Yellowcab data analysis

jakub.roman.borkowski

February 2021

1 Introduction

As a party of the recruitment process I was asked to perform broadly understood „analysis" of the data regarding yellow taxi trips in New York City (NYC). The period of interest was late 20s of 21st century (2018-2019). The issue of modelling this data set have been drawing a fair bit of interest lately (e. g this kaggle competition), for its volume, noisiness and variety of rules to explore. That is why I am sure that my analysis barely scratches the surface of what hides in realms of NYC taxi trips.

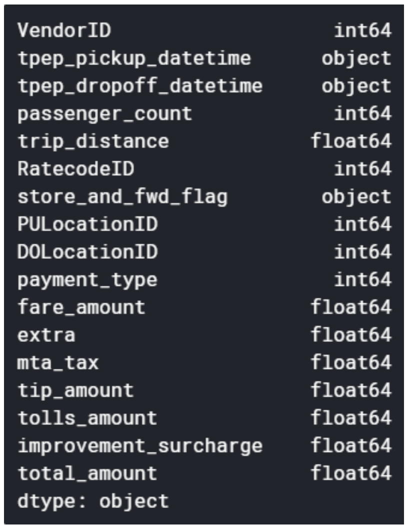
This report crowns the analysis from this repository. Starting from the foundations, the project structure starts from low level tool kits and model generating functions goes through the three notebooks (domain understanding, exploratory data analysis - EDA and modelling), which helps to gain high level orientation in the project but it is this report that is not in a draft form and offers an insight into reasoning and outcomes that this analysis consists of.

2 Domain understanding

To prepare this section domain understanding notebook and dictionary prepared by the TLC were used. For analysis the data from 01.2018 and 01.2019 was used to gain some additional intuition regarding the changes in distributions between the "sister" months. In further inquiry the data (resp. column) drawn from a next year was referred to as "sister data" (resp. "sister column").

Storing the data from January only consumed 2.8 GB of RAM, fortunately the Dask library came to help in latter storing the data and processing it further. I was operating on sample of size 8759874 and 17 columns. What was interesting was the index of that data which came out to consists of 728580 unique values hence there approximately ten times less "unique" records (based on index). It was also impossible for the index to represent the specific driver (there would be less than twenty drivers in NYC city then). I decided to inspect that matter.

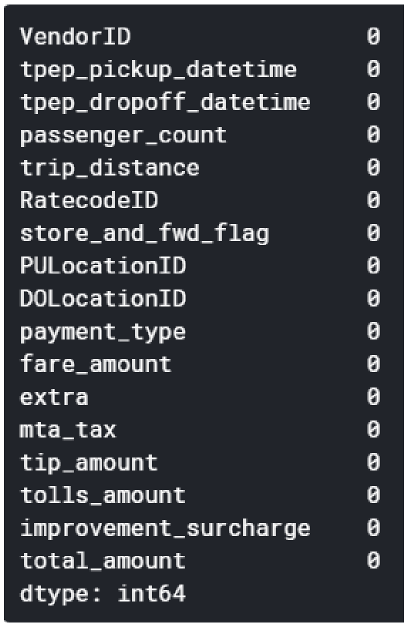
There was no clear similarity in these rows, except an improvement surcharge which is said to take only values from 0, 0.5, so it could not be related to the index. Nonetheless from then on I was referring to records with the same index as unique trips. In the EDA notebook this fact was used once more.



```
VendorID                   int64
tpep_pickup_datetime      object
tpep_dropoff_datetime     object
passenger_count            int64
trip_distance            float64
RatecodeID                 int64
store_and_fwd_flag        object
PULocationID               int64
DOLocationID               int64
payment_type               int64
fare_amount              float64
extra                    float64
mta_tax                  float64
tip_amount               float64
tolls_amount             float64
improvement_surcharge    float64
total_amount             float64
dtype: object
```

Figure 1: Types of data in the dataset from 01.2018

I started off by inspecting the types and missing values in data, which turned out to be pretty clean, at least in the context of missing values. In that moment I was not bothered by the mismatching types of data, processing functions dealt with that swiftly.



```
VendorID                 0
tpep_pickup_datetime     0
tpep_dropoff_datetime    0
passenger_count          0
trip_distance            0
RatecodeID               0
store_and_fwd_flag       0
PULocationID             0
DOLocationID             0
payment_type             0
fare_amount              0
extra                    0
mta_tax                  0
tip_amount               0
tolls_amount             0
improvement_surcharge    0
total_amount             0
dtype: int64
```

Figure 2: Missing values counts

2.1 Column insights

2.1.1 VendorID

This feature indicates the TPEP (Taxicab Passenger Enhancement Program) provider of a record. Unique values:

- 1 := Creative Mobile Technologies, LLC
- 2 := VeriFone Inc.

It was treated as categorical variable. I then proceeded with inspecting its relevancy i.e. the impact that it makes on distribution of the rest of the data. For this purpose I used a handy tool by Facebook: HiPlot which is great choice when it comes for interactive alluvial plotting.

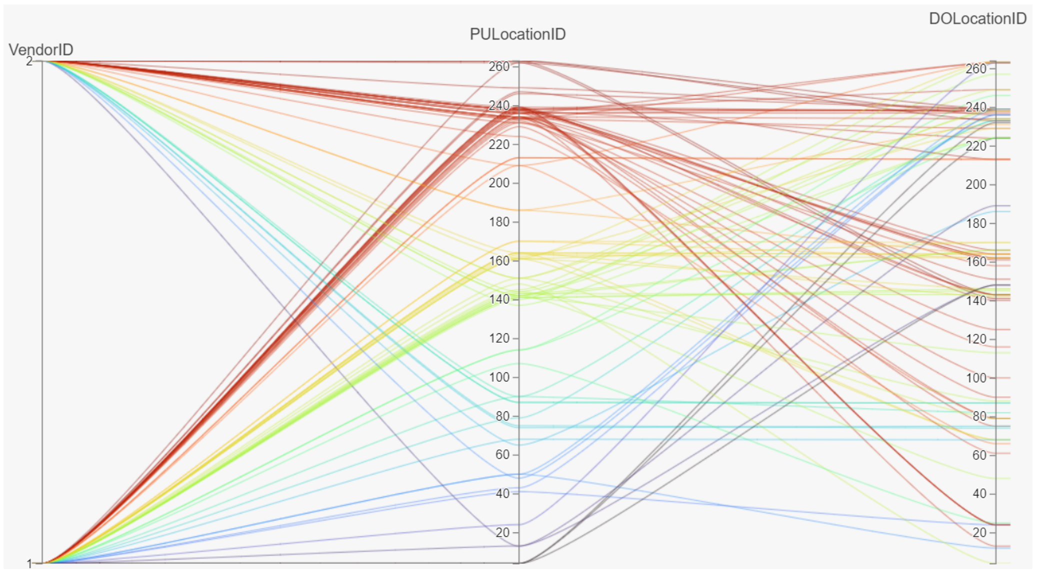
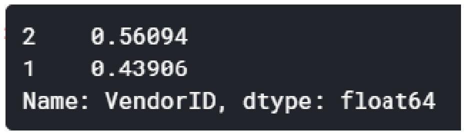


Figure 3: Scenario plot vendor Id versus pick-up and drop-off locations

It seemed that there were no separate districts in which the both providers have operated. That is why this column was not included in the further processes.



```
2    0.56094
1    0.43906
Name: VendorID, dtype: float64
```

Figure 4: Distribution of the VendorID variable

2.1.2 tpep_pickup_datetime, tpep_dropoff_datetime

The date and time when the meter was engaged/disengaged.

Format of the string: $'yyyy - mm - dd\ hh : mm : ss'$ was used for calculating how long the specific trip lasted. In the first notebook the operation was designed to be column-wise but later I replaced that with row-wise operation for better efficiency.

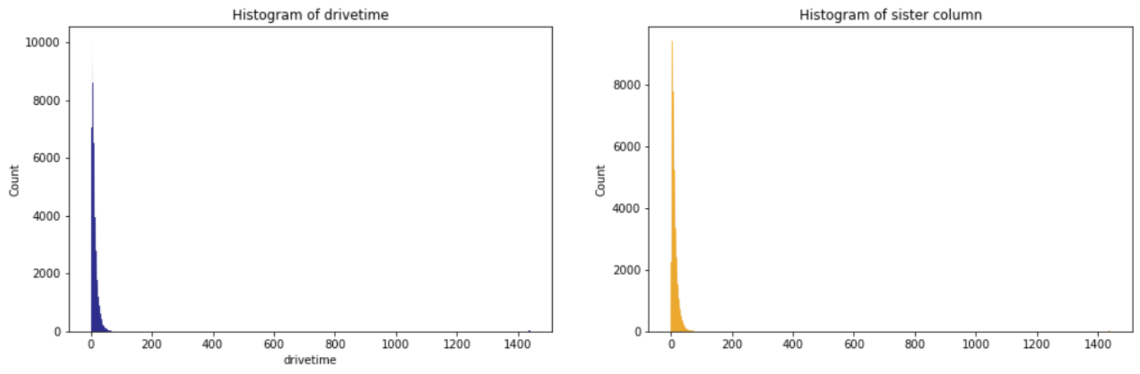
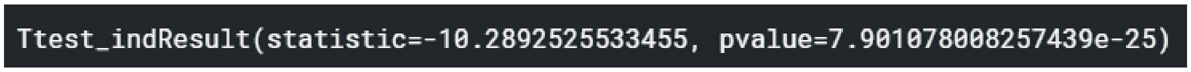


Figure 5: Distributions of the drive time variable in sister months

Sister data appeared to be distributed in similar manner, although there was a slight decreasing trend from month to month.

Drive times seemed to be concentrated on a shorter rides (time given in minutes), which was of high importance taking under consideration initial fees, more quick drives means shorter distances, less time spent on returning to the city centre and more engaging fees collected. Nonetheless the outliers should be inspected. But before that the equal expected values hypothesis was tested using the rule of thumb T Test.

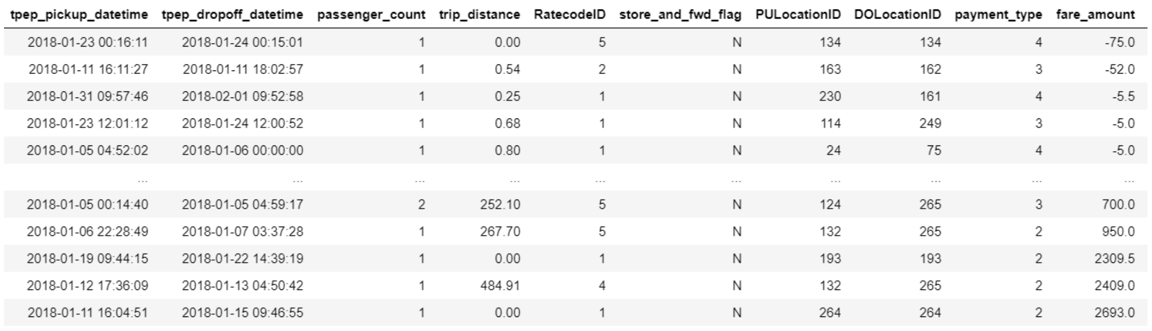


```
Ttest_indResult(statistic=-10.2892525533455, pvalue=7.901078008257439e-25)
```

Figure 6: T Test outcome, sample size of 500 000 records

The outcome meant that the null hypothesis had to be turned down in favor of the alternative one: expected value is changing. It could mean that while predicting the drive time, data from a specific month (or season) should be used.

It seemed that the outliers might be related both to the personal errands or even trips (negative fare) and longer drives. The odd thing is that some of the heavily charged services were not related to driving the passengers at all. That is why I would recommend more care when using these records in EDA and modelling processes.



tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	payment_type	fare_amount
2018-01-23 00:16:11	2018-01-24 00:15:01	1	0.00	5	N	134	134	4	-75.0
2018-01-11 16:11:27	2018-01-11 18:02:57	1	0.54	2	N	163	162	3	-52.0
2018-01-31 09:57:46	2018-02-01 09:52:58	1	0.25	1	N	230	161	4	-5.5
2018-01-23 12:01:12	2018-01-24 12:00:52	1	0.68	1	N	114	249	3	-5.0
2018-01-05 04:52:02	2018-01-06 00:00:00	1	0.80	1	N	24	75	4	-5.0
...	...	...	...	...	...	...	...	...	...
2018-01-05 00:14:40	2018-01-05 04:59:17	2	252.10	5	N	124	265	3	700.0
2018-01-06 22:28:49	2018-01-07 03:37:28	1	267.70	5	N	132	265	2	950.0
2018-01-19 09:44:15	2018-01-22 14:39:19	1	0.00	1	N	193	193	2	2309.5
2018-01-12 17:36:09	2018-01-13 04:50:42	1	484.91	4	N	132	265	2	2409.0
2018-01-11 16:04:51	2018-01-15 09:46:55	1	0.00	1	N	264	264	2	2693.0

Figure 7: Drive time outliers

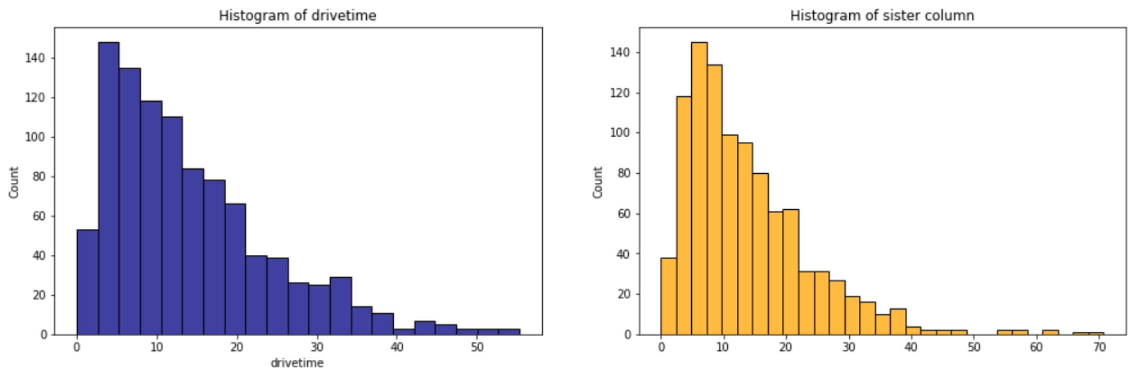


Figure 8: Truncated drive times distribution

Having ignored the drive time outliers it was possible to visualize the sister distributions once more.

2.1.3 passenger_count

The number of passengers, driver included. Should be checked for any anomalies. I decided to inspect its relation to the length of the ride and drive speed.

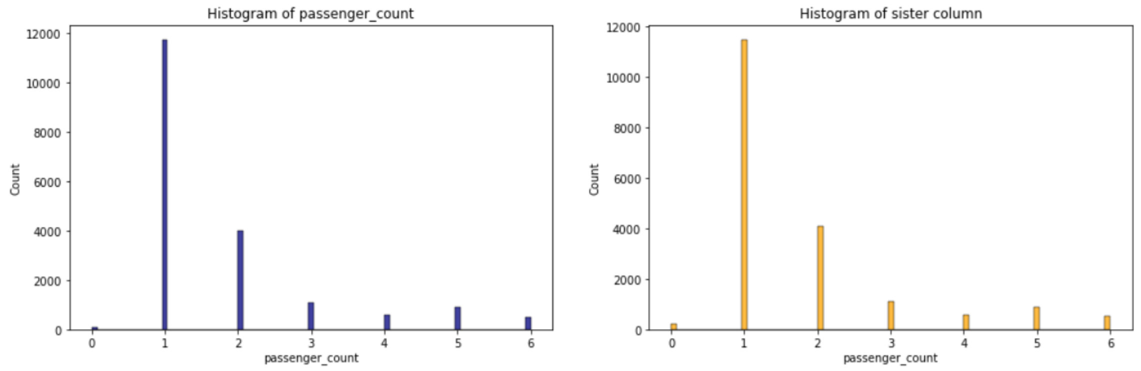
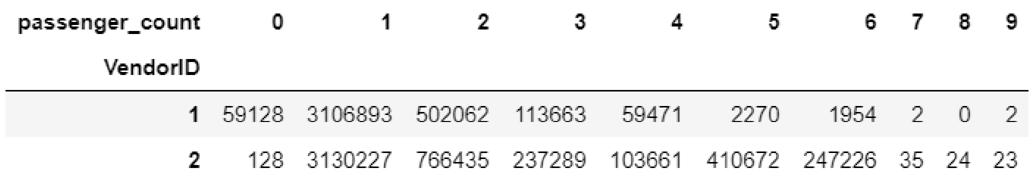


Figure 9: Distribution of the number of passengers

There seems to be some personal or anomaly trips included in the data. Let's inspect them before dropping.



passenger_count	0	1	2	3	4	5	6	7	8	9
VendorID										
1	59128	3106893	502062	113663	59471	2270	1954	2	0	2
2	128	3130227	766435	237289	103661	410672	247226	35	24	23

Figure 10: Conditional distribution of payment type on VendorID

Occurrences of the 'zero-passengers' records were related to the VendorID. I thought there might be some difficulties in collecting this data hence these records were be excluded from modelling but it is possible that including them would bring some value.

2.1.4 store_and_fwd_flag

This flag indicates whether the trip record was held in vehicle memory before sending to the vendor, because the vehicle did not have a connection to the server.

Unique values:

- Y= store and forward trip
- N= not a store and forward trip

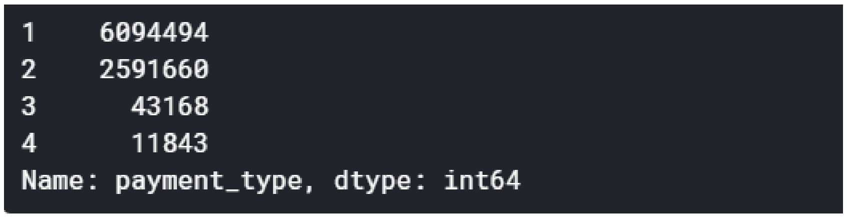
This column was not be used in further analysis for its highly probable irrelevance.

2.1.5 payment_type

A numeric code signifying how the passenger paid for the trip.
Unique values:

- 1= Credit card
- 2= Cash
- 3= No charge
- 4= Dispute
- 5= Unknown
- 6= Voided trip

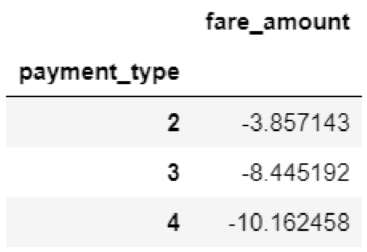
After taking a look at the dictionary I almost would have called this column one of the most important ones. That was because that grim ″dispute" value. Intuitively it could be related to default in payment. Only after short inquiry did I realized that it was not the case.



```
1    6094494
2    2591660
3      43168
4      11843
Name: payment_type, dtype: int64
```

Figure 11: Conditional distribution of fares for disputed payments

Soon it stroke me that the "dispute" value not necessarily indicates that the whole charge was lost. Some negative values were encountered but their sparsity cut made a mockery of the idea that modelling this variable would bring some greater value.



payment_type	fare_amount
2	-3.857143
3	-8.445192
4	-10.162458

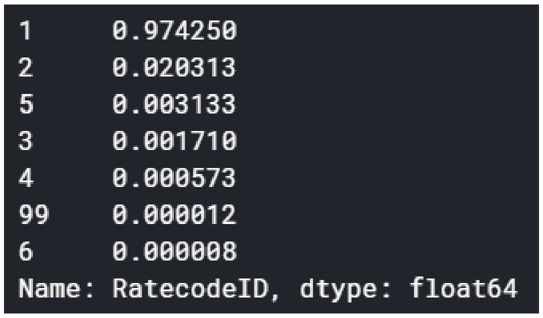
Figure 12: Counting the negative values for payment types

2.1.6 ratecodeID

The final rate code in effect at the end of the trip.
Unique values:

- 1= Standard rate
- 2=JFK
- 3=Newark
- 4=Nassau or Westchester
- 5=Negotiated fare
- 6=Group ride

It was worth checking if predicting the negotiated fare would have any business value but there were only a few negotiated fares and category "99" which origins are rather blurry hence the column was dropped before EDA.



```
1     0.974250
2     0.020313
5     0.003133
3     0.001710
4     0.000573
99    0.000012
6     0.000008
Name: RatecodeID, dtype: float64
```

Figure 13: Distribution of the rate codes

2.1.7 Financial data

There are few columns in this category:

- fare_amount: The time-and-distance fare calculated by the meter.
- extra: Miscellaneous extras and surcharges. Currently, this only includes the 0.50 dollar and 1 dollar rush hour and overnight charges.
- mta_tax: 0.50 dollar MTA tax that is automatically triggered based on the metered rate in use.
- improvement_surcharge: 0.30 dollar improvement surcharge assessed trips at the flag drop.
- tip_amount: Tip amount – This field is automatically populated for credit card tips. Cash tips are not included.

- tolls_amount: Total amount of all tolls paid in trip.
- total_amount: The total amount charged to passengers. Does not include cash tips.

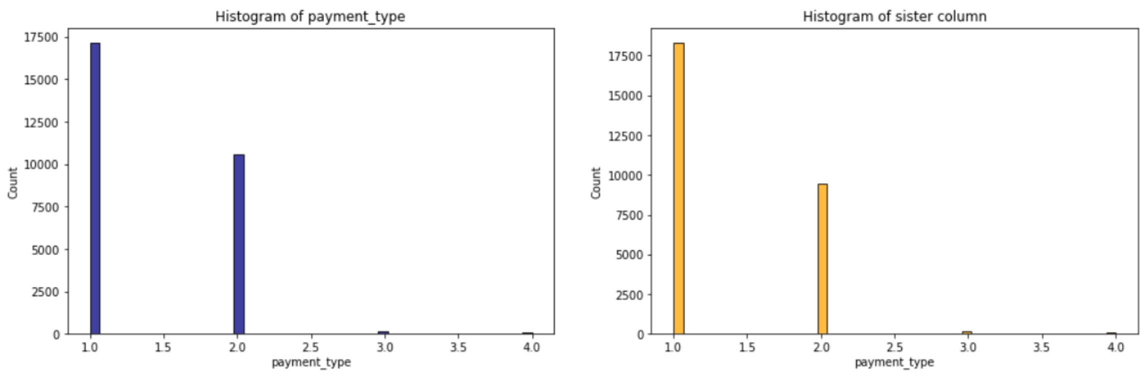


Figure 14: Distribution of fares

This data was used to check whether the "dispute" payment type really would have any impact on the final charge. Also the presence of the tip amount-related information was expected to indicate that there had been a card payment, which could be used to fill some of the missing payment type data.

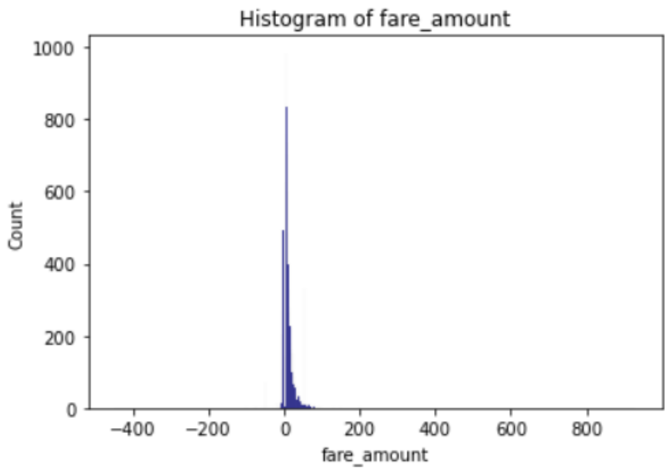


Figure 15: Distribution of fares for disputed payments

From then I decided to use just "fare_amount", "tip_amount" and "extra" variables for their relevance.

Negative charge records were sparse and for their sparsity were not included in further analysis. Their relation to pickup and drop-off location also seemed unclear.

2.1.8 PULocationID, DOLocationID

Respectively: TLC Taxi Zone in which the taximeter was engaged, TLC Taxi Zone in which the taximeter was disengaged.

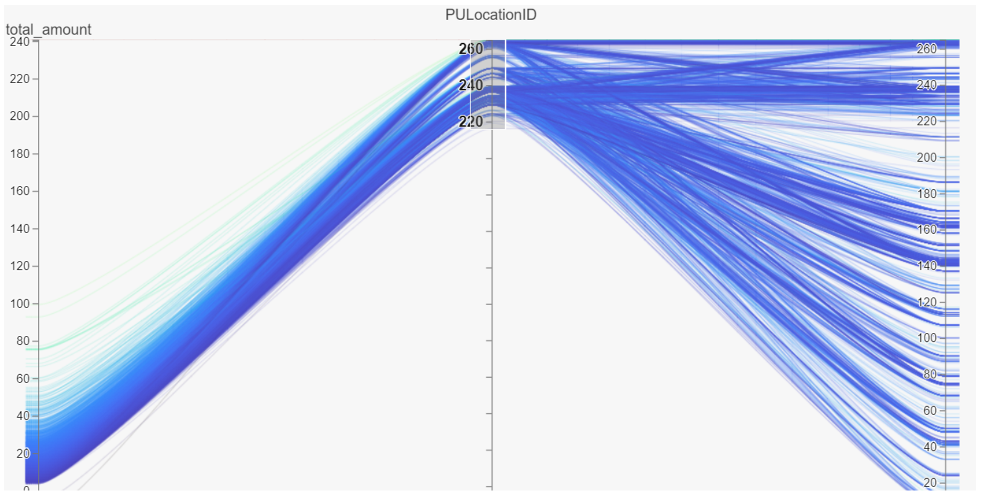


Figure 16: Alluvial plot regarding the pick-up and drop-off locations, zooming on medium fares

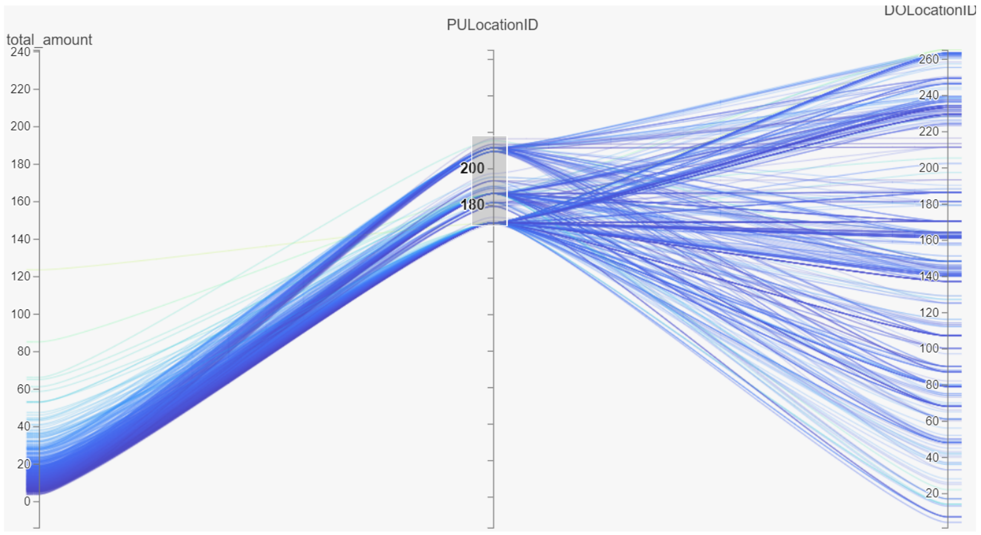
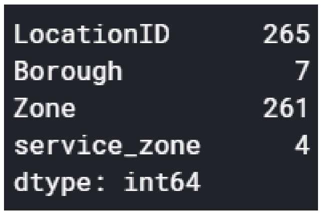


Figure 17: Alluvial plot regarding the pick-up and drop-off locations, zooming on cheaper trips

These were used for engineering trip duration. There is lookup table (Amazon S3 bucket) provided by TLC, able to decipher the Borough, Zone and the information which type of taxi services the specific zone.



```
LocationID      265
Borough           7
Zone            261
service_zone      4
dtype: int64
```

Figure 18: Scope of the lookup table

Obviously there was a correlation between pickup/drop-off location and the total amount charged but I wondered if it could be simplified by using the Boroughs.

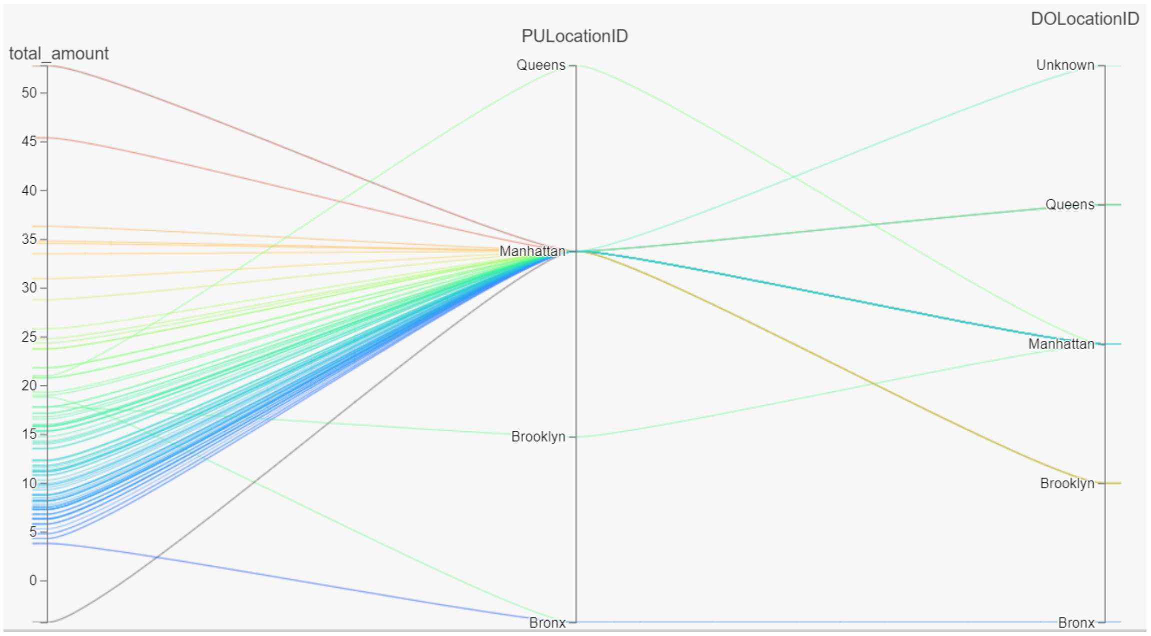


Figure 19: Alluvial plot regarding the pick-up and drop-off boroughs

It could be easily spotted that both least and most expensive trips routes lies entirely in Manhattan. It was obvious that some of the Boroughs are just disproportionately large. For a more in depth analysis it would be an asset to collect the data regarding boroughs size and population. In the next notebook was going to (and did!) engineer the feature "borough size" by simply counting the IDs for every Borough in the zone lookup table.

The pickups from zone other than yellow taxi zone was not taken into account as formally drivers cannot take passengers from other taxi zone. Because of that I expected some fare anomalies in these cases.

2.2 Pre-EDA notes

Matters that needed to be tended in the pre-processing:

- Dropping unused columns
- Cleaning missing values if any encountered
- Converting timestamp columns to the proper type and format
- Cleaning negative trip durations, trip distances
- Cleaning negative fares
- Cleaning the trips with no passengers
- Encoding the variables that should be categorical

Feature engineering:

- Calculate Borough sizes using number of zones that every one of them contains
- Calculate average drive speed and the drivetime
- Create the indicator whether it was a night/rush hour course
- Create a season indicator (for models trained on many months)
- Optionally merge the outlying payment types together into the "uncommon" category.

Features to keep from the original dataset:

- PULocationID and DOLocationID
- tpep_pickup_datetime and tpep_dropoff_datetime
- passenger_count
- trip_distance
- payment_type, fare_amount, extra, tip_amount.

Erase rows where:

- PULocationID or DOLocationID is in $\{0, 264, 265\}$
- Total amount is negative
- "extra" value is negative
- "tip_amount" is negative
- Trip lasts longer than 100 minutes or its duration is less than 1 minutes
- Erase rows with missing values.

3 Exploratory Data Analysis

It is rather a continuation of the previous section than a full separate EDA but it will offer a few additional insights, including yearly heatmaps. I was wondering if there would be some kind of trend visible from a yearly perspective. Moreover I wanted to at least briefly analyze which months forms clusters based on quantity of trips.

It seems that in times of transportation businesses growing in accelerated pace there is a strong competition for our beloved yellow cabs. Because of that, the idea that taxis will leave the ring unharmed is done in by the first touch of technological advancements. That is why it is crucial for them to automate their processes and optimize the trips.

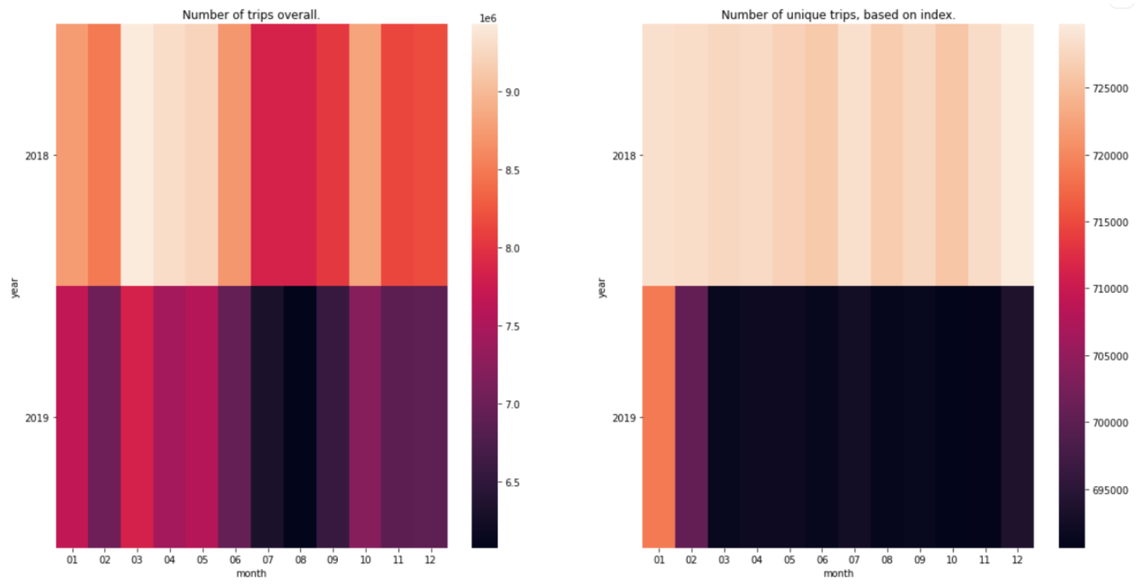


Figure 20: Heatmap of trips and index length

As we can see, the intuition built while roving around our data in the previous section was on spot. There is less and less trips over the years. It is true also for the unique trips. Now it is time to take a look at the cluster maps that I have produced during the EDA part. On the figure below the month clusters can be spotted. One cluster could represent months of spring while there is a lower number of trips in the high season. Generally the distribution of monthly trip numbers could be similar during the specific year but there is a clear decreasing tendency.

Then the EDA notebook performed calculations of the correlation matrix, without engineered features (aside of the drive time which is calculated in the preprocessing stage).

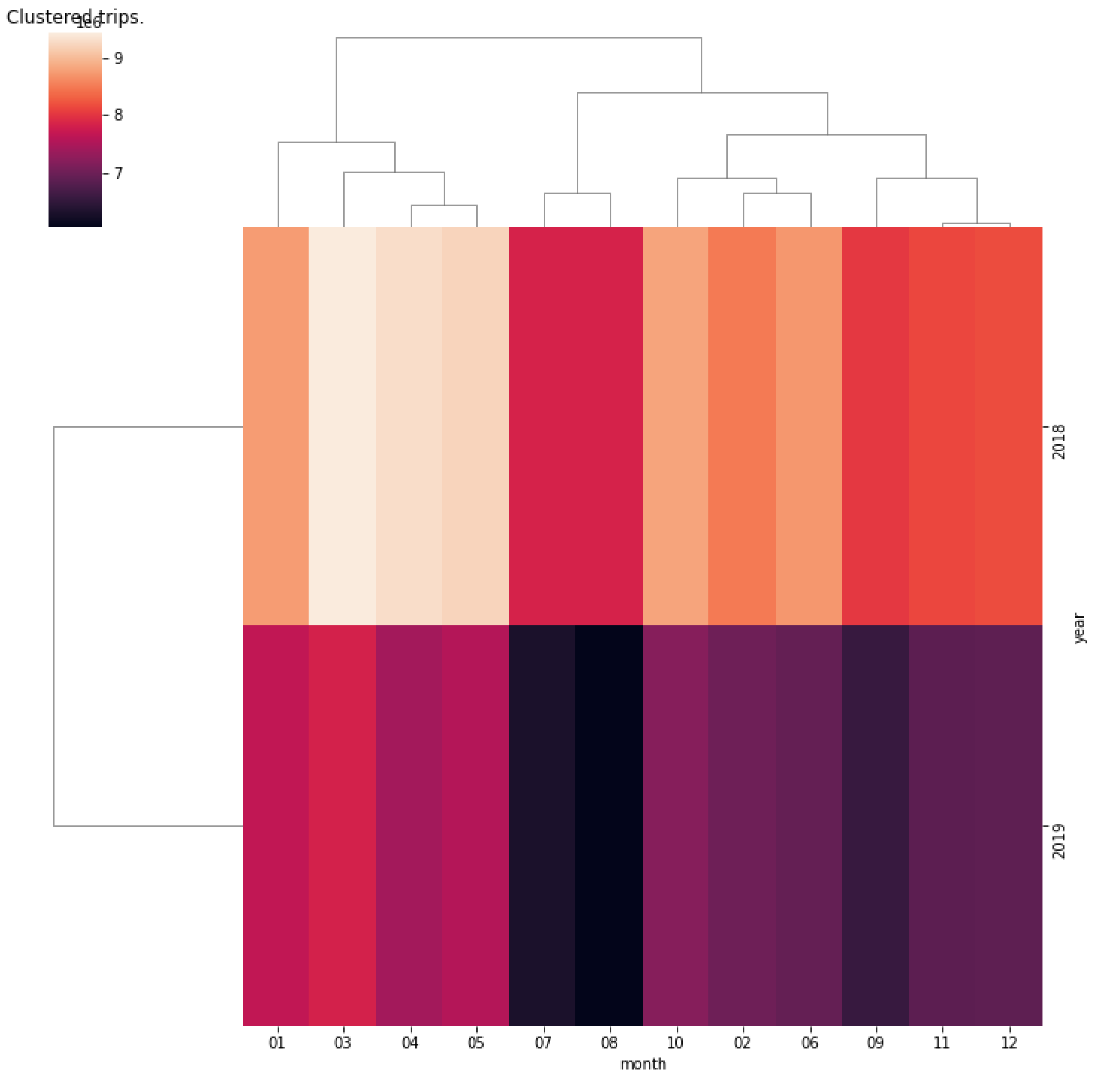


Figure 21: Clusters based on monthly trips

Then I would say that "extra" column was not necessarily used in a proper way. It was further processed to represent whether it had been a rush hour or a night ride (of what I thought of as the more compact approach than extracting this information from the timestamps). Before proposing any modelling approach I engineered some additional features like the type of trip, season indicator, pick-up and drop-down Borough size and the driving speed.

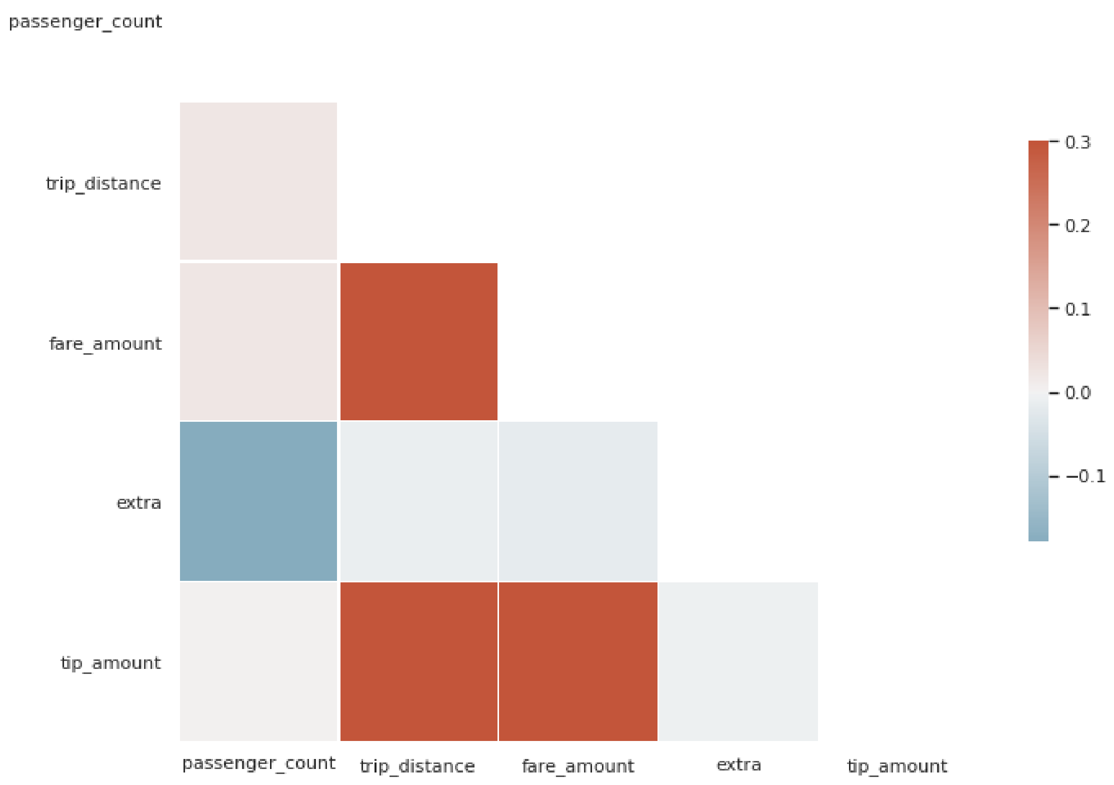


Figure 22: Correlations before feature engineering (except of the drive time)

After analyzing both correlation plots and remembering the domain understanding notebook results I had on my mind idea for two modelling approaches.
Draft ideas of modelling were:

- Trained on sister months (01.2018 01.2019, 02.2018 02.2019 etc.):
  - Predict the trip duration
  - Features: distance to drive, pickup/dropoff borough size, type of trip (night/rush hour/day), passenger count and optionally season
- Trained on whole dataset (but just on trips with card payments, for cash tips are not included in the data):
  - Predict tip amount
  - Features: distance, type of trip (night/rush hour/day), season, passengers count (after erasing the zero-passengers trips), pickup/dropoff borough size and optionally payment type.

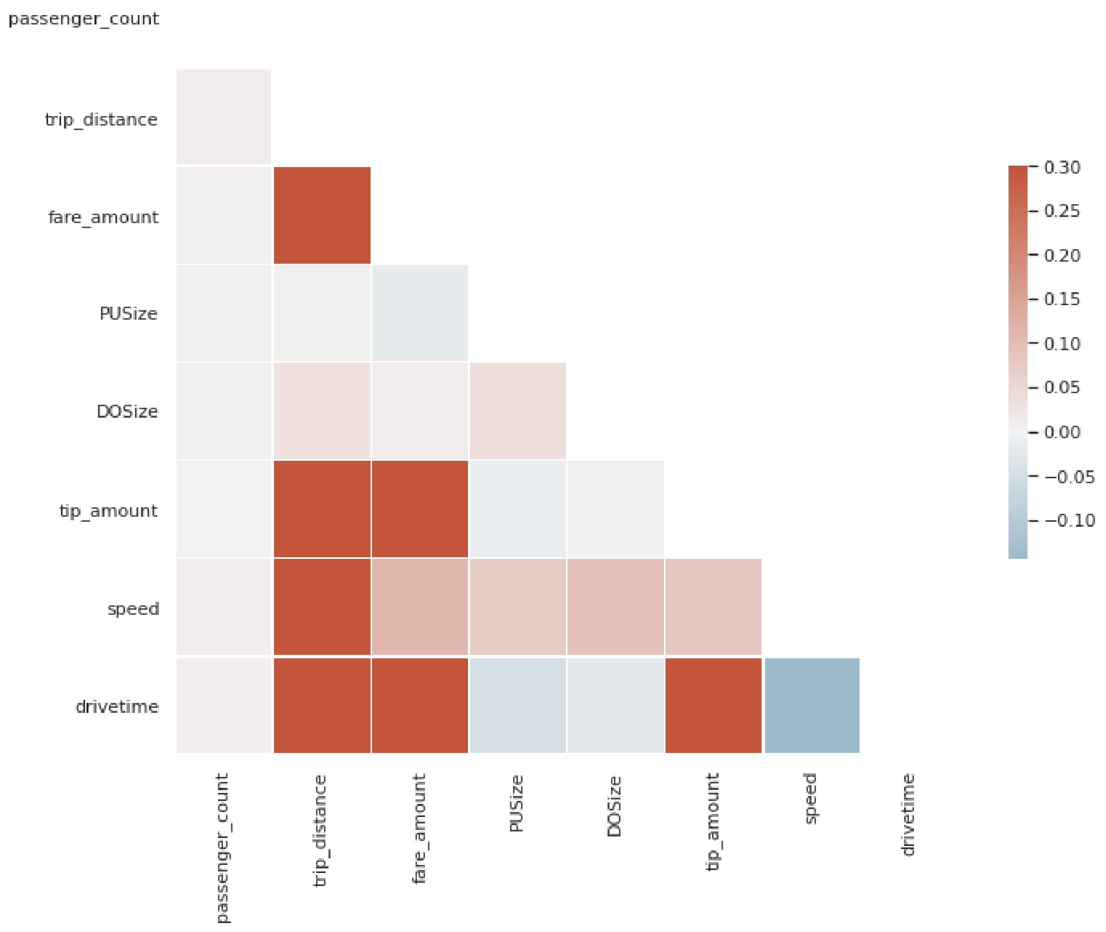


Figure 23: Correlation matrix with some of the engineered features included

3.1 Notes before modelling:

- Model 1) columns:
  - For scaling: passenger_count, trip_distance, PUSize, DOSize
  - For one-hot- encoding: trip_type, season
- Model 2) columns:
  - For Scaling: passenger_count, trip_distance, PUSize, DOSize, PULocationID (Optional), DOLocationID (Optional), speed
  - For one-hot- encoding: trip_type, season

4 Modelling

Finally the modelling part had came upon me, then it was time for choosing the problem to solve and the suitable architecture. I chose the first approached of the two that I had sketched while finishing the EDA part. After loading the batch of data, which was a bit over data points from 06.2019 and processing it I had the table with 5 columns at my disposal. Those column were telling us about the number of passengers, distance to travel, pick-up and drop-off Boroughs sizes and if it rush hour or night trip. That is what a driver usually learns in the very first moment after picking up passenger and I intended to predict the drive time based on this data.

I was using XGBoost regressor for predicting the drive time for its ability to estimate non-linear relations and robustness when it comes to the noisy data. Optimization goal was to reduce the mean squared error and the trained model was evaluated with the Root MSE (RMSE) at the end of every epoch both on training and validation set (0.9 and 0.1 train/test split).

Firstly I trained what I call the quick model, for 100000 epochs with early stopping after 100 epochs of no increase in the validation accuracy.. Then I measured the mean absolute percentage error (MAPE) just to have some intuitive measure of the booster performance. After that the feature importances were inspected just to show that the engineered features were relevant.

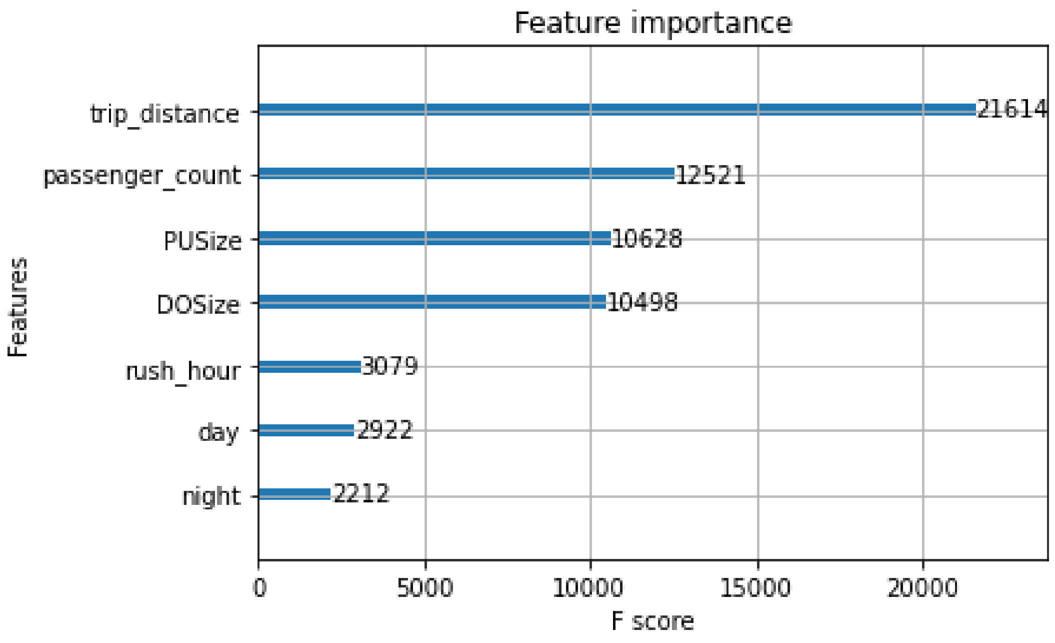
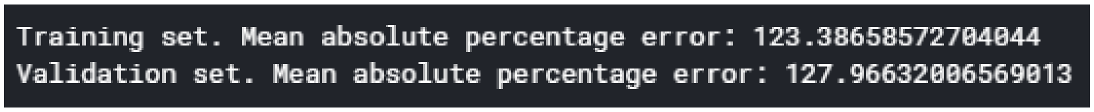


Figure 24: Feature importances: Quick model

Common as it is, the loss (as I call the RMSE evaluated residuals) sharply decreased in the first iterations, just to start flattening later on. While looking at the first plot, we could say that both losses hit a plateau, but I have added the zooming functionality to the evaluative plot so it would be possible to take a closer look at the losses. It could be seen that the training set performance still slightly improves over the epochs, hence the longer training could be expected to yield better performance although not necessarily.

I introduced MAPE mainly to have the information regarding whether the avoidable bias or high variance problem occurs (resp. underfitting or overfitting). It is problematic to approximate the Bayes Optimal Error in this case. Usually the human level performance is the proper estimator but it is true in the perception-based task not in remembering the high level rules present in the vast volume data. Deducing from the MAPE values, there is a overfitting issue.



```
Training set. Mean absolute percentage error: 123.38658572704044
Validation set. Mean absolute percentage error: 127.96632006569013
```

Figure 25: MAPE for the quick model

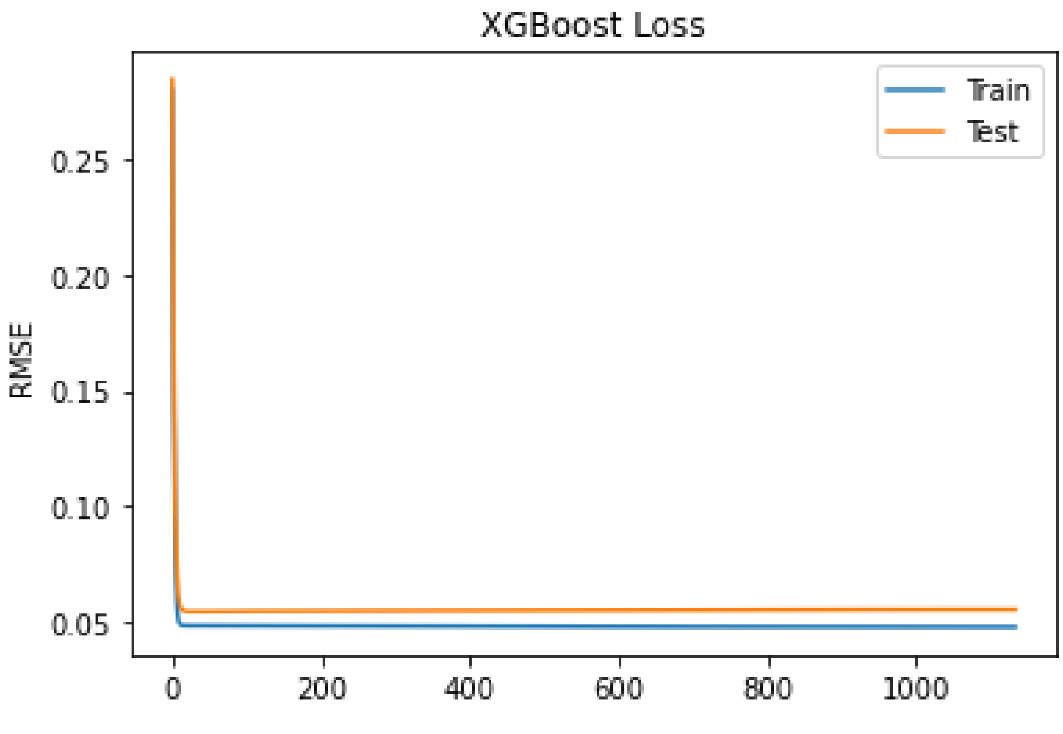


Figure 26: Quick model RMSE- based evaluation.

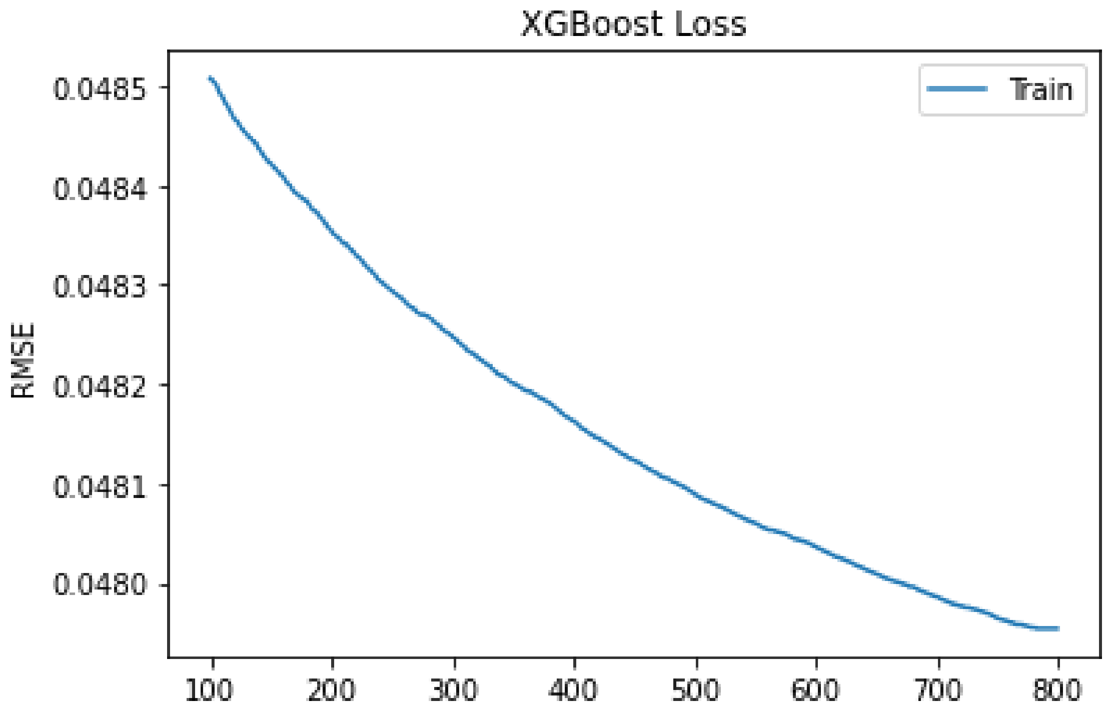
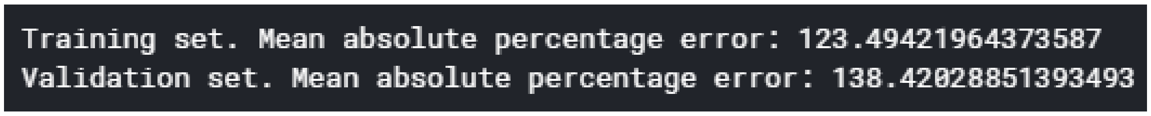


Figure 27: Quick model RMSE- based evaluation. Zoomed in on training set performance during the final epochs

I decided to train the model for a higher number of epochs (or rounds as creators of the XG-Boost call it) by practically switching off the early stopping callback. The results of evaluating MAPE showed that further improvement while using the chosen sample is probably not within my reach.



```
Training set. Mean absolute percentage error: 123.49421964373587
Validation set. Mean absolute percentage error: 138.42028851393493
```

Figure 28: MAPE of the prolonged model

Also the evaluative plot shows that there is no significant improvement of the model performance when training for more than 1000 epochs. It was time to leave the quick, default architectures behind and take care of parameters tuning. The random search was chosen for hyperparameters optimization

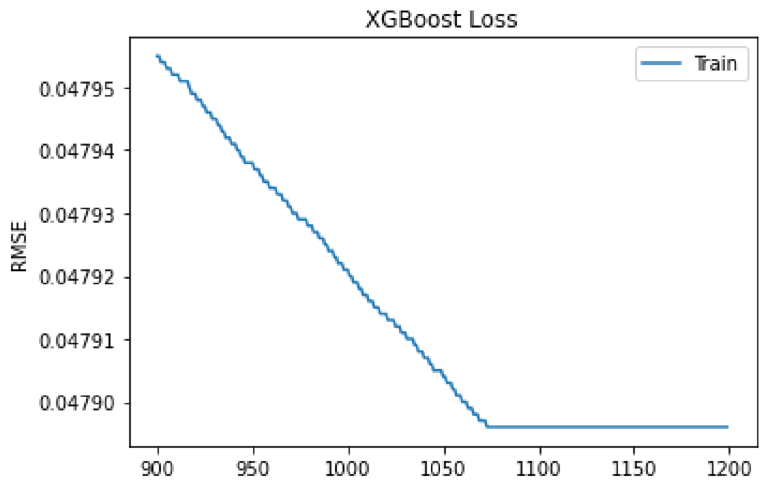


Figure 29: Training RMSE over epochs: Prolonged model, zoomed

The random search algorithm performed 1000 draws for each of them training the model using the parameter's variate drawn from the specified distribution. I used early stopping after 100 rounds and configured the seeking aim to be minimizing the validation set (or dev set) loss. Architecture chosen after that process will be called hereafter the optimal model.

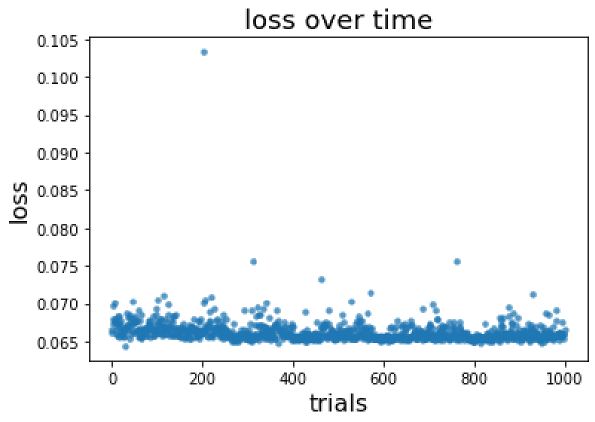
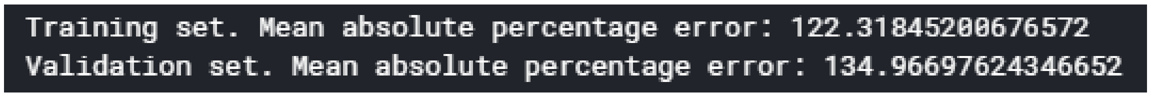


Figure 30: Random search process visualized

As expected the tuning lowered dev set MAPE, but surprisingly it had even greater impact on the training set performance. Still, while looking at the loss over trials plot that optimization could not yield spectacular improvement, for aside of a few outliers the loss variance is rather not startling.



```
Training set. Mean absolute percentage error: 122.31845200676572
Validation set. Mean absolute percentage error: 134.96697624346652
```

Figure 31: MAPE of the optimal model

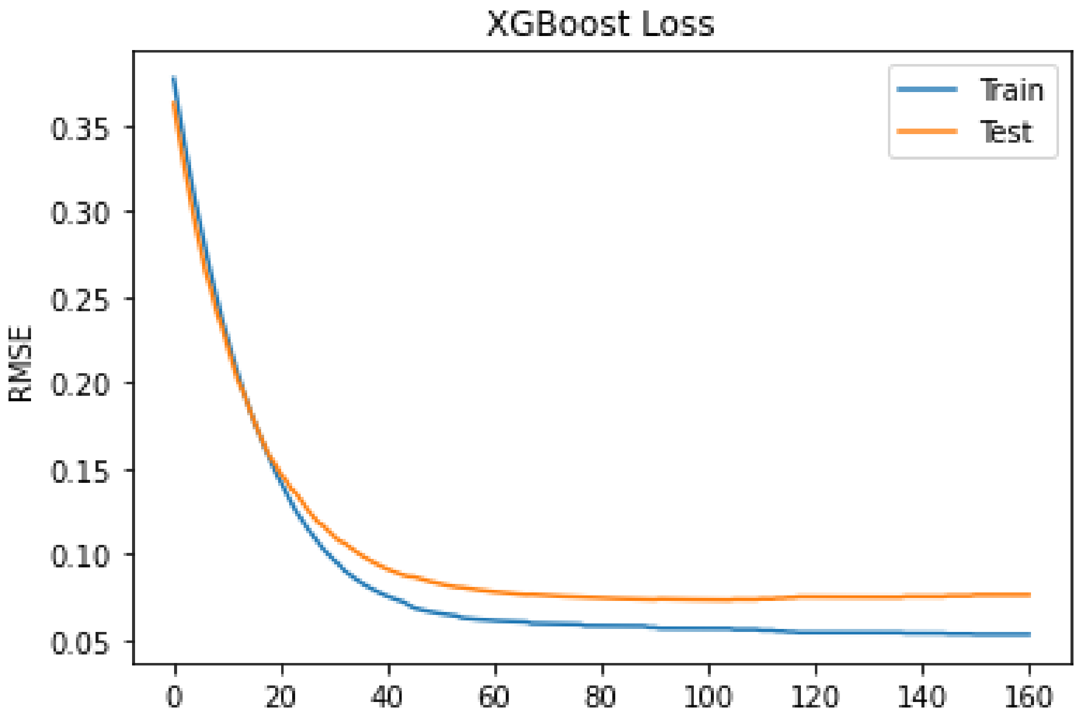
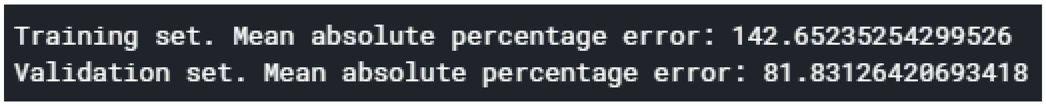


Figure 32: RMSEs of the optimal model

Then it was a time for an experiment. I decided to train a model picking one month from every 2019 season while using the engineered season indicator and specific pick-up and drop-off Borough instead of just their sizes. I thought that it might be possible for a booster to gain the spatial orientation in driving around New York City with the yellow taxi drivers. The batches of size 100000 were used and the Booster came out not to be the perfect choice for incremental learning. It is shown on the graphics below that the training was pretty unstable. It could be because of the batch size being too small. Also there is that low validation loss but it is due to the sample specifics, apparently it has an easier distribution than the training set.



```
Training set. Mean absolute percentage error: 142.65235254299526
Validation set. Mean absolute percentage error: 81.83126420693418
```

Figure 33: Bulk model MAPE

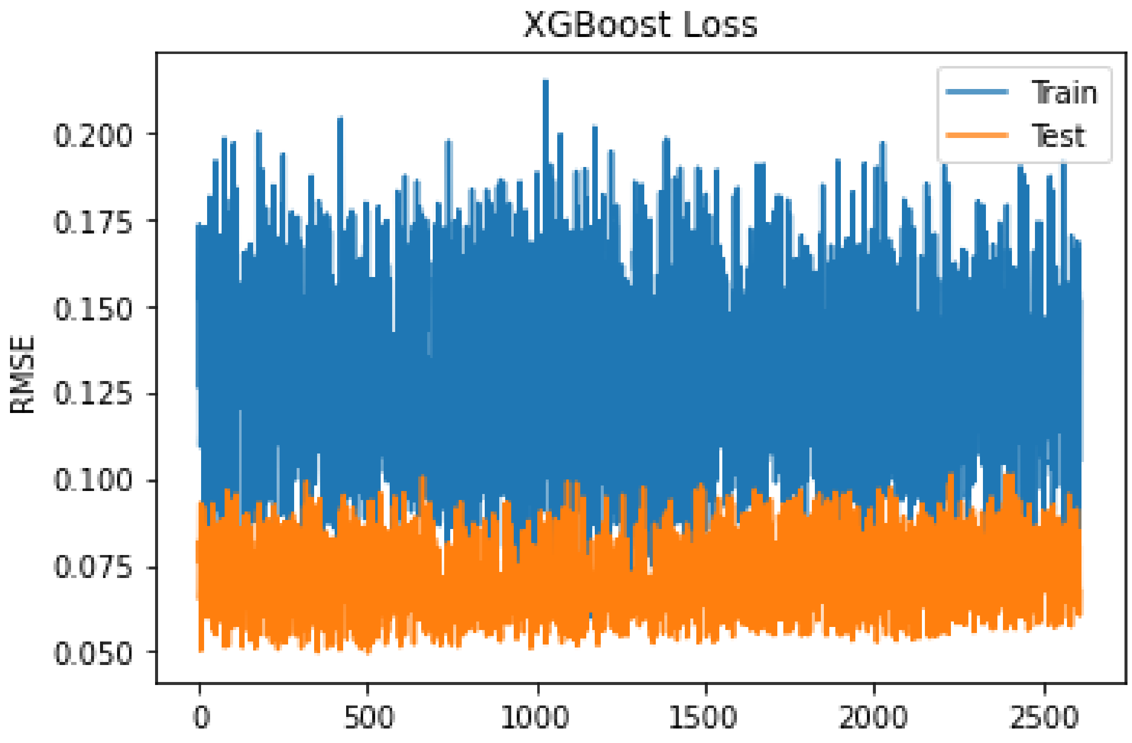


Figure 34: Bulk model evaluation

5 Summary

In a regular project, this analysis would be perceived as a pre-development stage. It offers the insight into what the data is, proposes some possible approaches to the modelling task, along with quick evaluations. Henceforth the next steps could be chosen.

Taking into account pick-up and drop-off locations the drive time prediction is highly nonlinear problem. That is why in the next stages I would recommend considering the use of neural network with specific locations, to learn the underlying map of NYC. Also training the XGBoost bulk model with the batches of larger sizes could yield better results.

Another idea would be to use Dask ML module which offers integration with XGBoost, Scikit-learn and Tensorflow libraries. I avoided that because of intention to present one of the methods for writing custom generators and using them in training. But it does not change the fact that this library is the great asset when it comes to working with large datasets.