MS4610 (Introduction to Data Analytics)

Project Report

Team: Data_Poltergeists
Shashwat Patel (MM19B053)
Pragalbh Vashishtha (MM19B012)
Hrishabh Srivastava (MM19B033)
Pradeepa Selvaraj (MS22Z101)

18th December 2022

Contents

1 Problem Statement

A bank offers two types of cards: Charge cards and lending cards. An individual can apply for any one of the two types of card. In order to extend the card to individuals, banks must first underwrite the applicant. Underwriting is the process by which the lender decides whether an applicant is creditworthy and should receive a credit line. Along with the data present in application forms, banks also have access to the consumer bureau. Bureau is an agency that aggregates consumer borrowing and payment information for the purpose of assessing the creditworthiness of an individual and setting a limit on the cumulative credit that can be extended to an individual by lenders.

2 Objective

The objective is to predict if an applicant **goes default or not in the next 12 months** from a new credit card application. A set of 51 features has been given. Customer application data and bureau data with the default tagging, i.e., if a customer has missed a cumulative of 3 payments across all open trades, his default indicator is "1" else "0". The data consists of independent variables at the time T_0 and the actual performance of the individual (Default/ Non Default) after 12 months.

3 List of Data-Preprocessing Techniques used

We employed different data-preprocessing techniques for data cleaning and addressing the dataset's imbalance.
Missing values in the dataset can cause problems. Identifying and replacing missing values in the input data is good before modeling the prediction task. This is called missing data imputation.

The dataset contains twice as many non-defaults(0) than defaults(1); hence it is of prime importance that preprocessing be conducted.

3.1 KNN imputation

The KNN imputation uses the K-Nearest Neighbour algorithm to predict missing data values. The KNN algorithm is pretty simple. A distance metric like Euclidean distance metric is first found for a sample, and then it is compared to every other sample of the dataset. For prediction, the average of the **k** most similar samples is taken for imputation. Different values of k might lead to different imputed values.

3.2 SMOTE

SMOTE refers **to Synthetic Minority Oversampling Technique**. It is one of the approaches for addressing imbalanced datasets. New synthetic data for the minority class is generated from the existing data of the minority class.

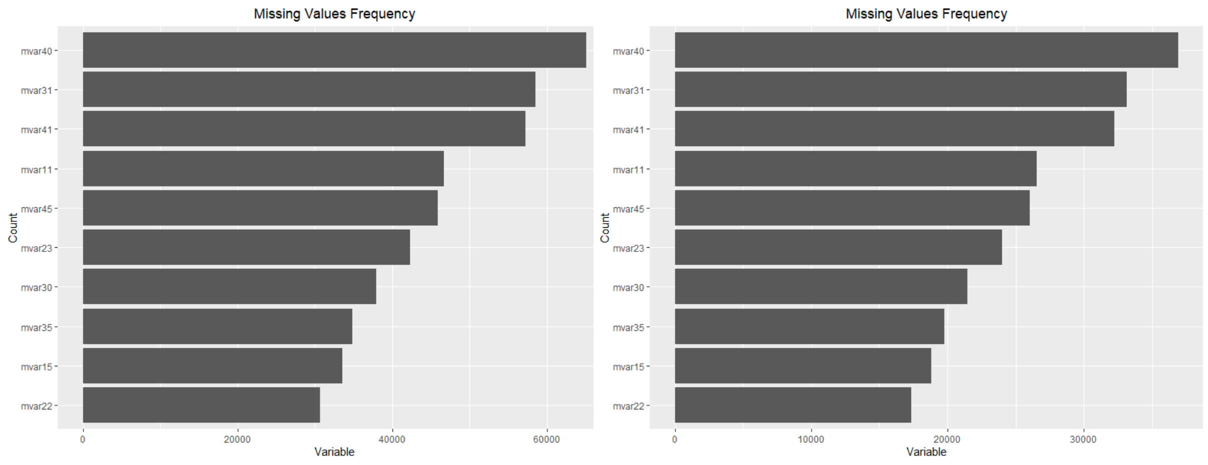
The working of SMOTE consists of first drawing a random sample from the minority class. For these observations in the sample, the k nearest neighbors are identified. One of those neighbors is taken, and then a vector is identified between the current data point and the selected neighbor. The vector is multiplied by a random number between 0 and 1 to obtain the synthetic data point, and the vector is then added to the current data point.

Another technique was used for data pre-processing but using that led to a bit decrease in test scores. SMOTE-ENN was tried. This method is a hybrid technique involving both under and over-sampling.

4 Data Set: Description & Pre-processing

Two datasets were provided, training and test data. The training data consisted of 83000 observations with 51 variables and 1 target column. The test data consisted of 47000 observations.

Out of 51 variables present in the training/test dataset, only 4 of them had no missing values others had atleast one missing value. Mvar11, Mvar30, Mvar 40, Mavr 41 and Mvar45 columns had the most missing values as seen from figure 1.



(a) Top 10 Variables having highest missing value frequency in **Training Set**

(b) Top 10 Variables having highest missing value frequency in **Test Set**

Figure 1: Missing Values Count(Made in R)

To address these missing values KNN imputation was performed, the imputation was done such that, the variables having unique values had the missing values replaced by the most frequent value. In our case, k=10 was used. (KNN imputation was performed in R, the code is attached).

The dataset is found to be imbalanced, with default (1) having a count of 23855 observations and 59145 observations with no-default(0). A problem with an imbalanced dataset is that there are only a few examples of the minority class for a model to effectively learn the decision boundary. Oversampling is a straightforward solution to tackle this problem. That's why an oversampling technique, **SMOTE** has been used for addressing the imbalance problem. The advantage of SMOTE is that it does not generate duplicates, but rather creates synthetic data points that are slightly different from the original data points. This led to our training dataset consisting of 118000 observations.

5 Model Building

After performing the data preprocessing, the model building was done. Since the given problem is a binary classification problem, ensemble methods like Random Forest, and XG-Boost(A boosting model) were used to train the model. Random forest and XGBoost have different parameters that be changed to fit our model. **Grid search cross-validation (Defined in the sub-section)** was used for tuning the parameters. Grid search with 5-fold Cross-Validation was performed for hyperparameter tuning.

In random forest, the maximum depth and number of estimators were tuned while in XGBoost, the maximum depth, number of estimators, and learning rate was tuned.

The best estimator for random forest obtained from GridSearch CV was 280 estimators and the maximum depth was 14. The best estimator obtained for XGBoost was 200 estimators, Maximum depth to be 4 and a learning rate to be 0.05.

5.1 Random Forest & XGBoost

Random Forest algorithm utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions. It is a tree-based machine learning algorithm that involves building several decision trees and then combining their output to improve the generalization ability of the model. These decision trees are grown in parallel. While doing classification in the random forest, the training data is fed to train the many decision trees. A subset of features is then chosen randomly while splitting the nodes in a decision tree. Every decision tree in the random forest consists of decision nodes, terminal nodes, and a root node. The terminal node of each tree is the final output produced by that specific decision tree. The selection of the final output follows the majority-voting system. In this system, the output chosen by the majority of the decision trees becomes the final output of the random forest.

XGBoost, In this, the decision trees are created sequentially instead of parallel. Some weights are assigned to all the independent variables, which are then fed into a decision tree that predicts some results. Then the weight of the variables predicted wrong by the tree is increased, and these variables are then fed to a second decision tree, and it goes on depending upon the number of trees grown. These individual classifiers are then ensemble to give a better model.

5.2 Grid Search CV

GridSearch CV is used to do hyperparameter tuning to determine the best possible value of parameters out of a given grid of parameters for a given model. There is no way to know in advance the best values for hyperparameters so all possible values have to be tried to know the best values. Manually, that is not possible and that's why GridSearchCV is used to automate the tuning of hyperparameters.

A given set of predefined values for hyperparameters is provided to the GridSearchCV function. This is done by defining a dictionary in which we mention a particular hyperparameter along with the values it can take. GridSearchCV tries all the combinations of the values passed in the dictionary and tries out the model for each combination using the Cross-

Validation method. The accuracy for every combination of hyperparameters is obtained and the one with best performance is chosen.

5.3 Variable Importance

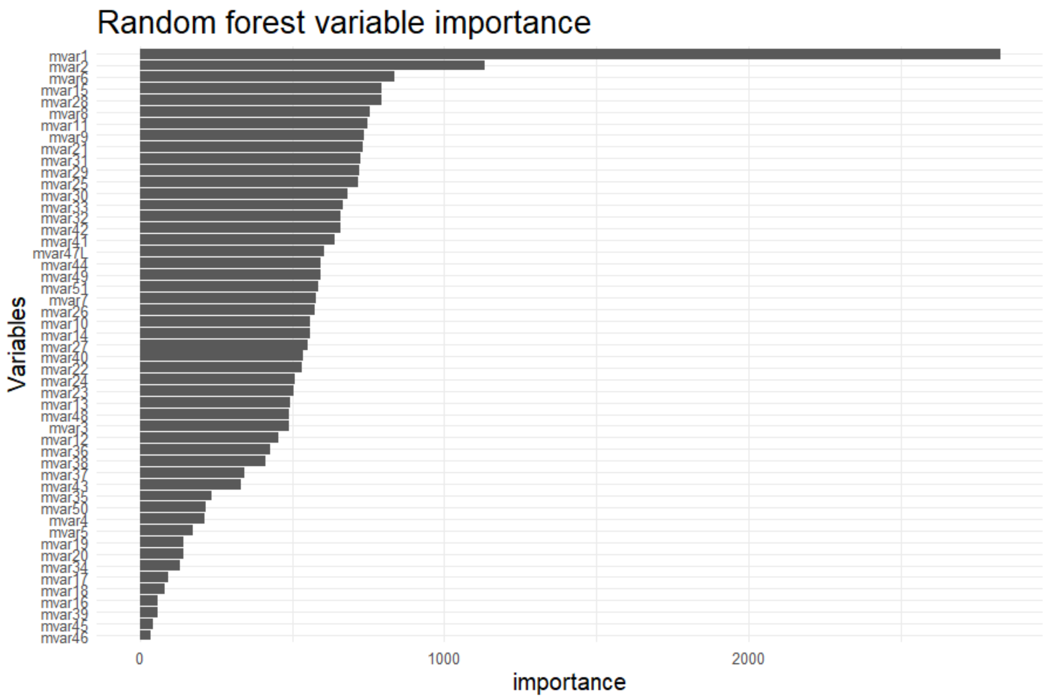


Figure 2: Variable Importance plot(Random Forest)

Figure 2 shows the variable importance plot. It is found that **mvar1, mvar2** are the most important variable.

6 Performance

Model	**Parameter**	**Test Score**
Random Forest (No SMOTE)	n_estimators=200, max_depth=12	51.41%
XGBoost(No SMOTE)	n_estimators=100, max_depth=8, learning_rate=0.3	51.60%
Random Forest (SMOTE)	**n_estimators=280, max_depth=14**	**58.54%**
XGBoost(SMOTE)	n_estimators=200, max_depth=4, learning_rate=0.05	58.22%
Random Forest(SMOTEENN)	n_estimators=260, max_depth=10	57.64%
XGBoost(SMOTEENN)	n_estimators=150,max_depth=10, learning_rate=0.05	57.62%

Different types of models were built using different data-preprocessing techniques.

When no SMOTE was used, the score was less compared to when different kinds of sampling techniques were used to address the imbalance in the dataset. Since the minority class (1) was very less compared to the majority class, the model predicted a lot of defaults (1) to not_default (0), the prediction accuracy of these kinds of models was high due to good prediction accuracy of not_default, but the Recall of these models was low.

When SMOTE was used, there was an equal number of defaults and not_defuals present in the training data, so the prediction accuracy of default (1) increased, and an improvement in the score was seen. The downside of this technique was that the model tended to overfit the training data.

Instead of using SMOTE when SMOTEENN was used, here a different thing happens since SMOTEENN is a hybrid sampling technique consisting of both oversampling of the minority class and undersampling of the majority class. The number of defaults (1) observed becomes more than the number of not_default (majority class) observed in the training data. This led to a bit decrease in prediction accuracy, but the Recall score improved. This technique also had one major problem since the number of defaults present in the dataset was a bit more than the number of not_default; this led to some not_defaults being predicted as defaults, and a minor decrease in test score was seen if compared to test score obtained when SMOTE was used.

7 Workflow

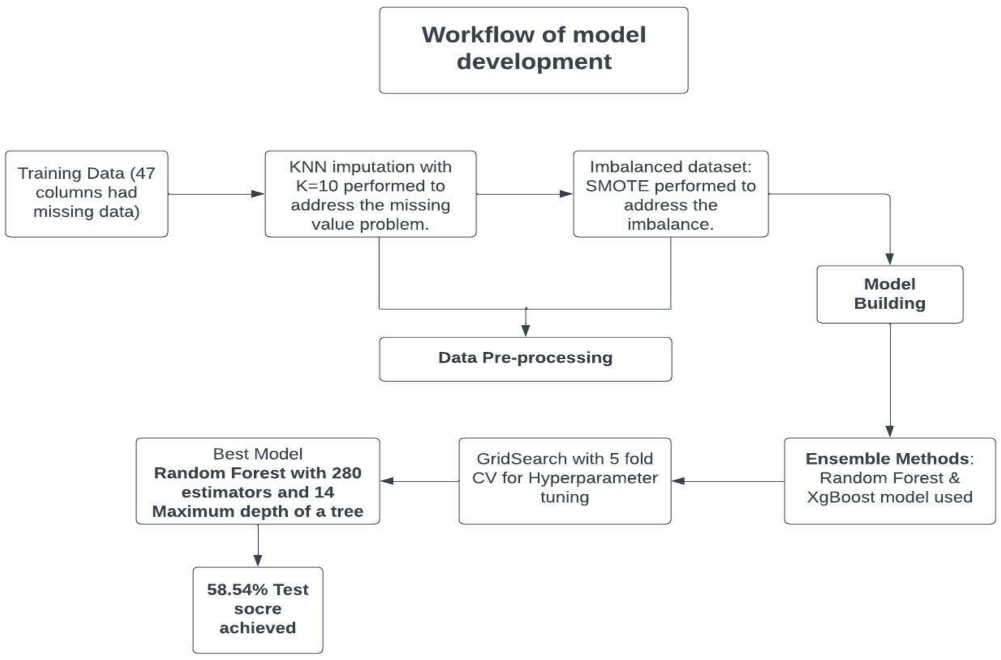

Workflow of model development
Training Data (47 columns had missing data)
KNN imputation with K=10 performed to address the missing value problem.
Imbalanced dataset: SMOTE performed to address the imbalance.
Model Building
Data Pre-processing
Ensemble Methods: Random Forest & XgBoost model used
GridSearch with 5 fold CV for Hyperparameter tuning
Best Model Random Forest with 280 estimators and 14 Maximum depth of a tree
58.54% Test socre achieved

Figure 3: Model devlopment workflow

8 Further Work

Further work can be done to improve the model's performance. There are various themes of improvement we tried which can be worked upon. The first of which is using feature selection. Feature selection can be made using the backward mode, forward mode, and Recursive Feature Selection with cross-validation(RFECV). We tried RFECV with SVMs, but the selection of features took a lot of time, and hence we were not able to generate results when appropriate. Logistic regression with Lasso regression could be tried out.

We also attempted a Neural Network based on a sequential architecture. There were consecutive dense layers and dropout layers. Still, the convergence of the NN did not occur for the choice of Adam optimizer, and SGD optimization was very slow and seemed to not improve beyond 65.5%. This may be due to the incorrect architecture of layers, and while tuning was attempted, the results weren't satisfactory. Hence a better tuning method can be employed.

9 Conclusion

The random forest model with SMOTE performed the best out of all tested models. The best test score achieved was 58.54%. Using SMOTE led to a bit of overfitting on the data but it was better than SMOTEENN or not using SMOTE at all. The random forest model, which gave us the best test score, had a number of estimators of 280 and a Maximum depth of 14. These parameters were obtained from hyperparameter tuning using GridSearch CV. KNN imputation with k=10 was performed to address the missing value problem. The Random Forest models had a better performance compared to the XGBoost models.