Credit Card Fraud Prediction

Dave Maharaj

July 29, 2021

Executive Summary

The objective of this paper is to analyze credit card fraud data from Kaggle. The data contains just over 280 thousand records in total. These records are credit card transactions from September 2013 by European cardholders. The data has been transformed using PCA to protect the confidentiality of the cardholders.

We will complete exploratory analysis on the dataset to gain insights into the features and the relationships between these. The dataset is imbalanced since there is a small number of transactions marked as fraud. The analysis will need to account for this imbalance to allow us to predict with higher confidence.

Confidence will be measured using accuracy, sensitivity and specificity. Since this is a classification problem, that is, the transaction is fraudulent or not, these metrics will help us understand how good our algorithm performs. Our objective is to have high specificity, that is, we want to have high rates of correctly predicting the fraudulent transactions. This means that our false negative values (non-fraudulent transaction) should be minimized.

We will test a number of algorithms and conclude with the findings for each algorithm. We will use just over 120 thousand records for the analysis. This will include all the fraudulent transactions from the original dataset, plus 120 thousand randomly selected records from the non-fraudulent transactions. To validate our models, we will keep 20% of our data for testing. This set will not be used for training to simulate transactions that the model has not seen. We elected to go with 20% to have sufficient number of samples for training the algorithms.

Method/Analysis

Overview of dataset

The dataset contains 30 potential predictors and the outcome value. A summary of the data is below:

Column 1 contains the time value of the transaction. This value is defined as the time between each transaction. Column 2 to 30 have been encoded using principle component analysis (PCA). The final column contains the outcome.

Time	V1	V2	V3	V4	V5	V6	V7	V8	V9
0	-1.35981	-0.07278	2.53635	1.37816	-0.33832	0.46239	0.23960	0.09870	0.36379
0	1.19186	0.26615	0.16648	0.44815	0.06002	-0.08236	-0.07880	0.08510	-0.25543
1	-1.35835	-1.34016	1.77321	0.37978	-0.50320	1.80050	0.79146	0.24768	-1.51465
1	-0.96627	-0.18523	1.79299	-0.86329	-0.01031	1.24720	0.23761	0.37744	-1.38702
2	-1.15823	0.87774	1.54872	0.40303	-0.40719	0.09592	0.59294	-0.27053	0.81774
2	-0.42597	0.96052	1.14111	-0.16825	0.42099	-0.02973	0.47620	0.26031	-0.56867

Table continued..

V10	V11	V12	V13	V14	V15	V16	V17	V18	V19
0.09079	-0.55160	-0.61780	-0.99139	-0.31117	1.46818	-0.47040	0.20797	0.02579	0.40399
-0.16697	1.61273	1.06524	0.48910	-0.14377	0.63556	0.46392	-0.11480	-0.18336	-0.14578
0.20764	0.62450	0.06608	0.71729	-0.16595	2.34586	-2.89008	1.10997	-0.12136	-2.26186
-0.05495	-0.22649	0.17823	0.50776	-0.28792	-0.63142	-1.05965	-0.68409	1.96578	-1.23262
0.75307	-0.82284	0.53820	1.34585	-1.11967	0.17512	-0.45145	-0.23703	-0.03819	0.80349
-0.37141	1.34126	0.35989	-0.35809	-0.13713	0.51762	0.40173	-0.05813	0.06865	-0.03319

Table continued

V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0.25141	-	0.27784	-	0.06693	0.12854	-	0.13356	-	149.62	0
	0.01831		0.11047			0.18911		0.02105		
-	-	-	0.10129	-	0.16717	0.12589	-	0.01472	2.69	0
0.06908	0.22578	0.63867		0.33985			0.00898			
0.52498	0.24800	0.77168	0.90941	-	-	-	-	-	378.66	0
				0.68928	0.32764	0.13910	0.05535	0.05975		
-	-	0.00527	-	-	0.64738	-	0.06272	0.06146	123.50	0
0.20804	0.10830		0.19032	1.17558		0.22193				
0.40854	-	0.79828	-	0.14127	-	0.50229	0.21942	0.21515	69.99	0
	0.00943		0.13746		0.20601					
0.08497	-	-	-	-	-	0.10591	0.25384	0.08108	3.67	0
	0.20825	0.55982	0.02640	0.37143	0.23279					

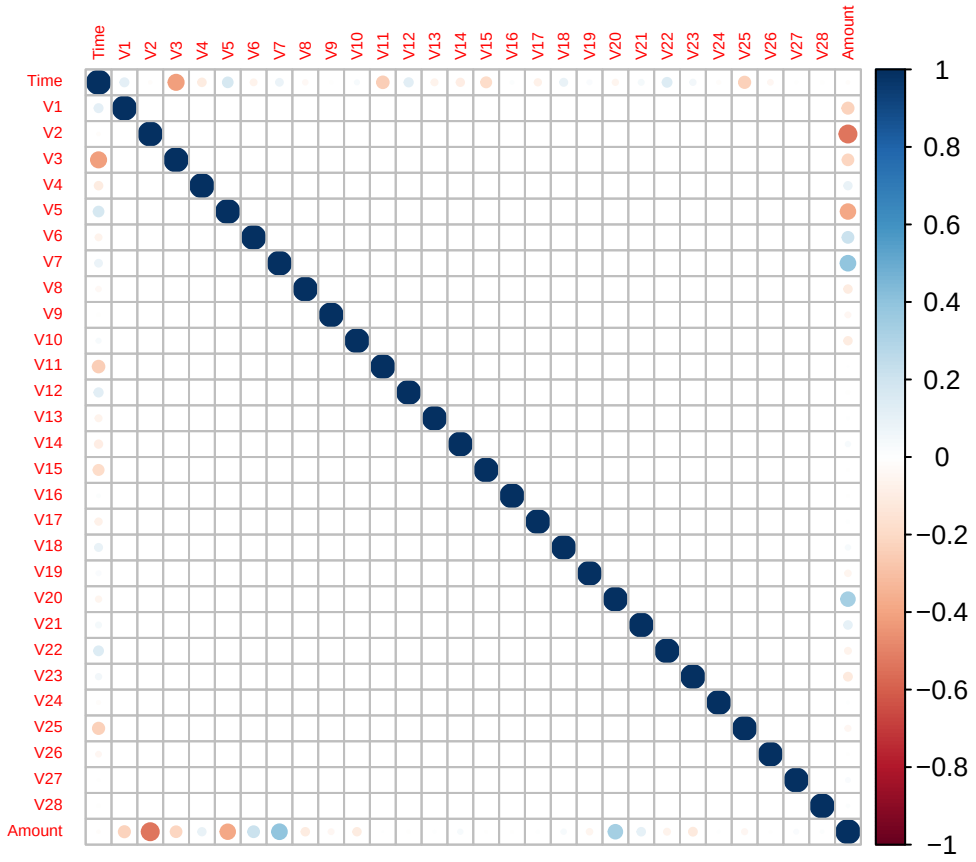
The dataset is imbalanced, meaning that the number of fraudulent transactions are very small compared to legitimate transactions. The table below provides a high level summary.

Class	RecordCount	Proportion
0	284315	0.998
1	492	0.002

Visual analysis and summary statistics

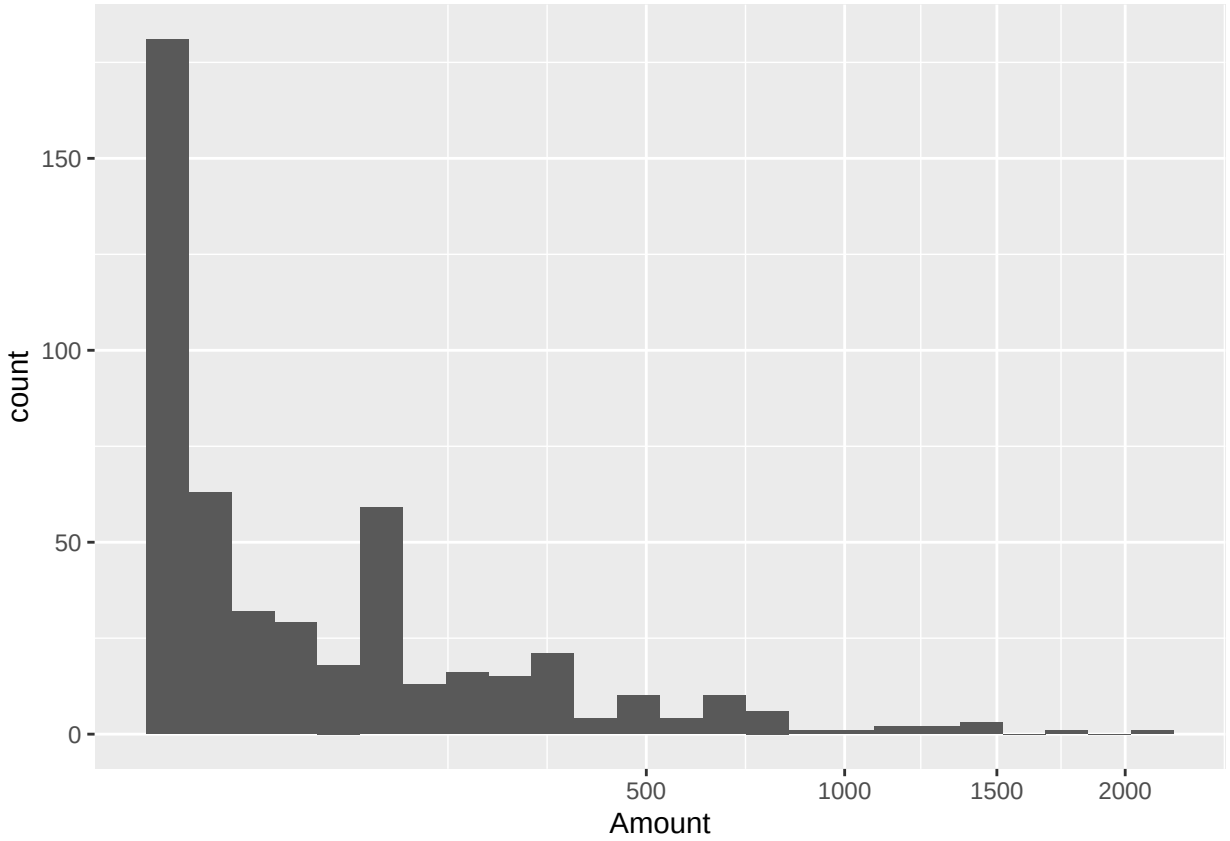
Below we explore the data visually and with summary statistics. This will help us gain a high level understanding of the dataset, which will help us define parameters for our model definitions.

Correlation Matrix: The plot below shows the correlation between the variables. In particular, you can see that a handful of variables are highly correlated to the amount (either negative or positive).



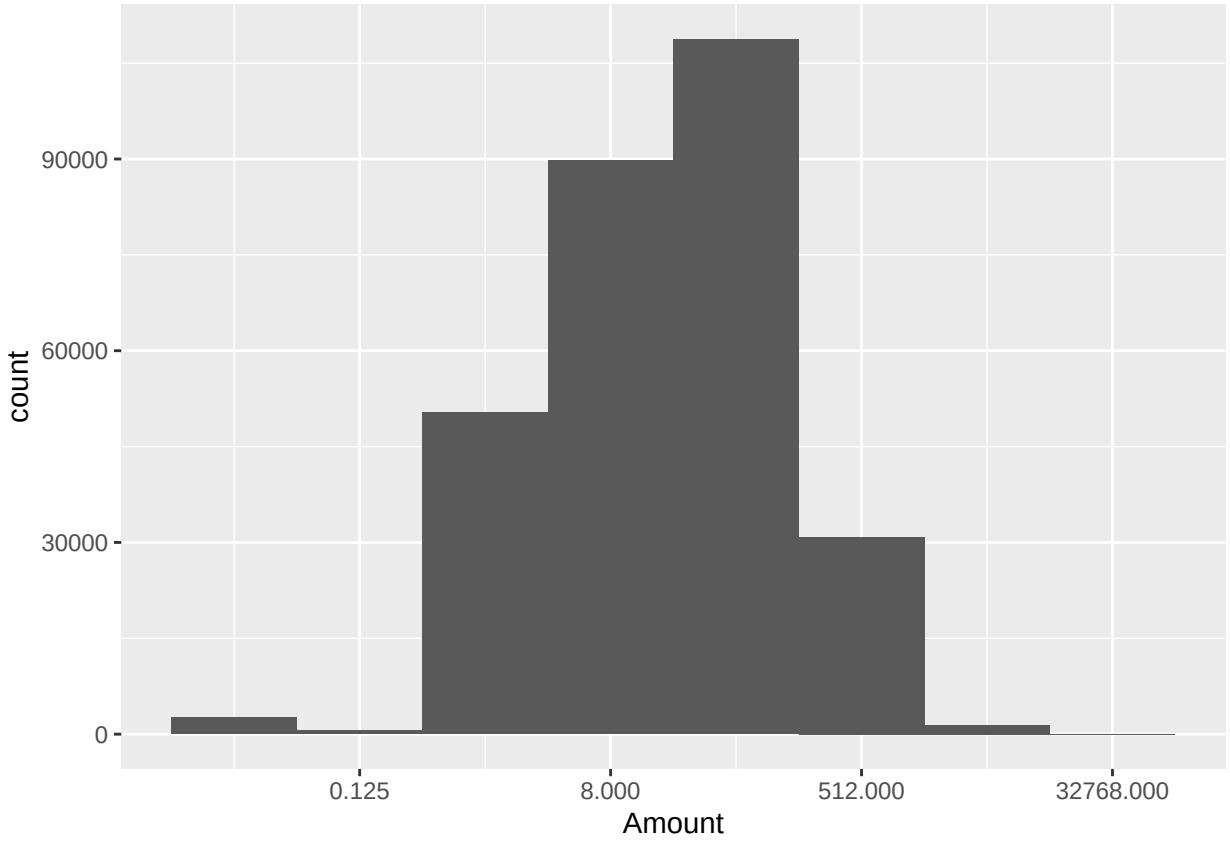
Distribution of fraudulent transactions:

This shows that most of the fraud transactions are under $1000.

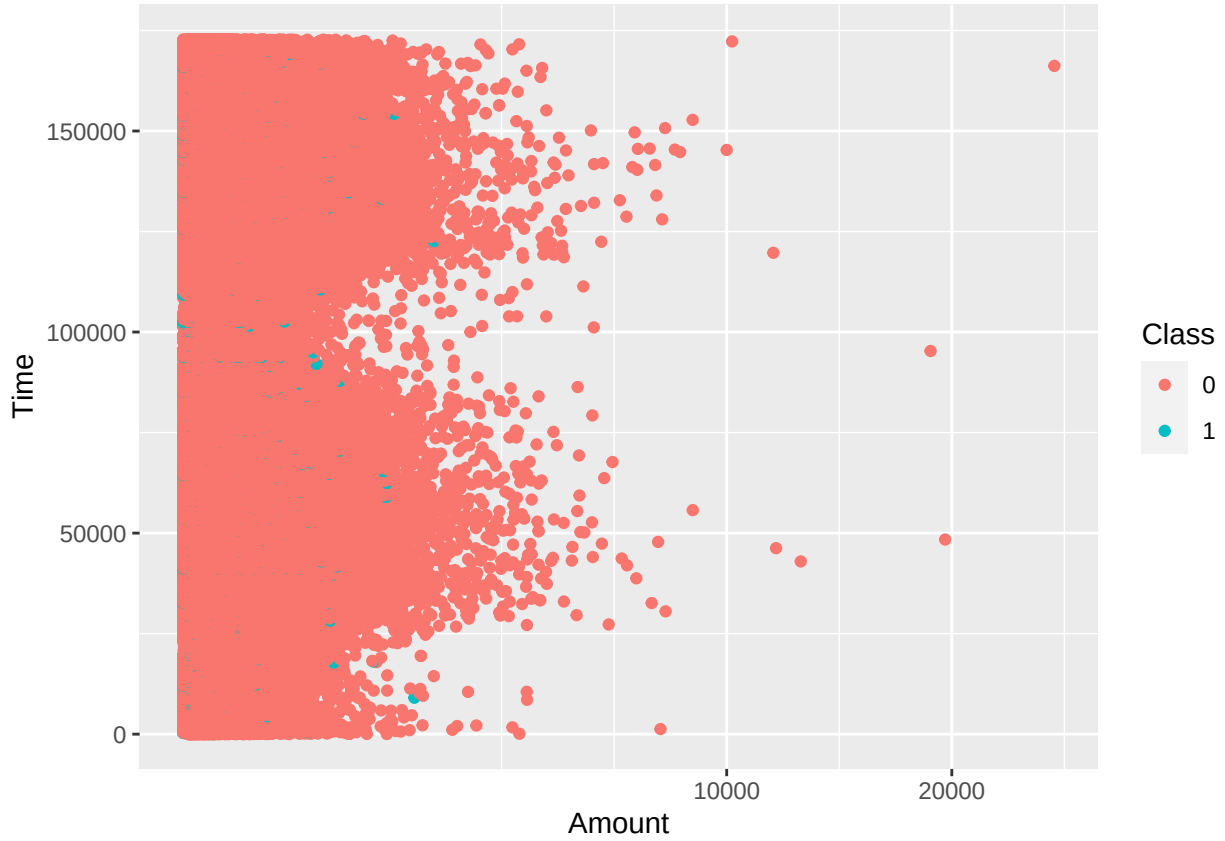


Distribution of non-fraudulent transactions:

The plot shows that non-fraud transactions has a wide range of amounts, from below 8 dollars to more than $20,000.



Transaction type by time: The plot below shows the transaction type by time. As you can see there is no clear pattern between time and the transaction amount by Class.



Statistical summary values

Below are the summary values for fraudulent transactions.

Mean_0	SD_0	Max_0	min_0
122.21	256.68	2125.9	0

Below are the summary values for non-fraudulent transactions.

Mean_0	SD_0	Max_0	min_0
88.291	250.11	25691	0

These statistic summary values shows that the mean is different by a number of points between both classes. The SD value is close between the classes and the Max is significantly different between both classes.

Pre-processing

This dataset is already processed well and does not necessarily require additional processing depending on the algorithm you elect to use. Since we do plan to use Support Vector machines as one of our algorithms, we will scale the data to center the values. In this project we scaled the data by subtracting the mean and then dividing by the standard deviation.

Modeling

Because the data is imbalanced, modeling will not yield accurate results with the data as is. We therefore need to look at doing one of the following:

Class weights - generate heavier cost when the minority class generate errors Up-sampling - create additional samples for the minority class Down-sampling - reduce the number of samples in the majority class.

We will try 3 different algorithms for modeling. Logistic regression, support vector machines and random forest. In order to allow the algorithms to run on a laptop, we have reduced the dataset by sampling 120 thousand non-fraudulent records and all the fraudulent records.

For all models we will standardize on the number of cross validation (cv) sets and the percentage of data to use for training. We will use 3 cv's and 90% of the data for training during cross validation. The code to implement this is below. Note that cross validation is not being used for the support vector machine algorithm.

Additionally, a confusion matrix was generated after predicting with the validation data to test each model.

```
train_params <- trainControl(method = 'cv', number = 3, p=0.9)
```

Logistic Regression - imbalanced data

Logistic regression models the probability of a class in the dataset. The algorithm uses the predictors to determine the probability.

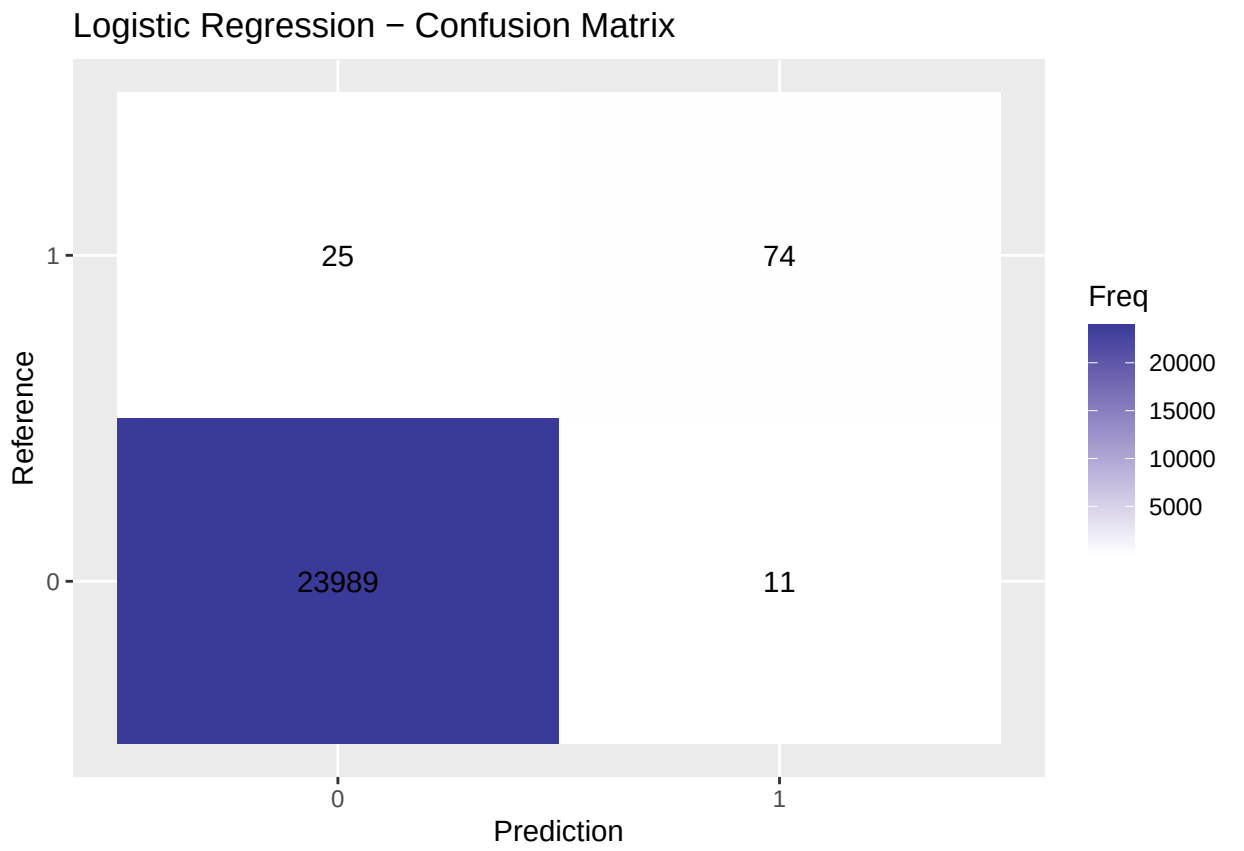
Before we use a method to address the imbalanced nature of the data, we will first look at the impact of having a significantly small number of records that are fraudulent versus not non-fraudulent. This will be done using logistic regression without any of the options above to address imbalanced data. The code for this is below.

```
# GLM model
glm_fit <- train(Class ~ ., data=training_set, method='glm',
                 trControl=train_params, family=quasibinomial, maxit=100)
glm_predictions <- predict(glm_fit, newdata=validation_set)
glm_cm <- confusionMatrix(glm_predictions, factor(validation_set$Class))
```

This model produced the results below. The results show that the specificity is less than 75%, this means that the model is predicting a number of false negatives (non-fraudulent transaction).The results also show a near 100% value for sensitivity since we are predicting almost all transactions as non-fraudulent.

Method	Sensitivity	Specificity	Accuracy
Logistic Regression	0.99954	0.74747	0.99851

The confusion matrix shows a summary of the predictions against the actual value.



Logistic Regression - class weights

The results above does not meet our objective, since we want our specificity to be higher. Let's adjust our model to include class weights. This approach will allow us to weigh the classes based on the number of records for that class in the dataset.

The code below does this for us.

```
# Use class weights given the data is imbalanced. Calculate the weight for each class
weights_0_1 <- training_set %>% group_by(Class) %>%
  summarise(counts = n()) %>% mutate(weight = (1/counts)*0.5) %>% pull(weight)
# create vector of training weights for each record in the dataset
training_weights <- training_set %>%
  summarise(weight = ifelse(Class == 0, weights_0_1[1], weights_0_1[2])) %>% pull(weight)
```

The model is now trained using the class weights using the code below.

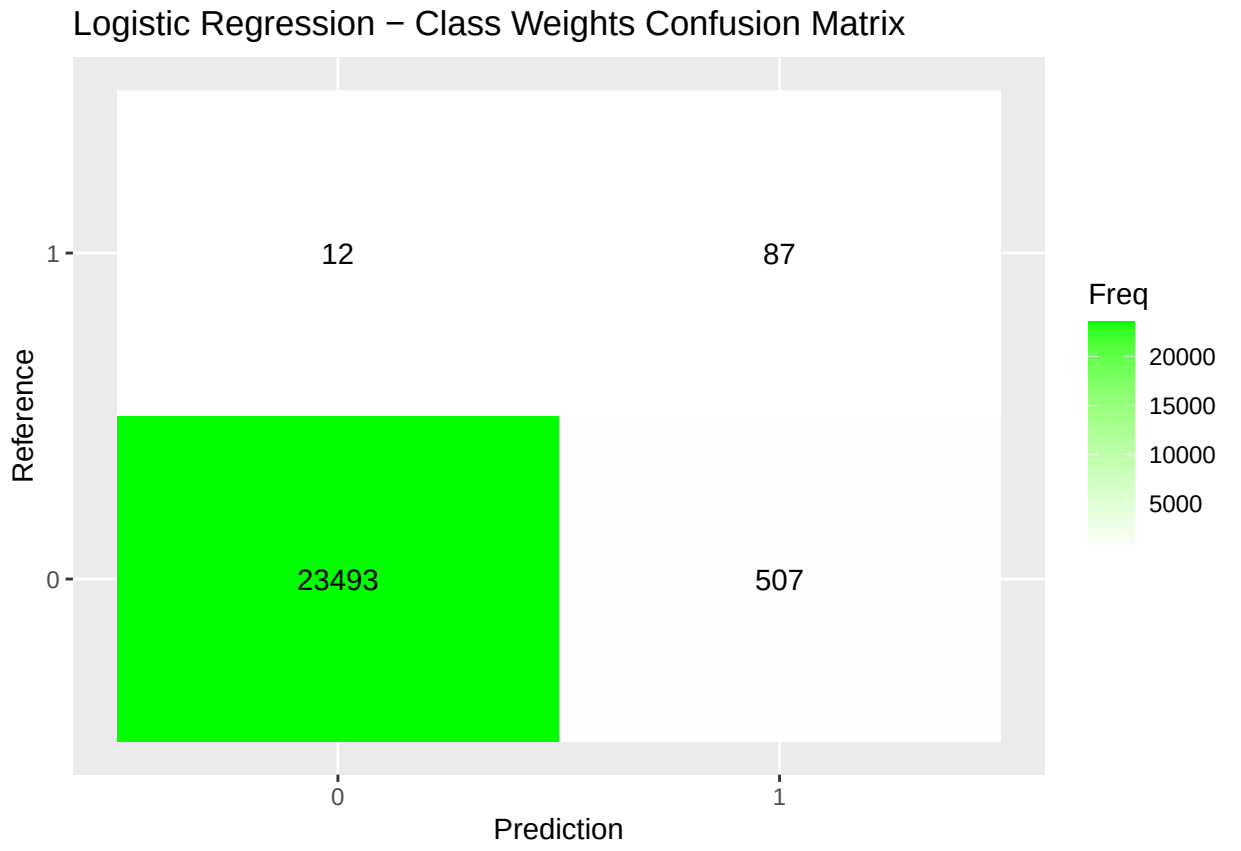
```
# GLM model with class weights
glm_fit_cl_weight <- train(Class ~ ., data=training_set, method='glm',
                           trControl=train_params, weights = training_weights,
                           family=quasibinomial, maxit=100)

glm_predictions_cl_weight <- predict(glm_fit_cl_weight, newdata=validation_set)
glm_cl_weight_cm <- confusionMatrix(glm_predictions_cl_weight, factor(validation_set$Class))
```

The logistic model with class weights produced the results below

Method	Sensitivity	Specificity	Accuracy
Logistic Regression - with class weights	0.97888	0.87879	0.97846

These results are much more favorable. The specificity has increased to 88% and our sensitivity is still over 97%. Note that sensitivity has decreased a little, this is normal, As specificity increases, sensitivity usually decreases.

A confusion matrix showing the predictions verses the actual values is in the plot below.



Support Vector Machines

We will now try another model to understand if we can do better. The model we will use is Support Vector Machines (SVM). SVM is used to classify the data. As it does the classification process, the algorithm also defines a boundary that separates the classes.

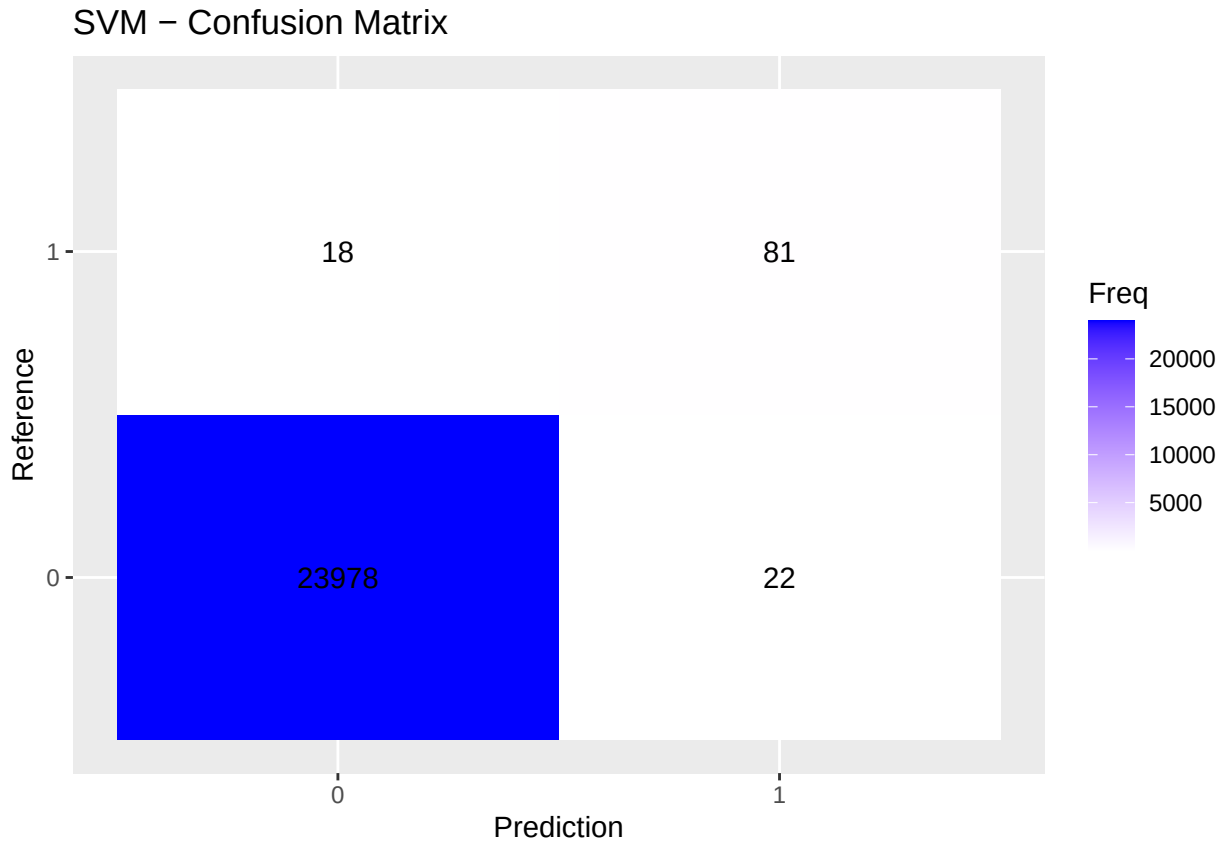
Class weights have been used with the SVM algorithm address imbalanced data.The code below was used to execute the model.

```
# SVM Model
svm_fit <- svm(Class ~ ., data=training_set,
               class.weights=c("0"=weights_0_1[1], "1"=weights_0_1[2]), kernel='linear')
svm_predictions <- predict(svm_fit, newdata=validation_set)
svm_cm <- confusionMatrix(svm_predictions, factor(validation_set$Class))
```

After testing with the validation data, we got the results below. The specificity was over 81% and our sensitivity 99.9%.

Method	Sensitivity	Specificity	Accuracy
SVM - with class weights	0.99908	0.81818	0.99834

Confusion matrix with the results of testing is below.



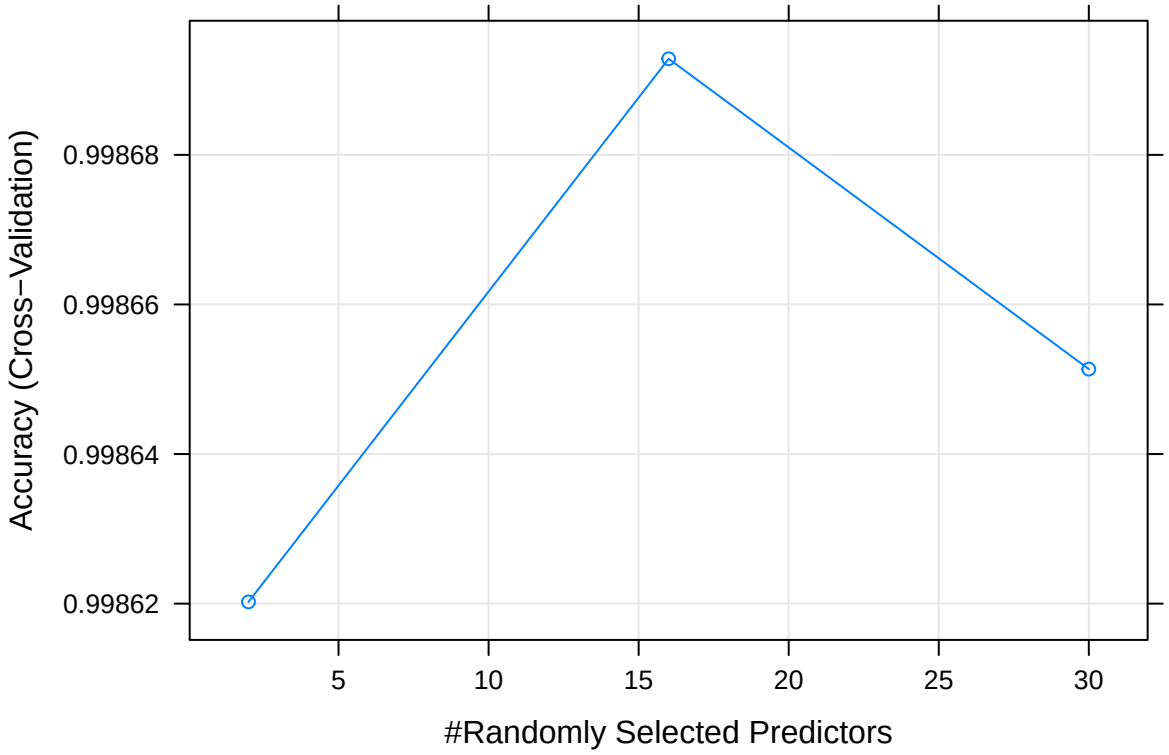
Random Forest

We will now try the random forest algorithm. This algorithm builds decision forest for classification. Multiple decision trees are generated and then the algorithm selects the tree that is most common in the forest as the final output. The random forest algorithm does not benefit from feature scaling so we did not use the scaled version of the data.

This model was used with class weights on the dataset. The code for modeling with random forest is below.

```
# random forest
# results of running the model showed an mtry value of 16 as optimal
rf_fit <- train(Class ~ ., data=training_set_rf, method='rf',
                classwt=c("0"=weights_0_1[1], "1"=weights_0_1[2]),
                trControl=train_params)

rf_predictions <- predict(rf_fit, newdata = validation_set_rf)
rf_cm <- confusionMatrix(rf_predictions, factor(validation_set_rf$Class))
```

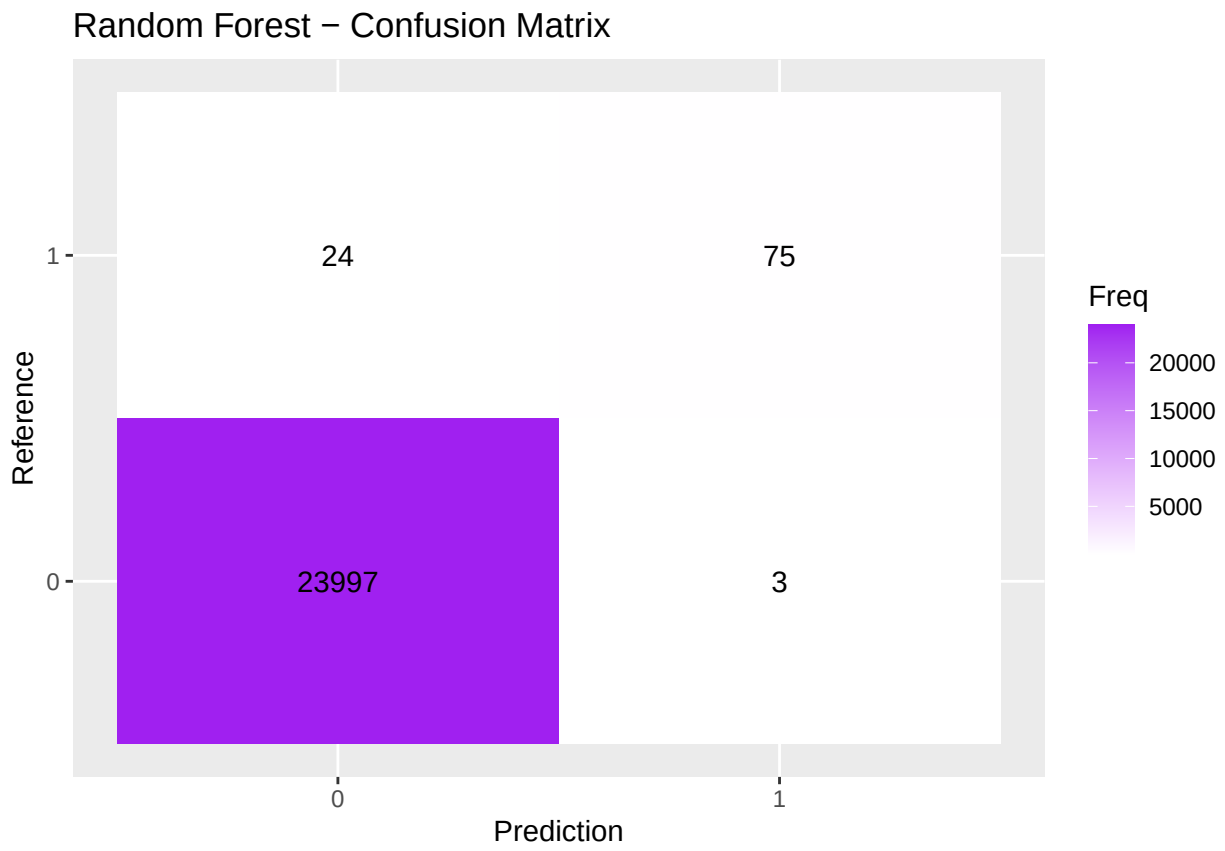
The algorithm landed on 16 predictors, the plot below shows this after running the learning process.



The results of this model can be found in the table below. The results produced a low specificity of less than 76% and a perfect sensitivity of 100%. This means that we classified more than 24% of the fraudulent transactions as non-fraudulent, which does not meet our objective.

Method	Sensitivity	Specificity	Accuracy
Random Forest - with class weights	0.99987	0.75758	0.99888

Confusion matrix with the results of the random forest algorithm.



Conclusion

It is important to note that the analysis was done on a subset of the data. With more data and additional parameter turning, there is a possibility of getting better results. Using cloud resources to run the analysis will allow you to efficiently model using complex algorithms on the entire dataset. This is a good next step. Additionally, using neural networks (NN) on the dataset may further improve the results since NNs can learn complex patterns in the dataset.

The table below summarize the results of our findings.

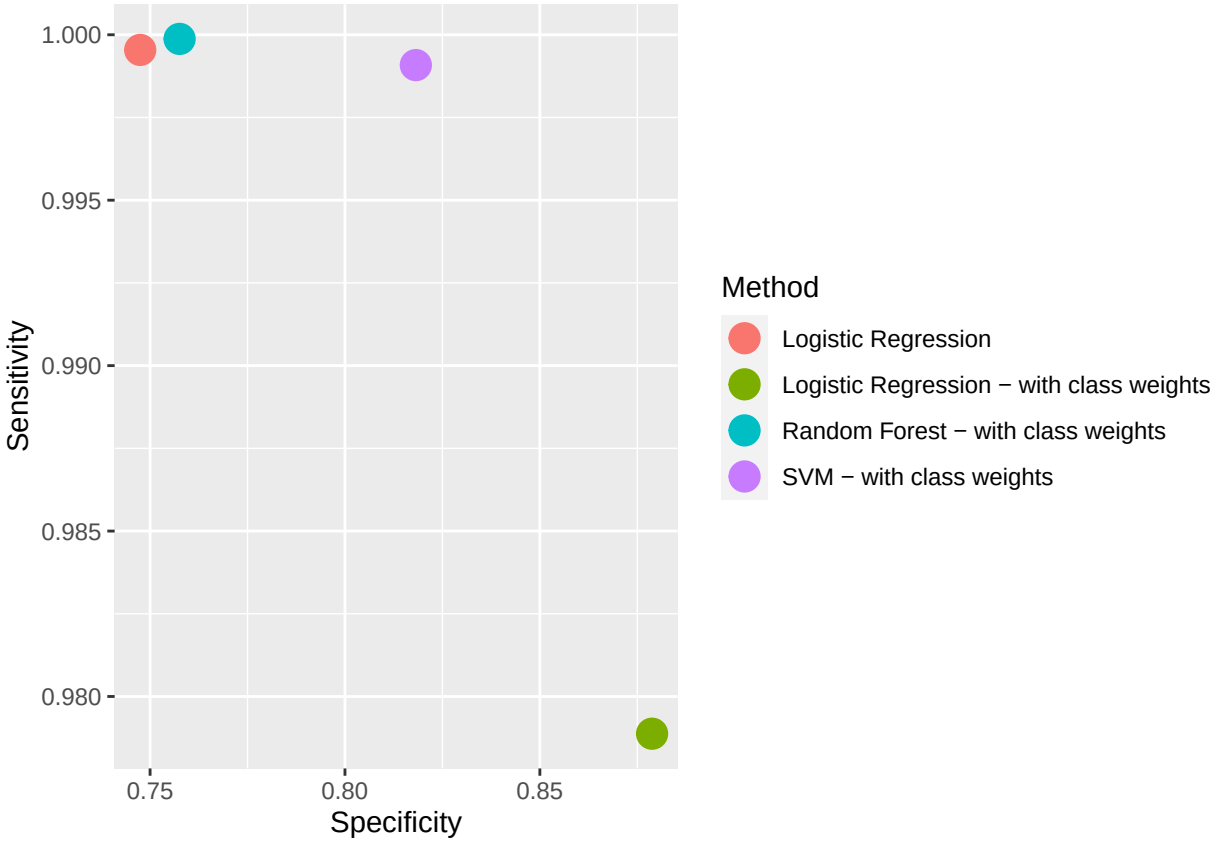
Method	Sensitivity	Specificity	Accuracy
Logistic Regression	0.99954	0.74747	0.99851
Logistic Regression - with class weights	0.97888	0.87879	0.97846
SVM - with class weights	0.99908	0.81818	0.99834
Random Forest - with class weights	0.99987	0.75758	0.99888

Our results show that both logistic regression and support vector machines with class weights produced favorable results. This gave us over 87% and 81% specificity respectively, which means we correctly identified more than 80% of the fraudulent transactions using these two algorithms. Accuracy on the SVM algorithm is over 99%, while the logistic regression (with class weights) produced an accuracy of close to 98%. Random Forest produced less than 76% on specificity with an overall accuracy of over 99%.

All the models produced significantly high accuracy, this was mainly due to predicting the non-fraud transaction correctly in over 97% of the cases. If we had more data on non-fraud transactions, there is a possibility

that we could get better results on the fraud transactions predictions. One approach to try is up-sampling to address the imbalanced data. We did not try that in this project as it does take a while to train on a laptop. Up-sampling will create additional entries for fraudulent transactions, essentially increasing the overall dataset during training. This approach for all 3 models may produce better results. Given the performance on logistic regression and support vector machines, it may be more beneficial to focus on one of those algorithms to further improve the strength of predictions.

The scatter plot below visualizes sensitivity and specificity for the models.



References

Link to Kaggle website where further details can be found on the database used in this analysis is below.

https://www.kaggle.com/mlg-ulb/creditcardfraud