Cyber Data Analytics
Assignment 1

Ahmet Güdek (4307445)
Takang Kajikaw Etta Tabe (4373758)

May 13, 2018

Introduction

To be able to safely and trustingly perform online payments, it is important to be able to identify fraudulent transactions. But identifying fraudulent transactions in a sea of valid ones is a difficult task, mainly due to two reasons. First, all transactions look alike with little to distinguish fraudulent transactions from valid ones. And second, the number of fraudulent data points is extremely small in comparison to the entire set of transaction data. This asymmetry in class probabilities makes it diffifult to create robust and reliable classifiers, leading to high false negative rates. In this report we will explore data sampling and manipulation methods to improve the data imbalance and create classifiers building upon this. For this, we will use real transaction data with anonymised data from a bank in Mexico.

For this project we used KNIME, R and several python libraries. To be able to reproduce our results, the following need to be installed.

- KNIME (instructions on `https://www.knime.com/installation-0`)
- R and its pROC package (instructions on `https://www.r-project.org/`)
- Python3 with libraries Scikit learn, NumPy, Pandas, Seaborn and SciPy (libraries can be installed via pip)

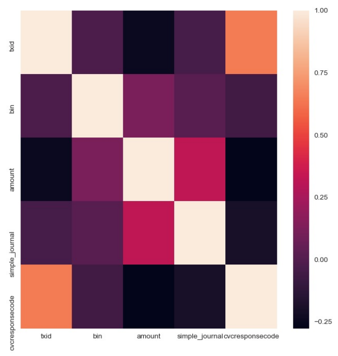
All code used in this assignment can be found on `https://github.com/Ettatabe/CyberDataAnalytics`

Finding patterns

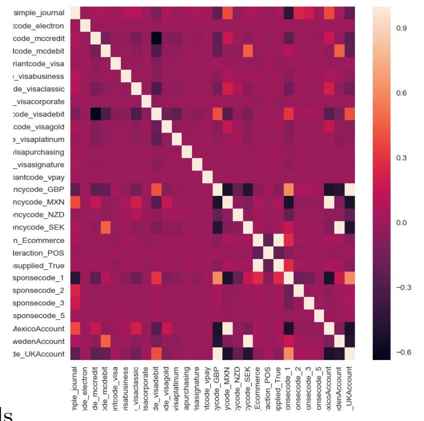
Our dataset consists of several columns with information on single transactions. dataset contains information such as the creation date, country, amount and currency, and the shopper's interactions for every transaction and provides information about the shopper such as their email_id, card_id and ip address. The simple_journal column contains our class labels in the form of *Settled*, *Refused*, *Chargeback* for valid, refused and fraudulent transactions respectively. As the exact fraud status of *Refused* transactions are unknown, we will no longer consider these transactions and all operations from here onward, except where mentioned, has these transactions filtered out.

To better understand the importance and interaction between these values, we attempted to create visualisations depicting this.

We see in the heatmaps in figure 1 that there is some correlation between the classes and transaction amounts. Furthermore, relatively strong correlation



(a) Heatmap of integer values against labels



(b) Heatmap of string values against labels

Figure 1: Data interaction visualisation

is found between the class labels and the cvcresponsecode and Mexican account code and currency. We can now build our classifiers depending on several of these values.

Balancing data

As mentioned earlier, a large part of the problem with classifying credit card fraud is the dataset sizes being skewed. In our case, the non-fraudulent cases dominate over the fraudulent ones in a 236691 to 345 ratio. To alleviate this, several methods exist that either grow the size of the minority class or shrink the majority class. We performed 3 such operations on their own and combined, and compared their effects on the classification with a random forest classifier. The methods we used are SMOTE, oversampling and undersampling. With SMOTE we generated new data points for the minority class. Oversampling was performed on the minority class to give the majority class additional weight in the learning phase. Finally, we performed undersampling on the majority class to use a similar number of data points as the minority class during learning. The results can be found in table 1.

From these results we can see that balancing methods severely affect the classification performances of classifiers with unbalanced datasets. While performing no balancing causes misclassification of almost all fraudulent records, with the above methods over 80% of accuracy is obtained. This does increase the number of false negatives, however, and should therefore be used carefully. If false negatives are as dangerous as true negatives, these methods should not be used. We also see that oversampling performes the worst of these methods, while undersampling seems to be the clear winner. However, the bad perfor-

Table 1: Random forest classification results of balancing

Random Forest		No special sampling		Oversampling Chargeback		Undersampling Settled	
Label	Prediction	No SMOTE	SMOTE	No SMOTE	SMOTE	No SMOTE	SMOTE
Chargeback	Chargeback	0	72	1	72	74	78
Chargeback	Settled	86	14	85	14	12	8
Settled	Chargeback	0	5429	0	5483	5149	7841
Settled	Settled	59173	53744	59173	53690	54024	51332

mance of oversampling is most probably caused by not oversampling the data enough times such that the datasets are still skewed. To find out whether these results are generic enough to use on other classifiers, we performed them also with decision tree and nearest neighbour classifiers and averaged the results. These can be found in table 2 below. Also the ROC analyses can be found in figure 2. From these results we can conclude that undersampling outperforms the other methods.

Table 2: Average classification results of balancing

		No special sampling		Oversampling Chargeback		Undersampling Settled	
Label	Prediction	No SMOTE	SMOTE	No SMOTE	SMOTE	No SMOTE	SMOTE
Chargeback	Chargeback	1.67	46.67	4.33	44.33	72.00	75.67
Chargeback	Settled	84.33	39.67	81.67	41.67	14.00	10.33
Settled	Chargeback	4.00	3232.67	50.33	3048.33	5821.33	7815.67
Settled	Settled	59169.00	55940.33	59122.67	56124.67	53018.33	51357.33

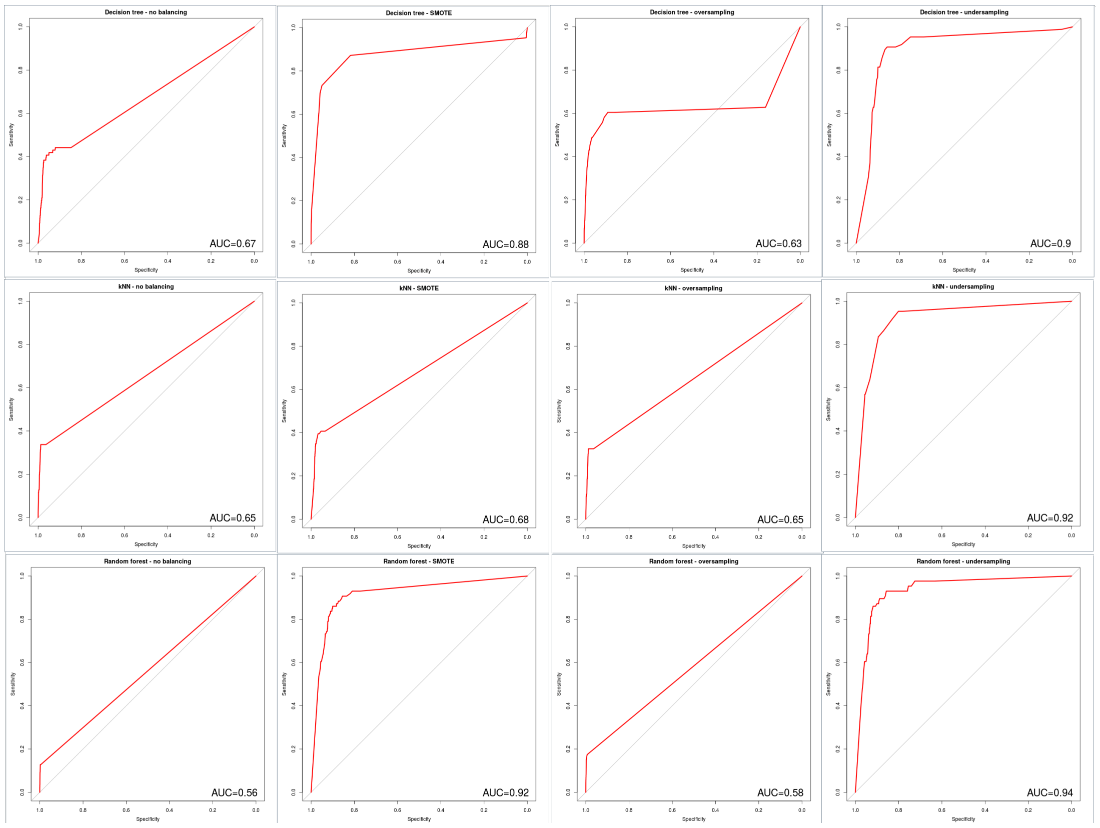
Building classifiers

White-Box Algorithm

In order to classify transactions as either fraudulent or bening, we applied a logistic regression model.

First of all, data was read in using pandas library. The first thing we did was look at the number of columns, rows and the types of the data entries in the data file. Since we are interested in just chargedback or settled transactions, we map those to 1 and 0 respectively, discarding refused transactions. For proper analysis, our classifier needs to work with integer or float types so we went on to discard the ”card”, ”ip” and ”email” strings in those columns so just integers are left. The next we took was to create dummy values of all those variables which contained strings as values. Checking out the target variable ”Simple_journal” showed us that the data was highly unbalanced. To handle such a problem of highly unbalanced classification data, it is good to first test the data with re-sampling and then sampling so as to compare the result. Before resampling was

Figure 2: ROC analysis of balancing methods



applied, we normalized the 'amount' variable using the StandardScaler library from sklearn.

We after that made use of traditional under-sampling and created a half-half train and test set for this by randomly selecting thesame number of samples in the minority class records from the majority class . When the undersampled data was ready, we then split the dataset and undersampled data set into test and training sets.

Due to the fact that dataset is highly unbalanced, we looked at recall more than accuracy and precision. Using the kfold score of 10, we found our best model with the best recall score for a number of iterations. This is then used to predict the undersampled data.

We also used a confusion matrix to have an overview of the number of true positive counts.

In this phase, the logistic regression model is used to predict the undersampled data and the entire data, with the Kfold being used to find the best model to be applied.

After the recall values were obtained from the logistic regression model, we make use of the AUC and ROC curves of true positive to false positive rate. This gives us a curve with an area under the curve. This value shows if the model is doing a good classification job or not.

Black-Box Algorithm

For the black box algorithm, we implemented a neural network.

In this stage, the preprocessing steps were somewhat different from those of the white box algorithm. The preprocessing steps in the white box algorithm were applied here except that no dummy values were made. The data was just prepared to be fed into the model without creating any dummy values. We did convert the data in the csv into a matrix and then split our data into training and test set. We then checked the size of the training, test and dev sets. Instead of just normalizing data, we flattened it out and then normalized the features and created the final training set.

We used neural networks to classify transactions as benign or fraudulent. Function is implemented to initialize weights for forward propagation, which is tested and works fine. The next step was to create and test the sigmoid, relu functions and implement forward propagation step. All the necessary steps such as backward propagation, linear backward were implemented and tested.

To test the accuracy of the model, we analyzed a curve of the cost on the y-axis and iterations per tens on the x axis for a learning rate of 0.0065.

Although this was all done and tested, the model exhibited strange behavior due to an unfound bug. Due to this, no comparison could be made between its performance and that of the logistic regression model.

Bonus assignment

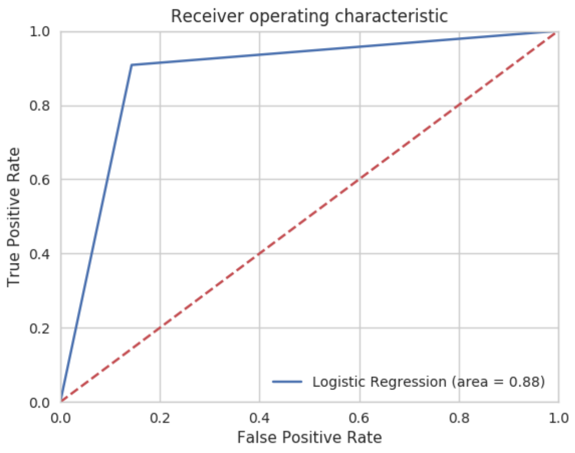
Another way to look at this data is not on a transaction basis, but rather on a credit card or ip address. We created two separate logit classifiers relying on such information. The classifiers rely on the transaction amount, the average amount until current transaction, the total amount in last 24 hours, the time since last transaction and the fraud/transaction ratio until current transaction

These values are calculated for every transaction of a user, where a user is defined by the card_id or ip_id columns. The data is viewed as coming in real-time, so it is first sorted by creationdate and converted to epoch timestamps for easy calculation. Next, we traverse over all transactions and calculate the desired values. Finally, we combine the outputs of the two classifiers with the classifier we created earlier by using the voting rule

$$Avg(confidence(logit_1), confidence(logit_2), confidence(logit_3)) > 0.5$$

and obtain the following results.

Figure 3: ROC of voting classifier


Receiver operating characteristic
True Positive Rate
False Positive Rate
Logistic Regression (area = 0.88)