Context Ad Clicks

Metis Project Mcnulty - Classification

Mailei Vargas

February 2019

Motivation

Websites such as *Craigslist* and *Avito* provide a platform for regular people and businesses a place to post "ads" of items they are looking to sell. While there is a free option to post your item for sale, Avito in particular, offers additional pay options that increase the visibility of your being seen by potential buyers. Is it worth paying for such an ad? The answer lies in the outcome of machine learning.

Problem Statement

Avito is the website in question. They offer three levels of advertising for a seller. As mentioned earlier, there is a free or "regular" tier. This tends to get pushed lower in the list of results from a buyers search. There is a "highlighted" tier, a onetime fee, where the ad is litterally highlighted in yellow in order to draw attention to it. Lastly, the "contextual" option, where the seller pays everytime their ad is clicked. The contextual ads are more likely to bubble up to the top of the list of a buyers search.

The challenge is to predict whether a user of the site clicked on a context ad or not. The data contatined eight relational tables which reflects information about searches, ads and users. Since this is a classification problem, classifier models such as logistic regression (to set a benchmark) and tree based models were used. The dataset had an imbalance of *yes/no* clicks and precision/recall in this case is best balanced, so I settled on the $F1$ score as a metric for measuring the model performance.

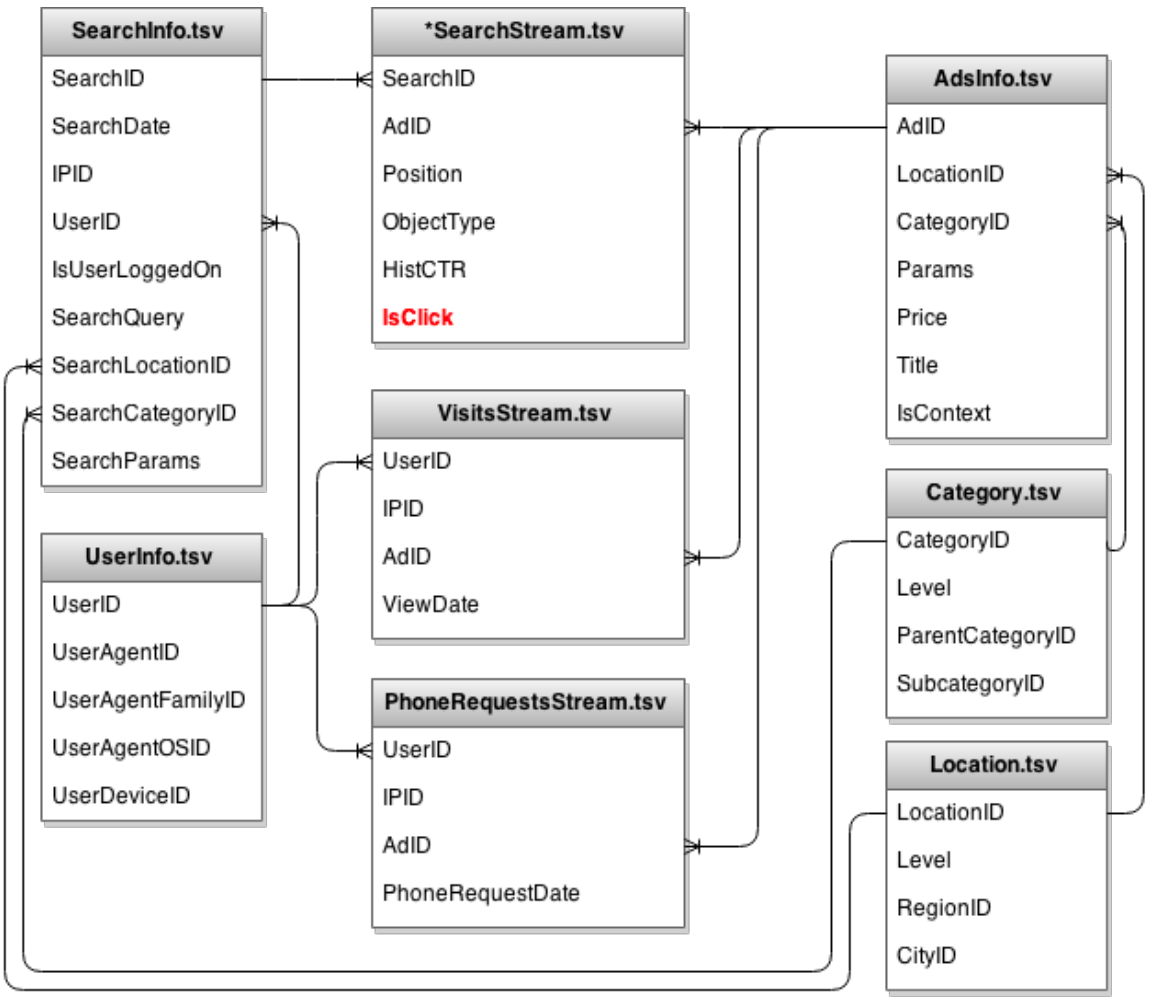
Tools

Below is a table of the libraries, sites, and tools that aided in the completion of this project.

Websites	Tools	Libraries for Analysis/Modeling	Custom functions scripts
Kaggle	Python	numpy	visualize.py
Avito	Jupiter Notebook	pandas	modeling.py
	PyCharm	matplotlib	
	GitHub	seaborn	
	PostgreSQL	xgboost	
		catboost	

Data Processing

The dataset was provided in eight relational tables some of which contained hundreds of millions of rows (not exagerating). Below is the schema of the tables.



Since there was an abundance of data that I was ill equiped to handle here is a brief path of the journey the data took:

`.tsv` -> `.csv` -> `AWS` -> `postgreSQL` -> `local postgreSQL`

Once I had a proportion (1 million+) of the data to work with locally, this made for faster run times on queries. The original dataset contained too many entries that a simple select query would take a long time to run, even after inserting indexes on certain columns.

Feature Enginering

SQL was very a useful means of extracting features and joining tables. I was able to save those new tables as CSV files and import into a Jupyter Notebook for additional feature engineering. Below is a list of the 15 features used in the models. Some of these features were engineered after some exploration on basic model performance.

- *category* - a value 1 - 57 representing an undisclosed category of the item in the ad.
- *day_of_week* - a value 0 - 6 representing Sunday - Saturday respectively of the search.
- *hour_of_day* - a value 0 - 23 representing the hour of the search.
- *region* - a value 1 - 84 representing an undisclosed region.
- *position* - a value of 1 or 7 representing the position of the context ad in the returned list.
- *hist_ctr* - a given CTR rate
- *price* - price of the item in the ad
- *num_searches* - the number of prior searches of a user
- *num_days* - the number of a days between a user first search and last search
- *num_land_page_views* - the number of landing page views of a user
- *num_phone_req* - the number of phone requests by a user
- *price_ratio* - the rank of the price relative to the mean price per category
- *title_length* - the length of the title of the ad
- *price_search* - the interaction variable between the price and num_searches
- *pos_hist* - the interaction variable between the postion and the hist_ctr

Algorithms and Techniques

Prior to finalizing some of the above features a quick "out-of-the-box" run through some models was done as a benchmark. Below are the results for each model:

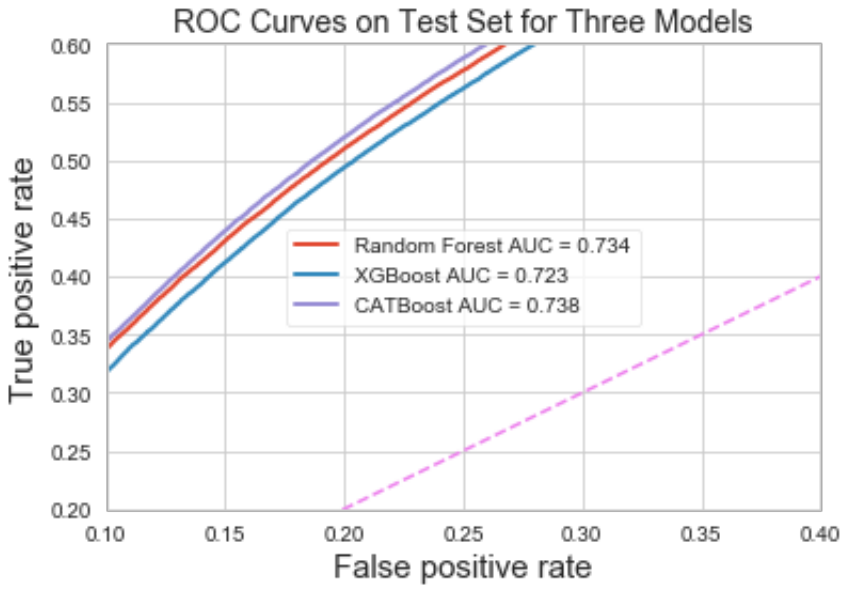
```
logistic F1 Score = 0.5658277117408227
logistic ROC AUC score = 0.5428506610806293
random-forest F1 Score = 0.653372742500729
random-forest ROC AUC score = 0.7202440754886555
knn F1 Score = 0.588154600637882
knn ROC AUC score = 0.6290507752419141
nb F1 Score = 0.6585748850326074
nb ROC AUC score = 0.5860326093467949
xg-boost F1 Score = 0.6713712639132153
xg-boost ROC AUC score = 0.7389025698611109
cat-boost F1 Score = 0.6672154439457837
cat-boost ROC AUC score = 0.7312150867967241
```

This gave me a quick idea of which models performed best and I would work on tuning those.

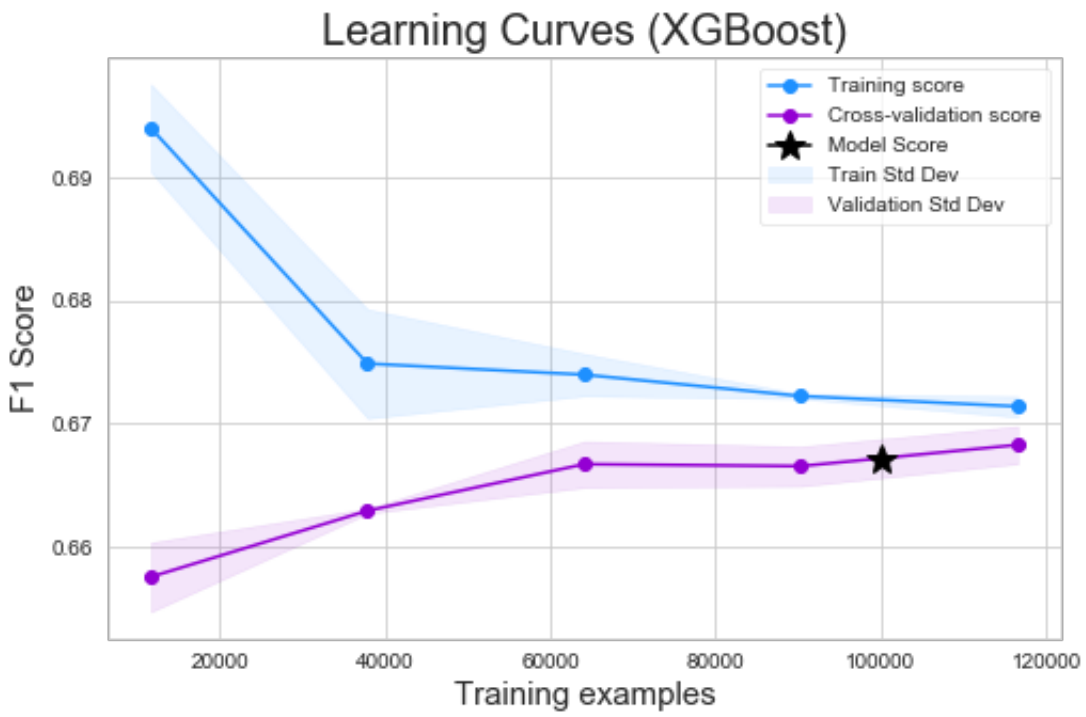
- Random Forest
- XGBoost
- CATBoost

I preformed first a `RandomGridSearch` which gave me a basis for `GridSearch` and I used the results to train, validate and lastly test the data. This entire process was cyclical and refined each time.

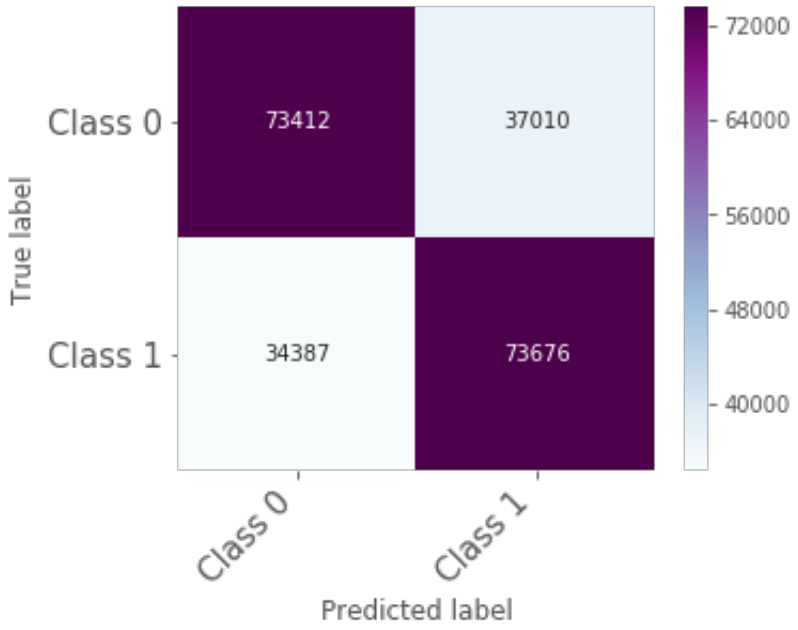
The results of the ROC curve was basically the same for the three tree based models with an AUC score of ~0.72 or 0.73 on the test data. Note that this graph is "zoomed in" to capture the slight difference in the 3 models.



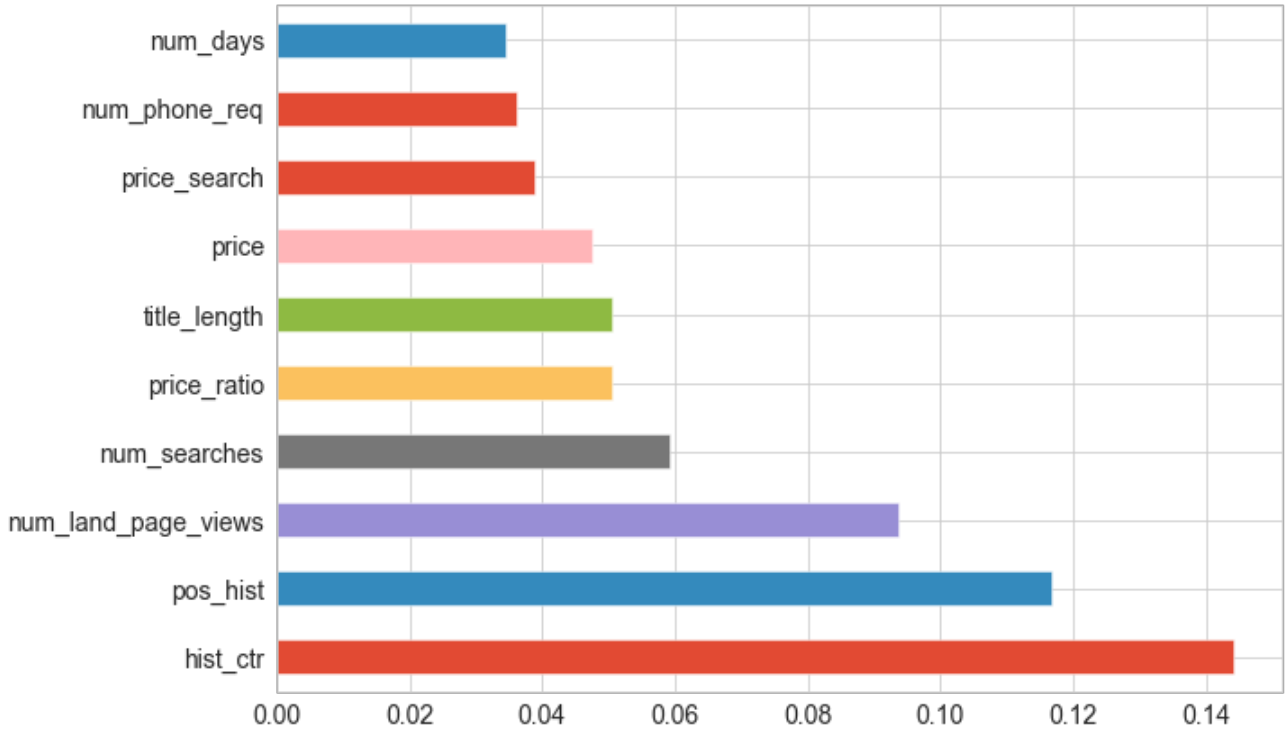
The three models also had very similar $F1$ scores ~0.66 - 0.67 on the test data. Below is the convergence of the learning curve for the XGBoost model.



An image of the confusion matrix for the CATBoost model. Again, they all behaved very much the same. Here you can see that the diagonal(s) appear to have the expected balance. However, there are still a lot of false positives and false negatives. Room for future imrpovement.



Below displays the significance of the features towards the XGBoost model. As expected, the `host_ctr` feature had the largest impact since is is reflective of users prior click through rate.



Future work and Improvements

- There is a lot more feature engineering that could be done with this dataset. For example, the `SearchParams`, `Params` and `title` fields contain language that could have strong predictive power on the result of the model.
- Use of all the data for training and validation may increase the results.
- A useful Flask app would be useful for someone in digital marketing and/or sales in which to guide them towards a sale of arrangment of details in the ad to increase its click power.