Inference and Prediction on Credit Approval

Pragna Bhatt

Banks nowadays receive large numbers of credit card applications and often the decision on whether an application gets approved or not depends on a variety of factors such as income level, credit score, debt, etc. Making such decisions may prove to be a difficult, time-consuming task if it were not for statistical modeling. The aim of this analysis was to create such a statistical model that could firstly predict approval status based on information provided in the application and secondly make inferences about the effects of certain attributes of the applicant on approval status. The methods used to achieve this objective were logistic regression, classification tree, and random forest. The data for this analysis was sourced from the UCI Machine Learning Repository and contained 690 observations with each observation representing an individual applicant. Each observation contained information on whether the application was approved and on fifteen other attributes of the applicant. However, the names and values of these fifteen variables had been changed to meaningless symbols to protect the confidentiality of the data. In the analysis, the variables were simply referred to as V1 to V15. All of them were considered as potential predictors in the three models except for V14. V14 was a factor variable with 170 levels and without any further information on what it represented, it was difficult to extrapolate any meaningful information from it so it was dropped at the beginning of the analysis. Additionally, about 0.6% of the data had missing values therefore multiple imputation was used to fill in the missing values. Lastly, a test set was created at the beginning of the analysis to check and compare the prediction accuracy of the three models at the end.

The first model fitted in this analysis was the logistic regression model. The form for the logistic regression model was determined using Agresti's Purposeful Selection Method. First fourteen models were created that each used approval as the response variable and a sole predictor. From the fourteen models, the predictors that did not show evidence of being significant at P-value less than 0.2 were dropped including V1 and V4. The remaining variables were used to create an initial main-effects model and then backwards elimination was used to come up with a final model. The formula for the final model was approval ~ V5 + V6 + V9 + V11 + V13 + V15. However, when fitting this model on the imputed data sets, the software reported a problem with complete or quasi-complete separation in the data. Therefore, while the pooled estimated odds ratios are provided in the output, the likelihood ratio test was used to test for significance of the effects and 95% profile likelihood confidence intervals were constructed instead of using Wald intervals. The results of the likelihood ratio test showed that all the explanatory variables included in the model were significant having p-values less than 0.05. The p-value for V9 especially was reported by the software as very close to 0 (< 2.2e-16). According to the profile likelihood interval for V9, holding all other variables fixed, the estimated odds of approval for an applicant with V9 = t were between 21.2587 and 91.6619 times the estimated odds for an applicant with V9 = f. Therefore, V9 was evidently a strong predictor of approval status. For V11, holding all other variables fixed, a one unit increase in V11 had a multiplicative effect between 1.0932 and 1.386 on the odds of approval for an applicant. V6 on the other hand was a factor variable with 14 levels and the confidence intervals for those categories, provided in the appendix, contained one many times and tended to be very wide. This may be perhaps due to sparse data as many categories had relatively low counts. Similar problems can be seen in the confidence intervals for V13 and V5 which had an upper bound of infinity on the profile likelihood interval of one of its subcategories.

The second model fitted in this analysis was a classification tree. Using a complexity parameter of 0.03, the result was a tree that classified approval based solely on the value of V9. The tree predicted that applications with V9 = t would be approved while those with V9 = f would not be approved. The complexity parameter was also decreased to see if a more complex classification tree would give higher prediction accuracy, however the more complex tree led to overfitting and worse performance on predicting approval in the test set. The last model fitted in this analysis was a random forest. After fitting the model on the training data, a variable importance plot was created to show what the random forest considered to be the most important variables based on how much accuracy the model lost by excluding the variable. The top three variables in the plot in order of importance were V9, V15, and V11. These three variables were also among the most significant effects in the logistic regression model.

Lastly, when checking prediction accuracy of the models, the holdout sample created at the beginning of the analysis was used to see how often the models accurately predicted the actual response values. The prediction accuracy was 85.6%, 83.65%, and 86.06% for the logistic regression model, classification tree, and random forest, respectively. Therefore, the random forest did the best job in predicting the actual values with logistic regression coming close in second place. While the classification tree had the worst performance, it performed relatively well for how simple it was to implement and understand. For the logistic regression, two other indicators of predictive accuracy were the area under the ROC plot, 0.9469, and the correlation between the fitted and actual values, 0.8056.

To sum up, the objectives of this analyses were to accurately predict approval status and to make inferences on which attributes of the applicant impacted approval status the most. The former objective was best achieved by the implementation of the random forest which had a prediction accuracy of 86.06%. The latter was achieved by the logistic regression model which showed that V5, V6, V9, V11, V13, and V15 were among the most significant effects on approval status. In future studies, it may be helpful to consider interaction terms as they were not looked at in this analysis. Something else to consider may be dropping outliers in both categorical and numeric variables. The numeric variables tended to be strongly skewed to the right with some outliers to the far right. The categorical variables also often had subcategories with very low counts that had either 100% approval rate of 100% rejection rate. It may be useful to at least explore what the impact is of removing certain subcategories or grouping them together with other subcategories. Of course, one must be cautious here especially as there is no information on what the true values and variables are. A better solution here though may be to choose Bayesian modeling and modified types of likelihood-based analyses that Agresti talks about in Chapter 5. These methods tend to work better especially when we are getting infinite ML estimates as we are in our logistic regression model. Lastly, to improve prediction accuracy, one may always try out other binary classification methods, such as linear discriminant analysis, k-nearest neighbor classifier, support vector machines, etc.

Appendix

Structure of Data

```
## 'data.frame':    690 obs. of  16 variables:
##  $ V1      : Factor w/ 2 levels "a","b": 2 1 1 2 2 2 2 1 2 2 ...
##  $ V2      : num  30.8 58.7 24.5 27.8 20.2 ...
##  $ V3      : num  0 4.46 0.5 1.54 5.62 ...
##  $ V4      : Factor w/ 3 levels "l","u","y": 2 2 2 2 2 2 2 2 3 3 ...
##  $ V5      : Factor w/ 3 levels "g","gg","p": 1 1 1 1 1 1 1 1 3 3 ...
##  $ V6      : Factor w/ 14 levels "aa","c","cc",..: 13 11 11 13 13 10 12 3
9 13 ...
##  $ V7      : Factor w/ 9 levels "bb","dd","ff",..: 8 4 4 8 8 8 4 8 4 8 ...
##  $ V8      : num  1.25 3.04 1.5 3.75 1.71 ...
##  $ V9      : Factor w/ 2 levels "f","t": 2 2 2 2 2 2 2 2 2 2 ...
##  $ V10     : Factor w/ 2 levels "f","t": 2 2 1 2 1 1 1 1 1 1 ...
##  $ V11     : int  1 6 0 5 0 0 0 0 0 0 ...
##  $ V12     : Factor w/ 2 levels "f","t": 1 1 1 2 1 2 2 1 1 2 ...
##  $ V13     : Factor w/ 3 levels "g","p","s": 1 1 1 1 3 1 1 1 1 1 ...
##  $ V14     : Factor w/ 170 levels "00000","00017",..: 69 12 97 32 38 116 5
5 24 63 16 ...
##  $ V15     : int  0 560 824 3 0 0 31285 1349 314 1442 ...
##  $ approval: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

```
# Pooled estimates for each model (5 models build on 5 imputed datasets) into
 a single set of estimates and standard errors

summary(combine, conf.int = TRUE, exponentiate=TRUE)
```

```
##          term     estimate    std.error   statistic       df      p.value
## 1  (Intercept) 3.314369e-02 5.547950e-01 -6.14083148 458.8241 1.779837e-09
## 2         V5gg 1.346462e+07 8.827439e+02  0.01859608 458.9665 9.851714e-01
## 3          V5p 5.515360e-01 3.907092e-01 -1.52299499 417.4702 1.285171e-01
## 4          V6c 1.422936e+00 5.610392e-01  0.62869410 455.6860 5.298643e-01
## 5         V6cc 5.707512e+00 8.698426e-01  2.00241199 458.6232 4.582868e-02
## 6          V6d 2.665772e+00 9.221911e-01  1.06322190 458.8655 2.882405e-01
## 7          V6e 3.235806e+00 9.075964e-01  1.29383282 458.9665 1.963739e-01
## 8         V6ff 1.654515e-01 1.090740e+00 -1.64940947 250.7448 1.003160e-01
## 9          V6i 6.007362e-01 7.143362e-01 -0.71338869 458.9665 4.759678e-01
## 10         V6j 5.061850e-01 1.349761e+00 -0.50442504 458.9665 6.142048e-01
## 11         V6k 7.660923e-01 7.122533e-01 -0.37409812 458.9665 7.085041e-01
## 12         V6m 7.698230e-01 7.539622e-01 -0.34695991 458.9665 7.287805e-01
## 13         V6q 2.667038e+00 6.378882e-01  1.53783753 458.9013 1.247775e-01
## 14         V6r 3.445525e-04 2.817603e+02 -0.02829804 458.9665 9.774367e-01
## 15         V6w 2.356665e+00 6.439447e-01  1.33124411 458.9665 1.837696e-01
## 16         V6x 8.045539e+00 9.178106e-01  2.27183893 458.7951 2.355842e-02
## 17         V9t 4.187866e+01 3.705859e-01 10.07803162 458.7943 0.000000e+00
## 18         V11 1.224076e+00 6.074528e-02  3.32843220 458.6798 9.437031e-04
```

```
## 19        V13p 3.928837e+01 1.088150e+00  3.37354900 294.5199 8.414190e-04
## 20        V13s 8.316127e-01 5.321917e-01 -0.34646995 458.8548 7.291484e-01
## 21         V15 1.000627e+00 2.394160e-04  2.61936031 458.8189 9.101264e-03
##              2.5 %        97.5 %
## 1   1.114062e-02  9.860349e-02
## 2   0.000000e+00           Inf
## 3   2.558790e-01  1.188812e+00
## 4   4.724502e-01  4.285627e+00
## 5   1.032943e+00  3.153678e+01
## 6   4.352879e-01  1.632561e+01
## 7   5.437414e-01  1.925629e+01
## 8   1.930759e-02  1.417795e+00
## 9   1.475815e-01  2.445319e+00
## 10  3.567410e-02  7.182331e+00
## 11  1.889761e-01  3.105669e+00
## 12  1.749525e-01  3.387361e+00
## 13  7.614159e-01  9.341925e+00
## 14 1.170489e-244 1.014247e+237
## 15  6.648473e-01  8.353605e+00
## 16  1.325095e+00  4.884986e+01
## 17  2.021693e+01  8.675019e+01
## 18  1.086339e+00  1.379278e+00
## 19  4.615374e+00  3.344423e+02
## 20  2.922273e-01  2.366582e+00
## 21  1.000157e+00  1.001098e+00
```

```
# Showing results of fitted logistic regression model on first imputed datase
t, likelihood Ratio Test, and Profile Likelihood Intervals

fit <- fit2$analyses[[1]]
summary(fit)

##
## Call:
## glm(formula = approval ~ V5 + V6 + V9 + V11 + V13 + V15, family = binomial
(link = "logit"))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8117  -0.3298  -0.1448   0.4044   3.0139
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.392e+00  5.542e-01  -6.119  9.4e-10 ***
## V5gg         1.639e+01  8.827e+02   0.019 0.985182
## V5p         -6.748e-01  3.816e-01  -1.768 0.076979 .
## V6c          2.933e-01  5.583e-01   0.525 0.599335
## V6cc         1.761e+00  8.745e-01   2.014 0.044052 *
## V6d          9.938e-01  9.221e-01   1.078 0.281140
```

```
## V6e          1.171e+00  9.116e-01   1.285 0.198831
## V6ff        -1.313e+00  9.143e-01  -1.436 0.151110
## V6i         -5.094e-01  7.147e-01  -0.713 0.475968
## V6j         -6.915e-01  1.353e+00  -0.511 0.609199
## V6k         -2.659e-01  7.125e-01  -0.373 0.708995
## V6m         -2.674e-01  7.553e-01  -0.354 0.723283
## V6q          9.692e-01  6.384e-01   1.518 0.128941
## V6r         -7.836e+00  2.716e+02  -0.029 0.976980
## V6w          8.634e-01  6.434e-01   1.342 0.179614
## V6x          2.063e+00  9.055e-01   2.279 0.022682 *
## V9t          3.745e+00  3.705e-01  10.107  < 2e-16 ***
## V11          1.997e-01  6.055e-02   3.299 0.000971 ***
## V13p         3.980e+00  1.058e+00   3.761 0.000169 ***
## V13s        -1.978e-01  5.282e-01  -0.375 0.708012
## V15          6.196e-04  2.369e-04   2.616 0.008904 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 661.67  on 481  degrees of freedom
## Residual deviance: 279.02  on 461  degrees of freedom
## AIC: 321.02
##
## Number of Fisher Scoring iterations: 13

library(car)

## Loading required package: carData

##
## Attaching package: 'car'

## The following objects are masked from 'package:faraway':
##
##     logit, vif

Anova(fit)

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Analysis of Deviance Table (Type II tests)
##
## Response: approval
##     LR Chisq Df Pr(>Chisq)
```

```
## V5     6.835  2  0.0327889 *
## V6    25.730 13  0.0184763 *
## V9   160.991  1  < 2.2e-16 ***
## V11   14.991  1  0.0001080 ***
## V13   13.300  2  0.0012941 **
## V15   12.599  1  0.0003861 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
library(profileModel)
```

```
## Warning: package 'profileModel' was built under R version 4.0.5
```

```
exp(confintModel(fit, objective = "ordinaryDeviance", method = "zoom", endpoi
nt.tolerance = 1e-08))
```

```
## Preliminary iteration ..................... Done
##
## Profiling for parameter (Intercept) ... Done
## Profiling for parameter V5gg ... Done
## Profiling for parameter V5p ... Done
## Profiling for parameter V6c ... Done
## Profiling for parameter V6cc ... Done
## Profiling for parameter V6d ... Done
## Profiling for parameter V6e ... Done
## Profiling for parameter V6ff ... Done
## Profiling for parameter V6i ... Done
## Profiling for parameter V6j ... Done
## Profiling for parameter V6k ... Done
## Profiling for parameter V6m ... Done
## Profiling for parameter V6q ... Done
## Profiling for parameter V6r ... Done
## Profiling for parameter V6w ... Done
## Profiling for parameter V6x ... Done
## Profiling for parameter V9t ... Done
## Profiling for parameter V11 ... Done
## Profiling for parameter V13p ... Done
## Profiling for parameter V13s ... Done
## Profiling for parameter V15 ... Done
## Zooming for parameter (Intercept) ...
## Zooming for parameter V5gg ...
## Zooming for parameter V5p ...
## Zooming for parameter V6c ...
## Zooming for parameter V6cc ...
## Zooming for parameter V6d ...
## Zooming for parameter V6e ...
## Zooming for parameter V6ff ...
## Zooming for parameter V6i ...
## Zooming for parameter V6j ...
## Zooming for parameter V6k ...
## Zooming for parameter V6m ...
```

```
## Zooming for parameter V6q ...
## Zooming for parameter V6r ...
## Zooming for parameter V6w ...
## Zooming for parameter V6x ...
## Zooming for parameter V9t ...
## Zooming for parameter V11 ...
## Zooming for parameter V13p ...
## Zooming for parameter V13s ...
## Zooming for parameter V15 ...

##                     Lower        Upper
## (Intercept) 1.091085e-02   0.09624379
## V5gg        7.484312e-01          Inf
## V5p         2.381744e-01   1.06924069
## V6c         4.475209e-01   4.03348039
## V6cc        1.111356e+00  33.34030472
## V6d         4.296172e-01  15.76343518
## V6e         5.587517e-01  19.41644186
## V6ff        3.691956e-02   1.45098414
## V6i         1.434609e-01   2.41013940
## V6j         2.983182e-02   7.10201543
## V6k         1.867237e-01   3.10072996
## V6m         1.718329e-01   3.38336924
## V6q         7.676685e-01   9.48846459
## V6r         1.208750e-10 403.92469345
## V6w         6.789289e-01   8.52741114
## V6x         1.435356e+00  48.55438081
## V9t         2.125873e+01  91.66196070
## V11         1.093271e+00   1.38606544
## V13p        6.904681e+00 522.63996934
## V13s        2.858393e-01   2.29858546
## V15         1.000202e+00   1.00108793
## attr(,"fitted object")
## fit
```

```
# Predictions - I only kept one table here since all five models gave same re
sults.

imptest <- mice(test, nnet.MaxNWts = 3000, seed=500)

##
##  iter imp variable
##   1   1  V1  V2*  V4  V5  V6  V7
##   1   2  V1  V2  V4  V5  V6  V7
##   1   3  V1  V2*  V4  V5  V6  V7
##   1   4  V1  V2  V4  V5  V6  V7
##   1   5  V1  V2*  V4  V5  V6  V7
```

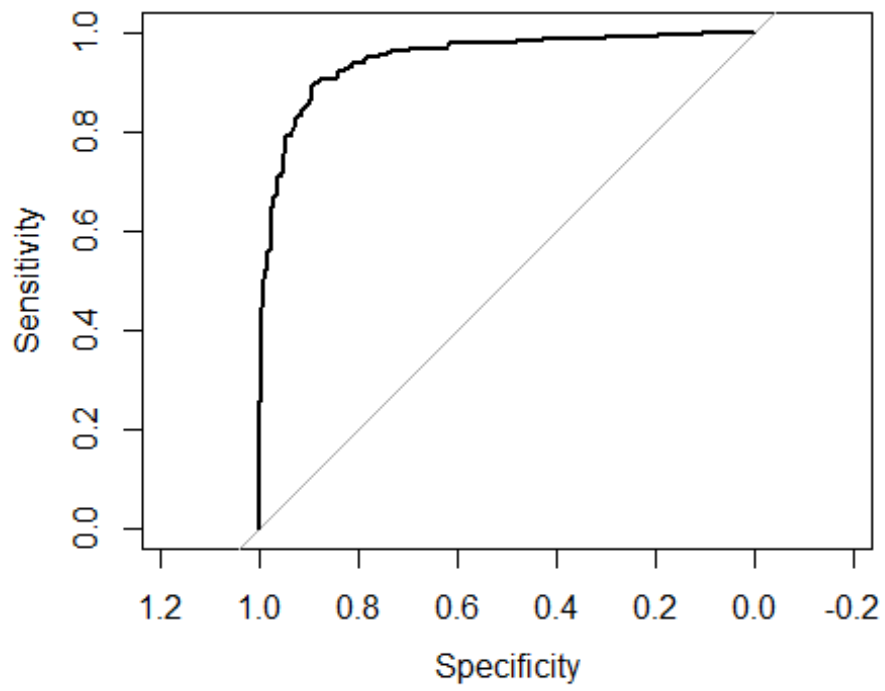
```
##   2   1  V1  V2  V4  V5  V6  V7
##   2   2  V1  V2  V4  V5  V6  V7
##   2   3  V1  V2*  V4  V5  V6  V7
##   2   4  V1  V2*  V4  V5  V6  V7
##   2   5  V1  V2*  V4  V5  V6  V7
##   3   1  V1  V2*  V4  V5  V6  V7
##   3   2  V1  V2*  V4  V5  V6  V7
##   3   3  V1  V2  V4  V5  V6  V7
##   3   4  V1  V2*  V4  V5  V6  V7
##   3   5  V1  V2*  V4  V5  V6  V7
##   4   1  V1  V2*  V4  V5  V6  V7
##   4   2  V1  V2*  V4  V5  V6  V7
##   4   3  V1  V2*  V4  V5  V6  V7
##   4   4  V1  V2  V4  V5  V6  V7
##   4   5  V1  V2*  V4  V5  V6  V7
##   5   1  V1  V2  V4  V5  V6  V7
##   5   2  V1  V2  V4  V5  V6  V7
##   5   3  V1  V2*  V4  V5  V6  V7
##   5   4  V1  V2*  V4  V5  V6  V7
##   5   5  V1  V2*  V4  V5  V6  V7

## Warning: Number of logged events: 167
```

```
completedtest <- complete(imptest, 1)
completedtest2 <- complete(imptest, 2)
completedtest3 <- complete(imptest, 3)
completedtest4 <- complete(imptest, 4)
completedtest5 <- complete(imptest, 5)

thresh  <- 0.5      # threshold for categorizing predicted probabilities
pred <- predict(fit2$analyses[[1]], newdata=completedtest, type="response")
predFac <- cut(pred, breaks=c(-Inf, thresh, Inf), labels=c("0", "1"))
cTab    <- table(completedtest$approval, predFac, dnn=c("actual", "predicted"
))
addmargins(cTab)
```

```
##       predicted
## actual   0   1 Sum
##    0    92  22 114
##    1     8  86  94
##    Sum 100 108 208
```

```
# AUC plot
library(pROC)

## Warning: package 'pROC' was built under R version 4.0.4

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'
```

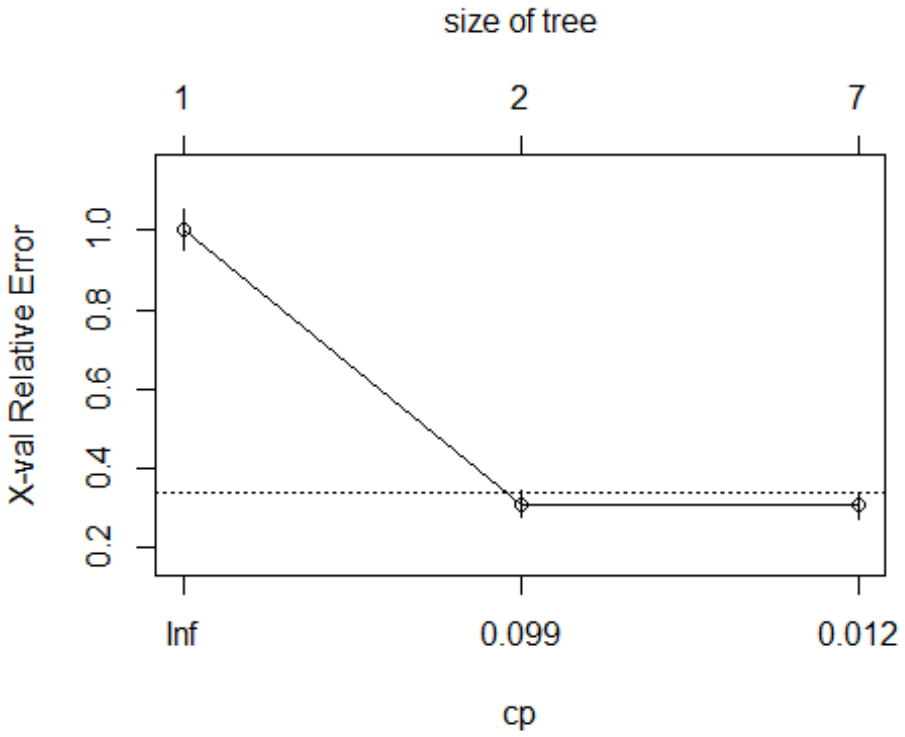
```
## The following object is masked from 'package:colorspace':
##
##     coords

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

rocplot <- roc(approval ~ fitted(fit2$analyses[[1]]), data = completedtrain)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

plot.roc(rocplot, legacy.axis=TRUE)
```



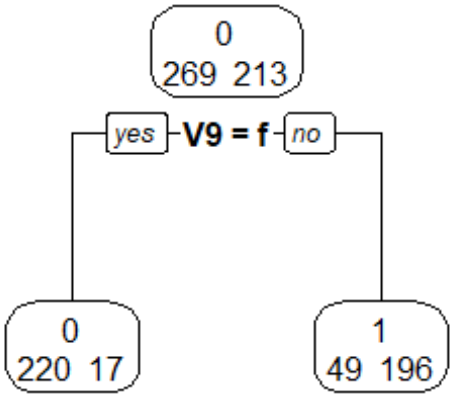

```
auc(rocplot)

## Area under the curve: 0.9469

cor(as.numeric(completedtrain$approval), fitted(fit2$analyses[[1]]))

## [1] 0.8055886
```

```
### NEXT MODEL: Classification Tree and Random Forest

# Similar results on all 5 imputated data sets so I've only included one here
```

```
.

completedtrain$approval <- factor(completedtrain$approval)
library(rpart)

## Warning: package 'rpart' was built under R version 4.0.5

##
## Attaching package: 'rpart'

## The following object is masked from 'package:faraway':
##
##     solder

fit <- rpart(approval ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 + V1
1 + V12 + V13 + V15, method = "class", data = completedtrain)
plotcp(fit)
```




```
p.fit <- prune(fit, cp = 0.03)
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 4.0.5

rpart.plot(p.fit, extra=1, digits=4, box.palette = 0)
```




```
pred <- predict(p.fit, newdata = completedtest, type = "class")
pred <- as.numeric(pred)
pred <- ifelse(pred == 2, 1, 0)
predFac <- cut(pred, breaks=c(-Inf, thresh, Inf), labels=c("0", "1"))
cTab <- table(completedtest$approval, predFac, dnn=c("actual", "predicted"))
addmargins(cTab)
```

```
##       predicted
## actual   0   1 Sum
##    0    86  28 114
##    1     6  88  94
##    Sum  92 116 208
```

```
# Random Forest

set.seed(12345)
library(randomForest)
```

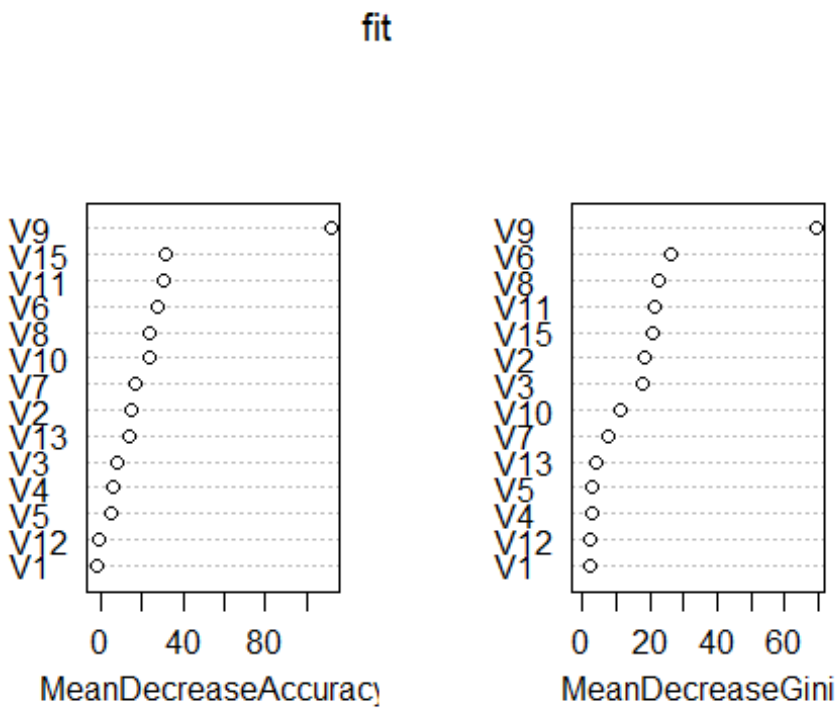
```
## Warning: package 'randomForest' was built under R version 4.0.5
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin

fit <- randomForest(approval ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V
10 + V11 + V12 + V13 + V15, data = completedtrain, ntree=2000, importance = T
RUE)
varImpPlot(fit)
```




```
pred <- predict(fit, newdata = completedtest, type = "class")
pred <- as.numeric(pred)
pred <- ifelse(pred == 2, 1, 0)
predFac <- cut(pred, breaks=c(-Inf, thresh, Inf), labels=c("0", "1"))
cTab <- table(completedtest$approval, predFac, dnn=c("actual", "predicted"))
addmargins(cTab)
```

```
##       predicted
## actual   0   1 Sum
##    0    93  21 114
##    1     8  86  94
##    Sum 101 107 208
```

References

Agresti, A. (2018). An Introduction to Categorical Data Analysis, 3rd ed. Wiley.

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.