What's behind the price of ride sharing?

Zhifei Jin, Shelley Li, Yunan Wu

December 13, 2020

Introduction

Ridesharing has been one of the major commute options. To price-sensitive users, the predicted price of each ride might influence their choice of ride-sharing providers. However, the algorithm for pricing calculation is a black box for consumers. We then formulate our question: What are the potential factors that influence the price? What company offers cheaper rides?

In this project, we look at what factors influence the price of ridesharing. Specifically, with the data on Uber and Lyft rides, we train a model that takes factors including the platform, time, day of the week, weather, distance, the model of the ride, etc, to predict the price of the ride.

The data

In this project, we use data from kaggle.com. This dataset contains a sample size of 693,071 ridesharing events, with 56% Uber rides and 44% Lyft rides. All ridesharing events occurred during November to December 2018 in Boston, MA. There are 57 features in total, including information on time, location, car-model, and the weather. We used the following 8 features to construct our models.

Column name	Data type	Description
cab_type	categorical string	The platform of the rideshare. “Uber” or “Lyft”.
type	categorical string	The model of the rideshare. E.g.: “Shared”, “XL”.
distance	float	The distance of the ride.
temperature	float	The temperature when the ride event happened.
humidity	float	The humidity when the ride event happened.
visibility	float	The visibility when the ride event happened.
wind_speed	float	The wind speed when the ride event happened.
day_of_week	categorical string	The day of the week.

Data processing

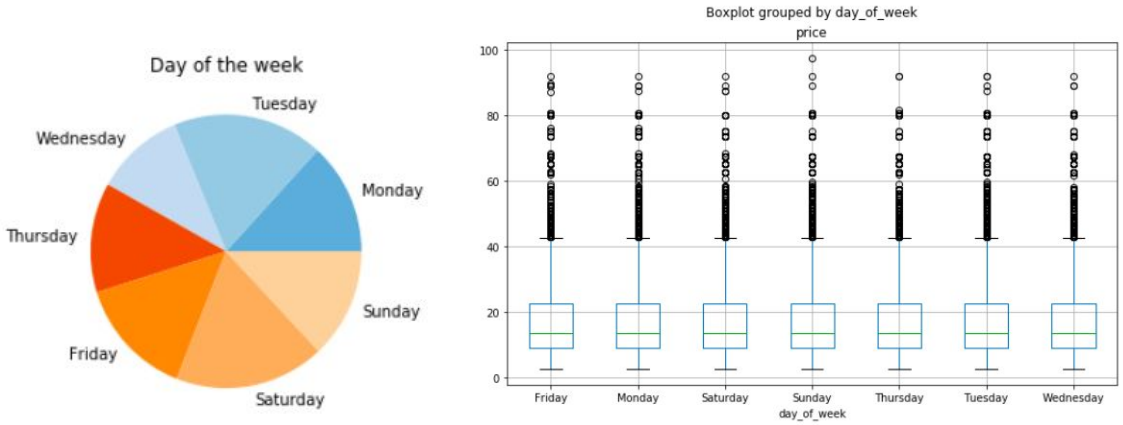
Missing Data

We removed all missing values in the table and changed the datatype of the continuous columns to float. There are in total 55095 entries containing missing data, which comprises 7.9% of all data and all of them are in the price column. Because the price is what we try to predict, we think there is no need to impute the missing prices.

Feature Engineering

We standardized all real features and used one-hot encoding for boolean and categorical features.

Columns `month` and `day` describe the date of the event. However, the particular date provides little information relevant to the price of a ride. Instead, we are able to infer the day of the week from the provided month and day, which we believe would be relevant to the price. Therefore, we combined the columns `month` and `day` into one column `day_of_week`.



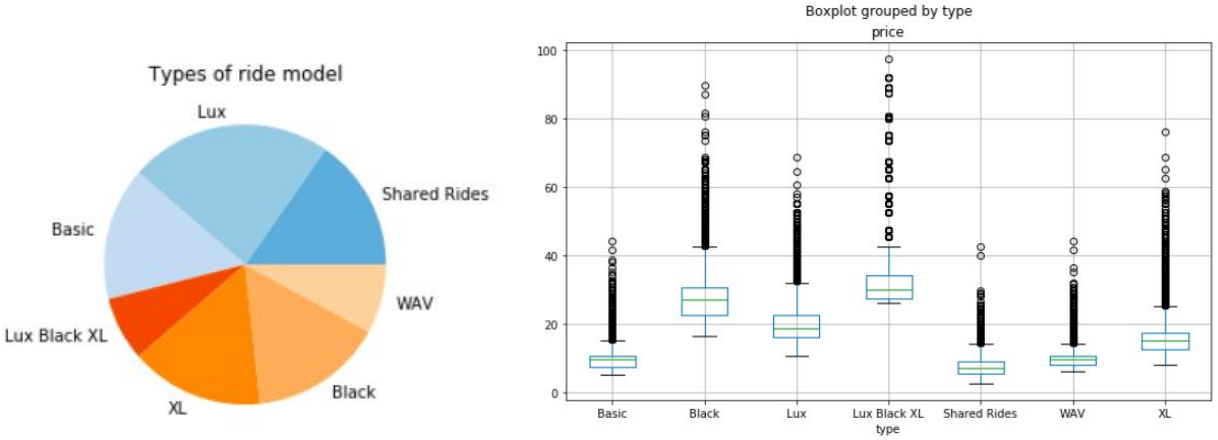
On the left is a pie chart of days of the week. Saturday and Tuesday have the most ridesharing events. On the right is a boxplot of prices grouped by the day of the week. There is no variance of price based on the day of the week.

The `type` column of the dataset describes the model of the ride. We found that the values in the feature `type` were repetitive and depended on the platform. For example, values `Shared` and `UberPool` both mean a shared ride, but the value `Shared` value only shows up in Lyft rides, and the value `UberPool` only shows up in Uber rides. To clean up the `type` column and avoid dependence between features, we reorganized this column by grouping values that mean the same thing.

Original types	New types	Original types	New types	Original types	New types
Shared	Shared Rides	LyftXL	XL	Lux Black	Black
UberPool		UberXL		Black SUV	

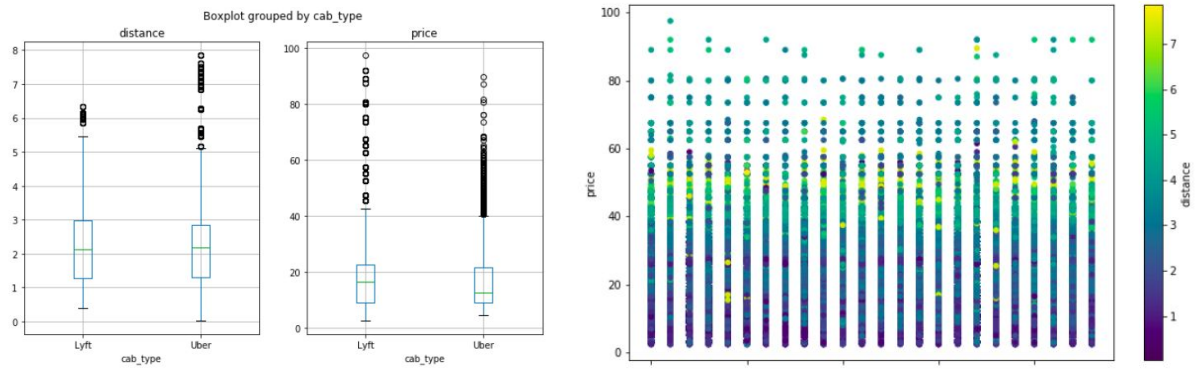
Lyft	Basic	Lux Black XL	Lux Black XL	Lux	Lux
UberX		WAV	WAV		

After regrouping the values, we show the distribution of ridesharing models:



On the left is a pie chart of different types of ride models. The most popular models are "Lux", "Shared Rides", "Basic", and "Black". On the right is a boxplot of prices grouped by type of ride models. Type "Lux Black XL" has the highest average price, and type "Shared Rides" has the lowest average price.

Exploratory Data Analysis

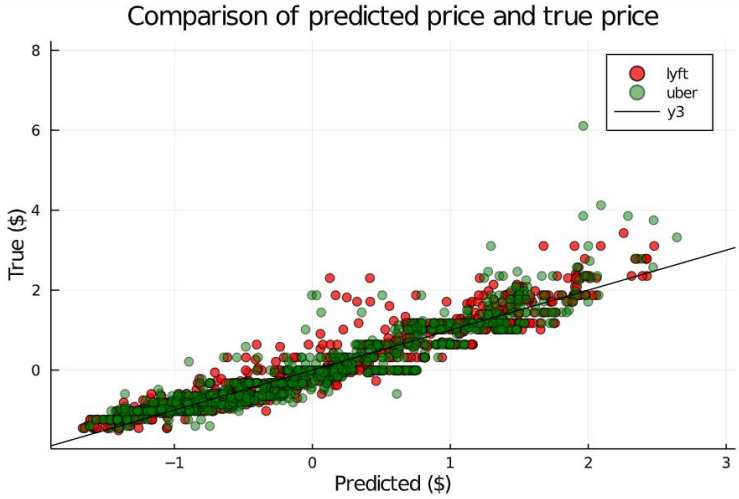


To explore the features available, we did an exploratory data visualization of the features. In the left graph above, we drew two boxplots of two features, `distance` and `price`, grouped by the rideshare platform. We could see that although the ride distance is similar for the two platforms, Lyft rides have a higher median than Uber rides. However, this is the median of the dataset we have, so it could be that in this dataset, more higher fare Lyft rides were collected. In the right graph above, we plotted the `price` for different `hours` of the day, colored by `distance`. From the graph, we are able to see that the price and distance do not change with the hours of the day, but a short distance is correlated with cheaper prices, but the highest prices are not correlated with the longest distance.

Regression Models

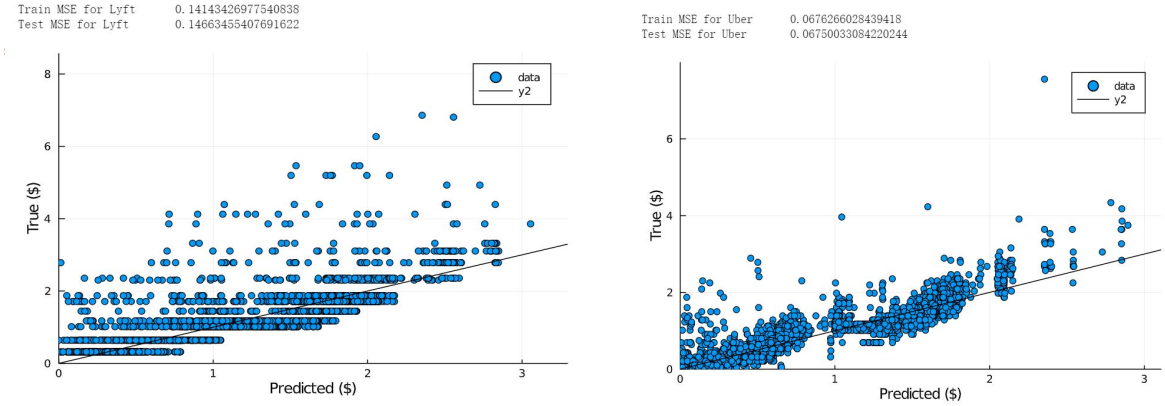
Least Squares

We first trained a linear regression model with least squares. We split the dataset into training and testing data, and the ratio of training to testing data is 0.8:0.2. We evaluated the model by calculating the mean squared error (MSE) for both training and testing data, and achieved a training MSE of 0.1221 and a testing MSE of 0.1219.



Then we separated the dataset into Uber rides and Lyft rides to see how well our model performs on both groups.

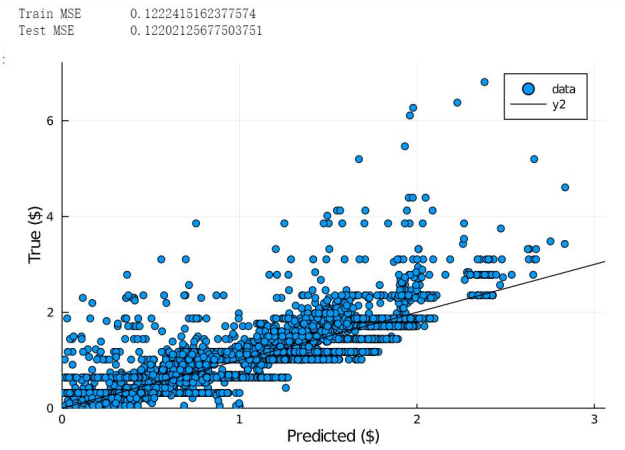
For Uber rides, we achieved a training MSE of 0.0676 and testing MSE of 0.0675 as shown below on the right. For Lyft rides, we achieved a training MSE of 0.141 and a testing MSE of 0.146 as shown below on the left.



In general, the model performs better in Uber rides. One of the possible reasons is that compared to Lyft, Uber has 12% more data in this dataset so it is reasonable that the model fits Uber rides better. Another possible reason is that the prices for Uber have a smaller variance (73.28) than Lyft (100.38). Because the error is the sum of noise, squared bias, and variance, and noise and squared bias are the same for the Uber and Lyft model, the difference between variance of Uber and Lyft data contributes to the difference between the accuracy for both models.

SVD

We also applied Singular Value Decomposition to the dataset and then fit a least-squares regression. We achieved a training MSE of 0.1222 and a testing MSE of 0.1220, which is similar to the errors we get for the previous model.



feature	coefficient	feature	coefficient	feature	coefficient
distance	0.3453	type	0.1806 (“Lux”)	day_of_week	9.704e-3 (“Monday”)
temperature	1.483e-4		-0.2070 (“XL”)		8.573e-3 (“Tuesday”)
wind_speed	6.701e-4		-0.8303 (“Basic”)		8.377e-3 (“Wednesday”)
humidity	-1.213e-3		-0.9580 (“WAV”)		6.403e-3 (“Thursday”)
visibility	-4.137e-4		1.005 (“Black”)		6.546e-3 (“Friday”)
cab_type	-0.1124 “Lyft”		-1.072 (“Shared Rides”)		9.266e-3 (“Saturday”)
	0.1676 “Uber”		1.742 (“Lux Black XL”)		6.262e-3 (“Sunday”)

Proxgrad

We tried 3 different Loss functions and 5 different Regularizers on the proximal gradient method. Because we have a real-valued response variable, we choose to try all the losses that work with real-valued y, namely the quadratic loss, L1 loss, Huber loss, and quantile loss. However, we are not looking at any quantile so we decided to only try the first three. As for the regularizer, we decided to try QuadReg and OneReg because we projected our **w** to be probably small and sparse because some features

may not really matter like humidity and temperature which turns out to be true. As we take a look at the w we got, it is pretty small and sparse and since the MSE is smaller as lambda decreases we figured maybe there is no need for a regularizer so we also tried ZeroReg.

We summarized the result of each combination in the chart below. For each field, we document the training MSE and the testing MSE in the order of train MSE / test MSE. The best result for each loss function is **bolded**.

	QuadReg ($\lambda = 0.01$)	QuadReg ($\lambda = 0.1$)	QuadReg ($\lambda = 1$)	OneReg ($\lambda = 0.01$)	ZeroReg
QuadLoss	0.128 / 0.129	0.287 / 0.288	0.745 / 0.745	**0.124 / 0.126**	0.126 / 0.127
L1Loss	0.136 / 0.137	0.400 / 0.400	0.857 / 0.856	0.134 / 0.135	**0.130 / 0.132**
HuberLoss	0.146 / 0.148	0.455 / 0.456	0.878 / 0.877	0.134 / 0.136	**0.122 / 0.124**

For Quad Loss, the smallest MSE occurs when we use OneReg with $\lambda = 0.01$. For L1 Loss and Huber Loss, the smallest MSE both occur when we use ZeroReg. However, even though the overall smallest MSE occurs when we use HuberLoss and ZeroReg, we can see from the above table that there is no significant difference among the three smallest MSE from each loss.

Conclusion

All three models applied perform relatively well on this dataset so we are confident with our results. Also, as we take a look at the covariates before each feature from each analysis, we found out that `day_of_week`, `humidity`, `temperature`, `wind_speed`, and `visibility` do not have a profound impact on the price, as these features have covariates much smaller than others'. In addition, when we perform analysis on the whole dataset, we observed that the covariate value `Lyft` of type `cab_type` is negative, and the covariate of the other value, `Uber`, is positive. We believe it indicates that when holding all other features constant, requesting a Lyft will be cheaper than requesting an Uber. We noticed that this seemingly contradicts with the higher median price we observed of Lyft. We believe that it is due to the sampling of the dataset and higher fared Lyft rides are collected. We also observed that distance affects the price the most. The `type` of the ride, such as whether the ride is `Shared`, `XL`, or `Basic`, is also correlated with the price. For example, the covariate for level `Shared` is around -1 while for level `Lux Black XL` the covariate is about 1 .

Since this model only produces a prediction of prices and companies have their own algorithms to calculate prices, it will not change the way Uber or Lyft's future pricing of their products, our model will not produce a Weapon of Math Destruction. Unfortunately, rideshare prices are highly related to a city's transportation system, so we need to retrain them if we want to apply the models to data collected from somewhere else. Personally, we would love to use this model to pick which one to use if we were ever in Boston.

Fairness is not an important criterion to consider in our project when choosing models since all features we used are objective and not about individuals but focus on the external environment and quality of the ride itself, so all models are equally fair inherently.