Fraud Detection Dataset Analysis and Machine Learning Classifiers' Comparison for Imbalanced Data

Dimitris Petratos

Abstract

In this study we will compare the application of machine learning classifiers to imbalance data for fraud detection. We will use techniques of oversampling such as SMOTE and we will focus to the influence of them at False Negative Rate, which we consider as the most important metric of rare events' classification such as the one we study.

Imbalance Data

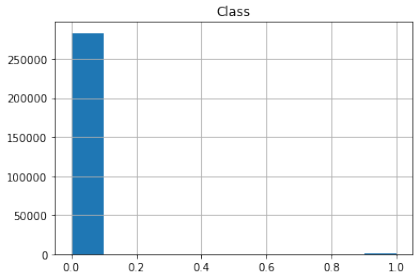
We come across with imbalance data when we have to deal with classification tasks that do not have equal represenation from both of the classes. We consider one class to be minor and the other one major class. Cases that could be interpreted as classification of imbalanced data are the one we study here, or others that contain the context of rarity of one of classes such as the occurence of a rare disease.
There have been studied a lot of approaches to problems of imbalance classification. Here we will use oversampling and specifically SMOTE [1] and a special case of that method, Borderline SMOTE [2]. SMOTE creates synthetic data geometrically, producing new registrations between k neighbors of the minor class. Borderline SMOTE does the same by not taking under consideration of the noisy data.

Methodology and Metrics

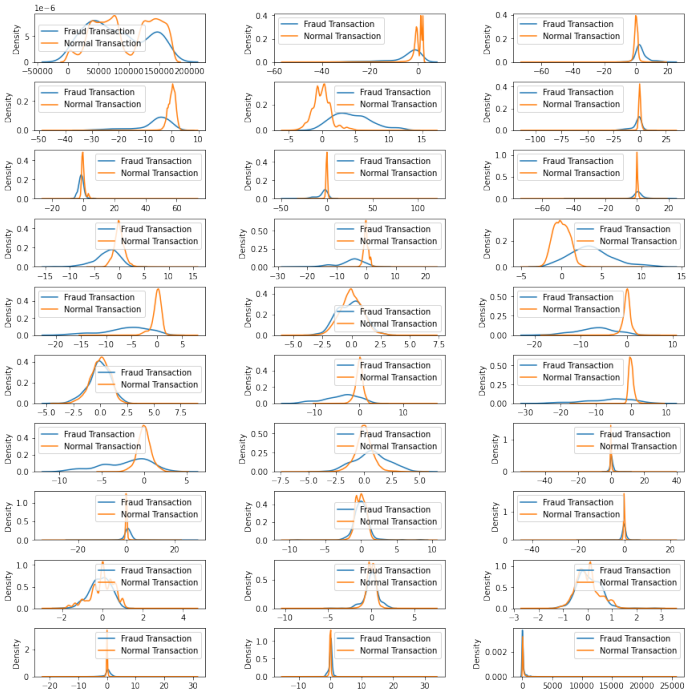
In the present study, we will use 25/75 test/train split. As evaluaton metrics we will measure accuracy, precision, f1-score and recall. What we consider as the most important measure for datasetes such as the one we study here, is False Negative Rate (FNR). FNR refers to the cases that an actual rare event is classified as a normal event. FNR if formally defined as $\frac{fn}{fn+tp}$ and our goal is to minimize it.
As one can understand, in cases of real time classification, we need a classifier that without any supervision minimize that error. In a real case we would only need human inspection to cases that have been classified as fraud transactions. As we will notice later, when we try to minimize FNR, we will increase False Positive Rate (FPS), formally defined as FPS=$\frac{fp}{fp+tn}$. This is the trade-off that comes up to cases like this one.
We will compare classifiers' performance with and without oversampled train set, holding test set stable.

Data Cleaning and Statistics

For our study we will use Credit Card Fraud Detection dataset from Kaggle. It consists of 284,807 transactions, 492 of which are labeled as fraud. We have 28 features produced from PCA for anonimity reasons and one feature that corresponds to the seconds ellapsed between each transaction and the first transaction in the dataset. We also have the feature Amount that corresponds to the amount of the transaction. In the process of data cleaning we found out that the dataset contains 1,081 dublicates that were removed. No null values were found. After dublicates removal, 473 fraud transactions were left.



So finally we are left with 99.83% of normal transactions and 0.17% of fraud transactions. That indicates the demand of the oversampling that we used.
After train/test split we are left with the same ratio of normal/fraud transactions to train and test set.
We give below the distributions of all the features per class:

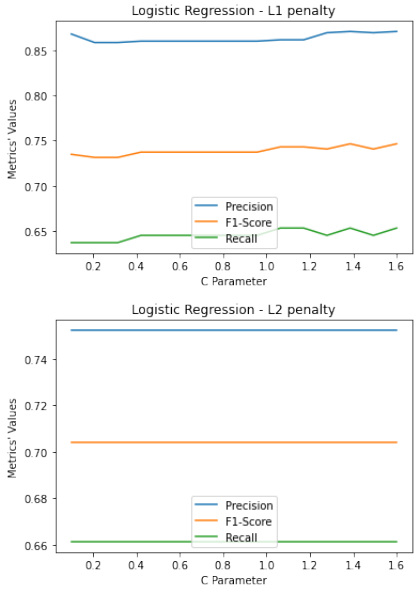


We do not observe many differences between the two classes.

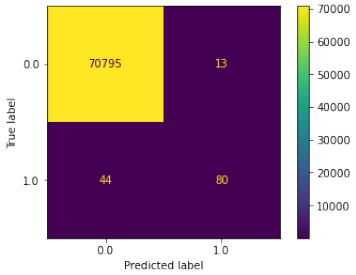
Measurements and Results

Logistic Regression

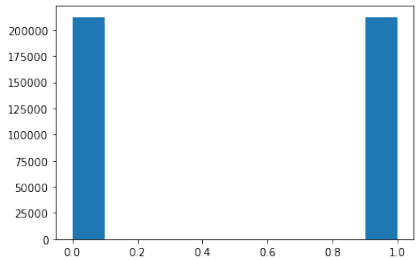
At first we searched for the hyperparameters that we will use for our model. After the following measurements using the test set:



we choose to use l1 penalty with C=1.
For the above model we take the following results:
Accuracy: 0.9991
Precision: 0.8602
F1-Score: 0.7373
Recall: 0.6451
We observe a huge gap between precision and recall.

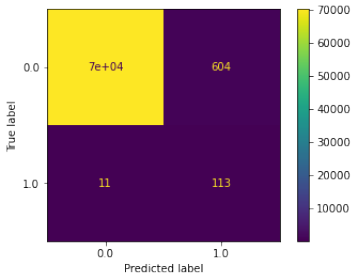


FNR: 0.3548
FPR: 0.0001
We see that we have almost 35% FNR and that is considered as a lot.
After creating synthetic data with SMOTE, using 5 neighbors, we balance the classes at train set:

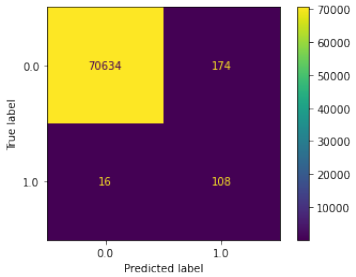


and continue implementing our classifier, getting the following results for test set:
Accuracy: 0.9913
Precision: 0.1576
F1-Score: 0.2687
Recall: 0.9112

We take a huge difference for precision and recall, comparing the values we got without oversampling.



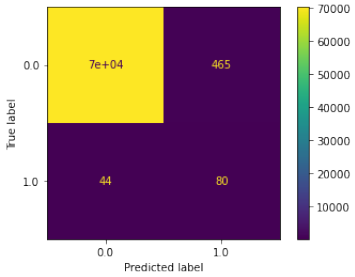
FNR: 0.0887
FPR: 0.0085
We take a big drop at the value of FPR and proportionaly a big increase at the value of FPR (almost 100 times rise).
Finally we use Borderline SMOTE getting the following results:
Accuracy: 0.9973
Precision: 0.3829
F1-Score: 0.5320
Recall: 0.8709



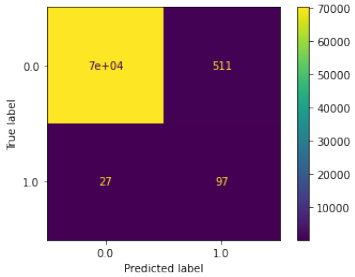
FNR: 0.1290
FPR: 0.0024
For the case of logistic regression we saw that using SMOTE, we take better results regarding thw values of FNR, loosing from the perspective of precision and FPS. At every case, we have good values for the accuracy.
Borderline SMOTE doesn't seem to affect our results better than the classic implementation of SMOTE, probably because we don't have noise between our classes.

Gaussian Naive Bayes

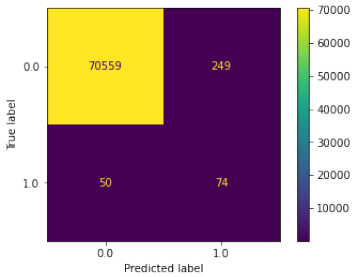
Using Gaussian Naive Bayes implementation from scikit-learn we don't have to worry about hyperparameter tuning, because there are no extraordinary hyperparameters.
Without oversampling we gaet the following results:
Accuracy: 0.9928
Precision: 0.1467
F1-Score: 0.2391
Recall: 0.6451



FNR: 0.3548
FPR: 0.0065
We don't get good metrics, as expected.
Below we present our results after SMOTE is used to generate synthetic data: Accuracy: 0.9924
Precision: 0.1595
F1-Score: 0.2650
Recall: 0.7822



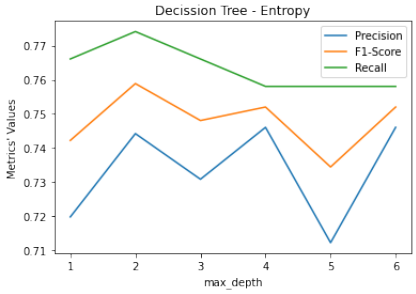
FNR: 0.2177
FPR: 0.0072
We don't get better results comparing to the case of logistic regression. Below are the results we get from Borderline SMOTE: Accuracy: 0.9957
Precision: 0.2291
F1-Score: 0.3310
Recall: 0.5967

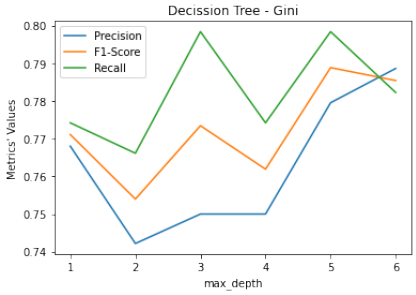


FNR: 0.4032
FPR: 0.0035
We see that with Gaussian Naive Bayes things go better with SMOTE, but are getting worse when Borderline SMOTE is applied.

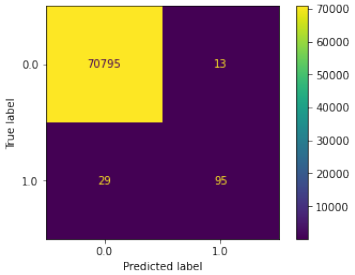
Random Forest

Now we will conduct hyperparameter tuning for Random Forest Classifier searching for max_depth between 1 and 6 and criterion between Gini and Entropy. We get the following:

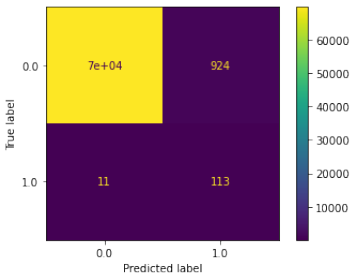




We choose max_depth=5 and criterion=gini, getting the following results:
Accuracy: 0.9994
Precision: 0.8796
F1-Score: 0.8189
Recall: 0.7661

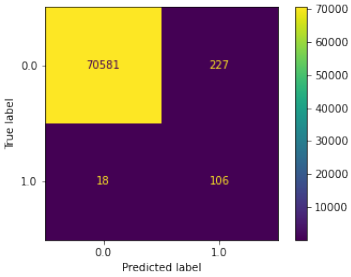


FNR: 0.2338
FPR: 0.0001
At first, it seems that we get good results. Below are the values after SMOTE oversampling:
Accuracy: 0.9868
Precision: 0.1089
F1-Score: 0.1946
Recall: 0.9112



FNR: 0.0887
FPR: 0.0130
FNR seems to 've been dropped significantly.
Finally the results after Borderline SMOTE are given below:
Accuracy: 0.9965

Precision: 0.3183
F1-Score: 0.4638
Recall: 0.8548



FNR: 0.1451
FPR: 0.0032
We see that with random forest classifier, as measured with all the other classifiers, we get better results with the classic approach of SMOTE, getting a little worse with Borderline SMOTE.

Results and Future Wordk

We studied 3 classifiers and the influence that synthetic data has to teir performance, since we dealt with imbalance data. We saw that decision tree classifier outperforms logistic regression and Gaussian naive Bayes classifiers using SMOTE technique for syntehtic data production. For that reason we considered False Negative Rate as the most important metric measure, since in real cases where there would not be any human interaction with the classifier, we would like to have as few False Negative missclassified documents as we could.
As future work we could try other oversampling methods such as GANs or ADASYN, or even undersampling techniques.

References

1. SMOTE: Synthetic Minority Over-sampling TechniqueNitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer, 2002
2. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning, Hui Han, Wen-Yuan Wang, Bing-Huan Mao, International Conference on Intelligent Computing ICIC 2005: Advances in Intelligent Computing pp 878-887