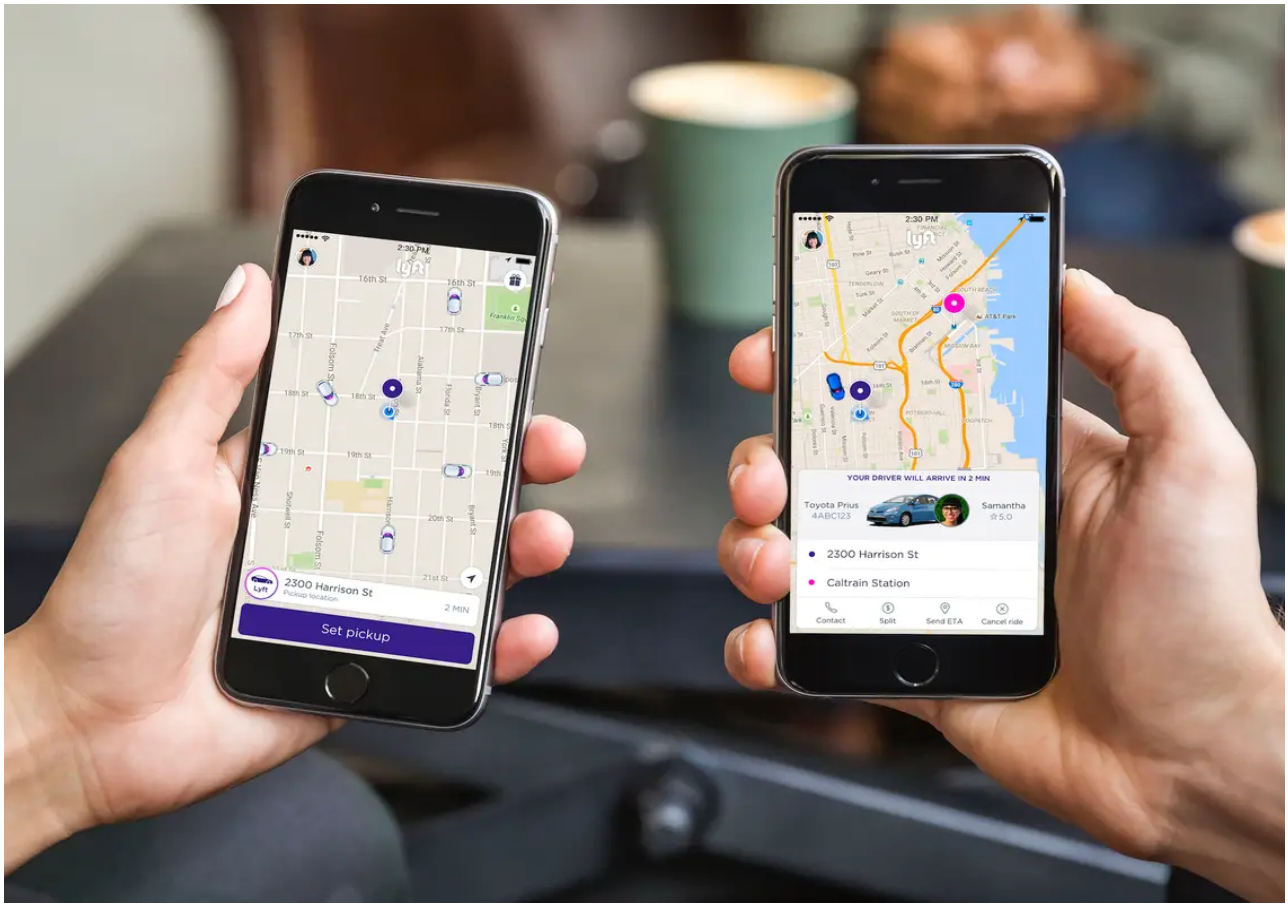
BA305 Lyft Pricing Predictions

Riya Parikh, Roshni Tandon, Dhruv Shastry, Dhanvi Shah, Nicholas Meeks



1. Introduction

1.1 Problem Statement

The primary problem we are addressing is the seemingly unpredictable nature of rideshare applications' pricing models in Boston. It is often unclear what conditions prompt the companies to activate "surge pricing" and which neighborhoods are most expensive to travel in and out of.

1.2 Our Interest

As college students currently living in Boston, often relying on applications like Uber and Lyft for transportation, we have often also been confused by this lack of stability. In particular, paying tuition to Boston University has left us with few spare allowances for additional expenses like transportation. Additional research shows intense complaints from consumers regarding Lyft's price-surging strategies. In recent news, the company charged surge prices during a heavy rainstorm in Manhattan.[1] Therefore, we are intrigued by historical data on rideshare transportation and the weather conditions surrounding them; with this information, we hope to try to predict what our ride prices may look like for future trips.

1.3 Our Guiding Mission:

We hope to understand rideshare companies' pricing models, specifically Lyft. Ultimately, we would like to help other cost-conscious students, like ourselves, use our findings to predict their ride prices without inputting their requirements into rideshare applications, as this could risk triggering surge pricing.

2. Data Description

2.1 The Data

Our team was mainly deciding between two datasets for this project. The first dataset was about levels of happiness. However, we wanted our dataset to relate more to our lives as Boston University students and to be more meaningful. Additionally, there weren't enough data points in this dataset to make accurate predictions about specific scenarios. Boston University students use Uber and Lyft frequently, so it would be beneficial and applicable to find out what factors impact prices and how we can predict prices.

The main objective of using the online Kaggle dataset called "Uber and Lyft Cab Prices" by Ravi Munde was to predict the price of an Uber or Lyft based on various factors[2]. This dataset was collected between mid-November 2018 and mid-December 2018. Within this Uber/Lyft dataset, there were two separate files. The first file had price and venue rideshare data, and the second file had data about the weather for any given day. To consolidate all the data into one dataset we merged both datasets. Following the merging of both datasets, there were 22 dimensions in total. The dataset has over 307,000 data points, each corresponding to a ride taken in Boston in this period.

---

[1] Crane, Emily. "Uber, Lyft Ripped for Surging NYC Prices during Storm, Flooding: 'Slime Balls.'" New York Post. New York Post, September 29, 2023. https://nypost.com/2023/09/29/new-yorkers-rip-uber-lyft-for-surging-prices-during-storm/.

[2] RaviMunde. "Uber & Lyft Cab Prices." Kaggle.com, 2019. https://www.kaggle.com/datasets/ravi72munde/uber-lyft-cab-prices.

2.2 Data Pre-Processing

The first step we took was to merge both datasets. The weather DataFrame had a temperature for each hour at each location. So, we aligned that with the rideshare, DataFrame to get the weather data for both the source and destination of each ride.

The next step was to clean our dataset to filter on the rideshare brand. After realizing that our merged dataset had many observations and the Uber data had more N/A variables, we kept only Lyft with 307,408 observations. Even after dropping rows with N/A values in the Lyft-focused dataset, we were still left with 92% of our data, making this a large sample suitable for analysis.

We then dropped three variables: “id”, “product_id”, and “cab_type”. “id” was dropped because it was the same as “index”, “product_id” was dropped due to the lack of information it provided and its inherent similarity to “name”, and “cab_type” was dropped because we decided to only work with Lyft as a car service.

We also found that the raw data used to describe the date and time that each ride took place made it difficult to standardize and analyze our data. There were unnecessary symbols, and Python also could not recognize the cyclicality of time. Therefore, we split time_stamp into “weekday”, which accounted for the day of the week, and also “time” which described the time of day (in microseconds since midnight) that the ride was ordered.

The resulting data that we were left with had many covariate components, greatly increasing the dimensionality of the dataset. However, many of the covariates were repetitive and did not seem to add any information gain to our analysis. Our primary observation was that there were many repeated columns in the source and destination categories, such as “pressure”, “humidity”, “temp”, “clouds”, “rain”, and “wind”. Each of these variables was addressed in slightly different ways. We assumed that pressure and humidity would not affect price greatly based on prior knowledge, especially when significant weather patterns are accounted for through other variables, so these columns were dropped for both the source and destination. We also believed these two variables were not as intuitive to potential users of our algorithm. As for the temperature, clouds, rain, and wind columns, since all registered rides occurred within the same city, we assumed weather patterns would be approximately the same. Since we do not need to differentiate between the patterns in one small area so greatly, we decided to take the average of the source and destination columns for each remaining weather category and format a new column containing the mean value of the source and destination weather.

Our final variables include: “distance”, the distance in miles between locations; “destination”, the Boston-based destination of the ride; “source”, the Boston-based source of the ride; “price”, price in dollars; “surge_multiplier”, Lyft’s surge multiplier on a value from 1-3; “name”, they type of Lyft ordered; “weekday”, the day of the week; “time”, microseconds after midnight; “temp”, temperature in Fahrenheit; “clouds”, percentage of cloud coverage; “rain”, rain in inches for the past hour; and “wind”, wind speed in miles per hour.

2.3 Checking Correlations

Following our data pre-processing, we investigated the correlation between our variables. To better visualize the correlations between our covariates, we created a correlation matrix to look solely at the variable's magnitude of the correlation, as we are not interested in whether the correlation is positive or negative. We observed that our weather variables had a significantly high correlation with one another, specifically clouds having values of 0.93 and 0.90 with wind and rain, respectively. Additionally, we noticed that price is somewhat highly correlated with “name”, which specifies the Lyft ride type, and “surge_multiplier”. Because many of the variables are highly correlated, it would be beneficial to run a principal component analysis (PCA) to reduce the correlation between variables and reduce the dataset’s dimensionality.

Figure 1: Correlation Heatmap of All Features

2.4 Principal Component Analysis

PCA is a descriptive approach involving projections where the goal is to reduce the number of columns/covariates by removing the overlap of info between variables. To do this, we rely on correlations to create new principal components that are linear combinations of the original variables. The resulting principal components have no overlap.

To begin our PCA, we had to ensure that all of our variables were metric by dummy encoding categorical variables and standardized by sklearn.preprocessing to ensure that we were not biased towards specific features.

We looked at three methods to pick the number of principal components for our analysis: latent root criterion, explained variance criterion, and the scree plot. The latent root criterion showed that we should use five principal components as the first five eigenvalues given by the PCA explained_variance_ method are greater than one; eigenvalues greater than one are considered super variables having greater than average

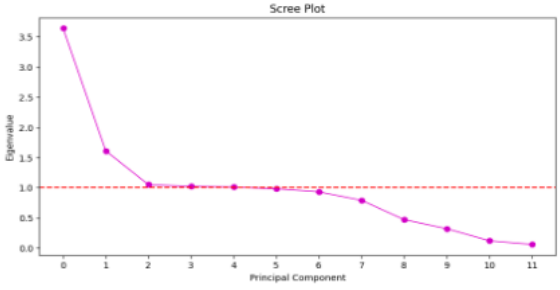


Figure 2: Scree plot to pick Number of Principal Components

explaining power. The explained variance criterion also showed that we should use five principal components as this was sufficient to explain approximately 70% of the variance in the data. Lastly, the scree plot showed that we should use three principal components as this was the component at the "elbow" of the plot shown below in Figure 2. Ultimately, we decided to go with five principal components in accordance with the latent root and explained variance criteria.

After examining the loadings for each of the five principal components, we decided to propose the following naming:
First Component: Bad Weather (clouds, rain, wind)
Second Component: Price Sensitivity (price, surge multiplier, name)
Third Component: Distance and Location (distance, source, destination)
Fourth Component: Good Temperature (temp)
Fifth Component: Time Stamp (weekday, time)

	F1	F2	F3	F4	F5
F1	1.0	-0.0	-0.0	-0.0	-0.0
F2	-0.0	1.0	0.0	0.0	-0.0
F3	-0.0	0.0	1.0	0.0	-0.0
F4	-0.0	0.0	0.0	1.0	-0.0
F5	-0.0	-0.0	-0.0	-0.0	1.0

Figure 3: Correlation Heatmap of Principal Components

To ensure that our components were calculated properly, we decided to build another correlation matrix to visualize the correlations between components one through five. The matrix we built aligned with our original expectations that the correlations between all components other than the component itself would be low or close to zero. As a result, we could move on with our analyses requiring PCA.

3. Methodology

3.1 Guiding Questions and Our Thought Process

Our main guiding question is: What factors influence Lyft prices in the Boston area, and what models best predict the price? Initially, our null hypothesis was that weather conditions and time most heavily influence Lyft ride prices. Throughout this process, we continued to simultaneously find out which variables had the greatest impact on the pricing model associated with each ride.

3.2 Analyzing the Dataset

We produced several plots for preliminary visualization of the different variables and would like to dive into further detail on the insights we gained.

Figure 4: Histogram of Price of Lyft versus Number of Lyfts Called

The first possible relationship we investigated was the correlation between the price of Lyft and the number of Lyfts ordered within that price range. With Boston being a densely populated city with a relatively small radius, we predicted cheaper rides would be more prominent than expensive rides, as people will typically have to travel less distance. The results of our histogram support this, as it displays an extreme influx of rides between 0 and 20 dollars of about 200,000 rides and a dramatic decrease in rides to 100,000 as prices increase above $20.

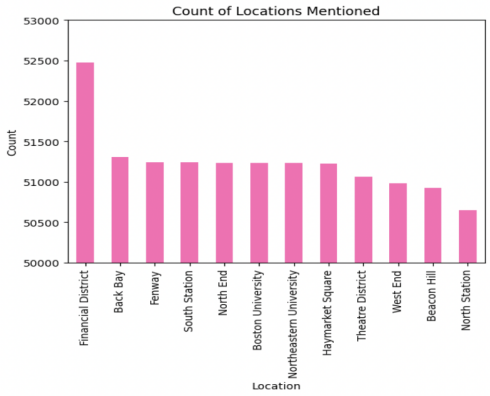


Figure 5: Bar Chart of District of Boston versus Number of Lyfts Called

Next, we decided to investigate the correlation between the district of Boston and the number of Lyfts called in said district. As seen below in our bar graph depicting the relationships, there is an overwhelming peak within the Financial District of almost 1,250 more Lyfts called than any other district, while the rest are relatively similar. Other locations mentioned, such as Boston University and Fenway, are mostly student housing. Being that students are in a proximity to their campus and have less money to spend, it seems plausible that its count would be much lower than the financial district.

We then decided to look into the average price of a Lyft per time of day. Apart from the assumption that Lyft prices may spike at peak times due to the surge multiplier, we had no prior knowledge of how the price would fluctuate throughout the day. The data indicates that there is minimal fluctuation between each time of day, with the greatest difference being about $0.02 between morning and night.

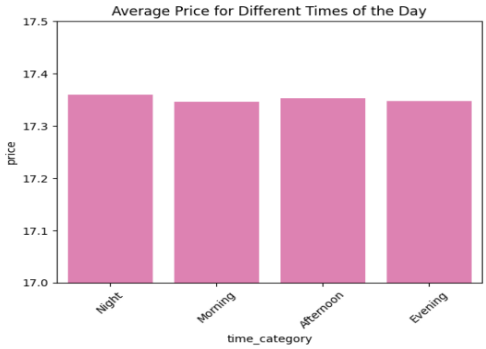


Figure 6: Bar Chart of the Time of Day versus the Average Price of the Lyfts Called

Finally, we investigated the average price of each different type of Lyft that could be ordered. We suspected that we already had a firm understanding of what this relationship would look like – as college students, we are constantly looking to find the cheapest deal to travel. Thus, it was no surprise that Lyft Share and Lyft were the cheapest options, while Black options were the most expensive. However, with such a dramatic price gap between different types of Lyfts, with the largest average gap being about $28 between Lyft Share and Lux Black XL, it is clear that the type of Lyft is extremely important when predicting prices.

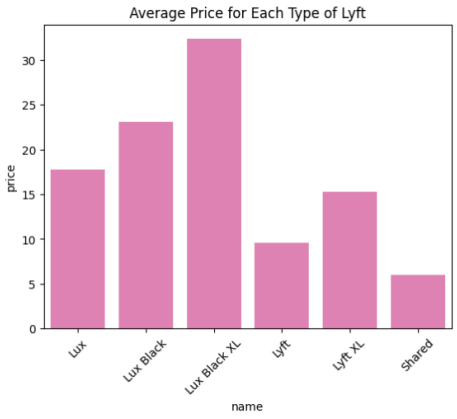


Figure 7: Bar Chart of the Type of Lyft Versus the Average Price

3.3 Modes of Interpretation and Individual Findings

3.3a Linear Regression

We conducted a linear regression on the PCA data to analyze which variables have the most impact on prices. This method shows the relationship between a dependent variable and one or multiple independent variables.

We found that F4 (‘good_temperature’) had the highest relative weight on price. This variable consists of factors related to weather, specifically good temperature. If all else is held constant, one unit increase corresponds to an average of 0.003 unit increase in price. After that, F3 (‘distance_location’) was the second most impactful group of variables. This consisted of variables related to location, like source, destination, and distance. If all other factors were held constant, one unit increase in F3 corresponded to an average increase of 0.002 units in price. Lastly, F5 (‘time_stamp’) corresponded to an average decrease of 0.000108 units in price.

Although these amounts seem nominal, it is important to note that the PCA data is scaled at 0-1, and therefore, these coefficients are more impactful than they seem. This model had a mean squared error (MSE) of 1.604 and a root mean square error (RMSE) of 1.267, showing that the predicted pricing values are fairly far from the actual values.

3.3b k-Nearest Neighbors

Another simple-to-understand, yet meaningful model we used was k-Nearest Neighbors. By fitting a clustering model to a training subset, consisting of about 60% of the data in our dataset, we could make groups that best fit most of our data.
Using this, we sorted the untouched testing data into these groups to make predictions about the prices for testing data. We measured the accuracy of these predictions using mean squared error and root mean squared error. This way, we could compare the predictions to the actual prices. The lowest RMSE occurred for k=2 at around 0.382.

3.3c Decision Trees and Random Forests

Decision trees were one of the most complex models we attempted to create to predict Lyft prices in the Boston area. To better understand how the structuring of the covariates would impact the decision trees, and, ultimately, the prediction accuracies, we decided to analyze both all covariates and only the five principal components using a decision tree and random forest.

For the first set, we wanted to analyze how accurately we could predict the prices with trees based on all components. This tree we created could be known as a "full tree" which is likely overfitting by capturing noise and data. This tree was not built to make further predictions, but rather served as a benchmark for comparing the accuracies of the other trees- it had a root mean squared error of 1.874. By introducing additional parameter settings such as minimum records to split and max depth in tandem with grid search on training data, we built multiple trees using different values for the parameters. We then measured accuracy on the testing fold and chose the tree settings with the best performance on the validation, i.e., test data. We chose to create a grid search function so that we could most efficiently test multiple combinations of minimum records to split. Max depth to see which one resulted in a tree with the lowest mean squared error. If the RMSE for the currently created tree was better than the baseline or a tree created before it, the best RMSE value would be saved as that of the current tree and the parameter settings would be saved as well. We tested the trees where min_samples_split_values were [2, 5, 10, 15, 20] and max_depth_values were [3, 5, 10, 15, 20]. After our grid search, we found that the best hyperparameters were {'min_samples_split': 20, 'max_depth': 15}. Finding the best parameters alone for a single tree led to a 36.393% decrease in the mean squared error, a promising improvement.

For the second set, we wanted to assess the accuracy of predictions for trees constructed with just the PCA component. This tree also served as a benchmark to compare against the accuracies of the other trees. Again, we then took steps to limit tree growth for optimal accuracy. By reusing the grid search again, we tested the trees where min_samples_split_values were [2, 5, 10, 15, 20] and max_depth_values were [3, 5, 10, 15, 20]. After running our grid search, we found that the best hyperparameters were {'min_samples_split': 10, 'max_depth': 20}. Finding the best parameters alone for a single tree led to a 14.863% decrease in the mean squared error, a promising improvement.

The decision trees created with PCA components are more accurate than those created with all components, and the difference is greater for the baseline tree than for the grid search-based tree.

Moreover, we can create random forests to allow ensemble methods. While random forests increase prediction accuracy and provide better performance by several extensions to trees that combine results from multiple trees. However, they reduce interpretability, and you lose the rules embodied in a single tree. We can also make a random forest regressor while limiting the min samples to split to 20 and max depth to 15. Since these are the best hyperparameters that resulted in the lowest RMSE for a single tree, we wanted to experiment with seeing random forest RMSE with these hyperparameters as well. We can also make a random forest regressor while limiting the min samples to split to 10 and max depth to 15. In both of the aforementioned cases, we see that the random forests cause RMSEs to decrease in comparison to the single tree, no matter whether the forests are created with all or solely PCA components.

RMSEs of Decision Trees and Random Forest

	All Components	PCA Components
Single Tree - Baseline Tree	1.874	1.386
Single Tree - Best Grid Search	1.192	1.180
Forest (1000) - Baseline Tree	1.425	1.035
Forest (1000) - Best Grid Search	1.356	1.116

Figure 8: Comparing the Decision Trees and Random Forests built on Different Components

3.3d Neural Networks with Deep Learning Aspect

Another model we decided on to predict prices was neural networks with an extension to deep learning. We chose neural networks as they have high predictive performance and can capture complicated, non-linear relationships between variables.

The first thing we had to do to construct our neural networks was to preprocess our data so that all features must be numerical. We dummy encoded categorical variables and normalized the data into [0,1] range using the MinMaxScaler() function so that the different features are on a similar scale, which helps to stabilize the gradient descent step, allowing us to use larger learning rates or help models converge faster for a given learning rate.

We decided to make a function called LyftNNCreator(size), where size can include information on the number of hidden layers and the number of nodes in each hidden layer. We created two neural networks, one with two nodes and another with five nodes, and two deep learning networks, one with 2 nodes in each layer and another with five nodes in each layer. We picked these values based on the models we examined in class. Additionally, we chose our activation function to be of type 'identity', as the identity solver returned the lowest errors compared to ‘logistic’, ‘tanh’, and ‘relu’. We chose the ‘adam’ solver as it can handle large datasets efficiently, making it suitable for scenarios like ours with a substantial amount of training data, and as it tends to work well with deep neural networks and complex models. Since it adapts the learning rates for each parameter individually, it makes it well-suited for models with many parameters. The resulting root mean square errors were as follows:

Neural Network with 1 hidden layer, 2 nodes:
6.128
Neural Network with 1 hidden layer, 5 nodes:
6.081
Neural Network with 3 hidden layers, 2 nodes each:
6.146
Neural Network with 3 hidden layers, 5 nodes each:
6.091

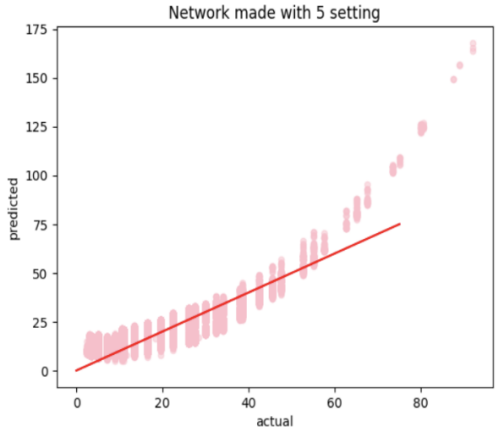


Figure 9: Neural Network built with 1 layer, 5 nodes Residual Plot

While the best accuracy is provided by the neural network with one hidden layer and 5 nodes, the difference is not significantly different between all the variations. This may be possible since the architectures we experimented with do not vary significantly in complexity. Since the models have a similar number of parameters or layers, they might capture the same level of complexity, resulting in similar performance. As seen in Figure 9, with all models, we see that predictive abilities taper for rides with costs over $60. Although it seems as if the RMSE for this data is unusually high, this can be credited to the fact that raw data is reported on a much larger scale than the PCA data used for the other models.

4. Conclusions and Diving Deeper

4.1 Major Conclusions

	Naive Model	Linear Regression	K-Nearest Neighbors	Single Tree (with Best Grid Search)	Random Forest (with Best Grid Search)	Best Neural Network
Mean Squared Error	100.383	1.604	0.146	1.386	1.245	36.979
Root Mean Squared Error	10.019	1.267	0.382	1.180	1.116	6.081

4.2 Analysis of Insights

We found that the best model for predicting price, in terms of accuracy as measured by the mean squared error, was k-Nearest Neighbors. The mean squared error for this model, which calculates the mean difference between predicted and actual price, was 0.146, which is significantly lower than the mean squared errors for the other models.

If we had further resources, time, etc., we would love to fine-tune the other models to make a portfolio of methods that can be used in tandem to create the best prediction algorithm for Lyft pricing. This would also be more possible if we had more computing power on our laptops to create more models with increasing complexities.

4.3 Data Limitations

It is important to concede that this dataset and our models have significant limitations. Firstly, we would like to highlight that this data is slightly outdated. These rides took place in 2018 and since then, COVID-19 has significantly impacted pricing models. The value of the US dollar increased by around 22 cents since this data was collected[3]. Another limitation was that Thanksgiving and Black Friday, when many Americans choose to travel, occurred during this period. This dataset also does not account for how

[3] Bls.gov. “CPI Inflation Calculator,” 2023. https://www.bls.gov/data/inflation_calculator.htm.

many rides are requested at a given time; the variable for surge pricing cannot account for the various levels of demand and any other incentives for consumers to choose certain rides like promotions. Additionally, for neural networks, it is possible to keep experimenting with additional hidden layers and nodes that could also impact the weights and biases calculated to predict pricing. However, we were not able to achieve this perfect model because we did not have access to this additional data and also did not have the capability to run experiments with these additional hidden layers and nodes.

4.4 General Implications

The primary motivation for this project was for our team, as Boston University students who rely on public transport, to understand how ride-share pricing is done. Firstly, through the use of our visualizations, we observed how cab-type and surge multipliers had a significant impact on Lyft cab pricing, but time-related variables such as day of the week and time of day seemed to have smaller impacts. Even simple conclusions like these will hopefully inspire a college student who may think walking late at night is much cheaper than a Lyft, to take the safe ride home instead. Additionally, through the K-Nearest Neighbor model we created, and hopefully further interactions, we hope to be able to give residents of Boston an opportunity to predict the price of a future planned Lyft across popular Boston locations.

4.5 Next Steps

We would like to incorporate other data into our analysis and potentially try unexplored models that may be better suited for this project. For additional data, we may look into incorporating the demand for rides at any given time, the specific location of rides from popular places like the airport, and holidays. Furthermore, it would be interesting to see these trends over a couple of years to account for inflation, seasonality, and other factors related to cyclicality. Lastly, it is important to account for internal trends, like promotions or coupons issued by the company to get the best predictive model that can be applied to a larger sample space of scenarios.

Bibliography

Bls.gov. “CPI Inflation Calculator,” 2023. https://www.bls.gov/data/inflation_calculator.htm.

Brown, Davis “The Algorithms Behind Pricing Your Ride,” 2023. https://www.fuqua.duke.edu/duke-fuqua-insights/algorithms-behind-pricing-your-ride.

Crane, Emily. “Uber, Lyft Ripped for Surging NYC Prices during Storm, Flooding: ‘Slime Balls.’” New York Post. New York Post, September 29, 2023. https://nypost.com/2023/09/29/new-yorkers-rip-uber-lyft-for-surging-prices-during-storm/.

Encinas, Amaris. “Late to the Airport during the Holidays? Lyft Will Pay You for That ‘Even If You Take a Taxi.’” USA TODAY. USA TODAY, November 14, 2023. https://www.usatoday.com/story/travel/news/2023/11/14/lyft-promo-airport-late-holidays/71583407007/.