Sarah Hollingsworth
DATA 621
Franklin University
Classification Models to Predict Credit Worthiness

I. Abstract

The decision of a financial institution to extend credit to a customer is a complex one, which must consider the ability of a customer to repay the debt and weigh this against the potential profit for the financial institution. As techniques in statistical modeling have improved, these models can be used to identify potential patterns of behavior which make a customer more likely to not be extended credit. In this paper we will explore the German Credit Data dataset to compare classification accuracy using two classification techniques, the Random Forests and Support Vector Machine (SVM). The goal of the classification models will be to correctly classify customers as either being extended or declined credit. The German Credit Data dataset contains twenty attributes, which includes both quantitative and qualitative variables ranging from the customer's age to the type of loan. Utilizing classification accuracy, sensitivity, precision, and AUC ROC curves, we will compare the different classification methods and make a recommendation on the best approach.

II. Introduction

The history of lending practices has changed dramatically over the past one hundred years, from the introduction of credit cards and mortgages to the financial landscape around the turn of the twentieth century to the subprime mortgage collapse in 2007. The ability for a statistical model to provide financial institutions a way of identifying customers who might default on their loan has broad reaching implications. (Zurada, Kunene, & Guan, 2014) Financial institutions can lower their risk exposure by correctly classifying high-risk customers and at the same time the extension of credit to low risk customers is a cornerstone for how modern economies work by adding cash flow to the system. (Zurada, Kunene, & Guan, 2014) There are varying opinions around the spending habits of customers and how the extension of credit to high-risk customers can result in further risky behaviors and changes to the marketplace. Some assert by extending credit to these high-risk individuals, consumers are using these loans not necessarily for items beyond their financial means but rather for basic living necessities. (Scott III, June 2007) This would indicate these customers are being pushed into patterns of behavior due to financial factors such as loss of income or rising costs of goods.

The use of classification models to predict a customer's credit worthiness can also have negative implications, since there may be several unintended assumptions made about customers in the creation of the model. These can include the presumption high income earning customers with secondary educations are more likely to pay back a loan than a low-income earning customer with no secondary education. (Krishnamurthi, 2007) With higher scrutiny being placed on financial institutions lending practices through such laws as the Fair Housing Act of 1968 , Community Reinvestment Act of 1977 and Equal Credit Opportunity Act of 1974. The burden is placed on the financial institution to demonstrate they do not treat customers in a disparate manner based on their age, sex, religion, or other protected statuses. At some point in the future, financial institutions may also be required to provide the evidence of how a model was built to evidence there was no disparate treatment. When determining the factors to use in the classification model, financial institutions must take all the above mentioned into consideration and weigh the risk and reward associated with each.

III. Data Exploration

In the German Credit Data dataset provided by Dr. Hans Hofmann, there are one thousand observations which include the target variable, seven quantitative and thirteen qualitative predictor variables.

Qualitative Variables

The qualitative variables include:

- Status of existing checking account
- Credit history
- Loan purpose
- Amount in savings or bonds
- Years in present employment
- Gender and marital status
- Other debtors or guarantors
- Property owned
- Other installment plans
- Housing status
- Employment status
- Telephone status
- Foreign worker status.

Quantitative Variables

The quantitative variables include:

- Duration in months of the loan
- Credit amount
- Installment rate in percentage of total income
- Present residence since
- Age
- Number of existing credits at this bank
- Number of people being required to provide maintenance for

Cost Matrix

A cost matrix in which the rows represent the actual classification and the columns the predicted classification. The cost matrix will need to be applied to the final models to determine a weighted precision score.

```
     1   2
1    0   1
2    5   0
```

(1 = Good, 2 = Bad)

Target Variable

Also included is the target variable, which indicates if the financial institution extended credit or not to the customer. When creating the classification model one of the most important items to address first is if there is class imbalance in the dataset. Figure 3.1 shows the distribution of our target variable and indicates a class imbalance does exists, with approximately thirty percent of the observations belonging to the minority class which contains declined customers. Class imbalance can lead to decreased accuracy in the model because the majority class may influence the model and therefore not pick up the features in the minority class which lead to higher accuracy.

Figure 3.1

Exploratory Analysis

Among the exploratory variables, the status of a customer's checking account shows a larger proportion of customers who had balances less than two hundred deutschmarks or negative balances in their checking account. (Figure 3.2) While Figure 3.3 shows customers with balances less

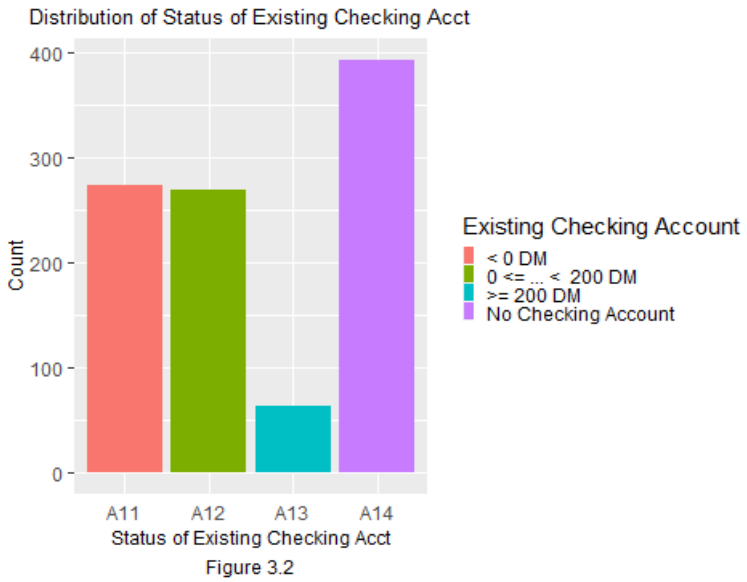

Figure 3.2

than two hundred deutschmarks or negative balances were equally likely to have credit approved or declined and customers with balances greater than two hundred deutschmarks or no existing checking account are more likely to have credit extended to them. Figures 3.4 and 3.5 show there

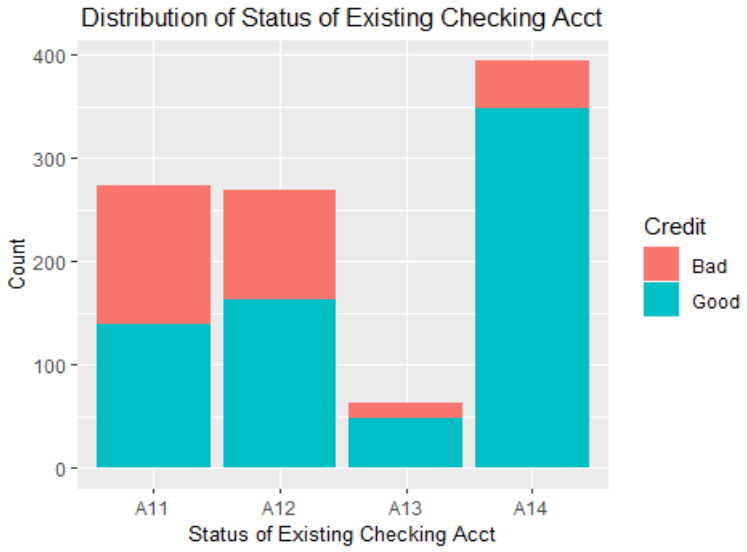

Figure 3.3

are four loan types which have a higher proportion of customers who had credit extended to them, including used cars, radio or television, domestic appliances, and retraining. When the dollar amount for each loan type is considered, Figure 3.6 shows the amount of credit extended is

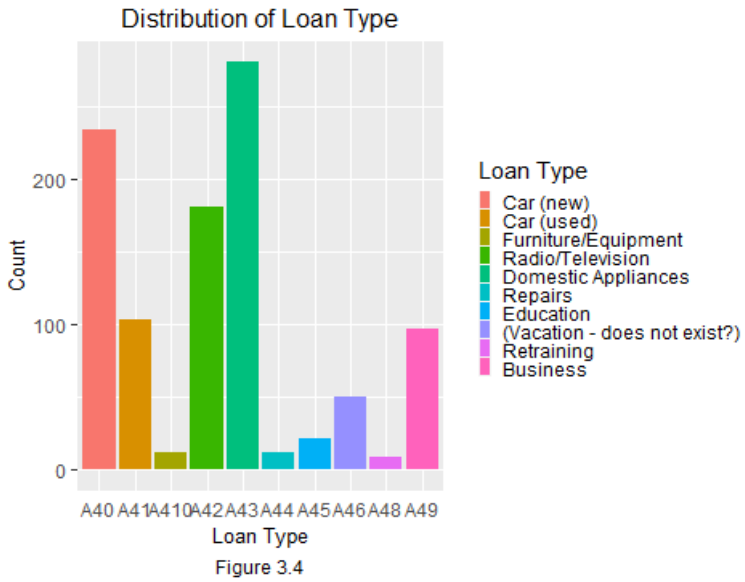

Figure 3.4

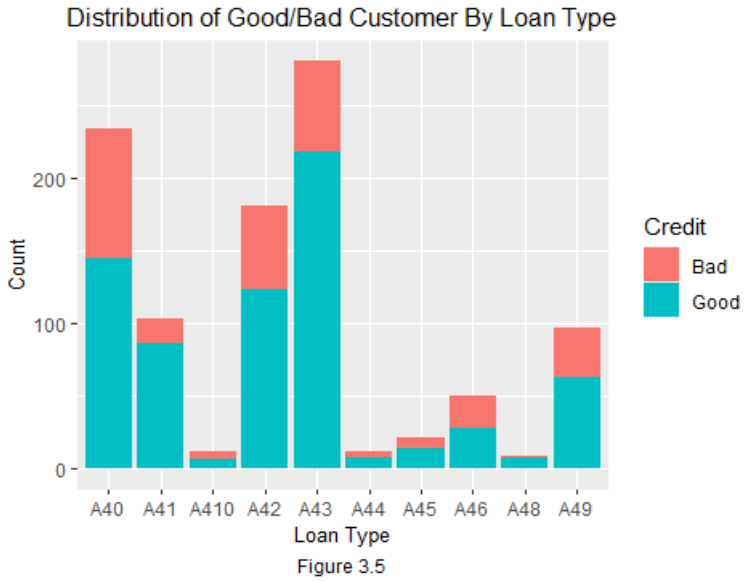

Figure 3.5

distributed equally among all loan types apart from furniture/equipment, repairs, and retraining. For furniture and equipment, the distribution contains several outliers as well as a distribution ranges from extremely low to high amounts. However, the repairs and retraining

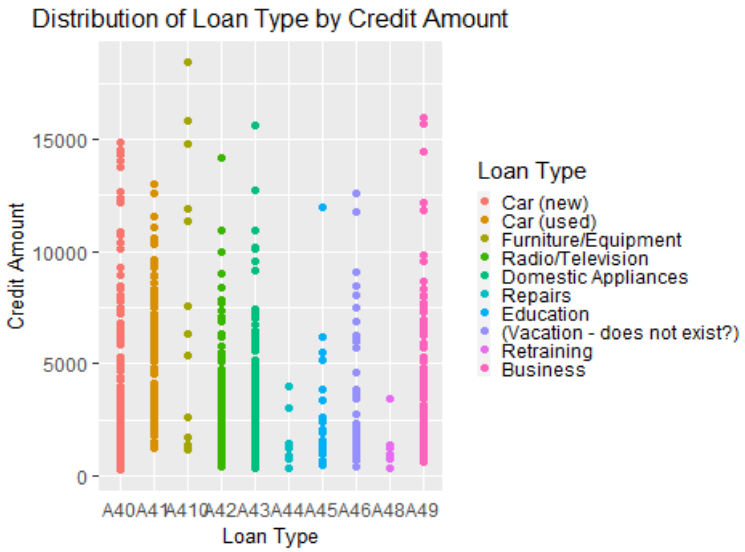

Figure 3.6

categories have a small distribution and contain low credit amounts. These loan types are among the smallest proportions of customers represented in the dataset but may also introduce bias into the model. The credit history of the customer is another important indicator in the decision of the whether credit is extended or declined. Figure 3.7 shows the largest proportion

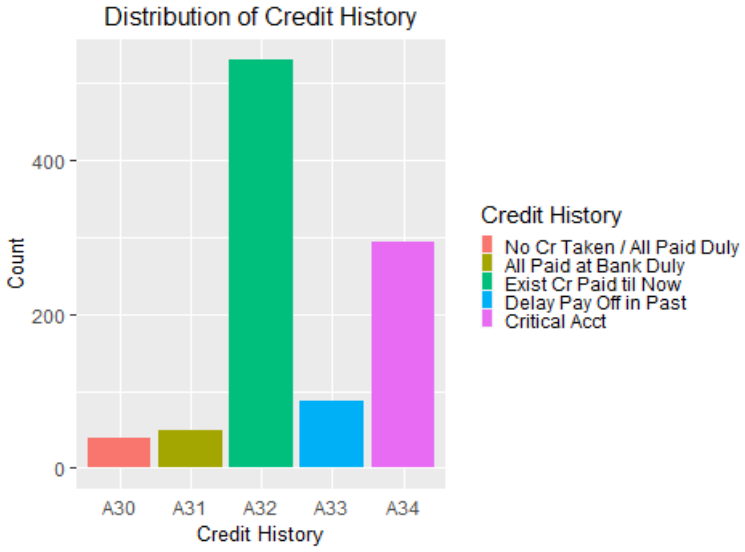

Figure 3.7

of customers who are extended credit are customers who have paid their credits until the current time or are critical accounts with credits at other banks. Figure 3.8 displays that among

FIgure 3.8

these customers, the largest proportion of customers declined credit are those who have paid their credits until the current time. In Figure 3.9, we can see for the term of the loan, the proportion of customers extended credit or not converges at thirty months. This indicates loan

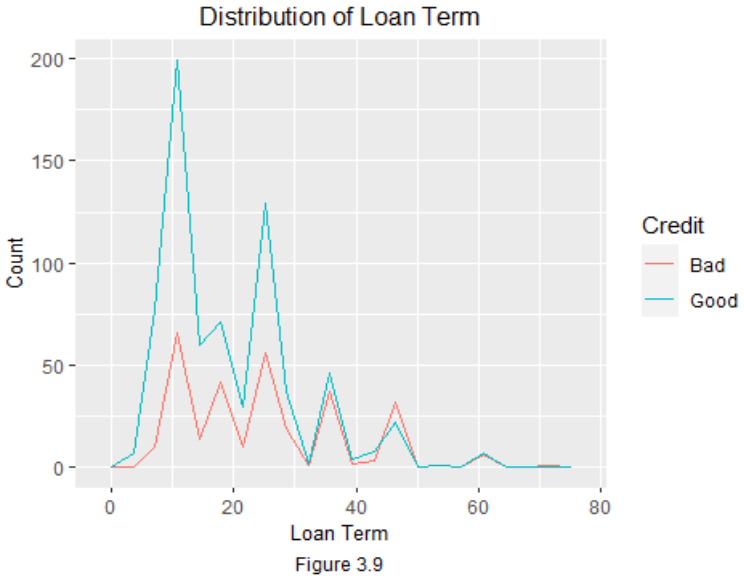

Figure 3.9

terms greater than thirty months are riskier for the financial institution and should be a considering factor in the final model. Figure 3.10 displays the distribution of credit amount extended by approved or declined. The proportion of approved or declined customers

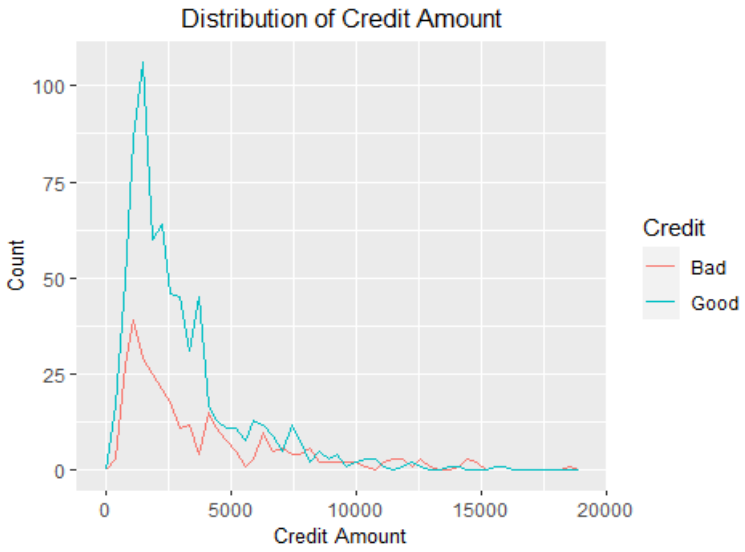

Figure 3.10

converges at five thousand deutschmarks, loan term is another benchmark to include in the final model. The age of the customer displayed in Figure 3.11 and the proportion of customers extended or declined credit is the same apart from convergences at ages sixty and seventy-five. Further inspection of age by credit amount proves this theory, as shown in Figure 3.12, the

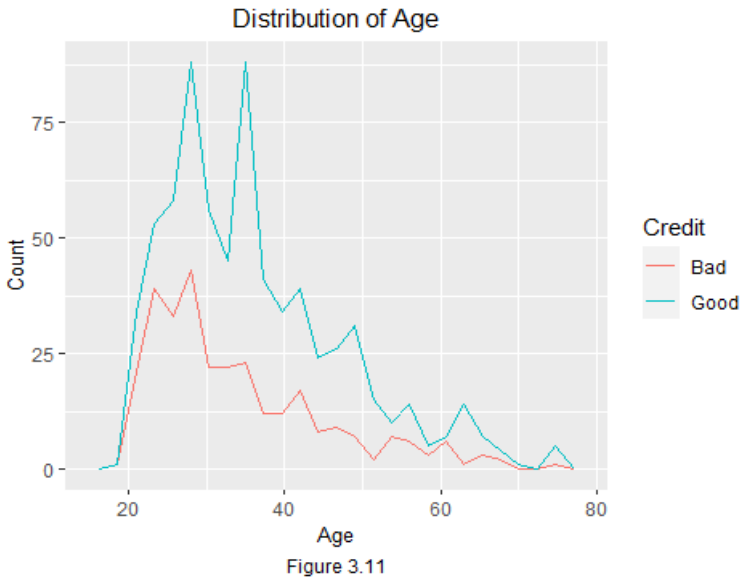

Figure 3.11

scatterplot shows older customers with smaller credit amounts are approved more frequently than younger customers. As the credit amount increases, the approve or decline rate for customers appears to be evenly dispersed. The distribution of age for each loan type appears to

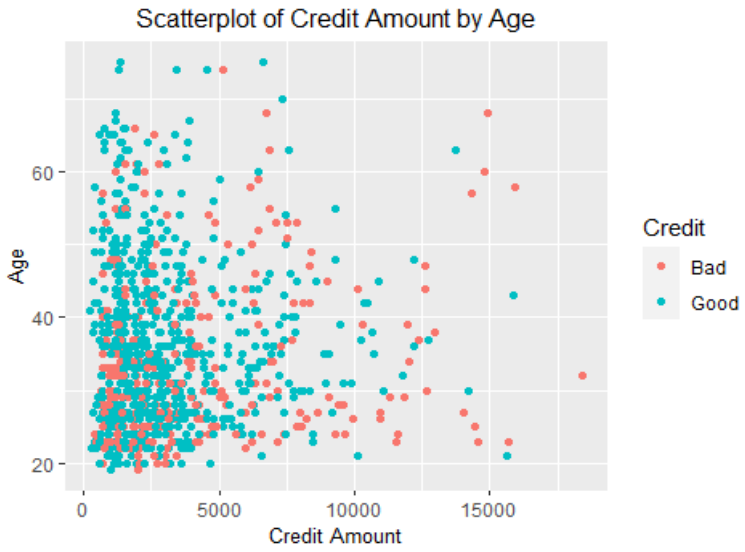

Figure 3.12

be evenly distributed, Figure 3.13 shows there are outliers for five of the loan types which includes new cars, used cars, radio/television, appliances, and business. This could indicate if the customers age and loan type are both included in the final model, it may exert undue influence. The scatterplot in Figure 3.14 shows the financial institution approves customers more

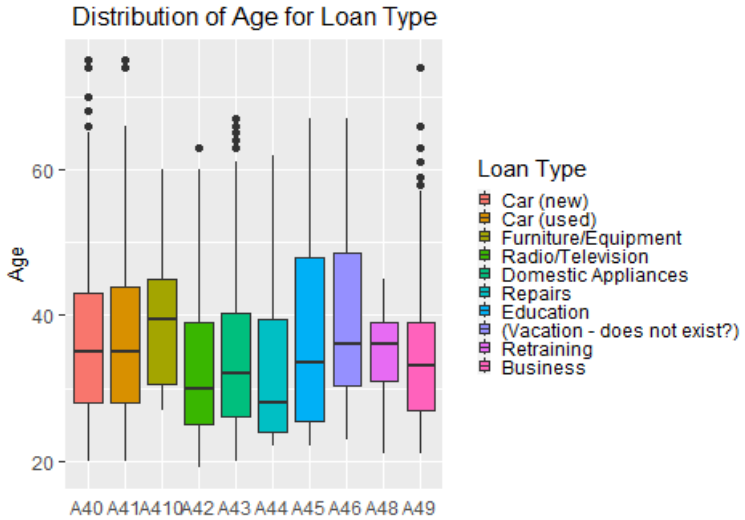

Figure 3.13

frequently when the loan term and credit amount are smaller. Alternatively, it also shows loans with longer terms and any credit amount are declined more frequently. This indicates loans with higher credit amounts and longer loan terms are considered riskier by the financial institution.

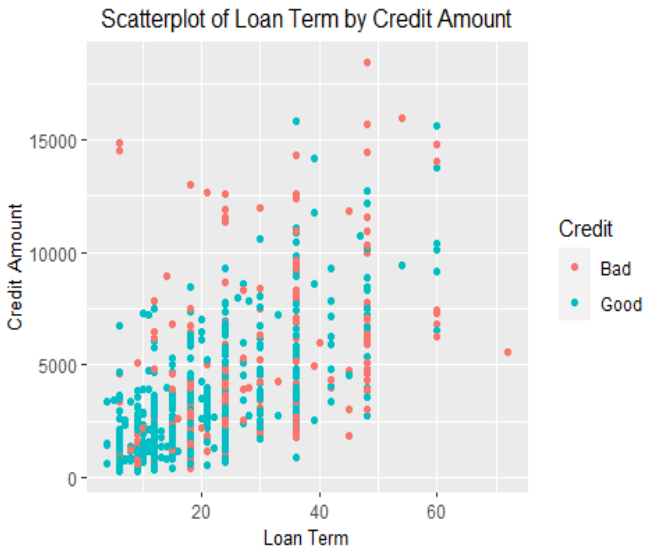

Figure 3.14

Feature Selection

Feature selection is the process of selecting a subset of the original features which exert the most influence on the target variable. With feature selection, the goal is to reduce the noise in the dataset and therefore have a model which generalizes well to unseen data. (Raschka & Mirjalili, 2017) For this problem feature importance was selected as the method used for feature selection. Feature importance is the process of assessing which features provide the most information gain to the model, this can be either through residual sum of squares or impurity measurement such as deviance. We used Random Forest and Gradient Boosting to assess the feature importance for the German Credit Data dataset. Figure 3.15 shows the results from the Random Forest model, with the left-hand side showing the mean decrease in accuracy and the right-hand side showing the mean decrease in deviance for each variable in the dataset.

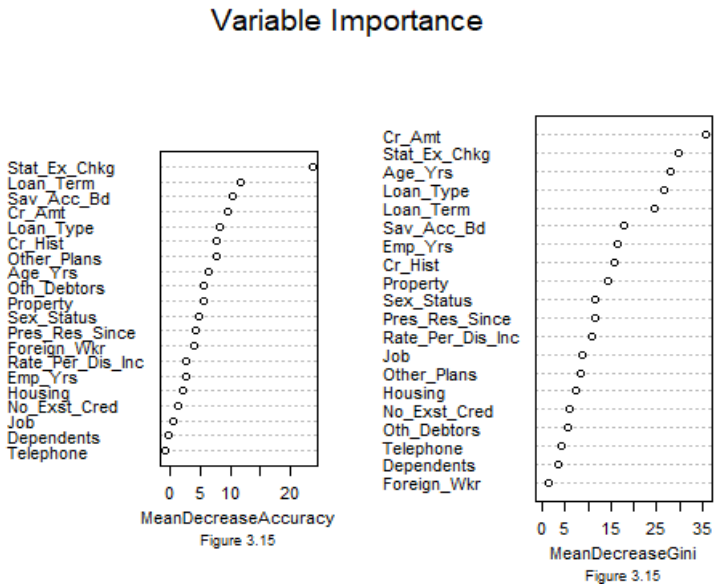

Figure 3.15
Figure 3.15

For feature selection we focused on the variables in the right-hand plot, as increasing the node purity of the model will lead to better classification results. We then performed gradient boosting on the dataset to compare with the results from Random Forest and find the best combination of features. Figure 3.16 shows the results of gradient boosting, when compared the

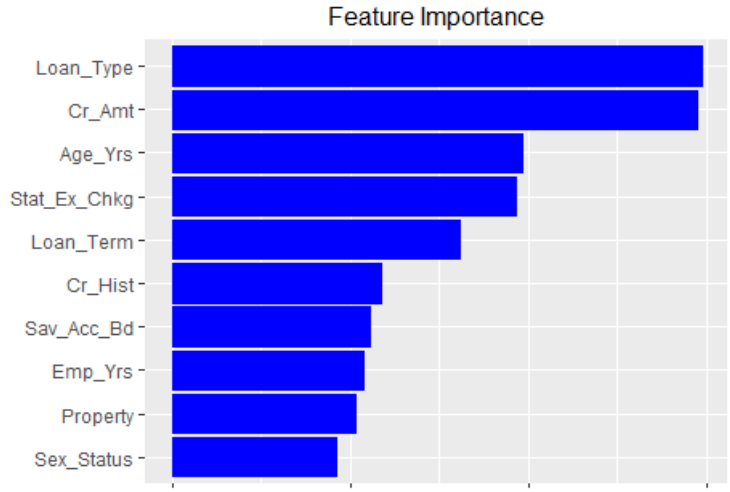

Figure 3.16

top ten results of the two models were the same. To reduce the dimensionality of the models, instead of including all twenty variables, the top ten features will be considered when building the models.

IV. Modeling

Random Forests

Random Forests are an ensemble method which creates a forest of decision trees and is used to address the potential for decision trees to pick up variance in the dataset. We build the Random Forest by taking a bootstrap sample from the dataset, growing a decision tree by using a subset of the predictors until a k number of trees are reached, and majority vote from among the assembled trees is used to assign the class. By selecting a random subset from the dataset, Random Forest helps with the tendency of models to overfit since the variance of the predictors will balance over the model. Random Forest also addresses the top-down greedy approach of decision trees, which does not look ahead when building the model and only considers the current split. Since Random Forests takes only a p subset of predictors, this allows for features to be considered which exert a smaller influence when building the tree. This not only creates a robust model which captures these less influential predictors, it also allows for a model which is not correlated since the strongest predictor is not used as the base for all models. (James, Witten, Hastie, & Tibshirani, 2013)
The randomForest() function was used to build the Random Forest model, the target variable, the top nine predictor variables and the training subset were passed into the Random Forest function. The results of the Random Forest model were plotted to determine what the best number of trees

```
    rf_df =
randomForest(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Emp_Yrs+Prope
rty+Sav_Acc_Bd, data=credit_df[train,], mtry=5, ntree=500)
```

were based on the classification error rate. Figure 4.1 shows the out-of-bag error rate for the Random Forest model. The out-of-bag error rate is calculated by taking the out-of-box observation which is not part of the specified Random Forest tree and predicting the response for the said observation. (James, Witten, Hastie, & Tibshirani, 2013) The out-of-box classification error rate provides an accurate estimate for how the model will perform on unseen data. For the Random

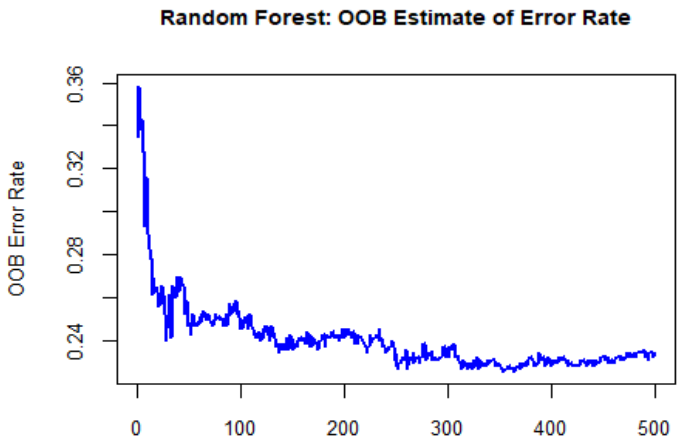

Figure 4.1

Forest model, the error rate drops significantly as the number of trees increases initially and then levels out at two hundred and fifty trees with a classification error rate of approximately 23%. This indicates the model is more accurate as the number of trees grows larger.

We implemented a grid search cross-validation approach to fine tune the Random Forest model. Grid search is a fine-tuning technique, where an exhaustive search of all the hyperparameters is performed to find the optimal combination of hyperparameters based on the best accuracy score. (Raschka & Mirjalili, 2017) The cv.rforest() function was used to evaluate which hyperparameters best improve the performance of the Random Forest model. The average classification rate for each model assessed were calculated and returned for the unique set of hyperparameters. Figure 4.2 shows the results from the cross-validation model, with Area Under the ROC Curve (AUC ROC) as

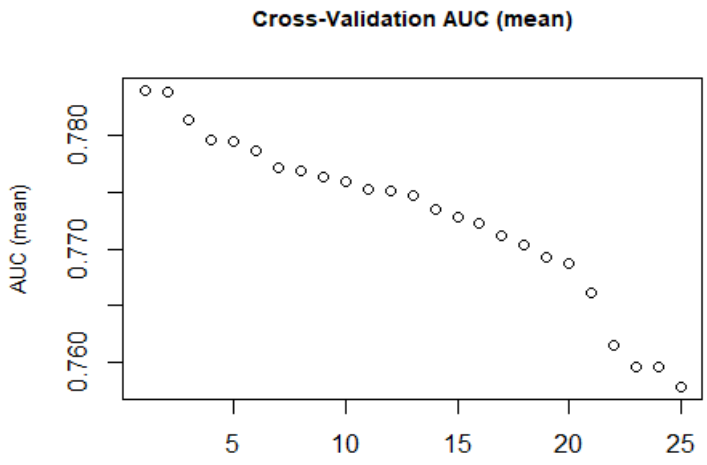

Figure 4.2

the measurement for the results of the cross-validated models. The AUC for the models starts at 78% and steadily decreases to 75%, this indicates the first five models were the most accurate of all models evaluated. The top five models with the AUC score and corresponding hyperparameters

AUC(mean)	std	min	max	mtry	num.trees	sample.fraction
0.7839293	0.03932700	0.7160494	0.8588972	1	500	1
0.7837426	0.04013488	0.7061267	0.8587211	2	500	1
0.7813471	0.04152141	0.6928310	0.8641840	5	250	1
0.7795313	0.04314482	0.6861920	0.8843168	3	250	1
0.7794016	0.03260767	0.7185802	0.8402746	4	500	1

Table 4.1

are shown in Table 4.1. All five models have two hundred and fifty or more trees, which indicates when more trees are grown this will increase the accuracy of the model. Also of note is the number of predictor variables considered at each split, which ranges between one and five. This indicates the number of predictors to consider at each split should be a range of numbers between these two values and both the number of trees and the number of variables should be taken into account when building the final model.

Support Vector Machines

Support Vector Machines (SVM) start with a decision boundary which is used to separate the dimensional space into regions based on a hyperplane. The optimal decision boundary is determined by finding the key support vectors which minimize the distance between the decision boundary and these observations, creating the maximal margin of the decision boundary. (James, Witten, Hastie, & Tibshirani, 2013) SVMs allow for more flexibility when dealing with data which is nonlinear or noisy, by allowing for observations to violate the margin without a larger penalty. This softening of the maximal margin lets a certain number of misclassifications occur, while still capturing sufficient support vectors to correctly classify other observations. To increase or decrease the maximal margin width, the c hyperparameter in the SVM model can be adjusted based on the linearity of the data and be tuned through cross-validation. Along with the maximal margin, kernels can be used to increase the feature space of the SVM model by using the inner products of the observations. (James, Witten, Hastie, & Tibshirani, 2013) These inner products can then be transformed using polynomial or radial computations to extend the inner products into a new dimensional feature space. A limiting factor can be placed on the kernel with the γ hyperparameter, which will limit or increase the influence of the training observations. The SVM model was created using the svm() function from the e1071 library and we passed in the nine predictor variables, a cost parameter, and the linear kernel function. The SVM model performs well, with an

```
svm_fit =
svm(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Emp_Yrs+Property+Sav_Ac
c_Bd, data=credit_df[train, ], kernel="linear", cost=0.1)
```

```
      train
 truth Good Bad
  Good 467  26
   Bad 152  55
```

accuracy of 74.57% on the training data. However, when viewing the confusion matrix, we can see the model misclassifies 152 or 21.7% of the customers. As identified earlier the risk of a financial institution misclassifying a customer as good when they are bad, incurs a stiffer penalty since the risk associated with this misclassification results in a potential loss for the financial institution. Grid search cross-validation approach was then used to assess which hyperparameters best improve the performance of the SVM model. The cross-validation was done through the tune() function in the e1071 library, where a list of the hyperparameters we wanted to search were passed into the

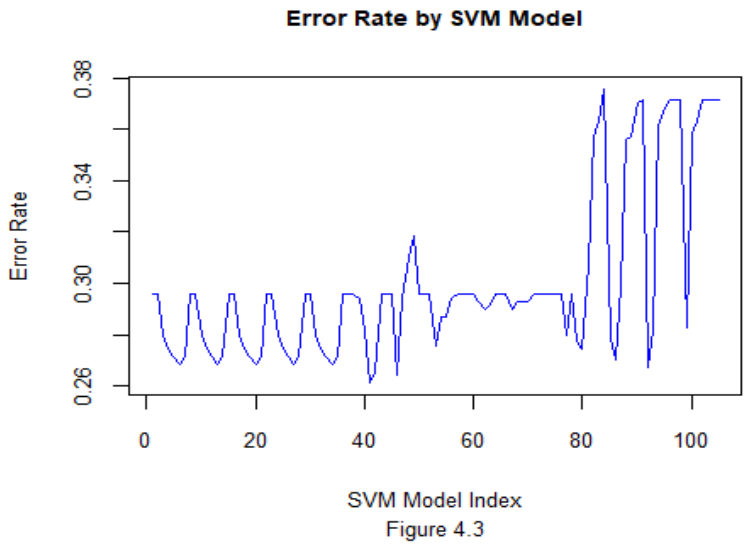

Figure 4.3

function. For the SVM model, using the tune() function we searched by the kernel type, c, and γ variables. Figure 4.3 displays the error rate of the models generated from the grid search cross-validation. The graph shows a 0.12 difference between the lowest and highest error rates, and the best performing model is the forty-first model which has the lowest error rate. The hyperparameters from this model indicate the radial kernel with a c value of ten, produce the best SVM model.

V. Evaluation

The performance of a classification model can be measured through several different methods, including accuracy, misclassification error rate and the AUC ROC curve.

Random Forests

Based on the results from the cross-validation performed on the Random Forest model, Table 4.1 shows the for the top five trees the number of underlying trees ranges from two hundred and fifty to five hundred and the number of predictors range from one to five for each node split. Four trees were built based on the results from the cross-validation model, two of the trees were built using the number of underlying trees as five hundred with the number of predictors considered as two and four. The other two models were built with the number of underlying trees set to two hundred and fifty and the number of predictors as two and four. The Area Under the ROC Curve (AUC ROC) is a way to display the true positive rate and false positive rate of a classification model. In the graph, a true positive rate of one would fall into the top left corner and would represent a perfect classifier. On the other hand, the false positive rate would be in the bottom right corner and represents a model which misclassifies all observations. The gray line crossing the center of the graph is indicates the probability of randomly guessing the class of an observation, where the model would perform no better than a random coin toss. From the results shown in Figure 5.1 for the training data, the performance of the models appears very similar with the indication the models correctly classify an observation approximately 75% of the time.

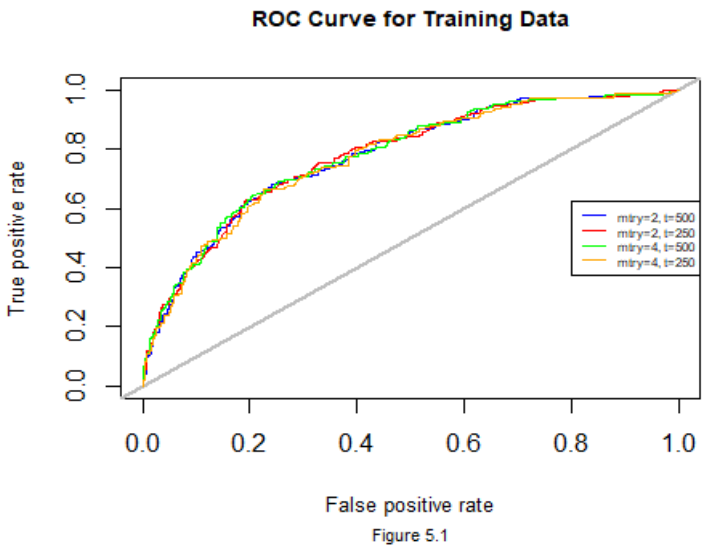

Figure 5.1

When the confusion matrices for the models are compared, we can see the two trees which performed the best are the tree with five hundred underlying trees and two variables at each split. Accuracy is the percentage of correctly classified observations out of the total number of

```
                     Number of trees: 500
 No. of variables tried at each split: 2

          OOB estimate of  error rate: 24%
 Confusion matrix:
    Good Bad  class.error
Good 451  42   0.0851927
 Bad 126  81   0.6086957
```

```
(451 + 81) / 700
```

```
 [1] 0.76
```

observations in the dataset. For the training data we see the accuracy for this tree is 76% and the percentage of incorrectly classified observations is 24%. We can also see the sensitivity for the model is 91.48%, this indicates the model correctly identifies a good customer among all true good customers nine times out of ten. However, when looking at the precision we can see the model

```
                     Number of trees: 250
 No. of variables tried at each split: 4

         OOB estimate of  error rate: 23.86%
 Confusion matrix:
    Good Bad  class.error
Good 443  50   0.1014199
 Bad 117  90   0.5652174
```

```
(443 + 90) / 700
```

```
## [1] 0.7614286
```

incorrectly identifies a bad customer as good 21.84% of the time. The other tree which performed well was the tree with two hundred and fifty underlying trees and four variables considered at each node split. Again, the overall accuracy of the model is high at 76.14% and we again correctly identify a good customer approximately 89.86% of the time for all good customers. This model also appears to suffer from the same problem as our first model, where the model precision is 20.89% and we identify a bad customer as good two out of ten times.

The models were then run using our test dataset to determine how well they performed on unseen data. Figure 5.2 shows the results of these models, and we can see the Random Forest with five hundred underlying trees and two predictor variables at each split performs the best. Given the

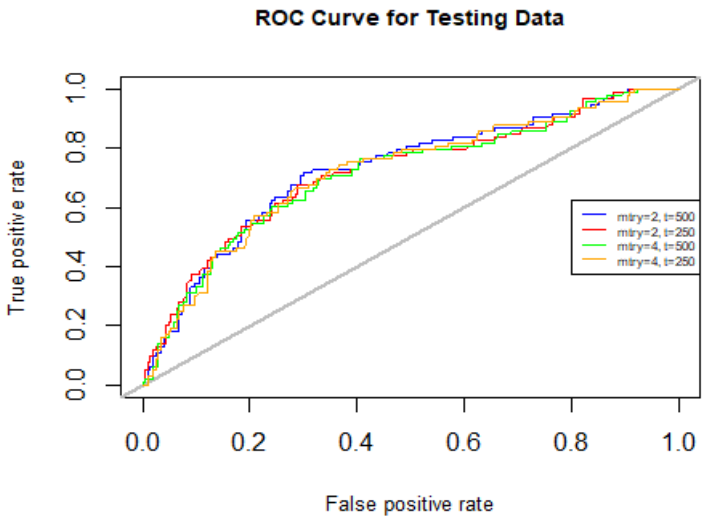

Figure 5.2

results from training data, we anticipated this result. When we look at the confusion matrix for this model, the accuracy is 73%, which is lower than the training dataset, but is still higher than random

```
                     Number of trees: 500
 No. of variables tried at each split: 2

         OOB estimate of  error rate: 26.67%
 Confusion matrix:
    Good Bad  class.error
Good 180  27   0.1304348
 Bad  53  40   0.5698925

(180 + 40) / 300

 [1] 0.7333333
```

guessing. When we look at the sensitivity it is lower at 86.96%, yet still indicates we are correctly identifying these customers almost nine times out of ten. The precision is higher as well, with the model incorrectly identifying these bad customers 22.75% of the time. This leads us to consider a model which may not have as high an accuracy percentage but will perform better at identifying these bad customers. The tree with five hundred underlying trees and four predictor variables has a

```
                     Number of trees: 500
 No. of variables tried at each split: 4

         OOB estimate of  error rate: 27%
 Confusion matrix:
    Good Bad  class.error
Good 176  31   0.1497585
 Bad  50  43   0.5376344
```

```
(176 + 43) / 300
```

```
 [1] 0.73
```

lower accuracy of 73% but does a slightly better identifying bad customer with a precision score of 22.12%.

Support Vector Machines

The results from the cross-validation of the SVM model indicated the radial kernel and $c = 10$, hyperparameters yield the best model. Based on these hyperparameters, an additional four models were created with varying values for the c hyperparameter to determine which value produced the best model. When we plot the AUC ROC curve for the five models created for the training dataset in

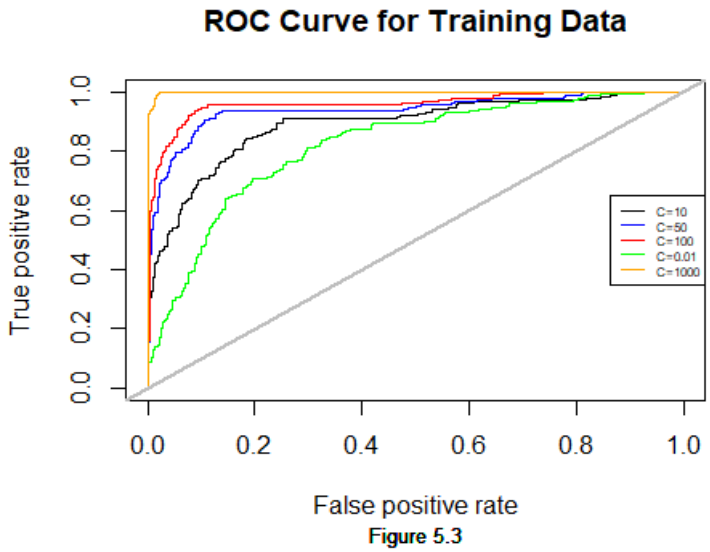

Figure 5.3

Figure 5.3 we can see the model with $c = 1000$ performs the best with an approximate accuracy of 98%. The confusion matrix for this model confirms the accuracy of this model is 98.29%. When we

```
       predict
 truth Good Bad
  Good  486   7
   Bad    5 202
```

```
predict_mostflex_train <- predict(svm_mostflex_fit_train, credit_df[train, ], type="class")
acc_mostflex_train <- mean(predict_mostflex_train==credit_df[train, "Credit"]) * 100
acc_mostflex_train
```

```
[1] 98.28571
```

look at the sensitivity for this model, it correctly identifies 98.58% of all good customers and similarly for the precision it incorrectly identifies bad customers as good 1.02% of the time. The model is then tested on the test dataset to evaluate how well it performs. Figure 5.4 shows the results of the five models, and the models with $c = 1000$ and $c = 100$ show the best results for the SVM models. When we view the confusion matrix for $c = 100$, we can see the accuracy of the

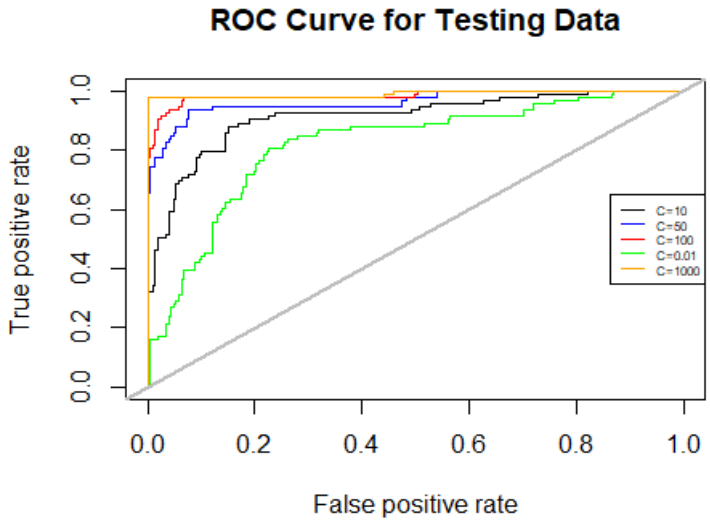

Figure 5.4

model is 94%, and the sensitivity is 98.55% which indicates we correctly identify good customers almost ten out of ten times. The precision score is also low at 6.85%, however it does suggest the

```
      predict
 truth Good Bad
  Good  204   3
   Bad   15  78

predict_moreflex <- predict(svm_moreflex_fit, credit_df[-train, ], type="class")
acc_moreflex <- mean(predict_moreflex==credit_df[-train, "Credit"]) * 100
acc_moreflex

 [1] 94
```

model would potentially inaccurately indicate a customer is good when they are not one out of ten times. The confusion matrix for $c = 1000$ shows this model performs much better, with an accuracy score of 99% and sensitivity score of 100%. This shows our model correctly identifies all good

```
      predict
 truth Good Bad
  Good  207   0
   Bad    3  90

predict_mostflex <- predict(svm_mostflex_fit, credit_df[-train, ], type="class")
acc_mostflex <- mean(predict_mostflex==credit_df[-train, "Credit"]) * 100
acc_mostflex

[1] 99
```

customers and we incorrectly identify bad customers less than one out of ten times with a precision score of 1.43%. When we compare the weighted results of the best Random Forests and SVM models, the results show the Random Forest model does far worse. With a new precision score

```
Confusion matrix:
     Good Bad
Good 176  31
 Bad 250  43
```

of 58.68%, this shows the Random Forest model incorrectly identifies bad customers as good five out of ten times. On the other hand, the results from the SVM weighted confusion matrix are far

```
      predict
truth Good Bad
 Good  207   0
  Bad   15  90
```

superior. The precision score is 6.76%, which shows we are misclassifying bad customers as good less than one out of ten times.

VI. Conclusion

The machine learning methods proposed in this paper offer promising results for correctly classifying credit extension to customers. We selected the classification methods of Random Forest and Support Vector Machine and created a training subset with seventy percent of the observations. A grid-search approach was selected as the cross-validation technique, which was chosen in part due to the fact Random Forest is inherently a form of cross-validation. The performance of both models was evaluated with the remaining thirty percent of the dataset through confusion matrices, accuracy and AUC ROC Curves. While the Random Forest classification method performed well, the Support Vector Machine model was superior in both accuracy and precision. The model we recommend is the Support Vector Machine with a radial kernel and c=1000, which provides the best results both for accuracy at 99% and weighted precision of 6.76%. Accuracy is an important indicator of the overall performance of the model, but for this recommendation we chose the model with the lowest weighted precision as this will mitigate the risk associated with extending to credit customers who are not credit worthy.

The weighted precision for the Support Vector Machine was calculated after the model was trained and validated, however further research should be conducted to determine if there are other methods which can be used to weight the model prior to calculating the accuracy. Special consideration should be given to a Cost-Sensitive Machine Learning methods such as resampling, algorithms or ensembles. These methods use a cost matrix similar to the one we used to calculate the weighted precision to create machine learning models. Consideration should also be given to other classification methods including Logistic Regression and Neural Networks. Overall the performance of the Support Vector Machine outperformed expectations and we consider this a good recommendation for the problem put forth.

References

James, G., Witten, D., Hastie, T., & Tibshirani. (2013). *An Introduction to Statistical Learning.* New York: Springer.

Krishnamurthi, M. (2007). Improving Credit Card Operations with Data Mining Techniques. *Journal of International Technology and Information Management*.

Raschka, S., & Mirjalili, V. (2017). *Python Machine Learning.* Birmingham: Packt Publishing Ltd.

Scott III, R. H. (June 2007). Credit Card Use and Abuse: A Veblenian Analysis . *JOURNAL OF ECONOMIC ISSUES*, 567*574.

Zurada, J., Kunene, N., & Guan, J. (2014). The Classification Performance of Multiple Methods and Datasets: Cases from the Loan Credit Scoring Domain. *Journal of International Technology & Information Management*, 57-82.

Appendix

Data Load

```
library(ggplot2)
library(tidyverse)
library(randomForest)
library(gbm)
library(radiant)
library(ROCR)
```

```
credit_df <- read.csv("C:\\Users\\rewar\\OneDrive\\Documents\\DATA 621\\Final Project\\german_credit.csv", header=TRUE)

Credit = as.factor(ifelse(credit_df$V21==1, "Good", "Bad"))
credit_df$V21 <- NULL
credit_df <- data.frame(credit_df, Credit)
colnames(credit_df) <- c("Index", "Stat_Ex_Chkg", "Loan_Term",
                         "Cr_Hist", "Loan_Type", "Cr_Amt", "Sav_Acc_Bd",
                         "Emp_Yrs", "Rate_Per_Dis_Inc", "Sex_Status",
                         "Oth_Debtors", "Pres_Res_Since", "Property",
                         "Age_Yrs", "Other_Plans", "Housing",
                         "No_Exst_Cred", "Job", "Dependents", "Telephone",
                         "Foreign_Wkr", "Credit")
```

Figure 3.1

```
ggplot(data = credit_df) +
  geom_bar(mapping = aes(x=Credit, fill=Credit)) +
  labs(title="Distribution of Customer Credit", caption= "Figure 3.1", x="Customer Credit", y="Count") +
  theme(plot.title = element_text(hjust=0.5)) +
  theme(plot.title = element_text(hjust=0.5)) +
  theme(plot.title = element_text(size=12)) +
  theme(axis.title = element_text(size=10)) +
  theme(plot.caption = element_text(hjust=0.5))
```

Figure 3.2

```
ggplot(data = credit_df) +
  geom_bar(mapping = aes(x = Stat_Ex_Chkg, fill = Stat_Ex_Chkg)) +
  labs(title="Distribution of Status of Existing Checking Acct", caption = "Figure 3.2", x="Status of Existing Checking Acct", y="Count") +
  theme(plot.title = element_text(hjust=0.5)) +
  theme(plot.title = element_text(size=10)) +
  theme(axis.title = element_text(size=9)) +
  scale_fill_discrete(name = "Existing Checking Account", labels = c("< 0 DM", "0 <= ... < 200 DM", ">= 200 DM", "No Checking Account")) +
  theme(legend.background = element_rect(size=0.5)) +
  theme(legend.key.size = unit(0.2, 'cm')) +
  theme(axis.ticks.x=element_blank()) +
  theme(plot.caption = element_text(hjust=0.5))
```

Figure 3.3

```
ggplot(data = credit_df) +
  geom_bar(mapping = aes(x = Stat_Ex_Chkg, fill = Credit)) +
  labs(title="Distribution of Status of Existing Checking Acct", caption = "Figure 3.3", x="Status of Existing Checking Acct", y="Count") +
  theme(plot.title = element_text(hjust=0.5)) +
  theme(plot.title = element_text(size=12)) +
  theme(axis.title = element_text(size=10)) +
  theme(plot.caption = element_text(hjust=0.5))
```

Figure 3.4

```
ggplot(data = credit_df) +
  geom_bar(mapping = aes(x=Loan_Type, fill=Loan_Type)) +
  labs(title="Distribution of Loan Type", caption = "Figure 3.4", x="Loan Type", y="Count") +
  theme(plot.title = element_text(hjust=0.5)) +
  theme(plot.title = element_text(size=12)) +
  theme(axis.title = element_text(size=10)) +
  scale_fill_discrete(name = "Loan Type", labels = c("Car (new)", "Car (used)",
"Furniture/Equipment", "Radio/Television", "Domestic Appliances", "Repairs", "Education",
"(Vacation - does not exist?)", "Retraining", "Business", "Others")) +
  theme(legend.background = element_rect(size=0.5)) +
  theme(legend.key.size = unit(0.2, 'cm')) +
  theme(axis.ticks.x=element_blank()) +
  theme(plot.caption = element_text(hjust=0.5))
```

Figure 3.5

```
ggplot(data = credit_df) +
  geom_bar(mapping = aes(x=Loan_Type, fill=Credit)) +
  labs(title="Distribution of Good/Bad Customer By Loan Type", caption = "Figure 3.5", x="Loan
Type", y="Count") +
  theme(plot.title = element_text(hjust=0.5)) +
  theme(plot.title = element_text(size=12)) +
  theme(axis.title = element_text(size=10)) +
  theme(plot.caption = element_text(hjust=0.5))
```

Figure 3.6

```
ggplot(data = credit_df) +
  geom_point(mapping = aes(x=Loan_Type, y=Cr_Amt, color=Loan_Type)) +
  labs(title="Distribution of Loan Type by Credit Amount", caption = "Figure 3.6", x="Loan
Type", y="Credit Amount") +
  theme(plot.title = element_text(hjust=0.5)) +
  theme(plot.title = element_text(size=12)) +
  theme(axis.title = element_text(size=10)) +
  scale_colour_discrete(name = "Loan Type", labels = c("Car (new)", "Car (used)",
"Furniture/Equipment", "Radio/Television", "Domestic Appliances", "Repairs", "Education",
"(Vacation - does not exist?)", "Retraining", "Business", "Others")) +
  theme(legend.background = element_rect(size=0.5)) +
  theme(legend.key.size = unit(0.2, 'cm')) +
  theme(axis.ticks.x=element_blank()) +
  theme(plot.caption = element_text(hjust=0.5))
```

Figure 3.7

```
ggplot(data = credit_df) +
  geom_bar(mapping = aes(x=Cr_Hist, fill=Cr_Hist)) +
  labs(title="Distribution of Credit History", caption = "Figure 3.7", x="Credit History",
y="Count") +
  theme(plot.title = element_text(hjust=0.5)) +
  theme(plot.title = element_text(size=12)) +
  theme(axis.title = element_text(size=10)) +
  scale_fill_discrete(name = "Credit History", labels = c("No Cr Taken / All Paid Duly", "All
Paid at Bank Duly", "Exist Cr Paid til Now", "Delay Pay Off in Past", "Critical Acct")) +
  theme(legend.background = element_rect(size=0.5)) +
  theme(legend.key.size = unit(0.2, 'cm')) +
  theme(axis.ticks.x=element_blank()) +
  theme(plot.caption = element_text(hjust=0.5))
```

Figure 3.8

```
ggplot(data = credit_df) +
  geom_bar(mapping = aes(x=Cr_Hist, fill=Credit)) +
  labs(title="Distribution of Credit History by Good/Bad Customers", caption = "Figure 3.8",
x="Credit History", y="Count") +
  theme(plot.title = element_text(hjust=0.5)) +
  theme(plot.title = element_text(size=12)) +
  theme(axis.title = element_text(size=10)) +
  theme(plot.caption = element_text(hjust=0.5))
```

Figure 3.9

```
ggplot(credit_df, aes(Loan_Term, colour=Credit)) +
  geom_freqpoly(bins=20) +
  labs(title="Distribution of Loan Term", caption = "Figure 3.9", x="Loan Term", y="Count") +
  theme(plot.title = element_text(hjust=0.5)) +
  theme(plot.title = element_text(size=12)) +
  theme(axis.title = element_text(size=10)) +
  theme(plot.caption = element_text(hjust=0.5))
```

Figure 3.10

```
ggplot(credit_df, aes(Cr_Amt, colour=Credit)) +
  geom_freqpoly(bins=50) +
  labs(title="Distribution of Credit Amount", caption = "Figure 3.10", x="Credit Amount",
y="Count") +
  theme(plot.title = element_text(hjust=0.5)) +
  theme(plot.title = element_text(size=12)) +
  theme(axis.title = element_text(size=10)) +
  theme(plot.caption = element_text(hjust=0.5))
```

Figure 3.11

```
ggplot(credit_df, aes(Age_Yrs, colour=Credit)) +
  geom_freqpoly(bins=25) +
  labs(title="Distribution of Age", caption = "Figure 3.11", x="Age", y="Count") +
  theme(plot.title = element_text(hjust=0.5)) +
  theme(plot.title = element_text(size=12)) +
  theme(axis.title = element_text(size=10)) +
  theme(plot.caption = element_text(hjust=0.5))
```

Figure 3.12

```
ggplot(data = credit_df) +
  geom_point(mapping = aes(x=Cr_Amt, y=Age_Yrs, color=Credit)) +
  labs(title="Scatterplot of Credit Amount by Age", caption = "Figure 3.12", x="Credit
Amount", y="Age") +
  theme(plot.title = element_text(hjust=0.5)) +
  theme(plot.title = element_text(size=12)) +
  theme(axis.title = element_text(size=10)) +
  theme(plot.caption = element_text(hjust=0.5))
```

Figure 3.13

```
ggplot(data = credit_df, aes(x=Loan_Type, y=Age_Yrs)) +
  geom_boxplot(mapping = aes(fill=Loan_Type)) +
  labs(title="Distribution of Age for Loan Type", caption = "Figure 3.13", x="Loan Type",
y="Age") +
  theme(plot.title = element_text(hjust=0.5)) +
  theme(plot.title = element_text(size=12)) +
  theme(axis.title = element_text(size=10)) +
  scale_fill_discrete(name = "Loan Type", labels = c("Car (new)", "Car (used)",
"Furniture/Equipment", "Radio/Television", "Domestic Appliances", "Repairs", "Education",
"(Vacation - does not exist?)", "Retraining", "Business", "Others")) +
  theme(legend.background = element_rect(size=0.5)) +
  theme(legend.key.size = unit(0.2, 'cm')) +
  theme(axis.ticks.x=element_blank()) +
  theme(plot.caption = element_text(hjust=0.5))
```

Figure 3.14

```
ggplot(data = credit_df) +
  geom_point(mapping = aes(x=Loan_Term, y=Cr_Amt, color=Credit)) +
  labs(title="Scatterplot of Loan Term by Credit Amount", caption = "Figure 3.14", x="Loan
Term", y="Credit Amount") +
  theme(plot.title = element_text(hjust=0.5)) +
  theme(plot.title = element_text(size=12)) +
  theme(axis.title = element_text(size=10)) +
  theme(plot.caption = element_text(hjust=0.5))
```

Figure 3.15

```
cols <- c("Stat_Ex_Chkg", "Cr_Hist", "Loan_Type", "Sav_Acc_Bd", "Emp_Yrs", "Sex_Status",
"Oth_Debtors", "Property", "Other_Plans", "Housing", "Job", "Telephone", "Foreign_Wkr")
credit_df[cols] <- lapply(credit_df[cols], factor)
credit_df$Index <- NULL

train = sample(1:nrow(credit_df), 0.7 * nrow(credit_df))
credit_df_test = credit_df[-train,]
Credit_test = credit_df$Credit[-train]

rf_df = randomForest(Credit~., data=credit_df[train, ], importance=TRUE)
varImpPlot(rf_df, cex=0.7, main = "Variable Importance", sub = "Figure 3.15", cex.main=0.9, ce
x.sub=0.8)
```

Figure 3.16

```
set.seed(1)
credit_df$Credit = as.numeric(ifelse(credit_df$Credit=="Good", 0, 1))
boost_credit_df = gbm(Credit~., data=credit_df[train, ], distribution="bernoulli", n.trees=200
0, interaction.depth=4)
vip::vip(boost_credit_df, aesthetics=list(colour="blue", fill="blue")) +
  labs(title="Feature Importance", caption = "Figure 3.16") +
  theme(plot.title = element_text(hjust=0.5)) +
  theme(plot.title = element_text(size=12)) +
  theme(plot.caption = element_text(hjust=0.5)) +
  theme(plot.caption = element_text(size=10))
```

Random Forests

```
credit_df$Credit = as.factor(ifelse(credit_df$Credit==0, 1, 2))
train = sample(1:nrow(credit_df), 0.7 * nrow(credit_df))
credit_df_test = credit_df[-train,]
Credit_test = credit_df$Credit[-train]

set.seed(1)
rf_df = randomForest(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Emp_Yrs+Pr
operty+Sav_Acc_Bd, data=credit_df[train,], mtry=5, ntree=500)
ytest = predict(rf_df, credit_df[train, ], type="class")
table(truth=credit_df[train, "Credit"], Train=ytest)
```

Figure 4.1

```
plot(rf_df$err.rate[,1], type="l", main="Random Forest: OOB Estimate of Error Rate", sub="Figu
re 4.1", xlab="Number of Trees", ylab="OOB Error Rate", lwd=2, col="blue", cex.main=0.9, cex.l
ab=0.8, cex.axis=0.8, cex.sub=0.8)
```

Cross Validation for Random Forests

```
set.seed(1)
rand_for <- rforest(credit_df[train, ], "Credit", c("Stat_Ex_Chkg", "Cr_Amt", "Age_Yrs", "Loan
_Term", "Loan_Type", "Cr_Hist", "Emp_Yrs", "Property", "Sav_Acc_Bd"), type="classification", s
eed=1)
cv_rf <- cv.rforest(rand_for, repeats=5, mtry=1:5, num.trees=c(25, 50, 100, 250, 500), min.nod
e.size=1, sample.fraction=NA, trace=TRUE, seed=1234)
```

Figure 4.2

```
plot(cv_rf$`AUC (mean)`, main="Cross-Validation AUC (mean)", sub="Figure 4.2", xlab="Cross-Val
idation Model Index", ylab="AUC (mean)", cex.main=0.9, cex.lab=0.8, cex.sub=0.8)
```

Table 4.1

```
table_cv_rf <- as.data.frame(cv_rf)
head(table_cv_rf, 10)
```

Support Vector Machine

```
svm_fit =
svm(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Emp_Yrs+Property+Sav_Acc_Bd
, data=credit_df[train, ], kernel="linear", cost=0.1)
summary(svm_fit)
```

Cross Validation for Support Vector Machine

```
set.seed(1)
tune_lin_svm = tune(svm, Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Emp_Yr
s+Property+Sav_Acc_Bd,
                    data=credit_df[train, ],
                    ranges=list(cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100),
                                gamma=c(0.01, 0.25, 0.5, 1, 2),
                                kernel=c("linear", "radial", "polynomial")
                                ))
summary(tune_lin_svm)
```

Figure 4.13

```
plot(tune_lin_svm$performances[,4], type="l", col="blue", main="Error Rate by SVM Model", sub=
"Figure 4.3", xlab="SVM Model Index", ylab="Error Rate", lwd=1.5, cex.main=0.9, cex.sub=0.8, c
ex.axis=0.8, cex.lab=0.8)
```

Figure 5.1

```
rf_train_opt <-
randomForest(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Emp_Yrs+Property+S
av_Acc_Bd, data=credit_df[train, ], mtry=2, ntree=500)
rf_train_least <-
randomForest(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Emp_Yrs+Property+S
av_Acc_Bd, data=credit_df[train, ], mtry=2, ntree=250)
rf_train_flex <-
randomForest(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Emp_Yrs+Property+S
av_Acc_Bd,  data=credit_df[train, ], mtry=4, ntree=500)
rf_train_mid <-
randomForest(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Emp_Yrs+Property+S
av_Acc_Bd, data=credit_df[train, ], mtry=4, ntree=250)

yhat_rf_train_opt <- predict(rf_train_opt, data=credit_df[train, ], type="prob")[, 2]
pred_yhat_rf_opt <-prediction(yhat_rf_train_opt, credit_df[train, "Credit"])
perf_train_rf_opt <- performance(pred_yhat_rf_opt, "tpr", "fpr")
plot(perf_train_rf_opt, main="ROC Curve for Training Data", sub="Figure 5.1", xlab="False
positive rate", ylab="True positive rate", col="blue", cex.main=0.9, cex.lab=0.8, cex.sub=0.7,
cex.axis=0.7)
legend("right", legend=c("mtry=2, t=500", "mtry=2, t=250", "mtry=4, t=500", "mtry=4, t=250"),
lty=c("solid", "solid", "solid", "solid"), col=c("blue", "red", "green", "orange"), cex=0.5)
abline(a=0, b=1, lwd=2, col="gray")

yhat_rf_train_least <- predict(rf_train_least, data=credit_df[train, ], type="prob")[, 2]
pred_yhat_rf_least <- prediction(yhat_rf_train_least, credit_df[train, "Credit"])
perf_train_rf_least <- performance(pred_yhat_rf_least, "tpr", "fpr")
plot(perf_train_rf_least, add=TRUE, col="red")

yhat_rf_train_flex <- predict(rf_train_flex, data=credit_df[train, ], type="prob")[, 2]
pred_yhat_rf_flex <- prediction(yhat_rf_train_flex, credit_df[train, "Credit"])
perf_train_rf_flex <- performance(pred_yhat_rf_flex, "tpr", "fpr")
plot(perf_train_rf_flex, add=TRUE, col="green")

yhat_rf_train_mid <- predict(rf_train_mid, data=credit_df[train, ], type="prob")[, 2]
pred_yhat_rf_mid <- prediction(yhat_rf_train_mid, credit_df[train, "Credit"])
perf_train_rf_mid <- performance(pred_yhat_rf_mid, "tpr", "fpr")
plot(perf_train_rf_mid, add=TRUE, col="orange")
```

Random Forest Train Confusion Matrices

```
rf_train_opt

rf_train_mid
```

Figure 5.2

```
set.seed(1)
rf_test_opt <- randomForest(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Emp
_Yrs+Property+Sav_Acc_Bd, data=credit_df[-train, ], mtry=2, ntree=500)
rf_test_least <- randomForest(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+E
mp_Yrs+Property+Sav_Acc_Bd, data=credit_df[-train, ], mtry=2, ntree=250)
rf_test_flex <- randomForest(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Em
p_Yrs+Property+Sav_Acc_Bd,  data=credit_df[-train, ], mtry=4, ntree=500)
rf_test_mid <- randomForest(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Emp
_Yrs+Property+Sav_Acc_Bd, data=credit_df[-train, ], mtry=4, ntree=250)

yhat_rf_test_opt <- predict(rf_test_opt, data=credit_df[-train, ], type="prob")[, 2]
pred_test_rf_opt <-prediction(yhat_rf_test_opt, credit_df[-train, "Credit"])
perf_test_rf_opt <- performance(pred_test_rf_opt, "tpr", "fpr")
```

```
plot(perf_test_rf_opt, main="ROC Curve for Testing Data", sub="Figure 5.2", xlab="False positi
ve rate", ylab="True positive rate", col="blue", cex.main=0.9, cex.lab=0.8, cex.sub=0.7, cex.a
xis=0.7)
legend("right", legend=c("mtry=2, t=500", "mtry=2, t=250", "mtry=4, t=500", "mtry=4, t=250"),
lty=c("solid", "solid", "solid", "solid"), col=c("blue", "red", "green", "orange"), cex=0.5)
abline(a=0, b=1, lwd=2, col="gray")

yhat_rf_test_least <- predict(rf_test_least, data=credit_df[-train, ], type="prob")[, 2]
pred_test_rf_least <- prediction(yhat_rf_test_least, credit_df[-train, "Credit"])
perf_test_rf_least <- performance(pred_test_rf_least, "tpr", "fpr")
plot(perf_test_rf_least, add=TRUE, col="red")

yhat_rf_test_flex <- predict(rf_test_flex, data=credit_df[-train, ], type="prob")[, 2]
pred_test_rf_flex <- prediction(yhat_rf_test_flex, credit_df[-train, "Credit"])
perf_test_rf_flex <- performance(pred_test_rf_flex, "tpr", "fpr")
plot(perf_test_rf_flex, add=TRUE, col="green")

yhat_rf_test_mid <- predict(rf_test_mid, data=credit_df[-train, ], type="prob")[, 2]
pred_test_rf_mid <- prediction(yhat_rf_test_mid, credit_df[-train, "Credit"])
perf_test_rf_mid <- performance(pred_test_rf_mid, "tpr", "fpr")
plot(perf_test_rf_mid, add=TRUE, col="orange")
```

Random Forest Test Confusion Matrices

```
rf_test_opt

rf_test_flex
```

Figure 5.3

```
set.seed(1)
rocplot = function(pred, truth, ...){
  predob = prediction(pred, truth)
  perf = performance(predob, "tpr", "fpr")
  plot(perf, ...)
  title(sub="Figure 5.3", cex.main=0.9, cex.lab=0.8, cex.sub=0.8)
  legend("right", legend=c("C=10", "C=50", "C=100", "C=0.01", "C=1000"),
         cex=0.5, lty=c("solid", "solid", "solid", "solid", "solid"),
         col=c("black", "blue", "red", "green", "orange"))
  abline(a=0, b=1, lwd=2, col="gray")
}

svm_fit_train_opt = svm(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Emp_Yrs
+Property+Sav_Acc_Bd, data=credit_df[train, ], kernel="radial", cost=10, decision.values=T)
fitted = attributes(predict(svm_fit_train_opt, credit_df[train, ], decision.values=TRUE))$deci
sion.values

rocplot(fitted, credit_df[train, "Credit"], main="ROC Curve for Training Data", col="black")

svm_flex_fit_train = svm(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Emp_Yr
s+Property+Sav_Acc_Bd, data=credit_df[train, ], kernel="radial", cost=50, decision.values=TRUE
)
fitted = attributes(predict(svm_flex_fit_train, credit_df[train, ], decision.values=TRUE))$dec
ision.values

rocplot(fitted, credit_df[train, "Credit"], add=T, col="blue")

svm_moreflex_fit_train = svm(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Em
p_Yrs+Property+Sav_Acc_Bd, data=credit_df[train, ], kernel="radial", cost=100, decision.values
```

```
=TRUE)
fitted = attributes(predict(svm_moreflex_fit_train, credit_df[train, ], decision.values=TRUE))
$decision.values

rocplot(fitted, credit_df[train, "Credit"], add=T, col="red")

svm_lessflex_fit_train = svm(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Em
p_Yrs+Property+Sav_Acc_Bd, data=credit_df[train, ], kernel="radial", cost=0.01, decision.value
s=TRUE)
fitted = attributes(predict(svm_lessflex_fit_train, credit_df[train, ], decision.values=TRUE))
$decision.values

rocplot(fitted, credit_df[train, "Credit"], add=T, col="green")

svm_mostflex_fit_train = svm(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Em
p_Yrs+Property+Sav_Acc_Bd, data=credit_df[train, ], kernel="radial", cost=1000, decision.value
s=TRUE)
fitted = attributes(predict(svm_mostflex_fit_train, credit_df[train, ], decision.values=TRUE))
$decision.values

rocplot(fitted, credit_df[train, "Credit"], add=T, col="orange")
```

Support Vector Machine Confusion Matrices

Figure 5.4

```
rocplot = function(pred, truth, ...){
  predob = prediction(pred, truth)
  perf = performance(predob, "tpr", "fpr")
  plot(perf, ...)
  title(sub="Figure 5.4", cex.main=0.9, cex.lab=0.8, cex.sub=0.8)
  legend("right", legend=c("C=10", "C=50", "C=100", "C=0.01", "C=1000"),
         cex=0.5, lty=c("solid", "solid", "solid", "solid", "solid"),
         col=c("black", "blue", "red", "green", "orange"))
  abline(a=0, b=1, lwd=2, col="gray")
  }

svm_fit_opt =
svm(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Emp_Yrs+Property+Sav_Acc_Bd
, data=credit_df[-train, ], kernel="radial", cost=10, decision.values=T)
fitted = attributes(predict(svm_fit_opt, credit_df[-train, ],
decision.values=TRUE))$decision.values

rocplot(fitted, credit_df[-train, "Credit"], main="ROC Curve for Testing Data", col="black")

svm_flex_fit =
svm(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Emp_Yrs+Property+Sav_Acc_Bd
, data=credit_df[-train, ], kernel="radial", cost=50, decision.values=TRUE)
fitted = attributes(predict(svm_flex_fit, credit_df[-train, ],
decision.values=TRUE))$decision.values

rocplot(fitted, credit_df[-train, "Credit"], add=T, col="blue")

svm_moreflex_fit =
svm(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Emp_Yrs+Property+Sav_Acc_Bd
, data=credit_df[-train, ], kernel="radial", cost=100, decision.values=TRUE)
fitted = attributes(predict(svm_moreflex_fit, credit_df[-train, ],
decision.values=TRUE))$decision.values
```

```
rocplot(fitted, credit_df[-train, "Credit"], add=T, col="red")

svm_lessflex_fit =
svm(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Emp_Yrs+Property+Sav_Acc_Bd
, data=credit_df[-train, ], kernel="radial", cost=0.01, decision.values=TRUE)
fitted = attributes(predict(svm_lessflex_fit, credit_df[-train, ],
decision.values=TRUE))$decision.values

rocplot(fitted, credit_df[-train, "Credit"], add=T, col="green")

svm_mostflex_fit =
svm(Credit~Stat_Ex_Chkg+Cr_Amt+Age_Yrs+Loan_Term+Loan_Type+Cr_Hist+Emp_Yrs+Property+Sav_Acc_Bd
, data=credit_df[-train, ], kernel="radial", cost=1000, decision.values=TRUE)
fitted = attributes(predict(svm_mostflex_fit, credit_df[-train, ],
decision.values=TRUE))$decision.values

rocplot(fitted, credit_df[-train, "Credit"], add=T, col="orange")
```