Project Report

Real-Time Data Extraction and Machine Learning for Optimized Uber Ride Booking With Continuous Learning

Pramod Kondur

September 19, 2024

Real-Time Data Extraction and Machine Learning for Optimized Uber Ride Booking with Continuous Learning

Abstract

This report presents a comprehensive overview of a project aimed at optimizing Uber ride bookings through real-time data analysis and machine learning techniques, incorporating a continuous learning framework. By leveraging data extraction methods and predictive modelling, the project seeks to enhance user experience by forecasting optimal booking times based on various factors, including ride prices and wait times. This paper explores the project's objectives, methodology, key features, technological components, and future enhancements.

Introduction

The rise of ride-sharing services like Uber has transformed urban transportation. However, users often struggle with fluctuating ride prices and varying wait times. This project addresses these challenges by leveraging real-time data and machine learning models to predict the best times for booking Uber rides. The goal is to provide a data-driven approach that ensures cost savings and improved convenience for users.

Significance of the Project

As urban populations continue to grow, the demand for efficient transportation solutions increases. By optimizing ride booking times, this project can lead to:

Cost Savings: Predicting lower fare times can help users save money.

Reduced Wait Times: Identifying optimal booking periods can minimize waiting for rides.

Enhanced User Experience: Providing users with reliable predictions can improve satisfaction.

Project Features

Continuous Real-time Data Collection: Data is scraped from Uber for seven locations, capturing all possible routes at one-hour intervals from 7 AM to 11 PM.

Database Management: A MySQL database is employed for data storage, facilitating continuous collection through job scheduling.

Geolocation API Integration: Latitude and longitude are calculated using the Nominatim API, and distances are computed via the Open Route Service API to improve model scope outside the scraped locations data.

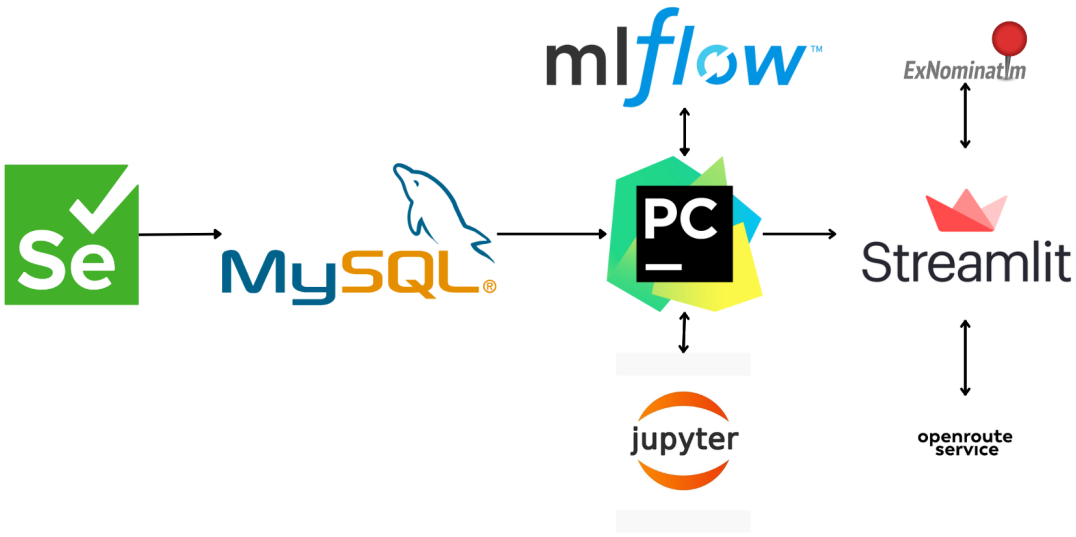
Machine Learning Models: Various ML models, including Random Forest and XGBoost, are trained, tested and tuned on historical data to predict optimal booking times.

Interactive Web Interface: A Streamlit application allows users to select locations and view predictions for future booking times while visualizing maps using Pydeck

Continuous Learning Framework: The system is designed to continuously learn from new data, automatically retraining models to adapt to changing patterns in ride demand and pricing using MlFlow.

Model Architecture and Tools Used:

Model Architechture



Programming Language: Python

IDE: PyCharm and Jupyter notebook

Web Scraping: Selenium

Mapping: Pydeck, Geopy

APIs: Nominatim, Open Route Service

Database: MySQL

Machine Learning Frameworks: Scikit-learn

Version Control: Git, GitHub

Model Logging and Continuous Learning: MlFlow

Methodology

1. Data Collection

Data Scraping:

Data was scrapped from the Uber website using Selenium and is continuously scrapped at 1 hour intervals between 7 AM and 11 PM IST for 7 locations in the city of Chennai, Tamil Nadu and all it's possible routes among them.

Types of Data:

Data collected included

- Ride type (Uber Go, Uber Sedan, Uber XL, Uber Auto, Uber Moto, Uber Premier),
- Maximum ride persons (1,2,3,4,6)
- Route location from
- Route location to
- Ride request date
- Ride request time
- Waiting time (minutes)
- Reaching_time (minutes)
- Ride time (minutes)
- Ride price in Rupees

The following images illustrate a row, total rows x columns and date period of the collected data as of September 19, 2024:

	id	route_from	route_to	ride_type	ride_max_persons	ride_request_time	ride_waiting_time	ride_request_date	ride_reaching_time	ride_time	ride_price
0	1	Chennai Lighthouse	Chennai Citi Centre	Uber Auto	3.0	16:34:22	2.0	2024-09-10	16:43:00	0:06:38	70.00

20544 rows × 11 columns

2024-09-10
2024-09-19

2. Data Preprocessing and Exploration

Data Formatting:

Since data was got at hourly intervals, not all data are got at exactly at the hour as selenium takes a while to scrape all the data from these 42 routes (For example starts scrapping at 9:00 AM but may finish at 9:05 AM or 9:10 AM). Thus we round it off to the hour and if there are multiple occurrences of the same data (I.e. same ride type, locations) then the data is averaged among them to give unique values of data for that time slot. Date formatting and time formatting was done as well

Feature Engineering:

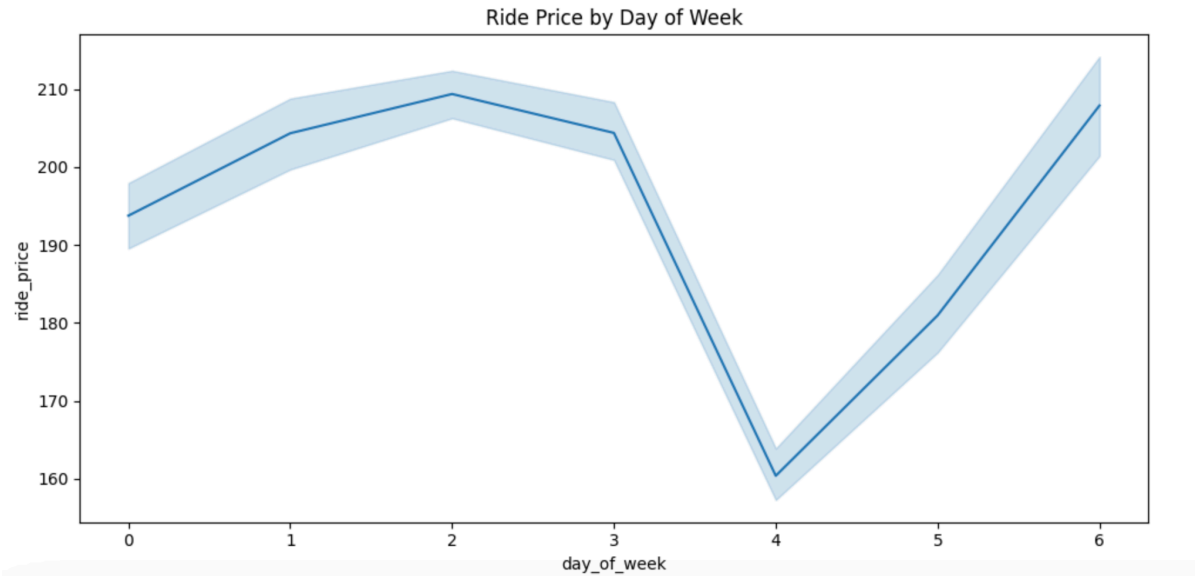
New features, including day of the week, and hour of the day, were added to enhance model predictions while removing unwanted columns.

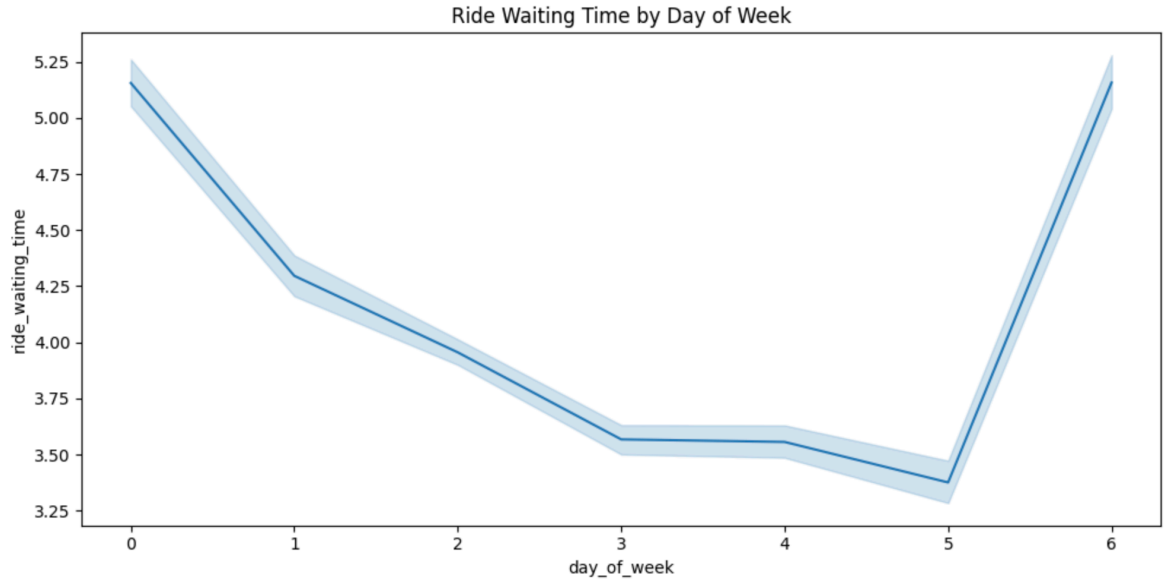
A depiction of the that is shown below.

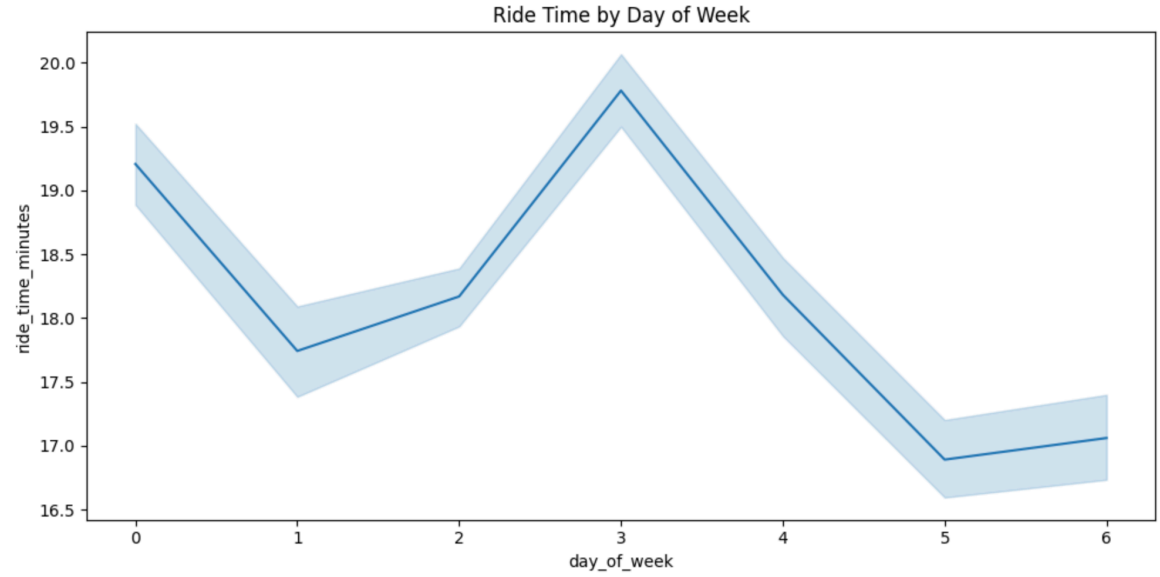
	route_from	route_to	ride_type	ride_max_persons	hour	day_of_week	ride_waiting_time	ride_time_minutes	ride_price
0	Chennai Citi Centre	Chennai Lighthouse	Go Sedan	4.0	17	1	6.000000	7.983333	147.06
1	Chennai Citi Centre	Chennai Lighthouse	Moto	1.0	17	1	5.333333	7.983333	35.416667
2	Chennai Citi Centre	Chennai Lighthouse	Premier	4.0	17	1	6.000000	8.316667	199.183333
3	Chennai Citi Centre	Chennai Lighthouse	Uber Auto	3.0	17	1	1.333333	7.650000	70.0
4	Chennai Citi Centre	Chennai Lighthouse	Uber Go	4.0	17	1	6.000000	8.650000	143.01
...	...	...	...	...	...	...	...	...	...
21439	Semmozhi Poonga	Sai Baba Temple Mylapore	Moto	1.0	13	3	2.000000	17.266667	56.33

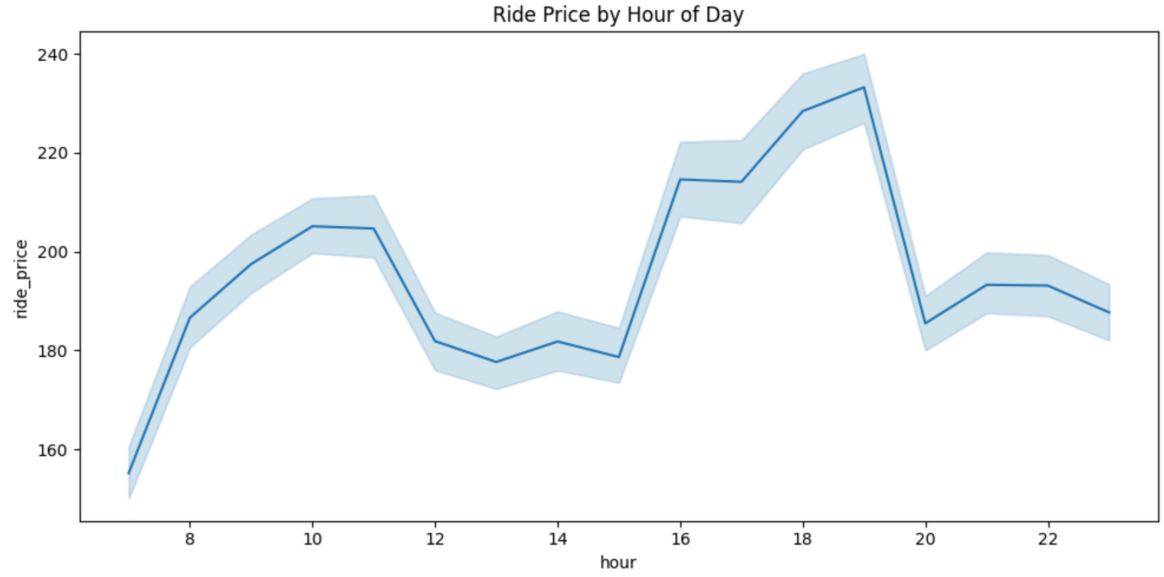
Exploratory Data Analysis (EDA):

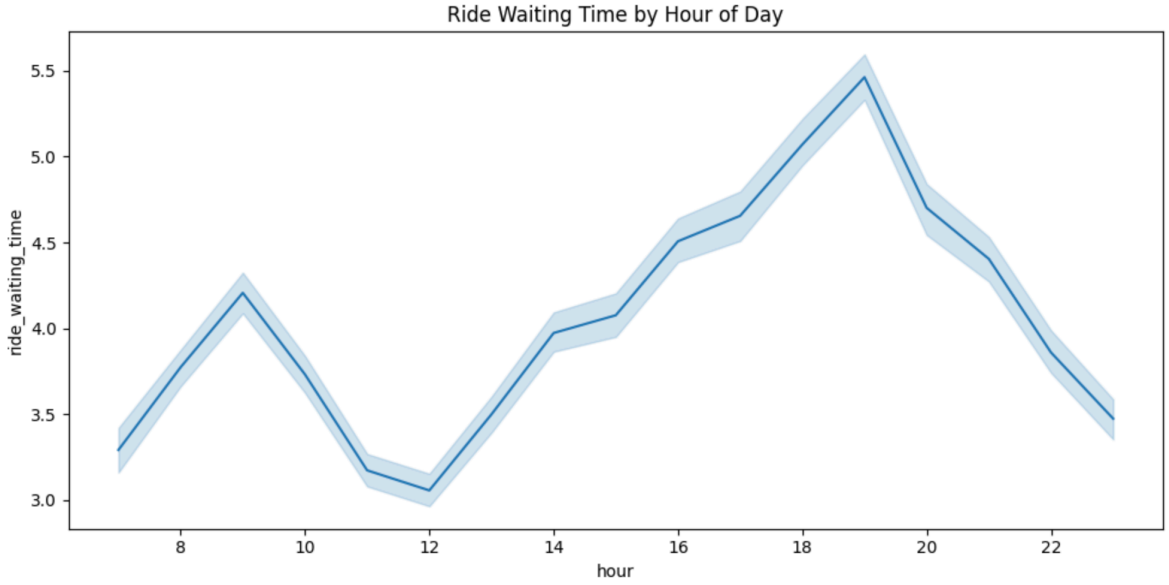
EDA was performed to analyze key metrics such as the distribution of ride times, prices, and waiting times by day of week and hour of day. They are represented below.

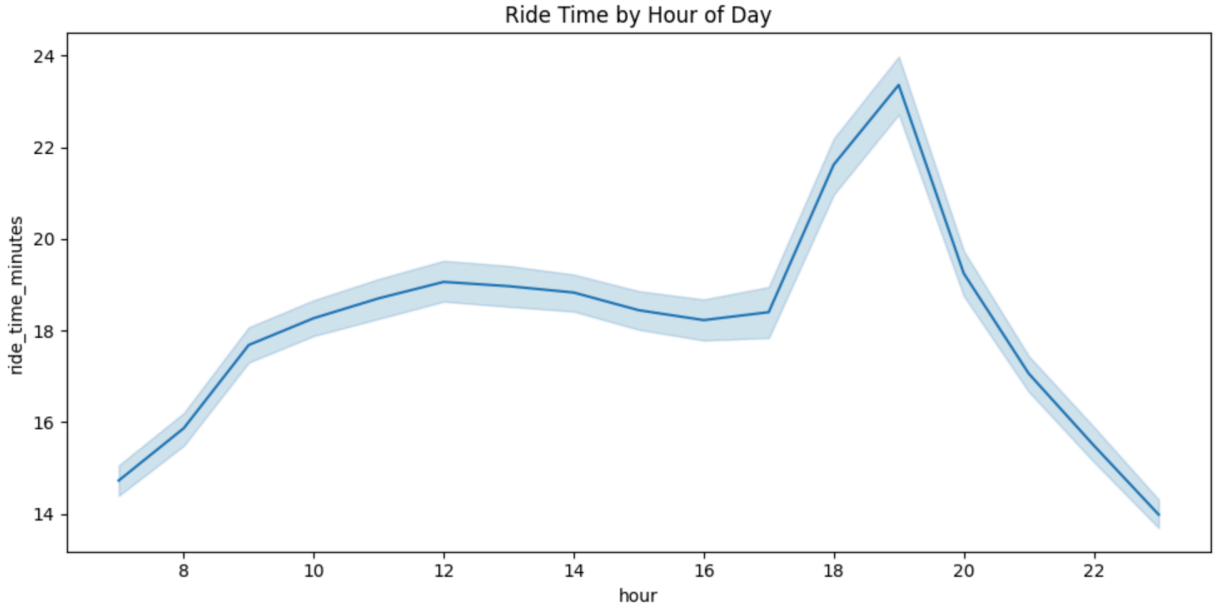

Ride Price by Day of Week
ride_price
210
200
190
180
170
160
0
1
2
3
4
5
6
day_of_week


Ride Waiting Time by Day of Week
ride_waiting_time
5.25
5.00
4.75
4.50
4.25
4.00
3.75
3.50
3.25
0
1
2
3
4
5
6
day_of_week

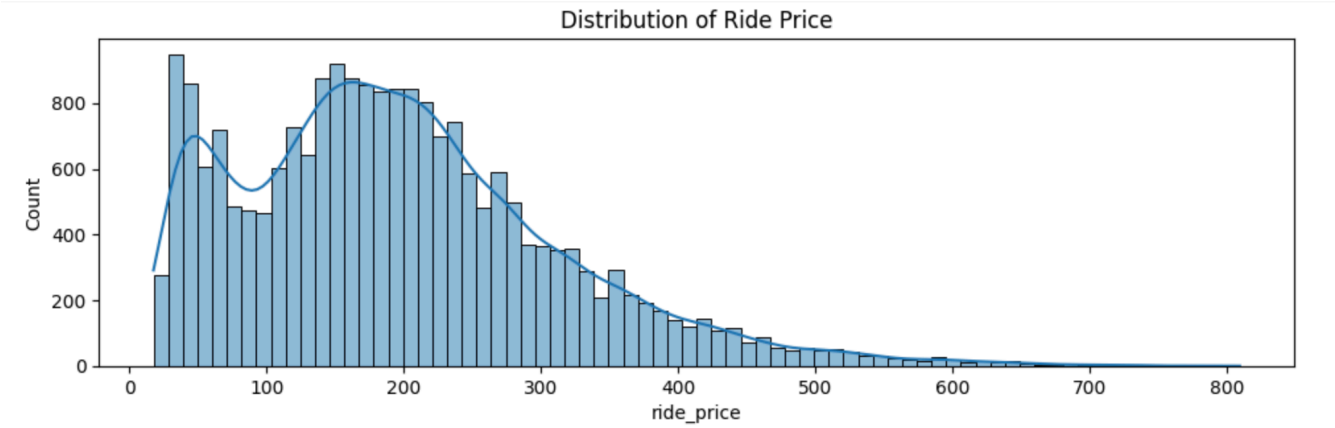

Ride Time by Day of Week
ride_time_minutes
20.0
19.5
19.0
18.5
18.0
17.5
17.0
16.5
0
1
2
3
4
5
6
day_of_week

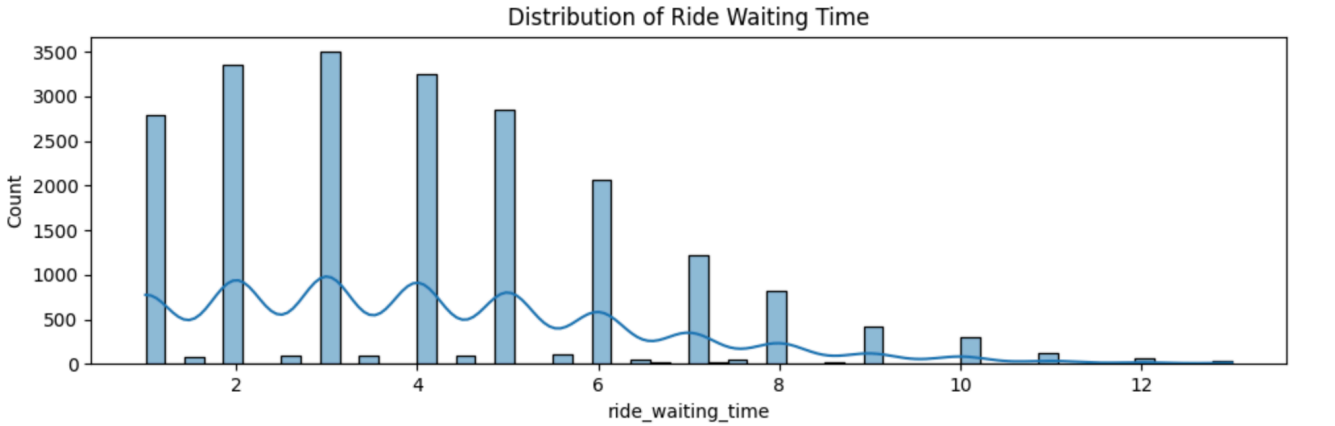

Ride Price by Hour of Day
ride_price
240
220
200
180
160
8
10
12
14
16
18
20
22
hour

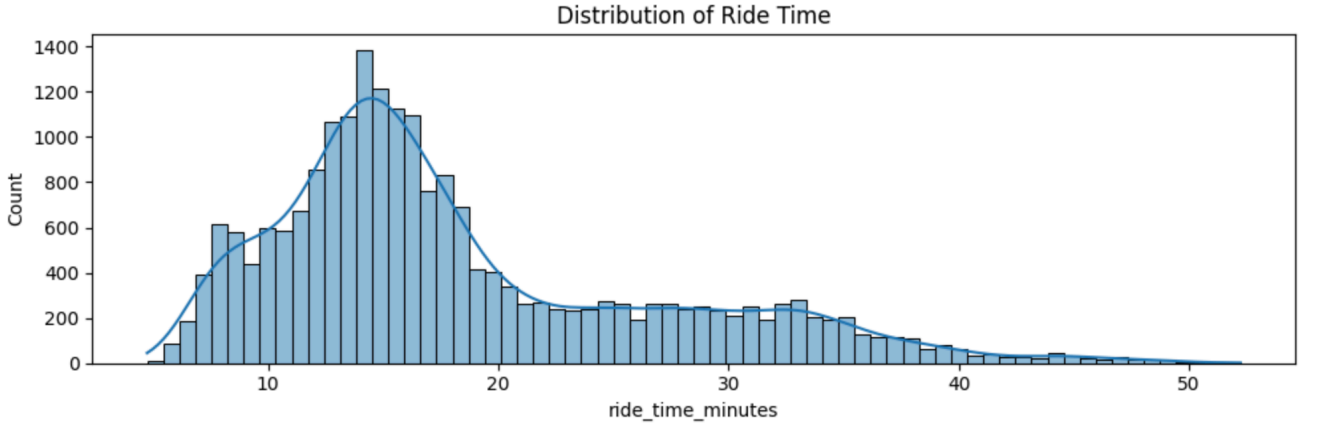

Ride Waiting Time by Hour of Day
ride_waiting_time
5.5
5.0
4.5
4.0
3.5
3.0
8
10
12
14
16
18
20
22
hour


Ride Time by Hour of Day
ride_time_minutes
24
22
20
18
16
14
8
10
12
14
16
18
20
22
hour

Distribution of ride price, ride waiting time, ride time also show us that they are not skewed much and can thus be utilized by the machine learning models. They are shown below.

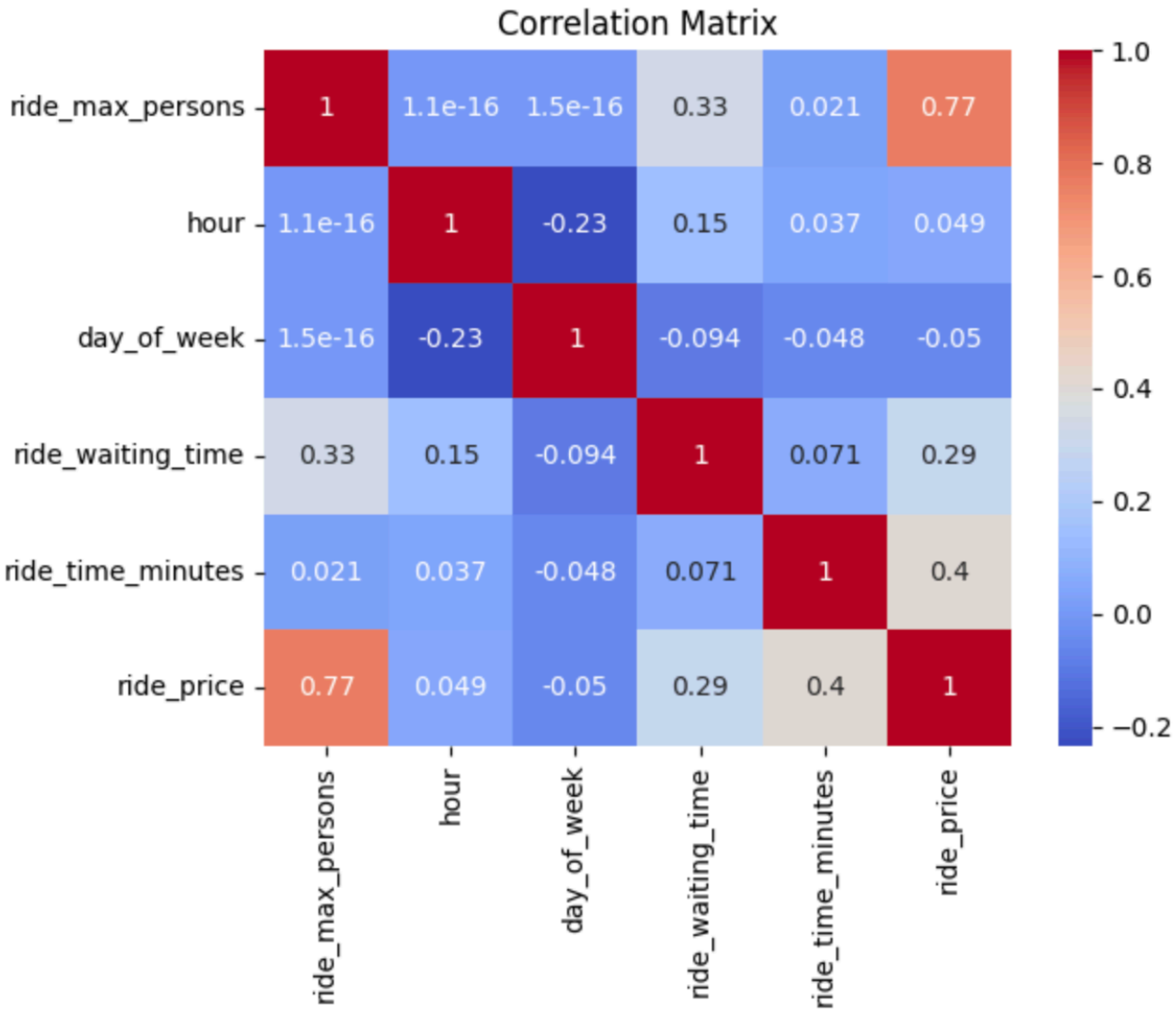






Correlation analysis among the numerical are shown as below.

Although hours, day of week likely affect the price, they are not totally co-related as they are fluctuations among them thus we don't need to drop any columns but proceed with next steps.



3. Model Selection, Training, and Testing

Model Selection:

For our project we went ahead and chose Linear Regression as our base model and ensemble learning techniques like Random Forest Regressor and XGBoost Regressor as they are able to deal with both normal distribution data and even skewed data and since our model is slightly skewed we went ahead and chose these models. Overall, these ensemble technique work the best for linear regression problems.

Model Evaluation:

Model was then trained after conducting a train-test split with encoding the categorical features and scaling numerical features. Model metrics of Mean Absolute Error (MAE), Mean Squared Error(MSE), and R2 Score were utilized as metics

The results obtained were as follows:

```
--- Linear Regression Price Prediction Model ---
Mean Absolute Error (MAE): 33.3068383392167
Mean Squared Error (MSE): 2298.816590761756
R² Score: 0.8087676739605123

--- Linear Regression Waiting Time Prediction Model ---
Mean Absolute Error (MAE): 1.3201258525689152
Mean Squared Error (MSE): 2.902264975361982
R² Score: 0.46864084061963085

--- Linear Regression Ride Time Prediction Model ---
Mean Absolute Error (MAE): 3.0053159556271702
Mean Squared Error (MSE): 16.92624728254662
R² Score: 0.7721016406263651
```

```
--- Random Forest Price Prediction Model ---
Mean Absolute Error (MAE): 10.017634153005465
Mean Squared Error (MSE): 373.0349721128934
R² Score: 0.9688103596681626

--- Random Forest Waiting Time Prediction Model ---
Mean Absolute Error (MAE): 0.38259514907487296
Mean Squared Error (MSE): 0.39639956699581375
R² Score: 0.9291120850372716

--- Random Forest Ride Time Prediction Model ---
Mean Absolute Error (MAE): 0.6325770140286968
Mean Squared Error (MSE): 0.6675361321222005
R² Score: 0.9908885757758246

--- XGBoost Price Prediction Model ---
Mean Absolute Error (MAE): 10.102199054007942
Mean Squared Error (MSE): 314.7897112639407
R² Score: 0.9738134965006694

--- XGBoost Waiting Time Prediction Model ---
Mean Absolute Error (MAE): 0.6035203063037675
Mean Squared Error (MSE): 0.6893802861069517
R² Score: 0.8737852909955264

--- XGBoost Ride Time Prediction Model ---
Mean Absolute Error (MAE): 0.5865173989346546
Mean Squared Error (MSE): 0.6334017144045959
R² Score: 0.9914717533586962
```

From the observations, we can clearly see that Random Forest performs the best, while Linear Regression performs the worst due to it's nature of not being able to capture non-linearity in data.

Model was hyper-tuned using RandomizedSearchCV for different values of no. of estimators, max depth of the model, min samples split, and min samples leaf.

Results indicated that the default model performed the best. Thus we can utilize that.

4. Further Improvement

Why?

So why do we need further improvement? The Current model does give us good performance metrics but it is limited to only the 7 locations and the 42 unique routes among them. In order to expand the scope of our model, we need to go beyond just the 42 routes. In order for our model in real world scenario where we can use these data for multiple locations in the city we can consider utilizing geo-locational features

Geo-locational features using APIs:

Thus latitudes and longitudes of these locations and the distances between the 41 routes were calculated using APIs. Nominatim API was used to obtain latitudes, longitudes while Open Route Service API was used to get the distances among these routes. Results are as follows:

	name	Latitude	Longitude
0	Express Avenue Mall	13.058821	80.264103
1	Chennai Citi Center	13.043025	80.273870
2	Chennai Lighthouse	13.039716	80.279442

	route_from	route_to	distance_meters
0	Express Avenue Mall	Chennai Citi Centre	2912.4
1	Express Avenue Mall	Chennai Lighthouse	3699.5
2	Express Avenue Mall	Marina Beach	3359.9
3	Express Avenue Mall	Semmozhi Poonga	2948.5
4	Express Avenue Mall	Sai Baba Temple Mylapore	3833.1
5	Express Avenue Mall	PVR Ampa SkyOne	7417.2
6	Chennai Citi Centre	Chennai Lighthouse	1902.5
7	Chennai Citi Centre	Marina Beach	3266.2
8	Chennai Citi Centre	Semmozhi Poonga	2701.3

5. Model Training and Testing on New Data

Different Train-Test Split:

Train-Test split was not done randomly but by completely hiding two locations and all it's occurrence among the two and also any occurrence between it and other locations in the proposed train set, thereby these locations and it's routes become completely new, unseen data to the model.

Training, Testing, and Evaluation: The results at this step are as follows:

```
--- Linear Regression Price Prediction Model ---
Mean Absolute Error (MAE): 34.33033067279943
Mean Squared Error (MSE): 2437.434861980726
R² Score: 0.7989533156845213

--- Linear Regression Waiting Time Prediction Model ---
Mean Absolute Error (MAE): 1.430394182301531
Mean Squared Error (MSE): 3.4763925383341405
R² Score: 0.39232712842875617

--- Linear Regression Ride Time Prediction Model ---
Mean Absolute Error (MAE): 2.9083422974865845
Mean Squared Error (MSE): 15.57062256755357
R² Score: 0.7762077078263596

--- XgBoost Price Prediction Model ---
Mean Absolute Error (MAE): 23.841591777242794
Mean Squared Error (MSE): 1184.3443473853758
R² Score: 0.902311849295486

--- XgBoost Forest Waiting Time Prediction Model ---
Mean Absolute Error (MAE): 0.842777184224963
Mean Squared Error (MSE): 1.3628716457509094
R² Score: 0.7617702496412159

--- XgBoost Forest Ride Time Prediction Model ---
Mean Absolute Error (MAE): 2.7410356842318997
Mean Squared Error (MSE): 11.621104222346432
R² Score: 0.8329730529255069
```

```
--- Random Forest Price Prediction Model ---
Mean Absolute Error (MAE): 21.91002510281385
Mean Squared Error (MSE): 998.9461468641157
R² Score: 0.9176040296422323

--- Random Forest Waiting Time Prediction Model ---
Mean Absolute Error (MAE): 0.6894978354978355
Mean Squared Error (MSE): 1.0576721711159212
R² Score: 0.8151190700371859

--- Random Forest Ride Time Prediction Model ---
Mean Absolute Error (MAE): 2.442126379870129
Mean Squared Error (MSE): 9.683781697262054
R² Score: 0.8608176587970612
```

Random Forest performs the best and although the performance is slightly lower than our model using location names instead of geo-locational features, the model still performs very well and thus we can utilize this model as it increases the scope of our project beyond the limited number of locations to choose from.

6. An Interactive Web Application

Web App:

In order for users to be able to utilize what we have created we leveraged Streamlit for them to be able to use the application in real-time.

Location Restriction:

Since the data collected was only from 7 locations in the city of Chennai, the best method was to restrict the selection of locations for from and to to be within 20 km of the mean of the other 7 locations. That way we can get accurate predictions of our values.

Dynamic geo-locational features:

Since we are using new locations, we need to fetch their corresponding latitudes, longitudes, and distances as well. This was dynamically done using the Nominatim API and Open Route Service API.

Routes Mapping:

Further, routes of selected locations was also mapped to give a visual appearance to the users of the road route using Open Route Service API.

Error Handling:

Errors were also handled if location exceeds the 20km radius and if there are invalid addresses.

Prediction of Value:

Upon selection of features, the app generates ride price, ride waiting time, and ride time for the selected date and hour. It also provides the values for the next three hours with percentage change and colour coding to help users with selecting the best ride enabling cost savings, convenience, and satisfaction.

7. Continuous Learning Framework

New data = New Model:

To ensure that the model remains effective over time, a continuous learning framework is implemented. This involves:

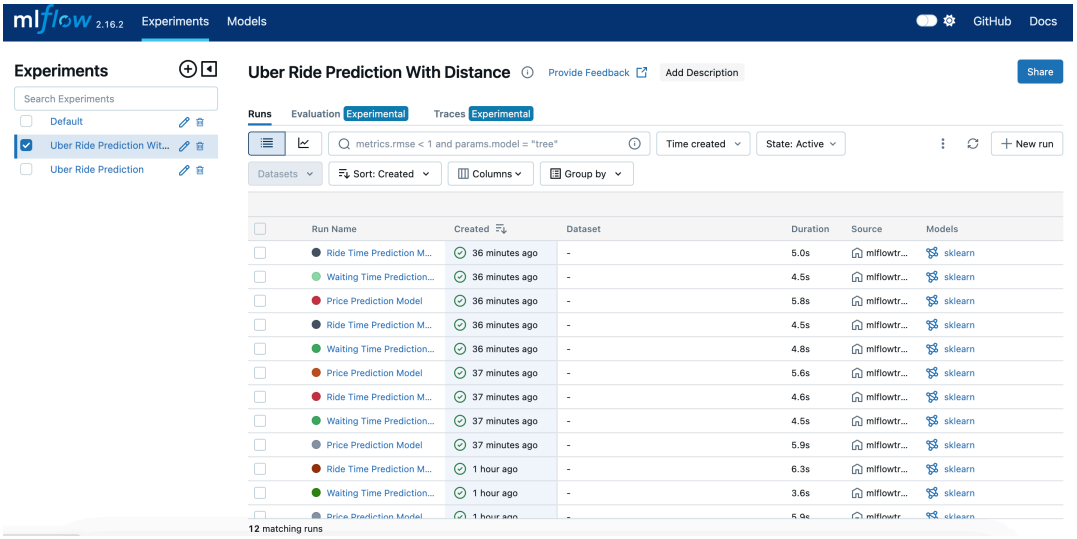
Scheduled Retraining:

Models are automatically retrained on new data that is collected daily at 9 AM, allowing them to adapt to the latest trends and patterns.

Model Performance Tracking and Best Model Selection:

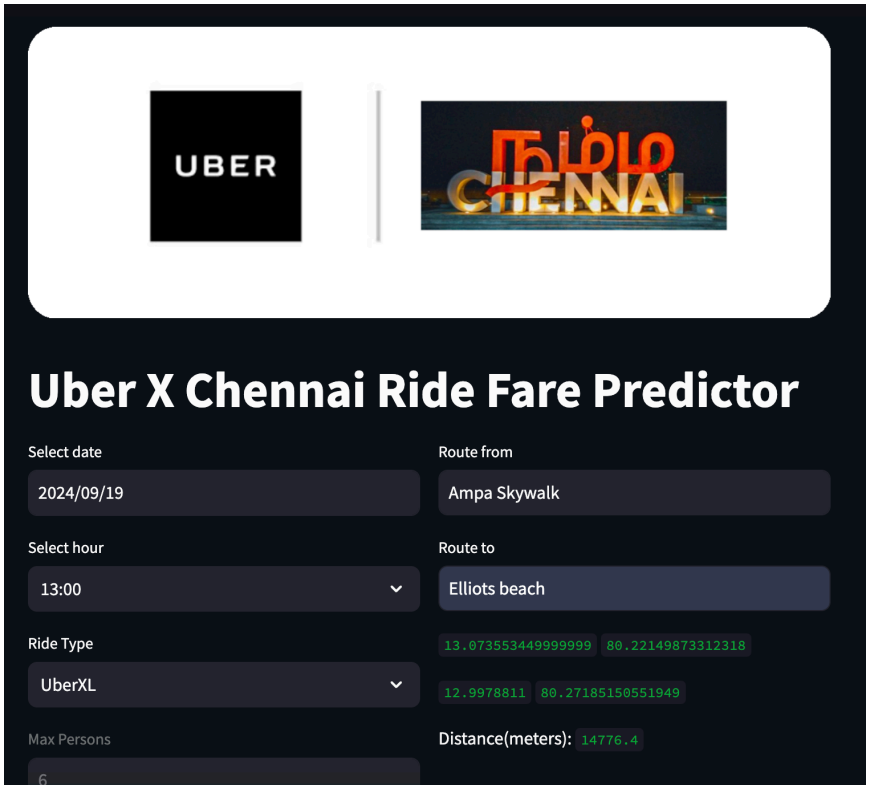
Using MLFlow, the performance of various models is tracked, while automatically selection the best-performing model.

The MlFlow application UI is show below to see it's structure



8. Demo

A demo of the application is shown below, it shows features selected, it's latitudes and longitudes with distance and the predicted values, a route map, and also the 7 locations used for modelling



Start location is within 20km radius.

End location is within 20km radius.

Predict

Predictions for the Selected Time:

Price: Rs. 465.76

Waiting Time: 3.02 minutes

Ride Time: 35.52 minutes

Predictions for the Next 3 Hours:

Hour: 14:00

Price: Rs. 428.65 (-7.97%)

Waiting Time: 3.09 minutes (2.54%)

Ride Time: 31.63 minutes (-10.96%)

Hour: 15:00

Price: Rs. 423.80 (-9.01%)

Waiting Time: 3.09 minutes (2.43%)

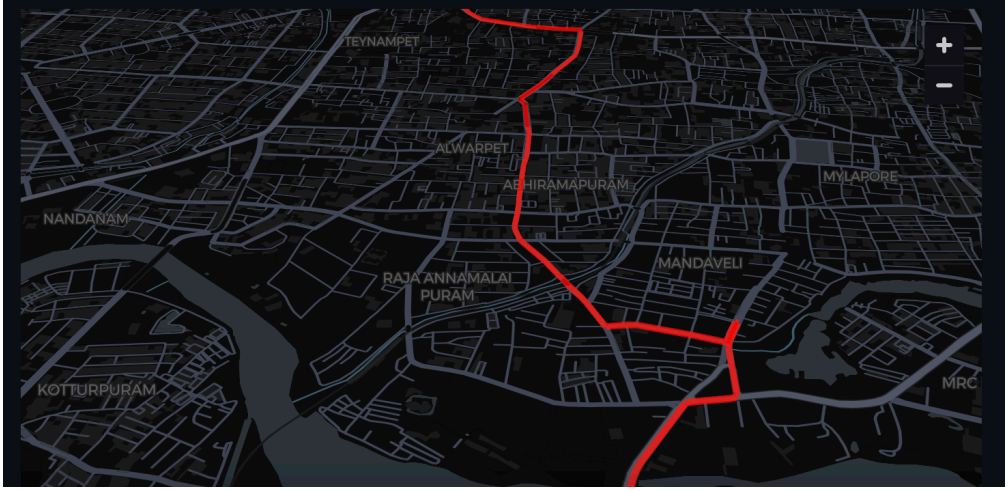
Ride Time: 30.69 minutes (-13.60%)

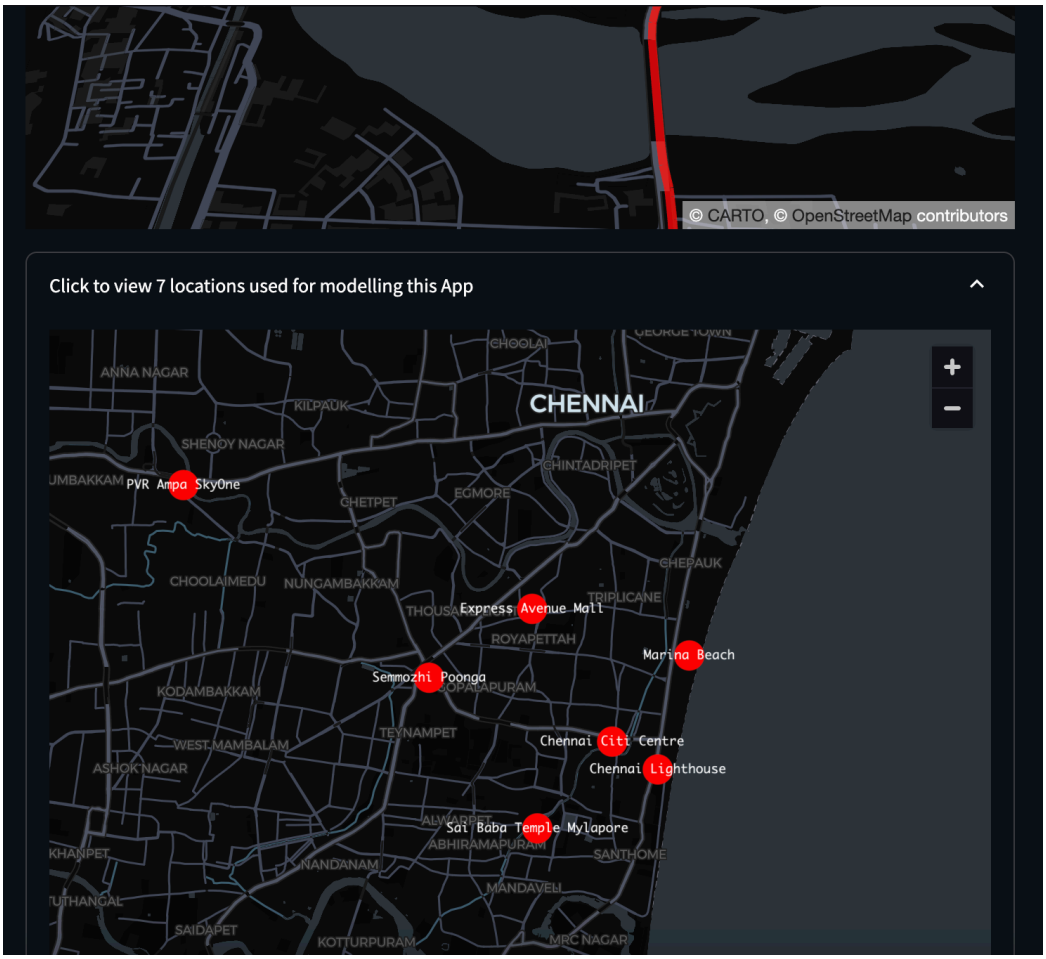
Hour: 16:00

Price: Rs. 562.12 (20.69%)

Waiting Time: 3.52 minutes (16.69%)

Ride Time: 32.18 minutes (-9.40%)





9. Future Enhancements

The project presents multiple avenues for future expansion:

Additional Routes: Extend the model's functionality to other cities and locations.

Real-Time API Integration: Establish direct connections with the Uber API for enhanced data accuracy and responsiveness.

Enhanced Continuous Learning: Incorporate user feedback and behavioural data to refine prediction algorithms further.

Conclusion

This project successfully integrates real-time data extraction and machine learning techniques to optimize Uber ride bookings. The inclusion of a continuous learning framework ensures that the system adapts to evolving patterns in ride demand and pricing. By providing users with predictive insights into ride prices and wait times, it enhances the overall transportation experience. The project highlights the potential of data-driven approaches in urban mobility, paving the way for future innovations.

References

- https://openrouteservice.org/
- https://nominatim.org/
- https://www.uber.com/in/en/
- https://scikit-learn.org/stable/
- https://mlflow.org/docs/latest/index.html
- https://www.selenium.dev/documentation/
- https://dev.mysql.com/
- https://deckgl.readthedocs.io/en/latest/layer.html
- https://docs.streamlit.io/