Modelling Credit Fraud detection through Random Forests

Samuel Vélez Arango

27/08/2019

Introduction

The Credit Card Fraud dataset contains transactions made by credit cards in September 2013 by European cardholder in a two-day period, during which 492 frauds occurred out of 284,807 transactions, making this dataset notably unbalanced, with these frauds accounting for 0.172% of the observations. The variables V1 through V28 are the result of a Principal Component analysis (PCA) transformation, which, due to confidentiality issues, don't carry further details on their information. The only other features are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount. The variable 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

To solve this exercise we will tune and employ Random Forests (Rborist package) through three different methods, but we will evaluate it using a 10% of the data set every time.:

1) Random Forests without further changes.
2) We will introduce class weights to our calculation of Random Forests.
3) The data will be partitioned to teach our Random Forest using a 50% fraud, 50% non-fraud subset.

In this dataset, 99.82% of observations are non-frauds, while only 0.172% are the actual class we are most interested in distinguishing. Two of our attempts will use a random sample of 90% of the data set, while the third one will extract the same number of frauds and non-frauds, like this:

Pre-processing

The following lines create our main data set and the balanced subset.

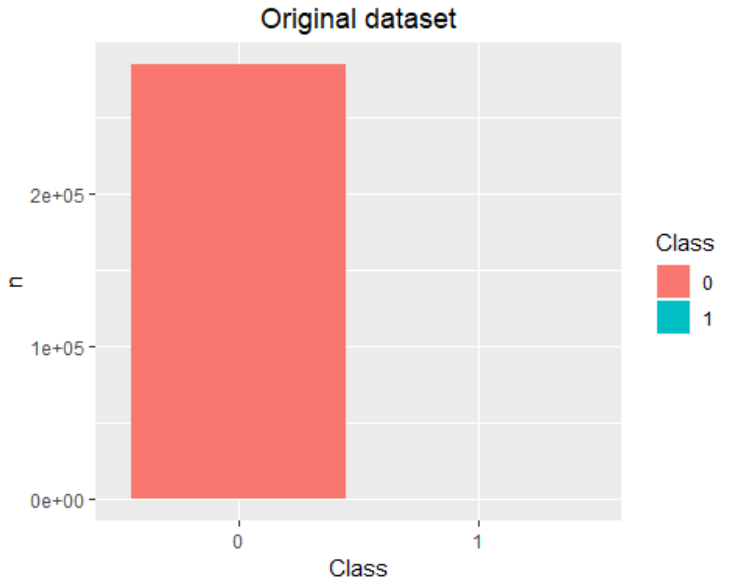
```
set.seed(1)
index_full <- sample(nrow(creditfraud), dim(creditfraud)*0.9)
x_full <- creditfraud[index_full,-c(31)]
y_full <- creditfraud$Class[index_full]

x_test_full <- creditfraud[-index_full,-c(31)]
y_test_full <- creditfraud$Class[-index_full]
```

```r
set.seed(1)
index_subset_x_nonfraud_0 <- filter(creditfraud,creditfraud$Class != "1")
index_subset_x_nonfraud_1 <-
sample(nrow(filter(creditfraud,creditfraud$Class == "1")),
dim(filter(creditfraud,creditfraud$Class == "1")))
index_subset_x_nonfraud <-
index_subset_x_nonfraud_0[index_subset_x_nonfraud_1,]

index_subset_x_fraud <- filter(creditfraud,creditfraud$Class == "1")
index_subset <- rbind(index_subset_x_nonfraud,index_subset_x_fraud)

x_subset <- index_subset[,-c(31)]
y_subset <- index_subset[,c(31)]
```

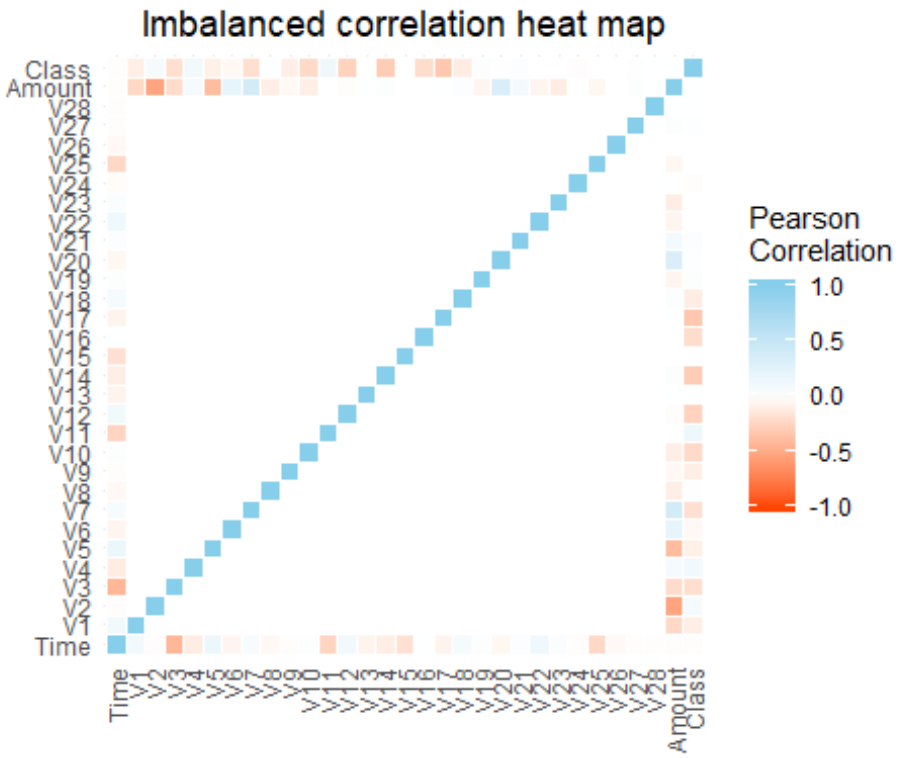


Correlation heat map

The following lines plot the correlation heat maps for both the full data set and the subsets with balanced data using ggplot2 and reshape2:

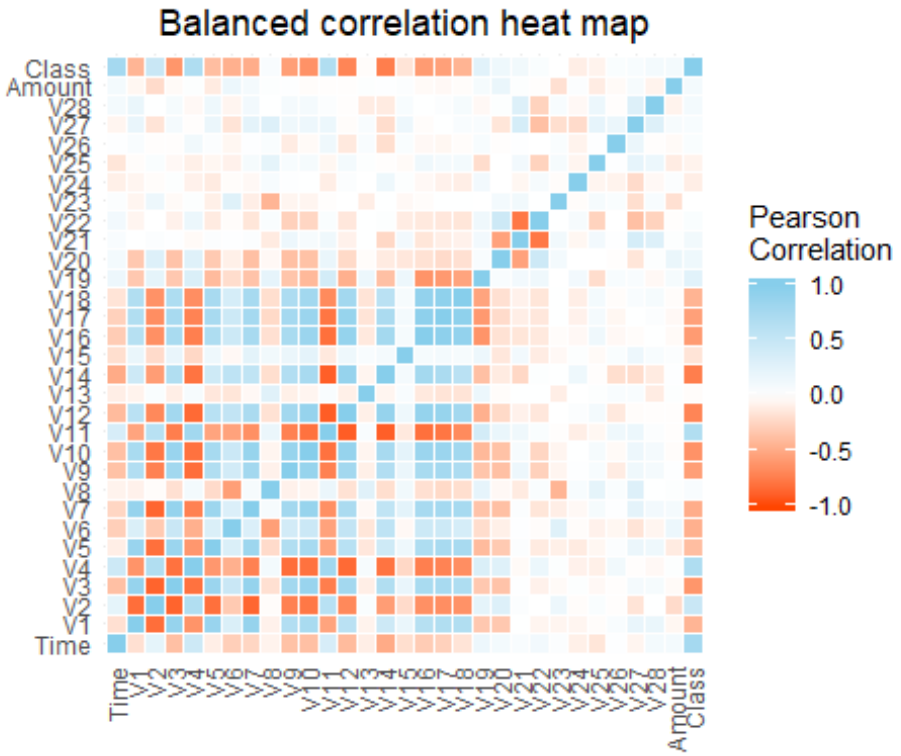
```
creditfraud$Class <- as.numeric(creditfraud$Class)
melted_cormat <- melt(round(cor(creditfraud),2))

ggplot(data = melted_cormat, aes(Var2, Var1, fill = value))+
  geom_tile(color = "white")+
  scale_fill_gradient2(low = "orangered", high = "sky blue", mid =
"white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5,
                                   size = 9, hjust = 1))+
  theme(plot.title = element_text(hjust = 0.5),
        axis.title.x = element_blank(),
        axis.title.y = element_blank()
        )+
  ggtitle("Imbalanced correlation heat map")+
  coord_fixed()
```




```
cormat_subset <- index_subset %>% mutate(Class =
as.numeric(index_subset$Class))
melted_cormat_subset <- melt(round(cor(cormat_subset),2))
```

```
ggplot(data = melted_cormat_subset, aes(Var2, Var1, fill = value))+
  geom_tile(color = "white")+
  scale_fill_gradient2(low = "orangered", high = "sky blue", mid =
"white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5,
                                   size = 9, hjust = 1))+
  theme(plot.title = element_text(hjust = 0.5),
        axis.title.x = element_blank(),
        axis.title.y = element_blank()
        )+
  ggtitle("Balanced correlation heat map")+
  coord_fixed()
```



As we can see, the balanced correlation heat map provides a much better visualization of what truly impacts a fraudulent transaction:

- Negative Correlations: The most significantly negatively correlated variables are V4 and V15, the lower they are, the higher the chances of a fraudulent transaction.
- Positive Correlations: V1, V5 and V5 are the most positively correlated variables; the higher they are, the higher the chances of a fraudulent transaction.

Modelling through Random Trees

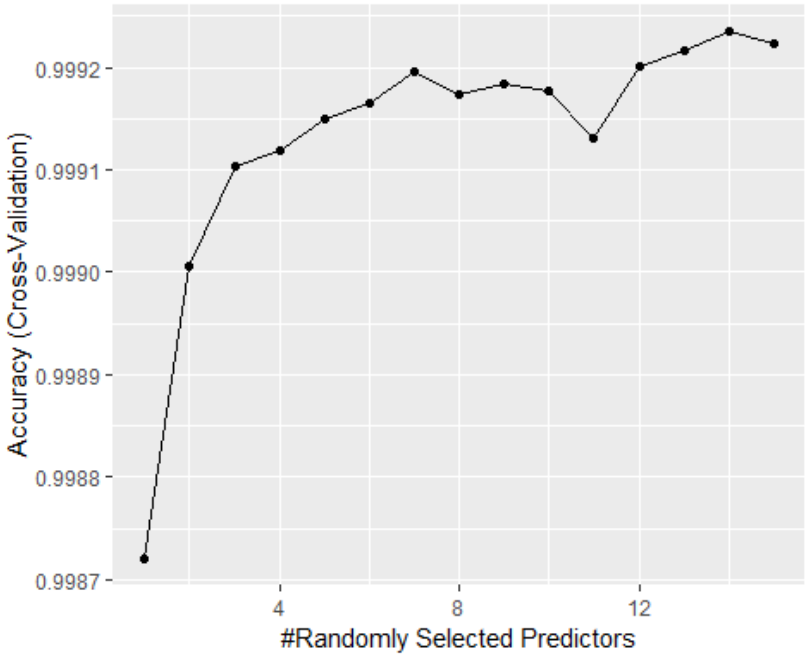
The following lines use the caret package to tune the first two random trees models, both using the full data set.

First, we will tune our predictors for both models:

```
control_set <- trainControl(method="cv", number = 5, p = 0.8)
tunegrid <- expand.grid(minNode = c(1) , predFixed = seq(1, 15, 1))

set.seed(1)
train_randforest_base <-  train(x_full,y_full,
                                method = "Rborist",
                                nTree = 50,
                                trControl = control_set,
                                tuneGrid = tunegrid,
                                nSamp = 5000)

ggplot(train_randforest_base)
```




```
train_randforest_base$bestTune

##    predFixed minNode
## 14        14       1
```

Random Forests using the imbalanced data set, not accounting for imbalance

Using the previous tuning, we train the first random trees model, not accounting for imbalance:

```
set.seed(1)
model_randforest_base <- Rborist(x_full, y_full,
                                 nTree = 500,
                                 minNode =
train_randforest_base$bestTune$minNode,
                                 predFixed =
train_randforest_base$bestTune$predFixed)

y_hat_rf_base <- factor(levels(y_full)[predict(model_randforest_base,
x_test_full)$yPred])
#factor(levels(y_full)[predict(model_randforest_base, x_test_full)])
confusionmatrix_base <- confusionMatrix(y_hat_rf_base, y_test_full)
confusionmatrix_base$overall["Accuracy"]

##  Accuracy
## 0.9996489
```

Weighted Random Forests using the imbalanced data set

And now Weighted random trees, giving more weight to the accurate prediction of fraudulent transactions using the classWeight component of the Rborist package:

```
set.seed(1)
model_randforest_weighted <- Rborist(x_full, y_full,
                                     nTree = 500,
                                     minNode =
train_randforest_base$bestTune$minNode,
                                     predFixed =
train_randforest_base$bestTune$predFixed,
                                     classWeight = c(50.0,1.0))

y_hat_rf_weighted <-
factor(levels(y_full)[predict(model_randforest_weighted,
x_test_full)$yPred])
confusionmatrix_weighted <- confusionMatrix(y_hat_rf_weighted,
y_test_full)
confusionmatrix_weighted$overall["Accuracy"]

##  Accuracy
## 0.9996138
```

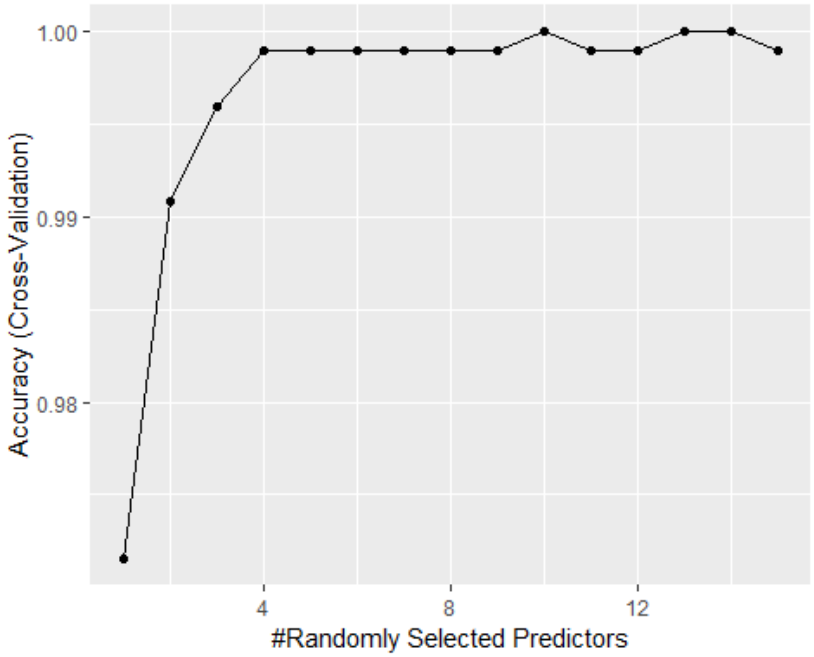
Random Forests using the balanced data set

Now we tune the model to the balanced data set, train it using the optimal number of predictors and then test it against the same train set as before.

```
set.seed(1)
control_set <- trainControl(method="cv", number = 5, p = 0.8)
tunegrid <- expand.grid(minNode = c(1) , predFixed = seq(1, 15, 1))

set.seed(1)
train_randforest_subset <-  train(x_subset,y_subset,
                                  method = "Rborist",
                                  nTree = 50,
                                  trControl = control_set,
                                  tuneGrid = tunegrid,
                                  nSamp = 5000)

ggplot(train_randforest_subset)
```




```
train_randforest_subset$bestTune

##    predFixed minNode
## 10        10       1
```

And now we create the random forest model using the balanced data set:

```
model_randforest_subset <- Rborist(x_subset, y_subset,
                                   nTree = 500,
                                   minNode =
train_randforest_subset$bestTune$minNode,
```

```
                                        predFixed =
train_randforest_subset$bestTune$predFixed
)

y_hat_rf_subset <- factor(levels(y_full)[predict(model_randforest_subset,
x_test_full)$yPred])

confusionmatrix_subset <- confusionMatrix(y_hat_rf_subset, y_test_full)
confusionmatrix_subset$overall["Accuracy"]

##    Accuracy
## 0.002879112
```

Results and conclusions

The following are the resulting confusion matrices:

Random Trees with no adjustments

```
confusionmatrix_base$overall["Accuracy"]

##  Accuracy
## 0.9996489

confusionmatrix_base$table

##            Reference
## Prediction     0     1
##          0 28441     7
##          1     3    30
```

Weighted Random Trees

```
confusionmatrix_weighted$overall["Accuracy"]

##  Accuracy
## 0.9996138

confusionmatrix_weighted$table

##            Reference
## Prediction     0     1
##          0 28440     7
##          1     4    30
```

Random Trees using a balanced subset

```
confusionmatrix_subset$overall["Accuracy"]

##    Accuracy
## 0.002879112

confusionmatrix_subset$table

##            Reference
## Prediction     0     1
##          0    45     0
##          1 28399    37
```

As we can see, accounting for the imbalance created a bias towards over classifying the fraudulent transactions, and the original Random Trees application outperfomed the others. While this method isn't the best approach to tackle this problem, Random Trees still delivered a very good approximation, and trying to balance the dataset provided very good insight on the interaction of our variables.