Delft University of Technology

Cyber Data Analytics

CS4035

Assignment 1: Credit Card Fraud Detection

Authors:	*Student Numbers:*
Alina Bianca Iancu	4930509
Gabriele Mazzola	4815505

May 10, 2019

Contents

1 Task 1: Visualization

For the visualization task we decided to initially perform a few basic checks. We created two barplots (as observed in Figure 1) for checking the distribution of the fraudulent and non-fraudulent transactions over the converted to euros amounts. We observed that fraudulent transactions are distributed over lower amounts than non-fraudulent ones (for the fraudulent ones the maximum is 71984.263 euros, whereas for non-fraudulent ones the maximum amount is 348326.104 euros). Additionally, we generated a barplot for counting the number of fraudulent cases per card type (due to space constraints we included this plot in the Appendix, Figure 6). We observed that the card with the highest number of recorded fraudulent transactions is ’mccredit’, while five types of cards have no such cases observed in the data (namely ’cirrus’, ’mc’, ’visacorporate’, ’electron’, ’vpay’).

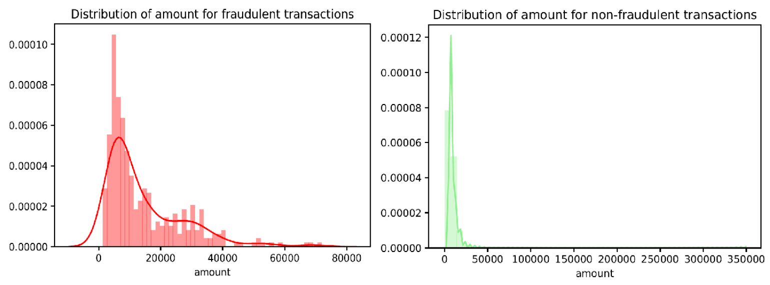


Figure 1: Stacked bar showing the amount of transaction type per different card

Subsequent to these findings, we plotted a heatmap, shown in Figure 2, to detect where we can identify the highest amount of fraudulent transactions, based on amount and card type. What we can conclude is that the fraudulent transactions are more frequent in the case of smaller amounts (below 20000 euros) and for the ’mccredit’ and ’visaclassic’ types of cards.

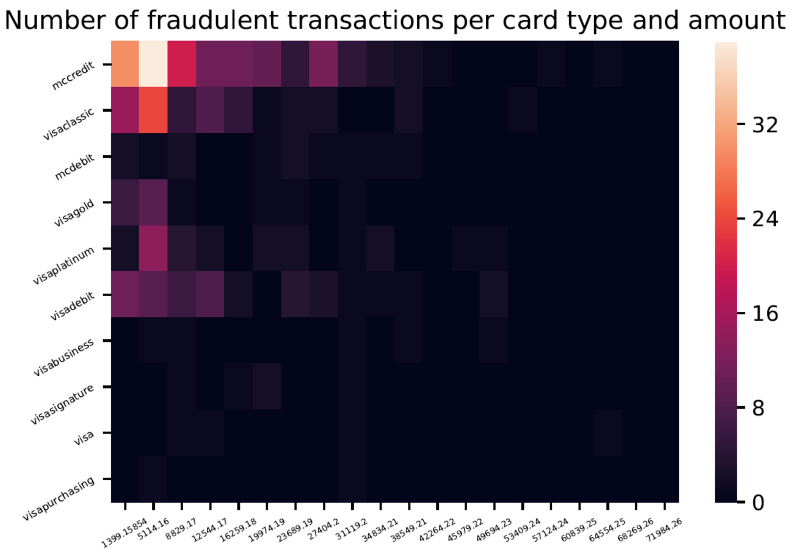


Figure 2: Heatmap showing the amount of frauds per card type and amount bin

2 Task 2: Imbalanced Data

2.1 Data processing

We performed the following preprocessing steps to the data (all the details can be consulted in the Jupyter Notebook):

- Remove transactions with *simple_journal* = 'Refused'. They do not provide meaningful information.
- From *creationdate* we extracted the hour of the day, the weekday, and the month number. We then discarded the *creationdate*.
- We converted *amount* based on *currencycode* in order to have transactions in Euros.
- We encoded categorical variables into discrete numerical values.
- We removed *currencycode* because of high correlation (0.97) with *accountcode*.
- We normalized the *amount* between -1 and 1.
- We removed features we did not consider meaningful (the full final list of features can be seen in the notebook).

2.2 ROC curves

The classifiers that we have experimented with in order to analyze the effects of SMOTE are AdaBoost, Random Forest, Logistic Regression, kNN, NaiveBayes. Due to space limitations, we only included and discussed three of the tested classifiers. The corresponding ROC curves are visualized in Figure 3. For the additional two ROC curves (corresponding to Logistic Regression and NaiveBayes) please refer to the accompanying Jupyter Notebook.

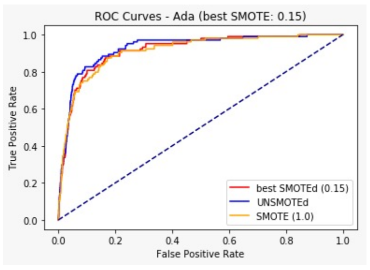
We tested each of these classifiers with different SMOTE ratios[1] and we adopted F0.5 to select the best one (for each classifier). The setting was as follows: 30% of the original dataset was used as a *test* set to only plot the ROC curves, no decision was taken based on the results on the *test* set. The remaining 70% was further split in a 80%-20% fashion, namely *train* and *validation*. By means of *train* and *validation* sets we selected the best SMOTE ratio for each classifier, and then plotted the ROC curves by retraining on the full *train+validation* set and testing on the initial *test* set.

By analyzing the generated ROC curves shown in Figure 3 we concluded that generally using SMOTE for dealing with imbalanced data results in a better performance. Keep in mind that when visualizing the performance using ROC curves, a good classifier aims for the upper left corner of the chart, thus increasing the area under the ROC curve. The performance improvement is clearly visible for the Random Forest (Figure 3b) and kNN with n = 5 neighbors (Figure 3c) classifiers, in which cases the area under the ROC curve increases when using SMOTE (both the best SMOTE parameter, as well as the perfectly balanced SMOTE case) compared to when the data is UNSMOTEd. In the case of the Random Forest we can see that the best SMOTE configuration slightly outperforms the balanced SMOTE (with coefficient 1.0), whereas for kNN it is the opposite case. What we infer from this is that overall, SMOTEing the data at a certain level, helps the classifier to better distinguish between the classes.

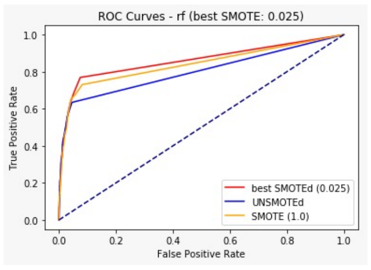
In the case of the AdaBoost classifier (Figure 3a), the performance is very similar in the SMOTEd and UNSMOTEd cases. This can be explained through the fact that generally ROC curves are not a very

[1] https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html

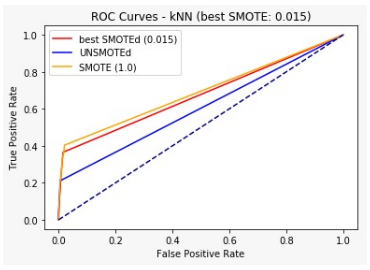
appropriate visualization when dealing with imbalanced data This is the case because the ROC curve plots the True Positive Rate against the False Positive Rate. In the case of highly imbalanced data, as our current case, when the total number of negative cases (so non-fraudulent cases) is very high, the False Positive Rate, which is computed as $FalsePositives/TotalNegatives$, does not drop drastically.



(a) ROC Curve for the AdaBoost classifier



(b) ROC Curve for the Random Forest classifier



(c) ROC Curve for the kNN (with n = 5) classifier

Figure 3: ROC Curves for the three different classifiers

3 Task 3: Classification

As in Task 2 we concluded that using SMOTE for dealing with the imbalanced data situation generally helps the classifiers to better distinguish between the classes and achieve a better performance, for this task we will use SMOTE in order to experiment with white-box and black-box classifiers.

3.1 Black-Box classifier

For the data processing part, please refer to the dedicated section in **Task 2**.

First of all, we decided which classifiers to use as baselines, before starting investigating possible ensembles. We employed the classifiers for which we performed ROC analysis in the Task 2, namely AdaBoost, Random Forest, Logistic Regression, kNN, NaiveBayes. We also associated a SMOTE ratio parameter to each of these classifier, taking into account the one suggested by the analysis performed in Task 2. These parameters have been chosen to try and meet the requirements given in the assignment paper. Also, running again the 'imbalance task' procedure would yield slightly different parameters each time, which means that they are not too significant nor strict. We performed a 10-fold stratified cross-validation, summing up the values of the confusion matrices at each iteration[2].

Classifier	**Smote ratio**	**Precision**	**Recall**	**F0.5**
AdaBoost	0.025	0.0185	0.0145	0.0175
R. Forest	0.015	0.2473	0.0667	0.1604
Log. Reg.	0.5	0.0120	0.8348	0.0150
5-NN	0.015	0.0639	0.2029	0.0740
N.Bayes	0.015	0.0119	0.6609	0.0148

Table 1: Performances single classifier baselines

Figure 1 shows the results of the baselines in terms of Precision, Recall, and F0.5. Again, we chose F0.5 because it's geared more towards Precision, which is necessary if we want to reach the requirements. Although Random Forest is the one with the highest F0.5 score, 5-NN is the one that gets closest to the asked requirements: the recall of the Random Forest classifier is way too low (all the confusion matrices can be consulted on the provided Jupyter Notebook).

Given these results, we then tried to create an ensemble model by ensembling the above-mentioned models. We did it both in a *hard* and a *soft* fashion, which means *majority voting* and *model averaging*, respectively [3]. We employed the same testing procedure used for testing the single classifiers (10-fold stratified cross validation).

Figure 4 shows the confusion matrices for the two ensemble models. All the details can be checked in the Jupyter Notebook.

The requirement on the TP(almost 100) is just met by the SOFT ensemble with a F0.5 score of 0.0586, while the HARD one meets the requirement on FP (maximum 1000), with a F0.5 score of 0.0725. A future experiment would be trying to combine these classifiers.

[2]https://stats.stackexchange.com/questions/147175/how-is-the-confusion-matrix-reported-from-k-fold-cross-validation
[3]https://scikit-learn.org/stable/modules/ensemble.html

		Predicted	
		0	1
True	0	222544	953
	1	281	64

(a) Confusion matrix - Hard ensemble

		Predicted	
		0	1
True	0	221550	1947
	1	245	100

(b) Confusion matrix - Soft ensemble

Figure 4: Ensemble models comparison

3.2 White-Box classifier

For the data processing part, please refer to the dedicated section in **Task 2**.

For the white-box classifier case we decided to experiment with Decision Tree and Logistic Regression. In both cases we performed a Grid Search among the hyperparameter space, both for SMOTE as well as for the actual classifier, in order to determine the best performing configuration. For the hyperparameter values that we have experimented with, please refer to the accompanying Jupyter notebook.

The performance of the classifiers for each possible configuration was assessed by computing the average F0.5 score over the iterations of Stratified 10-Fold cross validation. We employed Stratified 10-Fold cross validation as we are dealing with imbalanced data and we wanted to keep the class distribution in the folds similar to the one in the original data. The best configurations achieved for the two classifiers and their corresponding performances are visualized in Table 2.

Classifier	**Smote ratio**	**Precision**	**Recall**	**F0.5**
Decision Tree (criterion = 'gini', max_depth = 5 , min_samples_split = 0.1)	0.05	0.0514	0.1797	0.0557
Logistic Regression (penalty = 'l2', C = 31.62)	0.05	0.0361	0.1014	0.0429

Table 2: Performance classifiers white-box approach

As we can observe in the results by comparing the F0.5 score, the Decision Tree classifier has a better performance than the Logistic Regression. That is why, for further discussion, we will consider the Decision Tree classifier as our white-box algorithm choice.

Since for the white-box algorithm we want to focus on having an interpretable solution that makes it easy to explain why a certain transaction is being fraudulent, we decided to plot the resulting decision tree, in which we can clearly follow the path from the root of the tree, when the classification starts, to one of the leaves containing the final decision for the classification. The tree was generated by defining it using the best obtained configuration from the grid search and fitting it on the data obtained using stratified train-test split. The plot is included in the Appendix in Figure 8.

When comparing the two different approaches, black-box and white-box, we can observe that in the case of the black-box approach we can reach a significantly better performance, very close to satisfying the performance requirements for this assignment. More specifically, the best F0.5 score we obtain in the case of the white-box method is 0.0557 for the Decision Tree classifier, whereas in the case of the black-box method we obtain 0.0740 for the 5-NN. It might be the case, that considering the complexity of the data, the white-box classifiers are too simplistic. However, in the case of the white-box method we have the advantage of a clearer interpretation, as we can follow the classification path in the decision tree from the root to the decision leaves. On the other hand, in the other case, since we are using an ensembling of black-box models, we cannot trace back the decisions that led to the classification label.

4 BONUS point (dedicated Jupyter Notebook)

In this part of the assignment we decided to start again from the RAW dataset that was provided. This is done because, during the preprocessing part made for Task 2 and 3, some of the features that we need for the aggregations were dropped. Most of the preprocessing done for the previous two sections is repeated, but this time we kept and made use of the *bookingdate* and *creationdate* fields.

We hypothesize that if a transaction corresponding to a card/ip/mail has been detected as fraud, then it is likely that a new transaction with the same card/ip/mail might be fraudolent. Grouping by either card, ip, or mail, we investigated those transactions for which there was a previously reported fraud in the dataset. In order to do this, we require the *bookingdate* of the reported fraud to be earlier in time than the *creationdate* of the transaction we are investigating.

RESULT: Unfortunately, as it can be seen in the attached rendered notebook, all the transactions for which we found previously fraudolent data (card, ip, or mail) were not fraudolent, and therefore this approach would not be useful for helping the classifier in achieving its goal, at least with the given data.

Our second approach consisted of aggregating the transactions based on three grouping criteria (**card**, **ip**, and **mail**) and computed the following measures (repeated for each one of the three grouping criteria):

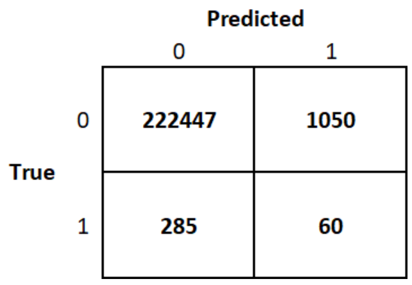
- Number of transactions happened in the day up to the transaction in analysis
- Amount obtained by summing the transaction of the previous point
- Average amount obtained by averaging the total amount obtained in the previous point

This led to the generation of 3 new features per criterion, for a final total of 9 new features.

4.1 Testing

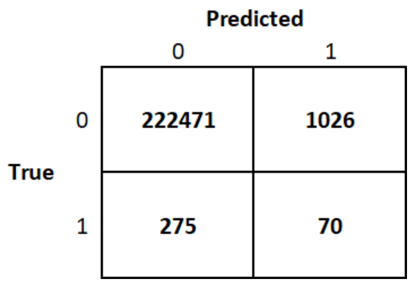
We decided to test the performance of 5-NN (the single classifier with the highest F0.5 score found in the previous tasks) with this new improved set of features. We employed the same exact testing procedure employed for the previous classifiers (10-fold stratified cross validation). Results are shown in Figure 5a. The achieved F0.5 score is 0.0627.

When we ran the very same classifier on the data without the aggregated features, we got the results shown in 5b, therefore it does not seem the generated features can help achieving the goal of fraud detection. Instead, they even seem counterproductive.



		Predicted	
		0	1
True	0	222447	1050
	1	285	60

(a) Confusion matrix - 5NN with aggregated features



		Predicted	
		0	1
True	0	222471	1026
	1	275	70

(b) Confusion matrix - 5NN with standard features

Figure 5: 5-NN comparison with and without aggregated features

Appendix

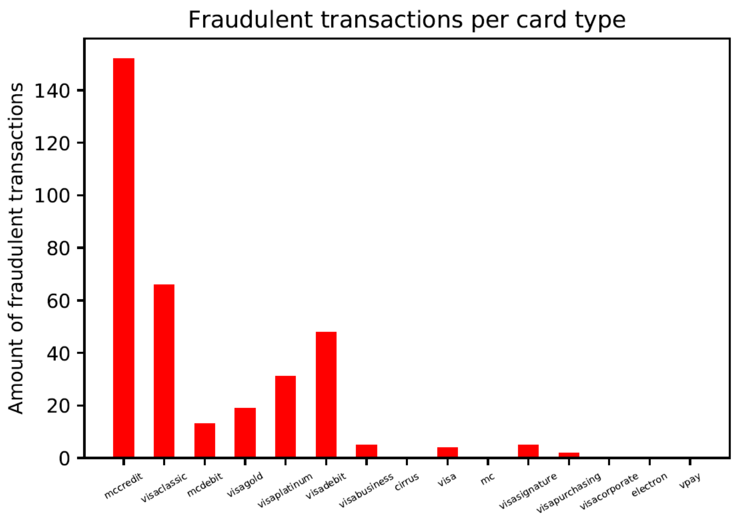
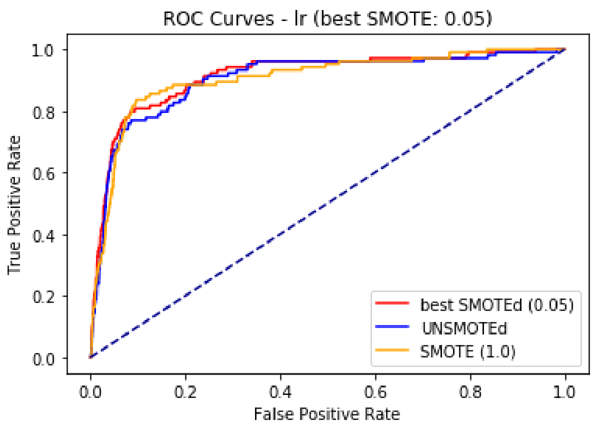
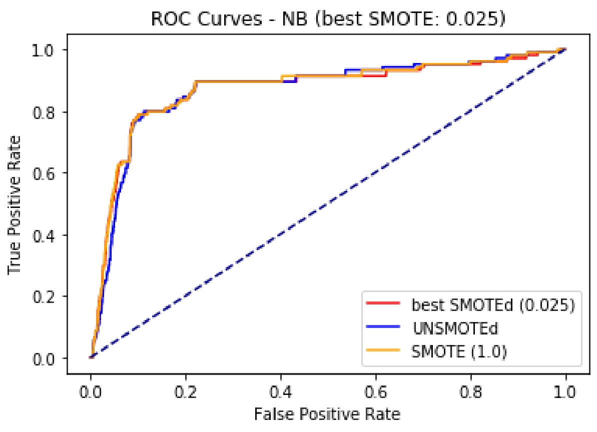


Figure 6: Stacked bar showing the amount of transaction type per different card



(a) ROC Curve for the Logistic Regression classifier



(b) ROC Curve for the Naive Bayes classifier

Figure 7: ROC Curves for the additional two classifiers

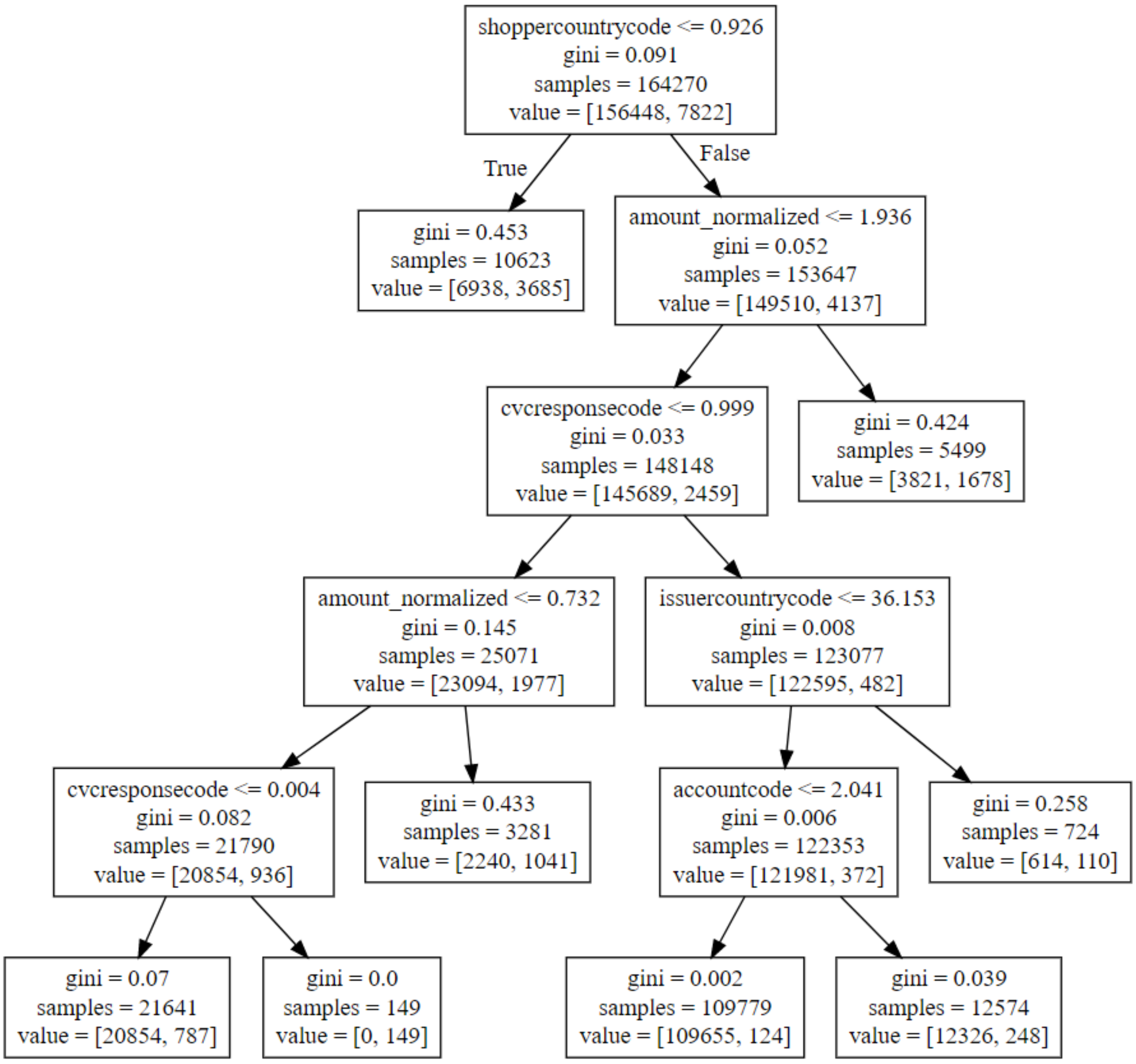


Figure 8: Visualization for the Decision Tree classifier used for the white-box algorithm