

Prediction Model for Avito Duplicate Ads Detection Challenge in Kaggle

Exploratory Data Analysis

I. Abstract

Online shopping has become a more and more common method for many customers. Customers could look through different choices when they want to pick the ideal products. However, there is also an issue that some sellers will post the same advertisement multiple times with very minor changes, which might lead the customers to check the same products multiple times. If we could develop a model to predict whether two ads are the same or not, the customers will benefit from it and avoid looking at identical ads.

The goal of our project is to develop a model that predicts whether the pairs of ads are duplicated or not. During this project, we fitted different models and checked the best AUC scores among those models. We also added new features and tried parameter adjustments to further improve the accuracy. After fitting various models, we found that the Boosting method generates the highest scores, which is 0.76812.

II. Data Description

The original data contains several datasets, and we needed to merge the datasets first to make clean training set and test set that contain information of the paired items in each observation.

The original training set contains two datasets, ItemPairs_train and ItemInfo_train.

ItemPairs_train contains four variables that are identifier of first ad, identifier of second ad, if the two items are duplicates or not, and training data generation method. Iteminfo_train contains eleven variables that are ad identifier, identifier of ad category, ad title, full text with the ad's description, list of image identifiers associated with this ad, additional parameters of the ad in JSON format, ad price, ad location identifier, identifier of a closest metro station to ad location, latitude of ad location, and longitude of ad location.

The original test set also has two datasets with the same variables as the training set except that the ItemPairs_test set does not have the information of whether the two items are duplicates or not, which is what we are going to predict.

We first merged the ItemPairs and ItemInfo sets to display the paired ads in the same observations. We also created new variables that will be used for building models and making

predictions. Since this project wants to predict whether two ads are duplicate or not, we assumed if the seller post the same items twice, he or she probably will use same information for the two ads. When we first created variables, we focused on whether the information completely match or not. As we tried different models, we also though comparison of the differences of the items could also be helpful features, so we created several more features.

Therefore, the final dataset contains 2991396 observations and 18 variables that are: identifier of first ad(itemID_1), identifier of second ad(itemID_2), whether the two items have the same category ID(categoryID), whether the two items have the same title(title), whether the two items have the same description(description), whether the two items have the same additional parameters in JSON format(attrs), the absolute difference of the two items' prices(price), whether the two items have the same location ID(locationID), whether the two items have the same metro ID(metroID), the absolute difference of the two items' latitudes(lat), the absolute difference of the two items' longitudes(lon), the pairwise string distances between elements of the two items' titles by using 'jw' method(titleStringDist), the pairwise string distances between elements of the two items' titles by using 'lcs' method(titleStringDist2), the pairwise string distances between elements of the two items' descriptions by using 'jw' method(descStringDist), the difference of the number of characters of the two items' titles(tc), the difference of the number of characters of the two items' descriptions(dc), whether the two items have same price(pm).

Additionally, variable categoryID, title, description, attrs, price, locationID, metroID and pm are binary variables with value 1 and 0. 1 means the content or value of the two items are the same for this variable, while 0 means they are different. Variable lat, lon, titleStringDist, titleStringDist2, descStringDist, tc, and dc are numeric variables.

Among these variables, we first created itemID_1, itemID_2, categoryID, title, description, attrs, price locationID, metroID, lat and lon to try various models. We transformed lat and lon while trying to improve the model, and titleStringDist, titleStringDist2, descStringDist, tc, dc, and pm were created as we tried various methods to improve the final prediction's AUC score. More details about the effects of these variables will be discussed in the model building section.

To eliminate missing values, we chose to replace the missing values with 0 if it is a factor variable, which means these the two items do not have the same values. For the numeric variables, we replaced the missing value with the mean of that variable.

Considering the size of the original dataset, it was difficult to fit models with the full dataset, therefore, we used a random sample of the training set when we fitted the models. The sample contains 29914 observations

III. Data Visualization

After cleaning the data and creating some explanatory variables, we wanted to generate some plots to get more insight of this dataset and the relationship among the variables. By looking at various plots, we could find if some variables have strong correlation with the response variable.

And if some features show strong correlations, we would also consider combining those features or conducting variable transformations.

i. Correlation Matrix

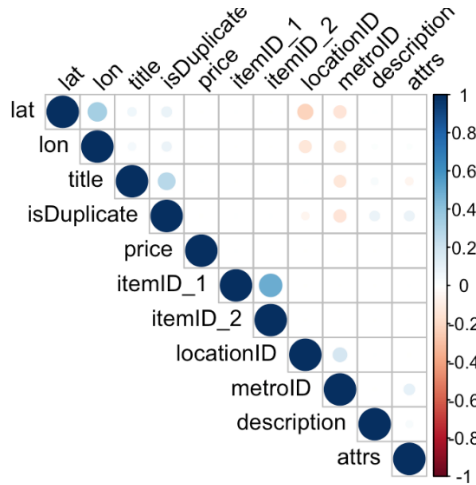


Figure 1. Correlation Matrix Plot

The correlation matrix plot above indicates that the response variable isDuplicate has relatively stronger positive correlation with title, while variable lat, lon, description, attrs also shows positive correlation with isDuplicate. In addition, metroID has negative correlation with isDuplicate, and locationID has relatively weaker negative correlation with isDuplicate compared with metroID.

Among the explanatory variables, lat and lon have obvious positive correlation, considering the nature of latitude and longitude, we would add distance which is the square root of lat-square and lon-square as a variable to check if adding distance will help improve the AUC score or not, and this will be further discussed in the model building section.

IV. Unsupervised Methods

ii. K-Means Clustering

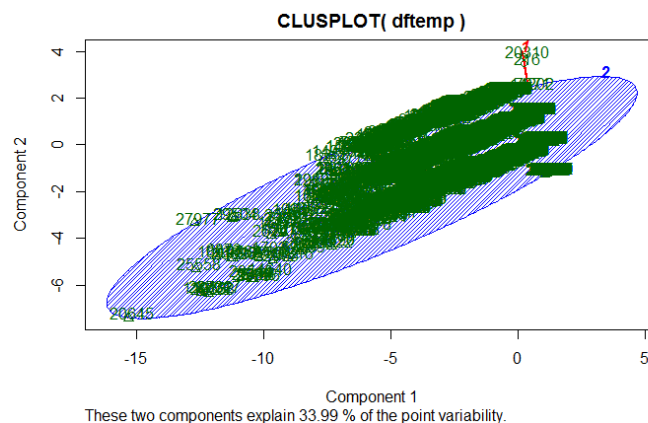


Figure 2. K-Means Clustering Plot

The clustering method groups and separates observations based on their similarities and dissimilarities. Since the response variable has two levels, we wanted to check if the K-Means cluster could separate the observations into two clusters. The plot above shows that the two components explain 33.99% of the point variability, and it is not easy to get an obvious separation of these two clusters from this plot. Therefore, we considered dimension reduction approach to get further understanding of the dataset and variables.

iii. Dimension Reduction Approach

In the dimension reduction process, we used principle component analysis(PCA) method to reduce the dimensions, and analyzed if any variable shows significant effect.

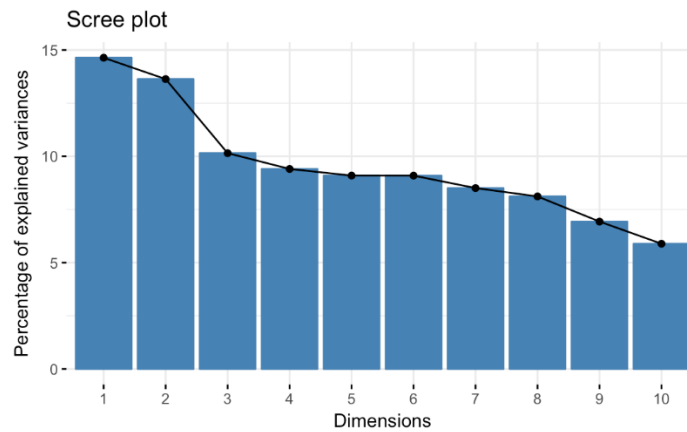


Figure 3. Principle Components Analysis Plot

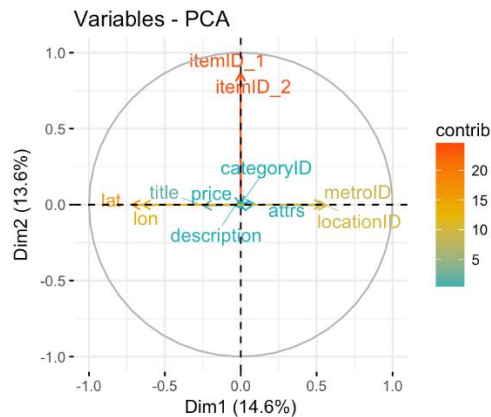


Figure 4. PCA Component Contribution Plot

From the plots above we could see that the first seven components represent about 80% of the variances, among which the first two components represent about 28.2% of the variances. The plot indicates that itemID_1 and itemID_2 have the most contributions, however these two variables are just identifiers that will not be used in the models. Besides, lat and lon also show relatively high contributions, metroID and locationID also have high contributions compare with

the rest variables. It is obvious to see that the PCA component contribution plot indicates the same pattern as the correlation matrix plot.

After having some insights about the dataset, we moved to the model building section.

Model Building

In this section, we will present our models and show how we improved our final AUC score in the test data.

We first used title, description, attrs, price, location ID, metro ID, lat, lon as explanatory variables, and isDuplicate as response variable. We excluded category ID because all the paired observations in the train set have the same category ID, because of the lack of levels, this variable does not show significant effect to the prediction.

Therefore, the equation for the models we originally fitted is:

Model 1: isDuplicate~title+description+attrs+price+location ID+metroID+lat+lon

We fitted *Model 1* with linear discriminant analysis(LDA) model, quadratic discriminant analysis(QDA) model, logistic regression model, decision tree model, random forest model, various GBM boosting models with different parameters, various neural network models with different layers and parameters, and xgboost models with different parameters. After picking the best method, we started to create new variables to see if the AUC score can be improved. At the meantime, we also tried to tune the parameters within the functions to test which are the best parameters for this model.

In general, the boosting models have the best AUC scores, while the rest models do not seem to have AUC scores that are close to the boosting method. After adding new variables and tuning the parameters, we also used various ensembles of the best predictions to check if we could further improve the AUC scores.

I. Successful Models

i. Boosting Models

Among the models we first fitted, the boosting models seem to have the highest AUC scores. Therefore, we further explored how to improve the boosting model. We first tried to adjust the parameters within the function to test what parameter values perform better for this model. Since the sample size of the dataset is relatively large, we predicted that the higher the n.trees and interaction depth, the better the boosting model will perform.

The results of the boosting models with different parameters are listed as below:

- (1) The AUC score for the Boosting model with n.trees=1000 is 0.6896
- (2) The AUC score for the Boosting model with n.trees=2000 is 0.7054
- (3) The AUC score for the Boosting model with n.trees=5000 and interaction depth=4 is 0.7358

- (4) The AUC score for the Boosting model with n.trees=5000, interaction depth=4, and shrinkage=0.1 is 0.7227
- (5) The AUC score for the Boosting model with n.trees=5000, interaction depth=4, and shrinkage=0.1, and n.minobsinnode=1 is 0.701
- (6) The AUC score for the Boosting model with n.trees=5000, interaction depth=4, and shrinkage=1, and n.minobsinnode=1 is 0.6827
- (7) The AUC score for the Boosting model with n.trees=5000, interaction depth=4, and shrinkage=0.5, and n.minobsinnode=1 is 0.666

From the list above we could see that the higher n.trees, the better the prediction. However, the change of shrinkage and n.minobsinnode does not seem to improve the performance. Since we could not find shrinkage and n.minobsinnode values that could generate better AUC scores than the default values, we decided to use the default values for these two parameters. We will also try different interaction depth levels in the following steps while we add more variables.

ii. New Variables, Variable Transformation and Parameter Adjustments

- **titleStringDist, descStringDist, tc, dc variables**

Then we found that we could study more about the title and description variables, and make better use of the information these variables provided.

Since these two variables contains lots of text, we wanted to see more of the similarity or difference of the paired items' texts, so we created a variable that shows the pairwise string distances between elements of the two items' titles by using 'jw' method(titleStringDist), jw method is an algorithm that calculates the Jaro distance between the two items. We also created a variable that shows the pairwise string distances between elements of the two items' descriptions by using the same method(descStringDist).

Sometimes the seller might post the same item with slightly different titles and descriptions, therefore, the duplicated ads may not have the exactly same titles and descriptions. But if there is a small difference between the length of the two items' titles or descriptions, that might also indicate that the two ads are identical or have high similarities.

Therefore, we created two more variables that represent the difference of the number of characters of the two items' titles(tc), and the difference of the number of characters of the two items' descriptions(dc). The new model is:

Model 2: isDuplicate~title+description+attrs+price+location ID+metroID+lat+lon+titleStringDist+descStringDist+tc+dc

We used the same n.trees=5000, but increased the interaction depth parameter to 8, after fitting the model, the AUC score in the test data was improved to 0.75261.

Next, we increased the n.trees to 6000, while kept the interaction depth at 8, and the AUC score increased to 0.75364. Based on the improvement of the AUC score above, it seems that adding features and increasing n.trees could improve the predictions.

- **pm variable**

To further improve the prediction result, we planned to add more features into the models. The price variable we used before represents the difference of the two items' prices. Since the sellers possibly will post the duplicated ads with the same prices, we also created a variable called pm, which indicates whether the two items have the same prices. The variable pm is a binary variable that has two levels 1 and 0, of which 1 means the prices are the same and 0 means they are different. And the model became:

Model 3: isDuplicate~title+description+attrs+price+location ID+metroID+lat+lon+titleStringDist+descStringDist+tc+dc+pm

After adding this new variable, and fitting the boosting model with n.trees=5000, interaction depth=8, the AUC score increased to 0.76205. Then we tried when n.trees=6000, the AUC score increased to 0.76229. Next, we tried to increase n.trees to 8000, but the AUC score decreased to 0.76229.

Through the comparison of these three results, it seems that n.trees=6000, and interaction depth=8 are the best parameters.

- **distance variable**

As the correlation matrix and pca component contribution plot show before, the variable lat and lon has strong positive correlation, and they also show relatively high correlation with the isDuplicate variable. Sometimes even the two items have different latitude and longitude, their locations might still be very close. If we could create a variable that reflects how far the two locations are, it might help to make the prediction more accurate. Therefore, we decided to transform these two variables to make a new variable that shows the distance between the two items' locations. We took the square root of the lat-square and lon-square, and obtained the new variable distance. We first fitted the model below which still contains the original lat and lon variables:

Model 4: isDuplicate~title+description+attrs+price+location ID+metroID+lat+lon+titleStringDist+descStringDist+tc+dc+pm+distance

Fitting Model 4 with boosting that has n.trees=6000, interaction depth=8, we obtained a higher AUC score of 0.76240.

However, since distance already contains information about lat and lon, we wanted to check how the AUC score change if we exclude lat and lon, and the model became:

Model 5: isDuplicate~title+description+attrs+price+location ID+metroID+titleStringDist+descStringDist+tc+dc+pm+distance

Then we fitted the boosting model with the same parameters, the AUC score was higher, which was 0.76273. Therefore, we decided not to include lat and lon in the future models.

- **titleStringDist2 variable**

Since we found adding titleStringDist was helpful for improving the AUC score, we wanted to create a new variable that reflects the pairwise string distances between elements of the two items' titles by using the 'lcs' method and named it titleStringDist2. Now the model is:

Model 5: isDuplicate~title+description+attrs+price+location ID+metroID+titleStringDist+descStringDist+tc+dc+pm+distance+titleStringDist2

After fitting the boosting model with n.trees=6000 and interaction depth=8, the final AUC score increased to 0.76812, which is also the best AUC score we obtained.

II. Unsuccessful Models

i. Models without Tuning Parameters

We also used LDA, QDA, logistic regression, decision tree, random forest methods when fitted *Model 1*. Since these methods did not generate good AUC scores compared with the boosting method, we did not try to further use these methods to improve the AUC scores.

ii. Xgboost Models

Since xgboost method computes very fast, we also tried to fit xgboost models and tuning the parameters. The major parameters we tuned are max.depth, eta, nthread, and nround. We tried different values of max.depth from 4 to 10, eta from 0.05 to 0.5, nthread of 50 and 100, and nround of 50 and 100. We first changed one parameter while kept the other parameters fixed, and then we tried different combinations of the parameters. However, none of these models had a good AUC score.

Even though the xgboost models do not have complex computation, it does not seem to be the best method for the model we wanted to fit. But we found that when max.depth is 6 and eta is smaller, it seems to have a better prediction result. However, the change of nthread and nround does not seem to affect the predictions very significantly.

iii. Neural Network

We also conduct various neural network models with different layers and parameter combinations. We tried to use different unit, activation, input_shape, rate, and optimizer, however, we could not find good parameters that could generate an AUC score above 0.7.

However, we found that when activation is sigmoid, input shape is 8, rate is 0.7, lr is 0.1, epochs is 5, and batch size is 128, the accuracy of the neural network model is the best, even though it is still lower than the Boosting models.

III. Ensembles

After trying various models above, we found that the best AUC scores were generated by the GBM boosting models. Then we wanted to try various ensembles to check if we could get a higher AUC score.

i. Average Ensemble and Weighted Ensemble

We first created an average ensemble by using the predictions of the best 4 boosting models. The average ensemble was formed by multiplying the top 4 accurate predictions by 0.25, and then calculating the sum of them. However, it did not generate a good AUC score.

Then we tried a weighted ensemble by multiplying the top 4 accurate predictions by 0.5, 0.2, 0.2, 0.1. The weighted ensemble could not provide a higher AUC score either.

ii. Random Forest Ensemble

Even though the random forest method was not able to provide a good prediction when we fitted the models before, we still wanted to check if we could get a good random forest model by training the best predictions.

We trained a random forest model by using the predictions of the best 4 boosting models in the training set, by setting the predictions as the new explanatory variables, and isDuplicate as the response variable. Then we created a new dataset with the 4 boosting models' predictions in the test set, and fit the random forest model with this new test set. However, the final AUC score of the random forest ensemble is lower than 0.76812.

iii. Neural Network Ensemble

We also tried to build neural network ensembles with various models we fitted before. We used different combinations of activation, input shape, rate, lr, epochs, and batch size values. After tuning the parameters, the best AUC score we obtained is still much lower than 0.76812.

In summary, even though we tried various ensembles, they did not generate predictions that are more accurate compared with the best boosting model we obtained before. However, among these ensembles, the weighted ensemble with appropriate weight seems to perform better. Based on all the scores we obtained and the analysis, we decided to choose the best GBM boosting model as our final model.

Discussion

I. Summary

In this project, we fitted various models and tried different variables to improve the AUC score in the test data. We tested different models, and added new features at the same time. However, not all features would help to improve the AUC score, we dropped several features as we found that they have negative effects to the final predictions. In the end, we were able to improve the final AUC score to 0.76812. Our final boosting model contains 13 explanatory variables that are title, description, attrs, price, location ID, metroID, titleStringDist, descStringDist, tc, dc, pm, distance, titleStringDist2. And the best AUC score was obtained from the GBM boosting model with n.trees=6000 and interaction depth=8 by using the features above.

II. Possible Future Work

If we had more time and known more about R, we would like to add the images as predictors, and try more variables that are related to title and description, for example, extracting key words within the text. We would also like to run everything on more advanced computers for computational efficiency. Since the AUC we got from neural network is low, we would love to change our learning rate to a changing learning rate instead of a fixed learning rate. However, this requires better support of the computation work.

III. Hurdles and Problems

The major problem we encountered is the size of the dataset. It is difficult to fit some model which are computationally heavy such as SVM, and also it is not feasible to use caret package in R to do parameter tuning. In addition, Google Cloud seems to be unstable sometimes, and it is not responding sometimes because of the heavy computation work. Another problem we encountered is since the random sample of the training set is relatively small compared with the complete training set, the models that performed well on the training set might not give good predictions on the test set. Therefore, if we used caret on the training set to find optimal parameters, the parameters might not fit very well on the test set.

Reference

- [1] <https://www.kaggle.com/c/avito-duplicate-ads-detection>
- [2] <http://cran.r-project.org/web/packages>