FRA MAIN PROJECT

BUSINESS REPORT (CONSISTING OF BOTH THE PARTS)

JAYA PREETHI R M

TABLE OF CONTENTS	
CONTENT	PAGE NUMBER

PART A:

Problem Statement:

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

A balance sheet is a financial statement of a company that provides a snapshot of what a company owns, lowes, 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.

BRIEF INFORMATION ON THE DATA SET

- Following is a brief glimpse into the data:

	Co_Code	Co_Name	_Operating_Expense_Rate	_Research_and_development_expense_rate	_Cash_flow_rate	_Interest_bearing_debt_interest_rate	_Tax_rate_A	_Cash_Flow_Per_Sha
0	16974	Hind.Cables	8.820000e+09	0.000000e+00	0.462045	0.000352	0.001417	0.3225
1	21214	Tata Tele. Mah.	9.380000e+09	4.230000e+09	0.460116	0.000716	0.000000	0.3155
2	14852	ABG Shipyard	3.800000e+09	8.150000e+08	0.449893	0.000496	0.000000	0.2998
3	2439	GTL	6.440000e+09	0.000000e+00	0.462731	0.000592	0.009313	0.3198
4	23505	Bharati Defence	3.680000e+09	0.000000e+00	0.463117	0.000782	0.400243	0.3251

5 rows × 58 columns

- Following is the shape of the data:

```
The number of rows (observations) is 2058
The number of columns (variables) is 58
```

- There are few null variables, and most variables tend to be either float or integer variables. Thus most of the variables are numerical in nature.

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

- There are no duplicate variables.
- I dropped the Co_Code and Co_Name variables, Since both those variables are descriptive and not necessary for our analysis.
- Most variables are rate or percentage based variables. There are also some variables that are categorical and continuous.

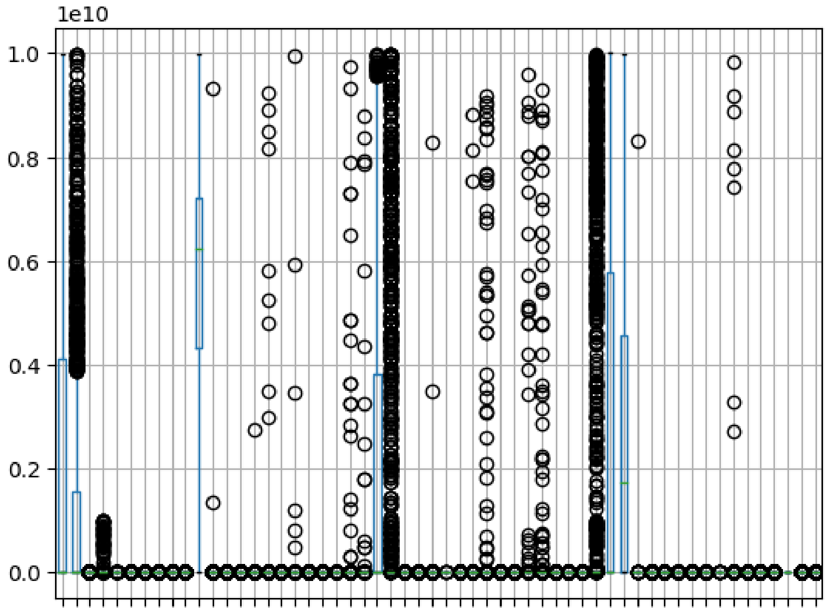
	_Operating_Expense_Rate	_Research_and_development_expense_rate	_Cash_flow_rate	_Interest_bearing_debt_interest_rate	_Tax_rate_A	_Cash_Flow_Per_Share
count	2.058000e+03	2.058000e+03	2058.000000	2.058000e+03	2058.000000	1891.000000
mean	2.052389e+09	1.208634e+09	0.465243	1.113022e+07	0.114777	0.319986
std	3.252624e+09	2.144568e+09	0.022663	9.042595e+07	0.152446	0.015300
min	1.000260e-04	0.000000e+00	0.000000	0.000000e+00	0.000000	0.169449
25%	1.578727e-04	0.000000e+00	0.460099	2.760280e-04	0.000000	0.314989
50%	3.330330e-04	1.994130e-04	0.463445	4.540450e-04	0.037099	0.320648
75%	4.110000e+09	1.550000e+09	0.468069	6.630660e-04	0.216191	0.325918
max	9.980000e+09	9.980000e+09	1.000000	9.900000e+08	0.999696	0.462227

8 rows × 56 columns

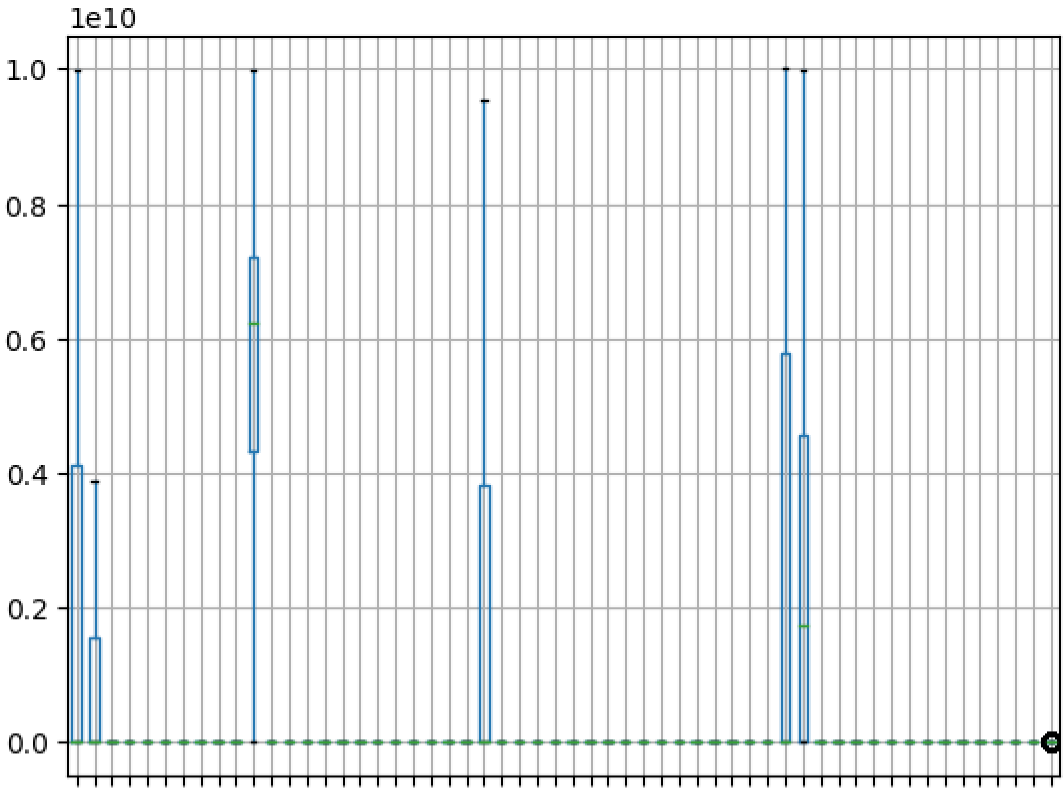
PART A: Outlier Treatment

I performed the outlier treatment on the dataset by using the IQR method to impute the outlier data points. Following is the result of the same.

BEFORE OUTLIER TREATMENT



AFTER OUTLIER TREATMENT



PART A: Missing Value Treatment

I imputed the null values with the median values, to treat missing values in the dataset. Results are as follows:

BEFORE MISSING VALUE TREATMENT

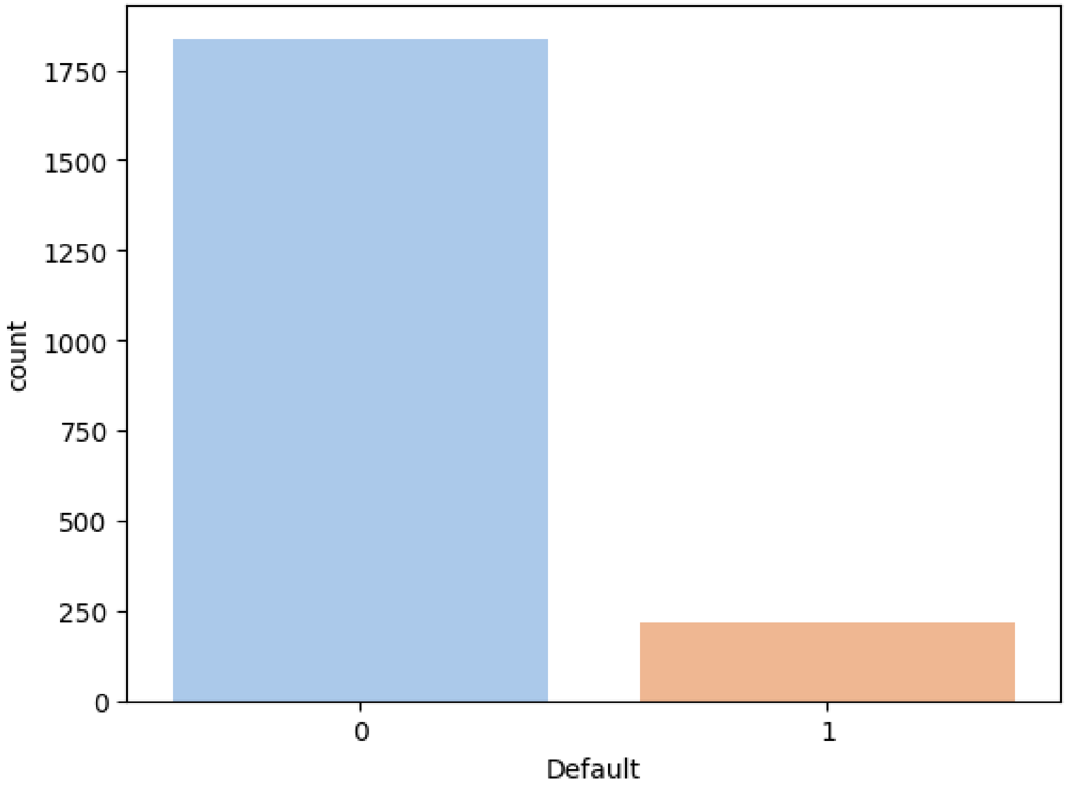
```
_Operating_Expense_Rate                                    0
_Research_and_development_expense_rate                     0
_Cash_flow_rate                                            0
_Interest_bearing_debt_interest_rate                       0
_Tax_rate_A                                         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

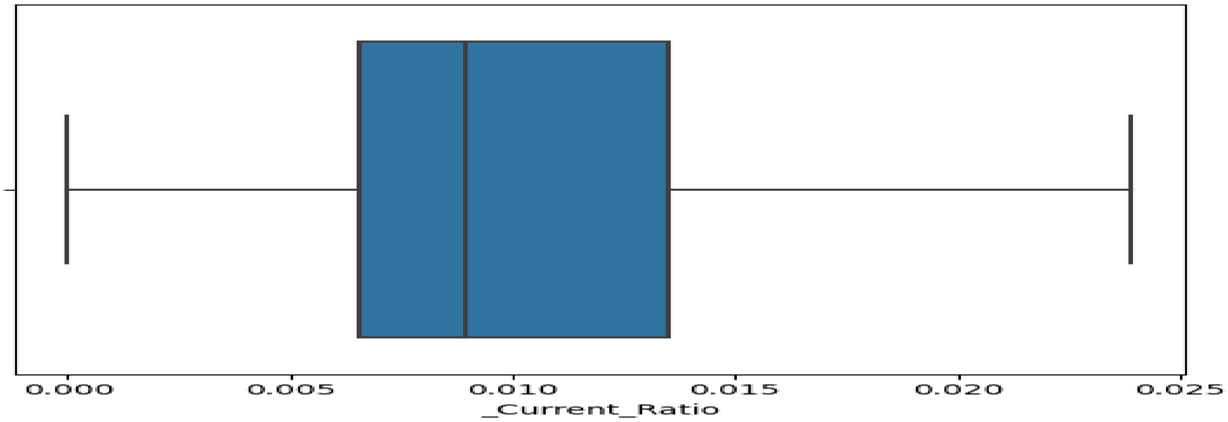
AFTER MISSING VALUE TREATMENT

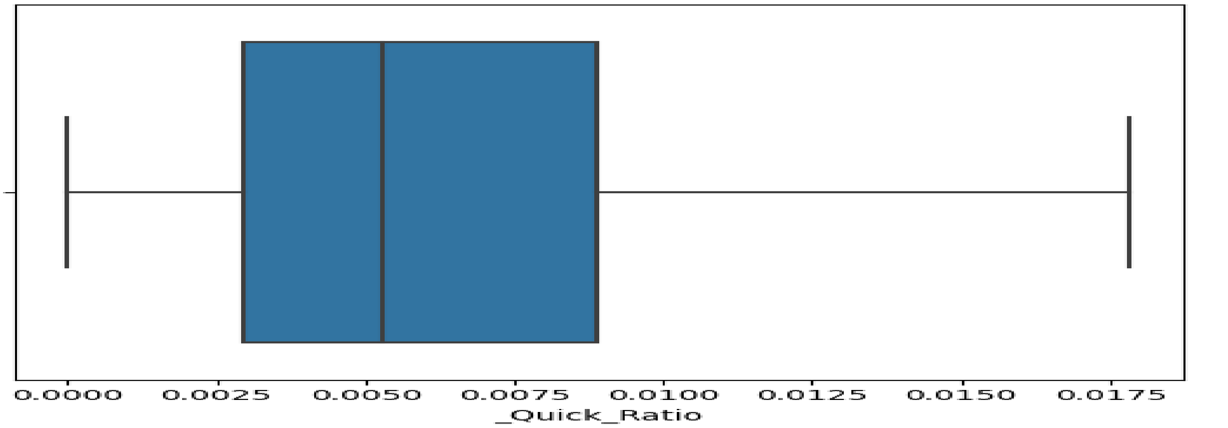
```
_Operating_Expense_Rate                                  0
_Research_and_development_expense_rate                   0
_Cash_flow_rate                                          0
_Interest_bearing_debt_interest_rate                     0
_Tax_rate_A                                              0
_Cash_Flow_Per_Share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

PART A: Univariate & Bivariate analysis with proper interpretation.

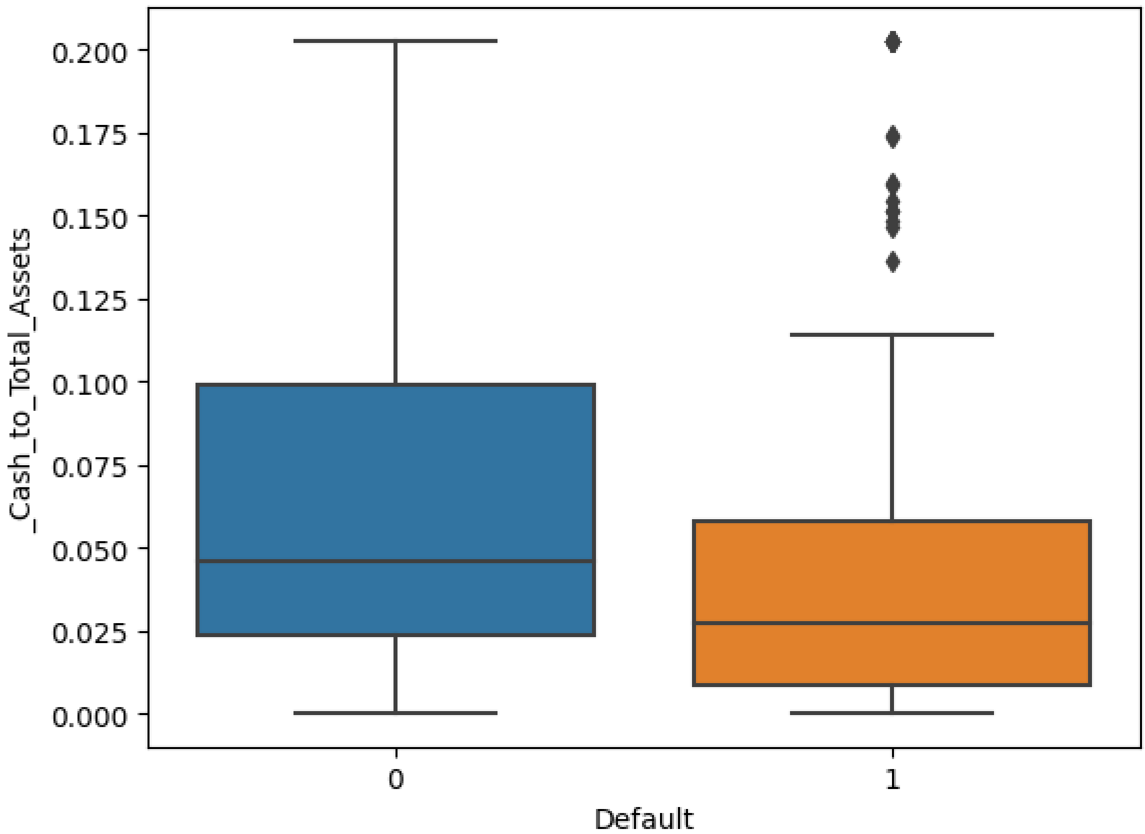
Following is the countplot representing the defaulter of the payment. This variable is our dependent variable and it represents the financial risk of a customer. We can see that there way less defaulters than the non-defaulters. This is a good thing, because there is less financial risk at hand.





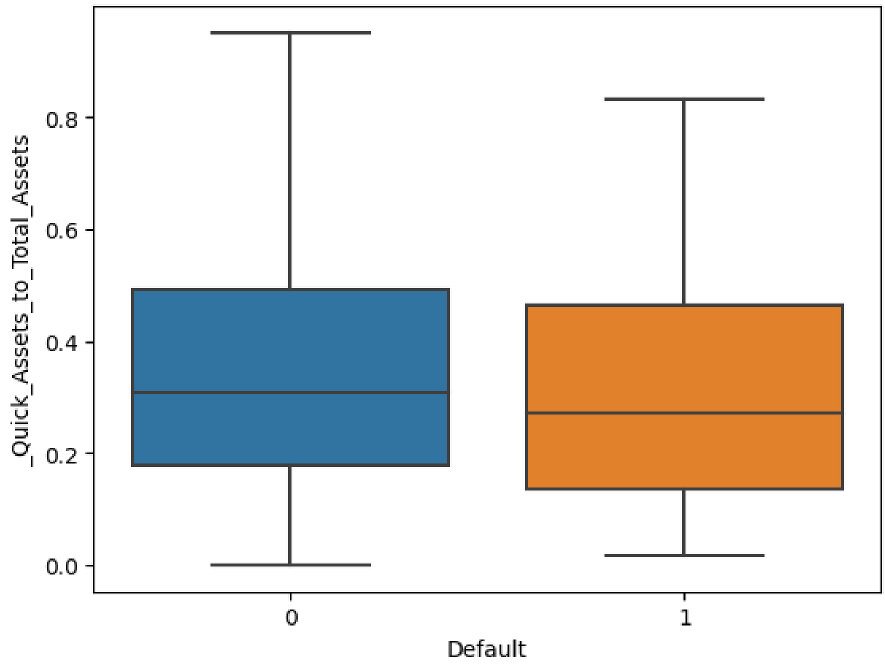


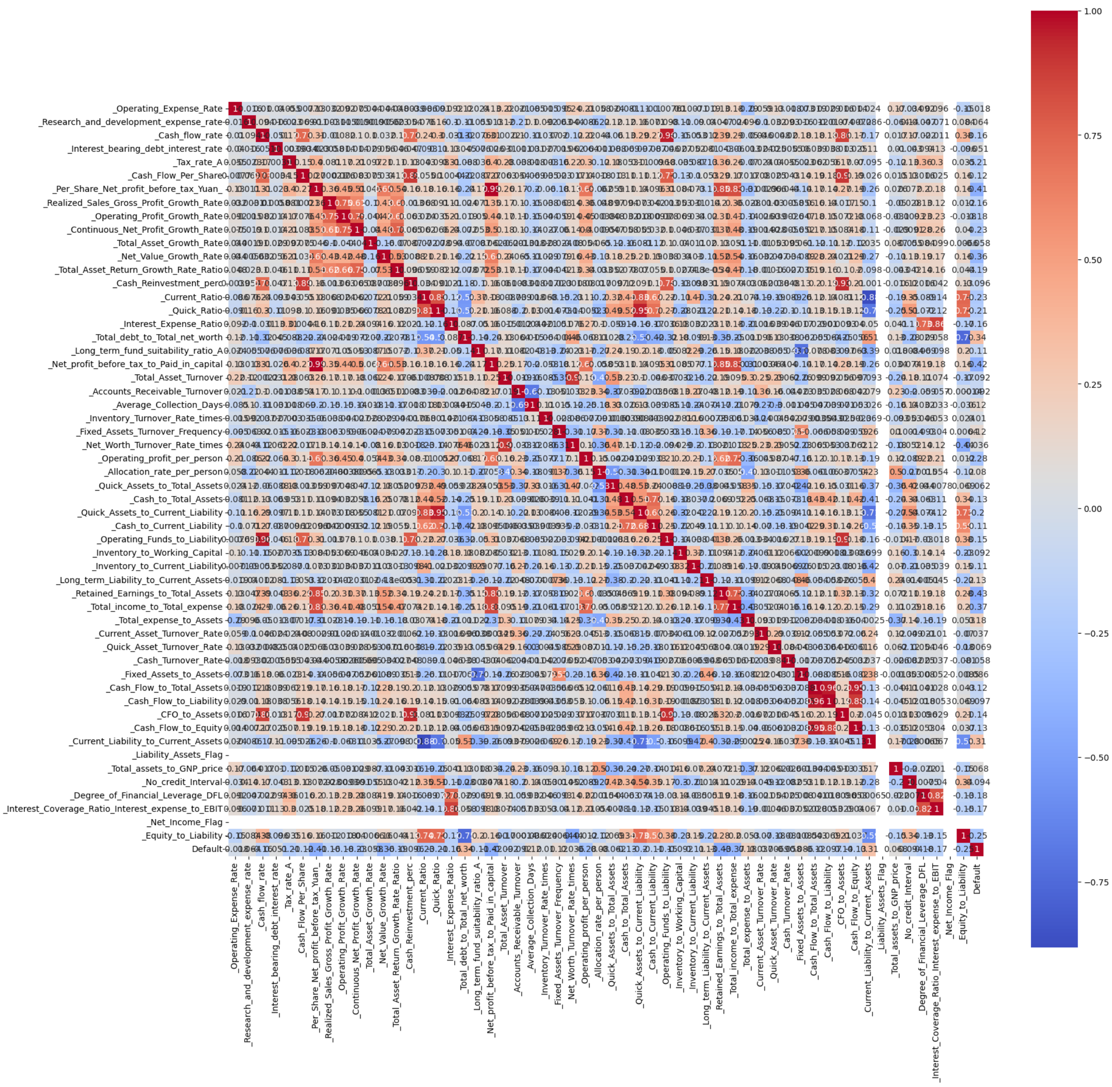
The quick ratio measures a company's ability to convert liquid assets into cash to pay for short-term expenses and weather emergencies. The current ratio measures a company's ability to pay current, or short-term, liabilities (debt and payables) with its current, or short-term, assets (cash, inventory, and receivables). We can see that most companies have a higher current ratio than quick ratio. This might be because the companies could anticipate payment from short term liabilities more frequently than once in a while expenses like emergencies.



We can see that the liquidity from cash is higher among non-defaulters (close to average of 0.05) than defaulters(close to average of 0.025). This is a good indication of the customer's ability to pay off loans without any default.

Quick assets include cash on hand or current assets like accounts receivable that can be converted to cash with minimal or no discounting.We can that the customers with high liquidity through quick assets are non-defaulters (close to 0.4), whereas defaulters have less liquidity from quick assets (close to 0.2).


_Quick_Assets_to_Total_Assets
0.0
0.2
0.4
0.6
0.8
0
1
Default


1.00
0.75
0.50
0.25
0.00
-0.25
-0.50
-0.75

The above given heatmap is the representation of the correlation between variables. Though the heatmap is not very legible or interpretable due high amounts of variables, we can see that there are some variables with high correlation between each other to the point that they are identical in nature. There are also some cases where the variables are highly negatively correlated.

PART A: Train Test Split

I have split the main dataset into Y set as the default variable and the rest of the variables into X variables using train_test_split tools from Sklearn. I split the dataset into a 67:33 ratio with random_state set at 42. Following are the results:

	_Operating_Expense_Rate	_Research_and_development_expense_rate	_Cash_flow_rate	_I
631	1.053450e-04	3.875000e+09	0.462934	
1799	1.569190e-04	0.000000e+00	0.480024	
1924	5.556330e-04	0.000000e+00	0.480024	
1629	8.520000e+08	3.460000e+09	0.463998	
363	7.870000e+09	0.000000e+00	0.480024	
...	...	...	...	
1638	8.480000e+09	5.090000e+07	0.460247	
1095	2.693170e-04	3.875000e+09	0.480024	
1130	2.932540e-04	3.875000e+09	0.459425	
1294	6.880000e+09	0.000000e+00	0.464892	
860	1.246560e-04	1.830000e+09	0.471560	

1378 rows × 35 columns

631	0
1799	0
1924	0
1629	0
363	0
	..
1638	1
1095	0
1130	1
1294	0
860	0

PART A: Build Logistic Regression Model on most important variables on train dataset and choose the optimum cut-off. Also showcase your model building approach.

Before building the logistic model, I checked the VIF (variance inflation factor) for all variables and gradually checked the VIF for each model by dropping the ones with highest inflation. I had to perform this function before building the model because I repeatedly get the LinAlg error due high correlation between the variables. After repeating this tep, until all variables had VIF below the rate of 5. I began to build the model with the following logistic formula:

```
f_1 = 'Default ~ _Total_Asset_Growth_Rate + _Equity_to_Liability +
_Cash_Flow_to_Liability + _Cash_Flow_to_Equity +
_Total_assets_to_GNP_price + _No_credit_Interval +
```

```
_Interest_bearing_debt_interest_rate + _Cash_Flow_Per_Share +
_Realized_Sales_Gross_Profit_Growth_Rate +
_Operating_Profit_Growth_Rate + _Quick_Assets_to_Total_Assets +
_Cash_to_Total_Assets + _Cash_to_Current_Liability +
_Inventory_to_Working_Capital + _Inventory_to_Current_Liability +
_Long_term_Liability_to_Current_Assets +
_Retained_Earnings_to_Total_Assets + _Total_expense_to_Assets +
_Current_Asset_Turnover_Rate + _Quick_Asset_Turnover_Rate +
_Cash_Turnover_Rate + _Fixed_Assets_to_Assets + _Cash_Flow_to_Liability
+ _Total_assets_to_GNP_price + _No_credit_Interval'
```

Following are the results of the model building:

Logit Regression Results

Dep. Variable:	Default	**No. Observations:**	2058
Model:	Logit	**Df Residuals:**	2035
Method:	MLE	**Df Model:**	22
Date:	Sun, 19 Nov 2023	**Pseudo R-squ.:**	0.3847
Time:	02:46:31	**Log-Likelihood:**	-430.49
converged:	False	**LL-Null:**	-699.69
Covariance Type:	nonrobust	**LLR p-value:**	7.002e-100

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	362.2679	699.317	0.518	0.604	-1008.367	1732.903
_Total_Asset_Growth_Rate	-1.016e-11	3.59e-11	-0.283	0.777	-8.05e-11	6.02e-11
_Equity_to_Liability	-75.4458	11.247	-6.708	0.000	-97.490	-53.402
_Cash_Flow_to_Liability	-20.3061	52.074	-0.390	0.697	-122.369	81.757
_Cash_Flow_to_Equity	-22.8321	46.711	-0.489	0.625	-114.383	68.719
_Total_assets_to_GNP_price	64.5702	18.822	3.431	0.001	27.680	101.460
_No_credit_Interval	-5.0798	131.399	-0.039	0.969	-262.618	252.458
_Interest_bearing_debt_interest_rate	718.8311	363.419	1.978	0.048	6.544	1431.118
_Cash_Flow_Per_Share	-15.0624	11.404	-1.321	0.187	-37.413	7.289
_Realized_Sales_Gross_Profit_Growth_Rate	-2440.0055	1208.264	-2.019	0.043	-4808.159	-71.852
_Operating_Profit_Growth_Rate	-203.0914	837.351	-0.243	0.808	-1844.269	1438.087
_Quick_Assets_to_Total_Assets	-0.7219	0.747	-0.966	0.334	-2.186	0.742
_Cash_to_Total_Assets	-4.3054	2.498	-1.724	0.085	-9.201	0.590
_Cash_to_Current_Liability	44.3261	23.270	1.905	0.057	-1.281	89.934
_Inventory_to_Working_Capital	-88.1228	130.210	-0.677	0.499	-343.330	167.084
_Inventory_to_Current_Liability	-22.7397	17.648	-1.289	0.198	-57.329	11.850
_Long_term_Liability_to_Current_Assets	-3.9209	12.176	-0.322	0.747	-27.785	19.943
_Retained_Earnings_to_Total_Assets	-94.5936	10.061	-9.402	0.000	-114.312	-74.875
_Total_expense_to_Assets	12.9565	6.139	2.111	0.035	0.925	24.988
_Current_Asset_Turnover_Rate	-130.4089	83.152	-1.568	0.117	-293.383	32.565
_Quick_Asset_Turnover_Rate	-1.3e-11	2.75e-11	-0.472	0.637	-6.69e-11	4.09e-11
_Cash_Turnover_Rate	-6.359e-11	3.7e-11	-1.718	0.086	-1.36e-10	8.97e-12
_Fixed_Assets_to_Assets	0.2282	0.568	0.402	0.688	-0.885	1.341

You can see that there were a lot of variables that were insignificant in predicting the dependent variable. Therefore, I built a new model by dropping the most insignificant variables.

As you can see below, the 2nd model also had some insignificant variables where the p-value was greater than 0.05. Therefore, I built a third and final model with the most important variables.

2ND MODEL

Logit Regression Results

Dep. Variable:	Default	**No. Observations:**	2058
Model:	Logit	**Df Residuals:**	2042
Method:	MLE	**Df Model:**	15
Date:	Sun, 19 Nov 2023	**Pseudo R-squ.:**	0.3838
Time:	02:56:30	**Log-Likelihood:**	-431.18
converged:	True	**LL-Null:**	-699.69
Covariance Type:	nonrobust	**LLR p-value:**	8.225e-105

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	173.0456	77.489	2.233	0.026	21.169	324.922
_Equity_to_Liability	-74.5210	10.955	-6.802	0.000	-95.993	-53.049
_Total_assets_to_GNP_price	62.3680	18.018	3.461	0.001	27.053	97.683
_No_credit_Interval	-0.3742	120.949	-0.003	0.998	-237.430	236.681
_Interest_bearing_debt_interest_rate	716.3612	362.499	1.976	0.048	5.876	1426.846
_Cash_Flow_Per_Share	-15.1024	11.169	-1.352	0.176	-36.994	6.789
_Realized_Sales_Gross_Profit_Growth_Rate	-2670.7410	877.688	-3.043	0.002	-4390.977	-950.505
_Quick_Assets_to_Total_Assets	-0.8096	0.555	-1.458	0.145	-1.898	0.278
_Cash_to_Total_Assets	-4.3295	2.481	-1.745	0.081	-9.192	0.533
_Cash_to_Current_Liability	46.4223	23.136	2.007	0.045	1.077	91.768
_Inventory_to_Current_Liability	-28.4465	15.194	-1.872	0.061	-58.227	1.334
_Retained_Earnings_to_Total_Assets	-97.0186	9.703	-9.999	0.000	-116.035	-78.002
_Total_expense_to_Assets	12.9628	5.775	2.245	0.025	1.645	24.281
_Current_Asset_Turnover_Rate	-129.0086	80.159	-1.609	0.108	-286.118	28.101
_Cash_Turnover_Rate	-6.42e-11	3.68e-11	-1.742	0.081	-1.36e-10	8.02e-12
_Cash_Flow_to_Liability	-42.5207	27.792	-1.530	0.126	-96.992	11.951

MODEL 3

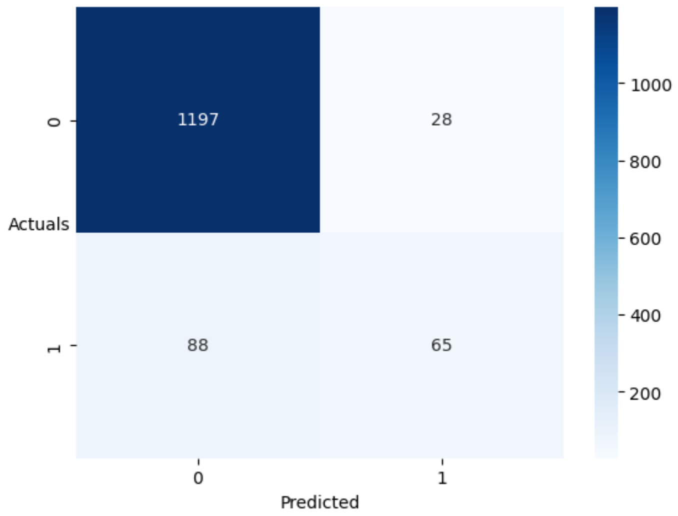
Logit Regression Results

Dep. Variable:	Default	**No. Observations:**	2058
Model:	Logit	**Df Residuals:**	2045
Method:	MLE	**Df Model:**	12
Date:	Sun, 19 Nov 2023	**Pseudo R-squ.:**	0.3810
Time:	03:01:16	**Log-Likelihood:**	-433.09
converged:	True	**LL-Null:**	-699.69
Covariance Type:	nonrobust	**LLR p-value:**	1.888e-106

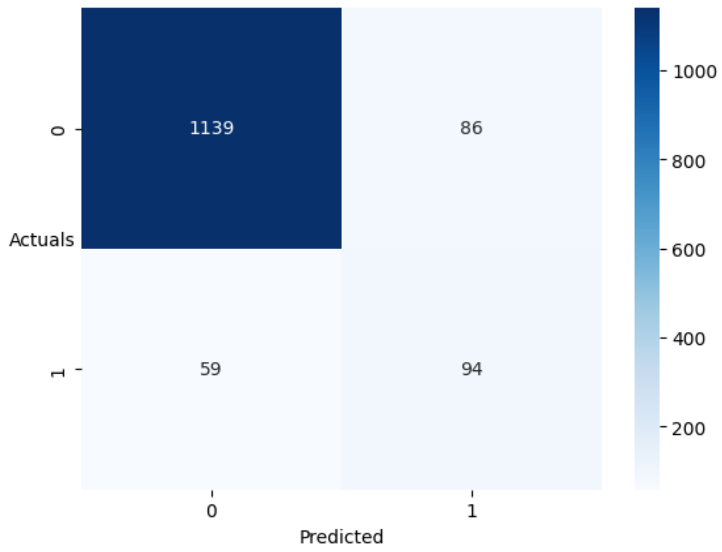
	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	174.8251	22.736	7.689	0.000	130.264	219.386
_Equity_to_Liability	-73.0601	10.476	-6.974	0.000	-93.592	-52.528
_Total_assets_to_GNP_price	65.9749	17.637	3.741	0.000	31.407	100.543
_Interest_bearing_debt_interest_rate	757.8037	353.973	2.141	0.032	64.030	1451.577
_Realized_Sales_Gross_Profit_Growth_Rate	-2734.3940	877.536	-3.116	0.002	-4454.333	-1014.455
_Cash_to_Total_Assets	-5.4870	2.296	-2.390	0.017	-9.987	-0.987
_Cash_to_Current_Liability	49.1181	22.990	2.136	0.033	4.058	94.179
_Inventory_to_Current_Liability	-19.4327	13.992	-1.389	0.165	-46.857	7.992
_Retained_Earnings_to_Total_Assets	-101.5508	9.400	-10.804	0.000	-119.974	-83.128
_Total_expense_to_Assets	11.1171	5.623	1.977	0.048	0.096	22.138
_Current_Asset_Turnover_Rate	-116.4870	78.592	-1.482	0.138	-270.525	37.551
_Cash_Turnover_Rate	-5.967e-11	3.65e-11	-1.635	0.102	-1.31e-10	1.19e-11
_Cash_Flow_to_Liability	-45.8916	27.197	-1.687	0.092	-99.196	7.413

There are a total of 13 independent variables in the final model, with most variables being significantly predictive of the dependent variables and the final model has a current function value of 0.21.

PART A: Validate the Model on Test Dataset and state the performance metrics. Also state interpretation from the model
I validate the final model on the train dataset and following ifs the confusion matrix for the same:



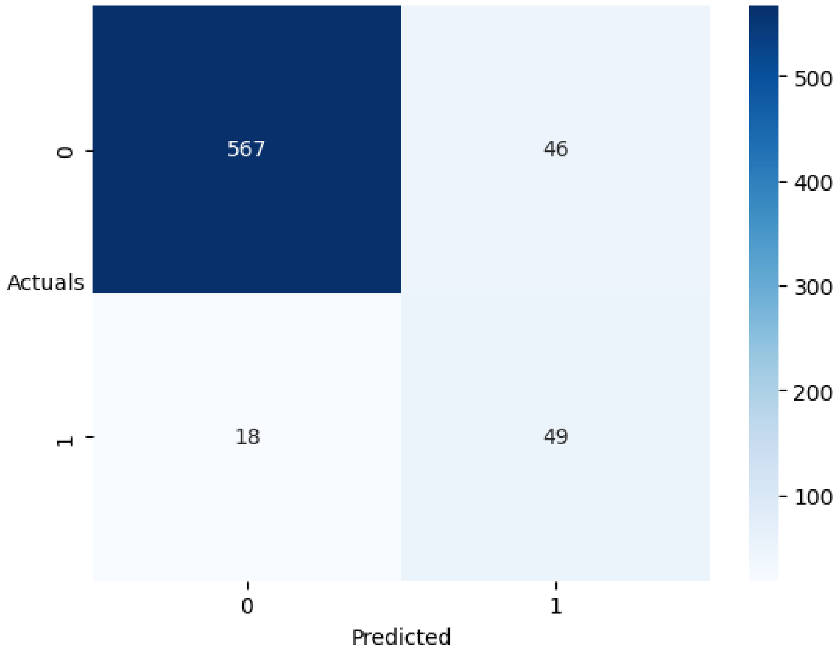
The precision is 0.42 and the recall is 0.7. Therefore we can see that the model has not done a very good job at predicting the dependent variable. Therefore, to boost the performance of the model, I implemented the optimal threshold to the validated model.
Following are the results:
TRAIN DATASET



	precision	recall	f1-score	support
0	0.951	0.930	0.940	1225
1	0.522	0.614	0.565	153
accuracy			0.895	1378
macro avg	0.736	0.772	0.752	1378
weighted avg	0.903	0.895	0.898	1378

TEST MODEL

	precision	recall	f1-score	support
0	0.969	0.925	0.947	613
1	0.516	0.731	0.605	67
accuracy			0.906	680
macro avg	0.743	0.828	0.776	680
weighted avg	0.925	0.906	0.913	680



We can see that the test model has a slight case of overfitting where the train accuracy is 0.89 and test accuracy is 0.91. The recall rate for trains is higher than the test. Thus, we can say that this model has performed better than all other previous models, in predicting the dependent variable.

PART A: Build a Random Forest Model on a Train Dataset. Also showcase your model building approach

When I built the Random Forest Model using the train model, I used the bestparams function to check the most optimum parameters to build the model. Here are the following parameter:

```
{'max_depth': 7,
 'min_samples_leaf': 5,
 'min_samples_split': 30,
 'n_estimators': 50}
```

I decided to build the final model for validation using these parameters.

PART A: Validate the Random Forest Model on test Dataset and state the performance metrics. Also state interpretation from the model.

I validated the final model on the test dataset. Following are the performance metrics:

```
[ ] print(metrics.classification_report(y_train, pred_train_rf))

              precision    recall  f1-score   support

           0       0.94      0.99      0.96      1225
           1       0.87      0.48      0.62       153

    accuracy                           0.93      1378
   macro avg       0.90      0.73      0.79      1378
weighted avg       0.93      0.93      0.93      1378
```

```
[ ] print(metrics.classification_report(y_test, pred_test_rf))

              precision    recall  f1-score   support

           0       0.93      0.98      0.95       613
           1       0.62      0.37      0.47        67

    accuracy                           0.92       680
   macro avg       0.78      0.67      0.71       680
weighted avg       0.90      0.92      0.91       680
```

We can see that the test model has performed significantly great at predicting the train model. The accuracy of the train model is 0.93 and the accuracy of the test model is 0.92. The recall of the train model is higher than the recall of the test model. Although there is a slight case of underfitting, the predicting model does a good job at replicating the original model.

PART A: Build a LDA Model on Train Dataset. Also showcase your model building approach

I built the LDA model using the LDA tool and got the results for ther same by validating the model on the test model:

```
[ ] print(metrics.classification_report(y_train, pred_train_lda))
```

```
              precision    recall  f1-score   support

           0       0.94      0.96      0.95      1225
           1       0.63      0.55      0.59       153

    accuracy                           0.91      1378
   macro avg       0.79      0.75      0.77      1378
weighted avg       0.91      0.91      0.91      1378
```

```
[ ] print(metrics.classification_report(y_test, pred_test_lda))
```

```
              precision    recall  f1-score   support

           0       0.96      0.95      0.95       613
           1       0.57      0.61      0.59        67

    accuracy                           0.92       680
   macro avg       0.76      0.78      0.77       680
weighted avg       0.92      0.92      0.92       680
```

We can see that the test model has some overfitting and the overall model cna do better than this. Therefore, I implemented the optimum threshold for the predicing model to validate on the test dataset.

PART A: Validate the LDA Model on test Dataset and state the performance metrics. Also state interpretation from the model

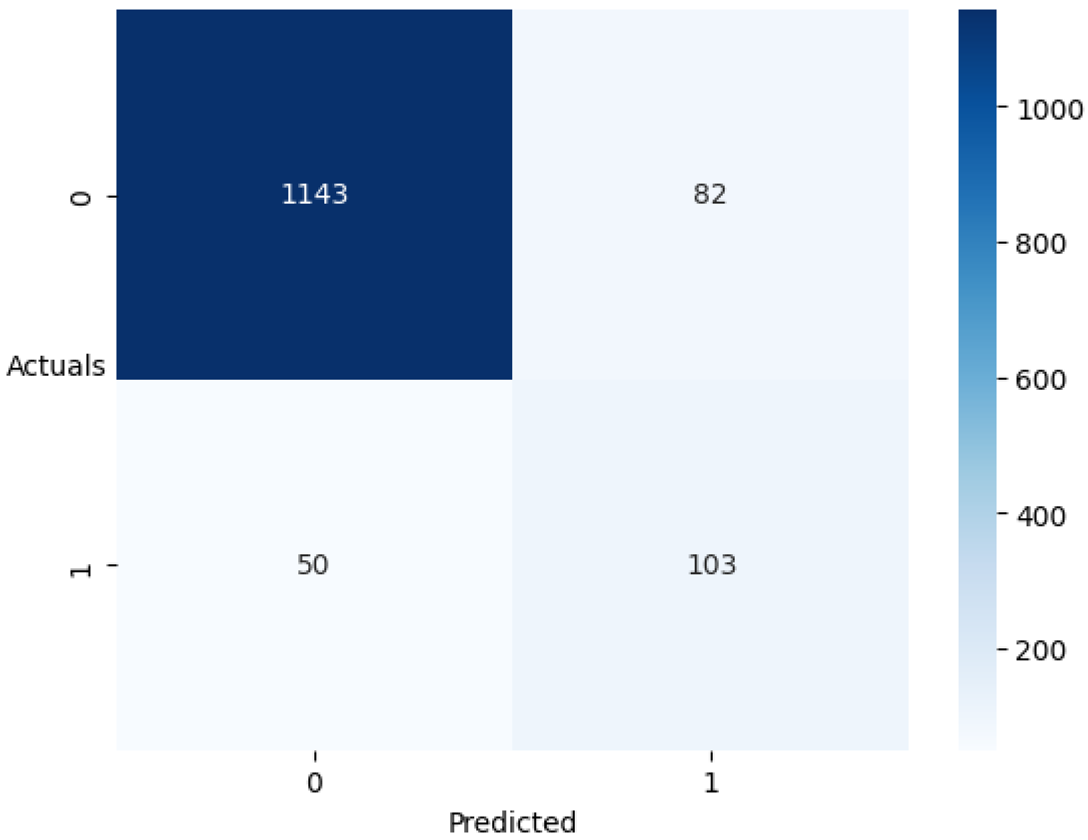
After validating the final dataset here are the performance metrics:

TRAIN MODEL

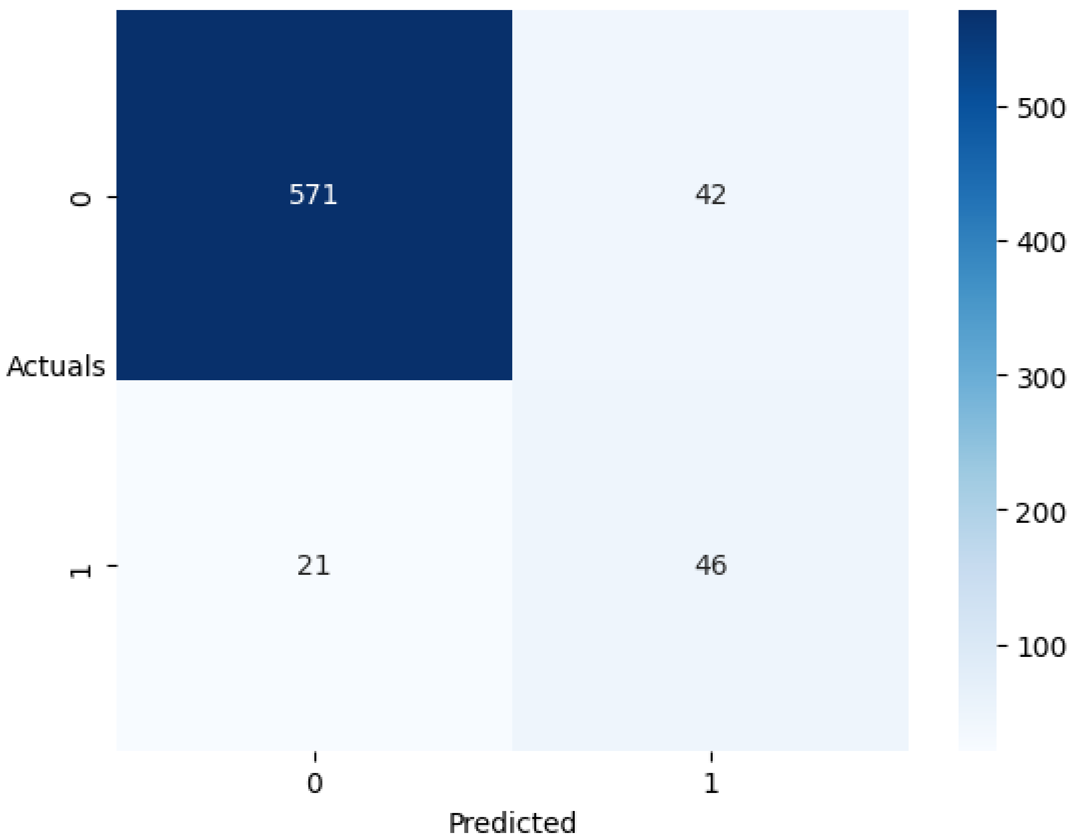
```
              precision    recall  f1-score   support

           0      0.958     0.933     0.945      1225
           1      0.557     0.673     0.609       153

    accuracy                          0.904      1378
   macro avg      0.757     0.803     0.777      1378
weighted avg      0.914     0.904     0.908      1378
```


1000
800
600
400
200
0
1
Actuals
1143
82
50
103
0
1
Predicted

TEST MODEL


500
400
300
200
100
0
1
Actuals
571
42
21
46
0
1
Predicted

	precision	recall	f1-score	support
0	0.965	0.931	0.948	613
1	0.523	0.687	0.594	67
accuracy			0.907	680
macro avg	0.744	0.809	0.771	680
weighted avg	0.921	0.907	0.913	680

There is a case of overfitting in the test model, where the accuracy of the train model is 0.904 and the test model is 0.907. The recall rate has also improved from the train md=model significantly. But, I believe that the previous model is better suited in the prediction of the dependent variable since I believe it is easier to interpret the results on the initial model.

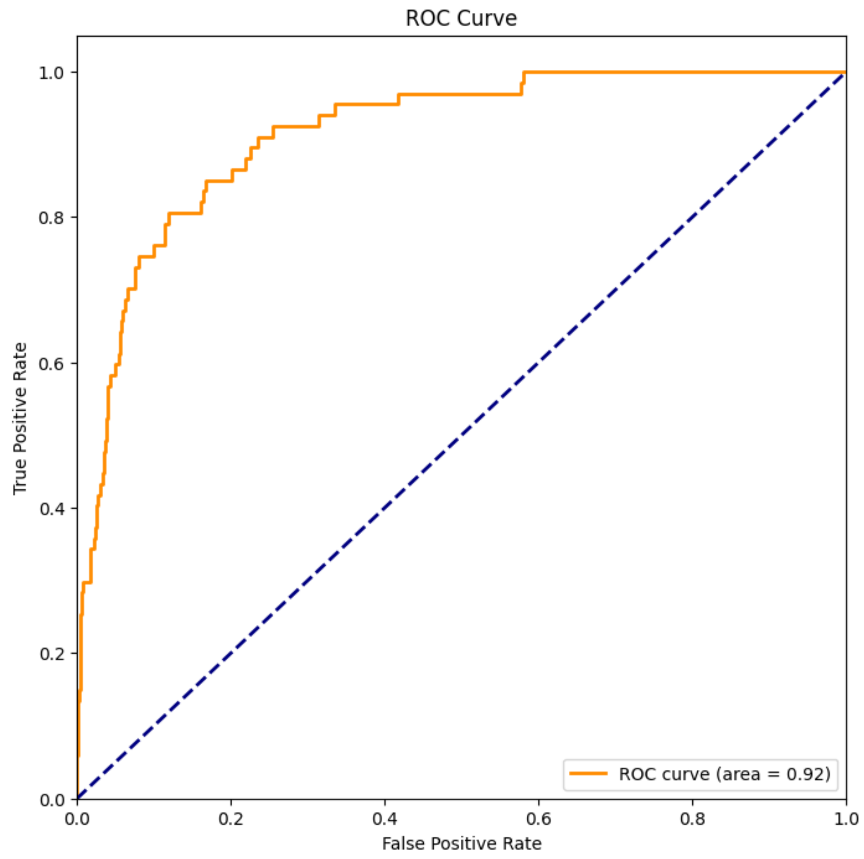
PART A: Compare the performances of Logistic Regression, Random Forest, and LDA models (include ROC curve)

Let us compare the classification report of the performance of all final models on the test dataset.

LOGISTIC MODEL

	precision	recall	f1-score	support
0	0.969	0.925	0.947	613
1	0.516	0.731	0.605	67
accuracy			0.906	680
macro avg	0.743	0.828	0.776	680
weighted avg	0.925	0.906	0.913	680

The ROC curve has significant area coverage and it tends to be much more curvier, resulting in a good model representation.



RANDOM FOREST MODEL

	precision	recall	f1-score	support
0	0.93	0.98	0.95	613
1	0.62	0.37	0.47	67
accuracy			0.92	680
macro avg	0.78	0.67	0.71	680
weighted avg	0.90	0.92	0.91	680

LDA MODEL

	precision	recall	f1-score	support
0	0.965	0.931	0.948	613
1	0.523	0.687	0.594	67
accuracy			0.907	680
macro avg	0.744	0.809	0.771	680
weighted avg	0.921	0.907	0.913	680

All the models have a considerable amount of accuracy in predicting the dependent variable.However, I believe that the logistic model is the best suited model to interpret and derive inferences. Although, Random forest model has the highest accuracy, I believe that the logistic regression model is the most inclusive of all the most important variables and it has the most interpretable results while taking the problem statement into consideration.

PART A: Conclusions and Recommendations

- In conclusion, the logistic model is the best model in predicting the defaulters in prospective customers.
- Particularly, checking the

1. equity to liability ratio (make sure that it doesn't exceed 2.0)
2. Total assets to GNP price (make sure that the total assets are in appropriate proportion to the GNP price)
3. Interest bearing debt interest rate (make sure that the interest bearing debt is serviceable)
4. Retained earnings to total asset (make sure that it is cas close to 100 percent as possible)
5. Realised sales gross profit growth rate (make sure that is sustainably high)
6. Total expense to assets (must be from 0.5% to 0.75%)

PART B:

Problem Statement:

The dataset contains 6 years of information(weekly stock information) on the stock prices of 10 different Indian Stocks. Calculate the mean and standard deviation on the stock returns and share insights. You are expected to do the Market Risk Analysis using Python.

BRIEF INFORMATION ON THE DATA SET

- This is a brief glimpse into the dataset: The variables are stockrice values of some indian companies

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

- This the shape of the dataset

```
The number of rows (observations) is 314
The number of columns (variables) is 11
```

- Following is the variable information: Most of the variables are integers and there are no null variables.

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

- Following the description of the dataset: The average stock prices of the dataset tends to oscillate between 50 to 15,000.

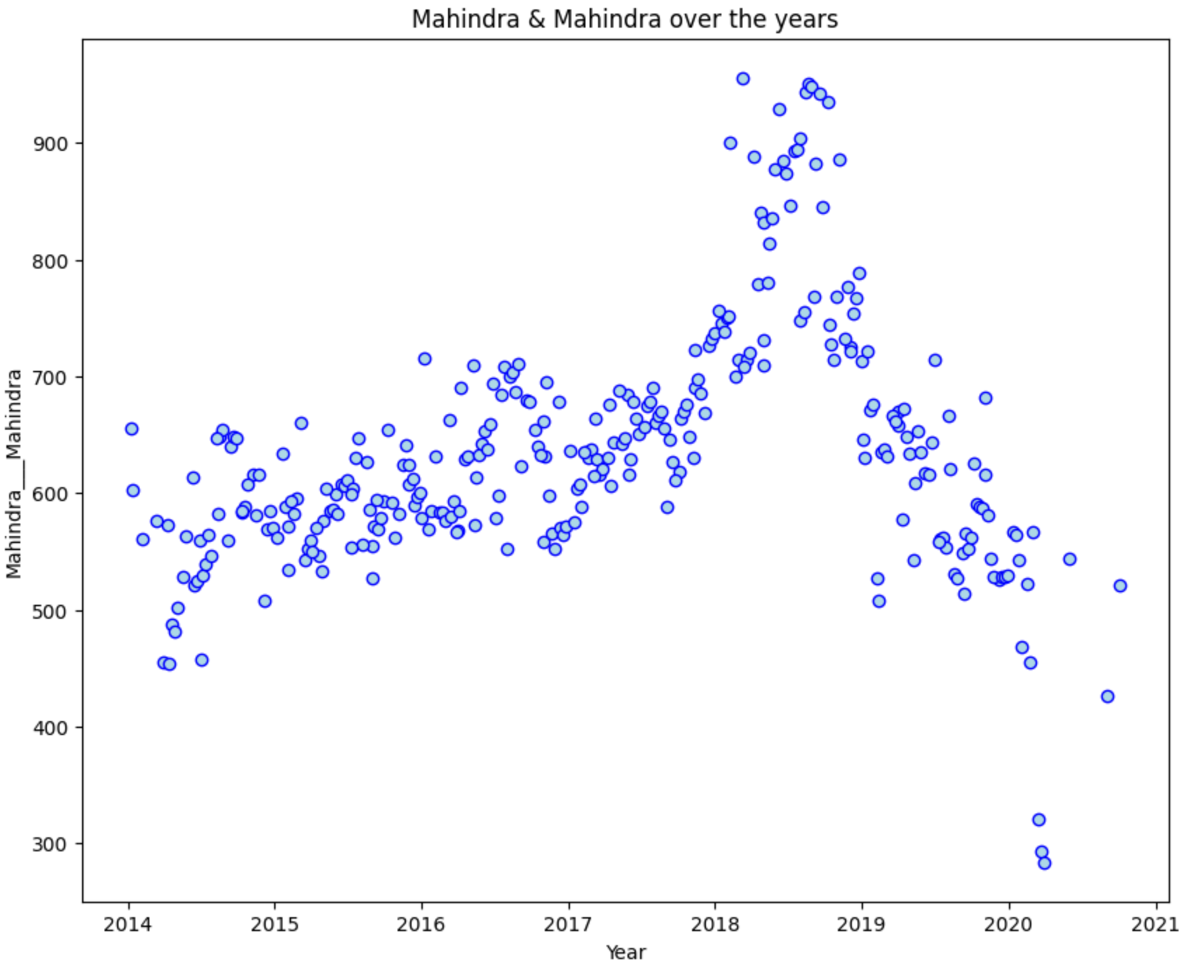
	Infosys	Indian_Hotel	Mahindra___Mahindra	Axis_Bank	SAIL	Shree_Cement	Sun_Pharma	Jindal_Steel	Idea_Vodafone	Jet_Airways
count	314.000000	314.000000	314.000000	314.000000	314.000000	314.000000	314.000000	314.000000	314.000000	314.000000
mean	511.340764	114.560510	636.678344	540.742038	59.095541	14806.410828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

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

STOCK PRICES OF MAHINDRA AND MAHINDRA

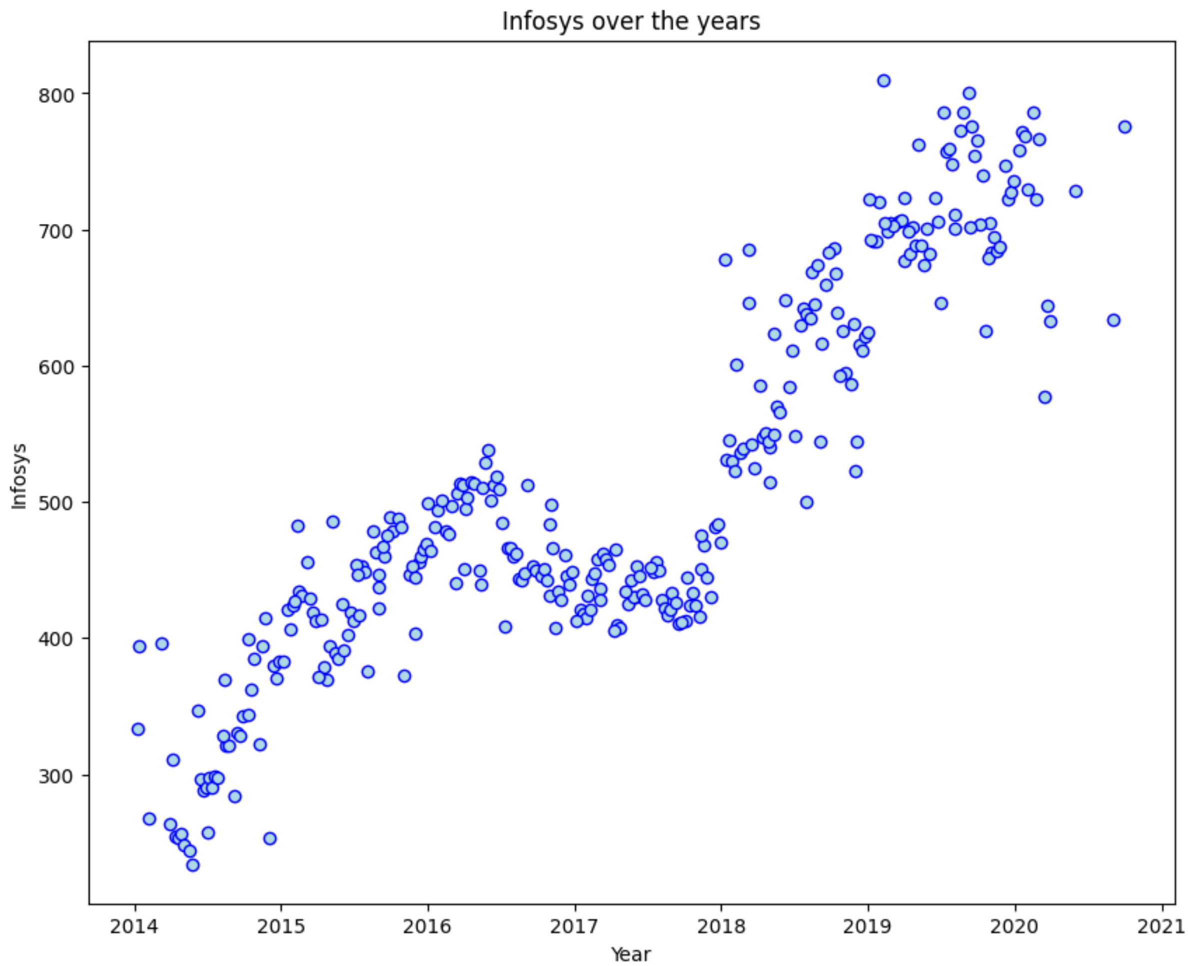
The stock prices of mahindra and mahindra has an increasing pattern from 2014 to

2019 and has steadily declined from 2019 to 2021. Therefore, Mahandra and Mahindra have an average stock price of 535 per stock. Beyond very few outliers, there is a discernible pattern with the stock prices of the company.



STOCK PRICES OF INFOSYS

The stock prices of Infosys have increased steadily from 2014 to mid-2016. The prices then slowly declined till 2018 and began to increase till 2021. The average stock price for Infosys is 511 per stock. Similar to Mahindra and Mahindra, beyond few outliers there is a discernible pattern with the stock prices of the company.



PART B: Calculate Returns for all stocks with inference

	Infosys	Indian_Hotel	Mahindra__Mahindra	Axis_Bank	SAIL	Shree_Cement	Sun_Pharma	Jindal_Steel	Idea_Vodafone	Jet_Airways
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	-0.026873	-0.014599	0.006572	0.048247	0.028988	0.032831	0.094491	-0.065882	0.011976	0.086112
2	-0.011742	0.000000	-0.008772	-0.021979	-0.028988	-0.013888	-0.004930	0.000000	-0.011976	-0.078943
3	-0.003945	0.000000	0.072218	0.047025	0.000000	0.007583	-0.004955	-0.018084	0.000000	0.007117
4	0.011788	-0.045120	-0.012371	-0.003540	-0.076373	-0.019515	0.011523	-0.140857	-0.049393	-0.148846

First step is to calculate the log version of the stock prices to regulate the dataset and calculate the differences from the previous stock price to calculate returns. The first line of the stock returns is null because there are no previous values to calculate the differences from. We can see that there are a lot of negative than positive returns and there are some cases of no returns from the result table.

PART B: Calculate Stock Means and Standard Deviation for all stocks with inference
Following the mean and the standard deviation of stock returns:

MEAN

```
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
```

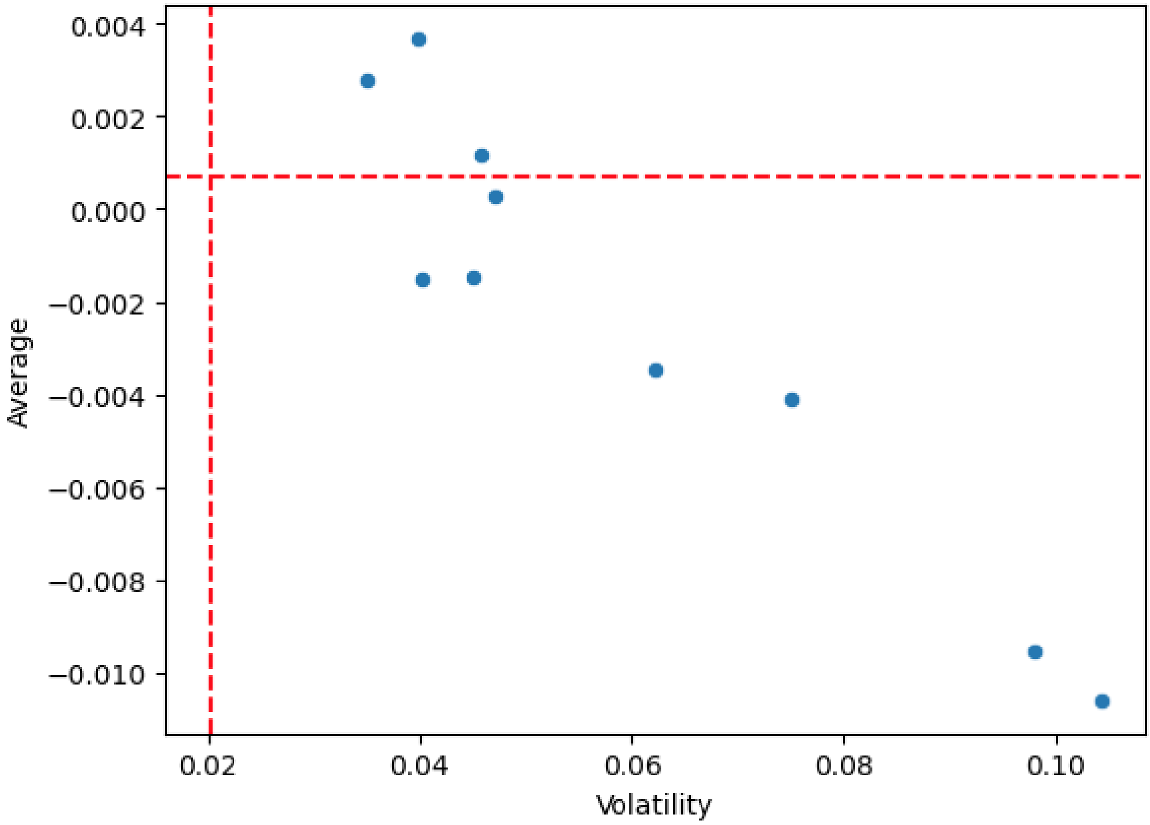
STANDARD DEVIATION

```
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
```

The mean stock returns are the average return a company is making on a regular basis. It measures the volatility of the stock, i.e., if the stock return is most varying from the average stock the more volatile the stock is. Shree cements has the highest average return whereas, Jet Airways has the lowest average return in comparison. Idea and Vodafome have the most volatile stocks and Infosys has the least volatile stock.

PART B: Draw a plot of Stock Means vs Standard Deviation and state your inference.

With the index value set close to 0.02 volatility and 0.000 average, Most companies with low volatility have the highest returns and as the volatility of the stocks increases the average returns from the stocks also decreases. Therefore, with the given set of companies it is better to stick with companies that have low volatility. We must also take into consideration that there are some companies that are below this index line of average returns even if they have low volatility but they are outnumbered by stocks that prived higher average returns.



PART B: Conclusions and Recommendations

- Following are the companies with low volatility and high average returns:

	Average	Volatility
Infosys	0.002794	0.035070
Shree_Cement	0.003681	0.039917
Axis_Bank	0.001167	0.045828
Indian_Hotel	0.000266	0.047131

- These companies could provide much better returns when invested into than the other companies.
- This is because the stock prices of these companies do not oscillate further from their average stock price level and they tend to have positive or no returns even in worst market conditions.
- Thus, investing in these companies' stocks yields better for low risk.
- The profitability with low risk factor is highest with Infosys, thus making it the most profitable stock off of the bundle.