MS4610
Introduction to Data Analytics

Group Project – Loan Default Prediction

Project Report

Team Members:

Kandra Pavan – ME18B148

D Sai Krishna – ME18B135

Eaga Gowtham Kumar – ME18B136

K Devendranath Reddy – ME18B111

Shaik Muzaffar Rehman – ME18B170

Penta Manoj – ME18B162

Table of Contents

Introduction

About the Report:

The report contains details of our approach to prediction of Loan Default based on previously available data. This is a classic example of Supervised Learning. We were given a dataset of 80,000 unique identifiers each with several features like Income, Expense, Age, Loan Type, Occupation Type and few other metrics and each labelled as default or not-default. We were also provided another dataset for which the labels were not available but had to be predicted using the previous data.

Objective of the Project:

The objective of the project is to predict Loan Default on a new dataset with the help of an old dataset using Supervised Learning.

Dealing with Missing Data

Missing Data:

Whenever we use any of the Supervised learning methods, we have to first pre-process the data. The first step of pre-processing is to deal with the missing data. There are several methods to deal with the missing data. To choose the most appropriate method we have to first take a look at our Data and the pattern of missing data.

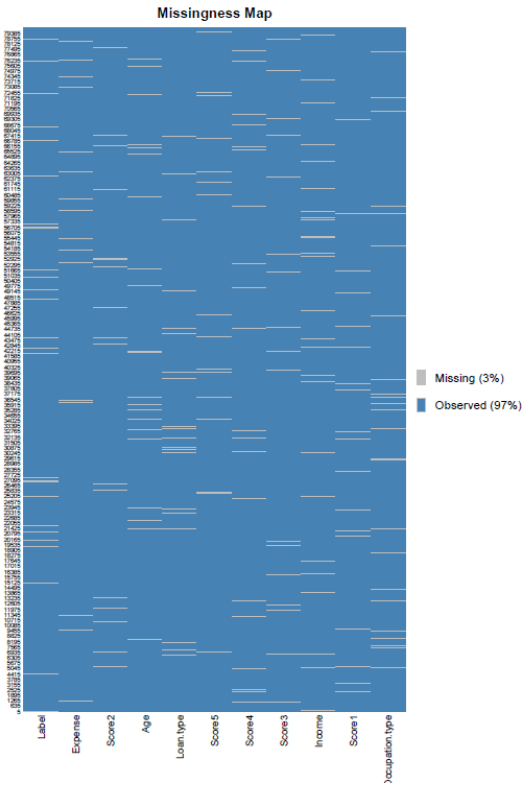


Figure: Missingness map of the give data

If missing data was small in number compared to the total available data, we can drop the rows in which data is missing. In the dataset provided to us we had about 3% of data missing. We decided not to drop these rows but use some method to impute data. Because using this data will increase the accuracy of our model.

Imputing Missing Data:

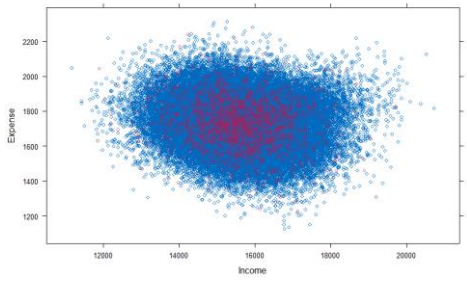
There are different methods to impute missing data. We can impute using mean, mode or median, which is a basic method widely followed by everyone. Newer, more accurate and more sophisticated methods have been developed over the years for imputing missing data. We have decided to use one such method to impute missing data in our data set.

MICE Imputations:

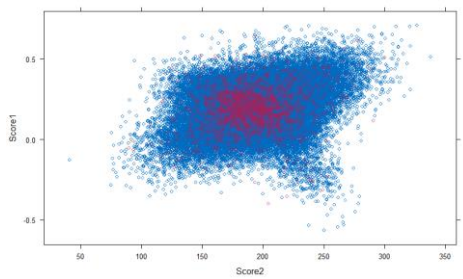
Multivariate imputations via Chained Equations (MICE) is commonly used by Data analysts to impute missing data. In this method the missing data is predicted based on the remaining features using multiple imputations. For example, if data is missing in feature 'A', the data is predicted using the remaining features 'B','C' and so on. Contrary to taking mean, median or mode of the same feature this method improves accuracy since we are predicting using other features. Different methods are used for predicting data, usually Predictive Mean Matching (PMM) is used for numeric data and other methods for categorical data. We used PMM for numeric data and CART (Classification and Regression trees) for categorical data.

Note: Imputations of missing data was done in R and the R Notebook with the code used for imputations was provided with the report.

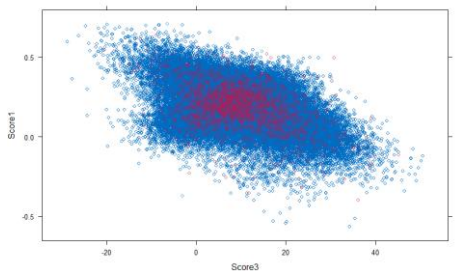
The following images show the imputed data in feature vs feature plots. (Blue points are pre-existing data; red points are imputed data)



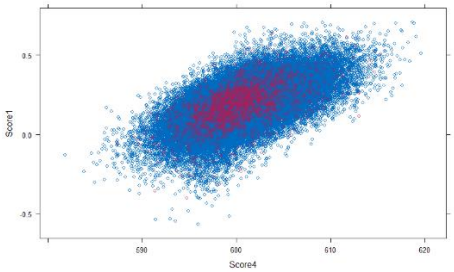
Expense vs Income



Score 1vs Score 2



Score 1 vs Score 3



Score 1 vs Score 4

It is visible that the imputed data is very similar to the pre-existing data in almost all the feature plots. This implies that the imputed data is predicted based on all the features.

Implementation of Supervised Learning

Model Selection:

The task here is a classification task, and there are many different classification models used in Supervised Learning. Each of these models work well on different kinds of Datasets. So, we tried few of the most used models on the given dataset. We tried different models like – Ridge Classifier, Lasso Classifier, SVM – Linear Kernel, SVM – Non-linear Kernel, LDA, QDA, Logistic Regression and classification using KNN. All of these models are standard models with good accuracy rates. We reached about **95.7% accuracy using SVM – Linear Kernel and LDA on training set**. Though all these methods are standard methods and widely used methods, some ensemble methods like Random Forests merged with Bagging or Boosting techniques give better accuracy (Sometimes at the cost of Over-fitting the data, but this can be reduced using regularisation methods). These ensemble methods are relatively new compared to previously mentioned methods. We decided to use ensemble methods, since the objective was to obtain a model which can predict with high accuracy. **Note –** We didn't provide the code for the standard models since our final model was XGBoost, hence the python code only contains the XGBoost model.

One of the most popular and recently developed ensemble methods is called **XGboost** (eXtreme Gradient Boosting) which usually performs the classification tasks with high accuracy. We decided to use this model, and tune the hyperparameters so as to increase training accuracy while also checking the accuracy on validation set (Bias-Variance Dichotomy).

Brief Introduction to XGBoost:

XGBoost (Extreme Gradient Boosting) is a new algorithm introduced by Chen & Guestrin in 2016. The model was developed to increase speed and performance while using regularisation to reduce overfitting. The method is similar to the method used for Gradient boosting trees, where regression trees in a sequential learning process as weak learners are used. This means that several weak learners or small trees which weakly fit the data are combined to create a better prediction model. These regression trees are similar to decision trees but they use a continuous score assigned to each leaf, and few other scoring criteria. Gradient boosting attempts to learn by minimising this score using Gradient Descent technique. XGBoost introduces a regularisation terms and also a feature called Subsampling where only few features are used for a weak learner, this further reduces overfitting. Overfitting was a drawback of Gradient Boosting and XGBoost has solved this issue.

Training the XGBoost Model for the Objective:

We used the Python package for XGBoost to train our model and predict the final output. The following sections cover several aspects of training the model –

Parameters of the Model - One of the drawbacks of XGBoost is that it has a lot of parameters to tune. We decided to focus on only few parameters so as to not complicate the procedure. The parameters which we considered were most important are-

1. ***max_depth*** – Maximum depth of a tree. A larger value implies a more complex model. Hence this parameter increases the complexity of the model and hence decreases the bias. (It may increase the variance, if the value is too high)
2. ***gamma (Ω) or the regularisation parameter*** – gamma is used to reduce overfitting of the model. Hence gamma tries to reduce variance at the cost of bias.
3. ***Learning task parameter – Objective*** – The error used for gradient descent is chosen by this parameter. We fixed this parameter to binary logistic, that is logistic regression for binary classification.

Tuning the Parameters - We tuned the parameters using a method called early stopping rounds, which is provided by the python package. This method stops the gradient descent iterations when there is no considerable decrease in the evaluation metric provided. The evaluation metric we provided was mean accuracy on validation set. Since the objective is to obtain better accuracy on the test dataset. We decided to tune only the max depth and **Ω** parameters so as to not complicate the procedure. We found out that a max depth of 25 and a regularisation parameter of 10 gave good training set and validation set accuracies, with less variance.

All the remaining parameters of the model were set to default. We used a 1000 total number of iterations and early stopping rounds as 20. This implies if the evaluation metric doesn't

decrease after 20 rounds then the iterations stop. This happens until either the metric doesn't decrease or 1000 iterations complete.
Note- We did not use any Cross-Validation methods, since XGBoost is relatively new the Cross-Validation methods are not very easy to implement. Rather we divided the provided data into Train and Validation sets. The validation set contains 10% of the whole data. The accuracy on the validation set was the metric used to tune the parameters of the model.

Summary of the Model:

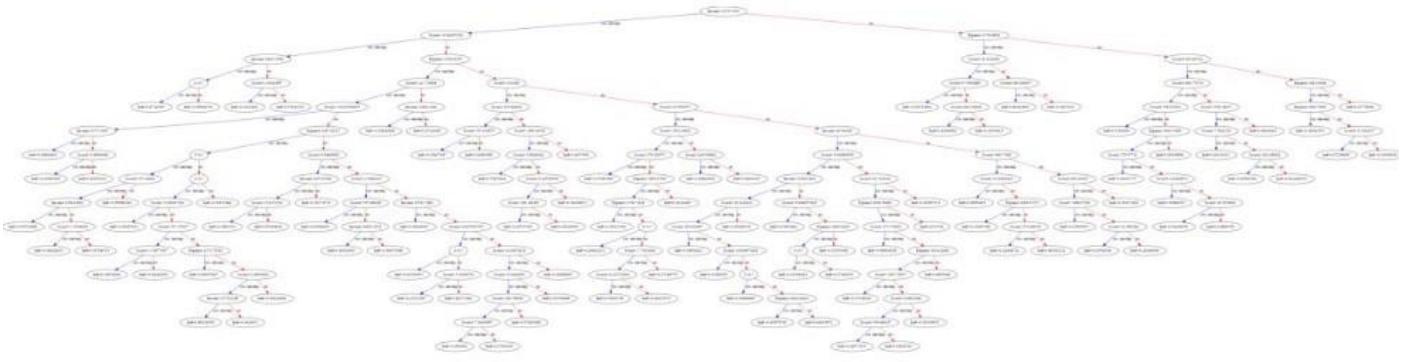
Final Parameters used for training the model –

Parameter	Value
Maximum Depth of Trees	25
Gamma (Ω)	10
Number of Iterations	1000
Early Stopping rounds	20
Objective	Logistic Regression for binary Classification

Accuracy Values –

Dataset	Mean Accuracy of the trained Model
Training set (90% of whole Data)	98.79%
Validation set (10% of whole Data)	98.43%

An Image of the ensemble Tree trained –

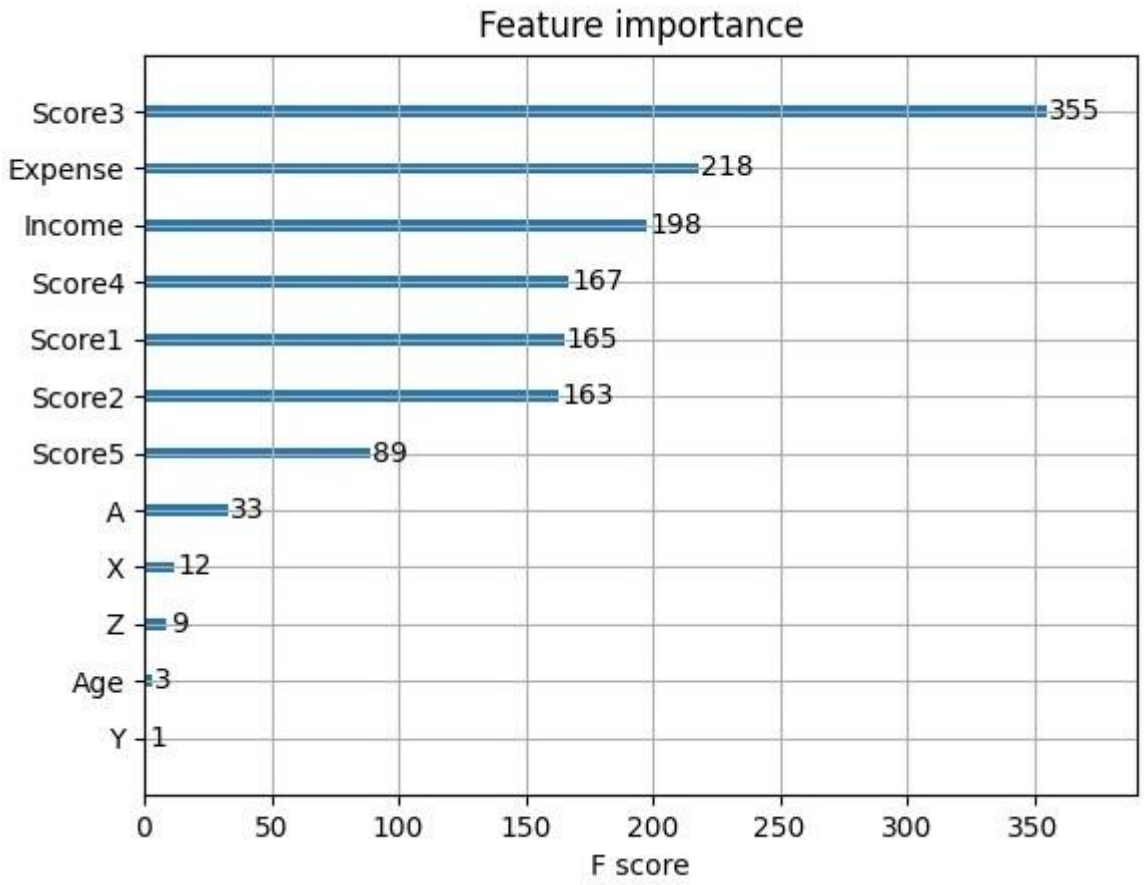


This image shows a branch (2 – tree branch) of the model. This was obtained with the functions provided by the python package.

Feature Importance:

One of the advantages of any tree-based method is the ease of interpreting the data. This is also the case with XGBoost. The python package provides a function which can easily calculate the feature importance on the trained model. The function calculates the F-score of each feature. The score basically implies the frequency of the variable being split on. As the variable is split on more often, its importance in the model increases. Hence feature importance can be evaluated on this metric.

The following image is a plot of F-scores for different features –



Inference – The metric **Score 3 has the highest importance, whereas Age has the least**. Note that even though Y has the least F-score X, Y and Z all come under same feature – 'Occupation Type'; Since the data was one-hot encoded for Categorical variables this feature has been split into 3 new features.

References

- *MICE Imputations-*
  - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3074241/#:~:text=Multivariate%20imputation%20by%20chained%20equations%20(MICE)%2C%20sometimes%20called%20%E2%80%9C,method%20of%20addressing%20missing%20data.
  - https://datascienceplus.com/imputing-missing-data-with-r-mice-package/
- *XGBoost-*
  - https://xgboost.readthedocs.io/en/latest/python/python_intro.html
  - https://towardsdatascience.com/the-ultimate-guide-to-adaboost-random-forests-and-xgboost-7f9327061c4f
- *Feature Importance and F-Score:*
  - https://stackoverflow.com/questions/34218245/how-is-the-feature-score-importance-in-the-xgboost-package-calculated