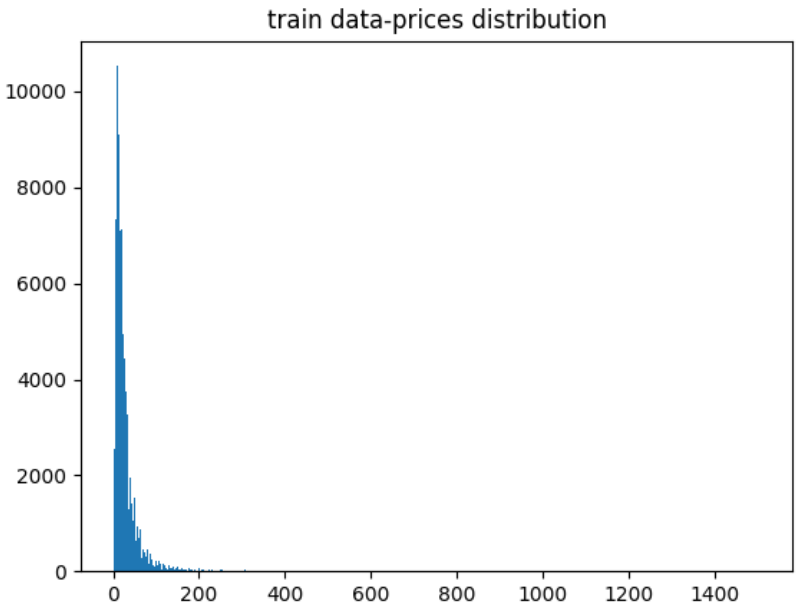
eBay interview, analysis of my solution of Mercari Price Suggestion Challenge

Nir Diamant

June 2020

1 Data analysis

The first thing I did was observing the data. The data contains about a million lines of entries, which varies by all kinds of parameters, especially by the categorical ones, which are the sentences. I looked over the prices distribution, and as expected for a lot of item prices, the distribution is log-normal:



2 Data Pre-processing

2.1 data cleaning

The data had a lot of invalid entries, there for what I did: 1) create a pandas dataframe of the table, with corresponding column titles
1) remove all entries where there is no price
2) remove all entries where the brand name is 'Nan'
3) The relevance of the item description seemed low to me, as a factor of the price, while it contains a lot of information, which would be interpreted as a lot of noise to the model, so I decided to omit it.

2.2 Sentence shortening

Currently, the categorical columns are name, item category, and item brand. the last two are informative, but the name is not always.
Therefore I was looking for a way to extract the two words that describe the item name the best, out of the name description.
To do that, I used the TF-IDF method over the first 100,000 item name sentences and picked for each sentence the two words with the highest value.

2.3 Creating a vocabulary

The next thing to do is to translate all names into number, therefore I created a vocabulary containing all the words of the item names (two words each). For the item category and brand, I referred to the whole string as one word for the vocabulary, because the whole category path seemed more relevant to me than splitting the words, i.e. 'Electronics/Computers' may be treated differently than 'Electronics/Batteries', even though they have the Electronics section in common. For the brand, it is more obvious that i.e. 'Factory 54' should be treated as one word and not two. the vocabulary size is about 23k words.

3 model training

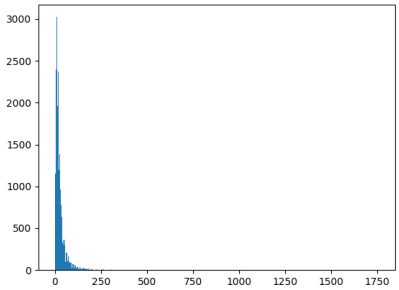
Now there is a table containing only numbers, yet some of them are numerical, and some refer to categorical data. I used a fully connected neural net model: the first layer was a concatenation of embedding of the each of the categorical entries with the continues entries. I chose the embedding output dimension to be 50. The network architecture consists of 3 fully connected layers with 25 neurons each, and a prediction one neuron at the top.
Due to that we are solving a regression problem, having a data that distributes log-normal, which means there are a lot of low-medium priced items, but a very few with a high price, I chose to use a mean square log error loss.
network hyper-parameters:
LR = 0.08

decay = 1e-03
num epochs = 3
batch size = 128

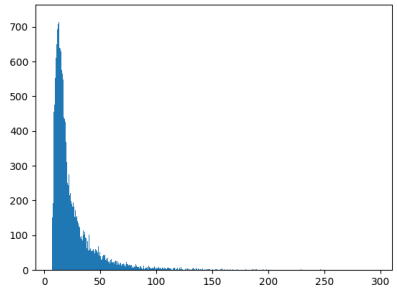
the number of epochs was low, due to fast convergence of the validation set, and after that, the train set started to over-fit.
Due to the long computation time of a massive amount of data, I trained my model on an 80,000 entries training set and tested it on 20,000 entries.
The results have beaten the first place in the Mercari Price competition, yielding a loss of 0.289 (first place 0.377). (though the results may be different when testing on another test file).

4 Results analysis and post processing

First I looked at the test data price distribution with compare to the prediction price distribution:



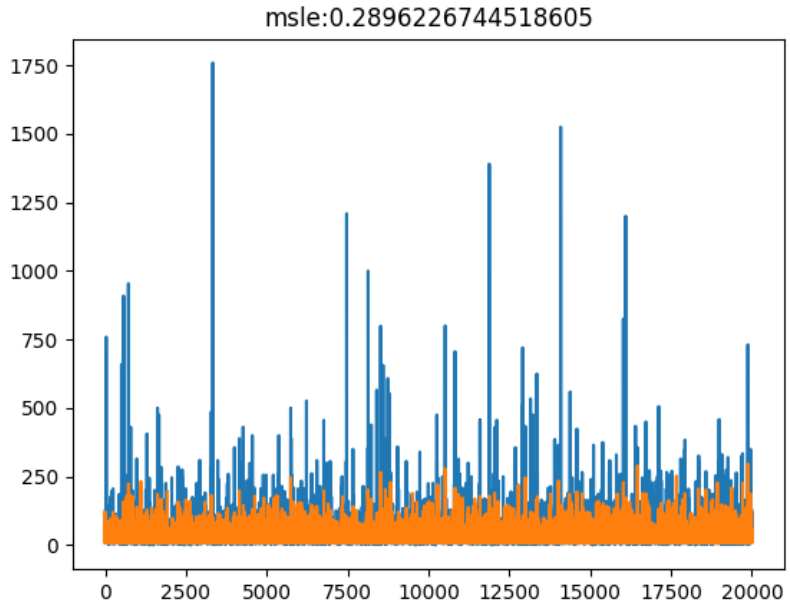
(a) test price distribution



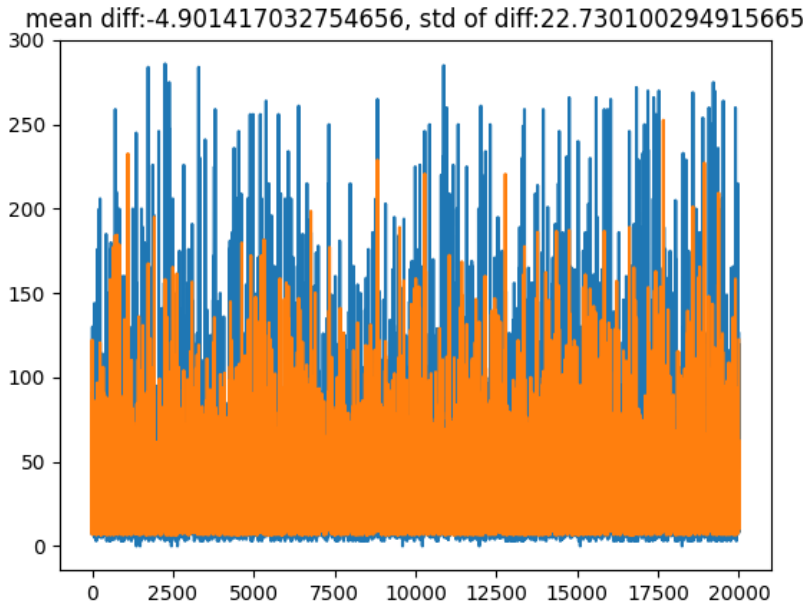
(b) prediction price distribution

Both distribute log-normal, yet it seems that the model tends to predict smaller values on average and is smoother. the big values in the real prices are so rare that they can not even be seen in the histogram

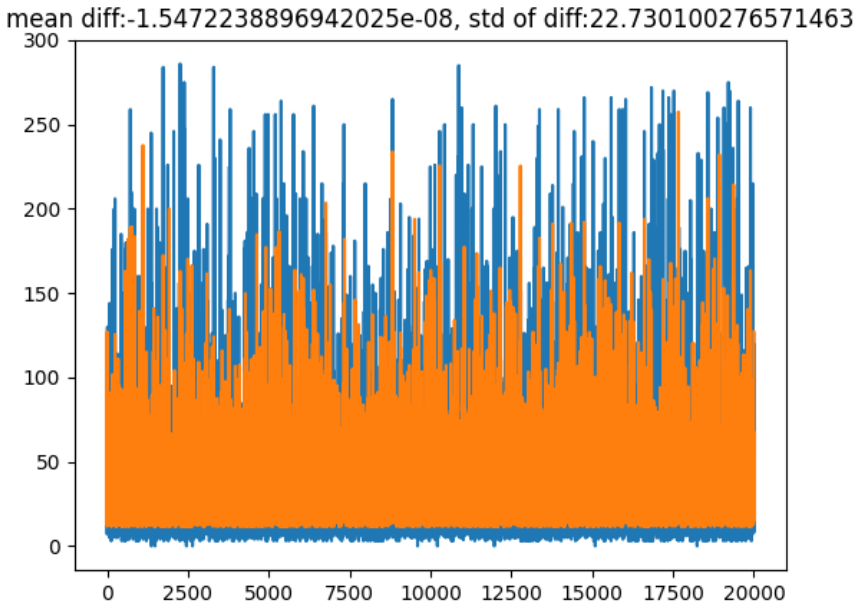
Now lets look at the prices themselves together (blue is the real ones, orange predicted):



Now I chose to check what is the number of values in the real prices that are higher than the highest value the model predicted. I found out that it is about only 91 out the 20,000 test entries, which is only 0.455% of the test data.
I decided to refer to these entries as outliers, so let's look at the results without them:



Now it can be seen how good the model predicts.
It can be seen the model still predicts always a bit less the real prices, so I decided to add manually to each prediction, the mean difference which is 4.9.
The last model performs this way:



5 Answering questions

1) The business problem is by definition predicting the correct price for a new item according to some parameters, including numerical and categorical.
Yet the seller's purpose is to maximize his profits.
Let's define the following variables:
item price: p
predicted price : $\hat{p}$
total items sold: q
total expenses: s

The total profits are defined by: $p \cdot q - s$
s is an unknown variable for us, but hence it's constant, we shall maximize the term $p \cdot q$.

For doing this we may predict a **reasonable** price $\hat{p}$ for our product, and experimentally over each category find what modification we may apply for the model price prediction to maximize the profits.
Hence the formal definition for an unknown price for an item would be:
$\arg\max_{\hat{p} \pm \Delta} p \cdot q$

2) The model's pros:

- It omits a lot of irrelevant data (subjective to my opinion)
- The loss metric fits the data distribution - rare high values do not affect the model the same it might if we would use i.e. MSE.
- Very short running time (a matter of minutes)
- Very high accuracy level (mean and std of difference regarding the real test prices.

The model's cons:

- the omission of the data might lose data which is informative
- bigger error rate when the actual price is very high (yet it relates only to less than 0.5% of the data).

3) The first thing I could do with more data is to enhance the name section information. the titles given by the sellers are many times not informative/accurate enough, or contain a lot of irrelevant information for the predicting model.
I could train a CNN network for classifying an object by its image. Then I could replace each name title by the classification output of its image (For those who have images, I guess it's the most of them). I believe it would be much more accurate and informative.
also, for any new item name, I could measure statistics over all the same items, make sure the predicted price lands in a plausible range.
furthermore, I could feed the network with additional information on the mean and std of the group.