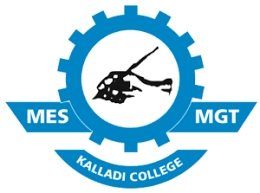
**MES KALLADI COLLEGE MANNARKKAD**

**Mannarkkad, Palakkad, Kerala**



A Project Report

On

“DYNAMIC PRICING SYSTEM”

Submitted in Partial Fulfillment of the requirement for the award of the degree

**BACHELOR OF VOCATION**

**IN**

**DATA SCIENCE & ANALYTICS**

Submitted by

Name: RAHUL K

Reg.No: KIAWBOE024

Under the guidance of

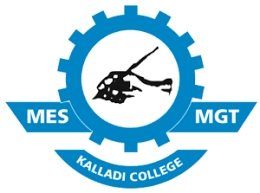
Internal Guide

**Hasfiya KP**

**Dept. of B.Voc Data Science & Analytics**

**MES KALLADI COLLEGE MANNARKKAD**

**Mannarkkad, Palakkad, Kerala**



**Department of Vocational Studies (B.Voc)**

**CERTIFICATE**

This is to certify that the project entitled "DYNAMIC PRICING SYSTEM " has been carried out by RAHUL uregno:KIAWB0E024 in partial fulfillment of the requirement for the Award of the degree of vocational studies in Data science & Analytics of Calicut University, during The year 2024-2025.

**Principal**

**Head of the Department**

**Project Guide**

Certified that the candidate was examined by us in the Project Viva Voce Examination held on………………….and his Register Number is……………………………………..

**Examiners:**

**1.**

**2.**

**ACKNOWLEDGEMENT**

I would like to express my sincere gratitude to everyone who supported me throughout the completion of this DYNAMIC PRICING SYSTEM project. First, I would like to thank HASFIYA KP for their invaluable guidance, insightful feedback, and constant encouragement, which were instrumental in the successful completion of this project. Their expertise helped me navigate the various challenges involved in data preprocessing, machine learning model training, and deployment.

I would also like to thank my family and friends for their unwavering support and understanding during the course of this project. Their encouragement allowed me to remain focused and committed throughout the development process. Additionally, I am grateful to my peers and colleagues for their collaboration, shared insights, and constructive suggestions, which significantly improved the quality of the project.

Finally, I would like to acknowledge the various online resources, open-source tools, and libraries that made this project possible. The availability of Python libraries such as Pandas, Scikit-learn, and Matplotlib was essential for conducting data analysis, building machine learning models, and visualizing results. Without the support and resources from the open-source community and online tutorials, the successful completion of this project would not have been possible. Thank you to all those who contributed to the realization of this project.

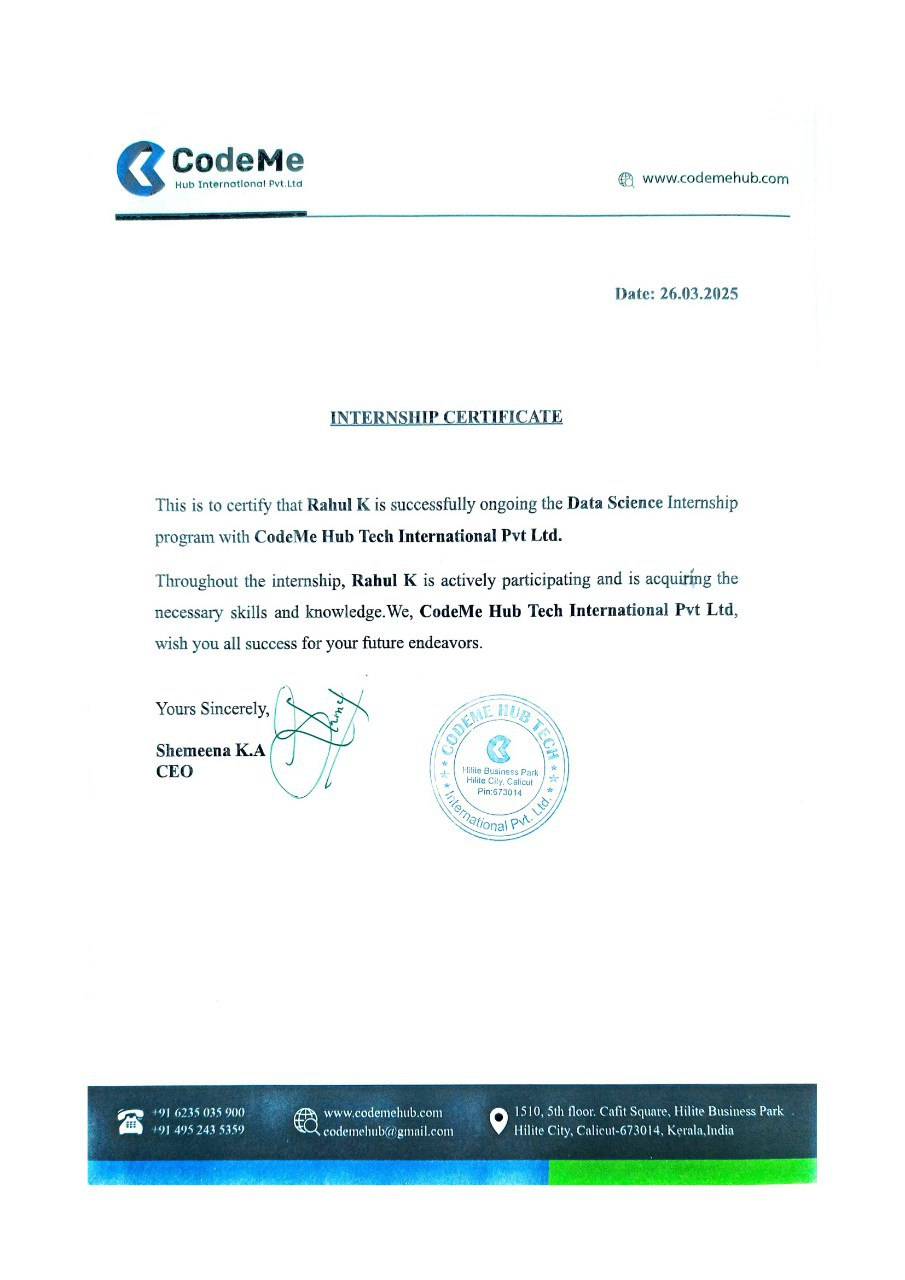
**DECLARATION**

I, **RAHUL K**, student of VI th semester B.voc Data science & Analytics a M.E.S Kalladi College Mannarkkad, Palakkad, hereby declare that the PROJECT work entitled “DYNAMIC PRICING SYSTEM” has been independently carried out by me under the supervision of **SHABNA**, Head of department of B.voc data science & analytics, and the coordinator **HASFIYA KP**, Assistant Professor, submitted in partial fulfillment of the course requirement for the award of degree **Bachelor of vocation in Data Science & Analytics** of Calicut University, during the year 2025. I further declare that the report has not been submitted to any other University for the award of any other degree.

**PLACE: MANNARKKAD Date: 20 APRIL 2025**

**STUDENT NAME: RAHUL K REG.NO:KIAWBOE024**

**CERTIFICATE**



**Abstract**

This project presents a Dynamic Pricing System for Cab Rides developed using a Random Forest Regression model. The objective of this system is to simulate real-time price adjustments in response to fluctuating supply-demand conditions, customer behavior, and ride-specific parameters. While the dataset used is static, consisting of 1,000 curated historical cab booking records, the system is designed to accept real-time-like inputs from users—such as the number of active riders and drivers, vehicle type, booking time, ride duration, and customer loyalty status—to dynamically compute a fair and optimized ride fare.

**Key features influencing the model include:**

**Number of Riders and Drivers** – serving as a proxy for current demand and supply.

**Location Category** – Urban, Suburban, or Rural classifications to reflect local pricing trends.

**Customer Loyalty Status** – recognizing frequent or valued customers.

**Time of Booking** – capturing temporal effects like peak hours or late-night surges.

**Vehicle Type and Expected Ride Duration** – to factor in ride quality and fuel/time consumption.

The Random Forest Regressor was selected for its ability to manage both numerical and categorical data efficiently and produce robust, non-linear predictions. The trained model is deployed using Streamlit, enabling users to input hypothetical or observed ride conditions via a web interface and instantly receive data-driven pricing estimates.

Although the system does not currently integrate with a live data feed, its architecture and interface are built to simulate a real-time environment, making it an ideal prototype for future integration with live operational data from ride-hailing platforms. This solution can assist companies in enhancing customer satisfaction, balancing market dynamics, and boosting profitability by offering intelligent, flexible pricing strategies.

**CONTENTS**

| **S. No.** | **Content** | **Page No.** |
| --- | --- | --- |
|  |  |  |
| 1 | Introduction | 6 |
| 2 | Problem Statement | 7 |
| 3 | Objective | 9 |
| 4 | Literature Review / Existing Systems | 10 |
| 5 | Proposed System | 13 |
| 6 | System Architecture | 15 |
| 7 | Methodology | 17 |
| 8 | Data Preprocessing | 18 |
| 9 | Model Training | 34 |
| 10 | Model Evaluation | 39 |
| 11 | Implementation | 40 |
| 12 | Technology Stack | 43 |
| 13 | Conclusion and Future Scope | 45 |
| 14 | References | 48 |

**INTRODUCTION**

In today’s fast-paced, on-demand economy, businesses face a critical challenge: how to dynamically price their services in a way that reflects current market conditions. From ride-hailing platforms to airline ticketing and even hotel bookings, **any system governed by the principles of demand and supply** requires an intelligent and adaptable pricing strategy.

Traditional static pricing models often fall short, as they cannot react to **real-time variables** such as sudden spikes in demand, temporary drops in supply, customer loyalty behavior, or operational costs. This mismatch can lead to lost revenue opportunities, dissatisfied customers, and operational inefficiencies.

To tackle this issue, we propose a **machine learning-powered Dynamic Pricing System**, specifically trained on a cab booking dataset. While this system is currently applied to the cab ride scenario, it is designed as a **prototype for a generalized real-time prediction model** that could be extended to other industries where dynamic pricing is critical.

Our model leverages key features such as:

* **Number of Riders (demand)**
* **Number of Drivers (supply)**
* **Vehicle Type and Expected Ride Duration (service-specific factors)**
* **Time of Booking**

Using these variables, the system generates **intelligent, data-driven price suggestions** that better reflect dynamic market scenarios.

Though the dataset used is historical and static, the deployed system accepts **user-defined, real-time-like inputs** and processes them through a trained **Random Forest Regressor**. The entire solution is wrapped in a **Streamlit web app**, providing an interactive and user-friendly interface for experimentation and real-world simulation.

This project lays the foundation for future integration with live operational data, demonstrating the potential of ML-based dynamic pricing across various industries where agile pricing can enhance competitiveness and profitability

**Problem Statement**

Ride-hailing companies operate in an environment where pricing directly impacts **profitability, operational efficiency, and market competitiveness**. Traditional pricing models—whether fixed-rate or basic surge mechanisms—often fail to reflect the dynamic nature of the market. They typically do not account for real-time operational factors such as **demand fluctuations, driver availability, ride duration, or regional characteristics**, leading to **suboptimal pricing** decisions.

Even modern dynamic pricing engines employed by major platforms can function as **opaque black-box systems**, offering little control or interpretability to business stakeholders. These systems often generalize behavior without giving companies the ability to simulate, test, or adapt pricing strategies based on **specific operational variables**.

This results in challenges such as:

* **Revenue leakage** due to underpricing during high-demand periods.
* **Loss of competitiveness** from overpricing during low-demand scenarios.
* **Inefficient driver utilization** due to non-aligned pricing signals.
* **Limited insights** into how specific factors influence pricing outcomes.

To address this, we developed a **Dynamic Pricing System prototype** tailored specifically for **business use cases**, not end users. Built using a **Random Forest Regression model** and trained on a curated cab booking dataset of 1,000 records, the system leverages critical real-world features including:

* **Number of Riders and Drivers** (real-time demand-supply simulation)
* **Vehicle Type and Expected Ride Duration**
* **Time of Booking** (to capture temporal pricing patterns)
* **Location Category** (Urban/Suburban/Rural)

The system is deployed as an **interactive Streamlit application**—not for customer-facing use—but to enable **internal pricing teams, analysts, and product managers** to experiment with real-time-like inputs and evaluate how pricing would respond under different operational scenarios.

What distinguishes this system is its **business-centric design**: it acts as a **strategic simulation tool**, allowing companies to **test pricing policies**, analyze feature impact, and make **data-informed adjustments** to existing pricing engines. It sets the stage for integrating live data pipelines, making it a scalable prototype for deploying **real-time, AI-powered pricing systems** across not only the ride-hailing industry but any sector driven by demand-supply economics.

**Objective**

The objective of this project is to **design and implement a machine learning-based dynamic pricing model** specifically tailored for **business use**, with the goal of helping **ride-hailing platforms and transportation businesses** optimize their pricing strategies. The system is designed to dynamically predict cab ride prices based on **real-time supply-demand conditions**, **ride-specific features**, and **booking context**, without relying on customer loyalty or historical ride data.

Key features influencing pricing predictions include:

* **Number of Active Riders and Drivers** – to simulate real-time demand and supply.
* **Location Category** (Urban/Suburban/Rural) – to account for region-specific demand and pricing trends.
* **Time of Booking** – to capture temporal fluctuations in ride demand (e.g., peak hours, late-night surges).
* **Vehicle Type and Expected Ride Duration** – to estimate service costs and optimize ride pricing.

The model is trained using a comprehensive dataset of **1,000 historical cab booking records** and deployed as an **internal web application** using **Streamlit**. This web application serves as a **business-facing tool** that enables **pricing teams, data analysts, and product managers** to input real-time-like conditions, simulate different ride scenarios, and evaluate how pricing might evolve under various demand-supply situations.

The solution is intended to serve as a **strategic decision support tool**, empowering businesses to:

* **Make data-driven adjustments** to dynamic pricing algorithms.
* **Evaluate the impact of operational variables** on pricing outcomes.
* **Optimize revenue and competitiveness** by balancing price sensitivity with market conditions.

By focusing on **real-time operational features** and **business logic**, this system aims to provide actionable insights that help businesses implement intelligent, context-aware pricing models, which can be integrated into **live data pipelines** in the future for full-fledged real-time dynamic pricing.

**Literature Review / Existing Systems**

Dynamic pricing is an increasingly common approach in industries where demand and supply fluctuate in real time, such as **ride-hailing**, **hotel bookings**, and **airline ticketing**. The ability to adjust prices based on real-time conditions allows companies to **maximize revenue**, **optimize resource allocation**, and **improve customer satisfaction**. This section reviews existing dynamic pricing systems and highlights their limitations, with a focus on how this project improves upon current methods.

**1. Traditional Pricing Models in Ride-Hailing**

Traditional pricing in the ride-hailing industry often follows **fixed pricing models** or **basic surge pricing**, which apply a flat rate or a multiplier to the fare during periods of high demand. These methods are relatively simple and rely on predefined rules to adjust prices. However, such models are often **inflexible** and fail to account for the full range of factors influencing both demand and supply, leading to suboptimal pricing in many scenarios.

* **Surge Pricing** (Uber, Lyft): Surge pricing adjusts rates during periods of high demand, often seen during peak hours or in busy areas. While this method responds to demand, it does not account for finer variables like **ride duration, customer preferences, or vehicle type**, and may result in **customer dissatisfaction** when fares become too high.
* **Fixed Pricing** (Traditional Taxi Services): In contrast, some traditional cab services still rely on fixed pricing models, setting fares based on distance and time without considering real-time market fluctuations. This approach can result in **overpricing during low-demand periods** and **underpricing when demand spikes**.

**2. Machine Learning Approaches to Dynamic Pricing**

With the rise of machine learning, more sophisticated dynamic pricing models have been proposed. These models attempt to predict optimal pricing by incorporating more granular data inputs, enabling platforms to adjust prices more intelligently.

* **Reinforcement Learning (RL)**: One of the most promising areas in dynamic pricing is **Reinforcement Learning**, where models continuously learn from feedback and adjust pricing strategies based on observed outcomes. For example, companies like Uber and Lyft are exploring RL techniques for pricing, where the algorithm makes decisions based on customer behavior, ride history, and demand-supply patterns (Zhao et al., 2017). RL allows for the continuous adaptation of pricing strategies, but it is complex to implement and may require large-scale, real-time data streams.
* **Gradient Boosting and Random Forests**: In contrast, more interpretable models like **Random Forest** and **Gradient Boosting** are also widely used for dynamic pricing. These models can handle a mix of categorical and numerical data and can capture non-linear relationships between input features. Many of these systems use factors such as **time of day**, **location**, **ride duration**, and **traffic conditions** to make pricing predictions (Chien et al., 2012).

**3. Existing Commercial Dynamic Pricing Systems**

A number of **ride-hailing companies** and **third-party vendors** have deployed dynamic pricing algorithms, but their systems are often proprietary and lack transparency, making it difficult for businesses to adapt the models to their unique needs.

* **Uber's Surge Pricing**: Uber's surge pricing system adjusts rates based on demand and availability but can lead to unpredictable pricing, especially during extreme conditions. Additionally, Uber’s pricing model tends to increase prices significantly during periods of high demand, leading to customer complaints regarding **price fairness** and **lack of transparency**.
* **Lyft’s Prime Time**: Lyft uses a similar **Prime Time pricing** system, which activates when demand exceeds supply, adjusting fares accordingly. While the system adapts to local conditions, it still uses simplistic rules that do not incorporate a broader set of features, such as customer-specific factors or vehicle type.

**4. Limitations of Existing Systems**

While current systems have made strides in dynamic pricing, they still face several challenges:

* **Lack of Transparency and Control**: Many commercial systems, such as those used by Uber and Lyft, operate as “black boxes,” leaving businesses with limited ability to understand or control how pricing decisions are made.
* **Limited Factors Considered**: While many systems use demand and supply data, they fail to incorporate other crucial features like **ride duration**, **location category**, or **time of day**, which can lead to overpricing or underpricing.
* **Customer Satisfaction**: Dynamic pricing models often prioritize revenue maximization, sometimes at the expense of customer satisfaction, with price fluctuations being seen as unpredictable and unfair.

**5. Key Gaps and Innovations**

While machine learning techniques like **Random Forests** and **Gradient Boosting** have shown promise in improving dynamic pricing, existing models often suffer from a **lack of interpretability**, making them difficult to adapt for businesses. Additionally, many of the models are not easily integrated into existing **business workflows** or **real-time systems**.

The approach presented in this project stands out by addressing the following gaps:

* **Business-Focused Tool**: Unlike customer-facing systems that focus on maximizing business revenue, this system is designed to help **businesses make informed pricing decisions** by offering a transparent and **interpretable model** that can be adjusted based on operational input.
* **Real-Time-Like Simulation**: The system uses a **static dataset** for training but is designed to accept **real-time-like inputs**, allowing businesses to simulate various market conditions and evaluate different pricing strategies without relying on live data feeds.
* **Comprehensive Operational Features**: The model takes into account critical features like **demand-supply balance**, **vehicle type**, **booking time**, and **ride duration**, providing a much more **holistic and actionable pricing model** than typical surge-based systems.

**6. Future Directions**

The system developed in this project can be extended to a **real-time dynamic pricing engine** by integrating live data streams from ride-hailing platforms. Future improvements could include the use of more advanced machine learning algorithms, such as **deep learning** or **reinforcement learning**, to further refine price predictions and account for even more complex patterns of demand and supply.

**Proposed System**

The proposed system introduces a **business-focused dynamic pricing model** tailored for ride-hailing platforms, designed to enhance profitability by aligning ride pricing with real-time-like operational conditions. Built using a **Random Forest Regressor** and deployed via **Streamlit**, this system allows pricing strategists to input key ride features and receive intelligent, data-driven price suggestions—complete with indicators of profitability.

**Core Functionalities**

1. **Input of Operational Ride Features**  
   Businesses can simulate dynamic pricing by inputting critical real-time-like features such as:
   * **Number of Riders (Demand)**
   * **Number of Drivers (Supply)**
   * **Time of Booking** (Morning, Evening, Night)
   * **Vehicle Type** (Economy, Premium)
   * **Expected Ride Duration**

These inputs reflect the operational environment and allow businesses to test pricing strategies across varied market conditions.

1. **Intelligent Price Prediction via Random Forest Regressor**  
   The system uses a machine learning model trained on historical cab ride data to predict a fair yet optimal price for any ride configuration. By analyzing patterns in past data, the model learns how different combinations of inputs influence ride cost, enabling more accurate, context-sensitive pricing.
2. **Profitability**  
   A key business utility feature is the system’s ability to **flag whether a ride is profitable**. This allows pricing managers to quickly assess if the suggested fare covers operational costs and aligns with revenue goals. This flag can help in adjusting strategies to prevent underpricing in low-margin conditions.
3. **Demand and Supply Multipliers for Enhanced Price Contextualization**  
   To further align pricing with market conditions, the system incorporates **demand and supply multipliers**:
   * **Demand Multiplier**: Quantifies rider demand intensity by comparing the current number of riders to historical percentiles. Higher demand increases the price suggestion proportionally.
   * **Supply Multiplier**: Assesses driver availability in the current context. Lower driver availability leads to a higher multiplier, increasing the predicted price to reflect supply scarcity.

These multipliers allow the pricing model to **dynamically scale prices** based on real-time fluctuations in market behavior, bringing a nuanced and mathematically backed adjustment layer to traditional pricing models.

1. **Deployed as an Interactive Streamlit Web Application**  
   The system is deployed using **Streamlit**, making it accessible and interactive for business analysts and pricing teams. Users can enter different values, simulate demand/supply changes, and instantly view pricing predictions and profitability indicators—without needing technical expertise.

**Unique Aspects**

* **Simulation-Driven Decision Making**: While based on static historical data, the system is designed for simulating **real-time market scenarios**, allowing businesses to test, forecast, and adjust pricing models in a controlled, strategic way.
* **Business-Centric Utility**: Unlike customer-facing tools, this system is meant for **internal use by pricing strategists**, providing them with transparency, interpretability, and control.
* **Modular Pricing Strategy**: The use of **demand and supply multipliers** ensures the pricing model remains sensitive to external fluctuations while maintaining a consistent pricing logic.

**Future Potential**

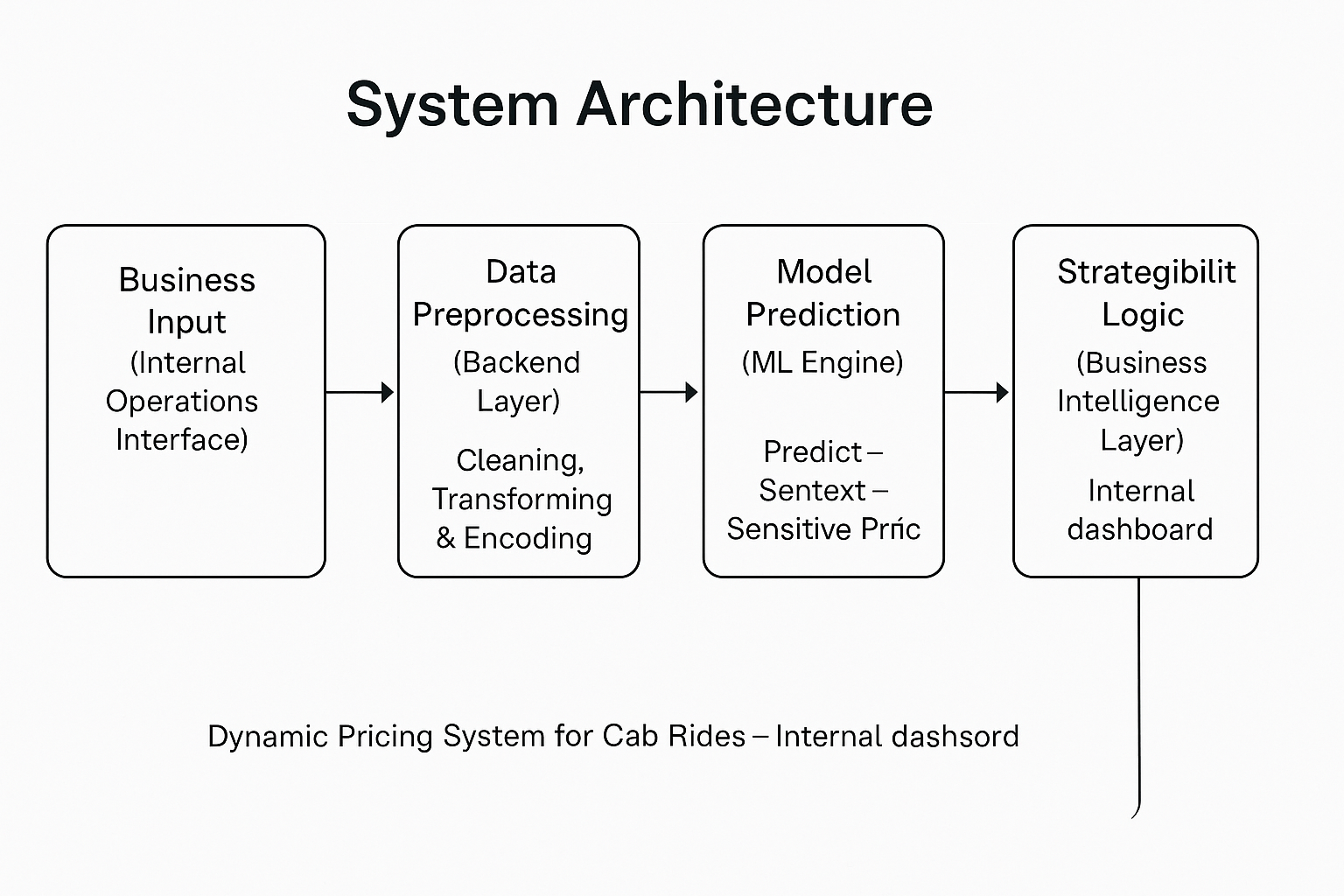
In future iterations, this prototype could evolve into a real-time pricing engine by integrating **live operational data**, APIs, and more advanced AI techniques like **reinforcement learning** for continual learning and optimization

**System Architecture**

The architecture of the proposed **Dynamic Pricing System for Cab Rides** is designed to support **business decision-making** rather than end-user interaction. It provides a backend tool for pricing strategists and operations teams to **simulate ride scenarios, predict prices**, and assess profitability based on internal operational conditions. The system is structured in a modular pipeline to enable **transparent, data-driven pricing strategies**.

**Architectural Workflow**

**Business Input → Data Preprocessing → ML Prediction → Profitability Logic → Strategic Output**



1. **Business Input (Internal Operations Interface)**
   * Internal teams input operational conditions like rider volume, driver availability, booking context, and expected duration.
   * This serves as a **controlled simulation environment** where various pricing scenarios can be tested under different supply-demand conditions.
2. **Data Preprocessing (Backend Layer)**
   * Input data is cleaned, transformed, and encoded to align with the format expected by the trained machine learning model.
   * Includes the calculation of **demand and supply multipliers** to dynamically adjust the pricing context based on percentile-based thresholds, enhancing the sensitivity of the model to realistic market variations.
3. **Model Prediction (ML Engine)**
   * A **Random Forest Regressor**, trained on historical cab ride data, processes the input features to predict a context-aware ride price.
   * This model leverages patterns across multiple factors such as time of booking, vehicle type, location, and ride duration to generate a robust and business-aligned price estimate.
4. **Profitability Logic (Business Intelligence Layer)**
   * Beyond pricing, the system applies **custom business logic** to flag whether a predicted price is **profitable** under given conditions.
   * Profitability checks are based on internal thresholds, ride characteristics, and vehicle categories—making it easier to avoid low-margin or loss-making rides.
5. **Strategic Output (Insight Delivery)**
   * The final output is presented within an internal-facing **Streamlit dashboard**, displaying:
     + Predicted Ride Price
     + Profitability Indicator
     + Input Condition Summary for Traceability
   * This enables **quick evaluation of pricing strategies**, fine-tuning of business models, and strategic decision support.

**Component Overview**

| **Component** | **Description** |
| --- | --- |
| **Input Layer** | Internal input module for simulating ride conditions |
| **Backend Layer** | Data preprocessing, multiplier calculations, and feature transformation |
| **ML Engine** | Random Forest Regressor trained to predict context-sensitive pricing |
| **Logic Layer** | Embedded business rules for profitability evaluation |
| **Insight Layer** | Internal dashboard for displaying pricing suggestions and strategic flags |

**Key Highlights**

* **Not User-Facing**: Designed exclusively for **business teams**, not end customers.
* **Simulation-Ready**: Can model hypothetical or forecasted conditions to support **scenario planning**.
* **Profitability-Driven**: Goes beyond fare prediction to align pricing with **business goals and margins**.
* **Modular & Scalable**: Easily adaptable for other sectors where dynamic pricing is essential (e.g., logistics, retail, events)

**Methodology**

**Data Understanding**

The dataset used in this project serves as **prototype data**, designed to simulate real-world ride booking conditions in a dynamic, on-demand transportation environment. It consists of **1,000 ride records**, each representing individual booking scenarios influenced by various operational and contextual features.

**✅ Key Features in the Dataset:**

| **Feature Name** | **Description** |
| --- | --- |
| Number\_of\_Riders | Total number of ride requests in a specific time/location window. |
| Number\_of\_Drivers | Total number of available drivers in the area. |
| Location\_Category | Categorized as Urban, Suburban, or Rural, affecting traffic and availability. |
| Customer\_Loyalty\_Status | Status tier of the customer (e.g., Silver, Gold). Not used in final model. |
| Number\_of\_Past\_Rides | Count of previous rides by the customer. Also not included in final model. |
| Average\_Ratings | Historical customer rating average. |
| Time\_of\_Booking | Time slot of the ride (e.g., Morning, Afternoon, Evening, Night). |
| Vehicle\_Type | Type of vehicle selected – Economy or Premium. |
| Expected\_Ride\_Duration | Estimated time duration of the ride in minutes. |
| Historical\_Cost\_of\_Ride | Actual cost of the ride (used as the target variable for model training). |

**📌 Additional Notes:**

* The dataset does **not contain missing values** and is structurally consistent, allowing for direct preprocessing and modeling.
* This dataset was synthesized to reflect **demand-supply dynamics**, ride duration, and vehicle selection trends commonly observed in real-time cab booking ecosystems.
* It is intended to serve as a **functional prototype** to test dynamic pricing logic before scaling to live deployment with real-time data pipelines.

**Data Preprocessing**

Efficient preprocessing was performed to prepare the dataset for robust and meaningful machine learning predictions. The preprocessing phase focused on cleaning, encoding, and selecting features that directly impact pricing from a business perspective.

**Handling Missing Values**

* The dataset was verified to be **complete** with no missing values across any of the 10 features.
* As a result, **no imputation or removal** of data was necessary, ensuring the dataset retained its original integrity.

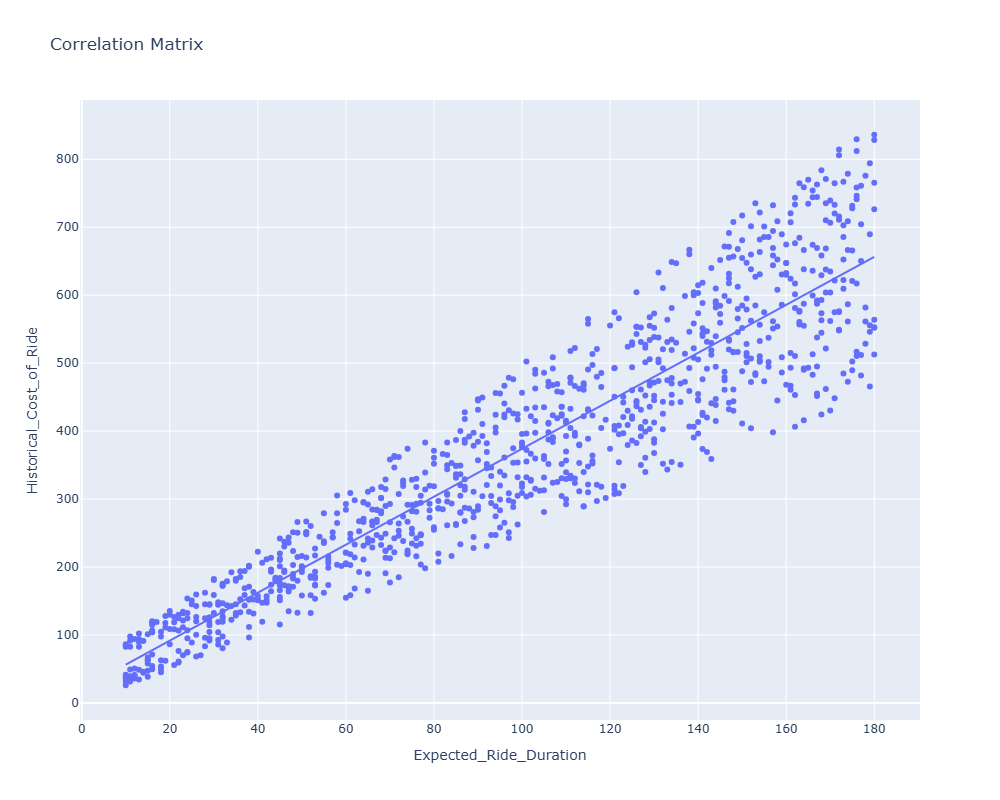
**Data Cleaning**

* Columns were reviewed for datatype consistency.
* Redundant or low-impact features such as Customer\_Loyalty\_Status and Number\_of\_Past\_Rides were **excluded** from the model pipeline after correlation and feature importance analysis showed they had negligible influence on pricing outcomes.

**Exploratory Analysis Used for Feature Assessment**

Exploratory Data Analysis (EDA) played a critical role in identifying the most influential features for ride pricing, ensuring that the final model aligns with real-world business goals such as profitability, demand-supply sensitivity, and operational efficiency.

**Expected Ride Duration vs. Historical Cost of Ride**

****

**What the Chart Shows:**

This scatter plot visualizes the relationship between:

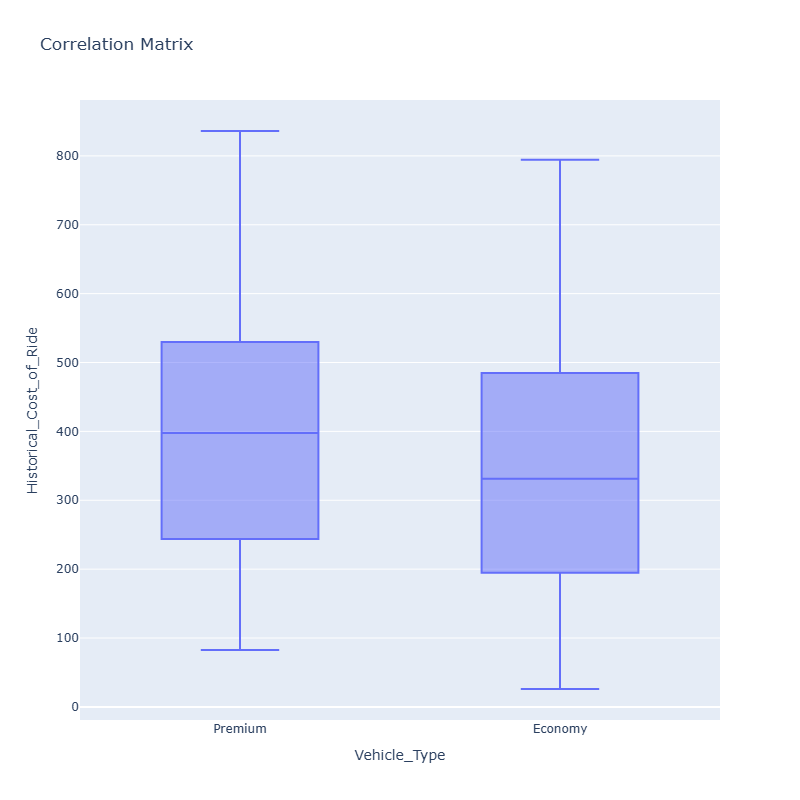
* **X-axis:** Expected\_Ride\_Duration (in minutes)
* **Y-axis:** Historical\_Cost\_of\_Ride (in currency units)

It includes a **trendline** fitted using **Ordinary Least Squares (OLS)** regression to observe the linear trend.

**Key Observations and Insights:**

1. **Strong Positive Correlation:**
   * The plot clearly demonstrates a **strong positive linear correlation** between ride duration and historical cost.
   * As the expected duration of a ride increases, the cost also increases proportionally.
   * This validates that duration is a **highly influential feature** in determining pricing.
2. **Tight Clustering Around the Trendline:**
   * Data points are tightly clustered around the regression line, indicating **low variance** and a consistent pricing pattern.
   * This makes Expected\_Ride\_Duration a **reliable predictor** for pricing models like Random Forest or Linear Regression.
3. **Outliers and Spread:**
   * A few data points deviate from the trend (above and below the line), possibly due to:
     + Surge pricing scenarios
     + Discounts applied
     + Varying vehicle types or customer loyalty impacts
   * This further supports the need for **multi-feature** modeling instead of a basic duration-cost ratio.
4. **Linear Pricing Behavior:**
   * The trendline supports the idea that the platform historically follows a **linear pricing model** based on time.
   * However, actual pricing may be improved using **non-linear models** like Random Forest, which can account for other interacting variables.
5. **Business Relevance:**
   * The graph justifies using historical ride duration and cost as **core inputs** for dynamic pricing.
   * Businesses can rely on this correlation to forecast baseline pricing and then adjust based on supply-demand and customer variables.

**Historical Cost of Ride by Vehicle Type**

****

**What the Chart Shows:**

This box plot visualizes the **distribution of historical ride costs** across two vehicle categories:

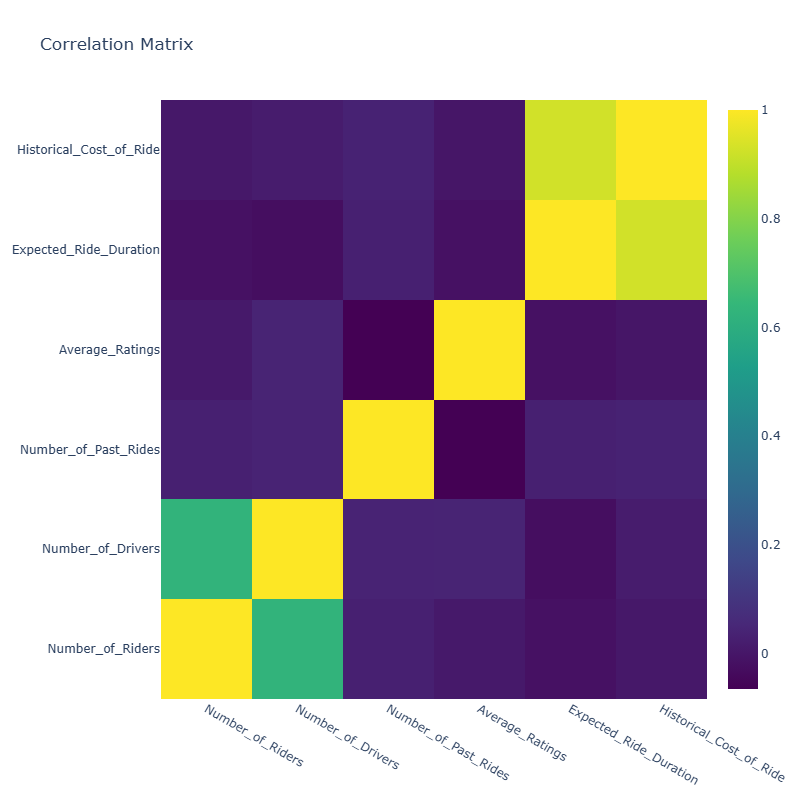
* **Premium**
* **Economy**

It highlights the **median**, **interquartile range (IQR)**, and **potential outliers** for each category.

**Key Observations and Insights:**

1. **Cost Disparity Between Vehicle Types:**
   * As expected, **Premium rides** are generally more expensive than **Economy rides**.
   * The **median** cost for Premium is significantly higher, indicating a consistent price premium.
2. **Wider Cost Range for Premium:**
   * Premium rides show a **wider interquartile range (IQR)** and more variability in pricing.
   * This suggests that Premium ride costs fluctuate more based on ride conditions, possibly influenced by duration, time of day, or customer loyalty.
3. **Overlapping Whiskers:**
   * There is **some overlap in the cost ranges** between Economy and Premium, especially at the higher end of Economy and the lower end of Premium.
   * This could occur when:
     + Economy rides are unusually long (raising the cost),
     + Premium rides are short but in high-demand areas.
4. **Presence of Outliers:**
   * Both vehicle types exhibit outliers, but they are **more prominent in Premium**, likely due to:
     + Long-distance luxury trips
     + Surge pricing events
     + Added services (e.g., waiting time, special routes)
5. **Business Relevance:**
   * Differentiating pricing strategies by vehicle type is validated.
   * Models should treat Vehicle\_Type as a **categorical feature** since it significantly affects pricing behavior.

**Correlation Matrix (Heatmap)**

****

**What This Chart Shows:**

This heatmap displays the **Pearson correlation coefficients** between all numerical variables in the dataset. The closer a value is to **1 or -1**, the stronger the linear relationship between the two variables.

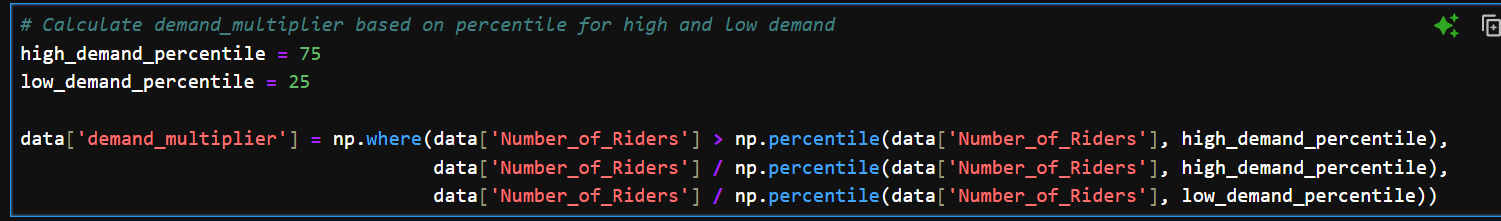
**Key Insights from the Correlation Matrix:**

1. **Strong Positive Correlation Between Duration and Cost:**
   * Expected\_Ride\_Duration and Historical\_Cost\_of\_Ride show a **very high correlation (~0.95+)**.
   * This makes sense—longer rides tend to cost more.
   * *Implication:* Duration is a **key driver** of pricing in the model.
2. **High Correlation Between Rider and Driver Counts (~0.65):**
   * Number\_of\_Riders and Number\_of\_Drivers are strongly correlated.
   * This reflects **supply-demand balancing** within the dataset, which might be influenced by system allocation rules or surge pricing logic.
   * *Implication:* This ratio can help define **demand pressure**, a useful engineered feature.
3. **Low or Negligible Correlation in Other Features:**
   * Variables like Average\_Ratings and Number\_of\_Past\_Rides show **very weak correlation** with cost and duration.
   * *Implication:* These may still hold **non-linear or categorical significance** even if Pearson correlation is low.
4. **Multicollinearity Watch:**
   * The **strong correlation between duration and cost** might introduce **multicollinearity** if both are used as predictors in the same model.
   * *Suggestion:* Consider modeling with only one of them or using **feature importance metrics** to evaluate their necessity.

**Feature Engineering**

**Demand Multiplier**

To capture the real-time market pressure, we introduced a new feature named demand\_multiplier. This variable is engineered using the number of riders in each instance, scaled based on predefined demand percentiles.



**What’s Happening Here:**

**1. Define Percentiles**

high\_demand\_percentile = 75

low\_demand\_percentile = 25

* This sets **75th percentile** as the threshold for **high demand**.
* And **25th percentile** as the threshold for **low demand**.
* These percentiles help categorize the data into high and low demand scenarios.

**2. np.percentile(...)**

np.percentile(data['Number\_of\_Riders'], high\_demand\_percentile)

* Calculates the value below which 75% of the rider count data falls.
* Suppose that value is 120, then:
  + If Number\_of\_Riders is more than 120, it's considered high demand.
  + If less than that, it's in normal/low demand.

**3. np.where(...)**

data['demand\_multiplier'] = np.where(condition, value\_if\_true, value\_if\_false)

* A **vectorized if-else** operation from NumPy.
* It checks the condition across all rows in the dataframe and assigns a new value based on it.

**Condition Logic Explained:**

data['Number\_of\_Riders'] > np.percentile(data['Number\_of\_Riders'], 75)

* **If True (High Demand):**  
  Divide the Number\_of\_Riders by the **75th percentile value** (e.g., 150 / 120 = 1.25) → Higher than 1
* **If False (Normal/Low Demand):**  
  Divide the Number\_of\_Riders by the **25th percentile value** (e.g., 60 / 30 = 2.0) → Still reflects pressure but on a lower scale

So the value of demand\_multiplier becomes:

* 1 when there's **high demand** (pressure is building)
* Close to or <1 when it's **low or average demand**

**Example:**

Assume:

* 75th percentile = 120 riders
* 25th percentile = 40 riders

Now for a few data points:

| **Number\_of\_Riders** | **demand\_multiplier** |
| --- | --- |
| 150 | 150 / 120 = **1.25** (high demand) |
| 30 | 30 / 40 = **0.75** (low demand) |
| 100 | 100 / 40 = **2.5** (mid-range but below 75th percentile) |

**Why It’s Useful:**

* Helps capture **non-linear effects** of demand on price.
* Adds a **context-aware feature** into the model.
* Can improve model interpretability and accuracy.

**Why This Matters:**

* **Capture Market Conditions:**

This feature reflects **surge-like pricing behavior**. When rider demand exceeds the 75th percentile (high demand), the multiplier increases relative to that threshold. Similarly, in low-demand situations (below 25th percentile), the multiplier adjusts downward.

* **Dynamic Sensitivity:**

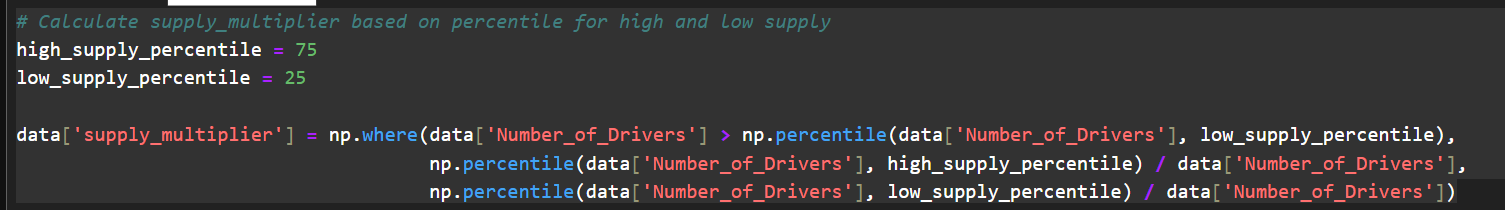
Unlike a binary high/low demand flag, the demand\_multiplier provides a **continuous value** that better reflects subtle market shifts.

* **Improves Model Accuracy:**

Models can now learn not just that demand is high or low, but **how much** higher or lower it is relative to normal conditions.

**Supply multiplier**

A new feature supply multiplier was created to reflect fluctuations in driver availability. It adjusts according to the number of available drivers at any given time and their percentile distribution in the dataset.

  
To create a new feature called supply\_multiplier that reflects how the number of available drivers affects ride dynamics. It uses percentile-based thresholds to scale the value, making it sensitive to high and low supply situations.

**Code Breakdown:**

# Set percentile thresholds

high\_supply\_percentile = 75

low\_supply\_percentile = 25

* These define the 75th and 25th percentiles of the number of drivers.
* They act as thresholds to determine what is considered **high supply** and **low supply**.

data['supply\_multiplier'] = np.where(

data['Number\_of\_Drivers'] > np.percentile(data['Number\_of\_Drivers'], low\_supply\_percentile),

np.percentile(data['Number\_of\_Drivers'], high\_supply\_percentile) / data['Number\_of\_Drivers'],

np.percentile(data['Number\_of\_Drivers'], low\_supply\_percentile) / data['Number\_of\_Drivers']

)

* np.where(condition, value\_if\_true, value\_if\_false) is a vectorized if-else condition.
* data['Number\_of\_Drivers'] > np.percentile(data['Number\_of\_Drivers'], low\_supply\_percentile)

This checks if the current row's number of drivers is **greater than the 25th percentile** (i.e., not extremely low supply).

**Logic:**

* If the number of drivers is **above the low threshold (25th percentile)**:

python

CopyEdit

np.percentile(data['Number\_of\_Drivers'], 75th percentile) / data['Number\_of\_Drivers']

→ The higher the drivers, the **lower** the multiplier (inverse relationship), which means **more supply reduces prices**.

* If the number of drivers is **below or equal to the 25th percentile** (indicating **low supply**):

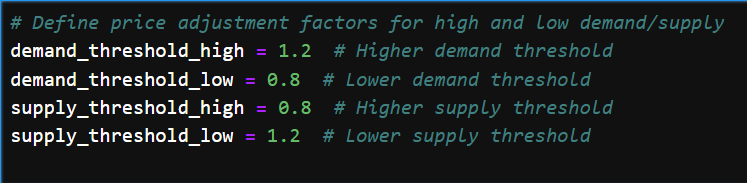
np.percentile(data['Number\_of\_Drivers'], 25th percentile) / data['Number\_of\_Drivers']

→ Since drivers are scarce, this creates a **higher multiplier**, simulating price increase due to **supply shortage**.

**Summary:**

* The feature supply\_multiplier is designed to **scale inversely** with the number of drivers.
* **More drivers → Lower multiplier**, indicating less market tension.
* **Fewer drivers → Higher multiplier**, capturing scarcity and potential price surge effects.

**Define price adjustment factors for high and low demand/supply**



 **demand threshold high = 1.2**

* When your computed demand\_multiplier exceeds 1.2, you’re in a “high‑demand” scenario (surge‑like conditions).
* You might, for example, multiply the base fare by 1.2 (or more) to reflect extra demand pressure.

 **demand threshold low = 0.8**

* If demand\_multiplier falls below 0.8, demand is unusually weak.
* You could apply a discount factor (e.g., multiply base fare by 0.8) to incentivize ridership.

 **supply threshold high = 0.8**

* A supply\_multiplier under 0.8 indicates **very high driver availability** (supply surplus).
* In that case, you might reduce prices (e.g., ×0.8) since there’s plenty of supply.

 **supply threshold low = 1.2**

* A supply\_multiplier over 1.2 means driver availability is scarce.
* You’d raise prices (×1.2) to reflect the supply shortage.

**Price Adjustment Thresholds**

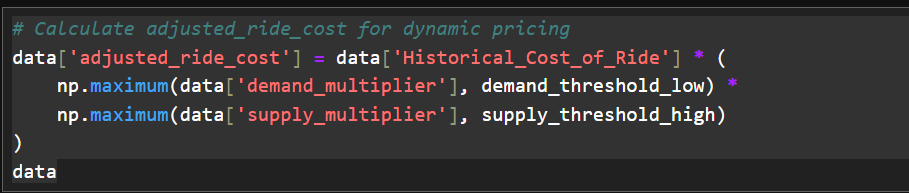
To operationalize our demand\_multiplier and supply\_multiplier, we defined four thresholds:

demand\_threshold\_high = 1.2 and demand\_threshold\_low = 0.8 specify when to apply surge or discount based on rider elasticity.

supply\_threshold\_high = 0.8 and supply\_threshold\_low = 1.2 trigger price decreases or increases when driver availability is abundant or scarce.

These thresholds allow the model to dynamically adjust the base predicted fare, ensuring responsiveness to real‑time market conditions.

**Calculate adjusted\_ride\_cost for dynamic pricing**

****

**Objective:**

To compute the **adjusted ride cost** using the Historical\_Cost\_of\_Ride as a baseline, adjusted by the **real-time demand** and **supply multipliers**, ensuring that the final price is dynamically adjusted based on current market conditions.

* **data['Historical\_Cost\_of\_Ride']**

This represents the baseline or historical cost of the ride. It’s the starting point for the adjusted price calculation.

* **np.maximum(data['demand\_multiplier'], demand\_threshold\_low)**

This part ensures that the **demand multiplier** doesn't fall below the demand\_threshold\_low (0.8).

* + If the demand multiplier is below 0.8, it will be **clipped to 0.8** (ensuring the lowest adjustment is a reduction, not a drastic drop).
  + Otherwise, it retains its value.
* **np.maximum(data['supply\_multiplier'], supply\_threshold\_high)**

Similarly, this ensures that the **supply multiplier** doesn't fall below the supply\_threshold\_high (0.8).

* + If the supply multiplier is below 0.8, it will be **clipped to 0.8**, meaning that even if supply is high (which would typically reduce prices), it can't lower the price too much.
  + Otherwise, it remains as is.
* **Final Formula:**

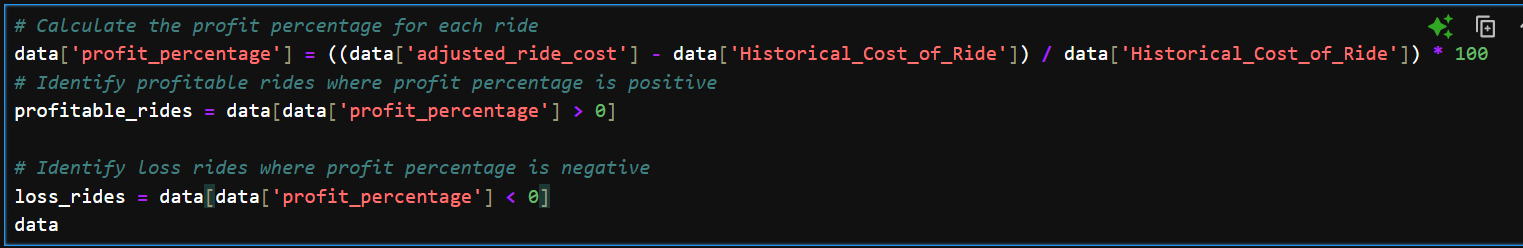
The **adjusted\_ride\_cost** is the product of the historical ride cost and the adjusted demand and supply factors. This gives the final dynamic price, incorporating both market demand and supply conditions.

**How the Formula Works:**

1. **When demand is high** (e.g., demand multiplier > 1.2):
   * The ride cost will increase because the **demand multiplier** is scaled up.
2. **When supply is low** (e.g., supply multiplier > 1.2):
   * The ride cost will increase as well, as there are fewer drivers to meet the demand.
3. **When both demand and supply are in balance** (e.g., demand multiplier between 0.8 and 1.2, and supply multiplier around 1):
   * The price will be adjusted slightly, but it will stay relatively close to the historical cost.
4. **When supply is high and demand is low** (e.g., demand multiplier < 0.8 and supply multiplier < 0.8):
   * The price will decrease as both factors push the price down.

**Profit percentage for each ride**

**Objective:**

****

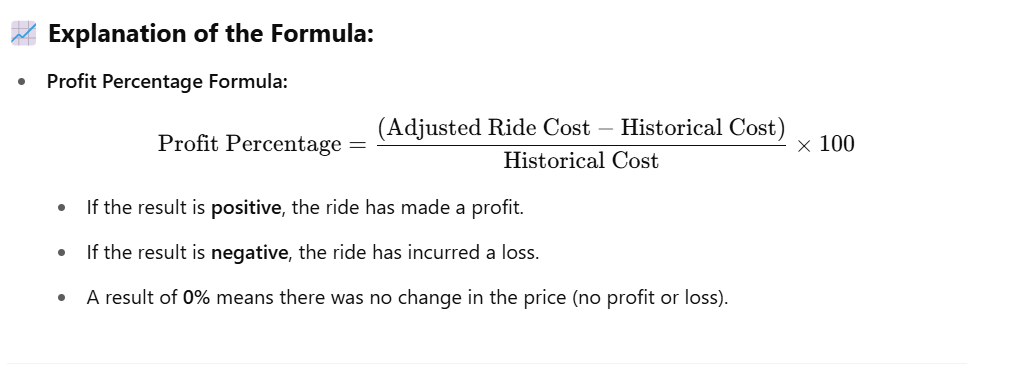
To calculate the profit percentage for each ride based on the difference between the adjusted ride cost and the historical cost of the ride. Then, to identify rides that are profitable or loss-making based on the calculated profit percentage.

 **data['adjusted\_ride\_cost']**: This is the dynamic, adjusted price for the ride (after considering demand and supply factors).

 **data['Historical\_Cost\_of\_Ride']**: This is the baseline price of the ride (without adjustments).

Positive result: Profit, i.e., the adjusted price is higher than the historical price.

Negative result: Loss, i.e., the adjusted price is lower than the historical price.



Profit Percentage Calculation

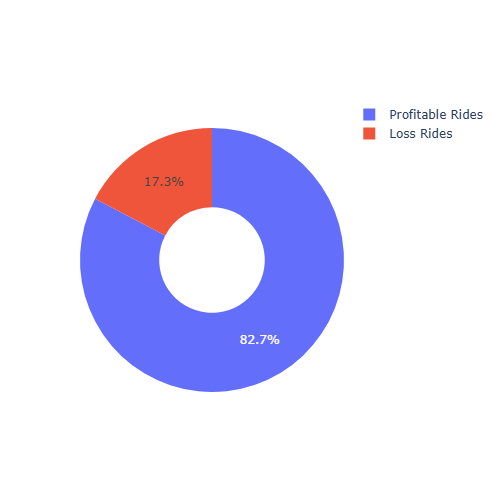
The profit percentage is calculated for each ride by comparing the adjusted ride cost (after dynamic pricing) with the historical cost of the ride. This allows us to identify:

Profitable Rides: Rides where the dynamic pricing resulted in an increase in the price.

Loss-Making Rides: Rides where the dynamic pricing resulted in a decrease in the price.

The profit/loss percentage provides valuable insights into how well the dynamic pricing model performs under varying market conditions.

**Profitability of Rides Visualization**

****

The donut chart above shows the distribution of **profitable** and **loss-making rides** based on the adjusted prices (dynamic pricing) compared to the historical cost of the ride.

* **Profitable Rides**: Represent the rides where dynamic pricing resulted in a price increase.
* **Loss-Making Rides**: Represent the rides where dynamic pricing led to a price decrease.

This visualization provides a quick insight into the overall performance of the dynamic pricing model in generating revenue for the platform.

**Model Training**

**Feature and Target Selection**

In this dynamic pricing system, the dataset was strategically split into independent variables (features) and dependent variables (targets) based on their relevance to the predictive business objective—accurately estimating ride costs based on operational and contextual ride factors.

**Independent Variables (X)**

The selected features used as input to the machine learning model are:

| **Feature** | **Description** | **Relevance** |
| --- | --- | --- |
| **Number\_of\_Riders** | Total number of ride requests at a specific time. | Indicates **demand pressure**; higher values suggest a possible surge in pricing. |
| **Number\_of\_Drivers** | Available drivers during that time frame. | Reflects **supply availability**; influences cost based on scarcity or abundance. |
| **Time\_of\_Booking** | Time slot when the booking is made (e.g., Morning, Evening). | Captures **peak and off-peak hours**; a critical factor for demand cycles. |
| **Vehicle\_Type** | Type of ride selected (Economy or Premium). | Differentiates **base fare categories**; Premium typically commands higher prices. |
| **Expected\_Ride\_Duration** | Estimated duration of the trip in minutes. | Directly proportional to cost; longer trips consume more resources. |

These variables were carefully chosen after exploratory analysis and business logic validation, focusing only on attributes that had a significant influence on ride pricing. Features like Customer\_Loyalty\_Status and Number\_of\_Past\_Rides were deliberately excluded, as analysis showed they had minimal impact on pricing decisions from a business profitability standpoint.

**Dependent Variables (y)**

Two distinct target variables were defined to serve different business analysis perspectives:

| **Target Variable** | **Description** | **Purpose** |
| --- | --- | --- |
| **y1: Historical\_Cost\_of\_Ride** | Actual historical prices recorded for each ride. | Used to **train the initial model** and benchmark historical pricing strategies. |
| **y2: Adjusted\_Ride\_Cost** | Business-optimized ride price considering demand-supply multipliers. | Used to **predict more accurate and profitable prices** from a business standpoint, helping to avoid revenue loss or underpricing. |

The use of two target variables enables a dual-model setup, supporting both retrospective analysis and forward-looking pricing optimization. This separation ensures the business can compare past performance against machine-generated intelligent pricing suggestions.

**Label Encoding**

Label Encoding of Categorical Features

In order to prepare the dataset for machine learning model training, categorical variables needed to be converted into a numerical format, as models like Random Forest do not natively handle non-numeric inputs. The following categorical columns were encoded:

1. Vehicle\_Type This column contained categories such as "Economy" and "Premium".

A label encoding approach was applied to convert these into integer labels.

The encoded values numerically represent the vehicle classes, preserving the categorical nature while making them model-compatible.

This feature plays a critical role in cost prediction due to the inherent pricing differences between vehicle types.

1. Time\_of\_Booking This feature included values like "Morning", "Evening", "Night", and "Afternoon".

Each time slot was encoded into a unique integer to reflect the temporal context of the booking.

Encoding this allowed the model to learn time-dependent patterns in ride pricing, such as demand surges during peak hours.

By encoding these features, the system maintained the categorical distinctions while ensuring compatibility with the machine learning pipeline. These transformations were essential to preserve feature integrity and support accurate pattern recognition in the model.

**Feature Scaling**

After encoding the categorical variables and completing feature engineering, **feature scaling** was applied to the dataset. This step ensures that all input variables are brought to a **uniform scale**, which is particularly important for distance-based models and can improve the stability and performance of tree-based models like Random Forest.

**⚙️ Why Scaling Was Needed:**

* The dataset includes features with varying ranges—e.g., Number\_of\_Riders could range in tens or hundreds, while encoded values like Time\_of\_Booking are single-digit integers.
* Such disparity in feature magnitudes can **skew the model’s learning**, especially when the model assigns undue importance to features with larger numeric values.

**🔧 Applied Method:**

* A **scaler** was used to normalize the entire feature set (dataEncoded), ensuring that all features contribute **equally** to the model’s learning process.
* Although Random Forest is not scale-sensitive in a strict sense, **standardizing the data** helps maintain consistency and can improve training efficiency when paired with downstream models or evaluations.

This preprocessing step plays a supportive role in enhancing model generalization and interpretability.

**Train-Test Split**

To evaluate the model’s ability to generalize to unseen data, the dataset was split into **training and testing sets** using a standard **80-20 split**. This process was applied separately for predicting:

**1. Historical Cost of Ride (y1)**

* This represents the **original ride cost** based on historical records in the dataset.
* A machine learning model was trained on 80% of the data (X\_train, y\_train) and tested on the remaining 20% (X\_test, y\_test) to evaluate performance.
* This prediction serves as a **benchmark** for validating how closely the model can replicate known pricing patterns.

**2. Adjusted Ride Cost (y2)**

* This is a **modified version of the ride cost**, accounting for business-defined pricing adjustments based on demand, supply, and other dynamic factors.
* A separate train-test split was conducted to build and validate a model specifically focused on **business-optimized pricing strategies**.

Both splits were performed with a **fixed random seed (42)** to ensure reproducibility. This step is vital for fair evaluation and comparison of model results on consistent data partitions.

**Model Selection and Training**

To build a reliable and robust prediction system, the **Random Forest Regressor** was selected as the core machine learning model. This algorithm was used to train **two separate models** tailored for different business objectives:

**1. Model for Adjusted Ride Cost**

* This model was trained to predict the **business-adjusted ride cost**, factoring in demand-supply dynamics and operational goals.
* It was trained on the features (X\_train1) and target (y\_train1) derived from the adjusted cost.
* The adjusted model reflects **real-time, optimized pricing** tailored to business profitability, not just historical patterns.

**2. Model for Historical Ride Cost**

* This model was trained on historical ride cost data (y\_train), which serves as a **baseline reference**.
* It helps the business compare actual historical pricing with dynamically predicted pricing.
* This model aids in understanding how traditional pricing aligns or deviates from the proposed optimized pricing system.

**Why Random Forest?**

* **Ensemble Learning**: Combines multiple decision trees to improve accuracy and reduce overfitting.
* **Handles Non-Linear Relationships**: Well-suited for complex real-world data.
* **Feature Importance Insights**: Useful for understanding which variables drive price prediction.
* **Scalability**: Efficient for moderate-sized datasets like the 1000-record dataset used here.

Both models serve distinct but complementary purposes in the overall system: one for maintaining a historical baseline, and the other for supporting **data-driven, business-oriented price optimization**.

**Model Evaluation**

After training the two Random Forest Regressor models, their performance was evaluated using standard regression metrics: **Mean Absolute Error (MAE)**, **Mean Squared Error (MSE)**, and **R² Score**. These metrics help assess how accurately the models can predict ride costs based on the input features.

**Adjusted Ride Cost Prediction Model**

* **Mean Absolute Error (MAE):** 55.38
  + On average, the predicted adjusted price deviates from the actual adjusted cost by around ₹55.
* **Mean Squared Error (MSE):** 5323.64
  + Captures the magnitude of large errors more strongly, indicating low variance in predictions.
* **R² Score:** 0.85
  + The model explains approximately **85% of the variance** in adjusted ride pricing, indicating **strong predictive performance**.

This model is designed for **business-focused pricing optimization** and performs well, making it a valuable tool for adjusting prices based on current conditions.

**📊 Historical Ride Cost Prediction Model**

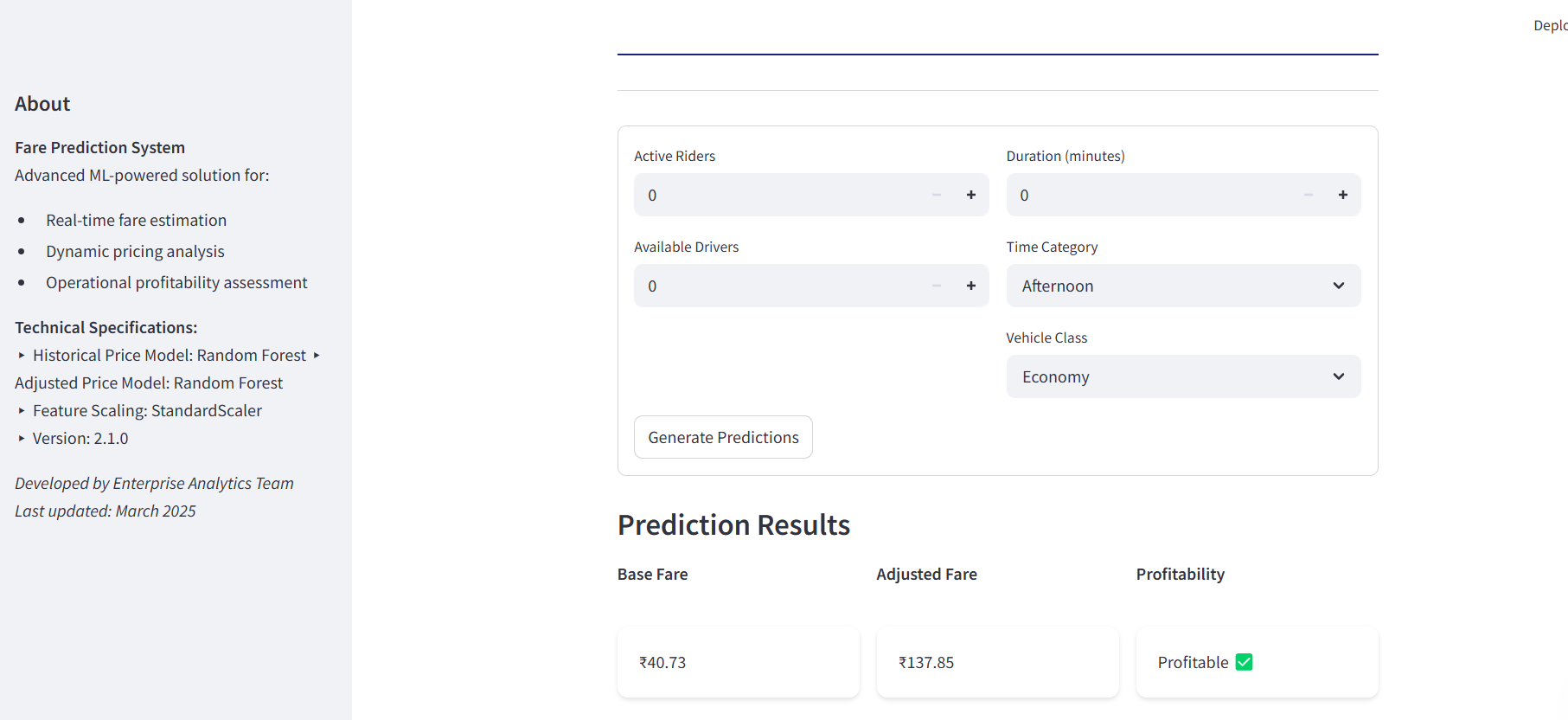
* **Mean Absolute Error (MAE):** 121.63
  + The predictions deviate more from the historical ride cost, which is expected as this baseline may contain inconsistencies or rigid pricing.
* **Mean Squared Error (MSE):** 31,859.30
  + Significantly higher than the adjusted model, reinforcing the presence of outliers or legacy pricing patterns.
* **R² Score:** 0.86
  + Despite the higher error, the model still explains **86% of the variance**, showing it can effectively capture historical pricing trends.

**Insights:**

* The **adjusted pricing model performs better** in terms of absolute error and variance, aligning with the goal of providing **business-optimized, demand-driven pricing**.
* The **historical model**, while accurate in learning past data, serves primarily as a reference and highlights the limitations of static pricing systems.

**Implementation Details**

**Streamlit Interface (Frontend)**

****

The system is deployed using **Streamlit**, providing an intuitive web-based interface that allows businesses to input relevant real-time parameters. The UI is designed with interactive elements to simulate ride conditions:

* **Number Inputs**:
  + Number\_of\_Riders: Active demand at the time.
  + Number\_of\_Drivers: Available supply.
  + Expected\_Ride\_Duration: Estimated ride time in minutes.
* **Dropdown Menus**:
  + Vehicle\_Type: Selectable options (e.g., Economy, Premium) encoded via label encoder.
  + Time\_of\_Booking: Categorical time slots (e.g., Morning, Evening, Night), also label encoded.
* **Submit Button**:
  + Triggers the backend logic and runs the trained models for predictions.
  + Outputs the **Historical Cost**, **Adjusted Price**, and a **Profitability Indicator**.

**Output:**

Upon submission:

* The **historical cost** of the ride (based on past data patterns) is predicted using model\_hist.
* The **adjusted ride price** (reflecting current demand-supply conditions) is predicted using model\_adj.
* The system then evaluates whether the adjusted ride is **profitable**, based on a threshold:

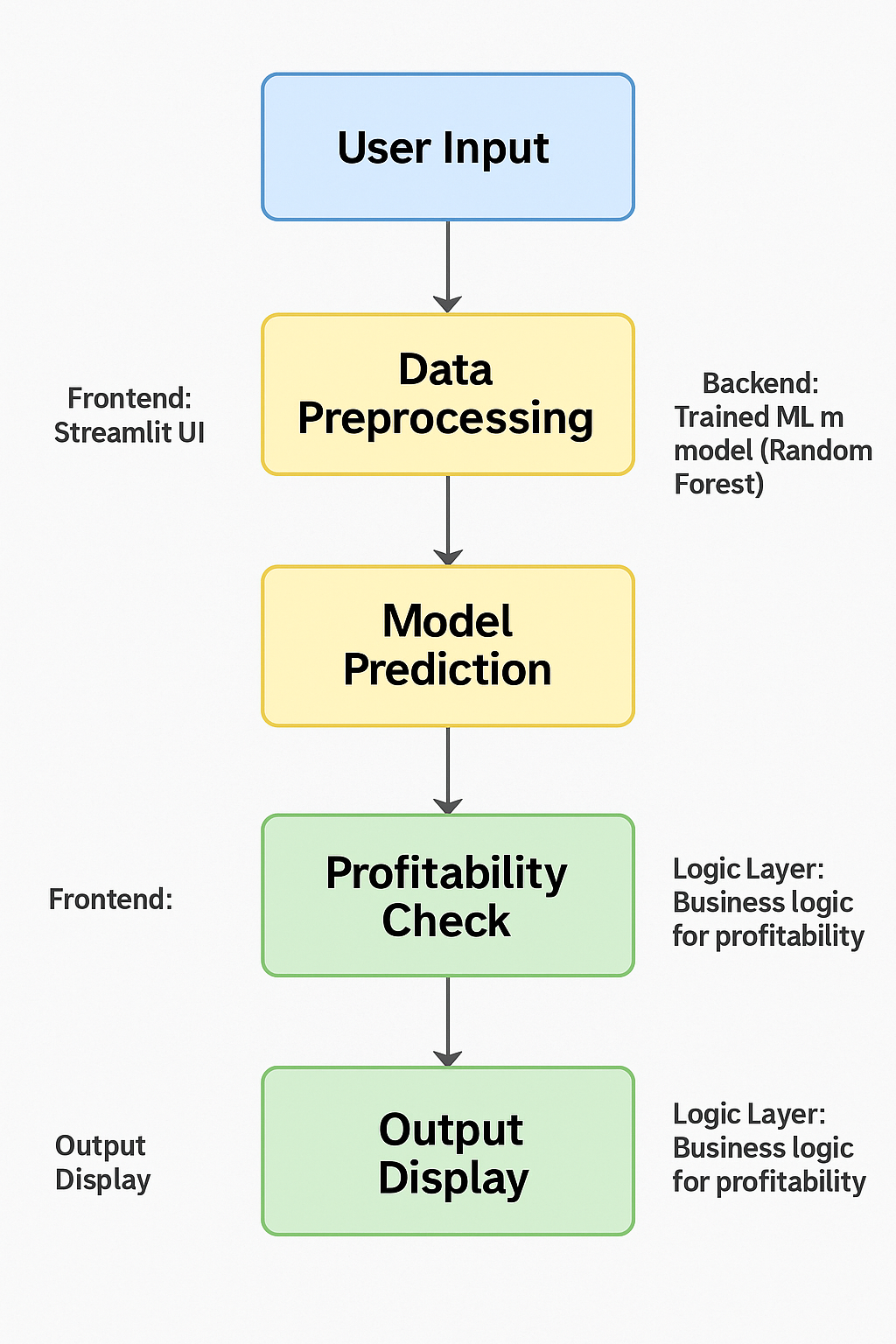
is\_profitable = adjusted\_price > actual\_price + 10

This profitability check provides actionable insight for the business in decision-making and pricing adjustments.

**Backend Architecture**

The backend handles model execution, data transformation, and result generation using pre-trained artifacts:

* Models & Tools Loaded via joblib:
  + model\_hist.pkl: Trained Random Forest model to predict historical price.
  + model\_adj.pkl: Trained Random Forest model to predict adjusted price.
  + scaler.pkl: StandardScaler used to normalize input features during training.
  + time\_label.pkl: LabelEncoder for time-of-booking categories.
  + vehicle\_label.pkl: LabelEncoder for vehicle types.
* **Processing Flow**:
  + User inputs are collected from the frontend.
  + Inputs are label encoded and scaled using the respective encoders and scaler.
  + Both models are run in parallel for their respective predictions.
  + The profitability logic is executed.
  + The results are sent back to the frontend for display.



**Technology Stack**

**Programming Language: Python**

Python was chosen due to its simplicity, readability, and vast ecosystem of libraries that are well-suited for data analysis and machine learning. It provides efficient handling of large datasets, supports various ML algorithms, and offers seamless integration with deployment tools.

**Development Environment:**

* **Visual Studio Code (VS Code):**  
  Used as the primary Integrated Development Environment (IDE) for writing and organizing Python scripts. It supports extensions like Python, Jupyter, and Git for better development and version control.
* **Jupyter Notebook:**  
  Utilized during the experimentation and model development phase. Jupyter is ideal for interactive coding, data visualization, step-by-step debugging, and documenting code alongside outputs and visualizations.

**Libraries & Frameworks Used:**

**1. Data Handling & Manipulation:**

* **Pandas:**  
  Used extensively for data loading, cleaning, transformation, and analysis. Enabled handling of tabular data (DataFrames), selection of features, and merging and grouping of data during feature engineering.
* **NumPy:**  
  Used for numerical operations and efficient array manipulation. It underpins most other libraries (like Pandas and Scikit-learn) and is essential for data transformation and calculations.

**2. Data Visualization:**

* **Matplotlib & Seaborn:**  
  These libraries were used for static visualizations like box plots, histograms, and correlation matrices. Seaborn, built on top of Matplotlib, was especially helpful in generating visually appealing plots for understanding feature distributions.
* **Plotly:**  
  Used for interactive visualizations. It provided dynamic charts such as scatter plots and heatmaps, enabling deep exploratory data analysis and pattern detection between ride features and cost.

**3. Machine Learning & Model Building:**

* **Scikit-learn:**  
  The backbone of the ML pipeline. Used for:
  + Data preprocessing (label encoding, scaling, splitting)
  + Model selection and training (Random Forest Regressor)
  + Model evaluation (using MAE, MSE, R²)
  + Deployment support (saving models using joblib)

**Model Used: Random Forest Regressor**

* The **Random Forest Regressor** was chosen for its ability to handle non-linear data, reduce overfitting, and provide accurate predictions.
* It works by building multiple decision trees and combining their outputs, making it robust against noise and variations in the data.
* Two models were trained: one to predict **historical cost**, and another for **adjusted ride cost** based on supply-demand logic and business objectives.

**Deployment Platform: Streamlit**

* **Streamlit** was used to build and deploy the web-based application.
* It offers a fast way to create interactive interfaces for ML models with minimal frontend development.
* Features include:
  + Interactive input fields (e.g., number of riders, drivers, vehicle type, time of booking)
  + Real-time prediction output (historical vs. adjusted ride prices)
  + Profitability check logic implemented for business decision-making

This complete stack enabled the development of a **prototype-level intelligent pricing system** that is scalable, interactive, and business-focused.

**Conclusion and Future Scope**

**Conclusion**

This project successfully demonstrates the design and implementation of a **prototype dynamic pricing system for cab rides**, using machine learning techniques tailored for business optimization. By leveraging **Random Forest Regression**, the system accurately predicts ride prices based on core factors such as:

* **Demand and Supply Dynamics** (Number of Riders and Drivers)
* **Ride Context** (Time of Booking, Vehicle Type, Duration)

Unlike traditional static pricing models, this system integrates **feature-driven price prediction** and evaluates **ride profitability**, empowering businesses to make informed, real-time pricing decisions that align with operational goals.

Deployed through **Streamlit**, the solution features an intuitive business dashboard interface, making it highly accessible for managerial use cases such as:

* Operational cost control
* Dynamic revenue optimization
* Strategic pricing during peak or low-demand periods

This project stands out by excluding customer loyalty or history-based bias, thereby promoting **context-aware pricing** based purely on real-time ride and supply-demand conditions—making it both **fair** and **scalable**.

**Future Scope**

While the current system presents a strong foundation, several enhancements can elevate it from a prototype to an enterprise-grade solution:

**1. Real-Time API Integration**  
Incorporating APIs to feed **live data** into the model will increase pricing accuracy and context sensitivity. This may include:

* **Traffic congestion data**
* **Weather conditions**
* **Event-based city activity (concerts, rallies, etc.)**

Such integrations can refine the prediction further by adjusting prices in accordance with environmental and temporal factors.

**2. Surge Pricing with Reinforcement Learning**  
Implementing reinforcement learning (RL) techniques could automate surge pricing decisions:

* RL agents could learn optimal pricing policies based on historical demand-supply patterns.
* The system could dynamically adjust prices while balancing **customer satisfaction** and **profit maximization**.

**3. Multi-City & Regional Model Adaptation**  
Currently designed on a prototype dataset, future models can be:

* **Customized for specific cities or regions** by incorporating local ride patterns.
* Adjusted for **currency, fuel cost, or policy differences** to improve localization.
* Enhanced with **geospatial clustering** for region-specific pricing zones.

**4. Business Intelligence Integration**  
Embedding the system into a broader **BI dashboard** could enable:

* Weekly/monthly pricing performance reports
* Heatmaps of profitable vs. non-profitable rides
* Alerts on unusual demand-supply imbalances

**5. Model Feedback Loop for Continuous Learning**  
Deploying feedback mechanisms (e.g., comparing predicted vs. actual ride success) would allow the system to **retrain and adapt over time**, improving its predictive power.

This project not only lays the groundwork for an intelligent pricing tool but also opens pathways for developing **data-driven, context-aware decision-making engines** that businesses can scale and customize across geographies and services.

**References**

**Scikit-learn Documentation**

Official documentation for the Scikit-learn library used for implementing the Random Forest model and preprocessing techniques.  
🔗 https://scikit-learn.org/stable/documentation.html

**Streamlit Documentation**

Guides and API references for building and deploying the interactive machine learning application.  
🔗 https://docs.streamlit.io/

**Pandas Documentation**

Reference for data manipulation, cleaning, and exploration.  
🔗 https://pandas.pydata.org/docs/

**NumPy Documentation**

Used for numerical operations, array handling, and statistical analysis.  
🔗 https://numpy.org/doc/

**Matplotlib & Seaborn Documentation**

For data visualization in exploratory analysis.  
🔗 https://matplotlib.org/stable/contents.html  
🔗 https://seaborn.pydata.org/

**Plotly Documentation**

For building interactive and publication-quality charts and visualizations.  
🔗 https://plotly.com/python/

**Jupyter Notebook**

Development environment used for prototyping and experimentation.  
🔗 <https://jupyter.org/>

**Dynamic Pricing in the Ride-Hailing Industry: A Review**  
*Chen, M.K., Sheldon, M. (2016).*  
Research paper on price elasticity and dynamic pricing strategies in ride-sharing platforms.  
🔗 https://www.nber.org/papers/w23827

**Uber’s Dynamic Pricing Model: Determinants of Surge Pricing**  
A deep dive into the logic behind Uber's real-time surge pricing mechanism.  
📄 Available via research repositories and economic modeling journals.

**Dynamic Pricing Algorithms and Techniques – ACM Digital Library**  
An overview of machine learning applications in pricing strategies.  
🔗 https://dl.acm.org/

**Dataset Source**  
A simulated dataset modeled for educational and research purposes, resembling real-world cab booking conditions.  
*(Include dataset link if public or mention it's a prototype dataset manually curated.)*

**Machine Learning Yearning – Andrew Ng**  
Reference guide for applying machine learning to real-world problems.  
🔗 https://www.deeplearning.ai/machine-learning-yearning/

**Reinforcement Learning: An Introduction – Sutton & Barto**  
Book reference for proposed future integration of RL-based pricing systems.  
🔗 http://incompleteideas.net/book/the-book.html