MS4610 - FINAL PROJECT REPORT

Conducted by
Course Coordinators, Teaching Assistants
and American Express

CREDIT CARD DEFAULT PREDICTION

by

RAGHAV JANGID (ME20B143)
ARUN PALANIAPPAN (ME20B036)
SWAPNIL PARESH MEHTA (ME20B183)

TEAM NAME: mech.ai

to

Department of Management Studies
Indian Institute of Technology Madras
March 1, 2024

Contents

List of Figures

List of Tables

1 Introduction

Team Name: **mech.ai**
Team Member 1: **Arun Palaniappan (ME20B036)**
Team Member 2: **Swapnil Paresh Mehta (ME20B183)**
Team Member 3: **Raghav Jangid (ME20B143)**
Accuracy Achieved: **60.10**%
GitHub Repository: **GitHub Repository**
Date of Report Submission: **March 1, 2024**

---

2 Problem Statement

The dataset provided has the customer application and bureau data with the default tagging i.e., if a customer has missed cumulative of `3` payments across all open trades, his default indicator is `1` else `0`. Data consists of independent variables at the time T_0 and the actual performance of the individual `(Default/ Non Default)` after `12` months i.e., at time T_{12}.

Some of the variables include: `credit-worthiness, annual-income, card-tenure, etc.`

The Problem Statement is to predict the applicant going default in next `12` months from new credit card application.

3 Exploratory Data Analysis

3.1 Missing Values

The missing values in the data were visualized using the `seaborn` library. An average of `17.3%` of data points were missing in each column. The visualization is as follows:

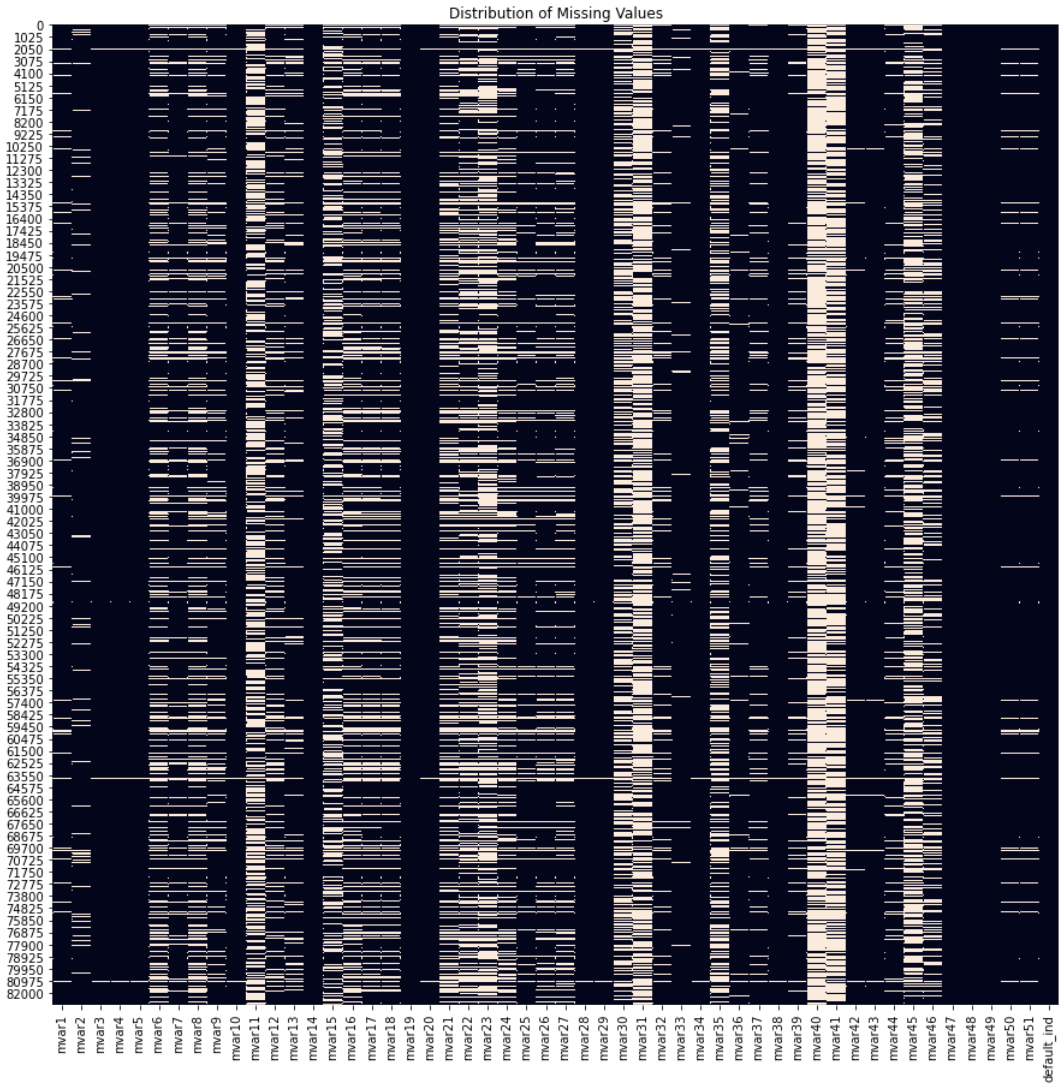

Figure 1: Visualization of missing values. The white bars indicate missing values

Some observations:

- Some columns have missing values along the same row, hence, a possibility of correlation between them.
- 12 out of 52 features have more than 25% of Data points missing.
- Few rows have almost all columns with missing values; dropping these rows will make some columns free from missing values, which is better suited for some models.
- The feature `"Utilization of line on active education loans (%)"` has the highest missing count of 78.4%.

3.2 Correlation Plot

We will examine feature interactions in this part, based on their correlations. In order to reject some of the features from analysis and model training, our objective is to identify redundant features. Multicollinearity and overfitting problems may be partially resolved as a result. We have shown correlations through a correlation heatmap below.

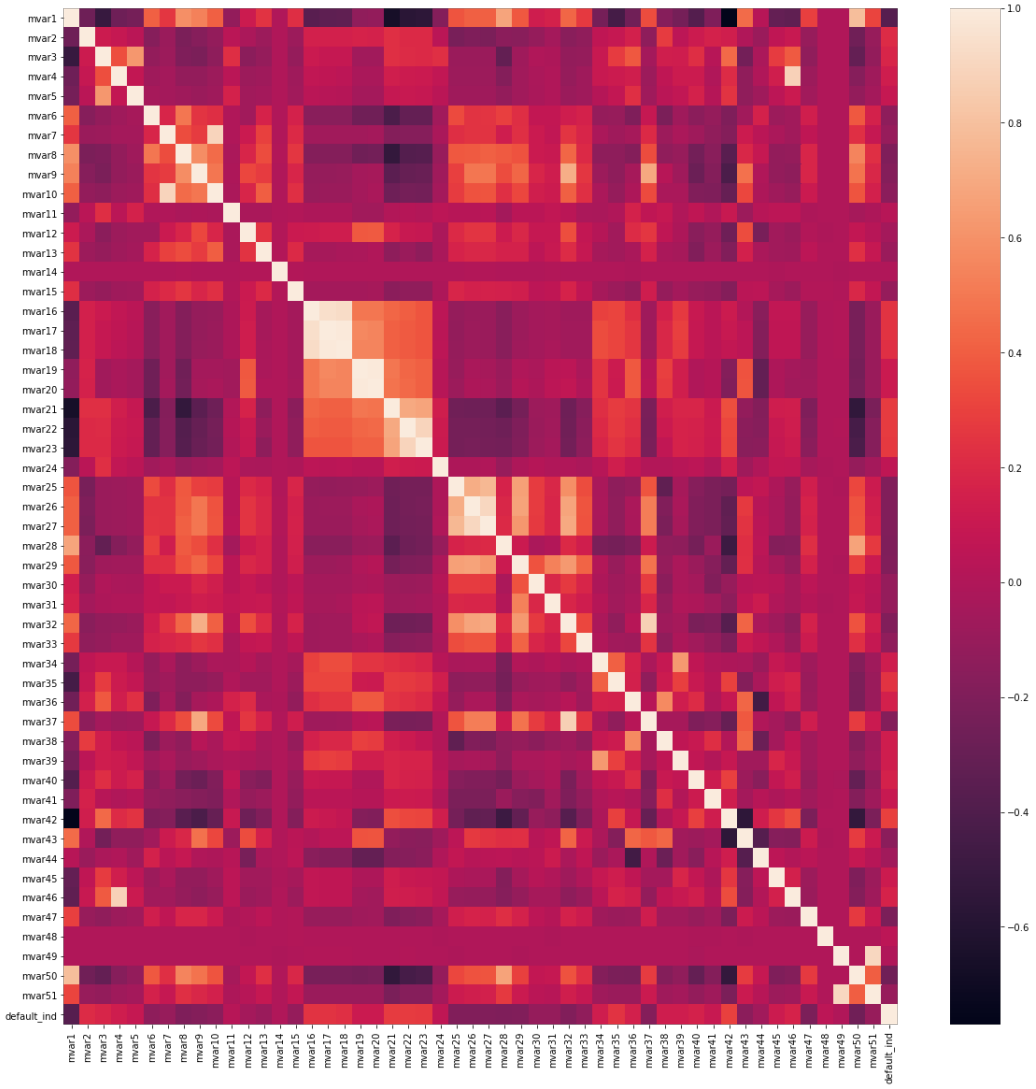

Figure 2: Visualization of correlation between features

- It can be observed that `default_ind` only has very weak correlations with other features, hence, confirming no data leakage.
- For example, from the plot it is visible that,
  - `Maximum of credit available on all active credit lines (in $)` and `Total amount of credit available on accepted credit lines (in $)` are correlated.
- The removal of correlated features helps reduce multicollinearity in certain models.

3.3 Multicollinearity and VIF

We calculated the **Variance Inflation factor(VIF)** on the features which is a measure of multicollinearity. Here are the features with very high VIF:

Feature	VIF	Feature	VIF
mvar1	713.89	mvar18	53.68
mvar50	650.62	mvar40	52.75
mvar51	307.97	mvar20	49.32
mvar49	293.48	mvar19	47.57
mvar17	61.74	mvar41	30.19

Table 1: VIF values of features

We observe that `Credit worthiness score` and `Compound feature created as a product of bucketized credit worthiness and mvar48` have very high VIF score of 713.89 & 650.62 respectively, and hence multicollinearity among the features.

3.4 Principal Component Analysis

In this section, we analyze the principal components of data. `PCA` helps us in dimensionality reduction and identifying the components which explains the variance effectively. We plotted data along 2 most significant Principal Components to check the presence of any visible decision boundary, and also the Explained variance of each component.

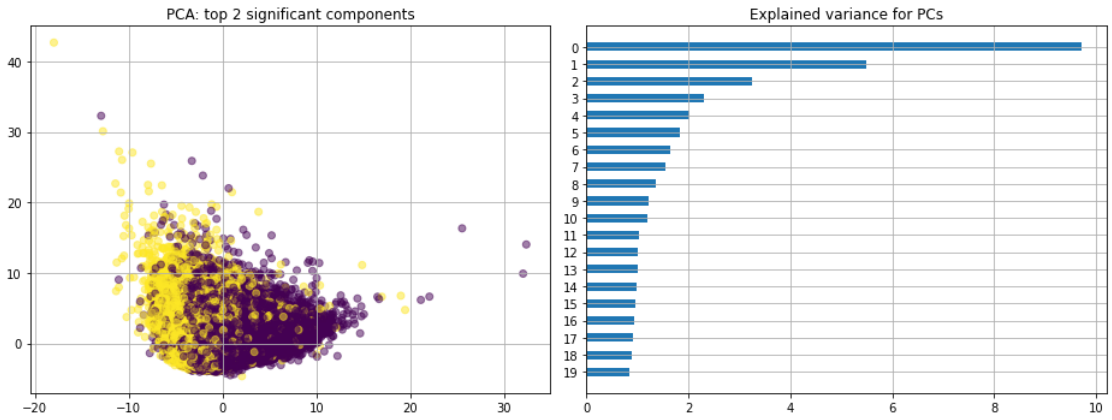

Figure 3: Principal Component Analysis

Some observations:

- We can observe that there is no clear decision boundary.
- There are lot of overlaps of data in the center part of the data cloud. Therefore even if we draw a decision boundary through the center part of the plot, there will be lot of outliers for the drawn decision boundary.

3.5 Feature Importance

In this section, we will fit a `XGBoost` model on the dataset and analyze the importance of all features. For any feature its importance for a model is a measure of its influence on the model. Here is the Bar plot of all the feature importances.

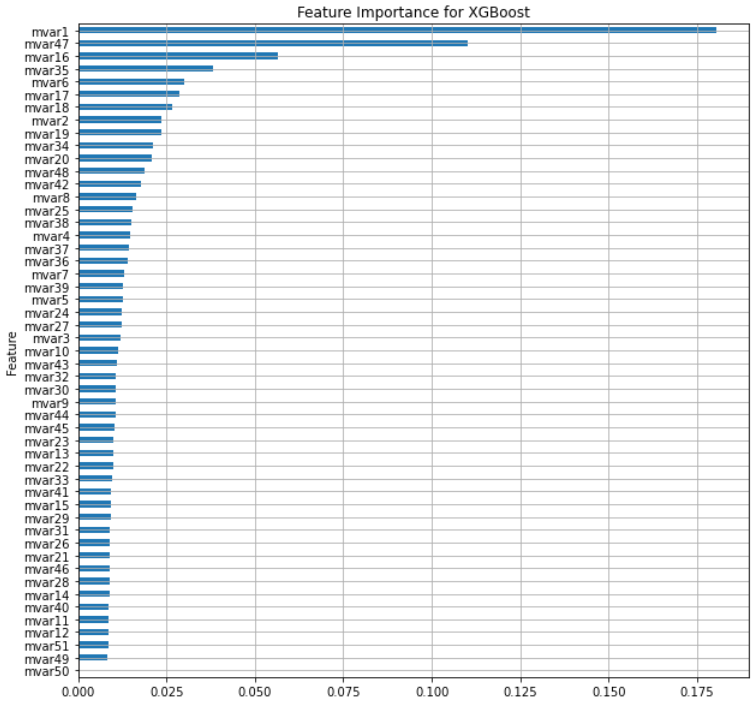

Figure 4: Feature Importance-XGBoost

Some observations:

- Features other than `mvar1` and `mvar47` don't show up as much in the feature importance curve. We can infer that these features explains majority of the variance.
- `"mvar1-Credit worthiness Score"` & `"mvar47-Type of Credit Card applied for"` are the most and second most important features respectively.

The Exploratory Data Analysis notebook can be found here.

4 Data Processing

4.1 Data Imputation

Firstly, the dataset had many missing values in various features which were written as - `"na"`, `"missing"`, `"N/A"` `etc.` which had to be replaced with the value `"NaN"` and encoding was done on the card type feature column.

- After trying to implement `Iterative Imputer` on the data, which models each feature with missing values as a function of other features in a round-robin fashion. It was seen that some of the imputed values were negative and didn't perform well.
- Therefore, we implemented `MiceForest`[1] imputation method on the dataset which utilises fast, memory efficient Multiple Imputation by Chained Equations (MICE) with `lightgbm`. It does this through iterative series of predictive models. In each iteration, each specified variable in the dataset is imputed using the other variables in the dataset. These iterations should be run until it appears that convergence has been met.

4.2 Data Generation

The dataset was quite imbalanced with 23,855 entries as defaulting in comparison to 59145 non-defaulting entries.

- Before we train the model on the dataset, it was needed to be balanced as the machine learning might be sensitive to imbalanced data, so we applied an over sampling technique called `SMOTE`[2] on the dataset which generates minority class data synthetically which in this case was defaulting.
- Prior to the `SMOTE` data generation, the dataset was split into the train set and the validation set in the ratio 4 : 1. The samples for class 1, in the train dataset alone were generated using `SMOTE`.

The distribution of samples before and after the `SMOTE` data generation is as follows:

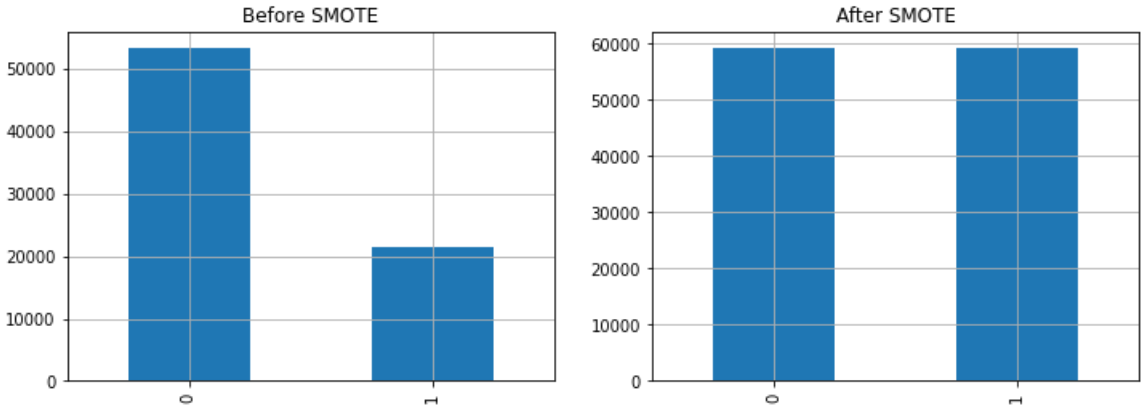

Figure 5: SMOTE class balancing

The Data Processing notebook can be found here.

5 Model Selection

The following models were implemented on the training dataset. If parameter tuning was required, it was carried out using `RandomizedSearchCV`. The validation parameters used were, Accuracy, Precision, Recall and ROC-AUC. The performance of the models is as follows:

Model	Accuracy	Precision	Recall	ROC AUC
KNN	60.74	69.43	60.74	74.16
GaussianNB	51.75	72.45	51.76	62.79
Adaboost	69.33	73.87	69.33	74.15
RandomForest	67.14	74.19	67.14	72.20
RandomForest with PCA	67.78	74.31	67.78	71.75
XGBoost	69.30	75.73	69.30	79.08
LGBM	70.39	75.96	70.39	79.57
Soft Voting on XGBoost and LGBM	69.90	75.95	69.90	79.42

Table 2: Model Performance

After careful Hyperparameter tuning we finalized these Hyperparameters for `XGBoost` and `LGBM` models, the constituents of the Voting Classifier.

Hyperparameters	XGBoost	LGBM
n_estimators	130	130
learning_rate	0.05	0.05
objective	binary:logistic	binary
subsample	0.5	0.5
colsample_bytree	0.5	0.5
scale_pos_weight	2.47	2.47

Table 3: Hyperparameters

The performance of the models with the best tuned parameters, on the validation data can be seen below. F1 score of each model is depicted in the form of a histogram below.

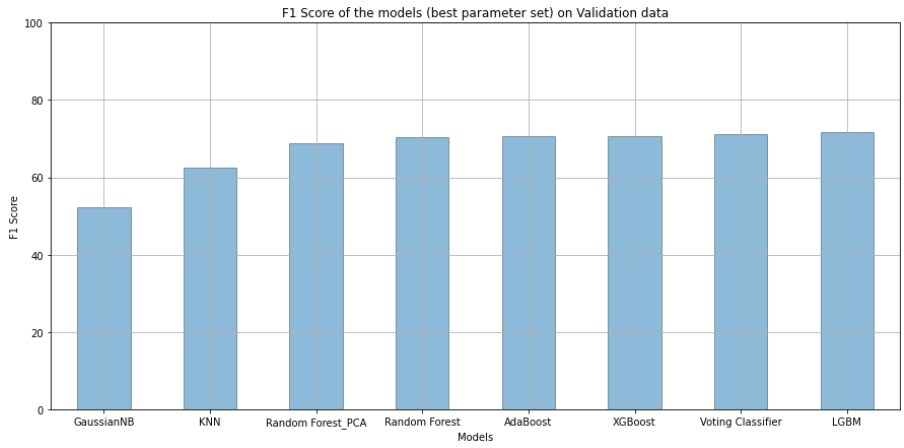

Figure 6: F1 scores of all the models

The Model Selection notebook can be found here.

6 Final Model

After careful model validation and selection we came up with this pipeline:

- `Mice Forest` Imputation gives us the most consistent and best results.
- `SMOTE` over-sampling is done to eliminate class imbalance.
- We chose a `voting classifier(soft)` comprising of best performing `XGBoost` and `LGBM` models

Model	Accuracy	Precision	Recall	ROC AUC
Voting Classifier (soft)	69.90	75.95	69.90	79.42

Table 4: Model Performance of Voting Classifier

The `ROC Curve` of the ensemble model is as follows:

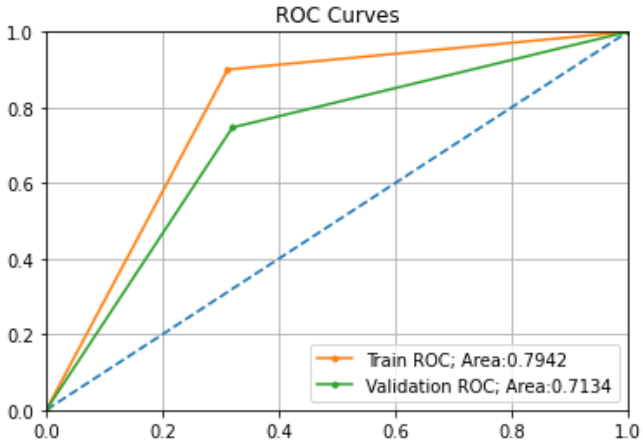

Figure 7: ROC curve

We generated the prediction file using this model and achieved an Accuracy of **60.10%** on the test data.

The Final Model notebook can be found here.

7 Insights

- We get better results for tree-based than other models and of these tree-based, the best ones are XGBoost & LGBM model. Trees takes care of the non-linear relations between the significant features.
- From the correlation plot of the features , it can be seen that some features are highly correlated. For example `sum of tenure of active credit cards` is highly correlated with the `number of credit cards with active tenure in last 2 years` and is quite intuitive.
- Another such example is that the `severity of defaults by borrower on auto loans` is highly correlated with the `number of defaults on auto loans by the borrower in the last 2 years`.
- Analysing the `default_ind` column suggested that there is a class imbalance in the training data with defaulting class in minority, therefore `SMOTE` method was used to balance the training set before applying it to the model.
- The feature importance plot depicts that some of the most informative features include `Credit Worthiness Score` as it is based on the borrower's credit history and `Type of Card` as it shows the way in which borrower is used to pay back the balance.

8 Conclusions

- Hence, we have imported, cleaned, imputed(pre-processed) & analysed the given data, fitted it to the best model.
- We noticed that among all the imputation methods tried, mice forest imputer gave the best results.
- Thus, we have implemented a `Voting Classifier` with `XGBoost` and `LGBM` models, and achieved a accuracy of **60.10%** on the test data.
- The codes written as a part of the course project can be accessed in the following GitHub Repository

———*Thank You*———

9 References

[1] Shah, Anoop D et al. *"Comparison of random forest and parametric imputation models for imputing missing data using MICE: a CALIBER study."* American journal of epidemiology vol. 179,6 (2014): 764-74. doi:10.1093/aje/kwt312

[2] Chawla, Nitesh & Bowyer, Kevin & Hall, Lawrence Kegelmeyer, W.. (2002). *SMOTE: Synthetic Minority Over-sampling Technique.* J. Artif. Intell. Res. (JAIR). 16.321-357. 10.1613/jair.953.