**Business Report**

**Project – FRA Project(Milestone-2)**

**Predicting Credit Risk for Company data**

**&**

**Predicting Market Risk**

**Created by Amit Jain**

**Note: This is in continuation for Milestone -1 Project , in this week we have analyzed our model based on Logistic Regression, LDA and Random forest. Also, we have solved one more problem related to Market Risk analysis**

Contents

[1. Credit Risk Dataset : Introduction 4](#_Toc120436130)

[Problem statement : 4](#_Toc120436131)

[1.1 Read the data as Dataframe in python and analyze the data. 5](#_Toc120436132)

[1.2 Fixing messy column names (containing spaces) for ease of use : 6](#_Toc120436133)

[1.3 Data dictionary : 7](#_Toc120436134)

[1.4 Create dependent variable: 10](#_Toc120436135)

[1.5 Data types of all variables: 11](#_Toc120436136)

[1.6 Dropping unnecessary columns: 12](#_Toc120436137)

[1.7 NULL Checks: 13](#_Toc120436138)

[1.8 Treat missing values: 15](#_Toc120436139)

[1.9 Outlier detection: 15](#_Toc120436140)

[1.10 Outlier Treatment: 16](#_Toc120436141)

[1.11 Univariate analysis: 17](#_Toc120436142)

[1.12 Distinct Value Counts of each field: 17](#_Toc120436143)

[1.13 Correlation heatmap 19](#_Toc120436144)

[1.14 Model building using stats model: 20](#_Toc120436145)

[Stats Model definitions: 20](#_Toc120436146)

[1.15 Partitioning the data into train and test: 21](#_Toc120436147)

[Start with the model 21](#_Toc120436148)

[1.16 Model 1 (With all columns ): 22](#_Toc120436149)

[1.17 Removing multicollinearity using VIF: 25](#_Toc120436150)

[1.18 Model 2 with VIF threshold 4: 26](#_Toc120436151)

[1.19 Model 3 with VIF threshold 4: 29](#_Toc120436152)

[1.20 Model 4 - Balance data using SMOTE and threshold VIF as 4: 34](#_Toc120436153)

[1.21 Perform Logistic Regression 37](#_Toc120436154)

[1.22 Building a Random Forest Classifier 39](#_Toc120436155)

[1.23 Perform LDA 43](#_Toc120436156)

[1.24 Compare the performances of Logistics, Radom Forest and LDA models (include ROC Curve) 46](#_Toc120436157)

[1.25 Conclusion and Recommendations from the above models: 48](#_Toc120436158)

[2. Market Risk Analysis: Introduction 50](#_Toc120436159)

[Problem statement : 50](#_Toc120436160)

[2.1 Data dictionary : 51](#_Toc120436161)

[2.2 Draw Stock Price Graph(Stock Price vs Time) for any 2 given stocks with inference 52](#_Toc120436162)

[2.3 Calculate Returns for all stocks with inference 55](#_Toc120436163)

[2.4 Means & Standard Deviations of these returns 57](#_Toc120436164)

[2.5 Draw a plot of Stock Means vs Standard Deviation and state your inference: 58](#_Toc120436165)

[2.6 Conclusion and Recommendations 59](#_Toc120436166)

**List of Figure**

[Figure 1 HeatMap 14](#_Toc120436167)

[Figure 2 BoxPlot for all data elements 15](#_Toc120436168)

[Figure 3 ScatterPlot 19](#_Toc120436169)

[Figure 4 Confusion Matrix and AUC Curve for Train data using Logit model 38](#_Toc120436170)

[Figure 5Confusion Matrix and AUC Curve for Test data using Logit model 39](#_Toc120436171)

[Figure 6 Feature importance graph for Random Forest 40](file:///C:\Users\Amit.Jain\Desktop\PGP-DSBA\11-Finance_and_Risk_analytics\Week-2\AmitJain_BusinessReport_25_Nov_2022.docx#_Toc120436172)

[Figure 7 Confusion Metrix and AUC Curve on Train data for Random forest 41](#_Toc120436173)

[Figure 8 Confusion Metrix and AUC Curve on Test data for Random forest 42](#_Toc120436174)

[Figure 9 Confusion Metrix and AUC Curve on Test data for LDA 44](#_Toc120436175)

[Figure 10 Confusion Metrix and AUC Curve on Train data for LDA 45](#_Toc120436176)

[Figure 11 AUC Curve for all the models on Train data 47](#_Toc120436177)

[Figure 12 AUC Curve for all the models on Test data 47](#_Toc120436178)

[Figure 13 Jet\_Airways Stock LinePlot 52](#_Toc120436179)

[Figure 14 Jet Airways Stock Scatter plot 53](#_Toc120436180)

[Figure 15 Infosys Stock Line Plot 54](#_Toc120436181)

[Figure 16 Infosys Stock Scatter Plot 54](#_Toc120436182)

[Figure 17Mean and Std Dvt for Stock returns 58](#_Toc120436183)

# **Credit Risk Dataset : Introduction**

This report explains the business requirements and provide the detailed solution based on the data provided for each problem statement. given in the assignment. Also, the purpose of this exercise is to execute Stats Model supervised learning techniques on the given data, combine all predictions and find out the model with best prediction or accuracy. In supervised learning techniques, there are clearly defined X and Y variables. Supervised Learning is used to predict either a continuous response (as in regression) or a categorical response (as in classification). These are machine learning models for combining predictions from multiple separate models. Both regression and classification can be done using Ensemble Learning. Combining all the individual predictions can be done using either voting or averaging.

# **Problem statement :**

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

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

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

**Dataset for Problem 1: Company\_Data2015-1.xlsx**

To understand the problem, for Credit risk checking company has given sample of 3586 Company records collected data in the Company\_Data2015-1.xlsx , which have financial information of the companies for previous year 3025 and we need to predict Company status based on next Year Networth.

**Dependent variable -** We need to create a default variable that should take the value of 1 when net worth next year is negative & 0 when net worth next year is positive.

**Assumption:**

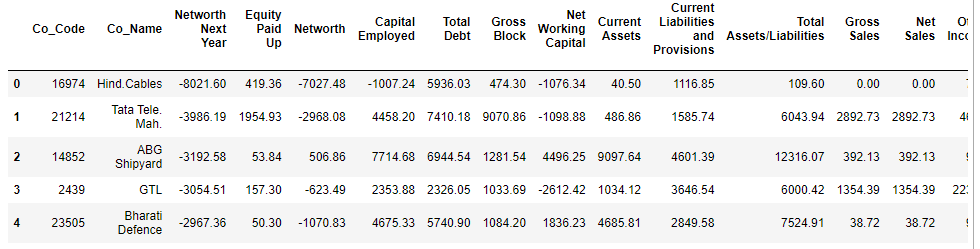
Assume that the data follows a normal distribution. In reality, the normality assumption may not always hold if the sample size is small.

Step of understanding the data:

Import the data: Imported the data using Python notebooks and analyzed the effects of Education and Occupations over salary field.

# **Read the data as Dataframe in python and analyze the data.**

This is how the data look like:



Shape of the data:

The number of rows (observations) is 3586

The number of columns (variables) is 67

Insights:

1. There are only 3586 Rows in sample, data , which are Companies previous year financial statements
2. We have given total 67 different fields for the data, so good amount of observation points.
3. Column names of the data:

'Co\_Code', 'Co\_Name', 'Networth Next Year', 'Equity Paid Up',, 'Networth', 'Capital Employed', 'Total Debt', 'Gross Block ',, 'Net Working Capital ', 'Current Assets ',, 'Current Liabilities and Provisions ', 'Total Assets/Liabilities ',, 'Gross Sales', 'Net Sales', 'Other Income', 'Value Of Output',, 'Cost of Production', 'Selling Cost', 'PBIDT', 'PBDT', 'PBIT', 'PBT',, 'PAT', 'Adjusted PAT', 'CP', 'Revenue earnings in forex',, 'Revenue expenses in forex', 'Capital expenses in forex',, 'Book Value (Unit Curr)', 'Book Value (Adj.) (Unit Curr)',, 'Market Capitalisation', 'CEPS (annualised) (Unit Curr)',, 'Cash Flow From Operating Activities',, 'Cash Flow From Investing Activities',, 'Cash Flow From Financing Activities', 'ROG-Net Worth (%)',, 'ROG-Capital Employed (%)', 'ROG-Gross Block (%)',, 'ROG-Gross Sales (%)', 'ROG-Net Sales (%)',, 'ROG-Cost of Production (%)', 'ROG-Total Assets (%)', 'ROG-PBIDT (%)',, 'ROG-PBDT (%)', 'ROG-PBIT (%)', 'ROG-PBT (%)', 'ROG-PAT (%)',, 'ROG-CP (%)', 'ROG-Revenue earnings in forex (%)',, 'ROG-Revenue expenses in forex (%)', 'ROG-Market Capitalisation (%)',, 'Current Ratio[Latest]', 'Fixed Assets Ratio[Latest]',, 'Inventory Ratio[Latest]', 'Debtors Ratio[Latest]',, 'Total Asset Turnover Ratio[Latest]', 'Interest Cover Ratio[Latest]',, 'PBIDTM (%)[Latest]', 'PBITM (%)[Latest]', 'PBDTM (%)[Latest]',, 'CPM (%)[Latest]', 'APATM (%)[Latest]', 'Debtors Velocity (Days)',, 'Creditors Velocity (Days)', 'Inventory Velocity (Days)',, 'Value of Output/Total Assets', 'Value of Output/Gross Block'

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

We can also observe that above listed column names have multiple Special characters in it , example [, (, ], ), %, - white space etc.

In order to start analyzing them in out Python tool, we need to clear this clutter and modify Column names only.

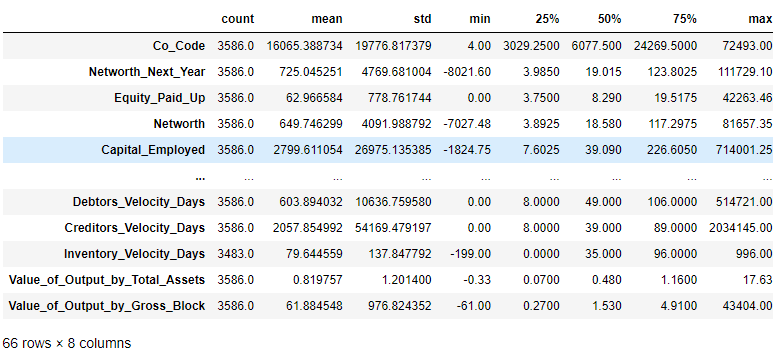
After fixing column names, this is how the Column names look like:

'Co\_Code', 'Co\_Name', 'Networth\_Next\_Year', 'Equity\_Paid\_Up', 'Networth', 'Capital\_Employed', 'Total\_Debt', 'Gross\_Block', 'Net\_Working\_Capital', 'Current\_Assets', 'Current\_Liabilities\_and\_Provisions', 'Total\_Assets\_by\_Liabilities', 'Gross\_Sales', 'Net\_Sales', 'Other\_Income', 'Value\_Of\_Output', 'Cost\_of\_Production', 'Selling\_Cost', 'PBIDT', 'PBDT', 'PBIT', 'PBT', 'PAT', 'Adjusted\_PAT', 'CP', 'Revenue\_earnings\_in\_forex', 'Revenue\_expenses\_in\_forex', 'Capital\_expenses\_in\_forex', 'Book\_Value\_Unit\_Curr', 'Book\_Value\_Adj\_Unit\_Curr', 'Market\_Capitalisation', 'CEPS\_annualised\_Unit\_Curr', 'Cash\_Flow\_From\_Operating\_Activities', 'Cash\_Flow\_From\_Investing\_Activities', 'Cash\_Flow\_From\_Financing\_Activities', 'ROG\_Net\_Worth\_perc', 'ROG\_Capital\_Employed\_perc', 'ROG\_Gross\_Block\_perc', 'ROG\_Gross\_Sales\_perc', 'ROG\_Net\_Sales\_perc', 'ROG\_Cost\_of\_Production\_perc', 'ROG\_Total\_Assets\_perc', 'ROG\_PBIDT\_perc', 'ROG\_PBDT\_perc', 'ROG\_PBIT\_perc', 'ROG\_PBT\_perc', 'ROG\_PAT\_perc', 'ROG\_CP\_perc', 'ROG\_Revenue\_earnings\_in\_forex\_perc', 'ROG\_Revenue\_expenses\_in\_forex\_perc', 'ROG\_Market\_Capitalisation\_perc', 'Current\_RatioLatest', 'Fixed\_Assets\_RatioLatest', 'Inventory\_RatioLatest', 'Debtors\_RatioLatest', 'Total\_Asset\_Turnover\_RatioLatest', 'Interest\_Cover\_RatioLatest', 'PBIDTM\_percLatest', 'PBITM\_percLatest', 'PBDTM\_percLatest', 'CPM\_percLatest', 'APATM\_percLatest', 'Debtors\_Velocity\_Days', 'Creditors\_Velocity\_Days', 'Inventory\_Velocity\_Days', 'Value\_of\_Output\_by\_Total\_Assets', 'Value\_of\_Output\_by\_Gross\_Block'

# **Data dictionary :**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Field Name** | **Description** | **New Field Name** |
| 1 | Co\_Code | Company Code | Co\_Code |
| 2 | Co\_Name | Company Name | Co\_Name |
| 3 | Networth Next Year | Value of a company as on 2016 – Next Year(difference between the value of total assets and total liabilities) | Networth\_Next\_Year |
| 4 | Equity Paid Up | Amount that has been received by the company through the issue of shares to the shareholders | Equity\_Paid\_Up |
| 5 | Networth | Value of a company as on 2015 – Current Year | Networth |
| 6 | Capital Employed | Total amount of capital used for the acquisition of profits by a company | Capital\_Employed |
| 7 | Total Debt | The sum of money borrowed by the company and is due to be paid | Total\_Debt |
| 8 | Gross Block | Total value of all of the assets that a company owns | Gross\_Block |
| 9 | Net Working Capital | The difference between a company’s current assets (cash, accounts receivable, inventories of raw materials and finished goods) and its current liabilities (accounts payable). | Net\_Working\_Capital |
| 10 | Current Assets | All the assets of a company that are expected to be sold or used as a result of standard business operations over the next year. | Curr\_Assets |
| 11 | Current Liabilities and Provisions | Short-term financial obligations that are due within one year (includes amount that is set aside cover a future liability) | Curr\_Liab\_and\_Prov |
| 12 | Total Assets/Liabilities | Ratio of total assets to liabailities of the company | Total\_Assets\_to\_Liab |
| 13 | Gross Sales | The grand total of sale transactions within the accounting period | Gross\_Sales |
| 14 | Net Sales | Gross sales minus returns, allowances, and discounts | Net\_Sales |
| 15 | Other Income | Income realized from non-business activities (e.g. sale of long term asset) | Other\_Income |
| 16 | Value Of Output | Product of physical output of goods and services produced by company and its market price | Value\_Of\_Output |
| 17 | Cost of Production | Costs incurred by a business from manufacturing a product or providing a service | Cost\_of\_Prod |
| 18 | Selling Cost | Costs which are made to create the demand for the product (advertising expenditures, packaging and styling, salaries, commissions and travelling expenses of sales personnel, and the cost of shops and showrooms) | Selling\_Cost |
| 19 | PBIDT | Profit Before Interest, Depreciation & Taxes | PBIDT |
| 20 | PBDT | Profit Before Depreciation and Tax | PBDT |
| 21 | PBIT | Profit before interest and taxes | PBIT |
| 22 | PBT | Profit before tax | PBT |
| 23 | PAT | Profit After Tax | PAT |
| 24 | Adjusted PAT | Adjusted profit is the best estimate of the true profit | Adjusted\_PAT |
| 26 | CP | Commercial paper , a short-term debt instrument to meet short-term liabilities. | CP |
| 27 | Revenue earnings in forex | Revenue earned in foreign currency | Rev\_earn\_in\_forex |
| 28 | Revenue expenses in forex | Expenses due to foreign currency transactions | Rev\_exp\_in\_forex |
| 29 | Capital expenses in forex | Long term investment in forex | Capital\_exp\_in\_forex |
| 30 | Book Value (Unit Curr) | Net asset value | Book\_Value\_Unit\_Curr |
| 31 | Book Value (Adj.) (Unit Curr) | Book value adjusted to reflect asset’s true fair market value | Book\_Value\_Adj\_Unit\_Curr |
| 32 | Market Capitalisation | Product of the total number of a company’s outstanding shares and the current market price of one share | Market\_Capitalisation |
| 33 | CEPS (nnualized) (Unit Curr) | Cash Earnings per Share, profitability ratio that measures the financial performance of a company by calculating cash flows on a per share basis | CEPS\_annualised\_Unit\_Curr |
| 34 | Cash Flow From Operating Activities | Use of cash from ongoing regular business activities | Cash\_Flow\_From\_Opr |
| 35 | Cash Flow From Investing Activities | Cash used in the purchase of non-current assets–or long-term assets– that will deliver value in the future | Cash\_Flow\_From\_Inv |
| 36 | Cash Flow From Financing Activities | Net flows of cash that are used to fund the company (transactions involving debt, equity, and dividends) | Cash\_Flow\_From\_Fin |
| 37 | ROG-Net Worth (%) | Rate of Growth – Networth | ROG\_Net\_Worth\_perc |
| 38 | ROG-Capital Employed (%) | Rate of Growth – Capital Employed | ROG\_Capital\_Employed\_perc |
| 39 | ROG-Gross Block (%) | Rate of Growth – Gross Block | ROG\_Gross\_Block\_perc |
| 40 | ROG-Gross Sales (%) | Rate of Growth – Gross Sales | ROG\_Gross\_Sales\_perc |
| 41 | ROG-Net Sales (%) | Rate of Growth – Net Sales | ROG\_Net\_Sales\_perc |
| 42 | ROG-Cost of Production (%) | Rate of Growth - Cost of Production | ROG\_Cost\_of\_Prod\_perc |
| 43 | ROG-Total Assets (%) | Rate of Growth – Total Assets | ROG\_Total\_Assets\_perc |
| 44 | ROG-PBIDT (%) | Rate of Growth- PBIDT | ROG\_PBIDT\_perc |
| 45 | ROG-PBDT (%) | Rate of Growth- PBDT | ROG\_PBDT\_perc |
| 46 | ROG-PBIT (%) | Rate of Growth- PBIT | ROG\_PBIT\_perc |
| 47 | ROG-PBT (%) | Rate of Growth- PBT | ROG\_PBT\_perc |
| 48 | ROG-PAT (%) | Rate of Growth- PAT | ROG\_PAT\_perc |
| 49 | ROG-CP (%) | Rate of Growth- CP | ROG\_CP\_perc |
| 50 | ROG-Revenue earnings in forex (%) | Rate of Growth - Revenue earnings in forex | ROG\_Rev\_earn\_in\_forex\_perc |
| 51 | ROG-Revenue expenses in forex (%) | Rate of Growth - Revenue expenses in forex | ROG\_Rev\_exp\_in\_forex\_perc |
| 52 | ROG-Market Capitalisation (%) | Rate of Growth – Market Capitalisation | ROG\_Market\_Capitalisation\_perc |
| 53 | Current Ratio[Latest] | Liquidity ratio, company’s ability to pay short-term obligations or those due within one year | Curr\_Ratio\_Latest |
| 54 | Fixed Assets Ratio[Latest] | Solvency ratio, the capacity of a company to discharge its obligations towards long-term lenders indicating | Fixed\_Assets\_Ratio\_Latest |
| 55 | Inventory Ratio[Latest] | Activity ratio, specifies the number of times the stock or inventory has been replaced and sold by the company | Inventory\_Ratio\_Latest |
| 56 | Debtors Ratio[Latest] | Measures how quickly cash debtors are paying back to the company | Debtors\_Ratio\_Latest |
| 57 | Total Asset Turnover Ratio[Latest] | The value of a company’s revenues relative to the value of its assets | Total\_Asset\_Turnover\_Ratio\_Latest |
| 58 | Interest Cover Ratio[Latest] | Determines how easily a company can pay interest on its outstanding debt | Interest\_Cover\_Ratio\_Latest |
| 59 | PBIDTM (%)[Latest] | Profit before Interest Depreciation and Tax Margin | PBIDTM\_perc\_Latest |
| 60 | PBITM (%)[Latest] | Profit Before Interest Tax Margin | PBITM\_perc\_Latest |
| 61 | PBDTM (%)[Latest] | Profit Before Depreciation Tax Margin | PBDTM\_perc\_Latest |
| 62 | CPM (%)[Latest] | Cost per thousand (advertising cost) | CPM\_perc\_Latest |
| 63 | APATM (%)[Latest] | After tax profit margin | APATM\_perc\_Latest |
| 64 | Debtors Velocity (Days) | Average days required for receiving the payments | Debtors\_Vel\_Days |
| 65 | Creditors Velocity (Days) | Average number of days company takes to pay suppliers | Creditors\_Vel\_Days |
| 66 | Inventory Velocity (Days) | Average number of days the company needs to turn its inventory into sales | Inventory\_Vel\_Days |
| 67 | Value of Output/Total Assets | Ratio of Value of Output (market value) to Total Assets | Value\_of\_Output\_to\_Total\_Assets |
| 68 | Value of Output/Gross Block | Ratio of Value of Output (market value) to Gross Block | Value\_of\_Output\_to\_Gross\_Block |

**Description of the data elements:**



Insights:

1. We can clearly see that data have different range of Min, max and Median.
2. We can Scale the data, based on our what prediction method we are using.

# **Create dependent variable:**

Dependent variable – We need to create a default variable that should take the value of 1 when net worth next year is negative & 0 when net worth next year is positive.

Lets check for the proportion of the data:

1. 3198
2. 388

Now lets check for the summary of the “default” target data :

count 3586.000000

mean 0.108199

std 0.310674

min 0.000000

25% 0.000000

50% 0.000000

75% 0.000000

max 1.000000

Name: default, dtype: float64

Insights:

1. Target variable is 1, which denotes the Default companies
2. Value 0 denotes, non default company
3. Default companies data is very less as compare to Non-default company.
4. Ratio of the target variable is 10.8%

# **Data types of all variables:**

Data columns (total 68 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Co\_Code 3586 non-null int64

1 Co\_Name 3586 non-null object

2 Networth\_Next\_Year 3586 non-null float64

3 Equity\_Paid\_Up 3586 non-null float64

4 Networth 3586 non-null float64

5 Capital\_Employed 3586 non-null float64

6 Total\_Debt 3586 non-null float64

7 Gross\_Block 3586 non-null float64

8 Net\_Working\_Capital 3586 non-null float64

9 Current\_Assets 3586 non-null float64

10 Current\_Liabilities\_and\_Provisions 3586 non-null float64

11 Total\_Assets\_by\_Liabilities 3586 non-null float64

12 Gross\_Sales 3586 non-null float64

13 Net\_Sales 3586 non-null float64

14 Other\_Income 3586 non-null float64

15 Value\_Of\_Output 3586 non-null float64

16 Cost\_of\_Production 3586 non-null float64

17 Selling\_Cost 3586 non-null float64

18 PBIDT 3586 non-null float64

19 PBDT 3586 non-null float64

20 PBIT 3586 non-null float64

21 PBT 3586 non-null float64

22 PAT 3586 non-null float64

23 Adjusted\_PAT 3586 non-null float64

24 CP 3586 non-null float64

25 Revenue\_earnings\_in\_forex 3586 non-null float64

26 Revenue\_expenses\_in\_forex 3586 non-null float64

27 Capital\_expenses\_in\_forex 3586 non-null float64

28 Book\_Value\_Unit\_Curr 3586 non-null float64

29 Book\_Value\_Adj\_Unit\_Curr 3582 non-null float64

30 Market\_Capitalisation 3586 non-null float64

31 CEPS\_annualised\_Unit\_Curr 3586 non-null float64

32 Cash\_Flow\_From\_Operating\_Activities 3586 non-null float64

33 Cash\_Flow\_From\_Investing\_Activities 3586 non-null float64

34 Cash\_Flow\_From\_Financing\_Activities 3586 non-null float64

35 ROG\_Net\_Worth\_perc 3586 non-null float64

36 ROG\_Capital\_Employed\_perc 3586 non-null float64

37 ROG\_Gross\_Block\_perc 3586 non-null float64

38 ROG\_Gross\_Sales\_perc 3586 non-null float64

39 ROG\_Net\_Sales\_perc 3586 non-null float64

40 ROG\_Cost\_of\_Production\_perc 3586 non-null float64

41 ROG\_Total\_Assets\_perc 3586 non-null float64

42 ROG\_PBIDT\_perc 3586 non-null float64

43 ROG\_PBDT\_perc 3586 non-null float64

44 ROG\_PBIT\_perc 3586 non-null float64

45 ROG\_PBT\_perc 3586 non-null float64

46 ROG\_PAT\_perc 3586 non-null float64

47 ROG\_CP\_perc 3586 non-null float64

48 ROG\_Revenue\_earnings\_in\_forex\_perc 3586 non-null float64

49 ROG\_Revenue\_expenses\_in\_forex\_perc 3586 non-null float64

50 ROG\_Market\_Capitalisation\_perc 3586 non-null float64

51 Current\_RatioLatest 3585 non-null float64

52 Fixed\_Assets\_RatioLatest 3585 non-null float64

53 Inventory\_RatioLatest 3585 non-null float64

54 Debtors\_RatioLatest 3585 non-null float64

55 Total\_Asset\_Turnover\_RatioLatest 3585 non-null float64

56 Interest\_Cover\_RatioLatest 3585 non-null float64

57 PBIDTM\_percLatest 3585 non-null float64

58 PBITM\_percLatest 3585 non-null float64

59 PBDTM\_percLatest 3585 non-null float64

60 CPM\_percLatest 3585 non-null float64

61 APATM\_percLatest 3585 non-null float64

62 Debtors\_Velocity\_Days 3586 non-null int64

63 Creditors\_Velocity\_Days 3586 non-null int64

64 Inventory\_Velocity\_Days 3483 non-null float64

65 Value\_of\_Output\_by\_Total\_Assets 3586 non-null float64

66 Value\_of\_Output\_by\_Gross\_Block 3586 non-null float64

67 default 3586 non-null int32

dtypes: float64(63), int32(1), int64(3), object(1)

Insights:

1. All of the data is in Numeric Format, except Co\_Code and Co\_Name, and these 2 fields are not required , so looks good to me.

# **Dropping unnecessary columns:**

Drop the fields CO\_Code and CO\_name , since these are not required for our model. Also Drop column Networth\_Next\_Year, because we used this field to build the Dependent Field "default"

# **NULL Checks:**

Equity\_Paid\_Up 0

Networth 0

Capital\_Employed 0

Total\_Debt 0

Gross\_Block 0

...

Creditors\_Velocity\_Days 0

Inventory\_Velocity\_Days 103

Value\_of\_Output\_by\_Total\_Assets 0

Value\_of\_Output\_by\_Gross\_Block 0

default 0

Insights:

1. We checked that data have NULL values in all of these positional columns:

(array([26, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 61], dtype=int64),)

1. Total NULL values in data set are : 103
2. Let’s check how many percentage values are missing in each columns.

if there are more than 30% of the values are missing, we will drop them

Inventory\_Velocity\_Days 0.028723

Book\_Value\_Adj\_Unit\_Curr 0.001115

Total\_Asset\_Turnover\_RatioLatest 0.000279

CPM\_percLatest 0.000279

PBDTM\_percLatest 0.000279

...

ROG\_Net\_Worth\_perc 0.000000

Networth 0.000000

PBIDT 0.000000

Capital\_Employed 0.000000

default 0.000000

No Column have values more than 30%, so we are good in this case, and no need to drop any field because of Missing values. Let's visually inspect the missing values in our data



Figure 1 HeatMap

Insights:

1. We can see that data have NULL values in a few columns only, and that is spotted in white dashes in the data . This data is too less. And we can impute them
2. We can see that maximum percentage of missing are 2.8% in Column **Inventory\_Velocity\_Days**

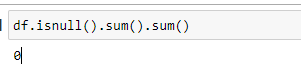
# **Treat missing values:**

There are many ways to Treat Missing values.

1. Either Replace them with Median
2. Impute them with any imputer

We will use both methods and analyze them separately . First of all we will impute missing values with Median value of that column:

Replace NULL with Median and this is how the data look like :



# **Outlier detection:**

Plot all data n box plot:

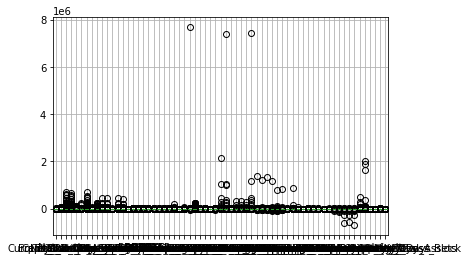
****

Figure 2 BoxPlot for all data elements

**Insights:**

1. **Also individual box plot can not be draw in this document, as there are 65 box plot to draw, which wont fit into page.**
2. **But we can clearly see some bubbles in the above graph , which is depicting outliers in the data**

# **Outlier Treatment:**

**We have created a function for checking upper and lower limit of the data and we will use that function for calculating, total no of records in each field, which are beyond this range.**

**List the total no of values in each column, which are above upper limit or less than lower limit of that column.**

Equity\_Paid\_Up 905

Networth 1342

Capital\_Employed 1220

Total\_Debt 1247

Gross\_Block 1236

...

Debtors\_Velocity\_Days 738

Creditors\_Velocity\_Days 770

Inventory\_Velocity\_Days 719

Value\_of\_Output\_by\_Total\_Assets 633

Value\_of\_Output\_by\_Gross\_Block 984

Length: 64, dtype: int64

**Insights:**

1. **There are total of 76984 records, which have values beyond upper and lower range of the value in that field.**
2. **There are many ways we can Treat Outliers ,** 
   1. **Either we can replace Higher values to Upper Limit or very less value to Lower limit of outliers**
   2. **Or we can impute them to K Nearest neighbor Method.**
   3. **In our Case, we dont know Company Segmentations, and Companies can have different levels of Revenue, Profits and all can be Valid at same time. So we should Treat them with K nearest neighbor method , and Treat them same as we did for Missing values.**
   4. **Still I will be using both method for KNN as well as replacing higher values with Upper/lower values**

**First of all, we would replace the outlier data with upper and lower value.**

**Again , since data fields are too large, we can display the Box plot in the Business report.**

# **Univariate analysis:**

We will analyze this data with Distinct Counts, Distribution of the data and box plots with respect to dependent variables , its co-relations, Skewness , unique values, etc. We will use different methods plots for understanding this information.

# **Distinct Value Counts of each field:**

Equity\_Paid\_Up:1586

Networth:2334

Capital\_Employed:2512

Total\_Debt:1557

Gross\_Block:1877

Net\_Working\_Capital:2078

Current\_Assets:2199

Current\_Liabilities\_and\_Provisions:1698

Total\_Assets\_by\_Liabilities:2565

Gross\_Sales:2081

Net\_Sales:2079

Other\_Income:545

Value\_Of\_Output:2097

Cost\_of\_Production:1975

Selling\_Cost:531

PBIDT:1418

PBDT:1179

PBIT:1327

PBT:969

PAT:885

Adjusted\_PAT:883

CP:1129

Revenue\_earnings\_in\_forex:431

Revenue\_expenses\_in\_forex:528

Capital\_expenses\_in\_forex:1

Book\_Value\_Unit\_Curr:2540

Book\_Value\_Adj\_Unit\_Curr:2483

Market\_Capitalisation:1469

CEPS\_annualised\_Unit\_Curr:1315

Cash\_Flow\_From\_Operating\_Activities:1340

Cash\_Flow\_From\_Investing\_Activities:922

Cash\_Flow\_From\_Financing\_Activities:989

ROG\_Net\_Worth\_perc:1725

ROG\_Capital\_Employed\_perc:1972

ROG\_Gross\_Block\_perc:1103

ROG\_Gross\_Sales\_perc:2084

ROG\_Net\_Sales\_perc:2085

ROG\_Cost\_of\_Production\_perc:2069

ROG\_Total\_Assets\_perc:2107

ROG\_PBIDT\_perc:2208

ROG\_PBDT\_perc:2201

ROG\_PBIT\_perc:2216

ROG\_PBT\_perc:2150

ROG\_PAT\_perc:2097

ROG\_CP\_perc:2175

ROG\_Revenue\_earnings\_in\_forex\_perc:1

ROG\_Revenue\_expenses\_in\_forex\_perc:1

ROG\_Market\_Capitalisation\_perc:1600

Current\_RatioLatest:455

Fixed\_Assets\_RatioLatest:741

Inventory\_RatioLatest:1134

Debtors\_RatioLatest:1161

Total\_Asset\_Turnover\_RatioLatest:352

Interest\_Cover\_RatioLatest:776

PBIDTM\_percLatest:1835

PBITM\_percLatest:1654

PBDTM\_percLatest:1707

CPM\_percLatest:1551

APATM\_percLatest:1268

Debtors\_Velocity\_Days:241

Creditors\_Velocity\_Days:203

Inventory\_Velocity\_Days:225

Value\_of\_Output\_by\_Total\_Assets:283

Value\_of\_Output\_by\_Gross\_Block:764

default:2

insights:

1. These fields have distinct count of value as only 1

Column name is Capital\_expenses\_in\_forex and its unique value count is : 1

Column name is ROG\_Revenue\_earnings\_in\_forex\_perc and its unique value count is : 1

Column name is ROG\_Revenue\_expenses\_in\_forex\_perc and its unique value count is : 1

1. Since these columns have only 1 value in it, we can not use it for our Predictions. So we should Drop these fields, as they are not going to contribute anything in Target variable prediction.
2. Individual Box plot and distribution of the data . Please refer Notebook for the diagram.
3. Data Skewness :

Equity\_Paid\_Up 1.141900

Networth 0.903328

Capital\_Employed 1.137131

Total\_Debt 1.197970

Gross\_Block 1.228742

...

Creditors\_Velocity\_Days 1.143302

Inventory\_Velocity\_Days 1.206446

Value\_of\_Output\_by\_Total\_Assets 1.110709

Value\_of\_Output\_by\_Gross\_Block 1.183618

default 2.523672

1. Data is not normally distributed and it is Right skewed and left skewed in both directions for many fields.
2. Checking proportion of default

0 0.891801

1 0.108199

# **Correlation heatmap**

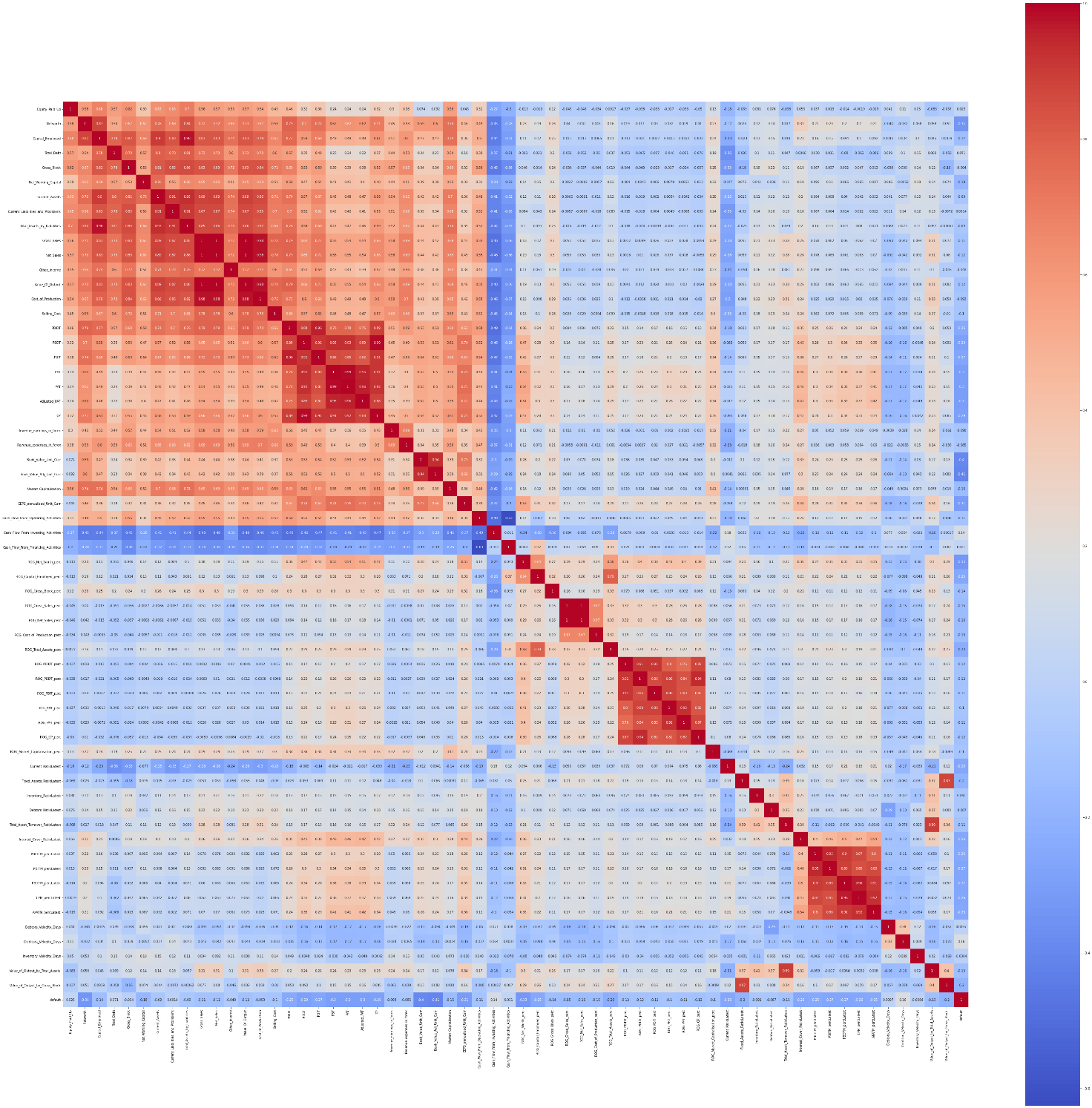


Figure 3 ScatterPlot

Insights:

1. We don’t need to look at the Score of co-relation for each column, we can just look at the color coding and Dark Red and Dark Blue showing Strong Positive and negative co-relations, among fields. We can clearly see that there are many fields, which can be avoided for analyzing and building prediction models, because they have co-relation between them,

# **Model building using stats model:**

# **Stats Model definitions:**

We will use statsmodels module to implement Ordinary Least Squares(OLS) method of linear regression for predicting “Default “ Companies based on the data provided for us.

Introduction : 

A linear regression model establishes the relation between a dependent variable(y) and at least one independent variable(x) as:



In OLS method, we have to choose the values of bi and b0 such that, the total sum of squares of the difference between the calculated and observed values of y, is minimized.   
Formula for OLS:



Where,   
= predicted value for the ith observation   
= actual value for the ith observation   
= error/residual for the ith observation   
n = total number of observations  
To get the values of  b0 and b1 which minimize S, we can take a partial derivative for each coefficient and equate it to zero.

**Approach :**

First of all we define the variables **x** and **y**. In the example below, the variables are read from a csv file using pandas.   
 Next, We need to add the constant b0 to the equation using the **add\_constant()** method.

The OLS() function of the statsmodels.api module is used to perform OLS regression. It returns an OLS object. Then fit() method is called on this object for fitting the regression line to the data.

The summary() method is used to obtain a table which gives an extensive description about the regression results

# **Partitioning the data into train and test:**

We divided the data into train and Test data set. With given parameters:

test\_size = 0.33, random\_state = 42, stratify = y

**Test Size specify**, what should be the size of train and test data set, in our case, train set will have 66 % of data and test will have about 33% of data elements

**Random state** given as 42.

Check the shape of the data :

Train size: (2402, 61)

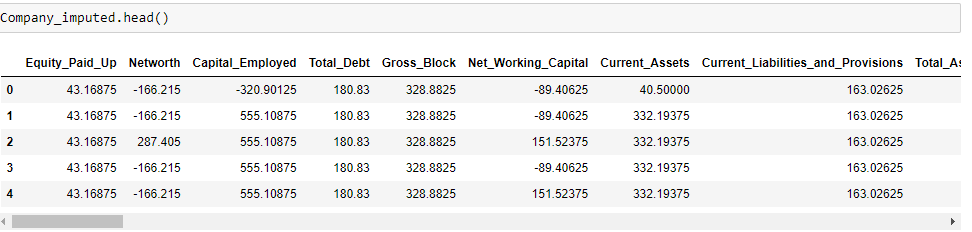
Test Size : (1184, 61)

**Why stratify = y?**

Please note, because this data is highly imbalanced and could possibly result into different proportions in the y variable between train and test set.

# **Start with the model**

This is how the final data looks like :



Once the data is split into 4 parts : X\_train, X\_test, y\_train, y\_test

We need to Concatenate train data (X\_train, y\_train) with name as TRAIN and test data (X\_test, y\_test) as TEST separately, because Stats model work on Complete data, it does need Separation of independent variables into 2 form of train and test.

After concatenation of data, we have got these list of fields in Train data , which are all fields and all rows of independent variables:

Index(['Equity\_Paid\_Up', 'Networth', 'Capital\_Employed', 'Total\_Debt',

'Gross\_Block', 'Net\_Working\_Capital', 'Current\_Assets',

'Current\_Liabilities\_and\_Provisions', 'Total\_Assets\_by\_Liabilities',

'Gross\_Sales', 'Net\_Sales', 'Other\_Income', 'Value\_Of\_Output',

'Cost\_of\_Production', 'Selling\_Cost', 'PBIDT', 'PBDT', 'PBIT', 'PBT',

'PAT', 'Adjusted\_PAT', 'CP', 'Revenue\_earnings\_in\_forex',

'Revenue\_expenses\_in\_forex', 'Book\_Value\_Unit\_Curr',

'Book\_Value\_Adj\_Unit\_Curr', 'Market\_Capitalisation',

'CEPS\_annualised\_Unit\_Curr', 'Cash\_Flow\_From\_Operating\_Activities',

'Cash\_Flow\_From\_Investing\_Activities',

'Cash\_Flow\_From\_Financing\_Activities', 'ROG\_Net\_Worth\_perc',

'ROG\_Capital\_Employed\_perc', 'ROG\_Gross\_Block\_perc',

'ROG\_Gross\_Sales\_perc', 'ROG\_Net\_Sales\_perc',

'ROG\_Cost\_of\_Production\_perc', 'ROG\_Total\_Assets\_perc',

'ROG\_PBIDT\_perc', 'ROG\_PBDT\_perc', 'ROG\_PBIT\_perc', 'ROG\_PBT\_perc',

'ROG\_PAT\_perc', 'ROG\_CP\_perc', 'ROG\_Market\_Capitalisation\_perc',

'Current\_RatioLatest', 'Fixed\_Assets\_RatioLatest',

'Inventory\_RatioLatest', 'Debtors\_RatioLatest',

'Total\_Asset\_Turnover\_RatioLatest', 'Interest\_Cover\_RatioLatest',

'PBIDTM\_percLatest', 'PBITM\_percLatest', 'PBDTM\_percLatest',

'CPM\_percLatest', 'APATM\_percLatest', 'Debtors\_Velocity\_Days',

'Creditors\_Velocity\_Days', 'Inventory\_Velocity\_Days',

'Value\_of\_Output\_by\_Total\_Assets', 'Value\_of\_Output\_by\_Gross\_Block',

'default'],

And similar fields will be part of Test data as well.

# **Model 1 (With all columns ):**

We have used all of the fields listed above and build our first model.

And this is the final result for our first model:

|  |  |  |  |
| --- | --- | --- | --- |
| Logit Regression Results | | | |
| **Dep. Variable:** | default | **No. Observations:** | 2402 |
| **Model:** | Logit | **Df Residuals:** | 2340 |
| **Method:** | MLE | **Df Model:** | 61 |
| **Date:** | Tue, 15 Nov 2022 | **Pseudo R-squ.:** | 0.6745 |
| **Time:** | 18:07:50 | **Log-Likelihood:** | -268.02 |
| **converged:** | True | **LL-Null:** | -823.47 |
| **Covariance Type:** | nonrobust | **LLR p-value:** | 1.198e-192 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **coef** | **std err** | **z** | **P>|z|** | **[0.025** | **0.975]** |
| **Intercept** | -0.1885 | 0.258 | -0.732 | 0.464 | -0.693 | 0.316 |
| **Equity\_Paid\_Up** | -0.0079 | 0.014 | -0.585 | 0.559 | -0.035 | 0.019 |
| **Networth** | -0.0056 | 0.005 | -1.215 | 0.224 | -0.015 | 0.003 |
| **Capital\_Employed** | -0.0124 | 0.010 | -1.294 | 0.196 | -0.031 | 0.006 |
| **Total\_Debt** | 0.0213 | 0.007 | 3.008 | 0.003 | 0.007 | 0.035 |
| **Gross\_Block** | 0.0012 | 0.004 | 0.288 | 0.773 | -0.007 | 0.009 |
| **Net\_Working\_Capital** | -0.0040 | 0.010 | -0.400 | 0.689 | -0.023 | 0.015 |
| **Current\_Assets** | 0.0053 | 0.008 | 0.640 | 0.522 | -0.011 | 0.022 |
| **Current\_Liabilities\_and\_Provisions** | 0.0010 | 0.013 | 0.081 | 0.935 | -0.024 | 0.026 |
| **Total\_Assets\_by\_Liabilities** | 0.0065 | 0.008 | 0.795 | 0.427 | -0.010 | 0.023 |
| **Gross\_Sales** | -0.0114 | 0.010 | -1.113 | 0.266 | -0.031 | 0.009 |
| **Net\_Sales** | 0.0203 | 0.022 | 0.939 | 0.348 | -0.022 | 0.063 |
| **Other\_Income** | -0.0076 | 0.079 | -0.097 | 0.923 | -0.162 | 0.147 |
| **Value\_Of\_Output** | -0.0188 | 0.014 | -1.348 | 0.178 | -0.046 | 0.009 |
| **Cost\_of\_Production** | 0.0065 | 0.013 | 0.510 | 0.610 | -0.018 | 0.031 |
| **Selling\_Cost** | -0.0350 | 0.103 | -0.340 | 0.734 | -0.237 | 0.167 |
| **PBIDT** | -0.0177 | 0.040 | -0.438 | 0.661 | -0.097 | 0.062 |
| **PBDT** | -0.1551 | 0.145 | -1.069 | 0.285 | -0.440 | 0.129 |
| **PBIT** | 0.0184 | 0.050 | 0.368 | 0.713 | -0.080 | 0.117 |
| **PBT** | 0.0480 | 0.212 | 0.226 | 0.821 | -0.368 | 0.464 |
| **PAT** | -0.0803 | 0.247 | -0.325 | 0.745 | -0.565 | 0.404 |
| **Adjusted\_PAT** | 0.0038 | 0.077 | 0.050 | 0.960 | -0.147 | 0.155 |
| **CP** | 0.1585 | 0.154 | 1.028 | 0.304 | -0.144 | 0.461 |
| **Revenue\_earnings\_in\_forex** | -0.0277 | 0.037 | -0.754 | 0.451 | -0.100 | 0.044 |
| **Revenue\_expenses\_in\_forex** | 0.0533 | 0.038 | 1.414 | 0.157 | -0.021 | 0.127 |
| **Book\_Value\_Unit\_Curr** | -0.0215 | 0.035 | -0.622 | 0.534 | -0.089 | 0.046 |
| **Book\_Value\_Adj\_Unit\_Curr** | -0.0684 | 0.036 | -1.878 | 0.060 | -0.140 | 0.003 |
| **Market\_Capitalisation** | -0.0043 | 0.004 | -1.205 | 0.228 | -0.011 | 0.003 |
| **CEPS\_annualised\_Unit\_Curr** | -0.0612 | 0.050 | -1.218 | 0.223 | -0.160 | 0.037 |
| **Cash\_Flow\_From\_Operating\_Activities** | 0.0100 | 0.027 | 0.375 | 0.707 | -0.042 | 0.062 |
| **Cash\_Flow\_From\_Investing\_Activities** | -0.0427 | 0.051 | -0.834 | 0.404 | -0.143 | 0.058 |
| **Cash\_Flow\_From\_Financing\_Activities** | 0.0023 | 0.043 | 0.053 | 0.958 | -0.083 | 0.087 |
| **ROG\_Net\_Worth\_perc** | -0.0236 | 0.012 | -1.899 | 0.058 | -0.048 | 0.001 |
| **ROG\_Capital\_Employed\_perc** | 0.0237 | 0.011 | 2.224 | 0.026 | 0.003 | 0.045 |
| **ROG\_Gross\_Block\_perc** | -0.0316 | 0.021 | -1.476 | 0.140 | -0.074 | 0.010 |
| **ROG\_Gross\_Sales\_perc** | 0.0896 | 0.120 | 0.748 | 0.454 | -0.145 | 0.325 |
| **ROG\_Net\_Sales\_perc** | -0.0914 | 0.119 | -0.766 | 0.444 | -0.325 | 0.143 |
| **ROG\_Cost\_of\_Production\_perc** | -0.0060 | 0.004 | -1.432 | 0.152 | -0.014 | 0.002 |
| **ROG\_Total\_Assets\_perc** | -0.0256 | 0.010 | -2.519 | 0.012 | -0.046 | -0.006 |
| **ROG\_PBIDT\_perc** | -0.0058 | 0.006 | -1.016 | 0.310 | -0.017 | 0.005 |
| **ROG\_PBDT\_perc** | 0.0064 | 0.006 | 1.148 | 0.251 | -0.005 | 0.017 |
| **ROG\_PBIT\_perc** | 0.0049 | 0.005 | 0.954 | 0.340 | -0.005 | 0.015 |
| **ROG\_PBT\_perc** | -0.0021 | 0.005 | -0.435 | 0.663 | -0.012 | 0.008 |
| **ROG\_PAT\_perc** | 0.0014 | 0.004 | 0.351 | 0.726 | -0.006 | 0.009 |
| **ROG\_CP\_perc** | -0.0042 | 0.004 | -0.928 | 0.353 | -0.013 | 0.005 |
| **ROG\_Market\_Capitalisation\_perc** | -0.0024 | 0.003 | -0.799 | 0.424 | -0.008 | 0.003 |
| **Current\_RatioLatest** | -0.5445 | 0.094 | -5.763 | 0.000 | -0.730 | -0.359 |
| **Fixed\_Assets\_RatioLatest** | -0.0015 | 0.112 | -0.014 | 0.989 | -0.221 | 0.218 |
| **Inventory\_RatioLatest** | -0.0678 | 0.026 | -2.561 | 0.010 | -0.120 | -0.016 |
| **Debtors\_RatioLatest** | -0.0396 | 0.026 | -1.523 | 0.128 | -0.091 | 0.011 |
| **Total\_Asset\_Turnover\_RatioLatest** | -0.0678 | 0.207 | -0.328 | 0.743 | -0.473 | 0.337 |
| **Interest\_Cover\_RatioLatest** | -0.1067 | 0.053 | -2.023 | 0.043 | -0.210 | -0.003 |
| **PBIDTM\_percLatest** | 0.0133 | 0.032 | 0.419 | 0.675 | -0.049 | 0.076 |
| **PBITM\_percLatest** | -0.0783 | 0.040 | -1.943 | 0.052 | -0.157 | 0.001 |
| **PBDTM\_percLatest** | 0.0095 | 0.054 | 0.177 | 0.859 | -0.096 | 0.115 |
| **CPM\_percLatest** | -0.0490 | 0.068 | -0.723 | 0.470 | -0.182 | 0.084 |
| **APATM\_percLatest** | 0.1252 | 0.069 | 1.802 | 0.072 | -0.011 | 0.261 |
| **Debtors\_Velocity\_Days** | -0.0042 | 0.002 | -2.771 | 0.006 | -0.007 | -0.001 |
| **Creditors\_Velocity\_Days** | 0.0011 | 0.002 | 0.676 | 0.499 | -0.002 | 0.004 |
| **Inventory\_Velocity\_Days** | 0.0015 | 0.002 | 0.827 | 0.408 | -0.002 | 0.005 |
| **Value\_of\_Output\_by\_Total\_Assets** | 0.7860 | 0.358 | 2.196 | 0.028 | 0.085 | 1.487 |
| **Value\_of\_Output\_by\_Gross\_Block** | -0.0875 | 0.111 | -0.790 | 0.430 | -0.305 | 0.130 |

Insights:

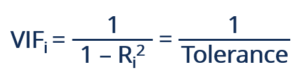
1. We can clearly see that results from our 1st Model have all 61 columns,,after removing unnecessary fields of Co\_num, next\_year\_networth etc.
2. There are many columns with more than 0.05 P value, which denotes those fields are not required for predicting the Target field.
3. As per initial Heatmap, we already know, that data have multi collinearity in the data, so we don’t need all of the columns for predicting target values

# **Removing multicollinearity using VIF:**

**What is VIF:**

The Variance Inflation Factor (VIF) measures the severity of multicollinearity in regression analysis. It is a statistical concept that indicates the increase in the variance of a regression coefficient as a result of collinearity.

In ordinary least square (OLS) regression analysis, multicollinearity exists when two or more of the independent variables demonstrate a linear relationship between them. VIF can be calculated by the formula below:



Where **Ri2**represents the unadjusted coefficient of determination for regressing the ith independent variable on the remaining ones. The reciprocal of VIF is known as **tolerance**. Either VIF or tolerance can be used to detect multicollinearity, depending on personal preference.

**Interpreting the Variance Inflation Factor**

Variance inflation factors range from 1 upwards. The numerical value for VIF tells you (in decimal form) what percentage the variance (i.e. the standard error squared) is inflated for each coefficient. For example, a VIF of 1.9 tells you that the variance of a particular coefficient is 90% bigger than what you would expect if there was no multicollinearity — if there was no correlation with other predictors.  
A **rule of thumb** for interpreting the variance inflation factor:

* 1 = not correlated.
* Between 1 and 5 = moderately correlated.
* Greater than 5 = highly correlated.

Exactly how large a VIF has to be before it causes issues is a subject of debate. What is known is that the more your VIF increases, the less reliable your regression results are going to be. In general, a VIF above 10 indicates high correlation and is cause for concern. Some authors suggest a more conservative level of 2.5 or above.

**Summary:**

* Variance inflation factor (VIF) is used to detect the severity of multicollinearity in the ordinary least square (OLS) regression analysis.
* Multicollinearity inflates the variance and type II error. It makes the coefficient of a variable consistent but unreliable.
* VIF measures the number of inflated variances caused by multicollinearity.
* Eliminate fields with Greater than 5 , which means there is highly correlated. We should do it one by one for each field.

# **Model 2 with VIF threshold 4:**

To Notice, since we have lot of independent fields, we should go and check for each single columns and check VIF multiple times.

Rather we have used a Loop function, its checking VIF in Iterations and remove one column in one iteration and calculate VIF again. This loop will run until we get all the fields with lesser value of VIF than threshold limit.

For this testing we will keep VIF threshold value as 5.

Keep VIF threshold value as 5:

After executing Loop function, we received 34 columns and about half of the fields are eliminated, which had very high co-relation in it.

Remained fields are:

Index(['Total\_Debt', 'Net\_Working\_Capital', 'Other\_Income', 'Selling\_Cost',

'Adjusted\_PAT', 'Revenue\_earnings\_in\_forex',

'Revenue\_expenses\_in\_forex', 'Book\_Value\_Adj\_Unit\_Curr',

'Market\_Capitalisation', 'CEPS\_annualised\_Unit\_Curr',

'Cash\_Flow\_From\_Operating\_Activities',

'Cash\_Flow\_From\_Investing\_Activities',

'Cash\_Flow\_From\_Financing\_Activities', 'ROG\_Net\_Worth\_perc',

'ROG\_Capital\_Employed\_perc', 'ROG\_Gross\_Block\_perc',

'ROG\_Net\_Sales\_perc', 'ROG\_Cost\_of\_Production\_perc',

'ROG\_Total\_Assets\_perc', 'ROG\_PBIT\_perc', 'ROG\_CP\_perc',

'ROG\_Market\_Capitalisation\_perc', 'Current\_RatioLatest',

'Inventory\_RatioLatest', 'Debtors\_RatioLatest',

'Total\_Asset\_Turnover\_RatioLatest', 'Interest\_Cover\_RatioLatest',

'PBITM\_percLatest', 'CPM\_percLatest', 'Debtors\_Velocity\_Days',

'Creditors\_Velocity\_Days', 'Inventory\_Velocity\_Days',

'Value\_of\_Output\_by\_Gross\_Block', 'default'],

We ran the stats Model on above listed field, and this is the Stats summary for same:

|  |  |  |  |
| --- | --- | --- | --- |
| **Dep. Variable:** | default | **No. Observations:** | 2402 |
| **Model:** | Logit | **Df Residuals:** | 2368 |
| **Method:** | MLE | **Df Model:** | 33 |
| **Date:** | Tue, 15 Nov 2022 | **Pseudo R-squ.:** | 0.6546 |
| **Time:** | 18:08:40 | **Log-Likelihood:** | -284.41 |
| **converged:** | True | **LL-Null:** | -823.47 |
| **Covariance Type:** | nonrobust | **LLR p-value:** | 3.364e-205 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **coef** | **std err** | **z** | **P>|z|** | **[0.025** | **0.975]** |
| **Intercept** | -0.2422 | 0.226 | -1.072 | 0.284 | -0.685 | 0.201 |
| **Total\_Debt** | 0.0157 | 0.004 | 3.831 | 0.000 | 0.008 | 0.024 |
| **Net\_Working\_Capital** | -0.0067 | 0.004 | -1.643 | 0.100 | -0.015 | 0.001 |
| **Other\_Income** | -0.0227 | 0.066 | -0.341 | 0.733 | -0.153 | 0.108 |
| **Selling\_Cost** | -0.0587 | 0.081 | -0.725 | 0.469 | -0.218 | 0.100 |
| **Adjusted\_PAT** | -0.0126 | 0.049 | -0.259 | 0.796 | -0.108 | 0.083 |
| **Revenue\_earnings\_in\_forex** | -0.0358 | 0.033 | -1.084 | 0.278 | -0.101 | 0.029 |
| **Revenue\_expenses\_in\_forex** | 0.0480 | 0.034 | 1.433 | 0.152 | -0.018 | 0.114 |
| **Book\_Value\_Adj\_Unit\_Curr** | -0.1004 | 0.010 | -10.195 | 0.000 | -0.120 | -0.081 |
| **Market\_Capitalisation** | -0.0081 | 0.003 | -3.015 | 0.003 | -0.013 | -0.003 |
| **CEPS\_annualised\_Unit\_Curr** | -0.0670 | 0.037 | -1.820 | 0.069 | -0.139 | 0.005 |
| **Cash\_Flow\_From\_Operating\_Activities** | 0.0037 | 0.022 | 0.169 | 0.866 | -0.039 | 0.047 |
| **Cash\_Flow\_From\_Investing\_Activities** | -0.0143 | 0.041 | -0.346 | 0.730 | -0.095 | 0.067 |
| **Cash\_Flow\_From\_Financing\_Activities** | 0.0071 | 0.038 | 0.186 | 0.853 | -0.067 | 0.082 |
| **ROG\_Net\_Worth\_perc** | -0.0267 | 0.012 | -2.317 | 0.021 | -0.049 | -0.004 |
| **ROG\_Capital\_Employed\_perc** | 0.0220 | 0.010 | 2.230 | 0.026 | 0.003 | 0.041 |
| **ROG\_Gross\_Block\_perc** | -0.0308 | 0.020 | -1.512 | 0.131 | -0.071 | 0.009 |
| **ROG\_Net\_Sales\_perc** | -0.0020 | 0.004 | -0.474 | 0.636 | -0.010 | 0.006 |
| **ROG\_Cost\_of\_Production\_perc** | -0.0062 | 0.004 | -1.551 | 0.121 | -0.014 | 0.002 |
| **ROG\_Total\_Assets\_perc** | -0.0216 | 0.010 | -2.231 | 0.026 | -0.041 | -0.003 |
| **ROG\_PBIT\_perc** | 0.0028 | 0.002 | 1.188 | 0.235 | -0.002 | 0.008 |
| **ROG\_CP\_perc** | -0.0016 | 0.002 | -0.716 | 0.474 | -0.006 | 0.003 |
| **ROG\_Market\_Capitalisation\_perc** | -0.0008 | 0.003 | -0.290 | 0.772 | -0.006 | 0.005 |
| **Current\_RatioLatest** | -0.5418 | 0.091 | -5.931 | 0.000 | -0.721 | -0.363 |
| **Inventory\_RatioLatest** | -0.0537 | 0.023 | -2.382 | 0.017 | -0.098 | -0.010 |
| **Debtors\_RatioLatest** | -0.0274 | 0.023 | -1.186 | 0.236 | -0.073 | 0.018 |
| **Total\_Asset\_Turnover\_RatioLatest** | 0.2183 | 0.132 | 1.648 | 0.099 | -0.041 | 0.478 |
| **Interest\_Cover\_RatioLatest** | -0.0728 | 0.047 | -1.547 | 0.122 | -0.165 | 0.019 |
| **PBITM\_percLatest** | -0.0307 | 0.015 | -2.040 | 0.041 | -0.060 | -0.001 |
| **CPM\_percLatest** | 0.0044 | 0.017 | 0.251 | 0.802 | -0.030 | 0.038 |
| **Debtors\_Velocity\_Days** | -0.0042 | 0.001 | -2.909 | 0.004 | -0.007 | -0.001 |
| **Creditors\_Velocity\_Days** | 0.0011 | 0.002 | 0.735 | 0.462 | -0.002 | 0.004 |
| **Inventory\_Velocity\_Days** | 0.0011 | 0.002 | 0.689 | 0.491 | -0.002 | 0.004 |
| **Value\_of\_Output\_by\_Gross\_Block** | -0.0515 | 0.044 | -1.157 | 0.247 | -0.139 | 0.036 |

Still we see lot of Columns have P value more than 0.05, so lets run VIF function one more time, and eliminate fields , which have VIF more than 4

# **Model 3 with VIF threshold 4:**

We have observed that threshold limit 5 is not eliminating many fields, which have co-relations among them, so we have eliminated fields with threshold more than 4.

After adjusting threshold limit to 4 , we were successfully able to eliminate 3 more fields. Which means we have now below listed fields for our Third model:

Index(['Total\_Debt', 'Net\_Working\_Capital', 'Other\_Income', 'Adjusted\_PAT',

'Revenue\_earnings\_in\_forex', 'Revenue\_expenses\_in\_forex',

'Book\_Value\_Adj\_Unit\_Curr', 'Market\_Capitalisation',

'Cash\_Flow\_From\_Operating\_Activities',

'Cash\_Flow\_From\_Investing\_Activities',

'Cash\_Flow\_From\_Financing\_Activities', 'ROG\_Net\_Worth\_perc',

'ROG\_Capital\_Employed\_perc', 'ROG\_Gross\_Block\_perc',

'ROG\_Net\_Sales\_perc', 'ROG\_Cost\_of\_Production\_perc',

'ROG\_Total\_Assets\_perc', 'ROG\_PBIT\_perc', 'ROG\_CP\_perc',

'ROG\_Market\_Capitalisation\_perc', 'Current\_RatioLatest',

'Inventory\_RatioLatest', 'Debtors\_RatioLatest',

'Total\_Asset\_Turnover\_RatioLatest', 'Interest\_Cover\_RatioLatest',

'CPM\_percLatest', 'Debtors\_Velocity\_Days', 'Creditors\_Velocity\_Days',

'Inventory\_Velocity\_Days', 'Value\_of\_Output\_by\_Gross\_Block', 'default']

**We have also adjusted cut-off limit to 0.4, earlier it was 0.5, which means, items more than 0.5 predicted values will be marked as 1, which is targeted value.**

**But we will consider Company as Default, even if there are 40% chances of this being Default, this will give us more accurate results for getting targeted values.**

Again, run the stats model on top of above listed columns.

|  |  |  |  |
| --- | --- | --- | --- |
| **Dep. Variable:** | default | **No. Observations:** | 2402 |
| **Model:** | Logit | **Df Residuals:** | 2371 |
| **Method:** | MLE | **Df Model:** | 30 |
| **Date:** | Tue, 15 Nov 2022 | **Pseudo R-squ.:** | 0.6501 |
| **Time:** | 18:08:42 | **Log-Likelihood:** | -288.13 |
| **converged:** | True | **LL-Null:** | -823.47 |
| **Covariance Type:** | nonrobust | **LLR p-value:** | 5.990e-206 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **coef** | **std err** | **z** | **P>|z|** | **[0.025** | **0.975]** |
| **Intercept** | -0.2233 | 0.225 | -0.993 | 0.321 | -0.664 | 0.218 |
| **Total\_Debt** | 0.0143 | 0.004 | 3.595 | 0.000 | 0.006 | 0.022 |
| **Net\_Working\_Capital** | -0.0064 | 0.004 | -1.687 | 0.092 | -0.014 | 0.001 |
| **Other\_Income** | -0.0546 | 0.065 | -0.843 | 0.399 | -0.182 | 0.072 |
| **Adjusted\_PAT** | -0.0448 | 0.044 | -1.021 | 0.307 | -0.131 | 0.041 |
| **Revenue\_earnings\_in\_forex** | -0.0454 | 0.031 | -1.457 | 0.145 | -0.106 | 0.016 |
| **Revenue\_expenses\_in\_forex** | 0.0477 | 0.033 | 1.463 | 0.143 | -0.016 | 0.112 |
| **Book\_Value\_Adj\_Unit\_Curr** | -0.1006 | 0.010 | -10.439 | 0.000 | -0.120 | -0.082 |
| **Market\_Capitalisation** | -0.0079 | 0.003 | -3.007 | 0.003 | -0.013 | -0.003 |
| **Cash\_Flow\_From\_Operating\_Activities** | -0.0026 | 0.022 | -0.119 | 0.905 | -0.045 | 0.040 |
| **Cash\_Flow\_From\_Investing\_Activities** | -0.0153 | 0.039 | -0.388 | 0.698 | -0.092 | 0.062 |
| **Cash\_Flow\_From\_Financing\_Activities** | 0.0049 | 0.037 | 0.132 | 0.895 | -0.068 | 0.078 |
| **ROG\_Net\_Worth\_perc** | -0.0284 | 0.011 | -2.531 | 0.011 | -0.050 | -0.006 |
| **ROG\_Capital\_Employed\_perc** | 0.0197 | 0.010 | 2.033 | 0.042 | 0.001 | 0.039 |
| **ROG\_Gross\_Block\_perc** | -0.0265 | 0.020 | -1.310 | 0.190 | -0.066 | 0.013 |
| **ROG\_Net\_Sales\_perc** | -0.0022 | 0.004 | -0.537 | 0.591 | -0.010 | 0.006 |
| **ROG\_Cost\_of\_Production\_perc** | -0.0060 | 0.004 | -1.519 | 0.129 | -0.014 | 0.002 |
| **ROG\_Total\_Assets\_perc** | -0.0229 | 0.010 | -2.372 | 0.018 | -0.042 | -0.004 |
| **ROG\_PBIT\_perc** | 0.0025 | 0.002 | 1.049 | 0.294 | -0.002 | 0.007 |
| **ROG\_CP\_perc** | -0.0017 | 0.002 | -0.769 | 0.442 | -0.006 | 0.003 |
| **ROG\_Market\_Capitalisation\_perc** | -0.0008 | 0.003 | -0.304 | 0.761 | -0.006 | 0.005 |
| **Current\_RatioLatest** | -0.5498 | 0.091 | -6.054 | 0.000 | -0.728 | -0.372 |
| **Inventory\_RatioLatest** | -0.0503 | 0.023 | -2.224 | 0.026 | -0.095 | -0.006 |
| **Debtors\_RatioLatest** | -0.0281 | 0.023 | -1.224 | 0.221 | -0.073 | 0.017 |
| **Total\_Asset\_Turnover\_RatioLatest** | 0.1895 | 0.130 | 1.458 | 0.145 | -0.065 | 0.444 |
| **Interest\_Cover\_RatioLatest** | -0.0896 | 0.047 | -1.902 | 0.057 | -0.182 | 0.003 |
| **CPM\_percLatest** | -0.0253 | 0.011 | -2.345 | 0.019 | -0.046 | -0.004 |
| **Debtors\_Velocity\_Days** | -0.0037 | 0.001 | -2.616 | 0.009 | -0.006 | -0.001 |
| **Creditors\_Velocity\_Days** | 0.0010 | 0.001 | 0.689 | 0.491 | -0.002 | 0.004 |
| **Inventory\_Velocity\_Days** | 0.0011 | 0.002 | 0.659 | 0.510 | -0.002 | 0.004 |
| **Value\_of\_Output\_by\_Gross\_Block** | -0.0611 | 0.044 | -1.392 | 0.164 | -0.147 | 0.025 |

Check for VIF for all the fields in this model :

|  | **variables** | **VIF** |
| --- | --- | --- |
| **8** | Cash\_Flow\_From\_Operating\_Activities | 3.804148 |
| **7** | Market\_Capitalisation | 3.769023 |
| **12** | ROG\_Capital\_Employed\_perc | 3.671089 |
| **0** | Total\_Debt | 3.620201 |
| **2** | Other\_Income | 3.483271 |
| **5** | Revenue\_expenses\_in\_forex | 3.418506 |
| **18** | ROG\_CP\_perc | 3.262816 |
| **17** | ROG\_PBIT\_perc | 3.251599 |
| **23** | Total\_Asset\_Turnover\_RatioLatest | 3.135009 |
| **16** | ROG\_Total\_Assets\_perc | 3.108078 |
| **3** | Adjusted\_PAT | 3.033204 |
| **1** | Net\_Working\_Capital | 2.902452 |
| **4** | Revenue\_earnings\_in\_forex | 2.872264 |
| **10** | Cash\_Flow\_From\_Financing\_Activities | 2.863477 |
| **11** | ROG\_Net\_Worth\_perc | 2.719630 |
| **9** | Cash\_Flow\_From\_Investing\_Activities | 2.503964 |
| **6** | Book\_Value\_Adj\_Unit\_Curr | 2.460811 |
| **22** | Debtors\_RatioLatest | 2.346833 |
| **26** | Debtors\_Velocity\_Days | 2.345360 |
| **29** | Value\_of\_Output\_by\_Gross\_Block | 2.316941 |
| **24** | Interest\_Cover\_RatioLatest | 2.250516 |
| **21** | Inventory\_RatioLatest | 2.221555 |
| **27** | Creditors\_Velocity\_Days | 2.180764 |
| **20** | Current\_RatioLatest | 2.175016 |
| **14** | ROG\_Net\_Sales\_perc | 2.081818 |
| **15** | ROG\_Cost\_of\_Production\_perc | 1.995917 |
| **28** | Inventory\_Velocity\_Days | 1.860591 |
| **25** | CPM\_percLatest | 1.744019 |
| **19** | ROG\_Market\_Capitalisation\_perc | 1.621405 |
| **13** | ROG\_Gross\_Block\_perc | 1.477094 |
| **30** | default | 1.397951 |

We can see that all of the fields have less than 4 VIF value for this.

#### **Validating the model on train set**

Confusion Matrix for train data:

[[2094 48]

[ 55 205]]

Calculate Recall and other matrix:

precision recall f1-score support

0 0.97 0.98 0.98 2142

1 0.81 0.79 0.80 260

accuracy 0.96 2402

macro avg 0.89 0.88 0.89 2402

weighted avg 0.96 0.96 0.96 2402

#### **Validating the model on test set**

Confusion Matrix for train data:

[[1027 29]

[ 27 101]]

Calculate Recall and other matrix:

precision recall f1-score support

0 0.97 0.97 0.97 1056

1 0.78 0.79 0.78 128

accuracy 0.95 1184

macro avg 0.88 0.88 0.88 1184

weighted avg 0.95 0.95 0.95 1184

Insights:

1. We are more interested in RECALL value as it denotes, how much “default” value or Targeted value we are predicting correctly.
2. Recall value for train data and test data are 81% and 79% in sequence, which is not bad.
3. Still data is imbalanced, because our targeted values are very less in Sample data.

# **Model 4 - Balance data using SMOTE and threshold VIF as 4:**

We will be using SMOTE functionality to generate data for target values, and we will upscaling the data for this case.

Steps:

1. Again, we will perform Smote on full sample data , so total fields are 62
2. Upscale data using imblearn.over\_sampling library
3. We also run VIF function to check best features, which can be used for Stats model, after executing VIF calculation function, we found 31 fields and we kept threshold limit as 4 for eliminating features with High co-relation.
4. We generated the Stats model technique and following is the result for Stats model:

|  |  |  |  |
| --- | --- | --- | --- |
| **Dep. Variable:** | default | **No. Observations:** | 3748 |
| **Model:** | Logit | **Df Residuals:** | 3717 |
| **Method:** | MLE | **Df Model:** | 30 |
| **Date:** | Tue, 15 Nov 2022 | **Pseudo R-squ.:** | 0.7452 |
| **Time:** | 18:08:45 | **Log-Likelihood:** | -652.27 |
| **converged:** | True | **LL-Null:** | -2559.5 |
| **Covariance Type:** | nonrobust | **LLR p-value:** | 0.000 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **coef** | **std err** | **z** | **P>|z|** | **[0.025** | **0.975]** |
| **Intercept** | 1.6931 | 0.162 | 10.462 | 0.000 | 1.376 | 2.010 |
| **Total\_Debt** | 0.0240 | 0.003 | 7.305 | 0.000 | 0.018 | 0.030 |
| **Net\_Working\_Capital** | -0.0053 | 0.003 | -1.782 | 0.075 | -0.011 | 0.001 |
| **Other\_Income** | -0.1006 | 0.050 | -2.028 | 0.043 | -0.198 | -0.003 |
| **Adjusted\_PAT** | -0.0177 | 0.029 | -0.602 | 0.547 | -0.075 | 0.040 |
| **Revenue\_earnings\_in\_forex** | -0.0699 | 0.024 | -2.966 | 0.003 | -0.116 | -0.024 |
| **Revenue\_expenses\_in\_forex** | 0.0389 | 0.024 | 1.609 | 0.108 | -0.008 | 0.086 |
| **Book\_Value\_Adj\_Unit\_Curr** | -0.1220 | 0.007 | -17.710 | 0.000 | -0.135 | -0.108 |
| **Market\_Capitalisation** | -0.0097 | 0.002 | -4.788 | 0.000 | -0.014 | -0.006 |
| **Cash\_Flow\_From\_Operating\_Activities** | -0.0262 | 0.016 | -1.620 | 0.105 | -0.058 | 0.006 |
| **Cash\_Flow\_From\_Investing\_Activities** | -0.0400 | 0.031 | -1.305 | 0.192 | -0.100 | 0.020 |
| **Cash\_Flow\_From\_Financing\_Activities** | -0.0079 | 0.028 | -0.283 | 0.777 | -0.062 | 0.047 |
| **ROG\_Net\_Worth\_perc** | -0.0340 | 0.008 | -4.402 | 0.000 | -0.049 | -0.019 |
| **ROG\_Capital\_Employed\_perc** | 0.0231 | 0.007 | 3.354 | 0.001 | 0.010 | 0.037 |
| **ROG\_Gross\_Block\_perc** | -0.0330 | 0.015 | -2.201 | 0.028 | -0.062 | -0.004 |
| **ROG\_Net\_Sales\_perc** | -0.0058 | 0.003 | -1.793 | 0.073 | -0.012 | 0.001 |
| **ROG\_Cost\_of\_Production\_perc** | -0.0070 | 0.003 | -2.447 | 0.014 | -0.013 | -0.001 |
| **ROG\_Total\_Assets\_perc** | -0.0314 | 0.007 | -4.631 | 0.000 | -0.045 | -0.018 |
| **ROG\_PBIT\_perc** | 0.0043 | 0.002 | 2.548 | 0.011 | 0.001 | 0.008 |
| **ROG\_CP\_perc** | -0.0021 | 0.002 | -1.300 | 0.194 | -0.005 | 0.001 |
| **ROG\_Market\_Capitalisation\_perc** | -0.0027 | 0.002 | -1.358 | 0.174 | -0.007 | 0.001 |
| **Current\_RatioLatest** | -0.6289 | 0.056 | -11.244 | 0.000 | -0.739 | -0.519 |
| **Inventory\_RatioLatest** | -0.0543 | 0.014 | -3.784 | 0.000 | -0.082 | -0.026 |
| **Debtors\_RatioLatest** | -0.0576 | 0.016 | -3.514 | 0.000 | -0.090 | -0.025 |
| **Total\_Asset\_Turnover\_RatioLatest** | 0.3287 | 0.089 | 3.707 | 0.000 | 0.155 | 0.503 |
| **Interest\_Cover\_RatioLatest** | -0.0856 | 0.034 | -2.515 | 0.012 | -0.152 | -0.019 |
| **CPM\_percLatest** | -0.0507 | 0.008 | -6.185 | 0.000 | -0.067 | -0.035 |
| **Debtors\_Velocity\_Days** | -0.0059 | 0.001 | -5.662 | 0.000 | -0.008 | -0.004 |
| **Creditors\_Velocity\_Days** | 0.0010 | 0.001 | 0.909 | 0.363 | -0.001 | 0.003 |
| **Inventory\_Velocity\_Days** | 0.0005 | 0.001 | 0.414 | 0.679 | -0.002 | 0.003 |
| **Value\_of\_Output\_by\_Gross\_Block** | -0.0945 | 0.028 | -3.384 | 0.001 | -0.149 | -0.040 |

#### **Validating the model on train set**

Confusion Matrix for train data:

[[2009 133]

[ 84 1522]]

Calculate Recall and other matrix:

precision recall f1-score support

0 0.96 0.94 0.95 2142

1 0.92 0.95 0.93 1606

accuracy 0.94 3748

macro avg 0.94 0.94 0.94 3748

weighted avg 0.94 0.94 0.94 3748

#### **Validating the model on test set**

Confusion Matrix for train data:

[[972 84]

[ 11 117]]

Calculate Recall and other matrix:

precision recall f1-score support

0 0.99 0.92 0.95 1056

1 0.58 0.91 0.71 128

accuracy 0.92 1184

macro avg 0.79 0.92 0.83 1184

weighted avg 0.94 0.92 0.93 1184

Insights:

1. We are more interested in RECALL value as it denotes, how much “default” value or Targeted value we are predicting correctly.
2. Recall value for train data and test data are 95% and 91% in sequence, which is very good.
3. Accuracy on Train and Test data is also very good which is more than 92% with both train and test data.
4. Precision is Bad after SMOTE is implemented, because data was imbalanced, and we had to upscale data using SMOTE.

# **Perform Logistic Regression**

We have used Sklearn’s LogisticRegression library for building Logistic regression model . Logistic regression model works on Separate train and test data set. So we have used our split data for building model.

We are making some adjustments to the parameters in the Logistic Regression Class to get a better accuracy. We have used grid search method for identifying best parameters for Logit model.

**Argument:**

solver{‘newton-cg’, ‘lbfgs’, ‘liblinear’, ‘sag’, ‘saga’}, default=’lbfgs’ Algorithm to use in the optimization problem.

For small datasets, ‘liblinear’ is a good choice, whereas ‘sag’ and ‘saga’ are faster for large ones.

For multiclass problems, only ‘newton-cg’, ‘sag’, ‘saga’ and ‘lbfgs’ handle multinomial loss; ‘liblinear’ is limited to one-versus-rest schemes.

‘newton-cg’, ‘lbfgs’, ‘sag’ and ‘saga’ handle L2 or no penalty

‘liblinear’ and ‘saga’ also handle L1 penalty

‘saga’ also supports ‘elasticnet’ penalty

‘liblinear’ does not support setting penalty='none'

Note that ‘sag’ and ‘saga’ fast convergence is only guaranteed on features with approximately the same scale. You can preprocess the data with a scaler from sklearn.preprocessing.

New in version 0.17: Stochastic Average Gradient descent solver.

New in version 0.19: SAGA solver.

Changed in version 0.22: The default solver changed from ‘liblinear’ to ‘lbfgs’ in 0.22.

Reference : [Article on Solvers](https://towardsdatascience.com/dont-sweat-the-solver-stuff-aea7cddc3451)

Following is list of parameters we have gathered after running Grid search:

{'penalty': 'l1', 'solver': 'liblinear', 'tol': 0.01}

Matrices for Logistic Regression Model on testing data

Accuracy of the Logistic Regression Model is 0.9527027027027027

Confusion Matrix

[[1038 18]

[ 38 90]]

Classification Report

precision recall f1-score support

0 0.96 0.98 0.97 1056

1 0.83 0.70 0.76 128

accuracy 0.95 1184

macro avg 0.90 0.84 0.87 1184

weighted avg 0.95 0.95 0.95 1184

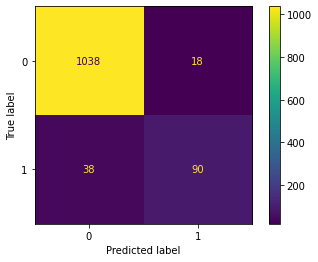
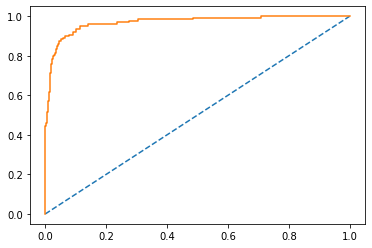


Figure 4 Confusion Matrix and AUC Curve for Train data using Logit model

**Matrices for Logistic Regression Model on Training data:**

Accuracy of the Logistic Regression Model is 0.9608659450457951

Confusion Matrix

[[2114 28]

[ 66 194]]

Classification Report

precision recall f1-score support

0 0.97 0.99 0.98 2142

1 0.87 0.75 0.80 260

accuracy 0.96 2402

macro avg 0.92 0.87 0.89 2402

weighted avg 0.96 0.96 0.96 2402

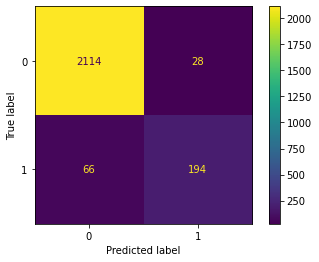
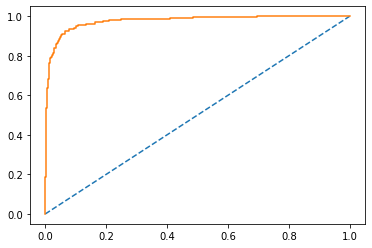


Figure 5Confusion Matrix and AUC Curve for Test data using Logit model

logit\_train\_precision 0.87

logit\_train\_recall 0.75

logit\_train\_acc 0.9608659450457951

logit\_train\_f1 0.8

logit\_test\_precision 0.83

logit\_test\_recall 0.7

logit\_test\_acc 0.9527027027027027

logit\_test\_f1 0.76

Insights:

* 1. Logistic regression model is not very bad on test and train both data sets in terms of predicting , its

giving 70% True positive rate on Test data and 75% on train data, which means, 70% times on test data its predicting correct true positives, which is our targeted column.

* 1. Still there are chances of improvements.
  2. There are about (18 out of 108 times ), its Type 2 error, its predicting wrong for DEFAULT Companies and marking them as Non Default. It has to be eliminated , because we have very less data, that too, if it is predicting wrong, then its not good.

# **Building a Random Forest Classifier**

Random Forest is a Supervised learning algorithm that is based on the ensemble learning method and many Decision Trees. Random Forest is a Bagging technique, so all calculations are run in parallel and there is no interaction between the Decision Trees when building them. RF can be used to solve both Classification and Regression tasks.

Some of the important parameters are highlighted below:

* n\_estimators — the number of decision trees you will be running in the model
* criterion — this variable allows you to select the criterion (loss function) used to determine model outcomes. We can select from loss functions such as mean squared error (MSE) and mean absolute error (MAE). The default value is MSE.
* max\_depth — this sets the maximum possible depth of each tree
* max\_features — the maximum number of features the model will consider when determining a split
* bootstrap — the default value for this is True, meaning the model follows bootstrapping principles (defined earlier)
* max\_samples — This parameter assumes bootstrapping is set to True, if not, this parameter doesn’t apply. In the case of True, this value sets the largest size of each sample for each tree.
* Other important parameters are min\_samples\_split, min\_samples\_leaf, n\_jobs, and others that can be read in the sklearn’s

We have again used Grid search method for finding best parameters using in this model.

Following are the results from Grid search:

{'max\_depth': 8,

'max\_features': 8,

'min\_samples\_leaf': 50,

'min\_samples\_split': 200,

'n\_estimators': 200}

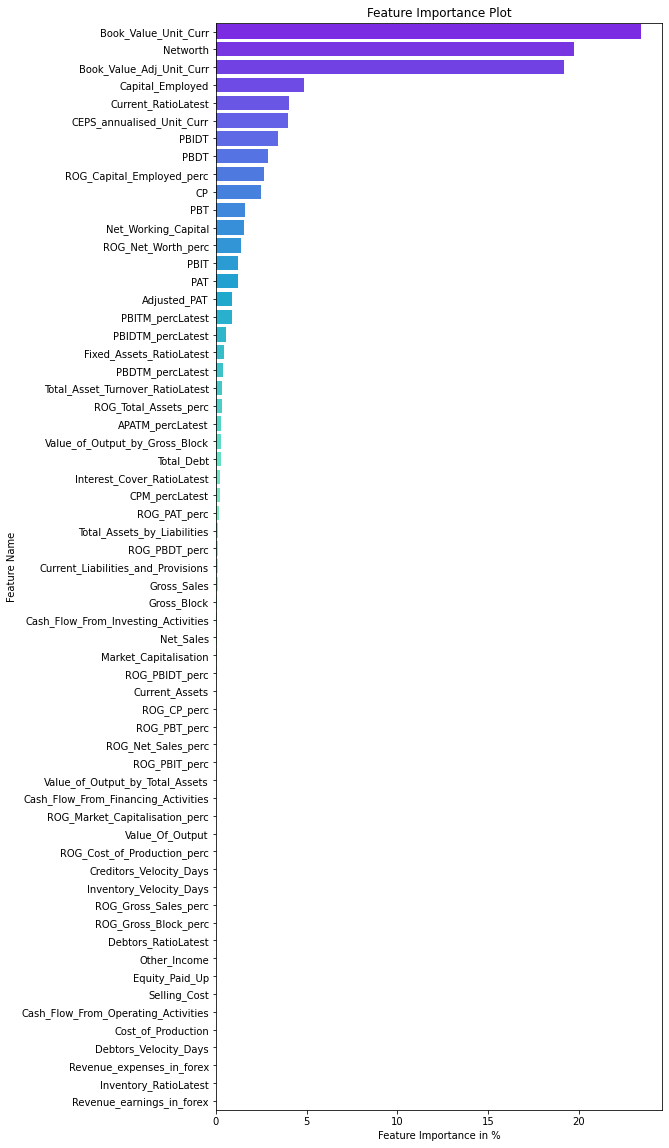
Feature importance graph:

Figure 6 Feature importance graph for Random Forest

**Following are importance of the features , used for building prediction models:**

Book\_Value\_Unit\_Curr 0.234250

Networth 0.197338

Book\_Value\_Adj\_Unit\_Curr 0.192101

Capital\_Employed 0.048405

Current\_RatioLatest 0.040189

... ...

Cost\_of\_Production 0.000004

Debtors\_Velocity\_Days 0.000002

Revenue\_expenses\_in\_forex 0.000002

Inventory\_RatioLatest 0.000002

Revenue\_earnings\_in\_forex 0.000000

**RF Model Performance Evaluation on Training data set:**

**Classification report for Training data set is as follows :**

precision recall f1-score support

0 0.99 0.99 0.99 2142

1 0.91 0.88 0.89 260

accuracy 0.98 2402

macro avg 0.95 0.94 0.94 2402

weighted avg 0.98 0.98 0.98 2402

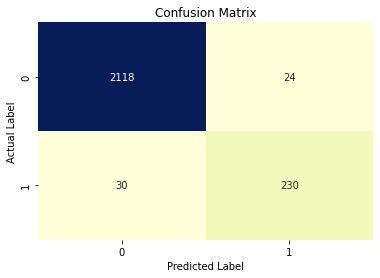
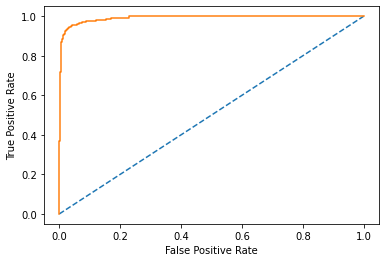
****

Figure 7 Confusion Metrix and AUC Curve on Train data for Random forest

**RF Model Performance Evaluation on Testing data set**

**Classification report for test data set is as follows :**

precision recall f1-score support

0 0.98 1.00 0.99 1056

1 0.96 0.87 0.91 128

accuracy 0.98 1184

macro avg 0.97 0.93 0.95 1184

weighted avg 0.98 0.98 0.98 1184

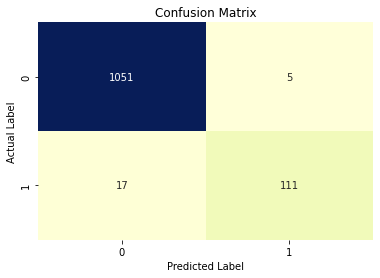
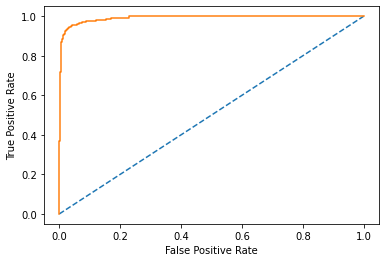
****

Figure 8 Confusion Metrix and AUC Curve on Test data for Random forest

rf\_train\_precision 0.91

rf\_train\_recall 0.88

rf\_train\_f1 0.89

rf\_accuracy\_train 0.98

rf\_auc\_train 0.99

rf\_test\_precision 0.96

rf\_test\_recall 0.87

rf\_test\_f1 0.91

rf\_accuracy\_test 0.98

rf\_auc\_test 0.99

**Insights:**

1. We have used all of the attributes of Sample data for building Random Forest model, but only a few fields are contributing in model predictions, list of those fields are with their weightage are:

Book\_Value\_Unit\_Curr 0.234250

Networth 0.197338

Book\_Value\_Adj\_Unit\_Curr 0.192101

Capital\_Employed 0.048405

Current\_RatioLatest 0.040189

1. Random Forest is performing very good on train and test data set and it’s recall value is 88% and 87%

respectively on train and test data set.

1. Recall denotes Of all the positive cases, what percentage are predicted positive? And if this model is predicting

88% time correctly on test data, which is a very good model.

1. Accuracy measures how often the model is correct. And this is also very good for train and test data set , and giving accuracy of 98% on Train and test both data set.

# **Perform LDA**

**Linear Discriminant Analysis** or **Normal Discriminant Analysis** or **Discriminant Function Analysis** is a dimensionality reduction technique that is commonly used for supervised classification problems. It is used for modelling differences in groups i.e. separating two or more classes. It is used to project the features in higher dimension space into a lower dimension space.

For example, we have two classes and we need to separate them efficiently. Classes can have multiple features. Using only a single feature to classify them may result in some overlapping as shown in the below figure. So, we will keep on increasing the number of features for proper classification.

Lightbox

We have used LinearDiscriminantAnalysis Library from Sklearn for performing LDA on our data. The assumptions made by an LDA model about your data:

* Each variable in the data is shaped in the form of a bell curve when plotted,i.e. Gaussian.
* The values of each variable vary around the mean by the same amount on the average,i.e. each attribute has the same variance.

**Model Performance Evaluation on Test data set:**

Accuracy of the LDA Model is 0.9366554054054054

Confusion Matrix

[[1037 19]

[ 56 72]]

Classification Report

precision recall f1-score support

0 0.95 0.98 0.97 1056

1 0.79 0.56 0.66 128

accuracy 0.94 1184

macro avg 0.87 0.77 0.81 1184

weighted avg 0.93 0.94 0.93 1184

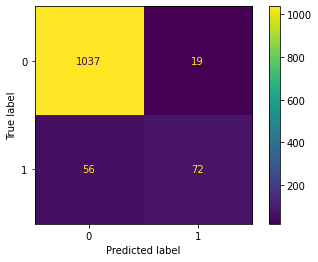
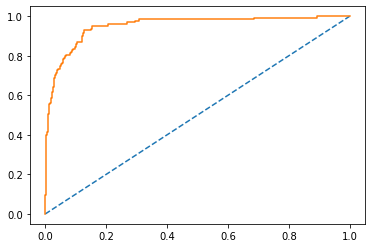


Figure 9 Confusion Metrix and AUC Curve on Test data for LDA

**Model Performance Evaluation on Train data set:**

Accuracy of the LDA Model is 0.9408825978351374

Confusion Matrix

[[2110 32]

[ 110 150]]

Classification Report

precision recall f1-score support

0 0.95 0.99 0.97 2142

1 0.82 0.58 0.68 260

accuracy 0.94 2402

macro avg 0.89 0.78 0.82 2402

weighted avg 0.94 0.94 0.94 2402

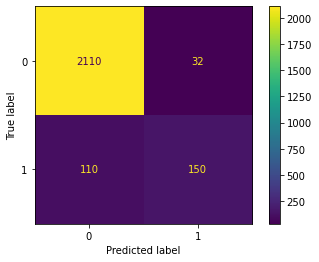
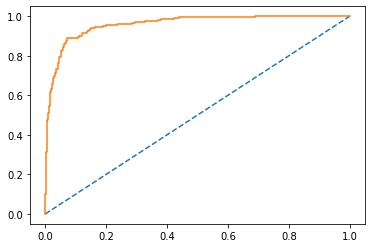
****

Figure 10 Confusion Metrix and AUC Curve on Train data for LDA

lda\_train\_precision 0.82

lda\_train\_recall 0.58

lda\_train\_acc 0.9408825978351374

lda\_train\_f1 0.68

lda\_test\_precision 0.79

lda\_test\_recall 0.56

lda\_test\_acc 0.9366554054054054

lda\_test\_f1 0.66

Insights:

1. This model is not working on our Sample data, its recall value is only 56% on test data and 58% on train data set.
2. Since LDA assumes that each input variable has the same variance, it is always better to standardize your data before using an LDA model. Keep the mean to be 0 and the standard deviation to be 1. And we can scale data for LDA analysis, that could be the reason, why we receive very less correct Recall value.

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

We have evaluated multiple models and calculated its performance metrices .

We put them together in Tabular format for better side by side understanding and comparing different models.

This is how the data looks like for all the models, we have built so far:

|  | **Accuracy** | **AUC** | **Recall** | **Precision** | **F1 Score** |
| --- | --- | --- | --- | --- | --- |
| **SM 4 Train** | 0.94 | Not applicable | 0.95 | 0.92 | 0.93 |
| **SM 4 Test** | 0.92 | Not applicable | 0.91 | 0.58 | 0.71 |
| **RF Train** | 0.98 | 0.99 | 0.88 | 0.91 | 0.89 |
| **RF Test** | 0.98 | 0.99 | 0.87 | 0.96 | 0.91 |
| **SM 3 Train** | 0.96 | Not applicable | 0.79 | 0.81 | 0.8 |
| **SM 3 Test** | 0.95 | Not applicable | 0.79 | 0.78 | 0.79 |
| **SM 1 Train** | 0.96 | Not applicable | 0.75 | 0.88 | 0.81 |
| **LR Train** | 0.960866 | 0.97619 | 0.75 | 0.87 | 0.8 |
| **SM 2 Train** | 0.96 | Not applicable | 0.73 | 0.87 | 0.79 |
| **SM 2 Test** | 0.95 | Not applicable | 0.72 | 0.83 | 0.77 |
| **SM 1 Test** | 0.95 | Not applicable | 0.7 | 0.81 | 0.75 |
| **LR Test** | 0.952703 | 0.96872 | 0.7 | 0.83 | 0.76 |
| **LDA Train** | 0.940883 | 0.959538 | 0.58 | 0.82 | 0.68 |
| **LDA Test** | 0.936655 | 0.951349 | 0.56 | 0.79 | 0.66 |

We have also compared Area under the curve for all the models on train and test data.

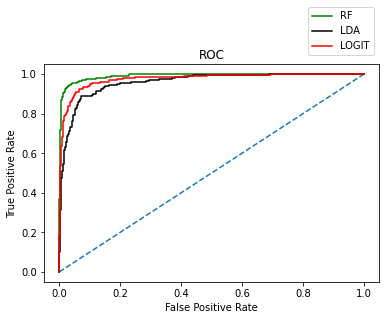


Figure 11 AUC Curve for all the models on Train data

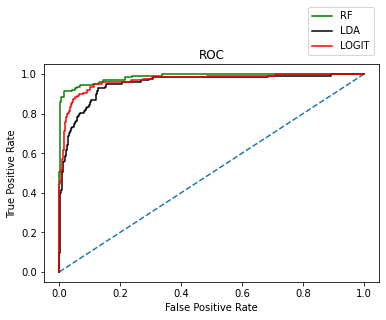


Figure 12 AUC Curve for all the models on Test data

**Comparing all the models:**

1. We have gathered stats from all the Models LDA, Logit, Random Forest, Stats Model and also done Smote data upscaling .
2. We arranged data from Descending order of Recall value, Recall denotes how accurately you are predicting True positives.
3. We have got the best Recall values from Stats Model with SMOTE upscaled data , with recall value of 91% on test data set
4. Second highest recall value is for Random forest Model and in this model have note upscaled data and data has been taken as it was given , (Only NULL value and Outliers imputed), it has given us Recall of 87%% on test data, which is very good.
5. Random Forest has also given very good Accuracy of 98% on both train and test data, which shows how good this model is in predicting correct data.
6. Random forest also has very good Precision and F1 Score is also about 90% for both Train and test data set.
7. I will consider Random Forest over Stats Model , even if SM-4 Model have slight better Recall value, because we have not done any data upscaling in Random forest and used data as it was given.

# **Conclusion and Recommendations from the above models:**

1. Sample data had 3586 rows and 67 columns; total number of column values were not bad.
2. We have not given Company types, like Mid-Size, Small Size or Large capital companies.
3. We have not given revenue of the companies , so that we can segregate companies based on their revenue
4. We have not given industry type of companies like, IT, Manufacturing, Retail , Pharma, etc . We consider them all of same type and not taken any action based on what type company it is
5. Data have about 100+ NULL values in sample data, which we imputed by taking median.
6. Data was very large number of outliers , and for our modelling purpose, we treated outliers, using bringing them back to Normal Upper and Lower limits
7. NULL Values and Outlier treatment:

Following methods were used in this exercise for treating NULL and outlier values:

* 1. Impute NULL with Median and Impute Outlier by bringing them back to Normal range. This Method is used for this project submission, because it has the best RECALL and Accuracy score.
  2. I have also tested NULL values and Outlier treatment with KNN method of data imputations, but recall values and precision are very less in this case , so I switch back to above point a only . Following attached is notebook, where I have tested using KNN model as well, but RECALL values are very less. This is just for reference purpose.:



This is a Notebook file, in this Notebook, I have tested KNN treatment and calculated RECALL after KNN treatment of Missing values and Outliers.

1. Data was very imbalanced , and we upscale data using SMOTE feature, and Built Stats model on top of it.

In the Final model , we have used 31 columns for deciding “DEFAULT” nature of companies, and there were total 67 features, were given.

1. We have found best model as Random forest for our analysis with best Recall , Accuracy and Precision on train and test both data set, and for building this model , we have not used any upscaling methodology , so this model is more close to real world scenarios and will not require too much of approval from managements.
2. According to Random forest model, only following features of data , contributing most in Model predictions and their weightage are as follows:

Book\_Value\_Unit\_Curr 0.234250

Networth 0.197338

Book\_Value\_Adj\_Unit\_Curr 0.192101

1. I would also like to Emphasize on following results,

|  | **Accuracy** | **AUC** | **Recall** | **Precision** | **F1 Score** |
| --- | --- | --- | --- | --- | --- |
| **SM 4 Train** | 0.94 | Not applicable | 0.95 | 0.92 | 0.93 |
| **SM 4 Test** | 0.92 | Not applicable | 0.91 | 0.58 | 0.71 |
| **RF Train** | 0.98 | 0.99 | 0.88 | 0.91 | 0.89 |
| **RF Test** | 0.98 | 0.99 | 0.87 | 0.96 | 0.91 |
| **SM 3 Train** | 0.96 | Not applicable | 0.79 | 0.81 | 0.8 |
| **SM 3 Test** | 0.95 | Not applicable | 0.79 | 0.78 | 0.79 |

Stats Model 3, which was created , after removing highly co-related attributes , is also a very good model after random forest, in terms of Recall , Accuracy and Precision model, on Train and test both data set.

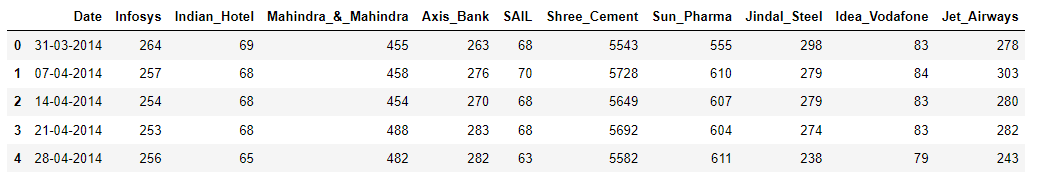
# **Market Risk Analysis: Introduction**

This report explains the business requirements and provide the detailed solution based on the data provided for each problem statement. given in the assignment. Also, the purpose of this exercise is to understand Market risks, given stock prices for different Stock prices for certain Time period, Calculating Mean and standard deviation of Stocks and calculate returns of those Stocks in longer terms.

# **Problem statement :**

The dataset contains 1 year of Stock information about 10 Stocks from Indian stock exchange . It has Stock price listed for all weekdays, whenever Market opens.

This is how the data look like:



Shape of the data:

The number of rows (observations) is 314

The number of columns (variables) is 11

Insights:

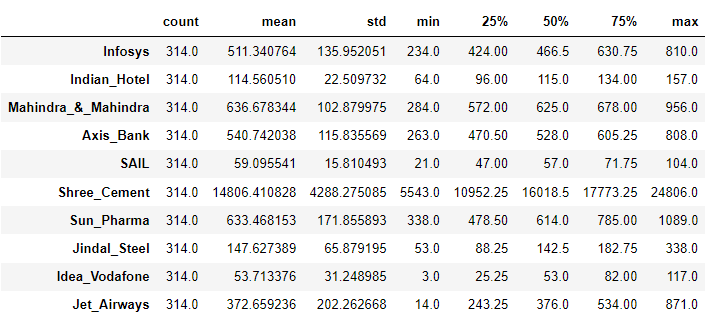
1. There are only 314 Rows (About a year worth of Stock prices , whenever Market opens) in sample, data
2. We have given total 10 Stock prices
3. Stocks given are :

'Infosys', 'Indian\_Hotel', 'Mahindra\_&\_Mahindra', 'Axis\_Bank',

'SAIL', 'Shree\_Cement', 'Sun\_Pharma', 'Jindal\_Steel', 'Idea\_Vodafone',

'Jet\_Airways'

# **Data dictionary :**



#### **Checking data types of all 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\_&\_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

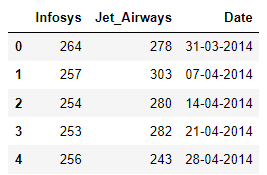
Data is given in correct format for us.

All stock prices are in Numeric format and Date field on which Stock price is captured is in Object format.

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

Lets us plot & see price trend over time for different companies . For the purpose of this exercise, we have asked to do analysis for Any 2 Stocks. For this exercise, we would Consider "Infosys" and "Jet\_Airways" Stocks for plotting Price against time.

This is how data looks like:



Jet airways over the year:

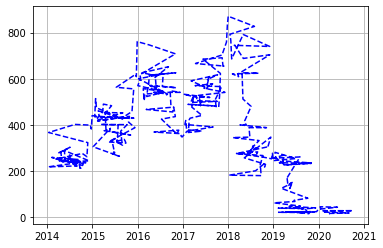


Figure 13 Jet\_Airways Stock LinePlot

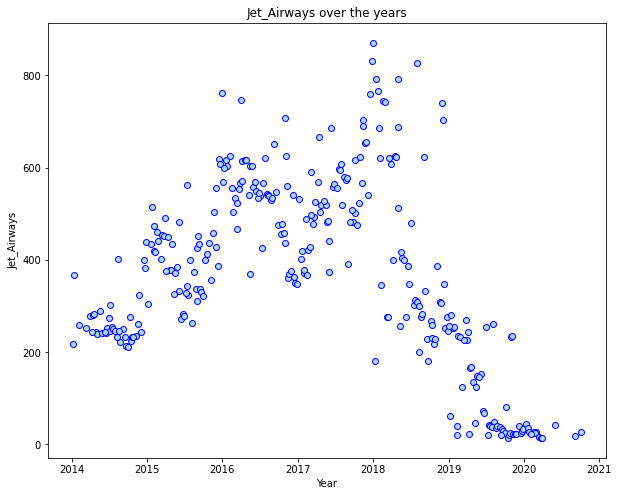


Figure 14 Jet Airways Stock Scatter plot

Insights:

1. JetAirways share grown up in year 2017-18 period,
2. It came down eventually after year 2019 and a big loss for the company,
3. It reached to all time low level from year 2019 to 2021 and collapse completely.

Infosys Stock Prices over the year:

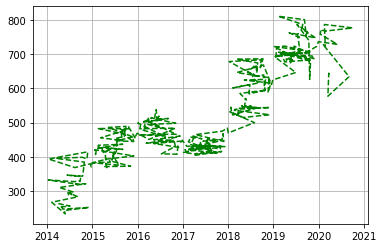


Figure 15 Infosys Stock Line Plot

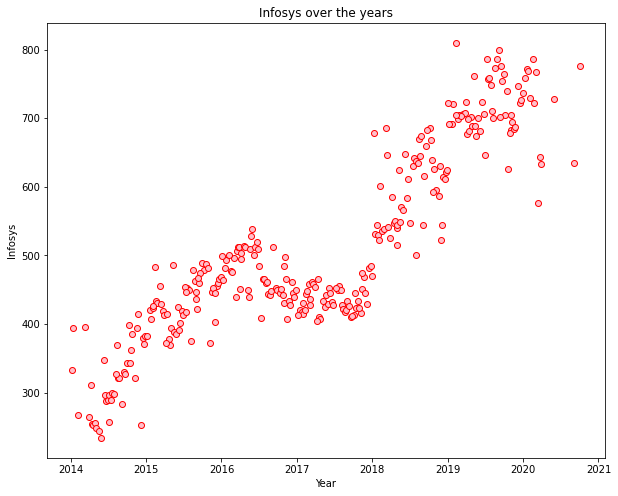


Figure 16 Infosys Stock Scatter Plot

Insights:

1. Infosys Stocks are going high every year,
2. There is always a positive trend in Infosys stocks
3. This is a safe Stock in Long term investment and can be trusted.
4. It is on it’s highest level in year 2019-20

# **Calculate Returns for all stocks with inference**

There are 2 ways we can calculate Returns of the Stocks :

Logarithmic returns

Athematic returns

The difference between a logarithmic and arithmetic chart scale can be seen on the vertical axis, which is the y axis. An arithmetic scale shows equal spacing between the chart units.

A semi-logarithmic scale, on the other hand, is set up to measure price distances in percentage terms. This means a 10% advance from 60 to 66 looks the same as a 10% advance from 100 to 110, even though the first advance is six Dollars and the second advance is ten Dollars.

So which scaling system is best? Both systems have their advantages and disadvantages. One's preference largely depends on analysis style and timeframe. Traders looking to capture short-term price movements or analyze trading ranges may prefer arithmetic scales for price purity. Chartists interested in trends and long-term price histories will likely prefer log scaling. Notice that a log scale is best used when prices have moved a significant amount, up or down. Chartists can switch between arithmetic and log scaling by using the "log scale" check box in the Chart Attributes sector under the SharpChart.

For our exercise purpose, we have chosen Logarithmic returns.

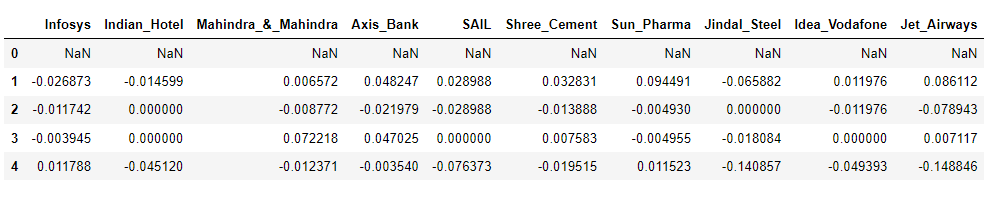
**Steps for calculating returns from prices:**

Take logarithms

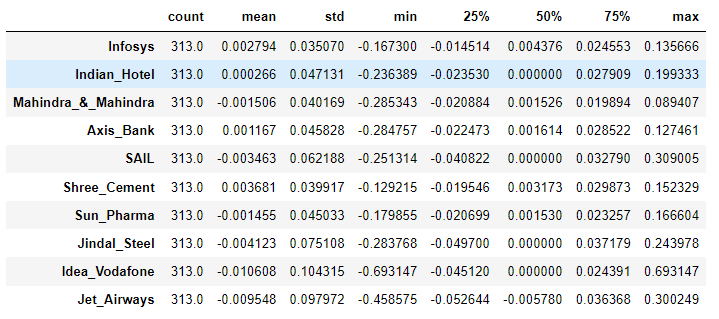
Take differences

Stock Price at time T *minus* Stock price at time (T-1) will give returns of that Stock at T point of time.

We have taken differences of the prices and taken log of the data. This is how the data looks like :



**Describer returns of all the Stocks:**



Insights:

* We can see that maximum returns gained was from ‘Idea Vodafone’ Stocks with 69% growth in any time period.
* We have added a new Row with Total in Stock data frame and calculated in total Returns of each Stock as compared to first value of Stock prices and this is how Total value looks like:



As per above data, we can see that maximum percentage returns from Sample data, given by Shree\_Cement Stock, which game 115% returns , followed by Infosys Stocks, which gave 87% returns.

Least returns given by Idea\_Vodafone, which ran into Loss for 332% Stocks down.

# **Means & Standard Deviations of these returns**

Stock Means: Average returns that the stock is making on a week to week basis

Stock Standard Deviation : It is a measure of volatility meaning the more a stock's returns vary from the stock's average return, the more volatile the stock

**Calculating stock means:**

Infosys 0.002794

Indian\_Hotel 0.000266

Mahindra\_&\_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

**Calculating stock standard deviation**

Infosys 0.035070

Indian\_Hotel 0.047131

Mahindra\_&\_Mahindra 0.040169

Axis\_Bank 0.045828

SAIL 0.062188

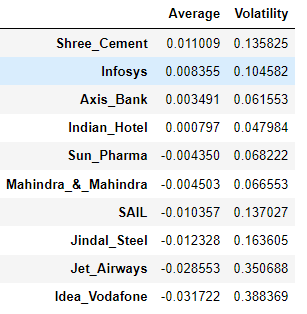
Shree\_Cement 0.039917

Sun\_Pharma 0.045033

Jindal\_Steel 0.075108

Idea\_Vodafone 0.104315

Jet\_Airways 0.097972

**Lets frame both mean and Standard deviation in a data frame and this is how, it look like in a combined manner:**

**Insights:**

* 1. Best Stock from sample data is Shree\_cement, which is giving average returns of 11% from each previous Stock value.
  2. Worst Stock is Idea\_vodafone, which is going down every time and average price decline by 3.1% on an average.
  3. Jet\_airways and Idea\_Vodafone remains the highly volatile Stocks, which are unpredicted with Max Standard deviation of 35-36% each.
  4. Indian Hotels are highly consistent in performance, though their returns are not good .
  5. Similarly Sun\_pharma and Mahindra\_&Mahindra are also least risk volatile Stocks, with consistent performance of about 4.3 to 4.5 % returns on average.

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

We have not given column for Sensex values, so we have calculated Mean for both Average and Volatility. And we will plot each Stock’s Average and Volatility against this mean of all stock’s and see , how that is performing against other Stocks from Sample data.

Mean for Averages of all the stocks from Sample data is : -0.004544117653834371

Mean for Volatility of all the stocks from Sample data is : 0.09334834662321471

Let us now plot each Stock against above means:

We have taken Volatility on X -axis and Average prices on Y axis and seen Volatility/Average of each stocks against means.

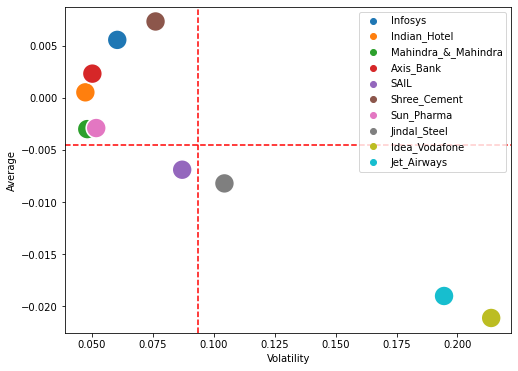


Figure 17Mean and Std Dvt for Stock returns

Insights:

1. We can see that Volatility us very high for Idea\_Vodafone, at the same time, it is giving consistent loss for it’s stocks
2. Same thing High Volatility and least profit with Jet\_airways as well after Idea\_vodafone
3. Jindal\_steel and SAIL are also giving average negative returns as well as High in Volatility.
4. Shree\_cement and Infosys remain the best Stocks for giving very good returns , but little volatile as well. In Long terms, they are always beneficiaries .

# **Conclusion and Recommendations**

1. There are only 314 Rows (About a year worth of Stock prices , whenever Market opens) in sample, data
2. We have given total 10 Stock prices
3. Stocks given are :

'Infosys', 'Indian\_Hotel', 'Mahindra\_&\_Mahindra', 'Axis\_Bank', 'SAIL', 'Shree\_Cement', 'Sun\_Pharma', 'Jindal\_Steel', 'Idea\_Vodafone', 'Jet\_Airways'

1. JetAirways and Idea\_vodafone mong the least performing Stocks and its prices going down always. These Stocks are very risky and Suggestions would be not buy these Stocks, unless there is any Magic happens, acquisition or some other Policy launches. Those are Sinking ships.
2. Infosys and Shree\_cement are the best Stocks, for giving consistent returns and these are Rocket stocks.
3. In Long terms investments, **Infosys** and **Shree\_cements** are best Stocks to purchase, which might give consistent returns of about 8-11% average returns, but they are volatile as well, so these Stocks are a very good suggestion

for **Long term traders**.

1. “**Indian\_Hotel**” is least risky Stocks for Intra-day traders. Because those are least Volatile. But the profit is also very less for it.
2. “Mahindra\_&\_mahindra” stocks are also among least Volatile Stock, but this is not a profit making Stock, average

returns is in **negative 4%**.

1. Axis\_bank is Safe stock with Least Volatility and giving Positive returns of average 2% always, so I would suggest to

buy Axis\_bank stock for **Intra-day traders**.