NYC Taxi – Data modelling

Author: Joan Borràs

1. Introduction

This document briefly explains the main considerations for creating the model. The notebook "*data_modelling*" shows a simple example of how to test some existing models using Spark.

2. Feature engineering

The function *add_feature_taxi_data* located at the module *preprocessing.py*, adds new features that it could be interesting for the model. The features created are the following:

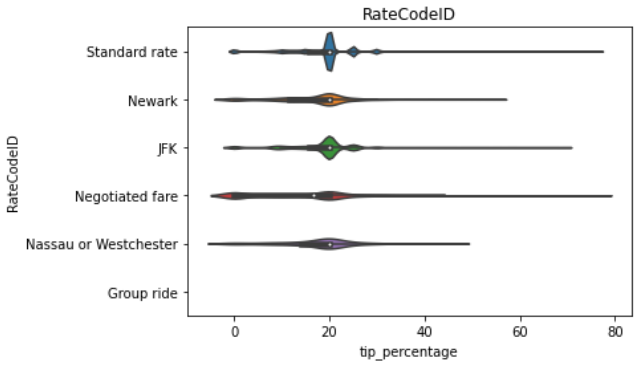
- pu_month: month of the pickup trip
- pu_year: year of the pickup trip
- do_month: month of the drop off trip
- do_year: year of the drop off trip
- duration_in_min: duration of the trip in minutes
- tip_percentage: % of the tip from the total amount (minus tip amount).
- day_of_week: day of the week
- day_of_month: day of the month
- hour: hour of the day

In the future it would be interesting to add a calendar of holidays. Probably tips during Christmas days or other special days are higher.

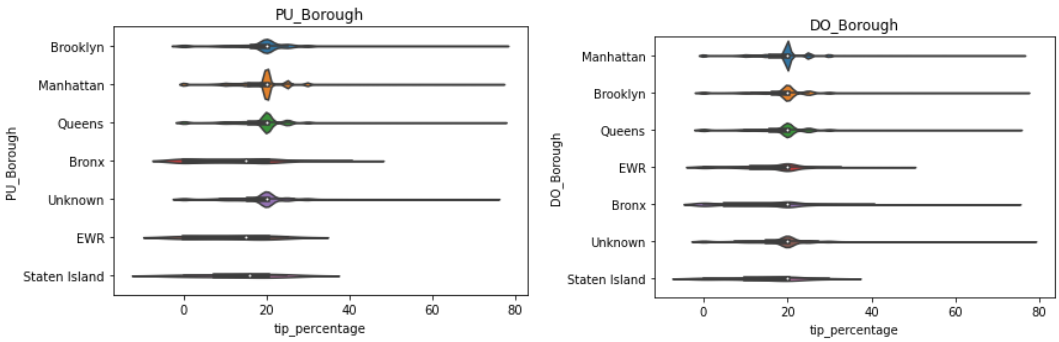
Weather condition could also affect the tip.

3. Feature selection

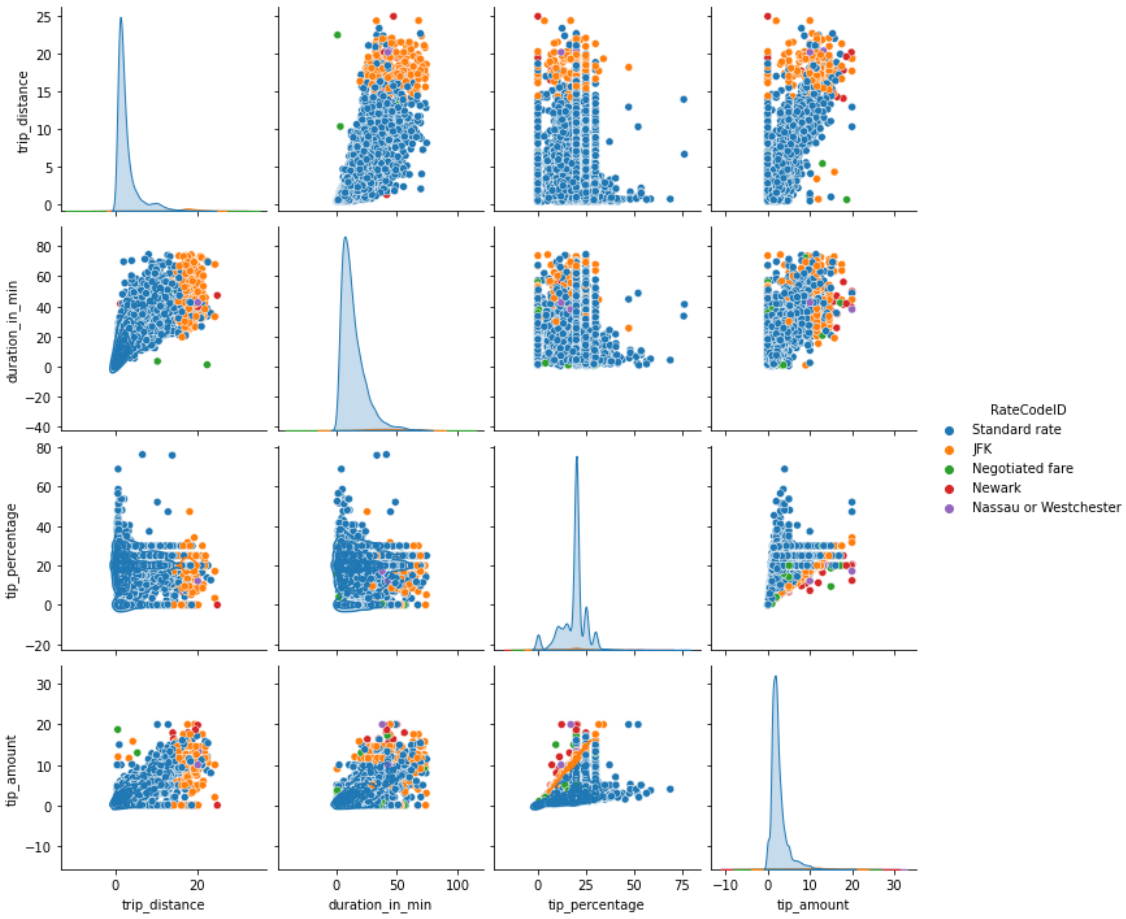
In the notebook *data_summary* we have seen interesting trends. For instance, as shown below, the distribution of percentages varies depending on the rate code, seeing that negotiated fare tend to have trips without tips.



Pick up and drop out boroughs have also different distributions. Some of them have high dispersion of tip_percentage.



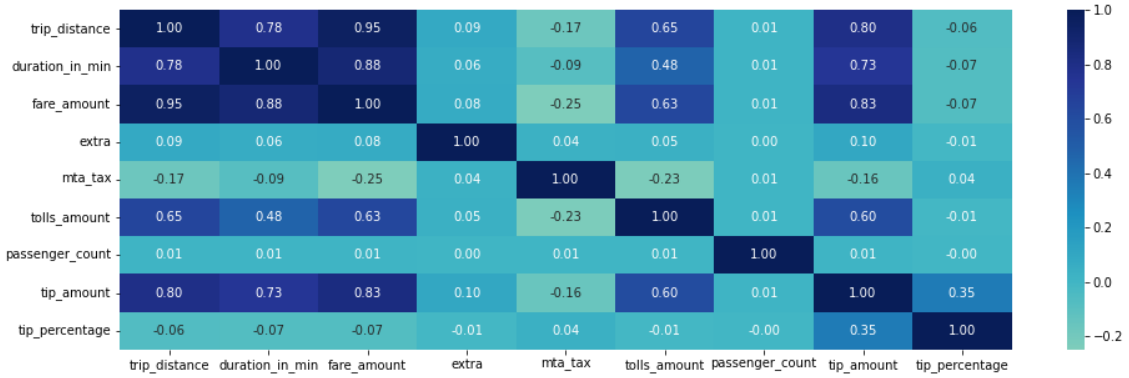
Looking at pairwise relationships between features we can discover interesting trends (figure below). JFK has higher trip distance and tip percentage. In standard rate, we can see high tip percentages with short distances.



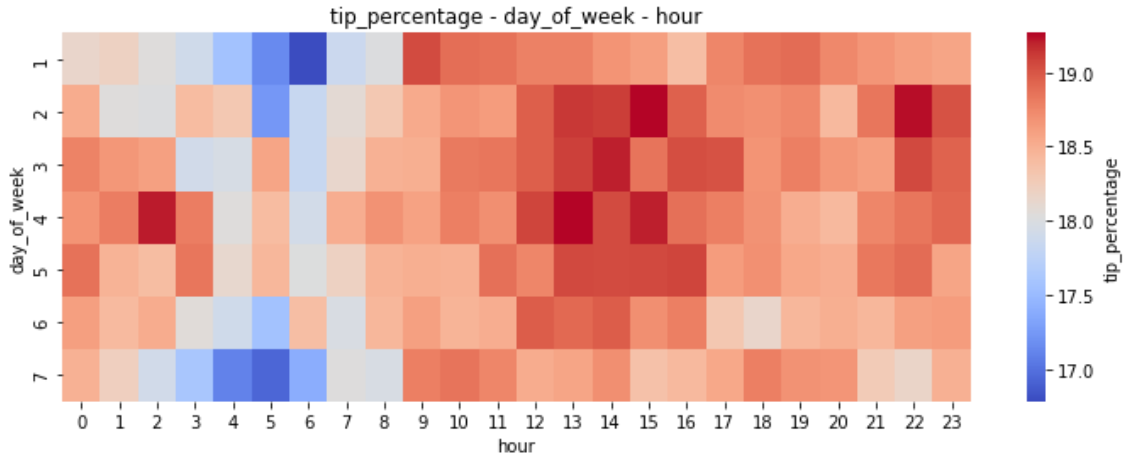
Below it is shown the pairwise correlation between numerical columns which can be concluded that:

- Features with higher correlations with tip_amount are: *total_amount, fare_amount, trip_distance and duration_in_min*
- *mta_tax* has a little negative correlation with *tip_amount*

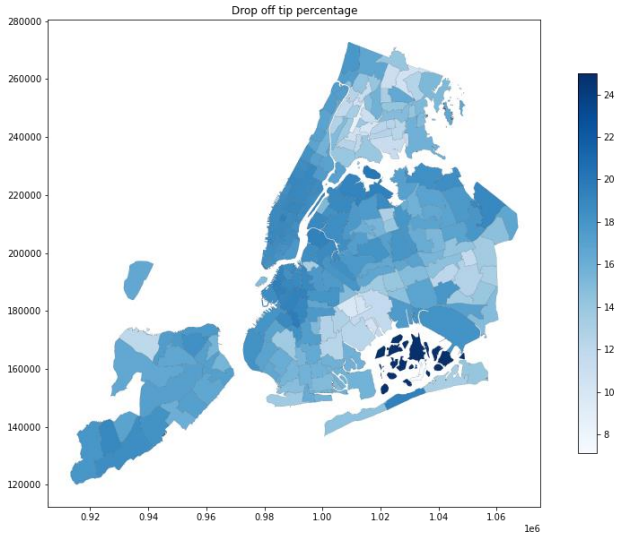
- *tip_percentage* only has some correlation with *tip_amount*
- As expected, we can see high correlations between trip_distance, total_amount, duration_in_min, fare_amount. This is known as multicollinearity. We will have to be aware during the selection of the model.



If we analyse temporal patterns (figure below), we can see that the combination of the hour and the day of the week affects the percentage of the tips. In the notebook we have seen some other interesting patterns depending on the month as well. For instance, last day of December tend to have higher tip percentages.



Finally, seeing the map below we can suspect that the individual zones (not only the borough and service zone) will affect the results. Moreover, as we can seen in the notebook, we highlighted differences depending on the routes.



In any case, for the first model we will try to include all the features that might make sense.

We have seen that tip_amount is highly correlated with fare_amount. In this case we would see that tip fare_amount would affect too much the prediction, leaving the other features with less weight. Instead, if we make the prediction to the percentage of the tip, the model would be able to combine more features with the prediction. For this reason, we use the tip_percentage as the target variable, which in a production environment would be converted with the total amount until then to propose the final tip amount.

It has been selected the following category features: *VendorID, RateCodeID, store_and_fwd_flag, payment_type, pu_month, day_of_week, hour, PU_Borough, PU_service_zone, DO_Borough, DO_service_zone*.

These category features have been one hot encoded with new features. This is a requirement to use categorical features in our model. We have used Spark methods that converts the one hot encoded feature to vectors.

It has been selected the following numeric features: *trip_distance, fare_amount, extra, mta_tax, tolls_amount, duration_in_min, passenger_count, Pick up location and Drop off location*.

We have included geometric locations, so the model would determine that in some nearby areas the patterns are similar. By the way, we have to be aware of that still closed areas could have totally different patterns. Maybe, because there is the limit between boroughs or the real connection between them are larger.

4. Model selection

We are trying to predict a percentage of the tip amount. We have a dataset of examples that are labelled with the target. This detection of a target variable must be done with a supervised regression learning algorithm.

Different basic regression models can be applied, such as Linear Regression, Lasso Regression (considered as linear models) or Decision Tree Regression and Random Forest (non-linear models), among others. Surely, non-linear models would work better for our case. However, since the volume of the data is high, we could even opt for a deep learning model such as Neural Network Regression.

Finally, we could use ensemble learning methods that combines different models. Boosting methods, for instance, improve the results based on the errors of the other methods and help to reduce bias. Baggin would help to reduce overfitting in case we detect it.

In any case, ideally, we would test different algorithms and hyper-parameters optimization to find the one with the best results.

5. Preliminary results

In our first tests, we have applied Gradient-Boosted Trees Regressor which is one of the most common models used for regression and usually provides good results.

This model does not require to normalize the features. In addition, it accepts null values in our features. However, using Spark, we encountered some bugs, and we had to remove those observations with null values.

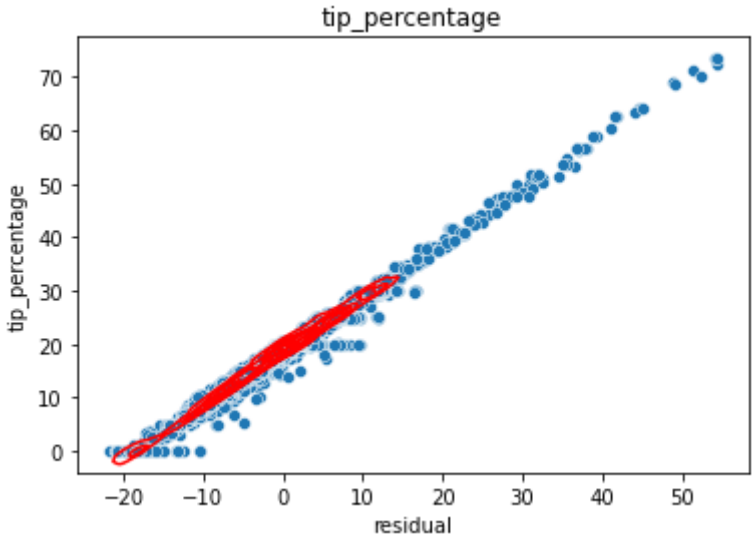
To evaluate the model, we use Root Mean Square Error (RMSE). This metric penalizes higher errors with his quadratic formula. The Root of (MSE) also gives the chance to interpret better the results.

In our first test without hyper-parameter optimization, we get an RMSE of 6.71.

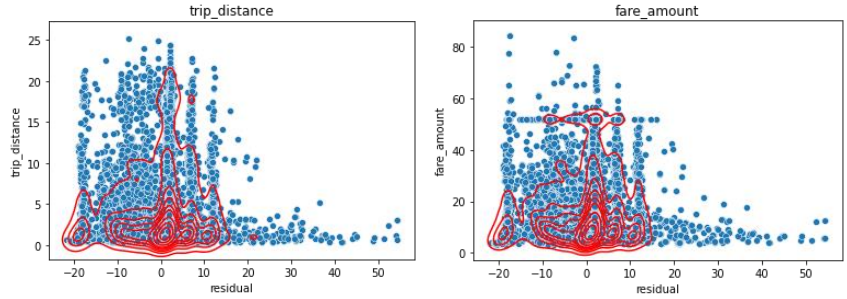
6. Residuals

The average residual is 0, so we are not over- or under-predicting

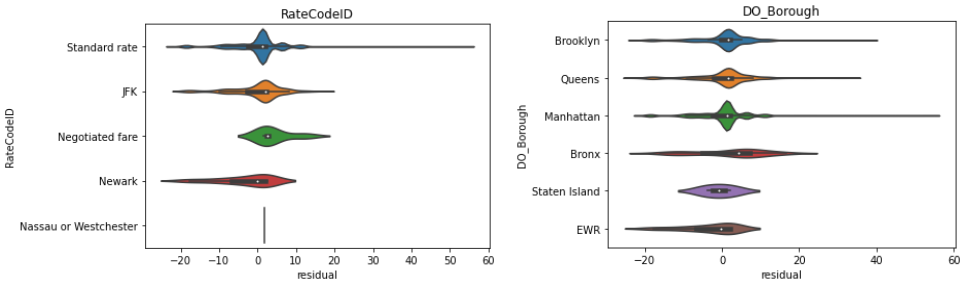
Analysing the residuals in our model we can see that we have -20 residual when tip percentage is 0.



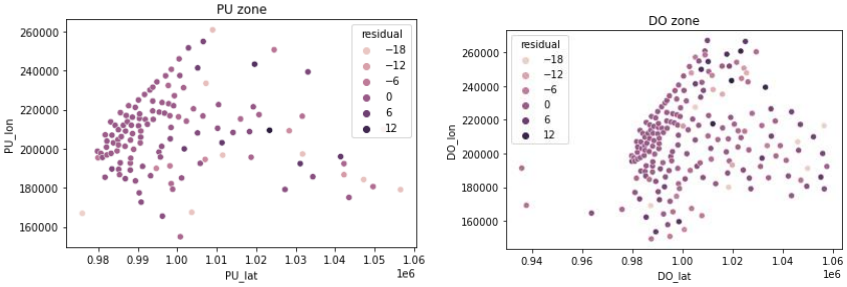
These specially happens when trip_distance and fare_amount are low.



Seeing the figures below we can see that temporal features have similar residual patterns within the different values. Rate codes have different patterns: negotiated fare is under-predicting their values. Finally, Bronx borough has higher residual than other boroughs.

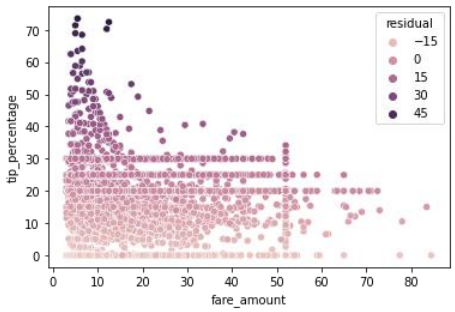


For these reasons we could try to investigate with those specific cases. Even we could explore to create a model only for Bronx borough.



Some specific zones seem to have higher residuals. We could investigate further which are those zones, and probably we could create a new feature grouping zones that have similar behaviour.

We produce residuals (>30) when fare_amount is low and tip_percentage is high (see figure below). We should investigat further the characteristics of those observations. We produce residuals of -15 when the tip_percentage is 0. We could investigate further these values. Even we could think about creating classification model that is able to predict if a trip will receive a tip or not.



7. Improving the model

The following points could be considered to improve the results of the detection:

- We can think to include other important features, such as holidays, events scholar periods, etc. that could affect the profile of users in the city in that moment (residents, workers, or tourists).
- We could apply a binning method for some features, such as passenger numbers or hours of the day.
- We can add some features specifying some zones that have different patterns or even create another model for some zones.
- Since we have are getting high residuals when tip percentage is 0. We could think about creating classification model that is able to predict if a trip will receive a tip or not.
- In case of detecting overfitting we would test the results with a validation and a test dataset. We could also apply bagging methods to reduce possible overfitting.
- Tunning hyperparameters.
- Test with deep learning methods.

8. API for the model

Once we find the best model, we would develop the data pipeline accessed through an API using FastAPI framework. This API would have 2 main endpoints, one for predicting the amount of a trip, and another one for training new models with updated data.

1. Prediction:
    a. This method would receive a trip with the current parameters
    b. Then it would execute the predict of the already trained model
    c. The estimated target value for this trip would be returned as a response to the query
    d. Optionally, the method would be able to receive a model identifier in case there are more than one model.
2. Training:
    a. This method would be useful for the company if they want to train again the model with new data. We could even consider including data from the own company.
    b. The method would receive the name of the algorithm to use and their hyper-parameters.

c. Optionally, the data pipeline of this method would extract new data, clean and add new features. Periods and URLs of the origin should be included as a parameter.
d. The results of the model and the identifier of the model would be returned as a response through the API
e. The company could use the new identifier of the model in the prediction API method.