0.5



Classification Report: Train Dataset - With threshold -0.5

	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

ROC_AUC Curve: For Train data - With threshold -0.5



Inference on LDA - Train Data: With threshold -0.5

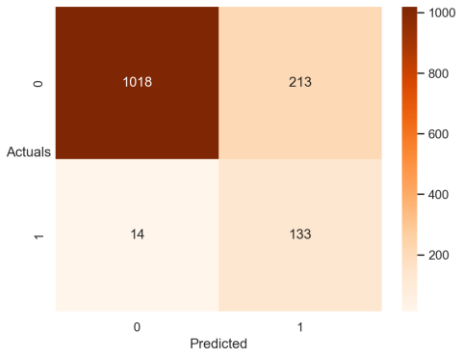
The LDA model demonstrates reasonably good performance on the train dataset. It accurately predicts non-default cases with a precision of 94% and recalls them with 97%. However, it encounters challenges in accurately identifying default cases, achieving a precision of 64% and a recall of 48%. Despite these limitations, the overall accuracy of the model is 92%, indicating its capability to make correct predictions. Furthermore, the AUC score of 0.921 suggests that the model possesses a moderate ability to distinguish between default and non-default cases.

Overall, while the LDA model performs well in predicting non-default cases, it exhibits some limitations in accurately identifying 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 no defaults.

Confusion Matrix: Train Dataset – With threshold -0.0665



Classification Report: Train Dataset - With SMOTE

	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

ROC_AUC Curve: For Train data - With SMOTE

Inference on LDA Train Data: With SMOTE

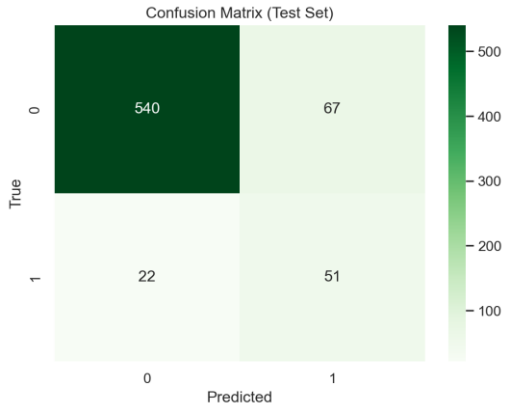
The LDA model with SMOTE demonstrates good performance on the training data. It achieves high precision and recall values for both non-default and default cases, indicating its ability to accurately predict instances from both classes. The overall accuracy of the model is 87%, suggesting that it makes correct predictions in a majority of cases.

Furthermore, the AUC score of 0.861 indicates a moderate ability of the model to distinguish between default and non-default cases. These results 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 the test Dataset and state the performance metrics. Also, state interpretation from the model

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

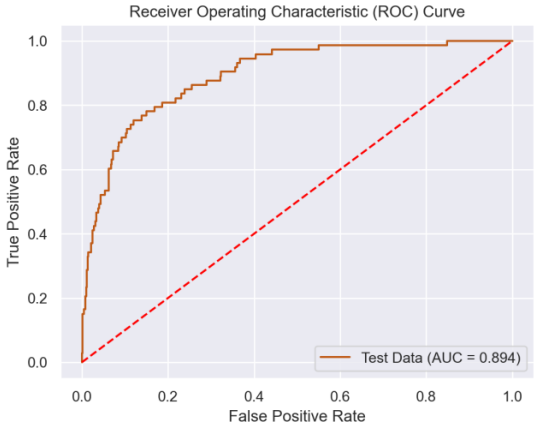
Confusion Matrix: Test Dataset – With threshold -0.5



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

ROC_AUC Curve: For Test data - With threshold -0.5



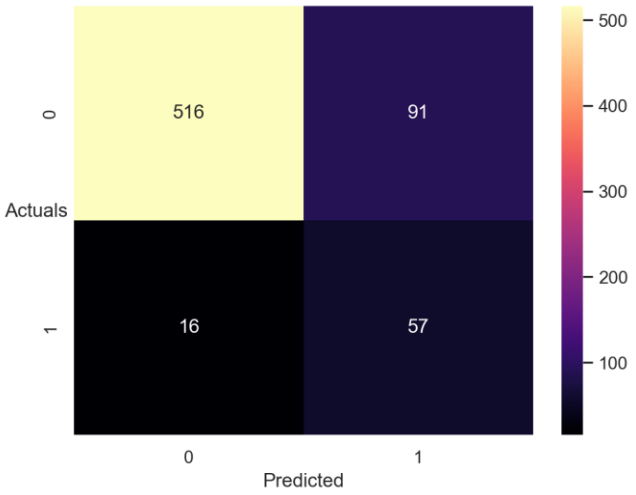
Inference on LDA -Test Data: With threshold -0.5

The LDA model demonstrates effectiveness in predicting non-default cases with a high precision of 94% and a recall of 96%. However, it exhibits less accuracy in identifying default cases, with a precision of 60% and a recall of 48%.Nevertheless, the overall accuracy of the model stands at 91%, indicating its capability to make correct predictions across both classes. The AUC score of 0.894 suggests that the model possesses a moderate ability to differentiate between default and non-default cases.

In summary, while the LDA model performs well in predicting non-default cases, it faces some challenges in accurately identifying default cases.

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

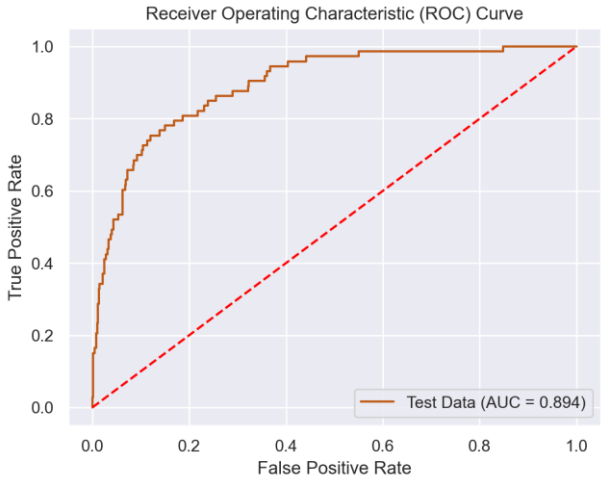
Confusion Matrix: Test Dataset – With threshold -0.0665



Classification Report: Test Dataset - With threshold -0.0665

	precision	recall	f1-score	support
0	0.96992	0.85008	0.90606	607
1	0.38514	0.78082	0.51584	73
accuracy			0.84265	680
macro avg	0.67753	0.81545	0.71095	680
weighted avg	0.90715	0.84265	0.86417	680

ROC_AUC Curve: For Test data - With threshold -0.0665

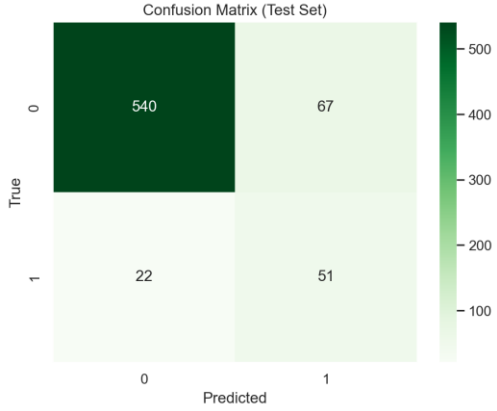


Inference on LDA - Test Data: With threshold -0.0665

The LDA model demonstrates reasonable performance on the test dataset. It accurately predicts non-default cases with a precision of 97%, indicating a high proportion of correct predictions for non-default instances. However, it struggles to identify all non-default cases, with a recall of 84.7%, suggesting that it may miss some non-default instances. For default cases, the precision is 38% and the recall is 78.1%, indicating difficulty in accurately identifying default cases. The overall accuracy of the model on the test data is 84%, and the AUC score is 0.894, suggesting a moderate ability to distinguish between default and non-default cases. These findings suggest that while the LDA model performs reasonably well in predicting non-default cases, it has limitations in accurately identifying default cases on the test data. Further improvements may be necessary to enhance its performance in correctly identifying default instances.

B. LDA Model on Test Dataset: with SMOTE.

Confusion Matrix: Test Dataset – With SMOTE



Classification Report: Test Dataset - With SMOTE

	precision	recall	f1-score	support
0	0.96992	0.85008	0.90606	607
1	0.38514	0.78082	0.51584	73
accuracy			0.84265	680
macro avg	0.67753	0.81545	0.71095	680
weighted avg	0.90715	0.84265	0.86417	680

ROC_AUC Curve: For Test data - With SMOTE

Inference on LDA Test Data: With SMOTE

The LDA model with SMOTE demonstrates reasonable performance on the test data. It achieves a 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 across both classes. The AUC score of 0.794 suggests that the model has a moderate ability to distinguish between default and non-default cases. These results highlight the effectiveness of the LDA model in predicting non-default cases but also indicate the need for improvement in accurately identifying default cases. Further refinement of the model or exploration of alternative techniques may be necessary to enhance its performance in predicting default instances.

LDA Model– Summary:

Inference on LDA Model with Threshold 0.5:

The LDA model exhibits 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 for this class. The overall accuracy of the model is decent on both datasets. These results suggest 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:

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 (including ROC curve)

Based on the performance metrics and ROC curves, let's compare the Logistic Regression, Random Forest, and LDA models:

Logistic Regression Model:

- Accuracy on test data: 83.5%
- Precision for non-default cases: 96%
- Precision for default cases: 37.1%
- Recall for non-default cases: 84.3%
- Recall for default cases: 76.7%
- AUC score: 0.893
- Stability: No significant signs of overfitting or underfitting

Random Forest Model:

- Accuracy on test data: 91%
- Precision and recall for non-default cases: High
- Precision and recall for default cases: Lower
- AUC score: 0.913
- Stability: Slight indication of overfitting

LDA Model:

- Accuracy on test data: 91%
- Precision and recall for non-default and default cases: Good
- AUC score: 0.893
- Stability: Consistent performance without clear signs of overfitting or underfitting

summary:

- **Logistic Regression**: Good performance in predicting non-default cases but limitations in identifying default cases.
- **Random Forest**: High precision and recall for non-default cases but challenges in accurately identifying default cases and slight overfitting.
- **LDA:** Reasonably good performance in predicting both non-default and default cases with consistent performance and no clear signs of overfitting or underfitting.

1.13 Conclusions and Recommendations Conclusions

Based on the 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 model's 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 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

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')
```

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
```

Based on the provided information:

Dataset Overview:

- The dataset contains 314 rows and 11 features.
- The features include stock information from various companies.

Data Types:

- There are 10 numerical features and 1 categorical feature.

Missing Values:

- There are no missing values present in the dataset.

Duplicates:

- 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

Based on the summary of stock prices for ten companies:

Average Stock Prices:

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

Standard Deviation:

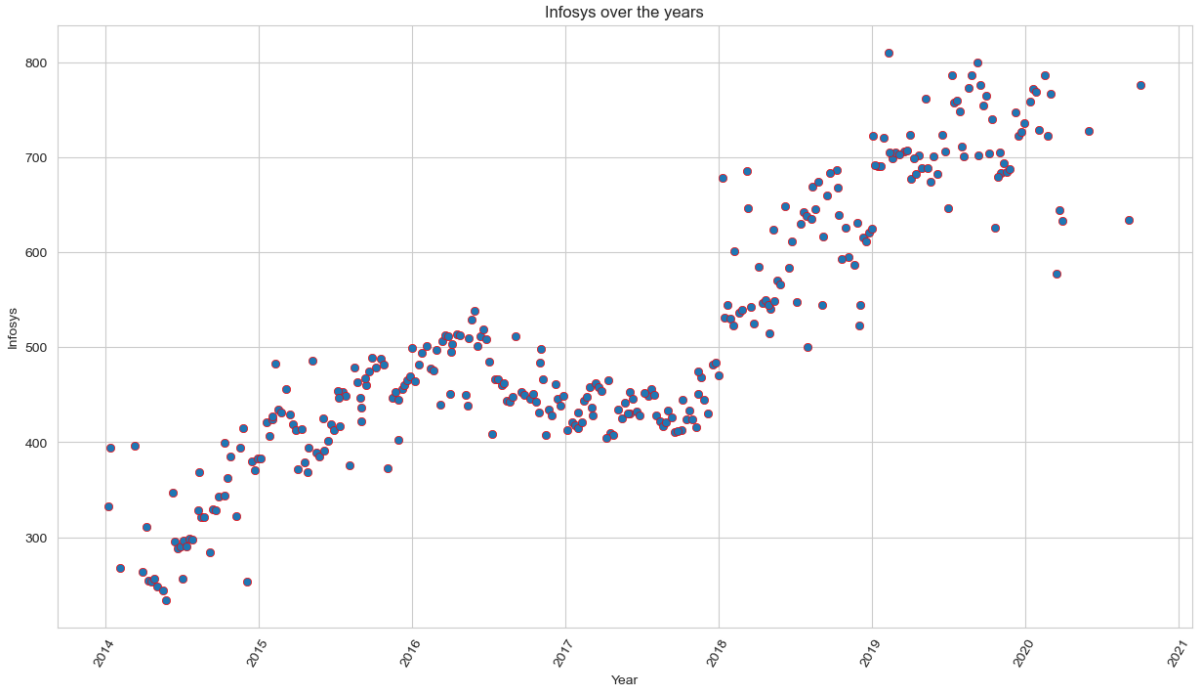
- The standard deviation indicates variability in stock prices, ranging from 135.95 for Infosys to 202.26 for Jet Airways.

Minimum and Maximum Stock Prices:

- The minimum stock price is 234 for Infosys, and the maximum stock price is 871 for Jet Airways.

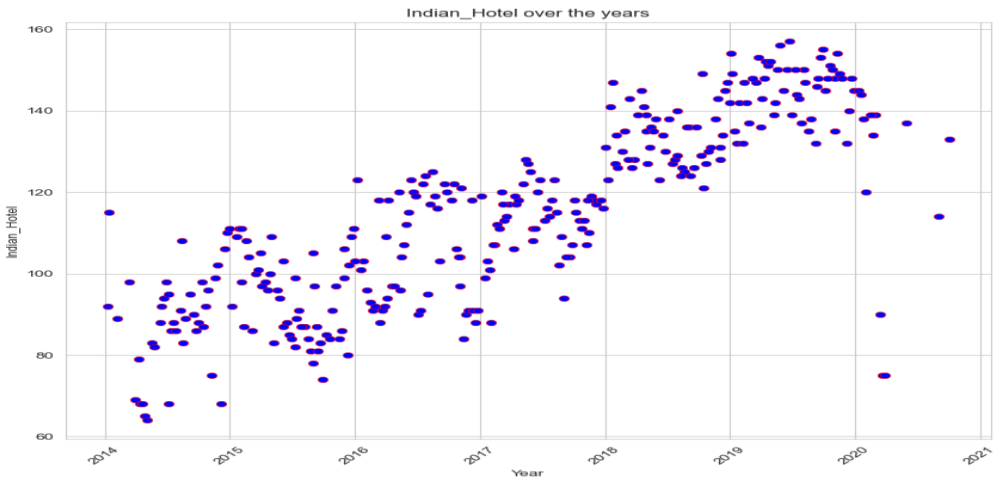
2.2 Draw Stock Price Graph (Stock Price vs Time) for any 2 given stocks with inference:

a. Infosys stock prices over the years



- The scatter plot illustrates the trend in Infosys stock prices over the observed years. It's evident from the plotted data points that there have been fluctuations in stock prices throughout this period.
- Between 2016 and 2018, Infosys experienced a steady increase in its stock value. However, after 2018, there was a decline in stock prices, reaching a low point. Subsequently, from 2019 onwards, the stock prices began to exhibit a gradual upward trend, showing consistent growth until 2021.

b. Indian Hotel stock prices over the years



The scatter plot displays the fluctuations in Indian Hotel stock prices over the years, showing noticeable variations until 2020. However, in 2020, there is a significant drop, likely attributed 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

The provided dataset comprises log returns of stock prices for ten different companies over a certain period, totalling 314 rows and 10 columns, each representing the returns for a specific company returns offers insights into the daily percentage changes in stock prices for each company. Negative log returns denote a decrease in stock prices, while positive log returns indicate an increase. It's worth noting that the log return for the first day is NaN, as there is no previous day's data available 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:

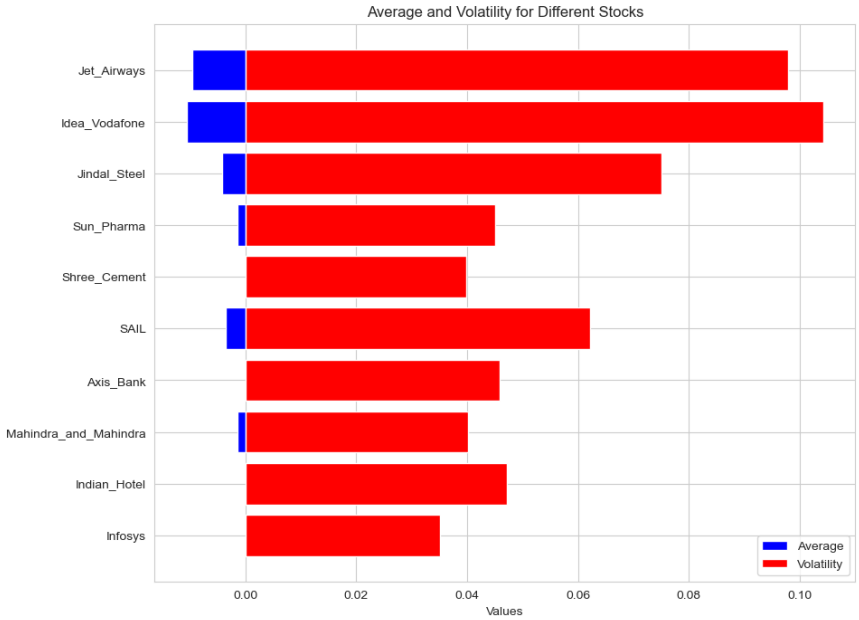
```
Infosys                   0.002794
Indian_Hotel              0.000266
Mahindra_and_Mahindra    -0.001506
Axis_Bank                 0.001167
SAIL                     -0.003463
Shree_Cement              0.003681
Sun_Pharma               -0.001455
Jindal_Steel             -0.004123
Idea_Vodafone   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

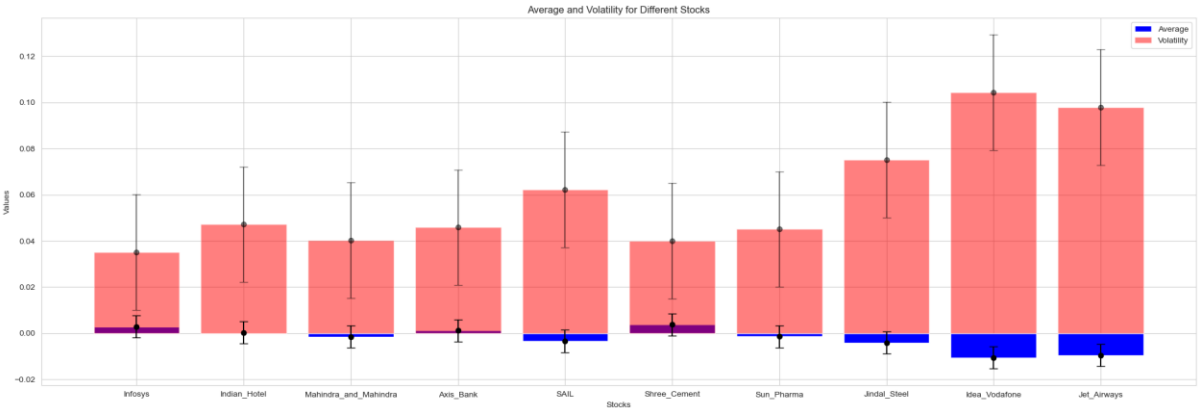
```
Infosys                   0.035070
Indian_Hotel              0.047130
Mahindra_and_Mahindra     0.040169
Axis_Bank                 0.045828
SAIL                      0.062188
Shree_Cement              0.039917
Sun_Pharma                0.045033
Jindal_Steel              0.075108
Idea_Vodafone             0.104315
Jet_Airways               0.097972
dtype: float64
```

The mean returns offer insights into the average daily performance of each stock. Some stocks exhibit positive returns, indicating growth, while others display negative returns, suggesting a decline in value. Meanwhile, the standard deviations reflect the level of volatility or risk associated with each stock. Stocks with higher standard deviations are prone to more significant price fluctuations, indicating greater volatility.

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

	Average	Volatility
Infosys	0.002794	0.035070
Indian_Hotel	0.000266	0.047130
Mahindra_and_Mahindra	-0.001506	0.040169
Axis_Bank	0.001167	0.045828
SAIL	-0.003463	0.062188
Shree_Cement	0.003681	0.039917
Sun_Pharma	-0.001455	0.045033
Jindal_Steel	-0.004123	0.075108
Idea_Vodafone	-0.010608	0.104315
Jet_Airways	-0.009548	0.097972





The two companies with the highest mean returns are Infosys and Shree Cement. Conversely, the two companies with the lowest mean returns are Idea Vodafone and Jet Airways. Among the companies with the highest mean returns, Shree Cement exhibits lower volatility compared to Infosys, indicating a relatively more stable investment choice. Similarly, among the companies with the lowest mean returns, Jet Airways displays slightly lower volatility than Idea Vodafone.

2.6 Conclusions and Recommendations

Conclusion:

Based on the analysis of stock data for different companies, several insights emerge. Infosys and Shree Cement emerge as companies with the highest mean returns, offering potentially lucrative investment opportunities. Conversely, Idea Vodafone and Jet Airways exhibit the lowest mean returns, warranting caution for potential investors.

Recommendations:

- Investors seeking higher profits should consider companies like Infosys and Shree Cement, which have consistently delivered higher returns. These firms present appealing choices for investors aiming to maximize their investment gains.
- For risk-averse investors, exercising caution when investing in companies like Idea Vodafone and Jet Airways, which demonstrate lower mean returns, is advisable.
- Market fluctuations are common in the short term, and holding onto investments for an extended period can help mitigate the impact of volatility and potentially yield higher returns.
- Regular monitoring of investments and staying informed about market trends are crucial for investors to make informed decisions and optimize their investment portfolios.