ANOMALY DETECTION:

CLASSIFYING CREDIT CARD TRANSACTIONS

Nathan Wong

June 2019

INTRODUCTION

This project examines data associated with legitimate and fraudulent credit card transactions. The purpose is to identify indicators that are associated with fraudulent cases and to build a model that recognizes fraudulent credit card transactions.

The data is two days of transactions made by European cardholders in September 2013. There are 492 fraud cases out of 284,807 transactions (0.172%). Due to confidentiality issues, 28 variables associated with each transaction were transformed with a principal component analysis and had their names removed; the two predictor variables that did not have their names removed are ‘Time’ and ‘Amount’. The variable ‘Class’ determines whether a transaction is legitimate or fraudulent: 0 is legitimate and 1 is fraudulent.

More information can be found here:

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

METHODS

Dataset

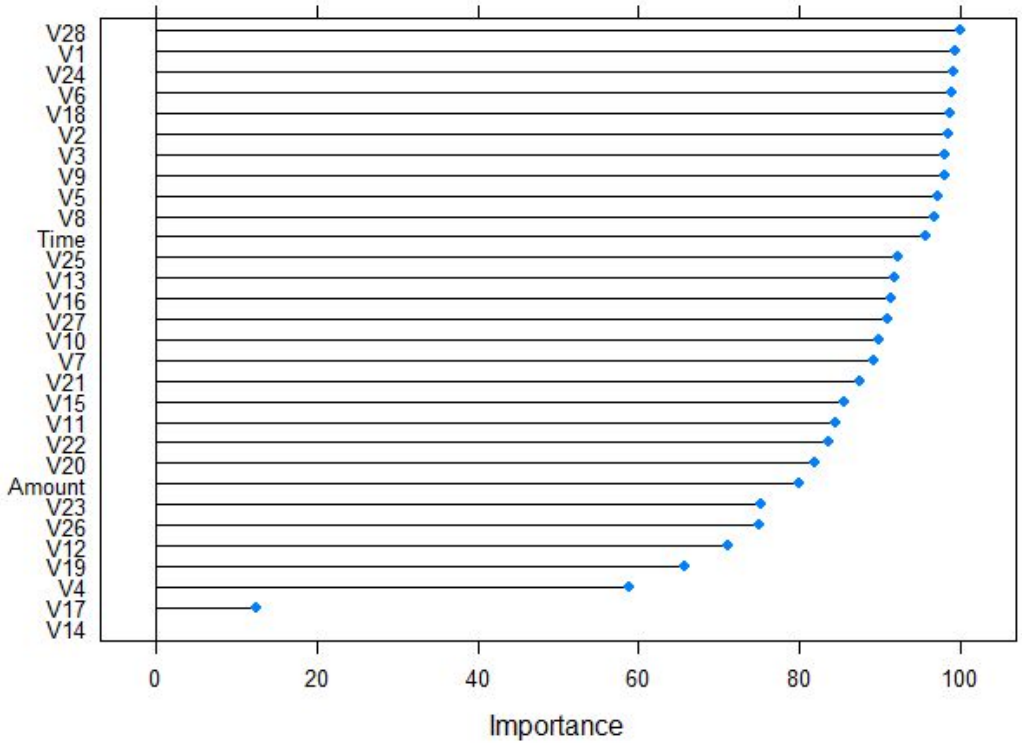
This dataset was already very clean; there were no NA values. After examining the structure of the data, I converted 'Class' to a factor.

Model

In order to have a reproducible model, I set the seed to '123'. I partitioned the data into two groups: a training group (80%) and a testing group (20%). To make the model more robust, I used a 5 fold cross-validation scheme. The 5 fold cross-validation scheme divides the training group into 5 iterations, and the model improves after trying each iteration. I tested a model using a 10 fold cross-validation scheme, but it made little difference in the final prediction.

In the package 'caret', I used the method 'ranger', which is a fast implementation of random forests. To balance the data when sampling, I set class weights to 0.1 for legitimate cases and 0.9 for fraudulent cases. Previously, I set class weights to 0.3 and 0.7, but it made little difference in my predictions. After trying various number of tree combinations (e.g. num.trees = 4, 20, 50, 101, 200, 501), I chose 101 because of its size, stability, and speed. To examine the importance of each variable in the model, I set 'importance' to "impurity" and plotted the results (see Figure 1).

Figure 1: Importance of each variable in the model



When the model was applied to the testing data, the model correctly identified 86 cases of fraud (true positive), and 56,845 cases of no fraud (true negative). There were 25 instances where there was actually fraud, and the model predicted no fraud (false negative); there were 6 instances where there was actually no fraud, and the model flagged the transaction as fraudulent (false positive). See Table 1 and Table 2 for more information.

Table 1: Predicted vs. Actual results with the testing data

	Actual Legitimate	**Actual Fraud**
Predicted Legitimate	56,845 True Negative (TN)	25 False Negative (FN)
Predicted Fraud	6 False Positive (FP)	86 True Positive (TP)

Table 2: Sensitivity, Specificity, Precision, and Recall

Sensitivity	TP / (TP + FN)	0.775
Specificity	TN / (TN + FP)	0.999
Precision	TP / (TP + FP)	0.935
Recall	TP / (TP + FN)	0.775