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**

Now we need to declare our metrics. Because we try to predict fare amount this is regression. In general we will measure some kind of difference between predicted value and the ground truth. The ground truth would be known fare amount from test set. Regression give us values, those values differ from true values. We need measure average of that differences. Popular metrics in machine learning regression problem is:

RMSE definition:

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

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

Example calculation:
predicted values: 5.5$ 11$ 20$
ground truth: 4.8$ 10.5$ 22$

RMSE = sqrt(1/3*((5.5 – 4.8)^2+(11-10)^2+(20-22)^2))= sqrt(1.83)=1.35

Example calculation:
predicted values: 5.5$ 11$ 20$
ground truth: 4.8$ 10.5$ 30$
RMSE = sqrt(1/3*(5.5-4.8)^2+(11-10)^2+(20-30)^2)) = sqrt(33.8)=5.81

because we square the difference between predicted and truth value that big errors has bigger impact for rmse. RMSE is good as loss function because it is easy and efficiently to calculate and differentiable.

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.

- **Exploratory visualization**

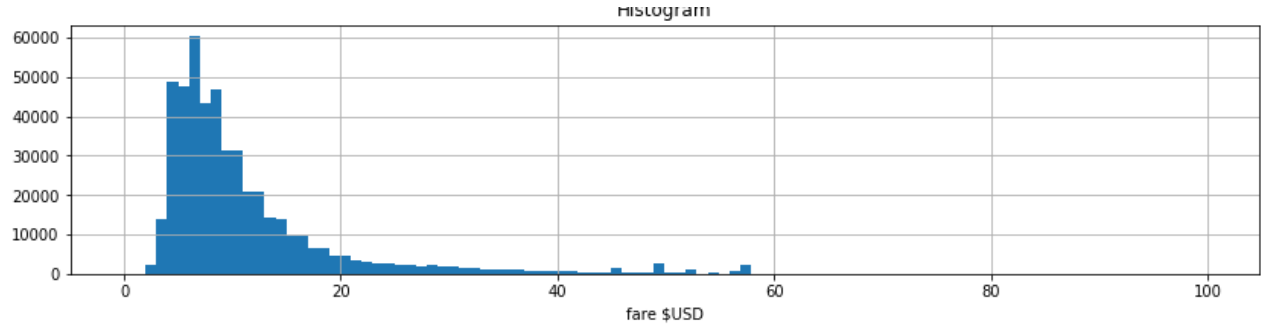
Training dataset contains 55M rows this is a lot of data. I'm trying to analyze and build model for all data. But it is very difficult at home on a Personal computer. It is possible to use some cloud services but I would prefer doing my computations at home. So I random choose 500k data rows and remove 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

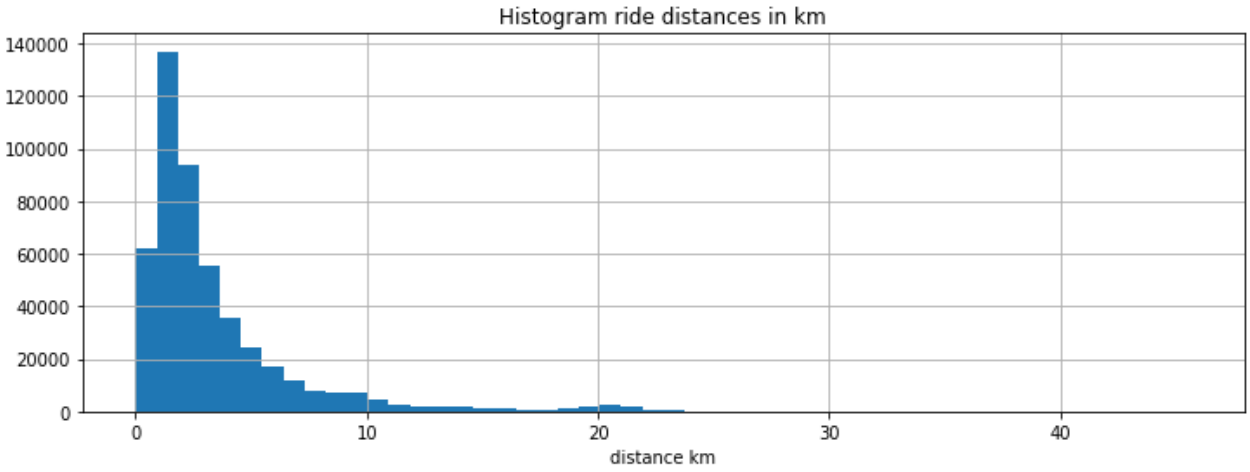
After that I remove from dataset fare_amount < 0 and passenger_count = 0.

Number of rides to fare amount.



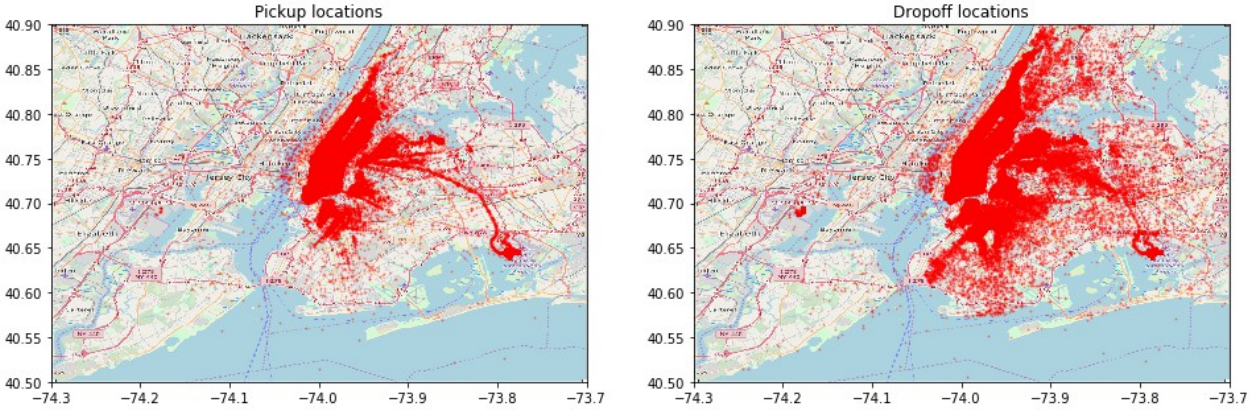
As you can see fare amount < 20 is most popular in our dataset.

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



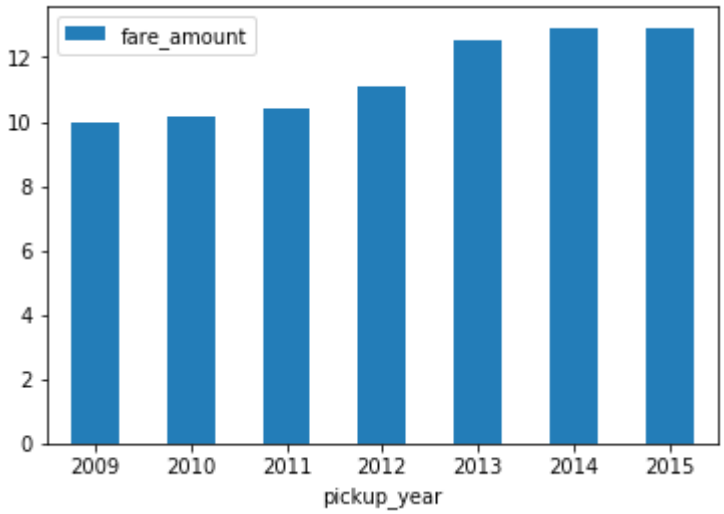
As we can see most of the rides is less than 10km.

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



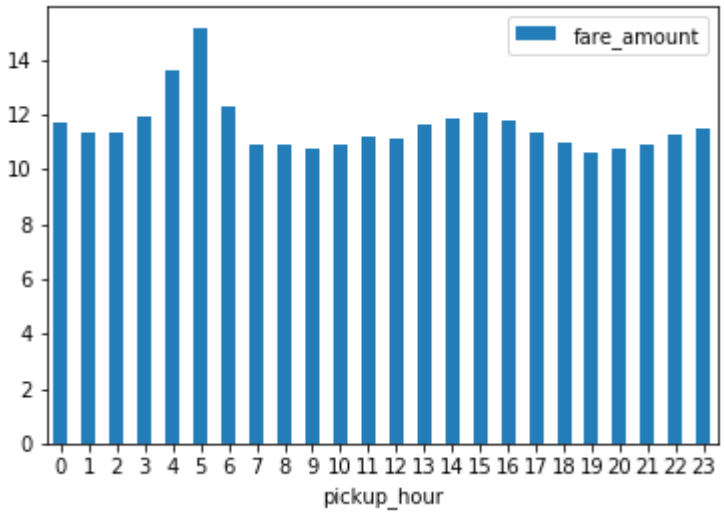
pickup_timestamp is in form: 2009-06-15 17:26:00+00:00 it is not good for analyze so we need to split it for separate columns: year, month, day, hour, minute. I need this for analyze time impact to fare_amount.

Now I plot correlation between fare price to year.



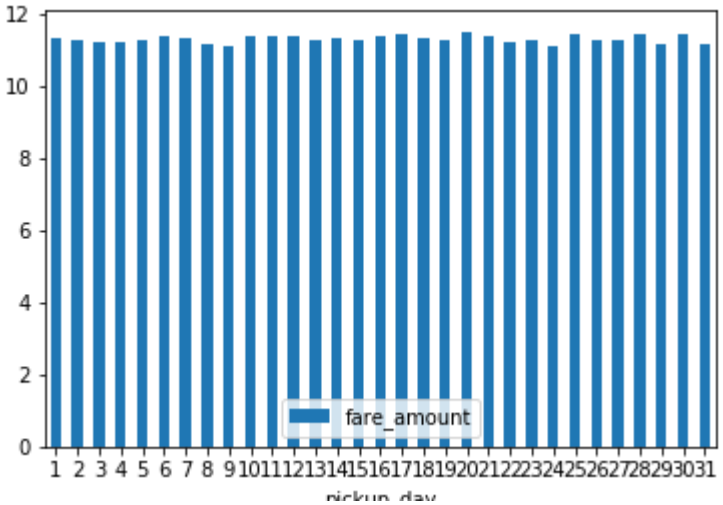
We see that price avg price increase in year.

Now we check fare_amount to hour of pickup.



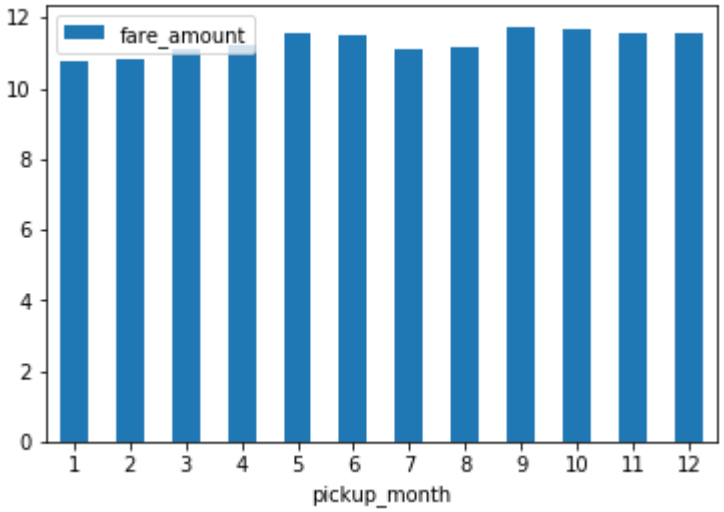
As we can see price depends on hour.

Now we check correlation between price and day of month.



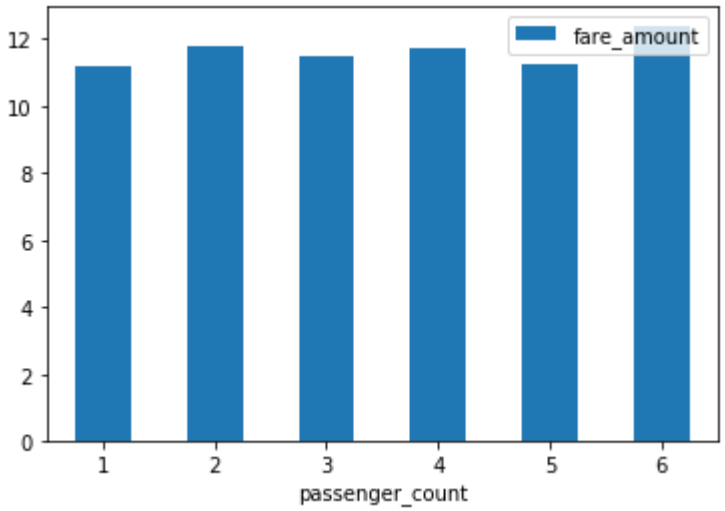
I think that we could skip this feature in our analyze.

Next we check price to month correlation.



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

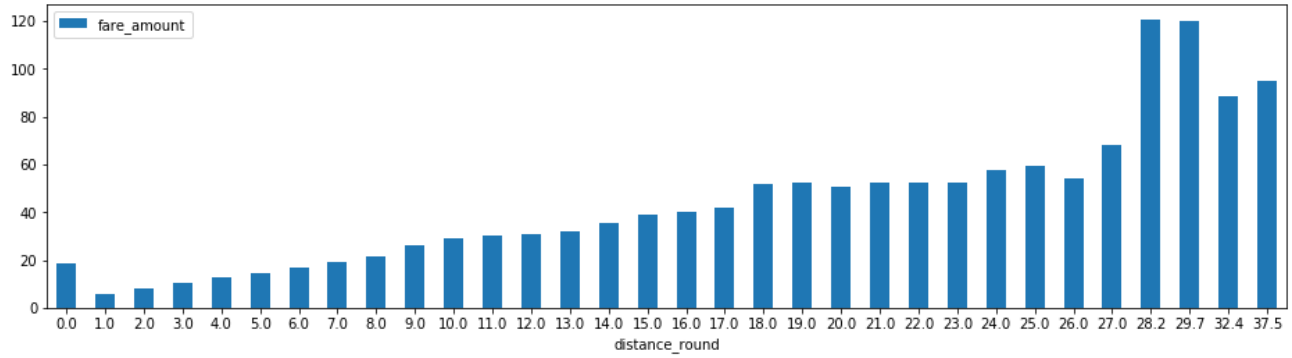
Next we check price to number of passengers.



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.

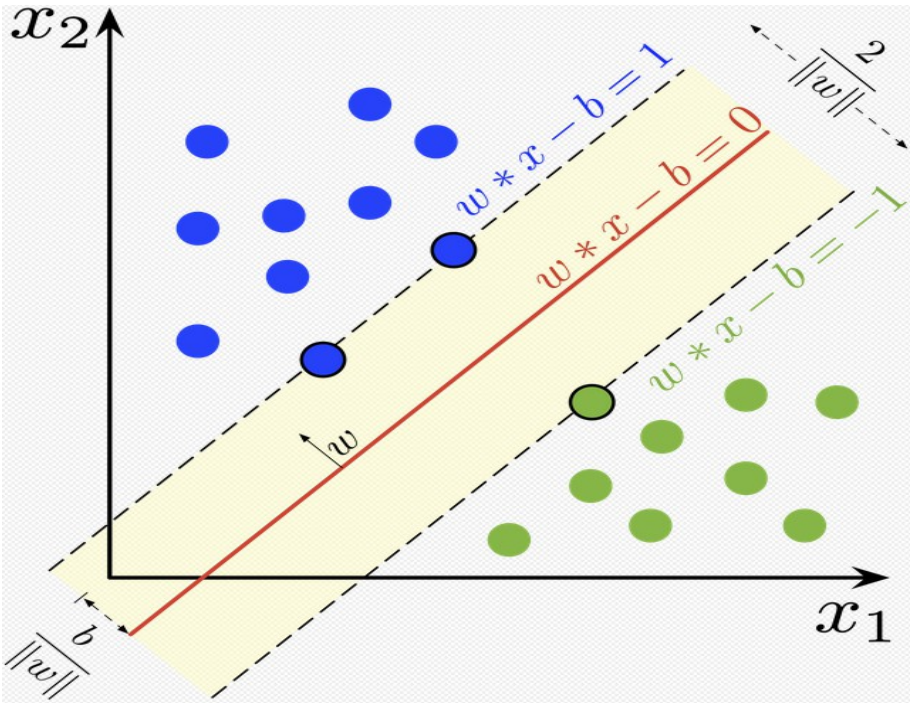
Last correlation is the price to ride distance. I use a rounded distance and every 5-th sample for plot readability



It is absolutely normal, long distance means higher prices.

- **Algorithms and Techniques**

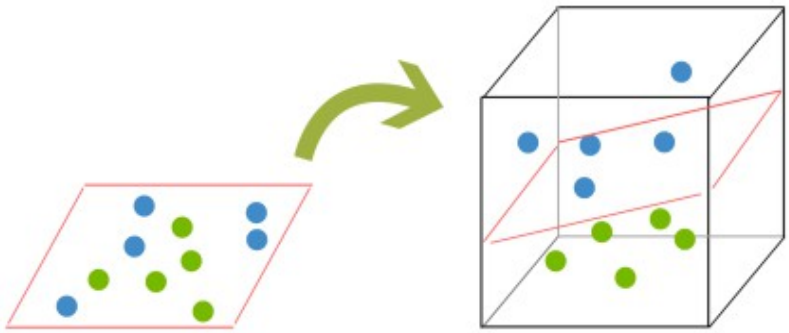
SVM support vector machine algorithm. SVR is commonly used in a classification problem but also is good for regression. I'm trying to explain in simple word how SVM (SVR) works. Using SVR we are trying to separate two class of data. This separation is made by hyperplane. Simple example in two dimensional space. Our hyperplane is the red line.



By Larhmam - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=73710028

Support vectors are the points nearest to the hyperplane. The distance from hyperplane to the nearest point called margin. Our goal is to maximize margin to any nearest points in training dataset.

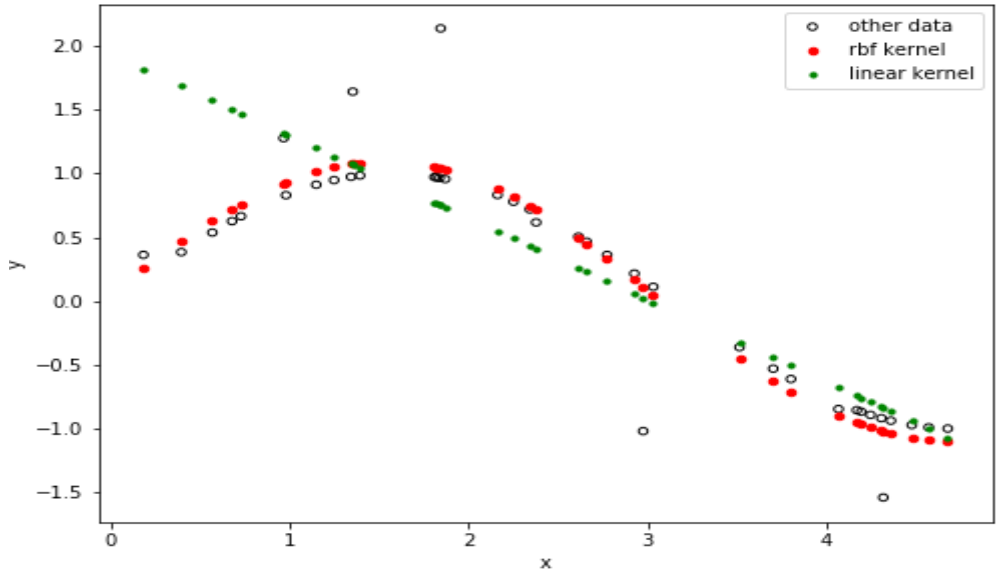
If data is not fine separable we need to go to the higher dimensions.



Picture source: https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html

We do some kind of lift blue balls over green. Now we can separate this two classes. This kind of dimensional expand is called kernelling. We do 'kernel trick'.

But in our project we are trying to use SVR for regression. Example 2D fitting using different type of kernels.



As you can see rbf kernel is more suitable for non-linear curves.

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.

tol : float, optional (default=1e-3) We don't change it Tolerance for stopping criterion.

cache_size : float, optional Specify the size of the kernel cache (in MB).

The other parameters is default or is not usable in 'RBF' kernel

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

- **Model benchmark**

We compare our model with some benchmark model. Thanks to the:
https://www.kaggle.com/dster/nyc-taxi-fare-starter-kernel-simple-linear-model

This is simple least square regression model. Working source is in simple_linear_benchmark_model.ipynb. Final score is 7.65 so we can compare our result to this score.

III. Methodology

- **Data preprocessing**

Data exploration section include data exploration and preprocessing. This is detailed description:
Removing NaN's – clear dataset from NaN's
Removing outliners – points out of the New York City area. New York City coordinates.

LEFT_LONGITUDE = -74.3, RIGHT_LONGITUDE= -73.7, TOP_LATITUDE = 40.9,BOTTOM_LATITUDE = 40.5
Removing taxi fare with price less or equal 0
Removing trips starts and ends at the same point

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.

Based on Data exploration section we choose this features for analyze:
'pickup_hour','pickup_year','pickup_latitude', 'pickup_longitude', 'dropoff_latitude', 'dropoff_longitude'

- **Implementation**

Implementation of SVM (SVR) for this project. My development enviroment was Anacoda for windows. Conda version : 4.6.7, python 3.6, scikit 0.20.2. Any code was developed in Jypyter Notebook. Because we need interaction with code, visualization etc.

Project development stages:

1. Data exploration and preprocessing. This stage is included in data_exploration.ipynb notebook.

2. Model development and training.

The biggest issue at the first stage is how to efficiently load and explore this big dataset (55M rows). Please note that everything was doing on a cheap home PC. Finally I have decided to get only 500k rows. Limited size also have big impact for computations time. We need from theory that SVR needs a lot of compute power. Generally speaking data engineering is crucial for machine learning. Training model for a long time on bad data could be frustrating. Output from first stage is data set located in data/data_preprocessed.csv

Second stage was the model development. I have not implemented SVR by myself because this will take a long time. I have decided to use SVR implementation from scikit-learn python package.
This stage steps:

1. load data.
2. Because we need train, test subsets we use this method:

   ```
   from sklearn.model_selection import train_test_split
   X_train, X_test, y_train, y_test = \
   train_test_split( data[['pickup_hour','pickup_year','pickup_latitude', 'pickup_longitude', \
   'dropoff_latitude', 'dropoff_longitude']], fare_amount, train_size=20000, test_size=2000, \
   random_state=42)
   ```

   Feature selection was justified in section visual data exploration. Train size has the bigest impact for computation time so I have to decided for a really small subset for test and experimets with model parameters

3. This is our model, because we use SVR implementation. This is very user friendly because you don't need to look under the hood.
   from sklearn.svm import SVR
   single_svr = SVR(kernel='rbf', gamma=57, C=50, epsilon=0.2, verbose=True)
   single_svr.fit(X_train, y_train)
4. Train model. single_svr.fit() with particular parameters.
   Training process could take a long time so I use limited size for trend observation how parameters change impact on metrics.
5. Calculate performance metrics.
   f*rom sklearn.metrics import mean_squared_error*
   rmse = np.sqrt(mean_squared_error(y_test, y_pred))
   print(rmse)

The biggest isssue at stage 2 was how to start? Which values should I use etc.

- **Refinement**

I have started using grid search for found optimal hyperparameters:

```
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV
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_
```

but it takes a lot of time. So I manually choose hyper parameters and training model with 10k samples

gamma=20, C=100, epsilon = 0.1
gamma = 30, c=100, epsilon = 0.1 and so on.
After many calculation and RMSE observation I choose values:
gamma=57
C=50
epsilon=0.2

IV. Results

- **Model Evaluation and validation**

During training and experiments I can assume that this parameters should be optimal:
kernel='rbf'
gamma=57
C=50
epsilon=0.2

Example output for two training dataset, that differ in size:

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

2 training set 200000 rows, validation set 20000 rows RMSE = 3.6899951784228455 but computations takes about 16hours (on my home PC) and I don't try more data.

Model is good when it performs good for new data, different data etc. Checking model robustness for another data for example. We use model learned at point 1.

```
#robustness test
#validation our model for different data
X_test_2 = data.sample(n=20000, random_state=33)
X_test_3 = data.sample(n=30000, random_state=85)
X_test_2_bias = X_test_2
y_test_1 = X_test_2['fare_amount']
y_test_3 = X_test_3['fare_amount']

y_pred_2 = single_svr.predict(X_test_2[['pickup_hour','pickup_year','pickup_latitude',
                'pickup_longitude', 'dropoff_latitude', 'dropoff_longitude']])
y_pred_3 = single_svr.predict(X_test_3[['pickup_hour','pickup_year','pickup_latitude',
                'pickup_longitude', 'dropoff_latitude', 'dropoff_longitude']])
```

rmse_2 = 5.491993372962419

rmse_3 = 5.448585805711815

as we can see our model for sampled different data with limited size perform worse but In my opinion it depends on training data size. We should have limited trust for this model.

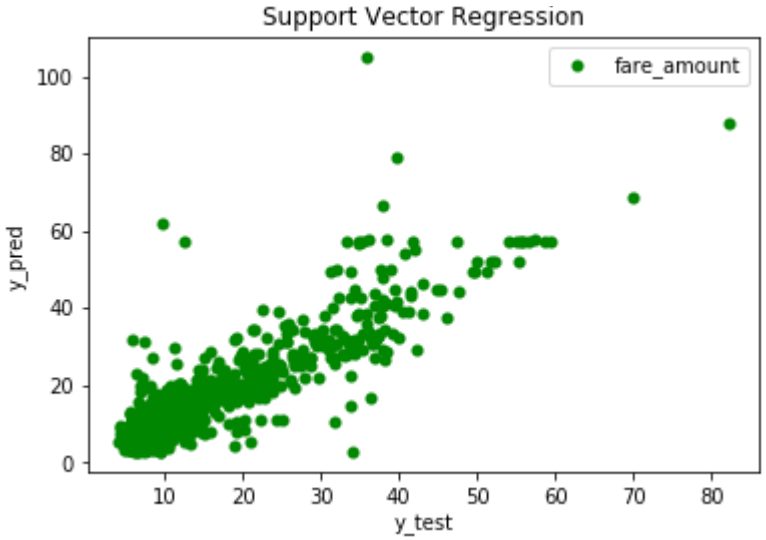
- **Justification**

Comparison to benchmark model.
Our model best score is about 3.69, benchmark score was 7.65. I think that my model could perform better. Is not good enough. Is better than our sample least square model but I think it needs more time. Training with more data and It also needs more parameters tuning.

V. Conclusion

- **Free-Form Visualization**

Vizualization of correlation between predicted values (y_pred) and true value (y_test).
1. training set 20000 rows , validation set 2000 rows RMSE = 4.543806777722515



For perfect model this line will be straight and more 'compacted' , but some correlation is visible. I think this model needs more improvement. Some points are totally out of range.

- **Reflection**

  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. I chose Support Vector Regression.

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

Most difficult thing at start is which features are relevant and how to choose hyper parameters. Relevant features select by plotting correlation between fare_price to particular feature. Details is in Data Exploration section.

- **Improvement**

I think that the most important improvement for our model is use more data for training it is crucial for machine learning in my opinion. More data also means more features, maybe 'distance' performs better than coordinates. There are a lot of things could help us improve our model. Maybe we should prioritize some ride direction more than others. For example from manhattan to the airport etc.