Credit Card Fraud Detection Using Supervised and Unsupervised Learning

Teoman Akyuz

December 18, 2025

Executive Summary

The Credit Fraud Detection dataset was used to determine whether a given transaction is fraudulent or not. There are no relevant insights from the features since they are encoded for confidentiality issues and the main challenges of this dataset are the class imbalance with only a 0.17% of fraudulent transactions and the large amount of instances. Given the nature of the dataset, the most reasonable metrics to evaluate the model are recall, precision and Area Under the ROC Curve (AUC), which we aim to maximise. We treated the imbalance problem with two different approaches based on weighting instances and the SMOTE technique. Furthermore, for this task anomaly detection plays an important role, hence, both supervised and unsupervised learning models are used. These include Random Forests, Isolation Forests and DBSCAN. The most reasonable and balanced model is the Weighted Random Forests that achieved a 80.73% recall, a precision of 92.63% and an AUC of 90.36%. Two models seem to be over-fitted: SMOTE Random Forests and DBSCAN. Finally, the two Isolation Forests models performed good in terms of recall and AUC but poorly in precision, therefore these were discarded.

Contents

1 Introduction and literature review

The Credit Card Fraud Detection[1] dataset provides transactions made by credit cards in September 2013 in Europe. The whole dataset represents transactions that occurred in two days and it only accounts for numerical variables that are a result of a PCA transformation with exception of the 'Time' and 'Amount' features. Due to confidentiality issues the PCA variables are encoded in the form "V1,..., V28" and no relevant insights into what they may represent can be obtained from them. The response variable takes value 1 if the transaction was deemed fraudulent and 0 otherwise.

The primary goal of this project is to ensure all fraudulent transactions are accurately identified. While the identification of legitimate transactions as fraudulent (false positives) is less of a concern, it is not entirely inconsequential. The focus, therefore, is on minimising the occurrence of false negatives, meaning we strive for a high recall rate. This dataset poses two important challenges to address: large number of instances and the class imbalance. The latter is a very relevant issue and it is therefore of great importance to treat it accordingly. The fraudulent transactions, the class of interest are overwhelmingly sparse, leading to an imbalance of 0.17% with respect to the minority. This translates to the issue that the learning algorithms will be unable to generalise and will not predict the fraudulent transactions well, this may lead to a misleadingly high overall accuracy rate because the model may only be correctly predicting the majority class. Most importantly, it suggests that the model may not be very good at predicting fraudulent transactions, exactly the outcomes we want to be sure to catch.

There are many ways to address this problem at a data level, such as weighting the loss function by giving more importance to the minority class, one can also vary the classification threshold for probabilistic classifiers and another option is to bootstrap the dataset in order to get a balanced class. The last option may incur in a loss of data and accuracy, therefore conclusions must be done carefully. In [7], they study analytically the under-sampling technique and argue that it affects negatively the classification accuracy and introduces bias. They propose a method based on the threshold to decrease bias for probabilistic methods and they test it in the credit card dataset.

Another relevant and effective method to treat the imbalance is the Synthetic Minority Oversampling Technique (SMOTE) [2]. SMOTE works by generating synthetic samples from the minority class instead of creating copies of existing samples. This is achieved by randomly selecting a point from the minority class and computing the k-nearest neighbors for this point; the synthetic samples are then created by interpolating between the chosen point and its neighbors. We take the enhanced diversity samples that SMOTE generates over techniques such as random oversampling, since we aim to produce data able to generalise for our models to be trained on in order to reduce the risk of over-fitting by potentially adding new information ideally without the duplicated details of noise from the data. One concern we had with utilising SMOTE, was whether the assumption that the minority class samples are similar enough that linear combinations of these samples are valid, may prove ineffective with our high dimensional data due to it's use of nearest neighbours.

We attempt to employ a Density based scanning algorithm, DBSCAN, whereby it classifies points into specific types, border points, and noise points, based on two parameters: the minimum number of points to form a dense region epsilon, that defines the maximum distance

[1] https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud?resource=download

between two points for them to be considered as in the same neighborhood. The motivation behind this is because of DBSCAN's ability to detect and isolate outliers or noise points, the algorithm also does not require the user to specify the number of clusters beforehand which may turn out to be useful with respect to the sparsity of fraudulent clusters. However, once again, one obstacle in the performance of this will link to the high-dimensional data since it can dilute the concept of proximity (traditional distance measures like Euclidean distance that become less effective) or neighborhood [10].

Alternatively, there are ways to address the class imbalance at an algorithm level. For instance, the Isolation Forest algorithm [4] [5] is a very efficient approach to detect outliers in a dataset. It works by isolating anomalies and it takes into account some of the properties of anomalies: they are the minority and their attribute values are very different from normal values. Some advantages are that it has only two parameters, the number of trees and the sub-sampling size, for which the authors in [4] argue that both can be small enough to converge in a small time. Also, this approach does not utilises distances and therefore makes it a very efficient algorithm with a linear complexity and few memory requirements. It has some variants and it has proven good and efficient results for anomaly detection. Another interesting algorithm is the Support Vector Machines (SVM) for one class. This is also a classic approach for detecting outliers, but in [3] they discuss some relevant drawbacks that make the algorithm not very useful for this particular dataset. Firstly, this method requires a large amount of positive data to create an accurate boundary. Secondly, it under-fits and over-fits easily and thirdly, the optimisation model to solve the problem is very inefficient for a large dataset, i.e., it is very computational expensive. This algorithm accounts for two parameters, namely, ν that controls the noise in the training data and λ that controls the smoothness of the boundary. In [1] they propose a methodology to balance these parameters and prevent under/over-fitting, but it is very inefficient, again. One important thing to highlight, is that these algorithms belong to unsupervised learning and thus it is not necessary to train them with the target. Nevertheless, we can evaluate how good they perform with the test target and predicted outcomes. Finally, Auto-encoders that use Neural Networks (NNs) are another relevant approach for this type of problems but we did not implement this.

2 Data modelling and insights

2.1 Insights and EDA

The Credit Card Fraud Detection is a very interesting one, because since it does not account for some data issues (such as missing values or huge amount of features), it does have many instances and a significant imbalance in the target class. There are 30 features plus the target and 28 of them have been already transformed using PCA. The other two are numerical and display the time of the transaction and the amount of it, so it is relevant to find insights regarding both the time and amount of the transactions. This dataset has 284807 instances for which only 0.17% are fraudulent, which certainly is a problem that must be treated and conclusions must be done accordingly.

As mentioned above, one advantage of this dataset is that there are no missing values. It is possible that the authors pre-processed this in order to delete all of these in a clever way. Also, one issue that may occur when testing these algorithms with new real data (without the PCA transformation) is that there may be a data leakage, since we do not have any information about it we cannot determine whether it is possible to account for all the data in a real scenario.

As mentioned above, the most relevant variables have been transformed using PCA. Figure 1 depicts the explained variance per component, which is indeed very low for the first two components as it is not supposed to be. It would be ideal to have at least the 70% of the variance explained in the first two to reduce the number of features in the models. However, this is not the case and it makes sense to use most, if not all, of the variables in the models. The variance was not provided in the dataset, but instead it was calculated. Let Z_i be the i-th component, then the variance of a single component is

$$\text{Var}(Z_i) = \frac{\sum_{j=1}^{N} (z_{ij} - \bar{z}_i)}{N - 1}$$

where N is the length of the dataset and $\bar{z}_i$ is the mean of the i-th component. Finally, the proportion of variance for each component is given by

$$\text{prop}_i = \frac{\text{Var}(Z_i)}{\sum_{i=1}^{p} \text{Var}(Z_i)}$$

where p is the number of PCA components [8]. The variance captured by the first component is around 12.48% and for the second component is 8.87%, together they explain a bit more than the 20% which implies that most unlabeled features are still relevant, making our task all the more difficult.

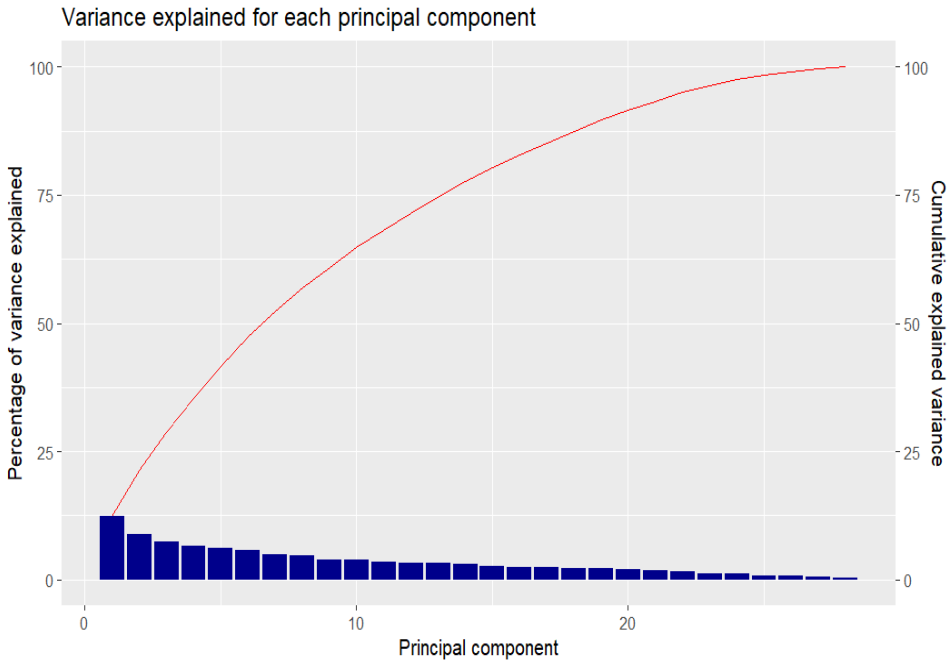

Figure 1: Proportion of variance explained for each principal component.

2.2 Imbalanced target

As mentioned above, there are different ways to address the imbalance problem. In this project we consider two data level methods: the inverse class frequency method and synthetic minority

oversampling technique. The first one weights the instances in the train set and penalises more misclassified predictions in the minority class. The second one uses an over-sampling technique.

2.2.1 The Inverse Class Frequency Method

Now consider the Inverse Class Frequency Method. From [9], let the usual 0-1 loss function be the instances where the classifications do not match:

$$l(y_i, \hat{y}_i) = \mathbf{1}(y_i \neq \hat{y}_i),$$

for every correct classification it takes 0 and value 1 otherwise. Tthe error rate for the test set

$$\text{error}_{\text{Te}} = \frac{1}{|\text{Te}|} \sum_{(x,y)\in \text{Te}} \mathbf{1}(y \neq \hat{f}(x))$$

If we weight the instances to give more importance to the unbalanced class, the error expression is modified as follows.

$$\text{error}_{\text{Te}} = \frac{1}{|\text{Te}|} \sum_{(x,y)\in \text{Te}} \mathbf{1}(y \neq \hat{f}(x)) w_x$$

Where w_x is the weight for instance x. There are only two weights: for the positive class and for the negative class. Each of the weights were calculated as follows:

$$w_0 = \frac{|\text{Tr}|}{2|\{(x,y) \in \text{Tr} : y = 0\}|}$$

$$w_1 = \frac{|\text{Tr}|}{2|\{(x,y) \in \text{Tr} : y = 1\}|}$$

Note that the weighting of instances was only performed to train the model and penalise in a greater manner the fraudulent transactions. Once the model has learned to pay more attention to fraudulent transactions, there is no need to weight the test set.

2.2.2 Synthetic Minority Oversampling Technique (SMOTE)

When addressing class imbalance in datasets, particularly when the minority class is the primary focus, conventional classifiers can exhibit a bias towards the majority class. The Synthetic Minority Over-sampling Technique (SMOTE) offers an innovative solution to this issue by generating synthetic instances for the minority class. It draws from the nearest neighbors of the sparsest regions; the synthetic instance x_{new} is generated through linear interpolation using the following formula:

$$x_{new} = x_i + \lambda(x_{zi} - x_i) \tag{1}$$

- x_i is the vector of the original minority class instance.
- x_{zi} is the vector of the randomly selected nearest neighbor.
- λ is a random number drawn from a uniform distribution between 0 and 1.

The term $(x_{zi} - x_i)$ defines the vector between the original point and its neighbor. Multiplying this vector by the scalar λ determines a random position along the line segment. Adding this value to x_i places the synthetic instance x_{new} within the feature space between the two existing observations.
There is a trade-off between keeping the dataset small (or at least a similar size of the original) and having a better balanced target. This clearly inflates the false positive rate due to the increased presence of synthetics frauds holding up the structure of the data. The mention of structure here is important, as we do not know anything about the data and have to treat the metrics through the lens of an unsupervised learning project.

2.3 Anomalies visualisation

In out attempt to build a friendly visualisation of the clusters, we obtained a plot that in certain way exhibits the differentiation between fraudulent and non-fraudulent transactions. Nevertheless, since these two first components only explain around the 20% of the variance, it does not show clearly enough the clusters. From figure 2 it can be noticed, however, that the fraudulent transactions account for a higher value in the second component and a middle value in the first component as highlighted. In short, the relevant insight from figure 2 is that the fraudulent transactions (blue points) usually take higher values for the second component.

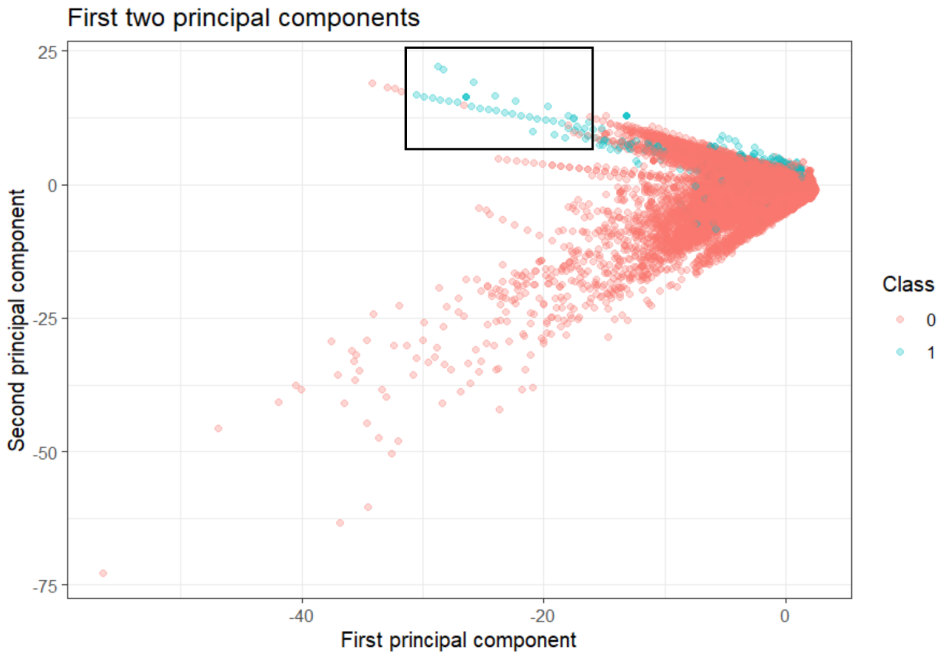

Figure 2: Scatter plot of the two first components V1 and V2.

3 Results

3.1 Learning algorithms and hyper-parameters tuning

In this part all the learning methods used and its results are discussed. It is important to emphasize in the following: to test the hyper-parameters, 5-fold cross validation using a train set, test set and a validation set is utilised. The latter is a subset of the training set that changes in every iteration of the cross validation and the test set is left untouched until the end. The training of models in this procedure is done with the complement of the validation set within the training set.
Our main metric is the recall, but we also consider the precision and the Area Under the ROC Curve (AUC). The train set corresponds to the 80% of the dataset and the test set is just the rest.

3.1.1 Random Forests

This model is not uniquely for anomaly detection but it is usually very powerful for any type of classification problems (and regression as well). Its computation training time depends on the number of trees and size of the data. Note that due to the high number of instances, the number of trees should be kept small in order to avoid an extensive training time. One of its advantages is that it is not sensible to not scaled data.

All the metrics were averaged and these were compared to determine the best hyper-parameter for the models and due to the high class imbalance, this model is tested using two approaches to address the problem: SMOTE sampling and weighting instances.

SMOTE approach Table 1 depicts the results of the SMOTE Random Forests. This model was evaluated using the 5-fold cross validation procedure for trees of size 1, 5, 10, 15 and 20 due to high computational time.

-

Table 1: 5-fold cross validation results for SMOTE Random Forests

Number of trees	Mean Recall	Mean Precision	Mean AUC
1	0.9393	0.9331	0.9684
5	0.9483	0.9905	0.9740
10	**0.9577**	**0.9907**	**0.9787**
15	0.9528	0.9932	0.9762
20	0.9573	0.9929	0.9785

It is of importance to highlight the fact that the increase of size due to SMOTE influenced negatively the computation time. Also, the last model chosen with 10 trees was evaluated using the test set, we want to evaluate this with the data as real as possible and avoided leakages. The model achieved a 96.15% recall, a precision of 99.33% and an AUC of 98.06%. Figure 3 depicts the ROC curve for this model with the best hyper-parameters. We suspect that this model may be over-fitted due to the almost perfect ROC curve.

Weighted Random Forests approach The results for the weighted random forest approach are shown in table 2. This model was evaluated using the 5-fold cross validation procedure for

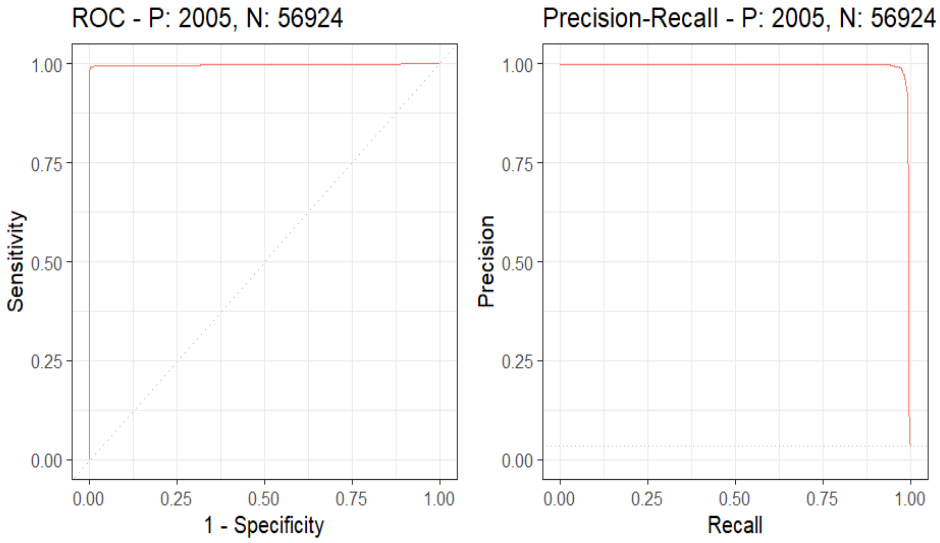

Figure 3: ROC curve for the SMOTE Random Forests with 10 trees.

trees of size 1, 5, 10, 20 and 30.

Table 2: 5-fold cross validation results for Weighted Random Forests

Number of trees	Mean Recall	Mean Precision	Mean AUC
1	0.7223	0.5770	0.8607
5	**0.8040**	**0.8828**	**0.9019**
10	0.7913	0.9036	0.8955
20	0.7805	0.9115	0.8902
30	0.7955	0.9200	0.8977

From table 2 one can notice that the number of trees with the highest recall is 5. After training the whole dataset using 5 trees, a recall of 80.73%, a precision of 92.63% and an AUC of 90.36% were obtained on the test set. Furthermore, the ROC curve for this choice of hyper-parameter and model is shown in figure 4.

3.1.2 Isolation Forests

This model is an special case of the Random Forests algorithm that isolates outliers, as mentioned in the introduction. The output of this model is an outlierness score from 0 to 1. If this value is close to 0, it means that the data-point is average and is not an outlier at all. Whereas a score closer to 1 indicates outlierness. From a score of 0.5 a data point can be considered as an average outlier and therefore this score can be seen as a probability of how outlier a point is. Note that the threshold to classify outliers can be varied, but we used a threshold of 0.5 to determine if a point is an outlier or not.

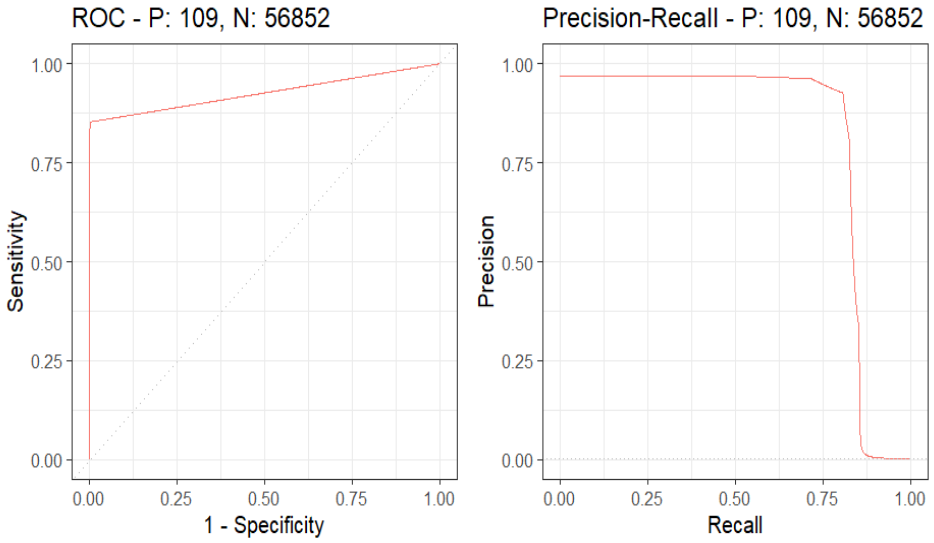

Figure 4: ROC curve for the Weighted Random Forests with 5 trees.

Variant 1: Isolation Forests The first variant tried for the Isolation Forests, consists of only modifying the sample size and number of trees. The results for the first variant are shown in table 3.

Table 3: 5-fold cross validation results for variant 1 of Isolation Forests

Number of trees	Mean Recall	Mean Precision	Mean AUC
1	0.8209	0.0075	0.8625
10	0.7588	0.0219	0.9253
20	0.8652	0.0198	0.9436
50	0.8710	0.0189	0.9455
100	**0.8722**	**0.0229**	**0.9481**

There are two hyper-parameters for the first variant of Isolation Forests: the sample size and the number of trees. The first one was set to be 100 after noticing over-fitting with a 1000 sample size. The number of trees were evaluated using 5-fold cross validation for 1, 10, 20, 50 and 100 trees. From table 3 one can notice that the number of trees with the highest recall is 100 and it is also the hyper-parameter with the highest AUC. Regarding the precision, there is a significant drop comparing with the results of the Random Forests approach. It is of special importance to define how costly it is to have a low precision. However, the best model is that with 100 trees. After training the whole dataset using 100 trees, a recall of 84.40%, a precision of 2.74% and an AUC of 95.57% were obtained on the test set. Figure 5 shows the ROC curve for this model and choice of hyper-parameters.

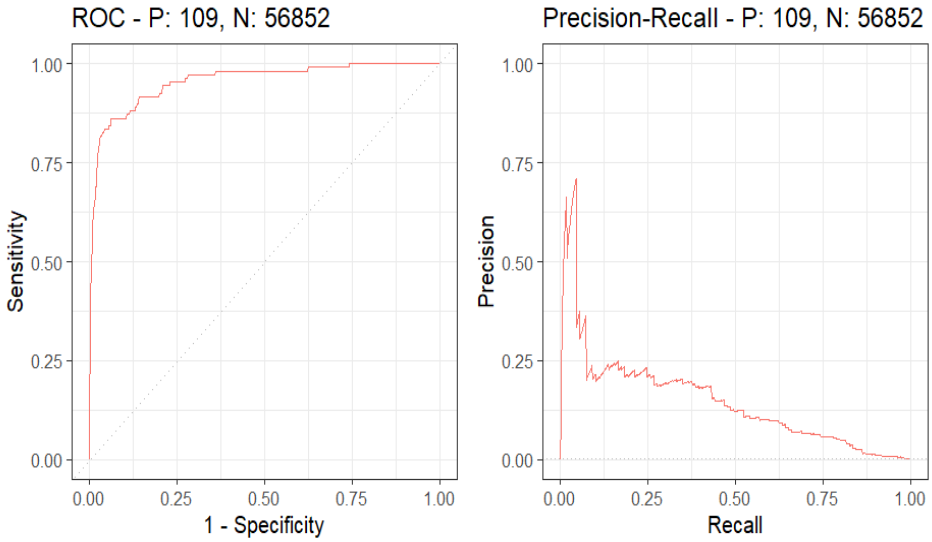

Figure 5: ROC curve for variant 1 of Isolation Forests with 100 trees.

Variant 2: Weighted Fair-Cut Forest The second variant tried is more known as the Fair-Cut Forest. This also accounts for the sample size and number of trees as parameters. It also has a parameter that indicates the probability of choosing the threshold on which to split a variable and we are weighting instances since this proved to make the model perform better. The results for the second variant are shown in table 4.

Table 4: 5-fold cross validation results for the weighted Fair-Cut Forest

Number of trees	Mean Recall	Mean Precision	Mean AUC
1	0.8073	0.0045	0.8302
10	0.8431	0.0130	0.9250
20	0.8591	0.0166	0.9351
50	**0.8813**	**0.0143**	**0.9365**
100	0.8751	0.0153	0.9365

We varied only one hyper-parameter in the 5-fold cross validation procedure: the number of trees. The sample size was set to be 500. The number of trees tried were 1, 10, 20, 50 and 100 trees. From table 4 one can notice that the number of trees with the highest recall is 50, which also accounts for one of the highest AUCs. Regarding the precision, there is a significant drop again comparing with the results from the Random Forests approach. After training the whole dataset using 50 trees, a recall of 88.07%, a precision of 1.50% and an AUC of 94.28% were obtained on the test set. Figure 6 shows the ROC curve for this model and choice of hyper-parameters.

One last thing to note about the Isolation Forests variants are their computational efficiency,

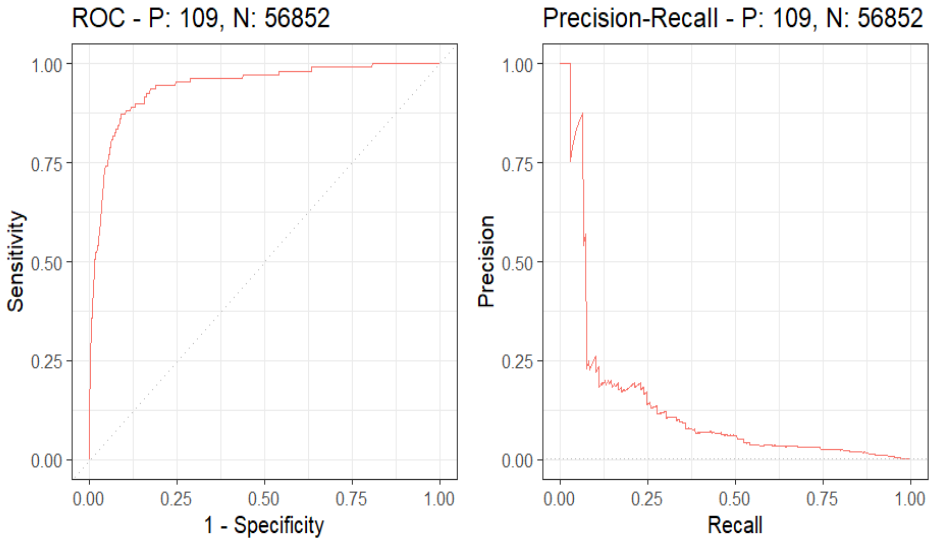

Figure 6: ROC curve for variant 2 of Isolation Forests with 50 trees.

these models where the quickest to train in less than minutes. Considering the size of the dataset, this is impressive and at the same time they achieve very good results.

3.1.3 Density-Based Clustering (DBSCAN)

This approach is implemented using only the first 5 principal components of the dataset, thus around the 45% of the variance is explained. This is done so that the algorithm is computational feasible. Also, the train data is scaled and the test (or validation) data is transformed using the same parameters from the train set. The radius parameter was changed and evaluated to determine the best for our purpose and the metrics used to evaluate this model are recall and precision, since this model is not a probabilistic classifier, there is no threshold to vary and plot a ROC curve to show.

Table 5: 5-fold cross validation results for the weighted Fair-Cut Forest

Radius	Mean Recall	Mean Precision
0.05	**0.9925**	**0.0022**
0.1	0.9747	0.0031
0.25	0.8863	0.0105

From table 5 the best radius is 0.05. The model is trained with this hyper-parameter and evaluated using the test set. The recall obtained is of 1, the precision is 0.24% and since we have a perfect recall makes sense to mention the accuracy which is 26.60%. This is clearly a bad model because it is basically classifying most of the predictions as fraudulent, which is not

desirable at all. One of the reasons might be that due to computational efficiency, we are only using a small set of the features and therefore the model performs poorly.

3.2 Evaluation of models

Now that the hyper-parameters for the models have been chosen, table 6 shows the results of the models with their best hyper-parameters run on the test set.

Table 6: Results on test set for model evaluation

Model	Recall	Precision	AUC
SMOTE Random Forests	96.15%	99.33%	98.06%
Weighted Random Forests	**80.73%**	**92.63%**	**90.36%**
Variant 1 Isolation Forests	84.40%	2.74%	95.57%
Variant 2 Isolation Forests	88.07%	1.50%	94.28%
DBSCAN	100%	0.24%	-

From table 6 one can notice that two of the models are over-fitted. The SMOTE approach seems too perfect to be real and therefore it should not be chosen. By the other hand, DBSCAN has a perfect recall, this means that the model is basically classifying most (or all) of the transactions as fraudulent. This is certainly not desirable and for these reasons, these models are discarded.

The most reasonable and balanced model is the Weighted Random Forests. It presents no indication of over/under-fitting it has a high recall (higher than 80%), a high precision (higher than 90%) and the AUC is also above 90%. So it is a very good and balanced model. The two variants of Isolation Forests present both high recall and AUC (higher than Weighted Random Forests) but a significant low precision and for this reason these two models are discarded.

4 Conclusion and future work

To sum up, two methods to treat the imbalance problem were implemented: the SMOTE over-sampling technique and the weighting of instances approach. The latter showed a great performance for treating the imbalance dataset using the Random Forests algorithm. Furthermore, a variety of algorithms were tried and implemented ranging from supervised and unsupervised learning. These include Random Forests, 2 variants of Isolation Forests and DBSCAN. One class SVM was also tried but it never converged in a reasonable time and so it was discarded by already having good and efficient results with other algorithms.

Regarding the results, it does not hurt to have a low precision in monetary terms. However, that could make clients annoyed if all of their transactions are being rejected and it does not make sense to discard every single transaction. Hence it is still important to maintain or assure a certain level of precision in the metrics. For this reason the Weighted Random Forests model was selected as the final and best model, with a recall of 80.73%, a precision of 92.63% and an AUC of 90.36%. Regarding the other learning and sampling methods used, the SMOTE approach showed signs of over-fitting and DBSCAN was basically classifying every transaction as fraudulent. It is of importance to highlight the computational efficiency of Isolation Forests, it did not take long to run even though the dataset has a great size. However, Isolation Forests presented a very low precision rate, which is not desirable.

For future work is left to evaluate in a better way why is the SMOTE approach over-fitting the models and combine anomaly detection algorithms with sampling techniques to enhance the overall performance. Also, it could be interesting to solve this problem using Neural Networks and auto-encoders.

Finally, it is worth mentioning that much of the information about the learning methods and procedures was obtained from [6] and from the lecture notes.

References

[1] L. M. Manevitz and M. Yousef, "One-class svms for document classification," *Journal of machine Learning research*, vol. 2, no. Dec, pp. 139–154, 2001.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[3] H. Yu, "Svmc: Single-class classification with support vector machines," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, ser. IJCAI'03, Acapulco, Mexico: Morgan Kaufmann Publishers Inc., 2003, pp. 567–572.

[4] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[5] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discov. Data*, vol. 6, no. 1, Mar. 2012, ISSN: 1556-4681. DOI: 10.1145/2133360.2133363. [Online]. Available: https://doi.org/10.1145/2133360.2133363.

[6] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. [Online]. Available: https://faculty.marshall.usc.edu/gareth-james/ISL/.

[7] A. Dal Pozzolo, O. Caelen, R. Johnson, and G. Bontempi, "Calibrating probability with undersampling for unbalanced classification," Dec. 2015. DOI: 10.1109/SSCI.2015.33.

[8] N. Olver, *Lecture 10 – Dimensionality reduction*. London School of Economics and Political Science, 2024.

[9] N. Olver, *Lecture 2 – Evaluation methods, bias–variance tradeoff, and the k-nearest neighbours algorithm (KNN)*. London School of Economics and Political Science, 2024.

[10] N. Olver, *Lecture 9 – Clustering*. London School of Economics and Political Science, 2024.