Cyber Data Analytics - Lab 1 Report

Enreina Annisa Rizkiasri (4701224) and Nivedita Prasad (4712099)

Source Code: *https://github.com/enreina/cs4035-lab/tree/master/lab1*[1]

Visualization

In order to visualize and find trends of the fraud transaction dataset, we first applied some basic preprocessing to the data. We ignored transactions marked as *refused, as* its ambiguous fraudulency, so we have **236,691** legitimate (*Settled*) transactions and **345** fraudulent (*Chargeback*) transactions. Furthermore, to normalize the dataset, we also converted the `amount` to EURO based on the corresponding `currencycode` of each transaction.

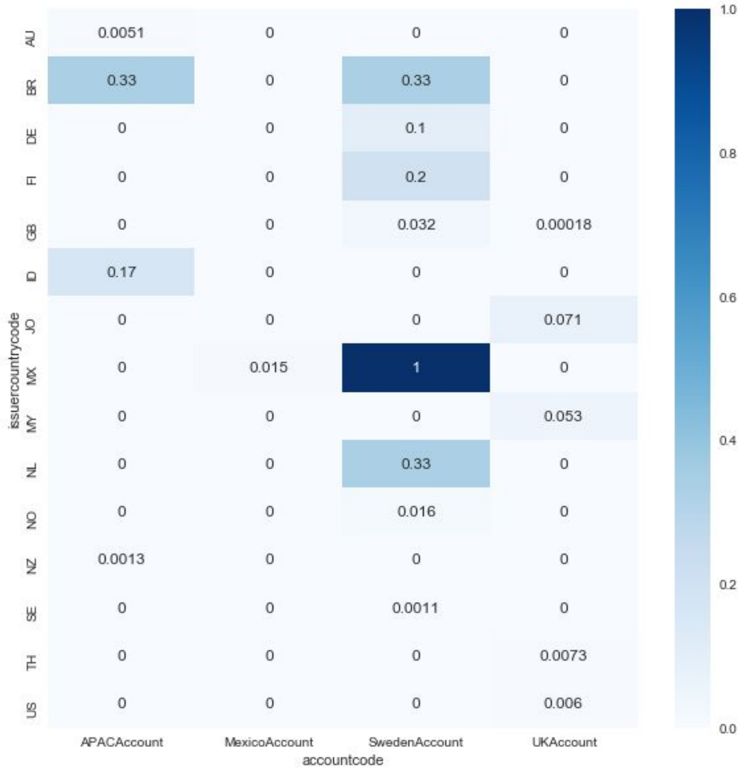


Figure 1: accountcode vs issuercountrycode

We explored data visualization in several ways: (1) using heat maps to explore categorical data and (2) using line chart to explore temporal trends and (3) distribution histogram.

The heat maps were generated on each pairs of categorical attributes, and each colored cells in the heatmap represent the fraction of fraudulent transactions in the dataset over all transactions which satisfies the corresponding pair of attribute values. One interesting heat map is shown in Figure 1 between the `accountcode` and `issuercountrycode`. We can see that there seems to be a high fraudulent possibility when the transaction has an `accountcode` of SwedenAccount and `issuercountrycode` of MX.

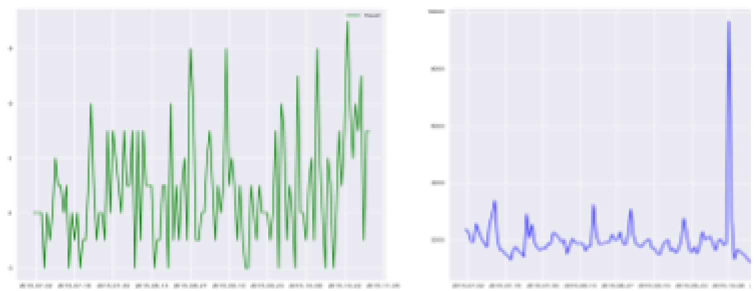


Figure 2: date vs number of transaction (left: fraud, right: legitimate)

Figure 2, on the other hand illustrates the daily trend of number of transaction between fraud and legitimate transaction. Note that the daily number of fraud transaction is relatively small and varies every day, while legitimate transaction has a more stable daily rate which the exception of a spike on the *black friday*.

Additionally, Figure 3 shows the comparison of the amount distribution between fraud and legitimate transaction. Fraud transaction tends to have a bit right tail, which indicates that fraud transaction tends to involve larger amount of money.

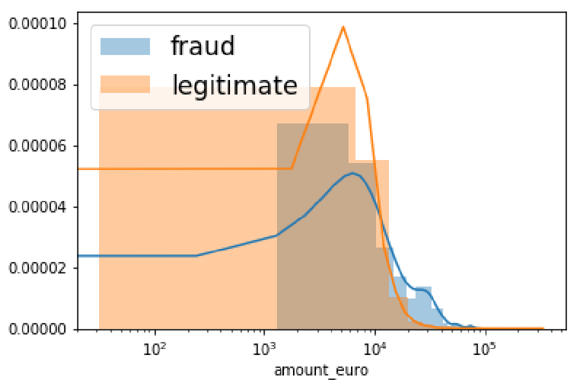


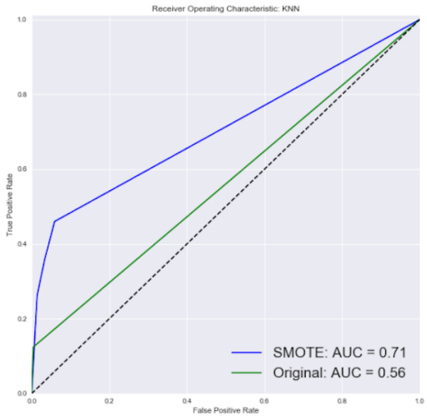
Figure 3: Distribution

---

[1] source code is uploaded to Brightspace, if you want to see the github repo, send your github username to e.a.rizkiasri@student.tudelft.nl

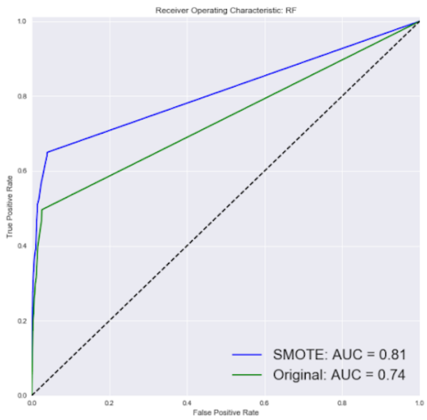
Handling Imbalanced Dataset

As stated before, we have an imbalanced dataset where the number of fraudulent transaction is way less than the number of legitimate transaction. The ratio is as low as 0,14%. To overcome this, we try to apply one of the most popular oversampling technique for imbalance data: SMOTE. We use the scikit learn implementation of SMOTE and compare the performance of three classifiers (K-Nearest Neighbor, Logistic Regression, and Random Forest) trained on the training set without and with SMOTE.



(a) KNN

For evaluation purpose, we split the original dataset into 60:40 split each as training set and test set respectively. Then, we plot the Receiver Operating Characteristic (ROC) curve of each classifier predicting the transactions in the test set. As we want our classifier to detect as much fraudulent activity as possible (i.e. true positive rate / TPR), while also minimizing the false positive rate (FPR), we calculate the Area Under the Curve (AUC) of the ROC (an ideal AUC should be 1.0 with TPR equal to 1 and FPR equal to 0).

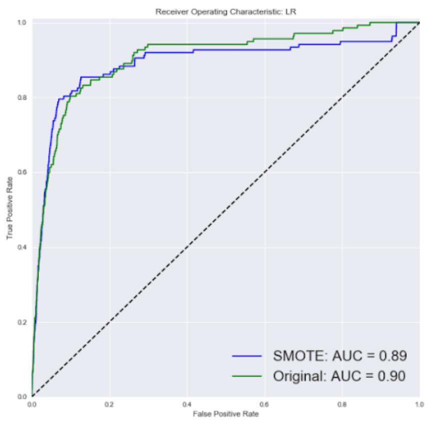


(b) Random Forest

Figure 4 (a), (b), (c) illustrates the ROC curve for K-Nearest Neighbor (KNN), Random Forest (RF), and Logistic Regression (LR) respectively. The AUC score is more clearly shown in the Table 1.

	Without Smote			With Smote		
	AUC	**Precision**	**Recall**	**AUC**	**Precision**	**Recall**
KNN	0.56	0.18	0.021	0.71	0.01	0.27
RF	0.74	0.08	0.014	0.81	0.08	0.043
LR	0.90	0.28	0.014	0.89	0.01	0.85

Table 1: AUC, Precision, Recall



(c) Logistic Regression

Figure 4: ROC Curve

As we can see, in the case of KNN, the AUC improves a lot with SMOTE being applied. There is also an improvement in the case of RF, but in the case of LR, the AUC tends to be not that different between using and not using SMOTE.

In conclusion, SMOTE seems to work well with distance-based classifier such as KNN. In the case of LR which already considers class probability, SMOTE does not affects the AUC that much. Additionally, we have to notice that both the precision and recall is still very low despite high AUC. This reflects that SMOTE which tries to balance the training set by oversampling the fraudulent transaction, will affect prediction in the real world where the dataset is highly skewed between fraudulent and legitimate transaction. A more appropriate metric for this kind of case would be the average precision or Area Under the Precision-Recall Curve.

Classification

Preprocessing

In addition to converting amount into euros, we first convert the `creationdate` attribute into three derived attributes creation month, creation day of the week, and creation day of the month. We don't include the original `creationdate` attribute for training the classifier. Then, we map the `cvcresponsecode` of value 3, 4, 5, and 6 to the value of 3 (because the information is the same). For categorical data such as `issuercountrycode`, `txvariantcode`, and `shopperinteraction`, we make sure that they are converted to discrete (categorical) data type rather than numeric data type. We also decide to not consider some attributes that seems to be an identifier such as `txid`, `mail_id`, `ip_id`, and `card_id` (see the Bonus Task section where we utilize some of these attributes for aggregation instead).

Black Box Classifier: Random Forest

We have chosen Random Forests as black box for our credit-card fraud dataset, as the classifier is not sensitive to the dataset during training phase. Random Forest [4] is an ensemble method that uses many bootstrapped 'm' samples of the dataset to construct group of independent decision trees. In the end, averaging is done on the trees to improve the accuracy and to keep an eye on the problem of overfitting. The n_estimators parameter is kept at 10 trees as default at first and the split criteria was chosen to be Gini impurity criterion instead of entropy. The Gini index decreases with respect to the corresponding parent node in the decision tree during the split. The leaves of the trees are the classes - Fraud or Legitimate; and the best predicted class is majority voted.

To base our intuition of using random forests on the credit card fraud dataset on more concrete results, we researched a bit. According to the authors of [1], suggest Random forests over SVM and Logistic regression and additionally, the paper by [2] show good performance in credit card fraud detection using Random forests as their classifier.

The n_estimator (number of trees), max_depth (maximum depth of the trees), and threshold hyperparameters are very important for the random forests. Hence, we played around with it a bit and settled for n_estimator of 200, max_depth of 10, and threshold of 0.6. These particular values yield the least false positives and high true positives to the best of our knowledge. As a next step, we then decided to get a deeper understanding of how the data was split based on the splitting criterion and explored Decision Trees as the white box classifier.

White Box Classifier: Decision Tree

The main reason we used decision trees is that is easy to understand, visualize and interpret even though they don't generalise really well. It uses a white box model in contrast to neural network or random forests where they are black box model. The best part is that we don't have to normalize the data. [5]

Decision trees - the name of this classifier clearly states that an if-then condition is applied on features/column value and as an end result that particular sample is decided whether it will belong to a Fraudulent or Legitimate class. The length of the decision tree is based on complexity of the decision rules.

The decision tree are greatly influenced if the classes are imbalanced, hence we have applied undersampling to the dataset before training our decision tree with maximum depth of 10 levels.

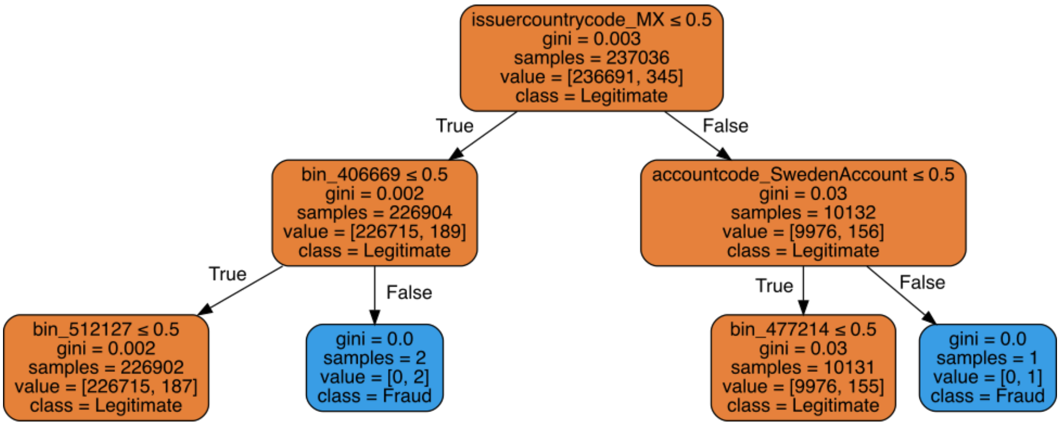


Figure 5: Decision Tree

Figure 5 shows how the classes represented as leaves of the tree (orange represents legitimate transaction and blue represents fraud) are arrived upon with `issuercountrycode`, `accountcode`, and `bin` being the feature criterion splitter and the gini index gradually decreases as compared to the parent node. The complete decision tree can be viewed here[2].

One leaf on the right of the figure indicates that a transaction can be suspected as a fraud (at least based on the training data) if the credit card issuer is from Mexico (`issuercountrycode` = MX) and the merchant's webshop is located in Sweden (`accountcode` = SwedenAccount). The other blue leaf indicates that if the `issuercountrycode` of a transaction is not MX (Mexico) and if its issuer (`bin`) has the id of 406669, the it is most likely to be fraud (again, at least based on the training set).

The number of fraudulent cases (true positives) we have detected on the trained decision tree is 215 and the number of false positives are 45,280. We performed cross validation on the data and the results have shown a fairly good enough recall but with really low precision and very high false positive which we think won't be acceptable in the real world.

Comparison

Both of our models are evaluated using the 10-fold cross-validation technique, and then we calculate the average precision, recall, true positive, false positive, true negative, false negative, and average AUC. The result is shown in the table below. We can see that even though the decision tree has identified more fraudulent cases (true positives), it has way lower average precision and the number of false positive is twice of the one identified by Random Forest.

	Avg. Precision	Avg. Recall	True Positive	False Positive	True Negative	False Negative	Avg. AUC
Random Forest	**60.09%**	48.22%	168	**22,259**	**214,432**	177	**0.916**
Decision Tree	3.28%	**62.36%**	**215**	45,280	192,411	**130**	0.618

[2] https://drive.google.com/open?id=1584ClMHiipWT06jBSRrOJkekyqCbZUSv

Bonus Task

Derived Attributes

For the bonus task, we try to contextualize each transaction linked by their card_id and mail_id. We refer to the work of [3] and we derive the following attributes by grouping the transaction based on same characteristics (e.g: same `card_id` and `mail_id` or same `card_id` and `shoppercountrycode`), and then sort them by creationdate. Then we iterate through the transaction while accumulating the transaction count and the average amount. For each transaction, a new attribute is added by calculating transaction count and average amount based on the accumulated data from the previous transaction. The derived attributes are:

- `prev_amount_mean`: average amount (in currency of `currencycode`) per each valid transactions before this transaction using the same `card_id`
- `prev_amount_euro_mean`: average amount (in euro) per each valid transactions before this transaction using the same card_id
- `prev_transaction_count`: number of transaction with the same card_id before this transaction
- `prev_transaction_count_mail_id`: number of transaction with the same `card_id` and `mail_id` before this transaction
- `prev_transaction_count_shoppercountrycode`: number of transaction with the same `card_id` and `shoppercountrycode` before this transaction
- `prev_transaction_count_currencycode`: number of transaction with the same `card_id` and `currencycode` before this transaction

Then we train a Random Forest (RF) classifier with the same parameters explained in the previous section. We evaluate the performance using 10-fold cross validation, and apply undersampling on each training fold with ratio 3:7 between fraud and non-fraud data. The result is shown in the table below included with the performance of RF on the data without the derived attributes. We can see that precision drops significantly, but both there is improvement on recall and decreased number of false positive. It shows that by deriving attributes based on each transaction context, the fraud detection system can be improved.

	Avg. Precision	Avg. Recall	True Positive	False Positive	True Negative	False Negative	Avg. AUC
Without Derived Attributes	**60.09%**	48.22%	168	22,259	214,432	177	0.916
With Derived Attributes	2.15%	**67.55%**	**233**	**10,630**	**226,061**	**112**	**0.939**

References

1. Weidong Tian. 2016. *Commercial Banking Risk Management: Regulation in the Wake of the Financial Crisis*. Springer.
2. C. Whitrow, D. J. Hand, P. Juszczak, D. Weston, and N. M. Adams. 2008. Transaction aggregation as a strategy for credit card fraud detection. *Data mining and knowledge discovery* 18, 1: 30–55.
3. 2011. Data mining for credit card fraud: A comparative study. *Decision support systems* 50, 3: 602–613.
4. RandomForestClassifier. Retrieved May 7, 2018 from http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
5. 1.10. Decision Trees — scikit-learn 0.19.1 documentation. Retrieved May 7, 2018 from http://scikit-learn.org/stable/modules/tree.html