University of Cologne
Faculty of Management, Economics and Social Sciences

Programming Data Science

Project Report

Written by:

Fenten, Julian <7364450>
Fiedler, Jan <7322954>
Pürlü, Selman <7336662>
Schwendner, Christian <7385458>
Weißhaar, Steffen <7325450>

Submission: 15.06.21

Content

List of Tables

List of Figures

Abbreviations

CW	calendar week
DO	drop off
DTR	decision tree regressor
HTML	hypertext markup language
KNN	k-nearest neighbor
MAE	mean absolute error
MLP	multilayer perceptron
MLPR	multilayer perceptron regressor
NYC	New York city
OLS	ordinary least squares
PU	pick up
RELU	rectified linear unit
RFR	Random Forest Regressor
RMSE	root mean squared error
SMOTE	Synthetic Minority Oversampling Technique
Std	Standard deviation
SVM	support vector machine

1 Executive Summary

Based on the NYC trip dataset we developed a python package named “Yellowcab”. The package was developed in the time span from April to June 2021 and provides methods to visualize and process trip data. As well as train models and predict certain features. Since we focused on 2020 the given data includes the beginning of the worldwide corona pandemic. The expected impact on mobility was analyzed and visualized. The reduction of trips after lockdown is clearly visible. After the analysis of the given data we trained promising prediction models for the ‘trip_distance’, ‘fare_amount’, ‘trip_count’ and ‘payment_type’. The finished package can be accessed via GitHub[1].

2 Problem Description

Transportation is an important aspect of our current society. As a strong support next to public transportation, taxicabs play an important role by conveying passengers between locations of their choice. Insights in their functioning and dependencies can offer various insights for companies, the government, and other stakeholders. The following analysis is based on a dataset of taxi rides within the city of New York [1]. The project is looking specifically at the year 2020 with a strong focus on New York's most populous borough Brooklyn. This includes the trips made inside Brooklyn, starting in Brooklyn, and starting in other boroughs but ending in Brooklyn.

The first result will be a well-documented GitHub project that can be imported and used, e.g. to analyze upcoming datasets of the same form or further analyze the given dataset. Furthermore, the most interesting insights will be shown and discussed in this report. To do so, the data will be prepared through filtering and adding columns, resulting from calculations or the integration of external data. In the exploration and visualization phase, several metrics, inspections, and analyses are visualized and documented. In this step, the Covid-19 pandemic of 2020, and its heavy influence on mobility in all areas is considered with particular attention. In the last step, machine learning models should be trained to test, if the prediction of important aspects is possible. Models will for example be designed to predict the distance of a trip, the amount of fares, the type of payment and the upcoming needed capacity.

For the project management, mainly the features of GitHub were used. Issues and tasks were assigned to the four milestones: Exploration and Description, Visualization, Prediction and Finish report. With an agile approach, members of the project team picked issues for themselves and worked on them. For coding tasks, branches were created that were later merged into the main code after successfully passing a review process. Each Wednesday and Sunday, the team met to present the status of their work, discuss problems, share ideas, and create new tasks. To comply with the requirements of the pandemic, all meetings were held via Zoom. For asynchronous communication, a slack channel offered the needed functionalities.

3 Data Description

As mentioned above, the dataset contains several million records of taxi rides within the city of New York [1] of the year 2020. These were provided in twelve parquet files, which had to be merged in a single file. These parquet files contain general information about trips with their start and end information as well as information about the payment. To enable the data preparation, it is firstly important to be familiar with the dataset and the meaning of all its current columns. The already existing 17 columns in the taxi dataset are explicitly named and described in Table A-1. The columns (‘tpep_pickup_datetime’, ‘tpep_dropoff_datetime’) include information about the time when the customer was picked up and dropped off at their target location. Due to privacy concerns the exact pick up and drop off location is not included. The location is therefore only roughly given by the columns ‘PULocationID’ and ‘DOLocationID’. PU denotes information about the pick up and DO denotes information about the drop off location. This column can be merged with the spatial dataset to get additional columns about the borough name, the assigned zone, and the service zone. Given the spatial data it is also possible to derive the centroid for each borough. The longitude and the latitude of the centroid

[1] https://github.com/lesar64/pds_brooklyn

can then be used as the location. For each trip, the distance travelled is given in the column *'trip_distance'*. The dataset also includes payment information like the fare amount (*'fare_amout'*) and tip amount (*'tip_amount'*). Information about taxes, tolls and other surcharges that can result from the use of certain routes, are split into different columns. There is also meta information included (*'stor_and_fwd_flag'*) that indicates wether the trip record was held in vehicle memory before it was sent. In later steps the data will be extended with several more columns.

As shown in Table A-2 and Table A-3, beside the main data set which provides the trip data, there are two additional data sets. The csv-file (*'taxi_zones.csv'*) gives information about the taxi zones in New York. The most important information in this file are the zone names because no other file contains this information. In addition to the csv-file, the JSON-file (*'taxi_zones.json'*) contains the associated spatial data about the taxi zones in New York.

4 Data Preparation

In the engineering process, we realized that we needed to prepare the data. Over the project, we added a broad range of new columns, to enable a better analysis or prediction. Furthermore, we identified aspects where the data needed to be filtered. Both these processes are described in this chapter. We implemented a method which automatically created a clean and full dataset and integrated all changes there, so that we always loaded the latest set of data. Therefore, working with old states of data could be avoided.

4.1 Adding Columns

Adding columns was an important step to enable the start of the exploration and visualization phase. These columns were either calculated from the given data or added by using external data. The columns that were added in the main process of data preparation are shown in the Table 4-1 below. These can be generally divided into three types. The first type contains temporal information. At first, the dataset just had one column containing the datetime information. This column was then split into its components like hour, day and month. We then also added columns for the weekday and weekend. The second type comprises spatial information. These columns were created by merging the *'PULocationID'* and *'DOLocationID'* for each trip with spatial data. The spatial data contains the centroid points (given through longitude and latitude) for each location id. We also added columns containing the absolute difference between the start longitude and end longitude (similar for latitude) which is used in the trip distance prediction (c.f. 7.1.1 Predicting the Trip Distance). Lastly a column stating the lockdown degree was added and used in the prediction for payment type (c.f. 7.1.3 Predicting the Trip Amount).

Furthermore, we decided that weather data can offer important additional information. Especially for the predictions (c.f. Data Prediction), aspects like temperature or precipitation can offer valuable support. We retrieved hourly weather data for Brooklyn from Visualcrossing[2]. The columns that could now be added were Maximum Temperature, Minimum Temperature, Temperature, Wind Chill, Heat Index, Precipitation, Snow, Snow Depth, Wind Speed, Wind Direction, Wind Gust, Visibility, Cloud Cover, Relative Humidity and Conditions. Still, we decided not to integrate the weather information in our main data preparation, as they need to be integrated manually for all trips of the upcoming future. The weather information that can be automatically merged in a taxi rides dataset of the respective form range from 01/01/2020 until 04/06/2021.

Table 4-1: Summary of the added columns

Field name	**Description**	**Data type**
'start_month'	The month of the start of the trip	Numercial
'start_day'	The day of the start of the trip	Numercial
'start_hour'	The hour of the start of the trip	Numercial

[2] visualcrossing.com

‘start_week’	The calendar week of the start of the trip	Numercial
‘end_month’	The month of the end of the trip	Numercial
‘end_day’	The day of the end of the trip	Numercial
‘end_hour’	The hour of the end of the trip	Numercial
‘end_week’	The calendar week of the end of the trip	Numercial
‘duration’	The duration of the trip	Numercial
‘weekend’	Binary Value, if the trip was on a weekend (1 = weekend)	Categorical
‘weekday’	Day of the week (1 - monday, 2 - tuesday, 3 - wednesday, 4 - thursday, 5 - friday, 6 - saturday, 7 - sunday)	Numercial
‘start_lat’	The latitude of centroid of the borough where the trip started (pick up)	Numercial
‘start_long’	The longitude of the centroid of the borough where the trip started (pick up)	Numercial
‘end_lat’	The latitude of the centroid of the borough where the trip ended (drop off)	Numercial
‘end_long’	The longitude of the centroid of the borough where the trip ended (drop off)	Numercial
‘long_dif’	The absolute difference between start_long and end_long	Numercial
‘lat_dif’	The absolute difference between start_lat and end_lat	Numercial
‘lockdown’	The lockdown measure status at the time of the trip start, with -1 to -3 as the lockdown regulations from loose to strict, 0 to 4 as the reopening phases 0 to 4 and 5 as no measures (pre-pandemic).	Categorical

4.2 Data Filtering

The next important step was to identify and handle data points with extreme or missing variable values. Due to the large dataset, we decided not to fill the values with dummy values and instead delete these entries. As the data does not represent a time series, but independent instances, there is no need to preserve continuity of the data. The small share of corrected data points would do more harm by distorting or impacting the results then there is a need for their preservation. To better visualize the extreme values we created a pandas profiling report, that is stored as an interactive HTML report[3]. The resulting filters that were applied are listed in detail in Table A-1. As the filters were designed based on the pandas profiling report, the practical impact was not always guaranteed, since the outliers caught by one filter sometimes overlapped outliers caught by another, especially if both features were heavily correlated (e.g., payment data like *‘fare_amount’* and *‘total_amount’*). For the sake of completeness and future proofing, those on first-sight “redundant” filters were not discarded.

There were three noteworthy categories of outliers. First, duration, *‘fare_amount’* or *‘passenger_count’* sometimes equaled 0, which would not be considered a valid trip. In some cases the trip also ended before it started. These trips are also invalid. Then there is no matching data for locations with ID above 263, which affects the IDs 264 and 265 in the given geojson file (c.f. Data Description). Furthermore the given geojson

[3] https://github.com/lesar64/pds_brooklyn/blob/main/data/output/report.html

has duplicates in the row *'location_id'*, we instead used the '*object_id*'. Probably the most interesting case was that some trips had extreme values for *'duration'*, with a high spike close to 24 hours visible in Figure 4-1. The right graph shows the overall distribution of the *'duration'*, while the left graph shows the spike at 86.400 seconds equal to 24 hours. Note that the right graph uses a logarithmic scale.

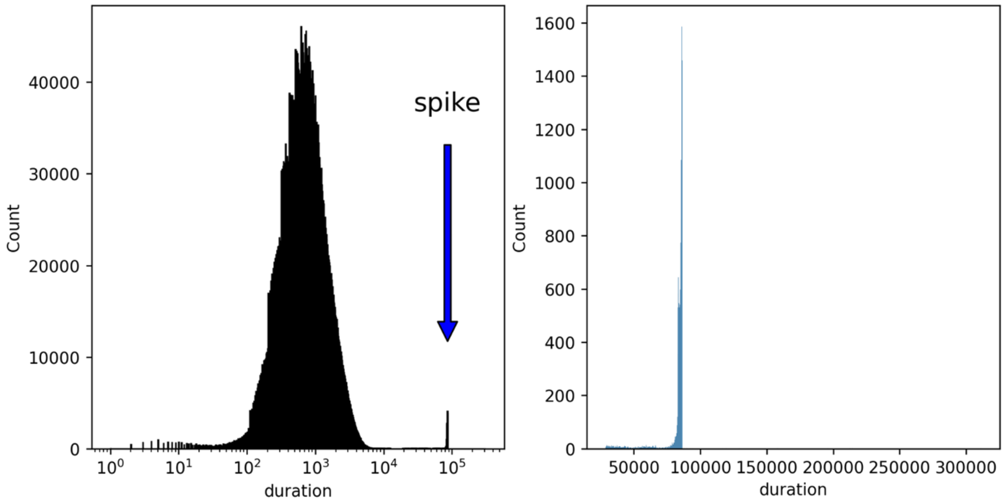

Figure 4-1: Histogram on logarithmic scale (left) and spike at 24h

This can be attributed to two errors. The first one being that the pick up and drop off time were close to 24 hours apart, minus an expected trip duration. This can probably be chalked up to a switch up of both times while logging the trip, with the system moving the now earlier drop off time to the next day to preserve the logical sequence. The second error is a number of trips with (mostly) the drop off or (rarely the) pick up time exactly on the second at midnight (c.f. Figure 4-2), while the other time is nowhere near, resulting in an extremely high duration. The most likely case being that the time, but not the date is missing, which results in the logged time being set to midnight.

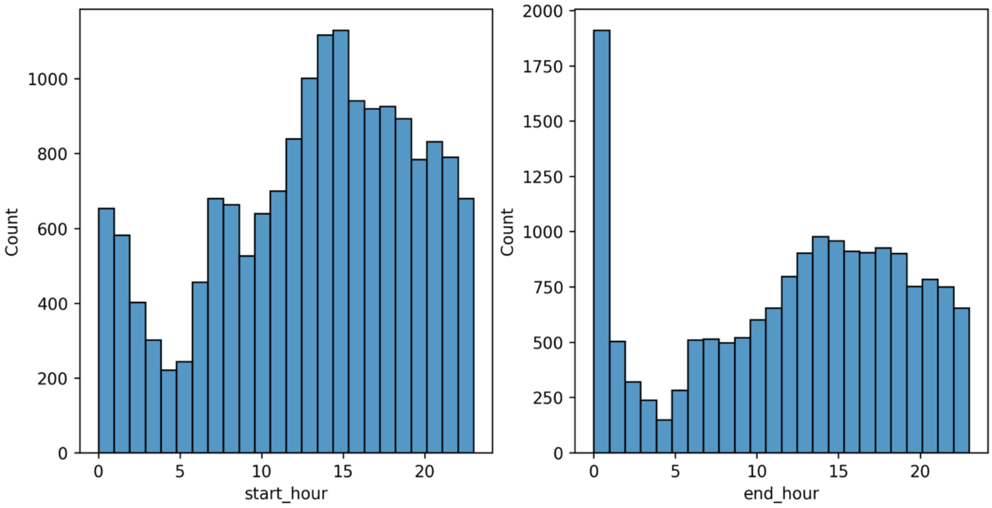

Figure 4-2: Histogram for hourly trips

5 Data Exploration

5.1 Overview of LocationID

To get an overview about the distribution of the LocationIDs in Brooklyn, the taxi zones were shown on map. In the map below (c.f. Figure 5-1), it is easy to see that the zone size differs a lot. Therefore, later in the analysis the ratio of size to number of trips should be analyzed. Furthermore, the map shows that some taxi zones do contain main arteries and some don't. This could possibly lead to a higher amount in such zones, where the traffic amount is generally higher throughout the main arteries. The impact of this distribution will be shown in the heatmaps, which will visualize the trip amount per taxi zone.

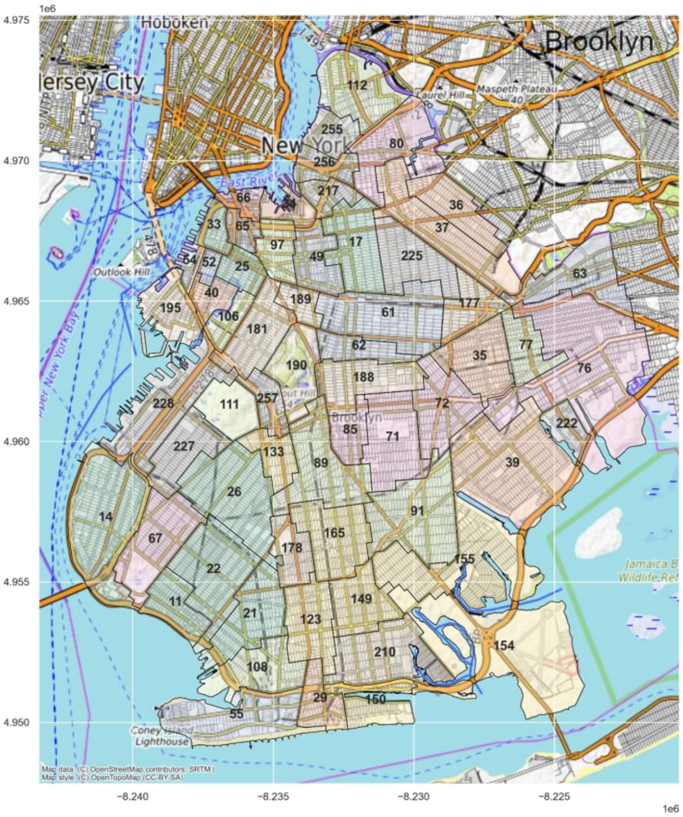

Figure 5-1: Overview of LocationIDs

5.2 Statistics

5.2.1 Trip Duration and Trip Distance

An interesting aspect of the statistics was the closer look at the trip lengths (*'duration'*, *'trip_distance'*) of the taxi rides. As a result, the mean and the standard deviation per month, day and hour were plotted (Figure A-4 and Figure A-5). Considering the plot for the trip duration per month, one could see that the average duration of a trip was clearly lower in the months of April and May (shortly after the first lockdown). They rose again for the months of June until December, but still did not reach the mean of the months of January to March. Considering Figure 5-2 for the trip distance per month, one could say that the median trip distance keeps almost the same over the months. There are slight decreases in the first lockdown in April. In the following months of reopening from April to June, the trip distance starts to increase again, but slowly decreases towards the second lockdown in November.

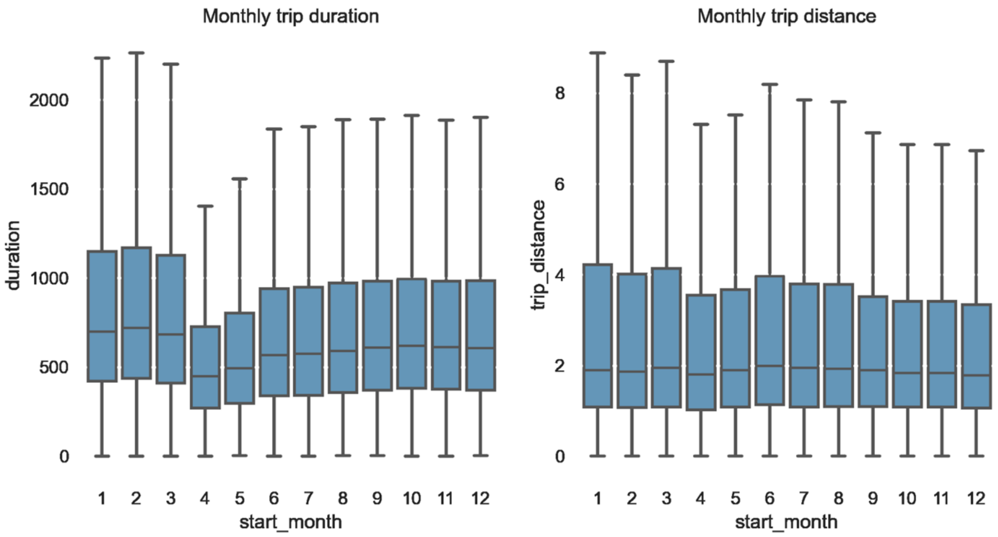

Figure 5-2: Monthly trip duration/ distance

Next, one could say that the median trip duration has its peaks at 5 a.m. and a smaller peak at 4 p.m. (c.f. Figure 5-3). The trip duration peak at 5 a.m. could be that high because one could assume that this is the time where people wake up and get ready to drive to their jobs to start at 8 a.m. Also, the variance of the trip duration at 5 a.m. seems to be higher than normal. These high trip durations may also be caused by high trip distances at 5 a.m. (c.f. Figure 5-4). Furthermore, the trip distance might be the highest at this point, due to faster taxi transportation in that pre rush hour time frame. Less traffic makes it possible to use taxis for longer distances, without getting stuck. In addition, the overall variance in the hourly data distributions (c.f. Figure 5-3, Figure 5-4) might be high and scattered because the data set consists of data collected before and after the Covid-19 situation.

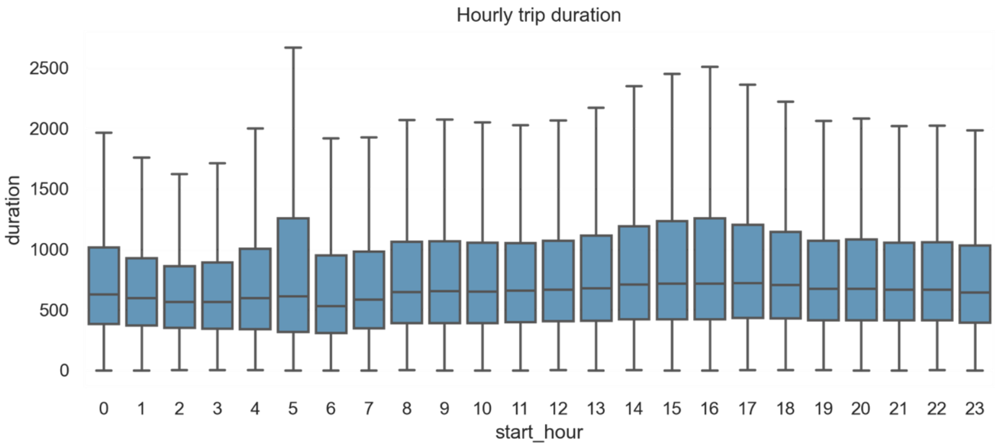

Figure 5-3: Hourly trip duration

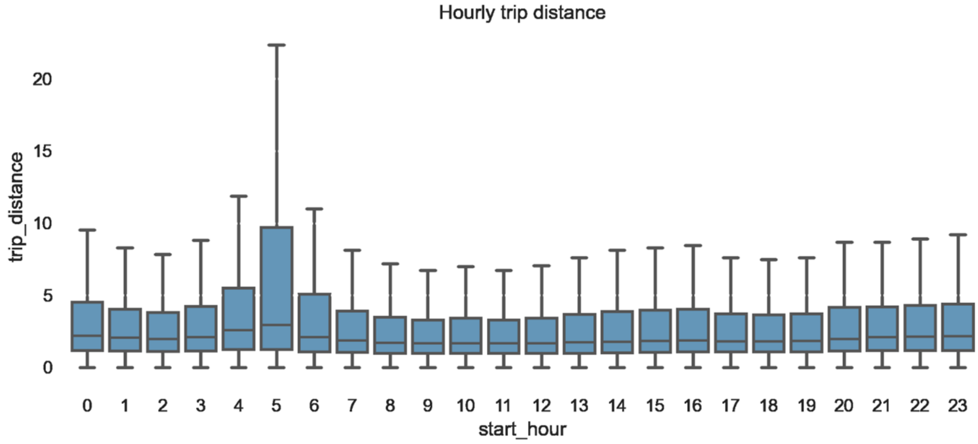

Figure 5-4: Hourly trip distance

5.2.2 Trip Length Distribution

The trip length can either be quantified as the duration (time) or the distance (way) of the trip. For a more precise analysis, the distance was used to look at the trip length. The distribution of the distance (blue color) was visualized and compared to the normal distribution (orange color) with the same mean and standard deviation (c.f. Figure 5-5). As a result, one could see that especially short trips occurred with a much higher density. Apparently, trip distance is not normally distributed due to the huge density spike difference in the first units of trip distance.

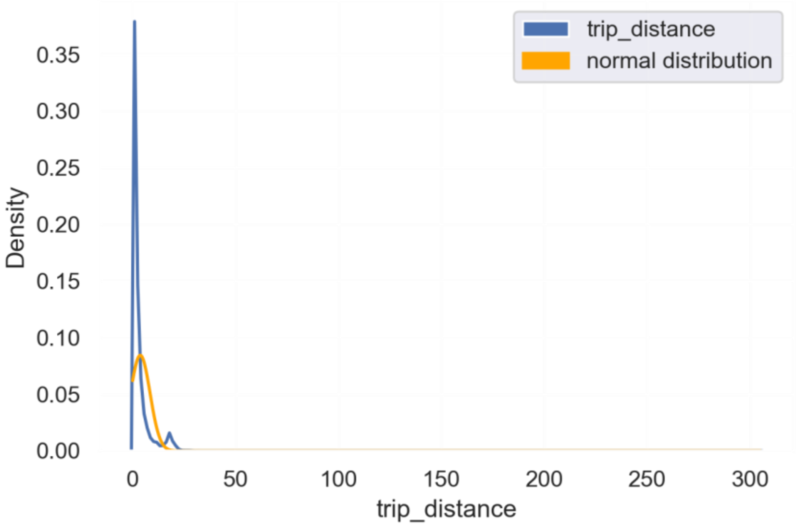

Figure 5-5: Overall distribution of trip distance

Plotting the trip data on a logarithmic scale results in the usual bell curve known from normal distributions (c.f. Figure 5-6). The histograms show a high amount of short trip distances and a high variance in trip lengths. Additionally, the same analysis and results also applies to the monthly distribution of the trip duration (c.f. Figure A-2, Figure A-3).

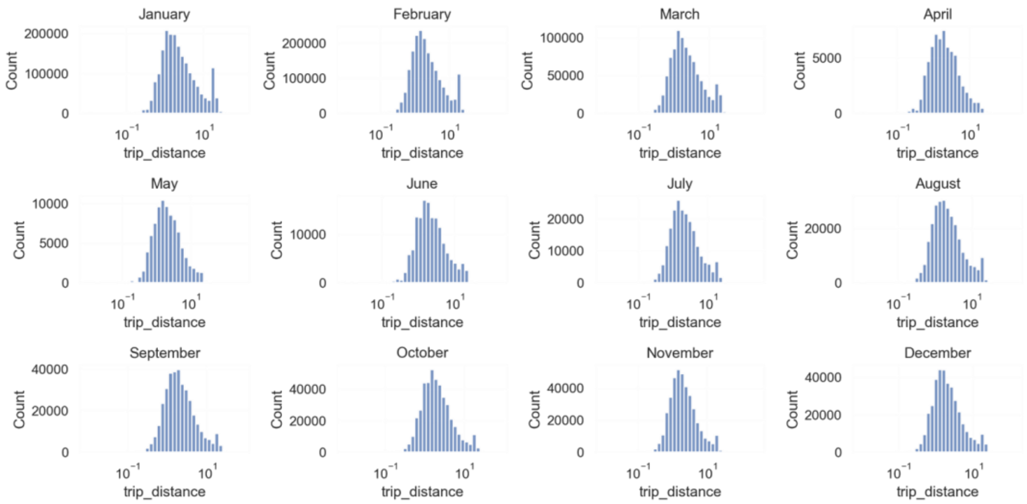

Figure 5-6: Monthly distribution of trip distance

5.3 Corona Exploration

The Covid-19 pandemic had a strong influence on the taxi rides in New York City in the year 2020. For a first overview, the amount of Covid-19 cases were interesting to look at. Therefore, in the figure below (c.f. Figure 5-7), the daily new Covid-19 cases in New York City were plotted. The cases rose fast in March and April, but declined over the summer months. At the end of the year, the cases started rising again.

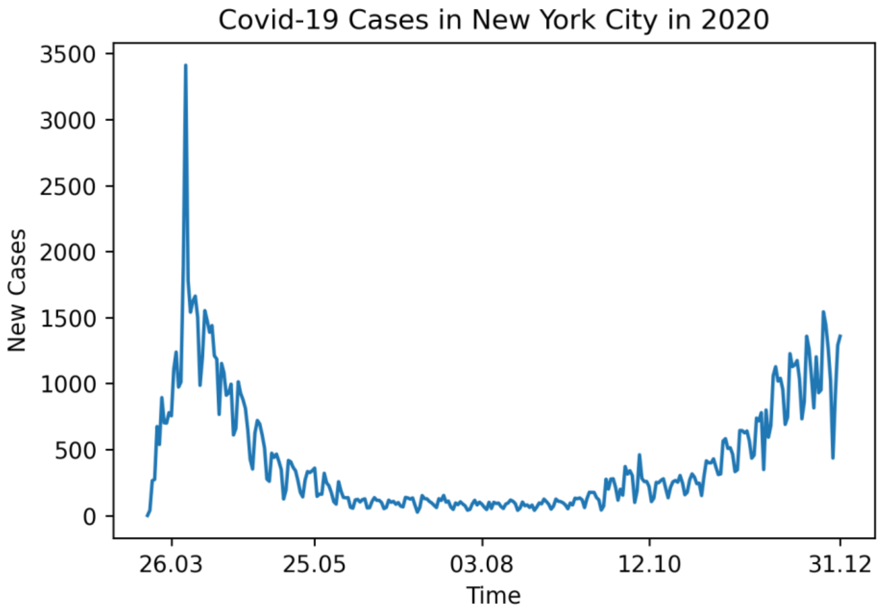

Figure 5-7: Covid-19 cases in NYC

To start with the analysis, it was important to further research the measures and policies taken by the government (c.f. Table 5-1). There were lockdowns in March and November, as well as reopening measures in May and June. These measures correlate with the new Covid cases.

Table 5-1: Lockdown and reopening measures

Lockdown and Reopening Measurement, concerning NYC [2] [3] [4] [5] [6] [7] [8] [9]		
Date	**Measure**	**CW**
March 12	**Lockdown**. Cancelling of all gatherings of more than 500 people. Gatherings of less than 500 people were ordered to cut capacity by 50%.	11

March 15	**Lockdown.** Closing all New York City schools and universities.	11
March 20	**Lockdown**. State-wide stay-at-home order declared. All non-essential businesses ordered to close. All non-essential gatherings are cancelled.	12
April 15	**Ordering** all state residents to wear face coverings in public places where social distancing is not possible.	16
May 15	**Reopening** drive-in theatres, landscaping, and gardening businesses state-wide. Regions that met qualifications (Also NYC) could open even more.	20
June 8	**Reopening** Phase 1. Reopening of construction, manufacturing, agriculture, forestry, fishing, and select retail businesses that can offer pick up.	24
June 22	**Reopening** Phase 2. Reopening of outdoor dining at restaurants, hair salons and barber shops, offices, real estate firms, in-store retail, vehicle sales, retail rental, repair services, cleaning services, and commercial building management businesses.	26
July 6	**Reopening** Phase 3. Reopening basketball, handball, tennis, bocce, and volleyball courts. Also, personal care salons, spas, and tattoo parlours.	28
July 20	**Reopening** Phase 4. Reopening Higher Education, low-risk indoor and outdoor Arts and Entertainment, Media Production, Professional Sports Competitions with no fans, gyms, and malls	30
August 19	**Ban** on ticketed music events at bars and restaurants.	34
October 6	**Implementing** a "micro-cluster strategy." The new plan places new restrictions in cluster areas that have spikes in Covid-19 cases.	41
November 12	**Lockdown.** Closing of bars, gyms, and any other business with a liquor license after 10 p.m. (restaurants as well, except for pick up). Household gatherings limited to ten people.	46

After examining the measures taken, the next step was to analyze how they influenced the taxi industry. The plot of total passengers per calendar week (c.f. Figure 5-8) gave a good overview on how impactful the lockdown measures were on transportation. The calendar weeks marked in red are weeks, in which lockdowns occur. The calendar weeks marked in green are weeks, in which reopening measures occur. One could see that the total count of passengers fell extremely fast after the first lockdown measures and slightly rose after the opening measures.

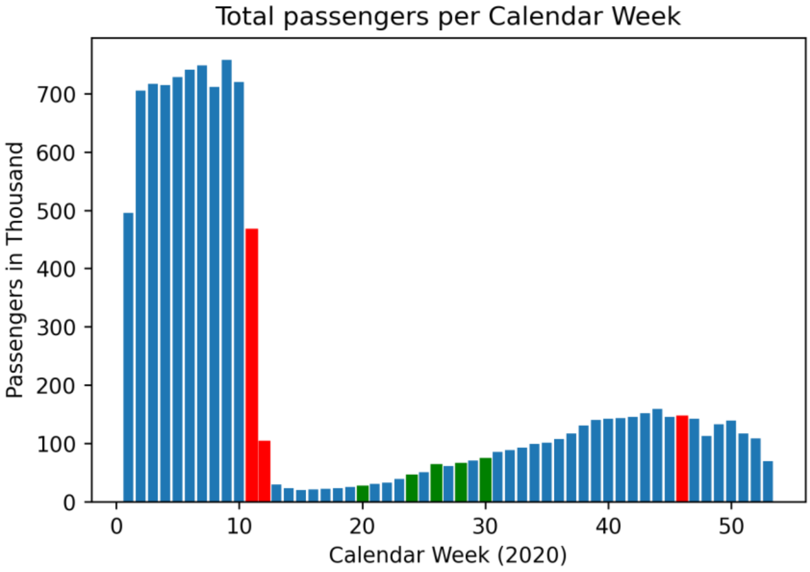

Figure 5-8: Passenger count per calendar week

To look at this specific impact, we also visualized the total duration per day in March 2020 (cf. Figure A-1). Furthermore, the impact of the lockdown on variations between hours (e.g., rush hour) can be seen. In the following plot (c.f. Figure 5-9), we compared a week before the first lockdown with a week after the first lockdown. 8am and 5pm are marked with a star. After the lockdown, there are not the usual peaks in the rush hour and the curve is flatter. This could be a result of the high number of companies that integrated home-office.

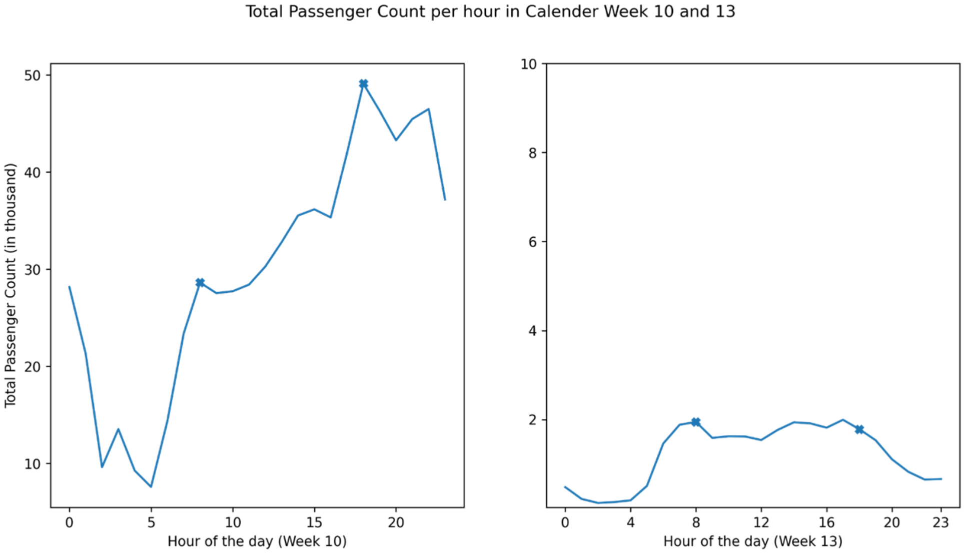

Figure 5-9: Passenger count per hour (CW 10 and 13)

6 Data Visualization

6.1 Trip Count

As needed for later steps, the trip count per hour was printed to get a brief overview about the impacts from the coronavirus pandemic (c.f. Figure 6-1). As shown in the previous Covid, the impact on the total passenger numbers correlates clearly with the beginnings of the lockdowns, mainly with the first one. Through the visualization below it can be clearly seen, that the trip count follows the same trend as the passenger amount. In the middle of June respectively calendar week 11 after the first lockdown regularities came into effect, there is a clear drop of the trip count in Brooklyn.

Independent from the coronavirus pandemic, it would be interesting to know which variables do impact the trip count in Brooklyn. Main factors could be the weather and the time. As proven through the figure below, there is a cycle, which scale is probably weekly. This is getting analyzed in a further prediction chapter.

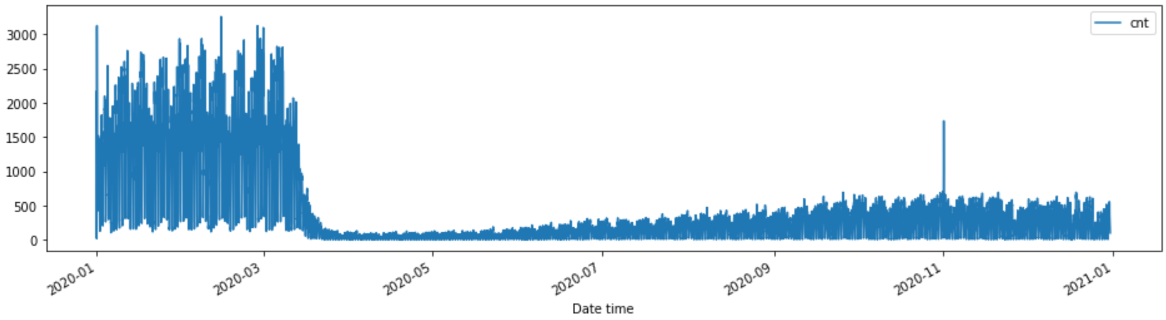

Figure 6-1: Trip count over the year 2020

6.2 Heatmap

As January was the month with the most taxi trips, it gives a good overview about the distribution under ‘normal’ conditions without the coronavirus. In numbers, there were 1,933,255 trips, which is about 26,2% of all trips in the whole year. With only 23,045 trips less than in January, the second highest amount was driven in February. If the trip count in January and February is added, it makes more than 52% of the trips of the given year.

To get more insights about the spatial distribution and hotspots in Brooklyn, the heatmap was created to visualize different metrics, which are available within a group_by-function. As a starting point, each month was put into the visualization. Furthermore, all metrics for all months were created and analyzed. The result of this analysis shows that the drop off hotspot in Brooklyn is placed around the Barclays center, which is located in the taxi zone with the location id 189. This spot is a multi-purpose arena where sport events, but also cultural events take place[4]. In the pre-corona map (c.f. Figure 6-2) it is shown that about 8,000 to 14,000 trips ended there, so this is probably a famous destination in Brooklyn.

Under corona conditions (c.f. Figure 6-3) it is clear to see that the absolute amount of trips dramatically decreased by a factor of ten. This shows the different scaling's of the heatmaps. Furthermore, the hotspot which was located surrounding the Barclays center, has decreased and the trip location is more distributed onto multiple zones in the north and north east of Brooklyn, where more residential units are placed. As mentioned in 5.1 Overview of LocationID, the size of a taxi zone does not have a big impact on the trip count pre corona. But as shown in the corona conditions map, the influence got bigger, most likely due to the higher total number of people who are living there and the decreasing relevance of event locations and office buildings.

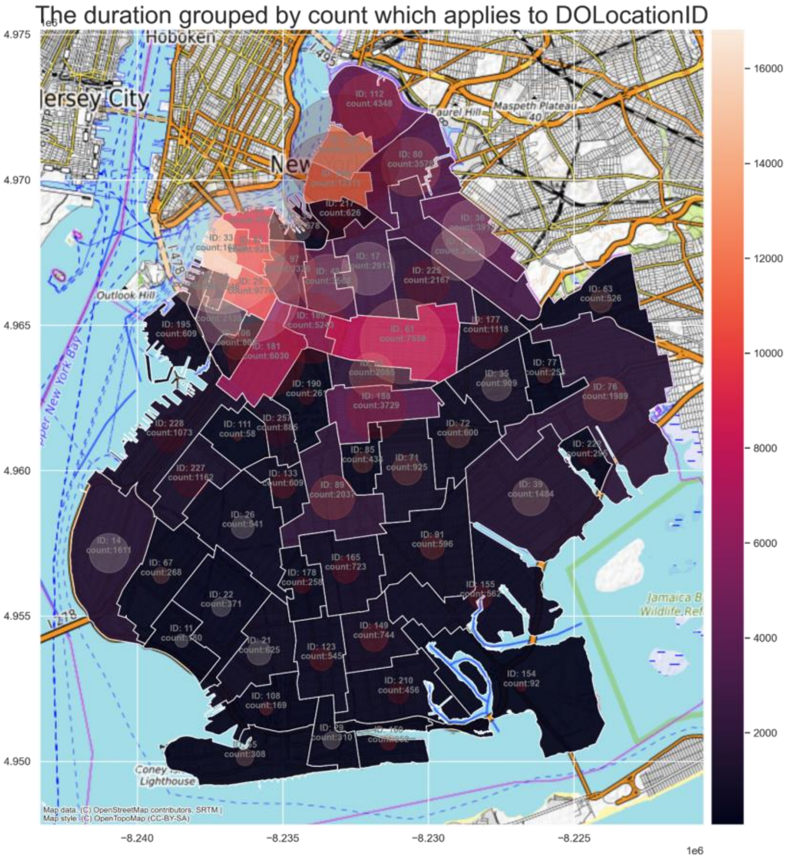

Figure 6-2: Pre-Covid conditions (from 2020-01-01 to 2020-01-31)

[4] https://www.barclayscenter.com/center-info/about-us

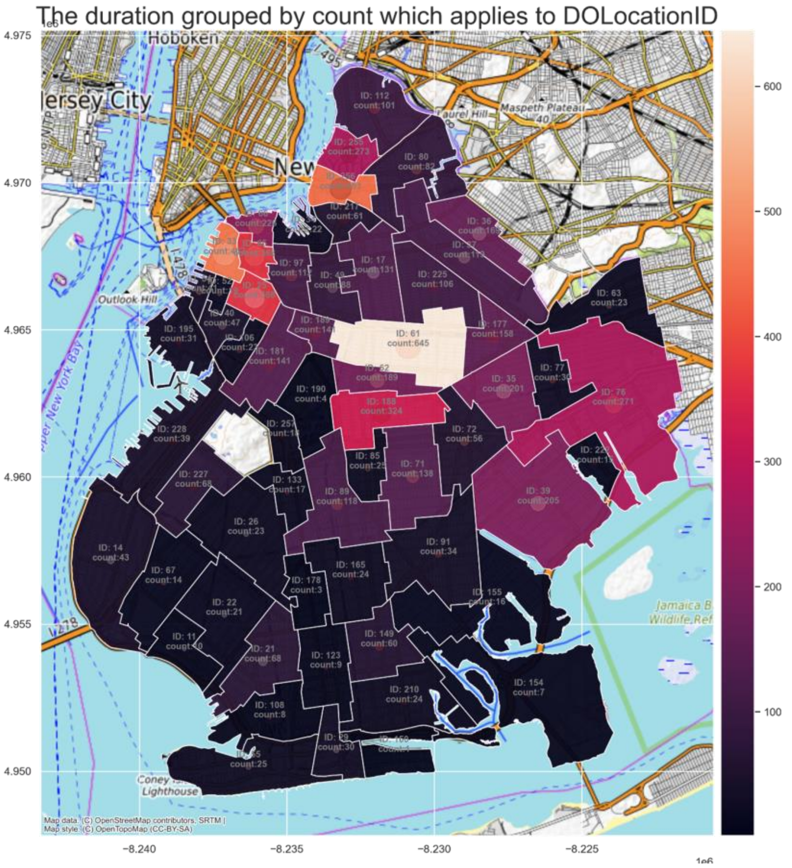

Figure 6-3: Under Covid conditions (from 2020-05-01 to 2020-05-31)

7 Data Prediction

Based on the given data we will train and compare different predictive models. As for regression we will predict the trip distance, fare amount and trip count. The regression model for the trip distance is based on information available at the start of the trip. The goal for the fare amount prediction is a parsimonious model based on start information and end information (time and place). For the trip count we will try to find a correlation between the weather and time data and the trip count. We will also classify the payment type based on all available information. All models are trained on the filtered and cleaned dataset given through the data cleaning process. This dataset only includes trips that were made in Brooklyn, start in Brooklyn or end in Brooklyn. The dataset totals around 7.3 million samples (unique trips). To train our models we used the sklearn library, pytorch and pytorch lightning.

7.1 Regression

7.1.1 Predicting the Trip Distance

Given the task to predict the trip distance based on the information available at the start of the trip we first need to clarify which features are used for the training. Since it is not clear which information can be considered to be available at the start of the trip we will discuss two scenarios. For the first scenario we assume that only passenger count, start time and the start location are known. For the second scenario we assume that the target location is also known. The target location can be seen as available at the start of the trip, since in the given setting the customer will tell the driver where to drive at the start of the trip. The resulting columns are shown in Table 7-1. For the second scenario we add *'DOLocationID'*, *'DOBorough'*, *'DOservice_zone'*, *'end_location_long'* and *'end_location_lat'* to the known features. It is important that due to privacy concerns the columns longitude and latitude (*'start_location_long'* and *'start_location_lat'*) are connected to the *'DOLocationID'* and do not represent the actual drop off location (the same applies to

‘PULocationID’ and the pick-up location). Based on these scenarios we will try to achieve the best performance separately.

Table 7-1: Known features for predicting the trip distance

Scenario 1	Scenario 2 - adding target location
● *'passenger_count'* ● *'PULocationID'* ● *'start_month'* ● *'start_day'* ● *'start_hour'* ● *'start_week'* ● *'weekend'* ● *'weekday'* ● *'start_location_long'* ● *'start_location_lat'* ● *'PUBorough'* ● *'PUservice_zone'*	● *'passenger_count'* ● *'PULocationID'* ● *'start_month'* ● *'start_day'* ● *'start_hour'* ● *'start_week'* ● *'weekend'* ● *'weekday'* ● *'start_location_long'* ● *'start_location_lat'* ● *'PUBorough'* ● *'PUservice_zone'* ● *'DOLocationID'* ● *'DOBorough'* ● *'DOservice_zone'* ● *'end_location_long'* ● *'end_location_lat'* ● *‘long_dif’* ● *‘lat_dif’*

Selecting features from the given dataset we also had the option to use *‘PUZone’*. The *‘PUZone’* is part of the information available at the start of the trip and also applies to the first and second scenario. We decided not to use *‘PUZone’* for the training, because the *‘LocationID’* (DO and PU) and the *‘PUZone’* are highly similar. After looking at the unique values for *‘PULocationID’*, *‘PUBorough’*, *‘PUZone’* and *‘PUService_zone’* the number of unique values of *‘PUZone’* and *‘PULocationID’* is off by one. After looking at the specific values it turned out that the IDs 56 and 57 are the same *‘PUZone’* - Corona. In the second scenario instead of using the longitude and latitude of the start and end location we decided to use the absolute difference respectively for longitude and latitude. The features *‘start_location_long’* and *‘end_location_long’* result in one new feature *‘long_dif’* (same for the latitude). The other case with two features for start and end location (four total) was tested in the beginning, but was dropped due to lower performance.

As possible regressors we considered linear regression (with Lasso and Ridge regularization), decision tree regression, k-nearest neighbors regression, support vector regressor (SVR) and a multi-layer perceptron regressor. The k-nearest neighbors and the support vector regressor were dropped in the beginning due to performance issues. “[T]he fit time complexity [for SVR] is more than quadratic with the number of samples which makes it hard to scale to datasets with more than a couple of 10000 samples” [10]. The SVR is therefore not feasible for our approach. The k-nearest neighbors regressor was dropped due to similar concerns [11].

After the first steps we realized that our linear models had problems capturing the complexity of the data. To improve the ability to do so, we decided to add polynomial features. We expected that this would help the linear models. The polynomial features were created using only numerical features. Categorical features are not used to create polynomial features. This results from the nature of categorical features. They can only be 0 or 1 and will only increase the number of polynomials without adding value. The resulting polynomial Features were then scaled using the RobustScaler from Sklearn. The categorical features were one-hot encoded. The remaining models were then trained on a subset of our samples. In the next step we determined the degree of the polynomial and the regularization parameter alpha (for Lasso and Ridge). To determine the hyperparameters of our linear models we used gridsearch on a subset with 160.000 samples with 5-fold cross validation (c.f. Table A-5, Table A-6)[5]. The non-linear models were trained on the same subset. The linear models with hyperparameters derived through gridsearch[6] and the non-linear models were then tested on a random subset with 40.000 samples. This corresponds to a 80/20 (train/ test) split. The results for scenarios one and two are shown in Table 7-2 and Table 7-3 respectively.

Table 7-2: Scenario 1 results on training subset[7]

	DTR		**RFR**		**MLPR**		**Lasso**		**Ridge**		**Lasso_log**	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
RMSE	1.82	4.40	2.06	3.6	3.17	3.31	3.28	3.28	3.28	3.27	4.69	4.69
MAE	0.76	2.61	1.19	2.23	1.98	2.06	2.02	2.02	2.02	2.02	3.59	3.59

Comparing the results of the given scenarios the errors for the first scenario are higher than in the second scenario. Having information about the end location drastically improves the predictive power of the models. On the given subset of 160.000 samples the trained linear models seem to perform pretty well compared to the other models (c.f. Table 7-2). This can be attributed to the fact that we mainly focused on the hyperparameter tuning for our linear models. The decision tree (DTR) for example seems to be heavily overfitting (c.f. Table 7-2). We also trained all the models only on a subset of 160.00 samples. Linear models tend to perform better on smaller sample sizes.

Table 7-3: Scenario 2 results on training subset[8]

	DTR		**RFR**		**MLPR**		**Lasso**		**Ridge**		**Lasso_log**	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
RMSE	0.13	1.59	0.42	1.14	0.84	1.11	1.1	1.13	1.17	1.19	424.4	5.93
MAE	0.02	0.7	0.2	0.53	0.45	0.55	0.57	0.58	0.63	0.63	4.97	3.9

For the second scenario the Random Forest Regressor (RFR) and the multi-layer perceptron regressor (MLPR) seem to capture the complexity too well. The training error is in both cases significantly lower than the test error and they seem to be overfitting on the training set. The same applies to the decision tree regressor (DTR). The DTR seems to be heavily overfitting and does not generalize as well as the RFR and MLPR, which is expected. In contrast the linear models still seem to have problems to capture the complexity, since the training and test error are similar, they seem to underfit. The overfitted nonlinear models generalize slightly better than the linear models. The idea to use a logarithmic target also seems to not be working. After having a look at the individual predictions, it seems like this is due to the high nonlinearity. The prediction model produces some unexpected outliers which result in the high error.

In both scenarios we only grid searched for polynomials with a maximal degree of three, so there was some room for improvement. In the next step the parameters for the gridsearch were adjusted to search for a higher degree and smaller alpha. For the non-linear models (RFR and MLPR) the sample size was increased to prevent overfitting. The performance of the final models is displayed in Table 7-4. The remaining MAE for both scenarios is still pretty high. When we do not know the target location it is obviously hard to predict the exact trip distance. Some of the remaining error might also occur due to the high variance in the trip distance (c.f. Figure 5-4). This can also be seen in the boxplots in 5.2 Statistics. Since we have a large amount of data it might be possible that we could have achieved higher performance with a more sophisticated neural

[5] additional information

[6] Lasso: degree 2 and alpha 0; Ridge: degree 3 and alpha 0

[7] values are rounded to the second decimal number

[8] values are rounded to the second decimal number

network. Due to limits of computational power and time constraints we only considered the MLPR from sklearn.

Table 7-4: Best models for each scenario[9] [10]

	Scenario 1 - MLPR		**Scenario 2 - MLPR**	
	Train	Test	Train	Test
RMSE	3.26	3.25	1.01	0.97
MAE	2.01	2.01	0.47	0.47

The final models for predicting the trip distance are inaccurate. In the first scenario higher polynomials did not yield any better results. The score of the MLPR slightly improved. In the second scenario with known target location, we reached a MAE of 0.47 miles (0.76km; c.f. Table 7-4). Regarding the given setting of short distance trips inside a city, this MAE can be interpreted as a quite inaccurate prediction. Especially if we have the option to use navigation systems which can calculate the exact trip distance if the start and end location is given. Our second scenario is therefore only interesting from an optimisation and technical standpoint, since the performance was increased step by step. In the end we spend a lot of time trying to improve the linear models. Further research could focus more on tuning models like the Random Forest Regressor. In the implemented command line interface, the model from the second scenario is used.

7.1.2 Predicting the Fare Amount

The focus of the prediction is on the information available at the start and end of the trip (time and place). Therefore, input features such as '*tip_amount*', '*total_amount*', '*payment_type*', and all IDs are thrown out of the original dataset. The first three mentioned features are known after the trip and the IDs are already represented by start/end location longitude/latitude. Thus, those features hold no additional information to predict '*fare_amount*'. The Pearson Correlation is used to identify highly correlated features for '*fare_amount*' (c.f. Figure A-6). Using a correlation rule to identify well correlated features, it is assumed that the correlation values should be higher than 0.2, thus the most relevant features for '*fare_amount*' are '*trip_distance*', '*tolls_amount*', '*congestion_surcharge*', '*start_location_long*', '*start_location_lat*', '*end_location_long*', and '*end_location_lat*' in this case. Furthermore, linear regression assumes that the independent variables need to be uncorrelated with each other. Therefore, the correlation of the most correlated '*trip_distance*' to all other relevant features is checked. Using another standard correlation rule (correlation should be higher than 0.5) again, '*tolls_amount*' is only highly correlated to '*trip_distance*', hence we would keep only one variable ('*trip_distance*'). In the end, all relevant features are '*trip_distance*', '*congestion_surcharge*', '*start_location_lat*', '*end_location_long*', and '*end_location_lat*'.

The data needs to be normalized in such a way that the range of all variables is almost similar. This is easily done in python by using the StandardScaler from sklearn. First, a linear regression model is trained on the data set and its performance is checked on the validation set. To check the model performance, the data set is randomly split into a test (20%)/ train set (80%) and RMSE and R² are used as evaluation metrics. On the one hand, Table 7-5 pictures the RMSE evaluation values of a simple linear model and the Random Forest Regressor. On the other hand, Table 7-6 pictures the R² evaluation values of the OLS, Lasso and Ridge linear regression. First, the model is trained on all five relevant features without considering their individual importance. The simple linear regression model calculates a low RMSE value on both training and validation data, which is decent. In addition, the Random Forest Regressor is applied on the dataset and it calculated an even lower RMSE yielding a better performance in this case. But unfortunately, this tree-based model is highly computationally demanding. Due to hardware limitations, there was a need for another well performing, but less resource intensive model. As a result, Lasso, OLS and Ridge linear regression are trained

[9] values are rounded to the second decimal number
[10] training time for scenario 1 - 6h and scenario 2 - 3.5h

and each of them provide a R^2 of 0.84. Thus, those models perform very well too. So, Lasso is picked out of those three because of less computational power, and slight performance and model modification benefits. The main goal of this prediction task is the development of a parsimonious model which is a less complex model without compromising on the overall model performance. Plotting the importance of the relevant input features, one could tell that *trip_distance* is highly important for predicting '*fare_amount*' (c.f. Table 7-7). The other features ('*start_location_lat*', '*congestion_surcharge*', '*end_location_lat*', '*end_location_long*') are unnecessary, since they seem to lack predictive power despite being correlated. Using '*trip_distance*' as the only input feature for the models did not reduce the performance of Ridge, Lasso and OLS. That means that all other input features from previous models can be thrown out after comparing the model performance of OLS, Ridge and Lasso regression because performance still stays the same. Hence, the task is to find a simple model with few variables, Lasso linear regression with only input feature '*trip_distance*' is chosen.

Table 7-5: Evaluation metric values for each model[11]

	Simple Linear Regression		**Random Forest Regressor**	
	Train	Test	Train	Test
RMSE	0.36	0.52	0.31	0.48

Table 7-6: Evaluation metric values for each model [12]

	OLS		**Lasso**		**Ridge**	
	Train	Test	Train	Test	Train	Test
R^2	0.84	0.84	0.84	0.84	0.84	0.84

[11] values are rounded to the second decimal number

[12] values are rounded to the second decimal number

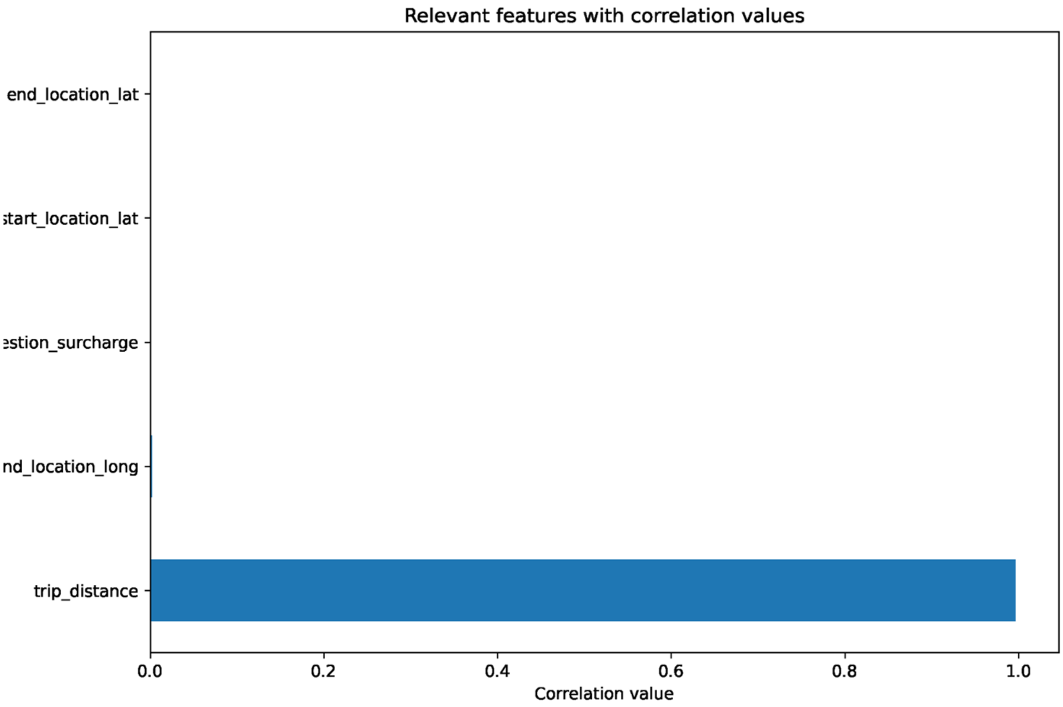

Figure 7-1: Relevant input featues for fare_amount

7.1.3 Predicting the Trip Amount

As additional information, we tried to predict the number of trips on an hourly basis, which could offer a wide range of business applications, as the possibility to predict capacity allows for better distribution management. Our assumption was that the trip amount is highly correlated with the time of the day and the weather. We developed a neural net with two layers which are connected by 20 nodes. We also tried a neural net with 10 nodes, but 20 nodes performed significantly better. For the activation function, we tried different activation functions like RELU, but in the end we used sigmoid because of the best results. As loss function the MSE-loss was chosen and the stochastic gradient descent for optimizing the model through a training time of 15 epochs. The model was developed with the pytorch lightning framework, and the categorical features were one-hot encoded. To test our prediction, we used the last three weeks of the year and visualized our accuracy (c.f. Figure 7-2). One can see that a correlation between time of the day, weather and number of trips exists. Unfortunately, pytorch lightning proposed difficulties, like the need to load the data each time when using the model. Further research could consider frameworks like keras that might be more suitable for the task. A model that is able to differentiate between service zones and use predicted weather information could even be able to predict the upcoming number of trips. This could offer a great benefit for the taxi industry, as it would be much better possible to plan the needed capacity and distribute the vehicles.

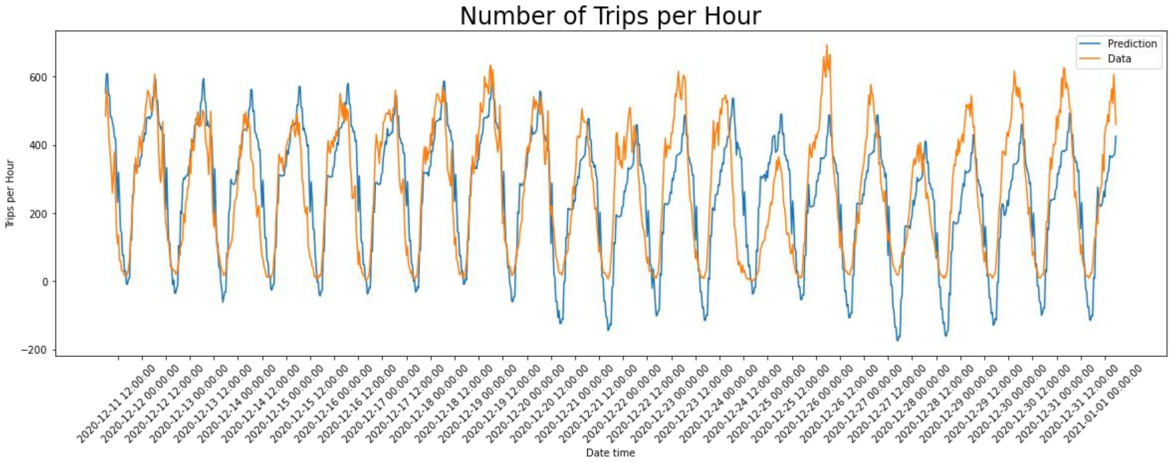

Figure 7-2: Prediction accuracy of the trip amount

7.2 Classification

7.2.1 Predicting the Payment Type

There are six payment type categories listed in the data dictionary [21], but only 1-4 are present in the filtered data which is heavily imbalanced, as the majority of payments are via credit card (c.f. Table 7-7).

Table 7-7: Payment type categories, count and share

Category	Name	Count	Share
1	Credit card	1072423	72.7 %
2	Cash	393297	26.7 %
3	No charge	7115	0.5 %
4	Dispute	2547	0.2 %
5	Unknown	0	0 %
6	Voided trip	0	0 %

Following a test for linear correlation on the with weather and lockdown data enhanced dataset, the following features proved a correlation above 0.04 to the *'payment_type'* target: *'tip_amount', 'congestion_surcharge', 'end_location_long', 'start_location_id', 'total_amount', 'Temperature', 'Minimum, Temperature', 'Maximum Temperature', 'lockdown'*. While a correlation of 0.05 is a common threshold, the *temperature* and *lockdown* features were so close to it, that we decided on taking them into consideration as well. Since '*Temperature', 'Minimum Temperature'* and '*Maximum Temperature'* had the exact same correlation and were highly correlated to each other, we decided on just using the '*Temperature'* feature. Furthermore, while the longitude of the start and end location seemed to be far higher correlated then the latitude, we decided on using the location id instead of the longitude for both start and end location, as the longitude is part of the information contained in the '*location_id'*, which means we do not lose information by switching this feature, but also might be able to gain information through the latitude. And since the '*location_id'* is categorical, it was not well represented in the linear correlation, further justifying the switch. So, the utilized features are as follows (c.f. Table 7-8):

Table 7-8: Training features

numerical:	categorical:
● *'tip_amount'* ● *'congestion_surcharge'* ● *'total_amount'* ● *'temperature'* ● *'lockdown'*	● *'do_location_id'* ● *'pu_location_id'*

To combat the heavy imbalance, data oversampling and SMOTE were tested, but just resulted in a worse performance, as the resulting model tried to predict the minority classes over proportionally, resulting in class precision of both under 10% and average recall of 20% for both minority classes. The overall performance dropped, as such that the classes and their relevance were deemed too minor to be considered further. Then the training pipeline was defined with a simple imputer und a robust scaler for the numerical features, and the categorical features were one-hot encoded. Furthermore, the data was split into a train and test set, with the test set being 0.2 the size of the complete dataset.

The following models were then trained on the training data to compare out of the box performance and decide on the model with the most potential for further tuning: logistic regression, KNN, decision tree, random forest, SVM, MLP. The logistic regression, as well as the MLP completely ignored both minority classes, while the SVM only ignored class 4. The KNN was rather fast to train, but it's runtime for the predictions scales extremely badly with the amount of data we have, rendering it practically useless. Only the decision tree and the random forest managed to even predict the minority classes, albeit especially bad. The random forest performed slightly better, and we then fine tuned the hyperparameters via a 5-fold cross validated GridSearch. The resulting performance can be seen in Table 7-9 and Table 7-10.

Table 7-9: Payment type metrics

category	**precision**	**recall**	**f1-score**	**support**
1	1.00	0.96	0.98	1072423
2	0.89	1.00	0.94	393297
3	0.72	0.00	0.01	7115
4	0.50	0.00	0.00	2547
accuracy			**0.97**	1475382

Table 7-10: Payment type confusion matrix

Actual:	Predicted:			
	1	2	3	4
1	1034236	38185	0	2
2	1943	391347	5	2
3	107	6989	18	1
4	62	2478	2	5

The model achieved an accuracy of 0.97, with a perfect precision on 1, perfect recall on 2 and an extremely good recall on 1. Categories 3 and 4 have a decent precision, but a non-existent recall. The metrics and confusion matrix show that while 1, 2 are really well predicted and 3, 4 have decent precision, the recall for 3, 4 is zero. The confusion matrix shows how slim the share of correct values actually is, undermining the validity of the precision for 3 and 4. While 3, 4 seem hard to capture, the accuracy shows how little impact those two have. Nevertheless, for category 3 and 4 a higher fraction of predictions are attributed to the respective 3rd and 4th category when compared to 1 and 2, although not significantly. In the end we get a model with a really good accuracy, but we could potentially try to improve on it in future work by testing other techniques for our imbalance problem.

8 Conclusion

Through the visualization and processing we gained insights into the (taxi) mobility within the city of New York (especially Brooklyn). Due to the focus on the year 2020, it was unavoidable to consider the strong influence of the pandemic on the taxi rides. As expected, the corona pandemic had a strong impact on mobility which is now visualized (c.f. 5.3 Corona Exploration). Especially the lockdown in March had a severe influence and the number of passengers fell sharply. 52% of the trips of the year 2020 were driven in the months January and February therefore before the pandemic started. Furthermore, the impact of the rush hour on passenger count was now lower. The numbers rose a little bit after reopening measures in May and June, but still stayed way below the numbers in the months of January and February. The lockdowns in November lead again to a small decrease. We can also say that the pandemic impacted the drop off location. Before the pandemic most drop off locations were centralized while under restrictions the drop off locations were more spread out (c.f. 6.2 Heatmap). This might be due to more people working from home.

As for the predictions, we trained promising models to predict the trip distance (c.f. 7.1.1 Predicting the Trip Distance), fare amount (c.f. 7.1.2 Predicting the Fare Amount), trip amount (c.f. 7.1.3 Predicting the Trip Amount) and payment type (c.f.7.2.1 Predicting the Payment Type). These models were specifically trained on the trips connected to Brooklyn and should therefore be used in that context. Generalization to other boroughs might reduce the predictive power. Since we focused on Brooklyn it is expected that a high number of ids correspond to boroughs inside of Brooklyn and a small sample size corresponds to locations outside of Brooklyn. Due to computer limitations and extensive training times only a selected number of models were trained on the whole dataset. We generally tried to evaluate the models on a smaller sample size and then train a final model on the whole dataset. To increase the performance and generalizability possible further work should consider using the whole NYC dataset. As for the trip distance prediction we propose that in further work the beeline between the centroid of PU and DO location is calculated. This should take our approach of calculating the absolute difference for longitude and latitude one step further. Further research can also use the implemented command line interface to compare their predictions with our results. The command line interface supports the prediction of trip distance, fare amount and payment type and can be used through the python package “Yellowcab”.

9 References

[1] Online Quelle (Stand 14.06.2021): TLC Trip Record Data

https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

[2] Online Quelle (Stand 14.06.2021): New York State on PAUSE.

https://coronavirus.health.ny.gov/new-york-state-pause

[3] Online Quelle (Stand 14.06.2021): NYC Business Reopening Guide.

https://www1.nyc.gov/nycbusiness/article/reopening-guide

[4] Online Quelle (Stand 14.06.2021): Covid-19 pandemic in New York City.

https://en.wikipedia.org/wiki/COVID-19_pandemic_in_New_York_City

[5] Online Quelle (Stand 14.06.2021): Covid-19 pandemic in New York (state).

https://en.wikipedia.org/wiki/COVID-19_pandemic_in_New_York_(state)

[6] Online Quelle (Stand 14.06.2021): As U.S. Breaks Hospitalization Records, N.Y. and Other States Add Restrictions.

https://www.nytimes.com/live/2020/11/11/world/covid-19-coronavirus-live-updates

[7] Online Quelle (Stand 14.06.2021): Timeline: The first 100 days of New York Gov. Andrew Cuomo's COVID-19 response.

https://abcnews.go.com/US/News/timeline-100-days-york-gov-andrew-cuomos-covid/story?id=71292880

[8] Online Quelle (Stand 14.06.2021): Everything You Need To Know About Phase 3 Of Reopening NYC.

https://gothamist.com/news/everything-you-need-know-about-phase-3-reopening-nyc

[9] Online Quelle (Stand 14.06.2021): During Novel Coronavirus Briefing, Governor Cuomo Announces New Mass Gatherings Regulations.

https://www.governor.ny.gov/news/during-novel-coronavirus-briefing-governor-cuomo-announces-new-mass-gatherings-regulations

[10] Online Quelle (Stand 12.06.2021): Sklearn - Support Vector Regressor

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

[11] Online Quelle (Stand 12.06.2021): Sklearn - Nearest Neighbor Regression

https://scikit-learn.org/stable/modules/neighbors.html#regression

[12] Online Quelle (Stand 12.06.2021): New York City - Data dictionary/ documentation

https://www1.nyc.gov/assets/tlc/downloads/pdf/data_dictionary_trip_records_yellow.pdf

A Appendix

Table A-1: Data description - trip data [12]

Column	Description	Data type
VendorID	A code indicating the TPEP provider that provided the record. 1= Creative Mobile Technologies, LLC; 2= VeriFone Inc.	Categorical
tpep_pickup_datetime	The date and time when the meter was engaged.	Numerical
tpep_dropoff_datetime	The date and time when the meter was disengaged.	Numerical
Passenger_count	The number of passengers in the vehicle (entered by the driver).	Numerical
Trip_distance	The elapsed trip distance in miles reported by the taximeter.	Numerical
PULocationID	TLC Taxi Zone in which the taximeter was engaged	Categorical
DOLocationID	TLC Taxi Zone in which the taximeter was disengaged	Categorical
RateCodeID	The final rate code in effect at the end of the trip. (1= Standard rate 2=JFK 3=Newark 4=Nassau or Westchester 5=Negotiated fare 6=Group ride)	Categorical
Store_and_fwd_flag	This flag indicates whether the trip record was held in vehicle memory before sending it to the vendor, aka “store and forward,” because the vehicle did not have a connection to the server.	Categorical
Payment_type	A numeric code signifying how the passenger paid for the trip. (1= Credit card 2= Cash 3= No charge 4= Dispute 5= Unknown 6= Voided trip)	Categorical
Fare_amount	The time-and-distance fare calculated by the meter.	Numerical
Extra	Miscellaneous extras and surcharges. Currently, this only includes the $0.50 and $1 rush hour and overnight charges.	Numerical
MTA_tax	$0.50 MTA tax that is automatically triggered based on the metered rate in use.	Numerical
Improvement_surcharge	$0.30 improvement surcharge assessed trips at the flag drop. The improvement surcharge began being levied in 2015.	Numerical
Tip_amount	Tip amount – This field is automatically populated for credit card tips. Cash tips are not included.	Numerical
Tolls_amount	Total amount of all tolls paid on the trip.	Numerical
Total_amount	The total amount charged to passengers. Does not include cash tips.	Numerical

Table A-2: Data description - taxi zones (taxi_zones.csv)

Column	Description	Data type
LocationID	Number of each TLC Taxi Zone in New York	Categorical
Borough	Name of the New Yorker borough according to the LocationID	Categorical
Zone	Name of each Zone according to the LocationID; Each LocationID got a Zone name	Categorical
service_zone	Multiple Zones are grouped in service zones; Name of LocationIDs service zone	Categorical

Table A-3: Data description - geo data (taxi_zones.json; imported as GeoDataFrame)

Column	Description	Data type
shape_area	Area size of the Taxi Zone	Numerical
objectid	ObjectID equals location_id from Taxi Zone	Categorical
shape_leng	Length of the geo polygon scope	Numerical
location_id	Equivalent to LocationID from Taxi Zones csv-file	Categorical
zone	Name of each Zone according to the LocationID; Each LocationID got a Zone name	Categorical
borough	Name of the New Yorker borough according to the LocationID	Categorical
geometry	Geometry Data, which contain the latitudes and longitudes of the zone border, which get converted to Shapes	Numerical

Table A-4: Data filering - resulting drops and rationale

Filter	Dropped trips	Rationale
Borough == brooklyn	16218357, or 68%	The problem is limited to Brooklyn.
Pick up in 2020	92, or 0.001%	The problem is limited to 2020.
Drop off in 2020	77, or 0.001%	The problem is limited to 2020.
passenger 0 < x	154638, or 2%	Trips without passengers are irrelevant
trip distance 0 < x < 1000	91196 or 1.2%	Trips without any covered distance, or an absurd amount, are negligible
fare amount 0 < x < 7000	29047, or 0.4%	Trips without any fare, or an absurd amount, are negligible.
tip amount 0 <= x	0, or 0%	Trips with a negative tip are likely faulty (Probably filtered by fare amount).
tolls 0 <= x	0, or 0%	Trips with a negative toll are likely faulty (Probably filtered by fare amount).

total 0 < x < 7000	0, or 0%	Trips without any total, or an absurd amount, are negligible (Probably filtered by fare amount).
congestion surcharge 0 <= x	0, or 0%	Trips with a negative congestion surcharge are likely faulty (Probably filtered by fare amount).
duration 0 < x <16h = 57.600	16068, or 0.22%	There is a high density of trips with a duration close to 24h = 86.400s, where drop off and pick up might have been switched, those, and trips with 0 or negative duration are negligible.
duration above 13k and time at midnight	1599, or 0.02%	Lots of rides with DO or sometimes PU times at midnight, that are likely faulty.
x is NaN	0, or 0%	All rows with NaN values are to be discarded

Table A-5: Scenario 1 gridserach for degree and alpha (Lasso) [13]

Degree	**Alpha (Lasso)**	**R2**
1	0.00	0.05
1	0.15	0.50
1	0.03	0.49
1	0.45	0.00
1	0.06	0.48
1	0.75	0.48
1	0.09	0.48
1	1,05	0.48
1	0.12	0.05
1	1,35	0.05
1	0.15	0.05
2	0.00	0.05
2	0.15	0.05
2	0.03	0.05
2	0.45	0.05
2	0.06	0.48

[13] values are rounded to the second decimal number

2	0.75	0.05
2	0.09	0.05
2	1,05	0.48
2	0.12	0.05
2	1,35	0.48
2	0.15	0.48
3	0.00	0.05
3	0.15	0.50
3	0.03	0.05
3	0.45	0.05
3	0.06	0.05
3	0.75	0.48
3	0.09	0.05
3	1,05	0.05
3	0.12	0.48
3	1,35	0.48
3	0.15	0.05

Table A-6: Scenario 2 gridsearch for degree and alpha (Lasso)[14]

Degree	**Alpha (Lasso)**	**R2**
1	0.00	0.94
1	0.15	0.92
1	0.03	0.09
1	0.45	0.92
1	0.06	0.92
1	0.75	0.92
1	0.09	0.92
1	1,05	0.92
1	0.12	0.92
1	1,35	0.92

[14] values are rounded to the second decimal number

1	0.15	0.92
2	0.00	0.94
2	0.15	0.09
2	0.03	0.94
2	0.45	0.09
2	0.06	0.93
2	0.75	0.93
2	0.09	0.93
2	1,05	0.93
2	0.12	0.01
2	1,35	0.93
2	0.15	0.09
3	0.00	0.95
3	0.15	0.94
3	0.03	0.94
3	0.45	0.93
3	0.06	0.93
3	0.75	0.93
3	0.09	0.93
3	1,05	0.93
3	0.12	0.93
3	1,35	0.93
3	0.15	0.93

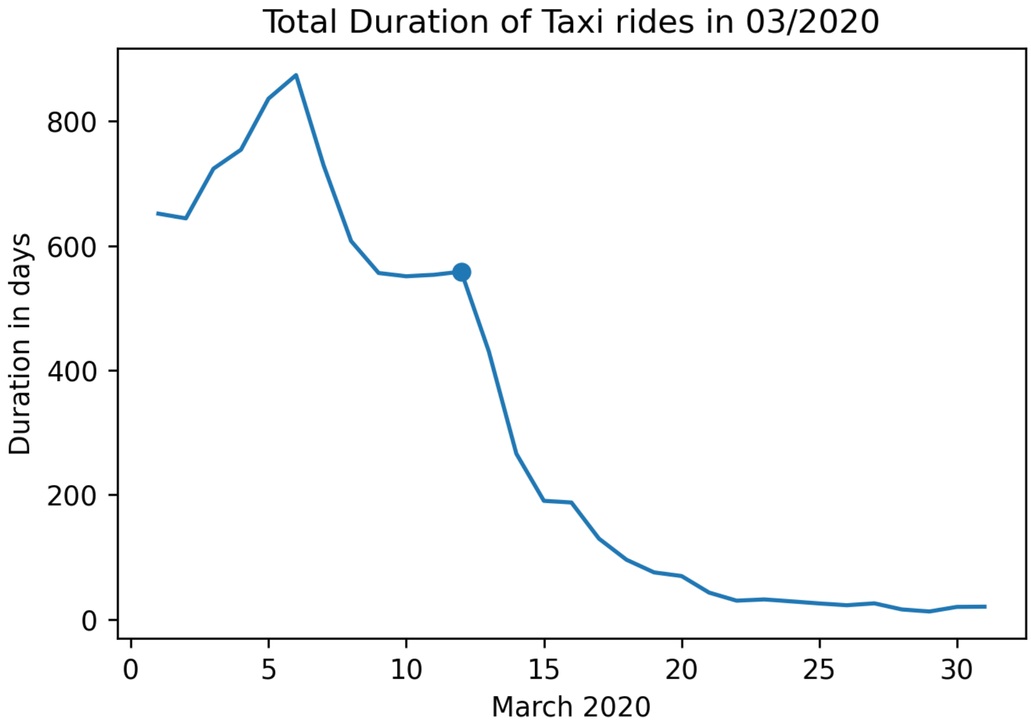

Figure A-1: Total duration of taxi rides in March 2020

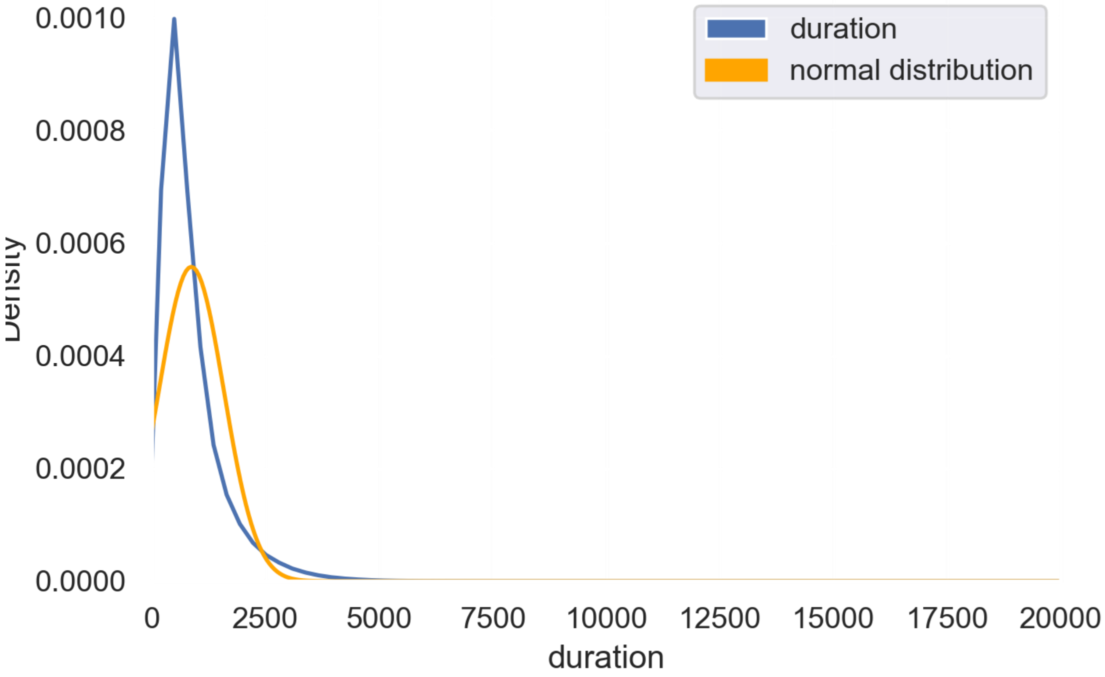

Figure A-2: Compare distributions trip duration

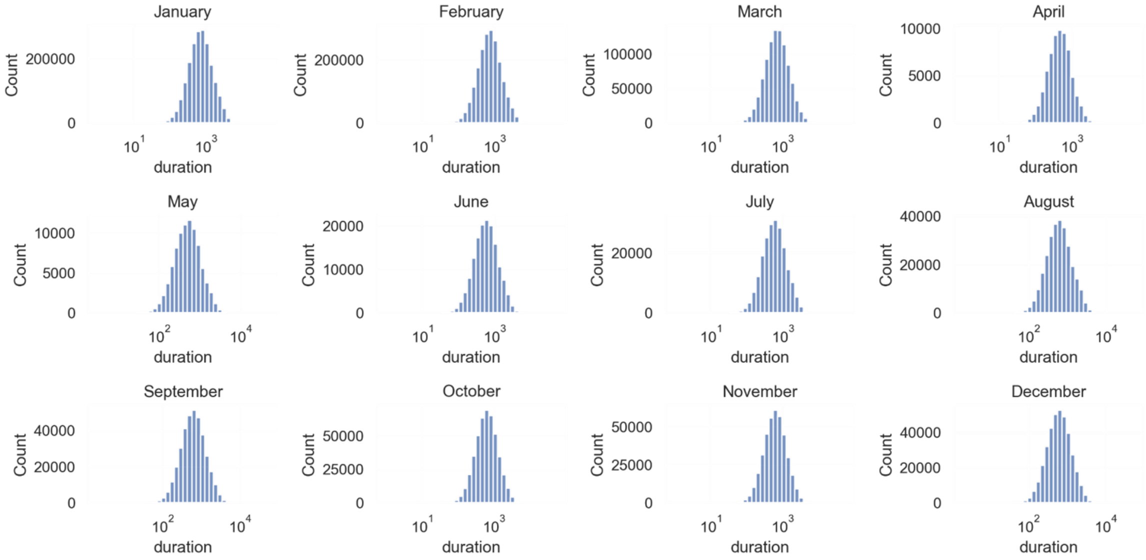

Figure A-3: Monthly histogram for trip duration (logarithmic scale)

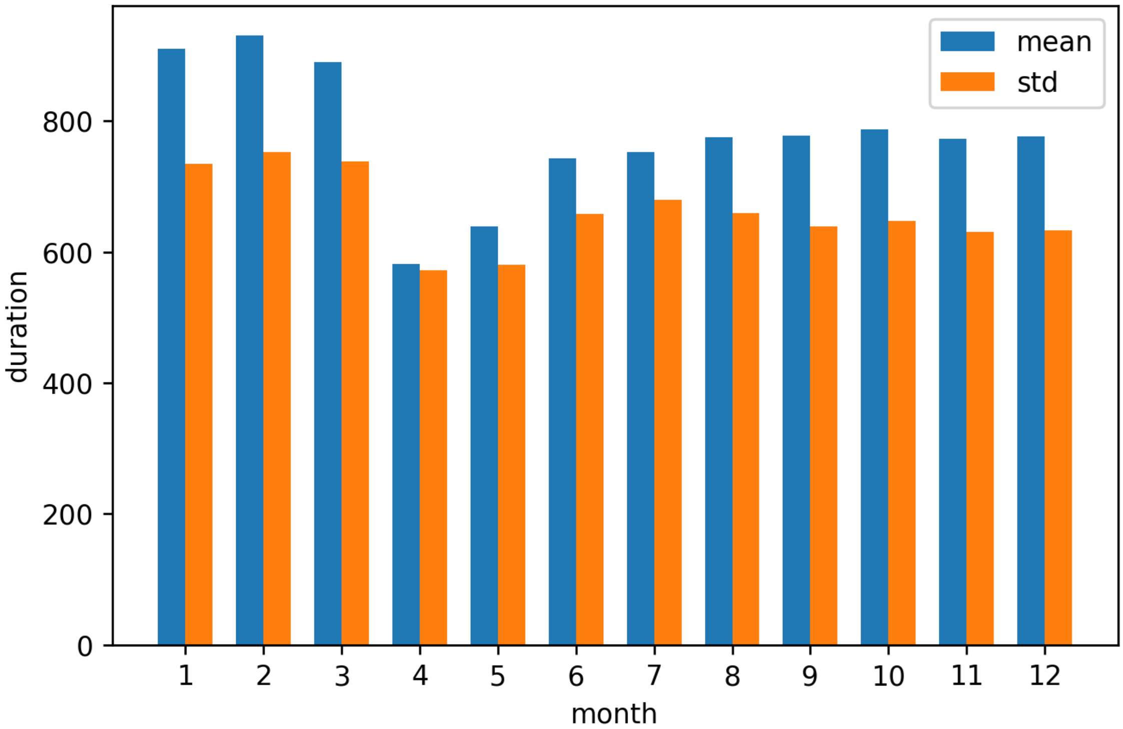

Figure A-4: Monthly mean and std for duration

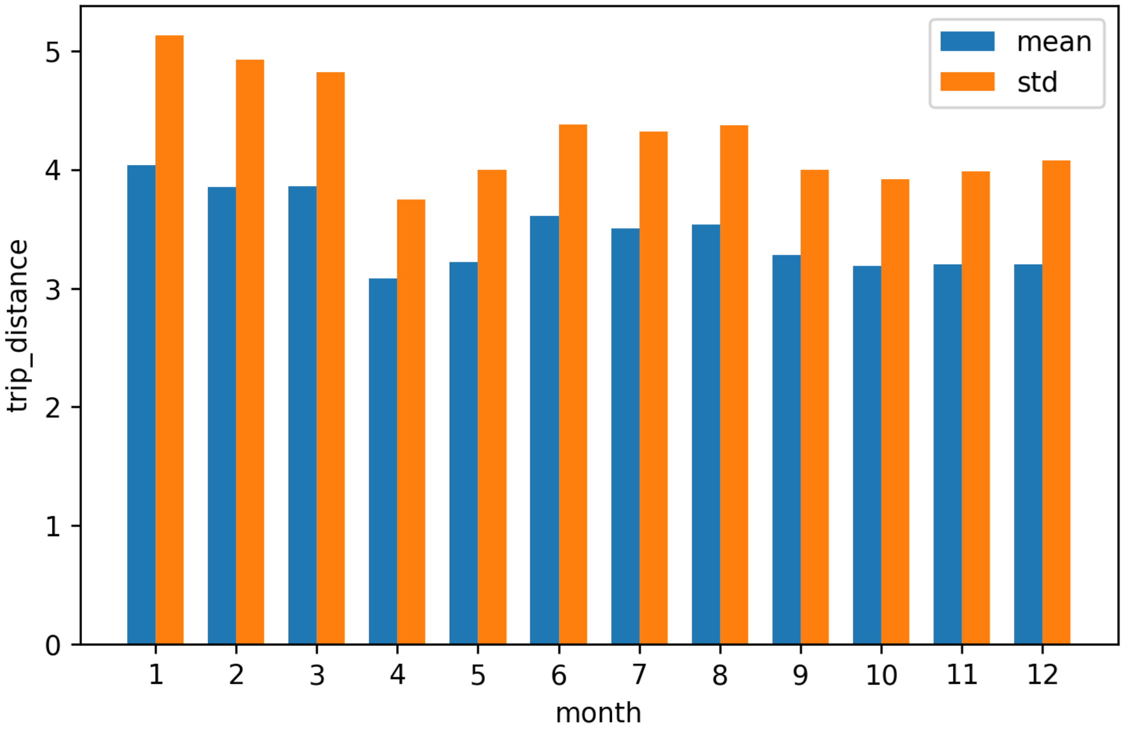

Figure A-5: Monthly mean and std for distance

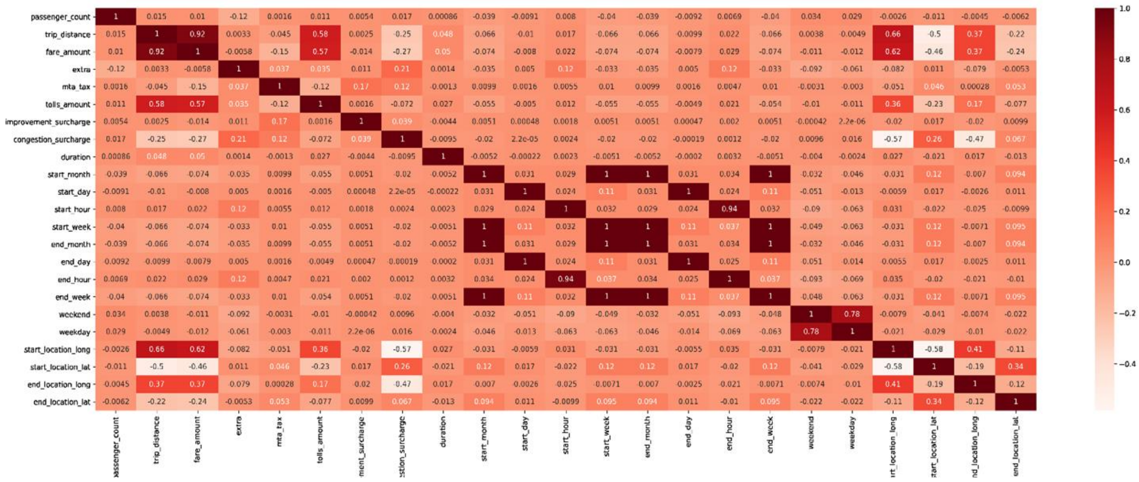

Figure A-6: Correlation matrix of fare amount