### Group 1: NYC taxi fare prediction

Guilherme, Alex, Rakshit, Seonhye, Michael

### **Introduction**

New York city taxis are still one of the most popular and common forms of transportation despite the rise of Uber and Lyft. Unlike Uber and Lyft, NYC taxis do not have the fare estimation feature: it’s imperative that customers have an idea of taxi fares in this digital age. We introduce taxi fare prediction where we utilize 55 million rows worth of data to predict taxi fare. For this project, we explored preprocessing, EDA, Modeling, and will discuss future work.

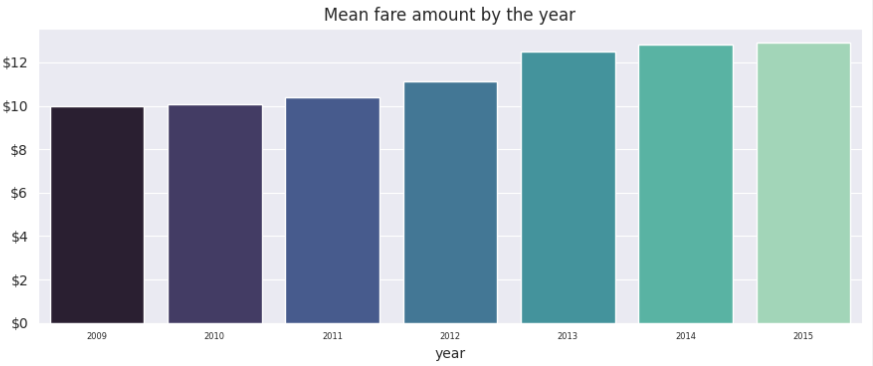
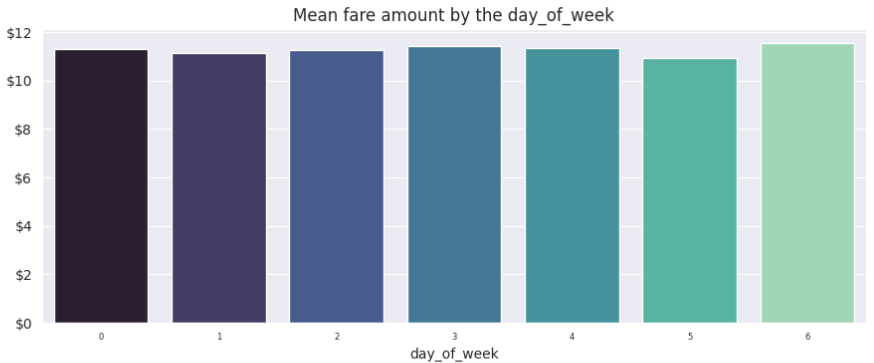
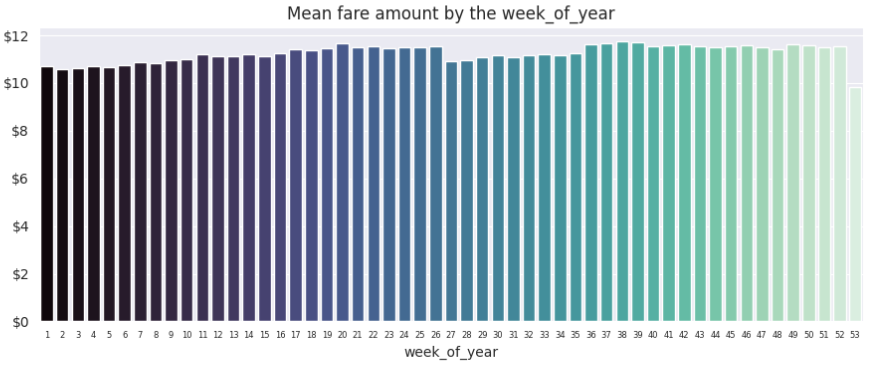
### **Preprocessing**

Since it’s critical to utilize “good” data, we decided to clean the raw data. For example, we dropped rows that were deemed “fake” such as rows with passenger counts indicated as 0 and 6 or more passengers. Given that taxis in New York City have a flat rate of $3, we dropped rows where the fares were less than or equal to $3 and fares higher than $150. Additionally, we dropped rows where the distance was less than 1km.

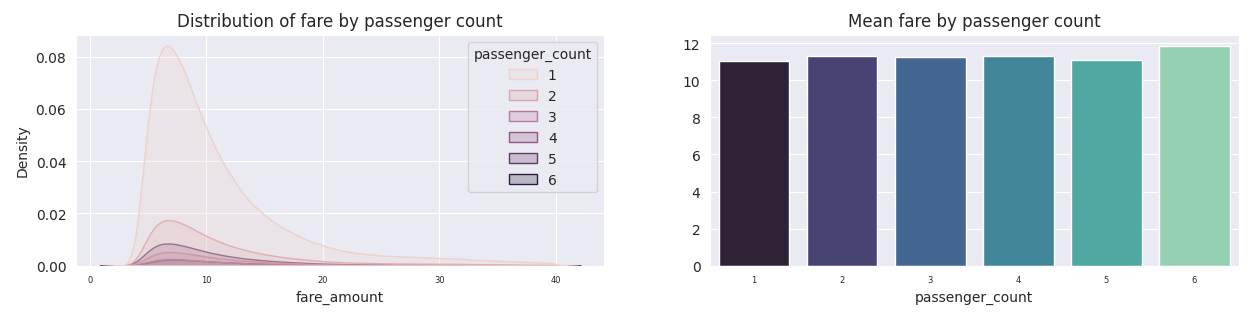
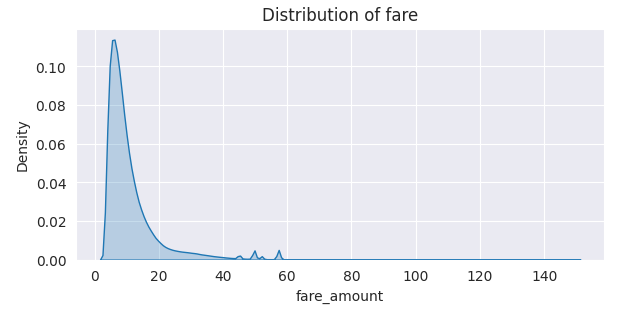
We labeled the pickup and drop off locations, and we also computed the geographical distance between the pickup locations and drop off locations. Lastly, we dropped the unused columns.

### **Exploratory Data Analysis**

The results are as follows:

**Figure 1**

The graphs above were average fare prices grouped according to hour, days, week, and year. Average fare was most expensive during the early mornings such as 4am-5am which is interesting because that is when there is the least amount of demand: average fare was consistent throughout the days and weeks: average fare increased throughout the years.

**Figure 2**

According to the graph above, the majority of the passengers were single passengers and the average fare by passenger count was consistent. Most travels were short term distances and most fares were less than $60.

### **Modeling**

The features are as follows; pickup date, pickup longitude, pickup latitude, dropoff longitude, dropoff latitude, and passenger count, respectively. The target variable is fare amount.

Three models were implemented; linear regression, random forest, and XGBoost. These three models are ideal for continuous variable predictions. For linear regression and random forest, Scikit-Learn and CuML versions were used. The model objective was to minimize the RMSE of taxi fare prediction. In fact, two measures were taken: RMSE penalizes larger errors and we report MAE for human understanding.

For GPU acceleration, we used CuDF and CuML libraries developed by NVIDIA for work on NVIDIA GPUs. CuDF is a GPU-accelerated dataframe library that is API-compatible with Pandas, and it utilizes similar pandas syntax with GPU acceleration. In addition to this, CuML is a machine learning library of machine learning algorithms optimized for GPU acceleration designed to work seamlessly with CuDF. The file size and device memory constrained how much data we could utilize which required us to use only 10% of given data to utilize CuML. We attempted to use pytorch but failed entirely.

### **Results**

**Figure 3**

Different machines have similar results; XGBoost performed the same nearly for CPU due to auto-parallelization. The best model was XGBoost which had the best MAE of **1.85** and R^2 score of **0.84**, and ran efficiently compared to random forest. The linear model did perform close to XGBoost and it did run faster with MAE of **3.26** and R^2 score of **0.78**. Linear models are more suitable for tasks demanding speed. GPU was fastest for training and suitable for tensor-like data, but harder to get devices with required memory. Although the best MAE is 1.85, it is still a large error for a fare prediction and needs improvement.

### **Future Work**

For future work, it would be ideal to utilize machines with more memory for large datasets and properly tune models with such hardware. It would also be good to use Google maps api to try GPS distances and traffic levels as features. Lastly, we would do more precise EDA and cleaning such as removing outliers and breaking down pick-up and drop-off locations more finely.

### **Github**