Avito Demand Predcition Challenge

Abhishek Bhatt
Kathakali Banerjee

Abstract

The online advertisement industry is filled with a variety of advertisements that suffer from lack of proper optimization. While some are too vague, others are not aesthetically pleasing or some might just not have enough demand in the market. The project aims at predicting the probability of online ads selling something based on a variety of categorical and numerical features like category, geographical location and price or date respectively along with historical demand for such ads.

1 Objective

The objective of the project is to implement a number of algorithms to predict the 'deal_probability', i.e.the target variable whose value lies between zero and one, compare the results from each of them and make conclusions about which algorithm gives us the best results.

2 Overview of Raw Data

The training and test sets consist of 18 columns which are as follows:

Categorical Features:

Region - Ad region
City - Ad city
Parent_category_name- Top level category by the Avito Model
Category_name - Fine grain level category by the Avito Model
Param_1 - Optional parameter
Param_2 - Optional parameter
Param_3 - Optional parameter
Title - Ad title
Description -Ad description
User_type - User type for the ad
Image - Id code of the image

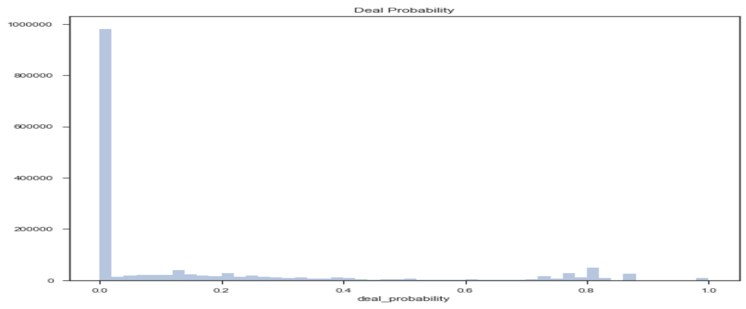
Numerical Features:

Item_Id - Item Id of the ad
User_Id - User ID of the ad
Price - Price of the ad
Item_sequence_number -Ad sequential number for user
Activation_Date - Date ad was placed
Image_top_1 - Avito's classification code for the image
Deal_probability - This is the likelihood of an ad actually selling something. The deal_probability

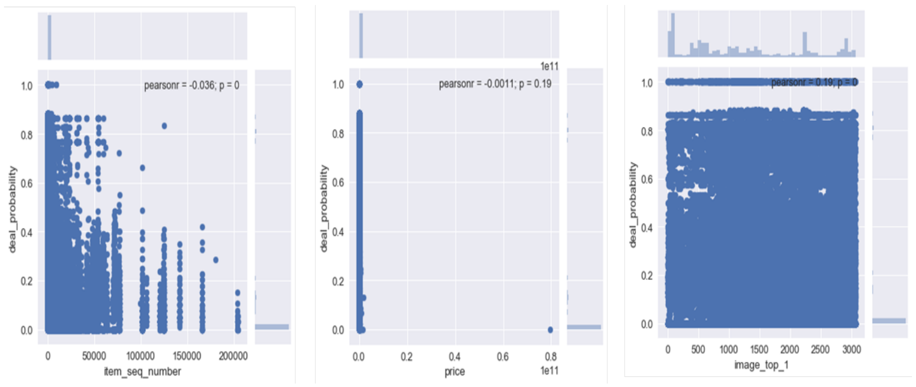
feature is part of the training set.
The train set has 1503424 rows while the test set has 508438 rows.

3 Exploratory Data Analysis

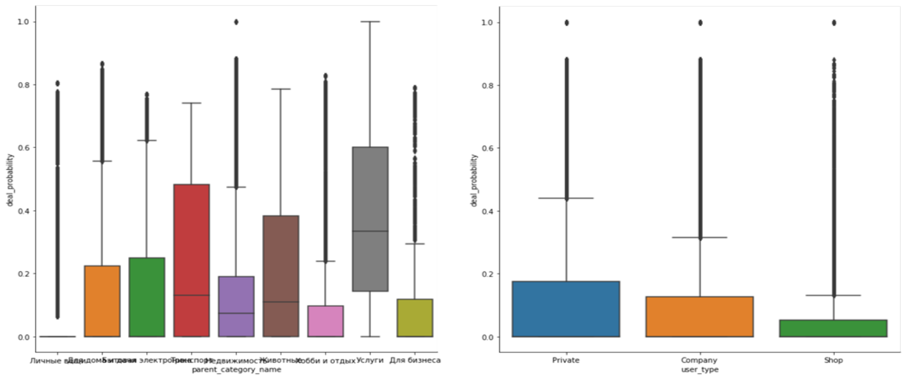
We have plotted a distribution of the *deal_probability* variable of the train set for preliminary visualization. The *deal_probability* for most instances are found to be zero.



Apart from this, the features do not show any significant correlation among themselves. When selectively they are plotted against the response, they do not convey any meaningful relationship as well.



Some categorical features with less unique values do not show any linear separability with the target variable which can be see from the plots below:



However, fitting the OLS model has generated results that show significant correlation between most features and the target.

4 Data Preprocessing

4.1 Handling Missing Data

The missing data constitutes a significant percentage of the entire data set size. We have imputed the missing values with mean values of the respective columns.

4.2 Handling Categorical Features

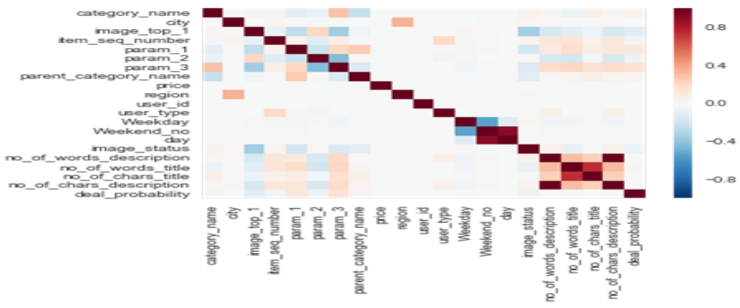
We have used the Label Encoding to convert each of the categorical features except *Activation_date*, *Title* and *Description*, into explicit numerical values.

We have parsed the *Activation_date* feature into 3 new columns each having the day, month and year individually.

For *Title* and *Description* features, we have followed two steps. First we have generated the number of words and number of characters for each of them, resulting in new 4 columns. Additionally, we have included an *image_status* column which denotes one if image filename is present in the raw data and zero otherwise. We will refer to this data as the *preprocessed_data* for future reference.

5 Implementation of Algorithms

First, we will present a heatmap of the correlation of the different features of our processed data.



It can be depicted that most of them are uncorrelated that makes it more important to perform various regression algorithms for prediction purpose.

We have implemented the following algorithms to predict the *deal_probability* of the test set. They are presented below:

5.1 Bayes Ridge Regression

Bayes Ridge Regression on the *preprocessed_data* gives us a score of 0.2912. The parameters chosen were:

```
BayesianRidge(n_iter=300, tol=0.001, alpha_1=1e-06, alpha_2=1e-06, lambda_1=1e-
06, lambda_2=1e-
06,compute_score=True, fit_intercept=True, normalize=False, copy_X=True, verbose=False)
```

5.2 Lasso

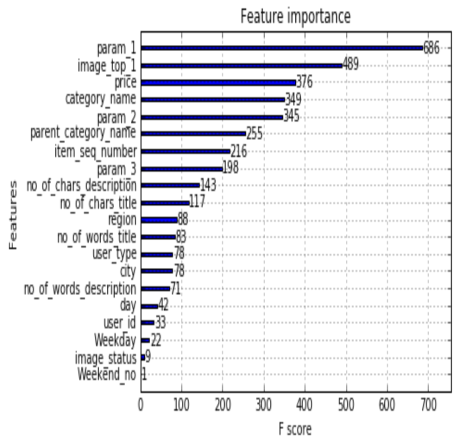
Lasso was 5 fold cross validated on the features *'image_top_1'*, *'item_seq_number'*,*'param_1'*,*'param_3'*,*'image_status'*,*'no_of_words_description*, *'no_of_chars_description* and the other features were suppressed. We obtained a score of 0.2617 which is better than Naive Bayes. The parameters chosen were *normalize=True, lassomodel.alpha_ = 5.5436516538551617e-06, lassomodel.intercept_ =0.14024966451290563*

5.3 XG Boost Regression

We performed basic XGBoost Regression with hyperparameter tuning over *max_depth in range(9,12), min_child_weight in range(9,12)*,*subsamplein [i/10. for i in range(7,11)]*,*colsamplein [i/10. for i in range(7,11)]* and *eta [.3, .2, .1, .05, .01, .005]*. The parameters chosen were:

```
{'base_score': 0.5,
 'booster': 'gbtree',
 'colsample_bylevel': 1,
 'colsample_bytree': 1.0,
 'eta': 0.3,
 'gamma': 0,
 'learning_rate': 0.1,
 'max_delta_step': 0,
 'max_depth': 9,
 'min_child_weight': 6,
 'missing': 9999999999,
 'n_estimators': 9,
 'nthread': 4,
 'objective': 'reg:linear',
 'reg_alpha': 0,
 'reg_lambda': 1,
 'scale_pos_weight': 1,
 'seed': 1301,
 'silent': 1,
 'subsample': 0.7}
```

Without hyperparameter tuning, the score obtained was 0.2717 while the score after hyperparameter tuning was 0.2364. In the subsequent stage, we have plotted the feature importance and eliminated features below Fscore =70.



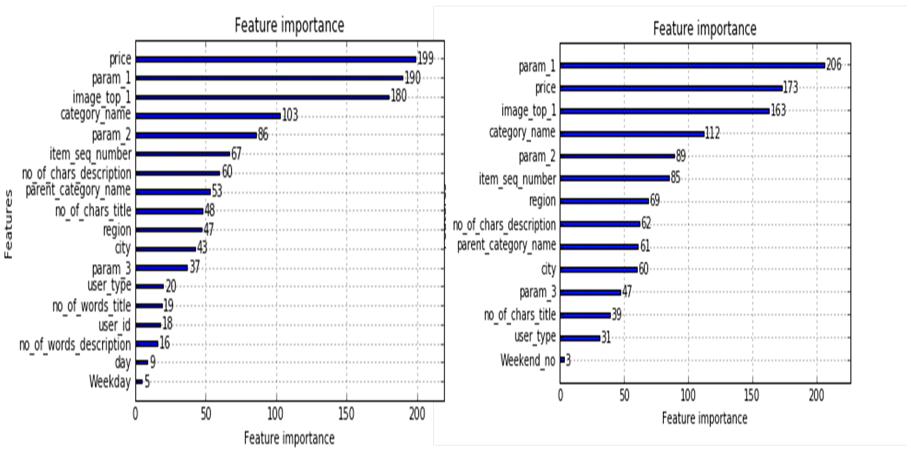
So after dropping the features - *'day'*,*'user_id'*,*'Weekday'*,*'image_status'*,*'Weekend_no'*, we obtain a score of 0.2363 which is an improvement over the previous score by 0.002. We randomly remove instances with *deal_probabilty* as 0 and chose *frac* as 0.2 of such rows and the score on XGBoost was 0.2662.

5.4 Light Gradient Boosting Regression

This is a variant of XGBoost but is five times faster. Initially, we have perform hyperparameter tuning over *learning_rate in [0.01, 0.1, 1]* and *n_estimators in [20, 40, 60, 80]*. The parameters chosen were:

```
LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
       learning_rate=1, max_depth=-1, min_child_samples=20,
       min_child_weight=0.001, min_split_gain=0.0, n_estimators=40,
       n_jobs=-1, num_leaves=31, objective='regression', random_state=None,
       reg_alpha=0.0, reg_lambda=0.0, silent=True, subsample=1.0,
       subsample_for_bin=200000, subsample_freq=0)
```

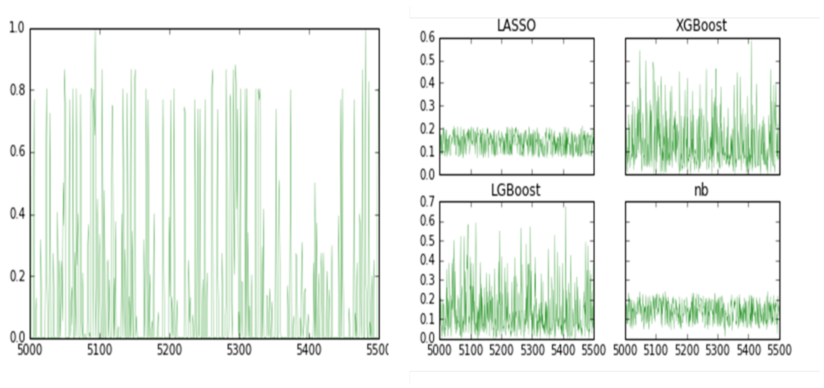
The LGBM score without parameter tuning was 0.2471 and with parameter tuning was 0.2343. Subsequently, we have plotted the feature importance after parameter tuning on the left figure. We eliminated the features below Fscore 20 like *'no_of_words_title'*,*'user_id'*,*'no_of_words_description'*, *'day'*,*'Weekday'* and obtained score of 0.2338. The new feature importance is plotted on the right figure:



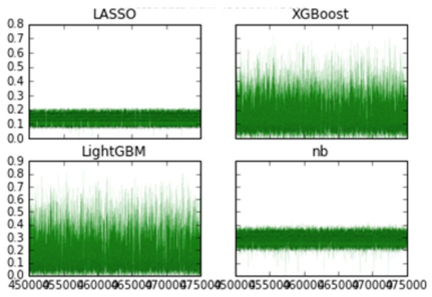
We eliminate three more features in the next step like *’Weekend_no’,’user_type’, ’no_of_chars_title’*. However, this time, we obtained an accuracy of 0.2344 which indicates that score starts getting worse now.

5.5 Evaluation of Algorithms

The left graph below shows the true probability on train data (rows-5000:5500) and the right graph indicates probability of different algorithms on the same train data(rows-5000:5500).



The graph below indicates the probability on test data (rows-450000:47500). We can see that XGBoost and LightGBM map the probabilities better than Lasso and Nave Bayes, i.e, their predicted range of probabilities is higher than the other two.



5.6 Comparison of Scores

We have tabulated the performances of all the algorithms. See Table 1.

Table 1: Comparison of Algorithms

ALGORITHM	LASSO	BAYESIAN RIDGE	LGBOOST	XGBOOST
PUBLIC SCORE	0.2617	0.2912	0.2346	0.2362
PRIVATE SCORE	0.2663	0.2937	0.2381	0.240
APPROX TIME TO RUN	15.9s	7.03s	11.7s	6 min 3 s
private scores are calculated on 69% of test data				

5.7 Stacking The Outputs

The following tables show what fitting the models on the output of the algorithms result in:

Fitting Lasso on outputs

ON ALL OUTPUT	0.2334
ON ALL OUTPUT EXCEPT LASSO	0.2335

Fitting LGBoost on outputs

ON ALL OUTPUT	0.2399
ON ALL OUTPUT EXCEPT LGBOOST	0.2450

OLS

OLS ON ALL OUPUT	0.2334

6 Conclusion

- An initial data analysis did not reveal much information about the relationship between the features with the predicted variable or relationship among the features themselves.
- The model that has given the best score among the ones used is the LGBoost Regression Algorithm.
- Removing features with less importance has steadily decreased the scores but for LGBoost, the score started to increase after a certain number of features had been dropped.
- Using simpler algorithm to fit the outputs improves the score. In the above case using LASSO on the output of Bayes Regression, LASSO, XGBoost, LGBM gave slight improvement over all individual algorithms.