Model Comparison in Credit Card Fraud Detection

Chris A. Shurtleff
Department of Business Analytics
University of Tennessee
Knoxville, TN 37996
cshurtle@vols.utk.edu

Abstract

Credit card fraud cost the U.S. economy $16 Billion in 2016. Financial institutions have very real incentives to create effective algorithms for detecting fraud in real time. Pursuant to this, I tested four common data mining algorithms (decision tree, random forest, logistic regression, and a neural network) on a data set consisting of 284,801 credit card transactions of which 492 were identified as fraudulent in an effort to identify which method produced the best results for detecting fraud. I used the area under the receiver operating curve as a measure of algorithm effectiveness and found that logistic regression outperformed, on that metric, the other three algorithms tested.

1 Background

Fraud cost the U.S. economy $16 Billion in 2016 [1]. This figure represents an increase of $2 million from 2015. Currently, all major credit card providers and most banks use data mining methods to identify fraud and minimize financial liability for fraudulent claims. There has been much research on various strategies to identify fraud in real time using machine learning techniques and automatically take steps to prevent financial accountability. Kim et al [2] proposed SVM ensembles for this purpose, several authors [3] [4] have utilized neural networks. Maes et al [5] compared neural networks and Bayesian networks. Considerably less research has been effected using modern deep learning techniques on this problem.

This project aims to compare the performance of four common data mining algorithms: decision trees, random forest, logistic regression, and a neural network. These four algorithms were chosen as the most likely to effectively identify fraudulent purchases on credit card data, as well as for ease of use and implementation. The performance of these models will be compared using the area under the receiver operating curve (AUROC) and area under the specificity curve (AUSEC).

1.1 Data

The data used for this project was uploaded to Kaggle by Pozzolo et al [6]. The data consist of 284,801 credit card transaction in Continental Europe, of which 492 were fraudulent. The fraudulent cases represent 0.17% of the available data, which presents unique problems that will be covered more thoroughly in a later section. The monetary value of the transactions was €25,162,590. The data consist of 28 features that have been anonymized by the application of principal component analysis (PCA), and two unmodified features. The unmodified features are time (from an unspecified beginning) and the purchase amount in Euros. The anonymization measures preclude many interesting forms of analysis, and contribute to the decision to test four conventional algorithms rather than more modern analyses based on deep learning (i.e. a recursive neural network).

2 Methods

This project concentrates on the comparative performance of four common data mining algorithms: decision trees, random forest, logistic regression, and a neural network. The models were trained and tuned on 66% of the data, and tested on the final 33% The performance of these models was evaluated using two primary measures: Area Under the Receiver Operating Curve (AUROC) and Area Under the Specificity Curve (AUSEC). These four algorithms and two measures were implemented in R, a statistical software chosen for its ease of use, that has several drawbacks that will be discussed in future sections. All code used in preparing these analyses is available in the appendix.

2.1 Data Preparation

The data, as delivered, was fundamentally clean. There was no need to eliminate observations due to missing data, as most preprocessing had been completed by Pozzolo et al, including principal component analysis, which ensured that the data would be sparse. Upon receipt of the data, I split the data into a training (33%), validation (33%) and test (33%) set. In splitting the data, I ensured that all sets received an approximately equal number of fraudulent and non-fraudulent observations. This was necessitated by the fact that the fraudulent observations represent less than 0.2% of the data, and a random distribution between the three data sets could very easily have resulted in one or more being entirely free from fraudulent observations entirely. To aid in analysis, I removed the response variables from the datasets as binary factors.

2.2 Decision Trees

The decision tree algorithm chosen for this project is an R implementation designed by Brieman et al [7], called 'rpart'. The rpart algorithm classifies members of the population by splitting on the independent variables, recursively splitting each sub-population until a stopping criterion is reached. In this case, the minimum number of observations required to perform a split was 20, the minimum bucket size was 7, the minimum acceptable decrease of lack of fit (the complexity parameter, or cp) was set at 0.01, the number of competitor splits retained in the output was 4 (this represents 'runner up' values to the chosen split), the maximum number of surrogate splits was capped at 5, the number of cross-validations was set at 10, the surrogate selection was determined by the total number of correct classifications, and the maximum depth of any node was set at 30.

2.3 Random Forest

The random forest algorithm used was an R implementation developed by Brieman [8]. The random forest algorithm grows a series of decision trees based on bootstrap samples of variables taken at every split of every node on every tree. The number of trees grown in this ensemble was set to 500, the average of which became the reported prediction. The bootstrap sample size taken at every split was set at 0.632*number of observations. The minimum number of nodes was set at 1, while there was no arbitrarily defined maximum number of nodes. Further details of the model can be found in the appendix.

2.4 Logistic Regression

The logistic regression formulation used in this experiment was based primarily on work done by Dobson [9] and implemented in R by the core R team as 'glm'. This formulation uses a generalized linear function designed to follow a binomial distribution. Further details can be found in the appendix.

2.5 Neural Network

The neural network algorithm used in this study is a single-hidden-layer model optimized for binomial output. In preparation for using this model, all of the training, validation, and test data were scaled to approximately the same range of values. This was done by scaling the train data to between 0 and 1, and scaling the validation and test data by the same multipliers. The neural network algorithm used is based on a design proposed and implemented in R by Ripley [10] as 'nnet'. The

fundamental design involves weights and activation functions between entry, hidden, and output layers. The weights and sensitivities of the activation functions are determined using a standard backpropogation algorithm. The initial random weights for the model were set at 0.5, due to the scaled nature of the data. The maximum number of iterations was capped at 10,000, the neural network size was set at 30, and the neural network decay parameter was set at 0.01. Further information about the neural network parameters can be found in the appendix.

2.6 Model Comparison

The AUROC and AUSEC calculations used in this study were developed by Ballings and Van den Poel [11] as a part of the R package 'AUC'. The receiver operating curve plots the true positive rate ($\frac{TP}{P_O}$, where TP is the number of true positive predictions, and P_O is the observed number of positives) against the false positive rate ($\frac{FP}{N_O}$, where FP is the number of false positive predictions, and N_O is the number of observed negative values). Calculating the area under the receiver operating curve (Equation 1)

$$Equation\ 1 : AUROC = \int_0^1 \frac{TP}{P_O} d\frac{FP}{N_O}$$

grants a single numeric statistic that is directly comparable between two models. The sensitivity curve plots the true positive rate against 1-the true positive rate, the area underneath which again provides a single number statistic directly comparable between models (Equation 2).

$$Equation\ 2 : AUSEC = \int_0^1 \frac{TP}{P_O} dt$$

Accuracy is another potential measure that directly compares the number of accurately classified observations by dividing the true positive (TP) and true negative (TN) rates by the total number of observations (Equation 3).

$$Equation\ 3 : Accuracy = \frac{TP + TN}{Total\ Observations}$$

Future sections will discuss the inadequacies of each measure.

3 Results

After following the methodology outlined in the previous section results for each model were obtained. These are discussed on a per-model basis in the following section.

3.1 Decision Tree

The decision tree methodology returned a model that scored an AUROC of 0.893, and an AUSEC of 0.892. As will become evident in this section, the AUROC and AUSEC values are very similar, for reasons that will be explored further in the discussion section. According to the very rough estimates of the academic points system [12], this is the equivalent represents 'good' accuracy. This interpretation is problematic, as will be discussed in greater detail later, but in many cases an AUROC of 0.893 is considered an adequate separator. Figure 1 shows a representation of the decision tree created by this model.

According to this tree, variables V17 and V14 are the most discriminatory, explaining between them the largest split of the dataset. Variables V7, V10, V16, V8, V26, and V27 also all play a significant role. Unusually, the actual value of a transaction seems to have a limited role in discerning fraud, according to this model. Granted, one or more of the anonymized features may connect historic spending trends of a given account with the current transaction price, which would include transaction price as an influential variable. While this model may not be the most effective at separating the two classes, it gives useful information about which variables are more important discriminators and which are less important.

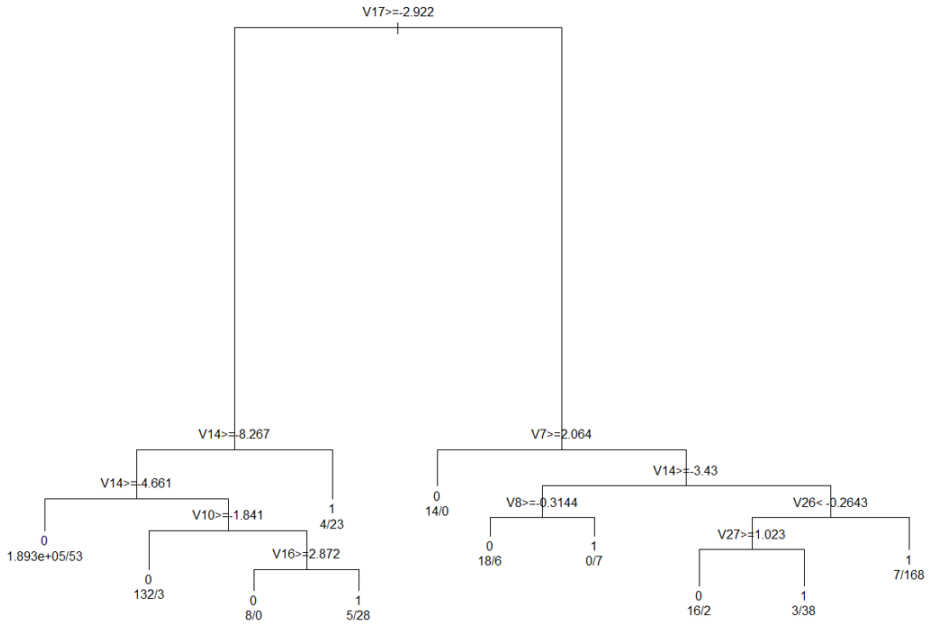

Figure 1: Decision tree that separates fraud from non-fraud. Variable V17 seems to be the most discriminatory, combined with V14 (the two of which can separate the vast majority of non-fraud cases) V7, V10, V16, V8, V26, and V27 also all play discriminatory roles in this tree. It would appear that the vast majority of features in this dataset are less discerning than these major classes.

3.2 Random Forest

The random forest algorithm is based on an ensemble of decision trees. It tends to predict better than individual trees due to the power of averaging many individual tree models. It is less prone to overfitting, and tends to perform well against other models. This experiment further supported this idea with an AUROC of 0.968 and an AUSEC of 0.967. Per the academic points system, this level of discrimination is 'Excellent'. To further display the effectiveness of the random forest model, Table 1 and Table 2 two show confusion matrices for a probability cutoff of 0.5 and 0.1, respectively.

Table 1: Random Forest Confusion Matrix at 0.5 cutoff

0.5 cutoff	P_P	N_P
P_O	131	33
N_O	12	94760

Table 2: Random Forest Confusion Matrix at 0.1 cutoff

0.1 cutoff	P_P	N_P
P_O	139	25
N_O	54	94718

Confusion matrices enumerate the true positives (where the prediction is positive, P_P and the observed value is positive P_O), the true negatives (where the prediction is negative N_P) and the observed value is negative N_O), the false positives (P_P and N_O), and the false negatives (N_P) and P_O). In this case, a 'positive' is fraudulent and a 'negative' is not-fraudulent, so false positives are

predicted cases of fraud that are not actually fraudulent, and false negatives are predicted as non-fraudulent but are in fact fraudulent. Table 1 shows the probability cutoff of 0.5, where 33 cases of fraud are not captured by the model, and only 12 cases of non-fraud are labeled as fraud. Table 2 shows the probability cutoff of 0.1, which increases the number of false positive predictions of the test set to 54, while reducing the false negative predictions from 33 to 25. The AUROC values capture the entirety of the tradeoff between the false positive and false negative rates, which is w hy the AUROC is directly comparable between models, there is no need to find the optimal cutoff values that are comparable between two models that have different distributions of prediction values.

3.3 Logistic Regression

Logistic regression often performs very well in tasks requiring binary separation. Additionally, logistic regression tends to be relatively easy to implement, is relatively simple to interpret, and increases understanding of which classifiers might be more important. In this case, the AUROC was calculated at 0.981, and the AUSEC was calculated at 0.980. Additionally, logistic regression models are often evaluated with p-values for each variable. This p-value suggest which variables are truly discriminatory and which ones are more likely to be noise. In this case, V4, V5, V8, V10, V13, V14, V20, V21, V22, V27, and V28 all were considered significant at the alpha value of 0.05.

The statistical significance, however, is only part of the measure of which features are most effective at discriminating between fraudulent and non-fraudulent purchases. The scale of each of the coefficients determined by the logistic regression function is related to how discriminatory each variable is, dependent on each variable having the same or nearly the same scale, which in the case of this data is approximately true. With that criteria, the top five discriminating variables according to the logistic regression model are V10, V27, V4, V22, and V14. This is broadly in line with which variables were picked out by the decision tree model above as the most discriminatory.

3.4 Neural Network

Neural networks are notoriously hard to implement and computationally expensive to train. The model used here was exceptionally simple (only a single hidden layer) but took by far the longest amount of time to run of all the models evaluated in this paper. Neural networks have more tunable hyperparameters than any of the other algorithms, which makes them both more flexible but also more prone to overfitting. The AUROC of the neural network evaluated to 0.968, and the AUSEC also evaluated to 0.968. This score is in line with the logistic regression and random forest models, although the computational cost was relatively staggering. This model, of all the models, is the most likely to be capable of improving, as will be covered in more depth in the discussion section.

3.5 ROC

The overall best model as measured by the AUROC is the logistic regression model, at 0.981. Figure 2 plots the ROC for each type of model. The logistic regression curve is by far the best performer in general, although it underperforms the other two models at the crucial point where all three models have a rapidly increasing true positive rate relative to the false positive rate. Depending on which scenario is deemed preferable (more false positives or more false negatives) it might be advantageous to try one of the other models.

From figure 2 it is obvious that the logistic regression, neural network, and random forest models all approximate each other very well. This is what would be expected if the models efficiently identified all or most of the discriminatory information in the dataset. While I suspect there is room for improvement in these models (as will be explored in the discussion section), there might be a cap on how much discriminatory information is truly available in this dataset. In that case, more sophisticated techniques might be used to more effectively identify feature classes in the raw data.

4 Discussion

In the previous section, logistic regression was tentatively declared the best model based on the AUROC. This declaration is somewhat problematic, in that the advantage enjoyed by logistic regression is slight, the AUROC is an imperfect measure of model discrimination, and there might be better

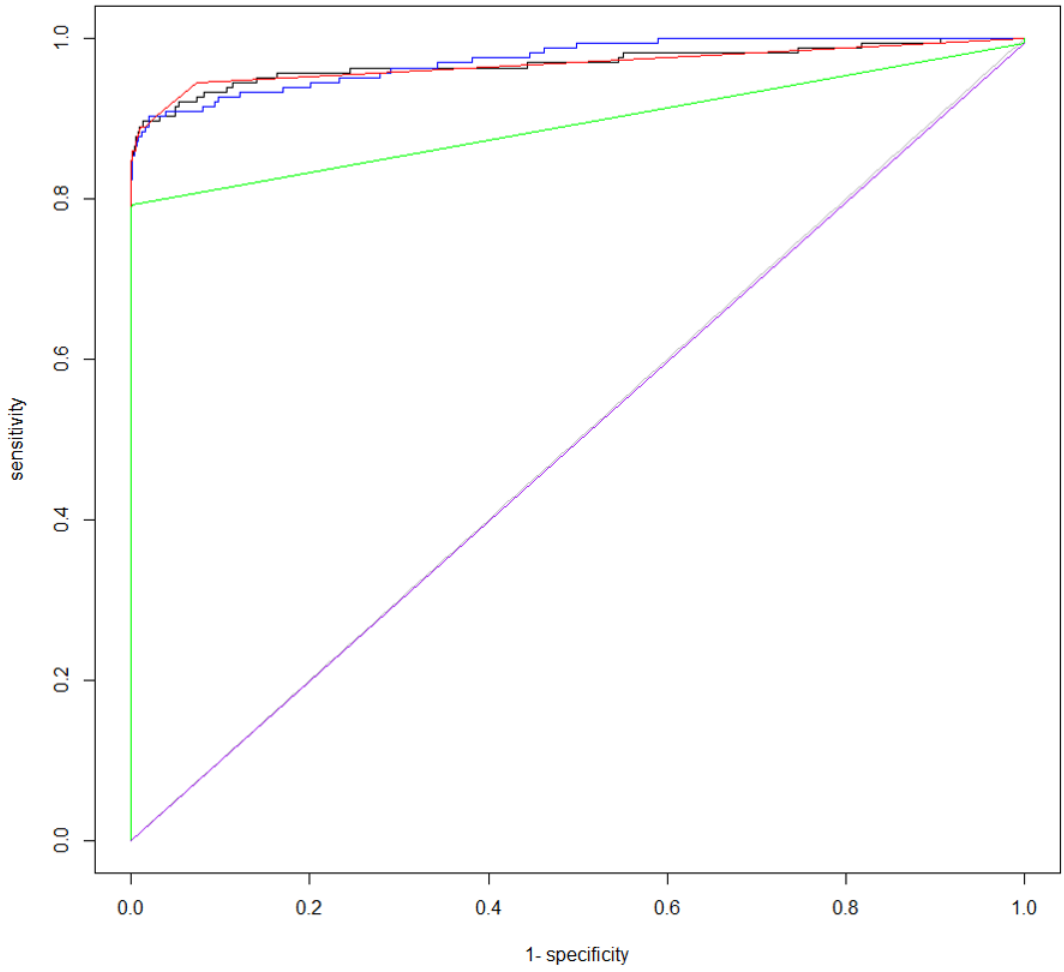

Figure 2: Receiver operating curve (ROC) for each model tested in this paper. Logistic regression (blue) had the highest AUROC at 0.98, although the random forest (red) and neural net (black) are also very similar, at 0.967 each. The decision tree (green) was certainly worse with an AUROC of 0.89. The purple line represents a base model where no fraud is predicted to occur and has an AUROC of 0.50.

hyperparameters for the neural network that result in a better model. Importantly, the nature of this dataset is fundamentally limiting in that feature selection has already been accomplished, while certain algorithms might outperform PCA in identifying more discriminatory features. Additionally, the scalability of this project as currently constituted is limited. Future research could focus on moving this project from R to Scala or Python, both of which are more easily scalable.

4.1 Imperfect Measures

The AUROC is a very convenient measure of model discrimination. Unlike accuracy, the AUROC equally weights positive and negative predictions, so it doesn't underperform too badly when one class is highly under sampled as in this dataset. However, there is some argument that the area under the precision-recall curve is a better measure with such unbalanced data. The precision-recall curve plots precision $\frac{TP}{TP+FP}$ against recall $\frac{TP}{TP+FN}$. Note that recall is equivalent to sensitivity, while precision measures the proportion of positive results. While the AUROC captures sensitivity

information, it does not capture precision information completely. A next step in this project would be to calculate the area under the precision-recall curve for each model and compare it with the AUROC. This could reveal that another model is more discriminatory than logistic regression, or at least verify that the difference between the models is slight.

4.2 Improved Models

Thin computational resources were a major factor in several decisions about what kind of models to run and how big of a search space for hyperparameters was possible. All three models could potentially be improved with model-specific hyperparameter tuning. The random forest model, for example, could be tuned on the control value (as could the decision tree, but that is less likely to be important anyway). This could reveal a better threshold for allowing variables, either making a better fit or preventing overfitting, both of which are detrimental to the model's performance. The logistic regression model could probably be improved with a form of ridge regression such as LASSO. However, the data is already sparse, which likely limits how effective ridge regression will be at improving the model. The neural network model, on the other hand, has several hyperparameter that could be tuned to create a more effective model. The most important tunable hyperparameter might be increasing the number of hidden layers. This might allow for emergent combinations of features that drastically improve model performance, or create an overfitting problem that drastically decreases model performance. Additionally, the network could be tuned on the number of nodes in the hidden layer (30 is likely a little small for this dataset), as well as the decay parameter (which could also improve computational efficiency). I suspect these methods could give a performance edge to the neural network, as many other similar projects have shown much more impressive results with neural networks. There is, however, reason to believe that there is something of a cap on how much information can be acquired from the current feature set.

4.3 Better Features

The need for anonymization of sensitive credit card data, while understandable, is extremely limiting in the breadth of analysis available. By utilizing PCA to anonymize the data, there is now no ability to make use of algorithms that might perform more effective feature creation. In particular, certain deep learning strategies would become more meaningful on raw transaction data. Recursive neural networks have seen significant use in natural language processing. One advantage this kind of model possesses is a limited ability to 'remember' past values and weight them for use on present predictions. In natural language processing, this allows for entire phrases to become significant where previously only individual words were considered. This type of action seems naturally suited to transaction data. The recursive neural network could use a series of several consecutive transactions to determine when a transaction is significantly different from those immediately prior. A wide breadth of transactions by different accounts would give context and weights to the importance of prior transactions relative to other features. While I suspect many variables that measure differences between past actions and the current transaction were included in the PCA features, there may be room for improvement over whatever method was used to select those variables. Better, more discriminatory features would be a method of improving model performance in identifying fraudulent behavior.

4.4 Scalability

R is a very convenient and useful language for performing many types of data mining/data analysis. However, R is limited in its scalability, and in the flexibility of implementation in a production environment. Python and Scala, on the other hand, are languages that have very strong support for highly scalable distributed computing. Future research could focus on implementing the algorithms tested in this paper in a scalable and resilient distributed computing system capable of handling higher velocity and volume of data than R. While the insights gained from analysis in R are useful, implementing and improving active fraud detection systems is a priority for most financial institutions.

5 Conclusion

Fraud is a significant cost to the world economy. Data mining offers valuable methods for minimizing the costs of fraud to companies and consumers. Of the four data mining algorithms tested in this paper, logistic regression was the top performer based on the AUROC measure, followed closely by a neural network and a random forest classifier. Future research should include utilizing different classifier performance measures such as the area under the precision-recall curve, tuning of hyperparameters in the neural network model, and better feature selection through deep learning methods. Future research can also focus on implementation of the algorithms tested in this paper in a production environment.

References

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D. S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609-616. Cambridge, MA: MIT Press.

[2] Kim, H., Pang, S., Je, H., Kim, D. & Bang, S. (2003). Constructing Support Vector Machine Ensemble. Pattern Recognition 36: 2757-2767

[3] Kim, M. & Kim, T. (2002). A Neural Classifier with Fraud Density Map for Effective Credit Card Fraud Detection. Proc. of IDEAL2002, 378-383.

[6] Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson & Gianluca Bontempi. Calibrating Probability with Undersampling for Unbalanced Classification. In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015

[7] Breiman L., Friedman J. H., Olshen R. A., & Stone, C. J. (1984) Classification and Regression Trees. Wadsworth.

[8] Breiman, L. (2001), Random Forests, Machine Learning 45(1), 5-32.

[9] Dobson, A. J. (1990) An Introduction to Generalized Linear Models. London: Chapman and Hall.

[4] Ghosh, S. & Reilly, D. (1994). Credit Card Fraud Detection with a Neural Network. Proc. of 27th Hawaii International Conference on Systems Science 3: 621-630.

[5] Maes, S., Tuyls, K., Vanschoenwinkel, B. & Manderick, B. (2002). Credit Card Fraud Detection using Bayesian and Neural Networks. Proc. of the 1st International NAISO Congress on [10] Ripley, B. D. (1996) Pattern Recognition and Neural Networks. Cambridge.

[11] Ballings, M., Van den Poel, D., Threshold Independent Performance Measures for Probabilistic Classifcation Algorithms, Forthcoming.

[12] Tape Thomas G. The Area Under an AUC Curve. University of Nebraska Medical Center. http://gim.unmc.edu/dxtests/Default.htm

6 Appendix: R Code

```
# Necessary modules
# AUC
if (!require("AUC")) {
  install.packages('AUC',
                   repos="https://cran.rstudio.com/",
                   quiet=TRUE)
  require('AUC')
}

# Random Forest
if (!require("randomForest")) {
  install.packages('randomForest',
                   repos="https://cran.rstudio.com/",
                   quiet=TRUE)
  require('randomForest')
}

# Neural Nets
if (!require("nnet")) {
  install.packages('nnet',
                   repos="https://cran.rstudio.com/",
                   quiet=TRUE)
  require('nnet')
}

# Decision trees
if (!require("rpart")) {
  install.packages('rpart',
                   repos="https://cran.rstudio.com/",
                   quiet=TRUE)
  require('rpart')
}

# LASSO log likelihood
if (!require("glmnet")) {
  install.packages('glmnet',
                   repos="https://cran.rstudio.com/",
                   quiet=TRUE)
  require('glmnet')
}

# Tunemember Function
# tuneMember is found at this address: source("http://ballings.co/hidden/aCRM/code/chapter2/tu
tuneMember <- function(call,tuning,xtest,ytest, predicttype=NULL,probability=TRUE){
  if (require(AUC)==FALSE) install.packages("AUC"); library(AUC)

  grid <- expand.grid(tuning)

  perf <- numeric()
  for (i in 1:nrow(grid)){
    Call <- c(as.list(call), grid[i,])
    model <-  eval(as.call(Call))

    predictions <- predict(model,xtest,type=predicttype, probability=probability)

    if (class(model)[2] == "svm") predictions <- attr(predictions,"probabilities")[,"1"]


    if (is.matrix(predictions)) if (ncol(predictions) == 2 ) predictions <- predictions[,2]
    perf[i] <- AUC::auc(roc(predictions,ytest))
  }
  perf <- data.frame(grid, auc=perf)
  perf[which.max(perf$auc),]
```

```
}


# Read in the dataset
cardfraud <- read.csv('C:/Users/chris/Documents/COSC_526/Final_Project/creditcardfraud/creditc

# Data Exploration
head(cardfraud)
length(which(cardfraud$Class == 1))
nrow(cardfraud)
length(is.na(cardfraud))
sum(is.na(cardfraud))
cardfraud[which(cardfraud$Class == 1),]


# Dataset looks clean, let's divvy it up into three equally sized categories.

# Create indicators for non-fraud
normind <- sample(x=1:nrow(cardfraud[which(cardfraud$Class == 0),]),size =nrow(cardfraud[which
trainindnor <- normind[1:round(length(normind)/3)]
valindnor <- normind[(round(length(normind)/3)+1):round(length(normind)*(2/3))]
testindnor <- normind[round(length(normind)*(2/3)+1):length(normind)]

# Create indicators for fraud
fraudind <- sample(x=1:nrow(cardfraud[which(cardfraud$Class == 1),]),size = nrow(cardfraud[whi
trainindfraud <- fraudind[1:round(length(fraudind)/3)]
valindfraud <- fraudind[(round(length(fraudind)/3)+1):round(length(fraudind)*(2/3))]
testindfraud <- fraudind[round(length(fraudind)*(2/3)+1):length(fraudind)]

# Make test, val, and train sets by combining fraud and non-fraud indicators
train <- rbind(cardfraud[which(cardfraud$Class == 0),][trainindnor,],cardfraud[which(cardfraud
val <- rbind(cardfraud[which(cardfraud$Class == 0),][valindnor,],cardfraud[which(cardfraud$Cla
test <- rbind(cardfraud[which(cardfraud$Class == 0),][testindnor,],cardfraud[which(cardfraud$C

# Set ytest, yval, and ytrain
ytrain <- as.factor(train$Class)
yval <- as.factor(val$Class)
ytest <- as.factor(test$Class)
ytrainbig <- as.factor(c(train$Class,val$Class))

# Drop the y values from the tables
train <- train[,!(names(train) %in% 'Class')]
val <- val[,!(names(val) %in% 'Class')]
test <- test[,!(names(test) %in% 'Class')]

# Make trainbig and ytrainbig
trainbig <- rbind(train,val)
ytrainbig <- as.factor(c(as.character(ytrain),as.character(yval)))

# Erase all variables except the necessary ones (saves on RAM)
rm(list=setdiff(ls(),c('train','val','test','ytrain','yval','ytest','trainbig','ytrainbig')))


## Decision Tree

# Grow a tree on trainbig
tree <- rpart(ytrainbig ~ .,
              trainbig,
              method = "class")

# Predict for all instances in test
predTree <- predict(tree,test)[,2]

# Compute the AUC
AUC::auc(roc(predTree,ytest))
```

```
AUC::auc(sensitivity(predTree,ytest))

# Make a nice plot of the tree
par(xpd = TRUE)
plot(tree, compress = TRUE)
text(tree, use.n = TRUE)

# Plot the AUROC
plot(roc(predTree,ytest))

## Random Forest
rFmodel <- randomForest(x=trainbig,
                        y=ytrainbig,
                        ntree=500,
                        importance=TRUE)
predrF <- predict(rFmodel,test,type='prob')[,2]

#assess model performance with AUROC
AUC::auc(roc(predrF,ytest))
# Assess model with AUSEC
AUC::auc(sensitivity(predrF,ytest))

# Plot the AUROC
plot(roc(predrF,ytest))

# A confusion matrix with a 0.5, 0.25, and 0.1 cutoff
table(ytest,predrF > 0.5)
table(ytest,predrF > 0.25)
table(ytest,predrF > 0.1)

## Zero Model
# An easy and foolish model
zeromod <- as.factor(numeric(94935))

# Accuracy of zero model
1-sum(as.numeric(as.character(ytest)))/nrow(test)

table(zeromod,ytest)

length(ytest)-sum(as.numeric(as.character(ytest)))

AUC::auc(roc(zeromod,ytest))


## Logistic Regression

LR <- glm(ytrainbig ~ .,
          data = trainbig,
          family = binomial("logit"))

(logisticsum <- summary(LR))
sort(abs(logisticsum$coefficients[,1]),decreasing = TRUE)

predLR <- predict(LR,
                  newdata=test,
                  type="response")


AUC::auc(roc(predLR,ytest))
AUC::auc(sensitivity(predLR,ytest))

# most important values
sort(abs(LR$coefficients))

## Neural Nets
```

```
# The data need to be scaled to train adequately
minima <- sapply(train,min)
scaling <- sapply(train,max)-minima

trainscale <- data.frame(base::scale(train,
                                     center=minima,
                                     scale=scaling))
sapply(trainscale,range)
valscale <- data.frame(base::scale(val,
                                   center=minima,
                                   scale=scaling))
testscale <- data.frame(base::scale(test,
                                     center=minima,
                                     scale=scaling))
trainbigscale <- data.frame(base::scale(trainbig,
                                        center=minima,
                                        scale=scaling))

# needs an NN.rang value and NN.maxit value
NN.rang <- 0.5
NN.maxit <- 10000
# Tuning values (can change depending on needs)
NN.size <- c(15,20,25,30)
NN.decay <- c(0.01,0.1)


# Create a call
call <- call("nnet",
             formula = ytrain ~ .,
             data = trainscale,
             rang = NN.rang,
             maxit = NN.maxit,
             trace = FALSE,
             MaxNWts = Inf)
tuning <- list(size=NN.size, decay=NN.decay)

# Call tuneMember
(result <- tuneMember(call=call,
                      tuning=tuning,
                      xtest=valscale,
                      ytest=yval,
                      predicttype='raw'))

# Test on ytrainbig
NN <- nnet(ytrainbig ~ .,
           trainbigscale,
           size = NN.size[4],
           rang = NN.rang,
           decay = NN.decay[3],
           maxit = NN.maxit,
           trace = TRUE,
           MaxNWtx= Inf)
predNN <- as.numeric(predict(NN, testscale,type='raw'))

AUC::auc(roc(predNN,ytest))
AUC::auc(sensitivity(predNN,ytest))

plot(roc(predNN,ytest))
plot(roc(predLR,ytest),add=TRUE, col = "blue")
plot(roc(predrF,ytest),add = TRUE, col = "red")
plot(roc(predTree,ytest),add=TRUE,col="green")
plot(roc(zeromod,ytest),add=TRUE,col="purple")

plot(sensitivity(predNN,ytest))
```

```
plot(sensitivity(predLR,ytest),add=TRUE, col = "blue")
plot(sensitivity(predrF,ytest),add = TRUE, col = "red")
plot(sensitivity(predTree,ytest),add=TRUE,col="green")
plot(sensitivity(zeromod,ytest),add=TRUE,col="purple")
```