Binary classification problem moneyveo

Yurii Baidala

August 4, 2019

Contents

1 Introduction and Statement of the problem **1**

2 Benchmark Model with Logistic Regression **1**

3 Exploratory Data Analysis **3**

4 Outliers Detection and Treatment **4**

5 Feature Engineering **4**

6 Modeling **4**

7 Conclusion **5**

1 Introduction and Statement of the problem

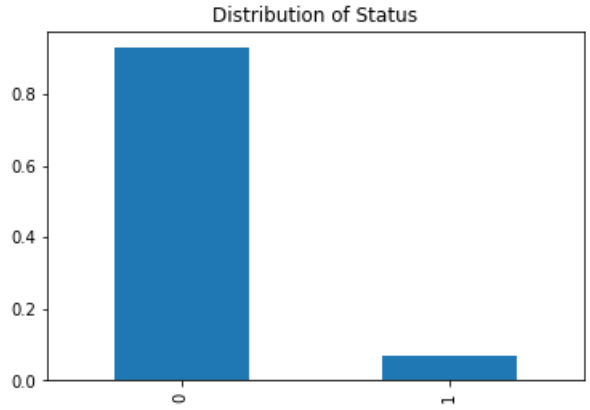
In this test task two data sets are given in order to solve a binary classification problem. A binary classification problem is one of the most common tasks in machine learning and data science. Essentially, from training data we need to make a prediction for testing data, in our scenario *Status* variable needs to be predicted. *Status* is a binary target variable, which means it only takes two values 0 and 1. In the description of the task, a clear business objective was not stated and no business metrics were given, therefore, in this analysis I will be using two standard classification metrics: area under the ROC-curve and average precision score. As a tool for data analysis I will be exploiting Python and its various library.

2 Benchmark Model with Logistic Regression

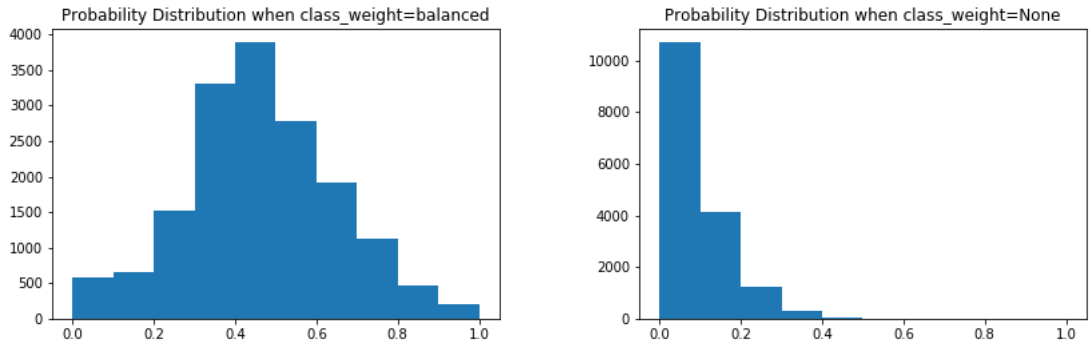
After first glance on training data we can conclude that this is imbalanced binary classification, which means that positive class (that takes value 1) is a minority class and occurs around 7% of times. Visualization of *Status* distribution is shown below.

Initial analysis also suggests that the training dataframe consists of 65974 rows (events) and 223 columns(some anonymized characteristics). Also, there are four types of variables i.e.

```
0    0.928927
1    0.071073
Name: Status, dtype: float64
```



continuous variables (109 columns), integers (108 columns including Id), boolean variables (3 columns, takes True and False values) and object type (3 columns, which need special treatment). Fist step of the analysis would be to build a benchmark model. This so-called benchmark model would allow us to compare results of more sophisticated approaches to it. Let's discard object type columns and proceed only with numeric columns. Hence, training data is split on training (75%) and validation (25%) subsets in order to first train the model and then evaluate performance of the model. Logistic Regression from sklearn used as a modeling tool. Since it is imbalanced problem, we use *class weight* equal to *balanced* parameter, which puts more weight on minority class. $AUC\ ROC$ score of this model is equal to .762, which is pretty decent, however AP is only .186. If we changed *class weight* to *None*, than AUC dropped to .598 and AP dropped to only .11.



On the left graph is displayed a histogram of estimated probabilities when *class weight* equal to *balanced* on the validation subset. It looks familiar and reminds us a bit skewed standard normal distribution. It corresponds to higher $AUCROC$ of .762 but does not reflect the observation that the mean of the distribution has to be close to empirical frequency of the positive class. On contrary, the right graph fits to this feature, but the model performs quite poorly by AUC and AP scores. Hence, we need to explore and analyse data more deeply to find the model that performs better and results into realistic probability distribution.

3 Exploratory Data Analysis

First of all, as was mentioned before we divided our train dataframe into training and validation subsets in order to avoid the leakage of information while exploring the data, which might lead to overfitting during the analysis. Secondly, we split training subset columns into 4 types: continuous, integer, boolean and object.

Let's start exploring data with correlation between columns of training data set. Visualization heatmap could be found in *'EDA.ipynb'*. The conclusion from there could be made that some columns are correlated with columns that go right after them (e.g. 'par108' and 'par109'). Also, this heatmap suggests that *Status* does not strongly correlate with any other columns. This could be confirmed directly by using *corr()* function and sorting values.

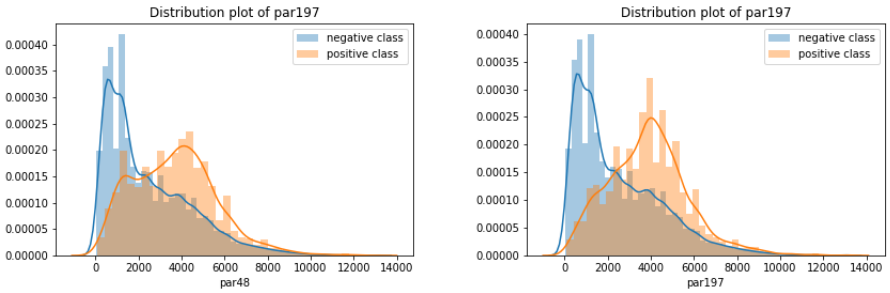
	Status
par85	0.090388
par101	0.095675
par74	0.101227
par75	0.102342
par49	0.133516
par48	0.156626
par108	0.158203
par109	0.165338
par197	0.176225
Status	1.000000

	Status
par174	-0.134240
par175	-0.131400
par194	-0.116936
par86	-0.114048
par132	-0.107839
par89	-0.103674
par166	-0.100275
par200	-0.100080
par167	-0.097777
par106	-0.087302

Two tables above show that only few column have correlation in absolute term higher than 0.1.

The next step in the analysis is to exploit various possibilities of Python libraries to visualize the data sets. For this purpose I used such tools as *scatter plots, histograms, boxplots and distribution plots* of both classes. The results of eyeballing this graph are as follows:

- quite a few a variables have outliers (value -10000 occurs in many columns).
- Scales of continuous columns are very different.
- Variables like 'par48' and 'par197' might have quite high predictive power.



Those two figures show that two classes are quite separable by a hyperplane and that earlier exploration by using correlation function is confirmed here.

4 Outliers Detection and Treatment

In previous section it was found out that there are quite a few outliers in every column, however we need to detect outliers for the whole dataset. For this purpose let's leverage different outlier detection techniques and figure out whether this might improve performance of our model. We will use sklearn and PyOD libraries, more specifically such algorithms as Isolation Forest, Elliptic Envelope and KNN for outlier detection.

First we started with KNN. For an observation its distance to k-th nearest neighbor might be viewed as the outlying score. Utilizing PyOD KNN for outlier detection with out of box parameters (contamination rate 10% and number of neighbors =5) allows to improve performance of both metrics for Logistic Regression, AUC rose from .762 to .78 and AP rose from .186 to .208. Elliptic Envelope algorithm assumes that classes have Gaussian distribution and, therefore, learns an ellipse and ignores everything outside the ellipse. By applying this algorithm we obtain .789 AUC score and .215 AP score. Lastly, Isolation Forest does not improve none of the scores AUC 0.76 and AP 0.21. To sum up, when it comes to outliers detection and treatment, some approaches might be helpful for this data set, but they do not improve performance of the classifier drastically.

5 Feature Engineering

In this part of the analysis we will try to create new features that might improve performance of our model. As might be concluded from EDA, there exist some values in variables that appears quite often and that might have good predictive power. To utilize this observation we will use such feature engineering technique as *frequency encoding*. Frequency encoding technique counts number of appearance of specific value in the data set. This approach is quite popular among kaggle competition participants, and in particular it was a part of a winning solution to Santander Customer Transaction Prediction. Let's see how it performs on this data set.

6 Modeling

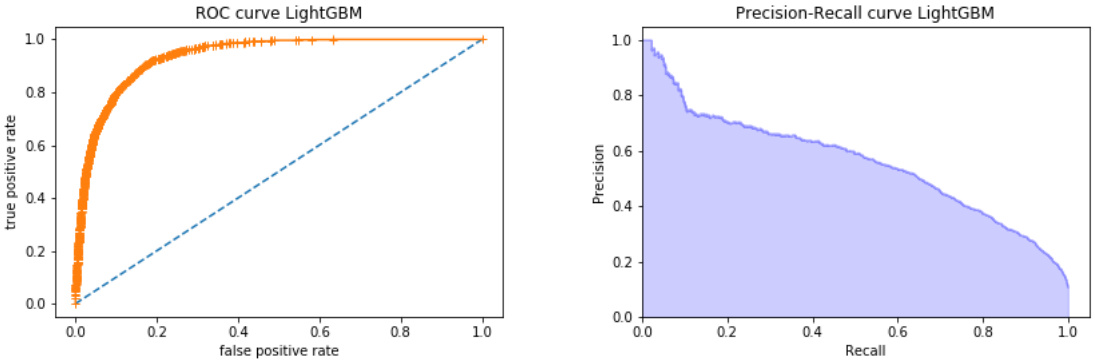
In the Modeling part of this analysis we will try to incorporate several machine learning algorithms and will find out which one performs best on this binary classification problem. We will explore following algorithms: Support Vector Classifier, Naive Bayes Classifier, Random Forest Classifier, Linear Discriminant Analysis and LightGBM. Additionally, we tried to apply a outlier handling technique but it doesn't seem to improve the model.

The table below shows the summary of the results of modeling exercise. Two things can be concluded from this table. First, features engineering technique such as frequency encoding is very effective in this setting, since all algorithms improved their performance

	no Feature Engineering		incl. Feature Engineering	
metrics	AUC ROC	AP	AUC ROC	AP
SVM	.671	.12	.779	.256
Naive Bayes	.72	.151	.742	.16
Random Forest	.84	.292	.924	.512
LDA	.843	.312	.849	.339
LightGBM	.863	.362	.937*	.562**

after adding new features. Secondly, tree-based models outperformed other types of models.

Since LightGBM was the best model according to both AUC and AP we will exploit it for making predictions on a test set. Below are show ROC and Precision-Recall curve. They suggest that this model is not ideal and there is always room for improvement.



7 Conclusion

On this classification problem LightGBM model with frequency encoding features performs best comparing to other models. The model is not ideal and might be improved in different ways but 0.93 AUC ROC score on validation data is pretty decent.