OML final project: credit card fraud detection

Lorenzo Di Filippo s277961
Pascal Christopher Iuliano s276181
Fabio Veneto s276211

January 2019

Abstract

In this report we're going to analyze the different performance of a variety of machine learning methods applied to credit card fraud detection. What we're going to show is that following a systematic procedure that includes undersampling the dataset and a second PCA-based dimensionality reduction performance increases and computational time decreases.

1 Data Preprocessing

1.1 Data Description

The dataset contains records of 199365 transactions, classified between fraud(1) and not fraud(0). PCA has been applied for privacy issues. The dataset has then 30 features in total: privatized features(V1-V28),Time and Amount. The dataset does not show neither any null nor NaN value. The main problem of our data is the strong bias towards class 0(0,17% frauds). In the context of classifiers it is a serious problem that does not allow to rely on the accuracy for performance evaluation.

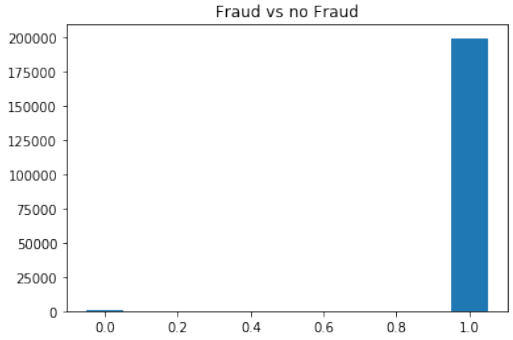


Figure 1: Extreme imbalance of the dataset

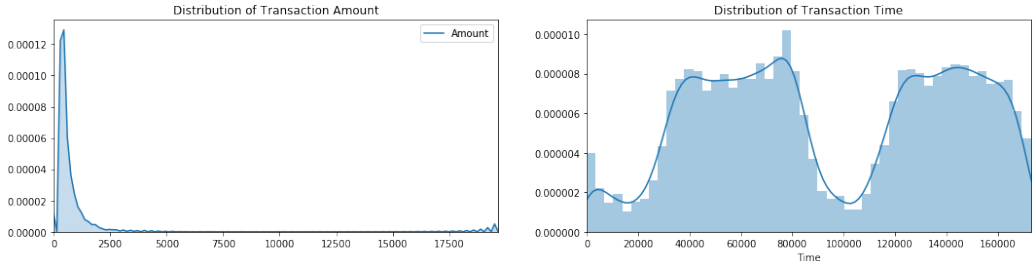


Figure 2: Time and amount distribution

1.2 Features analysis

Analyzing the only two easily interpretable features we notice the strong skew of the distribution of the Amount and the bimodal nature of the distribution of Time. What the graphs show is that the transaction amounts are mainly small(as expected).There are also some evident outliers in the extrema of the tail but we preferred not to ignore them because it is highly improbable that they correspond to some kind of measurement error. In this sense they could be helpful because they could be associated somehow to frauds(see later for the correlation analysis between Class and Amount). As far as Time is concerned, the two zones of maximum probability in time distribution could be associated to the morning of two subsequent days. Time could be a significant feature because, for example, fraudulent transactions could happen more in the night than they do in the morning.

1.2.1 Correlation analysis

Thanks to correlations we would like to test our hypoteses regarding the significance of Time and Amount features and possibly discover new significant features. What the correlation maps clearly shows is that there is clearly no information in these two features as far as links with Class is concerned(at least in a linear fashion). But what we also discover that some relevant features($|\rho| > 0.3$)emerge, namely V14 and V17.

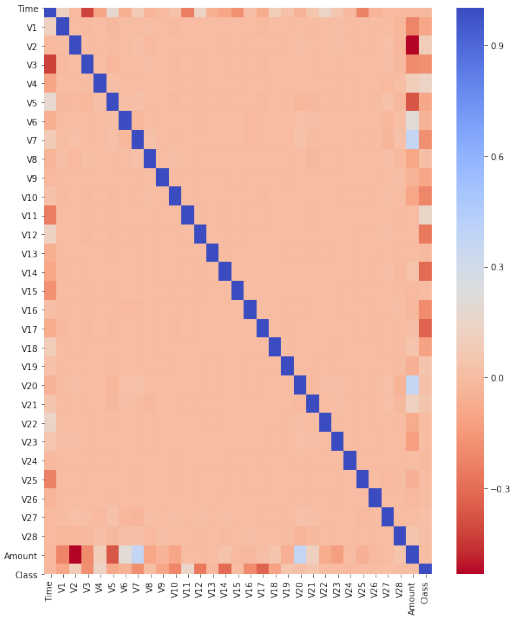


Figure 3: Correlation heatmap with imbalanced data

The next thing we did was trying to have a higher resolution with respect to correlations so we tried to rebalance the dataset with the hope of having a clearer vision of significant correlations. We used a technique called ClusterCentroids, which applies K-Means algorithm to data and then substitutes the actual data with centroids. We preferred to use this technique at this step instead of pure random undersampling because we would like ideally to preserve the underlying structure of the dataset. What started to emerge clearly were new relevant features. With an higher threshold($|\rho| > 0.7$) now we had features V4, V11, V12, V14 who were the most relevant.

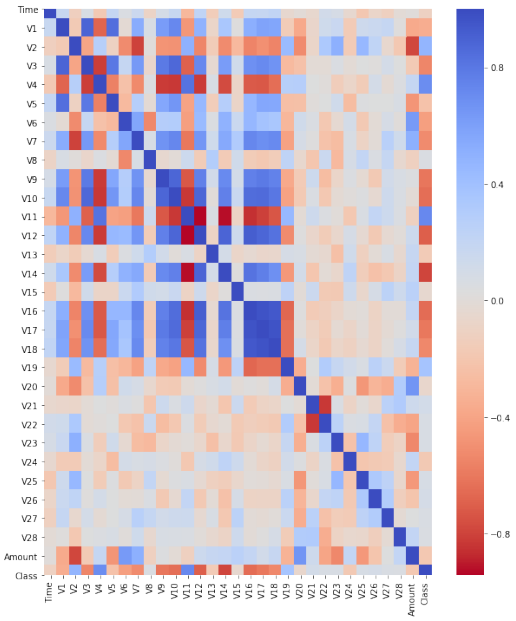


Figure 4: Correlation heatmap with balanced data

1.2.2 Data Visualization

Given the high dimensional nature of our dataset what we tried to do was using another PCA to select the two most relevant(highest variance)linear combinations of features(eigenvectors of covariance matrix)and then scatterplot the projection of the dataset onto these two features. We were conscious about the fact that one initial PCA was applied to data so it could be simply useless performing another one because we already had the eigenvectors of the covariance matrix. But what we wanted to check was the explained variance, we wanted to be sure to select the features with the most explained variance. What we did next was checking the presence of some kind of clustering but there was clearly none.

Our idea was then the following, performing the PCA on a subset of the features chosen by the following criteria:

1. Features with the highest correlation to Class

2. Features with the highest separation between histograms conditional on class 0 and 1(this was computed as the absolute difference in means)

(a) PCA performed on all the features (b) PCA on the most significant features

Figure 5: Result of PCA for features selection

Given the fact that a first PCA only performed on features following criterion 2. already gave good results in terms of separation and given the fact that some features respected both criterion 1. and 2.(which was reassuring we were more confident about them being the most significant) what was needed was simply adding V11. The final PCA on V14, V3, V17, V12, V10, V11, V4, V1 gave us the best performance in terms of clustering separation.

From now on they are to be interpreted as the most discriminating features.

2 Model choice

The choice of the models to use was performed taking into account the major problem of our analysis: dataset imbalance. It was evident we would encounter a tradeoff between *recall* and *precision* on class 1. In the context of money transaction we realized that more importance on recall should have been given because missing a fraud means missing money(often great amounts)whereas receiving a notification on possible fraudulent transactions would be annoying for a client but could also be solved by further human analysis on the transaction itself before notifying the client. We would expect from any algorithm high recall on class 0(if extremely high associated to high precision on class 1)but low recall on class 1 if we were to use the entire dataset given the fact that the algorithm was well trained on 0's but not on 1's. If we were to wisely rebalance the dataset we would expect recall on 1 to become higher and higher while precision on 1 dropping. On the basis of these considerations we set our benchmark: **keeping recall $\geq$ 90% while having a precision as high as possible(ideally >70%)**.

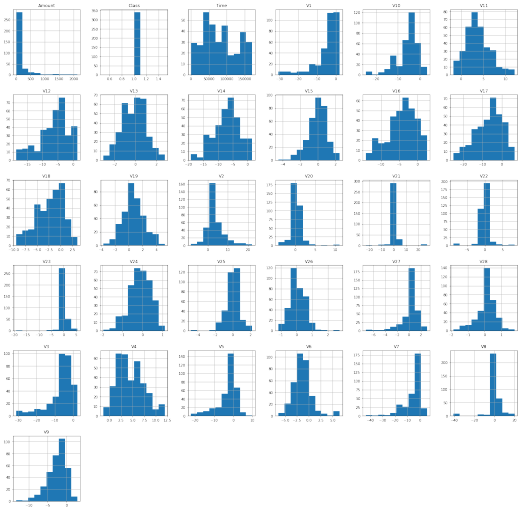


Figure 6: Histograms conditional on class 1 showing how some features have a mean significantly different from 0, which is the mean associated to each feature of class 0

The general approach we chose to adopt was articulated as follows:

1. Running the algorithm on the entire dataset
2. Running the algorithm over an undersampled dataset
3. Running the algorithm over an undersampled dataset including only relevant features
4. Performing cross validation to tune hyperparameters of the final model
5. Run the model on the test dataset

As far as rebalancing was concerned we faced three possibilities:

1. **Random undersampling**: the simplest solution which consists in simple random sampling keeping only a fraction of the class 0 data while preserving class 1 data
2. **Clustering undersampling**: which was the technique adopted in data preprocessing(see subsection 1.2.1)
3. **Oversampling**: inserting artificial copies of class 1 data

After some experiments and considerations we opted for techinque 1., in order to preserve the integrity of data.
The classifications techniques we have chosen are:

1. **Gaussian Naive Bayes**: an extension of Naive Bayes to continuous data. Maybe the simplest idea for a classifier. Relying on independence hypotesis of features we expect it to fail somehow(see figure 4 for correlations between features)

2. **Logistic Regression**: this classifier allows to drop independence hypotesis so is somehow more robust. However this algorithms is really sensitive to dataset imbalance given the shape of the loss it relies on.

3. **SVM**: simple, fast and reliable algorithm, thanks to the kernel trick that allows us to go beyond linear separability assumption. Problems could arise in presence of strong non-linearity.

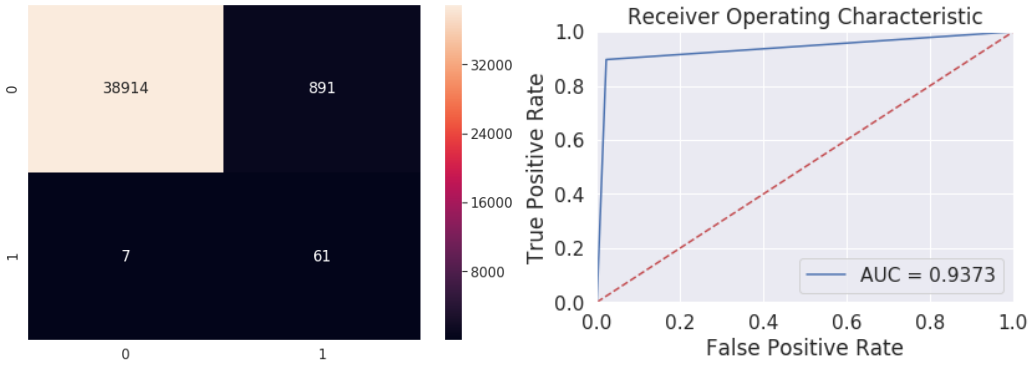
4. **Neural networks**: last resort as far as classification. Extremely reliable for non-linear relationships but slow and at risk of overfitting.

Note: in the subsequent sections *recall* and *precision* will refer to class 1 unless differently specified

2.1 Gaussian Naive Bayes

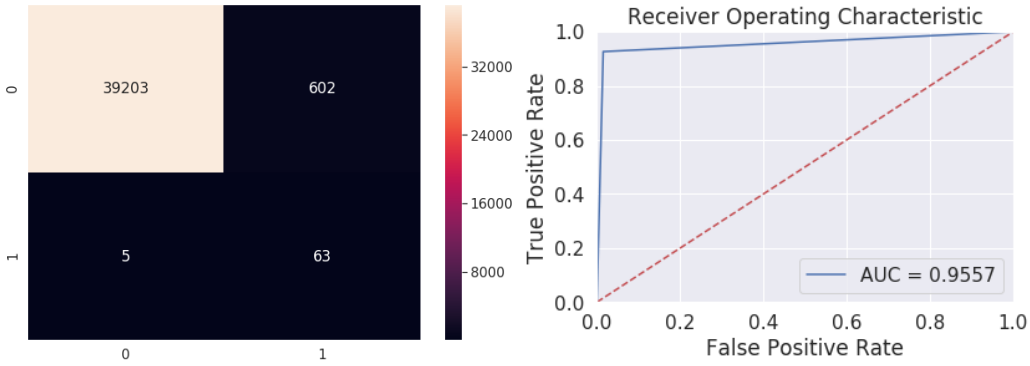
Running the GNB on the overall dataset shows how recall seems to be very high while precision very low. Altough independence hypotesis is strongly violated in this dataset GNB still seems to work very well as far as recall is concerned. This could be due to strong cancellations that occur between positive and negative correlations. Small performance improvements can be obtained with undersampling and undersampling+feature selection.

GNB performance summary			
	Recall	Precision	AUC
No undersampling	0.90	0.06	0.94
Undersampling	0.91	0.06	0.94
Under+Feat	0.93	0.09	0.96



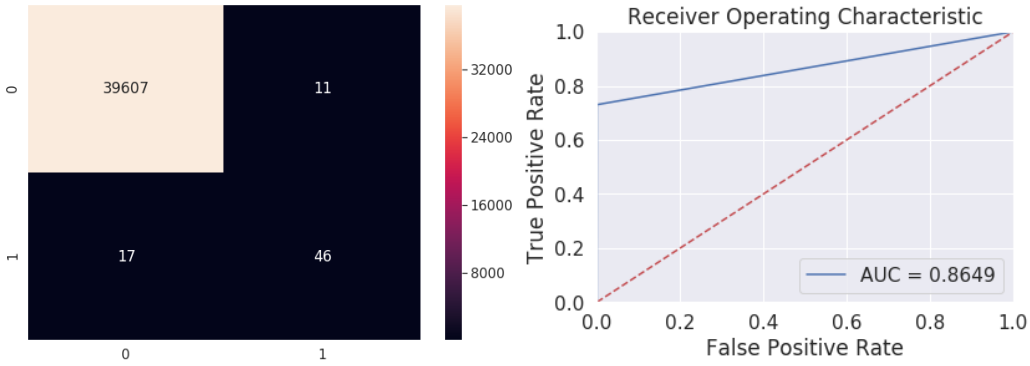
(a) Confusion matrix (b) AUC

Figure 7: GNB:Results with complete dataset/undersampling



(a) Confusion matrix (b) AUC

Figure 8: GNB:Results with undersampling and feature selection



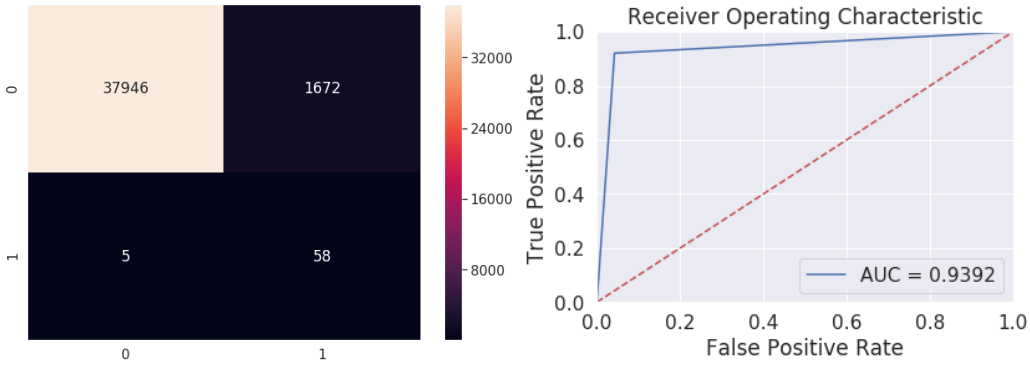
(a) Confusion matrix (b) AUC

Figure 9: LR:Results with complete dataset

2.2 Logistic Regression

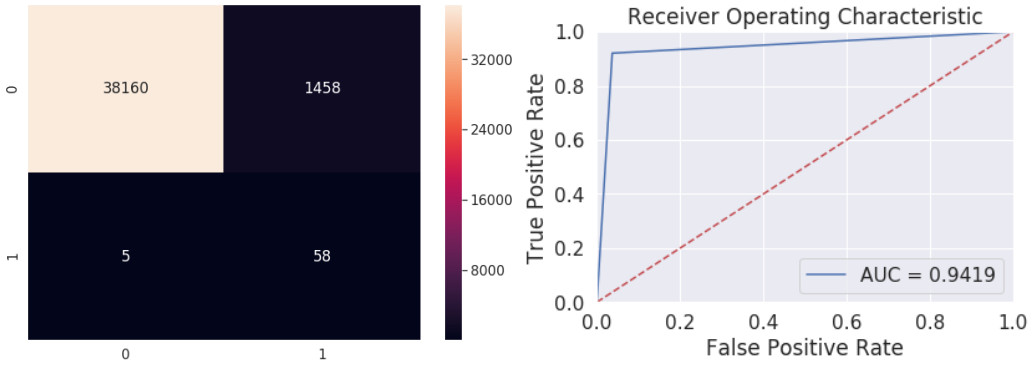
Running logistic regression on the entire dataset gives a balanced performance for recall and precision, probably due to capability of Logistic Regression to manage correlated features differently from GNB. Performing undersampling and a rebalancing on the dataset that uses the values of y to automatically adjust weights inversely proportional to class frequencies(a feature contained in Python library concerning Logistic Regression) a significant improvement on recall is obtained while precision hits almost 0. Dropping irrelevant features only increases speed but does not influence classification performance. The best overall performance was obtained using rebalancing+oversampling on class 1,obtaining a very small improvement on precision(some figures not shown due to small improvement in performance as shown in the tab).

LR performance summary			
	Recall	Precision	AUC
No rebalance	0.73	0.81	0.86
Rebalance	0.92	0.03	0.94
Undersample	0.92	0.06	0.93
Relevant feat	0.92	0.05	0.95
Oversampling	0.92	0.04	0.94
Over+relevant	0.92	0.05	0.95



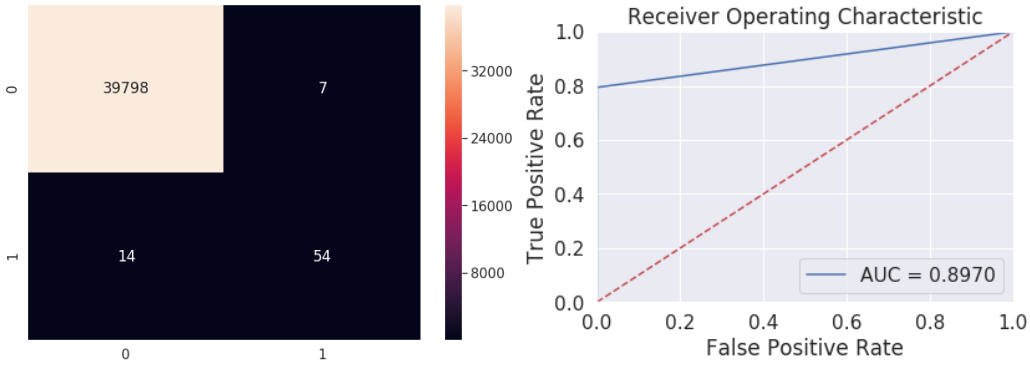
(a) Confusion matrix (b) AUC

Figure 10: LR:Results with reweighting



(a) Confusion matrix (b) AUC

Figure 11: LR:Results with oversampling



(a) Confusion matrix (b) AUC

Figure 12: SVM:Results with complete dataset

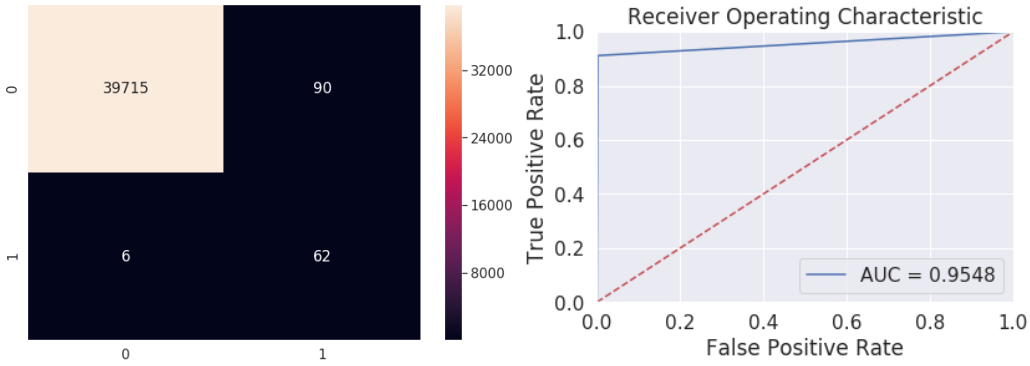
2.3 SVM

For SVM the run on the entire dataset is unsatisfying on recall. Undersampling improves performance but the actual improvement comes from using a undersampling+relevant features.The difference with respect to other algorithms is that recall can be made high while still preserving a satisfying performance on precision.Cross validation does not help in further improving precision, because custom parameters were already the best. The use of a gaussian kernel is crucial because we saw from PCA that the data weren't perfectly linearly separable so introducing a certain amount of non-linearity seems to help.

SVM performance summary			
	Recall	Precision	AUC
No undersampling	0.79	0.86	0.90
Undersampling	0.91	0.41	0.95
Selection+under	0.91	0.74	0.96
CV	0.90	0.76	0.95

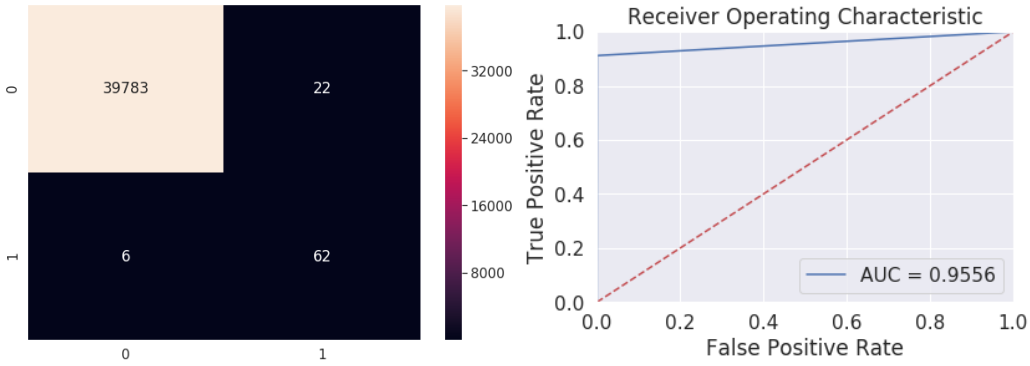
2.4 Neural Network

Running the neural network over the entire dataset shows a balanced performance in terms of recall and precision. Targeting a higher recall we undersample our dataset getting a very high recall but at the cost of a significantly



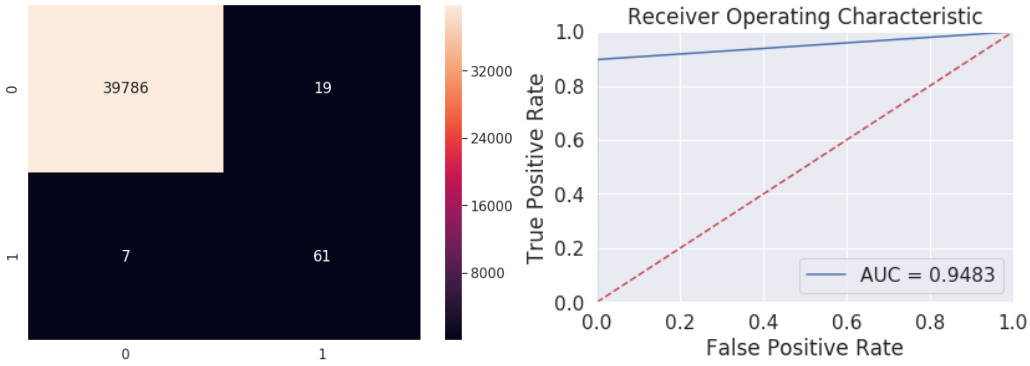
(a) Confusion matrix (b) AUC

Figure 13: SVM:Results with undersampling



(a) Confusion matrix (b) AUC

Figure 14: SVM:Results with undersampling+feature selection



(a) Confusion matrix (b) AUC

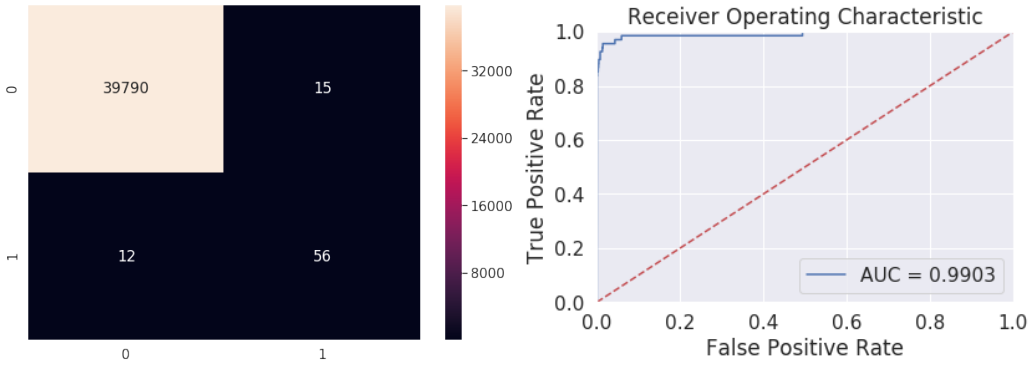
Figure 15: SVM:Results with gaussian kernel+feature selection+regularization

lower precision. We also notice how we needed to add some L2-regularization because the net tended to easily overfit. One more optimization factor we introduced was a reweighting in the loss based on the same criterion of logistic regression. Dropping irrelevant feature led to the same recall but increased precision. We finally set the optimal threshold which was near 0.5.

NN performance summary			
	Recall	Precision	AUC
No undersampling	0.82	0.79	0.99
Undersampling	0.93	0.22	0.99
Selection	0.93	0.34	0.99

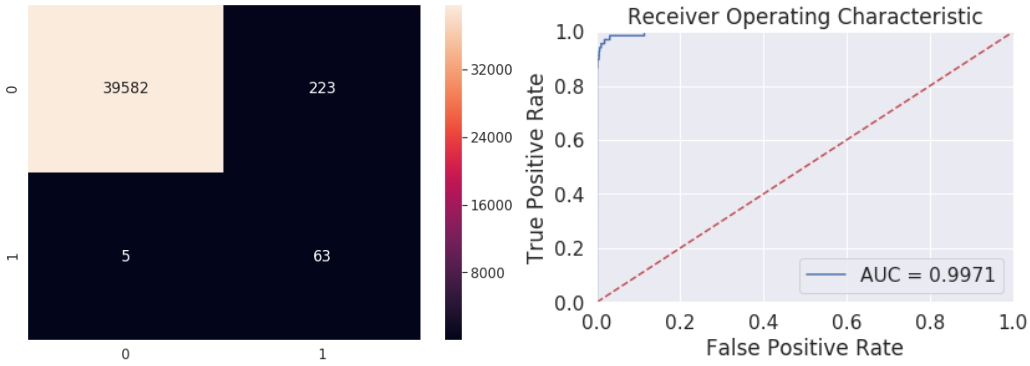
3 Final considerations

Based on our benchmark we see how SVM is probably the best algorithm to use if we want a good balance between speed, recall and precision(the Notebook is slow because of the first part which does not include any optimization and was just an experiment). On the other hand the best performance on recall was obtained with NN, as expected, although GNB and LR gave a similar recall but with a significantly higher speed. Our guidelines regarding oversampling and feature selection have shown to be solid enough especially with the best performing algorithms, NN and SVM, validating our initial



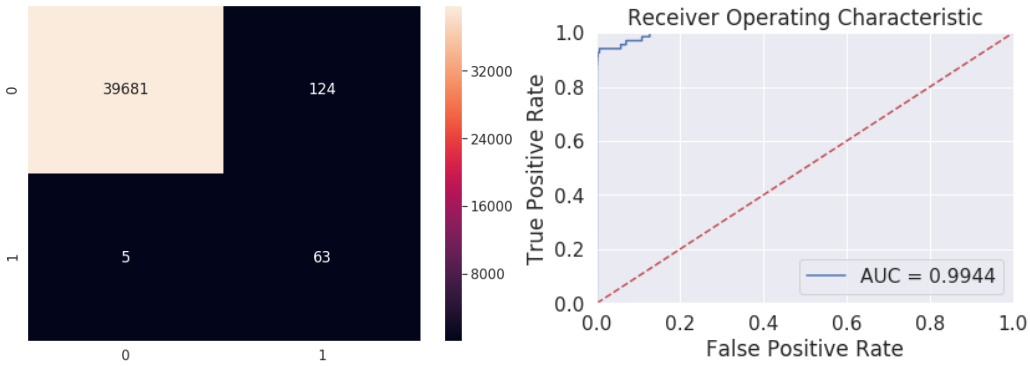
(a) Confusion matrix (b) AUC

Figure 16: NN:Results with complete dataset



(a) Confusion matrix (b) AUC

Figure 17: NN:Results with undersampling



(a) Confusion matrix (b) AUC

Figure 18: NN:Results with feature selection

Figure 19: Validation curve for the best performing NN: we clearly see how it does not overfit

analysis. Further improvement could be obtained with ensemble methods that exploit two algorithms with complementary features, high recall/low precision and low precision/high recall. We started to implement a combination of such NN's but results are still too preliminary so further work could be done in this direction.

Best performance summary			
	Recall	Precision	AUC
GNB	0.93	0.09	0.96
LR	0.92	0.05	0.99
SVM	0.90	0.76	0.96
NN	0.93	0.34	0.99