Machine Learning Engineer Nanodegree

Vitor Bona de Faria

April 2019

1 Definition

1.1 Project Overview

Banks pose as important actors in any person's life. Through them, with the correct guidance, someone may earn enough through investments to fulfill monetary goals, which could lead to some personal life goals too.

The amount of data available to banks is incredible, from customers income to consume habits and their posture when it comes to investments. Common challenges banks face may turn into binary classification problems: Is a customer satisfied? Will a customer invest in such stock? Will a customer be able to pay a loan? Successful applications of machine learning to this type of tasks are found in the literature [1].

The data used is available in Kaggle[2] and consists of anonymized data containing only numeric feature variables, the target binary column and a string column corresponding to the customer's ID. The data is composed of 2 files: train.csv; test.csv. Both files contain 200k observations and 201 features, named as: var_0, var_1, ..., var_199, target.

1.2 Problem Statement

This project will build a solution for the "Santander Customer Transaction Prediction" problem. In this problem what is being predicted is the proba-

[1] Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach topredict the success of bank telemarketing.Decision Support Systems,62:22–31, 2014.

[2] https://www.kaggle.com/c/santander-customer-transaction-prediction

bility of a customer making a specific transaction in the future, regardless of the amount of money transacted. The tasks involved are the following:

1. Downloading training and testing datasets from Kaggle
2. Explore data and perform needed preprocessing
3. Compare different models to determine the best approach
4. Train final model and compare to the chosen benchmark model

1.3 Metrics

The chosen evaluation metric was the area under the Receiver Operating Characteristic (ROC) curve. The ROC curve is a probability curve that plots True Positive Rate (TPR) against False Positive Rate (FPR), being the first in the y-axis and the latter in the x-axis. TPR and FPR are measured as follows:

$$True\ Positive\ Rate\ (TPR) = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

$$False\ Positive\ Rate\ (FPR) = \frac{False\ Positives}{True\ Negatives + False\ Positives}$$

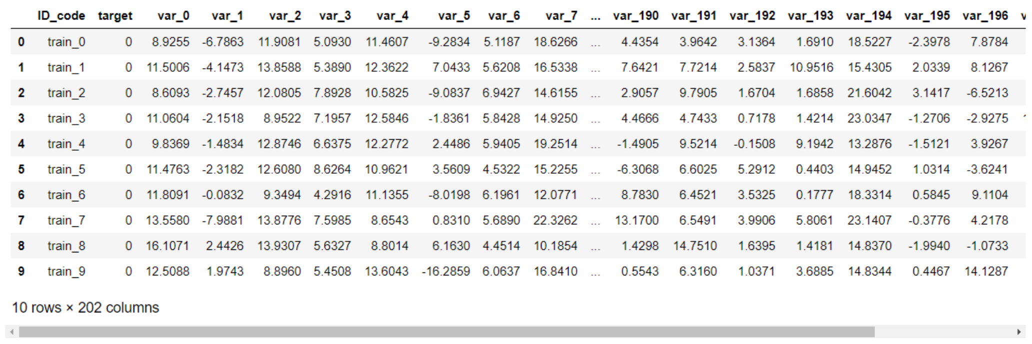
The area under ROC curve is well suited for representing the degree of separability in classification problems[3]. A value of 1 implicates in a perfect model, a value of 0 in a reciprocating model and a value of 0.5 a random model — providing no class separation.

2 Analysis

2.1 Data Exploration

The chosen dataset contains 200.000 observations with 200 features, a target binary column and an ID column. All features are numeric and anonymized, as shown in figure 1, not leaving much room for feature discussion.

[3]https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5



	ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	...	var_190	var_191	var_192	var_193	var_194	var_195	var_196
0	train_0	0	8.9255	-6.7863	11.9081	5.0930	11.4607	-9.2834	5.1187	18.6266	...	4.4354	3.9642	3.1364	1.6910	18.5227	-2.3978	7.8784
1	train_1	0	11.5006	-4.1473	13.8588	5.3890	12.3622	7.0433	5.6208	16.5338	...	7.6421	7.7214	2.5837	10.9516	15.4305	2.0339	8.1267
2	train_2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427	14.6155	...	2.9057	9.7905	1.6704	1.6858	21.6042	3.1417	-6.5213
3	train_3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428	14.9250	...	4.4666	4.7433	0.7178	1.4214	23.0347	-1.2706	-2.9275
4	train_4	0	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405	19.2514	...	-1.4905	9.5214	-0.1508	9.1942	13.2876	-1.5121	3.9267
5	train_5	0	11.4763	-2.3182	12.6080	8.6264	10.9621	3.5609	4.5322	15.2255	...	-6.3068	6.6025	5.2912	0.4403	14.9452	1.0314	-3.6241
6	train_6	0	11.8091	-0.0832	9.3494	4.2916	11.1355	-8.0198	6.1961	12.0771	...	8.7830	6.4521	3.5325	0.1777	18.3314	0.5845	9.1104
7	train_7	0	13.5580	-7.9881	13.8776	7.5985	8.6543	0.8310	5.6890	22.3262	...	13.1700	6.5491	3.9906	5.8061	23.1407	-0.3776	4.2178
8	train_8	0	16.1071	2.4426	13.9307	5.6327	8.8014	6.1630	4.4514	10.1854	...	1.4298	14.7510	1.6395	1.4181	14.8370	-1.9940	-1.0733
9	train_9	0	12.5088	1.9743	8.8960	5.4508	13.6043	-16.2859	6.0637	16.8410	...	0.5543	6.3160	1.0371	3.6885	14.8344	0.4467	14.1287

10 rows × 202 columns

Figure 1: First 10 data set rows.

The following tasks were performed during data exploration:

- Checking if data is balanced
- Checking if features have normal distribution
- Checking features correlation
- Checking univariate and multivariate outliers
- Checking missing values

While looking for data balance, we found out that 89.951% of the customers in our training data set wouldn't make the transaction of interest in the future. Thus, our group of interest in the training data is composed of 10.049% of the rows, resulting in an imbalanced data set.

We used D'Agostino and Pearson's [4,5] test to check our data normality. The test results assured that none of the features is normally distribution, though visual inspection of data histogram might fool some observers.

In order to calculate features correlation the Pearson correlation coefficient was used. The intensity of the correlations were judged according to table 1. No correlations were found between features.

[4]D'Agostino, R. B. (1971), "An omnibus test of normality for moderate and large sample size", Biometrika, 58, 341-348

[5]D'Agostino, R. and Pearson, E. S. (1973), "Tests for departure from normality", Biometrika, 60, 613-622

Correlation	**Intensity**
$score > 0.7$	strong
$0.4 < score \leq 0.7$	medium
$0.2 < score \leq 0.4$	weak
$score \leq 0.2$	none

Table 1: Correlation intensity.

In the pursuit of outliers, we used Tukey's[6] method. It consists of using the inter-quartile range (IQR) — distance between first quartile (q1) and third quartile (3q) — to detect whether a point belongs to the set or is an outlier. Points that fall out of the following interval are considered outliers: $[q1 - 1.5 * IQR, q3 + 1.5 * IQR]$. Looking from an univariate perspective, 12.448% of the observations may be considered outliers. Looking from a multivariate perspective, only 0.778% of the observations may be considered outliers.

Finally, while looking for missing values we found out that we had a perfect data set. No observations are missing, so no preprocessing that could harm our results will be needed for this particular task.

2.2 Data Visualization

Some graphics were created during the project execution in order to make the data understanding clearer. The first graphic, presented in figure 2, shows us how imbalanced our training data is.

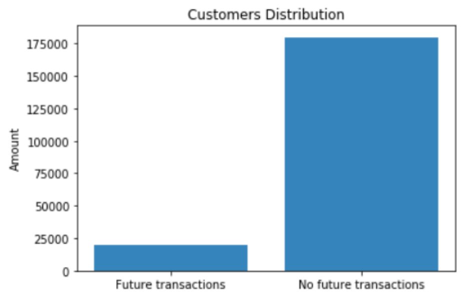


Figure 2: Customers distribution.

[6]Tukey, JW. Exploratory data analysis. Addison-Wesely, 1977

The following three graphics, presented in figures 3, 4 and 5, show us how some of our data features are distributed. A normal curve with the same mean and variance of the features is plotted along with its probability histogram. Due to our data extent, 200.000 rows, it would be expected that our data would have converged to a normal distribution, but that didn't happen. We can see clearly in our graphics that the histograms are not adjusted to the normal curve. Plotting the histogram alone could be misleading since its bell shape could be interpreted as a normal distribution in most features.

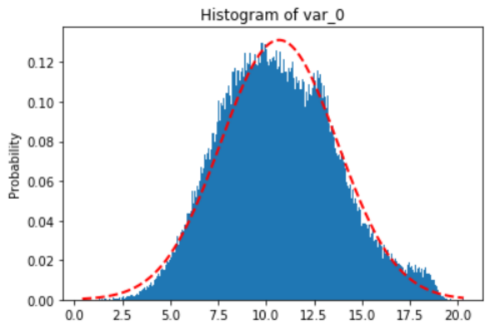


Figure 3: Histogram of feature var_0 with equivalent normal curve.

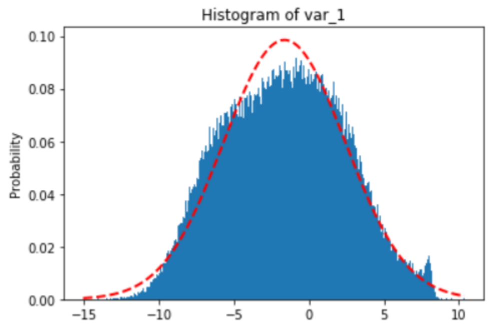


Figure 4: Histogram of feature var_1 with equivalent normal curve.

Finally, our last graphic is about the features importance, presented in figure 6. A random forest model was used without any parameter tuning only to provide us the features importance values. If any feature had great importance among the 200 then further exploration would be done regarding

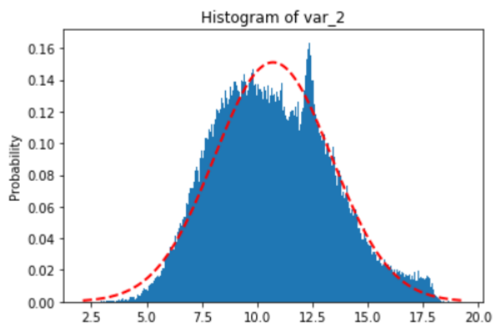


Figure 5: Histogram of feature var_2 with equivalent normal curve.

the feature. No feature had great importance in our data set, leading us to drop further explorations.

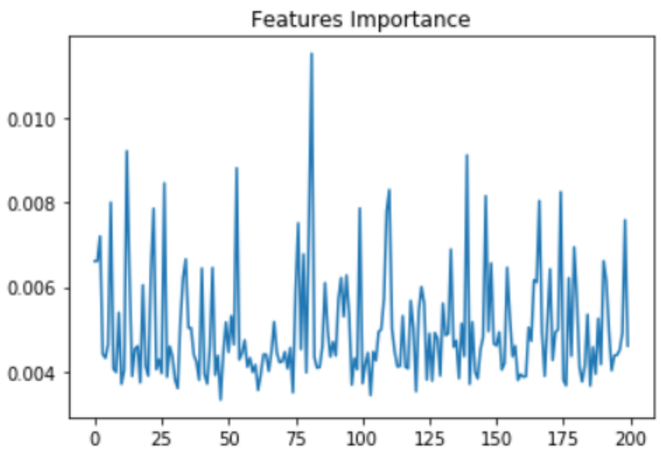


Figure 6: Features importance.

2.3 Algorithms and Techniques

1. Complement Naive Bayes: Naive Bayes Classifiers are based on Bayes' Theorem. Those kind of classifiers assume features independence and develop a model based on this assumptions. The Complement Naive Bayes in particular was created to correct severe assumptions made by the standard Multinomial Naive Bayes Classifier an is particularly suited for imbalanced data sets.

2. Logistic Regression: Logistic regression is and algorithm that implements regularized logistic regressions using several distinct solvers, chosen by the developer. It's a linear method that uses a logistic function. Although its results are easily understood it's quite sensitive to data anomalies.

3. Random Forest Classifier: The Random Forest Classifier is a meta estimator, it uses decision trees on sum-samples to improve the predictive accuracy. The algorithm tries to control over-fitting and has the option to select sub-samples with replacements if the bootstrap parameter is set to True. As a decision tree, results might be intutitive even for untrained observers.

4. XGBoost Classifier: XGBoost stands for eXtreme Gradient Boosting, which is an algorithm designed to be efficient, flexible and portable. It can be looked as a form of ensemble of weak prediction models. Adding to the stated points, it can be parallelized, distributed and chache optimized, providing an optimal performance regarding time.

5. LightGBM: Similar to XGBoost, LightGBM is yet another gradient boosting framework, this one based on tree algorithms. Designed to have fast training speed and low memory usage, it's an algorithm capable of dealing with large-scale data.

6. K Nearest Neighbours (KNN) Classifier: KNN Classifier makes predictions based on the closest neighbours to the point being predicted. To do this it has to calculate the distance between all points, which becomes very costly with the increase of data or number of features. It's a very flexible algorithm, but some preprocessing is needed to keep features in a similar range so that no feature has a bigger impact when calculating distances, unless intended.

2.4 Benchmark

The chosen benchmark for this project was the Complement Naive Bayes Classifier. Our problem consists of finding the probability of a customer making a specific transaction, therefore we expect that we'll be looking into a rather small group of clients. For cases where the data set is imbalanced the chosen benchmarks is expected to perform well[7].

3 Methodology

3.1 Data Preprocessing

During preprocessing we dealt with two things: outliers and data scaling. On the outlier side, we considered univariate outliers to be too big of a fraction of our data set, therefore we didn't do anything about them. Nevertheless we considered multivariate to be a small fraction of our data set, which lead us to create a copy data set without multivariate outliers for further model performance testing.

On the data scaling side, we used MinMaxScaler in order to make every feature contained in the [0,1] interval. This preprocessing step allowed us to use the chosen benchmark model and have a better performance regarding the KNN Classifier.

3.2 Implementation

The choice of an appropriate model was the most challenging part of this project. Initially we tried tuning the parameters for a Logistic Regression model, but couldn't get a performance over 70%, so this model was kept only for reference. LightGBM came up after some more tets.

Not all models tested are present in the project in order not to make it too large. The two models tested but not present in the project were Decision Trees and CatBoost. Like Random Forests, Decision Trees had great training but low testing performance, suggesting overfitting. CatBoost was one of the last models tested, at this stage we were looking for a implementation of gradient boost that suited the problem.

A troubling aspect of choosing the model was the time needed to rerun models. Since we used a pipeline, when deciding to change the models in the pipeline we had to rerun all of them due to the implementation used. This had a great impact in the pace of the project, delaying the results. CatBoost

[7]https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.ComplementNB.html#sklearn.naive_bayes.ComplementNB

was a model that took a long time, therefore it was excluded from the project due to bad performance.

The first model to be tested was the benchmark model. After that, a pipeline was created for the supervised learning algorithms that followed the sklearn standards, being the ones used in this project the following: Random Forest, Logistic Regression, XGBoost. Following the pipeline, the LightGBM and KNN Classifier were tested. For this initial comparison the models that had a random_state attribute had it set to 42 and no further parameter tuning was performed. All models, including the benchmark, were evaluated using stratified cross-validation with 10 folds. After comparing the area under the ROC curve, the model chosen as appropriate for the task at hand was the LightGBM.

3.3 Refinement

The parameter tuning of the LightGBM was done following the model's documentation[8] and community posts shared accross websites such as stackoverflow and other forums. The parameters are as follows:

Parameter	**Value**
random_state	42
metric	auc
bagging_freq	5
bagging_fraction	0.35
min_data_in_leaf	256
learning_rate	0.005
objective	binary
max_leaves	80
num_threads	4

Table 2: Tuned parameters for the LightGBM model.

With the model properly tuned we had two tasks at hand: deal with imbalanced data and decide whether to keep outliers or not. To make these decisions a random sampling of 10% of the data was done.

[8]https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html#tune-parameters-for-the-leaf-wise-best-first-tree

We decided between the complete data set or the one without outliers. Both had equivalent performances, therefore the one without outliers was chosen due to its smaller size requesting less computational power.

After that we had to decide between three approaches for the imbalanced data: oversampling the minority class, undersampling the majority class or a mix of oversampling and undersampling. As the three approaches had similar performances to the model ran without any of them, we decided to not use any of them since they made the model training times rather big.

After all those tests our final model used a data set without multivariate outliers and the tuned parameters present in table 2.

4 Results

During the project we tested several models in different ways. Our first result was the benchmark model, which had the following performance:

Model	[train] ROC AUC	[test] ROC AUC
Complement NB	0.777	0.777

Table 3: Average area under ROC curve for the benchmark model.

The results obtained during the model selection phase were the following:

Model	**[train] ROC AUC**	**[test] ROC AUC**
Random Forest	0.927	0.508
Logistic Regression	0.628	0.627
XGBoost	0.512	0.508
LightGBM	0.908	0.855
KNN	0.506	0.501

Table 4: Average area under ROC curve using the complete data set for each model.

The results found while deciding whether to use the complete data set or not were the following:

The results found while deciding whether to use the data set or not were the following:

Dataset	**[train] ROC AUC**	**[test] ROC AUC**
Normal	0.996	0.870
Filtered	0.997	0.861

Table 5: Average area under ROC curve using normal and filtered data sets for LightGBM.

Balance Method	**[train] ROC AUC**	**[test] ROC AUC**
Oversample	0.992	0.869
Undersample	0.992	0.869
Mixed Method	0.992	0.869

Table 6: Average area under ROC curve using complete and filtered data sets for LightGBM.

Although in the notebook where the code is implemented the [train] ROC AUC for the balancing methods differ, if we take a look at the cross validations splits we can see that they have the same values, so the difference might be a bug we couldn't find a solution to.

4.1 Model Evaluation

Our final model was executed with 100% of the data excluding multivariate outliers. The results obtained were as follows:

[train] ROC AUC	**[test] ROC AUC**
0.992	0.896

Table 7: Average area under ROC curve for the final model.

Along with the results obtained with the training data, there is the Kaggle submission result. By sumiting to Kaggle a mean of the predictions made with the testing data set a score of 0.898 was obtained.

4.2 Justification

Based solely on the results we obtained with our final model, which are over 80% for either the cross-validation or the kaggle submission, we can say we have done very well with our proposed solution. Looking at the chosen benchmark we can say we've done an excellent work. Not only our final model is better than the chosen benchmark, it has an improved score of over 10%.

5 Conclusion

5.1 Free-Form Visualization

Figures 7 and 8 bring to us the features importance information. The first one shows all 200 features in decreasing order of importance — text might be hard to distinguish in the report but is readable in the project file — and the second one shows the 50 most important features. The value associated with each feature is the number of times it was used in the model.

5.2 Reflection

Overall this project has been very challenging. The difficulty of dealing with anonymized data for the first time was quite big. Nevertheless, all barriers were overcome and the results surpassed what was initially expected, some score near low 80's.

One of the best aspects was the process was that during the proposal the reviewer's suggestions were on point with the problem at hand and helped a lot to develop a overall better project.

Although better solutions to this problem surely exist, I'm satisfied with the results being this the first time I've dealt with anonymized data, which kinda restrain you in the feature engineering and understanding overall.

5.3 Improvement

I think even greater results may be obtained through a thorough parameter tuning, but it may require quite some time investment from the developer. Another thing that could be tried is model stacking, to see if some combination of models suits the problem better than the one proposed in this project.

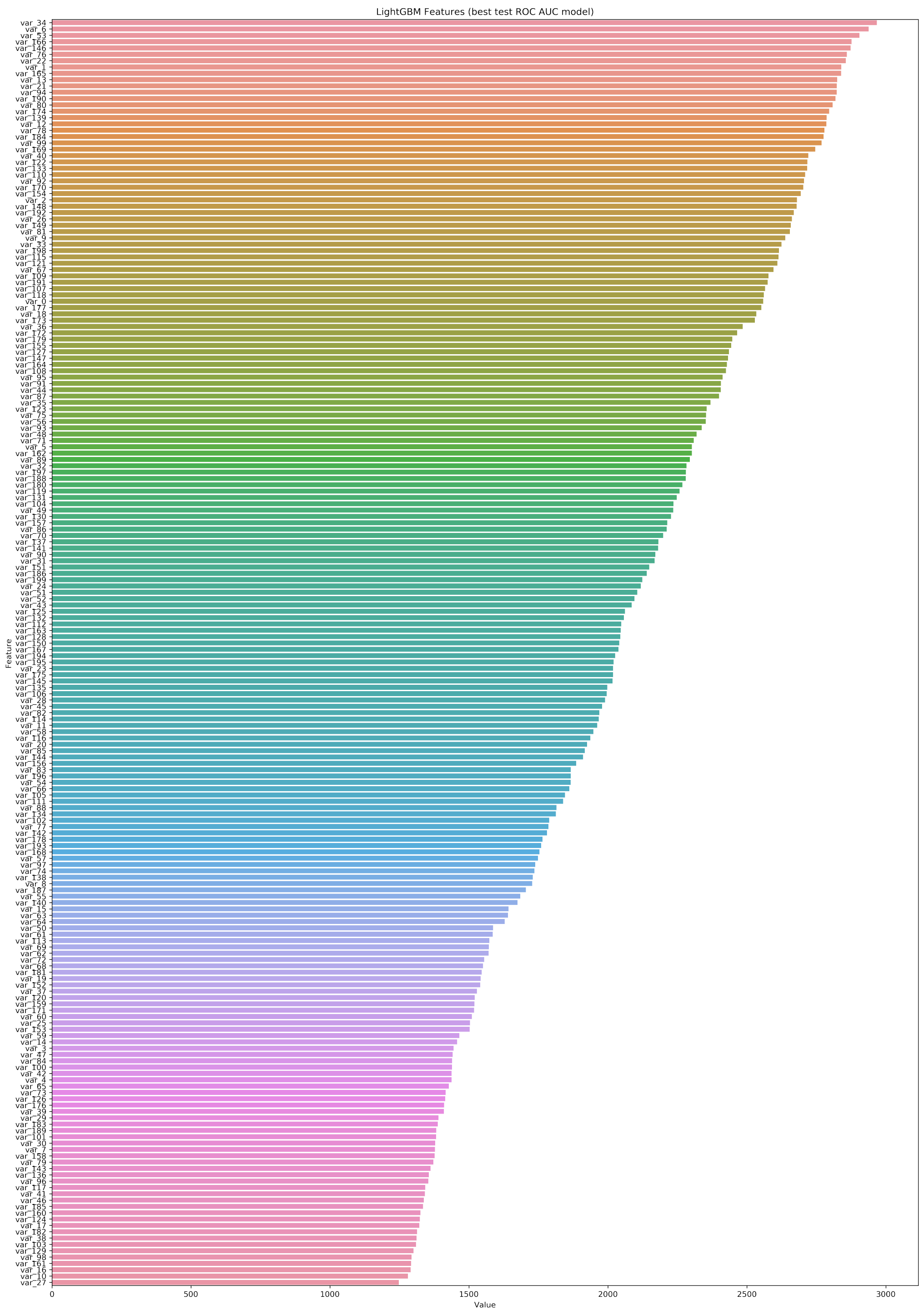


Figure 7: Plotting features importance from model with best test ROC AUC.

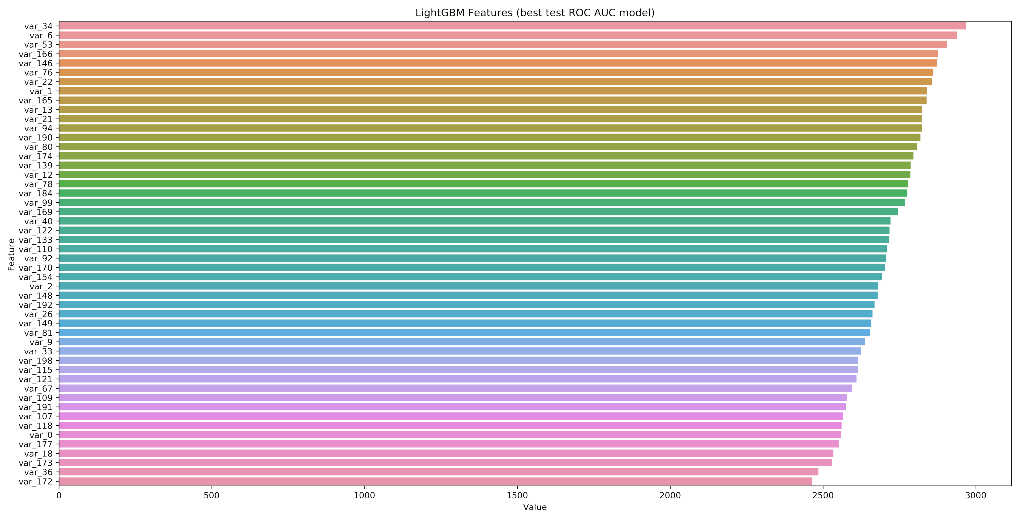


Figure 8: Plotting top 50 important features from model with best test ROC AUC.