FINANCE RISK ANALYIS

COMPANY CREDIT RISK ANALYSIS

MAY 5, 2024

SHUBHAM KUMAR

CONTENT: - Page No

PROBLEM STATEMENT

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

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

Data that is available includes information from the financial statement of the companies for the previous year.

Dependent variable - No need to create any new variable, as the 'Default' variable is already provided in the dataset, which can be considered as the dependent variable.

Test Train Split - Split the data into train and test datasets in the ratio of 67:33 and use a random state of 42 (random_state=42). Model building is to be done on the train dataset and model validation is to be done on the test dataset.

Basic Information About the Dataset: -

- The dataset has 2058 rows and 58 Columns.
- Out of 58 Columns,
    - 53 Columns are of float64 Datatype
    - 4 Columns are of int64 datatype
    - 1 column is object Datatype.
- Few Columns has null entries such as
    - _Cash_Flow_Per_Share,
    - _To_Total_debt_to_Total_net_worth
    - _Cash_to Total_Assets,
    - _Current_Liability_to_Current_Assets
- Looking at the Descriptive Statistics we assume that the dataset has an outliers also (for Descriptive Stats please refer jupyter notebook)

PART A.1 : Outlier Treatment

Checking Outliers using Boxplots: -

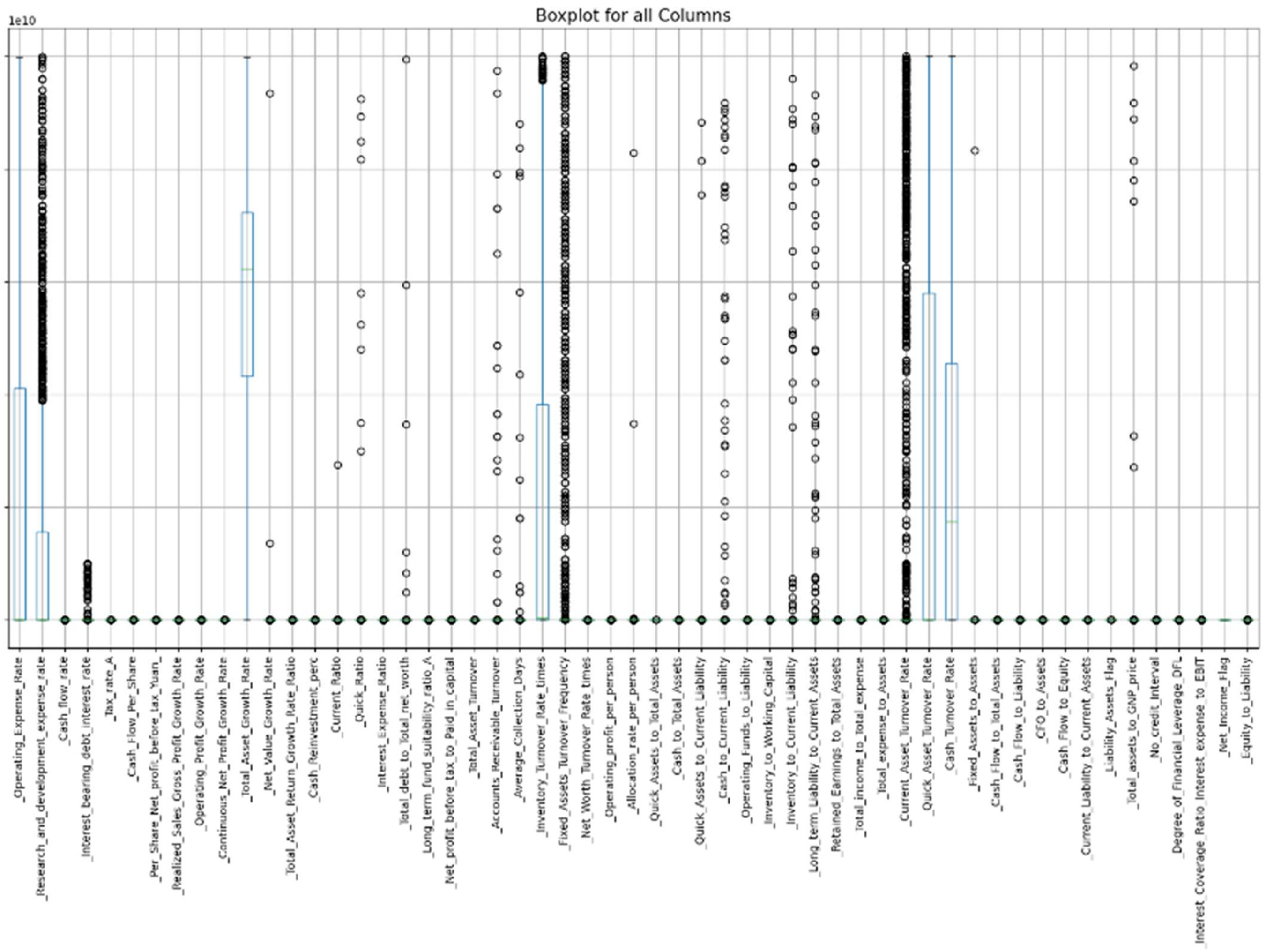


Fig. 1.A

Boxplot of all the numeric columns

From the Above Boxplot we can see that almost all the columns have outliers.

Total Number of Outliers in a dataset = 10864

Percentage of Outliers in a dataset = 0.09427

In order to treat these outliers, we have considered IQR (Inter-Quartile Method)

In IQR Method, we do capping and flooring: -

Q1 = 25 % of the data Q3 = 75 % of the data

IQR = Q3 – Q1

Capping or Upper Limit = Q3 + (1.5 * IQR)

Flooring or Lower Limit = Q1 – (1.5 * IQR)

Here all the value of a column which are above the Upper Limit will be replaced with Upper Limit and all the values of the column which are below the Lower Limit will be replaced with the lower limit.

In order to Calculate Upper and Lower limit we have defined the defined a function remove_outliers.

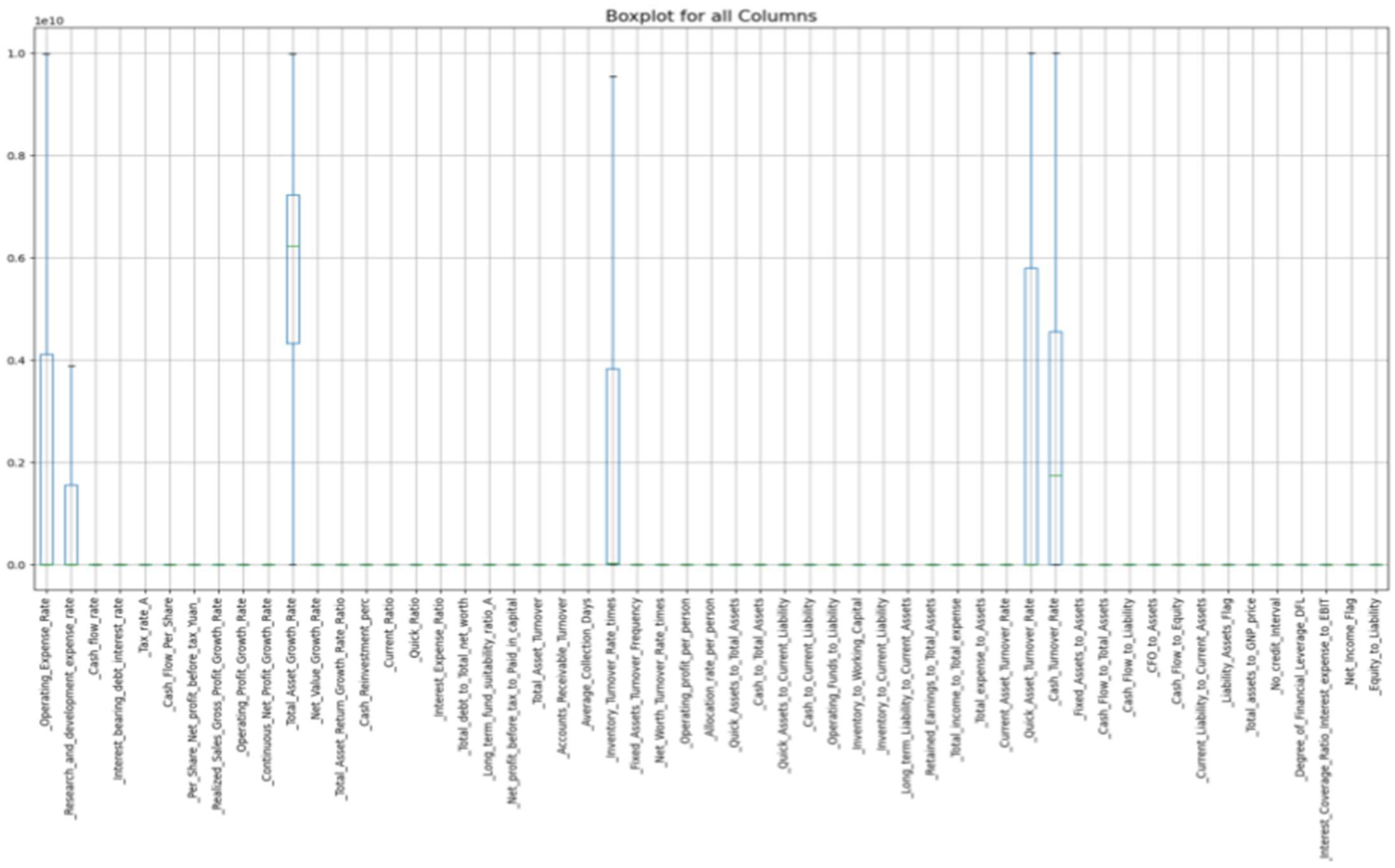
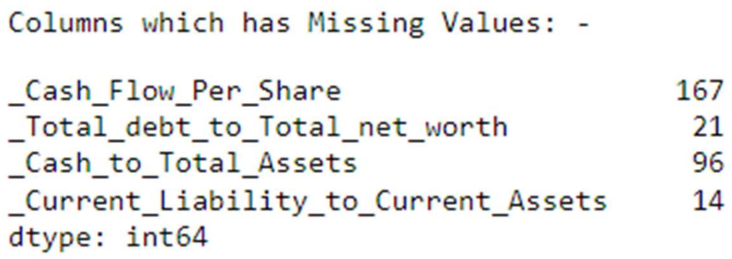


Fig 1.B

Boxplot After Treating Outliers

PART A.2 : Missing Value Treatment

Check for Missing Values: -



```
Columns which has Missing Values: -

_Cash_Flow_Per_Share                      167
_Total_debt_to_Total_net_worth             21
_Cash_to_Total_Assets                      96
_Current_Liability_to_Current_Assets       14
dtype: int64
```

Fig. 2.A

Missing Values Check

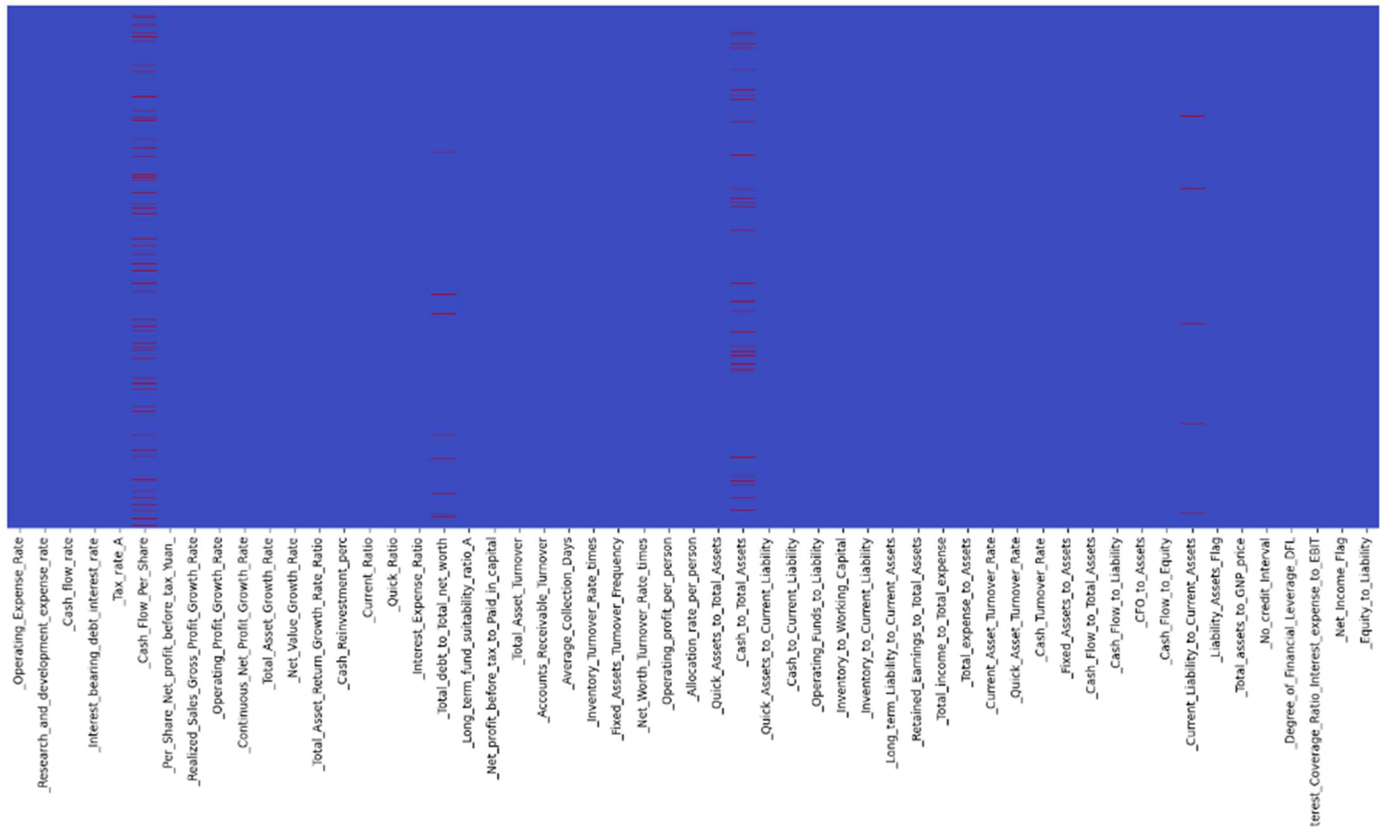


Fig. 2B

Visually Representing Missing Values in a dataset

Total Number of Missing Values: - 298

Percentage of missing values in a dataset: - 0.00249656512851446

For Missing Values Treatment, we considered KNN Imputer. It is stored in sklearn library

In this approach, we specify a distance from the missing values which is also known as the K parameter. The missing value will be predicted in reference to the mean of the neighbors.

Since KNN Imputer is a distance based algorithm, hence we need to scale our predictor variables to same scale. For Scaling we have considered StandardScaler from SKlearn library.

Also KNN imputer for train and test data is done separately.

For Train Dataset, we have used fit_tranform method while imputation using KNN Imputer

For Test Dataset, we have used transform method while imputation using KNN Imputer

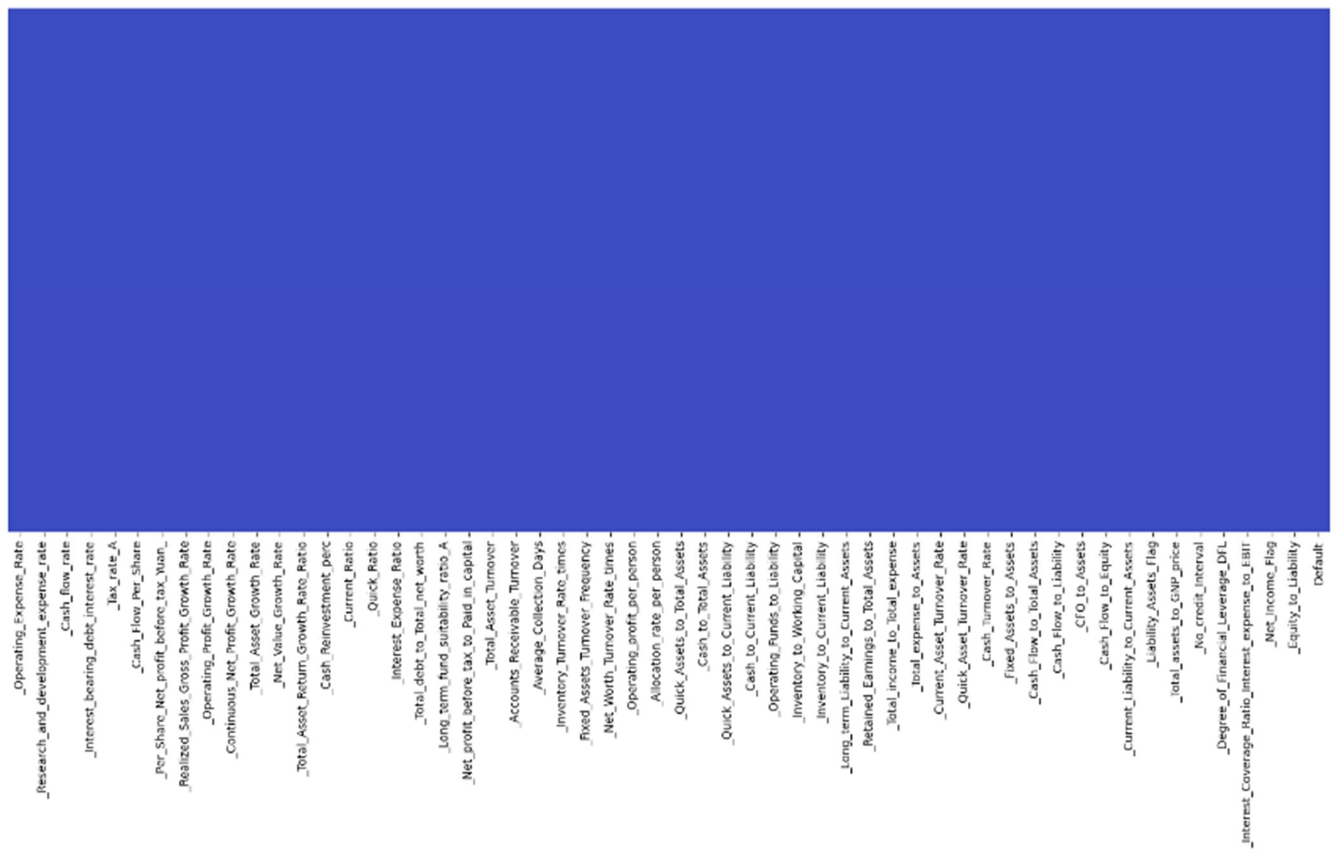
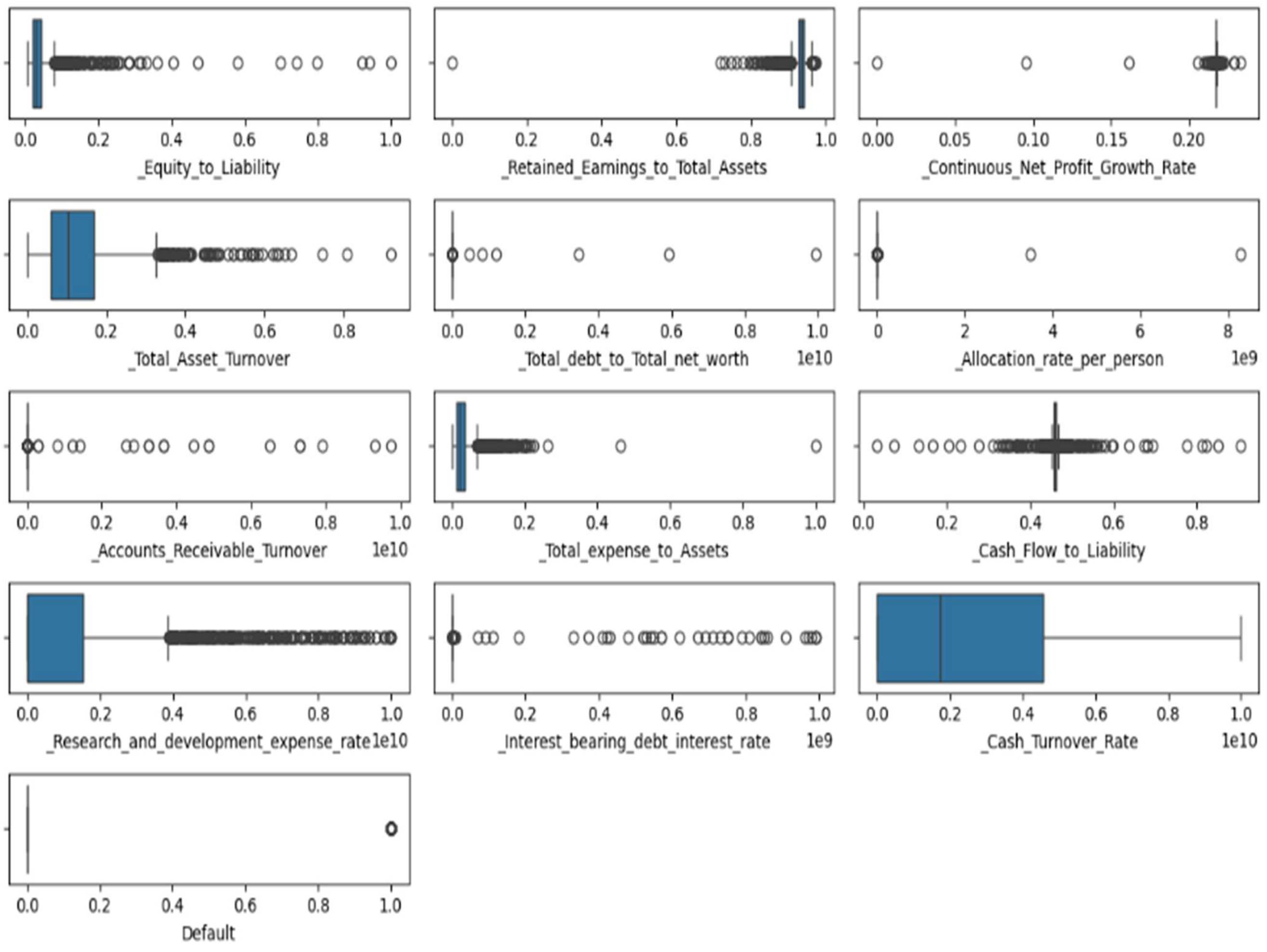


Fig. 2.C

Visually Representing Dataset after Missing Values Imputation

PART A: Univariate (4 marks) & Bivariate (6 marks) analysis with proper interpretation. (You may choose to include only those variables which were significant in the model building)

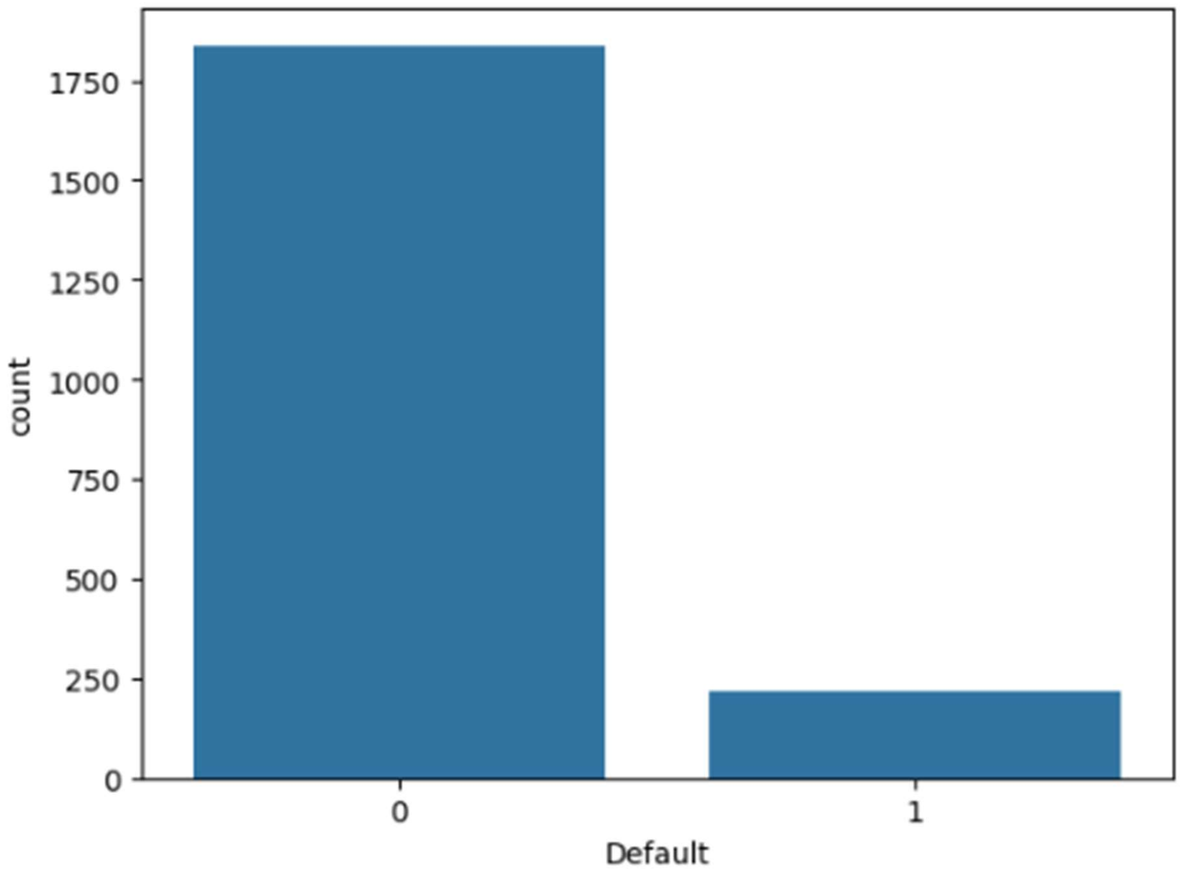
UNIVARIATE ANALYSIS



Inferences: -

- **Almost all the columns have an outlier except Cash Turnover Rate**
- **There are large number of outliers**
- **Data distribution of each of the columns are either right skewed or left skewed, none of the column data are normally distributed**

- **Default variable Univariate analysis**

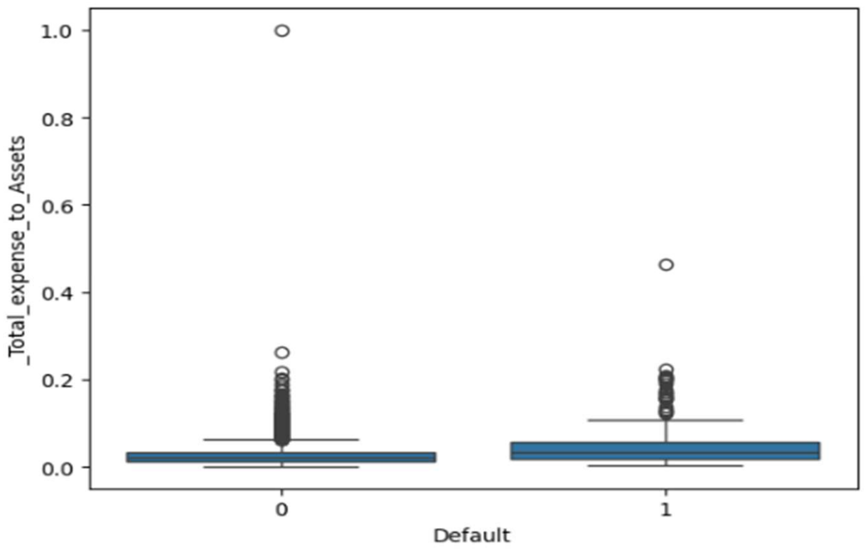


Inferences: -

- Total Number of Defaulters = 220
- Total Number of Non Defaulters = 1838

Multivariate Analysis

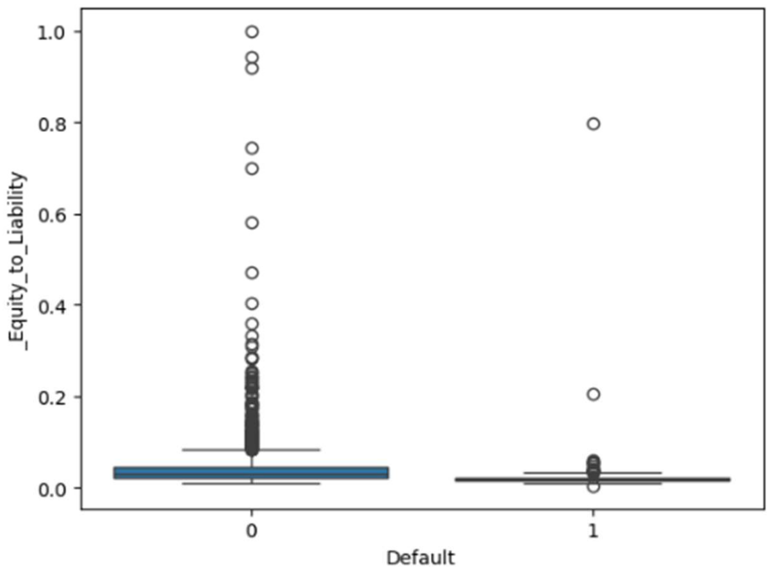
1. **Default Vs _Total_expense_to_Assets**



Inferences: -

Non Defaulters have Lower Total_expense_to_Assets as compared to Defaulters (exception for Outliers)

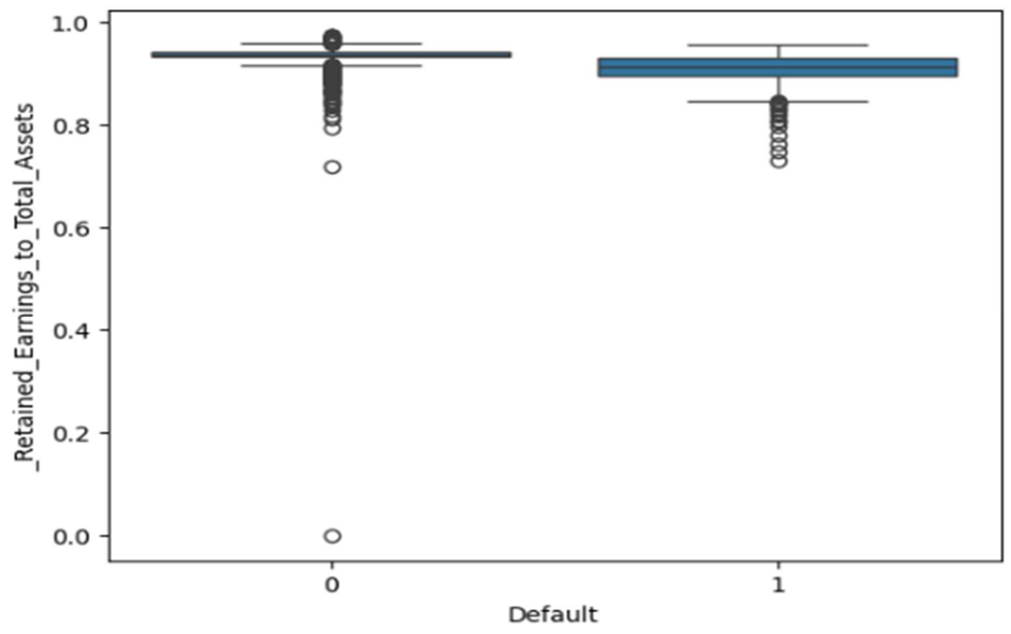
2. **Default vs _Equity_to_Liability**



Inferences: -

Non Defaulters have higher _Equity_to_Liability as compared to Non Defaulters

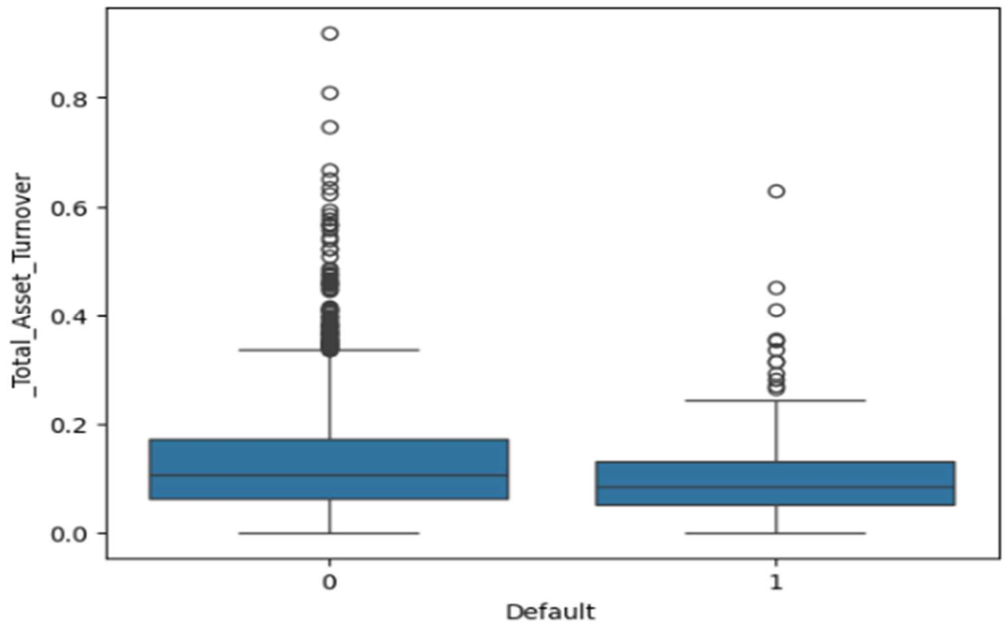
3. **Default vs _Retained_Earnings_to_Total_Assets**



Inferences: -

Non Defaulters have higher _Retained_Earnings_to_Total_Assets as compared to Defaulters

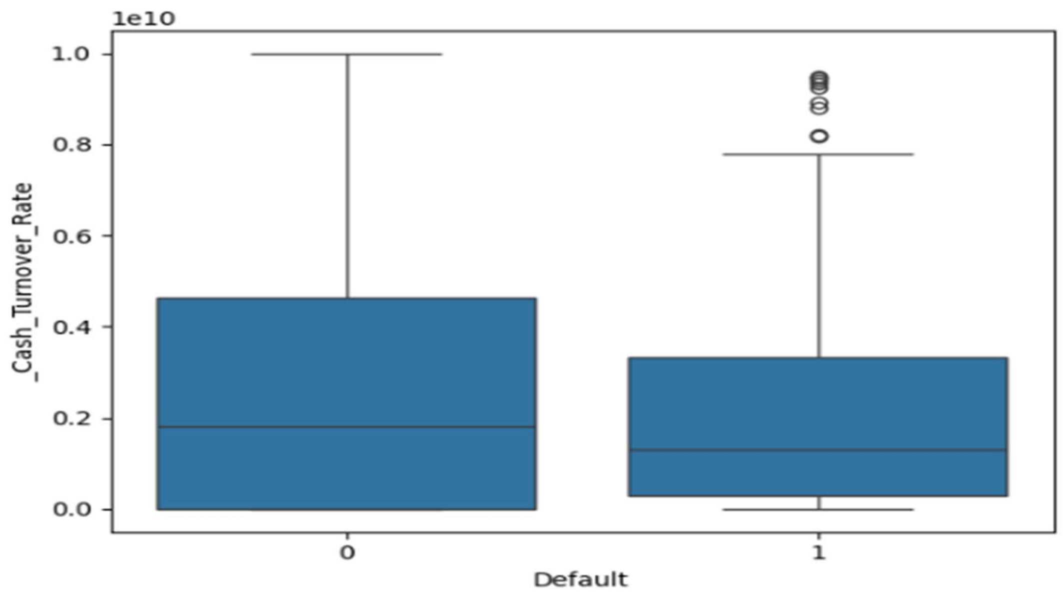
4. **Default vs _Total_Asset_Turnover**



Inferences: -

Defaulters have lower _Total_Asset_Turnover as compared to Non Defaulter

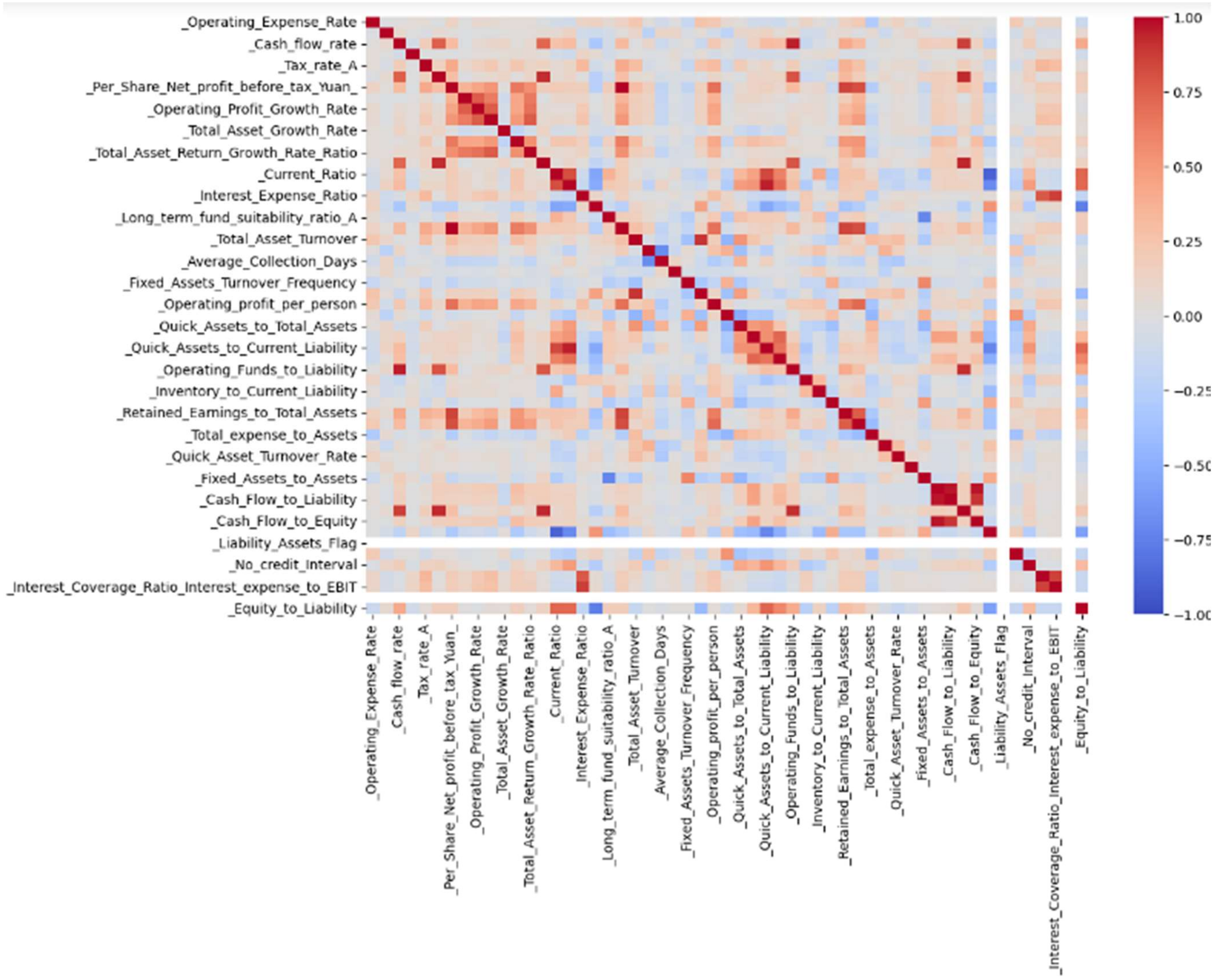
5. Default Vs Cash_Turnover_Rate



Inferences: -

Non Defaulters have higher Cash_turnover_Rate as compare to Defaulters

Multivariate Analysis of All the Numeric Variables: -



Inferences: -

From above Heatmap we can see there are several variables which having high correlation between themselves

PART A.4 : Train Test Split

Here we have split the dataset into 67:33 ratio i.e., 67 % dataset as train and 33% dataset as test.

Checking the number of defaulters in each dataset after split i.e., train and test dataset

Total Number of defaulters in a dataset = 220.

Number of Defaulter in Train dataset = 147 i.e., 67 % of Total defaulters

Number of Defaulter in Test dataset = 73 i.e., 33 % of Total Defaulters

PART A.5 : Build Logistic Regression Model (using statsmodels library) on most important variables on train dataset and choose the optimum cut-off. Also showcase your model building approach

<u>Logistic Regression Using statsmodels libraray</u>

Since Logistic Regression Model are sensitive to Multicolinearity, so we check for correlation first using heatmap.

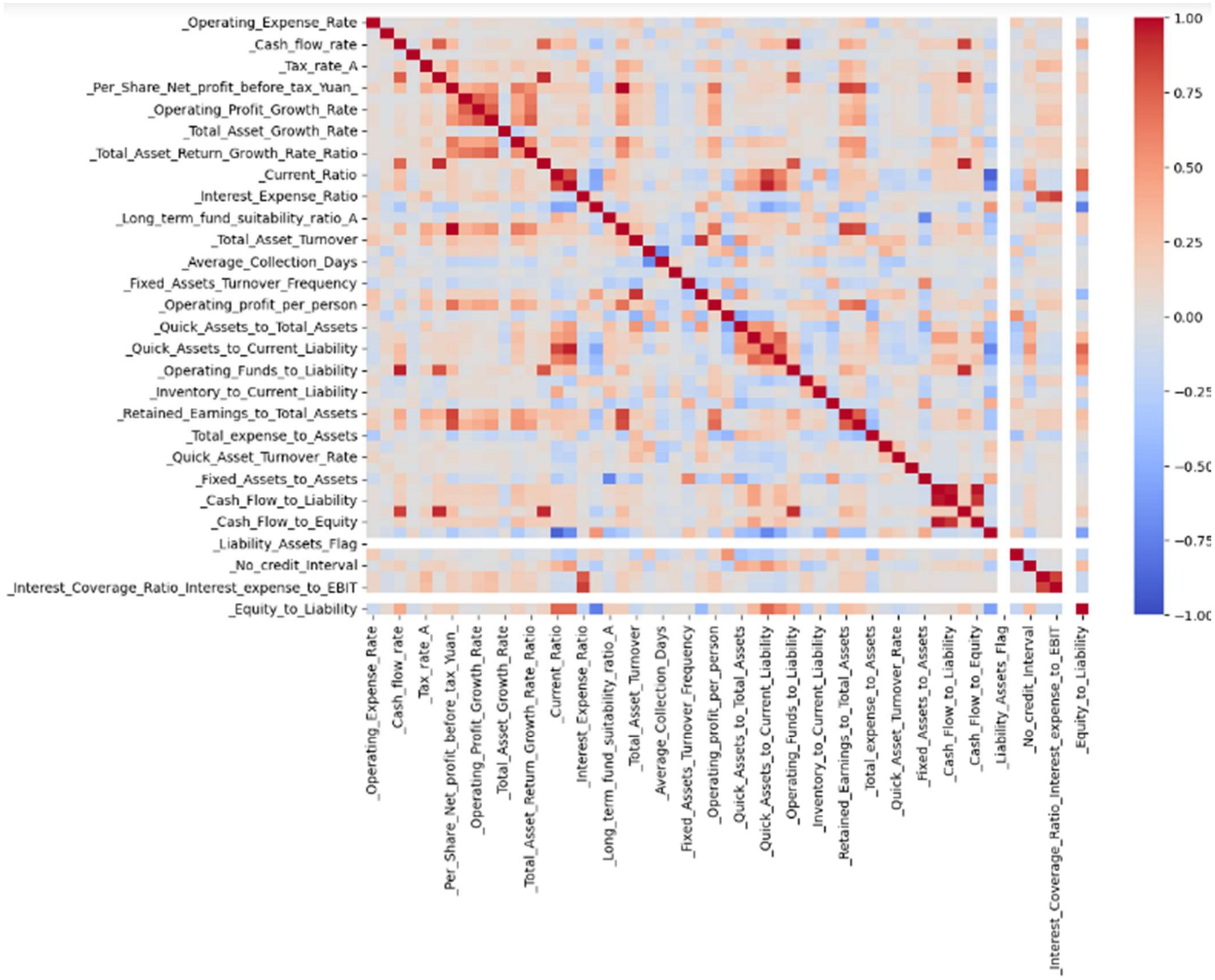


Fig. 5.A

Correlation plot between all the variables

From the above Correlation plot we can see that some of the variables are highly correlated with each other. We need treat this as logistic regression model is sensitive to multicolinearity

In order removed the curse of multicolinearity from our dataset we have used the Variation Inflation Factor (VIF) from statsmodel.stats.outliers_influence libraray.

For VIF we have set a cut of 5, which means we will drop variables which has VIF values higher than 5 one by one. We have calculated VIF, drop the variable which has the highest VIF (highest compared to 5) and again calculated VIF. We repeated this until we get variables having VIF equal to or less than 5.

So During Treatment for Multicolinearity we were able to drop 13 variables which were highly correlated with each other.

Variables which are dropped due to Multicolinearity in our dataset: -

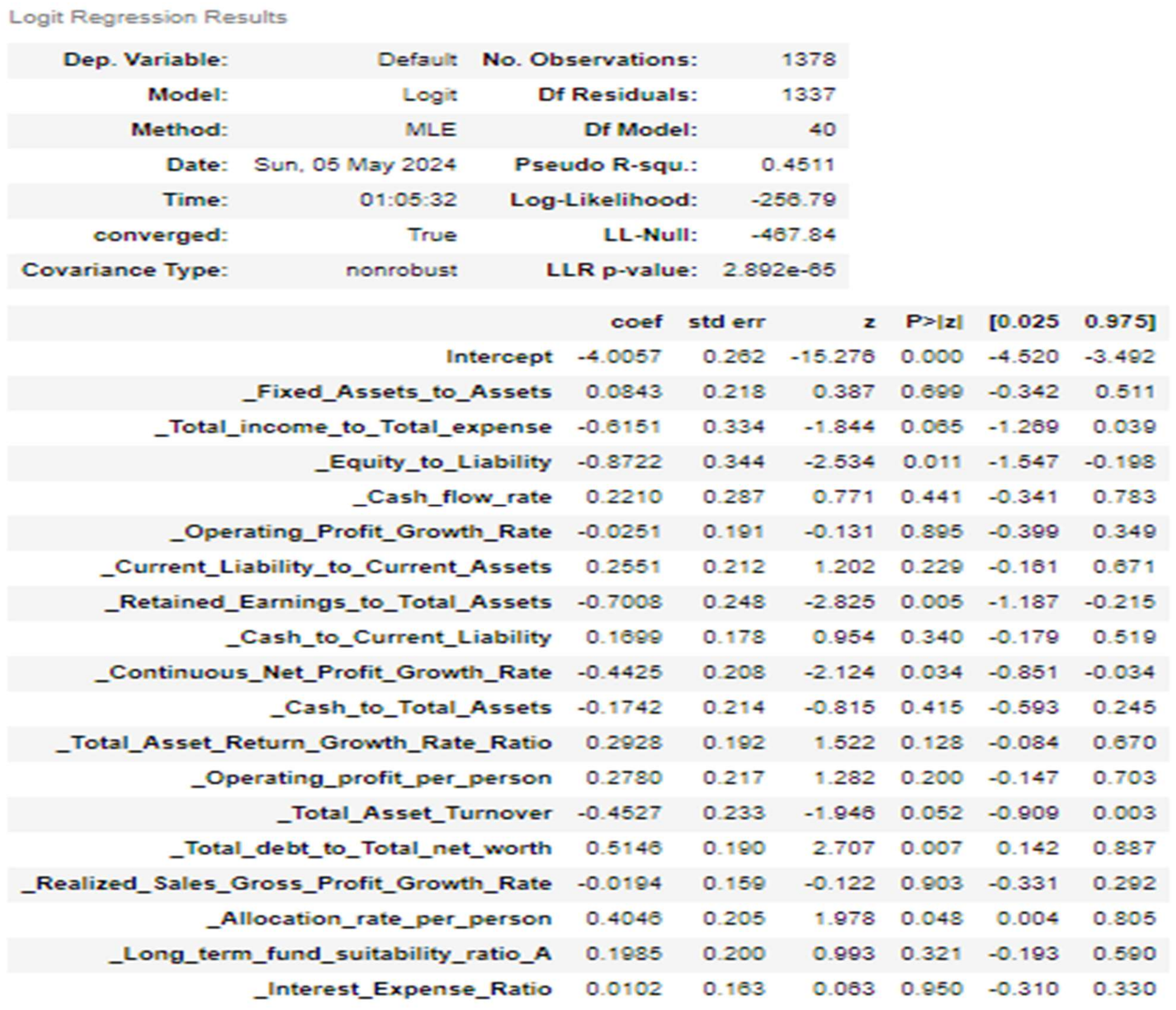
‘_Per_Share_Net_profit_before_tax_Yuan_’ , ‘_Cash_Flow_to_Total_Assets’ , ‘_CFO_to_Assets’ , ‘_Quick_Assets_to_Current_Liability’, ‘_Operating_Funds_to_Liability’, ‘_Net_Worth_Turnover_Rate_times’, ‘_Current_Ratio’, ‘_Cash_Flow_Per_Share’ , ‘_Net_profit_before_tax_to_Paid_in_capital’, ‘_Interest_Coverage_Ratio_Interest_expense_to_EBIT’, ‘_Quick_Ratio’ . ‘_Cash_Flow_to_Equity’ , ‘_Quick_Assets_to_Total_Assets’ , ‘

Also 2 Variables have consistent values along the columns. Hence we will drop these columns also '_Liability_Assets_Flag' , '_Net_Income_Flag' .

Note: - Refers to Jupyter Notebook for how we have calculated and dropped the variables.

After treating for curse of multicolinearity we have 40 variables. We have prepared the basic model using these 40 variables.

Basic logistic regression model using statsmodel: -



Logit Regression Results

Dep. Variable:	Default	No. Observations:	1378
Model:	Logit	Df Residuals:	1337
Method:	MLE	Df Model:	40
Date:	Sun, 05 May 2024	Pseudo R-squ.:	0.4511
Time:	01:05:32	Log-Likelihood:	-256.79
converged:	True	LL-Null:	-467.84
Covariance Type:	nonrobust	LLR p-value:	2.892e-65

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-4.0057	0.262	-15.276	0.000	-4.520	-3.492
_Fixed_Assets_to_Assets	0.0843	0.218	0.387	0.699	-0.342	0.511
_Total_income_to_Total_expense	-0.6151	0.334	-1.844	0.065	-1.269	0.039
_Equity_to_Liability	-0.8722	0.344	-2.534	0.011	-1.547	-0.198
_Cash_flow_rate	0.2210	0.287	0.771	0.441	-0.341	0.783
_Operating_Profit_Growth_Rate	-0.0251	0.191	-0.131	0.895	-0.399	0.349
_Current_Liability_to_Current_Assets	0.2551	0.212	1.202	0.229	-0.161	0.671
_Retained_Earnings_to_Total_Assets	-0.7008	0.248	-2.825	0.005	-1.187	-0.215
_Cash_to_Current_Liability	0.1699	0.178	0.954	0.340	-0.179	0.519
_Continuous_Net_Profit_Growth_Rate	-0.4425	0.208	-2.124	0.034	-0.851	-0.034
_Cash_to_Total_Assets	-0.1742	0.214	-0.815	0.415	-0.593	0.245
_Total_Asset_Return_Growth_Rate_Ratio	0.2928	0.192	1.522	0.128	-0.084	0.670
_Operating_profit_per_person	0.2780	0.217	1.282	0.200	-0.147	0.703
_Total_Asset_Turnover	-0.4527	0.233	-1.946	0.052	-0.909	0.003
_Total_debt_to_Total_net_worth	0.5146	0.190	2.707	0.007	0.142	0.887
_Realized_Sales_Gross_Profit_Growth_Rate	-0.0194	0.159	-0.122	0.903	-0.331	0.292
_Allocation_rate_per_person	0.4046	0.205	1.978	0.048	0.004	0.805
_Long_term_fund_suitability_ratio_A	0.1985	0.200	0.993	0.321	-0.193	0.590
_Interest_Expense_Ratio	0.0102	0.163	0.063	0.950	-0.310	0.330

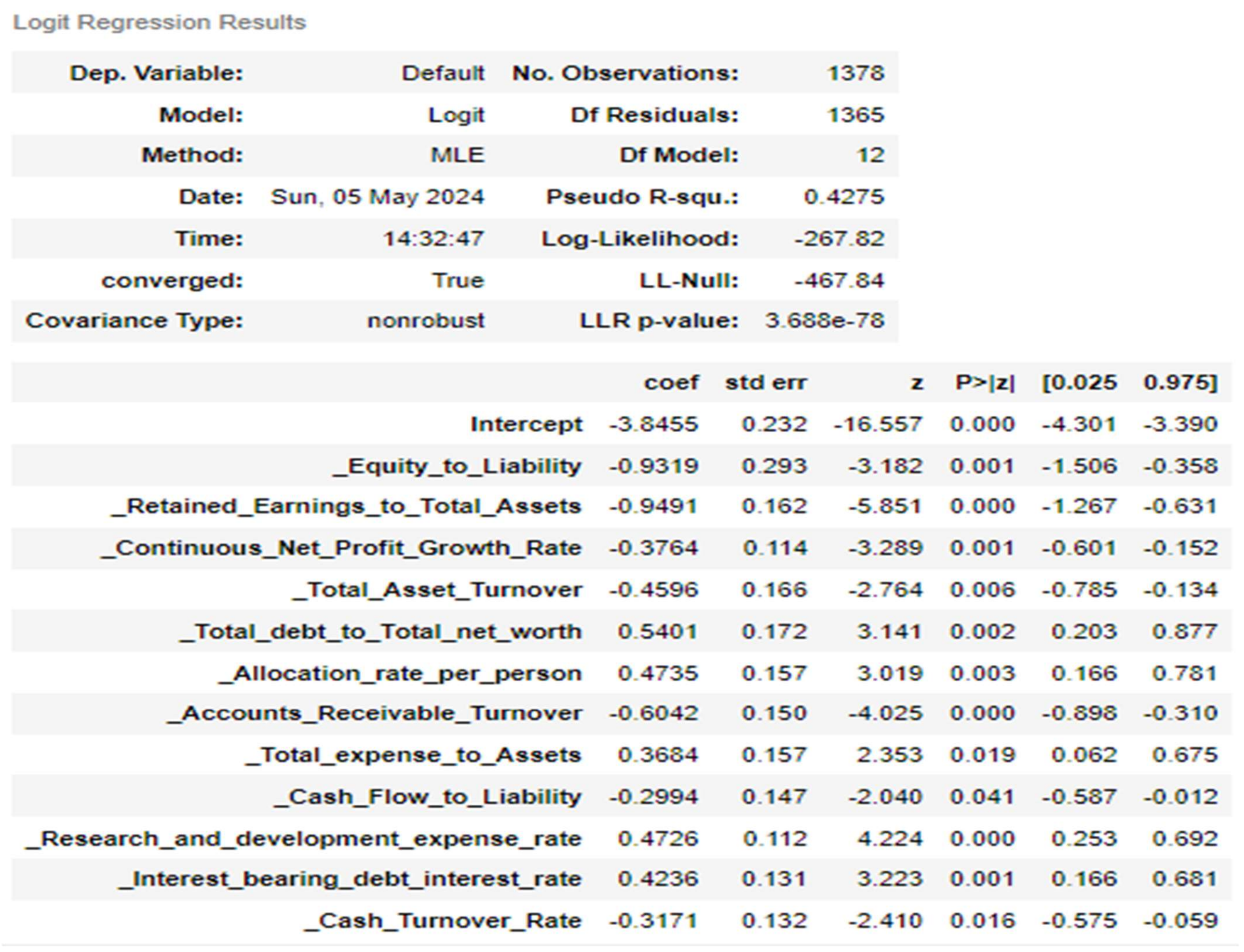
Fig. 5B

Summary of Basic Logistic Regression Model using statsmodel (Sample)

Note: - For complete summary please refer Jupyter Notebook

From above Fig 5B we can see that many variables have p_values greater than 0.05 whereas for optimum model p_value should be equal to or less than 0.05 . Hence we will drop variables having p_values greater than 0.05 one by one (from highest to lowest) in order to get an optimum or best model.

After repeated process of creating a model and dropping the variables (greater than 0.05 one by one), we were able to drop

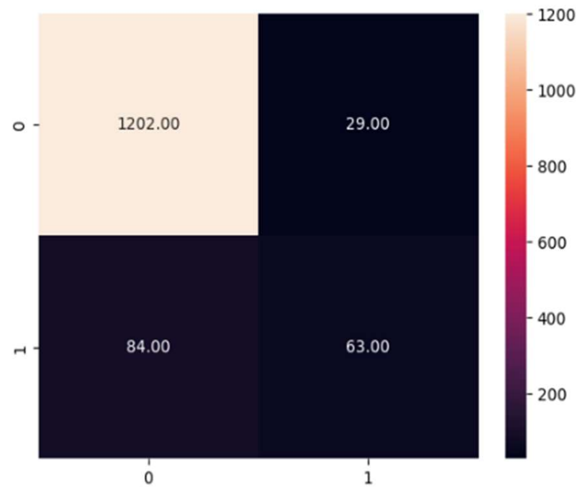


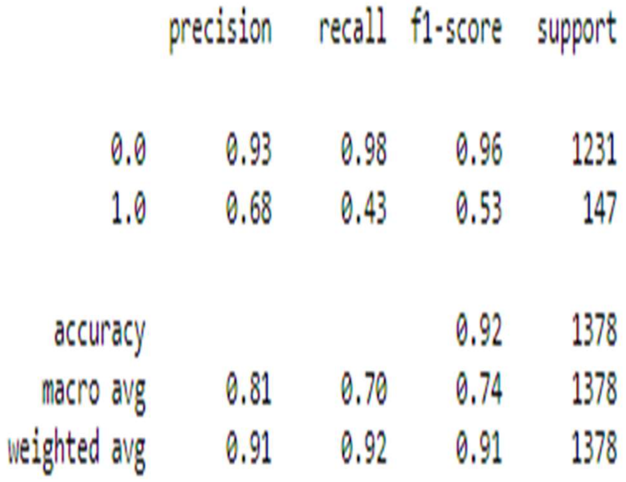
```
Logit Regression Results

      Dep. Variable:          Default   No. Observations:       1378
             Model:             Logit       Df Residuals:       1365
            Method:               MLE           Df Model:         12
              Date:  Sun, 05 May 2024     Pseudo R-squ.:      0.4275
              Time:          14:32:47     Log-Likelihood:    -267.82
         converged:              True            LL-Null:    -467.84
   Covariance Type:         nonrobust       LLR p-value:   3.688e-78

                                          coef   std err        z   P>|z|   [0.025   0.975]
                              Intercept  -3.8455   0.232   -16.557  0.000   -4.301   -3.390
                     _Equity_to_Liability -0.9319   0.293    -3.182  0.001   -1.506   -0.358
       _Retained_Earnings_to_Total_Assets -0.9491   0.162    -5.851  0.000   -1.267   -0.631
     _Continuous_Net_Profit_Growth_Rate  -0.3764   0.114    -3.289  0.001   -0.601   -0.152
                   _Total_Asset_Turnover -0.4596   0.166    -2.764  0.006   -0.785   -0.134
          _Total_debt_to_Total_net_worth  0.5401   0.172     3.141  0.002    0.203    0.877
               _Allocation_rate_per_person  0.4735   0.157     3.019  0.003    0.166    0.781
          _Accounts_Receivable_Turnover  -0.6042   0.150    -4.025  0.000   -0.898   -0.310
               _Total_expense_to_Assets   0.3684   0.157     2.353  0.019    0.062    0.675
                 _Cash_Flow_to_Liability -0.2994   0.147    -2.040  0.041   -0.587   -0.012
   _Research_and_development_expense_rate  0.4726   0.112     4.224  0.000    0.253    0.692
      _Interest_bearing_debt_interest_rate  0.4236   0.131     3.223  0.001    0.166    0.681
                     _Cash_Turnover_Rate -0.3171   0.132    -2.410  0.016   -0.575   -0.059
```

Fig. 5C

Final Logistic Regression Model Summary (with most important variables)

Model Performance on Train data : -





```
              precision    recall  f1-score   support

         0.0       0.93      0.98      0.96      1231
         1.0       0.68      0.43      0.53       147

    accuracy                           0.92      1378
   macro avg       0.81      0.70      0.74      1378
weighted avg       0.91      0.92      0.91      1378
```

Fig. 5D

Classification Report and Confusion matrix with Best Logistic Regression (Train Data)

Inferences: -

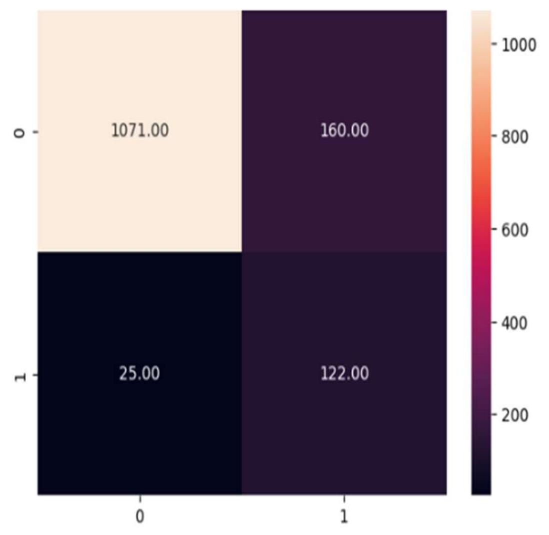
- For Train Data out of 147 number of Defaulters, 63 were actually predicted as 1's (Yes)
- Recall of 1 = 0.43
- From above reference we can see that our model is not performing well

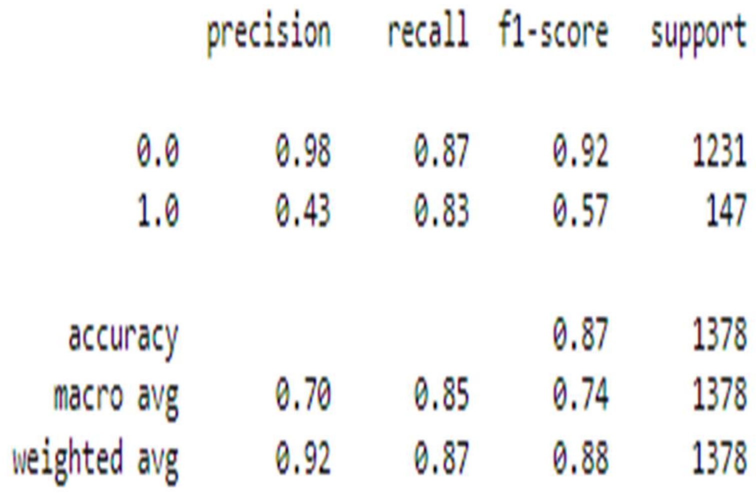
Since our model is not performing well we choose to shift our cut off and check if our model performance increases or not. Previously we set our cut off to 0.5 which means if a probability is greater than 0.5 we have replaces with 1 (Defaulted) and less than or equal to 0.5 with 0 (Not Defaulted)

Now we shift our Cut off from 0.5 to 0.13275. If the probabibility is greater than 0.13 we replaced it with 1 (Defaulted) and leass than 0.13 has been replaced with 0 (Not Defaulted)

Note: - In Order to check for calculation behind shifting Optimum Cut off from 0.5 to 0.13275 , please refer Jupyter notebook.

Calculating Model Performance after shifting Optimum Cut-Off from 0.5 to 0.132 on train data





```
              precision    recall  f1-score   support

         0.0       0.98      0.87      0.92      1231
         1.0       0.43      0.83      0.57       147

    accuracy                           0.87      1378
   macro avg       0.70      0.85      0.74      1378
weighted avg       0.92      0.87      0.88      1378
```

Fig. 5E

Train Data Classification Report and Confusion matrix with Best Logistic Regression Cut off (0.132)

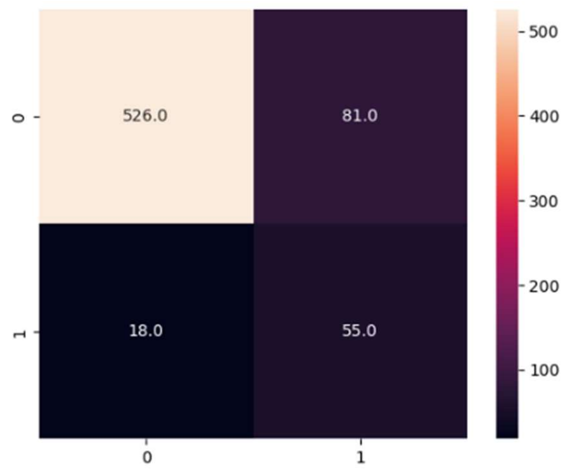
Inferences: -

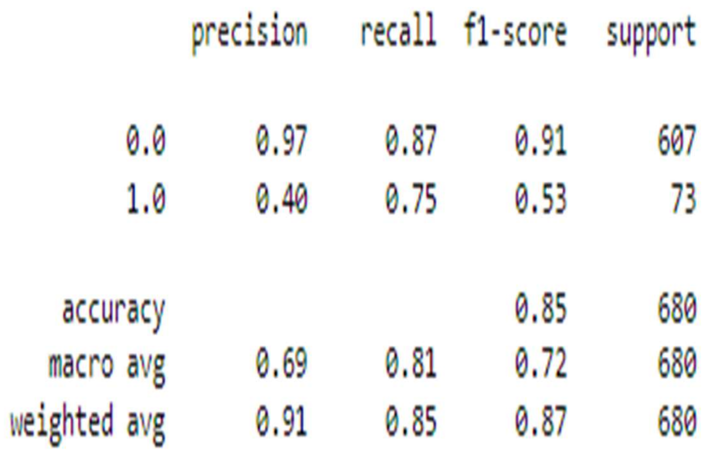
- For Train Data out of 147 number of Defaulters, 122 were actually predicted as 1's (Yes)
- Recall of 1 (Defaulted) = 0.83
- Although the precision has come down but Recall of 1 (Default) has increased from 0.43 to 0.83

PART A.6 : Validate the Model on Test Dataset and state the performance metrics. Also state interpretation from the model

Using Best Logistic Regression Model, we made probability prediction for the test data. Setting optimum cutoff to 0.132 we made prediction on test data where the company will default or not.

<u>Calculating Model Performance after shifting Optimum Cut-Off from 0.5 to 0.132 on test data</u>





```
              precision    recall  f1-score   support

         0.0       0.97      0.87      0.91       607
         1.0       0.40      0.75      0.53        73

    accuracy                           0.85       680
   macro avg       0.69      0.81      0.72       680
weighted avg       0.91      0.85      0.87       680
```

Fig. 6A

Classification Report and Confusion matrix with Best Logistic Regression (Test Data)

Inferences: -

- For Test Data out of 73 number of Defaulters, 55 were actually predicted as 1's (Defaulted)
- Recall of 1 (Defaulted) = 0.75
- We can see that the model performance for Recall of 1 (Default) is quite significant i.e., 0.75

Interpretation: -

In financial credit risk analysis, a high recall value and accepting a lower precision value can be a reasonable trade-off.

The precision and Recall values in both test and train sets are comparable and hence we can say that we have a fairly good model not showing presence of any overfitting and a Recall score of around 78%

VIF has been used to handle the multicolinearity in the model whereas the feature selection has been further improved by using the p-value to identify and eliminate any non-significant features from the model

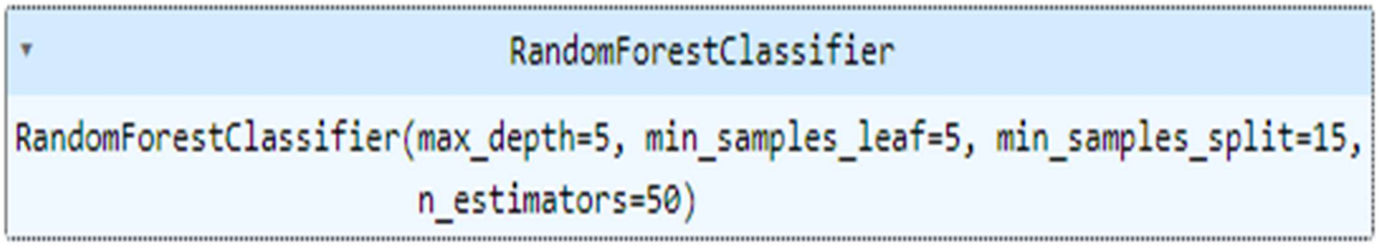
PART A.7 : Build a Random Forest Model on Train Dataset. Also showcase your model building approach

Random Forest algorithm is a powerful tree learning technique in Machine Learning. It works by creating a number of Decision Trees during the training phase. Each tree is constructed using a random subset of the data set to measure a random subset of features in each partition. This randomness introduces variability among individual trees, reducing the risk of overfitting and improving overall prediction performance.

In order to find the optimum values of the parameters for random forest classifier we have used GridSearchCV from sklearn.model_selection library.

GridSearchCV is a hyper tunning parameters which is used in order to find the optimum value of each parameters for the models.

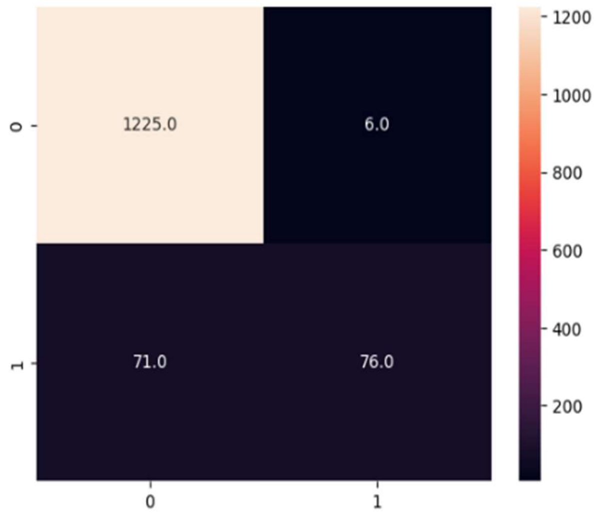
<u>Best Parameters after applying GridSearchCV: -</u>



```
▾                         RandomForestClassifier
RandomForestClassifier(max_depth=5, min_samples_leaf=5, min_samples_split=15,
                       n_estimators=50)
```

Using the above values on Random Forest Classifier Model parameters, we had created the model

Random Forest Classifier Model Confusion Matrix and Classification Report on train data



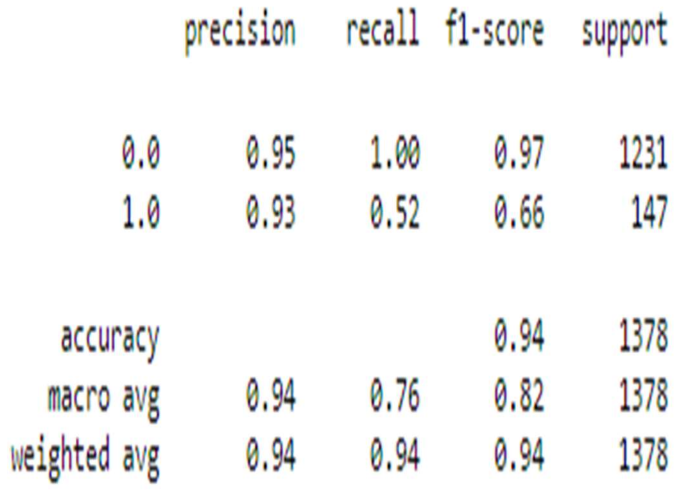

```
              precision    recall  f1-score   support

         0.0       0.95      1.00      0.97      1231
         1.0       0.93      0.52      0.66       147

    accuracy                           0.94      1378
   macro avg       0.94      0.76      0.82      1378
weighted avg       0.94      0.94      0.94      1378
```

Fig. 7A

Train Data Classification Report and Confusion matrix with Random Forest Classifier (with GridSearchCV)

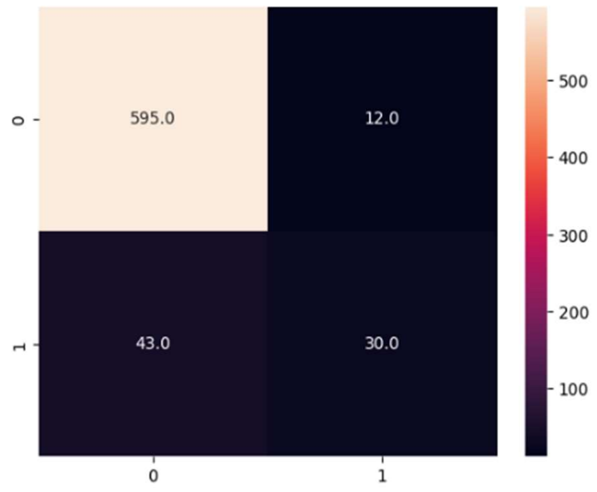
Inferences: -

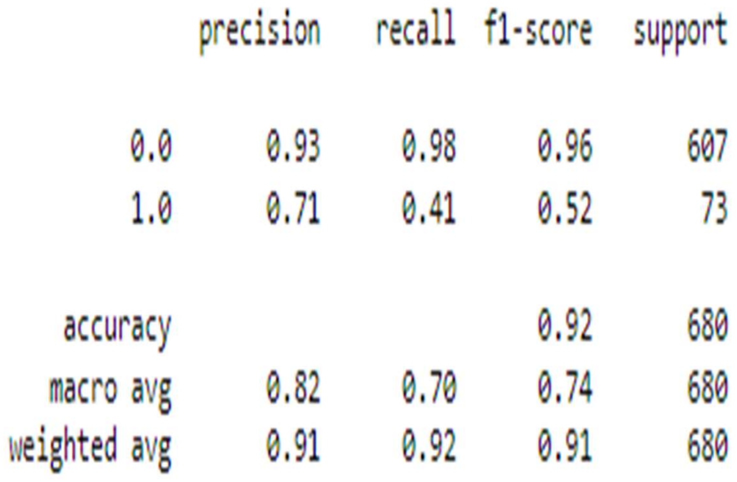
- For Train Data out of 147 number of Defaulters, 71 were actually predicted as 1 (Defaulted)
- Recall of 1 (Defaulted) = 0.52
- Model is having very low recall .

PART A.8 : Validate the Random Forest Model on test Dataset and state the performance metrics. Also state interpretation from the model

Using Random Forest Model (with GridSearchCV), we made prediction for the test data.

<u>**Calculating Model Performance on test for Random Forest Classifier Model (with GridSearchCV)**</u>





```
              precision    recall  f1-score   support

         0.0       0.93      0.98      0.96       607
         1.0       0.71      0.41      0.52        73

    accuracy                           0.92       680
   macro avg       0.82      0.70      0.74       680
weighted avg       0.91      0.92      0.91       680
```

Fig. 8A

Classification Report and Confusion matrix with Random Forest Classifier (with GridSearchCV) on test data

Inferences: -

- For Test Data out of 73 number of Defaulters, 30 were actually predicted as 1's (Defaulted)
- Recall of 1 (Defaulted) = 0.41
- We can see that the model performance for Recall of 1 (Default) is quite similar 0.41

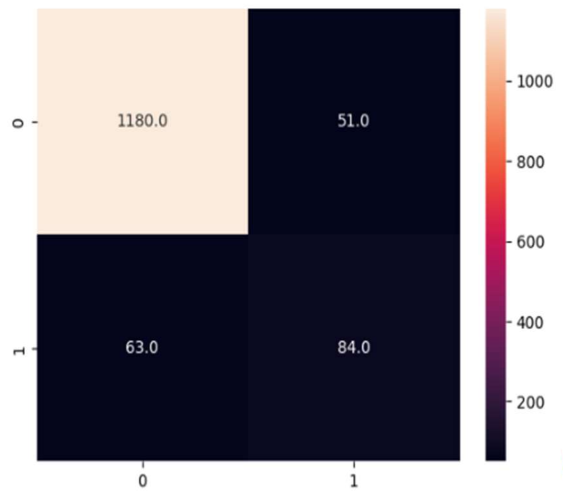
Interpretation: -

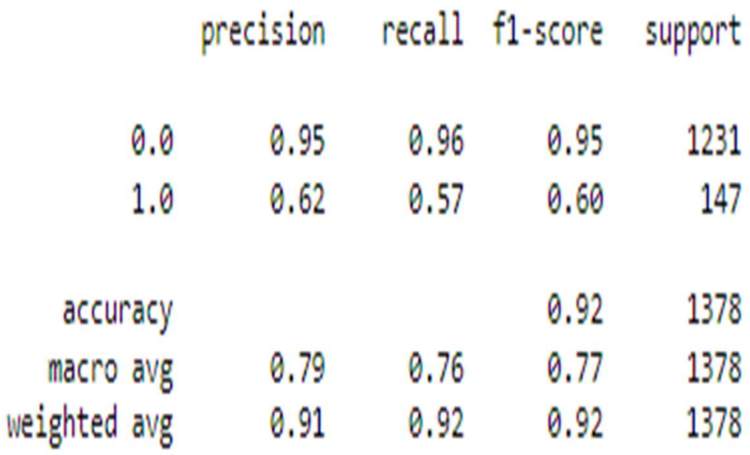
From above model performance on train and test we can say that it is poor model .

PART A.9 : Build a LDA Model on Train Dataset. Also showcase your model building approach

Linear Discriminant Analysis (LDA), also known as Normal Discriminant Analysis or Discriminant Function Analysis, is a dimensionality reduction technique primarily utilized in supervised classification problems

LDA Model Confusion Matrix and Classification Report on train data





```
              precision    recall  f1-score   support

         0.0       0.95      0.96      0.95      1231
         1.0       0.62      0.57      0.60       147

    accuracy                           0.92      1378
   macro avg       0.79      0.76      0.77      1378
weighted avg       0.91      0.92      0.92      1378
```

Fig. 9A

Classification Report and Confusion matrix with Random Forest Classifier on train data

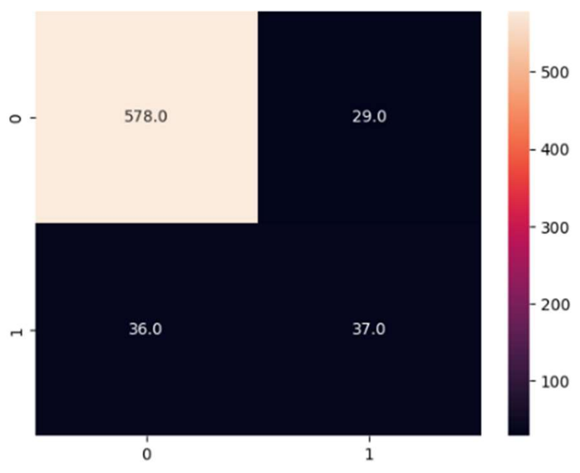
Inferences: -

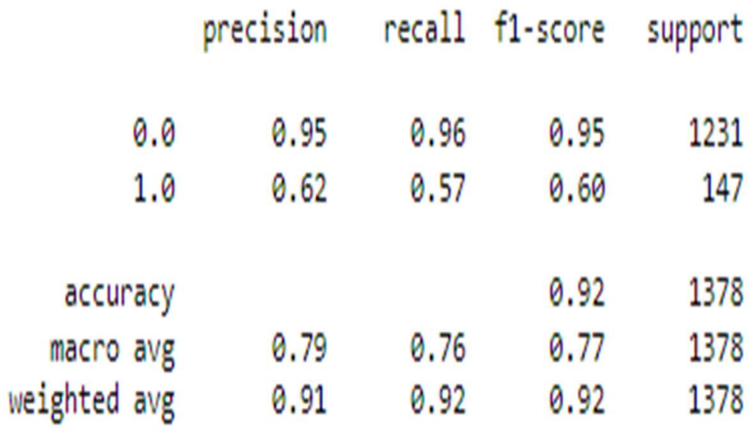
- For Train Data out of 147 number of Defaulters, 84 were actually predicted as 1 (Defaulted)
- Recall of 1 (Defaulted) = 0.57
- Model performance is quite average as we can see its recall is only 0.57, which means nearly half of the defaulters are predicted as non-defaulters

PART A.11 : Validate the LDA Model on test Dataset and state the performance metrics. Also state interpretation from the model

LDA Model Performance on Test Data: -

<u>**Calculating Model Performance on test for LDA Model**</u>





```
              precision    recall  f1-score   support

         0.0       0.95      0.96      0.95      1231
         1.0       0.62      0.57      0.60       147

    accuracy                           0.92      1378
   macro avg       0.79      0.76      0.77      1378
weighted avg       0.91      0.92      0.92      1378
```

Fig. 10A

Classification Report and Confusion matrix with LDA model

Inferences: -

- For Test Data out of 73 number of Defaulters, 37 were actually predicted as 1's (Defaulted)
- Recall of 1 (Defaulted) = 0.57
- We can see that the model performance for Recall of 1 (Default) is quite similar 0.57

PART A.12 : Compare the performances of Logistic Regression, Random Forest, and LDA models (include ROC curve)

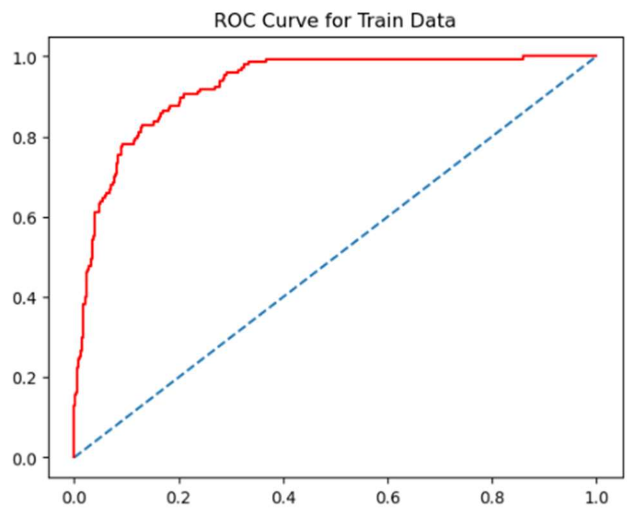
Logistic Regression, Random Forest & LDA model Performance Comparision for Default (1)						
	LR Train	**LR Test**	**RFCL Train**	**RFCL Test**	**LDA Train**	**LDA Test**
Accuracy	**0.87**	**0.85**	**0.94**	**0.92**	**0.92**	**0.9**
Recall	**0.83**	**0.75**	**0.52**	**0.41**	**0.57**	**0.51**
Precision	**0.43**	**0.4**	**0.93**	**0.71**	**0.62**	**0.56**
F1 Score	**0.57**	**0.53**	**0.66**	**0.52**	**0.6**	**0.53**
ROC AUC Score	**0.926**	**0.911**	**0.756**	**0.698**	**0.765**	**0.73**

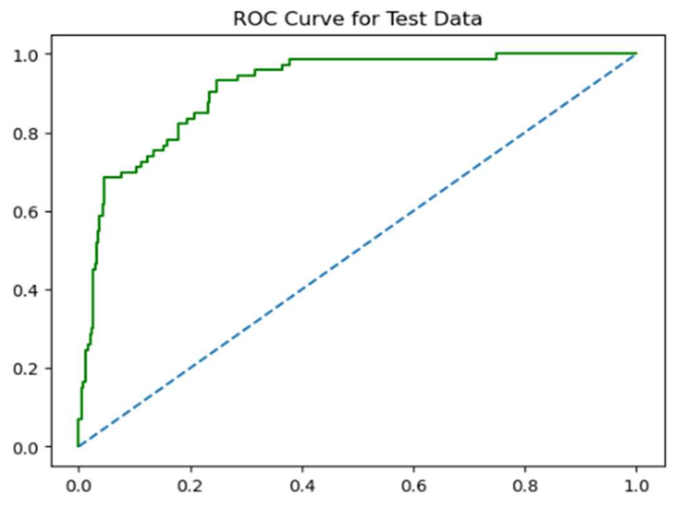
Since here it is important to predict the defaulter as defaulted. Hence Recall plays the major role in model performance.

Logistic Regression Model has the highest recall rate compared to other two models

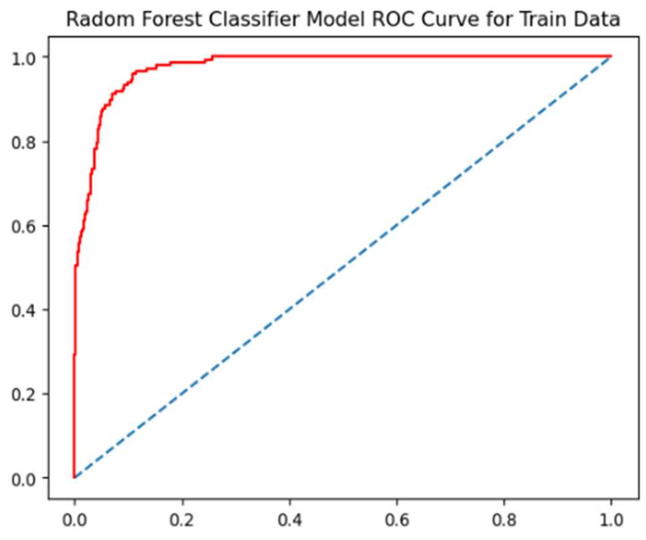
From above chart we can see that the model performance parameters of Logistic Regression (Optimum Cut off = 0.132) model for both train and test data are better than the Random Forest Classifier Model and LDA Model.

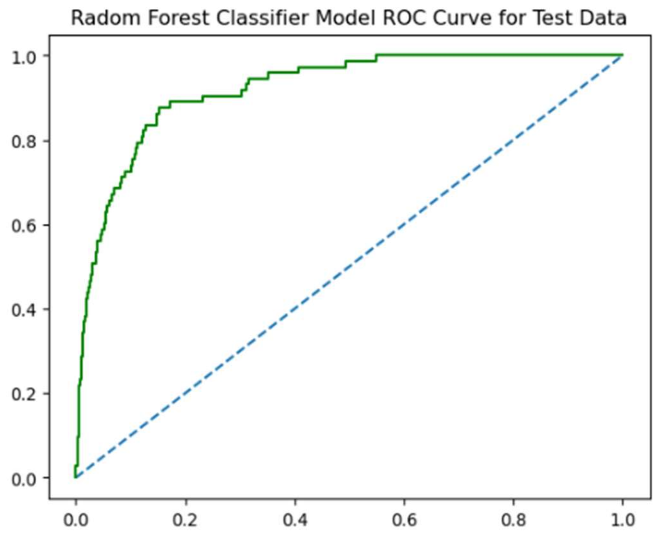
<u>ROC Curve for Logistic Regression (optimum cut off = 0.132) train and test data</u>



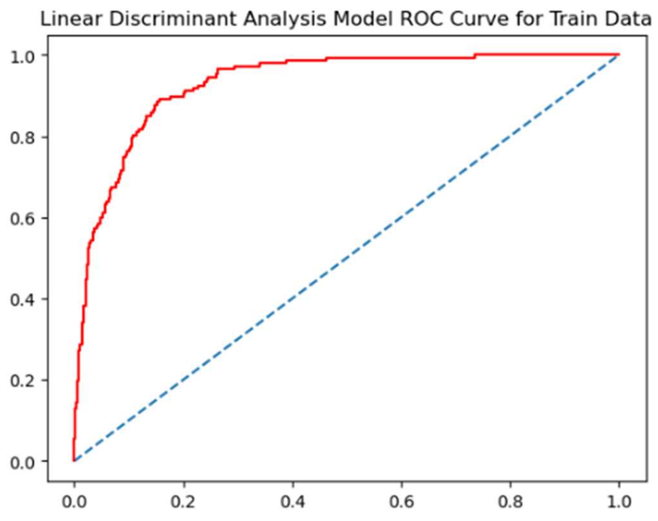


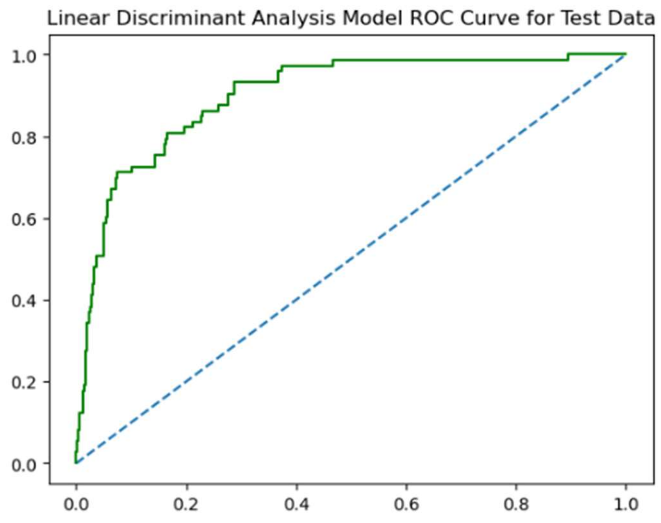
ROC Curve for Random Forest Classifier Model with GridSearch CV for train and test data





ROC Curve for LDA Model for train and test data





PART A.12 : Conclusions and Recommendations

Conclusion: -

The Logistic Regression Model outperforms random forest classifier (rfcl) and Linear Discriminant Analysis (LDA) in several metrics, achieving higher recall and ROC AUC on both training and test data.

However, LDA shows better performance in identifying potential defaulters based on higher test recall, making it more effective at correctly classifying companies at risk of default (Hence preferred for current case)

Recommendation: -

Finally, we are able to achieve a descent recall value with some overfitting. Considering the opportunities such as outliers, missing values and correlated features this is a fair model. It can be improved if we get better quality data where the features explaining the default are not missing to this extent. Of course we can try other techniques which are not sensitive towards missing values and outliers.