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

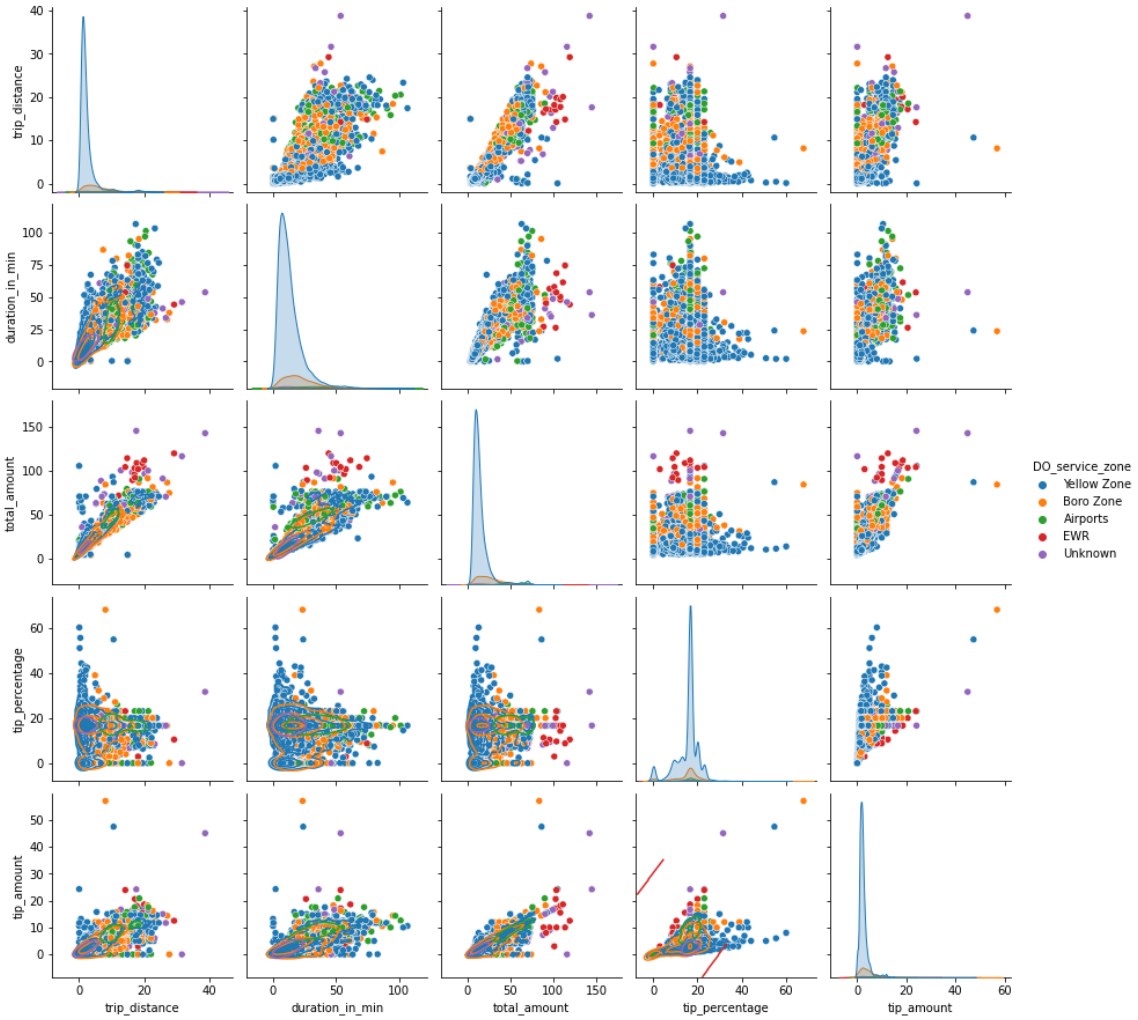
The function *add_feature_taxi_data* located at the module *preprocessing.py*, adds new features that it could be interesting for the model. The features created are the following:

- 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. Note that this variable will not be included in the model, since it needs the tip variable which is the one to be predicted.
- day_of_week: day of the week
- 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.

3. Feature selection

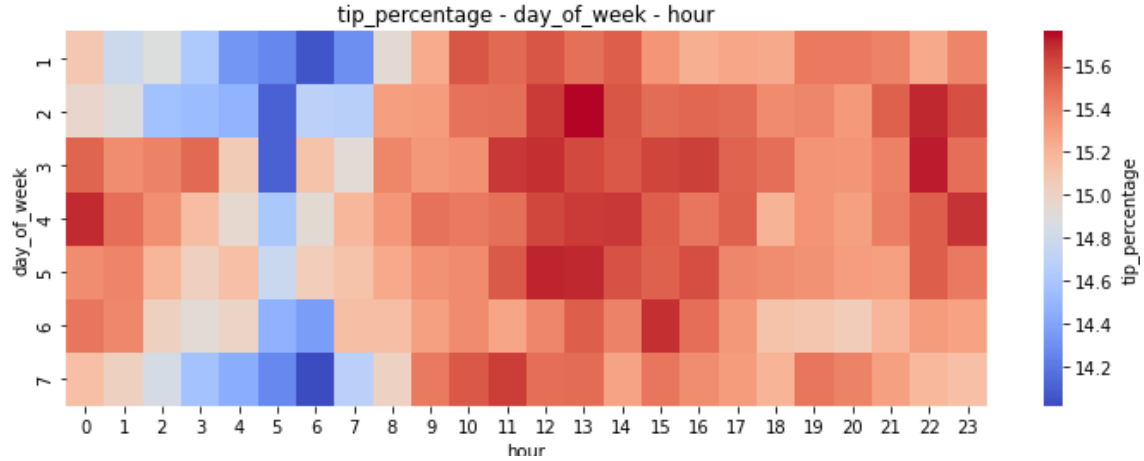
In the notebook *data_summary* we have seen interesting trends. For instance, looking at *tip_amount* and *tip_percentage* we see some trips that are grouped together (figure below). We can see more scatter plots at the notebook.



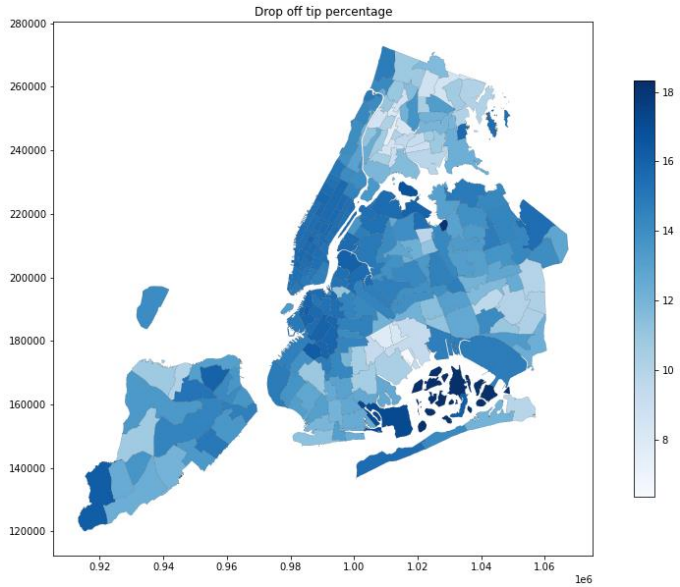
Below it is shown the pairwise correlation between numerical columns which can be concluded that:

- 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

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.



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.

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_Zone, PU_service_zone, DO_Borough, DO_Zone, 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 features to vectors.

It has been selected the following numeric features: *trip_distance, fare_amount, extra, mta_tax, tolls_amount, total_amount,duration_in_min, passenger_count*.

Since we detected in the first analysis that the distribution of most of the numerical features are right-skewed, we will need to Normalize them.

The normalization of the *passenger_count* feature it's discarded for the low range of numbers. In this case, we can explore to apply a binning method as a possible improvement of the model.

The combination of all features adds up to a total of 540 features. Note that DO_Zone and PU_Zone has a high number of possible values, which are one hot encoded.

4. Model selection

We are trying to predict a real value (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 Linear Regression, Decision Tree Regression and Gradient-Boosted Trees Regressor. Without hyper-parameter optimization we get the following results:

Model	RMSE	R^2
Linear Regression	1.71	0.58
Decision Tree Regression	1.31	0.75
Gradient-Boosted Trees Regressor	0.98	0.86

As we can see, Linear Regression is the model that performs worst since it is a linear model. Boosting methods highly improves the other two methods. Tunning hyper-parameters and test other models, such as Deep Learning, would help to improve the results.

6. Improving the model

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

- We could try other normalization methods to the numerical variables.
- Dimensionality reduction would help to reduce the number of features.
- 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.
- 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.

7. 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.