Machine Learning Engineer Nanodegree

Capstone Proposal

Ajay Kumar Misra
May 22th, 2018

Domain Background

When selling used goods online, a combination of tiny, nuanced details in a product description can make a big difference in drumming up interest. But, even with an optimized product listing, demand for a product may simply not exist–frustrating sellers who may have over-invested in marketing.

https://www.kaggle.com/c/avito-demand-prediction/data

Russia's largest classified advertisements website, is deeply familiar with this problem. Sellers on their platform sometimes feel frustrated with both, too little demand (indicating something is wrong with the product or the product listing) or too much demand (indicating a hot item with a good description was underpriced).

Problem Statement

Avito is challenging us to predict demand for an online advertisement based on its full description (title, description, images, etc.), its context (geographically where it was posted, similar ads already posted) and historical demand for similar ads in similar contexts. With this information, Avito can inform sellers on how to best optimize their listing and provide some indication of how much interest they should realistically expect to receive.

Datasets and Inputs

To make it easier to download the training images, Avito has added several smaller zip archives that hold the same images as train_jpg.zip. We are free to use either train_jpg.zip or the smaller zip archives.

File and column descriptions

- **train.csv** - Train data.
    - item_id - Ad id.
    - user_id - User id.
    - region - Ad region.
    - city - Ad city.
    - parent_category_name - Top level ad category as classified by Avito's ad model.
    - category_name - Fine grain ad category as classified by Avito's ad model.
    - param_1 - Optional parameter from Avito's ad model.

    - param_2 - Optional parameter from Avito's ad model.
    - param_3 - Optional parameter from Avito's ad model.
    - title - Ad title.
    - description - Ad description.
    - price - Ad price.
    - item_seq_number - Ad sequential number for user.
    - activation_date- Date ad was placed.
    - user_type - User type.
    - image - Id code of image. Ties to a jpg file in train_jpg. Not every ad has an image.
    - image_top_1 - Avito's classification code for the image.
    - deal_probability - **The target variable**. This is the likelihood that an ad actually sold something. It's not possible to verify every transaction with certainty, so this column's value can be any float from zero to one.
- **test.csv** - Test data. Same schema as the train data, minus deal_probability.
- **train_active.csv** - Supplemental data from ads that were displayed during the same period as train.csv. Same schema as the train data minus deal_probability, image, and image_top_1.
- **test_active.csv** - Supplemental data from ads that were displayed during the same period as test.csv. Same schema as the train data minus deal_probability, image, and image_top_1.
- **periods_train.csv** - Supplemental data showing the dates when the ads from train_active.csv were activated and when they were displayed.
    - item_id - Ad id. Maps to an id in train_active.csv. IDs may show up multiple times in this file if the ad was renewed.
    - activation_date - Date the ad was placed.
    - date_from - First day the ad was displayed.
    - date_to - Last day the ad was displayed.
- **periods_test.csv** - Supplemental data showing the dates when the ads from test_active.csv were activated and when they were displayed. Same schema as periods_train.csv, except that the item ids map to an ad in test_active.csv.
- **train_jpg.zip** - Images from the ads in train.csv.
- **test_jpg.zip** - Images from the ads in test.csv.
- **sample_submission.csv** - A sample submission in the correct format.
- **train_jpg_{0, 1, 2, 3, 4}.zip** - These are the exact same images as you'll find in train_jpg.zip but split into smaller zip archives so the data are easier to download.

Solution Statement

My solution would involve inputting the data into XGBoost/LightGBM Model. For each item_id in the test set, we will predict a probability for the deal_probability. These predictions must be in the range [0, 1].

Benchmark Model

At this point of time, the random guessing would be the benchmark for this project. Assumption is that deal probability prediction would be either all 0 (no ad sold anything) or all 1 (all ads sold something). Linear Regression would be used as benchmark model.

Evaluation Metrics

- Metric used would be root mean squared error. RMSE is defined as:
    - $\mathrm{RMSE}=(1/n\sum i=1\mathrm{to}n(yi-y\hat{}i))**2)**1/2$
- where y hat is the predicted value and y is the original value.

Project Design

Here is the suggested flow for this project.

- Load data using python’s pandas library in the .csv forms.
- Analyze the features contributing to deal_probability.
- Find the target variable for prediction..
- Develop various models like Linear Regression, XGBoost and LightGBM.
- Execute the models on training data, and validate the predictions on test data.
- Thinks of improving the model even further by tuning parameters.