Machine Learning Engineer Nanodegree

Capstone Project

Canburak Tümer
02.07.2018

Table of Contents

1. Definition

Project Overview

By completing this project, I would like to improve my Machine Learning skills and new learned abilities. Selecting the problem defined in Problem Statement I aimed to work on a real world problem, and enhance my regression skills since it is a very useful method for predictive analysis on continoues values.

So I decided to choose a competition from Kaggle as project. Kaggle is a website where you can find data mining and machine learning competitions opened by real companies to outsource and crowdsource their predictive analysis needs.

Demand prediction is one of the biggest problems for the retailers. By doing this right, retailers can handle their inventory and stocks also they can do the valuation (or pricing) better by using demand.

There are some research papers and patents about this issue;

You can find these by searching terms *retail demand prediction* or *retail demand forecasting* on Google Scholar:

- Forecasting aggregate retail sales:: a comparison of artificial neural networks and traditional methods; Ilan Alon, Min Qi, Robert J. Sadowski
- The Demand for Dairy Products: Structure, Prediction, and Decomposition; Dale M. Heien, Cathy Roheim Wessells

But Avito's problem is a bit different from the retailers' problem. Since Avito is a marketplace where users put their belonging for sale and other users come and buy these, Avito wants to make advertisements' demand increase so both sellers and buyers would have a better marketplace experience.

To achieve this, first step is to predict the demand value of the ads' current situation. And the predicted demand is somewhat lower than the expected then Avito will provide suggestions to seller to improve his/her ads' demand.

To sum up, this project aims to predict demand of an ad on an online marketplace to help marketplace platform to provide suggestions to sellers for a better ad.

Problem Statement

Avito, a Russia based online classifed advertisement company, wants to solve this demand problem. So they would like to predict demand from the features of advertisements and warn the seller about how to optimize the advertisements' features and get a better demand or better income.

Aimed output of the problem is a probability value between 0 and 1, inclusive and continous. This probaility is interpreted as demand. That's why problem seem to be a regression problem, but I believe it can be also solved by Bayesian approach and using Naive Bayes. The input which will be used is the data of former ads, which are explained in more details in Datasets and Inputs topic of this document.

Metrics

Project output will be scored by using Root Mean Squared Error formula. Formula can be seen below, where y-hat is the calculated probability and y is the real probability for the records. Since the output is a probability value. It would be easier and meaningful to calculate the distance between predicted and real values for evaluation. And to avoid negative and positive values to even-out themselves, we are using the square of errors. And since this task is an Kaggle competition originally, they've selected RMSE as the evaluation metric, that's why I will use it for evaluation.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

1 - Root Mean Squared Error

2. Analysis

Data Exploration

Our dataset consists of four files;

train.csv 307.61MB :

Train data with the following features

- item_id
- user_id
- region
- city
- parent_category_name
- category_name
- param_1
- param_2
- param_3
- title
- description
- price
- item_seq_number
- activation_date
- user_type
- image
- image_top_1
- **deal_probability**

train_active.csv 2.52GB :

Supplemental data from ads that were displayed during the same period as test.csv. Same schema as the train data, minus **deal_probability.**

train_jpg.zip 49.39GB :

Images from the ads in train.csv

periods_train.csv 170.26MB :

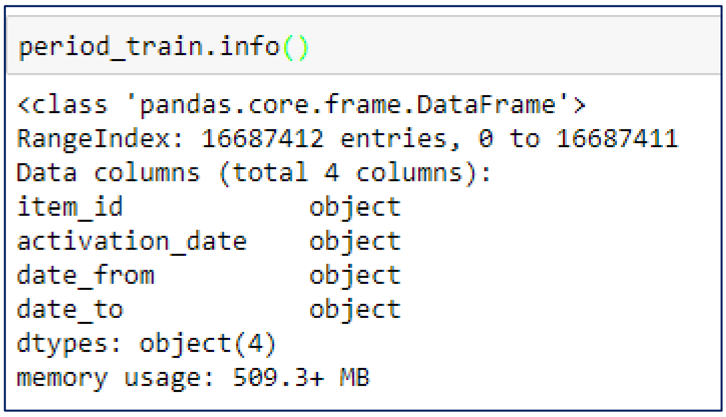
Supplemental data showing the dates when the ads from train_data.csv were activated and when they were displayed. With following features :

- item_id
- activation_date
- date_from
- date_to

All the data can be found on Kaggle by following this link : https://www.kaggle.com/c/avito-demand-prediction/data

Because of hardware limitations, we are only going to use train.csv file and may add some information from periods_train.csv. Unfortunately I am unable to handle image processing and big file processing on my current hardware.

So if we just load up the data, we'll see that data consists of 18+4 features in two different files.



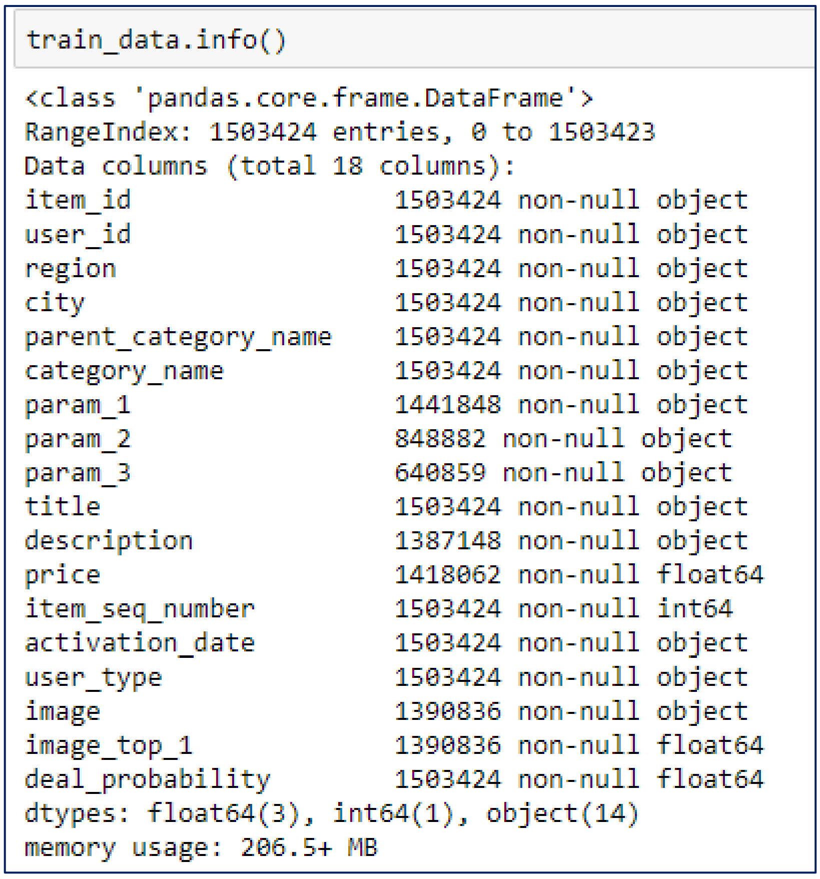
```
period_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16687412 entries, 0 to 16687411
Data columns (total 4 columns):
item_id            object
activation_date    object
date_from          object
date_to            object
dtypes: object(4)
memory usage: 509.3+ MB
```

2 Auxiliary Data Set Summary

As we can see from the metadata, we have three features and an ID in our data set, from this data set I am planning to extract an ad_online_period feature which is date_to minus date_from so I can see for how many days an ad stayed online. Other than that probably I will not use any more feature from this data set.

As this data set was just an auxiliary data set, I do not require to use it but I believe some feature engineering can help me to improve my model score at the end. That's why I will try and extract ad_online_period from this file.



```
train_data.info()
```

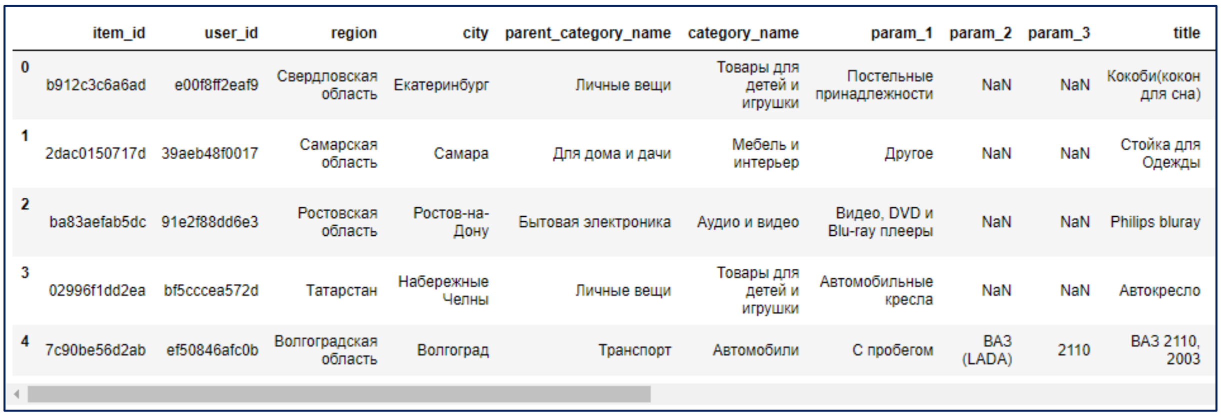
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1503424 entries, 0 to 1503423
Data columns (total 18 columns):
item_id                 1503424 non-null object
user_id                 1503424 non-null object
region                  1503424 non-null object
city                    1503424 non-null object
parent_category_name    1503424 non-null object
category_name           1503424 non-null object
param_1                 1441848 non-null object
param_2                 848882 non-null object
param_3                 640859 non-null object
title                   1503424 non-null object
description             1387148 non-null object
price                   1418062 non-null float64
item_seq_number         1503424 non-null int64
activation_date         1503424 non-null object
user_type               1503424 non-null object
image                   1390836 non-null object
image_top_1             1390836 non-null float64
deal_probability        1503424 non-null float64
dtypes: float64(3), int64(1), object(14)
memory usage: 206.5+ MB
```

3 Main Data Set Summary

Our main data set consists of 18 features, one of them is id (ite_id) and one is the target feature (deal_probability). From these 18 variables I will do some feature engineering, and also drop some of the features. For example, param_X features are optional features to describe ad details, I will create a new feature callaed param_count and calculate a numeric feature from params.

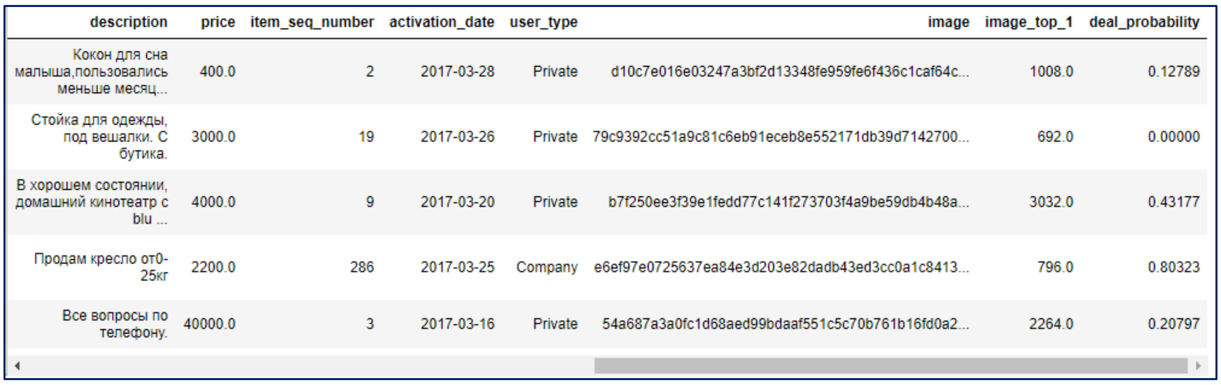
Also I had a problem with the all text fields, unfortunately all text fields are written in Cyrillic alphabet, becuase of this I will use them as they're caterogical data.

A part of data set can be seen below, to have an idea about the data



	item_id	user_id	region	city	parent_category_name	category_name	param_1	param_2	param_3	title
0	b912c3c6a6ad	e00f8ff2eaf9	Свердловская область	Екатеринбург	Личные вещи	Товары для детей и игрушки	Постельные принадлежности	NaN	NaN	Кокоби(кокон для сна)
1	2dac0150717d	39aeb48f0017	Самарская область	Самара	Для дома и дачи	Мебель и интерьер	Другое	NaN	NaN	Стойка для Одежды
2	ba83aefab5dc	91e2f88dd6e3	Ростовская область	Ростов-на-Дону	Бытовая электроника	Аудио и видео	Видео, DVD и Blu-ray плееры	NaN	NaN	Philips bluray
3	02996f1dd2ea	bf5cccea572d	Татарстан	Набережные Челны	Личные вещи	Товары для детей и игрушки	Автомобильные кресла	NaN	NaN	Автокресло
4	7c90be56d2ab	ef50846afc0b	Волгоградская область	Волгоград	Транспорт	Автомобили	С пробегом	ВАЗ (LADA)	2110	ВАЗ 2110, 2003

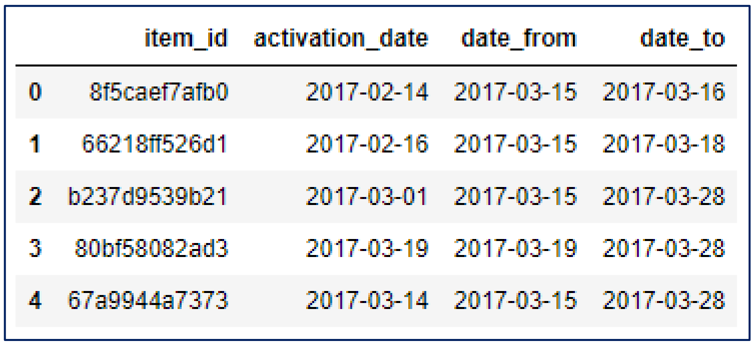
4 Main Data Set Sample part1



description	price	item_seq_number	activation_date	user_type	image	image_top_1	deal_probability
Кокон для сна малыша,пользовались меньше месяц...	400.0	2	2017-03-28	Private	d10c7e016e03247a3bf2d13348fe959fe6f436c1caf64c...	1008.0	0.12789
Стойка для одежды, под вешалки. С бутика.	3000.0	19	2017-03-26	Private	79c9392cc51a9c81c6eb91eceb8e552171db39d7142700...	692.0	0.00000
В хорошем состоянии, домашний кинотеатр с blu ...	4000.0	9	2017-03-20	Private	b7f250ee3f39e1fedd77c141f273703f4a9be59db4b48a...	3032.0	0.43177
Продам кресло от0-25кг	2200.0	286	2017-03-25	Company	e6ef97e0725637ea84e3d203e82dadb43ed3cc0a1c8413...	796.0	0.80323
Все вопросы по телефону.	40000.0	3	2017-03-16	Private	54a687a3a0fc1d68aed99bdaaf551c5c70b761b16fd0a2...	2264.0	0.20797

5 Main Data Set Sample part2

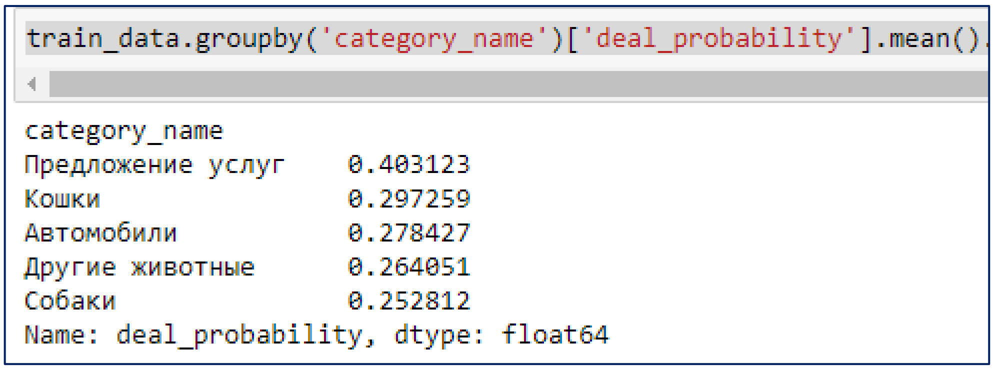
A sample of data from the auxiliary data set is also can be seen below :



	item_id	activation_date	date_from	date_to
0	8f5caef7afb0	2017-02-14	2017-03-15	2017-03-16
1	66218ff526d1	2017-02-16	2017-03-15	2017-03-18
2	b237d9539b21	2017-03-01	2017-03-15	2017-03-28
3	80bf58082ad3	2017-03-19	2017-03-19	2017-03-28
4	67a9944a7373	2017-03-14	2017-03-15	2017-03-28

6 Auziliary Data Set Sample

Also I have calculated some mean deal_probailities depending on different groups to see if any of them is leading to a correlation.

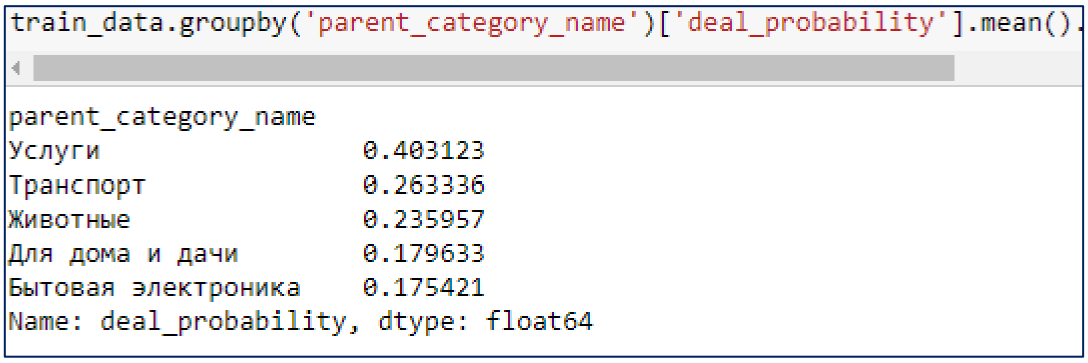


```
train_data.groupby('category_name')['deal_probability'].mean().
```

```
category_name
Предложение услуг    0.403123
Кошки                0.297259
Автомобили           0.278427
Другие животные      0.264051
Собаки               0.252812
Name: deal_probability, dtype: float64
```

7 Target Means Grouped By Category

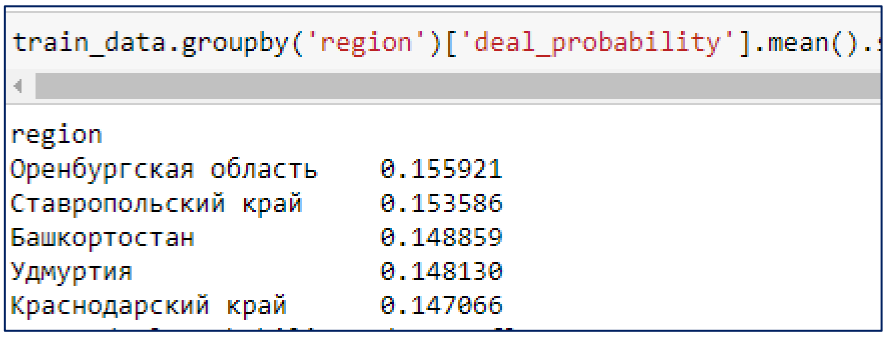
Category seems to have a valid correlation with the target variable, but I will also try and visualize the distribution of deal_probability values in categories. And we will have a better insight on next chapter (Exploratory Visualization)



```
train_data.groupby('parent_category_name')['deal_probability'].mean().

parent_category_name
Услуги                  0.403123
Транспорт               0.263336
Животные                0.235957
Для дома и дачи         0.179633
Бытовая электроника     0.175421
Name: deal_probability, dtype: float64
```

8 Target Means Grouped By Parent Category

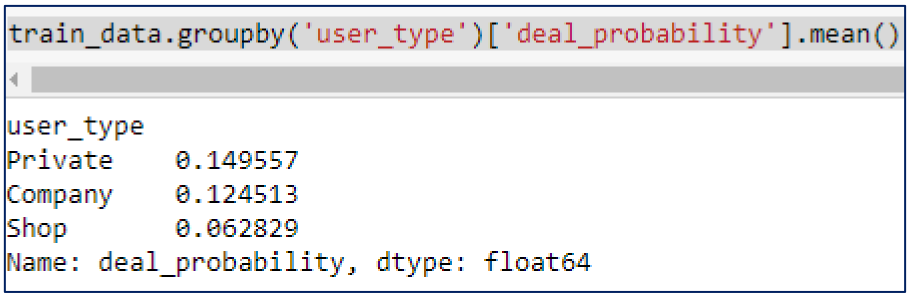
Parent category name also seems like to have a good correlation on demand. Some parent categories really have a higher mean than the others.



```
train_data.groupby('region')['deal_probability'].mean().

region
Оренбургская область    0.155921
Ставропольский край     0.153586
Башкортостан            0.148859
Удмуртия                0.148130
Краснодарский край      0.147066
```

9 Target Means Grouped By Region

Depending on my home country I believed that region can also help us to find a correlation. But it seems region does not play a powerful role in the Avito’s data.



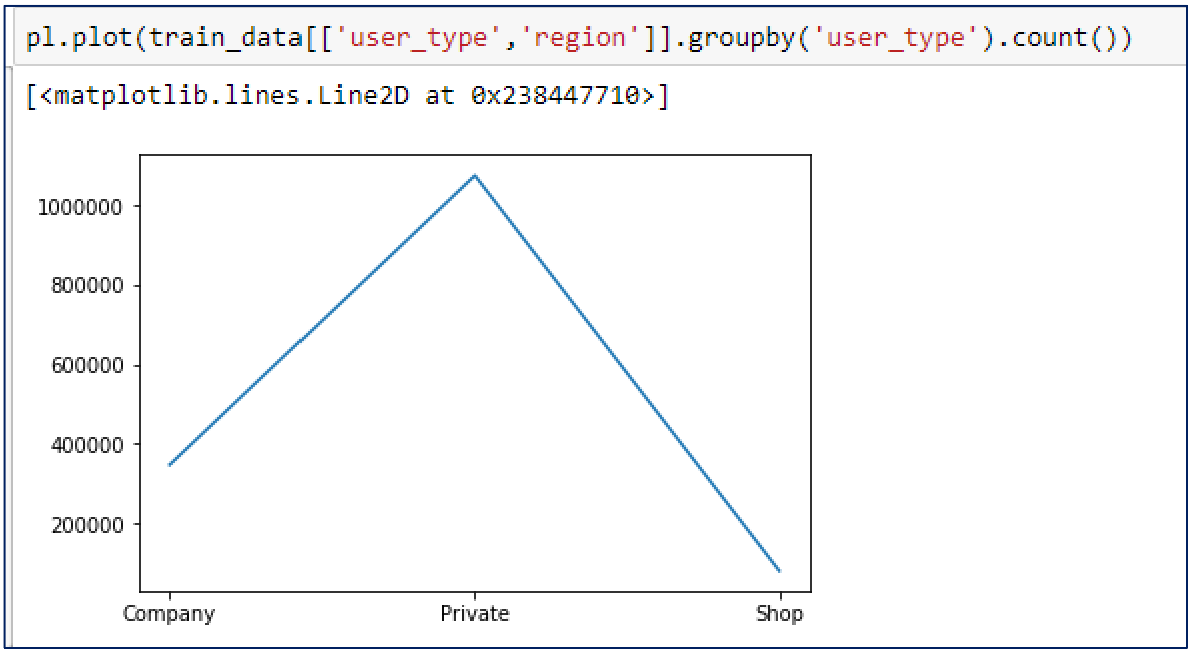
```
train_data.groupby('user_type')['deal_probability'].mean()

user_type
Private     0.149557
Company     0.124513
Shop        0.062829
Name: deal_probability, dtype: float64
```

10 Target Means Grouped By User Type

User type seems to be a powerful candidate as an input, shops have a great disadvantage compared to private and company ads.

Exploratory Visualization

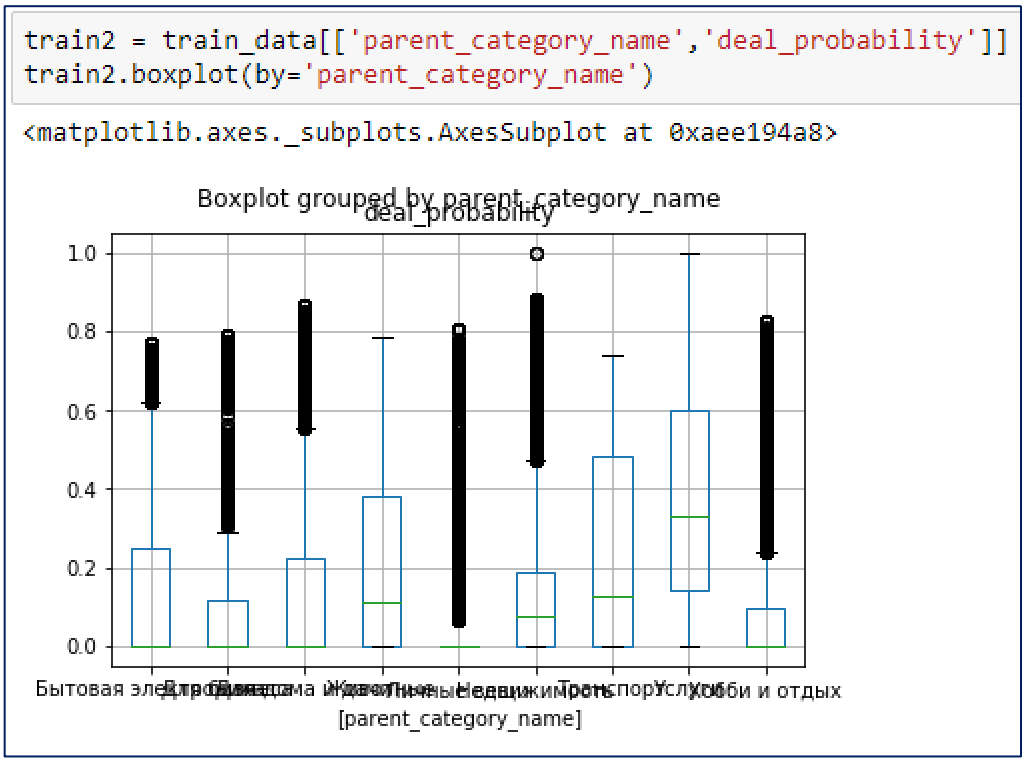
First of all I wonder about user_type counts, it may give us an insight about marketplace’s user base.



```
pl.plot(train_data[['user_type','region']].groupby('user_type').count())
```

```
[<matplotlib.lines.Line2D at 0x238447710>]
```

11 Ad Counts by User Type

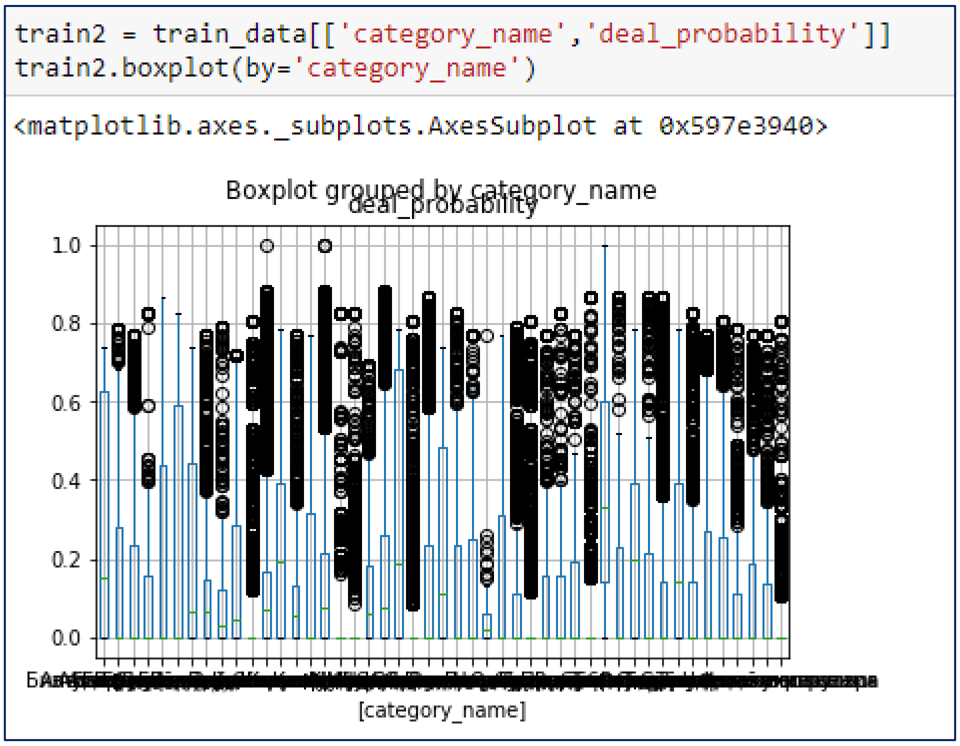


```
train2 = train_data[['parent_category_name','deal_probability']]
train2.boxplot(by='parent_category_name')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xaee194a8>
```

12 Deal Probability Distribution Over Parent Categories

As we can see, deal probability values have influenced by the parent categories, that means I need to consider parent category as a feature in my input set.

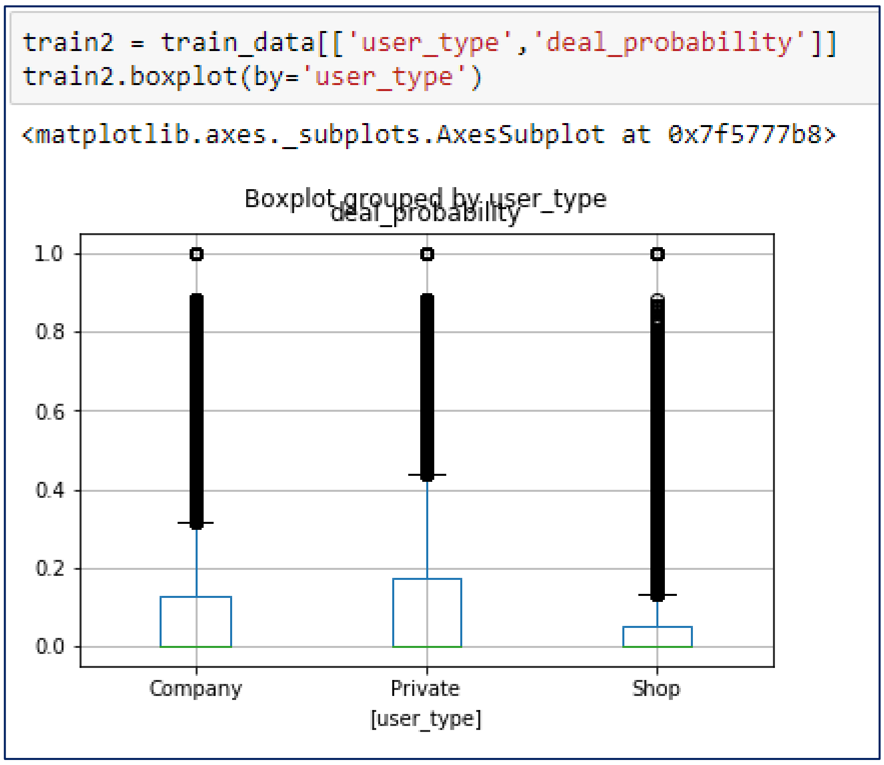


```
train2 = train_data[['category_name','deal_probability']]
train2.boxplot(by='category_name')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x597e3940>
```

13 Deal Probability Distribution Over Categories

Although category also seem to have an effect on probability disribution, because of there are too many categories, I will not add it in the first input set, but I can add it later on to see if it improves the score.



```
train2 = train_data[['user_type','deal_probability']]
train2.boxplot(by='user_type')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5777b8>
```

14 Deal Probabilty Distribution Over User Types

User types seem to have a direct impact on deal probability I will use this feature in the input set.

Algorithms and Techniques

The algorithms I decided to try on the data is Naive Bayes and Regression (first linear one then I may try some more complex regression algorithms).

The reason I am choosing these methods are I need to predict a continuous variable which is between 0 and 1. And Naive Bayes is doing the exact same thing, although it is known as a classification algorithm it calculates a probability value. So Naive Bayes will produce me a number betwen zero and one.

Regression also used for predict continuous variables. In a multi dimensional environment like our problem, linear regression is a fast and safe approach to calculate target variable.

For hyper parameters, I will use a grid search approach to find the best solutions. Also for both approaches I will try different feature sets to see the relation between features and the result.

Benchmark

Benchmark used by Kaggle (https://www.kaggle.com/c/avito-demand-prediction/leaderboard) is Naive Random Forest and its score is 0.2468 which means I aim to beat this score by using two algorithms explained above.

3. Methodology

Data Preprocessing

As I have partly mentioned in the data exploration part, in data preprocessing part I will drop some features from the data set but also I will engineer some new features from existing ones.

The features I am going to use are :

- item_id : the id of the ad
- parent_category_name : the parent category of the ad
- category_name : the sub category of the ad
- price : price of the products being sold
- user_type : user type of the ad owner
- image_top_1 : a score given to ad image by Avito
- deal_probability : target variable

The features I am going to engineer are :

- param_count : not every ad has a param_1, param_2, param_3 value so count of the filled variables may help us to decide ad deal_probability
- ad_online_period : how many days the ad stayed online, calculated from auxiliary data set

Also I will build a single dataset file with the features above. So it will be easier to maintain and keep data.

For these steps, I will use pandas and pandasql libraries of python.

Then while the model training and scoring phase I will divide data into two sets of 70% training and 30% test set.

Data preprocess is actually the hardest part of an machine learning and data science problem. Data provided is usually dirty, have missing values and quality problems. Also not every algorithm handles every type of data, for example I will need to transform categorical data into numerical data for use in Linear Regression Model.

So what I plan to do is, first try to generate a linear model with only numerical values, and see the score of model, then transform data into more acceptable input set for the linear regression model.

For missing values in numericals, I will fill them with a zero value.

Then I have exported the data as a CSV file. Running the code in jupyter notebook named “2 Data Preprocessing” will produce the same files for you to run on further notebooks.

Implementation

As in every machine learning project, this project also did not go as it’s planned.

First step was the data preprocessing which is planned in former topic, let’s deep dive into what really happened in implementation phase.

After completing the Exploratory Data Analysis phase, I’ve decided to drop some feature and generate some new feature from given set. Dropping the features went like a breeze, but the generating features was not successful. Pandasql library was unable to handle cyrillic alphabet, so I’ve decided to use join method from pandas, but after joining two files, I’ve seen there is not any match on item_id values, so I decided not to use the auxiliary data set.

For the first model, I decided to move on with only numerical values, since linear regression can not handle the categorical values. So I generated a new file with, price, image_top_1 and deal_probability features.

In jupyter notebook named “3 Linear Regression” I’ve splitted data into test and train with 30% - 70% ratio. Then I trained a linear regression model for the data.

Looking to the resources made me think twice about using the Naive Bayes, since Naive Bayes is a classification model and sci-kit learn only gives the class as output. It will not help me about getting the probability value. That’s why I decided not to train a Naive Bayes model.

Instead I decided to use other type of regressions to see if they fit better than the Linear model. Which can be found in jupyter notebook “4 Other Regressions”

This new notebook covers, Ridge Regression and Bayesian Regression.

Refinement

My first steps in refinement is always adding or dropping features to the model. Then I also play around with hyperparameters of the algorithm

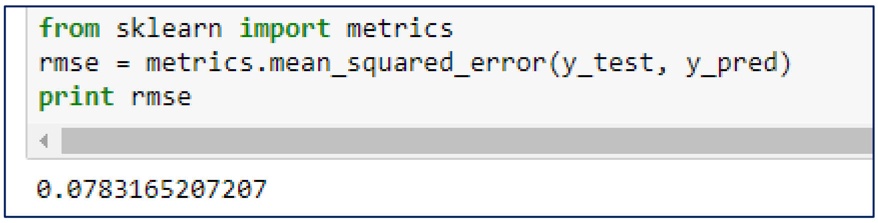
To the first linear regression model, I’ve added item_seq_number to see if it helps with enhancing the score. You may see the results in the following section about this refinement.

For the ridge regression, there are hyperparameters like max_iter and alpha. I've used grid search to find a good fit for this algorithm. For this purpose I used a method from sci-kit learn called RidgeCV with alpha values of 0.1, 0.5, 0.8, 1.5, 5, 10

4. Results

Model Evaluation and Validation

First model trained was a Linear Regression model and its output RMSE value is 0.0783 which is a pretty good value compared to the benchmark of 0.2468 from naive random forest value taken from Kaggle.

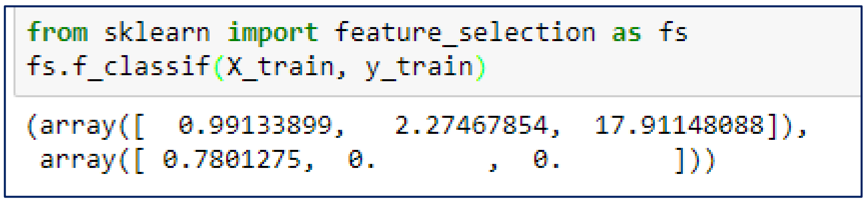


```
from sklearn import metrics
rmse = metrics.mean_squared_error(y_test, y_pred)
print rmse
```

```
0.0783165207207
```

15 Result from Linear Regression

Also I've checked ANOVA table from the model, and its p-values seems to be reasonable.

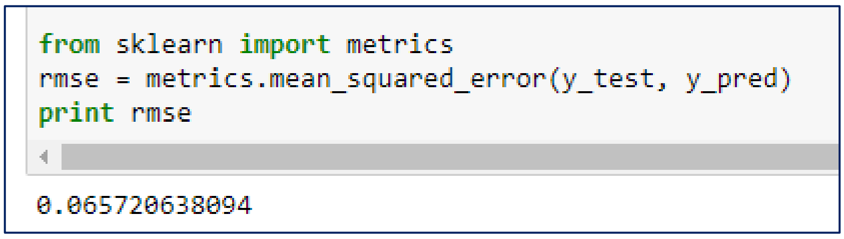


```
from sklearn import feature_selection as fs
fs.f_classif(X_train, y_train)
```

```
(array([  0.99133899,   2.27467854,  17.91148088]),
 array([ 0.7801275,  0.        ,  0.        ]))
```

16 ANOVA table for Linear Regression

Adding item_seq_number feature to the Linear Regression above increased our score to 0.0657.

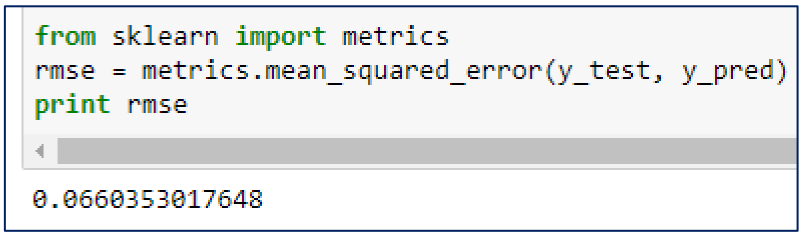


```
from sklearn import metrics
rmse = metrics.mean_squared_error(y_test, y_pred)
print rmse
```

```
0.065720638094
```

17 Result from Linear Regression with item_seq_number

As mentioned in Methodology I also implemented Ridge and Bayesian regression methods to see the score of these methods. Ridge Regression with only numerical values provide a good score of 0.066.

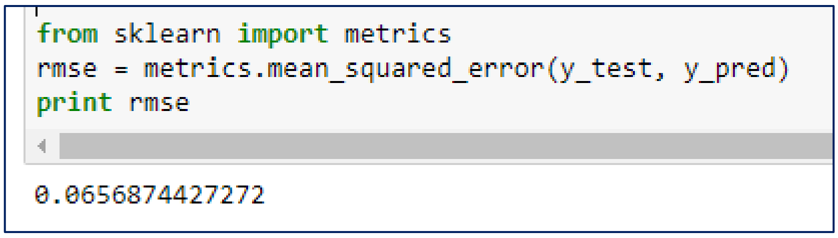


```
from sklearn import metrics
rmse = metrics.mean_squared_error(y_test, y_pred)
print rmse
```

```
0.0660353017648
```

18 Result from Ridge Regression

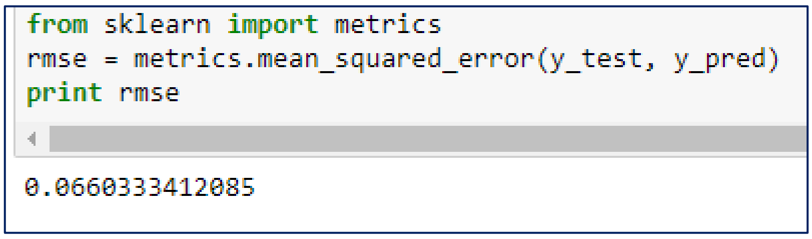
If we also add the item_seq_number feature to the Ridge Regression our score improves to the 0.0656 and this is the best score we get out of our data so far.



```
from sklearn import metrics
rmse = metrics.mean_squared_error(y_test, y_pred)
print rmse

0.0656874427272
```

19 Result from Ridge Regression with item_seq_number

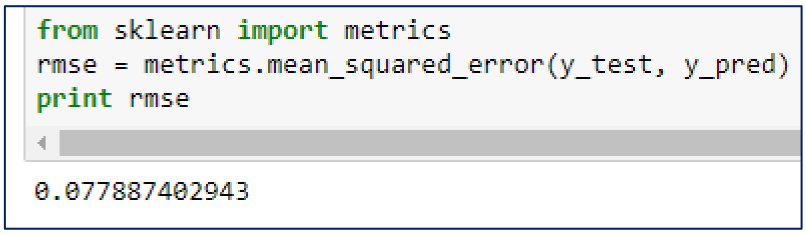
In the Bayesian regression, our model scores a 0.066 as the Ridge Regression did. They have a small difference in the fifth or sixth figure after the decimal point, so we can assume they are as good as each other.



```
from sklearn import metrics
rmse = metrics.mean_squared_error(y_test, y_pred)
print rmse

0.0660333412085
```

20 Result from Bayesian Regression

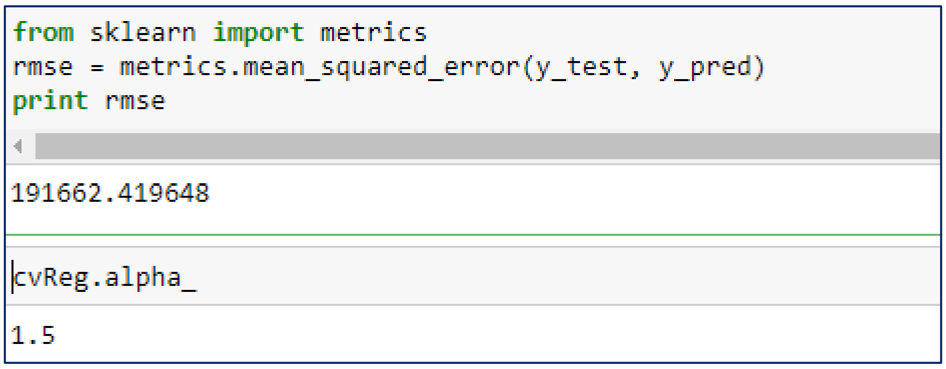
Adding item_seq_number to Bayesian Regression caused our score to decrease to 0.0778



```
from sklearn import metrics
rmse = metrics.mean_squared_error(y_test, y_pred)
print rmse

0.077887402943
```

21 Result from Bayesian Regression with item_seq_number

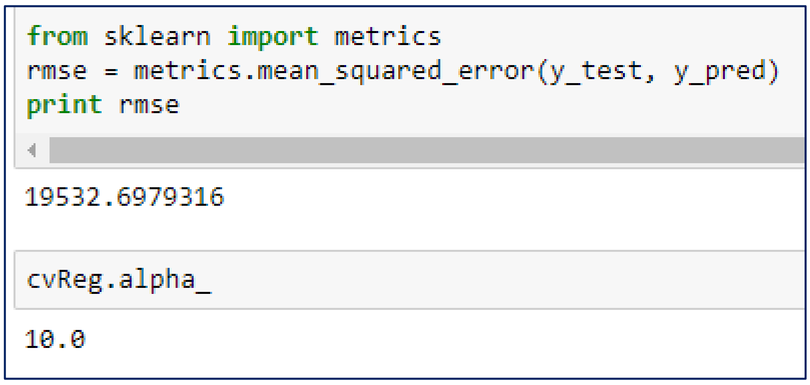
After all these, I've also trained another Ridge Regression model with built-in cross validation function and its results as follows.



```
from sklearn import metrics
rmse = metrics.mean_squared_error(y_test, y_pred)
print rmse

191662.419648

cvReg.alpha_

1.5
```

22 Ridge Regression with Cross Validation

From cross validation, our algorithm selected 1.5 alpha value as the best solution, but it has a mean squared error of 191,662.4196 and this seems a bit extreme to me. I believe there is something going on under the hood of sci-kit learn.



```
from sklearn import metrics
rmse = metrics.mean_squared_error(y_test, y_pred)
print rmse
```

```
19532.6979316
```

```
cvReg.alpha_
```

```
10.0
```

23 Ridge Regression with Cross Validation and item_seq_number

From cross validation with item_seq_number added, our algorithm selected 10 alpha value as the best solution, but it has a mean squared error of 19,532.6979 and this seems a bit extreme to me. I believe there is something going on under the hood of sci-kit learn.

After all trial and error session, the best solution for our problem seems to be the Ridge Regression with item_seq_number added. Since it has the lowest MSE value, which is 0.0656

Justification

So our best model produces a nice score of 0.0656 which is way better than the benchmark of 0.2468.

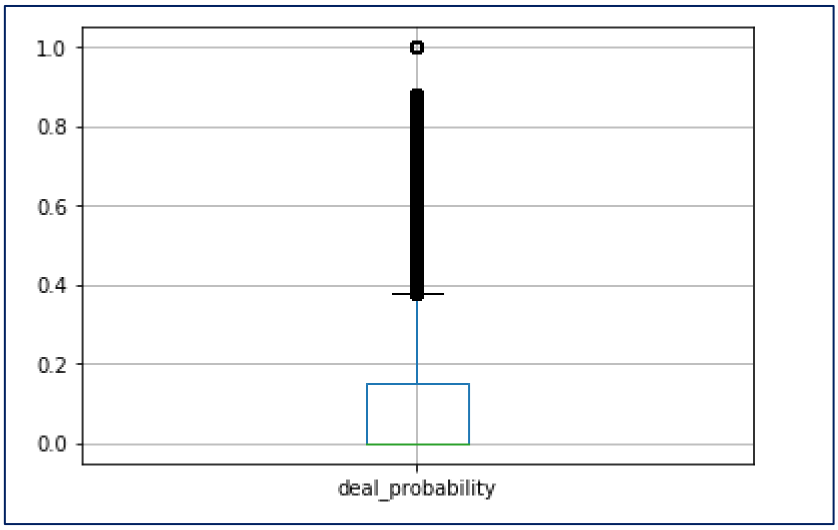
When I decided to work on this problem it seem clear to me to use a regression algorithm to get the result, since the problem is predicting a continuous number between zero and one. And from these options I have started with the simplest one the popular and loved Linear Regression although it performed well enough, it would be a shame to not trying other algorithms for a project like that. So I moved on to more complex regression algorithms.

Luckily I have found some better approaches in other regression algorithms and select the Ridge Regression as the best performing.

5. Conclusion

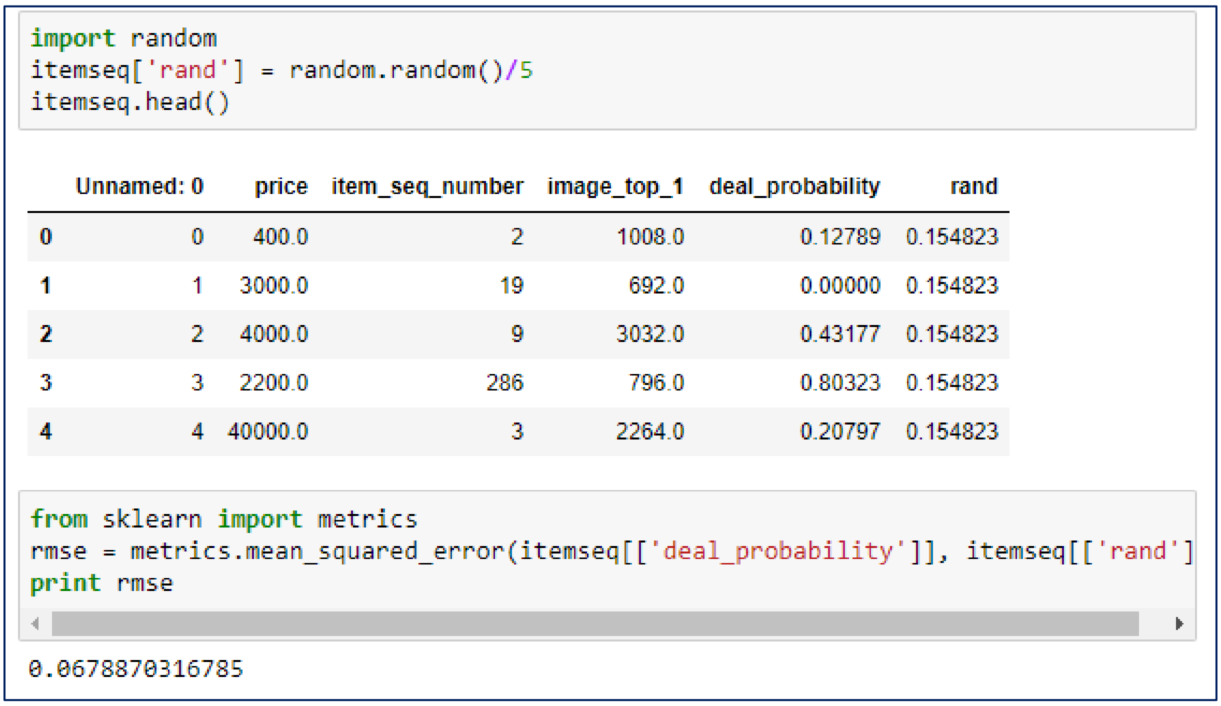
Free-Form Visualization

It's always good idea for me to visualize the target variable first. So I did visualize it.



24 Deal Probability boxplot

We can see that most of the ads' demands stay below 0.2, and all the values above 0.4 is drawn like they are outliers. Although this is interesting, it could be a pretty normal for an online marketplace and also we do not know how this probability value calculated in the first step. But looking through the deal probability, an algorithm which predicts some random numbers between 0 and 0.2 can also achieve a fair score in this problem. Well I actually did it to see the results which also you can see below.



```
import random
itemseq['rand'] = random.random()/5
itemseq.head()
```

	Unnamed: 0	price	item_seq_number	image_top_1	deal_probability	rand
0	0	400.0	2	1008.0	0.12789	0.154823
1	1	3000.0	19	692.0	0.00000	0.154823
2	2	4000.0	9	3032.0	0.43177	0.154823
3	3	2200.0	286	796.0	0.80323	0.154823
4	4	40000.0	3	2264.0	0.20797	0.154823

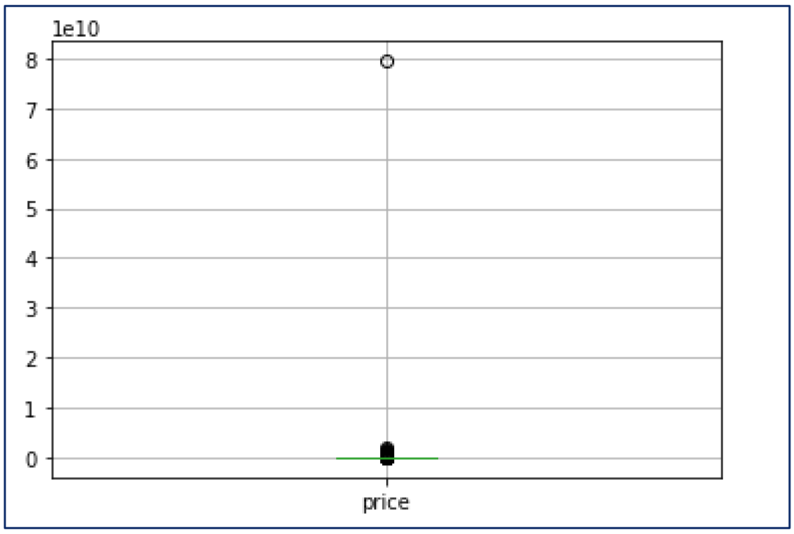
```
from sklearn import metrics
rmse = metrics.mean_squared_error(itemseq[['deal_probability']], itemseq[['rand']
print rmse
```

```
0.0678870316785
```

25 Score of random probability

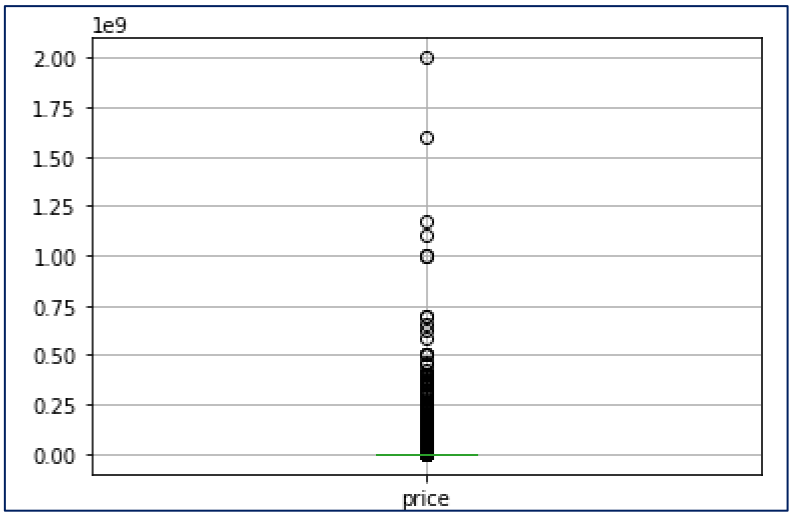
Our model is still better than the random probability, which makes me happy.

I also decided to visualize the price value since it is a highly correlated feature on demand prediction.



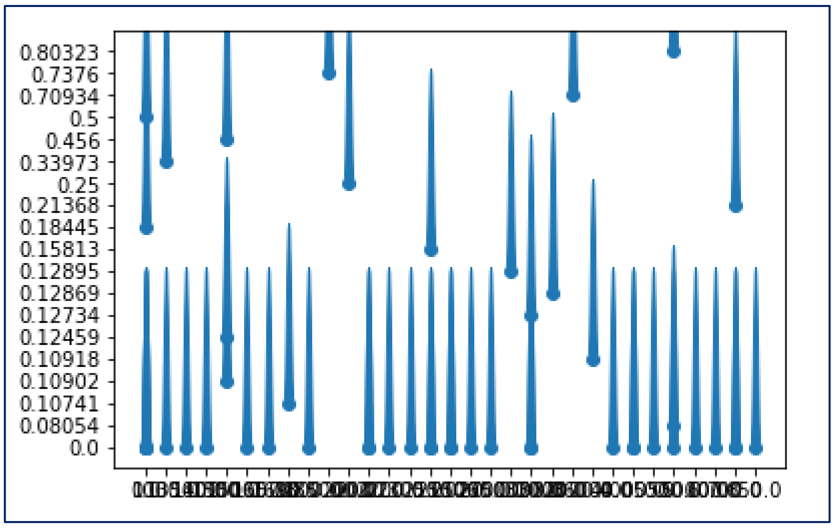
26 Price unhandled

When I visualized the price, I saw there is an outlier value, which is aroun 80 billion, so I thought this should be a data quality problem and I discarded this data point from my set.



27 Price handled

This is the new graph I got. Although the one with the price of 2 billion seems like an outlier, it seems more reasonable to me than the 80 billion so I left the data like that, but as you can see most of the data points are below 500 million, even we have a mean much less than a million, you may see the green line in the graph aroun 0.00 line. The actual price mean is 298,725 to be honest.



28 Price and Deal correlation

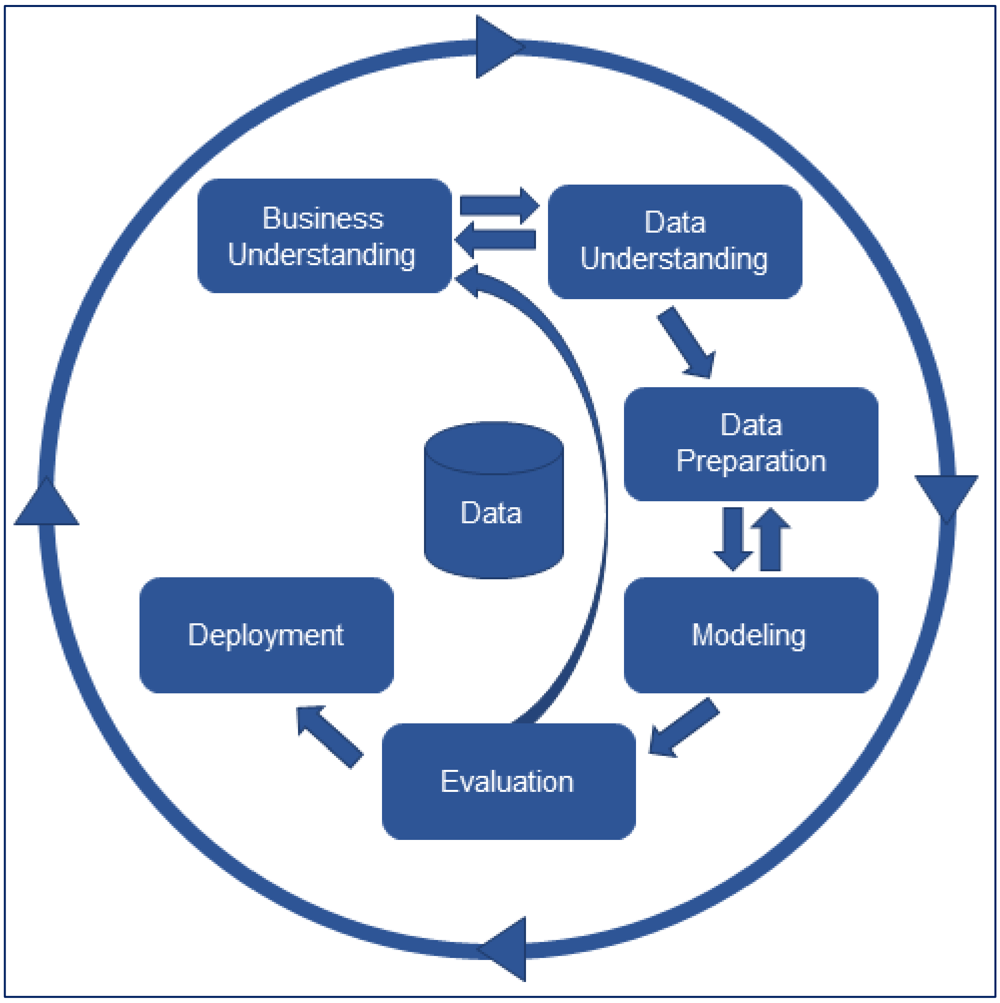
Then for the last thing I would lie to see if I can catch any visual correlation between price and deal probability, I was expecting the cheaper things get better deal probabilities, (X axis is deal_probability in the graphic) but I saw that price do not seem to have a great affect on the deal probability. Then it came to me that this data has lots of categories, and doing this visual is not the best thing to do. And I decided that actually this graph do not tell us something, but I leave it here to reflect my effort and my understanding of the domain.

Reflection

Avito's Demand Prediction project was an interesting project and it also was totally out of my domain.

In my industry, telecommunications, we do not have problems like this one. So predicting the effective features, wrangling the data and getting the results were interesting tasks for me. I believe I still have many things to learn about the domain and the needs of the domain.

It is also pretty hard to accomplish a machine learning project end-to-end. I tried to utilize CRISP-DM methodology which consists of the following loop.



29 CRISP-DM Loop

CRISP-DM tells us to first start with understanding the business needs; for this I've done some research about online marketplaces and how do they work and what they need. Then comes the data understanding, to be honest this should be done before selecting the capstone project. In real life it is not possible to select your project but in MLND Capstone I may find a project with a better data. Data understanding phase was tiring for Avito's challenge, it has cyrillic alphabet, it has images, it has some cryptic quality features, it has free-form text. Then I started to prepare data for the models in my mind. This task was also hard to finish because of the reasons I've stated above. Modelling and model evaluation phases are the easiest parts of the project. I have trained different type of models with different features and different hyperparameters and all of the outputs are compared to benchmark and each other.

I also would like to do some personal reflection. What would I do better if I do the project again? Probably I would use my time wiser. Because of conferences and local holidays, I have spent so much time without coding between proposal and completion of the project.

Improvement

Because of the time dimension data have, it could be a better choice to implement some time series algorithms like ARIMA but due to deadline constraint I have it was not possible to try it.

Also using some translation APIs some fields like description and category could be translated into latin words then using methods like tf-idf or bag of words approach a correlation may be caught between words in ad and ad demand.

Kaggle competitors have used neural networks for training. It may be another approach for this problem to utilize some neural network or deep learning solutions to get the results. But most of other competitors did not complete any feature engineering because of the power of deep learning method. And I believe as a MLND student I also need to learn about feature engineering and I am happy with my method selection.

References

http://scikit-learn.org/stable/documentation.html

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

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

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