Final model

Yoni Slutzky Itay Ofer

2023-06-12

Our final solution

Our method of modelling was split into the following steps: - Feature engineering and data imputation, for which we allocated 60% of the training set - Model tuning, for which we allocated 30% of the training set - Model validation, for which we allocated 10% of the training set

We began our work with feature engineering and imputation of missing values. Feature engineering was split into 4 feature-extraction functions, as-well as a recipe that was applied before training/inferring using the model. The first function was clean_features(), which was used for cleaning the original variables and performed the following: - lower casing character variables - imputing missing numerical / binary values with zeros - replacing character 'na' values with 'unknown' - standardizing the variables 'heel_height' and 'vintage' - generalizing the variable "location" to the country location

The second function was style_features(), which was used to standardize the 'style' variable. The function unified and cleaned many of the values, and then, using the results produced for the eng set, kept only 'style' values which appeared more than a cutoff parameter times. This was used to keep only relevant values.

The third function was brand_features(), which was used to standardize the 'brand' variable and enhance it by "taking a look at" the 'title' variable. The function first unified and cleaned many of the values, and then, using the results produced for the eng set, kept only 'brand' values which appeared more than a cutoff parameter times. This was used to keep only relevant values. Next, the function used the results of the eng set to also look for matches in the 'title', keeping the results for samples whose transformed 'brand' so far was empty. Lastly, the function used word embeddings and KNN to cluster the remaining samples.

The fourth and last function was title_features(), which was used to extract text features from the 'title' variable. The function cleaned the 'title' values from stop-words, extracted common-sense and statistically significant words (w.r.t the 'log_price' variable) in the form of new binary variables, and applied the textfeatures() function to extract text features such as sentiment measures and word statistics.

Before applying the modelling, we used a recipe which included step_novel() to treat new nominal values not seen in training, and step_dummy() to create binary variables from all nominal variables. The total amount of variables in our model was more than 1000.

For modelling, we chose to use the GBT model. We tuned our model on the tuning set by using 5-fold cross-validation. Pretty early on in our tuning process we realized the hyperparameters to be used are $lr = 0.01, sample\_size = 0.75$. In our first attempts we also tuned the amount of trees between one of $\{500, 750, 1000\}$, with 1000 being the clear favorite. As our attempts continued we raised the amount of trees, and since the optimization process became slower, we stopped tuning and when straight to the validation step. The final model used $lr = 0.01, sample\_size = 0.75, trees = 10000$.

When validating our model, in addition to examining the straight rmse results, we also compared them to the rmse results of the baseline model proposed above. Our validation set was reaching better results in the baseline than the results of the test set (results published on the leaderboards), and thus it was clear from the beginning that the results of our model on the test set won't be as good as they were on the validation set. Examining the proportion of rmses of the validation set allowed us to gauge our model's performance on the test set with much more confidence.