Machine Learning Models on Credit Card Acceptances

Andrew Ma

University of Illinois at Urbana-Champaign, Spring 2020

Abstract – Credit Card Applications are determined by algorithms that vary based off massive amounts of data banks and credit providers hold. While many of these algorithms are similar, there are differences between them that dictate whether a person is accepted or rejected for a credit card application. This paper observes the Credit Card dataset from the book, Applied Econometrics with R (AER), in order to observe: the significance of numerous inputs and which machine learning technique produces the best model in predicting the acceptance of a credit card application. I used several machine learning algorithms including, but not limited to, shrinkage methods, random forest, neural networks, and support vector machines. The findings of this paper can be used to assist in developing new or improving existing algorithms that automate credit card applications.

1 Introduction

According to data released by the American Bankers Association, there were 374 million open credit card accounts in the United States as of November 2019.[1] As more and more people open credit card accounts, banks need to create better algorithms in order to automate whether a credit card application gets rejected or accepted. My paper aims to (a) find which variables in my data weigh most in determining the status of a credit card application and (b) determine which machine learning model most accurately determines the status of a credit card application. Millions of new credit card accounts are opened every year so even a small percentage improvement will impact millions of people. [2]

Similar studies have been conducted by others and there are other datasets regarding credit card applications available online. A study done in 2017 by Khaneja, et al. determined that Income, Years of Employment, Credit Score, and history of default as the largest determinants in the outcome of a credit card application.[3] The dataset used in that paper was provided by the machine learning archives of the University of California, Irvine. It consisted of variables different from the ones analyzed in this paper; in addition to the variables mentioned above, it also had debt, marital status, education level, employed, drivers license, and citizen, etc. Additionally, the models they implemented were Logistic regression, K nearest neighbors, Linear Discriminant Analysis, Quadratic Discriminant Analysis, Neural Networks, and Classication and Regression Tree (CART) [3] Another study done by Kuhn, R. uses the same dataset from UCI and used logistic regression and CART as well. They found Years of employment, credit score, and income as the most significant variables in their logistic regression - which are similar results to that found by Khaneja. [4]

In my paper, I am working with a different dataset which contains different variables than the other studies. I find the variables that are consistently significant in different models. I also implemented more machine learning models to determine which gives the most accurate prediction. The machine learning algorithms used in this paper for classification were Logistic Regression, K Nearest Neighbors,Ridge Regression, Lasso Regression, Decision Trees, Bagging, Random Fores, Boosting, XGBoost, Neural Networks, and Support Vector Machines.

2 Dataset Description

The data set used is from the book Applied Econometrics with R. [5]

2.1 Variables

The y variable was "card"-whether or not an applicant was accepted for a Credit Card. This held

the value of "yes" or "no". The x variables were:
reports: number of derogatory reports.
age: the age in years plus twelfths of a year.
income: yearly income (in USD 10,000).
share: ratio of monthly credit card expenditure to yearly income.
expenditure: average monthly credit card expenditure to yearly income.
owner: does the individual own their home? (categorical:"Yes","No")
selfemp: is the individual self-employed? (categorcial:"Yes","No")
dependents: number of dependents.
months: months living at current address.
majorcards: number of active credit accounts.
active: number of active credit accounts.

2.2 Transformations

The categorical variables were owner, selfemp, and card. The continuous variables were income, share, and expenditure. The discrete variables were reports, age, dependents, months, major cards, and active. Because categorical variables were in "yes" or "no", I needed to transform them to 0 and 1, for no and yes, respectively. This is due to the fact that R will not run variables that are characters in the models.

3 Machine Learning Models

3.1 Splitting the Dataset

I used 70% of my data as the training set and the remaining 30% into a testing set. To determine the performance of each model, I can calculate the correct number of predicted values of the test subset using the machine learning model and divide them by the total number of y's in the test subset.

Training Error Rate =

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y_i})$$

where

$$\underbrace{I(y_i \neq \hat{y_i})}_{\text{Indicator Function}} = \begin{cases} 1, & \text{if } y_i \neq \hat{y_i} \\ 0, & \text{if } y_i = \hat{y_i} \end{cases}$$

Test Error Rate = $Ave(I(y_0 \neq \hat{y_0}))$

$\hat{y_0}$ is the predicted class label that was derived from our model and y_0 is the correct value from the original data.

Cross-Validation

I use cross validation to determine the accuracy of the machine learning models. Steps:
1. Randomly divide the observations into a training set and a validation set. (70/30 in this research)
2. Train the model using training subset
3. Use the trained model to predict the outcome or responses in our validation set
4. Calculate the error rate by comparing the predicted values with the actual values in the testing set. A quantitative method is preferred.

3.2 Logistic Regression

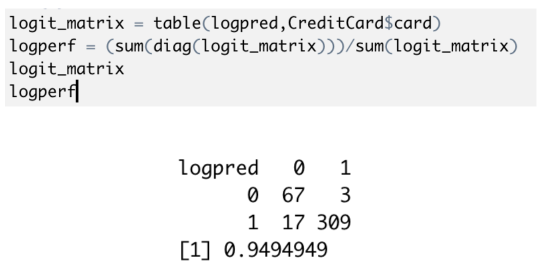
When the response variable falls into one of two categories, such as yes/1 or no/0, we need to use logistic regression models in order to determine the probability that the Y classifies in a certain category.

$$\text{p}(X) = \frac{e^{\beta_0+\beta_1 X_1+\ldots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\ldots+\beta_p X_p}} \quad (1)$$

$$log(\frac{p(X)}{1-p(X)}) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p \quad (2)$$

In order to compute or β_p values, the regression coefficients are estimated with maximum likelihood. Maximum likelihood works by seeking estimates for β's such that our predicted probability $\hat{p}(x_i$ yields a number close to 1 for credit card acceptances, and 0 for those that were denied.

Results of Logistic Regression



```
logit_matrix = table(logpred,CreditCard$card)
logperf = (sum(diag(logit_matrix)))/sum(logit_matrix)
logit_matrix
logperf
```

```
logpred   0   1
      0  67   3
      1  17 309
[1] 0.9494949
```

Figure 1. Logistic Confusion Matrix

Our logistic regression was able to correctly predict 94.94% of our test data correctly. Compared to other machine learning models, logistic regression is simple but it can be observed that it does perform well in our data.

Figure 2. provides us the values of the β's from Eq. 1 and Eq. 2. The five variables with the highest

```
               Estimate
(Intercept)  4.415505e+00
reports     -3.052424e+00
age          6.792541e-02
income      -1.531357e+00
share       -9.709443e+03
expenditure  5.600846e+01
owner        9.924877e-01
selfemp      1.206606e+00
dependents  -8.340682e-01
months      -6.172181e-03
majorcards   3.746914e-01
active       1.283828e-01
```

Figure 2. Logistic Regression Coefficients

weights are share, months, age, owner, and expenditure. We interpret the coefficients as the **change in the log odds of the outcome for a one unit increase in the variable** [Eq.2]. Thus, our logistic model weighs **share, months, age, owner, and expenditure** as the five largest determinants of classifying a credit card application. Furthermore, a bank that wants to implement a logistic regression model should focus on those five variables.

3.3 K Nearest Neighbors

K Nearest Neighbors is a supervised learning algorithm that classifies points based on existing known points. K Nearest Neighbors assumes that similar data points will be close together on a graph.

We are looking for

$$Pr(Y = j \mid X = x_0) = \frac{1}{K} \sum_{i \in N_0} I(y_i = j) \quad (3)$$

If we are given K and observation x_0, we then look for the K poins closes to x_0 and then it calculates the conditional probability for class j, which is accepted or denied, as a fraction of the points in N_0 whose response variable, card, is equal to j.
In my paper, I started by choosing an arbitrary number for K, 12.

If K is too small, the model is not smooth and will over fit the data. On the other hand, if K is too large, the model is very smooth but it does not perform well in train and test data and will under fit. So now, the question is, "What is the optimal value of K".

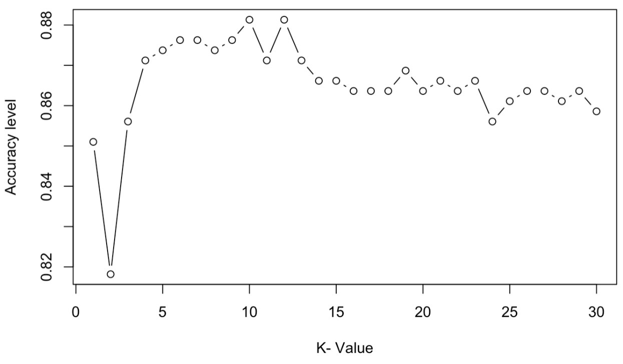
Results of KNN

When implementing K Nearest Neighbors, I tried to find the optimal value of K, which can answer the question above, allowing us to find which value of K will give us the highest prediction accuracy. In order to do this, I tried values of K between 1 and 1000 and after K = 245, the accuracy rate stayed at .7878788. I then tried values between 1 and 30 and graphed them, resulting in 「Fig.4 Accuracy of each K values」 and looking at the highest number, I observed that K=10 gave us the highest accuracy.

```
knn_matrix = table(card_pred, Y_card_tst)
knnperf = (sum(diag(knn_matrix)))/sum(knn_matrix)
knn_matrix
knnperf
```

```
          Y_card_tst
card_pred   0   1
        0  46   9
        1  38 303
[1] 0.8813131
```

Figure 3. KNN Matrix



```
1 = 0.8510101 2 = 0.8181818 3 = 0.8560606 4 = 0.8712121 5 = 0.8737374 6 = 0.8762626 7 = 0.8762626
8 = 0.8737374 9 = 0.8762626 10 = 0.8813131 11 = 0.8712121 12 = 0.8813131 13 = 0.8712121 14 =
0.8661616 15 = 0.8661616 16 = 0.8636364 17 = 0.8636364 18 = 0.8636364 19 = 0.8686869 20 =
0.8636364 21 = 0.8661616 22 = 0.8636364 23 = 0.8661616 24 = 0.8560606 25 = 0.8611111 26 =
0.8636364 27 = 0.8636364 28 = 0.8611111 29 = 0.8636364 30 = 0.8585859 [1] 0.8813131
[1] 10
```

Figure 4. Accuracy of each K values

Even with the optimal K value, 10, K Nearest Neighbors correctly classifies our data 88.13% of the time. When looking at the millions of people that open credit card accounts every year, I see that K Nearest Neighbors would not be an optimal model to use since it will still be misclassifying millions of people in the United States for a credit card application.

3.4 Shrinkage and Selection Methods

Bias-Variance Tradeoff

Our goal for good estimators is that they should have small prediction errors. However, there are issues with the bias and variance of models. Bias is the difference between our prediction of our model and the correct value of what we are trying to predict. Models with high bias will be prejudiced and oversimplify our model leading to high error on training and testing data. Variance is the error that is caused due to a model's sensitivity in changes in the data.

High variance will cause a model to be very specific to the training data and overfit.

Shrinkage Methods, or subset selection, aims to find a subset of our regressors that predict our model best. At first, Shrinkage models with use all regressors and then starts to constrain/regularize our estimates. The two shrinkage methods I perform in this paper are Ridge and Lasso. By "shrinking" the coefficients, we push them towards 0 which make them work better on new data sets. Shrinking the coefficients also allows a better evaluation of the individual effects of each variable.

Ridge Regression

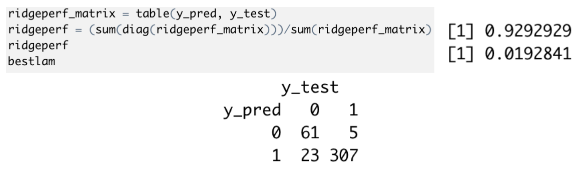
$$RSS = \sum_{i=1}^{n}(y_i - B_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 \tag{4}$$

Objective Function Ridge Regression and Least squares are similar but in ridge regressions, the coefficient estimates $\hat{\beta}^R$ that minimize:

$$\sum_{i=1}^{n}(y_i - B_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = RSS + \lambda\sum_{j=1}^{p}\beta j^2 \tag{5}$$

$\lambda \geq 0$ is a tuning parameter. What is being done is that ridge finds coefficient estimates that fit the data well and minimize RSS. The second term, $\lambda\sum_{j=1}^{p}\beta_j^2$, is the shrinkage penalty and is small when the β's are close to 0. But in Ridge, $(\beta_j x_{ij})^2$ will shrink the coefficients to zero, but are never zero.

Results of Ridge



```
ridgeperf_matrix = table(y_pred, y_test)
ridgeperf = (sum(diag(ridgeperf_matrix)))/sum(ridgeperf_matrix)
ridgeperf
bestlam
```

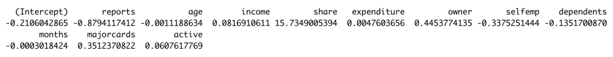
```
[1] 0.9292929
[1] 0.0192841
```

```
       y_test
y_pred   0   1
     0  61   5
     1  23 307
```

Figure 5. Ridge Confusion Matrix

When using Ridge Regression, I was able to achieve a classification rate of .929. Additionally, the best lambda was found to be .0192841 and is the value that minimizes our error.

In Figure 5 and 6, we see the coefficients of our variables when they are shrunk using Ridge Regression. Many variables are close to zero. In Figure



```
  (Intercept)       reports           age        income         share   expenditure         owner       selfemp    dependents 
-0.2106042865 -0.8794117412 -0.0011188634  0.0816910611 15.7349005394  0.0047603656  0.4453774135 -0.3375251444 -0.1351700870 
       months    majorcards        active 
-0.0003018424  0.3512370822  0.0607617769 
```

Figure 6. Values of the Ridge Coefficients

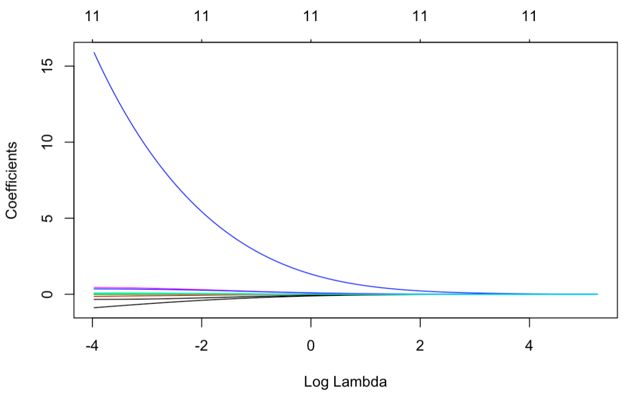


Figure 7. Values of the Ridge Coefficients Graphed

6., each curve represents ridge regression coefficient plotted as a function of log(λ). As λ increases, the ridge coefficients shrink towards zero.

In Ridge regression, I see that the largest coefficients are for the variables **share, owner, reports, and selfemp**. Based off of my Ridge Regression model, those four variables have the largest impact on my predictor variable.

Lasso Regression

LASSO Regression, or Least Absolute Selection and Shrinkage Operator), will force coefficient estimates toward zero.

Previously, I mentioned that the penalty $\beta_j x_{ij})^2$ in Ridge regression will shrink the coefficients to zero, however, will never make them zero. However, many banks and credit card companies have large data on many different people and they may also have a large number of variables, p. LASSO solves this issue and makes coefficient estimates equal to 0. Lasso performs variable selection which allows models generated to be easily interpreted.

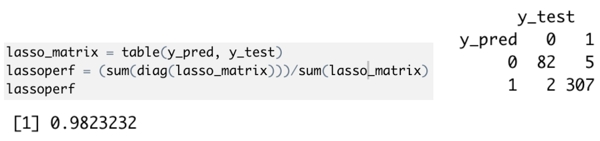
Objective Function

$$\sum_{i=1}^{n}(y_i - B_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p} \mid \beta_j \mid = RSS + \lambda\sum_{j=1}^{p} \mid \beta j \mid \tag{6}$$

Results of Lasso Regression

Lasso correctly classified 98.23% of our test data. Also, our lambda in Lasso Regression is .0002166.

The curves in Figure 9. display how Lasso makes



```
lasso_matrix = table(y_pred, y_test)
lassoperf = (sum(diag(lasso_matrix)))/sum(lasso_matrix)
lassoperf
```

```
      y_test
y_pred   0   1
     0  82   5
     1   2 307
```

```
[1] 0.9823232
```

Figure 8. Lasso Confusion Matrix

```
(Intercept)     reports      income       share       owner  dependents  majorcards      active 
-0.73851677 -1.05336272  0.07419872 70.23998753  0.22440783 -0.01879572  0.17443795  0.06705179 
```

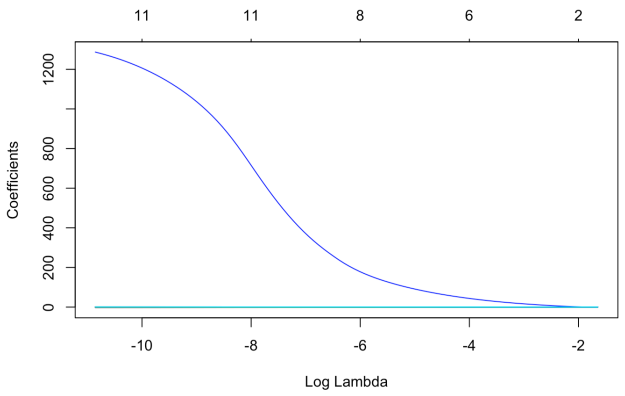


Figure 9. Lasso Coefficient Graphs

the coefficient estimates equal to zero. I also observe that Lasso is a good model in classifying our data with an accuracy of 98.23%.Lasso also made the following variables 0: age, expenditure, selfemp, and months. Thus, the variables that Lasso did not set equal to zero will be most important in determining the classification of a credit card application which are: **reports, income, share, owner, dependents, majorcards, and active**. I also will weigh this model more because 98.23% correct classifications are high.

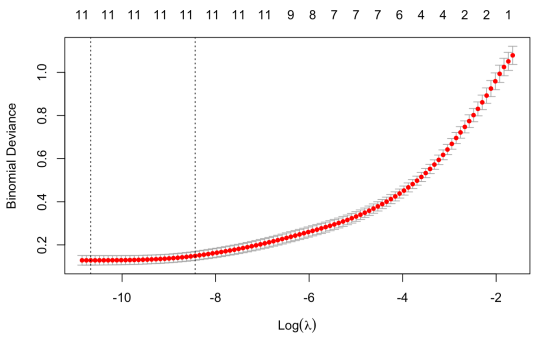


Figure 10. Lasso Cross Validation

Comparison between Shrinkage methods

With Ridge and Lasso, we see how they both perform variable selection and try to determine which variables are most significant in models. Ridge makes the coefficients towards zero but Lasso sets them to zero. However, with both, the value of λ is significant. λ, the tuning parameter, is determined for both with cross validation. In our paper, we chose a grid of λ values and tested the values of λ to see which gave us the least binomial deviance as seen in Figure 10 and Figure 11. For both, our value of λ is small(both are close to 0) which tells us that the optimal fit involves a small amount of shrinkage relative to the least squares solution. This may also mean that we might just use the least squares solution.

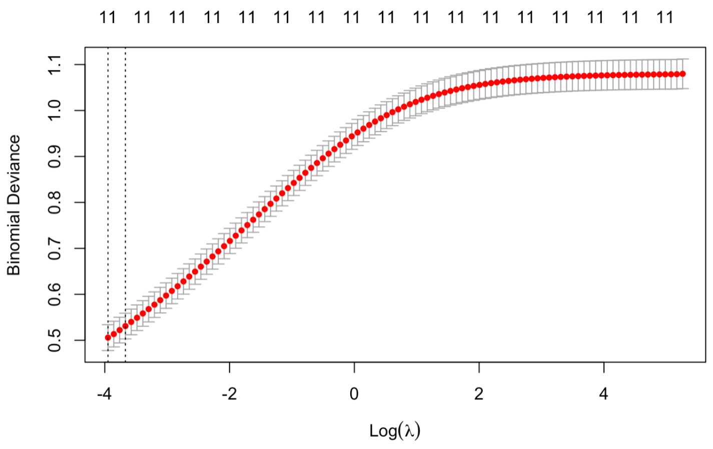


Figure 11. Ridge Cross Validation

Also, when comparing the performance of Ridge and Lasso Regression, .929292 and .9823232, respectively, there is a large difference between Lasso and Ridge. While the difference is about 5.303% – this **translates to hundreds of thousands of new credit card applicants every year**. Incorrectly classifying all those people can **be extremely detrimental for a bank or company**.

The **Lasso model is advantageous over ridge regression**, specifically i our model, due to its nature of producing simple and interpretable models. In fact, we see the superiority of variable selection and the difference it makes when setting variables to 0. However, the similarity of both is that as λ increases, variance and bias decreases.

3.5 Tree Based Methods

Decision Trees

In my paper, because my response variable was categorical, I implemented a classification tree. Classification Trees segment our predictor space into a number of simple regions. It performs recursive binary splitting in order to divide a predictor space into a distinct and non-overlapping region. The main objective is to predict that "each observation belongs to the most commonly occurring class of training observations in the region to which it belongs." (page 311) [6] When splitting, we have certain criteria for each case:

Classification Error Rate

$$E \;=\; 1 - \max_k(\hat{p}_{mk}) \tag{7}$$

$\hat{p}_{mk}$ is the proportion of training observations in the mth region from kth class. But, this classification error is not sensitive enough for pruning/tree building so two other objectives are preferred. [6]

Another possible objective is the Gini Index:

$$\sum_{k=1}^{K} = \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{8}$$

which measures total variance across K classes.

A second alternative is entropy:

$$D \;=\; -\sum_{k=1}^{K} = \hat{p}_{mk} log \hat{p}_{mk} \tag{9}$$

Both entropy and the Gini index are used to measure the quality of a particular split.

Results of Classification Tree

Running a Classification Tree Model gives us the decision tree in Figure 12. This tree has a depth of three and 4 leaf nodes. It also only uses the variables expenditure, reports, and active to prune the tree. This tree looks extremely simple however, performs well.

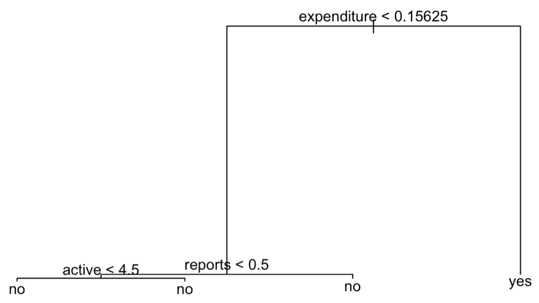
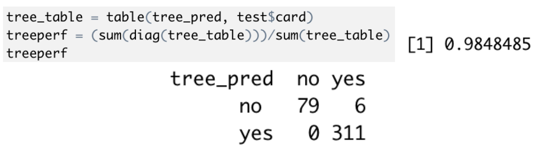


Figure 12. Credit Card Decision Tree

Performance

Figure 13. shows that our Classification Tree performs well in classification with correct predictions



```
tree_table = table(tree_pred, test$card)
treeperf = (sum(diag(tree_table)))/sum(tree_table)
treeperf
```

```
[1] 0.9848485
```

```
tree_pred  no yes
      no   79   6
      yes   0 311
```

Figure 13. Decision Tree Confusion Matrix

in 98.484% of my test data. **Recursive Binary Splitting** creates a decision tree that is accurate and credit card companies may use decision trees to determine the status of a credit card application. Additionally, I notice that when the node expenditure $\leq$ 0.15625 is true, the credit card application is accepted. However, In our testing data, we see that there were 6 cases where it misclassified that data and approved an application that was originally denied. I also see that the most important variables in a decision tree that determines the credit card application is **expenditure, reports, and active**. Another downfall of decision trees is that it suffers from high variance.

Bagging

Bagging, or bootstrap aggregation, tries to solve this issue of high variance by averaging a set of observations which will reduce the models variance. It takes many training sets and averages the resulting predictions.

However, we do not have multiple training sets - this is where bootstrap comes into play. Bootstrap works by "taking repeated samples from a single training data set." [6] This approach gives us B different bootstrap training sets and we can now train our models on these training sets and then average the predictions to get:

$$\hat{f}_{bag}(x) \;=\; \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x) \tag{10}$$

For classification, bagging works by recording the class predicted by each of the B trees and finds the overall prediction of the most commonly occurring class among the B predictions.

Results of Bagging

In Figure 14. bagging correctly classify 97.98% of our testing data. From these results, I see that a

```
yhat.bag  no yes
     no   80   4
     yes   4 308
[1] 0.979798
```

Figure 14. Bagging Confusion Matrix

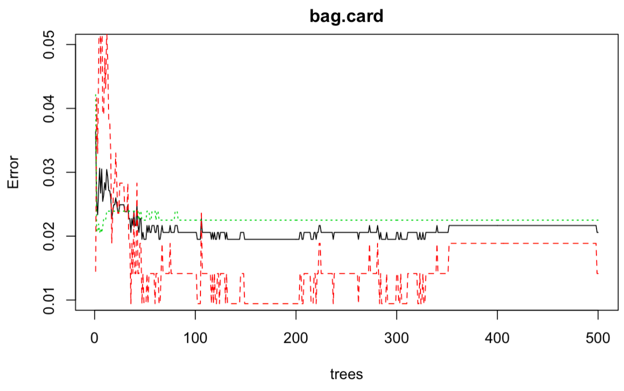


Figure 15. Bagging Graph

bank or credit company may not need to use bagging since a decision tree had a higher accuracy. In my paper, boosting did not perform as well as the classification-single tree previously mentioned so a company would probably prefer the single tree method over bagging. Furthermore, our most important variable is expenditure which seems logical because our Decision Tree had expenditure as the first prune. Additionally, if a bank was looking to implement Bagging in their credit card application model, they would want to emphasize expenditure for an applicant.

Random Forest

Random Forest is similar to bagging but the main objective is that everytime we split the tree, random subsets of predictors are used instead of all predictors. So at each split of the tree, we do not consider a majority of available predictors. This decorrelates the trees and solves the issue of one very strong predictor being used as the top split in numerous bagged trees.

Results of Random Forest

Random Forest is an improvement on Bagging. Figure 16 shows that it correctly classified 98.23% of our testing data. Furthermore, Figure 19. shows us that in our Random Forest Model, **expenditure, share, and reports** are our three most important variables. So credit card companies may consider a

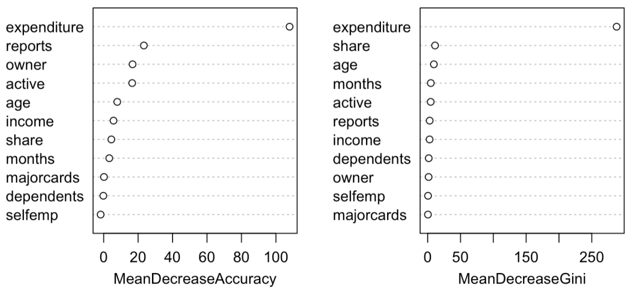


Figure 16. Bagging Relative Importance

```
yhat.rf  no yes
    no   81   4
    yes   3 308
[1] 0.9823232
```

Figure 17. Random Forest Confusion Matrix

random forest over bagging when developing models to classify credit card applications.

Boosting

Boosting is an approach to improve predictions from a decision tree and does not use bootstrap sampling. In boosting, trees are grown sequentially by using data from previous grown trees. This is different from a decision tree, which is fitting a single tree to the data, where boosting learns slowly by fitting a decision tree to the residuals from the model rather than the outcome, or our response variable. It also uses multiple tuning parameters.

B: number of trees - boosting can overfit if B is too large

λ: shrinkage parameter

d: number of splits in each tree

Algorithm[6]

1. Set $\hat{f}$(x) and $r_i = y_i$ $\forall i$ in the training set.

2. For b = 1,2,...,B,repeat:

(a) Fit a tree $\hat{f}^b$ with d splits (d+1 terminal nodes) to the training data (X,r).

(b) Update $\hat{f}$ by adding in a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x) \tag{11}$$

(c) Update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x^i) \tag{12}$$

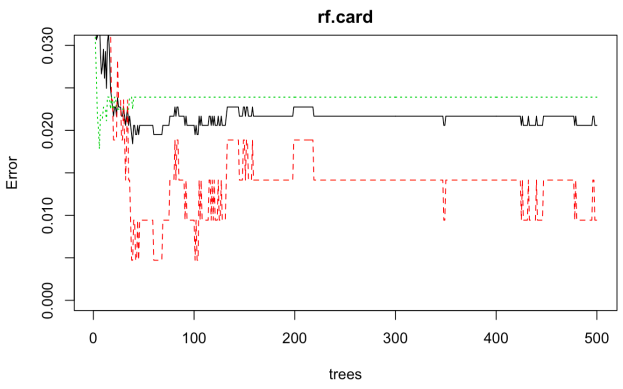


Figure 18. Random Forest Graph

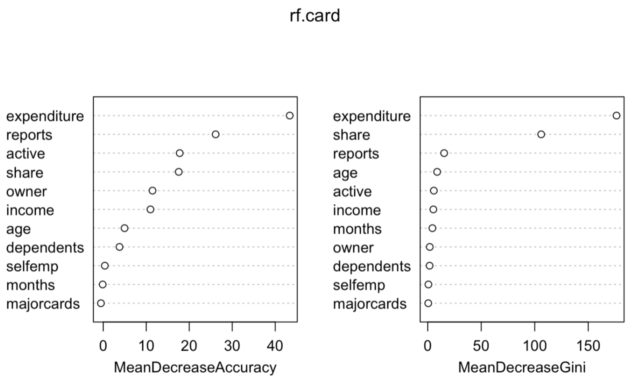


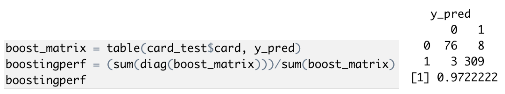
Figure 19. Random Forest Variable Importance

3. output the boosted model

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^{b}(x) \tag{13}$$

We then apply the boosted model to our dataset.

Results of Boosting



```
boost_matrix = table(card_test$card, y_pred)
boostingperf = (sum(diag(boost_matrix)))/sum(boost_matrix)
boostingperf
```

```
   y_pred
     0   1
  0 76   8
  1  3 309
[1] 0.9722222
```

Figure 20. Boosting Confusion Matrix

Boosting classified our testing data correctly 97.222% of the time. From Figure 20 and Figure 21, we see that the variables with the largest influence are **expenditure, share, age, and reports**. So a bank using a boosting model should prioritize those four variables and observe them closely when classifying a credit card application.

3.6 XGBoost

XGBoost, or Extreme Gradient Boosting, has specific differences from Boosting mentioned above. XGBoost implements gradient boosting but also includes

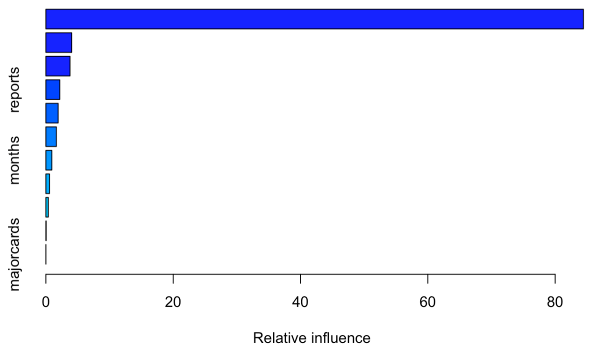
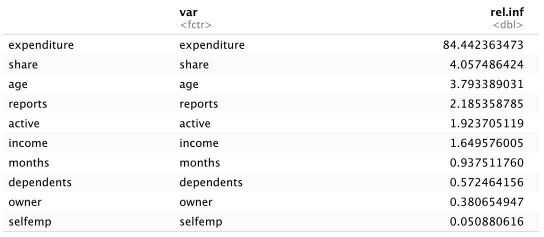


Figure 21. Boosting Relative Influence Plot



	var <fctr>	rel.inf <dbl>
expenditure	expenditure	84.442363473
share	share	4.057486424
age	age	3.793389031
reports	reports	2.185358785
active	active	1.923705119
income	income	1.649576005
months	months	0.937511760
dependents	dependents	0.572464156
owner	owner	0.380654947
selfemp	selfemp	0.050880616

Figure 22. Boosting Relative Influence Values

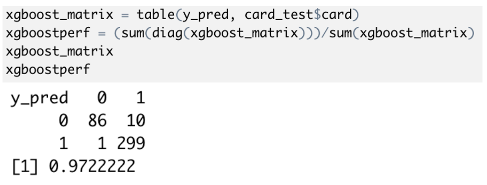
a regularization parameter and implements parallel processing.

Its objective function: [7]

$$\begin{aligned} obj &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{t-1} + f_t(x_t)) + \Omega(f_t) \\ &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{t-1} + f_t(x_t)) + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2 \end{aligned} \tag{14}$$

Tuning Parameters:
max_depth: max depth of tree
eta: controls learning rate
γ: min. loss reduction required to further partition on a leaf node. Larger value is more conservative.
λ: L2 regularization term on weights. ω is vector of scores on leaves

Results of XGBoost



```
xgboost_matrix = table(y_pred, card_test$card)
xgboostperf = (sum(diag(xgboost_matrix)))/sum(xgboost_matrix)
xgboost_matrix
xgboostperf
```

```
y_pred   0   1
     0  86  10
     1   1 299
[1] 0.9722222
```

Figure 23. XGBoost Confusion Matrix

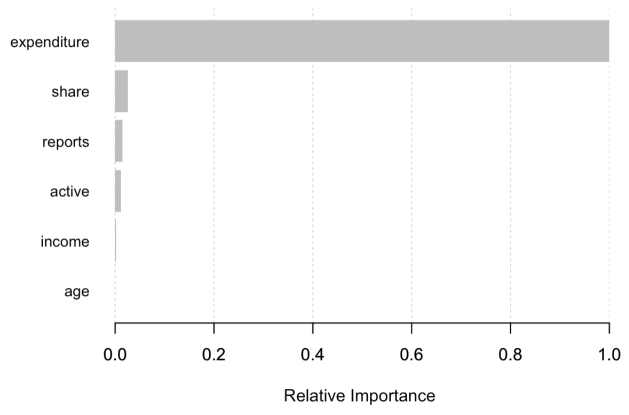


Figure 24. XGBoost Relative Importance

With an XGBoost model implementation, we see that it correctly predicted our test data 97.22% correctly(Figure 22). Furthermore, the graph of relative importance also showed that **expenditure, share, reports, and active** are the most important. So a bank or credit card company implementing XGBoost would want to put more weights on those four variables.

Comparison of Tree Models

During my research, I implemented five different tree based machine learning models: Decision Trees, Bagging, Random Forest, Boosting, and XGBoost. Their performances on the test data were 98.484% 97.98% 98.232% 97.222% and 97.222% respectively. The most accurate mode was the decision tree, which was the simplest model among all tree-based models. The different tree based models would give us different classification rates because they behave differently and have different models in their predictions and pruning. For example, Bagging and Random Forest fit separate decision trees and use bootstrapping to create different training sets while others do not.

It is also important to observe that our decision tree was the most accurate model which was also the simplest. Many of the other tree models were supposed to be an improvement off the previous tree model mentioned. For example, bagging is supposed to solve the issue of high variance in decision trees. However, our decision tree only had a depth of 3 and was still able to perform best. This may have been the case because the credit card data set used had only 1319 rows and 12 columns. Our data set is still relatively small compared to the large amounts of data available to large banks and credit corporations. This may mean that our dataset did not need the most complex models to perform well. This is important to take into account because if a bank was trying to build a tree based model of classifying credit card applications, it is not necessarily the case that more complex models are better. However, this may only be applicable to the limited dataset I have. The more complex model may perform at a much higher level when given more variables and information.

Furthermore, I can see which variables are consistently significant in the tree models. The significant variables in each tree models were:
Decision Tree: expenditure, reports, active
Bagging: expenditure
Random Forest: expenditure, share, reports
Boosting: expenditure, share, age, reports
XGBoost: expenditure, share, reports, active

From the tree based models, it is consistent that the three most significant variables in classifying a credit card application is expenditure, share, and reports. So if a bank or credit company was aiming to implement a tree based model for their applications, these would be the three most important variables they should focus on. Thus, applicants should also ensure that they keep those variables in a good standard. Applicants may need to make sure that they do not have a high average monthly credit card expenditure but also make ensure that they do spend and make use of a credit card. Furthermore, they should try to keep the number of derogatory reports they have low.

3.7 Neural Network

Neural Networks are data that goes through a network by layers until it outputs our classification. Each

Classification Softmax Function:

$$g_k(T) = \frac{e^{T_k}}{1 + e^{T_k}} \qquad (15)$$

Function for the Hidden layers: Sigmoid Function:

$$\sigma(v) = \frac{1}{1 + e^{-v}} \qquad (16)$$

We then perform fitting which finds unknown parameters or our weights: Cross Entropy(deviance):

$$R(\theta) \;=\; -\sum_{k=1}^{K}\sum_{i=1}^{N} y_{ik} log f_k(x_i) \qquad (17)$$

Neural Networks estimates many parameters and in order to do so efficiently, it performs gradient descent and back-propogation. Gradient descent is an optimization that moves step-wise toward the steepest direction. (New weight = old weight - Derivative Rate * learning rate). This process is called back-propogation. [7]

Results of Neural Network

```
error_table
neuralnetperf = (sum(diag(error_table)))/sum(error_table)
neuralnetperf
```

```
          test_class
             0   1
         0  82   2
         1   3 309
       [1] 0.9873737
```

Figure 25. Neural Network Confusion Matrix

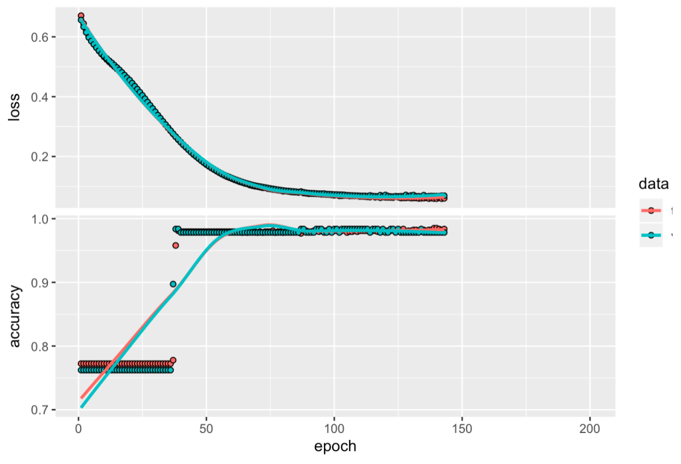


Figure 26. Neural Network Epochs

The neural network had the best performance and correctly classfied 98.737% of our testing data. However, this comes with a computational cost because it does take a long time to run the neural network. However, a company may be interested in implementing neural networks despite its long run time and energy use if it is able to have a better model of credit card application acceptances. Figure 26. also tells us the number of epochs we need to perform in order to get a high accuracy or when to stop.

3.8 Support Vector Machines

A support vector machine classifies the response variables based on the space that they are in. SVM's makes the assumption that points that classify similarly will be close to each other in a space. A support vector classifier is used for points that are linearly separable but there are instances where points are not linearly separable, as seen in Figure 27. SVM's adds higher polynomials of X to increase the dimension of the space and the hyperplane that solves the maximization problem with linear margins:

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i < x, x_i > \tag{18}$$

¡xi, xi'¿ is an inner product Figure 27. uses a radial kernel:

$$K(x_i, x_i^{'}) = exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2) \tag{19}$$

Results of Support Vector Machines

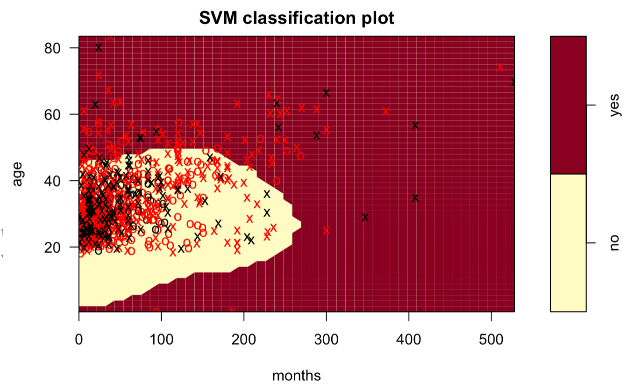


Figure 27. SVM of age and months

```
svmperf = (sum(diag(svm_table)))/sum(svm_table)
svmperf
```

```
[1] 0.8737374
```

Figure 28. SVM Matrix

The SVM Model only correctly classifies our test data 87.37% correctly. A credit card company may not want to implement a SVM model in this case. We could try to do further research to see if polynomial kernel will perform better.

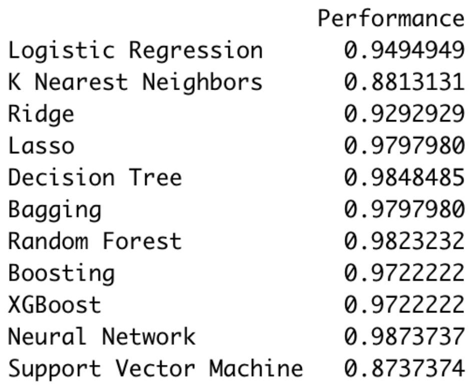
4 Machine Learning Models Comparisons

Significant Variables Different Machine Learning Models provided us information on what variables ended up being the most significant in their own respective model. They were as following:

Logistic Regression: months, age, owner, expenditure
Ridge: share, owner, reports, selfemp
Lasso: reports, income, share, owner, dependents, majorcards, active
Decision Tree: expenditure, reports, active
Bagging: expenditure
Random Forest: expenditure, share, reports, active
Boosting: expenditure, share, age, reports
XGBoost: expenditure, share, reports, active

From these models, we see that credit card companies may put a **higher emphasis on expenditure, share and reports**

Results of all Models



```
                         Performance
Logistic Regression        0.9494949
K Nearest Neighbors        0.8813131
Ridge                      0.9292929
Lasso                      0.9797980
Decision Tree              0.9848485
Bagging                    0.9797980
Random Forest              0.9823232
Boosting                   0.9722222
XGBoost                    0.9722222
Neural Network             0.9873737
Support Vector Machine     0.8737374
```

Figure 29. Performance of all Models

In the order of greatest to least performance, the models were: neural networks, decision trees, random forests, Bagging Lasso, Logistic Regression, Ridge, Boosting XGBoost, K Nearest Neighbors, and Support Vector Machines.

5 Conclusion

In my research, I was able to determine the most important variables in determining the status of a credit card application as well as finding out which machine learning model performs the best on the data set. Banks may use those variables (expenditure, share, and reports) and models in their own algorithms when deciding whether to accept an applicant for a credit card. Additionally, these results benefit the applicant as well so they know what factors credit card companies take into account the most in order to get approved for a credit card.

Neural networks, a powerful machine learning model, performed the best and we were also able to determine the rank the performance of the machine learning models. However, taking into account that decision trees also performed only slightly worse and takes up less computational power, it may not be worth it to use neural networks. In my opinion, with millions and millions of people applying for credit cards, even a small percentage improvement can make a large difference. The goal of the credit card company is also to minimize the risk - approving someone who was supposed to get denied is more costly than denying an applicant who was supposed to get approved.

Further research is to examine whether or not the computational power of running neural networks will be profitable or beneficial enough for a credit card company if it only performs better by a little amount. Another area to look into is how the data set can be used in conjunction with credit card default data.

References

[1] https://www.creditcards.com/credit-card-news/credit-card-use-availability-statistics-1276.php

[2] https://wallethub.com/edu/cc/number-of-credit-cards/25532/

[3] Smith, Maher, and Khaneja: ”Credit Approval Analysis”, November, 2017 https://www.researchgate.net/publication/ 321002603_Credit_Approval_Analysis_using_R

[4] Kuhn, R. Analysis of Credit Approval Data http://rstudio-pubs-static.s3.amazonaws.com/73039 _9946de135c0a49daa7a0a9eda4a67a72.htmlinterpret-the-model-and-research-questions-answered

[5] Kleiber, C. Zeileis, A. (2008) Applied Econometrics with R, Springer-Verlag, New York. ISBN 978.0.387.77316.2

[6] Gareth, J. et al. (2013) An Introduction to Statistical Learning New York. ISBN. 978.1.4614.7138.7 http://faculty.marshall.usc.edu/gareth-james/ ISL/ISLR%0Seventh%0Printing.pd

[7] Khazra, N. and Farhoodi, A. (2020) ECON 490: Applied Machine Learning Course Slides/Labs

*All Code used for this paper can be found at www.github.com/andrewma/CreditCardMLAnalysis

*Special thanks to Nazanin Khazra and Abdollah Farhoodi