ASML: Classification
Topic: Prediction of Bank Personal Loan Upsell
Lecturer: Dr Sarah Heaps and Dr James Liley
Github Repository: https://github.com/Amberisfree/datascience.github.io/blob/main/Bank_loan_Analysis/report.R

Part1: Executive Summary

The classification goal is to predict the likelihood of a liability customer buying personal loans. Our bank wants a new marketing campaign, so that we need information about the correlation between the variables given in the dataset. The objective of this analysis is to utilize finance data to gain insights and make informed decisions in classifying our clients into our banks' potential candidate for selling personal loans in the near future. Finance data encompasses a wide range of clients' information including Age, Income, Education, Income, Morgage and Securities Account. The significance of finance data analysis lies in its potential to guide investment strategies, boost our personal loan sales, and drive financial decision-making processes.
By analyzing finance data, we aim to:

- Predict our future clients who might need personal loans from our bank based on our current customer database

Overall, our loan model can help ensure more people who qualify get access to fair loan options. This can help people with important purchases like homes or cars. The model can quickly assess a borrower's situation and recommend the best loan options for them. This saves time for both the borrower and the lender, getting the right loan to the right person faster.

Part2: Technical Summary

I. Problem Description

1. Explanation of the objective of the Analysis:

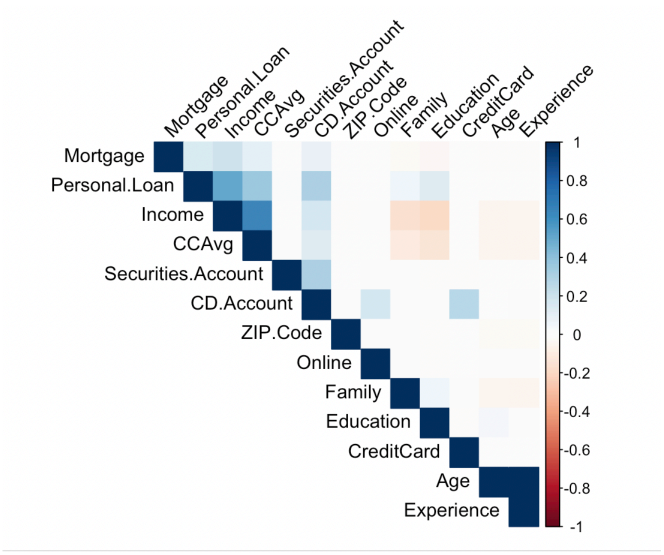
In order to sell more loans, our goal for this analysis is to identify the key features of potential client who might need to loan from our banks. We need to especially turn our attention to our potential clients and not to reject them for the benefits of business and also our banks's reputation. Our target for the classification task is therefore "Personal.Loan."

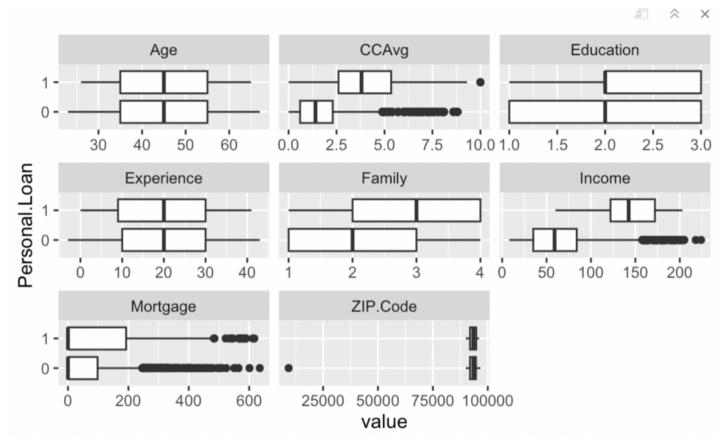
2. Data summary:

Our current dataset for banks contains 5000 observations with 13 features, including Age, Experience, Income, ZIP.Code, Family, CCAvg, Education, Mortgage, Personal.Loan, Securities.Acc…). Firstly, looking into our correlation matrix, we've noticed that our target variable "Personal.Loan" is highly correlated with Income, and moderately correlated with CD.account, CCvg, Education, and Family. Secondly, by focusing on the box plot, we can see that people who loan tend to have more family members, earned more income, more CCAVg, and have received more education. In contrast, Age and Experience seem to be less important in terms of loans or not.

Lastly, through our first exploratory, our target variable "Personal.Loan," has suffered significant from class imbalance, with 480 positive and 4520 negative. This indicates that we are required to perform data augmentation to help our model generalize better both in positive and negative classes.

3. Simple Visualization:





II. Model Fitting

1. Resampling Methods: Cross-validation and Nested Resampling

Given our feature space is not overwhelming large, we do not consider bootstrap method for selecting our variables. Instead, we proceed on sampling through cross-validation for estimating the error and also choosing our best model and its hyper parameter. During the data preprocessing phase ,we've chosen k-fold cross validation sampling to achieve accurate error estimation before completing model assessment. In addition, we've chosen 5 instead of 2 or n (our sample size) for cross validation, because our model tends to have high bias and low variance if k=2, whereas if k=n is chosen, our model will have very high variance since all models are so highly correlated. Therefore, to achieve a decent bias-and -variance tradeoff, 5-fold cross validation is implemented.

In addition, for a more sophisticated model assessment, we also adopted nested resampling. In particular, *nested* cross validation which is the correct way to do parameter selection without biasing test error. During our building of random forest, we've also implemented bootstrap as a bagging method to reduce variance.

Lastly, for the final model assessment, we perform train-test split to assess our final model's predictive performance.

2. Algorithm Design and Loss Function

Given our data's complexity and we need a very intuitive interpretation for our model, **non-linear and non-parametric** ML approaches, such as tree-based and support vector machine, are primarily considered. Both CARTs and SVM are Non-linear classifier, but they differ hugely in terms of training their algorithm and minimizing their loss function.

A. Loss function for Support Vector Machines

The hinge loss function acts as a guide for the SVM algorithm, directing it towards finding the hyperplane that best separates the data points while minimizing misclassifications, especially those with large margins.

B. Loss function for Tree-Based Algorithm,

Tree-based methods partition the feature space into a set of rectangles, and then fit a simple model (like a constant) in each one. Their goal is to predict the class of a new observation based on the most common class in the training data for the region the observation falls into. Classification trees use Gini index, classification error rate, and entropy to determine the best split for separating classes. $p\hat{}_{mk}$ represents the proportion of training observations in the mth region that are from the kth class.

- **Gini index** is referred to as a measure of node purity—a small value indicates that a node contains predominantly observations from a single class.

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

- **Entropy** it measures how mixed the classes are in a region. Lower entropy signifies a purer node.

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$

III.Model Improvement: Model Selection

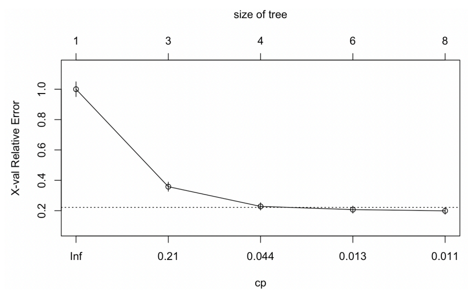
Initially, we've compared support vector machine and Tree-based algorithm with a baseline model, and we notice that accuracy in support vector machine and that of Cart algorithm are significantly higher than baseline model. In contrast, with respect to metrics in false negative rate and false positive rate, the CART algorithm outperforms SVM. Therefore, tree-based algorithm is preferred and to further explore our model selection, we further focus on Boosting / Bagging/ Random Forest

1. Summary of Tree-Based Approaches Tried

The decision trees normally suffer from high variance. This means that if we split the training data into two parts at random, and fit a decision tree to both halves, the results that we get could be quite different. We have experimented different tree-based approaches before selecting the final models, including boost, random forest, and pruning a single decision tree.

A. Single Decision Tree with Tree-Pruning

The single decision tree is fit by pruning its tree nodes to only 8 nodes; however, only the false negative rate has been improved by 0.004. Other metrics, such as accuracy, auc, error rate, and false positive rate, do not improve. We suspect that one major problem with trees is their high variance. Often a small change in the data can result in a very different series of



splits, making interpretation somewhat precarious. The major reason for this instability is the hierarchical nature of the process: the effect of an error in the top split is propagated down to all of the splits below.

B. Ensemble methods: Bagging and Random Forest

Random forests build on the strengths of bagging by introducing randomness in feature selection, resulting in a more robust and accurate ensemble model. This has resulted in lower error rate, lower significantly false positive rate. Also, the accuracy and auc have been improved. However, false negative rate is increased slightly. We have address a single tree's high variance and its unstable split of nodes by considering ensemble method. Random forests, as one of bagging methods, averages many trees to reduce this variance. Also, we build a number of decision trees on bootstrapped training samples. But when building these decision trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors.

C. Boosting: Training on errors

Boosting works in a similar way, except that the trees are grown sequentially: each tree is grown using information from previously grown trees. Boosting does not involve bootstrap sampling; instead each tree is fit on a modified version of the original data set. However, during our experiment, the over performance of Boosting do not improve.

2. Hyper Parameter Tuning

Random forests are generally less susceptible to overfitting compared to some other algorithms due to their ensemble nature. Given random forest outperforms the other two architectures, we will primarily focus on tuning the random forest. Number of trees, mtry, and alpha are to perform hyper parameter tuning during grid search. Gini Index is used as a criterion to split the node of trees. The argument mtry = 5 indicates that all 5 predictors should be considered for each split of the tree—in other words, that bagging should be done.

Finally, our final random forest model is chosen based on reducing classfication error is alpha=0.7, number of trees=300, and mtry=5.

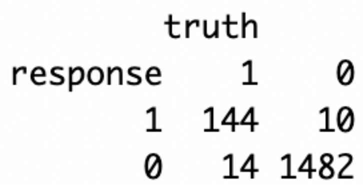
3. Insight into improvements through different architectures

To combat a single decision tree's high variance issue and tackle its inherent instability with respect to splitting its nodes, we consider ensemble methods. Averaging or focusing on errors helps to reduce the overall variance of the model, leading to more stable and consistent predictions. By combining diverse trees, ensemble methods can better capture the underlying patterns in the data and potentially generalize better to unseen data.

IV. Performance Report

1. Accuracy and Confusion matrix

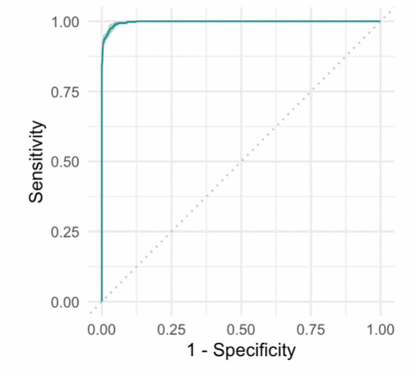
Our final random forest model of for loan task has a roughly 0.9854, which only takes into account the TPs and TNs, in the confusion matrix, the model seems to be morel likely to misclassify



```
        truth
response    1    0
       1  144   10
       0   14 1482
```

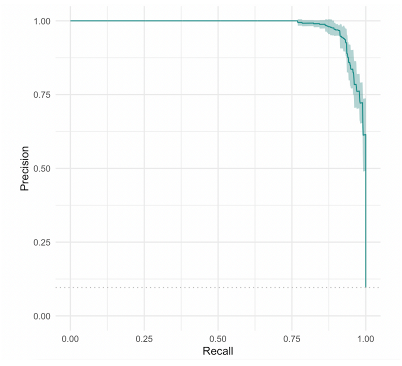
negative class.

2. ROC Curve and AUC score



ROC Curves towards the top-left corner. This indicates high TPR (correctly identifying positives) with low FPR (mistaking negatives as positives), which is around 0.088. Also , In our loan task, AUC score for the final random forest is 0.9970. The ROC curve's AUC (Area Under the Curve) summarizes the model's overall performance. It represents the probability that the model will rank a randomly chosen positive instance higher than a randomly chosen negative instance. However, we've also noticed that FNR is higher than FPR, with 0.088 versus 0.0067, which has also underlined our class imbalance problem.

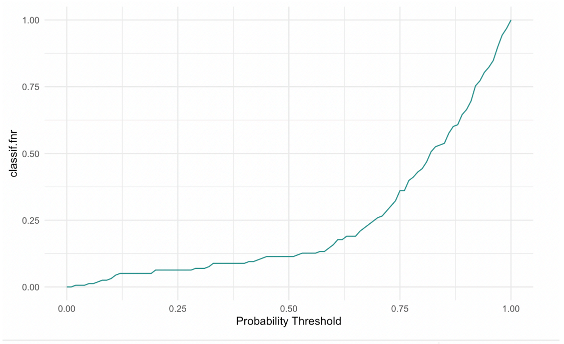
3. Precision And Recall Curve

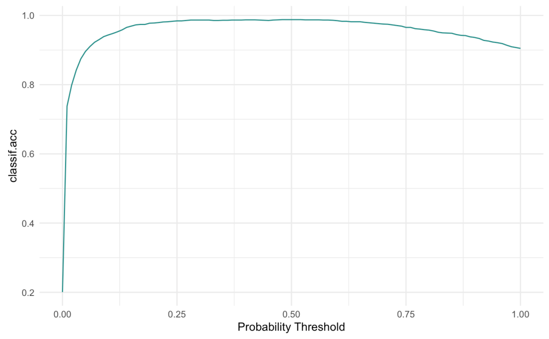


PR curves can be especially useful in our imbalanced populations where the positive class is rare, and where a classifier with high TPR may still not be very informative and have low PPV. Our final model can be deemed as an **ideal Curve,** that stays close to the top-left corner, signifying both high precision (few false positives) and high recall (identifies most positive cases). Yet, we can also notice a **Steeply Dropping Curve.** This Indicates a rapid decline in precision as recall increases. The model might be overly aggressive in identifying negatives (loan rejections), leading to a significant increase in false negatives (missed good loans).

4. Decision Threshold and False Negative Rate VS. Accuracy

In our loan task, I'd likely be **more worried about false negatives** (rejected applicants who should be approved) than false positives (approved applicants who shouldn't be approved).False negatives prevent creditworthy borrowers from accessing loans they need. It can also damage the bank's reputation for being too restrictive. Denying loans to qualified individuals can have a negative social impact, especially for those seeking loans for essential needs like starting a business or buying a home. By comparing false negative rate and accuracy in terms of probability threshold, the threshold is suggested to move from 0.5 to 0.25 to further make improvement on the false negative rate without reducing the accuracy.


classif.fnr
Probability Threshold


classif.acc
Probability Threshold