Fare Enough: Building ML Models for Predicting Uber Fare Amount

Muchen Zhong
Data Science Institute, Brown University
Github link: https://github.com/muchenzhong/DATA1030_FINAL_PROJECT.git

Part I. Introduction

Ride-hailing services like Uber have significantly impacted urban transportation in recent years. (1) With its development, understanding the factors impacting the fare amounts is crucial. For passengers taking the rides, analyzing fare prediction would help them to anticipate the cost of trips, providing them guidance in choosing optimal travel time; for drivers, a better understanding of the pricing strategy of the platform can optimize their trip selection and deliver them with earning forecasts; companies like Uber can adjust their pricing models to balance supply and demand efficiently, ensuring that riders are incentivized to use the service while drivers are motivated to meet demand during peak times.

Previous researchers have started to explore the factors that could potentially impact its price. For instance, one paper concluded that the prices are primarily influenced by distance and traffic surge, not time of day or weather. (2) Another former study believes that Uber nowadays has become a more cost-effective option for business travelers or individuals in areas with limited transit services, indicating Uber prices are influenced by travel time. (3)

Inspired by others' research and the importance of uncovering relationships in ride-hailing fare data, this report will focus on building predictive regression models to predict the Uber fare amount. The data being used was extracted from Kaggle and collected using Uber API. It is a large dataset with 192036 observations, ranging from 2009 to 2015, with no missing values. The dataset includes 6 features, namely pickup_datetime (including year, month, date, and the specific time), pickup_longitude, pickup_latitude, dropoff_longitude, dropoff_latitude, and passenger_count (manually entered by drivers), and a target variable of fare_amount.

Part II. Exploratory Data Analysis

After extracting the data, some EDA has been done to provide a decent insight into this data and an initial understanding of the relationship between each feature and the target variable. Figure 1 shows the log-scaled distribution of the target variable, the fare amount, where the y-axis shows the frequency of observations and the x-axis shows the log-transformed values of fare amounts (in $). The distribution appears to be closer to a bell shape after applying the log transformation. A log-scaled distribution enhances the clarity of patterns by reducing the dominance of extreme values.

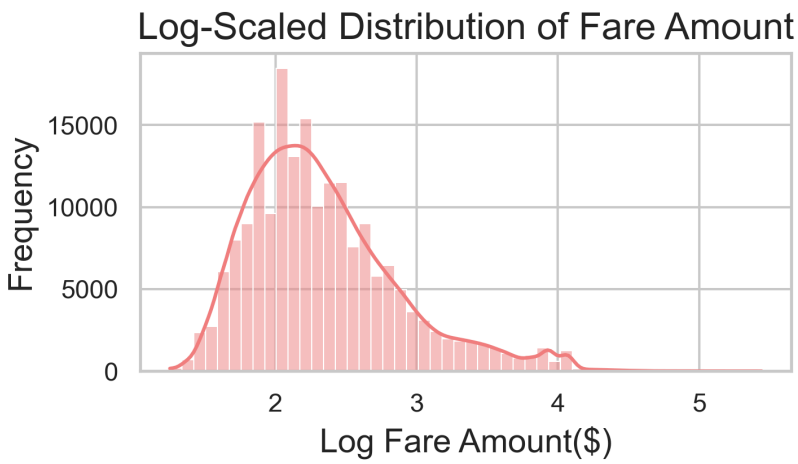

Figure 1: Fare Amount (Log-Scaled Distribution)

Figure 2 shows the relationship between the shortest distances between the drop-off points and the pick-up points calculated using the haversine formula (introduced later) and the fare amount. The scatter plot illustrates how distance (km) impacts the fare amount ($). A dense cluster of points is visible at shorter distances (0–25 km) with fares mostly below $50. Beyond 25 km, trips become sparser, but some high-fare outliers are visible. Overall, longer distances results in higher fares.

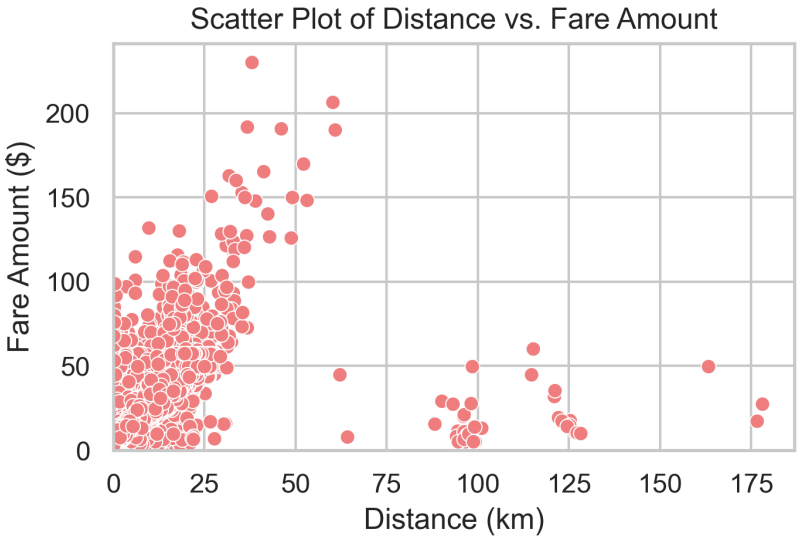

Figure 2: Distance vs. Fare Amount

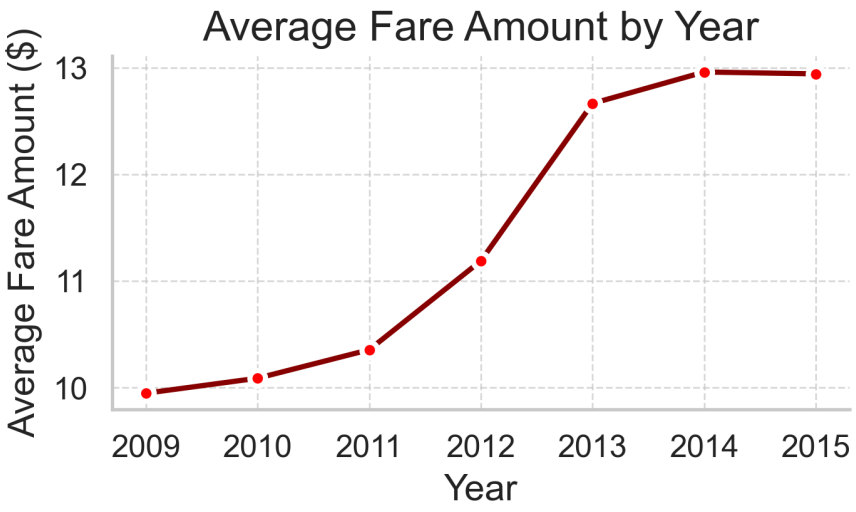

Figure 3: Average Fare Amount (Year)

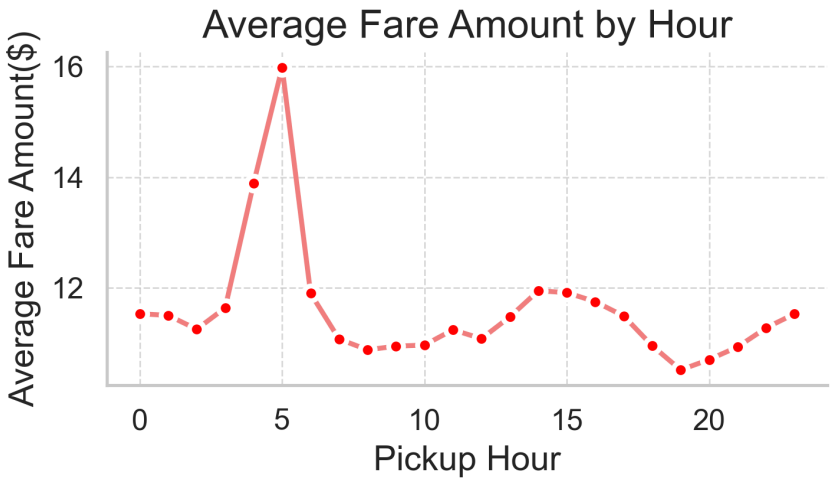

Figure 4: Average Fare Amount (Hour)

Figure 3 shows the average fare amount over the years from 2009 to 2015, while Figure 4 shows the average fare amount across different hours of the day (pickup times). The visuals indicate the average fare amounts steadily increased from 2009 to 2013 and stabilized afterward. In addition, the average fares peak sharply at 5 AM and gradually rise again during late evening hours after 8 pm.

Part III. Method

Feature Engineering

The shortest distance between the pickup and drop-off points is calculated using the Haversine formula, based on the four location features. Unlike the Euclidean formula, the Haversine formula calculates distances on a sphere, making it more suitable for geographic applications.(4) The Earth's radius (R) is 6371 km. The latitude and longitude of both points are converted to radians, and the differences in latitude (Δlat) and longitude (Δlon) are computed.(4) Then, the spherical distance has been computed as a =

$\sin^2(\Delta lat / 2) + \cos(lat1) * \cos(lat2) * \sin^2(\Delta lon / 2)$ and the angular distance has been calculated as $c = 2 * \text{atan2}(\sqrt{a}, \sqrt{1 - a})$. Finally, the distance is represented as $R * c$. (4)

ML Pipeline

To evaluate the performance of different models, a comprehensive pipeline was implemented. The pipeline integrates data splitting, preprocessing, cross-validation, hyperparameter tuning, and performance evaluation. To further enhance the robustness of the pipeline, the data splitting, model training, and testing process was repeated over 5 random states. This ensures that the results are not overly dependent on a single train-test split.

Data Splitting and Cross-Validation

The dataset was first split into an 80 (train + validation) - 20 (test) ratio to ensure that the final evaluation is conducted on unseen data, providing unbiased estimates of the generalization error. The data is randomly shuffled before the split to introduce more randomness and prevent any order-related biases.

A 4-fold cross-validation was applied to the data to split the train and validation set (Figure 5). This strategy iteratively splits the remaining data into four folds, with one fold used as validation and the other three folds serving as training data. With a shuffling, the splitting method ensures a more reliable evaluation of the model's performance.

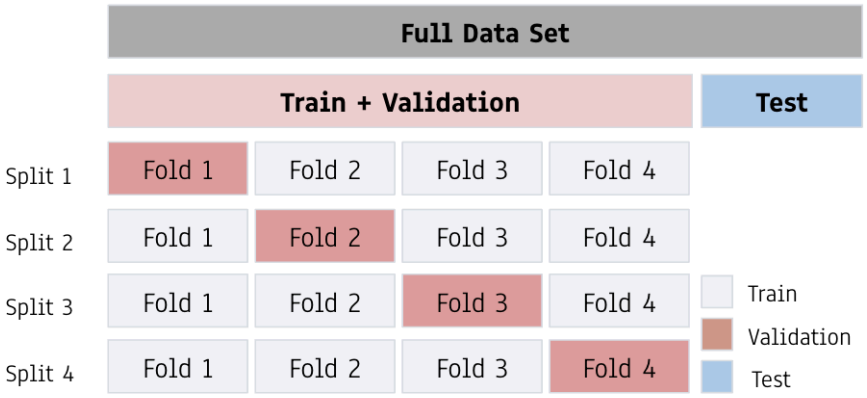

Figure 5. Data Splitting

Data Preprocessing

During cross-validation, data is preprocessed with a pre-defined pipeline. Some ordinal encoding and data normalization including: Ordinal encoding has been done on the pickup_datetime feature as there exists some trend between the date, time, and the Uber fare amount as shown previously in the EDA section and ordinal encoding preserves these patterns compared to one hot encoding. The pickup_datetime column was first split into two features: the pick-up day (year, month, and date) and the pick-up hour. The pick-up hour was ordinally encoded into values from 1 to 24. For the pick-up day, ordinal encoding was applied such that the earliest date in the dataset was assigned the value 1. Subsequent dates were assigned consecutive integers, skipping missing dates. (E.g., if the earliest date is 1, the third chronological date (skipping a missing second date) would be assigned 3). Then, the location features and 'pickup_hour' are scaled using Min-Max scaling for their known and fixed range, while other features are standardized. The normalization process ensures all features are on comparable scales, optimizing model performance.

ML Algorithms and Hyperparameter Tuning

Six ML algorithms have been chosen with the parameters being tuned shown in Table 1.

ML Algorithm	Parameter Tuned
Lasso Regression	alpha [0.01, 0.1, 1.0, 10.0, 100.0]
Ridge Regression	alpha [0.01, 0.1, 1.0, 10.0, 100.0]
ElasticNet Regression	alpha [0.01, 0.1, 1.0, 10.0, 100.0] L1_ratio [0.1, 0.3, 0.5, 0.7, 0.9]
XGBoost	N_estimators [100] Max_depth [1, 3, 10, 20] Reg_alpha [0.0, 0.01, 0.1, 1.0, 10.0, 100.0] Reg_lambda [0.0, 0.01, 0.1, 1.0, 10.0, 100.0] Learning_rate [0.05, 0.1, 0.2] Subsample [0.9] Colsample_bytree [0.66]
KNN Regressor	N_neighbors [1, 5, 10, 100, 1000], Weights ['uniform', 'distance']
Random Forest Regressor	n_estimators [100] max_depth [1, 3, 10, 20] max_features [0.25, 0.5, 0.75, 1.0]

Table 1. Algorithms Overview

Evaluation Metric & Uncertainty Measurements

RMSE is chosen as the evaluation metric because it is in the same unit as the target variable (dollars), making it easy to interpret, and it effectively captures variance and penalizes large errors, ensuring accurate fare predictions for user trust and profitability. To measure uncertainty, the model is evaluated across five random states, with the standard deviation of RMSE calculated from these runs. The results will be visualized in the later section.

Part IV. Results

Model	Best Parameters	Average Test RMSE	Std Dev of Test RMSE
Lasso	{'lasso__alpha': 0.01}	5.4405	0.2463
Ridge	{'ridge__alpha': 0.01}	5.4187	0.2468
ElasticNet	{'elasticnet__alpha': 0.01, 'elasticnet__l1_ratio': 0.9}	5.4401	0.2457
XGBoost	{'xgbregressor__colsample_bytree': 0.66,	3.2373	0.1020

	'xgbregressor__learning_rate': 0.2, 'xgbregressor__max_depth': 20, 'xgbregressor__n_estimators': 100, 'xgbregressor__reg_alpha': 100.0, 'xgbregressor__reg_lambda': 10.0, 'xgbregressor__subsample': 0.9}		
k-NN	{'kneighborsregressor__n_neighbors': 100, 'kneighborsregressor__weights': 'distance'}	3.8675	0.0889
Random Forest	{'randomforestregressor__max_depth': 20, 'randomforestregressor__max_features': 0.5, 'randomforestregressor__n_estimators': 100}	3.2546	0.0698

Table 2. Model Results

Table 2 shows the performance of each model and figure 6 visualizes the results. All models outperform the baseline RMSE of 9.49, calculated from the average Uber fare in the test set (mean of y_test). XGBoost appears to be the most predictive one, with the lowest mean RMSE of 3.2373 and a relatively low standard deviation of 0.1020. Figure 7 shows a scatter plot between the predicted value and the true value using the best combination of the parameters of the XGBoost model.

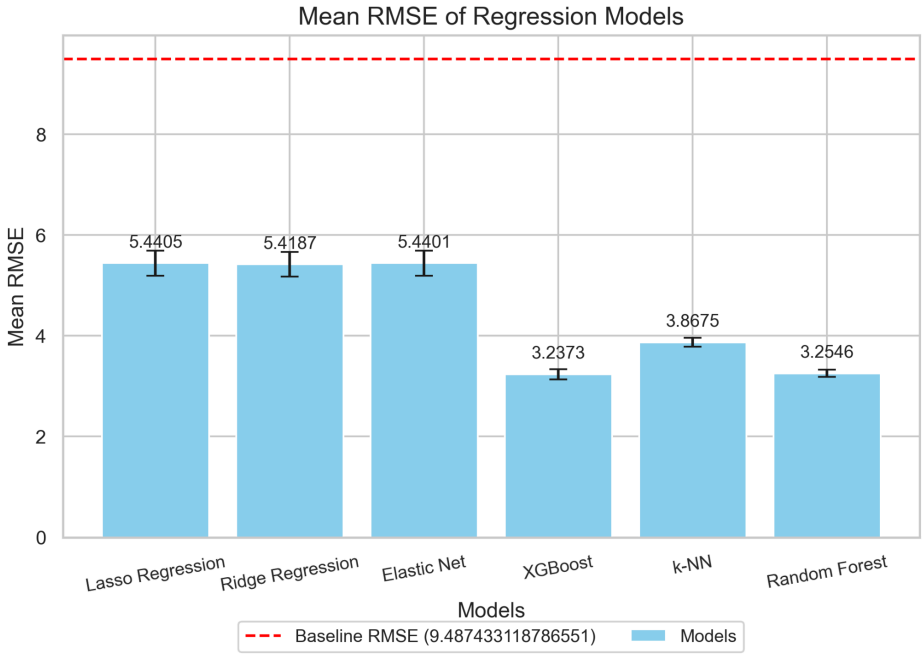

Figure 6. Mean RMSE Results Comparison

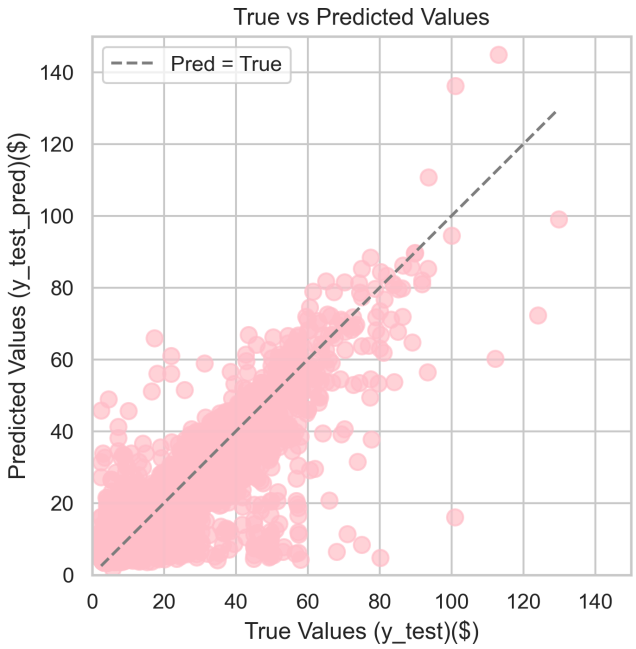

Figure 7. True v. Predicted Fare Amount

Feature Importances

The global importance is first calculated using permutation importances with standard deviation based on the best model shown in Figure 8. Distance_km is the most important feature, followed by dropoff and pickup locations. The least important feature is passenger_count. With the random forest approach (Figure 9), the top features are almost similar to the permutation importance which shows consistency in identifying the most influential features, except the random forest approach has the pickup day being the fifth important feature. The least important feature is also the passenger count. These results are reasonable as fare amounts are primarily influenced by travel distance and location, while passenger count has minimal impact on pricing by common sense.

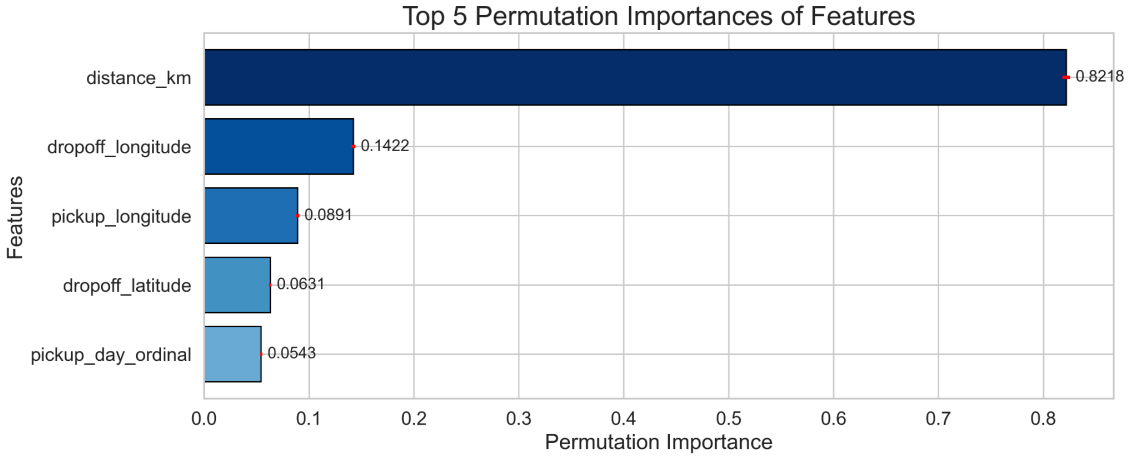

Figure 8. Top 5 Permutation Importance Features

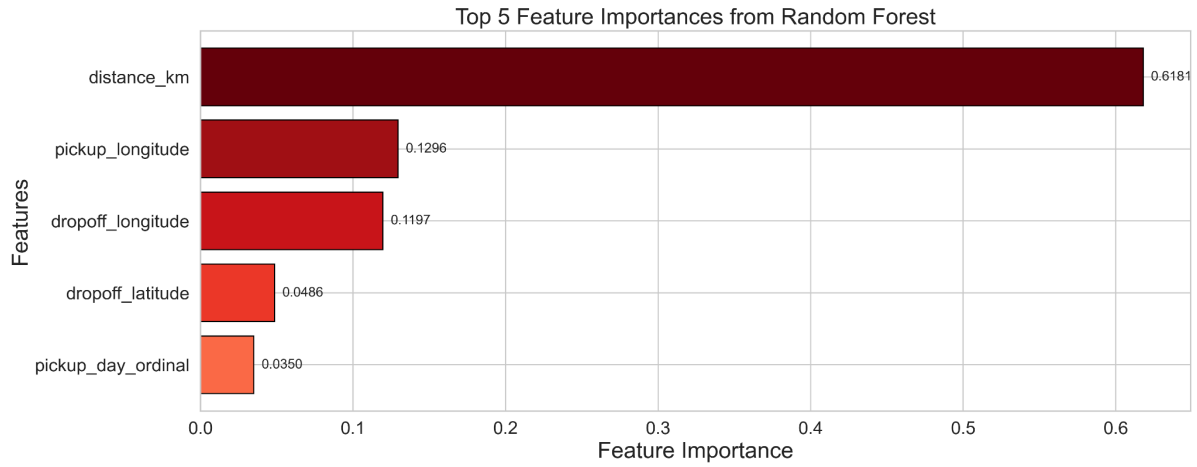

Figure 9. Top 5 Feature Importance (Random Forest)

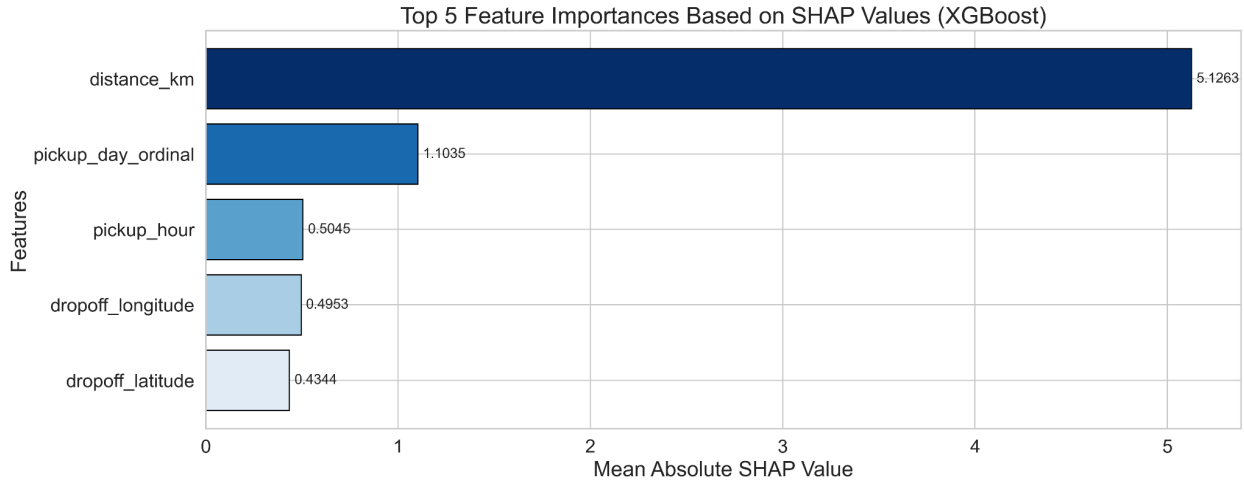

Figure 10. Top 5 Feature Importance (Mean Absolute SHAP Value)

The top importance feature is still distance_km as expected for MAE of SHAP Value (Figure 10). Pickup_day_ordinal and pickup_hour are also an expected important feature as they likely capture the impact of time-based patterns, such as traffic congestion during specific hours. The least important feature using this approach is still passenger_count.

Overall, the most important global feature regardless of the approach being chosen is distance_km, and most of the approach shows that the pickup and drop-off locations also strongly influence the fare amount. All of the methods show that passengers' numbers are the least important factors that contribute to the fare amount. In conclusion, the results are not surprising as they align with intuitive expectations: distance is a primary determinant of fare, and geographic locations influence travel costs, while a specific number of passengers has minimal impact on fare.

Figures 11 and 12 show force plots generated using random indices, illustrating how individual predictions are influenced by feature values. In Figure 11, the prediction is above the base value, with positive impact from distance_km and pickup_latitude; negative impact from dropoff_latitude and

pickup_day_ordinal. In Figure 12, the prediction is below the base value, with positive contributions from pickup_hour and dropoff_latitude, and negative contributions from distance_km and pickup_day_ordinal.

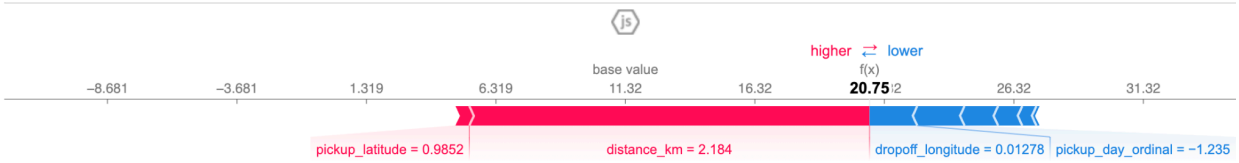

Figure 11. Force Plot (Index 12)

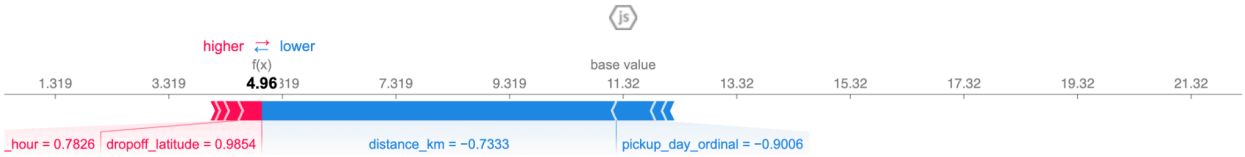

Figure 12. Force Plot (Index 15000)

Part V. Outlook

Some weak spots of this project and potential improvements are as follows. First, using the Haversine formula to calculate distances assumes the shortest path between two points, which does not account for real-world factors like traffic or detours. A potential approach is to calculate the travel distance using travel time estimates from APIs like Google Maps. Second, the lack of engineered interaction terms that can improve model interpretability and prediction accuracy is also one of the potential weak spots. Therefore, engineer interaction terms between features to capture combined effects would also be useful. Thirdly, broadening the hyperparameter grids would also be a potential improvement. Lastly, some external data collection (adding some weather data (e.g., rainfall, temperature) and traffic patterns to the model) would increase the performance, as they likely influence ride fares.

References

1. Sadowsky, N. & Nelson, E., (Year). The Impact of Ride-Hailing Services on Public Transportation Use: A Discontinuity Regression Analysis. Retrieved from https://digitalcommons.bowdoin.edu/econpapers/13/

2. Huang, M., (Year). Ridesharing Price Prediction: Exploring the Factors of Price Fluctuation. Retrieved from https://www.semanticscholar.org/paper/Ridesharing-Price-Prediction%3A-Exploring-the-of-Huang/336ea83699619443a27a5e09e511a66496cb053a

3. Schwieterman, J., (Year). Uber Economics: Evaluating the Monetary and Travel. Retrieved from https://www.semanticscholar.org/paper/Uber-Economics%3A-Evaluating-the-Monetary-and-Travel-Schwieterman/bbb1cdfa363aaa23ffbedc9c5d8f481d8ad42f34

4. ESRI, (2020). Distance on a Sphere: The Haversine Formula. Retrieved from https://community.esri.com/t5/coordinate-reference-systems-blog/distance-on-a-sphere-the-haversine-formula/ba-p/902128

5. YasserH., (2020). Uber Fares Dataset. Kaggle. Retrieved from https://www.kaggle.com/datasets/yasserh/uber-fares-dataset