Machine Learning Engineer Nanodegree

Capstone Project

Rodrigo Miike da Silva
August 19th, 2017

I. Definition

Project Overview

In this project, I'll try to solve a problem that affects most of the financial entities by using as resource the Credit Card Fraud dataset available in Kaggle[5]. Fraudulent transactions happen with a high frequency and the purpose is to obtain goods without paying for it. They are committed through identity/physical card theft or when personal data about accounts are leakaged[1]. Although the rate of fraudulent transactions are lower than 1%, the financial losses are huge due to the high amount value of the transactions[1].

Popular machine learning techniques such as supervised and unsupervised learning could be used to aid this problem. However, each approach have your own challenge and assumptions. Under the condition that fraudulent transactions incidence are highly imbalanced compared to legit transactions, anomaly detection is an amazing approach to find points that are not expected from the pattern.[3]

Chandola, Banerjee and Kumar[3] provide a research list with anomaly detection techniques and its applications. Thus, providing a general idea about how to apply anomaly detection under a domain. Barkan and Averbuch[4] propose the application of a statistical model on an original representation of the data in order to detect anomalies.

Problem Statement

The problem consists in find out which transaction is legit or fraudulent in order to minimize financial losses. Since there is an imbalance between them, an approach through anomaly detection could be effective.

I'll be doing this through Unsupervised Learning algorithm Expectation Maximization applying Multivariate Gaussian Mixture Model and Multivariate t Mixture Model and compare them. With models we could get an insight about the structure of the data and develop a way to predict the transactions.

Metrics

Due to imbalanced nature of the dataset, the Area Under the Precision-Recall Curve (AUPRC) would be a better approach to measure the performance since the precision is highly affected due to the false positives. The AUPRC is a scalar value that summary the

Precision-Recall (PR) curve. The summary is calculated by integrating out the area under the PR curve and it ranges from 0 to 1 (greater is better).

On Scikit-learn library, the *average_precision_score* function calculates this area by receiving the labels and the predicted labels.

II. Analysis

Data Exploration

The Credit Card Fraud dataset is available on Kaggle[5] providing resource for study about fraud detection. The dataset contains credit card transactions made by european cardholders in September 2013. The data provides 497 frauds out of 284807 transactions in total (0.172%) that occurred in two days. This is a highly imbalanced dataset and might present a challenge to model it.

	Class	Amount	V28	V27	V26	V25	V24	V23	V22	V21	...	V9	V8	V7	V6	
0	0	149.62	-0.021053	0.133558	-0.189115	0.128539	0.066928	-0.110474	0.277838	-0.018307	...	0.363787	0.098698	0.239599	0.462388	-0.338
1	0	2.69	0.014724	-0.008983	0.125895	0.167170	-0.339846	0.101288	-0.638672	-0.225775	...	-0.255425	0.085102	-0.078803	-0.082361	0.060
2	0	378.66	-0.059752	-0.055353	-0.139097	-0.327642	-0.689281	0.909412	0.771679	0.247998	...	-1.514654	0.247676	0.791461	1.800499	-0.503
3	0	123.50	0.061458	0.062723	-0.221929	0.647376	-1.175575	-0.190321	0.005274	-0.108300	...	-1.387024	0.377436	0.237609	1.247203	-0.010
4	0	69.99	0.215153	0.219422	0.502292	-0.206010	0.141267	-0.137458	0.798278	-0.009431	...	0.817739	-0.270533	0.592941	0.095921	-0.407

5 rows × 31 columns

Fig. 1 A sample from credit card fraud dataset

The dataset contains only numerical inputs. The features named as V1 to V28 are the principal components of the PCA transformation. Due to confidentiality issues, the original information could not be provided. The amount and time variables were the only ones that have not been transformed by PCA. The last feature “Class” labels if the transaction is fraudulent or legit.

	count	mean	std	min	25%	50%	75%	max
Class	284807.0	1.727486e-03	0.041527	0.000000	0.000000	0.000000	0.000000	1.000000
Amount	284807.0	8.834962e+01	250.120109	0.000000	5.600000	22.000000	77.165000	25691.160000
V28	284807.0	-1.206049e-16	0.330083	-15.430084	-0.052960	0.011244	0.078280	33.847808
V27	284807.0	-3.660161e-16	0.403632	-22.565679	-0.070840	0.001342	0.091045	31.612198
V26	284807.0	1.699104e-15	0.482227	-2.604551	-0.326984	-0.052139	0.240952	3.517346
V25	284807.0	1.453003e-15	0.521278	-10.295397	-0.317145	0.016594	0.350716	7.519589
V24	284807.0	4.458112e-15	0.605647	-2.836627	-0.354586	0.040976	0.439527	4.584549
V23	284807.0	5.367590e-16	0.624460	-44.807735	-0.161846	-0.011193	0.147642	22.528412

Fig. 2 A summary of some of the features of the dataset

About the dataset:

- The mean of all the features Vs are next to zero.
- Most of the features Vs have outliers above 75% and below 25%.
- The feature Amount have a huge range between quantile 75% and the max value.
- There aren’t missing values on the dataset.

Exploratory Visualization

The countplot below shows the ratio between legit and fraudulent transactions. Fraudulent transactions appear to be only 0.172% of all transactions. Due to this, a classifier might classifies all the transactions as legit. So, in order to deal with this issue an over-sampling or under-sampling approach should be considered.

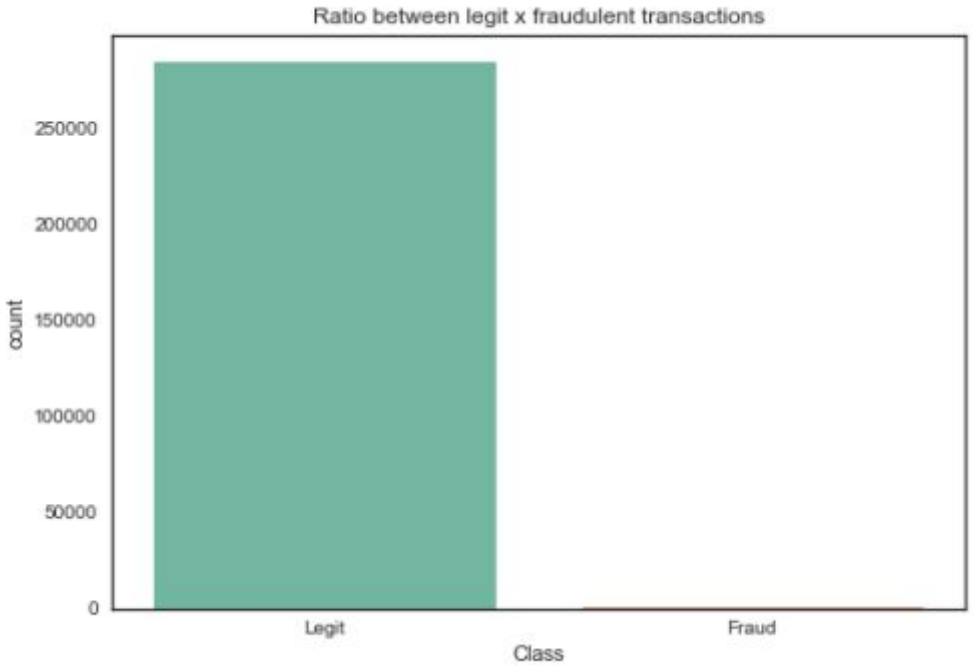

Fig. 3 The highly imbalanced might present a challenge to deal with

The features V1 to V28 are distributed around zero (actually reflects the summary), however each one with your own peculiarity. By the extension of the x-axis, we can see that there are some outliers.

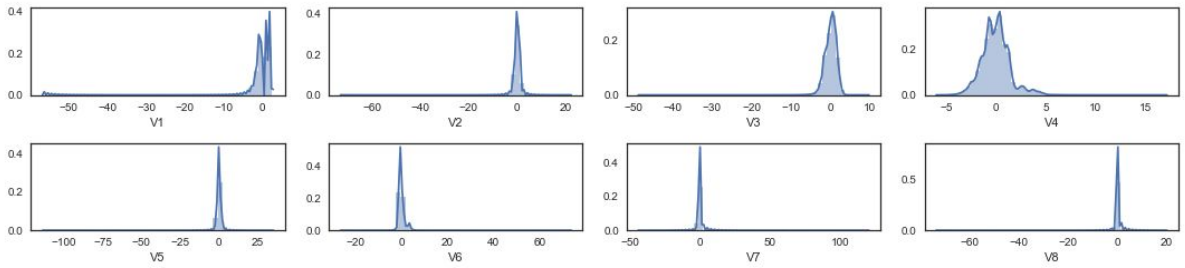

Fig. 4 Some of the Vs features distribution

The feature Amount follows a highly skewed distribution with a long tail pointing to the right side. Since It's a positive-skew distribution, a log transformation might handle this problem.

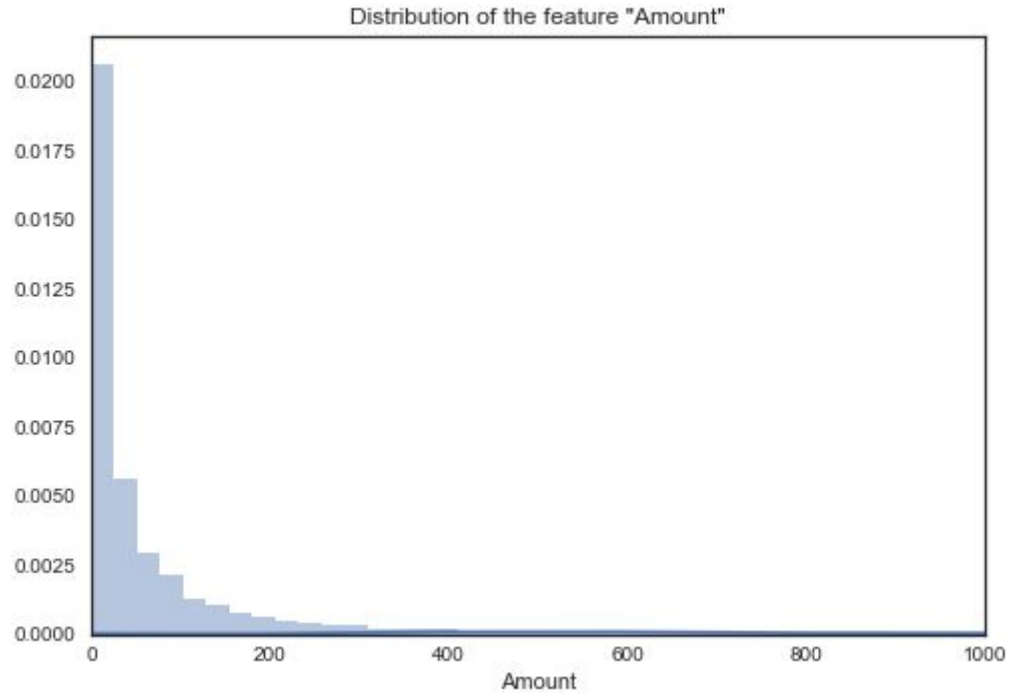

Fig 5 The positive skew distribution of the feature Amount

The feature Time presents some kind of a bimodal distribution. Since, the dataset contains transactions of two days, this might be the reflection of it. Since I won't be working with recurrent models, this feature might be ignored.

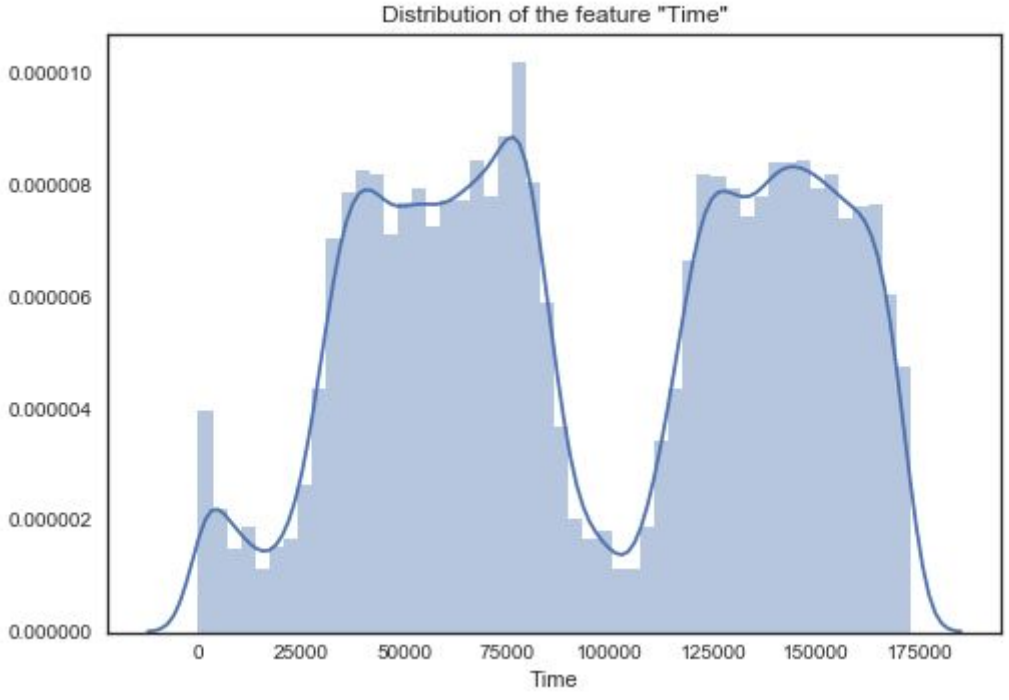

Fig 6 A bimodal distribution of the feature Time

Algorithms and Techniques

The classifier is a mixture model. It implements an expectation-maximization (EM) algorithm which fits a model on the data. The models are Gaussian mixture model and a t distribution mixture model.

The EM algorithm is a probabilistic soft-assignment unsupervised learning technique. Due to the imbalanced nature of the dataset, this approach should be nice since:

- It's hard to get fraudulent samples;
- It gives scores in order to support business rules;
- It should give an insight about the structure of the data or the relationship between the features;

The purpose of the EM algorithm is to estimate the parameters of the dataset that maximize the likelihood of each point under a model. And it's done by firstly guessing the parameters and then iterating over the E-step (Expectation step) and M-step (Maximization step) until convergence[9].
The E-step consists on calculating the probabilities with the current parameters and the M-step consists on taking the average of the probabilities in order to maximize the likelihood of the model while updating the current parameters.
The convergence condition is given when the probability doesn't change after iterating the E-step and M-step.

Both Gaussian and t distribution have in common the following parameters:

- The mean ($\boldsymbol{\mu}$);
- The standard deviation ($\boldsymbol{\sigma}$); or in multivariate model the covariance matrix ($\boldsymbol{\Sigma}$);

The t distribution has additionally the degrees of freedom to estimate.

After estimating the parameters, the classifiers should be able to calculate the score of a sample point.

Benchmark

I'll be applying two models on the Credit Card Fraud dataset in order to use them as performance measurement reference. The models are Gaussian Naive Bayes and One-class SVM.

The Gaussian Naive Bayes[8] is a supervised learning algorithm that use as a model a Gaussian distribution. By tuning the hyper-parameters, the Gaussian NB could properly handle a highly imbalanced problem.

The One-class SVM is an unsupervised learning algorithm that classifies new data as similar or different of the training set[7]. Here, we can use the classifier to model the boundary between the legit and fraudulent transactions.

The following table summarizes the results.

	Precision	Recall	AUPRC
GaussianNB	0.68	0.99	0.68
One-class SVM	0.66	0.72	0.47

III. Methodology

Data Preprocessing

First of all, we should choose the features that are most relevant. That is, the features that most tell us how the fraud is. We will do this by running a Random Forest classifier. Since decision trees aren't sensitive to outliers, it won't be necessary preprocess the data. However, I'll be undersampling the data in order to train the classifier.

Random Forest is an ensemble technique and it receives a lot of parameters. So, I'll be running the Random Forest with a GridSearch to choose the best parameters.

After running the Random Forest, the classifier performed very well. The classifier got an AUPRC score of .81. The five most relevant features that the classifier described were the features V11, V15, V18, V13 and V8.

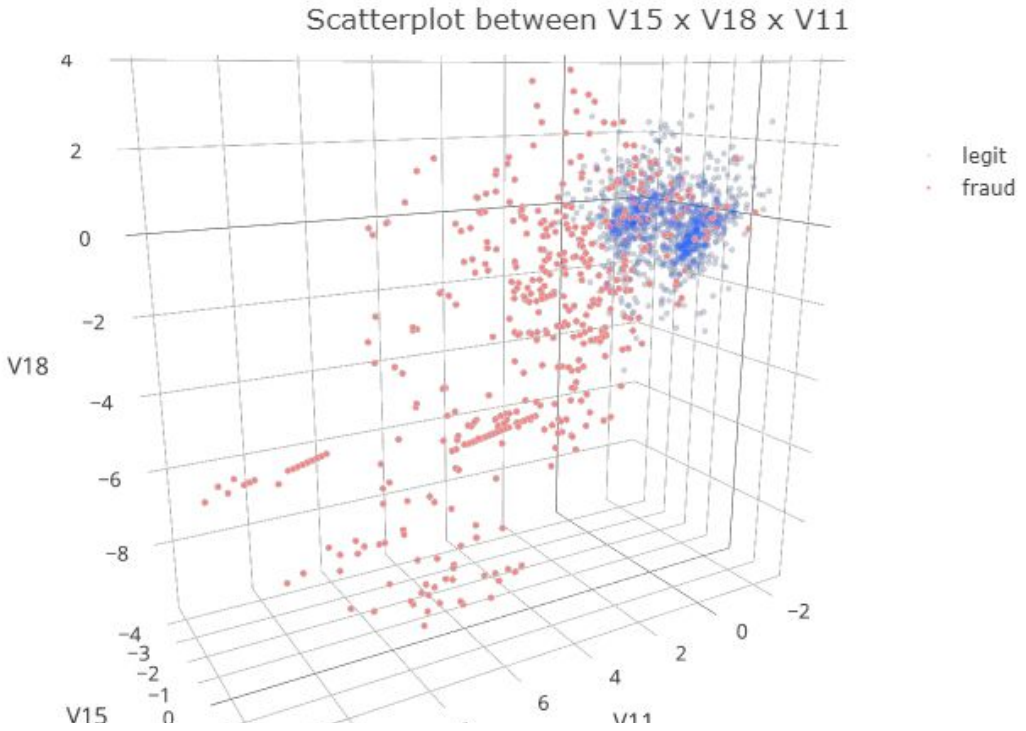

Fig 7. Relationship between three most relevant features

It’s clear to see the relationship between the three most relevant features. It seems that there are two clusters of the legit transactions closer to each other and the fraud transactions are spread away of them.

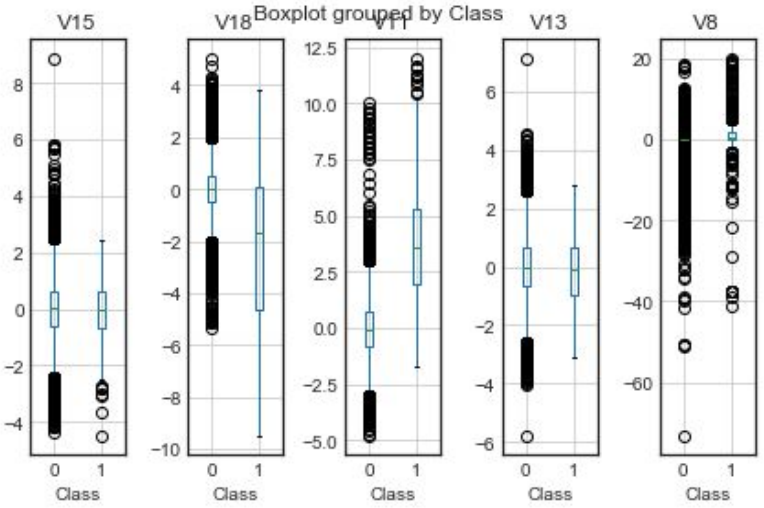

Fig 8 Distribution of the features

By taking a closer look on the features distribution, we can see that there are a lot of outliers. It might influence on the predictions of those models which use the mean. Since there is a lot of legit samples and the fraud ones are scarce, I’ll be removing the outliers of the legit samples and keep the fraud ones. The removed outliers were the ones above the 75th percentile plus 1.5 x IQR and the ones below the 25th minus 1.5 x IQR.

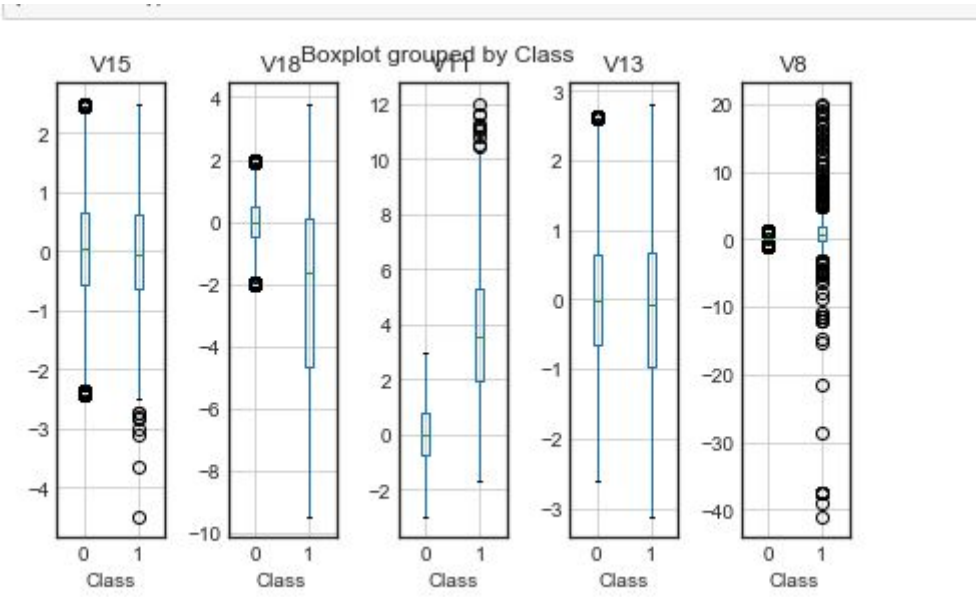

Fig 9 Distribution of the features after outlier removal

It was removed about 36000 samples. I guess it was a small quantity compared to the whole size of the legit samples.

Implementation

The models that I used were the Gaussian Mixture model provided on scikit-learn library and the t Mixture model implemented by me, but proposed by D. Peel and G. J. McLachlan[6].
The idea was to compare both of the models by estimating the parameters on training set with a fraud samples and on another training set without the fraud samples.

The main process was to estimate the parameters by using a undersampled training set and after that figuring out a threshold which best maximizes the AUPRC by using the respective validation set.
With only one mixture, we can calculate the mahalanobis distance and use it with the chi-squared distribution in order to outline a Confidence Interval (C.I.) under the estimated parameters.
With more than one mixture, it's hard to calculate the C.I., so we'll be using the log likelihood in order to outline the boundary between the legit transactions and the fraud transactions.

The prediction is similar to the One-Class SVM. Those transactions that are between the C.I. are considered legits, otherwise are frauds.

Refinement

The parameter used for tuning was the Confidence Interval (or the likelihood if more the one mixture) in order to outline the decision boundary.
The initial solution was with only one mixture and the results are summarized on the following table.

Model	Training Set	C.I.	Precision	Recall	AUC
Gaussian	No fraud	99.97%	.66	.94	.62
	With fraud	99.80%	.56	.97	.54
t	No fraud	99.991%	.628	.993	.625
	With fraud	99.99%	.62	.98	.61

As saw in the fig. 7, the fraud transactions are somehow outliers. This fact underperformed the Gaussian model. However, the t model kept the performance. This show us how robust the t model performs under outliers.

After that, I tried to increase the number of mixtures. Since the mixtures try to fit the shape of the data, it might increase the performance of the models.
In order to set a threshold, the log likelihood was used as tuning parameter.

Model	Training Set	Components	Precision	Recall	AUC
Gaussian	No fraud	3	.70	.98	.69
	With fraud	2	.63	.81	.51
t	No fraud	7	.69	.97	.676
	With fraud	2	.62	.95	.59

I had some issues building t Model with fraud on the training dataset. The model wasn't fitting due to ill-defined empirical covariance. The solution was to keep a small number of components.

The final solution was the Gaussian Model which got the highest AuC score.

IV. Results

Model Evaluation and Validation

The model used a validation set and a training set that did not have outliers. The main idea was to model what you expect to be normal and afterwards you'll be able to tell what is anomaly.

Both models performance were similar, however the one that best performed was the Gaussian Mixture with 3 components.

In order to verify if the model generalized the problem, a noise data was generated and tested against the models.

Justification

My final solution is the Gaussian Mixture with 3 component.
The results are summarized below:

- AuC: 0.69;
- Precision: 0.70;
- Recall: 0.98;

Comparing it with the benchmarks, the Gaussian Mixture performed better than both of the them, however not significantly better than GaussianNB which scored an AuC 0.68.
I'd like to point out an advantage over the GaussianNB that we don't need the labels since it's an Unsupervised Learning technique (under certain domains is hard to acquire), even though we used the labels to build the model.

We could take the scores on the validation set and then plotted a histogram in order to take a reference where the threshold should be.
For example, the threshold of the Gaussian Model was -16.19. It's close to the left tail of the distribution.

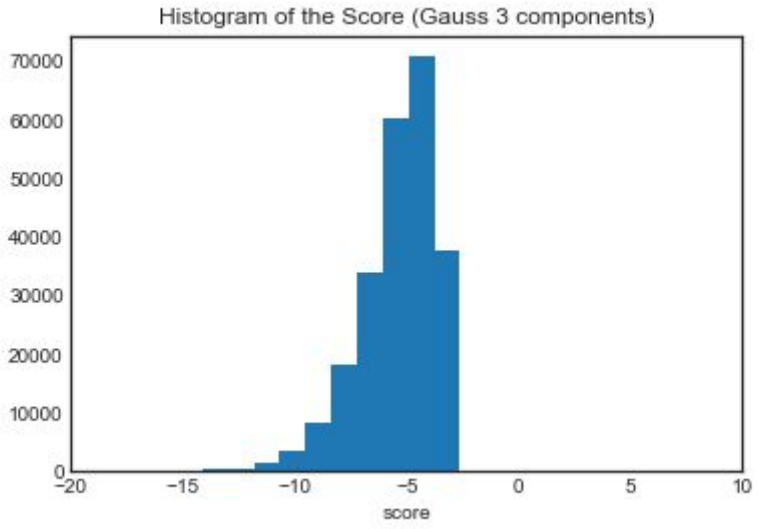

Fig 10 Scores of the validation set can gives an insight about the threshold

Another thing to notice is that most of the transaction might have a score next to -5 and as the score goes to $-\infty$, the probability to be a fraud increases.

V. Conclusion

Free-Form Visualization

The real point of a fraud prediction model is to predict frauds while avoiding the false positives. There is a tradeoff between the false negatives and the false positives. So, depending the domain where you want to apply it, this things should be considered. Under the scenario of a credit card fraud detection, the false positives mean that you're denying a legit user to spend his money. If the false positive rate is high, it might cause some issues to the bank. The bank might be avoiding a lot of frauds, however it might be losing the current customers.

The following figures show us that the models were able to capture most of the noise data. The points that are closer to the cluster are more difficult to find out if is a fraud or not. Consequently, the frauds that are embedded inside the cluster would be always predicted as legit transactions (false positives). So, in order to capture the embedded frauds, another approach should be considered.

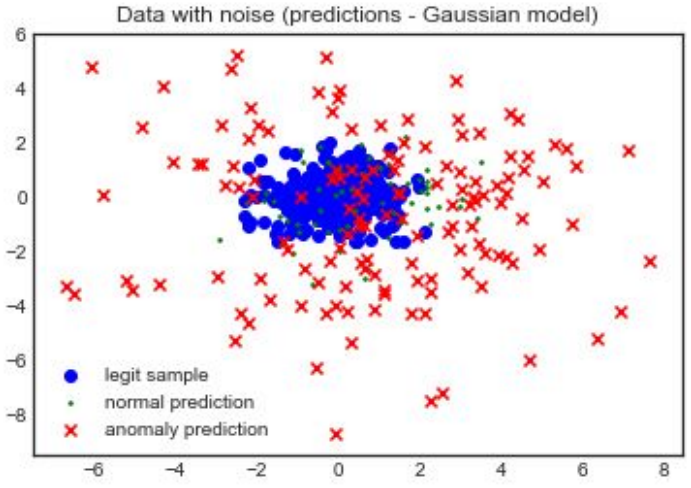

Fig 11 Gaussian Mixture prediction on a noise data

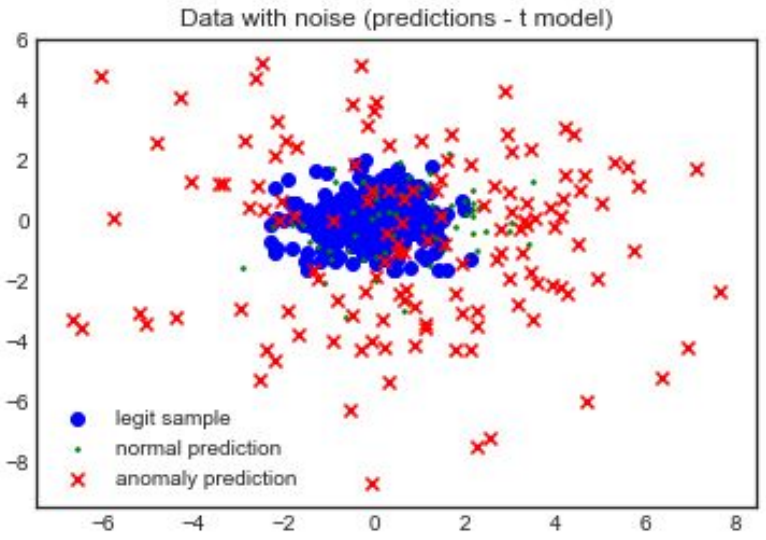

Fig 12 t Mixture prediction on a noise data

Reflection

The process used for this project was:

- Finding a public dataset;
- Data exploration and visualization;
- Define a metric based on nature of the dataset;
- Feature selection and outlier removal;
- Building benchmarks;
- Implementing the Multivariate t Mixture;
- Training the EM models;
- Comparing the results with the benchmarks.

I had a lot of trouble implementing the Multivariate t Mixture. First that it is hard to grasp the math stuff and the papers have a lot of complex formulas. I had to study a lot about probability distribution, linear algebra and some calculus stuff. The first implementation was taking minutes to train the model. The second implementation was based on the scikit-learn Gaussian Mixture. This one was taking seconds to train the model. So, it was a great improvement.

Both the EM models fitted my expectations, however I was expecting the Multivariate t Mixture to handle the problem better than the Gaussian Mixture. Even if the Multivariate t is less sensible to outliers than the Gaussian Mixture, it doesn't mean that it's bullet proof to outliers.

I guess that both models might be useful to understand the structure of the dataset and the advantage is that you don't need the labels (that might be hard to get under some scenarios).

Improvement

The model can be improved by applying a novelty approach. The main idea of the anomaly detection was to find out outliers under the distribution parameters. And since the business domain is dynamic, there is a chance to a new legit cluster appear far away from the old cluster. The novelty approach would identify this new cluster and then we would be able to identify it as legit or not.

There was a technique that I was wishing to implement, but I find it so complex that I gave up. The diffusion map[4] is a dimensionality reduction technique that gives a low dimensional representation of the original data. The new dimension is given by probabilistic affinity between the samples.
I was expecting to apply the EM on the dataset after applying the dimensionality reduction. I guess that would improve the model, because the outlier would be more highlighted.

References

[1] Wikipedia - Credit Card Fraud
[2] Dunning, T., Friedman, E.: Practical Machine Learning - A new look at Anomaly Detection.
[3] Chandola, V., Banerjee, A., Kumar, V.: Anomaly Detection - A Survey.
[4] Barkan, O., Averbuch, Amir.: Robust Mixture Models for Anomaly Detection.
[5] Kaggle - Credit Card Fraud Dataset
[6] Peel D., McLachlan G. J.: Robust Mixture modelling using the t distribution.
[7] Scikit-learn. One-class SVM with non-linear kernel (RBF). [ONLINE] Available at: http://scikit-learn.org/stable/auto_examples/svm/plot_oneclass.html. [Accessed 20 August 2017]
[8] Scikit-learn. Naive Bayes. [ONLINE] Available at: http://scikit-learn.org/stable/modules/naive_bayes.html#gaussian-naive-bayes. [Accessed 20 August 2017].
[9] McLachlan G. J., Krishnan T.: The EM Algorithm and Extensions - Second Edition