Dariusz Wapiński
31.03.2019

Capstone Project

New York city taxi fare prediction

this project is based on kaggle competition:
https://www.kaggle.com/c/new-york-city-taxi-fare-prediction

if I use any tips, helps etc from another projects/websites/code and so on which helps me to solve this problem a proper information and link to author will be included in text and code description.

I. Project definition

• Project overview

Have you ever had taxi ride? I'm pretty sure you do. In our modern world we have a lot things to do everyday. We dont have time to thinking about many things. I need to go to work, meet parents and so on. In busy life we take cab and go. Transport issues in big city is very important. Because nobody likes traffic jams etc. The taxi companies usually thinking about cost optimization. Cost optimization in transportation leads to less pollution etc. There are many taxi companies which provides smartphone apps for customer. Customer is able to check price before ride. It's our problem we have to provide model to predicting taxi fare total amount based on a historical data. This kind of problem is well documented and many scientist works on it. For example http://www.vivekchoksi.com/papers/taxi_pickups.pdf. In this scientific paper an author use machine learning for predicting pickups so they will be able to optimize where taxis should be located. Another scientific paper about our domain is http://cs229.stanford.edu/proj2016/report/AntoniadesFadaviFobaAmonJuniorNewYorkCityCabPricing-report.pdf. The Author try to predict fare and duration of taxi trip

• Project statement

Previous section contains general description. Now is the time for more details. Our goal is build mechanism for predicting taxi fare amount. We have a data contains many real life taxi trips in the New York City. This is our inputs: pickup date, pickup location, dropout date, dropout location, number of passengers, sample fare amount. Our desired output is total fare amount for new data (not known before). We have to found function of input variables which give us fare amount. We provide input data and have output data (fare amount) and we use an algorithm to learn mapping function from the input to the output. This function for new data give us unknown before taxi fare. In other words this is supervised learning. Because we want to approximate new output (real number) this is regression type problem. So we build supervised regression model. Data is provided by Google. Detailed data description is in section about data engineering/data analyzing.
This is solution path:

1. Data exploration.
   - total amount of our data
   - do we need full data set or can we use data range.

2. Feature selection.
– which feature is relevant
– maybe we should group features or use specific data ranges.
3. Select model for supervised regression task.
- SVM or DNN
4. Train model, make predictions.
5. Do some benchmarks. Look at section project metrics.
6. Make decision if model is good enough. Compare metrics, tune model parameters if needed and go to step 5.

- **Project metrics**

Evaluation metric for this project will be the root mean-square error (RMSE). RMSE measures difference between the predictions of a model and the ground truth.

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

where y_i is the ith observation and $\hat{y}_i$ is the prediction for that observation.

During model training we get current rmse values for every iteration. So we could draw the plot of rmse changes vs number of iterations.

II. Analysis

- **Data exploration**

Data description. Data for those project is provided by google.

- **train.csv** - Input features and target fare_amount values for the training set (about 55M rows).
- **test.csv** - Input features for the test set (about 10K rows). It is not necessary I could split train.csv to train test data, because we have a lot of data. We don't want to submit our result to kaggle.
- **sample_submission.csv** - a sample submission file in the correct format (columns key and fare_amount). This file 'predicts' fare_amount to be $11.35 for all rows, which is the mean fare_amount from the training set. This is not necessary for us because we dont want submit our score to kaggle

Descripion of datafields:

ID:

- **key** - Unique *string* identifying each row in both the training and test sets

Features:

- **pickup_datetime**- *timestamp* value indicating when the taxi ride started.
- **pickup_longitude** - *float* for longitude coordinate of where the taxi ride started.
- **pickup_latitude** - *float* for latitude coordinate of where the taxi ride started.
- **dropoff_longitude** - *float* for longitude coordinate of where the taxi ride ended.
- **dropoff_latitude** - *float* for latitude coordinate of where the taxi ride ended.
- **passenger_count** - *integer* indicating the number of passengers in the taxi ride.

Target

- **fare_amount** - `float` dollar amount of the cost of the taxi ride. This value is only in the training set; this is what you are predicting in the test set and it is required in your submission CSV.

- **Exploration visualization**

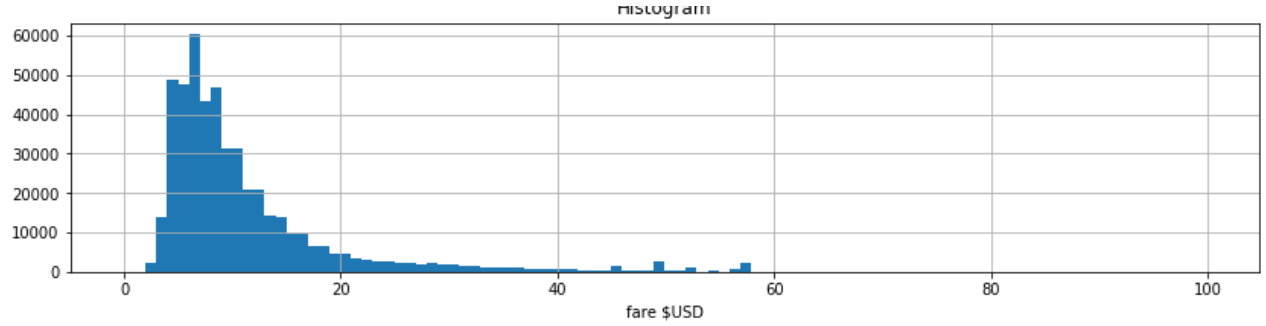
Training dataset contains 55M rows this is a lot of data. I'm trying to analyze and build model for all data. But it is very difficult at home on a Personal computer. It is possible to use some cloud services but I would prefer doing my computations at home. So I random choose 500k data rows and remove NaN etc from dataset. Next data analyze will be based on this subset. All data analyze/exploration is in **data_exploration.ipynb**

Data statistics.

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
count	499995.000000	499995.000000	499995.000000	499995.000000	499995.000000	499995.000000
mean	11.358182	-72.520091	39.920350	-72.522435	39.916526	1.683445
std	9.916069	11.856446	8.073318	11.797362	7.391002	1.307391
min	-44.900000	-2986.242495	-3116.285383	-3383.296608	-2559.748913	0.000000
25%	6.000000	-73.992047	40.734916	-73.991382	40.734057	1.000000
50%	8.500000	-73.981785	40.752670	-73.980126	40.753152	1.000000
75%	12.500000	-73.967117	40.767076	-73.963572	40.768135	2.000000
max	500.000000	2140.601160	1703.092772	40.851027	404.616667	6.000000

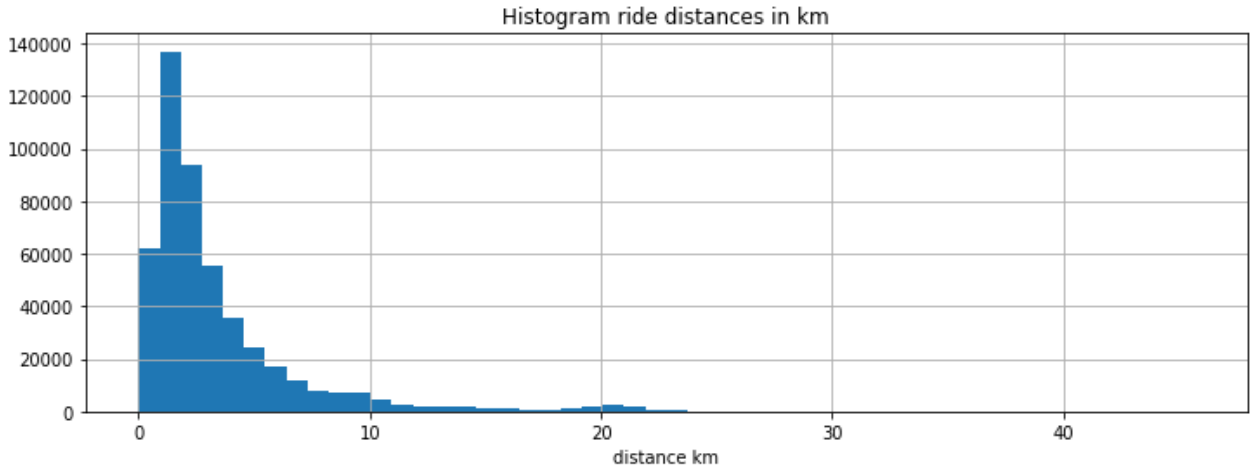
After that I remove from dataset fare_amount < 0 and passenger_count = 0.

Number of rides to fare amount.



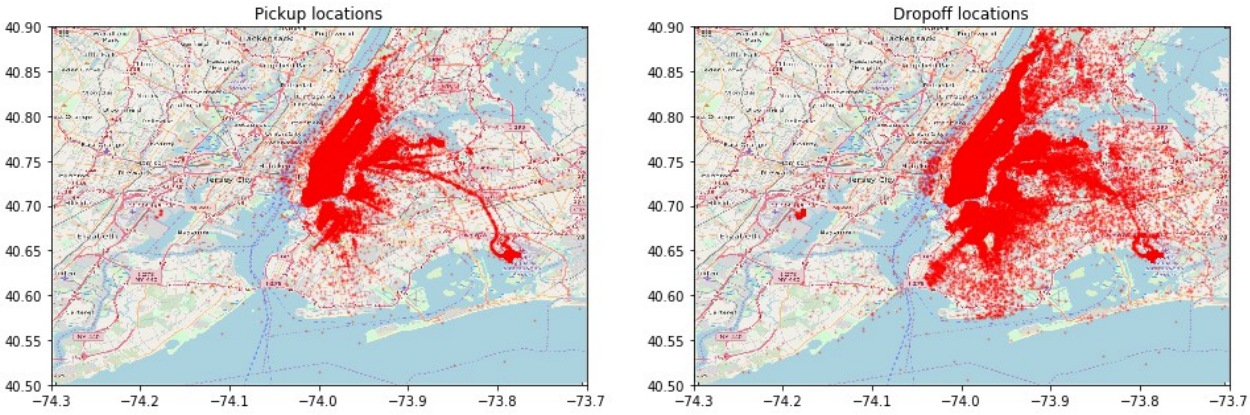
As you can see fare amount < 20 is most popular in our dataset.

Next lets see ride distance. We using haversine formula to calculate distance from pickup, dropoff coordinates.



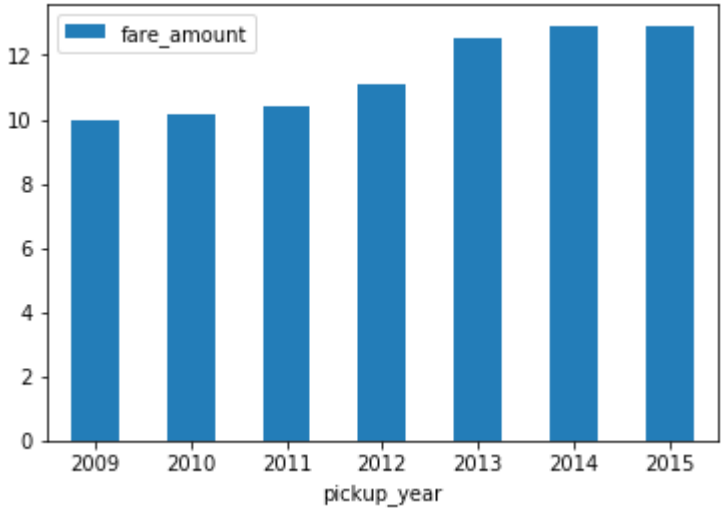
As we can see most of the rides is less than 10km.

We work on geographical data (coordinates) so it will be very usefull to draw pickup, dropoff coordinates on map. New york city coordinates is between 40.50 – 40.90 latitude and -73,7 – 74.3 longitude



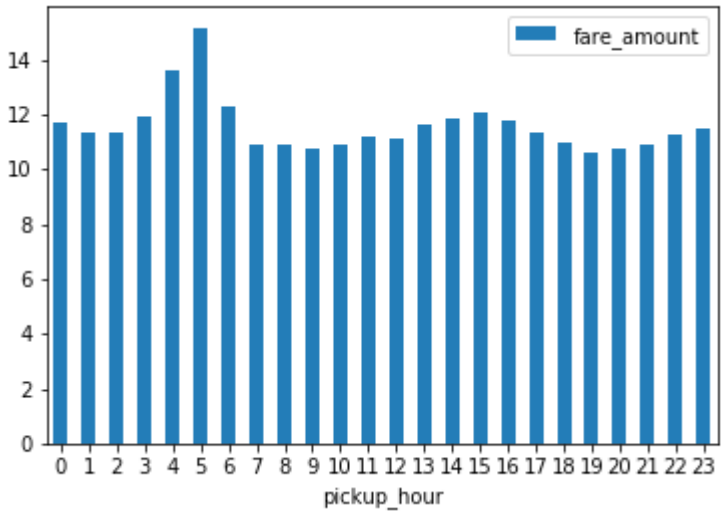
pickup_timestamp is in form: 2009-06-15 17:26:00+00:00 it is not good for analyze so we need to split it for separate columns: year, month, day, hour, minute. I need this for analyze time impact to fare_amount.

Now I plot correlation between fare price to year.



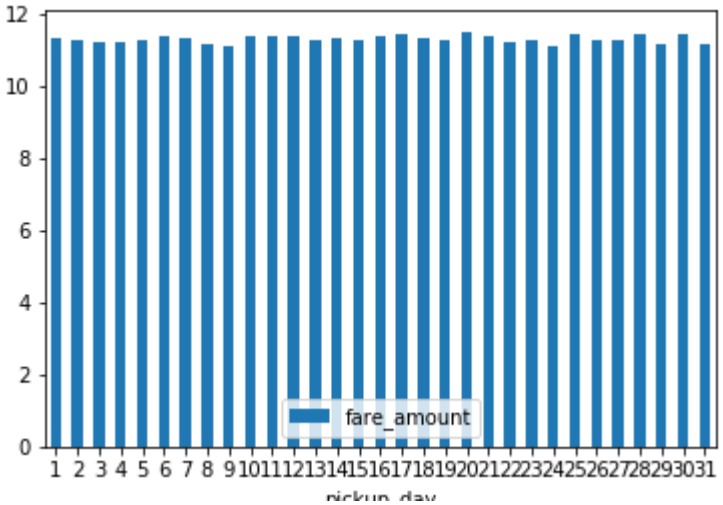
We see that price avg price increase in year.

Now we check fare_amount to hour of pickup.



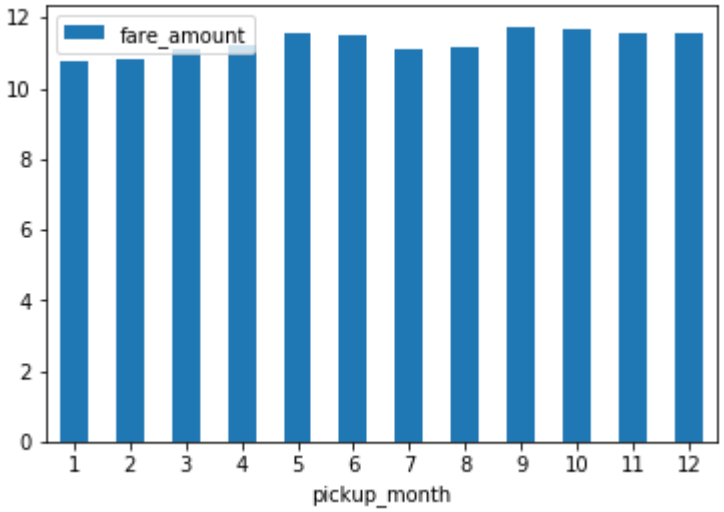
As we can see price depends on hour.

Now we check correlation between price and day of month.



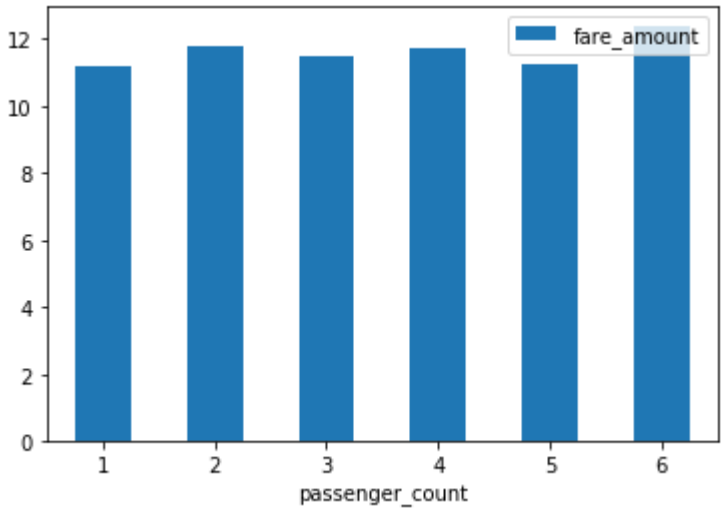
I think that we could skip this feature in our analyze.

Next we check price to month correlation.



I think we also could skip this feature in our analyze.

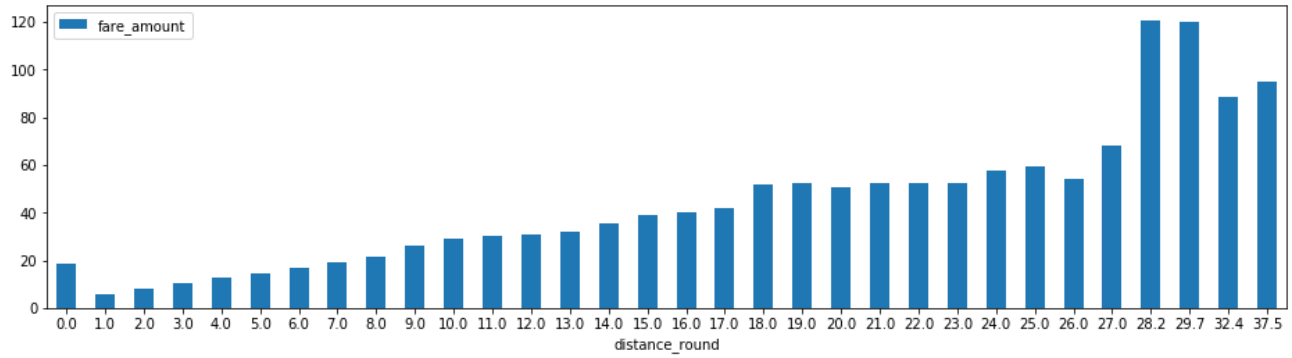
Next we check price to number of passengers.



It is really strange because I think than an particular taxi ride cost the same. It doesnt matter how many passengers is in car. Cost is the same so passengers could divide it.

I dont take this feature into consideration.

Last correlation is the price to ride distance. I use a rounded distance and every 5-th sample for plot readability



It is absolutely normal, long distance means higher prices.

- **Algorithms and Techniques**

This is regression type problem so we should use some regression model. As I write before I use Support vector regression. General information about SVR could be find here: https://medium.com/coinmonks/support-vector-regression-or-svr-8eb3acf6d0ff

and http://kernelsvm.tripod.com/

Our project is in python so I use SVR from Sci-Kit learn: https://scikit-learn.org/stable/modules/svm.html#svm-regression

Examples of approximation plot using svr https://scikit-learn.org/stable/auto_examples/svm/plot_svm_regression.html

SVR parameters:

kernel – linear, polynomial, rbf (our analyze is non-linear so we choose rbf – kernel)

gamma : float, optional (default='auto')
Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

Current default is 'auto' which uses 1 / n_features, if `gamma='scale'` is passed then it uses 1 / (n_features * X.var()) as value of gamma. The current default of gamma, 'auto', will change to 'scale' in version 0.22. 'auto_deprecated', a deprecated version of 'auto' is used as a default indicating that no explicit value of gamma was passed.

C : float, optional (default=1.0) Penalty parameter C of the error term.

epsilon : float, optional (default=0.1) Epsilon in the epsilon-SVR model. It specifies the epsilon-tube within which no penalty is associated in the training loss function with points predicted within a distance epsilon from the actual value.

All parameters: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

- **Model benchmark**

We can train selected model by using the train data set and test with the test data set. After training we measure difference between the predictions and the truth. We will measure RMSE error and we trying to minimalize this error.

III. Methodology

- **Data preprocessing**

Data exploration section include data exploration and preprocessing. Removing outliners – points not included in New York City area. Pickup timestamp was splitted into separate columns 'year', 'month', 'day', 'hour'. From coordinates calculated distance using haversine formula, column 'distance'. I use those features to match correlation between fare_amount and data.

- **Implementation**

Dataset from google 55M rows, I choose only 500k for analyze and preprocessing. SVR needs a lot of compute power so I do computations for limited dataset.

1. Split data_reduced.csv to train, test subsets.

2. train model with parameters:

```
kernel='rbf'
gamma=57
C=50
 epsilon=0.2
```

Those parameters was set experimental and tune during training with small dataset. At start I use gridsearch technique but this is very inefficient if you choose many parameters to test. For example

```
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV
#svr kernel rbf
K = 5
parameters = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4, 0.1],
                    'C': [10, 50, 100]}]

svr = SVR()
clf = GridSearchCV(svr, parameters, cv = K)

clf.fit(X_train, y_train)
clf.best_params_
```

- **Refinement**

As I write before, some initial parameters was chosen experimental. I have also tried find best parameters using GridSearchCV but it takes a long time. For getting better performance we should use more data, and tune parameters.

IV. Results

- **Model Evaluation and validation**

SVR training takes a long time on my PC. So I do some computations on two dataset,

1. training set 20000 rows , validation set 2000 rows RMSE = 4.543806777722515

2 training set 200000 rows, validation set 20000 RMSE = 3.6899951784228455 but computations takes about 16hours (on my home PC) and I dont try more data.

V. Conclusion

We passed through project.

1. Declare what we want to do. Which problem do we try to solve.

2. We identified type of problem (regression)

3. Analyze data, trying to find important features. Important means correlated with fare_amount.

4. We choose algorithm for this kind of problem. I choosed Support Vector Regressor.

5. experimental (and with little help of gridsearch for start) we tune parameters and check performance metrics for our model.

Further improvments could be done by using more data. Unfortunately I don't have time for doing this project using for example neural networks maybe it will be performs better.