NYC Taxi Fare Prediction: Predicting using machine learning algorithms

Non-technical report of the development of an algorithm which purpose is to predict the price of NYC Taxis.

Lucas Veteikis

Problem description

It’s not new that many enterprises use machine learning to solve business problems or to propose new business models, and specifically talking about means of transport, some of them like Uber or Cabify use these algorithms to predict the cost of different cab rides. In this case, we have almost 100 million cab observations with its pickup and dropoff place and the number of passengers and our objective is to predict the price of 9914 rides which are the ones being assessed.

Evaluation metric

To assess the model's performance, Kaggle suggests employing RMSE (Root Mean Squared Error). This metric provides us with the average error considering absolute values (owing to the squared terms), but then the root takes part scaling back to the normal values so we can compare them.

$$RMSE = \sqrt{\frac{1}{N}\sum (y - \hat{y})^2}$$

Preprocessing

The dataset is fairly straight to use luckily. However, its dimensions make it really hard to use complete unless you have enough computation power. In order to manage this, I decided to sample a portion of the dataset of only 20% of the data, of course throwing 80% of the data is not ideal but due to limitations in my resources I had no choice.

One positive aspect of this dataset is that it does not contain any NaN values. These kind of values can be quite problematic when attempting to execute a machine learning model, often needing some form of imputation.

Exploratory Data Analysis

The EDA is useful when deploying a ML model in order to do different things such as detecting patterns between the variables that could be useful in the feature engineering or detecting observations that do not make sense or have mistakes.

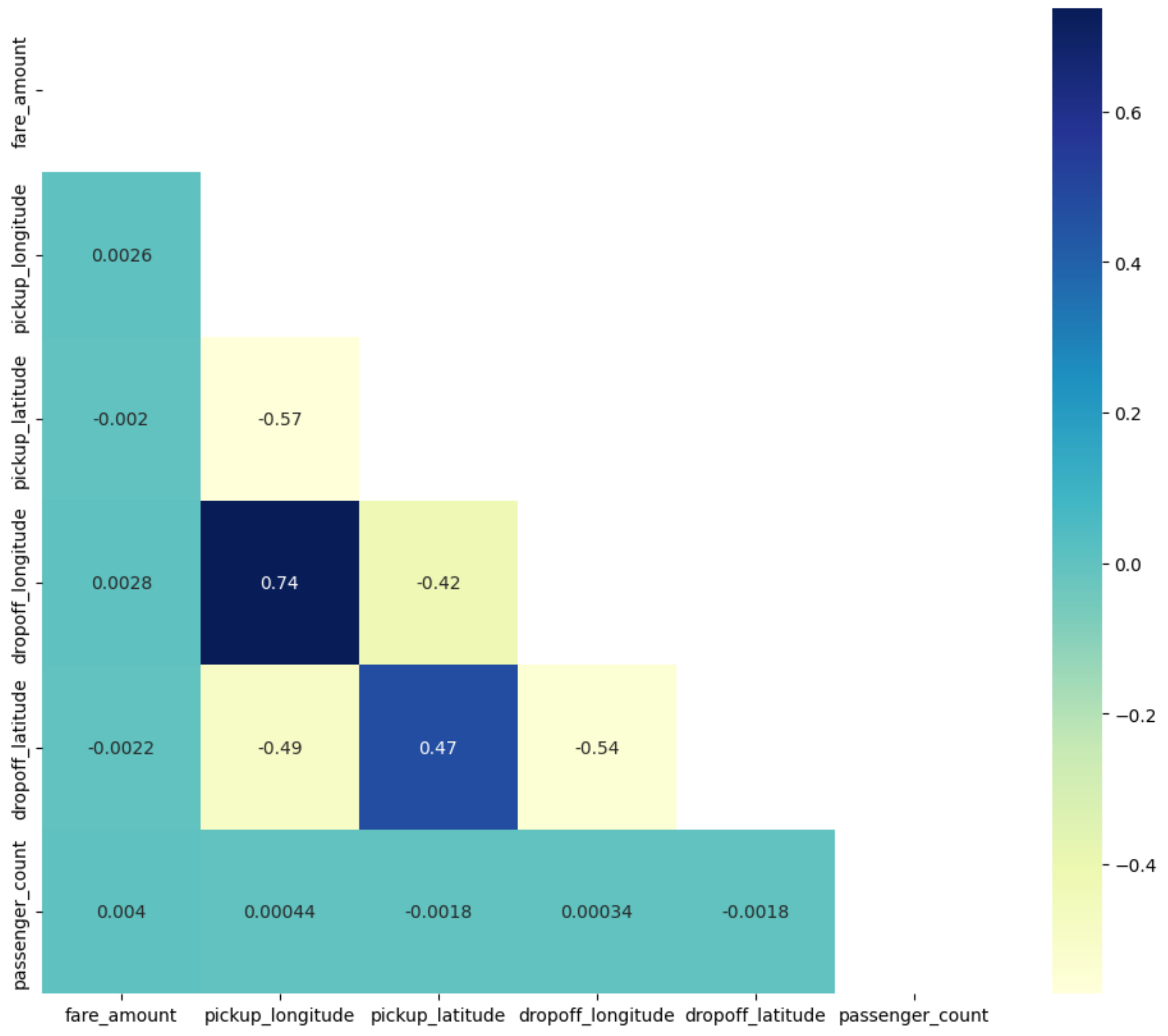
<u>Correlation matrix</u>

I decided to start building a correlation matrix which main purpose is to evaluate 2 key things:

- The limited number of variables we have should not be correlated, this would mean redundancy and the potential for reduction to a singular variable.
- Also, we would like to discover which are the variables that are actually useful when trying to predict the taxi fare amount, because a correlation that tends to 1 or -1 is better than correlation around the value 0.

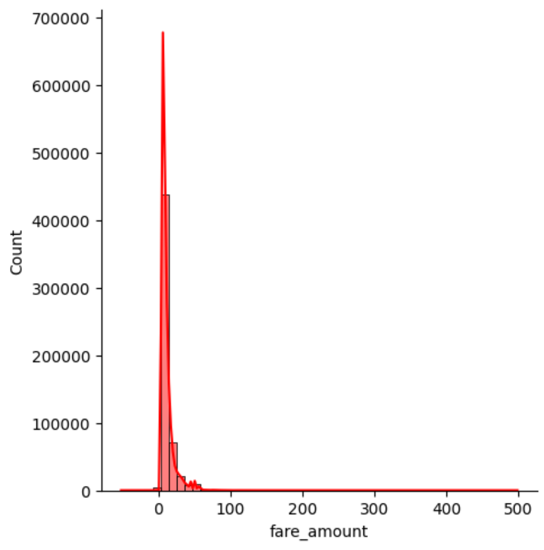
As shown, the results show us that none of the variables have a good correlation compared to fare amount, which is the dependent one, this means that these variables itselves are not able to predict the price of the cab rides.

Additionally, a moderate correlation exists among the independent variables. Though not optimal, it's far from the worst-case scenario. Watching this graph, we get the idea that the feature engineering process will be key to transform these variables so they can be more useful.



<u>Dependant variable analysis</u>

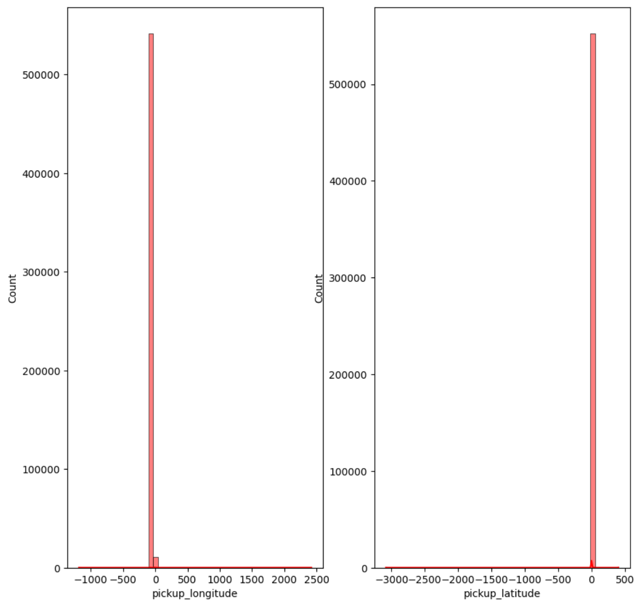
Through the analysis of the value distribution of the fare amount in taxis, it can be easily detected that there are some unusual values in some observations, for instance, values lower than zero. These can be instantly deleted as we know for sure that the value of a cab can't be negative or zero. Also, there are some values that are oddly high which might be true, but for the purpose of this model, those with fare amount >=400 were deleted as they can mislead the results.



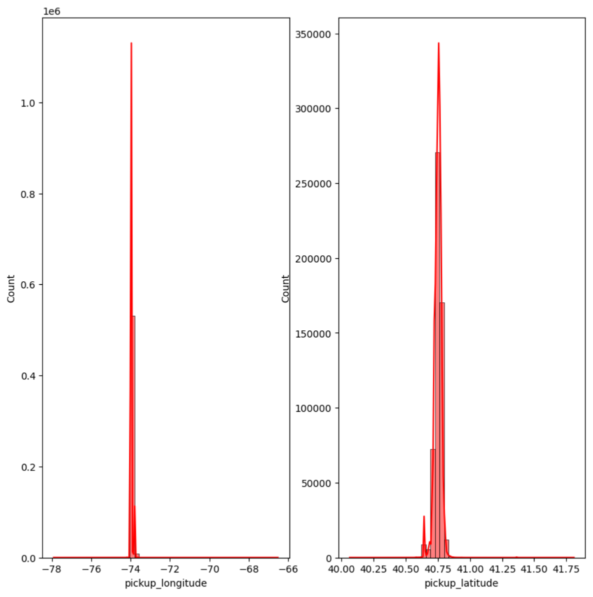
Coordinates

Four variables contained information about coordinates for dropoff or pick up point. These variables were useful to detect observations with irregularities in data:

- There were some observations with longitude >180 or <-180 which makes no sense
- There were some observations with latitude >90 or <-90 which demonstrates irregularity as well.
- Many observations contains coordinates that don't correspond to NYC, these observations should be out of the scope of the model.

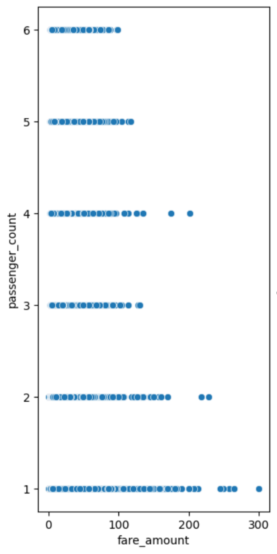


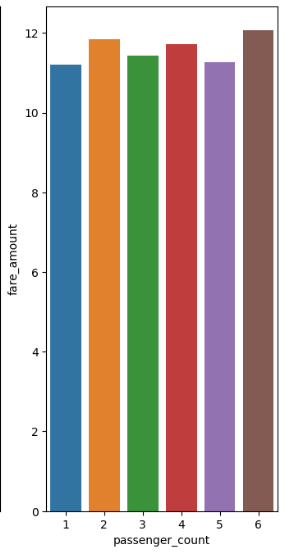
All these observations were removed which meant a big clean of observations in data, but still those are variables with really low variance so should be worked with feature engineering to extract insights from them.



Passenger count

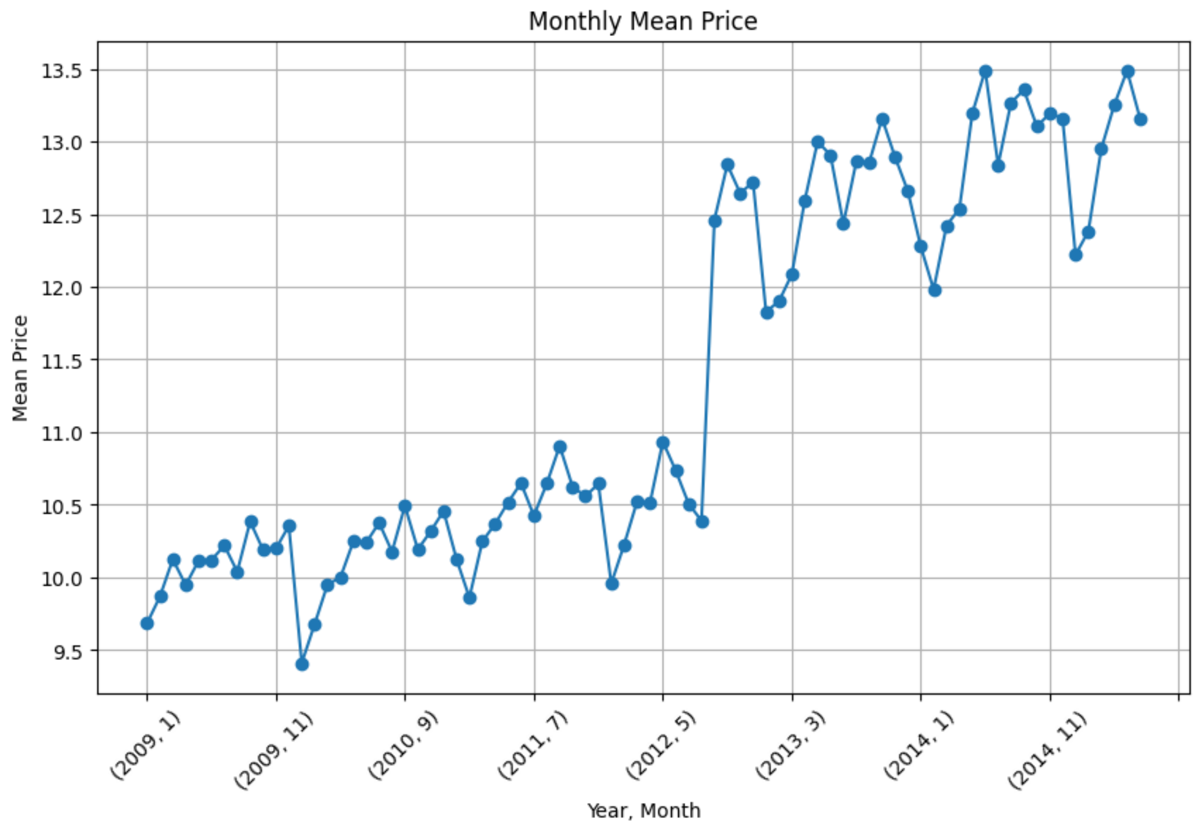
Through the process of EDA, we encountered some observations which had zero passengers, those were deleted as well as those which had >=6 passengers, considered to be a too large amount of passengers. Another analysis performed with this variable, was the average of fare amount by passengers on each cab, no significant insights were gained from it.



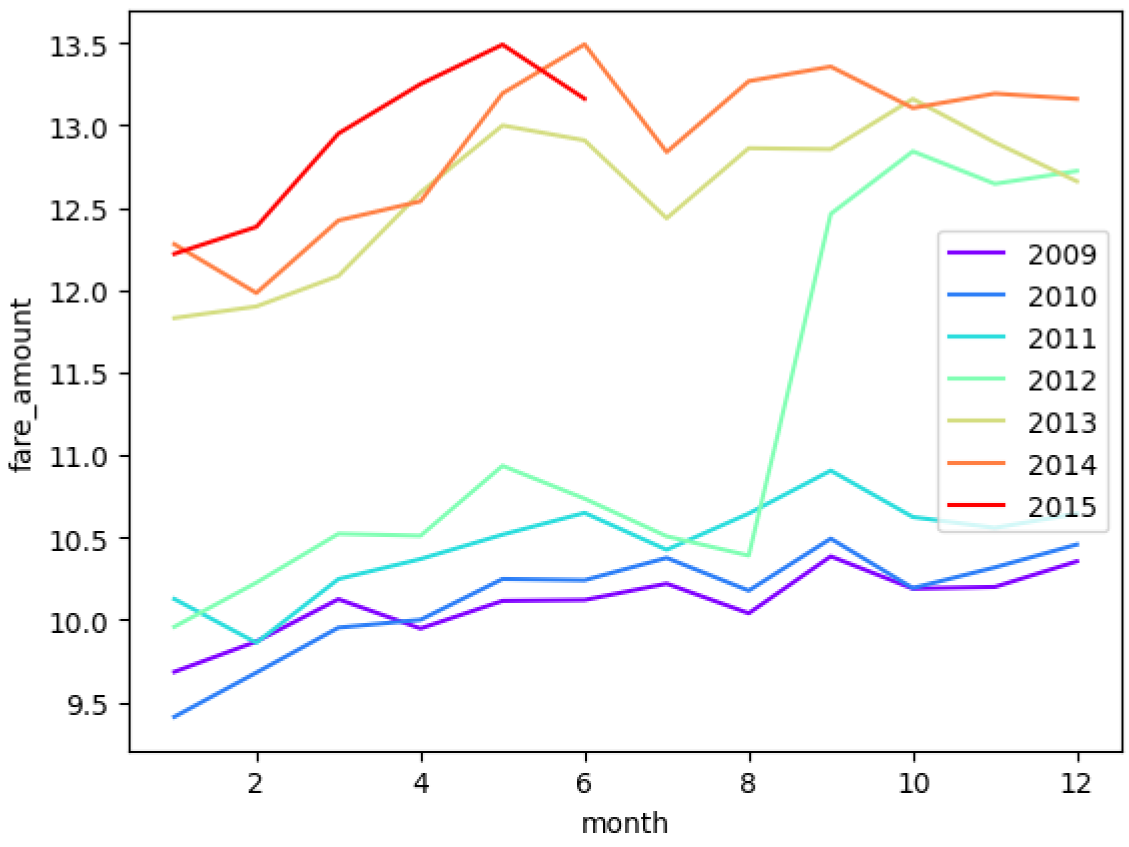


Time Analysis

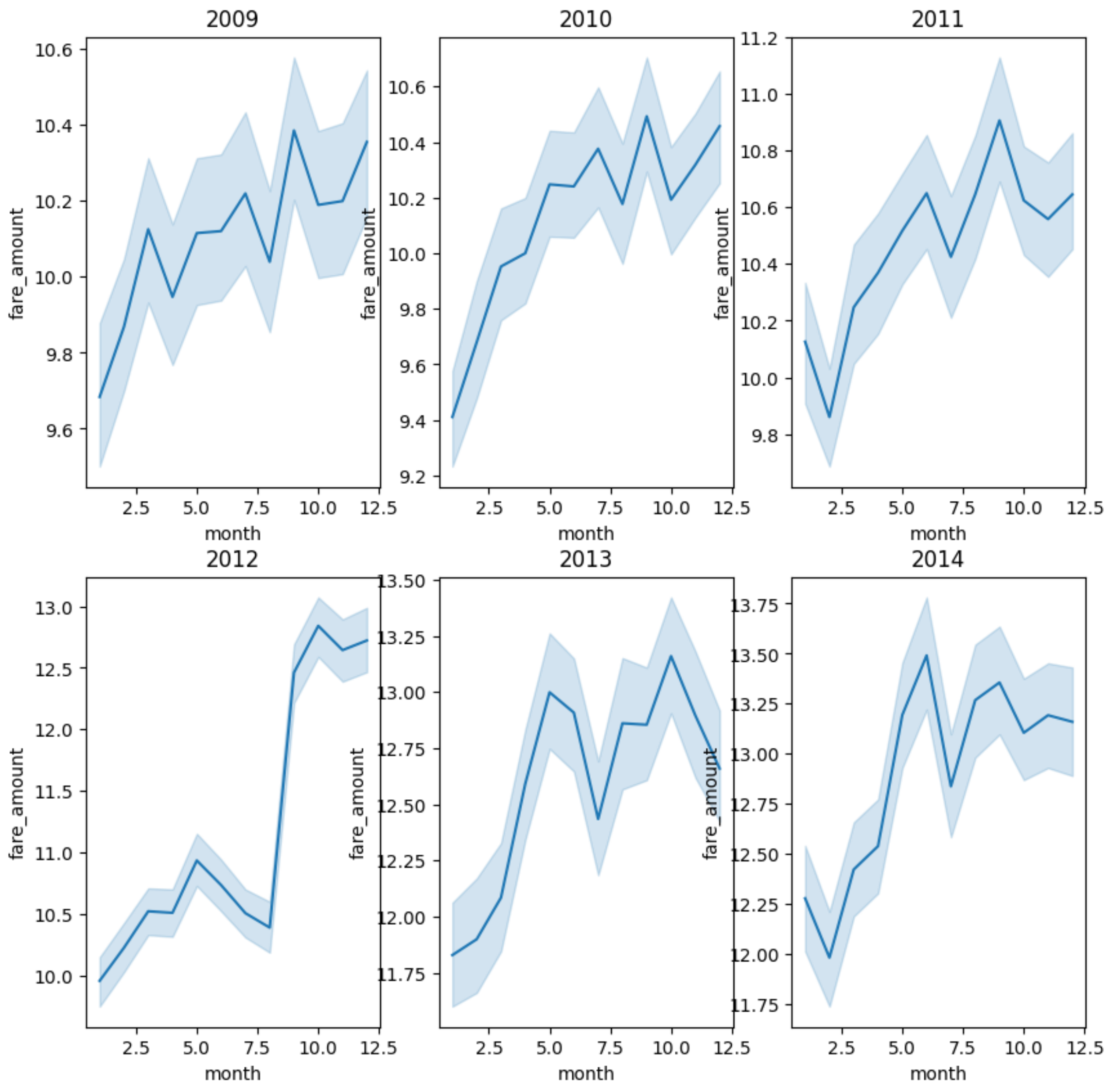
The objective of analyzing the time is to detect any change on fares along the time that covers the dataset, from 2009 to 2014.



According to this plot, there was a big change in a certain month of 2012 which changed the average fare amount drastically. This is good material for a dummy variable which will be significant when training a model.



Besides the big increase in september 2012, fare amount looks to behave similar along the months of the year.



Feature engineering

Given the short amount of variables of the dataset, feature engineering will play a key role in the performance of our machine learning model.

Time feature engineering

Given the pickup datetime, numerous variables can be derived from the timestamp, which will significantly benefit our model. For instance, the cab fare can fluctuate based on variables such as the day of the week, whether it's during rush hour, and more. Among the variables extracted are:

- Year
- Month

- Day of the year
- Weekend indicator (1 for weekend, 0 for weekdays)
- Quarter of the year
- Semester of the year

An additional avenue for feature engineering involving dates involves categorizing hours into distinct parts of the day, such as night, early morning, and prime morning. This categorization allows us to flag specific days and hours of the week, identifying periods like peak work hours or rush times, and even weekend nights when cab demand is notably higher. This granularity in time segmentation can contribute to identifying instances when taxi fares might surge due to increased demand.

Distance

By possessing the pickup and drop-off coordinates, we could compute the distance for each cab ride. Employing the Haversine distance method, we obtained a Python function from an online source and applied it to our dataset. Following distance calculations, observations with unusually high distance but low fare amounts were excluded from our analysis.

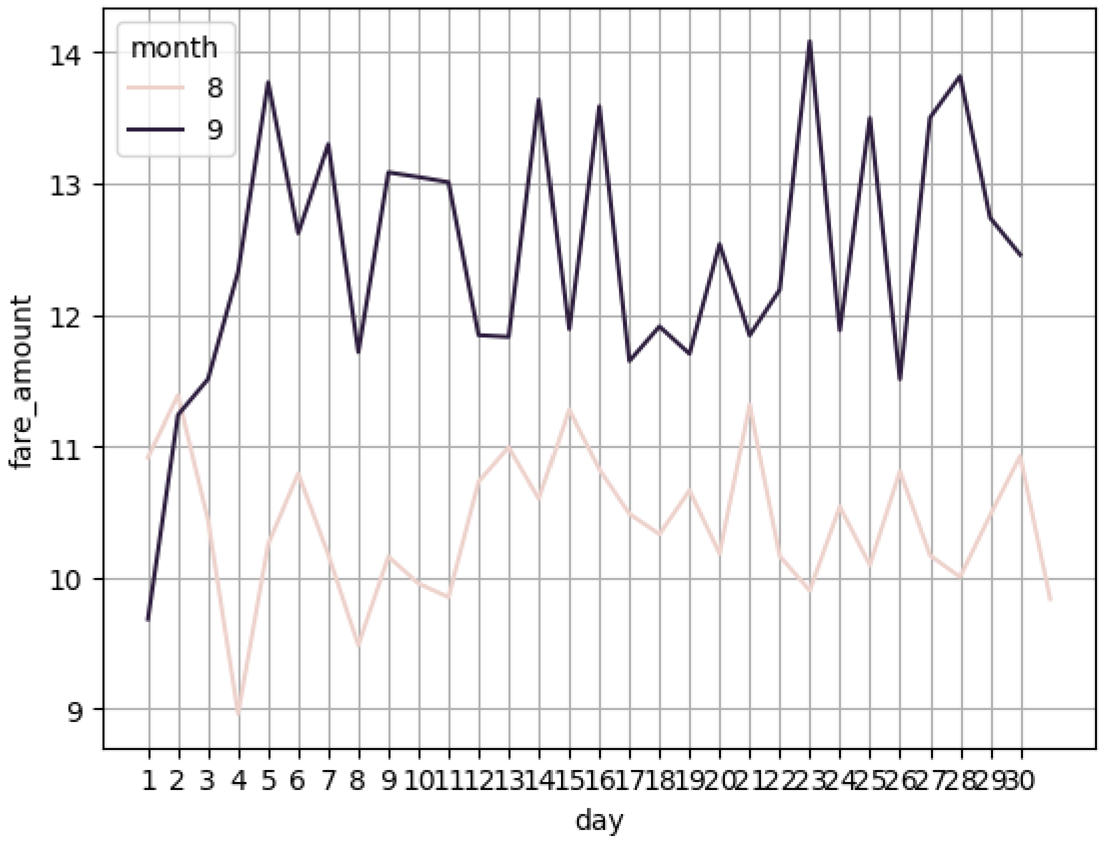
Furthermore, leveraging the distance data, we explored an intriguing feature—measuring distances from pickup points to various high-traffic locations like airports (JFK, EWR, LGA), train stations, and significant landmarks such as the Empire State Building. This additional analysis provides insights into proximity to key destinations driving high attendance.

Weather

Specific weather conditions can significantly impact taxi demand, causing fluctuations in fare amounts. Recognizing this, we incorporated New York City's weather data into our dataset as a variable for our Machine Learning model. This data was sourced from https://www.weather.gov.

2012 fare amount rise

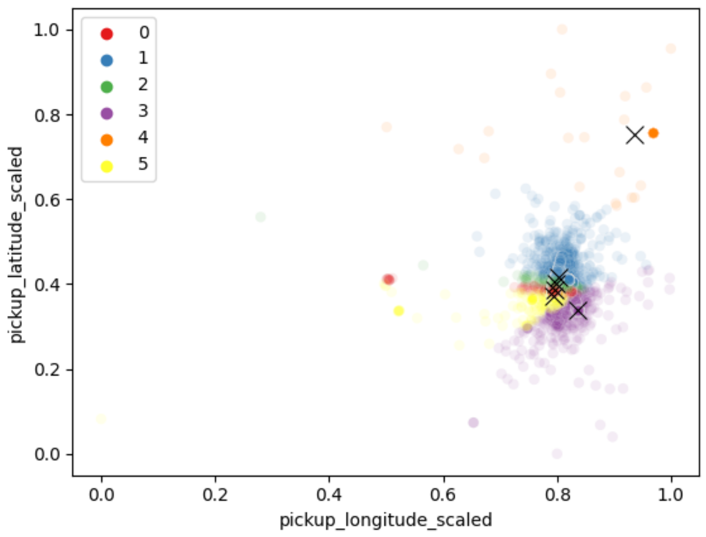
During the exploratory data analysis (EDA) in the Time analysis section, a notable pattern emerged: a sudden and substantial surge in taxi fares occurred in September 2012. Consequently, creating a dummy variable to identify this sharp increase becomes pivotal for our model's accuracy and relevance.



This change is visually noticeable in the plot, highlighting the variable's significance on 9/4/2012, the specific date when the price hike occurred.

K-means

An attempt was made to employ K-means clustering to group observations with similar pickup and drop-off coordinates. Unfortunately, these attempts did not yield distinct aggregations of observations. Nevertheless, despite the lack of clear clusters, the variables derived from this process were retained and incorporated into the model. This decision was made considering the nature of our model, where the inclusion of non-significant variables does not significantly impact its performance.



<u>Model Implementation</u>

To determine the best-fitting model for this data, My first step was to test several models without adjusting their hyperparameters, which means testing them **out of the box**. I calculated their RMSE in validation to understand which models perform better with this dataset.

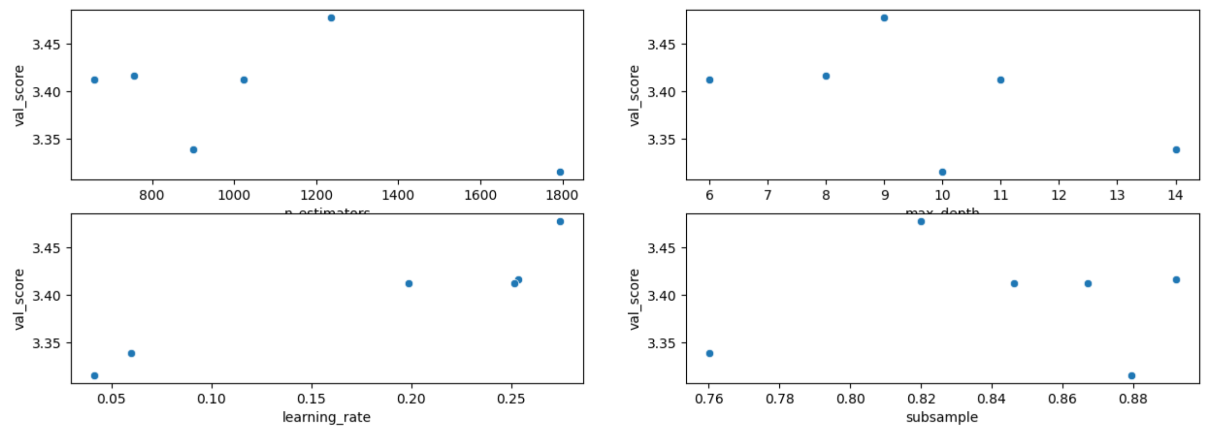
The outcomes clearly favored boosting algorithms, which isn't unexpected given their strong predictive performance with tabular data. However, it's worth noting that while boosting algorithms excel in prediction, they often lack interpretability. Given our priority is prediction, we'll proceed with xgboost, a specific type of boosting algorithm.

XGBoost is an ensemble learning method based on boosting. It sequentially builds models, each correcting the errors of its last models. By focusing on previously mispredicted data points, it creates a series of decision trees to refine predictions. Through a gradient descent approach, XGBoost optimizes a specific loss function, continually improving its accuracy. This iterative process makes it a potent and efficient algorithm, recognized for its predictive capabilities, particularly in tabular datasets.

<u>Hyperparameter tunning</u>

Adjusting the hyperparameters of a complex model like xgboost can be challenging. To tackle this, I employed a random search technique using the function "hyper_random_search" in the code. This method randomly picks values within specified limits for each hyperparameter, enabling the creation of multiple models with different settings. After this task, we analyze the model outcomes to understand what worked effectively, iteratively refining the process to identify the best-performing hyperparameters.

For instance, on the first iteration I got the following results, plotted by my custom function "analyze_results".



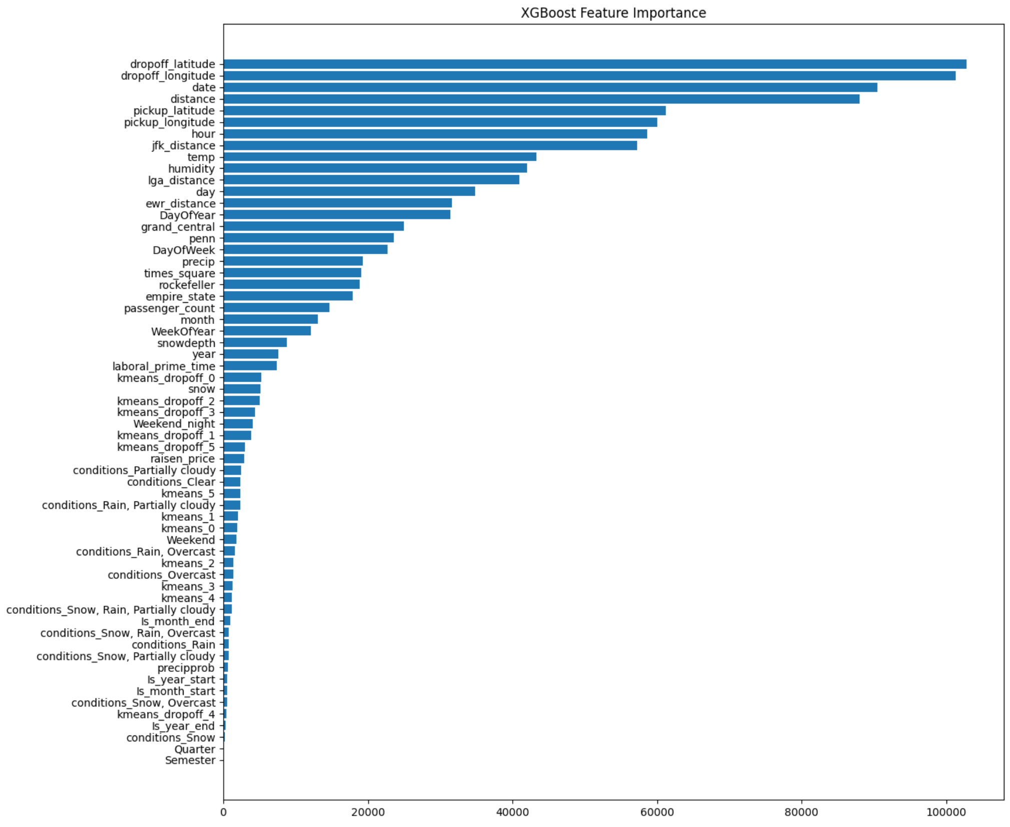
Although we don’t have as many observations (models) as we would like to drive conclusions, it’s worth noting that there is a trend favoring lower learning rates. As a result, subsequent iteration lower learning rates were used adjusting the limits to sample. Overall, taking a look at all the plots, we can get the sense that models with more trees (n_estimators) and lower learning rates perform better, so that is how the iteration went on.

Final model

After going through this process I described, I achieved my highest Kaggle score by using a mix of six XgBoost models. I trained them separately and then combined their predictions by averaging them for the final result. This combining method, similar to bagging, works best when the models' errors aren't consistently in the same direction. If they tend to consistently over or underestimate, averaging might not give the best result, unlike when using the single best model.
After implementing this approach, I managed to reach an RMSE of 4 on Kaggle. While the top 50 performers achieve around 2.5, this score is reasonable given specific limitations I'll detail in another section.

Feature Importance

To evaluate the feature importance, I got the feature importance from the best model although I used the six of them to predict.



To highlight key findings from the graph:

- Dropoff and Pickup coordinates showed unexpectedly strong performance. I suspect further feature engineering might have diminished their effectiveness, but I couldn't explore this due to computational constraints.
- Ride distance and distance from pickup to specific highly frequented locations proved to be valuable in predicting cab fares.
- Unfortunately, variables related to KMeans clustering and weather (except temperature) didn't yield successful outcomes.

Limitations

Working with this dataset was challenging due to its immense size. I could only utilize 10% of the dataset, which contained 55 million observations. This restriction was necessary due to limitations in my computational capacity, resulting in a lower performance compared to many participants in the Kaggle competition. That's why achieving an RMSE of 4 doesn't seem unfavorable to me. Despite using a small fraction of the dataset, computations were time-consuming. After several iterations and achieving an acceptable score, I discontinued the project. I believe there's room for improvement with better computational resources, as handling this dataset requires more power than what I currently have at my disposal.