SWIFTPICK Web Application
Business Report

CS-GY 6513: Big Data

Name:
Ping Chang (ypc231),
Kevin Lee (kl3642),
Guan Cherng Lin (gl2547),
Wei-heng Lin (whl318)

Table of Contents

Abstract

This project focuses on creating a web application that empowers users to input specific time windows and geographical locations to obtain insights into average trip costs and strategic recommendations for drivers to maximize profitability. The motivation behind this project stems from the increasing demand for data-driven decision-making in the ride-sharing industry. As the volume of data generated by these platforms continues to grow, there is a compelling need for efficient tools that can handle large datasets and extract meaningful insights.

The overarching goal is to provide ride-sharing companies with a robust solution that not only processes data efficiently but also translates it into actionable recommendations for drivers. By creating a user-friendly web application, we aim to provide access to valuable insights, enabling both technical and non-technical users to make informed decisions.

The report will delve into the technical aspects of the project, including the architecture, implementation details using Pyspark, incorporation of Matplotlib for data visualizations, machine learning to predict values, front end integration using HTML, CSS, and JavaScript, as well as bridging backend data with Flask. The development of this solution represents a significant step toward leveraging big data technologies to enhance operational efficiency and profitability in the ride-sharing sector.

Introduction

In the dynamic landscape of urban transportation, the advent of big data has opened unprecedented avenues for enhancing the efficiency and customer experience of taxi services. Our project introduces an innovative big data web application tailored to revolutionize the taxi industry. By leveraging a rich dataset of past taxi ride records and integrating real-time location data, our application stands at the forefront of predictive technology in transportation.

The core functionality of our application revolves around two critical aspects: accurately predicting the cost of taxi rides and intelligently guiding drivers to optimal pickup locations. This dual approach addresses both customer and driver needs, ensuring a harmonious balance between service quality and operational efficiency.

Our project is not just a technological innovation; it is a step towards redefining urban mobility. By harnessing the power of big data, we are poised to create a more efficient, reliable, and

customer-friendly taxi experience. The following sections will go into the details of our application, outlining its design, functionality, and the potential impact it holds for the future of taxi services.

Technologies and Workflow

Data Collection

Our primary dataset comes from the New York City Taxi & Limousine Commission (TLC), which provides extensive records of taxi trips, including details such as pickup and drop-off locations, times, distances, and fares. This rich dataset forms the backbone of our analysis, offering real-world insights into taxi service patterns.

Data Storage

We employed MongoDB to store our data. MongoDB is a sensible data store to use as it is document-oriented, which integrates well with our taxi data. As it is distributed, it enables efficient data storage and retrieval, as well as the capability to execute simple operations to extract insights during exploratory data analysis. Being a popular platform with a large development community, we may be able to seamlessly merge with other services such as Apache Kafka should we need to in the future.

Data Processing and Collaboration

With the data in hand, our next step was to organize and analyze it. We utilized JupyterLab, a versatile web-based interactive development environment, for initial data exploration and manipulation. To facilitate collaboration among our team members, we also employed Google Colab notebooks. Google Colab's cloud-based platform enabled seamless sharing and collective editing of our work, ensuring that all team members could contribute effectively, regardless of their location.

Data Visualization

To make sense of the trends and patterns in our data, we employed Matplotlib, a widely-used Python library for data visualization. Using Matplotlib, we created a series of graphs and charts that illustrated key insights from the historical taxi data. These visualizations helped us better understand passenger behaviors, peak demand times, and other critical factors influencing taxi services.

Data Manipulation

For data manipulation and processing, we turned to PySpark, the Python API for Apache Spark. PySpark's powerful data processing capabilities allowed us to handle the large volumes of taxi trip data efficiently. We used PySpark to clean, transform, and structure the data into a usable format for further analysis. Its ability to handle big data processing made it an ideal choice for our needs.

Data Analysis

The culmination of our workflow is the use of K-Means clustering, an unsupervised machine learning method, for in depth analysis of strategic taxi pickup locations and expected revenue during selected hours. K-Means was applied on a window of historical data using geographical coordinates to determine potential hotspots for passenger pickups. This analysis was combined with further data manipulation to determine the average potential revenue in each hotspot. Our analysis provides drivers with data-driven recommendations on where to be and when.

Front End Integration

Our team wrapped up our data into a web application using the fundamental triad of web development: HTML, CSS, and JavaScript. In the application, users can input their designated locations and times, and the application will give back the most popular pick-up locations based on their inputs with estimated earnings for each resulting pick-up location. The connection between the frontend display and the backend data manipulation processes was bridged by Flask, which receives and throws HTTP requests to the frontend applications, and processes the requested data in the backend.

Data Collection

The data used for this project was collected from the New York Government website, more specifically, the NYC Taxi & Limousine Commission page. At the time of data collection, only data from January to September for 2023 was available. Data prior to 2022 were not considered due to how much impact Covid-19 had on the transportation industry, which will make real-time predictions very skewed. The figure below shows how different the data is in 2020 compared to 2023:

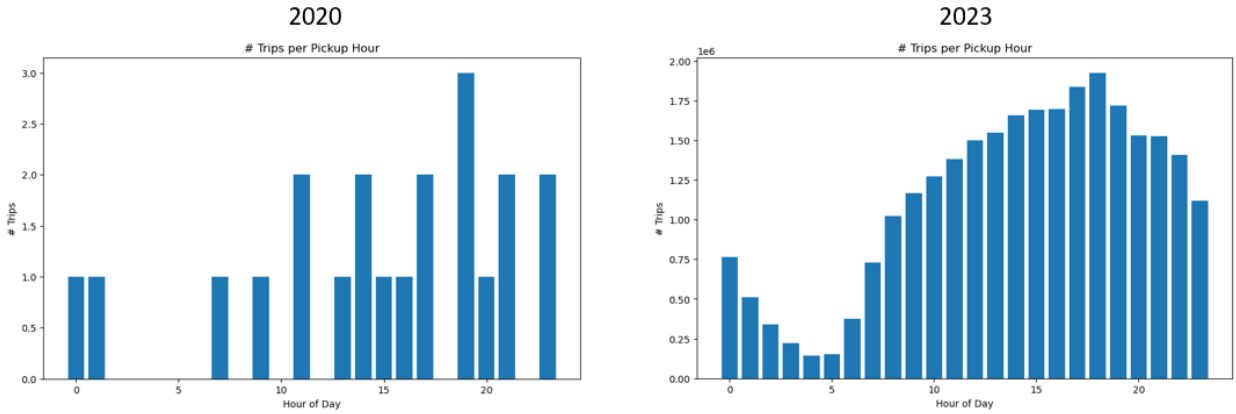

Figure 1: 2020 and 2023 Data Comparison

Only the yellow taxi data was utilized since we want to find the taxi fare for all of New York that are not bound by area restrictions. Each record in the yellow taxi dataset represents a single taxi ride, which includes the time and location for pickup and dropoff as well as how much the taxi ride was. Data preprocessing is needed to get the column values into the form where our team can effectively manipulate the data and use it for machine learning.

The taxi data only shows the location ID for the pickup and dropoff locations, our team merged a second dataset, which is the taxi zone data provided by the NYC Open Data, to show the names of the locations. This allows for more visible analysis and less confusion going into data processing. These records are then stored in a MongoDB database for use in further processing and analysis.

Data Visualization

A data exploration stage was performed to visualize the data in more detail, to observe if there are any trends that the team needs to be aware of before any assumptions are made.

Some of the most important attributes for analyzing taxi data are the pickup and dropoff locations, as they represent when records start and finish. The below figure visualizes the number of trips for each location in New York for the dataset of 2023.

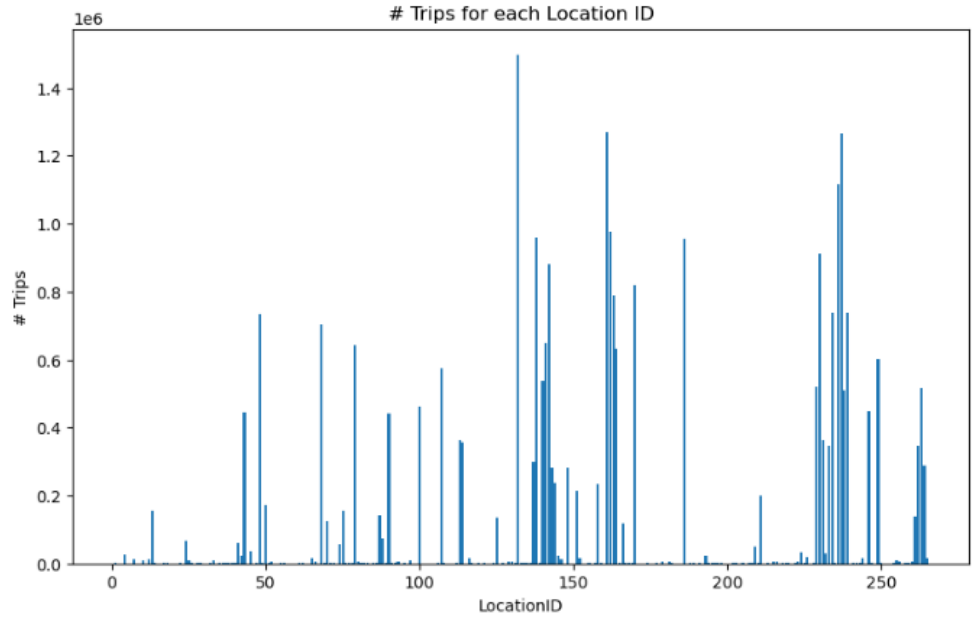

Figure 2: Number of Trips for Each Location

While the above graph is serviceable in showing trends for the number of trips, it is too vague to conduct an analysis on the pickup zones. Therefore, the next figure shows the top ten pickup locations as well as instead of showing the location ID, the names of the zones are provided.

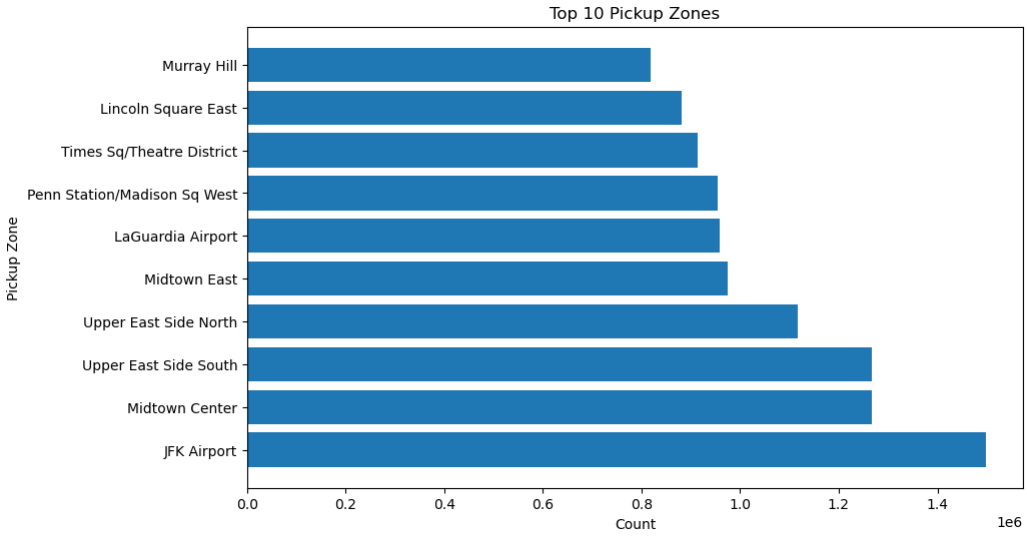

Figure 3: Top 10 Pickup Zones

The figure shows that JFK Airport is the most popular location for pickup, which matches expectations since the only available transportations at the airport are the subways and taxis, with subways taking up much more time to reach a certain location. Looking at the rest of the list, the majority of the pickup zones are either airports or tourist attractions.

Since time is also an important factor in our data processing, another summarization the team may need is the average trip distance by hour of day. The number of trips peak at around 6 pm, which is to be expected as work and school typically end around that time of day.

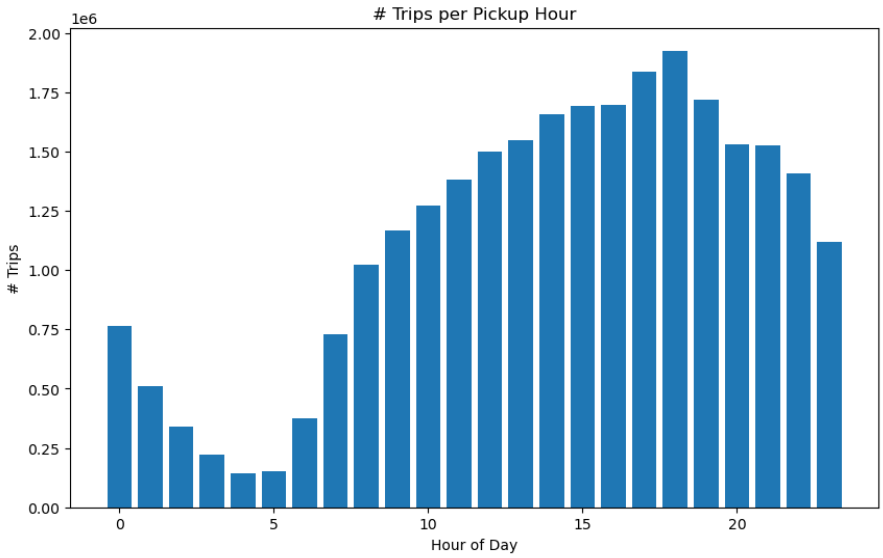

Figure 4: Number of Trips per Pickup Hour

Machine Learning (K-Means Clustering)

Our application leverages machine learning algorithms to analyze patterns in taxi usage, transforming raw data into strategic knowledge. This analysis enables the app to predict demand hotspots and recommend optimal pickup locations to drivers, thereby maximizing their efficiency and earnings potential.

A key component of our machine learning strategy is the use of K-means clustering, implemented through the PySpark.ML package, to identify the most favorable areas for driver positioning. K-means, a popular unsupervised learning algorithm, excels in grouping data into clusters based on the similarities of vectors in a feature space. In our case, we leverage this capability on the slightly different task of clustering the geographical coordinates of taxi pickup

locations to find the most densely clustered hotspots of potential taxi hailers. A unique aspect of our approach is the incorporation of time-specific data analysis. By windowing the historical data to be within an hour of the specified time (± 30 minutes), our application can provide time-sensitive recommendations. This feature is crucial as the demand for taxi services fluctuates significantly throughout the day (rush hours, lunch and dinner hours, etc.). In addition to determining the hotspot clusters, we perform another data manipulation procedure to calculate the average generated revenue across all trips in each cluster membership. This gives the drivers further insight into the profitability of each hotspot location. It should be noted that this analysis does not take into consideration the destination locations for each analyzed trip, where instead the average estimated cost from the hotspot is being displayed. A future improvement could include the average trip length (taking the destination in consideration) to provide further insight to the drivers.

Upon processing, the K-means algorithm identifies clusters representing high-demand areas. These clusters are then analyzed to generate location-specific recommendations for drivers. By positioning themselves in these areas, drivers are more likely to encounter frequent ride requests, reducing idle time and increasing profitability. By integrating advanced machine learning techniques and K-means clustering, our application is not just predicting where passengers might be; it's transforming how taxi drivers navigate the urban landscape, making every minute and mile count.

Web App Architecture

Our full-stack web application is designed to serve as a convenient tool for taxi drivers, empowering them to pinpoint the most profitable pickup locations across New York City. The architecture of our application is a combination of robust frontend and backend technologies, seamlessly integrated to deliver an intuitive and efficient user experience.

Frontend Design

Our team wrapped up the data analysis and machine learning approaches using the fundamental triad of web development: HTML, CSS, and JavaScript. This classic combination allows us to build a user interface that is both aesthetically pleasing and functionally dynamic. The front-end web applications are divided into three main sections, the Home, Application, and Others sections. The figure below presents the Home section of the application, where the key features are presented as well as the purposes of the application and buttons that are linked to other sections. To guide users to different pages depending on their needs, they may click “Start” to go

to the Application section to start their experience or click the other buttons to go to the Others section.

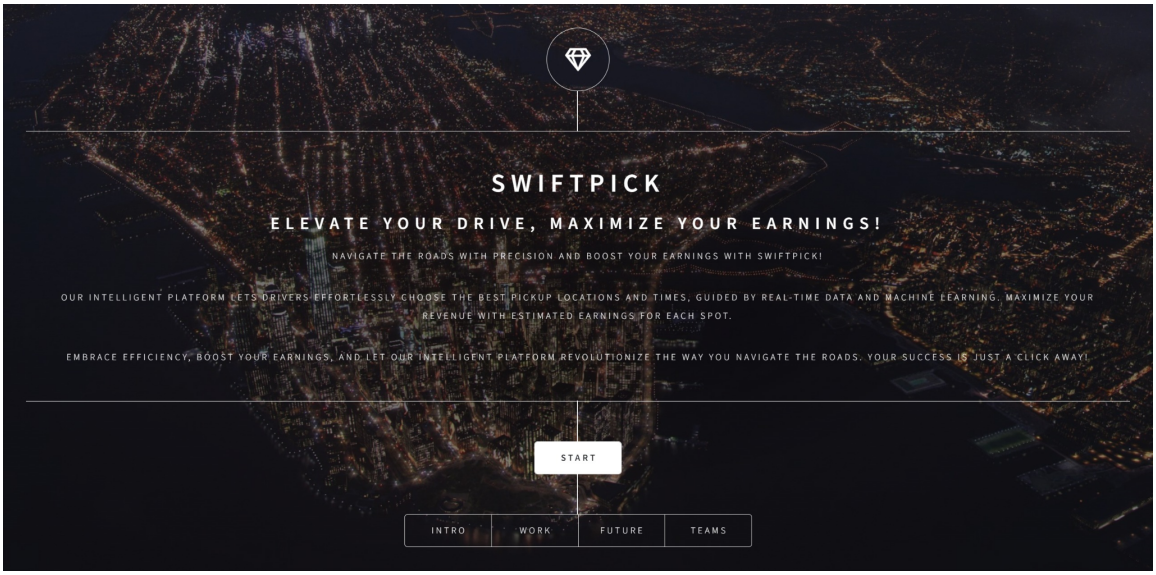

Figure 5: Webpage Home Section

Figure 6 shows the Others section of the application, where the team summarizes the goals of the project, including an introduction, a rough translation of our implementations, some future additions we would add, as well as the team involved for this project:

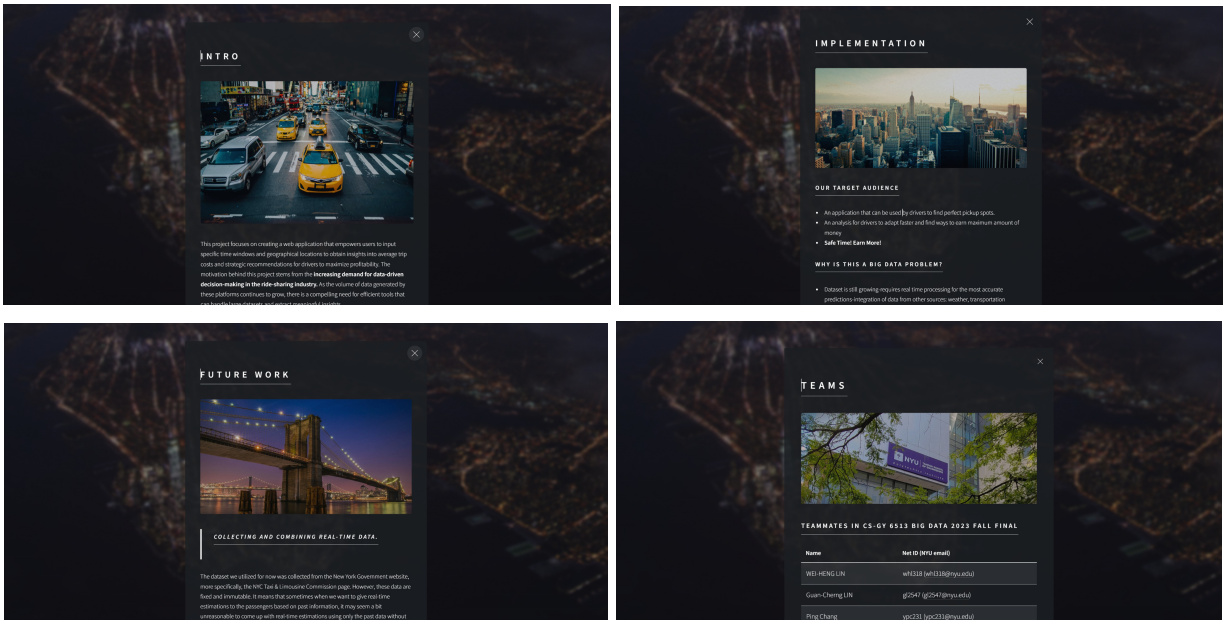

Figure 6: Others Section (Intro/ Work/ Future/ Teams pages)

The figures below dive into the details of the Application section. In this section, the user can select and input the locations of which borough and the zip code they would like to pick up passengers from. They can also get an estimated most popular pick-up location or time of day by selecting a specific date and range of time. Once the user presses “Start Searching”, the application will transfer the inputted data to our backend and show the calculated results on the front end in the interactive map format. The user can click on each mark that is highlighted on the map and see the estimated revenue for every estimated pick-up location. The user can also press “Reset” to hide and reset the map to do another search.

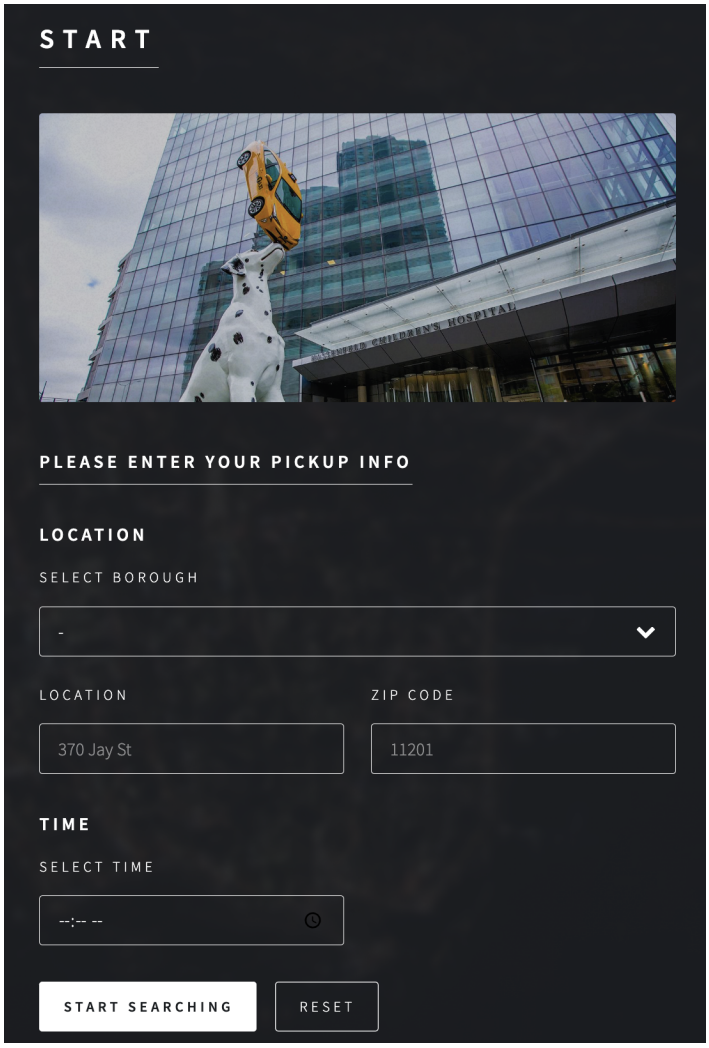

Figure 7: Application Section (Start Page)

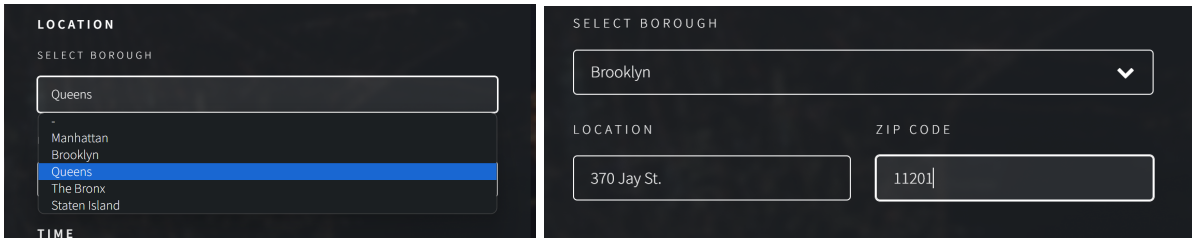

Figure 8: Select and Input System

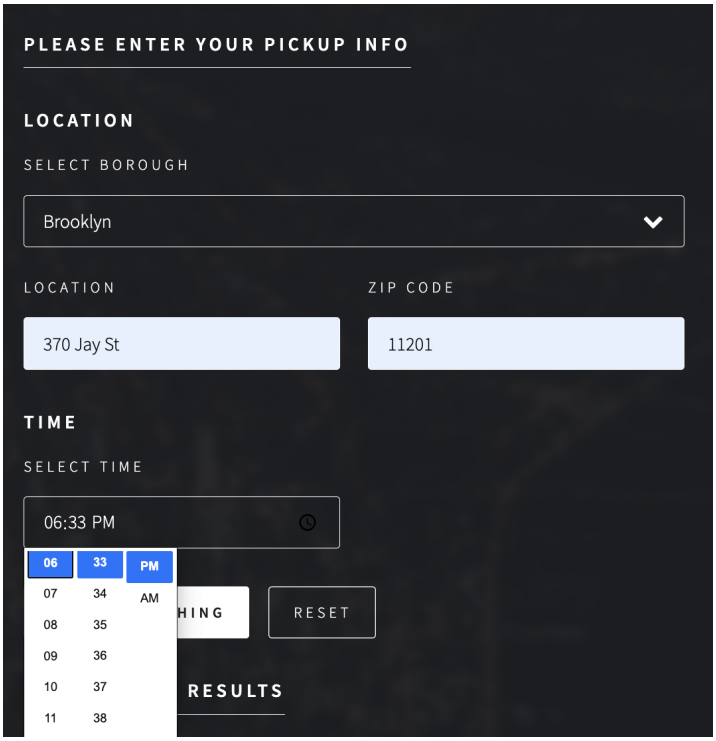

Figure 9: Time Selection

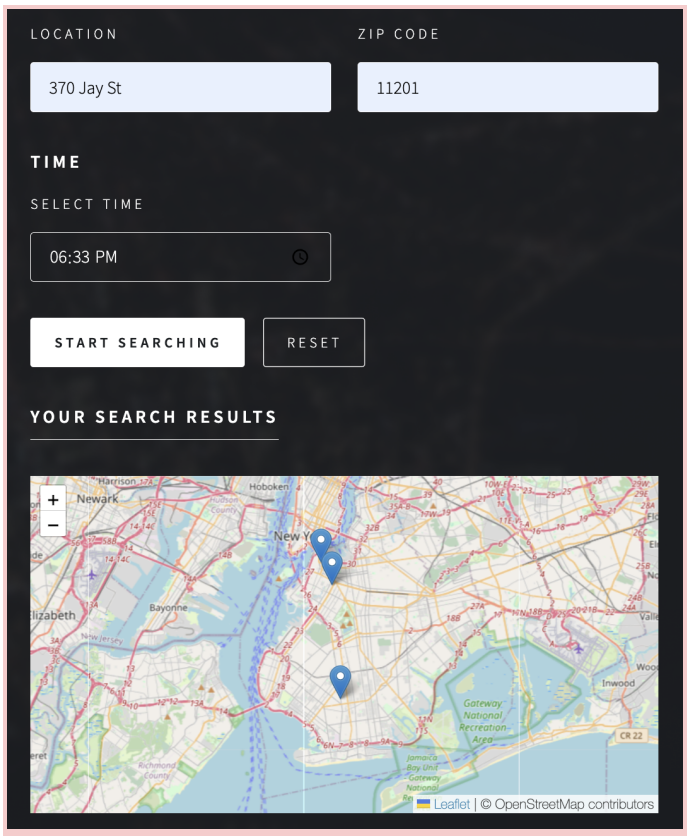

Figure 10: Interactive Map with Estimated Revenue

Backend Integration

Each time a user interacts with our application, by inputting data to search for optimal locations, an HTTP request is generated. This request is directed to our backend, which is powered by Flask. Flask was chosen as a lightweight and efficient web framework for proof of concept in our case. Upon receiving user input, the Flask app performs initial data processing and then passes the data onto the Spark module, where the core of our data analysis takes place.

Result Visualization

Once the Spark module concludes its computation, the results are sent back to the Flask application. Flask then relays these results to the front end.

Interactive Map Visualization: The frontend dynamically renders these results on an interactive map using JavaScript's capabilities and leaflet (an open-source library for creating interactive maps). This visualization enables drivers to easily comprehend and locate the suggested high-demand areas, thereby enhancing their decision-making process and operational efficiency.

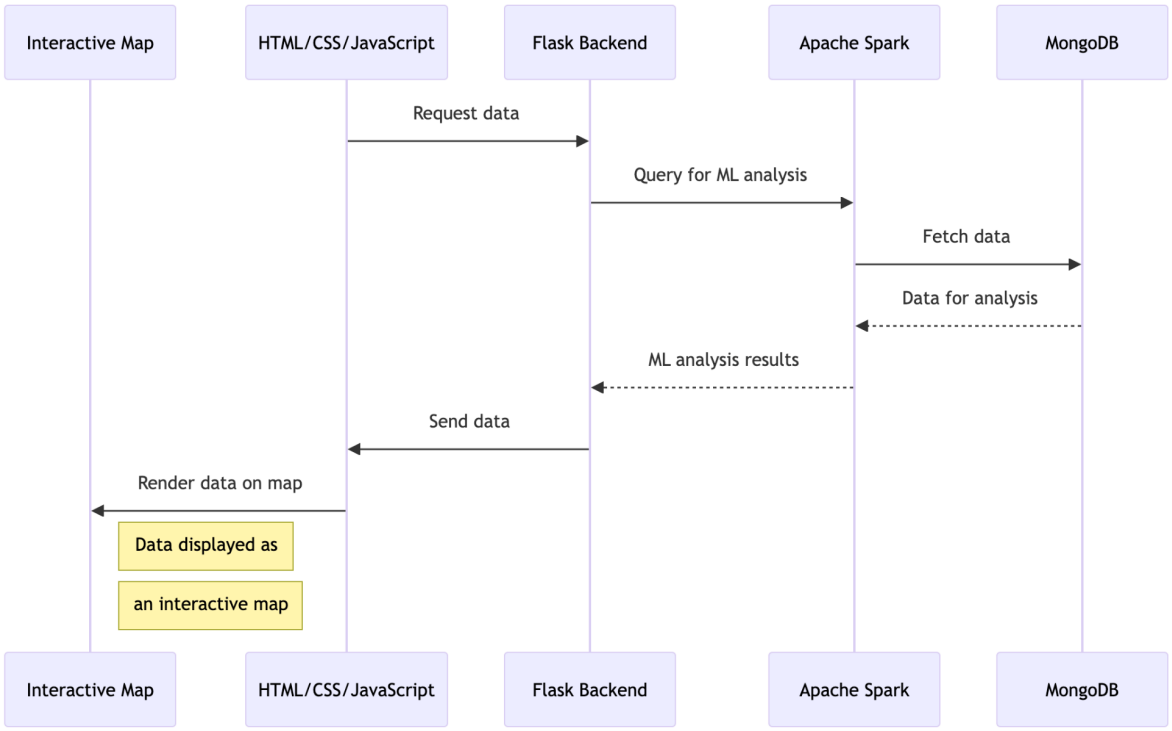

Figure 11: Frontend-backend workflow visualization

Results

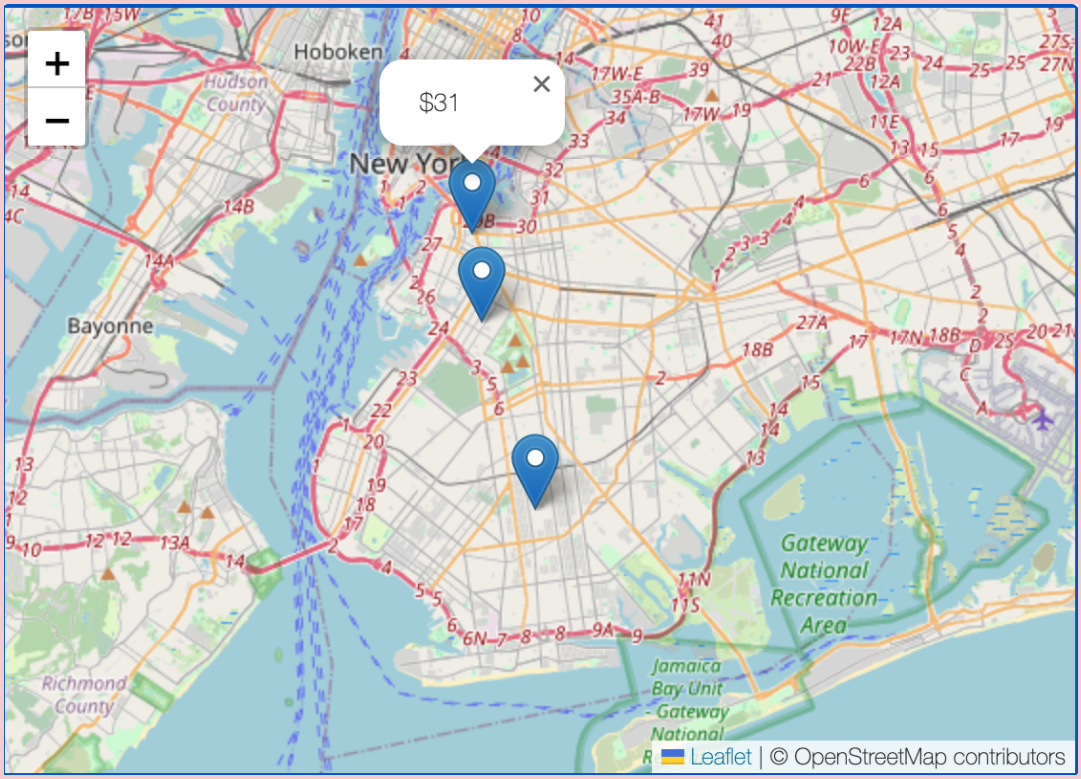

Figure 12: Interactive Map Showing the Top 3 PickUp Hotspots in Brooklyn

The interactive map shows the top 3 most profitable pick-up spots in Brooklyn based on the proximity to the address that the user has entered.

Future Implementations

In the next phase of development, our application will see significant enhancements aimed at further elevating its functionality and user experience. First, we would integrate real-time data collection and processing. The dataset used was fixed and immutable. It may be slightly unreasonable to present real-time estimations based on only past information without having data from the closest time frame. During the process, we did actually try to find some real-time data from NYC authorities, such as MTA (Metropolitan Transportation Authority) and Google Maps API to get real-time location and traffic information or OpenWeatherMap API and Dark Sky API to get real-time weather information. However, most of the data could not be obtained without a proper account or compensation.

In the event that we do implement real-time data ingestion, it would also be sensible to integrate data stream platforms such as Apache Kafka. Kafka would enable us the capability to efficiently ingest streams of data from multiple sources for analysis and use in downstream tasks. Kafka also has native integration with MongoDB.

Additionally, we plan to expand the scope of data used in processing and machine learning, incorporating a wider array of variables to refine our analytical capabilities. During the team's data processing stage, finding reasonable methods to perform machine learning on basic attributes was difficult as methods such as linear regression and clustering would benefit from having more information. As more details continue to be added to our backend, our team can more effectively process the data to prove more realistic estimations.

Perhaps the most user-centric upgrade will be the introduction of a passenger tab. This new feature is designed to benefit passengers directly, allowing them to identify the most popular pickup spots and thereby improving their overall service experience. These advancements collectively aim to make our application more comprehensive, intuitive, and beneficial for both drivers and passengers in the taxi industry.

Limitations

There are two working versions of the web application included in the submission. One version includes MongoDB in its implementation, and the other version is a PySpark-only version used during the live demo. We elected to create two different implementations due to unexpected behavior encountered when integrating MongoDB with PySpark. This behavior resulted in a performance issue that impacted the user experience, which we aimed to be real-time. We suspect the problem stems from MongoDB lacking support for a performant JOIN operation, and/or an underlying data partitioning issue.

Conclusion

This application not only meets the rising demand for data-driven solutions in the ride-sharing sector but also pioneers the use of predictive technology in urban transportation.

Our solution's core functionality, which revolves around predicting taxi ride costs and guiding drivers to optimal locations, represents a significant stride in enhancing both service quality and operational efficiency. By analyzing past ride records and integrating real-time location data, the application provides a unique dual benefit: it empowers drivers with strategic insights to maximize profitability and ensures a more customer-friendly taxi experience.

The technical exploration of this project includes a sophisticated architecture using Pyspark, Matplotlib for visualizations, and machine learning for predictive analysis. The successful integration of these technologies with front-end development (HTML, CSS, JavaScript) and Flask for backend connectivity showcases our commitment to creating a user-friendly and effective tool for both technical and non-technical users.

References

“Data Dictionary - Yellow Taxi Trip Records.” NYC.Gov. Accessed December 15, 2023. https://www.nyc.gov/assets/tlc/downloads/pdf/data_dictionary_trip_records_yellow.pdf

"NYC Taxi Zones.” NYC OpenData. Accessed December 15, 2023. https://data.cityofnewyork.us/Transportation/NYC-Taxi-Zones/d3c5-ddgc

“TLC Trip Record Data.” TLC Trip Record Data. Accessed December 15, 2023. https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page