COMPANY CREDIT RISK ANALYSIS

MILESTONE - 2

SUGANTHE RAMYA M K

Case study

Build a machine learning model, to predict Credit Defaulters from the financial statement of a company

Table of contents

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

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

Project Objective:

The Objective of the report is to explore the dataset "Credit Risk Dataset" in Python (JUPYTER NOTEBOOK) and generate insights about the dataset. This exploration report will consist of the following:

- Importing the dataset in jupyter notebook.
- Understanding the structure of dataset.
- Exploratory Data analysis
- Graphical exploration
- Prediction using various machine learning models
- Insights from the dataset

Read the dataset. Do the descriptive statistics and do the null value condition check. Write an inference on it.

Dataset: Credit Risk Dataset

Column names are changed for further analysis

	Co_Code	Co_Name	Networth_	Equity_Pa	Networth	Capital_Er	Total_Deb	Gross_Blo	Net_Work	Curr_Asse	...	PBIDTM_p	PBITM_pe	PBDTM_p	CPM_perc	APATM_p	Debtors_V	Creditors_	Inventory	Value_of_	Value_of_Output_to_Gross_Blo		
0	16974	Hind.Cable	-8021.6	419.36	-7027.48	-1007.24	5936.03	474.3	-1076.34	40.5	...	0	0	0	0	0	0	0	45	0	0		
1	21214	Tata Tele.	-3986.19	1954.93	-2968.08	4458.2	7410.18	9070.86	-1098.88	486.86	...	-10.3	-39.74	-57.74	-57.74	-87.18	29	101	2	0.31	0.24		

Information on dataset:

<class 'pandas.core.frame.DataFrame'>	
RangeIndex: 3586 entries, 0 to 3585	
Data columns (total 67 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 float	
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 float6	
9 Curr_Assets 3586 non-null float64	
10 Curr_Liab_and_Prov 3586 non-null float6	
11 Total_Assets_to_Liab 3586 non-null float6	
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 float6	
16 Cost_of_Prod 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 Rev_earn_in_forex 3586 non-null float6	
26 Rev_exp_in_forex 3586 non-null float6	
27 Capital_exp_in_forex 3586 non-null float6	
28 Book_Value_Unit_Curr 3586 non-null floa	
29 Book_Value_Adj_Unit_Curr 3582 non-null fl	
30 Market_Capitalisation 3586 non-null float	

31 CEPS_annualised_Unit_Curr 3586 non-null fl
32 Cash_Flow_From_Opr 3586 non-null floa
33 Cash_Flow_From_Inv 3586 non-null floa
34 Cash_Flow_From_Fin 3586 non-null float
35 ROG_Net_Worth_perc 3586 non-null flo
36 ROG_Capital_Employed_perc 3586 non-null
37 ROG_Gross_Block_perc 3586 non-null flo
38 ROG_Gross_Sales_perc 3586 non-null floa
39 ROG_Net_Sales_perc 3586 non-null float
40 ROG_Cost_of_Prod_perc 3586 non-null flo
41 ROG_Total_Assets_perc 3586 non-null flo
42 ROG_PBIDT_perc 3586 non-null float6
43 ROG_PBDT_perc 3586 non-null float6
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_Rev_earn_in_forex_perc 3586 non-null
49 ROG_Rev_exp_in_forex_perc 3586 non-null
50 ROG_Market_Capitalisation_perc 3586 non-null

51 Curr_Ratio_Latest 3585 non-null float64		
52 Fixed_Assets_Ratio_Latest 3585 non-null flo		
53 Inventory_Ratio_Latest 3585 non-null float		
54 Debtors_Ratio_Latest 3585 non-null float		
55 Total_Asset_Turnover_Ratio_Latest 3585 non-nul		
56 Interest_Cover_Ratio_Latest 3585 non-null fl		
57 PBIDTM_perc_Latest 3585 non-null float		
58 PBITM_perc_Latest 3585 non-null float6		
59 PBDTM_perc_Latest 3585 non-null float		
60 CPM_perc_Latest 3585 non-null float6		
61 APATM_perc_Latest 3585 non-null float		
62 Debtors_Vel_Days 3586 non-null int64		
63 Creditors_Vel_Days 3586 non-null int64		
64 Inventory_Vel_Days 3483 non-null float6		
65 Value_of_Output_to_Total_Assets 3586 non-null		
66 Value_of_Output_to_Gross_Block 3586 non-nul		
dtypes: float64(63), int64(3), object(1)		
memory usage: 1.8+ MB		

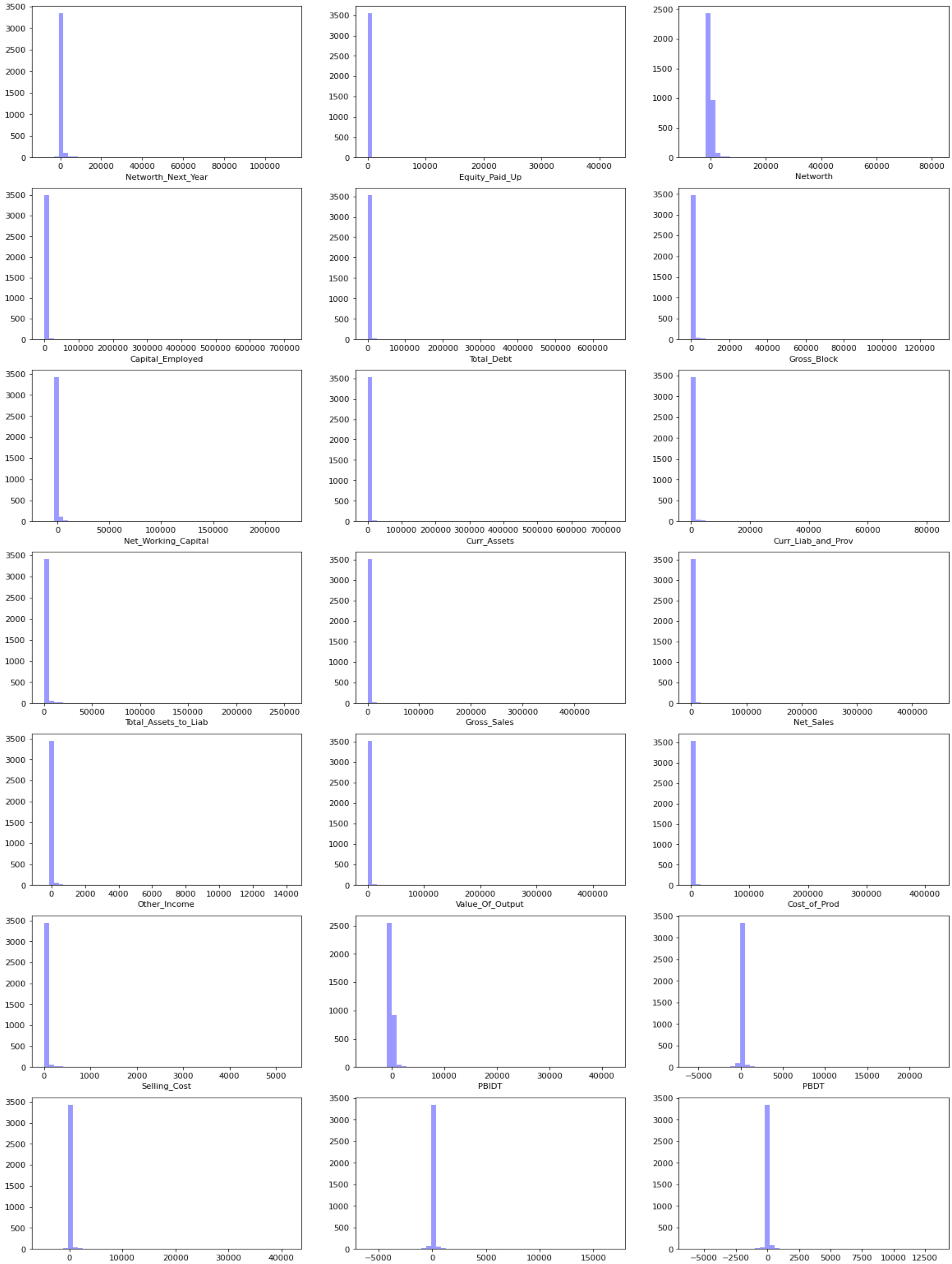
Summary of the dataset:

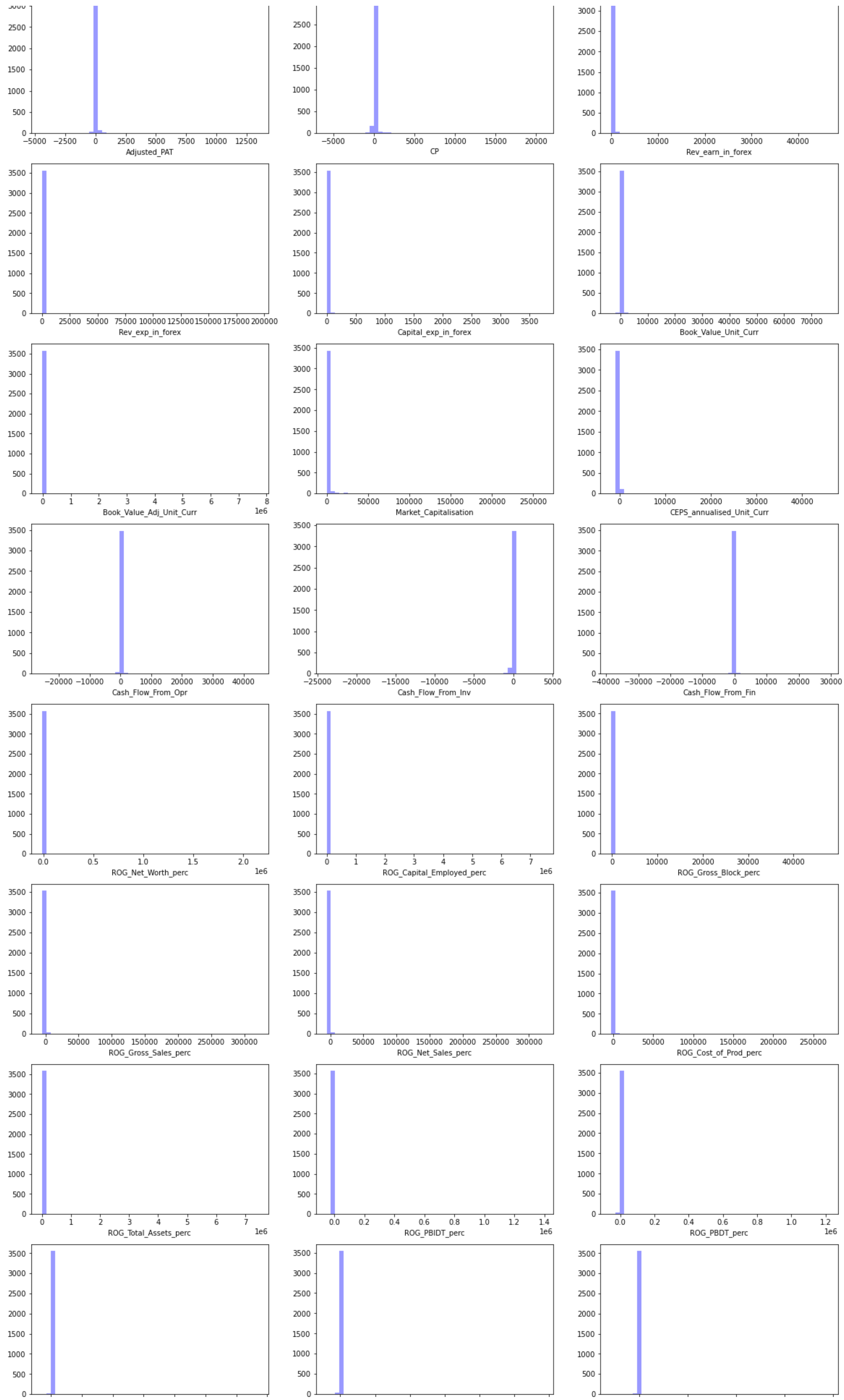
	count	mean	std	min	25%	50%	75%	max
Co_Code	3586	16065.39	19776.82	4	3029.25	6077.5	24269.5	72493
Networth_	3586	725.0453	4769.681	-8021.6	3.985	19.015	123.8025	111729.1
Equity_Pa	3586	62.96658	778.7617	0	3.75	8.29	19.5175	42263.46
Networth	3586	649.7463	4091.989	-7027.48	3.8925	18.58	117.2975	81657.35
Capital_Er	3586	2799.611	26975.14	-1824.75	7.6025	39.09	226.605	714001.3
Total_Deb	3586	1994.824	23652.84	-0.72	0.03	7.49	72.35	652823.8
Gross_Blo	3586	594.1788	4871.548	-41.19	0.57	15.87	131.895	128477.6
Net_Work	3586	410.8097	6301.219	-13162.4	0.9425	10.145	61.175	223257.6
Curr_Asse	3586	1960.349	22577.57	-0.91	4	24.54	135.2775	721166
Curr_Liab_	3586	391.9921	2675.002	-0.23	0.7325	9.225	65.65	83232.98
Total_Asse	3586	1778.454	11437.57	-4.51	10.555	52.01	310.54	254737.2
Gross_Sale	3586	1123.739	10603.7	-62.59	1.4425	31.21	242.25	474182.9
Net_Sales	3586	1079.703	9996.574	-62.59	1.44	30.44	234.44	443775.2
Other_Inc	3586	48.72982	426.0407	-448.72	0.02	0.45	3.635	14143.4
Value_Of_	3586	1077.187	9843.88	-119.1	1.4125	30.895	235.8375	435559.1
Cost_of_P	3586	798.5446	9076.703	-22.65	0.94	25.99	189.55	419913.5
Selling_Co	3586	25.555	194.2445	0	0	0.16	3.8825	5283.91
PBIDT	3586	248.1753	1949.593	-4655.14	0.04	2.045	23.525	42059.26
PBDT	3586	116.2688	956.1996	-5874.53	0	0.795	12.945	23215
PBIT	3586	217.6594	1850.973	-4812.95	0	1.15	16.6675	41402.96
PBT	3586	85.75291	799.9258	-6032.34	-0.06	0.31	7.4225	16798
PAT	3586	61.21831	620.2984	-6032.34	-0.06	0.255	5.54	13383.39
Adjusted_	3586	60.05896	580.4329	-4418.72	-0.09	0.21	5.3425	13384.11
CP	3586	91.7342	780.7906	-5874.53	0	0.74	10.91	20760.2
Rev_earn_	3586	131.1653	1150.73	0	0	0	7.2	46158
Rev_exp_i	3586	256.327	4132.34	0	0	0	6.9875	193979.7
Capital_ex	3586	7.655689	111.4321	0	0	0	0	3722.1
Book_Valu	3586	157.2378	1622.664	-3371.57	7.9625	21.665	71.6675	75790
Book_Valu	3582	2243.153	128283.7	-33715.7	7.06	18.925	60.01	7677600
Market_Ca	3586	1664.092	12805.17	0	0	8.37	111.4575	260865.1
CEPS_ann	3586	36.01871	828.4208	-1808	0	1.145	8.7725	45438.44
Cash_Flow	3586	65.77075	1455.048	-25469.2	-0.3075	0.45	12.6475	44529.4
Cash_Flow	3586	-60.8704	701.9747	-23843.5	-5.1175	-0.12	0.12	3732.98
Cash_Flow	3586	11.43645	1272.257	-38374	-5.8475	0	0.4575	28846
ROG_Net_	3586	1237.625	41041.93	-14485.7	-1.4875	1.84	11.3625	2144020
ROG_Capi	3586	2988.885	126472.9	-8614.63	-3.835	1.375	12.5875	7412700
ROG_Gros	3586	37.55431	893.6194	-116.12	0	0.25	6.72	47400
ROG_Gros	3586	242.673	6103.528	-5503.7	-8.0775	3.31	21.525	320200
ROG_Net_	3586	242.5885	6103.488	-5503.7	-8.1175	3.205	21.5675	320200
ROG_Cost	3586	310.4884	5573.215	-2130.23	-7.2425	4.415	23.1225	267150
ROG_Tota	3586	2793.283	125941.7	-136.13	-3.9725	1.475	12.5	7422120
ROG_PBID	3586	375.8522	23278.4	-52200	-23.3625	4.57	47.875	1386200
ROG_PBD	3586	336.3799	20353.4	-52200	-30.5975	3.365	52.915	1208700
ROG_PBIT	3586	374.7	22462.79	-58500	-31.3525	2.13	50.1425	1338000
ROG_PBT_	3586	224.0702	19659.23	-78900	-41.235	0.025	61.9575	1160500
ROG_PAT_	3586	112.2317	13480.52	-114500	-43.7325	0	65.3475	774200
ROG_CP_p	3586	221.0915	13980.2	-52200	-29.505	4.615	52.9075	822400
ROG_Rev_	3586	37.22784	658.666	-100	0	0	0	29084.77
ROG_Rev_	3586	364.8632	15233.64	-100	0	0	0	894591.7
ROG_Marl	3586	63.68222	1047.928	-98.05	0	0	47.515	61865.26
Curr_Ratic	3585	12.0566	108.4101	0	0.88	1.36	2.77	4813
Fixed_Asse	3585	51.53884	681.1509	0	0.27	1.56	4.74	22172
Inventory_	3585	37.79895	458.1894	0	0	3.56	8.94	15472
Debtors_R	3585	33.027	489.5635	0	0.42	3.82	8.52	22992.67
Total_Asse	3585	1.237236	2.673228	0	0.07	0.6	1.55	57.75
Interest_C	3585	16.38789	351.7378	-5450	0	1.08	3.71	18639.4
PBIDTM_p	3585	-51.1629	1795.131	-78870.5	0	8.07	18.99	19233.33
PBITM_pe	3585	-109.213	3057.636	-141600	0	5.23	14.29	19195.7
PBDTM_p	3585	-311.57	10921.59	-590500	0	4.69	14.11	15640
CPM_perc	3585	-307.006	10676.15	-572000	0	3.89	11.39	15640
APATM_p	3585	-365.056	12500.05	-688600	0	1.59	7.41	15266.67
Debtors_V	3586	603.894	10636.76	0	8	49	106	514721
Creditors_	3586	2057.855	54169.48	0	8	39	89	2034145
Inventory_	3483	79.64456	137.8478	-199	0	35	96	996
Value_of_	3586	0.819757	1.2014	-0.33	0.07	0.48	1.16	17.63
Value_of_	3586	61.88455	976.8244	-61	0.27	1.53	4.91	43404

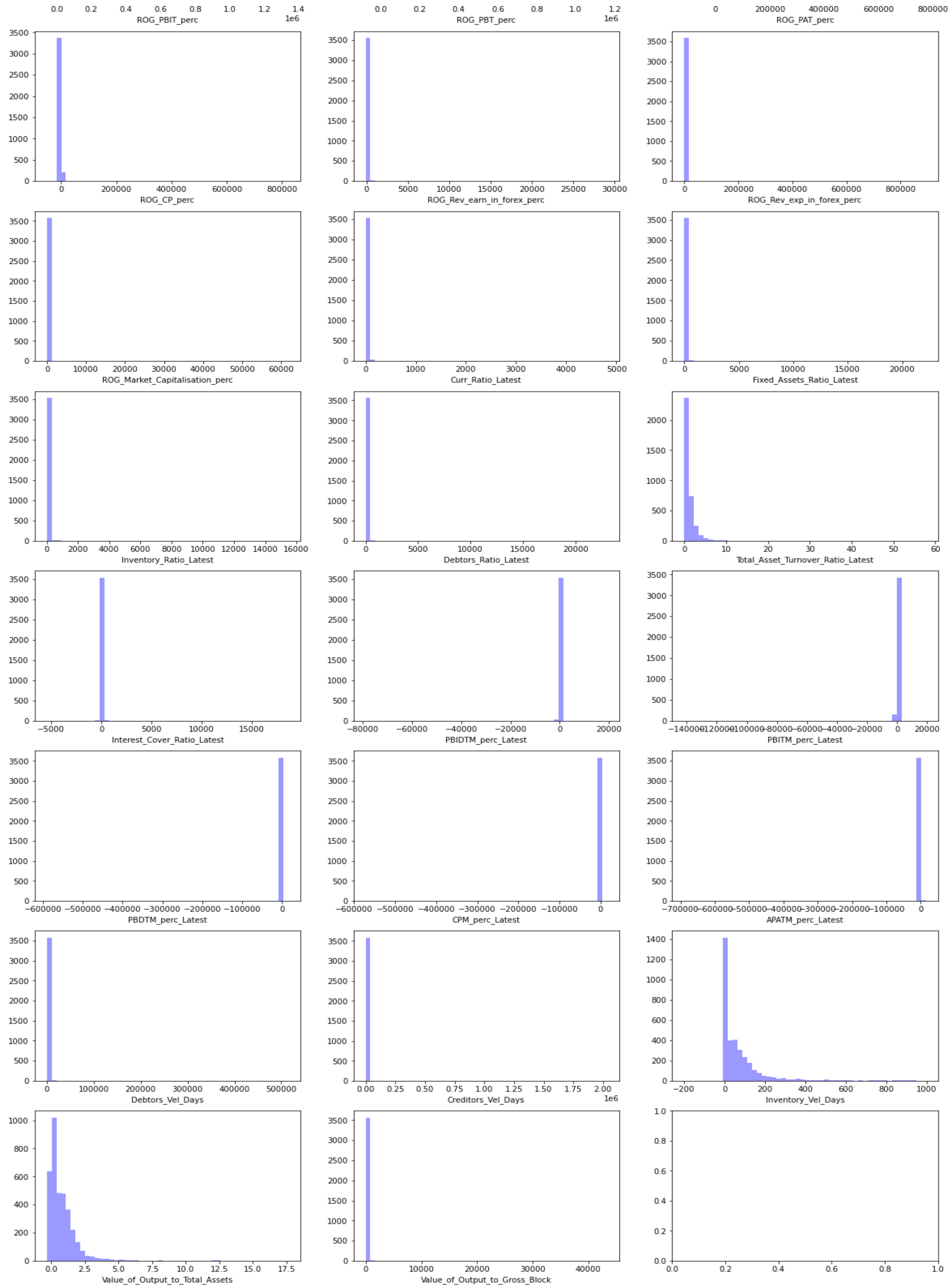
Inference

- The number of rows (observations) is 3586
- The number of columns (variables) is 67
- Minimum Networth_Next_Year - (-8021)
- Maximum Networth_Next_Year - (111729.10)
- Maximum Total Debt - 652823.81
- There are no duplicates in the dataset

Perform Univariate and Bivariate Analysis. Do exploratory data analysis. Check for Outliers.




Adjusted_PAT
CP
Rev_earn_in_forex
Rev_exp_in_forex
Capital_exp_in_forex
Book_Value_Unit_Curr
Book_Value_Adj_Unit_Curr
Market_Capitalisation
CEPS_annualised_Unit_Curr
Cash_Flow_From_Opr
Cash_Flow_From_Inv
Cash_Flow_From_Fin
ROG_Net_Worth_perc
ROG_Capital_Employed_perc
ROG_Gross_Block_perc
ROG_Gross_Sales_perc
ROG_Net_Sales_perc
ROG_Cost_of_Prod_perc
ROG_Total_Assets_perc
ROG_PBIDT_perc
ROG_PBDT_perc

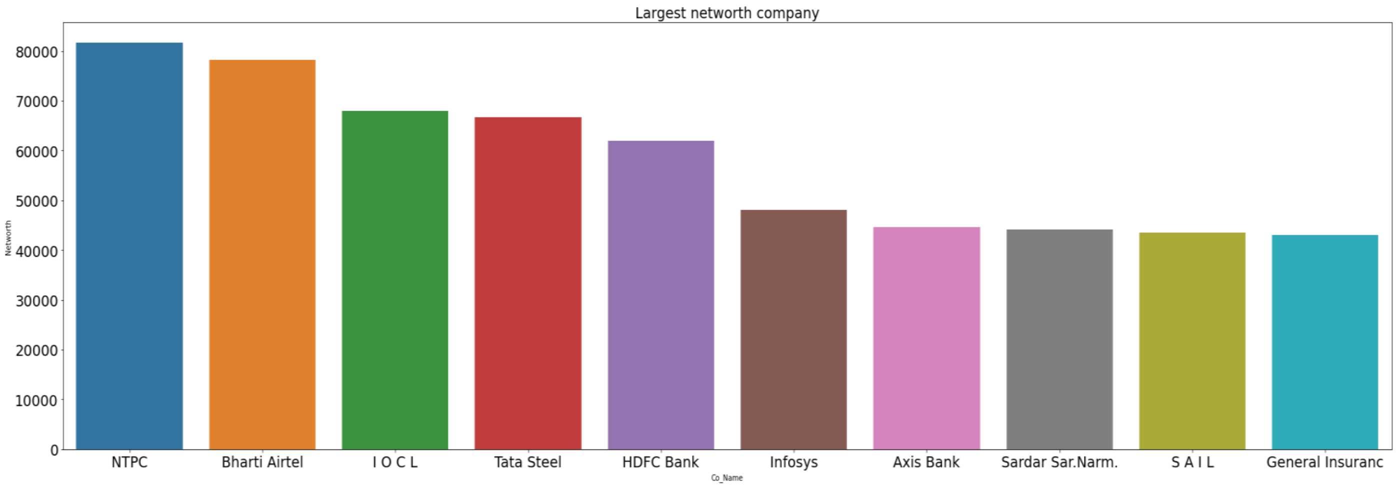


Inference

None of None of the variables show perfect normal distribution. Most of the variables have left positive skewness only six variable right negative skewness

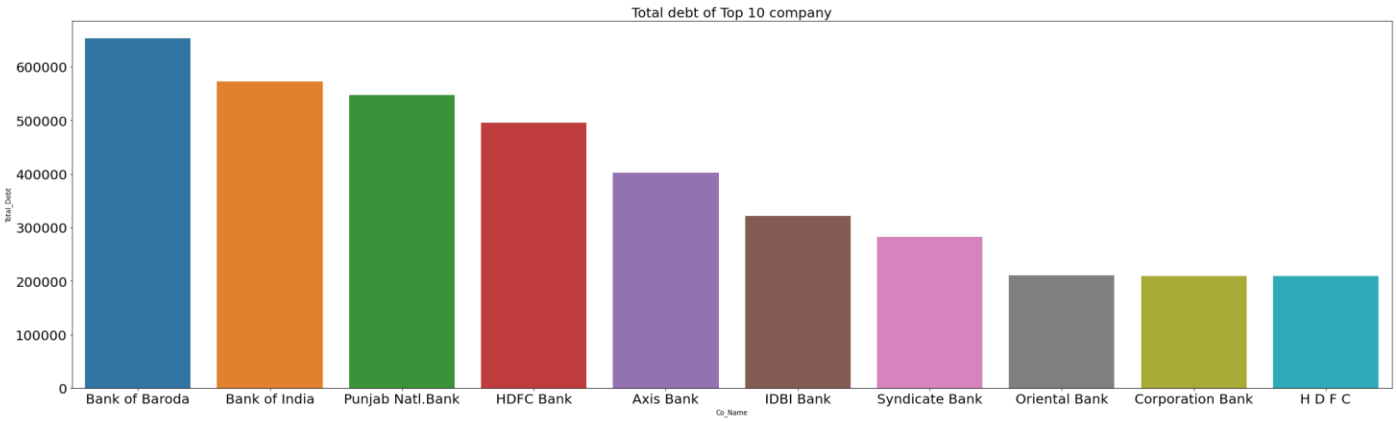
Bi-variate analysis

Largest net worth company



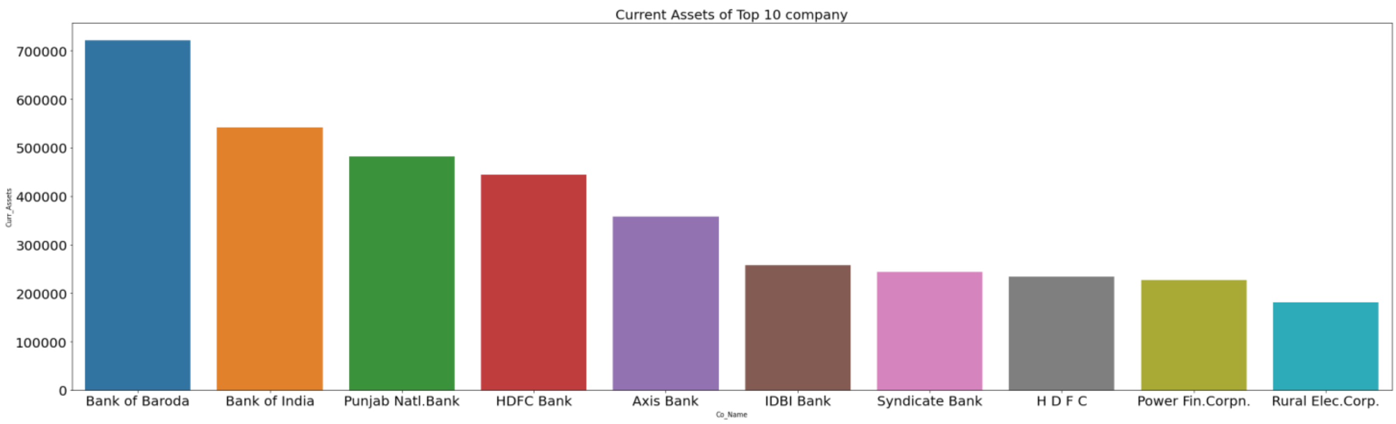
NTPC has highest net worth followed by the Bharti Airtel

Total debt of Top 10 company



Highest debt Company is Bank of Baroda Second Highest is Bank of India

Current Assets of Top 10 company



Highest current assets company is Bank of Baroda Second highest current assets company is Bank of India

Creating a binary target variable using 'Networth_Next_Year'

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.

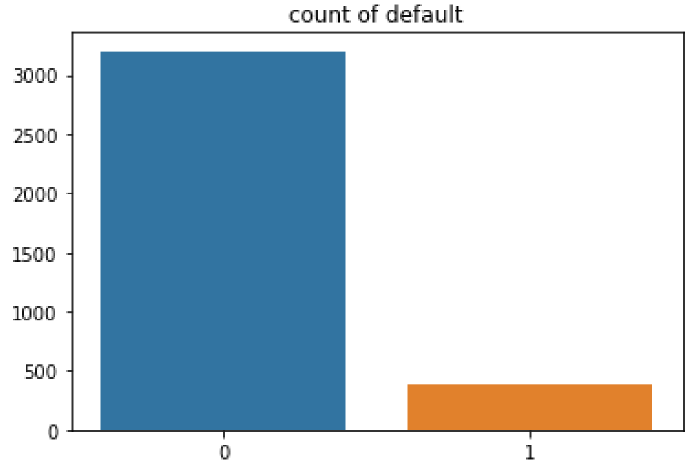
	default	Networth_Next_Year
0	1	-8021.6
1	1	-3986.19
2	1	-3192.58
3	1	-3054.51
4	1	-2967.36
5	1	-2519.4
6	1	-2125.05
7	1	-2100.56
8	1	-1695.75
9	1	-1677.18

Count of default in the dataset

```
0    3198
1     388
Name: default, dtype: int64
```

Percentage of default

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



Inference

- Nearly 11% of defaulters observed and value count of the dataset is 388

Grouping by default:

default	Networth_	Equity_Pa	Networth	Capital_Er	Total_Deb	Gross_Blo	Net_Work	Curr_Asse	Curr_Liab_	Total_Asse	...	PBIDTM_p	PBITM_pe	PBDTM_p	CPM_perc	APATM_p	Debtors_V	Creditors_	Inventory_	Value_of_	Value_of_
0	2667204	214372.5	2373939	9967573	7046659	2059120	1462211	6968271	1355095	6255114	...	-124452	-193784	-452373	-463495	-532827	1983309	5105452	243673	2757.62	220144.7
1	-67191.9	11425.65	-43948.7	71832.61	106779.3	71605.22	10952.31	61541.12	50588.63	122421.2	...	-58966.7	-197746	-664606	-637120	-775899	182255	2274016	33729	182.03	1773.34

Dropping Networth_Next_Year variable for further analysis.

Outlier Treatment

Significant number of outliers were present for almost all the variables.

```
Equity_Paid_Up                  448
Networth                        650
Capital_Employed                596
Total_Debt                      583
Gross_Block                     540
Net_Working_Capital             625
Curr_Assets                     577
Curr_Liab_and_Prov              581
Total_Assets_to_Liab            574
Gross_Sales                     554
Net_Sales                       556
Other_Income                    603
Value_Of_Output                 559
Cost_of_Prod                    560
Selling_Cost                    605
PBIDT                           671
PBDT                            815
PBIT                            720
PBT                             941
PAT                             959
Adjusted_PAT                    954
CP                              816
Rev_earn_in_forex               738
Rev_exp_in_forex                693
Capital_exp_in_forex            694
Book_Value_Unit_Curr            485
Book_Value_Adj_Unit_Curr        486
Market_Capitalisation           639
CEPS_annualised_Unit_Curr       602
Cash_Flow_From_Opr              801
Cash_Flow_From_Inv              876
Cash_Flow_From_Fin             1005
```

```
ROG_Net_Worth_perc                    747
ROG_Capital_Employed_perc             572
ROG_Gross_Block_perc                  830
ROG_Gross_Sales_perc                  671
ROG_Net_Sales_perc                    667
ROG_Cost_of_Prod_perc                 675
ROG_Total_Assets_perc                 483
ROG_PBIDT_perc                        611
ROG_PBDT_perc                         628
ROG_PBIT_perc                         616
ROG_PBT_perc                          611
ROG_PAT_perc                          598
ROG_CP_perc                           637
ROG_Rev_earn_in_forex_perc           1317
ROG_Rev_exp_in_forex_perc            1615
ROG_Market_Capitalisation_perc        497
Curr_Ratio_Latest                     565
Fixed_Assets_Ratio_Latest             495
Inventory_Ratio_Latest                375
Debtors_Ratio_Latest                  371
Total_Asset_Turnover_Ratio_Latest     201
Interest_Cover_Ratio_Latest           725
PBIDTM_perc_Latest                    595
PBITM_perc_Latest                     717
PBDTM_perc_Latest                     695
CPM_perc_Latest                       720
APATM_perc_Latest                     933
Debtors_Vel_Days                      398
Creditors_Vel_Days                    391
Inventory_Vel_Days                    262
Value_of_Output_to_Total_Assets       150
Value_of_Output_to_Gross_Block        481
dtype: int64
```

Since the outliers are too large in the number.it will affect the model. But also given the fact that this is a financial data and the outliers might very well reflect the information which is genuine in nature. Since data captured from different size of companies

Missing values in the dataset

```
Equity_Paid_Up                    0
Networth                          0
Capital_Employed                  0
Total_Debt                        0
Gross_Block                       0
Net_Working_Capital               0
Curr_Assets                       0
Curr_Liab_and_Prov                0
Total_Assets_to_Liab              0
Gross_Sales                       0
Net_Sales                         0
Other_Income                      0
Value_Of_Output                   0
Cost_of_Prod                      0
Selling_Cost                      0
PBIDT                             0
PBDT                              0
PBIT                              0
PBT                               0
PAT                               0
Adjusted_PAT                      0
CP                                0
Rev_earn_in_forex                 0
Rev_exp_in_forex                  0
Capital_exp_in_forex              0
Book_Value_Unit_Curr              0
Book_Value_Adj_Unit_Curr          4
Market_Capitalisation             0
CEPS_annualised_Unit_Curr         0
Cash_Flow_From_Opr                0
Cash_Flow_From_Inv                0
Cash_Flow_From_Fin                0
ROG_Net_Worth_perc                0
ROG_Capital_Employed_perc         0
ROG_Gross_Block_perc              0
ROG_Gross_Sales_perc              0
ROG_Net_Sales_perc                0
ROG_Cost_of_Prod_perc             0
ROG_Total_Assets_perc             0
ROG_PBIDT_perc                    0
ROG_PBDT_perc                     0
ROG_PBIT_perc                     0
ROG_PBT_perc                      0
ROG_PAT_perc                      0
ROG_CP_perc                       0
ROG_Rev_earn_in_forex_perc        0
ROG_Rev_exp_in_forex_perc         0
ROG_Market_Capitalisation_perc    0
Curr_Ratio_Latest                 1
Fixed_Assets_Ratio_Latest         1
Inventory_Ratio_Latest            1
Debtors_Ratio_Latest              1
Total_Asset_Turnover_Ratio_Latest 1
Interest_Cover_Ratio_Latest       1
PBIDTM_perc_Latest                1
PBITM_perc_Latest                 1
PBDTM_perc_Latest                 1
CPM_perc_Latest                   1
APATM_perc_Latest                 1
Debtors_Vel_Days                  0
Creditors_Vel_Days                0
Inventory_Vel_Days              103
Value_of_Output_to_Total_Assets   0
Value_of_Output_to_Gross_Block    0
default                           0
dtype: int64
```

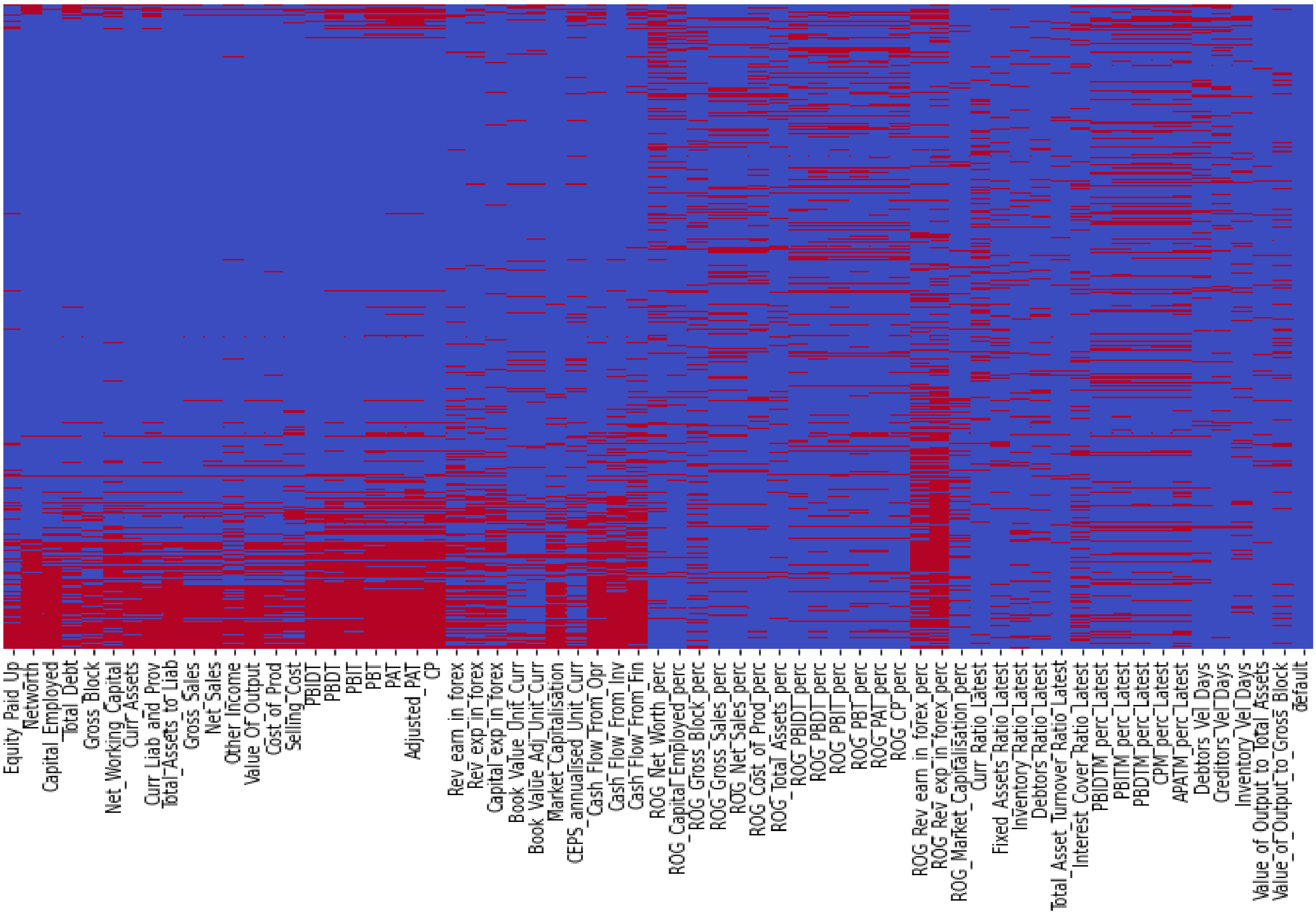
Size of the dataset = 233090

There are 118 missing values

Although most outliers have nan values which is a missing data which should be treated with missing data imputation method here KNN imputation method is used

Sum of the missing values = 41473

Visual inspect the missing values in our data



We should inspect total missing values by each row.

```
0       19
1       34
2       43
3       36
4       35
        ..
3581    30
3582    36
3583    34
3584    30
3585    36
Length: 3586, dtype: int64
```

If we consider availability of features for deciding the observations to be considered, we will end up losing more than 90% of the actual defaulters.so, Dropping columns with more than 30% missing values

Dropping variables like ROG_Rev_exp_in_forex_perc. ROG_Rev_earn_in_forex_perc which has more than 30% missing values

Segregate the predictors and response variables

Predictors:

All independent variables except default variable

Response variable

Dependent variable – Default variable which has binary variable 0 and 1

Scale the predictors

It can also be a good idea to scale the target variable for regression predictive modelling problems to make the problem easier to target variable with a large spread of values, in turn, may result in large error gradient values causing weight values to change dramatically, making the learning process unstable.

Scaling variables is a critical step in regression

Here StandardScaler is used for pre-processing the data. We will use the default configuration and scale values to subtract the mean to centre them on 0.0 and divide by the standard deviation to give the standard deviation of 1.0. First, a StandardScaler instance is defined with default hyperparameters.

Once defined, we can call the fit_transform() function and pass it to our dataset to create a transformed version of our dataset.

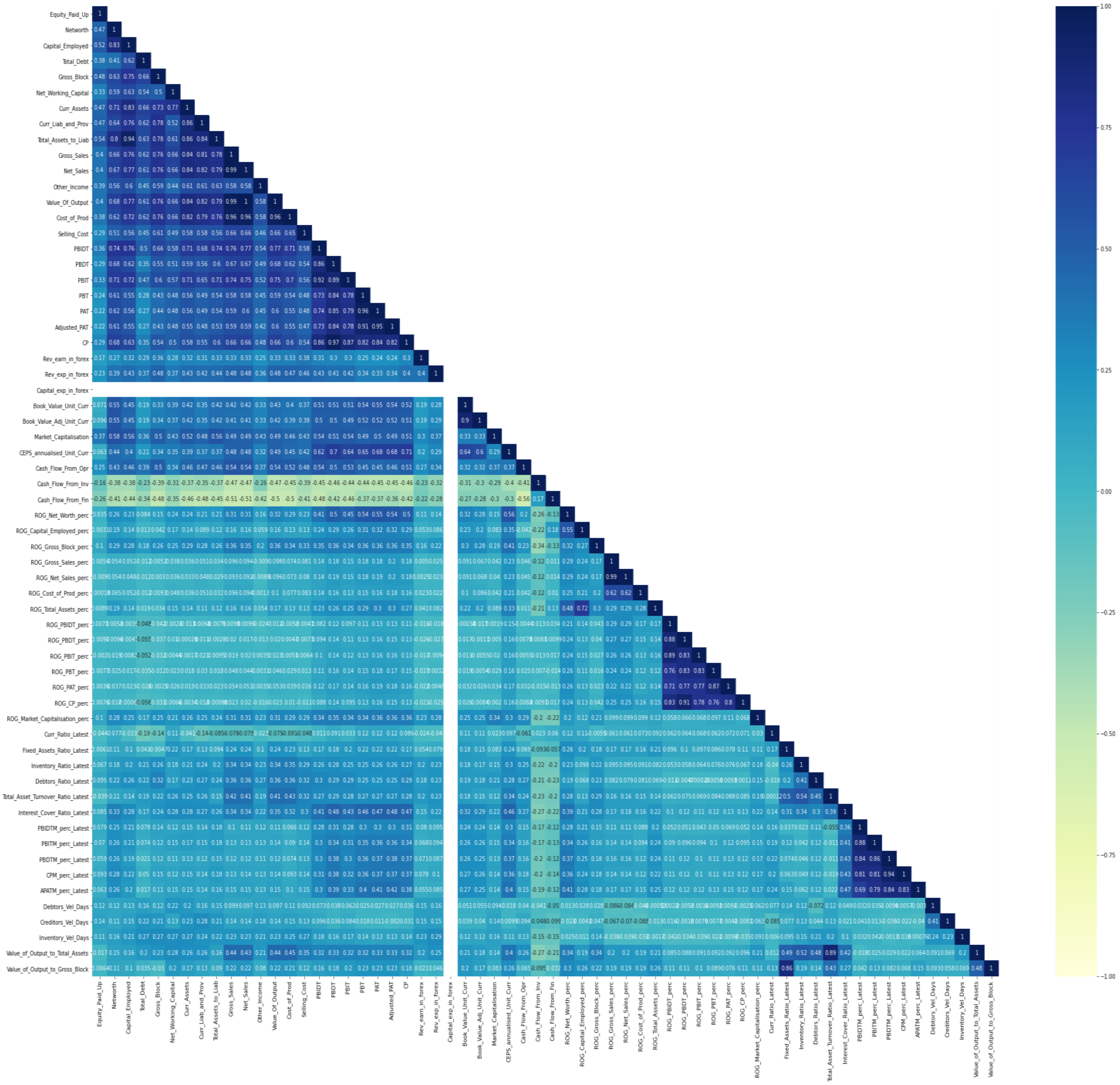
Imputing the missing values
KNNImputer

```
Equity_Paid_Up                 0
Networth                       0
Capital_Employed               0
Total_Debt                     0
Gross_Block                    0
Net_Working_Capital            0
Curr_Assets                    0
Curr_Liab_and_Prov             0
Total_Assets_to_Liab           0
Gross_Sales                    0
Net_Sales                      0
Other_Income                   0
Value_Of_Output                0
Cost_of_Prod                   0
Selling_Cost                   0
PBIDT                          0
PBDT                           0
PBIT                           0
PBT                            0
PAT                            0
Adjusted_PAT                   0
CP                             0
Rev_earn_in_forex              0
Rev_exp_in_forex               0
Capital_exp_in_forex           0
Book_Value_Unit_Curr           0
Book_Value_Adj_Unit_Curr       0
Market_Capitalisation              0
CEPS_annualised_Unit_Curr          0
Cash_Flow_From_Opr                 0
Cash_Flow_From_Inv                 0
Cash_Flow_From_Fin                 0
ROG_Net_Worth_perc                 0
ROG_Capital_Employed_perc          0
ROG_Gross_Block_perc               0
ROG_Gross_Sales_perc               0
ROG_Net_Sales_perc                 0
ROG_Cost_of_Prod_perc              0
ROG_Total_Assets_perc              0
ROG_PBIDT_perc                     0
ROG_PBDT_perc                      0
ROG_PBIT_perc                      0
ROG_PBT_perc                       0
ROG_PAT_perc                       0
ROG_CP_perc                        0
ROG_Market_Capitalisation_perc     0
Curr_Ratio_Latest                  0
Fixed_Assets_Ratio_Latest          0
Inventory_Ratio_Latest             0
Debtors_Ratio_Latest               0
Total_Asset_Turnover_Ratio_Latest  0
Interest_Cover_Ratio_Latest        0
PBIDTM_perc_Latest                 0
PBITM_perc_Latest                  0
PBDTM_perc_Latest                  0
CPM_perc_Latest                    0
APATM_perc_Latest                  0
Debtors_Vel_Days                   0
Creditors_Vel_Days                 0
Inventory_Vel_Days                 0
Value_of_Output_to_Total_Assets    0
Value_of_Output_to_Gross_Block     0
default                            0
dtype: int64
```

Data is scaled and pre-processed before imputing missing data using KNN Imputer. Imputation for completing missing values using k-Nearest Neighbours.

Each sample’s missing values are imputed using the mean value from n_neighbors nearest neighbours found in the training set. Two samples are close if the features that neither is missing are close.

Inspect possible correlations between independent variables



Some of the variable is high positively correlated and some of the variables are slightly negative correlated

Splitting the data into train and test sets

Test Train Split - Split the data into Train and Test dataset in a ratio of 67:33 and use random_state =42

```
The training set for the independent variables: (2402, 62)
The training set for the dependent variable: (2402,)
The test set for the independent variables: (1184, 62)
The test set for the dependent variable: (1184,)
```

Model using logistic regression

Logistic regression is one of the most important models for categorical response data. It is an example of a generalized linear model whose main use is to estimate the probability that a binary response occurs based on several predictor variables.

Recursive Feature Elimination, or RFE for short, is a popular feature selection algorithm.

RFE is popular because it is easy to configure and use and because it is effective at selecting those features (columns) in a training dataset that are more or most relevant in predicting the target variable. RFE is a wrapper-style feature selection algorithm that also uses filter-based feature selection internally. RFE works by searching for a subset of features by starting with all features in the training dataset and successfully removing features until the desired number remains.

This is achieved by fitting the given machine learning algorithm used in the core of the model, ranking features by importance, discarding the least important features, and re-fitting the model. This process is repeated until a specified number of features remains.

For modelling we will use Logistic Regression with recursive feature elimination

Important features found by RFE

Features are selected based on rank. Here we have taken 15 features with rank 1

Feature	Rank	
1	Networth	1
2	Capital_Employed	1
4	Gross_Block	1
7	Curr_Liab_and_Prov	1
8	Total_Assets_to_Liab	1
12	Value_Of_Output	1
13	Cost_of_Prod	1
15	PBIDT	1
17	PBIT	1
25	Book_Value_Unit_Curr	1
26	Book_Value_Adj_Unit_Curr	1
32	ROG_Net_Worth_perc	1
33	ROG_Capital_Employed_perc	1
46	Curr_Ratio_Latest	1
51	Interest_Cover_Ratio_Latest	1

These 15 features are used for stats model Logistic regression model

Fit the model to the training set

We now fit our model to the GridSearchCV for Logistic Regression model by training the model with our independent variable and dependent variables.

Inference

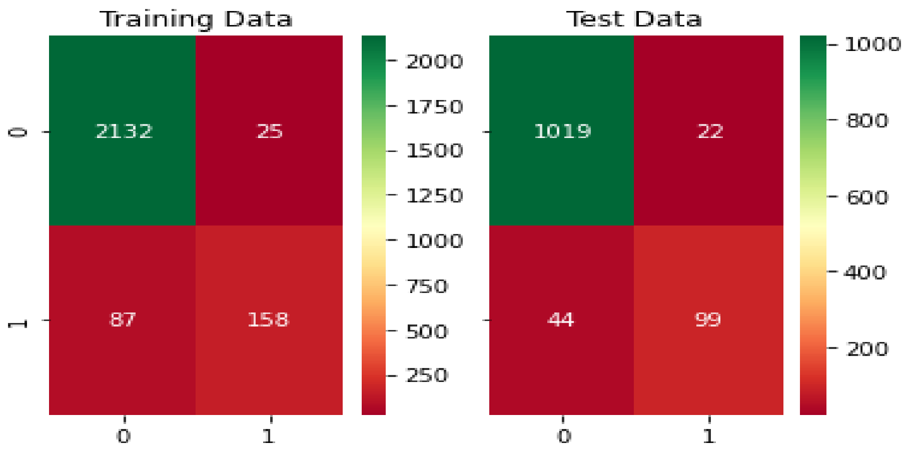
Using GridsearchCV, we input various parameters like 'max_iter', 'penalty',solver', 'tol' which will helps us to find best grid for prediction of the better model

max_iter is an integer (100 by default) that defines the maximum number of iterations by the solver during model fitting.

solver is a string ('liblinear' by default) that decides what solver to use for fitting the model. Other options are 'newton-cg', 'lbfgs', 'sag', and 'saga'.

```
Best grid: {'max_iter': 1000, 'penalty': 'l2', 'solver': 'saga', 'tol': 1e-05}
```

Confusion matrix on the training and test data



Inference

Training data:

True Negative : 2132 False Positive : 25

False Negative : 87 True Positive : 158

Test data:

True Negative : 1019 False Positive :22

False Negative : 44 True Positive : 99

Classification Report of training and test data

Training data

	precision	recall	f1-score	support
0.0	0.96	0.99	0.97	2157
1.0	0.86	0.64	0.74	245
accuracy			0.95	2402
macro avg	0.91	0.82	0.86	2402
weighted avg	0.95	0.95	0.95	2402

Test data

	precision	recall	f1-score	support
0.0	0.96	0.98	0.97	1041
1.0	0.82	0.69	0.75	143
accuracy			0.94	1184
macro avg	0.89	0.84	0.86	1184
weighted avg	0.94	0.94	0.94	1184

Inference

Train Data:

- Accuracy: 95%
- precision : 86%
- recall : 64%
- f1 :95%

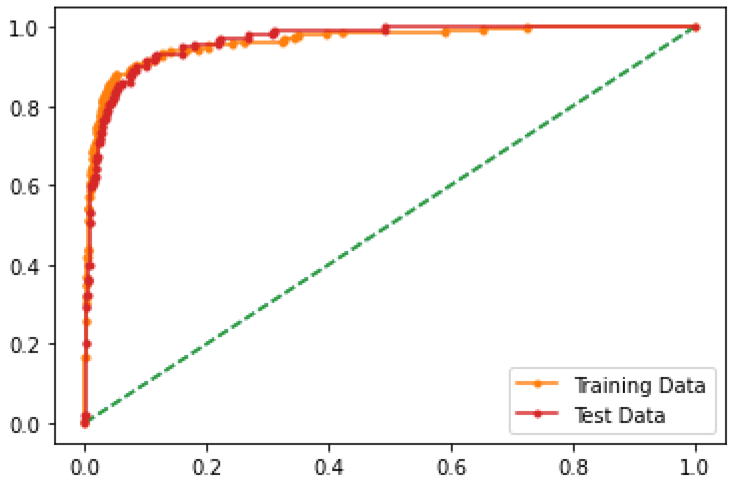
Test Data:

- Accuracy: 95%
- precision: 82%
- recall:69%
- f1:75%

AUC and ROC for the training data and test data

AUC for the Training Data: 0.965

AUC for the Test Data: 0.966



Stats model Logistic regression modelling

Statsmodels is a Python module which provides various functions for estimating different statistical models and performing statistical tests

First, we define the set of dependent(**y**) and independent(**X**) variables. If the dependent variable is in non-numeric form, it is first converted to numeric using dummies. The file used in the example for training the model

Statsmodels provides a **Logit()** function for performing logistic regression. The *Logit()* function accepts **y** and **X** as parameters and returns the *Logit* object. The model is then fitted to the data.

Logit Regression Results

Dep. Variable:	default	**No. Observations:**	2402
Model:	Logit	**Df Residuals:**	2386
Method:	MLE	**Df Model:**	15
Date:	Tue, 10 Aug 2021	**Pseudo R-squ.:**	0.5863
Time:	13:28:48	**Log-Likelihood:**	-327.37
converged:	True	**LL-Null:**	-791.34
Covariance Type:	nonrobust	**LLR p-value:**	3.686e-188

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	-5.2239	0.292	-17.872	0.000	-5.797	-4.651
Networth	-1.5555	0.334	-4.664	0.000	-2.209	-0.902
Capital_Employed	-0.7493	0.309	-2.424	0.015	-1.355	-0.143
Gross_Block	0.8500	0.228	3.733	0.000	0.404	1.296
Curr_Liab_and_Prov	0.7379	0.236	3.125	0.002	0.275	1.201
Total_Assets_to_Liab	0.7680	0.306	2.509	0.012	0.168	1.368
Value_Of_Output	-1.8154	0.552	-3.290	0.001	-2.897	-0.734
Cost_of_Prod	1.6849	0.489	3.447	0.001	0.727	2.643
PBIDT	-1.2197	0.257	-4.745	0.000	-1.724	-0.716
PBIT	0.9219	0.251	3.670	0.000	0.430	1.414
Book_Value_Unit_Curr	-2.0100	0.544	-3.693	0.000	-3.077	-0.943
Book_Value_Adj_Unit_Curr	-1.5899	0.539	-2.950	0.003	-2.646	-0.533
ROG_Net_Worth_perc	-0.5607	0.149	-3.768	0.000	-0.852	-0.269
ROG_Capital_Employed_perc	0.4830	0.132	3.672	0.000	0.225	0.741
Curr_Ratio_Latest	-1.0811	0.163	-6.639	0.000	-1.400	-0.762
Interest_Cover_Ratio_Latest	-0.7117	0.167	-4.265	0.000	-1.039	-0.385

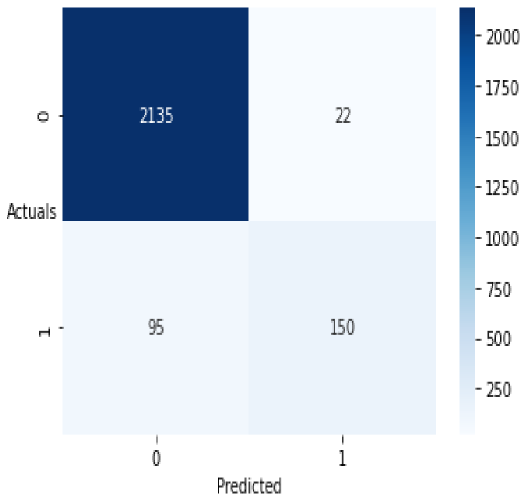
Inference:

- The sign of a regression coefficient tells you whether there is a positive or negative correlation between each independent variable the dependent variable. A positive coefficient indicates that as the value of the independent variable increases, the mean of the dependent variable also tends to increase. A negative coefficient suggests that as the independent variable increases, the dependent variable tends to decrease.
- Gross_Block, Curr_Liab_and_Prov, Total_Assets_to_Liab, Cost_of_Prod, ROG_Capital_Employed_perc has positive coefficients. When these features increase Credit Score also increases.
- Other features have negative coefficients. When these features increases then Credit score is decreases.

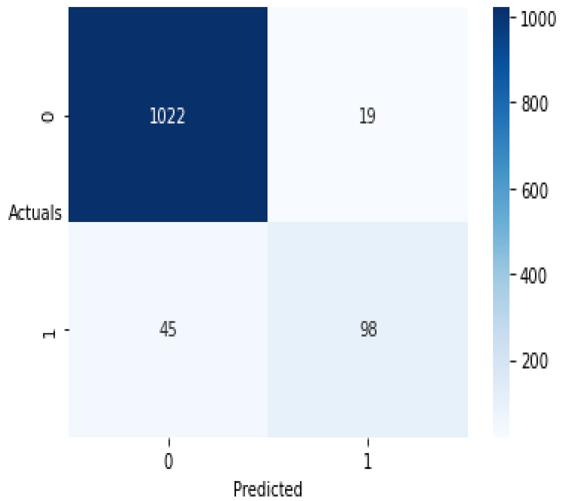
- The parameter estimates table summarizes the effect of each predictor.
- The ratio of the coefficient to its standard error, squared, equals the Wald statistic.
- If the significance level of the Wald statistic is small (less than 0.05) then the parameter is useful to the model.
- The predictors and coefficient Values shown in the last steps are used by the procedure to make predictions.

Confusion matrix on the training and test data

Training data



Test data



Inference

Training data:

True Negative : 2135 False Positive : 22

False Negative : 95 True Positive : 150

Test data:

True Negative : 1022 False Positive : 19

False Negative : 45 True Positive : 98

Classification Report of training and test data

Training data

	precision	recall	f1-score	support
0.0	0.96	0.99	0.97	2157
1.0	0.87	0.61	0.72	245
accuracy			0.95	2402
macro avg	0.91	0.80	0.85	2402
weighted avg	0.95	0.95	0.95	2402

Test data

	precision	recall	f1-score	support
0.0	0.96	0.98	0.97	1041
1.0	0.84	0.69	0.75	143
accuracy			0.95	1184
macro avg	0.90	0.83	0.86	1184
weighted avg	0.94	0.95	0.94	1184

Inference

Train Data:

- Accuracy: 95%
- precision : 87%
- recall : 61%
- f1 :72%

Test Data:

- Accuracy: 95%
- precision: 84%
- recall : 69%
- f1 : 75%

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

Using Scikit_Learn RandomisedSearchCV method, we can define a grid of hyperparameter ranges and randomly sample from the grid, performing K-Fold CV with each combination of values

Fit the model to the training set

We now fit our model to the GridSearchCV for Random Forest model by training the model with our independent variable and dependent variables

- n_estimators = number of trees in the forest
- max_features = max number of features considered for splitting a node
- max_depth = max number of levels in each decision tree
- min_samples_split = min number of data points placed in a node before the node is split
- min_samples_leaf = min number of data points allowed in a leaf node

```
Best grid: {'max_depth': 15,
 'min_samples_leaf': 20,
 'min_samples_split': 100,
 'n_estimators': 701}
```

The probabilities on the training set

	0	1
0	0.98	0.02
1	0.99	0.01
2	0.93	0.07
3	0.95	0.05
4	1.00	0.00

The probabilities on the test set

	0	1
0	0.99	0.01
1	0.97	0.03
2	0.86	0.14
3	0.20	0.80
4	0.93	0.07

Feature importance

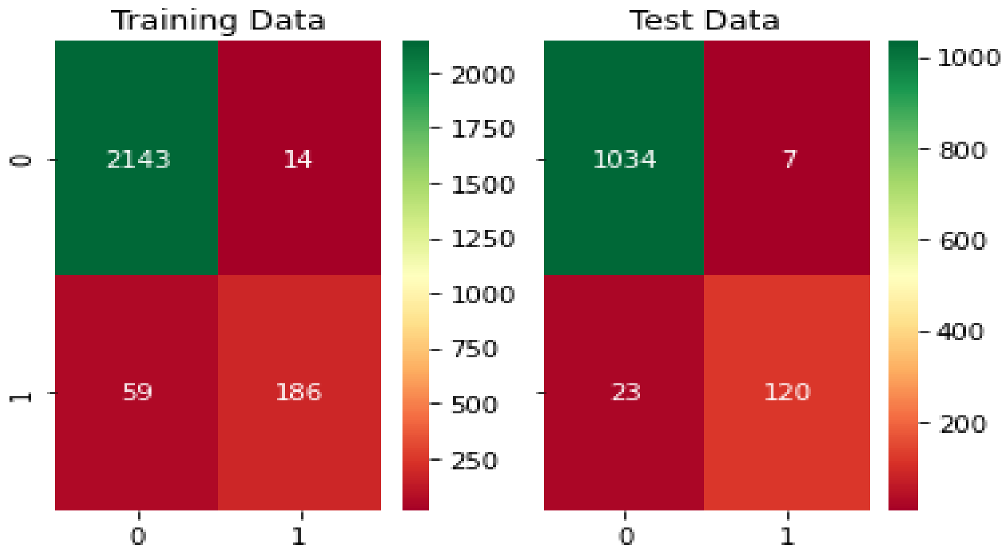
```
IMP
Book_Value_Unit_Curr                 0.23
Networth                             0.21
Book_Value_Adj_Unit_Curr             0.17
Curr_Ratio_Latest                    0.06
Capital_Employed                     0.05
PBIDT                                0.03
CEPS_annualised_Unit_Curr            0.02
CP                                   0.02
PBDT                                 0.02
Net_Working_Capital                  0.02
Total_Asset_Turnover_Ratio_Latest    0.02
Adjusted_PAT                         0.02
PBIT                                 0.01
Interest_Cover_Ratio_Latest          0.01
PAT                                  0.01
ROG_Net_Worth_perc                   0.01
Total_Debt                           0.01
PBITM_perc_Latest                    0.01
PBT                                  0.01
Total_Assets_to_Liab                 0.00
PBIDTM_perc_Latest                   0.00
PBDTM_perc_Latest                    0.00
CPM_perc_Latest                      0.00
APATM_perc_Latest                    0.00
Value_of_Output_to_Gross_Block       0.00
Curr_Assets                          0.00
Fixed_Assets_Ratio_Latest            0.00
Gross_Block                          0.00
Cost_of_Prod                         0.00
ROG_Cost_of_Prod_perc                0.00
Creditors_Vel_Days                   0.00
```

```
Creditors_Vel_Days                0.00
Curr_Liab_and_Prov                0.00
ROG_Total_Assets_perc             0.00
Selling_Cost                      0.00
Value_of_Output_to_Total_Assets   0.00
Net_Sales                         0.00
Value_Of_Output                   0.00
Cash_Flow_From_Inv                0.00
Equity_Paid_Up                    0.00
Gross_Sales                       0.00
Debtors_Vel_Days                  0.00
Market_Capitalisation             0.00
Inventory_Vel_Days                0.00
ROG_Capital_Employed_perc         0.00
ROG_Gross_Sales_perc              0.00
ROG_CP_perc                       0.00
Other_Income                      0.00
ROG_Net_Sales_perc                0.00
Rev_exp_in_forex                  0.00
ROG_PBIDT_perc                    0.00
Cash_Flow_From_Opr                0.00
Debtors_Ratio_Latest              0.00
Inventory_Ratio_Latest            0.00
ROG_PBIT_perc                     0.00
ROG_Gross_Block_perc              0.00
ROG_PBDT_perc                     0.00
ROG_Market_Capitalisation_perc    0.00
ROG_PAT_perc                      0.00
ROG_PBT_perc                      0.00
Cash_Flow_From_Fin                0.00
Rev_earn_in_forex                 0.00
Capital_exp_in_forex              0.00
```

Book_Value_Unit_Curr, Networth, Book_Value_Adj_Unit_Curr are most important features

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

Confusion matrix on the training and test data



Inference

Training data:

True Negative : 2143 False Positive : 14

False Negative : 59 True Positive : 186

Test data:

True Negative : 1034 False Positive : 7

False Negative : 23 True Positive : 120

Classification Report of training and test data

Training data

	precision	recall	f1-score	support
0.0	0.97	0.99	0.98	2157
1.0	0.93	0.76	0.84	245
accuracy			0.97	2402
macro avg	0.95	0.88	0.91	2402
weighted avg	0.97	0.97	0.97	2402

Test Data:

	precision	recall	f1-score	support
0.0	0.98	0.99	0.99	1041
1.0	0.94	0.84	0.89	143
accuracy			0.97	1184
macro avg	0.96	0.92	0.94	1184
weighted avg	0.97	0.97	0.97	1184

Inference

Train Data:

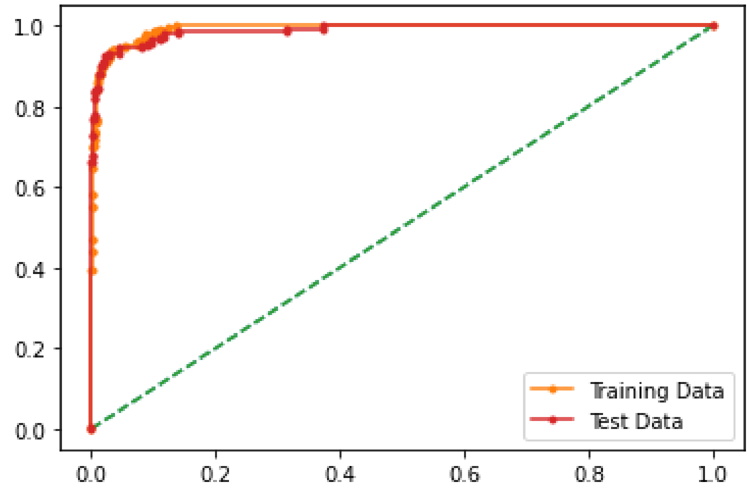
- Accuracy: 97%
- precision : 93%
- recall : 76%
- f1 :84%

Test Data:

- Accuracy: 97%
- precision: 94%
- recall : 84%
- f1 : 89%

AUC and ROC for the training data and test data

```
AUC for the Training Data: 0.991
AUC for the Test Data: 0.988
```



Here, recall has increased to 84% from 76% in test data even F1 Score is also increased to 89% with precision of 94%. It is good model

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

- Linear Discriminant Analysis (LDA) is a dimensionality reduction technique which is commonly used for the supervised classification problems.
- It is used for modeling 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.
- library used in LDA is sklearn
- Using GridsearchCV, we input various parameters like 'max_iter', 'penalty',solver', 'tol' which will he lps us to find best grid for prediction of the better model

The probabilities on the training set

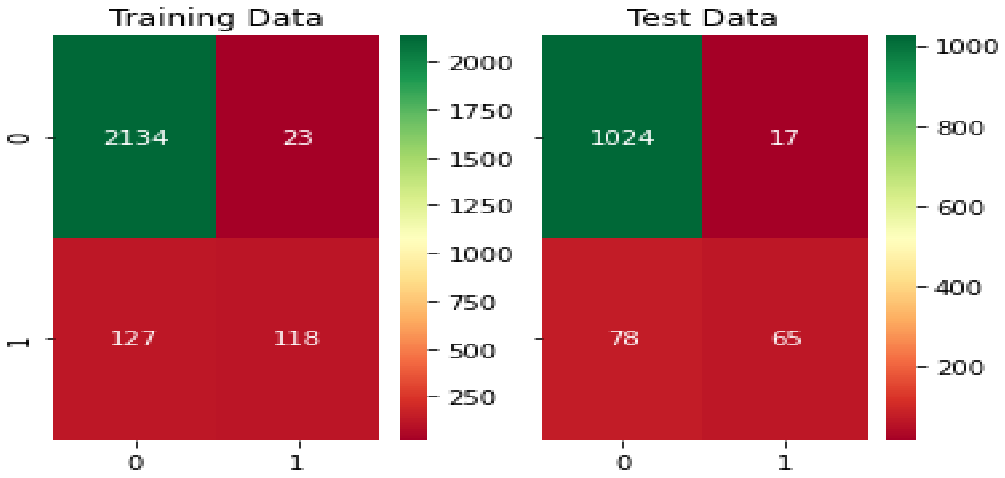
	0	1
0	0.96	0.04
1	1.00	0.00
2	0.55	0.45
3	1.00	0.00
4	1.00	0.00

The probabilities on the test set

	0	1
0	0.98	0.02
1	0.95	0.05
2	0.86	0.14
3	0.92	0.08
4	1.00	0.00

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

Confusion matrix on the training and test data



Inference

Training data:
True Negative : 2134 False Positive : 23
False Negative : 127 True Positive : 118
Test data:
True Negative : 1024 False Positive : 17
False Negative : 78 True Positive : 65

Classification Report of training and test data

```
              precision    recall  f1-score   support

         0.0       0.94      0.99      0.97      2157
         1.0       0.84      0.48      0.61       245

    accuracy                           0.94      2402
   macro avg       0.89      0.74      0.79      2402
weighted avg       0.93      0.94      0.93      2402
```

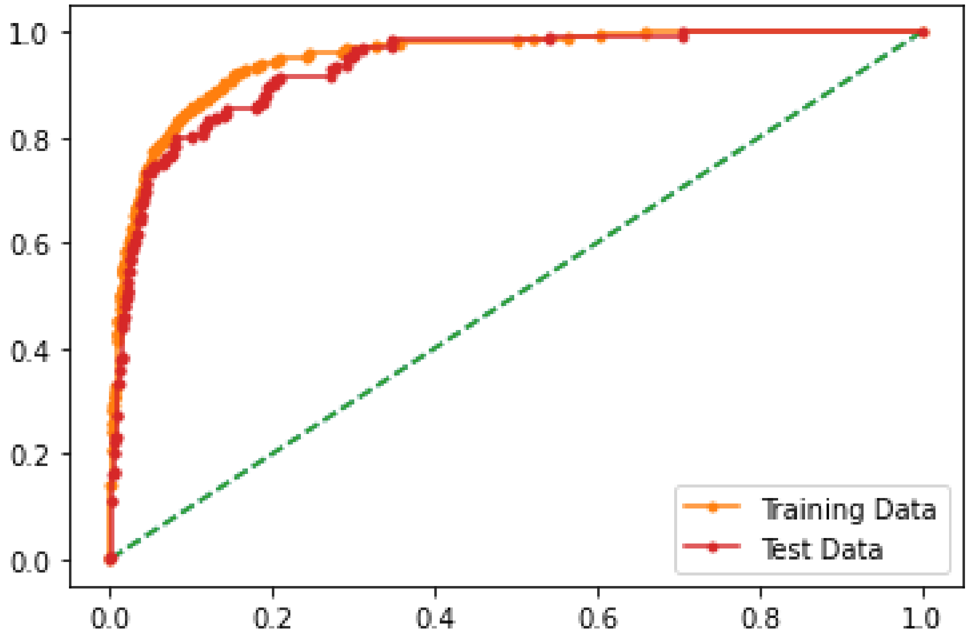
Test data

	precision	recall	f1-score	support
0.0	0.93	0.98	0.96	1041
1.0	0.79	0.45	0.58	143
accuracy			0.92	1184
macro avg	0.86	0.72	0.77	1184
weighted avg	0.91	0.92	0.91	1184

AUC and ROC for the training and test data

AUC for the Training Data: 0.950

AUC for the Test Data: 0.935



Inference

Train Data:

- AUC: 95%
- Accuracy: 94%
- precision : 84%
- recall :48%
- f1 :61%

Test Data:

- AUC: 93%
- Accuracy: 92%
- precision: 79%
- recall : 45%
- f1 : 58%

In this model recall is very low with 48%. It is not a good model

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

MODEL	DATA	ACCURACY	PRECISION	RECALL	F1-SCORE	AUC
RANDOM FOREST	TRAIN	97	93	76	84	99
	TEST	97	94	84	89	98
LDA	TRAIN	94	84	48	61	95
	TEST	92	79	45	58	93
LOGISTIC REGRSSION WITH RFE	TRAIN	95	86	64	95	96
	TEST	95	82	69	75	96

Random forest with grid search performed well with highest recall and good f1 score. Roc Curve shows it's not unfitting or overfitting. While comparing other models, it is observed that Random Forest is best model for credit risk analysis with accuracy of 97%.

Conclusion

Credit report analysis provides information on the credit worthiness of a potential customer The model with selected features will predict a relatively high probability of default. Next step is to integrate with classification model where defaulters further classified into "very high risk", "high risk", "medium risk", "low risk", etc. Later embed these models in Web and Database Integration

Market Risk Analysis

Problem:

Market Risk

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.

Project Objective:

The Objective of the report is to explore the Market risk dataset in Python (JUPYTER NOTEBOOK) and generate insights about the dataset. This exploration report will consist of the following:

- Importing the dataset in jupyter notebook.
- Understanding the structure of dataset.
- Exploratory Data analysis
- Graphical exploration
- Calculate the mean and standard deviation on the stock returns
- Insights from the dataset

Load and Explore Data

Import the market risk data using pandas with Parse_date. We can use the parse_dates parameter to convince pandas to turn things into real datetime types. parse_dates take a list of columns (since you could want to parse multiple columns into datetimes. Changing the messy column names for further analysis

Date	Infosys	Indian_Hotel	Mahindra_&_Mahindra	Axis_Bank	SAIL	Shree_Cement	Sun_Pharma	Jindal_Steel	Idea_Vodafone	Jet_Airways
2014-03-31	264	69	455	263	68	5543	555	298	83	278
2014-07-04	257	68	458	276	70	5728	610	279	84	303
2014-04-14	254	68	454	270	68	5649	607	279	83	280
2014-04-21	253	68	488	283	68	5692	604	274	83	282
2014-04-28	256	65	482	282	63	5582	611	238	79	243

```
The number of rows (observations) is 314
The number of columns (variables) is 10
```

Information of dataset

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 314 entries, 2014-03-31 to 2020-03-30
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Infosys              314 non-null    int64
 1   Indian_Hotel         314 non-null    int64
 2   Mahindra_&_Mahindra  314 non-null    int64
 3   Axis_Bank            314 non-null    int64
 4   SAIL                 314 non-null    int64
 5   Shree_Cement         314 non-null    int64
 6   Sun_Pharma           314 non-null    int64
 7   Jindal_Steel         314 non-null    int64
 8   Idea_Vodafone        314 non-null    int64
 9   Jet_Airways          314 non-null    int64
dtypes: int64(10)
memory usage: 27.0 KB
```

There are information stock prices from 10 different countries

Summary of the dataset

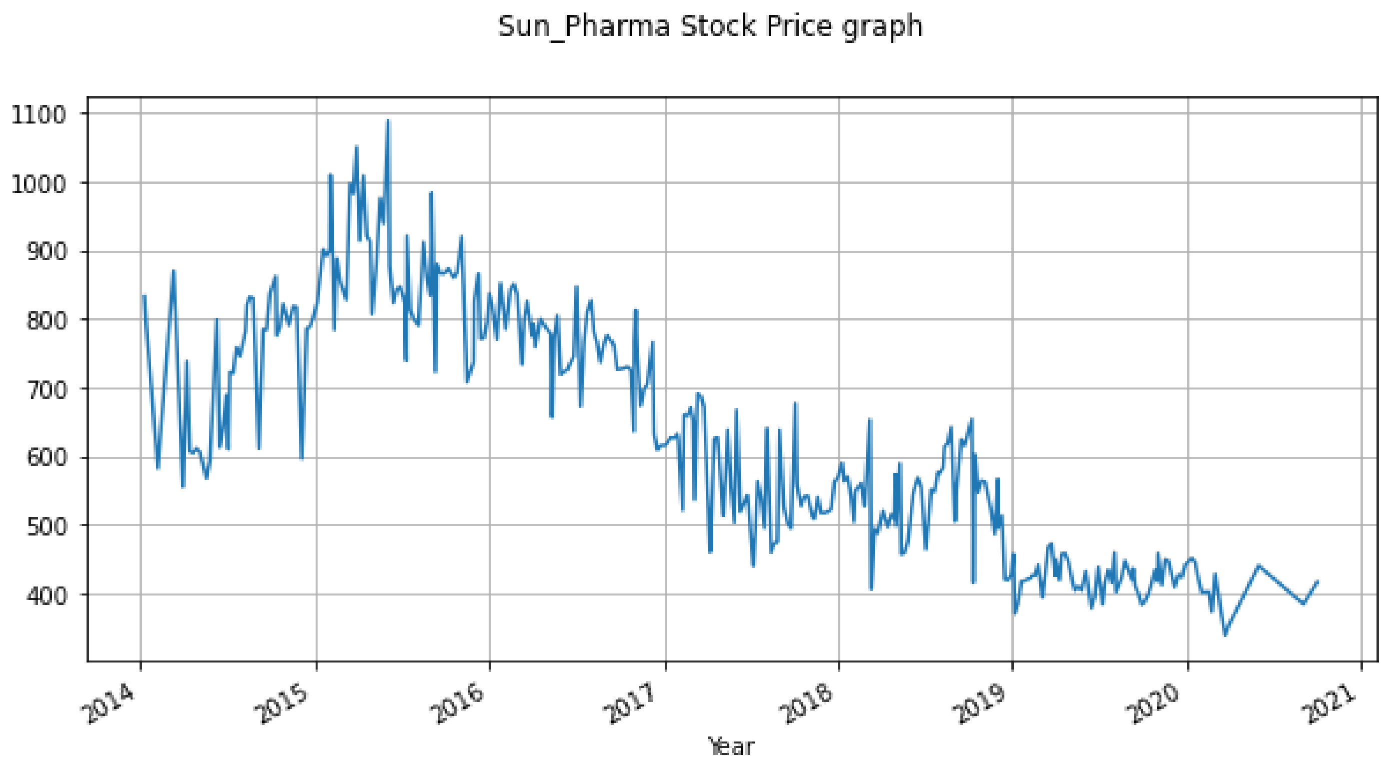
	Infosys	Indian_Hotel	Mahindra_&_Mahindra	Axis_Bank	SAIL	Shree_Cement	Sun_Pharma	Jindal_Steel	Idea_Vodafone	Jet_Airways
count	314	314	314	314	314	314	314	314	314	314
mean	511.3408	114.56051	636.678344	540.742038	59.09554	14806.41083	633.468153	147.627389	53.713376	372.659236
std	135.9521	22.509732	102.879975	115.835569	15.81049	4288.275085	171.855893	65.879195	31.248985	202.262668
min	234	64	284	263	21	5543	338	53	3	14
25%	424	96	572	470.5	47	10952.25	478.5	88.25	25.25	243.25
50%	466.5	115	625	528	57	16018.5	614	142.5	53	376
75%	630.75	134	678	605.25	71.75	17773.25	785	182.75	82	534
max	810	157	956	808	104	24806	1089	338	117	871

Inference

- Shree Cements have highest stock price
- SAIL Company have low stock price

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

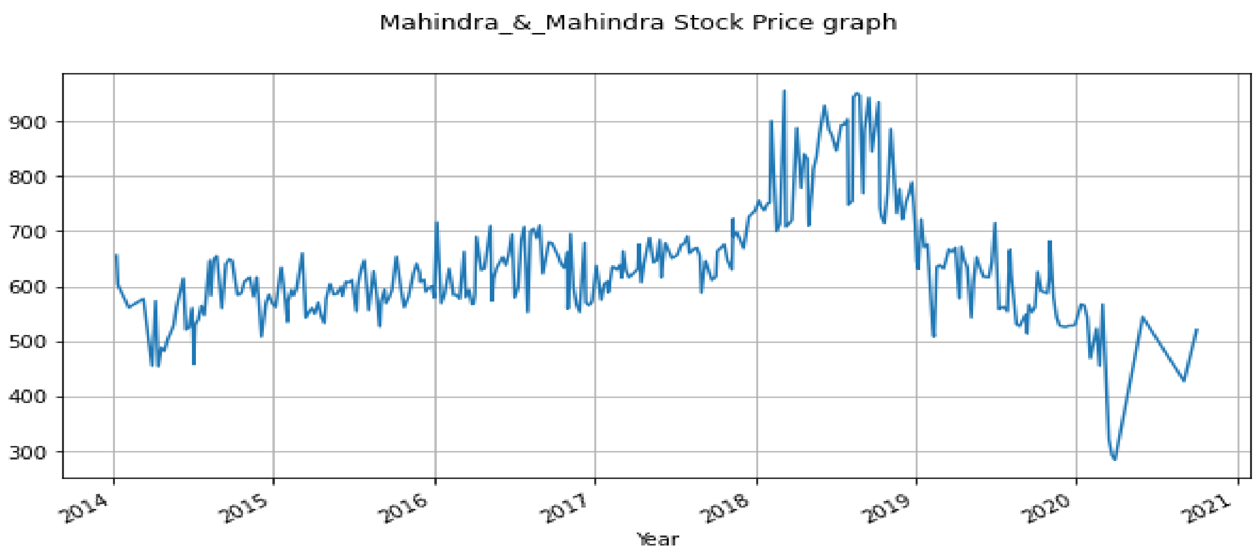
Sun Pharma Stock Price graph



Inference:

Although in the year 2015 Sun Pharma showed increasing stock prices, but Stock prices begin to fall from 2016 leading to downward trend

Mahindra & Mahindra Stock Price graph



Mahindra & Mahindra maintained Stock Price from 500 – 700 between 2014 -2018.In the beginning of 2018 stock price is reached its peek above 900 and there is steep drop in the beginning for 2020

2.2 Calculate Returns for all stocks with inference

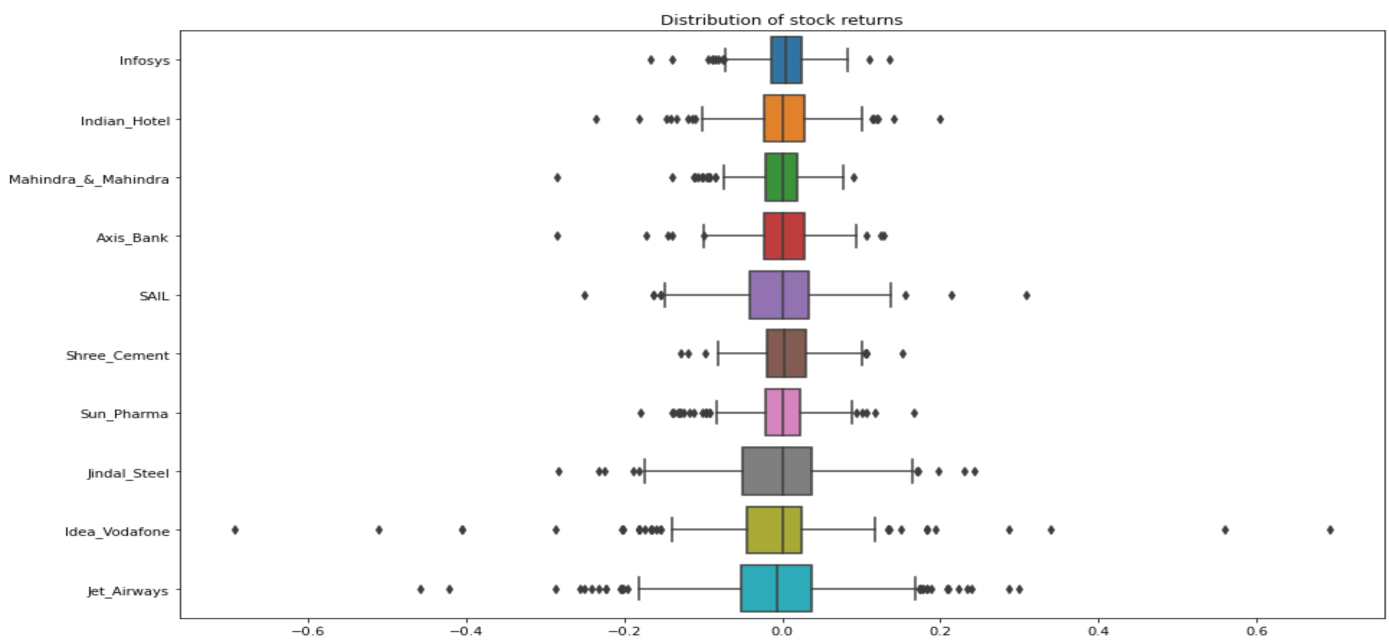
Analysing returns

steps for calculating returns from prices:

Take logarithms Take differences

Date	Infosys	Indian_Hotel	Mahindra_&_Mahindra	Axis_Bank	SAIL	Shree_Cement	Sun_Pharma	Jindal_Steel	Idea_Vodafone	Jet_Airways
2014-03-31	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
2014-07-04	-0.02687	-0.014599	0.006572	0.048247	0.028988	0.032831	0.094491	-0.065882	0.011976	0.086112
2014-04-14	-0.01174	0	-0.008772	-0.021979	-0.02899	-0.013888	-0.00493	0	-0.011976	-0.078943
2014-04-21	-0.00395	0	0.072218	0.047025	0	0.007583	-0.004955	-0.018084	0	0.007117
2014-04-28	0.011788	-0.04512	-0.012371	-0.00354	-0.07637	-0.019515	0.011523	-0.140857	-0.049393	-0.148846

Distribution of stock returns



2.3 Calculate Stock Means and Standard Deviation for all stocks with inference

We now look at Means & Standard Deviations of these returns

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

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

Inference

Idea Vodafone has the lowest returns, while Shree cements have the highest returns

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

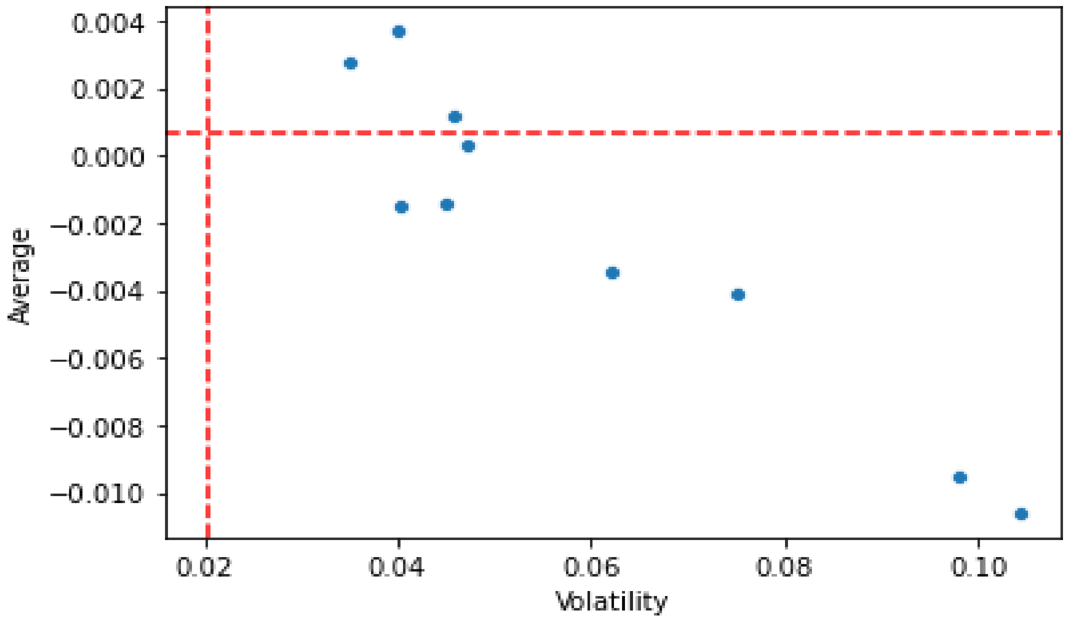
```
Idea_Vodafone          0.104315
Jet_Airways            0.097972
Jindal_Steel           0.075108
SAIL                   0.062188
Indian_Hotel           0.047131
Axis_Bank              0.045828
Sun_Pharma             0.045033
Mahindra_&_Mahindra    0.040169
Shree_Cement           0.039917
Infosys                0.035070
dtype: float64
```

Idea Vodafone has the highest risk factor for investment, while Infosys is the least risky investment option

Creating a data frame with companies, Stock means as Average and Stock standard deviation as volatility

	Average	Volatility
Infosys	0.002794	0.03507
Indian_Hotel	0.000266	0.047131
Mahindra_&_Mahindra	-0.00151	0.040169
Axis_Bank	0.001167	0.045828
SAIL	-0.00346	0.062188
Shree_Cement	0.003681	0.039917
Sun_Pharma	-0.00146	0.045033
Jindal_Steel	-0.00412	0.075108
Idea_Vodafone	-0.01061	0.104315
Jet_Airways	-0.00955	0.097972

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



- Stocks higher up but on the far left indicate high volatility and low returns, while the stocks on the bottom right indicate low volatility and high returns.
- During the investment, this graph is very useful in analysing the risk from different companies

Conclusion

Traders and analysts use several metrics to assess the volatility and relative risk of potential investments, but the most common metric is standard deviation.

- Standard deviation helps determine market volatility or the spread of asset prices from their average price.
- When prices move wildly, standard deviation is high, meaning an investment will be risky.
- Low standard deviation means prices are calm, so investments come with low risk.

In this data we are only left few stocks:

One with highest return and lowest risk & one with lowest risk and highest return

Good Returns:

Shree Cement, Infosys & Axis Bank may have good returns

less Risk (as measured by standard deviation):

Infosys, Shree Cement & Mahindra & Mahindra may have low risk

Recommendations

We would recommend using the stock means vs standard deviation plot to assess the risk to reward ratio. The smaller the standard deviation, an investment will be the less risky. On the other hand, the larger the variance and standard deviation, the more volatile a security. While investors can assume price remains within two standard deviations of the mean 95% of the time, this can still be a very large range. As with anything else, the greater the number of possible outcomes, the greater the risk of choosing the wrong one.