INDEX

Introduction

The following project offers exposure to a real-world scenario using a data set from the insurance industry. The data contains customer responses from a mail marketing campaign, and it will be used improve future launches of the same campaign. The goal is to improve the marketing campaign by selecting better target audiences that are likely to take the offer.

The dataset contains 69 predictors and 20k observations. Among the predictor variables, credit information, number of accounts, account types, credit limits, and utilization are given. Additionally, some demographic information such as the age, and location of the individuals is also given. As the goal is to estimate the probabilities of future customers taking the offer given by the insurance industry, at this stage it is safe to assume that for our classification models, the variable PURCHASE will be used as our response variable.

For training and validation purposes, the data was already split 50/50, 10k records were split for training and stored in the data frame "train", and 10k records were split for testing and stored in the data frame "valid".

Methodology

For the completion of this Case Study, the software R Studio (Version 4.1.2) was used. Additionally, several packages and libraries were used for the statistical analysis. The library "Information" was used to complete Information Value (IV) analysis. Libraries "ClustOfVar", "reshape2", and "plyr" were used for variable clustering and for eliminating highly correlated variables. Library "dplyr" was used to clean out the variable "PURCHASE". For modeling, the libraries "randomForest", and "e1071" were used; randomForest for modeling Random Forests, and e1071 for modeling Support Vector Machines of Polynomial and Radial kernels. Lastly, the library "caret" aided on the creation of confusion matrices, and the library pROC allowed for the creation of Receiver Operating Characteristic curve (ROC) and Area Under Curve (AUC) value.

Data Cleaning and Preparation

As the dataset is rather large, data preparation is needed. In the data preparation stage, it is sought to clean the variables that contain no relevant information, eliminate highly correlated variables, and eventually select the most informative variables.

To start our data exploration and preparation stage, we start with Weight Of Evidence (WOE) and Information Value (IV) Analysis. As stated by its creator, Kim Larsen, IV "is a precursory step designed to ensure the approaches deployed during the final modeling phases are set up for success", and "permits to clean out the variables that contain no information relevant to predicting the action of interest".

To start with IV analysis, the library Information is needed. In this step, we will remove all variables that have an information value (IV) less than 0.05. Consequently, we create new train and validation datasets were we subset only the variables that have an IV value higher than 0.05. Said variables are stored in the data frames train_new and valid_new. As expected, by performing the first step, our data set has been reduced greatly as the training and validation data frames now contain 34 predictor variables as a result of only keeping those predictors that had an IV higher than 0.05 and contain potentially relevant information.

Next, using ClustOfVar, reshape2, and plyr packages, highly correlated variables will be eliminated. Additionally, before building our models, the variable, PURCHASE is recoded into NewPurchase using the command ifelse, for every observation that is 1 in PURCHASE, will be kept as 1 in NEWPurchase, if else… it will be recoded into -1. After recoding, the variable PURCHASE is eliminated from the data set and NEWPurchase is our new predictor.

Random Forest Model

To create the random forests in R, the library randomForest was used. For this case study, 10,001 trees were used to build the random forest. Additionally, it was advisable to try mtry values ranging from 1 to 13, then evaluate the out of bag (OOB) error for each model and select the random forest with the lowest OOB to make predictions.

There are different ways to perform the tasks required by this project. For instance, random forests with differing mtry values can be run individually and then their OOB error compared. However, after attempting said method it not only resulted on an extensive run time for each forest, but also on a mismanagement of RAM. Said issue limited the ability to finish running future random forests and later tasks required for a conclusive case study.

Luckily, a different method was successfully coded that allowed to compare OOB error in an effective manner. First, an array to store the OOB errors was created and named oob.err. Then, a random forest with 10,001 trees was built, with NEWPurchase as the predictor variable, train_new1.2 as the data set, and a setseed value of 1551 that was arbitrarily chosen. Each mtry value was run through the random forest, and its value stored in oob.err. After the random forest was created, OOB errors for mtry 1-13 can be visualized by calling oob.err. In order to get the best fit, the which.min function is called and stored in the value fitbest. A list of the OOB errors is calculated by the random forest and the lowest value highlighted is attached in *Table 1*.

mtry value	Out-of-Bag (OOB) error
13	0.1818
12	0.1807
11	0.1811
10	0.1807
9	0.1818
8	0.1804
7	0.1809
6	0.1813
5	0.1806
4	0.1820
3	0.1814
2	0.1846
1	0.2017

***Table 1**: OOB errors for Random Forest of mtry values 1 through 13.*

After comparing the OOB errors, the random forest model with the lowest OOB value used an mtry of 8. This random forest was used to make predictions and visualize a variable importance plot.

Before making predictions, a random forest with mtry of 8, NEWPurchase as the predictor variable, and train_new1.2 as the data set was retrained making sure to respect the same setseed value of 1551. The random forest selected was named as bestRF.

After the model – bestRF – was trained, a variable importance plot was built using the command varImpPlot. In the variable importance plot, a breakdown of the variables that influenced the outcome of the random forest can be visualized. The importance ranking MeanDecreaseAccuracy and Gini measures can be visualized.

For this random forest, the ranking in both measures align similarly. For example, looking at the first five variables in MeanDecreaseAccuracy and Gini measures, 4 out 5 variables coincide. Although the ranking of said variables differ by one or two places, both measures relatively agree on the ranking of important variables. A further breakdown of the variable importance plot can be visualized in *Figure 1*.

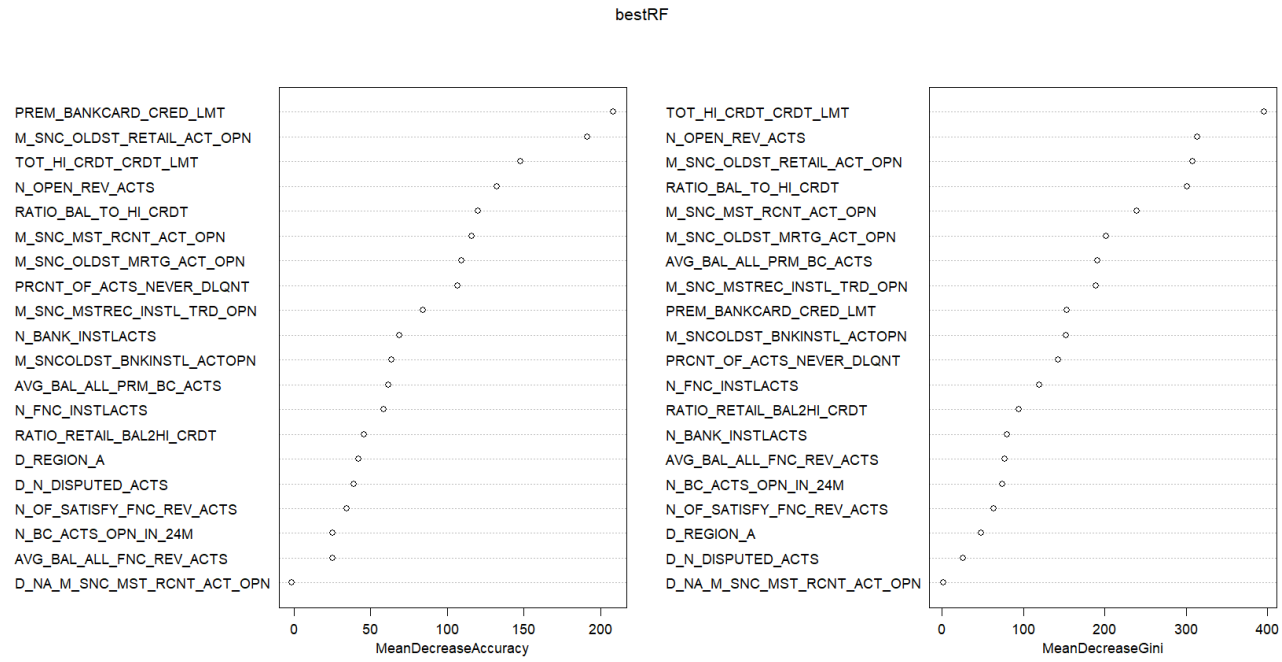


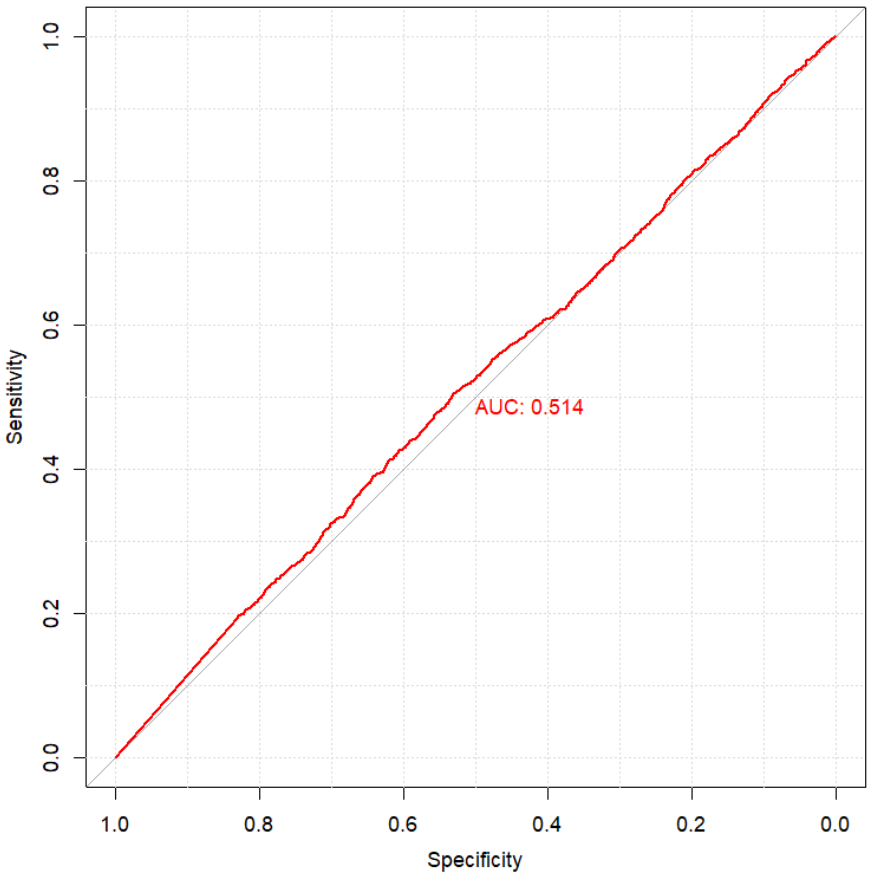
Figure 1: *Variable Importance Plot ni MeanDecreaseAccuracy and MeanDecreaseGini measures.*

To make the predictions, the predict command and the validation data set (valid_new) were used. With the caret library, a confusion matrix can be visualized and used to assess the model. Upon calling the confusion matrix, it is learned that 7,738 observations were correctly categorized as -1, and 510 predictions were correctly categorized as 1. Additionally, 285 predictions were classified as 1 instead of -1, and 1,467 predictions were categorized as -1 instead of 1. Thus, the proportion of customers that were predicted to take the offer out of those who actually took the offer (Sensitivity) of the model was 96.45%, and the proportion of customers that were predicted not to take the offer out of those who actually did not take the offer (Specificity) was 25.80%. The overall accuracy of the model was calculated to be 82.48%.

```
> pRF <-confusionMatrix(RF.pred, valid_new1$NEWPurchase)
> pRF
Confusion Matrix and Statistics

          Reference
Prediction   -1    1
        -1 7738 1467
        1   285  510

               Accuracy : 0.8248
                 95% CI : (0.8172, 0.8322)
    No Information Rate : 0.8023
    P-Value [Acc > NIR] : 5.346e-09

                  Kappa : 0.2871

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9645
            Specificity : 0.2580
         Pos Pred Value : 0.8406
         Neg Pred Value : 0.6415
             Prevalence : 0.8023
         Detection Rate : 0.7738
   Detection Prevalence : 0.9205
      Balanced Accuracy : 0.6112

       'Positive' Class : -1
```

Figure 2 *Confusion Matrix and Statistics for Random Forest Model*

To assess model performance, an ROC curve with an AUC value was plotted. In R, the ROC curve was built using the pROC library. In order to properly code for the ROC plot, we need to consider the probabilities in our predictions. The ROC curve with AUC value for the Random Forest Model is given by *Figure 3*.



A good model has an AUC near to 1, which means that the model is accurate at separating between classes. When a model has an AUC of 0.5, it is not pleasing news as it means that the model does not have an accurate capacity of separating between classes. Unfortunately, for the Random Forest model, the AUC value resulted in 0.514, meaning that the model is not able to distinguish between classes.

Figure 3 *ROC curve and AUC value for Random Forest Model.*

An additional way of classification is through support vector machines.

For this case study, two different approaches were used to construct the support vector. The first through a polynomial kernel, and the second through a Gaussian Radial Kernel. To construct the support vectors, the library e1071 was used.

Support Vector Machines (Polynomial Kernel)

To run the support vector machine, the svm function was used, the cost was set to be 0.01, the degree 3, the kernel polynomial, and the probability was set to true. The support vector machine fit was stored in the variable svm.poly. Within the data set, a number of 4,171 support vectors were detected. After training the support

vector machine model, predictions were made using the predict function and the validation data set (valid_new1).

```
> pSVM.poly <-confusionMatrix(svm.pred, valid_new1$NEWPurchase)
> pSVM.poly
Confusion Matrix and Statistics

          Reference
Prediction   -1    1
        -1 7979 1899
        1    44   78

               Accuracy : 0.8057
                 95% CI : (0.7978, 0.8134)
    No Information Rate : 0.8023
    P-Value [Acc > NIR] : 0.2003

                  Kappa : 0.0525

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.99452
            Specificity : 0.03945
         Pos Pred Value : 0.80775
         Neg Pred Value : 0.63934
             Prevalence : 0.80230
         Detection Rate : 0.79790
   Detection Prevalence : 0.98780
      Balanced Accuracy : 0.51698

       'Positive' Class : -1
```

Figure 4 *Confusion Matrix and Statistics for SVM Polynomial Kernel Model.*

From the confusion matrix of the polynomial kernel, the proportion of customers that were predicted to take the offer out of those who actually took the offer (Sensitivity) of the model was 99.45%, and the proportion of customers that were predicted not to take the offer out of those who actually did not take the offer (Specificity) was 3.94%. The overall accuracy of the model was calculated to be 80.57%. The confusion matrix for the support vector machine polynomial kernel can be seen in *Figure 4.*

Additionally, to assess model performance an ROC curve with AUC value was also plotted for smv.poly, and given in *Figure 5.*

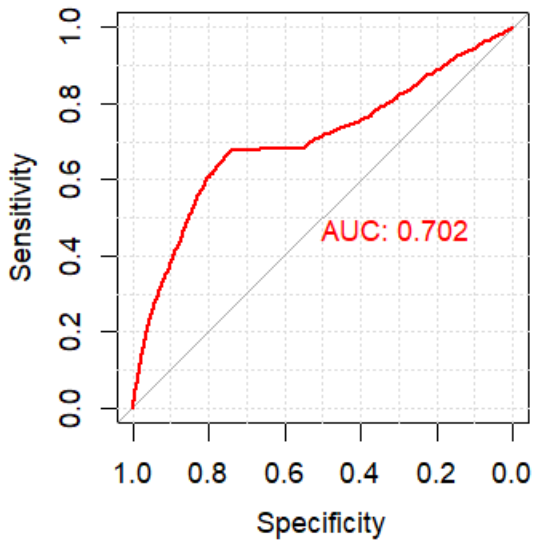


Figure 5 *ROC curve and AUC value for SVM Polynomial Kernel Model.*

The AUC for the Support Vector Machine using a Polynomial Model was calculated to be 0.702. As a good model has an AUC value closer to 1, the results for this SVM are positive as they indicate that the model performed well on classifying the data correctly. For this SVM, there is a 70.2% chance that the model will be able to correctly select customers for the insurance company that will take the offer.

Support Vector Machines (Gaussian Kernel)

A different type of support vector machine was also fitted also with the svm function provided by the e1071 library. This time, to run the support vector machine, the kernel was set to radial with gamma value of 0.000001, the probability was also set to true, and the cost was kept as 0.01. The support vector machine fit was stored in the variable svm.radial. Within the data set, a number of 4,046 support vectors were detected. Consequently, predictions were made using the predict function and the validation data set (valid_new1).

```
> pSVM.gauss <-confusionMatrix(svmg.pred, valid_new1$NEWPurchase)
> pSVM.gauss
Confusion Matrix and Statistics

          Reference
Prediction   -1    1
        -1 7335 1252
        1   688  725

               Accuracy : 0.806
                 95% CI : (0.7981, 0.8137)
    No Information Rate : 0.8023
    P-Value [Acc > NIR] : 0.1798

                  Kappa : 0.3148

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9142
            Specificity : 0.3667
         Pos Pred Value : 0.8542
         Neg Pred Value : 0.5131
             Prevalence : 0.8023
         Detection Rate : 0.7335
   Detection Prevalence : 0.8587
      Balanced Accuracy : 0.6405

       'Positive' Class : -1
```

Figure 6 *Confusion Matrix and Statistics for SVM Gaussian Kernel Model.*

From the confusion matrix of the Gaussian kernel, the proportion of customers that were predicted to take the offer out of those who actually took the offer (Sensitivity) of the model was 91.42%, and the proportion of customers that were predicted not to take the offer out of those who actually did not take the offer (Specificity) was 36.67%. The overall accuracy of the model was calculated to be 80.6%. The confusion matrix for the support vector machine polynomial kernel can be seen in *Figure 6.*

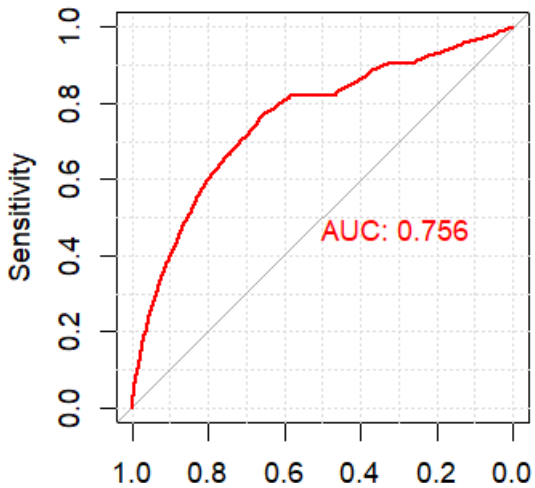


Figure 7 *ROC curve and AUC value for SVM Gaussian Radial Model.*

The ROC curve and AUC value are given in *Figure 7.* The AUC for the Support Vector Machine using a Gaussian Radial Model was calculated to be 0.756. The results for this SVM are positive as they indicate that the model performed even better on classifying the data accurately. For this SVM, there is a 75.6% chance that the model will be able to correctly select customers for the insurance company that will take the offer.

Discussion

In Table 2, a quick visualization for the values for Accuracy, Specificity, Sensitivity, and AUC is provided for all models.

Model	Accuracy	Sensitivity	Specificity	AUC
Best Random Forest	82.48%	96.45%	25.80%	**0.514**
SVM Polynomial	80.57%	99.45%	39.45%	**0.702**
SVM Gaussian Radial	80.6%	91.42%	36.67%	**0.756**

Table 2 *Statistics for all three models.*

Judging models by their Accuracy, the Random Forest performed better (82.48%). All models did well at predicting true positives, as all of them have a Sensitivity ranging from 91% to 99%. However, all three models could use some improvements towards Specificity as they all struggled on classifying true negatives accurately.

Within the context of AUC values, the Support Vector Machines performed better as they have the AUC value closest to 1. Out of all the models, the SVM using a Gaussian Radial Kernel performed the best having an AUC of 0.756.

As stated by British statistician George E.P. Box, “all models are wrong, but some are useful”, all models within this case study could use some improvements and further analysis investigating the *why* of some of the values. However, after comparing Accuracy, Sensitivity and Specificity for all three models, and relaying on the AUC value for a final recommendation, the SVM Gaussian Radial Model results in the most useful as it is likely that it will do a better job on selecting a target audience for the marketing campaign of the insurance company.

References

Hoare, Jake. “How Is Variable Importance Calculated for a Random Forest?” *Displayr*, 9 June 2021, www.displayr.com/how-is-variable-importance-calculated-for-a-random-forest.

Josh, Starmer. “ROC and AUC in R.” *YouTube*, uploaded by StatQuest with Josh Starmer, 18 Dec. 2018, www.youtube.com/watch?v=qcvAqAH60Yw.

Klarsen1. “GitHub - Klarsen1/Information: Information Package.” *GitHub*, 10 Apr. 2016, github.com/klarsen1/Information.

Narkhede, Sarang. “Understanding AUC - ROC Curve - Towards Data Science.” *Medium*, 15 June 2021, towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5.

Rodrigo, Hansapani. “Lecture Ch 8 Tree-Based Methods.” *OneDrive*, uploaded by Dr. Hansapani Rodrigo, 1 Nov. 2021, UTRGV Class Recordings.

---. “Lecture Ch 9 Support Vector Machines.” *OneDrive*, uploaded by Dr. Hansapani Rodrigo, 8 Nov. 2021, UTRGV Class Recordings.

---. “Lecture Decision Trees, Random Forests, Bagging, and Boosting.” *OneDrive*, uploaded by Dr. Hansapani Rodrigo, 3 Nov. 2021. UTRGV Class Recordings.