Transaction fraud detection

E. Gridneva

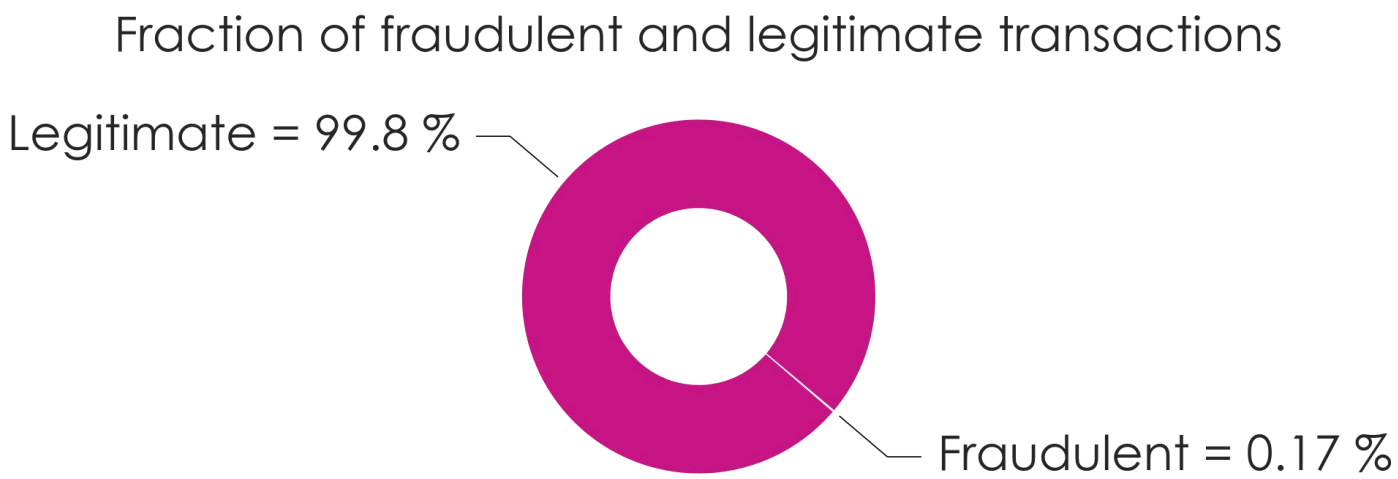
Introduction

- Credit card fraud rate grows together with the number of card transactions, which becomes the major mode of payment for both online and offline transaction.
- There can be inner or external card fraud. Inner fraud carried out via false identity to commit fraud while the external card fraud involves the use of stolen credit card to get cash or to pay for smth.
- Most of the credit card fraud is external, when a thief tryes to pay with the stolen card.
- Financial institutions nowadays focuse on computational methodologies to handle credit card fraud problem.

Introduction

- A person may typically pump gas one time a week, go grocery shopping every two weeks, and so on. The algorithm learns that this is a normal transaction sequence. Then the network predicts a probability for a test transaction to be fraudulent.
- A machine learning algorithm for fraud detection is trained by being fed the normal as well as fraud transaction data of lots and lots of cardholders.
- The model should be able to properly classify transactions as either legitimate or fraudulent, based on transaction data which includes for instance amount, merchant, location, time and others detailes of transaction.
- In the dataset that contain transactions and their details, the majority of transactions is not fraudulent. That means we are dealing with imbalanced data.

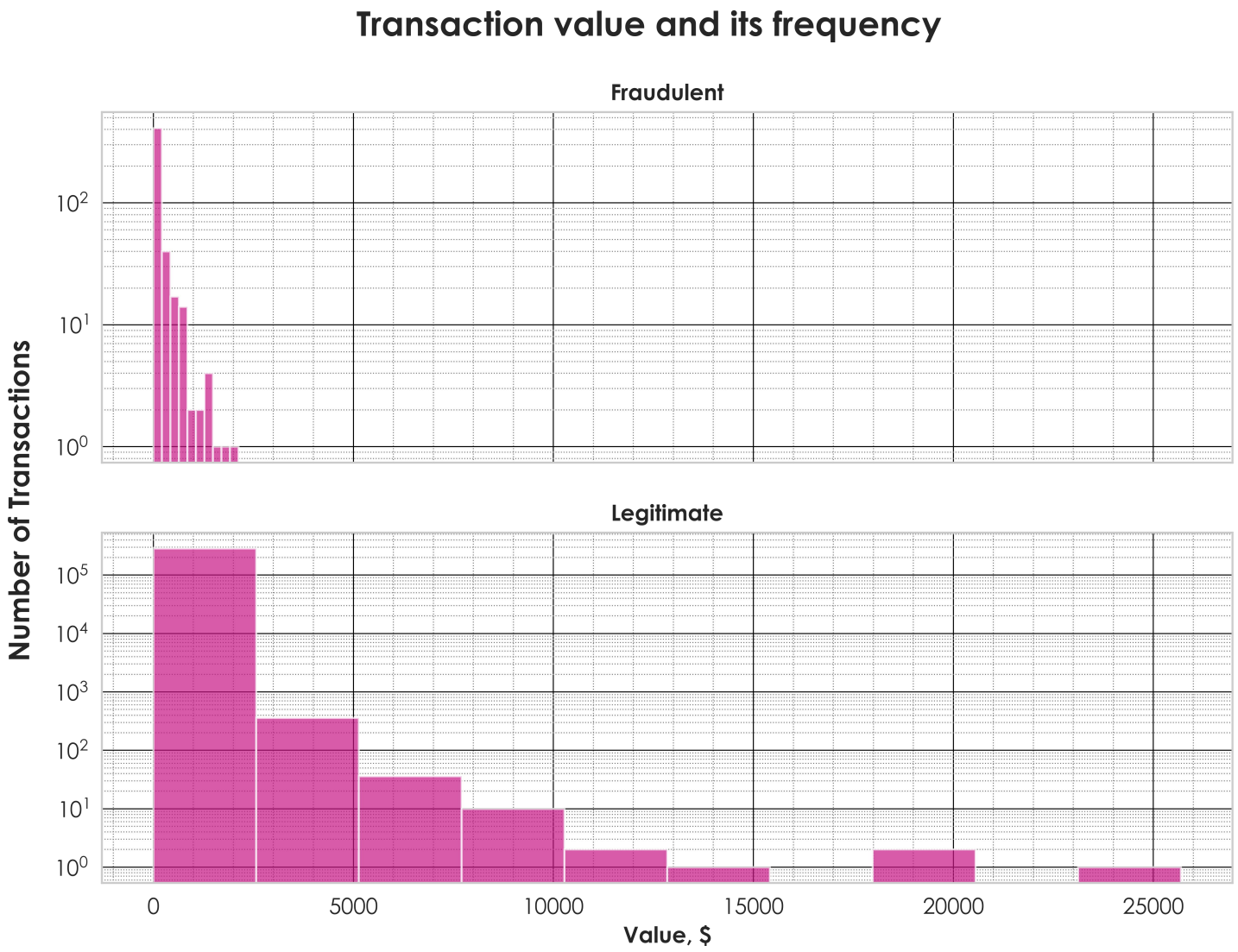
Data overview



The dataset contains transactions made by European cardholders back in September 2013. These are 2 days transactions, and there are 492 fraudulent out of 284,807 transactions. That means the dataset is highly imbalanced.

The frauds are just 0.172% of all the transactions. Saying that, if we predict these 492 fraudulent as “not fraud”, we’d get a 99.83% classification accuracy for our model. Quite high! But our model will be wrong as it cannnot do what we need - identify frauds.

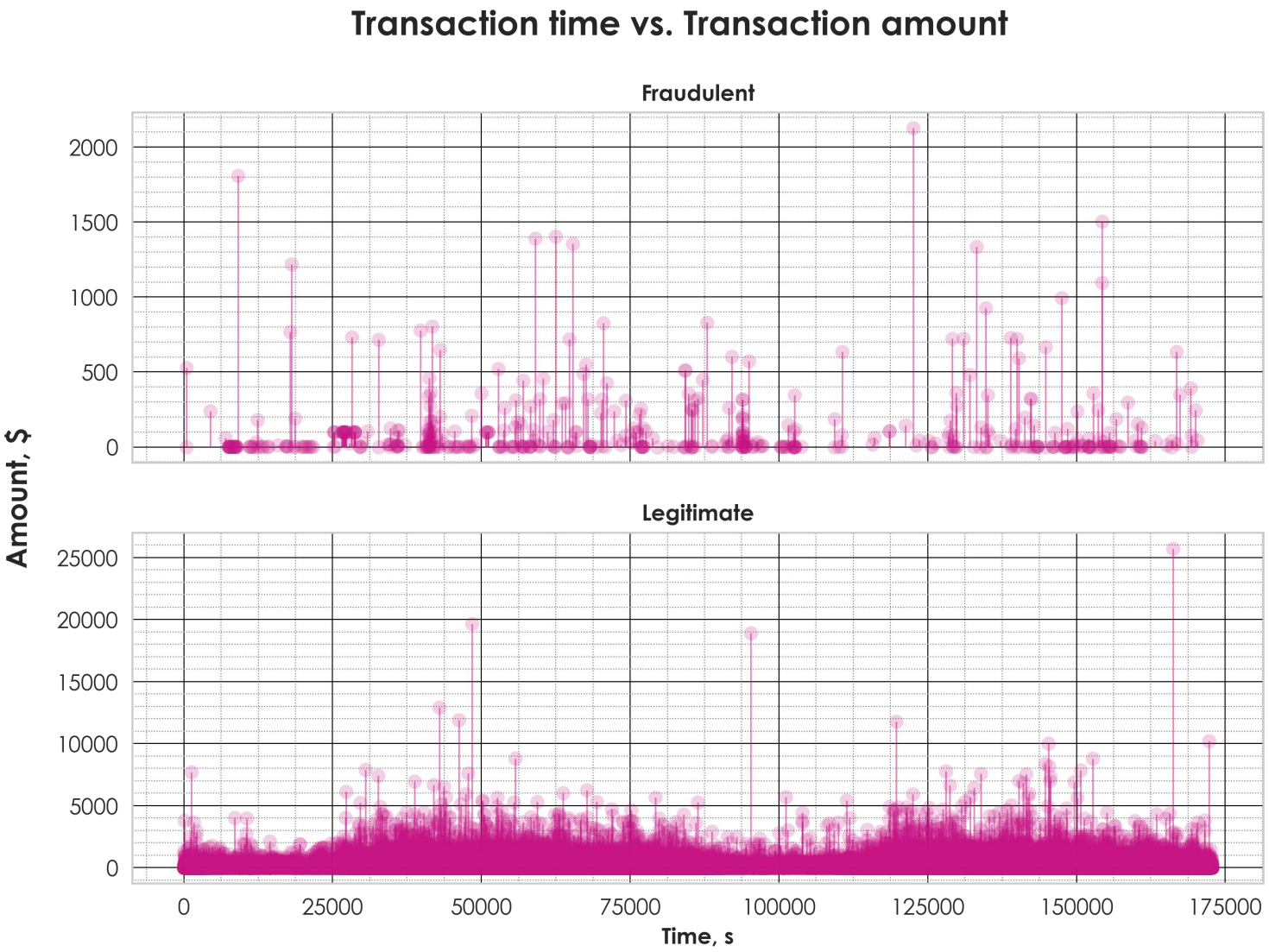
Data overview



Exploring the data! Let's plot the transaction value vs. the number of transactions for legitimate and fraudulent cases.

There are 28 features (in encoded form due to confidentiality reasons, named V1-V28), plus "Amount" and "Time". The feature "Amount" differs from other features values by several orders of magnitude. Therefore we'll rescale it so it has mean 0 and standard deviation 1.

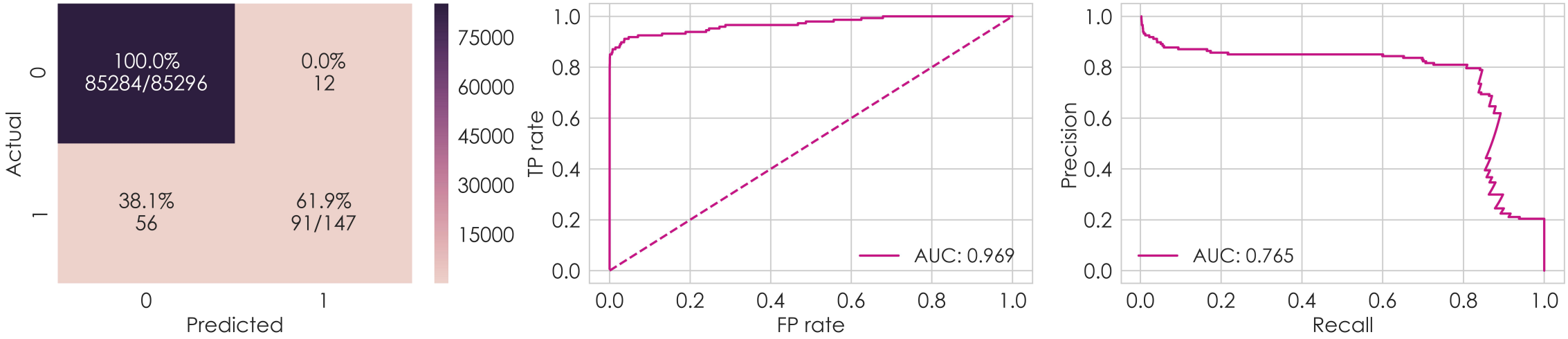
Data overview



The type of transaction doesn't realy show any dependency from the time of transaction. Therefore the time feature can be excluded from our prediction model.

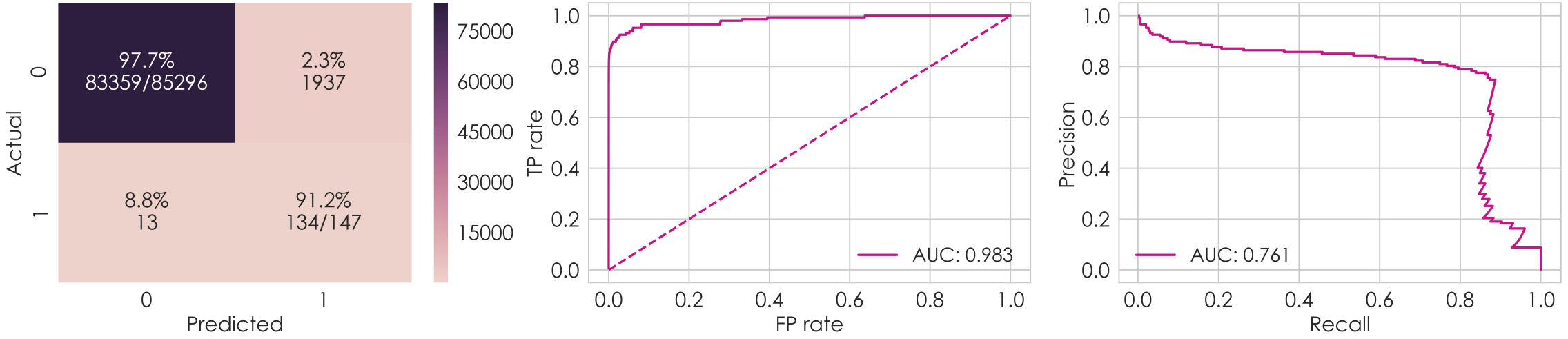
Baseline: logistic regression

We first train logistic regression classifier with the default parameters and later use the result as a baseline score to compare other models with. As it is seen, the model already learns very well about the legit transactions, but is unable to recognize the fraudelent ones.



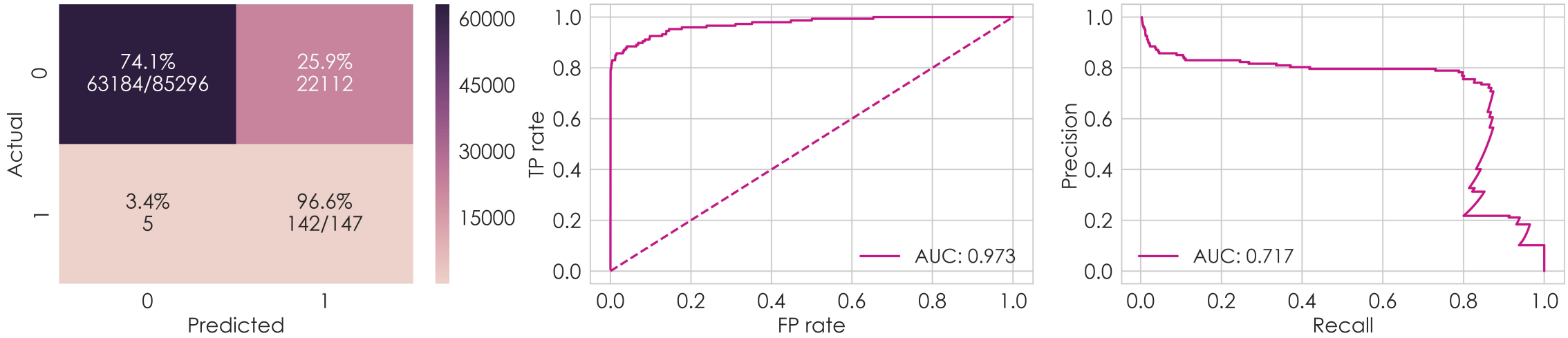
Overcome the data imbalance 1: weighted loss function

Applying weighted loss function can help to overcome the high data imbalance. In this case, the gradient updates are adjusted inversely proportional to the class frequency, i.e. the updates of the larger class are supressed and the updates of smaller class are prioritized. As the result, the recall score for prediction of fraudulent ones increased from 62% to 91% and the precision decreased.



Overcome the data imbalance 2: undersampling

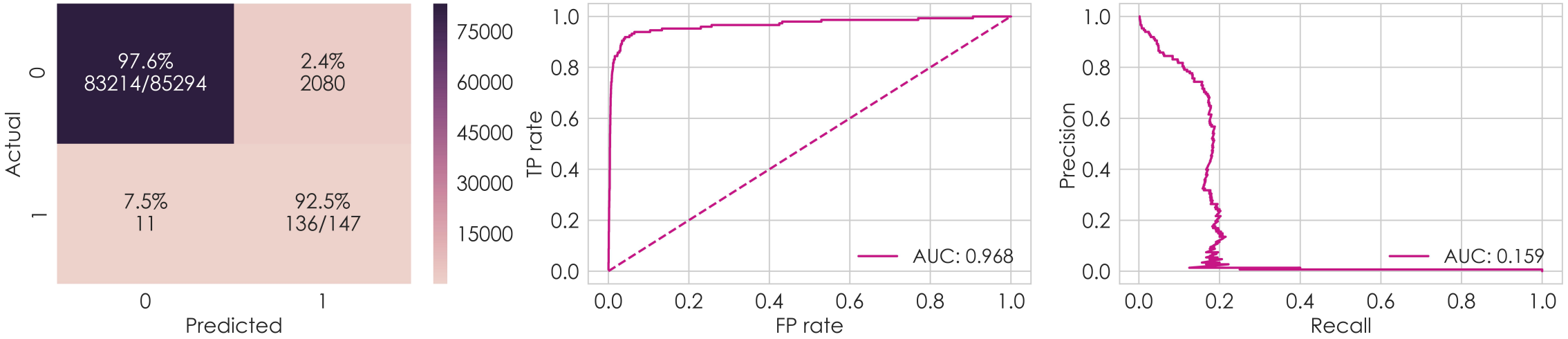
Another technique to overcome class imbalance is to undersample the larger class or oversample the smaller class. Undersampling basically means that the most of the legitimate transactions will be removed from the data so that there is approximately the same amount of both classes in the training and testing data. The disadvantage of undersampling is that a great amount of information is being removed from the majority class.



Due to loss of information induced by undersampling, the model is not realy able to make good predictions on real world data. The true negative rate (prediction of legit transactions) drops down to 74%.

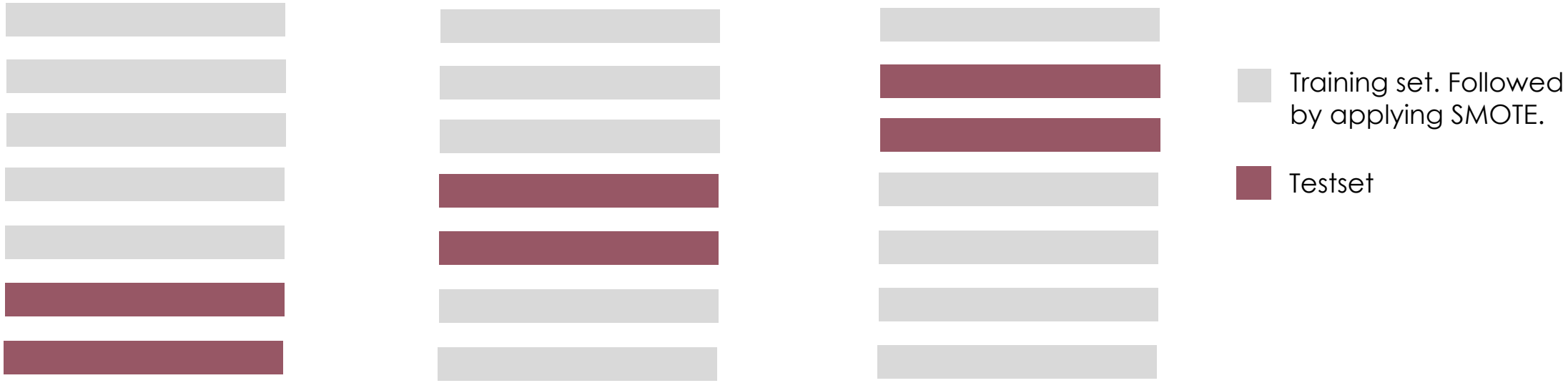
Overcome the data imbalance 3: SMOTE

Another technique to overcome the data imbalance is SMOTE—Synthetic Minority Over-sampling Technique. SMOTE creates synthetic observations of the minority class (in this case, fraudulent transactions). The technique first finds the k-nearest-neighbors for minority class observations, then randomly chooses one of the k-nearest-neighbors and creates a similar, but randomly tweaked, new observations. To implement SMOTE, we will use imblearn library. It is a toolbox for dealing with imbalanced data problems.



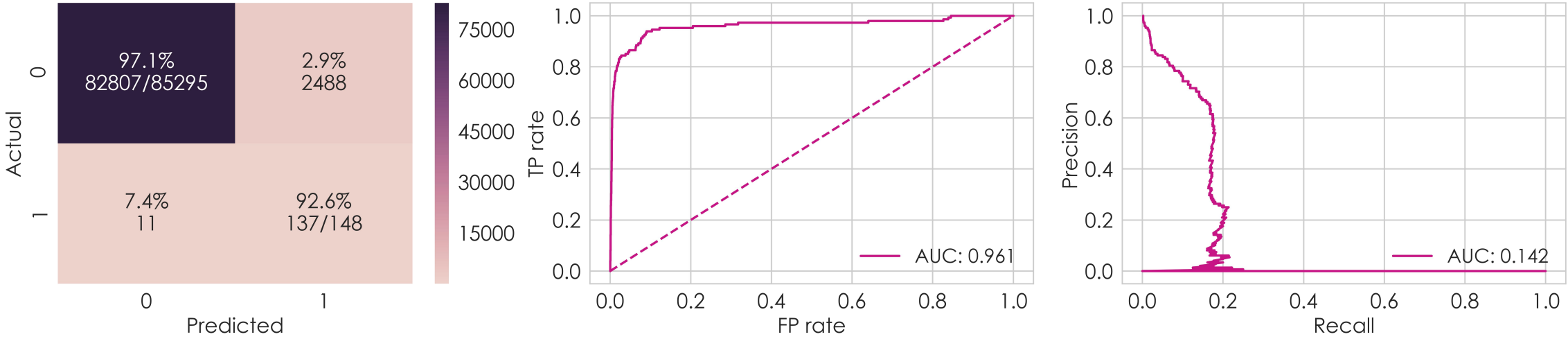
Overcome the data imbalance 3: SMOTE

In order to further verify our model, we can test it on several various testsets chosen randomly from the dataset. For that purpose, we will use stratified shuffle-split at the data separation step, then apply oversampling on the training set only, and then verify the model on the test set. The number of splits in this example is equal to 3.



Overcome the data imbalance 4: feature engineering

Applying clustering, we can organize our features by groups, and the group will be then additional feature in our training dataset.



Clustering improved true positive rate by +0.1%, but decreased true negative a bit, as well as increased false positive by +0.5%.

Conclusions and further work

Applying additional measures, we could highly increase the **True Positive** predictions. With just one classifier such as logistic regression, on the highly imbalanced dataset, our model could predict **91** out of **147** fraudelent transactions. With oversampling techniques and feature engineering, we could detect **137** out of **147** fraudelent transactions.

The area where we further should work on is the number of **False Positive** predictions. In this case, a transaction is predicted to be fraudelent when it actualy isn't. So, there would be around 2000 transaction where the model would settle false alarm, making worse the customer satisfaction rate, as the usual payment that a bank customer want to carry out, would be identified as suspicious and blocked.