

Name: Sheli Bekel Sela

Machine Learning Capstone Project

New York City Taxi Fare Prediction

1. Definition:

1.1 Project Overview

Domain Background: In this project, I will try to predict the fare amount for a taxi ride in New York Given pickup and dropoff locations, the pickup timestamp, and the passenger count.

When we book a flight ticket, we know before taking off how much the airline ticket costs, so why should taxi ride be any different? Extending the “Know before you go” concept also to taxi transportation will benefit both end users and drivers. Upfront pricing makes riding simpler, while giving the end user control over how much they spend and providing the drivers ability to predict and optimize their return. These days with the competition in the transportation domain due to services like uber, getTaxi, grab, providing the end user and driving with a fare prediction is a must .

Project origin: the project is taken from Kaggle competition:

<https://www.kaggle.com/c/new-york-city-taxifare-prediction>

Datasets and Input: The data provides the details of yellow taxi rides in the New York City from Jan 2016 to June 2016 The data is available to be download in Kaggle:

<https://www.kaggle.com/c/new-york-city-taxi-fare-prediction/data> The competition data is also hosted directly within BigQuery. Can be accessed directly in:

https://bigquery.cloud.google.com/dataset/cloudtraining-demos:taxifare_kaggle?pli=1

Data description:

File descriptions:

- 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).
- 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 .

Data field:

ID

- key - Unique string identifying each row in both the training and test sets. Comprised of pickup_datetime plus a unique integer, but this doesn't matter, it should just be used as a unique ID field. Required in your submission CSV. Not necessarily needed in the training set, but could be useful to simulate a 'submission file' while doing cross validation within the training set.

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 .

1.2 Problem Statement:

Predicting the fare amount (inclusive of tolls) for a taxi ride in New York City given the pickup and dropoff locations. The basic estimate can be based on just the distance between the two points, but this will result in an RMSE of \$5-\$8, depending on the model used (an example of this approach in Kaggle Kernel: <https://www.kaggle.com/dster/nyc-taxi-fare-starter-kernel-simple-linearmodel>). The challenge is to do better than this using Machine Learning techniques .

Solution Statement: At first glance, the New York city Taxi fare prediction seems like a classical supervised regression model and I thought to apply one of the regressions models like: Ensemble Methods (Bagging, AdaBoost, Random Forest, Gradient Boosting), decision tree or SVN .

The challenge in the computations was to learn how to handle large datasets with ease and solve this problem using deep learning .

Until now I had the impression the deep learning is mostly used for image recognition and it will be interesting to learn how to address our problem using deep learning technic .

To solve this problem, the following steps will be taken:

1. Understand the problem and data
2. Data exploration / data cleaning
3. Feature engineering / feature selection
4. Model evaluation and selection
5. Model optimization
6. Interpretation of results and predictions

1.3 Metrics:

Benchmark Model: The starter project, using linear regression model, referred in the Kaggle competition overview page: <https://www.kaggle.com/dster/nyc-taxi-fare-starter-kernel-simple-linear-model> will be used as a benchmark: an example of this approach in Kaggle Kernel: <https://www.kaggle.com/dster/nyc-taxi-fare-starterkernel-simple-linear-model>

Evaluation Metrics: I will use the same evaluation metric required for the competition. The evaluation metric is the root mean-squared error or RMSE. RMSE measures the difference between the predictions of a model, and the corresponding ground truth. A large RMSE is equivalent to a large average error, so smaller values of RMSE are better. The error is given in the units being measured, so we will be able to tell very directly how incorrect the model might be on unseen data .

RMSE is given by :

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

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

I chose to use RMSE (and MSE) metrics as they are most fit for regression models with continuous variables. Our problem is a linear regression problem and amount rate target variable is a continuous variable.

RMSE also gives relatively higher weight to large errors and I defiantly wish to avoid large errors. In our cases, it makes more sense to give a higher weight to points away from the mean in which RMSE would be preferred.

2. Analysis:

2.1 Data Exploration

The Taxi Fare dataset is relatively large at 55 million training rows, with only 6 features.

The fare_amount feature will be predicted using machine learning technic.

For the exploration phase I used only subset of the data (1M data points).

Example of the data:

passenger_count	dropoff_latitude	dropoff_longitude	pickup_latitude	pickup_longitude	pickup_datetime	fare_amount	key
1	40.712278	73.841610-	40.721319	73.844311-	17:26:21 2009-06-15	4.5	17:26:21.0000001 2009-06-15 0
1	40.782004	73.979268-	40.711303	74.016048-	16:52:16 2010-01-05	16.9	16:52:16.0000002 2010-01-05 1
2	40.750562	73.991242-	40.761270	73.982738-	00:35:00 2011-08-18	5.7	00:35:00.00000049 2011-08-18 2
1	40.758092	73.991567-	40.733143	73.987130-	04:30:42 2012-04-21	7.7	04:30:42.0000001 2012-04-21 3
1	40.783762	73.956655-	40.768008	73.968095-	07:51:00 2010-03-09	5.3	07:51:00.000000135 2010-03-09 4
1	40.758233	73.972892-	40.731630	74.000964-	09:50:45 2011-01-06	12.1	09:50:45.0000002 2011-01-06 5

Data statistics:

passenger_count	dropoff_latitude	dropoff_longitude	pickup_latitude	pickup_longitude	fare_amount	
1000000.000000	999990.000000	999990.000000	1000000.000000	1000000.000000	1000000.000000	count
1.684924	39.919954	72.527860-	39.929008	72.526640-	11.348079	mean
1.323911	8.201418	11.324494	7.626154	12.057937	9.822090	std
0.000000	3114.338567-	3383.296608-	3116.285383-	3377.680935-	44.900000-	min
1.000000	40.734046	73.991385-	40.734965	73.992060-	6.000000	25%
1.000000	40.753166	73.980135-	40.752695	73.981792-	8.500000	50%
2.000000	40.768129	73.963654-	40.767154	73.967094-	12.500000	75%
208.000000	1651.553433	45.581619	2621.628430	2522.271325	500.000000	max

From looking at the data distribution and statistics, reading article on New York Taxi rates and common sense :-), I can see that there are few anomalies in the data: latitude and longitude columns, passenger_count and fare_amount. I also have data points with missing values.

2.1.1 Fare amount feature exploration:

1. Minimal fare_amount is negative (-44.9 USD), whereas the initial charge is 2.5 USD (http://home.nyc.gov/html/tlc/html/passenger/taxicab_rate.shtml). Data points below 2.5 USD will be removed.

2. fare amount STD (9.91 USD) is high comparing to the average amount (11.35 USD)

Few important notes from reading

http://home.nyc.gov/html/tlc/html/passenger/taxicab_rate.shtml article:

1. Initial rate is 2.5USD

2. Different rate for day/nights and weekends/holidays.

2.1. There is a daily 50-cent surcharge from 8pm to 6am.

2.2. There is a \$1 surcharge from 4pm to 8pm on weekdays, excluding holidays

3. Different rates for outside the city and airport rides.

4. I didn't see any limitation on the max amount.

2.1.2 Longitude/latitude coordinates features explorations:

We will look at the distribution and exclude longitude/latitude coordinates that are off the norm.

New York city coordinates are (<https://www.latlong.net/place/new-york-city-ny-usa-1848.html>):

latitude = 40.730610

longitude = -73.935242

2.1.3 Passengers count feature exploration:

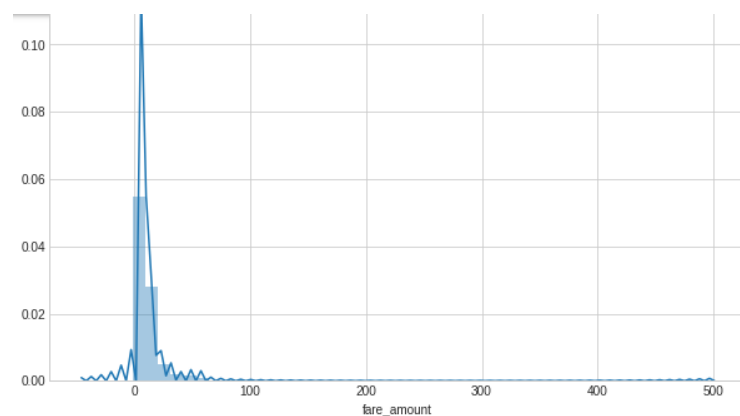
Also here it looks like we have values that doesn't make any sense. Like rides with 208 passengers and rides with zero passengers.

2.2. Exploratory Visualization

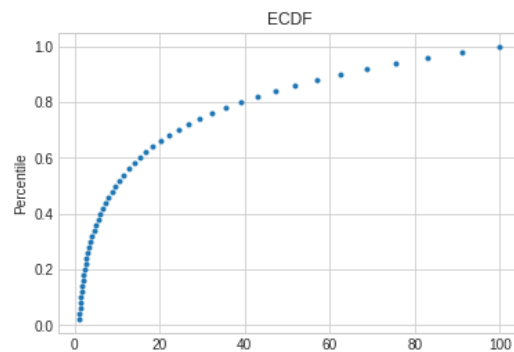
For each one of the features I plot graphs, maps and diagrams to better understand the feature distribution and the correlation to our target: ride fare amount.

2.2.1 Fare amount feature visualization:

Using seaborn's distplot, we will see both a kernel density estimates plot and a histogram



Another plot for showing the distribution of a single variable is the empirical cumulative distribution (ECD) function. This shows the percentile on the y-axis and the variable on the x-axis. ECD is good for finding outliers and the percentiles of a distribution.



From the graph we can see that most of the rides amount rate are smaller than 20. from 20 USD we can see that the graph is skewed right.

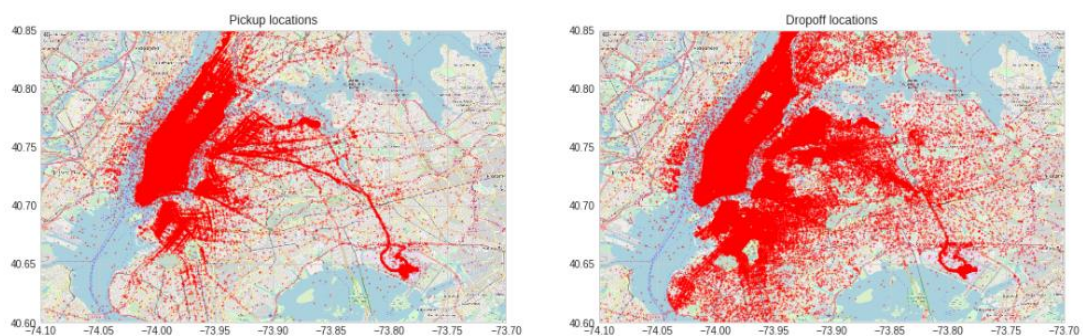
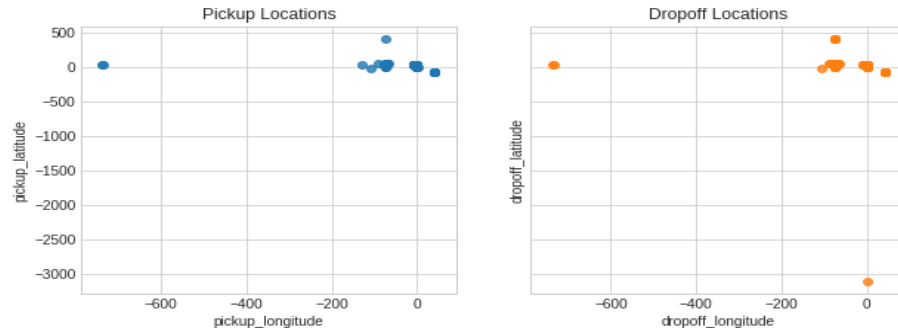
2.2.2 Longitude/latitude coordinates features visualization:

For a more contextualized representation, I plot the pickup and drop-off New York city map.

The map was extracted from OpenStreetMaps

(<https://www.openstreetmap.org/export#map=9/41.1528/-73.6496>)

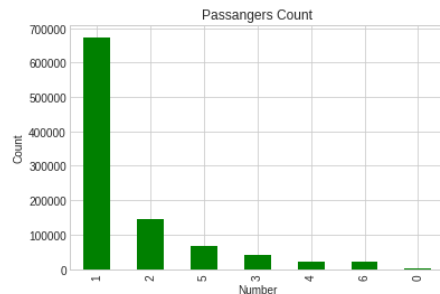
I also created a distribution diagram to get a better understanding of the data distribution.



From plotting the data on NYC map and from the distribution diagrams we can see that there are many outliers fall outside of New York coordinates or in the water.

2.2.3 Passengers count feature visualization:

A bar diagram was created to better understand the passenger count distribution. Here is was clear from the statistics that there are values that doesn't make any sense (too high for a taxi)



2.3 Algorithms and Techniques

Now that I have a better understanding of the problem (predicting the ride fare amount), domain (New York maps and fare amounts) and after analyzing, exploring and visualizing the data I can move on to the next steps.

Preprocessing the data based on my observations: the data must be cleaned, formatted, and restructured. Removing from the dataset invalid or missing entries and all the outliers that were identified. I didn't have categorial features neither nor a reason to perform scaling on numerical features. After preprocessing the data I rerun the visualization phase to ensure the outliers were corrected.

Shuffle and Split Data both features and their labels into training and validation sets. 80% of the data will be used for training and 20% for validation.

Metrics and native predictors: root mean-squared error (RMSE) and MSE metrics will be used. For the native predictor, I used the RSME value provided in Kaggle starter project referred in the competition page (RMSE=\$5.74)

I also compared my results with the leading boards and saw that I got very nice results.

In addition, I run few models and compared between them.

Feature engineering: creating new features, predictor variables, out of an existing dataset. New time and distance features were added as we know that the cost of a taxi ride is proportional to the distance and time (hour, weekday).

For each one of the features I added, I created also distribution diagram to see the correlation between the new features and the fare amount.

In addition, used correlation and heatmap diagrams.

Model evaluation and selection:

From the beginning it was clear to me that the NYC taxi ride prediction is a regression supervised learning problem.

Using the correlation and heatmap diagrams I could see also the linear correlation between few of the features and the rate amount I tried to predict.

I wanted to practice few models and compare between them. I choose simple linear model like naïve Bayes and SVR and more complex one from ensemble regressors like random forest.

I selected the model that I got the best RMSE result: Random forest.

In addition, I wanted also to practice deep learning to resolve regression problem and see if I can improve my results (was the competition challenge).

So how does the different algorithms work:

Random forest regression trees:

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction

In regression tree, the target variable is a real valued number, we fit a regression model to the target variable using each of the independent variables. Then for each independent variable, the data is split at several split points. We calculate Sum of Squared Error(SSE) at each split point between the predicted value and the actual values. The variable resulting in minimum SSE is selected for the node. Then this process is recursively continued till the entire data is covered.

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction

It adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Random Forest prevents overfitting most of the time, by creating random subsets of the features and building smaller trees using these subsets.

SVM/SVR

The goal of the SVM learning algorithm is to find a hyperplane which could separate the data accurately.

SVM works by finding the optimal hyperplane which could best separate the data, so basically it is an optimization problem

SVM maximize the distances between nearest data point and hyper-plane to decide the right hyper-plane. This distance is called as **Margin**. Higher margin makes the model more robustness. SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin

To handle non-linear separation problem (meaning curve rather than a line) SVM has a technique called the kernel **trick**. These are functions which takes low dimensional input space and transform it to a higher dimensional space. it converts not separable problem to separable problem; these functions are called kernels.

we define an epsilon range from both sides of hyperplane to make the regression function insensitive to outliers

Naive Bayes

Naïve Bayes is a classification technique with an assumption of independence among predictors. a Naive Bayes assumes that the presence of a feature in a class is unrelated to the presence of any other feature.

I think that Naive Bayes doesn't fit to regression problems but rather to classification problem and it was a mistake to choose this model.

CNN (Deep learning)

With deep learning we build a model architecture that contains layers. Each layer has

Using deep learning we build a network made of interconnected neurons. The network is built with multiple layers connected to each other.

Each neuron is characterized by its weight, bias and activation function.

The input is fed to the input layer. The neurons do a linear transformation on the input by the weights and biases. The non linear transformation is done by the activation function.

The information moves from the input layer to the hidden layers. The hidden layers process and send the final output to the output layer.

This is the forward movement of information known as the forward propagation.

We will repeat this process, called back-propagation, where we update the weights and biases of the neurons based on the loss function.

Once the entire data has gone through this process, the final weights and biases are used for predictions.

Model tuning: Fine tune the chosen random forest model using grid search and tuning it's parameters

I also adjust the CNN model and tune it and it's parameters.

Feature importance: find the features that contributed the most to predict the target.

2.4 Benchmark

As stated above, I used the RMSE and MSE metrics to choose between the models.

I ran few linear regression models and tuned the one with the best results.

Then, I compared between the results of the CNN and the supervised linear regression model.

3. Methodology:

3.1 Data Pre-processing

As part of the pre-processing phase, I removed all outliers described on section 2.1.

3.1.1 Removing all missing outliers

All null values were removed.

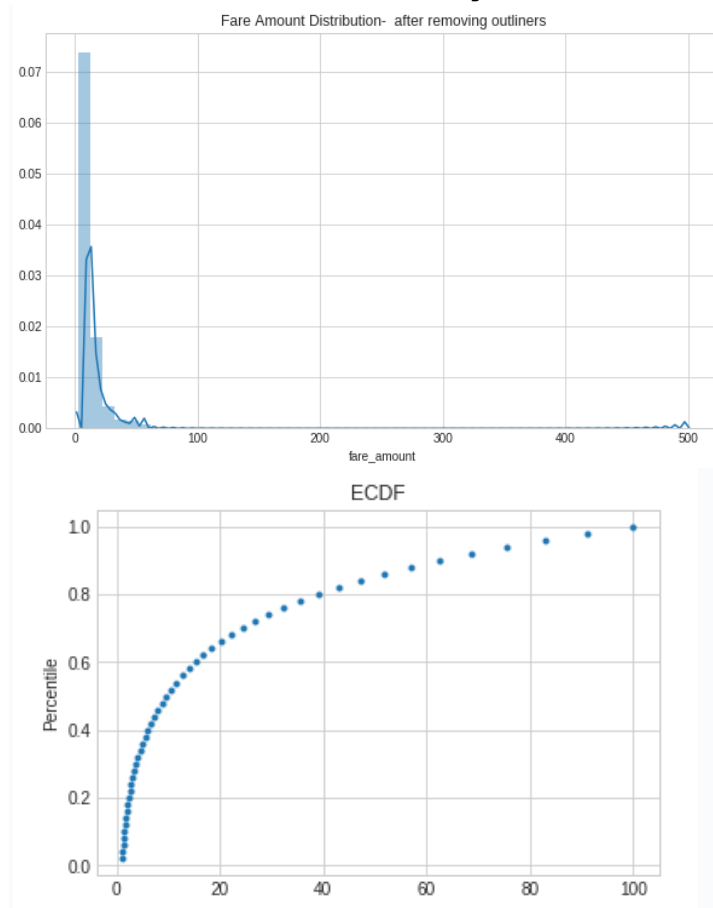
Data set size before clean up process: 1000000

Data set size after removing null outliers: 999990

3.1.2 Remove Fare amount outliers:

Remove all data point with value below 2.5USD (as this is the initial rate)

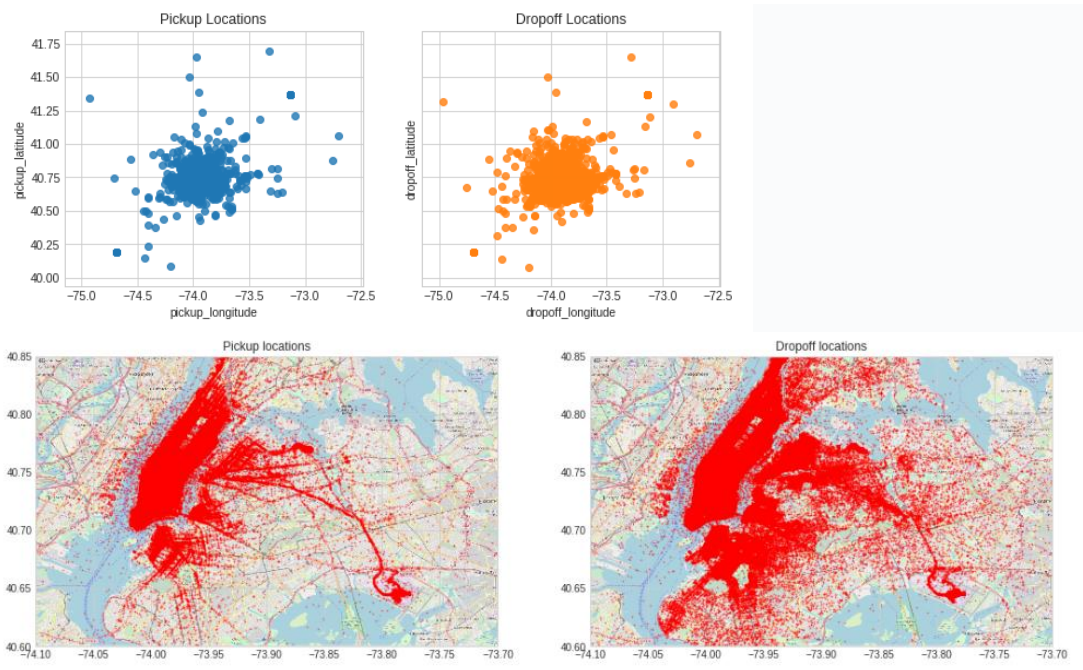
Data set size after removing fare amount outliers: 995910



3.1.2 Remove pickup/drop-off coordinates outliers:

All outliers that doesn't fall in New York city Longitude/latitude coordinates were removed

Data set size after removing latitude and longitude outliers: 975661



Looks much better, still not there are data point that appears in the_water.

3.1.2 Remove passenger count outliers:

All data point greater than 6. I decided not to remove the zero-data point as it might be rides to deliver packages.

Data set size after removing passenger count outliers: 975661

And now the statics looks much more better:

passenger_count	dropoff_latitude	dropoff_longitude	pickup_latitude	pickup_longitude	fare_amount	
975661.000000	975661.000000	975661.000000	975661.000000	975661.000000	975661.000000	count
1.685860	40.751430	73.974314-	40.751090	73.975204-	11.363810	mean
1.308202	0.033084	0.038128	0.029902	0.038887	9.749911	std
0.000000	40.041180	74.964263-	40.052722	74.968142-	2.510000	min
1.000000	40.735607	73.991577-	40.736592	73.992277-	6.000000	25%
1.000000	40.753886	73.980600-	40.753427	73.982090-	8.500000	50%
2.000000	40.768427	73.965375-	40.767597	73.968352-	12.500000	75%
6.000000	41.923820	72.196091-	41.800252	72.702870-	500.000000	max

For CNN (deep learning) I also run mix Max scalar on the training data.

3.2 Implementation

3.2.1 Shuffle and Split Data

As described in section 2.3 I wrote methods to split and shuffle the data to avoid overfitting.

3.2.2 Metrics and native predictors

I create methods to calculate the evaluation metrics : RMSE and MSE.

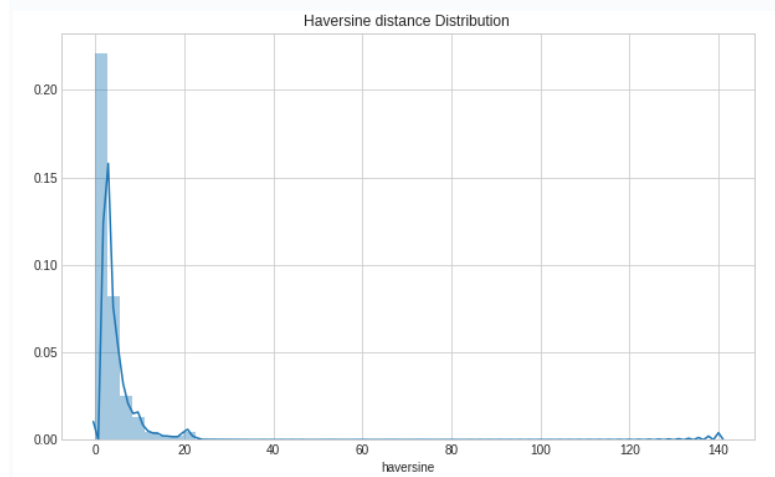
3.2.3 Feature engineering

Two major features have impact on the taxi ride:

1. Distance - From many discussions in Kaggle's it seems like using the Haversine distance will be the best to calculate a more realistic distance between the pickup and drop-off. It is more accurate than Manhattan and Euclidean distances which are relative and do not consider the spherical shape of the Earth.
2. Time (Different rate for day/nights and weekends/holidays), therefore I added year, hour, weekday features.

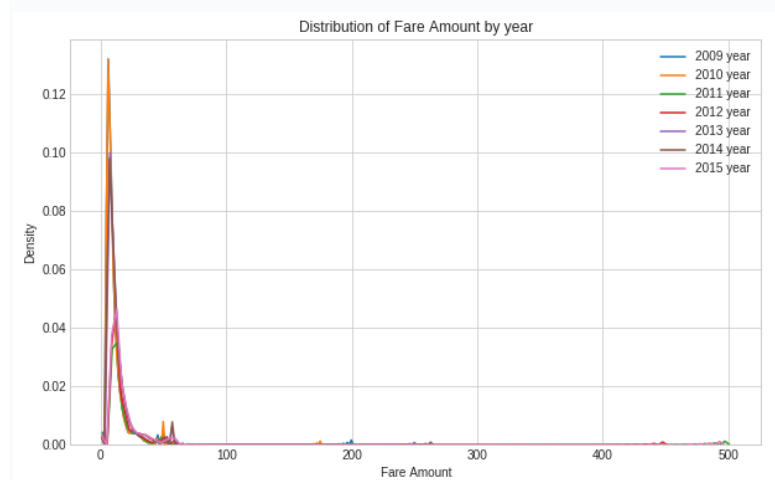
Then, I plot graph to prove my assumptions and see that these features have correlation to the rate amount.

Haversine distance distribution: we can see linear correlation between distance and fare amount.

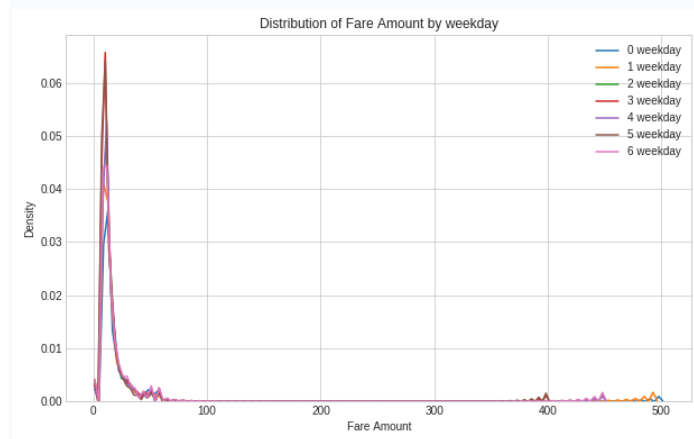


Time features (Year, Hour, Weekday) distributions:

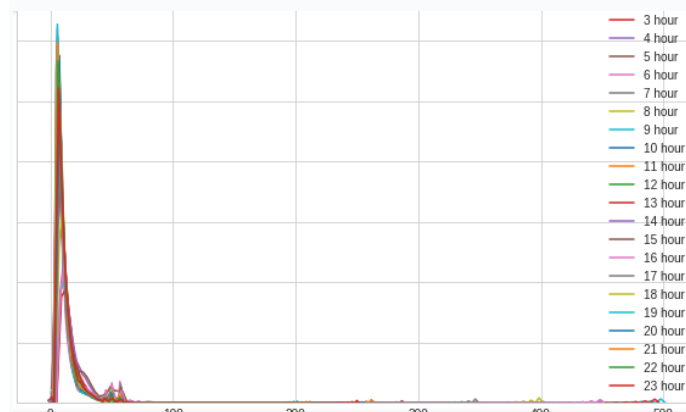
Year:



Day week:

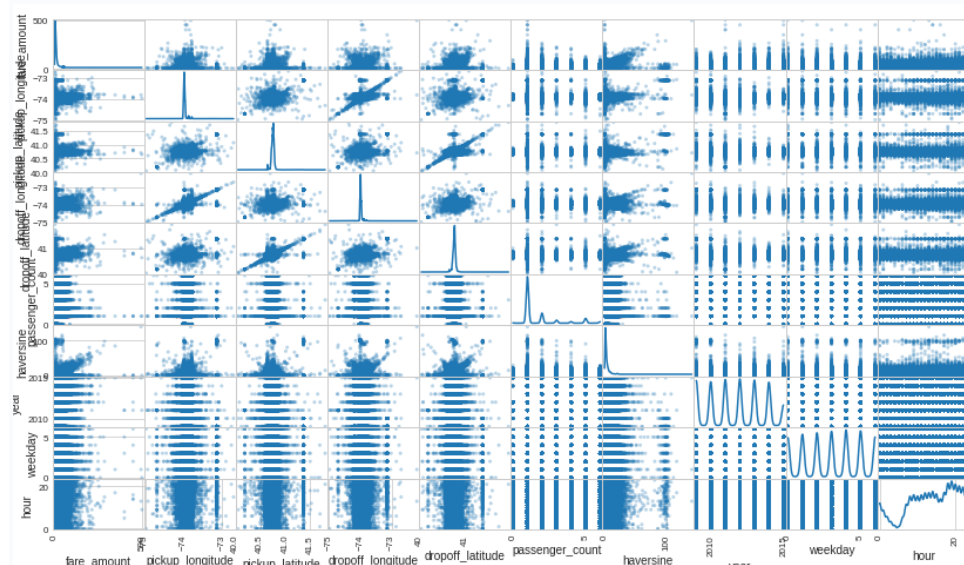


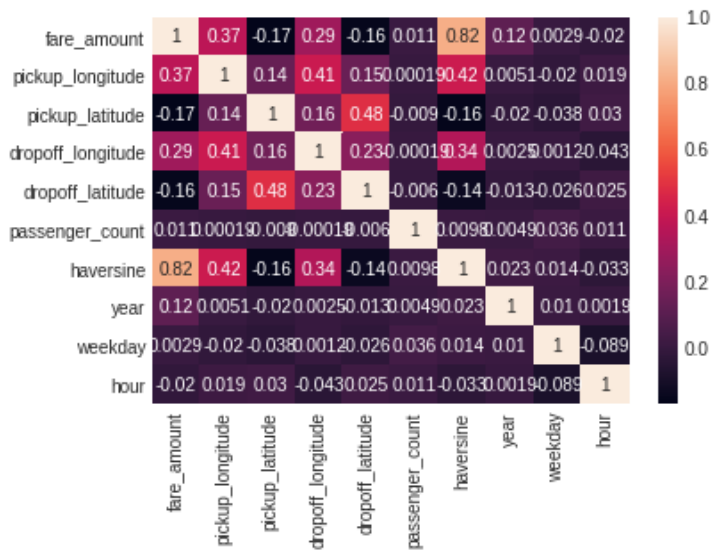
Hour:



From the diagrams, I can learn that the year and hour features have impact on the fare amount. the price per distance varies by the hour of day. I couldn't find much impact dues to week days, which was a bit surprising for me.

To get a better understanding of data correlation, we can construct a scatter matrix and heatmap to find correlation between features.





From the heatmap and scatter matrix we can find, as expected, a very strong linear correlation between the distance and the fare amount.

I also noticed a small correlation between the year and the rate amount.

I didn't see strong correlation to the hour/weekend.

3.2.4 Model evaluation and selection:

Supervised regression learners:

I trained the three learners that I've selected with the final list of features, including distance and time features (I dropped 'key', 'fare_amount', 'pickup_datetime').

I fit the learners to the training data.

I used the validation data and sub set of the training data to perform prediction and to calculate the evaluation metric RMSE.

Based on the RMSE I was able to compare between the models and choose the one that best performed.

CNN (Deep learning):

I build a CNN architecture based on 6 Convolution layers.

I added batchNormalization to normalize the activations of the previous layer at each batch, i.e. applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1

For each Convolution layer I used ReLU activation function to introduce none-linearity. I added dropout consists in randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting.

```

model = Sequential()

BatchNormalization(epsilon=1e-06, mode=0, momentum=0.9, weights=None)
model.add(Dense(128, activation='relu', input_shape=(X_train_scaled.shape[1],)))

model.add(Dropout(0.25))

model.add(Dense(64, activation='relu'))
model.add(Dropout(0.25))

model.add(Dense(32, activation='relu'))
model.add(Dropout(0.25))

model.add(Dense(16, activation='relu'))
model.add(Dropout(0.25))

model.add(Dense(8, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='relu'))

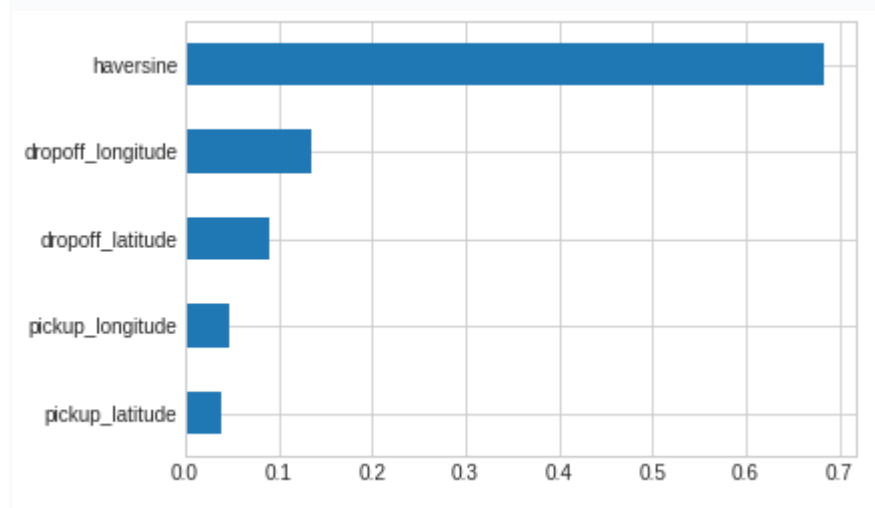
```

After building the model, I compiled it and train it. I used model checkpointing to save the model that attains the best validation loss. Here I used the MSE metric as I was not able to use my customized loss function.

As I used the MSE also the supervised regression models, I was able to compare between them.

3.2.5 Feature importance

Haversine distance was the most important feature to predict the fare_amount. Time features seems to have minimal impact.



Code complication I faced in the process:

In general, errors are not self descriptive. It was hard to understand the nature of the problem from the error message.

One mistake I did, is trying to train the models with features that were in datetime format (pickup_datetime, key features). To solve this issue I dropped them before splitting the data and training the model.

```
features_final= data.drop(['key','fare_amount','pickup_datetime'], axis = 1)
```

Another issues was related to creation the target fare_amount.

I had to convert it target to an array using:

```
fare_amount = np.asarray(data['fare_amount'], dtype="S1")
```

And then when calling RMSE function to cast it to float . I applied the asType(float) casting on the two objects (target, prediction) to solve the issue.

```
y_actual.astype(float), y_predicted.astype(float)
```

3.3 Refinement

Supervised regression learners:

In this final step, I choose from the three supervised learning models the *best* model to use on the trained data. I performed a grid search optimization for the model over the entire training set (X_train and y_train) by tuning its parameter to improve upon the untuned model's RMSE

From the supervised learning model, the model that perform the most was Random forest. I tuned its n_estimators', 'max_features': ['sqrt', 'log2'] and 'max_depth' : [3,10,20] parameters.

```
Parameters used by optimal model are: {'bootstrap': True, 'criterion': 'mse', 'max_depth': 3, 'max_features': 'sqrt', 'max_leaf_nodes': None, 'min_impurity_decrease': 0.0, 'min_impurity_split': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 20, 'n_jobs': 1, 'oob_score': False, 'random_state': 42, 'verbose': 0, 'warm_start': False}
```

CNN (Deep learning):

I tuned the number of epochs to train the model.

I didn't perform major modifications to the model architecture as I got good results and my machine resource were limited.

4. Results

4.1 Model Evaluation and Validation

Model	Train dataset		Validation data set		comments
	RMSE	MSE	RMSE	MSE	
Supervised learner: Gaussian Naive Bayes	3.533	12.486	3.237	10.483	Simple model.
SVR	-	-	-	-	Was not able to run. Require a lot of machine resource. The model didn't converge.
Random forest	0.929	0.864	2.053	4.217	
CNN	2.213	4.899	2.240	5.018	

Random forest was outperforms compared to the linear regression as it has more flexibility. It reduced variance because it combines the predictions of many decision trees. A linear regression is a simple method and as such has a high bias (it assumes the data is linear).

It also outperformed comparing to the CNN. It looks like the CNN is not the best fit to solve regression problems.

Random forest hyper parameters:

- `n_estimators: [20]`, The number of trees in the forest.
- `max_features: ['sqrt', 'log2']`, The maximum depth of the tree
- `max_depth : [3,10,20]`, The number of features to consider when looking for the best split

I used a limited set of parameter as random search is computationally expensive as it uses K Fold cross validation to assess the model. For each combination of hyperparameters, we are training and testing the model K times.

I started with more options but didn't have enough machine resources to handle.

Trying to be objective 🤖, I believe random forest is a robust model by its nature and not sensitive to outliers. It reduced variance because it combines the predictions of many decision trees. it should not overfit when increasing the number of trees in the forest. In addition the algorithm applies cross validation which also helps to avoid overfitting,

To validate my assumption, I will try to run it with 3 different samples.

Running on 3 different sample sizes:

```
RandomForestRegressor trained on 7805 samples
RandomForestRegressor RMSE for trained data is: 0.872066511224918
RandomForestRegressor RMSE for validated data is: 2.232991082302064
RandomForestRegressor MSE for trained data is: 0.7605
RandomForestRegressor MSE for validated data is: 4.986249173640544
```

```

Start running supervised regression models
Running RandomForestRegressor model
RandomForestRegressor trained on 78052 samples
RandomForestRegressor RMSE for trained data is: 0.9207967564379593
RandomForestRegressor RMSE for validated data is: 2.13794101786089
RandomForestRegressor MSE for trained data is: 0.8478666666666667
RandomForestRegressor MSE for validated data is: 4.57079179585206
Start running supervised regression models
Running RandomForestRegressor model
RandomForestRegressor trained on 780528 samples
RandomForestRegressor RMSE for trained data is: 0.9299103899480494
RandomForestRegressor RMSE for validated data is: 2.053588579899758
RandomForestRegressor MSE for trained data is: 0.8647333333333332
RandomForestRegressor MSE for validated data is: 4.217226055494705

```





I don't see major change in RSME value for the validation data, so I believe that we don't have overfit

Another improvement that I can do is the run the model again with outliers and check its performance

4.1 Justification

The RMSE in the starter project that is used as a reference was: \$5.74 USD.

Also comparing to other boards, I think I managed to get a good score (better than #3).

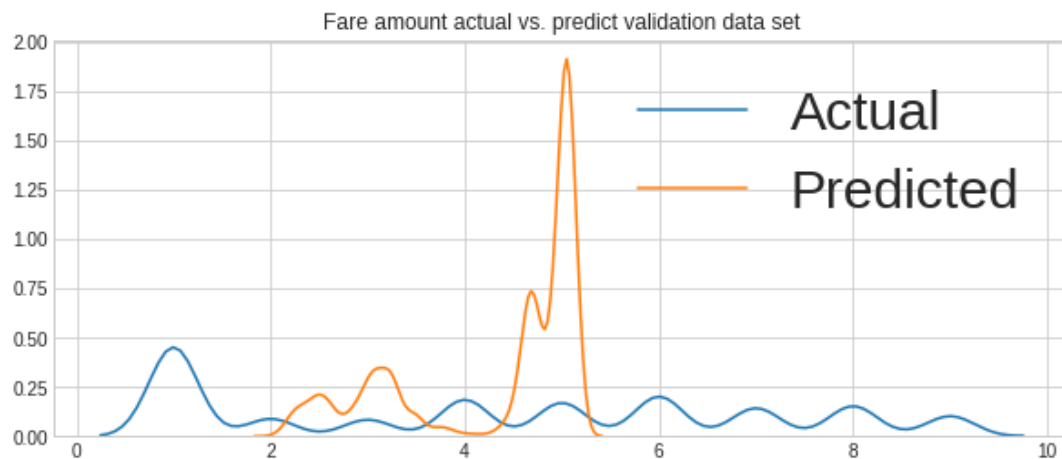
#	Δ1w	Team Name	Kernel	Team Members	Score 🏆	Entries	Last
1	—	rmtest2			1.38506	1	4mo
2	new	rmtest1			1.60451	8	3mo
3	▼ 1	Luis San Román			2.07148	4	3mo
4	▼ 1	cookie monster			2.59631	48	3mo

5. Conclusions

5.1 Free-From Visualization

I decide to visualize the predicted rate amount based on the validation data. I choose to use this visualization as I think it gives a good view of the model quality (prediction correction).

I was expected to see a better correlation between the graphs...



5.2 Reflection

To solve New York city rate prediction problem, I tried to use few machine learning techniques. I started with data explorations to better understand the data. I was surprised to see how much effort I've invested in data exploration, data pre-processing and feature engineering. I think it is the most important and complicated step in the pipeline. Once we identify all the features, potential features and correlation between them, then it was easier to understand the type of problem (linear regression in our case).

For selection the model, I tried few models. I think it is a good practice to try few models. To start from simple one and move to more complex.

I really wanted to try deep learning in a project that is not classification or image recognition.

Tuning the models was time consuming, hard task and some of the models that I tried got stuck and crashed my entire project.

It was also my first time to work on Kaggle project and I was amazed from the amount of knowledge a sharing that I could have found there. For sure I will use this platform in the future.

Finally, I really enjoyed it and feel that I learned a lot regarding to how to approach a problem and how to run a machine learning pipeline.

5.3 Improvement

The following improvements could have being done:

Data Explorations:

- There are still outliers related to latitude and longitude that I could handle.
- Rides to airports has fix price and could also be handled

Feature engineering:

- Add more time related features

Models:

- Try more complex models such as the gradient boosting machine. I saw few kernel that used it and got good results and also was recommended in the review
- Random forest: tune the models with more hyperparameters
- Tune the models with more training data
- CNN – tune the architecture (change the layers and try different activation methods), increase number of epochs

6. References

<https://towardsdatascience.com/another-machine-learning-walk-through-and-a-challenge-8fae1e187a64>

Other Kaggle projects:

<https://www.kaggle.com/breemen/nyc-taxi-fare-data-exploration>

<https://www.kaggle.com/willkoehrsen/a-walkthrough-and-a-challenge>

<https://www.kaggle.com/breemen/nyc-taxi-fare-data-exploration>

Other reference:

http://home.nyc.gov/html/tlc/html/passenger/taxicab_rate.shtml

<https://www.latlong.net/place/new-york-city-ny-usa-1848.html>

<https://www.openstreetmap.org/export#map=12/40.7250/-73.8999>

<https://towardsdatascience.com/automated-feature-engineering-in-python-99baf11cc219>

https://en.wikipedia.org/wiki/Haversine_formula

<https://stackoverflow.com/questions/45173451/scikit-learn-how-to-calculate-root-mean-square-error-rmse-in-percentage/45173579>

https://scikit-learn.org/stable/tutorial/machine_learning_map/

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>

<https://medium.com/usf-msds/choosing-the-right-metric-for-machine-learning-models-part-1-a99d7d7414e4>

<https://shuzhanfan.github.io/2018/05/understanding-mathematics-behind-support-vector-machines/>

<https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>