Capstone Project Report: Dynamic Pricing Analysis Using Uber and Lyft Data

Emeka Amadi, Yunchae Shin

Project Overview

What is dynamic pricing? At a basic level, most of us understand the concept: companies adjust the prices of their products based on the demand these products receive. A common example is seen in the airline industry. You may have noticed how flight prices change when you check back after some time, often increasing significantly. But how and why does this happen?

In the case of airlines, as the supply of seats decreases and more latecomers try to book flights, companies seize the opportunity to maximize profits. Why not increase prices if consumers, needing to travel, are essentially forced to pay? And with limited seats available, this strategy makes business sense. The true essence of dynamic pricing kicks in when these price changes are automated through algorithms.

In this report, we delve into the conditions that impact pricing using ride-share data from Uber and Lyft, combined with weather data and attempt our own dynamic pricing strategy. First, we'll examine how fares for base (non-dynamically priced) rides differ from dynamically priced rides, and whether certain conditions influence these variations. Uber and Lyft implement dynamic pricing through a 'surge multiplier' - when demand for rides spikes, this multiplier increases, and so does the ride's cost.

Another focus of our analysis is to simulate our own demand estimation, akin to a surge, with the aim of calculating a dynamic price based on this estimation. This exploration of the base data, dynamic data, and demand-estimated data will be conducted through predictive price models for each dataset. Additionally, we've developed a web application that allows users to compare how these three models differ in pricing based on conditions, demonstrating how price fluctuates with demand, the features that may affect demand, and how dynamic pricing affects revenue.

Data and GitHub Repository Documentation

Accessing the APIs of Uber and Lyft requires authorization, which we were unable to obtain. Instead, our data was sourced from Kaggle, where datasets were originally gathered through the Uber and Lyft APIs. This data represents what the price and surge data of a ride would be if a ride was taken at that time. It's important to note that Uber and Lyft actual ride data is not publicly available.

Our dataset encompasses ride data collected from 12 different locations in Boston. The pricing data was collected from the APIs every five minutes over a span of approximately two weeks.

Additionally, weather data was collected hourly. These datasets are referred to as 'cab_rides.csv' and 'weather.csv' in our repository and on Kaggle. The project is thoroughly documented and hosted on a GitHub repository, titled Dynamic_Project.

Pipeline

We have crafted a pipeline that encompasses the entire spectrum of our data handling, preprocessing,demand estimation, model training, and web application development.

1. Pipeline Stages

1.1 Data Cleaning and Feature Engineering

The first step in our project involved the foundational tasks of importing, preprocessing, cleaning, and feature engineering, all detailed in our 'preprocess.py' file. We started with the 'cab_rides.csv' and 'weather.csv' datasets sourced from Kaggle, focusing first on removing irrelevant data. Next, we tackled the timestamps, converting them from epoch numerical representations to a more user-friendly date-time format. This transformation was key to making the data more understandable and analytically useful. Given the impact of weather on ride-sharing demand, we then categorized the weather data. Rainfall data was simplified into a binary variable (rain or no rain), and temperature values were grouped into different ranges. These modifications allowed us to identify and analyze trends related to weather conditions.

Vehicle types also underwent a categorization process. Varieties like 'Black', 'Lux Black', and similar were grouped into a single “Luxury” category. We added columns to indicate weekdays and rush hours, enabling us to model how these time-related factors influence travel demand. The final step in our data preparation was merging the ride data (data_df) with the weather data (weather_df) based on the closest timestamp. This approach ensured that each trip’s data was accurately paired with the corresponding weather conditions.This cleaned and engineered data is available through our 'get_cleaned_data' function.

1.2 Data Model Preprocessing

Before the data can be modeled, it must undergo additional preprocessing steps. This is where our ‘prepare_data’ function comes into play. It's designed to ready the dataset for the predictive modeling phase. Here's how it works:

The target variable 'y', which in our case is the 'price', is then separated from the feature set. For the remaining features which are in 'X', we perform preprocessing tailored to their data types

Numerical features are identified and normalized using StandardScaler. Categorical features are also identified and transformed using OneHotEncoder. These preprocessing steps are then bundled using ColumnTransformer. By the end of this process we have a transformer to input into our modeling pipeline that follows.

After our preprocessing steps, the final input features we use for our model are:

"**cab_type**", "**source**", "**car_type**", "**weekday**", "**rush_hour**", "**is_raining**", "**temp_groups**"

The output variable for our model is "**price**", which represents the cost of a ride.

1.3 Demand Estimation

In our quest to understand and replicate dynamic pricing mechanisms similar to that of Uber and Lyft, we devised our own demand metric, simply titled 'demand'. Our approach utilized a Bayesian method known as Thompson sampling, which informed us about how prices impact demand. This Bayesian approach allowed us to construct a demand function defined by three key parameters: eta (η), a, and b. These parameters interact in the following way:

- eta (Price Elasticity of Demand): This is a measure of how sensitive the quantity demanded is to a change in price, represented by the equation:

  eta = % change in quantity demanded / % change in price

- a (Demand Scaling Factor): This acts as a modifier for the demand curve, representing factors other than price that could affect demand, such as seasonal effects or marketing campaigns.
- b (Baseline Demand): This signifies the quantity demanded when the price effect is negligible, essentially serving as the demand level baseline.

These parameters come together in our demand equation:

`Q(Quantity Demanded)= a ·P^−η + b`

where Q is the Demand and n is eta.

Following this, is our dynamic pricing equation:

Price=Base Price×(1+η×Demand Factor)

In order to maximize their profits in a competitive market, suppliers must strategically determine their own prices, taking into account the pricing policies of other suppliers. (Jiang, R., Qin, T., &

An, B., 2018) To this end, we leveraged the PyMC3 framework to build a Bayesian model and perform parameter estimation to find an effective approximate equilibrium under complex market conditions. Despite the lack of real-world data and computational difficulties, this approach allows us to derive a strategy that adapts to changing supply and demand conditions in real time to optimize profits. It focuses on how market participants can optimize their policies and seek maximum profit in the face of competition. Together, these parameters (eta, a, and b) in the demand function provide a comprehensive view of how demand varies with price, accounting for both the direct impact of price changes (through eta) and other factors influencing demand levels (a and b). By estimating these parameters using Bayesian methods in PyMC3, the function aims to create a nuanced and probabilistic understanding of demand dynamics in the given dataset.

Thompson Sampling Analysis

Parameter Estimation in Bayesian Modeling and Demand Analysis

To estimate demand, we used the PyMC3 framework, which is a powerful tool for constructing and inferring Bayesian statistical models. Our approach was systematic, starting with data preprocessing using Theano tensors, and ending with the complex process of defining and testing prior distributions for eta(η), a, and b, which are important parameters in demand analysis. We leveraged the strengths of PyMC3 to include Poisson processes in this process, which according to Agrawal (2021) is particularly useful for modeling demand in dynamic pricing and learning models.

Bayesian Modeling Approach

- Data Preparation: We began by structuring our dataset in chronological order and then grouping the data by unique combinations of Source, Destination, and Car Type. The estimation is based solely on the price column for each combination.
- Demand Function Definition: Our focus was on the non-linear impacts of ETA on demand, modeled using gamma functions and accounting for Gaussian noise to define the likelihood function.

Parameter Estimation via MCMC

- Sampling Methodology: We employed Markov Chain Monte Carlo (MCMC) techniques based on the reference(Mathias Leys, 2023), iterating thousands of times to gather estimates of eta, a, and b, inclusive of parameter uncertainty.
- Visualization: We plotted demand functions representing averages of all samples, providing a lucid visual representation of how demand fluctuates with price changes.

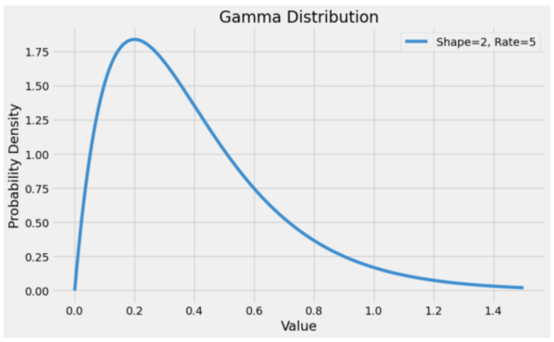
Trials and Results

We conducted multiple trials to ascertain the most fitting distributions for each parameter.

	1st trial	2nd trial	3rd trial	4th trial
eta	Gamma dist (shape=2, scale=⅕)	Gamma dist (shape=3, scale=⅙)	Uniform dist (from 0 to 2)	Gamma dist (shape=2, scale=⅕)
a	Uniform dist (from 0 to 100)	Uniform dist (from 0 to 70)	Uniform dist (from 0 to 10)	Uniform dist (from 0 to 10)
b	Uniform dist (from 0 to 20)	Uniform dist (from 4 to 40)	Uniform dist (from 4 to 70)	Uniform dist (from 4 to 70)

Table 1. Demand Estimation Trail Details

1. Eta (η) Estimation
- Initial Approach: Assumed a gamma distribution(Michael Kozak,, 2020), conducting several experiments with varying shape and scale.
- Third Trial: Shifted to a uniform distribution to identify potential skews, later approximating it with the closest gamma distribution.
- Final Decision: Upon reviewing the consistency and performance of each version against actual prices, we found the first version (Gamma distribution, shape=2, scale=⅕) to be the most effective, leading us to adopt these parameters in the 4th trail.



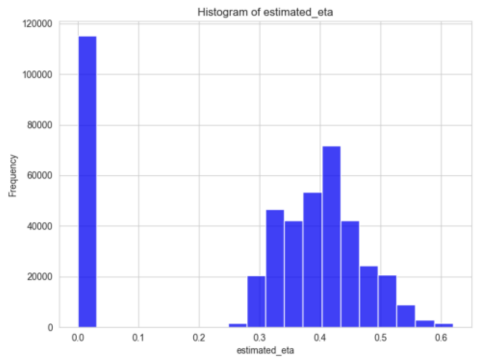


Figure 2. Gamma Distribution for generating eta

Figure 3. Histogram of Simulated eta Values

Figure 1 shows a gamma distribution, with initial settings of a shape parameter of 2 and a rate parameter of 5. Our research indicated that eta follows a gamma distribution leading to our choice in following a gamma distribution. Figure 2 shows a histogram of eta values derived from our MCMC simulation in trail one, aligning with the gamma distribution and indicating a concentration of values in a median range. Due to our estimation slightly differing from the gamma distribution we attempted a uniform distribution in trial three however the results were worse than the gamma distribution so we decided to stick with the gamma distribution. \

2. a and b Estimation

- a Parameter: Initially a uniform distribution ranging from 0 to 100, gradually narrowed based on MCMC findings.
- b Parameter: Began with a range of 0 to 20, incrementally adjusted upper limits to pinpoint the exact distribution.
- Conclusive Trends: Both parameters a and b showed converging trends, leading us to finalize our choice with the parameters from the third trial.

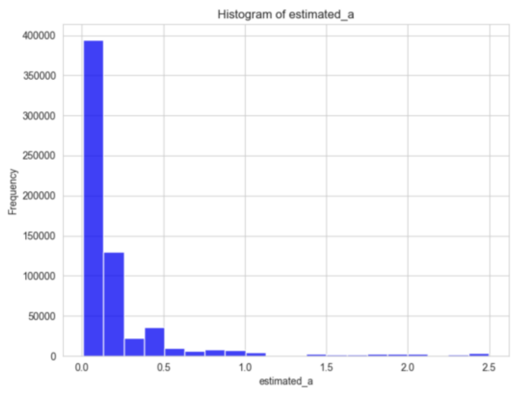


Figure 4. Distribution of Estimated Parameter a Trail 1

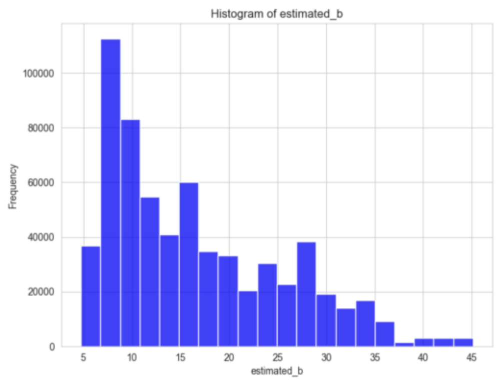


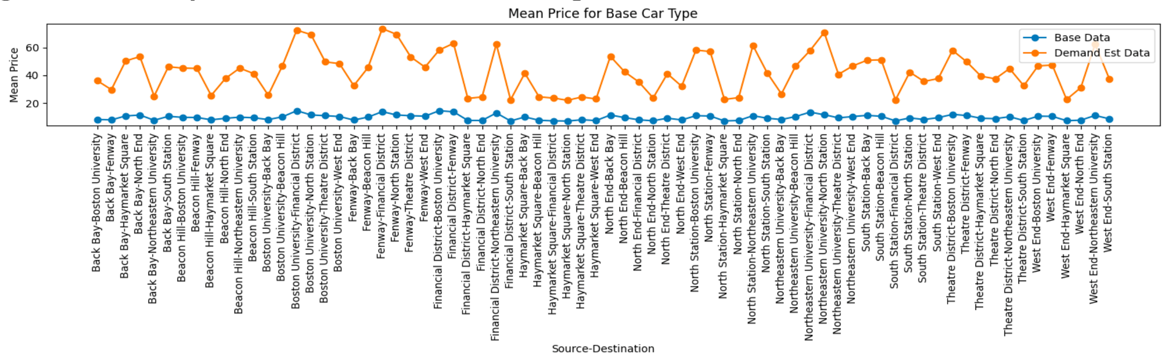
Figure 5. Distribution of Estimated Parameter b in Trail 1

Figure 4 and Figure 5 show the distribution of estimates from MCMC simulations of the parameters a and b used in the economic demand function. In Figure 4, parameter a most of the estimates are clustered around a low value, which can be interpreted as having a relatively stable value. On the other hand, parameter b in Figure 5 shows that the estimates are spread over a wider range. This suggests that the lower bound b can take on a wide range of values in demand forecasting, and these two parameters provide important information about how their respective bounds should be adjusted during the optimization process.

In evaluating the effectiveness of each trial, our primary method was to compare the mean estimated prices derived from our demand model against the actual prices observed for each source-destination pair. The goal was to identify the trial where our demand-based pricing closely mirrored the real-world prices, minimizing the gap between our estimations and the actual data.

Figure 6 below showcases this comparison process for two car types: the base car type and the luxury SUV car type. It was in trial one that we observed the most accurate predictions, showing the most divergence between our modeled prices and the actual prices.

During subsequent trials, we noticed a trend: while our model might yield a close fit for one car type, it could simultaneously produce wildly inaccurate predictions for another, with discrepancies ranging into the hundreds of dollars. Such variations underscore the challenges in creating a universally accurate demand estimation model and highlight the nuanced nature of price sensitivity across different service categories.



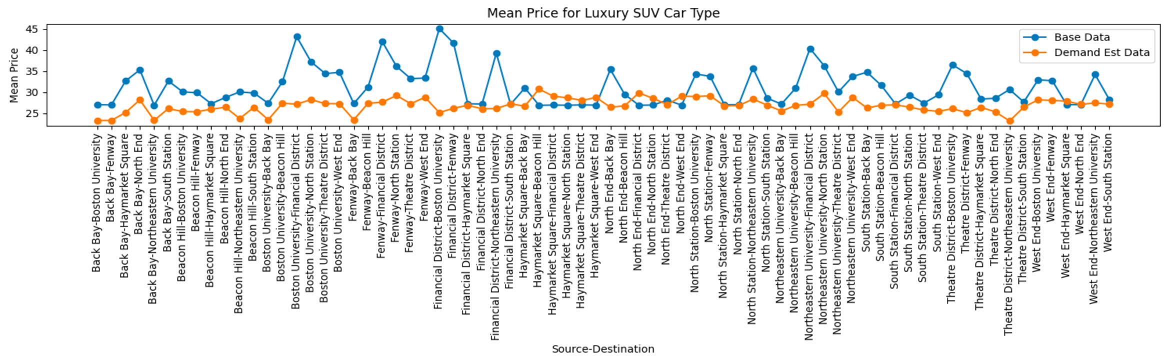


Figure 6. Comparison of the Mean Demand Est Price and the Real Price Per Source Destination Combination

Conclusion on MCMC

We found that our first tracked trail displayed the best results. We thought the 4th trail would be the best after getting parameters a and b to converge but these parameters led to extreme prices for some car types. Our comprehensive Bayesian modeling and parameter estimation process revealed key insights into dynamic pricing via demand estimation. Despite extensive testing, achieving an optimal model is a difficult task. Demand estimation is a complex, time-consuming process, as each iteration takes 6-8 hours to run. Ultimately to get an optimally fitting model the process takes extensive trial and error.

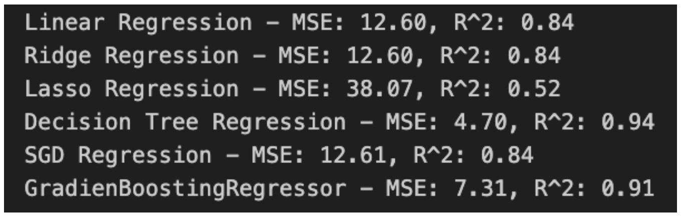
1.4 Modeling Structure

Our modeling process encompassed three distinct models: a base model generated from the non-dynamic data (where surge equals 1), a dynamic model that includes the entire dataset (encompassing both surge equals 1 and surge greater than 1 scenarios), and a demand estimation model derived from data created via PyMC Bayesian modeling.

The full dataset was quite extensive, with approximately 637,976 data points, of which around 20,000 had surge values exceeding one. This disproportionate distribution of surge values poses challenges for the dynamic model, potentially impacting its predictive accuracy.

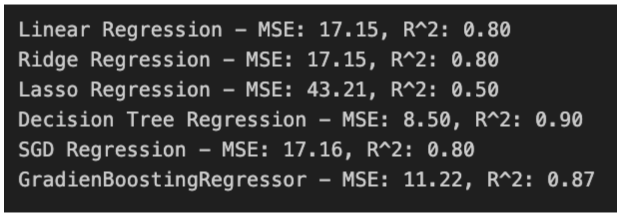
We employed various regression techniques—Linear, Ridge, Lasso, Stochastic Gradient Descent (SGD), Decision Tree, and Gradient Boosting—to identify the most accurate price estimation model. Decision Tree Regression emerged as the top performer for all models, evaluated based on Mean Squared Error (MSE) and R-squared (R2) metrics.

As demonstrated in the test results shown below, Decision Tree Regression had an MSE of 4.70 and an R2 of 0.94 for the base model, and for the dynamic model, an MSE of 8.50 and an R2 of 0.90. After extensive hyperparameter tuning through grid search, we observed no significant improvement over the default parameters, leading us to retain them.



```
Linear Regression - MSE: 12.60, R^2: 0.84
Ridge Regression - MSE: 12.60, R^2: 0.84
Lasso Regression - MSE: 38.07, R^2: 0.52
Decision Tree Regression - MSE: 4.70, R^2: 0.94
SGD Regression - MSE: 12.61, R^2: 0.84
GradienBoostingRegressor - MSE: 7.31, R^2: 0.91
```

Code Block 1. Base Model Test Results



```
Linear Regression - MSE: 17.15, R^2: 0.80
Ridge Regression - MSE: 17.15, R^2: 0.80
Lasso Regression - MSE: 43.21, R^2: 0.50
Decision Tree Regression - MSE: 8.50, R^2: 0.90
SGD Regression - MSE: 17.16, R^2: 0.80
GradienBoostingRegressor - MSE: 11.22, R^2: 0.87
```

Code Block 2. Dynamic Model Test Results

Despite the favorable metrics, the dynamic model's predictions, particularly for cases where surge was greater than 1, were not ideal. It underpredicted actual prices, as shown in Figure 7 and Figure 8 below. This is likely because the model lacks the ability to discern between surge and non-surge cases through the input features alone—since the surge multiplier itself was not included as a feature.

To delve deeper into this, we tested the high-surge scenarios directly, obtaining an MSE of 112.26 and an R2 of 0.44. This confirmed the belief we had that the dynamic model does not predict the cases where surge is greater than one well, as seen in the visual representation in Figure 7 and 8.

In this project, a challenge we faced was enabling the dynamic model to out predict the base model in the surge greater than one case. This pinpoints an area that is in need of further investigation. The core challenge lies not in a direct class imbalance, as surge values are not our target output, but rather in a more complex, underlying imbalance within the data. This subtle issue hinders the model's ability to adeptly predict surge pricing scenarios. Overcoming this hurdle is crucial and presents an exciting opportunity for future research. Delving into this aspect could yield significant improvements in the dynamic pricing model.

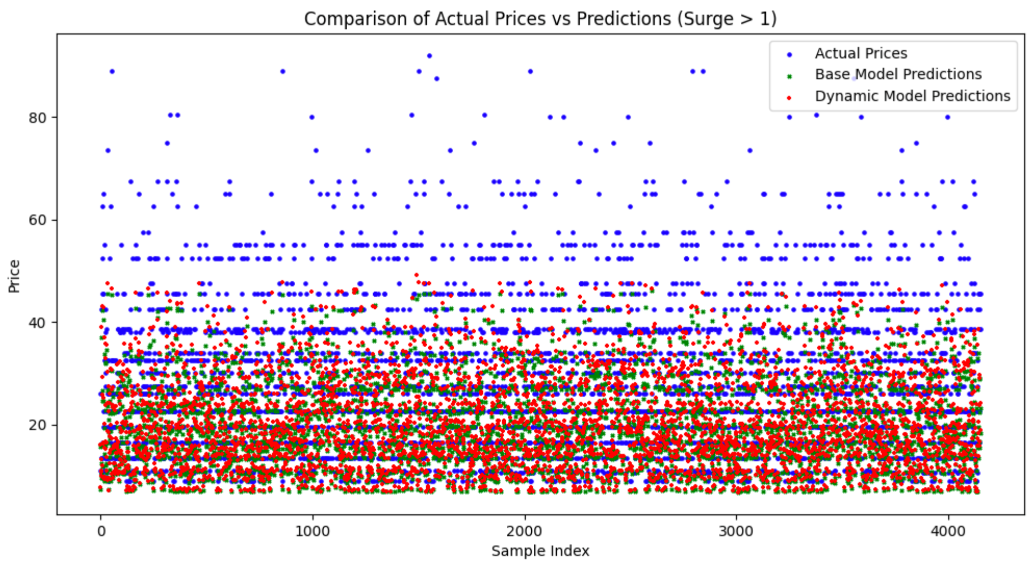


Figure 7. Scatter Plot of Actual Prices vs Predictions for Base and Dynamic Model on the Full Test Set

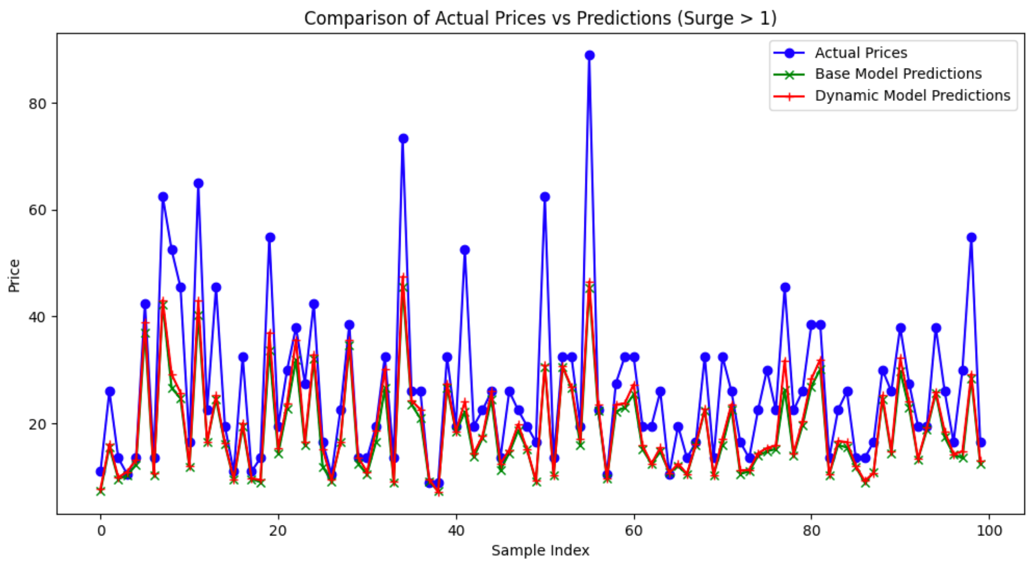


Figure 8. Line Plot of Actual Prices vs Predictions for Base and Dynamic Model on the first 100 points of the Test Set

2. Web Application

2.1 Run.py

The run.py file serves as the central execution script for the Capstone Project on Dynamic Pricing Analysis using Uber and Lyft Data. It orchestrates the complete pipeline from preprocessing to model training and finally to running the Streamlit application for data visualization.

The flow of Run.py file

1. Initializes the Process: Importing necessary libraries and modules such as Pandas, NumPy, and specific project modules like preprocess and train.
2. Data Loading and Preprocessing
3. Demand Estimation:
    - Includes commented-out lines for running the demand estimation process which is time-intensive.
    - Utilizes pre-generated demand data for efficiency.
4. Model Training:
    - Segregates the data into base, dynamic, and demand datasets.
    - Trains separate models for base, dynamic, and demand pricing strategies and saves them for prediction purposes.
5. Streamlit Application Execution:
    - Executes the Streamlit application (app.py) which provides an interactive interface for users to input conditions and visualize model predictions.

2.2 Visualization and Revenue Estimation Analysis (In web application)

We implemented a web application that harnesses the predictive power of three distinct models—Base, Dynamic, and Demand Estimation—to forecast potential demand and revenue generation under varying pricing strategies. This section delves into the visualizations generated by the app, illustrating the demand at different predicted prices and the corresponding revenue implications for each pricing strategy.

Demand Function Visualization

The demand curve, a fundamental concept in economics, represents the relationship between price and the quantity of a service that consumers are willing to purchase. Our application visualizes this curve and overlays the estimated prices from our three models: the Base Model, the Dynamic Model, and the Demand Estimation Model.

- Price of Base Model: This represents the standard pricing without considering demand fluctuations or other dynamic factors.

- Price of Dynamic Model: This price is adjusted based on varying factors such as time of day, weather, and real-time demand.
- Price of Demand Estimation Model: Developed from the economic demand function, this price takes into account the estimated parameters to optimize pricing according to predicted demand levels.

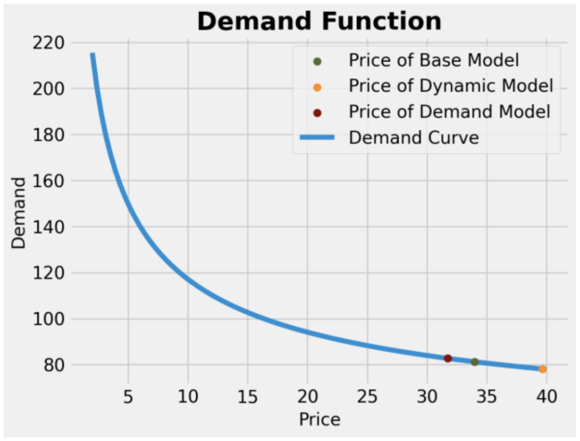


Figure 9. Demand Curve and predictions with three models

This visualization displays where each pricing model intersects the demand curve, indicating how much demand each price is expected to drive. This intersection provides an intuitive understanding of how the prices suggested by the base model, dynamic model, and demand model might perform in terms of sales volume . It also displays how an increase in price decreases demand and how a decrease in price increases demand. From the Demand and Price you can estimate the potential revenue difference for each model. On our web application we display how pricing dynamically could lead to greater revenue.

Conclusion

To wrap up, we have successfully developed a web application that serves as a visualization and analytical tool, encapsulating our three-model approach to estimate ride prices under varying conditions, and formulating our own dynamic pricing strategy. While we faced challenges with the dynamic model's predictive accuracy in specific cases, our application stands as a testament to the potential of data-driven solutions in understanding and innovating within the realm of dynamic pricing. It provides a foundation for further exploration and refinement in the dynamic pricing domain.

Ethics and Broader Impacts

Our platform is designed for both consumers and businesses . It allows consumers to dig into conditions that may lead to changes in their ride price. As a result users can plan accordingly depending on these conditions to save money if necessary. and It also enables businesses to consider the use of Dynamic Pricing in their products whatever they may be. The revenue estimation allows businesses to see how slight increases and prices in high demand moments can be very fruitful.

Ethical Concern: Information Asymmetry

Issue: Dynamic pricing algorithms often operate with vast amounts of data, including user behavior, location, and historical purchasing patterns. Customers may not have access to or understanding of the algorithms' intricacies, leading to a significant information gap between businesses and consumers.

Ethical Question: Is it ethical for businesses to utilize complex dynamic pricing algorithms that customers cannot fully comprehend, potentially putting them at a disadvantage in the marketplace?

Team Contributions and Feedback Integration

We have included a statement of work outlining each team member's contributions and have actively incorporated feedback from our project mentor, addressing major issues identified during check-ins. An optional appendix showcases efforts not included in the main report, providing a complete picture of our project journey.

	Emeka Amadi	Yunchae Shin
Contributions	Data Cleaning and Feature Engineering, Model Pipeline and Analysis, Overall Pipeline, Report	Early-stage research, experiments with MCMC(PyMC3), eta estimation model, building Streamlit app

Bibliography

Jiang, R., Qin, T., & An, B. (2018). Dynamic Pricing for Reusable Resources in Competitive Market with Stochastic Demand. In Proceedings of The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18).

Den Boer, A. V., & Keskin, N. B. (2019). Discontinuous Demand Functions: Estimation and Pricing. Management Science. First version: March 5, 2017; This version: June 13, 2019.

Agrawal, S. (2021). Dynamic Pricing and Learning under the Bass Model. arXiv:2103.05199v1 [cs.LG]. 9 Mar 2021.

Michael Kozak (2020). Predicting Revenue with Gamma Distribution, Medium Article(https://medium.com/analytics-vidhya/the-gamma-distribution-and-its-applications-in-the-mobile-app-industry-2ba891d8b7f4)

Mathias Leys (2023). Effective Dynamic Pricing in Practice, Medium Article(https://blog.ml6.eu/dynamic-pricing-in-practice-99fe2216a93d)