FH University of Applied Sciences TECHNIKUM WIEN

Machine Learning: Anomaly Detection

by Juan Manuel Alonso & Florencia Cavallin

Assignment

In this exercise, you shall work with

- The credit card fraud dataset
- The IEEE Fraud detection dataset

Your task is to work with anomaly / outlier detection methods to find the fraudulent cases. You should include

- Methods based on a two-class classifier, with sampling methods to help the classifier learn the minority class better, and can also consider using e.g. a higher cost associated with misclassification of the fraudulent cases.
- Semi-supervised methods such as one-class SVM, Gaussian Mixture Models
- Unsupervised methods, such as LOF, etc.

Compare with published results and try to get close to those results.

You can e.g. use the methods provided in sk-learn as an inspiration, but there are also other approaches published that can be reused. You can reuse existing code from the web as well.
Make sure to chose fitting criteria for evaluation! Best is to compare multiple ones, e.g. recall, an F-Measure, AUC, ...

Installation

```
$  pip3 install -r requirements.txt
```

Running

Custom hyperparameters in a textfile i.e. “./configs/config.txt”.

```
$  python3 experiments.py ./configs/config.txt
```

A results folder will contain a timestamp directory with the latest results, depending on the dataset.

Datasets

Credit Card Dataset[1]

The creditcard dataset contains transactions made by credit cards in September 2013 by european cardholders in the time lapse of two days.

There were 492 frauds out of 284,807 transactions. The dataset is highly unbalanced as the fraud transactions account for 0.172% of all transactions.

It contains only numeric input variables. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

IEEE Dataset[2]

This dataset has been used in a competition which aim was to benchmark machine learning models on a challenging large-scale dataset.The data comes from Vesta's real-world e-commerce transactions and contains a wide range of features from device type to product features.

The data is broken into two files identity and transaction, which are joined by TransactionID. Not all transactions have corresponding identity information and the idea is to predict the probability that an online transaction is fraudulent, as denoted by the binary target *isFraud*.

Detection Methods

There are three broad categories of anomaly detection techniques existing.

Unsupervised anomaly detection techniques detect anomalies in an unlabeled test dataset under the assumption that the majority of the instances in the dataset are normal. The techniques chosen for this category were Isolation Forest, Local Outlier Factor, an AutoEncoder and One-Class Support Vector Machine.

Supervised anomaly detection techniques require a dataset that has been labeled as "normal" and "abnormal" and involves training a classifier. With these datasets, normal is the *fraud* setting and abnormal or 'novelty' is the *not fraud*. For this category we chose XGBClassifier.

Semi-supervised anomaly detection techniques construct a model representing normal behavior from a given normal training dataset, and then test the likelihood of a test instance to be generated by the learned model. For this category we chose Gaussian Mixture.

[1] https://www.kaggle.com/mlg-ulb/creditcardfraud

[2] https://www.kaggle.com/c/ieee-fraud-detection/overview/description

Preprocessing

Credit Card Dataset

As part of the preprocessing steps necessary prior to the training, the ‘Amount’ column and the ‘Time’ column were normalize, while the other columns seem to be already in such state.

IEEE Dataset

For this dataset, significantly larger than the previous one, a series of steps had to be made. These steps are a recollection of the most popular notebooks on the topic found at Kaggle.

First of all, some of the train and test identity columns do not match in the column label specifically, so this had to be corrected for. Secondly, the merging of the transaction and identity files is done. Thirdly, a grouping of the email column is done by their parent domain/company[3]. Fourth, the D columns are normalized for both the train and test dataframes[4]. Fifth, a broad feature engineering is done on a set of columns, i.e., dropping columns with only one value or columns with more than 90% of nulls[5]. Fifth, any infinite values are replaced by numpy’s NaN[6]. Sixth, the memory usage of the dataframes is greatly reduced[7], to 30% of its original size. Lastly, categorical features are encoded appropriately.

Sampling

Three sampling techniques are available for the user to chose: undersampling, oversampling and SMOTE (Synthetic Minority Over-sampling Technique). All of them are used to adjust the class distribution of both of these greatly imbalanced datasets. The first two are done in a random manner, deleting instances of the majority *not fraud* class (undersampling). or adding to the training data copies of the minority *fraud* class (oversampling). Oversampling should increase the misclassification cost, while undersampling, which losses information, may yield too general learning. SMOTE generates artificial anomalies around edges of sparsely populated data regions, and was the one chosen for all runs, due to the fact that fraud data points are generated inside regions of novelty examples.

[3] https://www.kaggle.com/kabure/extensive-eda-and-modeling-xgb-hyperopt
[4] https://www.kaggle.com/cdeotte/xgb-fraud-with-magic-0-9600
[5] https://www.kaggle.com/artgor/eda-and-models#Feature-engineering
[6] https://www.kaggle.com/dimartinot
[7] https://www.kaggle.com/jesucristo/fraud-complete-eda

Results

Metrics

A major complication when working with unbalanced data is the choice of a metric to compare different techniques. If one were to choose accuracy, it would yield totally biased results, since the majority class will be selected as target naturally much more often than the minority class.

With 'positive' as being *fraud* in the context of this work, false positives are those data points predicted as *fraud* when they were not in reality, while false negatives are those predicted as *not fraud* when they actually were *frauds.* It is clear that false negative are critically more costly than false positives, because a fraud camouflaged as a normal transaction will be in prejudice of the client, while a transaction labelled as *fraud* when it was not only makes the client loose time, but not money.

In this sense, recall, the number of true positives divided by the number of true positive targets in the data, is more important than precision, the number of true positives divided by the number of all positive predictions. Since low recall means a high number of false negatives, it makes sense to use a metric such as F2 score, that add more weight or 'cost' for each false negative predicted.

Credit Card Dataset

Unsupervised

The unsupervised methods yielded the worst results compared to the other types of techniques. In particular, LOF (Local Outlier Factor) had the best F2 out of the 4 unsupervised methods, with a 0.142 score on the testing set. The AutoEncoder, Isolation Forest and OCSVM approaches had a 0.015, 0.010 and 0.012 respectively.

From the confusion matrices of Fig. 1, one can better understand the errors. For instance, IForest has perfect precision on the fraud class but poor recall. In those terms, OCSVM has both precision and recall as poor metrics.

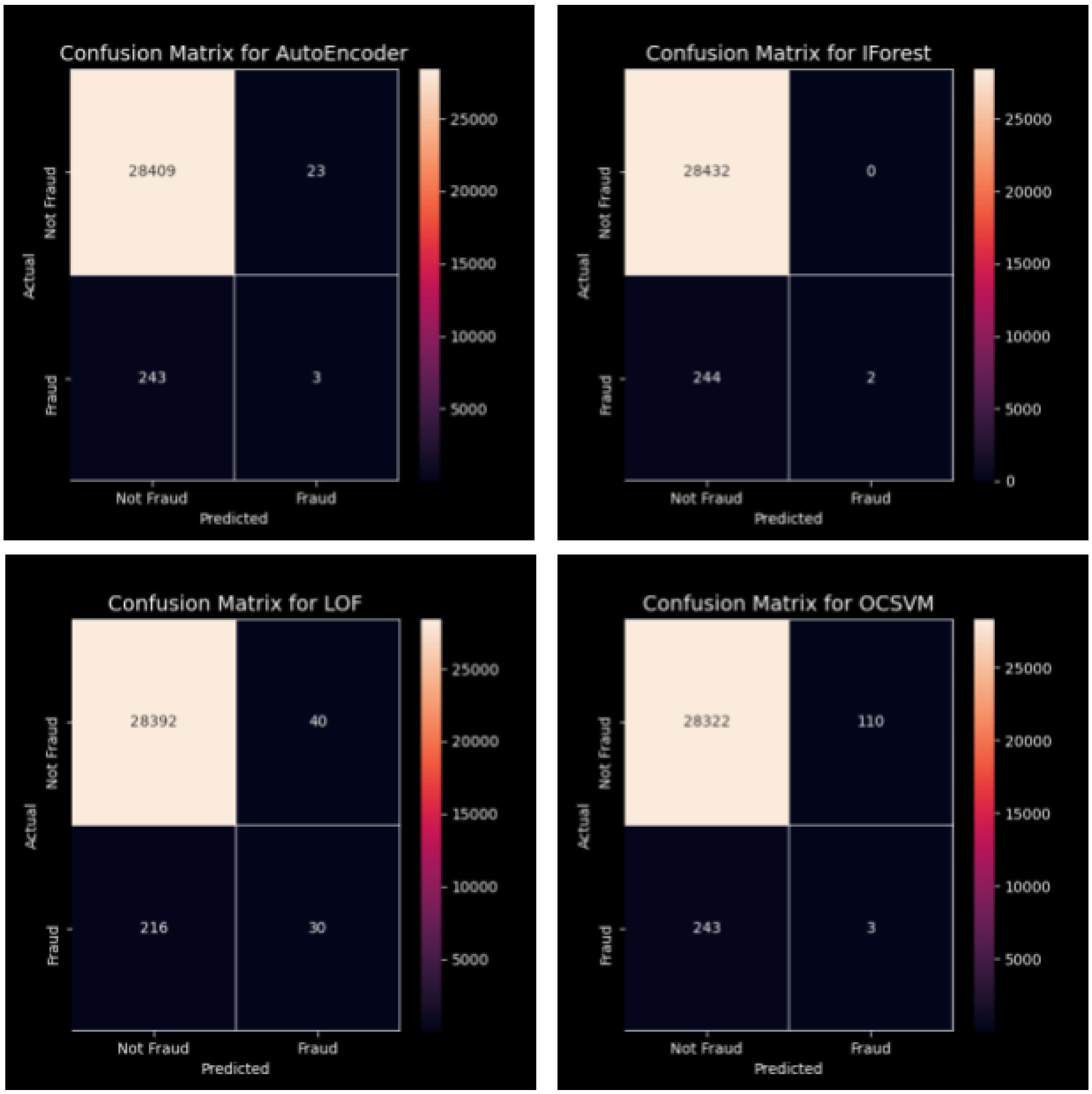


FIG 1. Confusion Matrices for Unsupervised techniques.

Semi-supervised

For this type of technique, use must assume that the data has labels for the normal or *not fraud* class. So the model, in this case, a Gaussian Mixture one, will learn only the normal behaviour. So only the probabilities of normal that are trained upon. During testing, we are interesting in knowing novelty instances, *frauds* or anomalies, whose probability of being normal is low. To determine the class for the testing set, a threshold hyperparameter is searched for, by means of a validation dataset separated at the beginning of the algorithm. The range for this threshold was set between -1000 and 0 in steps of 2. Those data points whose 'score' is below this threshold will be labeled as anomalies or *frauds*, and those above as *not frauds*.

To have an idea of what this threshold can have as a value, the mean and the standard deviation of the validation dataset are computed, and then a multivariate normal model is instantiated from those parameters. For each class, a threshold value is calculated by means of computing the median of the log of the probability density function for those data points of that particular class. The log is taken since the *pdf* values are very small.

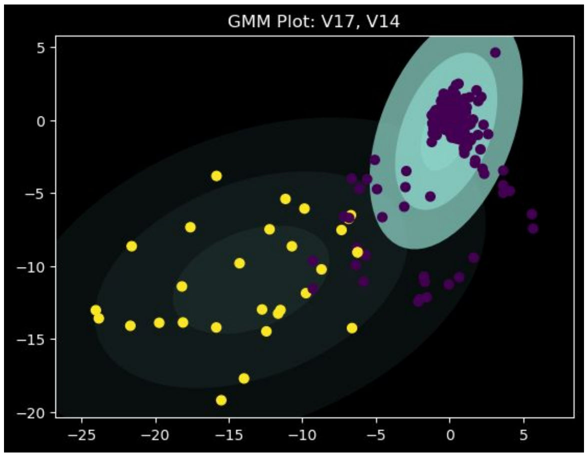
Not Fraud logpdf median:	-33.737
Fraud logpdf median:	-90.127
Best Threshold:	-158

For each split of the dedicated cross validation dataset, the threshold search was made, and the averaged scores are shown in Table 1.

Metric	***Mean*** +/ ***Std***
Precision	0.693 +/- 0.04
Recall	0.768 +/- 0.036
F2 Score	0.752 +/- 0.034
ROC-AUC	0.883 +/- 0.018

Table 1. Average scores of threshold hyperparameter search. CV 5 fold.

From the plot on the right, were two particular features were selected, one can observe the ellipses that describe with some strength the separate classes of the dataset.



Out of all the splits, the a best threshold is determined from the best local thresholds of each fold. This best value is determined by the highest F2 score, since it is the metric of interest we chose, as seen in the example of Fig. 2.

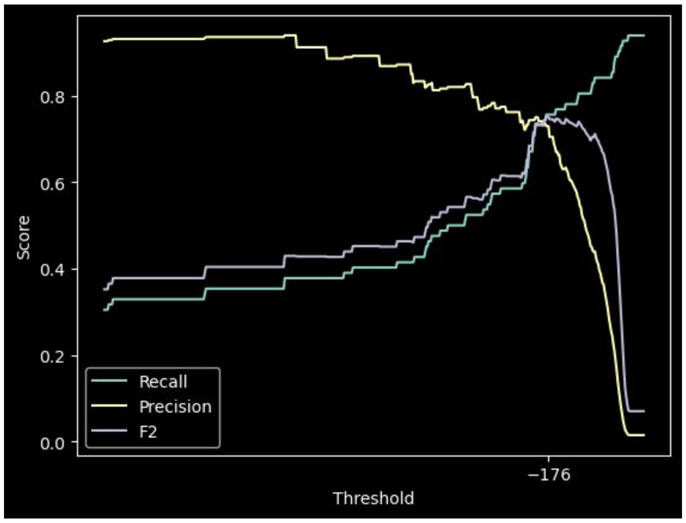


FIG 2. GMM threshold search CV fold example. Determining the best value.

From Fig. 3 one can notice how much better GMM results in terms of recall, compared to all the other unsupervised techniques. Indeed, the test F2 score was 0.744, thanks to a recall of 0.776 and a precision of 0.637, significantly better than the unsupervised alternatives. This means that learning only the normal behaviour can have an important impact in novelty identification.

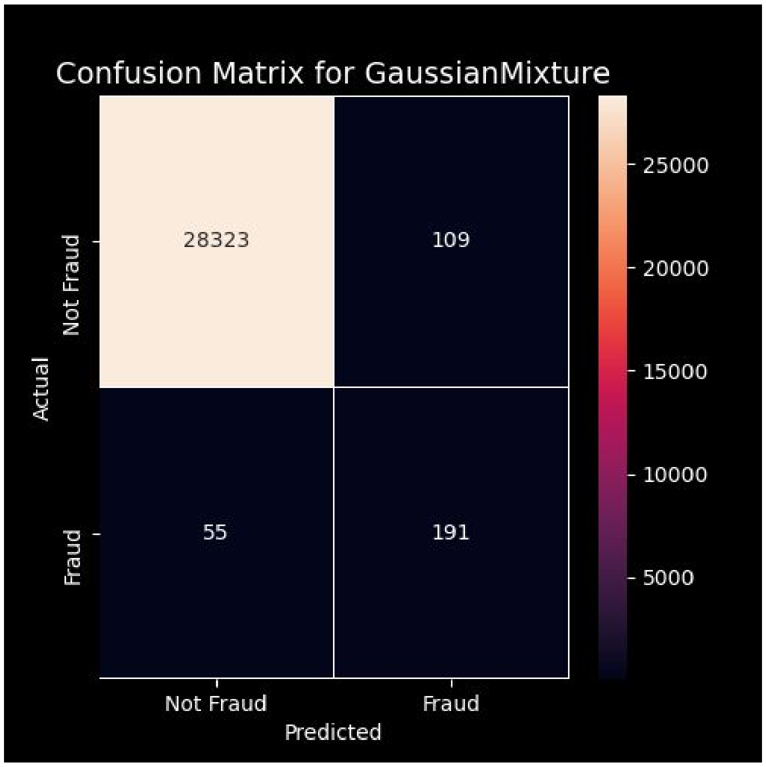


FIG 3. Confusion matrix for GMM.

Supervised

For this last type of technique, and based on the best performing Kaggle notebooks online, the XGBoost Classifier was chosen. This 'Extreme Gradient Boosting' technique is faster than the normal gradient technique thanks to its parallelized execution, among other benefits, it has an efficient handling of missing data[8].

Figure 4 shows the confusion matrix for XGBClassifier, whose comparison to GMM's results yields even better results. False negatives are halved, and the precision obtained is significantly better, from 0.637 with GMM to 0.973 with XGB. Together with a recall of 0.886, the final F2 score is a staggering 0.902, the highest of all techniques in this dataset. This may suggest that the representation of a Gaussian mixture model probability distribution has better novelty detection that any unsupervised method, but not as good as boosted decision trees of this supervised learning classifier.

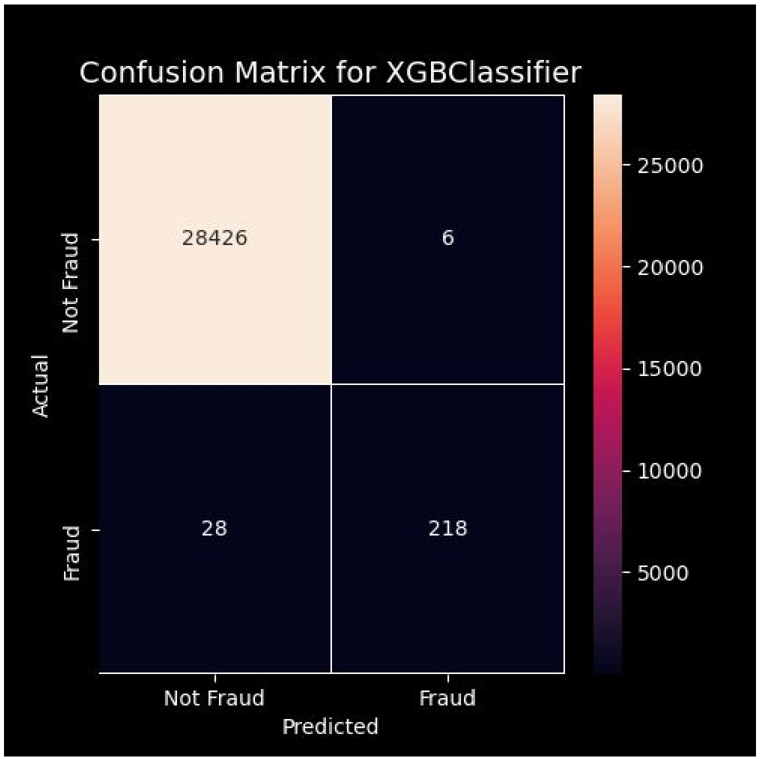


FIG 4. Confusion matrix for XGBClassifier.

[8] https://xgboost.readthedocs.io/en/latest/

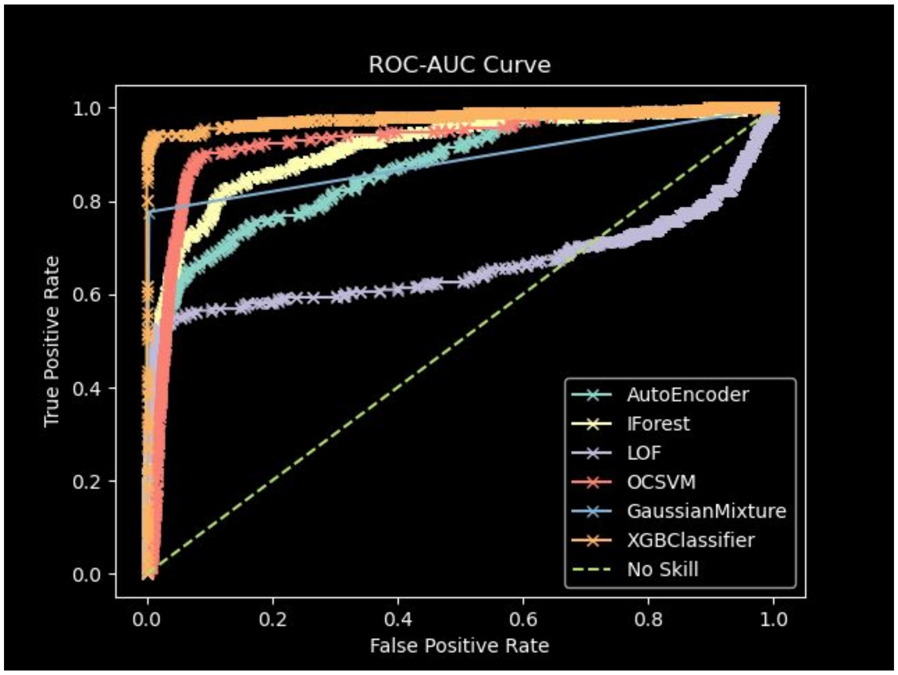


FIG 5. ROC-AUC Curves for all techniques

It is interesting to observe the ROC-AUC curves for all techniques in this dataset, though the most meaningful criteria of comparison is the F2 score. These curves are more of a tool to see how the precision and recall vary as one adjusts the decision threshold, which in this work was only searched for for the Gaussian Mixture semi-supervised technique. Also, one must take into consideration that the datasets used in this work are highly unbalanced.

Method	***Test AUC***
XGBClassifier	0.979
GaussianMixture	0.886
OCSVM	0.927
LOF	0.661
IForest	0.913
AutoEncoder	0.870

Table 2. *Test AUC* per method.

From Table 2, one would suggest that the unsupervised methods, which performed poorly in terms of F2, have high AUC scores. This means that there is another threshold, apart from the one used by the classifier's implementation, for which F2 is much better.

IEEE Dataset

This dataset differs completely in the way one measures the classifiers performance. Since the labels for the test dataset are unknown, one can only predict them and delegate the scoring to the official Kaggle kernel that takes care of the scoring for us.

Some compromises had to be made for the size of this dataset. For the OCSVM and LOF runs, the training datasets, that are sampled by SMOTE at every cross validation fold, were reduced to a tenth of their original size, for a total training size of about 200k rows, in addition to the 118 columns dropped at the preprocessing stage. This was done to reduce the total time spent, despite the fact that the predictions also were computationally expensive (500k rows).

Method	*Public Score*
XGBClassifier	0.895
GaussianMixture	0.500
OCSVM	0.726
LOF	0.240
IForest	0.769
AutoEncoder	0.269

Table 3. *Kaggle IEEE Fraud Detection public scores*.

Regardless of Kaggle's own scoring, we can still make notice of the training scores, though these are results of data already seen, which is not as scientifically valid as unseen testing data. Once again, the methods with any degree of supervision outperformed those unsupervised methods.

Method	*Mean* +/ *Std*
XGBClassifier	0.592 +/- 0.007
GaussianMixture	0.339 +/- 0.002
OCSVM	0.037 +/- 0.002
LOF	0.069 +/- 0.007
IForest	0.072 +/- 0.005
AutoEncoder	0.051 +/- 0.006

Table 4. Training F2 score per method, 5 fold CV.