Department of Management Studies

IIT Madras

Predictive and Prescriptive Data Analytics

MS6032

Loan Default Prediction

Submitted by:
Harshit Shekhar Jha, MS19S003

Submitted to:
Dr. Nandan Sudarsanam

1 Problem Statement

Providing a loan is a classic case of the fine balance between risk and reward. A lenient lender might hand out loans to riskier customers thereby reducing a guaranteed payback but handing out more loans also gives him a better chance of generating higher profits. Thus, it is imperative for financial institutions to carefully classify an applicant as a defaulter or non-defaulter. This reduces the potential losses of the lender.
Machine learning can be used for such classifications. A machine learning model can be trained on the data set of past defaulters and non-defaulters and following that, it can be applied in future to predict whether the customer with a given set of credentials will default or not.

2 Dataset

The data set provided consists of 80,000 past records with each candidate labelled as a defaulter or non-defaulter. There are a total of 11 columns containing personal information like ID number, Age, Income and Expenses, Loan type taken, Occupation type and 5 additional metrics labeled as Scores. The dependent variable is whether a candidate will default or not. A label of 0 means the loan will go non-default and 1 means the loan will be default.
The test data set consists of another 20,000 records which is to be classified using an appropriate machine learning model.

3 Methodology Adopted

The first step before training any model is to format the data as per our need. If the data is incorrect, the output of the algorithm might be unreliable. With the given data set, the following changes have been made:

- **Converting Numeric Variables to Factor**

  Initially, Age and Label are represented as integers with values 0 and 1. These are then converted to factors. This ensures that the modeling functions will treat them as categorical rather than continuous variables.

- **Handling missing values**

  Across all the columns, there are a total of 17,719 rows with missing values. The decision variable has total of 3903 missing labels.

  A simple approach is to impute the missing continuous variable with mean and the missing categorical variable with mode. But this method induces a lot of bias in the data. The variance of the data distribution reduces after substituting the missing value with mean.

  Instead, MICE package (Multivariate Imputation via Chained Equations) has been used for imputing the missing values. MICE package uses information from other variables in the data to predict the missing values.

Linear regression is used to predict continuous missing values whereas logistic regression is used for categorical missing values. The missing values of Label are not imputed.

The rows for which Label is missing are dropped. This step leaves us with 76097 rows without missing values.

- **Class Imbalance**

  The problem at hand has class imbalance, i.e the distribution of labels is skewed towards the non-defaulter class. Class imbalance results in poor predictive performance for the minority class.

  In the training set, there are 93% records with Label 0 and only about 7% records as Label 1. This is handled later by bagging-boosting models and using SMOTE data.

- **Correlation**

  Only those variables must be selected that best contribute to the resulting model. Redundant variables need to be removed as they reduce model's interpretability. One method to remove these redundant variables is to check the correlation scores.

  A correlation matrix is constructed of the columns (see appendix A). It is observed that Score 5 and Expense exhibit a high correlation of 0.97. Thus, we drop Score 5 column.

- **Normalization**

  When different columns of a data set are measured on different scales, it can create bias because different variables contribute differently. Normalization aims to change the numeric columns into a common scale without distorting the range of those values.

  All numeric columns are normalized using the min-max normalization.

- **Generating Synthetic Samples**

  To generate a balanced data set, a data augmentation process for the minority class, SMOTE (Synthetic Minority Oversampling Technique), is used. This technique synthesizes new data points for the minority class.

  SMOTE package has been used to create a balanced data set. It generated 46% 0 label values and 54% 1 label values.

- **Developing models**

  Five types of models were tried. Logistic regression, KNN and CART models were trained on SMOTE data set whereas Random forest and Boosting models were trained on the imputed data set. Imputed data corresponds to the data which we got after imputation of missing values on the training data while SMOTE data set corresponds to the data which we got on applying SMOTE on the Imputed data.

- **Cross Validation**

  K-fold cross validation was used to validate the models.The value of k was chosen to be 5 as this value has been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor very high variance (James et al., 2013).

4 Models

- Brief description of all the 5 types of models are given below:

  1. **Logistic Regression**

     The logistic regression was performed with Label (Defaulter or non defaulter) as the dependent variable and Expense, Income, Loan type, Age, Score1, Score2, Score3 and Score4 as independent variables. The summary of the logistic regression model applied on the SMOTE dataset can be seen in figure 2 of appendix B.

  2. **K-nearest neighbors (KNN)**

     K-nearest neighbors method(KNN) was also performed on the SMOTE dataset. The value of k was obtained from cross validation. Using 5 fold cross validation, we checked the recall value for different values of K. Recall value was maximum when the value of k was 5. Hence, we obtained a KNN classifier with k = 5. Income, Expense, Score1, Score4 and Score2 are the predictors in decreasing order of their importance.The model summary and variable importance can be seen in figure 3 and figure 4 of appendix C respectively.

  3. **Classification and Regression Tree (CART)**

     The independent variables used in construction of the model using smote dataset were: Expense, Score4 Score2 , Age, Income and Score3. Expense is chosen as the root of the tree. The complexity parameter (cp) which is the minimum improvement in the model needed at each node is found to be 0.392 using 5-fold cross validation.

  4. **Bagging: Random Forest**

     Random forest is a decision-tree based ensemble model. It consists of large number of un-correlated decision trees with each decision tree giving a class prediction and the class which gets maximum number of votes becomes our final prediction. Number of trees to be generated was kept at 500 and number of variables tried at each split (mtry) is 6. The most appropriate value of mtry was found by cross validation and it came out to be 6 (see figure 6 in appendix E).This combination

gave us the minimum out of bag (OOB) estimate of error rate of 1.81 %. The out-of-bag (OOB) error is the average error for each training observation calculated using predictions from the trees that do not contain that training observation in their respective bootstrap sample.

5. **Boosting: Gradient Boosting Method**

   Boosting is also another ensemble technique. It combines weak learners to create a strong learner. It is not prone to over fitting. It builds the first learner based on the training dataset, then calculates the loss and uses this to build an improved learner in the next stage. To create this model, we used the stochastic gradient boosting method from gbm package in r and fit 150 trees. The stochastic gradient boosting algorithm is faster than that of the conventional gradient boosting as the regression tree require fitting smaller data sets into every iteration(Breiman, 1996). In stochastic gradient boosting, the size of subsample is small and hence, randomness comes in the algorithm which reduces the chance of overfitting. The maximum depth of each tree is 3 and the learning rate is 0.2 (see figure 8 in appendix F). Expense, Income and Score5 have the maximum influence in this model (see figure 9 in appendix F)

- In order to handle class imbalance, logistic regression, KNN and CART was applied on the SMOTE data. Since, ensemble methods like bagging and boosting are effective in handling class imbalance, they were applied to imputed data. In addition to this, we used **class_weights** with the imputed dataset so that the model is not biased towards majority class. The class weights were inversely proportional to the respective frequencies of Labels (0 - 1).
- The summary of all the metrics obtained for each model after cross validation is listed in Table 1. We used 5-fold cross validation so that the results are not dependent on a particular random choice for the pair of (train, validation) sets.

5 Metrics optimized

- Metrics for each model is listed in the table below:

  summary$_m odels_k cv.csv$

Table 1: Metrics obtained after K fold cross validation

For our model, it is very important to avoid those situations where a candidate is a potential defaulter , but our model classifies as him as not being a defaulter and thus, we should be certainly be selecting a model with high recall.

From table 1, we can see that KNN classifier is giving the highest recall value of 0.95 . Although, the accuracy of random forest classifier is slightly higher than KNN classifier, we will still go for KNN classifier as our final model since it gives much better recall value.

6 Conclusion

- A total of 5 models were tried. Class imbalance in the training data was handled by either applying SMOTE technique on imputed data in case of Logistic regression, KNN and CART models or by using ensemble techniques likes Random Forest and Gradient Boosting Method. Since, in our case, recall is a very important metric, hence K nearest neighbour(KNN) model with highest recall value was selected as our final model. On 5-fold cross validation, K nearest neighbour gave the accuracy of 0.96 and recall of 0.95. Moreover, KNN also has the highest precision and F1 score amongst all models. The KNN model prediction on test dataset gave 12,836 defaulters and 7,164 non defaulters.

7 References

James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 18). New York: springer.

Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140.

Appendix A

```
##             Expense     Income      Score1     Score2      Score3    Score4
## Expense  1.00000000 -0.1178236 -0.08656248 0.01782257  0.72344776 0.2449390
## Income  -0.11782364  1.0000000  0.37344642 0.57134080 -0.36954922 0.4366910
## Score1  -0.08656248  0.3734464  1.00000000 0.20971132 -0.39075010 0.5834144
## Score2   0.01782257  0.5713408  0.20971132 1.00000000  0.02577676 0.7860669
## Score3   0.72344776 -0.3695492 -0.39075010 0.02577676  1.00000000 0.2440589
## Score4   0.24493901  0.4366910  0.58341440 0.78606690  0.24405890 1.0000000
## Score5   0.97296417 -0.1220925 -0.08889971 0.01761717  0.74181601 0.2502164
##              Score5
## Expense  0.97296417
## Income  -0.12209246
## Score1  -0.08889971
## Score2   0.01761717
## Score3   0.74181601
## Score4   0.25021637
## Score5   1.00000000
```

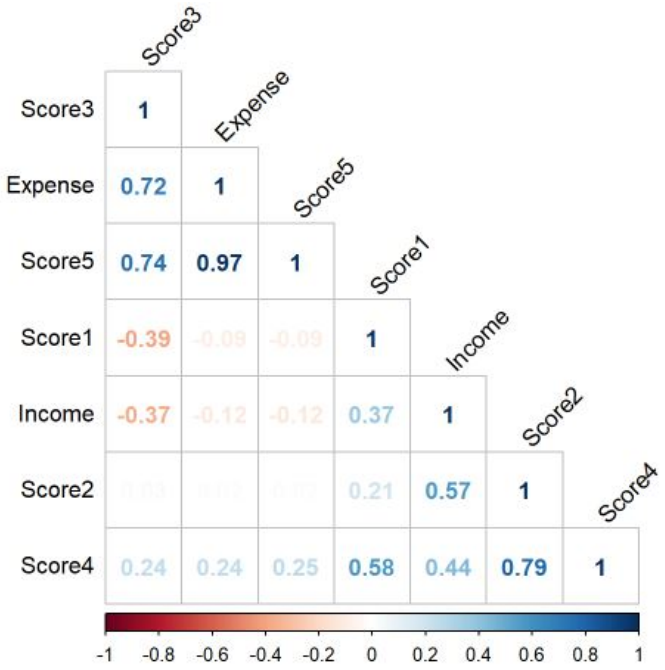

Figure 1: Correlation Matrix

Appendix B

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.1090  -0.5704   0.0955   0.5014   3.7604
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -6.199e+01  1.590e+00 -38.986  < 2e-16 ***
## Expense           1.094e+00  4.489e-01   2.437 0.014811 *
## Income            1.268e+01  2.340e-01  54.193  < 2e-16 ***
## Score1            8.569e+01  2.414e+00  35.496  < 2e-16 ***
## Score2            7.992e+01  1.852e+00  43.147  < 2e-16 ***
## Score3            6.380e+01  1.995e+00  31.971  < 2e-16 ***
## Score4           -1.323e+02  3.166e+00 -41.798  < 2e-16 ***
## Loan.typeB       -9.606e-02  2.749e-02  -3.494 0.000476 ***
## Occupation.typeY  4.588e-03  4.222e-02   0.109 0.913467
## Occupation.typeZ -1.792e-02  5.469e-02  -0.328 0.743103
## Age1              1.145e-01  2.801e-02   4.086  4.4e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 90316  on 65428  degrees of freedom
## Residual deviance: 48952  on 65418  degrees of freedom
## AIC: 48974
```

Figure 2: Logistic regression model summary

Appendix C

```
## k-Nearest Neighbors
##
## 65429 samples
##     6 predictor
##     2 classes: '0', '1'
##
## Pre-processing: centered (6), scaled (6)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 52344, 52344, 52343, 52342, 52343
## Resampling results across tuning parameters:
##
##   k  Recall  Precision  F1      Accuracy.Accuracy  Kappa.Kappa
##   5  0.9612  0.9632     0.9620  0.9591313          0.9177909
##   7  0.9590  0.9632     0.9610  0.9581379          0.9158090
##   9  0.9566  0.9628     0.9592  0.9566554          0.9128412
##
## Recall was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

Figure 3: 5 fold cross validation to get the most appropriate value of K

```
## ROC curve variable importance
##
##          Importance
## Income       100.00
## Expense       91.10
## Score1        35.17
## Score4        34.71
## Score2        20.03
## Score3         0.00
```

Figure 4: Variable importance table for KNN model

Appendix D

```
## CART
##
## 65429 samples
##     9 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 52342, 52343, 52344, 52343, 52344
## Resampling results across tuning parameters:
##
##   cp          Recall  Precision  F1      Accuracy.Accuracy  Kappa.Kappa
##   0.06642824  0.7008  0.8584     0.7716  0.7766287          0.5574337
##   0.08576727  0.7838  0.7556     0.7592  0.7338333          0.4607002
##   0.39250944  0.9166  0.6288     0.7392  0.6451890          0.2503171
##
## Recall was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.3925094.
```

Figure 5: Finding the appropriate value of cp using cross validation

Appendix E

```
## Random Forest
##
## 76097 samples
##     9 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 60878, 60878, 60877, 60878, 60877
## Resampling results across tuning parameters:
##
##   mtry  Recall  Precision  F1      Accuracy.Accuracy  Kappa.Kappa
##    2    0.7304  0.9376     0.8212  0.9789216          0.8098501
##    6    0.7926  0.9168     0.8500  0.9815236          0.8404538
##   10    0.7906  0.9090     0.8456  0.9809322          0.8357072
##
## Recall was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 6.
```

Figure 6: Choosing the correct value of mtry using 5 fold cross validation

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 1.81%
## Confusion matrix:
##       0    1 class.error
## 0 70703  361 0.005079928
## 1  1020 4013 0.202662428
```

Figure 7: Summary of random forest model

Appendix F

```
## Stochastic Gradient Boosting
##
## 76097 samples
##     9 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 60878, 60878, 60877, 60877, 60878
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Recall  Precision  F1      Accuracy.Accuracy
##   1                   50      0.8296  0.2594     0.3950  0.8311233
##   1                  100      0.8460  0.3238     0.4682  0.8727544
##   1                  150      0.8552  0.3506     0.4970  0.8854488
##   2                   50      0.8500  0.4554     0.5930  0.9227959
##   2                  100      0.8676  0.4832     0.6204  0.9298527
##   2                  150      0.8780  0.5050     0.6412  0.9348725
##   3                   50      0.8624  0.5048     0.6366  0.9348463
##   3                  100      0.8806  0.5188     0.6528  0.9379476
##   3                  150      0.8854  0.5366     0.6680  0.9417848
##   Kappa.Kappa
##   0.3267824
##   0.4118771
##   0.4451050
##   0.5545859
##   0.5854756
##   0.6078536
##   0.6035766
##   0.6210902
##   0.6382217
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Recall was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

Figure 8: Gradient boosting machine model summary

	var <fctr>	rel.inf <dbl>
Expense	Expense	28.35531386
Income	Income	21.26319490
Score3	Score3	19.79947842
Score4	Score4	11.66280749
Score1	Score1	10.57985725
Score2	Score2	4.82070777
Loan.typeB	Loan.typeB	2.69602563
Occupation.typeZ	Occupation.typeZ	0.74496257
Occupation.typeY	Occupation.typeY	0.07765211
Age1	Age1	0.00000000

1-10 of 10 rows

Figure 9: Relative influence of predictors in gradient boosting model