Fraud Detection: Logit GLM, Boosting, & SVM

Mikhail Lara

Aim of Study

The purpose of this study is to compare the out-of-box performance of different models that could used to predict the occurrence of credit card fraud in an extremely unbalanced dataset. The models evaluated in this study are:

1. Logistic Regression
2. Gradient Boosting with Trees
3. Two Class Support Vector Machine
4. One Class Support Vector Machine

Load Data

```
suppressMessages(options(warn = -1))
suppressMessages(library(data.table))
suppressMessages(library(ggplot2))
suppressMessages(library(gridExtra))
suppressMessages(library(glmnet))
suppressMessages(library(caret))
suppressMessages(library(corrplot))
suppressMessages(library(GGally))
suppressMessages(library(kernlab))
suppressMessages(library(gbm))
suppressMessages(library(plyr))

path <- '/Users/Mikey/Documents/ML-Case-Studies/Credit Card Fraud Detection'
setwd(path)
DT<- fread('creditcard.csv',sep = ',', colClasses = rep('numeric',31))
```

```
##
Read 91.3% of 284807 rows
Read 284807 rows and 31 (of 31) columns from 0.140 GB file in 00:00:03
```

Dataset Description

The dataset consists of **284807 transactions** with each observation classified as either legitimate(Class=0) or fraudulent(Class=1). Each observation has **30 quantitative fields** that can be used as predictors.

It should be noted that the 'Time' field is just the time lapse between each observation and the 1st observation in the dataset. It is essentially a row index and **provides no meaningful date-time information** so it should be excluded from any preditive analysis.

The data is extremely unbalanced with only **492 fraudulent transactions**, corresponding to only 1.727%. This degree of skewness is problematic for any predictive model because the majority Class will dominate the calculations, making it difficult to build an accurate classifier for fraudulent transactions. Data resampling is needed to create sub-training sets with more balanced occurrences of both Class levels.

```r
#How Many Ocurrences of Each Class
DT[,.N, by = Class]
```

```
##    Class      N
## 1:     0 284315
## 2:     1    492
```

Linear Correlation Analysis

There is poor linear correlation between the transaction Class and the possible predictors. Of the 29 non-trivial fields, only 11 have a correlation with a magnitude greater than 10% and only 2 of those correlations exceed 30%. For the sake of simplicity, only those predictors with a Class correlation of at least 10% is be used to build a logistic model for the odds that a transaction is fraudulent.

```r
#Linear Correlation of Fraud with Predictors
invisible(DT[,Class:= as.numeric(Class)])

M<-cor(DT)
tbl<-data.table(correlation = M[31,sort.list(abs(M[31,1:30]),decreasing = TRUE)],
                varname=names(DT)[sort.list(abs(M[31,1:30]),decreasing = TRUE)])
tbl
```

```
##       correlation varname
##  1: -0.3264810672     V17
##  2: -0.3025436958     V14
##  3: -0.2605929249     V12
##  4: -0.2168829436     V10
##  5: -0.1965389403     V16
##  6: -0.1929608271      V3
##  7: -0.1872565915      V7
##  8:  0.1548756447     V11
##  9:  0.1334474862      V4
## 10: -0.1114852539     V18
## 11: -0.1013472986      V1
## 12: -0.0977326861      V9
## 13: -0.0949742990      V5
## 14:  0.0912886503      V2
## 15: -0.0436431607      V6
## 16:  0.0404133806     V21
## 17:  0.0347830130     V19
## 18:  0.0200903242     V20
## 19:  0.0198751239      V8
## 20:  0.0175797282     V27
## 21: -0.0123225709    Time
## 22:  0.0095360409     V28
## 23: -0.0072209067     V24
## 24:  0.0056317530  Amount
## 25: -0.0045697788     V13
## 26:  0.0044553975     V26
## 27: -0.0042234023     V15
## 28:  0.0033077056     V25
## 29: -0.0026851557     V23
## 30:  0.0008053175     V22
##       correlation varname
```

Data Partitioning

There are 2 major issues with directly using the raw data for modeling:

1. The dataset is too large to use in model building all at once. This is true even if a 60/40 split is used.
2. The dataset is unbalanced and needs to be adjusted so the fraud data is not modeled as just 'noise'.

Both issues are addressed by making several smaller subsets of the training data that are only 42% the size of the original dataset and are rebalanced so that the fraud/legitimate disparity is less severe. Specifically, a **primary training set** is created using 60% of the raw data(i.e. 60% of all fraud & 60% of all legitimate) and **10 sub-training sets** are created from it via resampling without replacement. The sub-training sets are created in such a way that the frequency of legitimate transactions is only 8 times that of the fraudulent transactions.

Note: The choice of a 8:1 legitimate/fraud occurrence ratio is arbitrary. A more complete analysis should assess the effect of data rebalancing on the predictive power to detect fraud.

```r
invisible(DT[,Class:= as.factor(Class)])

set.seed(1234)
inTrain<-createDataPartition(DT$Class, p = 0.6,list=FALSE)
train <-DT[inTrain]  #419 Total Occurrences of Fraud
test  <-DT[-inTrain] #73 Occurrences of Fraud

nonfraud<-which(train$Class==0)
fraud<-which(train$Class==1)

#Use nonfraud-to-fraud ratio of 8:1
set.seed(1234)

train.sub<-list()
for(i in 1:10){
  train.sub[[i]]<-train[c(sample(fraud,length(fraud)*(.75)),sample(nonfraud,length(fraud)*8*(.75)))]
}
```

Traditional Logistic Regression

Regularized regression approach would normally be used to handle correlation between the predictors to create an accurate linear model with the fewest number of correlated predictors. However, the values provided in the dataset have been 'decorrelated' via principle component analysis(PCA) resulting is essentially independent predictors. Therefore, standard logistic regression considering only major effects is used.

```r
logit.models<-list()

set.seed(1234)
for(i in 1:10){
        logit.models[[i]]<- glm(Class~V17+V14+V12+V10+V16+V3+V7+V11+ V4+ V18+ V1 ,
                                    data = train.sub[[i]],family = 'binomial')
}
```

The predictors used for the final logistic model are selected based on the p-values for each term in the models built on the 10 sub-training datasets. A **p-value of 0.01** is used as the significance threshold for terms in the glm.

```r
# for(i in 1:10){
#  print(t(which(coef(summary(logit.models[[i]]))[,4]<.01)))
# }

set.seed(1234)
logistic.final<-glm(data= train,as.factor(Class)~V14+V10+V16+V3+V4,family = 'binomial')
summary(logistic.final)
```

```
##
## Call:
## glm(formula = as.factor(Class) ~ V14 + V10 + V16 + V3 + V4, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.9898  -0.0308  -0.0216  -0.0150   4.5990
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.33191    0.14333 -58.129  < 2e-16 ***
## V14         -0.78556    0.04144 -18.955  < 2e-16 ***
## V10         -0.39843    0.05774  -6.900 5.20e-12 ***
## V16         -0.32198    0.05779  -5.572 2.52e-08 ***
## V3           0.15246    0.03129   4.873 1.10e-06 ***
## V4           0.61770    0.05155  11.984  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4355.7  on 170884  degrees of freedom
## Residual deviance: 1459.3  on 170879  degrees of freedom
## AIC: 1471.3
##
## Number of Fisher Scoring iterations: 11
```

Logistic Model Accuracy

The standard confusion matrix metrics are calculated using both the **primary training set** and the **test set**:

1. Recall
2. True Negative Rate (TNR)
3. Positive Predictive Value (PPV)
4. Negative Predictive Value (NPV)

```r
#In-Sample Prediction
  pred.logit<-predict.glm(logistic.final,newdata=train)
  pred.expit<- exp(pred.logit)/(1+exp(pred.logit))

  a<-which(train$Class==1)
  b<-which(train$Class==0)

  TP<-length(which(pred.expit[a]>0.5))
  FN<-length(which(pred.expit[a]<=0.5))
  TN<-length(which(pred.expit[b]<0.5))
  FP<-length(which(pred.expit[b]>=0.5))
  Recall<-TP/(TP+FN)
  TNR<-TN/(TN+FP)
  PPV<-TP/(TP+FP)
  NPV<-TN/(TN+FN)

#Out of Sample Prediction
  pred.logit.test<-predict.glm(logistic.final,newdata=test)
  pred.expit.test<- exp(pred.logit.test)/(1+exp(pred.logit.test))

  a<-which(test$Class==1)
  b<-which(test$Class==0)

  TP.test<-length(which(pred.expit.test[a]>0.5))
  FN.test<-length(which(pred.expit.test[a]<=0.5))
  TN.test<-length(which(pred.expit.test[b]<0.5))
  FP.test<-length(which(pred.expit.test[b]>=0.5))
  Recall.test<-TP.test/(TP.test+FN.test)
  TNR.test<-TN.test/(TN.test+FP.test)
  PPV.test<-TP.test/(TP.test+FP.test)
  NPV.test<-TN.test/(TN.test+FN.test)
```

The training set and test set accuracy metrics show that the final logistic model is able to accurately and reliably label legitimate credit card transactions. This is demonstrated by TNR and NPV values exceeding 0.99. It does not, however, have the same degree of accuracy for fraudulent transactions with a test Recall values of only 0.59, which is marginally better than random guessing. Although the test set PPV suggests that any transaction labeled as fraud is reliable 87% of the time, the poor model recall disqualifies the final logistic model from having any in-field applications.

```
##Summary
logit.DT<-data.table(data_set=c('train','test'),
                     recall = c(Recall,Recall.test),
                     true.negative.rate=c(TNR,TNR.test),
                     pos.pred.value=c(PPV,PPV.test),
                     neg.pred.value=c(NPV,NPV.test))

logit.DT
```

```
##    data_set    recall true.negative.rate pos.pred.value neg.pred.value
## 1:    train 0.6047297          0.9998593      0.8817734      0.9993145
## 2:     test 0.5969388          0.9998505      0.8731343      0.9993057
```

Machine Learning Classifiers

Gradient Boosting with Trees: Cross-Validation

GBM classifiers use weighted decision trees as the weak learners. The tree depth needed to get high accuracy is typically less than 3. As with the logit regression model, multiple GBM classifiers are created using the sub-training data sets. Each of the 10 gradient boosting models is able to classify fraudulent transactions on the entire training set moderately well with the least accurate model having a fraud sensitivity of 0.878.

```
##Convert Class Data into 2-level factor variables
for(i in 1:10){
  invisible(train.sub[[i]][,Class:=as.factor(Class)])
}


gbm.fit<-list()

  set.seed(1234)
  for(i in 1:10){
      gbm.fit[[i]]<-train(data = train.sub[[i]], Class~.-Time,method='gbm',verbose=FALSE)
  }


#Calculate Recall for All GBM Models on Entire Training Set
    gbm.recall=c()
    gbm.specificity=c()

    a<-which(train$Class=='1')
    b<-which(train$Class=='0')

    for(i in 1:10){
          gbm.recall[i]<-length(which(predict(gbm.fit[[i]],newdata =train)[a]=='1'))/length(a)
          gbm.specificity[i]<-length(which(predict(gbm.fit[[i]],newdata =train)[b]=='0'))/length(b)
    }
      data.table(Model=c(1:10),recall=gbm.recall,spec=gbm.specificity)
```

```
##     Model    recall      spec
##  1:     1 0.8783784 0.9964652
##  2:     2 0.8885135 0.9960490
##  3:     3 0.9054054 0.9975966
##  4:     4 0.9324324 0.9941731
##  5:     5 0.8851351 0.9960255
##  6:     6 0.8851351 0.9963890
##  7:     7 0.8817568 0.9950348
##  8:     8 0.9222973 0.9965883
##  9:     9 0.8986486 0.9940149
## 10:    10 0.8952703 0.9962073
```

The final gbm model is a composite of all 10 classifiers. It labels transactions as fraudulent if a minimum number of the predictors classify it as fraudulent. To determine what that threshold should be, the true positive rate(TPR) was calculated for predictor thresholds varying from 1-10.

The results show that the final model is able to label a fraudulent training set transaction best when a minimum of 1 gbm model classifies it as fraudulent. However, when the voter threshold is set to 1, the PPV of the final model is very poor (PPV=0.11), which would make its predictions unreliable in practice. The voter threshold is set to 10 for prediction on the test set because:

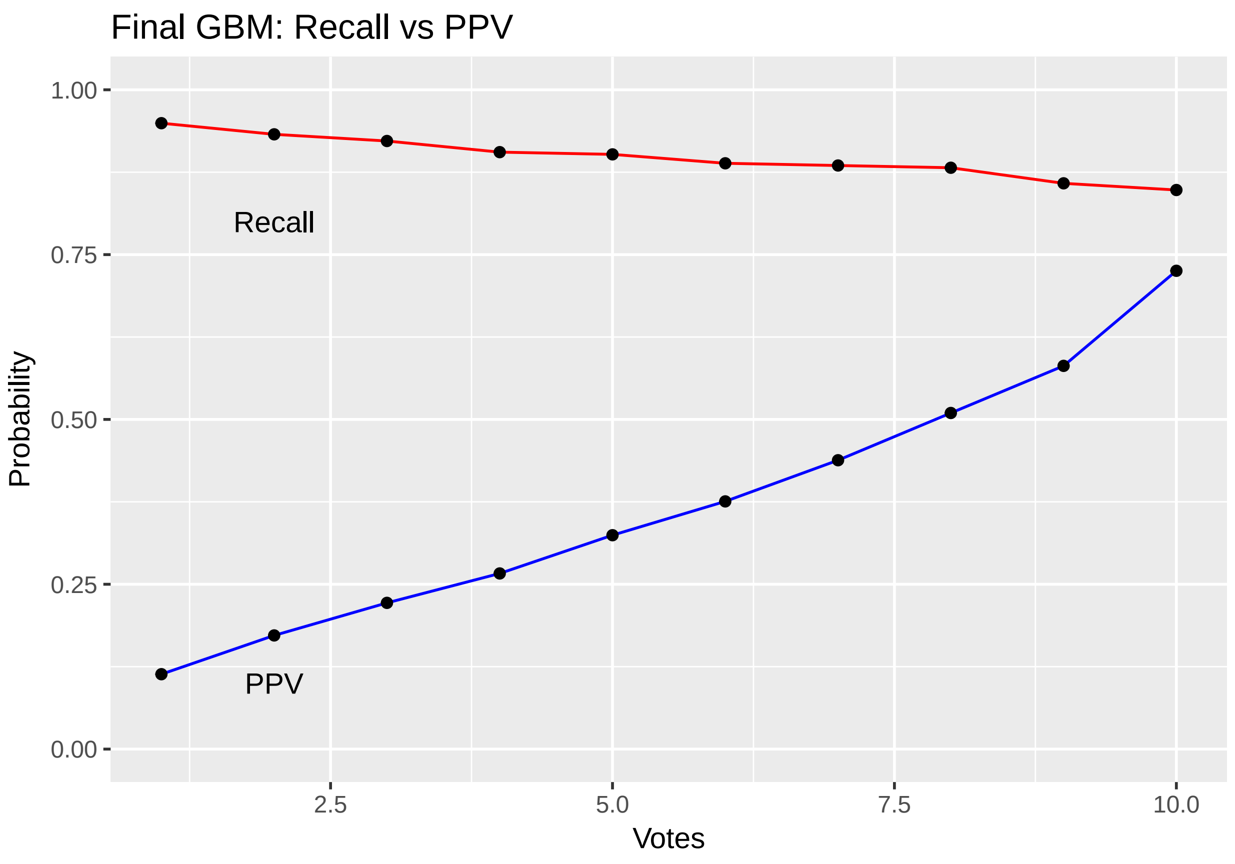
1. It results in an acceptable out-of-box value of 0.84 and a PPV of 0.72, which is a significant improvement over the 0.11
2. it as close as the model can get to the crossover point (Recall=PPV) with only 10 classifiers.

```
#Final GBM Model - Validation on Training Set
    v1<-predict(gbm.fit[[1]],newdata=train)
    v2<-predict(gbm.fit[[2]],newdata=train)
    v3<-predict(gbm.fit[[3]],newdata=train)
    v4<-predict(gbm.fit[[4]],newdata=train)
    v5<-predict(gbm.fit[[5]],newdata=train)
    v6<-predict(gbm.fit[[6]],newdata=train)
```

```r
v7<-predict(gbm.fit[[7]],newdata=train)
v8<-predict(gbm.fit[[8]],newdata=train)
v9<-predict(gbm.fit[[9]],newdata=train)
v10<-predict(gbm.fit[[10]],newdata=train)

v.agg<-as.numeric(v1)+as.numeric(v2)+as.numeric(v3)+as.numeric(v4)+as.numeric(v5)+
  as.numeric(v6)+as.numeric(v7)+as.numeric(v8)+as.numeric(v9)+as.numeric(v10)-10

final.gbm<-data.table(v.agg, truth=train$Class)

gbm.TPR<-c()
gbm.FPR<-c()

a<-which(train$Class==1)
b<-which(train$Class==0)

Recall.train<-c()
PPV.train<-c()
#Evaluate threshold effect
for(i in 1:10){
    final.gbm[,prediction:=as.factor(ifelse(v.agg>=i,'1','0'))]

    TP.train<-length(which(final.gbm$prediction[a]=='1'))
    FN.train<-length(which(final.gbm$prediction[a]=='0'))
    TN.train<-length(which(final.gbm$prediction[b]=='0'))
    FP.train<-length(which(final.gbm$prediction[b]=='1'))

    Recall.train[i]<-TP.train/(TP.train+FN.train)
    PPV.train[i]<-TP.train/(TP.train+FP.train)
}

data.table(votes=c(1:10),Recall=Recall.train,PPV=PPV.train)
```

```
##     votes    Recall       PPV
##  1:     1 0.9493243 0.1135812
##  2:     2 0.9324324 0.1723923
##  3:     3 0.9222973 0.2217709
##  4:     4 0.9054054 0.2664016
##  5:     5 0.9020270 0.3244228
##  6:     6 0.8885135 0.3757143
##  7:     7 0.8851351 0.4381271
##  8:     8 0.8817568 0.5097656
##  9:     9 0.8581081 0.5812357
## 10:    10 0.8479730 0.7254335
```

```r
ggplot(data=data.table(votes=c(1:10),Recall=Recall.train,PPV=PPV.train),aes(x=votes,y=Recall))+
    geom_line(colour='red')+geom_point()+
    geom_line(aes(c=votes,y=PPV),colour='blue')+geom_point(aes(c=votes,y=PPV))+
    ylim(0,1)+ylab('Probability')+xlab('Votes')+ggtitle('Final GBM: Recall vs PPV')+
    annotate('text',x=2,y=0.8,label='Recall')+
    annotate('text',x=2,y=0.1,label='PPV')
```



```
    # ggplot(data=data.table(threshold=c(1:10),TPR=gbm.TPR,FPR=gbm.FPR),aes(x=threshold,y=TPR))+
    #     geom_point(colour='blue')+geom_line()+
    #     ylim(0.8,1)+xlim(1,10)+xlab('Threshold')+ylab('TPR')+ggtitle('Threshold Effect on GBM TPR')
    #
    # ggplot(data=data.table(threshold=as.factor(c(1:10)),TPR=gbm.TPR,FPR=gbm.FPR),aes(x=FPR,y=TPR,colo
    #     geom_point()+geom_line()+
    #     ylim(0,1)+xlim(0,1)+
    #     geom_line(data=data.table(x=c(0,1),y=c(0,1)),aes(x=x,y=y),colour='black')+
    #     xlab('FPR')+ylab('TPR')+ggtitle('Partial ROC Curve for Final GBM')
    #
```

The final composite gradient boosting model is much better than as the traditional logistic regression at identifying fraud when it occurs, shown by a test set recall 0.87. However, It also has test set PPV of 0.69 meaning that the predictions of fraudulent transactions would be false alarms approximately 30% of the time. The pairing of a high recall and a moderate PPV is troublesome because it implies the final model correctly classifies fraud when it occurs but predictions of fraud are not as reliable.

The overly conservative fraud predictions may be due to either the rebalancing of the training data to an 8:1 legitimate-to-fraud ratio or setting the minimum predictor threshold to 1. A good next step to improve the composite gbm model would be to run a rebalancing study to see how it affects the PPV.

```
#Train
    invisible(final.gbm[,prediction:=as.factor(ifelse(v.agg>=10,'1','0'))])

    a<-which(train$Class==1)
    b<-which(train$Class==0)

    TP.train<-length(which(final.gbm$prediction[a]=='1'))
    FN.train<-length(which(final.gbm$prediction[a]=='0'))
    TN.train<-length(which(final.gbm$prediction[b]=='0'))
    FP.train<-length(which(final.gbm$prediction[b]=='1'))
    Recall.train<-TP.train/(TP.train+FN.train)
    TNR.train<-TN.train/(TN.train+FP.train)
    PPV.train<-TP.train/(TP.train+FP.train)
    NPV.train<-TN.train/(TN.train+FN.train)

#Final GBM Model (Test Set)
    v1<-predict(gbm.fit[[1]],newdata=test)
    v2<-predict(gbm.fit[[2]],newdata=test)
    v3<-predict(gbm.fit[[3]],newdata=test)
    v4<-predict(gbm.fit[[4]],newdata=test)
    v5<-predict(gbm.fit[[5]],newdata=test)
    v6<-predict(gbm.fit[[6]],newdata=test)
    v7<-predict(gbm.fit[[7]],newdata=test)
    v8<-predict(gbm.fit[[8]],newdata=test)
    v9<-predict(gbm.fit[[9]],newdata=test)
    v10<-predict(gbm.fit[[10]],newdata=test)

    v.agg<-as.numeric(v1)+as.numeric(v2)+as.numeric(v3)+as.numeric(v4)+as.numeric(v5)+
      as.numeric(v6)+as.numeric(v7)+as.numeric(v8)+as.numeric(v9)+as.numeric(v10)-10

    invisible(final.gbm<-data.table(v.agg, truth=as.factor(test$Class)))
    invisible(final.gbm[,prediction:=as.factor(ifelse(v.agg>=10,'1','0'))])

    a<-which(test$Class=='1')
    b<-which(test$Class=='0')

    TP.test<-length(which(final.gbm$prediction[a]=='1'))
    FN.test<-length(which(final.gbm$prediction[a]=='0'))
    TN.test<-length(which(final.gbm$prediction[b]=='0'))
    FP.test<-length(which(final.gbm$prediction[b]=='1'))
    Recall.test<-TP.test/(TP.test+FN.test)
    TNR.test<-TN.test/(TN.test+FP.test)
    PPV.test<-TP.test/(TP.test+FP.test)
    NPV.test<-TN.test/(TN.test+FN.test)

gbm.DT<-data.table(data_set=c('train','test'),
                     recall = c(Recall.train,Recall.test),
                     true.negative.rate=c(TNR.train,TNR.test),
                     pos.pred.value=c(PPV.train,PPV.test),
                     neg.pred.value=c(NPV.train,NPV.test))

gbm.DT
```

```
##    data_set   recall true.negative.rate pos.pred.value neg.pred.value
## 1:    train 0.847973          0.9994431      0.7254335      0.9997361
## 2:     test 0.872449          0.9993405      0.6951220      0.9997801
```

Support Vector Machine (Standard 2-Class Classification)

Support vector machine classifiers are formulated to distinguish between different groups by defining a hyperplane-based separation criteria. If the predictor data for the fraudulent and legitimate transactions do exhibit some form of high-dimensional grouping, a multi-class SVM should be able to define an accurate classification method that using a sparse number of data points(support vectors).

```
#Use nonfraud-to-fraud ratio of 4:1

set.seed(1234)
    svm1.fit<-list()
    for(i in 1:10){
        svm1.fit[[i]]<-train(data = train.sub[[i]], as.factor(Class)~.-Time,
                        method='svmRadial',
                        verbose=FALSE)
    }

    acc<-c()
    for(i in 1:10){
        acc[i]<-max(svm1.fit[[i]]$results[,3])
    }

    data.table(svm_model=c(1:10),Accuracy=acc)
```

```
##     svm_model  Accuracy
##  1:         1 0.9781442
##  2:         2 0.9758756
##  3:         3 0.9778860
##  4:         4 0.9738895
##  5:         5 0.9735214
##  6:         6 0.9794037
##  7:         7 0.9788731
##  8:         8 0.9737157
##  9:         9 0.9757146
## 10:        10 0.9701685
```

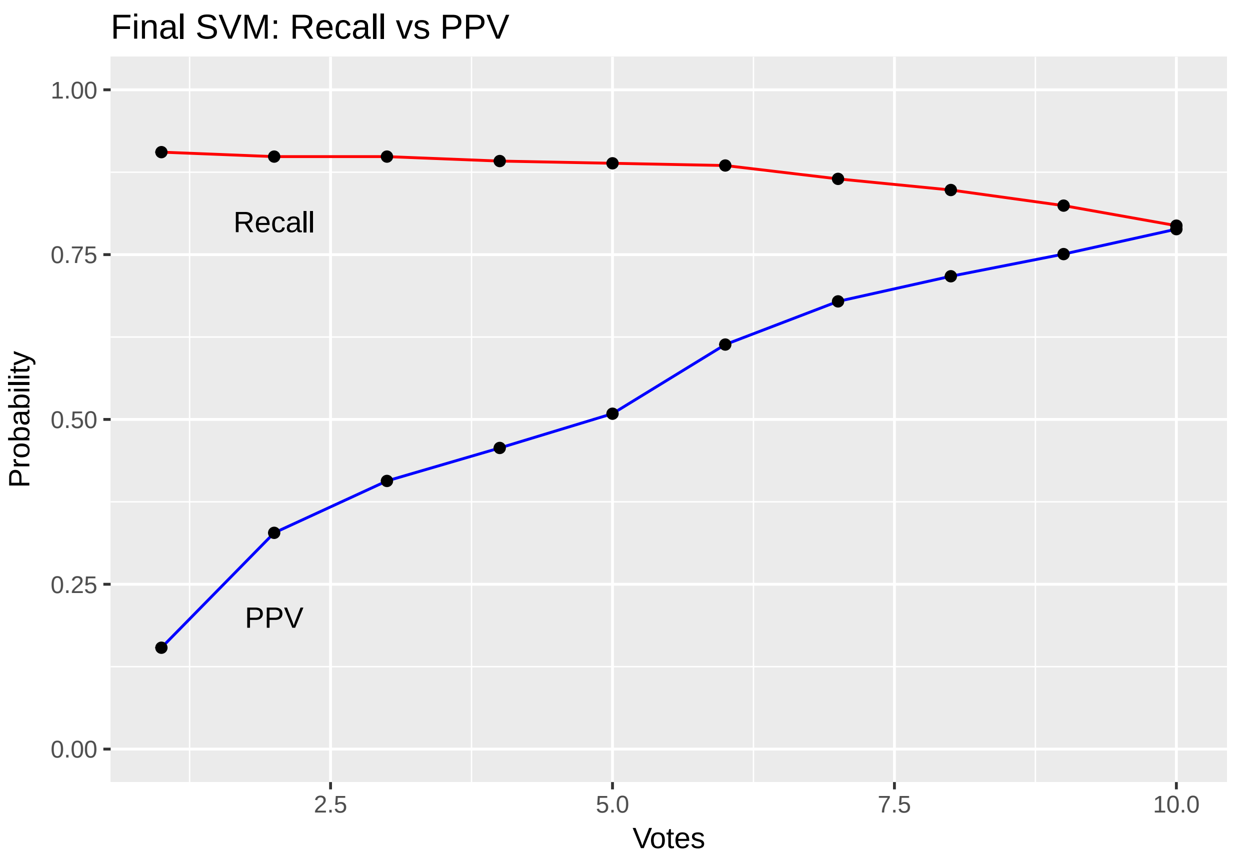
By comparing the recall and PPV on the **primary training set**, it is clear that increasing the voting threshold has significant value. Increasing the minimum voter threshold from 1 to 10 results in a significantly greater PPV with only a moderate decrease in the recall. The voting threshold that will be used for the final svm model will be 6. This threshold has been chosen because it is the cross-over point where the recall and PPV are roughly equal.

```
#Final SVM Model (Training Set)
    v1<-predict(svm1.fit[[1]],newdata=train)
    v2<-predict(svm1.fit[[2]],newdata=train)
    v3<-predict(svm1.fit[[3]],newdata=train)
    v4<-predict(svm1.fit[[4]],newdata=train)
    v5<-predict(svm1.fit[[5]],newdata=train)
    v6<-predict(svm1.fit[[6]],newdata=train)
    v7<-predict(svm1.fit[[7]],newdata=train)
    v8<-predict(svm1.fit[[8]],newdata=train)
    v9<-predict(svm1.fit[[9]],newdata=train)
    v10<-predict(svm1.fit[[10]],newdata=train)

    v.agg<-as.numeric(v1)+as.numeric(v2)+as.numeric(v3)+as.numeric(v4)+as.numeric(v5)+
      as.numeric(v6)+as.numeric(v7)+as.numeric(v8)+as.numeric(v9)+as.numeric(v10)-10
```

```
final.svm1.train<-data.table(v.agg, truth=as.factor(train$Class))

a<-which(train$Class==1)
b<-which(train$Class==0)

Recall.train<-c()
PPV.train<-c()
#Evaluate threshold effect
for(i in 1:10){
    final.svm1.train[,prediction:=as.factor(ifelse(v.agg>=i,'1','0'))]

    TP.train<-length(which(final.svm1.train$prediction[a]=='1'))
    FN.train<-length(which(final.svm1.train$prediction[a]=='0'))
    TN.train<-length(which(final.svm1.train$prediction[b]=='0'))
    FP.train<-length(which(final.svm1.train$prediction[b]=='1'))

    Recall.train[i]<-TP.train/(TP.train+FN.train)
    PPV.train[i]<-TP.train/(TP.train+FP.train)
}

data.table(votes=c(1:10),Recall=Recall.train,PPV=PPV.train)
```

```
##     votes    Recall       PPV
##  1:     1 0.9054054 0.1537579
##  2:     2 0.8986486 0.3279901
##  3:     3 0.8986486 0.4067278
##  4:     4 0.8918919 0.4567474
##  5:     5 0.8885135 0.5087041
##  6:     6 0.8851351 0.6135831
##  7:     7 0.8648649 0.6790451
##  8:     8 0.8479730 0.7171429
##  9:     9 0.8243243 0.7507692
## 10:    10 0.7939189 0.7885906
```

```
ggplot(data=data.table(votes=c(1:10),Recall=Recall.train,PPV=PPV.train),aes(x=votes,y=Recall))+
    geom_line(colour='red')+geom_point()+
    geom_line(aes(c=votes,y=PPV),colour='blue')+geom_point(aes(c=votes,y=PPV))+
    ylim(0,1)+ylab('Probability')+xlab('Votes')+ggtitle('Final SVM: Recall vs PPV')+
    annotate('text',x=2,y=0.8,label='Recall')+
    annotate('text',x=2,y=0.2,label='PPV')
```



The test set results for recall and PPV of the final svm composite classifier is comparable to those of the training data. This type of classifier does a good job of detecting credit card fraud without requiring too much tuning.

```
    invisible(final.svm1.train[,prediction:=as.factor(ifelse(v.agg>=6,'1','0'))])

    TP.train<-length(which(final.svm1.train[a]$prediction=='1'))
    FN.train<-length(which(final.svm1.train[a]$prediction=='0'))
    TN.train<-length(which(final.svm1.train[b]$prediction=='0'))
    FP.train<-length(which(final.svm1.train[b]$prediction=='1'))
    Recall.train<-TP.train/(TP.train+FN.train)
    TNR.train<-TN.train/(TN.train+FP.train)
    PPV.train<-TP.train/(TP.train+FP.train)
    NPV.train<-TN.train/(TN.train+FN.train)

#Final SVM Model (Test Set)
    v1<-predict(svm1.fit[[1]],newdata=test)
    v2<-predict(svm1.fit[[2]],newdata=test)
    v3<-predict(svm1.fit[[3]],newdata=test)
    v4<-predict(svm1.fit[[4]],newdata=test)
    v5<-predict(svm1.fit[[5]],newdata=test)
    v6<-predict(svm1.fit[[6]],newdata=test)
    v7<-predict(svm1.fit[[7]],newdata=test)
    v8<-predict(svm1.fit[[8]],newdata=test)
    v9<-predict(svm1.fit[[9]],newdata=test)
    v10<-predict(svm1.fit[[10]],newdata=test)
    v.agg<-as.numeric(v1)+as.numeric(v2)+as.numeric(v3)+as.numeric(v4)+as.numeric(v5)+
      as.numeric(v6)+as.numeric(v7)+as.numeric(v8)+as.numeric(v9)+as.numeric(v10)-10

    final.svm1.test<-data.table(v.agg, truth=as.factor(test$Class))
    invisible(final.svm1.test[,prediction:=as.factor(ifelse(v.agg>=6,'1','0'))])

    a<-which(test$Class==1)
    b<-which(test$Class==0)

    TP.test<-length(which(final.svm1.test[a]$prediction=='1'))
    FN.test<-length(which(final.svm1.test[a]$prediction=='0'))
    TN.test<-length(which(final.svm1.test[b]$prediction=='0'))
    FP.test<-length(which(final.svm1.test[b]$prediction=='1'))
    Recall.test<-TP.test/(TP.test+FN.test)
    TNR.test<-TN.test/(TN.test+FP.test)
    PPV.test<-TP.test/(TP.test+FP.test)
    NPV.test<-TN.test/(TN.test+FN.test)

svm1.DT<-data.table(data_set=c('train','test'),
                     recall = c(Recall.train,Recall.test),
                     true.negative.rate=c(TNR.train,TNR.test),
                     pos.pred.value=c(PPV.train,PPV.test),
                     neg.pred.value=c(NPV.train,NPV.test))

svm1.DT
```

```
##    data_set    recall true.negative.rate pos.pred.value neg.pred.value
## 1:    train 0.8851351          0.9990328      0.6135831      0.9998005
## 2:     test 0.8316327          0.9989800      0.5842294      0.9997096
```

Support Vector Machine (Novelty Detection via Single Class SVM)

An alternative to multi-class classification with SVM is 1-class classification, or novelty detection. It is an odd classification method because it constructs a 'separating' hyperplane based only on the data from 1 class. In theory, if all the information that uniquely describes the legitimate data is contained in the dataset, a descrimination algorithm can be created without having to make any adjustments to account imbalance between the legitimate and fraudulent data in the **primary training set** or the **test set**.

The SVM is trained on a subsample of the legitimate training date with the same number of observations as all the sub-training sets combined to make its results comparable to the previous analyses.

```
TP<-c()
FN<-c()
sens<-c()
set.seed(1234)

svm2.fit<- ksvm(Class~.-Time,data=train[nonfraud[sample(nonfraud,size=1776*10)]],type='one-svc')
```

The one-class SVM is good at correctly labeling fraud when it occurs, as demonstrated by a recall of 0.92. However, it is unable to clearly distinguish between outliers in legitimate transactions and fraudulent transactions (PPV=0.0078). This problem is expected since novelty detection looks for deviations from a 'bulk majority' NOT differences between distinct classes. With a PPV this low, the model could never be used reliably in practice.

```r
print('Model Performance on Test Set Data')
```

```
## [1] "Model Performance on Test Set Data"
```

```r
a<- which(test$Class=='1') #Fraud
b<- which(test$Class=='0') #Non-Fraud

svm2.test<-predict(svm2.fit,newdata=test,type='response')

TP.test<-length(which(svm2.test[a]==FALSE))
FN.test<-length(which(svm2.test[a]==TRUE))

TN.test<-length(which(svm2.test[b]==TRUE))
FP.test<-length(which(svm2.test[b]==FALSE))

Recall.test<-TP.test/(TP.test+FN.test)
PPV.test<-TP.test/(TP.test+FP.test)

data.table(data_set=c('test'),Recall=c(Recall.test), PPV=c(PPV.test))
```

```
##    data_set    Recall         PPV
## 1:     test 0.9285714 0.007742375
```