UC San Diego

Predictive Model for Cab Fare

Final Report for DSC148-WI23

Hede Yang
Jianjun Gao
Licheng Hu
Wenhua Tang
Yuchu Yan

Supervised by: Dr. Jingbo Shang

March 19, 2023

Introduction:

The price of Uber has always fluctuated since the ride based on a range of factors such as time of day, distance, location, and demand. With the increasing popularity of ride-sharing services such as Uber and Lyft, accurate pricing has become crucial for both companies and their users. Our project is aiming to predict the price of Uber and Lyft rides with given features of time, location, weather and distance etc.

1 Dataset

1.1 Identify dataset

Uber and Lyft are two of the most popular and well-known ride-sharing companies in the world. In order to purchase ride service, users have to sign up with their email, number, and card for the payment. Both Uber and Lyft offer a range of services, from economy to luxury, and have become a ubiquitous part of modern transportation.

The dataset project_sampled_data is about Uber and Lyft prices in Boston. The original dataset contains more than 693071 rows, which is massive in size. In order to perform a more efficient model, our group split the origin data into a training dataset and a test data set with a ratio of 0.66. Thus, the new training dataset contains around 464357 rides among Uber and Lyft from Boston, MA within one month. Each row represents one ride and each column represents the details about the ride.

1.2 EDA

Basic Statistics

We use the dataset *Train_Data.csv* as our training data which contains 464357 rows sampled from the original dataset *rideshare_kaggle.csv* containing 693071 rows. For test data, *Test_Data.csv* contains 228714 rows. Our dataset covers the date of rides from 2018-11-26 to 2018-12-18. The columns of dataset include the information about price of the ride, data about time(e.g. hour, day, and month), geographical location (e.g. longitude and latitude, distance etc) and weather(e.g. temperature, visibility, summary of weather condition) etc.

Data Cleaning

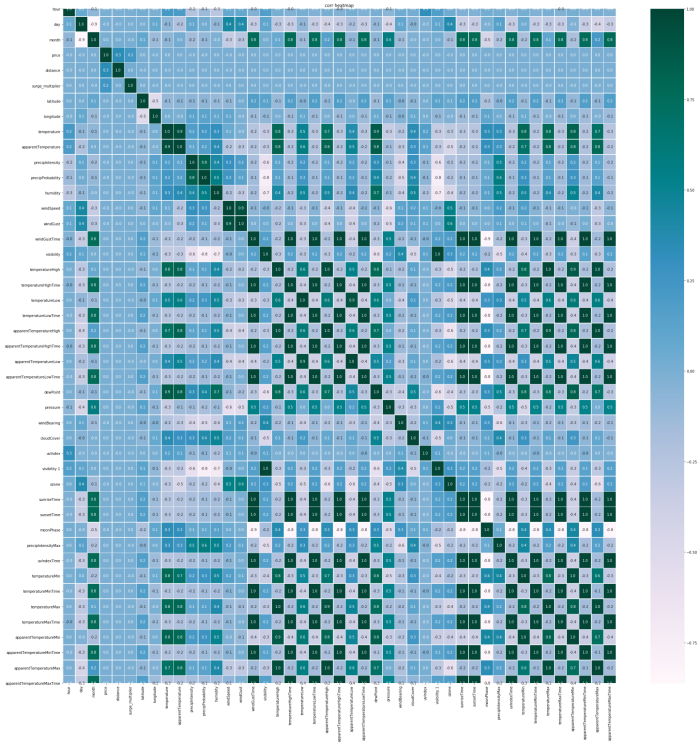

Figure 1: Heatmap of correlation of all features

Column drop: By looking at the unique value of each column in the dataset, it turns out that the *timezone* has only one value which is the "America/New_York". Thus, we decided to drop this column. In addition, since the column id is unique to each row, we decided to drop this column as well. In addition, since the number of unique values of the *long summary* column is greater than the *short summary*, we decided to drop the short summary column and applied TF-IDF to the *long summary*. Besides, from the above heat map, we see that all features related to weather and climate have weak correlation with the price, so we decided to drop these weather and climate related features.

Row drop: By looking at the mean missing value of each column, we found that only the *price* column has 0.079494 (around 8%) of the missing value. Since the total of missing values counts for only a very small proportion of the data, we decided to drop all the missing values from our original dataset.

Column Transformation: In order to improve the accuracy of ride price prediction, our team performed a column transformation on the timestamp column, which was originally in integer format. We recognized the significance of the day of the week in determining ride prices and therefore converted the timestamp column into datetime format. Additionally, we created a new column titled weekdays, which represented the day of the week in string format. This transformation allowed us to take into account the influence of weekdays on ride prices and thus make more precise predictions.

Findings in EDA

Cab_type: there are two types of cabs in this dataset: Uber and Lyft. The following histogram (Figure1) plots the amount of rides based on different types of cab. Uber has a higher frequency of a low price range. The price for Lyft rides is distributed more evenly. Thus the price based on the cab type can be one of the important features.

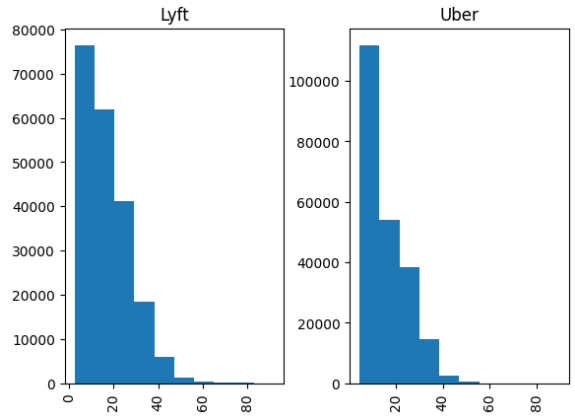

Figure 2: Price frequency within each Cab_type

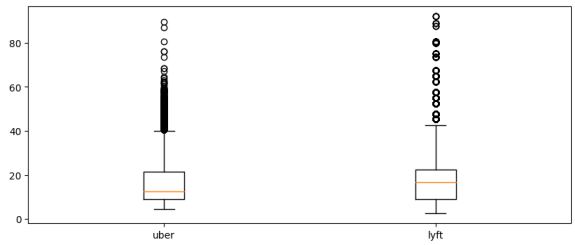

Figure 3: Distribution of price with each Cab_type

Subsequently, the above box plot (figure2) was constructed to visualize the distribution of prices for these two cab types. Based on the graphical representation, it can be

inferred that the median price of Lyft trips is comparatively higher than that of Uber trips. As a result, it was deemed appropriate to retain the cab types column as a feature in order to facilitate a more comprehensive analysis of the test data.

Hour:In order to understand the relationship between hour and price better, the bar chart shows the mean of the price difference between hours. The bar chart (Figure3) shows the gap in height which illustrates the price depends on the hour of the day. Working hours are more likely to have a lower cab price.

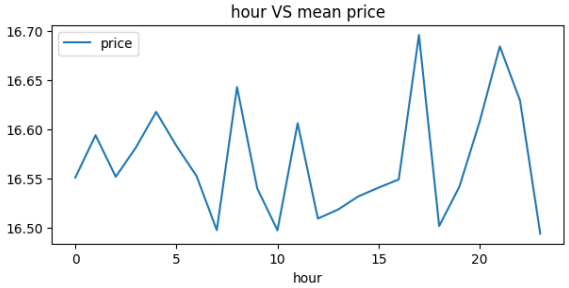

Figure 4: mean price over hours

Surge_multiplier: Dynamic pricing, which involves adjusting prices based on changes in supply and demand, has become a common strategy for ride-sharing companies such as Uber and Lyft. One of the most widely used dynamic pricing features is the "surge multiplier", which is employed during periods of high demand to increase the fare of a ride. This surge multiplier represents the factor by which the normal fare is multiplied to determine the final price.

To gain insight into the surge pricing patterns of these ride-sharing companies, we aggregated data on the hour and the corresponding surge multiplier as shown in Figure 4. Our analysis revealed that there are two surges in the multiplier that happen in both mornings from 9 a.m to 10 a.m and night from 20 p.m to 0 a.m, which follows our common sense that morning is usually the peak hour which involves traffic jam and it also gets harder to hail a cab when the nights get later.

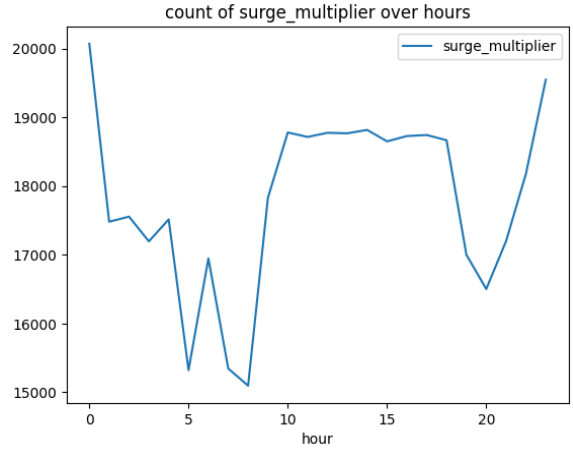

Figure 5: count of surge multiplier over hours

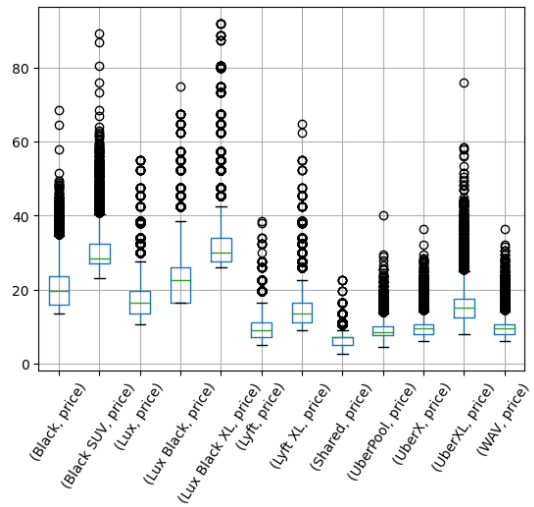

Figure 6: Distribution of price within each name (ride type)

Name: *Name* column in this dataset shows the different ride types (e.g UberXL, Lyft Xl, Shared etc). From the following boxplot (Figure5), it's clear to see that the LUX (product of Lyft) and Black(product of Uber) have higher prices than other ride types. The basic ride type such as UberX and Lyft have much lower prices.

Distance: Based on our analysis of the data on cab trips, we hypothesized that *distance* is a crucial feature for predicting the target variable, namely the price column. We postulated that, as the distance increases, the fare would correspondingly increase. To validate this hypothesis, we created a scatter plot (Figure 6) to explore the correlation between distance and price. While the data exhibited some degree of noise, we discerned a discernible positive trend for both types of cabs as the distance increased. Therefore, based on this observation, we posit that distance is a critical variable to consider when predicting the price of a cab trip. Our findings suggest that the relationship between the distance and price is the farther the distance, the higher the fare is likely to be, which follows our intuition.

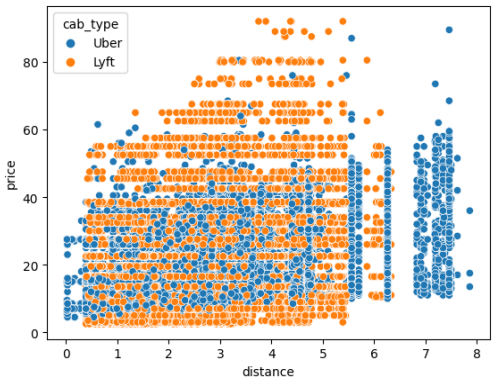

Figure 7: Scatter plot of trips with different distance by different cab type

Destination: Based on the statistical analysis of the data and the boxplot visualization (Figure 7), we can observe that trips with different cab types and destinations have different distribution of price. Notably, trips that have Financial District as its destinations tend to have a higher price than other trips and trips that have Haymarket square as its destinations tend to have a lower price than other trips. Overall, Lyft has a higher price than Uber across all destinations.

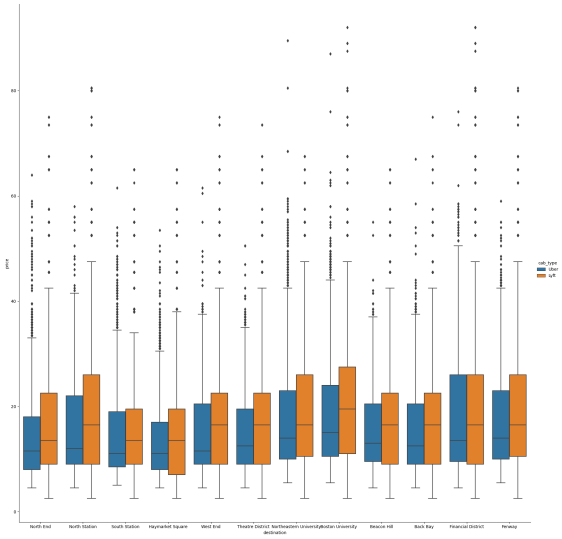

Figure 7: Distribution of price of trips within each destination of trips for both cab types.

2. Predictive Task

2.1 Predictive task

After basic cleaning and engineering, we decide to use features which include *weekday, hour, cab_type, name, distance,, long_summary, and destination* to predict the price of a given trip. This is a regression prediction problem.

2.2 Evaluation

Root Mean Square Error (RMSE) and R-Squared (R^2). Both measurements are important for accessing our model in different respects. **Root Mean Square Error (RMSE)** measures the error of how far away the predicted values are from the given values in the dataset on average of a model. On the other hand, **R-Squared (R^2)** measures how much the variation of a dependent variable can be explained by the independent variable. When the RMSE is lower and at the same time R^2 is closer to 1, the better the model fits the dataset. Therefore, in the following tasks, we will evaluate the model performance based on both RMSE and R^2. The performance of the model will be accessed by its RMSE and R^2 of the model prediction and testing data.

Generalization

During the data cleaning section, we figured out that there are some data that have null prices. Since we are trying to make a model to predict the price of the trip, these data with null prices are useless and even noisy for our prediction. Therefore, we decided to drop these data in our prediction process and evaluation. We have 36941 data that have null prices out of 464357 data in our train dataset. Thus, after we drop the data with null price, we still have enough data to analyze.

2.3 Baseline

Baseline dataset: *Train_Data.csv*

Baseline involves features: *weekday, hour, cab_type, name, distance, long_summary, and destination*

8 models are used as the baseline: *Linear regression model, Random Forest Regressor, XGB Regressor, Lasso regression, Ridge regression, Stochastic Gradient Descent Regressor, AdaBoostRegressor and GradientBoostingRegressor.*

Linear Regressor: Linear regressor is a type of model that finds a linear relationship between features and price through statistical approach, the model will fit a line to the data that best represents this relationship.

```
Linear RMSE: 6.795093990622744
R-squared: 0.4702443849219795
```

Figure 7: Linear Regression Performance

Random Forest Regressor: Random Forest Regressor is an ensemble learning algorithm that uses multiple decision trees to make predictions of price. It creates a set of decision trees on random subsets of the data, and then averages their outputs to make a final prediction.

```
RandomForest RMSE: 2.7546291352656826
R-squared: 0.9129415550877801
```

Figure 8: RandomForest Regression Performance

XGB Regressor: XGB Regressor is a type of gradient boosting algorithm that uses a combination of weak learners to create a strong learner. It works by iteratively adding new decision trees to the model of predicting price and adjusting the weights of the features based on the errors of the previous models.

```
XGB RMSE: 2.470667460488817
R-squared: 0.9299653080176011
```

Figure 9: XGB Regression Performance

Lasso and Ridge regression: Lasso and Ridge regression are two popular regularization techniques used in linear regression models. They work by adding a penalty term to the model that discourages large coefficients for the predictor features. Lasso regression adds an L1 penalty term, which can result in some coefficients being reduced to zero, while Ridge regression adds an L2 penalty term, which tends to shrink all coefficients towards zero.

```
Lasso RMSE: 6.800371402227667
R-squared: 0.46942119569453


Ridege RMSE: 6.795093993767946
R-squared: 0.47024438443157046
```

Figure 10: Lasso & Ridge Regression Performance

Stochastic Gradient Descent Regressor: SGDRegressor is a type of gradient descent algorithm that updates the model parameters based on a randomly selected subset of the training data at each iteration. In this way, it can lead to faster convergence compared to batch gradient descent, which updates the model parameters based on the entire training set at each iteration.

```
SGDRegressor RMSE: 7.46692217783275
R-squared: 0.3603124739189413
```

Figure 11: SGDRegressor Performance

AdaBoostRegressor: AdaBoostRegressor a weak regressor is trained on a subset of the data, and then the algorithm adjusts the weights of the data points based on the error of the previous iteration. This process is repeated with new subsets of the data until a predefined number of weak learners are created. The final prediction is then made by combining the predictions of all the weak learners, weighted by their performance during training.

```
AdaBoostRegressor RMSE: 5.405862455450146
R-squared: 0.6647146212342192
```

Figure 12: AdaBoostRegressor Performance

GradientBoostingRegressor: Gradient Boosting Regressor is a type of gradient boosting algorithm that is similar to XGB Regressor. However, it uses a different loss function, which is often more suitable for regression problems. The algorithm works by iteratively adding new decision trees to the model, and adjusting the weights of the observations based on the errors of the previous iterations.

```
GradientBoosting RMSE: 2.636234099406946
R-squared: 0.920264342720314
```

Figure 13: GradientBossting Performance

Baseline Result

According to the results shown in each baseline model, it turns out that XGBRegressor (RMSE = 2.47) and GBRegressor (RMSE = 2.63) have the best performance compared to the basic model such as Linear Regression (RMSE = 6.79). Thus, we decided to use XGBRegressor and GBRegressor as our final models.

3. Model

Initial Setting:

Dataset: **Uber and Lyft Dataset Boston, MA**

Initial features: We will use the same features before plus two new features, *icon* and *surge_multiplier* with XGB and Gradient Boosting Regressor since they are the top regressors that achieved higher R square and lower RMSE in our baseline model.

Optimization of the model is based on a decrease in RMSE and an increase in R square. Since most of the trips end up laid in a range of 0 to 40 dollars and there are a lot of trips that have the same price. Therefore, all operations that decrease RMSE and increase R^2 over a threshold (**difference RMSE=-0.5 and difference R^2 =0.02**) will be considered as a significant improvement in performance.

3.1 Basic Hyper parameter tuning

Our group chose two regression models that perform the best from the baseline, which are GBRegressor and XGBRegressor. We manually test out the hyperparameters that had the most effects of these regressors manually: learning rate, n_estimators, max_depth. With the setting (n_estimators=110, learning_rate=0.1, max_depth=8) of Gradient Boosting Regressor, the RMSE decreased 0.977, the R squared increased 0.0480. With setting (learning_rate=0.08, n_estimators=3000, verbosity=1, max_depth=7) of XGB Regressor, the RMSE decreased ; the R squared increased 0.0400.

3.2 Feature Engineering:

Quantitative Data:

Since we are random sampling from the original dataset, we automatically sampled the rows that do not contain null values.

Standard Scaler:

We used standard scaler on our numeric features. Standard Scaler is to process data according to the columns of the numeric features, and convert the feature values of the samples to the same dimension by calculating the z-score method. This process would standardize our numeric values which perform a better prediction on the price.

One Hot Encoder:

The categorical features used are 'weekday', 'cab_type', 'destination', ‘icon’, and 'hour', and each ride can only belong to one category from these features. By applying one hot encoding to these features, we can

represent each category as a binary vector, where only one element in the vector is 1, and the others are 0s. This representation allows the final predict model to handle categorical features in a more efficient way.

Ordinal Encoder:

We applied the ordinal encoder on the [‘Name’] column.The name feature contains the different type cars: Black, Black SUV, Lux, Lux Black, Lux Black XL, Lyft, Lyft XL, Shared, UberPool, UberX, UberXL, WAV. After looking up the different car types on Uber[1] and Lyft[2] Apps and websites, we decided to categorize the types into 7 different orders based on their different price ranking. The luxury type of car apparently cost more than the regular type. Thus we manually replaced the original name of the car into a number. The higher the number the more luxury car type is.

```
ordinal_enc = {'Shared': 1, 'UberPool': 1, 'Lyft':2, 'UberX': 3, 'WAV':3,
               "UberXL": 4, 'Lyft XL':4, 'Lux':5, 'Lux Black': 6, 'Black': 6,
               'Lux Black XL':7, 'Black SUV': 7}
```

Figure14: Ordinal Encoding Transformation

TF-IDF:

In order to analyze the textual content of our dataset, we employed the term frequency-inverse document frequency (TF-IDF) technique to process the "long_summary" column. TF-IDF is a commonly used method in natural language processing that evaluates the importance of each word in a document by taking into account its frequency in the document as well as its frequency across all documents in the corpus.

After applying TF-IDF, we obtained a total of 21 columns, each corresponding to a different word in the text. These columns represent the relative importance of each word in the "long_summary" column and are used to characterize the content of each document in the dataset.

To further analyze the data, we utilized singular value decomposition (SVD) to reduce the dimensionality of the TF-IDF matrix. SVD is a well-established method for reducing the dimensionality of high-dimensional datasets, such as those commonly encountered in natural language processing. By transforming the original TF-IDF matrix into a lower-dimensional representation, SVD allows us to identify patterns and relationships within the dataset that may be difficult to discern otherwise.

Overall, the combination of TF-IDF and SVD allowed us to effectively process the textual content of our dataset and identify key patterns and relationships within the data.

3.3 Model Optimization:

Following the feature selection and feature engineering steps, we trained and evaluated a variety of machine learning models on our dataset. In order to assess the effectiveness of each model, we compared their performance against their corresponding baseline models.

After conducting this evaluation, we identified two models that performed exceptionally well and outperformed their corresponding baseline models. The first

[1] https://www.uber.com
[2] https://www.lyft.com

model was an Extreme Gradient Boosting (XGBoost) model, which is a gradient boosting algorithm that utilizes a tree-based ensemble learning approach. The second model was a Gradient Boosting Regressor, which also uses a tree-based ensemble learning approach to make predictions.

To determine which of these models to use for our final analysis, we further evaluated their performance using additional metrics such as mean squared error and root mean squared error. After this evaluation, we selected both the XGBoost model and Gradient Boosting Regressor as our final models due to their consistently high performance.

After conducting feature selection and feature engineering, we performed hyper-parameter tuning on a variety of machine learning models. Our ultimate goal was to identify the best combination of hyperparameters for each model to achieve optimal predictive performance on our dataset.

After conducting a thorough search of the hyperparameter space, we selected two models with the following hyperparameters: Gradient Boosting Regressor (n_estimators =110, learning_rate=0.1, max_depth=8) and XGB Regressor(learning_rate=0.08, n_estimators=3000,verbosity=1,max_depth=7). These models were selected based on their consistently high performance during our evaluation process.

To optimize our model, we utilized one hot encoding for the categorical features (weekday, cab_taype, destination, short_summary, hour, and icon), TF-IDF for the text feature (long_summary) with SVD for dimension reduction, and ordinal encoding for the name column by assigning different weights to different car types based on research from Uber and Lyft apps and websites. For the numerical features (surge_multiplier and distance), we experimented with standardization, but ultimately decided to pass them through without standardization as it did not improve the RMSE.

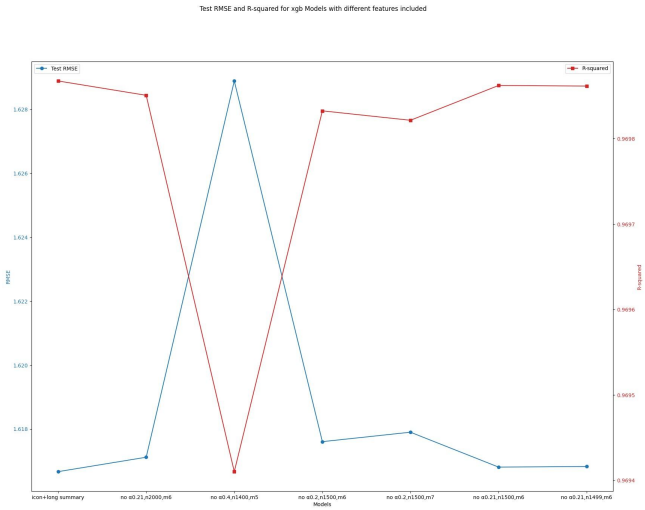

Figure 15: Test RMSE and R-squared for XGB Models with different features included

For the baseline models, we only used ordinal encoding for all categorical features and did not include the surge_multiplier column. However, after hyperparameter tuning and utilizing various feature engineering techniques, we were able to significantly improve the performance of our models.

Our final XGB model achieved a reduction in RMSE by 0.86 compared to the baseline XGB model and an increase in R^2 by 0.04. Similarly, the GBR model achieved a

decrease in RMSE by 0.98 compared to the baseline model and an increase in $R^2$by 0.048. These improvements were a result of the careful selection of hyperparameters and feature engineering techniques employed during the model optimization process.

3.4 Model Optimization Result:

After comparing the performance of the XGB model and Gradient Boosting Regressor (GBR) model, we ultimately chose the eXtreme Gradient Boosting model over the GBR model. The XGB model demonstrated a lower RMSE and a higher R^2 score compared to the GBR model. This decision was made based on our primary objective to minimize the prediction error on our dataset while maximizing the predictive power of the model.

The final XGB model achieved a significant improvement in performance compared to the baseline XGB model. The RMSE value decreased from the baseline value by 0.86, and the R^2 score increased by 0.04. After further optimization, the final XGB model achieved a remarkable RMSE value of 1.61 and an impressive R^2 score of 0.97, indicating its high accuracy and strong predictive power.

Overall, our model optimization process involved carefully selecting and tuning hyperparameters, implementing various feature engineering techniques, and comparing the performance of different models. The final selection of the XGB model was based on its superior performance and ability to achieve our primary objective of accurate and reliable predictions on our dataset.

4. Literature

Uber-Lyft Price Prediction by Daniel Beltsazar Marpaung[3]:

Daniel used the same dataset and also studied the price prediction task as we did. In his EDA section, he firstly analyzed the time features such as month, hour, and day, which is very intuitive since these features come first in the columns of the dataset.Then, he analyzed the location features such as source and destination. The highlight of his project is that he visualized his data by using a geospatial map, which enables the readers to clearly see the amount of rides within each location. In the data processing section, Daniel used the similar approach as we did to remove the unrelated feature by plotting the heat map of each feature to the price. Through the heatmap, both Daniel and our team reached the same conclusion that most weather features are unrelated to the price since all of them show very weak correlation. Thus, both Daniel and our team removed all the climate and weather related features. In the modeling section, compared to his regression model, we used more different types of regression models than he did. Another highlight of Daniel's project is that he removed the outliers which could be the potential noise in data that undermine the accuracy of model predictions.

[3] https://www.kaggle.com/code/danielbeltsazar/uber-lyft-price-prediction

Despite the different process of feature engineering, it is surprising that all of our linear models don't have good performance (R^2 = 0.47) in any linear models such as Linear Regression, Lasso Regression, and Ridge Regression etc. However, since we tried different other regression models such as XGBoostRegressor (R^2 = 0.97) which is even higher than Daniel's final model of DecisionTree(R^2 = 0.9644). Thus, the novelty of our project is that we have a more variety of models, and the model selections and feature engineering have better performance on price prediction.

5. Result:

5.1 Comparison:

Both XGB Regressor and Gradient Boosting Regressor were performed in our selection of baseline models for our dataset. We first split the dataset into train data (70% of the entire dataset) and test data (30% of the entire dataset). Then, we did all feature engineering on both train data and test data, model optimization on the modified train data, and testing modified test data.

After we tried adding and deleting features and making modifications on our feature engineering on the final model, we have improved our RMSE by around 34.79% and R square by 4.32% for XGBRegressor. At the same time, we improved our RMSE by around 37.09% and R square by 5.21% for Gradient Boosting Regressor. In our baseline, we used weekday, hour, cab_type, name, distance, long_summary, and destination and ordinal encoded for categorical features. After several tries, we find surge_multiplier and icons are also useful for our prediction. We also made TF-IDF on long_summary and hyperpara- meter tuning to prevent overfitting. There- fore, our final models outperformed the ba- seline model. The performance of improve- ment in our final model is significant, since we improved more than 30% for RMSE and 4% for R square on both models.

5.2 Effectiveness

Our group planned to standardize all the numerica features by standard scaler. However, the Standard Scaler didn't improve the final score. Instead, passing through numeric features can slightly increase the accuracy and RMSE. The combination of passing through (distance, surge multiplier), One Hot Encoder('TF-IDF','weekday','cab_type','destination',' hour', Ordinal Encoder('name') performed the best, which is what we used on the final model.

5.3 Hyperparameter:

After conducting a thorough search of the hyperparameter space, we selected two models with the following hyperparameters: Gradient Boosting Regressor (n_estimators =110, learning_rate=0.1, max_depth=8) and XGB Regressor(learning_rate=0.08, n_esti- mators=3000,verbosity=1,max_depth=7).

After conducting an extensive search for the optimal hyperparameter combinations for the XGBoost model, we determined that the most influential hyperparameters on the

model's accuracy are n_estimators, max_depth, and learning_rate.

For the n_estimators parameter, we observed that increasing its value led to a lower RMSE, which indicates an improvement in the model's performance. Nevertheless, due to computational limitations imposed by the available CPU capacity, we were constrained to select a maximum value of 3,000 for n_estimators.

Regarding the max_depth parameter, our analysis revealed that the optimal value is 7.

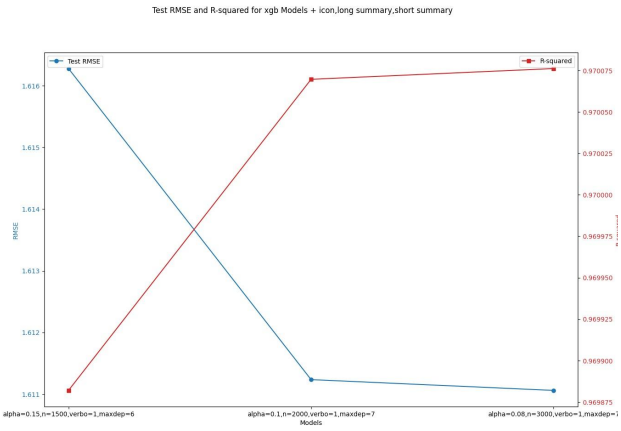

Figure 16: Test RMSE and R-squared for XGB Models + icon,long summary,short summary

Any values above or below this threshold resulted in a weaker model performance, suggesting that a max_depth of 7 represents a critical point in our model's configuration.

In terms of the learning_rate parameter, we initially experimented with a value of 0.05. As we increased the learning_rate, we found that the RMSE decreased, signaling enhanced model performance. Upon reaching a learning_rate of 0.2, however, we identified a turning point beyond which further increases in the learning_rate no longer led to improved model outcomes. Consequently, we concluded that a learning_rate of 0.2 represents the most effective value for our XGBoost model.

5.4 Major Takeaways:

1. Distance, surge multiplier and name of the car are the common features that can help predict the final price of the ride the most.
2. Choosing the right machine learning model is essential. In this case, the XGBoost Regressor and Gradient Boosting Regressor were used, which are powerful and fast for our huge dataset. However, other models like linear regression, random forest, or neural networks could also be used,but took too much CPU space and were less accurate.
3. It is hard to select the right feature transformations and combinations to improve model performance. One-hot encoding for categorical variables, ordinal encoding for ordinal variables, and TF-IDF for text data.

REFERENCES

[1] https://www.uber.com
[2] https://www.lyft.com
[3] https://www.kaggle.com/code/danielbeltsazar/uber-lyft-price-prediction