Credit Card Fraud Detection using RandomForest

Nitin Sinha

May 2, 2019

Introduction

Credit Card Fraud is an unfortunate reality in todays cashless world. It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

About the CreditCard Fraud Dataset

Credit Card Fraud Database can be downloaded from KAGGLE, one of the sites suggested by EDX - Credit Card DataSet. The dataset has been collected and analysed during a research collaboration of Worldline and the Machine Learning Group of ULB (Université Libre de Bruxelles) on big data mining and fraud detection.

The datasets contains transactions made by credit cards in September 2013 by european cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

Due to confidentiality issues, provider cannot provide the original features and more background information about the data. Features V1, V2, ... V28 are the principal components obtained with `Principal Component Analysis (PCA)`, the only features which have not been transformed with PCA 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, this feature can be used for example-dependant cost-senstive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

Goals of this Project

The Goal of this project is to:
- Detect most of the Fraudulent Transactions in the dataset using **Random Forest Machine Learning Algorithm**
- As cost of following up on suspected fradent transactions is high, it is important to have a **high degree of Accuracy**
- Also, it is important that a customer who has done a valid transaction should never be blamed for a fraudulent transaction, therefore **True Positive Rate (or Sensitivity)** of the prediction needs to be high; it does not matter if the True Negative Rate (or Specificity) is high, because even if some fradulent transactions escape the net impact is much lower than otherwise

Key Steps

- Step 1: Data was analyzed by plotting couple of charts that provided useful insight
- Step 2: Correlation between the various variables was determined to be minimum
- Step 3: Random Forest Algorithms is used to predict Fraud in the Credit Card Data set
- Step 4: Conclusions are drawn from the Result set

Analysis

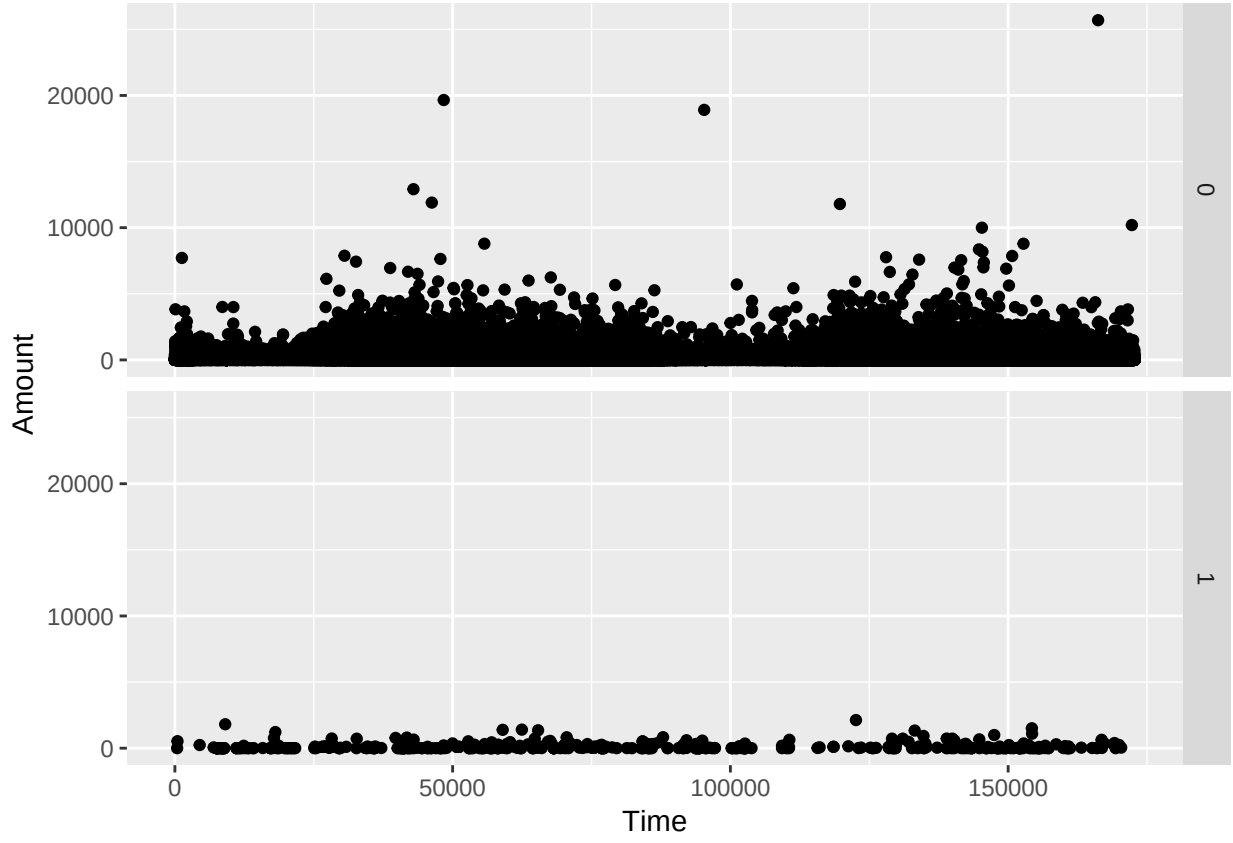
Data Exploration

There are a total of 31 columns in the data. One column, `Class` is the target value; it is a binary value, can have either `0` (not fraud) or `1` (fraud) value. Another two columns have clear meaning: `Amount` is the amount of the transaction; `Time` is the time of the transaction. The rest of the features (28), anonymized, are named from `V1` to `V28`.

The data is highly unbalanced with respect of `Class` variable values. There are only `0.1727486`% of the rows with value `Class = 1`. Typically, in such cases, we can either choose to preserve the data unbalancing or use a oversampling (of the data with minority value of target variable) or undersampling (of the data with majority value of the target variable). Here we will just preserve the unbalancing of the data. In terms of validation of the result, we will see that usual metric, using a confusion matrix or accuracy are not the most relevant. Rather, the metric that is key to teh success of this model is `Sensitivity`

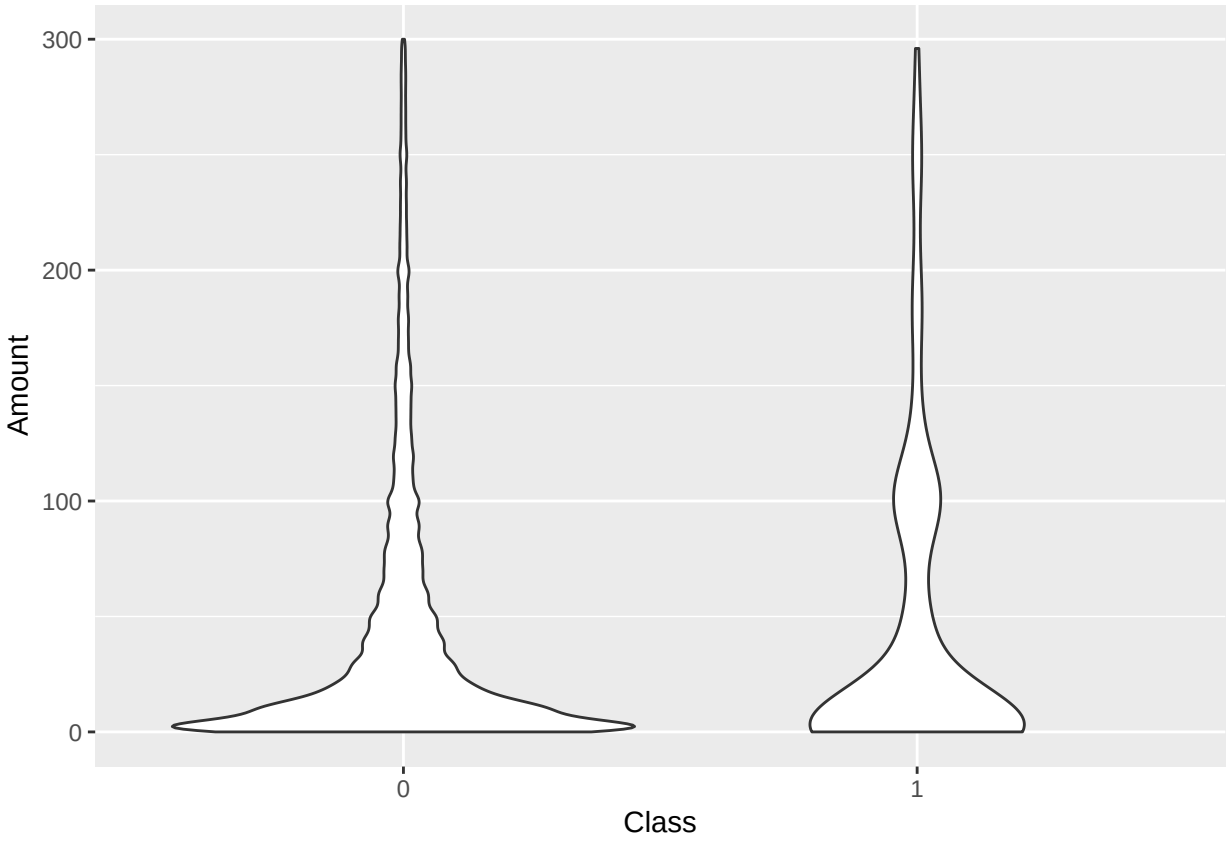
Data Visualization

Let's start by plotting all the $ Amounts for 'Non Fradulent' (0) and 'Fraudulent' (1) transactions seperately



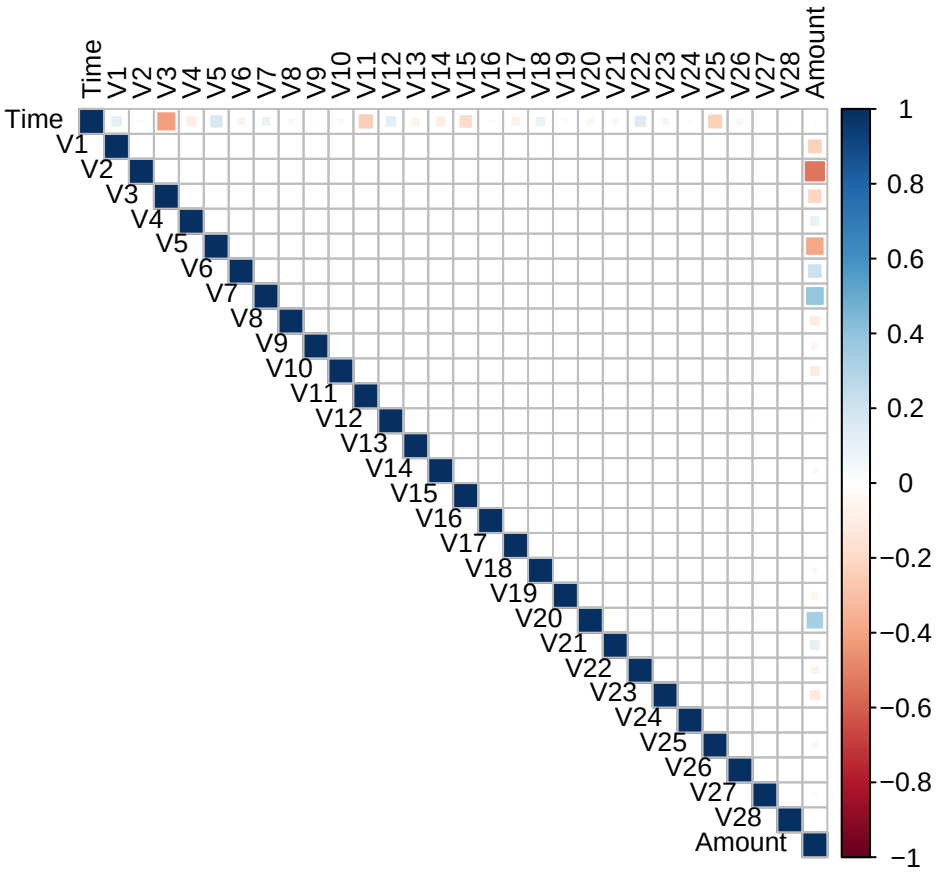
From the previous chart we see that Fraudlent Transactions are not of very high dollar values. This may be a deliberate attemp by the Fraudsters to keep these transactions 'under the radar'.

In the Chart below, I am filtering out transactions above $300, as they are not marked fraudulent and creating a `Violin Plot`. This kind of chart visually shows the most prelevant range of data.



From the Violin Chart, we see that the amount involved in fraudulent transactions seems more likely to be around $100. This is a useful insight that can be used while designing the algorithm.

Next we plot a `Correlation Plot` to visualize the correlation between all the variables. Note that as there are a large number of variables (Columns in input data) in this case it can take a while to Train the Random Forest Model.



We do not see any significant Correlation between variables, therefore we will use `Random Forest` as a `Classification` model rather than as `Regression` model (details below).

Results

Random Forest Algorithm

Random Forest Algorithm is one of the most polular Machine Learning Algorithm as it can be used for both `Classification` and `Regression` tasks. In simple terms Random Forest creates a `Forest` (an enseble of `Decision Trees`) that can be trained using the `Bagging` method.

In Random Forest, it is easy to measure the `Feature Importance` by looking at the tree nodes. Through looking at the feature importance, you can decide which features you may want to drop, because they don't contribute enough or nothing to the prediction process.

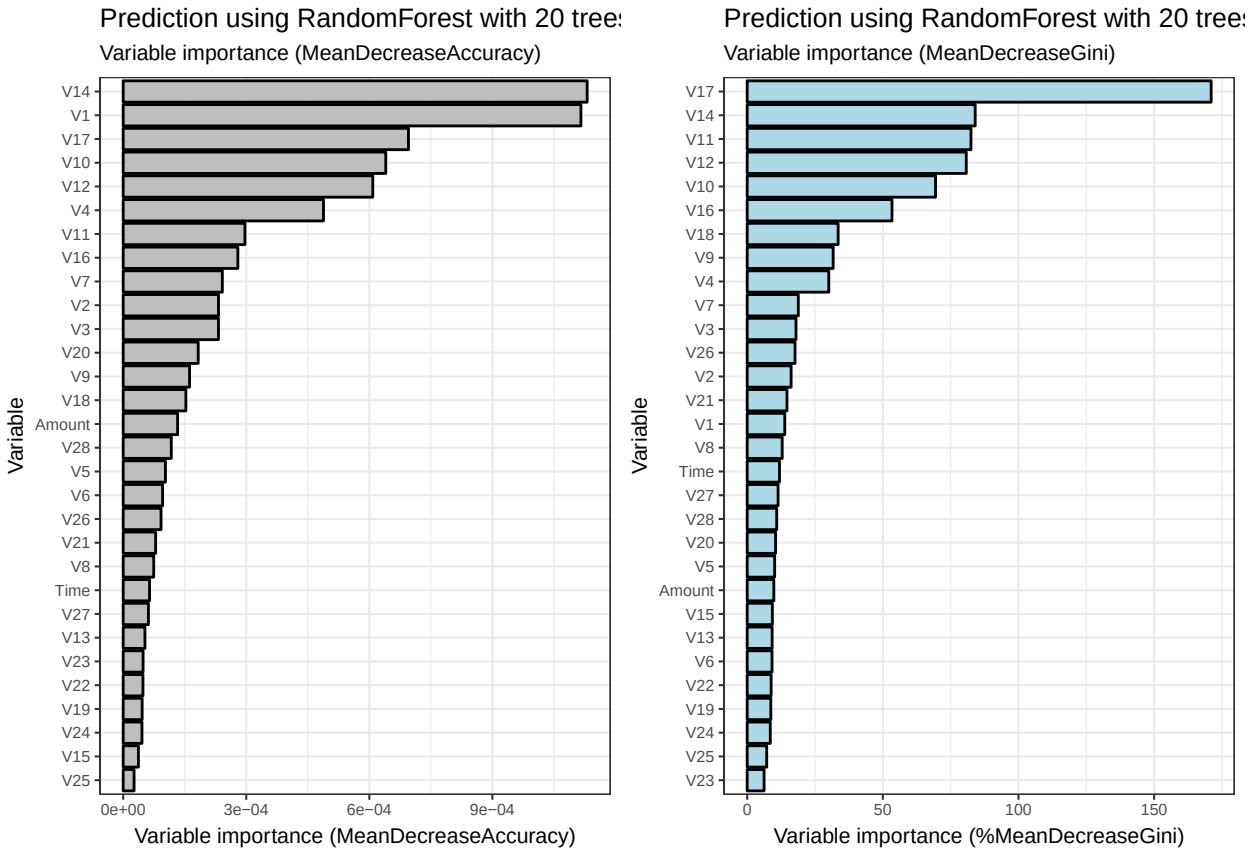
```
#Separate train and test sets of data
set.seed(1)
idxs <- sample(nrow(df), size = 0.1 * nrow(df))
train <- df[-idxs,]
test <- df[idxs,]
## Next we train the model using RandomForest
n <- names(train)
# Below code creates the formula using column names
rf.form <- as.formula(paste("Class ~", paste(n[!n %in% "Class"],
    collapse = " + ")))
set.seed(1)
```

```
rf <- randomForest(rf.form, train, ntree = 50, importance = T)
```

To determine which Features are important to Random Forest Classification, we plot the below chart using `MeanDecreaseAccuracy` and `MeanDecreaseGini` from the `Importance` feature data set provided by Random Forest.

`Gini impurity` is a metric used in Decision Trees to determine how (using which variable, and at what threshold) to split the data into smaller groups. Gini Impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. Gini Impurity reaches zero when all records in a group fall into a single category. This measure is essentially the probability of a new record being incorrectly classified at a given node in a Decision Tree, based on the training data. **A higher Mean Decrease in Gini indicates higher variable importance**

The `Mean Decrease in Accuracy` for a variable is determined during the out of bag error calculation phase. The more, the accuracy of the random forest decreases, due to the exclusion of a single variable, the more important that variable is deemed. Therefore **variables with a large mean decrease in accuracy are more important** for classification of the data.



We see that the following Variables are important for Classification using the Random Forest Algorithm: `V1, V4, V10, V11, V12, V14, V17`

Therefore in next cycle we can run the Random Tree with 100 trees with formula just using the above variables

```
rf.form = as.formula("Class ~ Time + V1 + V4 + V10 + V11 + V12 + V14 + V17 + Amount")
```

The machine used for running this code is limited in resources, and therefore forests were limited to `20 trees` only. This algorithm can be further enhanced by using the above formula and running for at least 100 trees in

a forest. Better still, an optimal number of trees for this dataset can be determined, to minimize the chances of this model giving `False Positives`, but that is for another day.

Now lets look at the **Accuracy, Sensitivity and Specifity** achieved by our model

```
## Accuracy: : 0.9995435
```

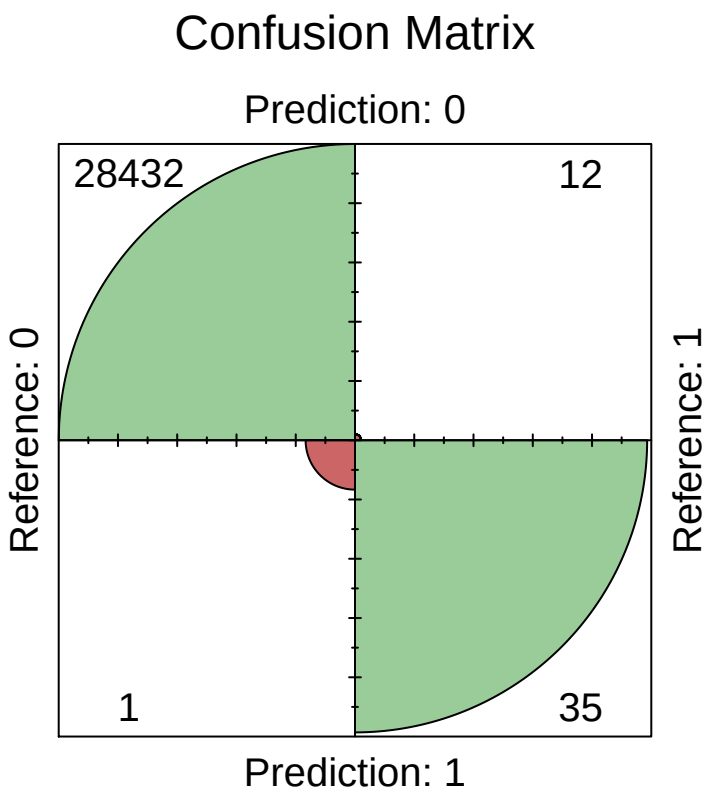
```
## Sensitivity:  0.9999648
```

```
## Specificity:  0.7446809
```

The Confusion Matrix

Confusion Matrix is a very important indicator of how well our Random Forest prediction model has done. Below I simplify the plotting of Confusion Matrix by using `Four Fold Plot` whihc is a great tool to plot `2 x 2` matrices. The 'good' is indicated in Green and 'not-so-Good' in Red.

```
fourfoldplot(conf_matrix$table, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1, main = "Confusion Matrix")
```



Conclusion

Following are key things to note from the Confusion Matrix:

```
## Total Number of Records in the Test Set : 28480
```

```
## Of these 47 are marked as Fraud
```

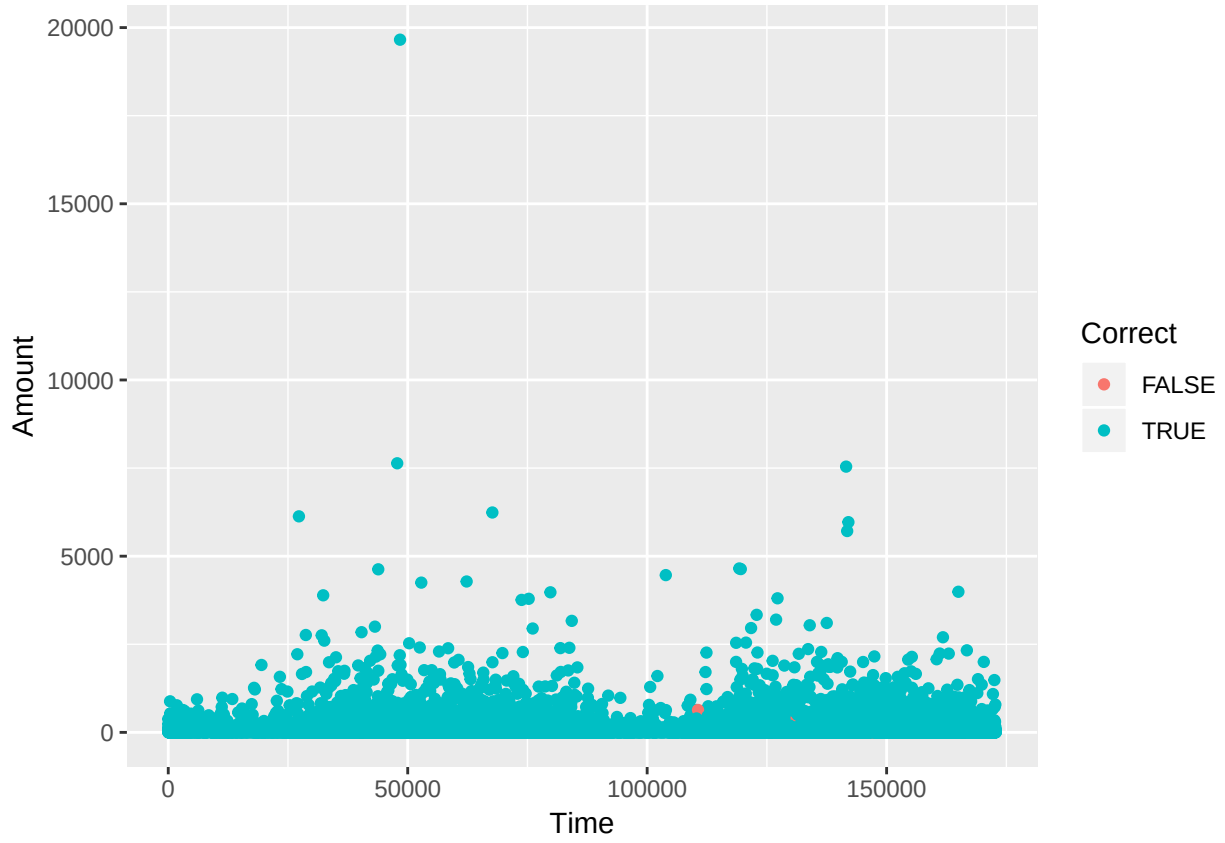
```
## Of the 47 Fraud cases, our model correctly predicts 35 as Fraud
```

```
## Our model does not detect 12 as Fraud. This is False Negative, but as mentioned before not so harmful

## Our model marks 1 as Fraud. This is False Positive

## Therefore Percentage of False Positive: 2.1 %. Not so bad eh!
```

As mentioned at the start of this report, it is important that a customer who has done a valid transaction should never be blamed for a fraudulent transaction, therefore **True Positive Rate(or Sensitivity)** of the prediction needs to be high. It does not matter if the **True Negative Rate (or Specificity)** is high, because even if some fradulent transactions escape the net impact is much lower than otherwise. Or in other words `An innocent should never be punished, even if some Guilty are let go`

We see from the above that our model has marked only `1` cases incorrectly (or **False Positives**). Let's plot the incorrect predictions against all the correct ones made by our model.

```
corr_incorr <- data.frame(Predicted = test$predicted, True = test$Class) %>%
    mutate(Correct = ifelse(Predicted == True, TRUE, FALSE)) %>%
    mutate(Time = test$Time, Amount = test$Amount)

ggplot(data = corr_incorr, aes(Time, Amount, col = Correct)) + geom_point()
```



As you can see, finding Fraudulent Credit Card transactions is like 'finding a needle in a hay stack', and this is with just 2 days of selected Credit Card Data. I think you will agree that our model has done really well!