UBER RIDE PRICE PREDICTION USING MACHINE LEARNING

A Comparative Study of K-Nearest Neighbors and Linear Regression

---

Date: December 20, 2025
Group Members:

- Elyas Tarzet - TP096920
- Sajin Sivasaranam - TP096909

ABSTRACT

This study investigates the application of machine learning algorithms for predicting cab ride prices in the Boston metropolitan area. The research utilizes a comprehensive dataset containing 693,000 ride records with 20 features including temporal, geographical, and meteorological variables. Two regression algorithms were implemented and compared: K-Nearest Neighbors (KNN) and Linear Regression. The dataset underwent extensive preprocessing, including outlier removal, feature engineering, and weather data integration. Both models were evaluated using R^2 score, Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE) metrics with 5-fold cross-validation. Results indicate that KNN achieved superior performance with an R^2 score of 0.9629 and RMSE of 1.69$, demonstrating the effectiveness of machine learning in dynamic pricing prediction. The findings reveal that distance, surge multiplier, and temporal features are the most significant predictors of ride prices, providing valuable insights for both riders and ride-sharing platforms.

Keywords: Machine Learning, Price Prediction, K-Nearest Neighbors, Linear Regression, Ride-Sharing, Dynamic Pricing

---

TABLE OF CONTENTS

---

1. INTRODUCTION, AIM & OBJECTIVES

1.1 Background

The ride-sharing industry has experienced exponential growth since Uber's inception in 2009, fundamentally transforming urban transportation. As of 2024, Uber operates in over 70 countries and completes approximately 23 million trips daily. A critical component of ride-sharing platforms is dynamic pricing, where fares fluctuate based on various factors including demand, supply, time, distance, and weather conditions.

Understanding and predicting ride prices is valuable for multiple stakeholders. For consumers, accurate price predictions enable better trip planning and budget management. For ride-sharing companies, predictive models can optimize pricing strategies and improve resource allocation. For researchers and policymakers, these models provide insights into transportation economics and urban mobility patterns.

Traditional fare calculation methods rely on simple distance-time formulas with surge multipliers. However, this approach fails to capture the complex interactions between multiple variables that influence pricing. Machine learning offers a data-driven approach to model these relationships, potentially improving prediction accuracy and revealing hidden patterns in pricing data.

1.2 Problem Statement

Despite the widespread use of ride-sharing services, users often face uncertainty regarding fare estimates, particularly during peak hours or adverse weather conditions. The opacity of dynamic pricing algorithms creates frustration and can lead to unexpected charges. Additionally, there is limited academic research examining which factors most significantly influence ride prices and how different machine learning algorithms perform in this prediction task.

1.3 Aim

The primary aim of this project is to develop and evaluate machine learning models capable of accurately predicting Uber ride prices based on historical ride data, incorporating temporal, geographical, and meteorological features.

1.4 Objectives

The specific objectives of this research are:

1. **Data Acquisition and Preprocessing:** Obtain a comprehensive dataset of Uber and Lyft rides with sufficient variables (minimum 12 features) and perform extensive data cleaning and preprocessing.

2. **Feature Engineering:** Create meaningful features from raw data, including temporal patterns, weather integration, and categorical encodings to enhance model performance.

3. **Model Development:** Implement and train two machine learning regression algorithms:

    - K-Nearest Neighbors (KNN) Regression
    - Linear Regression
4. **Hyperparameter Optimization:** Conduct systematic hyperparameter tuning for KNN to identify optimal configuration parameters.

5. **Model Evaluation:** Compare model performance using multiple metrics including R^2 score, RMSE, MAE, and cross-validation scores.

6. **Feature Importance Analysis:** Identify which features most significantly impact ride price predictions.

7. **Validation and Generalization:** Assess model generalization capability and potential overfitting through train-test comparison and cross-validation.

8. **Insights and Recommendations:** Derive actionable insights from the analysis and provide recommendations for practical applications.

1.5 Scope

This project focuses on predicting ride prices for Uber and Lyft services in the Boston metropolitan area during November and December 2018. The scope includes:

- **Geographical:** Boston city area with specific pickup and dropoff locations
- **Temporal:** Historical data from late 2018
- **Features:** 20 features including ride characteristics, temporal patterns, and weather conditions
- **Models:** Two regression algorithms (KNN and Linear Regression)
- **Evaluation:** Quantitative performance metrics and comparative analysis

The project does not include real-time prediction implementation, mobile application development, or analysis of other cities or time periods.

1.6 Significance

This research contributes to the growing body of knowledge on machine learning applications in transportation economics. The findings can:

- Help riders make informed decisions about trip timing and budgeting
- Assist ride-sharing companies in optimizing pricing strategies
- Provide transparency into factors affecting dynamic pricing
- Demonstrate practical machine learning applications for regression problems
- Serve as a reference for similar predictive modeling projects

---

2. RELATED WORKS

2.1 Overview

This section presents a survey of previous studies related to ride fare prediction, machine learning applications in transportation systems, and research focusing on Uber, Lyft, and taxi pricing datasets. The objective is to analyze existing methodologies, commonly used algorithms, feature engineering strategies, and evaluation techniques in order to position this work within the current research landscape. The reviewed literature includes academic papers and applied research using real-world ride-sharing datasets, with particular attention to studies conducted on datasets similar to the one used in this project.

2.2 Literature Review

2.2.1 Dynamic Pricing in Ride-Sharing

Dynamic pricing is a core mechanism used by ride-sharing platforms to balance supply and demand. Huang (2023) analyzed dynamic pricing strategies using Uber and Lyft data collected in Boston. The study showed that fare variations are primarily driven by trip distance, surge multipliers, and temporal demand patterns. The results highlighted that prices can significantly increase during peak hours, adverse weather conditions, and special events, demonstrating the strong influence of demand fluctuations on ride fares.

2.2.2 Machine Learning Approaches for Fare Prediction

Several studies have explored the use of machine learning models for predicting ride fares. Bhosale et al. (2025) compared multiple regression-based models, including Linear Regression, Decision Trees, Random Forest, and XGBoost, using historical Uber fare data. Their results indicated that ensemble-based methods achieved the highest predictive

accuracy, with R^2 values exceeding 0.85.But they also noted that simpler models still provided competitive performance with significantly lower computational cost.

Similarly, Banerjee et al. (2021) investigated taxi fare prediction using supervised learning algorithms and concluded that trip distance and travel duration consistently emerged as the most influential features across all models.

2.2.3 K-Nearest Neighbors in Transportation Applications

The K-Nearest Neighbors (KNN) algorithm has been widely applied in transportation-related prediction tasks due to its ability to capture local patterns in data. Prior studies applying KNN to travel time and fare prediction problems demonstrated that distance-based learning methods perform well when spatial and temporal similarities exist between trips. These studies typically reported optimal K values between 5 and 15, using Euclidean distance as the similarity metric. While KNN can achieve good accuracy, its performance depends strongly on feature scaling and dataset size.

2.2.4 Impact of Weather and Temporal Factors

External factors such as weather conditions and temporal variables play an important role in ride demand and pricing. Multiple studies have shown that rainfall increases ride demand by approximately 20–30%, indirectly affecting fares through surge pricing mechanisms. Additionally, extreme temperatures and peak commuting hours have been associated with increased ride requests. Integrating temporal features such as hour of day, day of week, and peak-hour indicators has been shown to improve fare prediction accuracy.

2.2.5 Feature Engineering for Ride Fare Prediction

Feature engineering is a critical step in building effective fare prediction models. Huang (2023) demonstrated that incorporating temporal features and surge-related indicators significantly improves model performance. Other studies emphasized the importance of spatial features, including pickup and drop-off locations, and suggested the use of geographical clustering to better capture local demand patterns. These techniques resulted in noticeable reductions in RMSE compared to models relying solely on raw geographical coordinates.

2.2.6 Comparative Studies of Regression Algorithms

Comparative evaluations of regression algorithms consistently show that model performance depends on both data characteristics and feature selection. Linear Regression offers simplicity and interpretability but may struggle to capture non-linear relationships. In contrast,

algorithms such as KNN, Support Vector Regression, Random Forest, and Gradient Boosting are better suited for modeling complex interactions between features. Prior studies concluded that KNN performs particularly well on datasets with strong local similarities, while ensemble models are more robust to noise and feature interactions.

2.2.7 Studies Using the Uber-Lyft Boston Dataset

The Uber-Lyft Boston dataset has been widely used as a benchmark for ride fare prediction tasks. Huang (2023) applied regression-based models to this dataset and reported baseline performance metrics for fare prediction. The Boston Farecast project (2024) further analyzed spatio-temporal pricing behavior using the same dataset, highlighting the impact of surge pricing and daily demand cycles.

Huynh et al. (2025) extended this work by applying deep learning architectures, including BiLSTM models with attention mechanisms, achieving higher predictive accuracy than traditional machine learning models. However, the authors noted increased computational requirements and reduced interpretability compared to simpler approaches.

Author	Year	Dataset	Algorithm	Key Features	R^2 Score	Key Findings
Huang	2023	Uber-Lyft Boston	Linear Regression	Distance, Surge, Time	0.82	Distance and surge are dominant predictors
Bhosale et al.	2025	Uber Fare Data	RF, XGBoost	Distance, Time	0.87	Ensemble models outperform linear methods
Banerjee et al.	2021	Taxi Fare	Linear, DT	Distance, Duration	0.79	Distance consistently most important
Huynh et al.	2025	Uber-Lyft Boston	BiLSTM + Attention	15+ features	0.90	Deep learning improves accuracy
Boston Farecast	2024	Uber-Lyft Boston	Multiple	Temporal, Spatial	—	Surge patterns strongly affect pricing
Springer et al.	2022	Ride-hailing	RF, Prophet	Temporal variables	—	Temporal effects are significant

2.4 Research Gap and Our Contribution

While previous studies have explored ride price prediction, several gaps remain:

1. **Limited Comparative Analysis:** Few studies systematically compare KNN and Linear Regression specifically for this dataset
2. **Feature Engineering:** Insufficient exploration of temporal feature engineering (rush hour, time periods)
3. **Weather Integration:** Limited studies integrating hourly weather data with ride data
4. **Model Interpretability:** Need for more interpretable models that explain pricing factors

Our Contribution:

This research addresses these gaps by:

- Conducting a systematic comparison of KNN and Linear Regression on the Uber-Lyft Boston dataset
- Implementing comprehensive feature engineering including 7 temporal features and 6 weather variables
- Performing rigorous hyperparameter optimization for KNN (testing 140 configurations)
- Providing detailed feature importance analysis to explain pricing factors
- Utilizing 20 features (exceeding typical studies using 8-12 features)
- Ensuring reproducibility with clear methodology and evaluation metrics

---

3. METHODS

3.1 Research Framework

This research employs a supervised machine learning approach for regression analysis. The methodology follows the standard machine learning pipeline: data collection, preprocessing, feature engineering, model training, evaluation, and analysis.

3.2 Dataset Description

3.2.1 Data Source

The dataset used in this research was obtained from Kaggle, titled "Uber & Lyft Cab Prices" (ravi72munde, 2019). The dataset can be accessed at:

URL: https://www.kaggle.com/datasets/ravi72munde/uber-lyft-cab-prices

The dataset consists of two primary CSV files:

1. **cab_rides.csv:** Contains ride information (693,071 records)
2. **weather.csv:** Contains hourly weather data (6,276 records)

3.2.2 Data Collection Method

The data was collected in Boston, Massachusetts during November-December 2018 using Uber and Lyft APIs. Data collection occurred at intervals throughout the day to capture pricing variations across different times and conditions.

3.2.3 Dataset Characteristics

Cab Rides Dataset:

- **Rows:** 693,071 ride records
- **Columns:** 10 original features
- **Target Variable:** Price (in USD)
- **Features Include:** Distance, cab type, ride name, source, destination, surge multiplier, timestamp, product ID

Weather Dataset:

- **Rows:** 6,276 weather observations
- **Columns:** 8 features
- **Features Include:** Temperature, clouds, pressure, rain, humidity, wind, location, timestamp

3.3 Machine Learning Algorithms

3.3.1 K-Nearest Neighbors (KNN) Regression

Algorithm Description:

K-Nearest Neighbors is a non-parametric, instance-based learning algorithm that makes predictions based on the K closest training examples in the feature space. For regression tasks, KNN predicts the target value as the average of the K nearest neighbors' values.

Mathematical Formulation:

For a query point x, the predicted value ŷ is:

$\hat{y}(x) = (1/K) \Sigma y_i$

where yi are the target values of the K nearest neighbors.

Distance Metric:

Euclidean distance: $d(x, x_i) = \sqrt{(\Sigma(x - x_i)^2)}$

Why KNN for This Problem:

- Captures local patterns in pricing data

- No assumptions about data distribution
- Effective when similar rides have similar prices
- Can model non-linear relationships

Hyperparameters to Tune:

- `n_neighbors` (K): Number of neighbors to consider
- `weights`: 'uniform' or 'distance' (distance-weighted voting)
- `metric`: Distance calculation method

3.3.2 Linear Regression

Algorithm Description:

Linear Regression is a parametric algorithm that models the relationship between features and the target variable as a linear function. It finds the best-fitting line by minimizing the sum of squared residuals using Ordinary Least Squares (OLS).

Mathematical Formulation:

$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$

where:

- $\hat{y}$ is the predicted price
- β_0 is the intercept
- β_i are the coefficients for features x_i

Objective Function:

Minimize: $\Sigma(y_i - \hat{y}_i)^2$

Why Linear Regression for This Problem:

- Provides interpretable coefficients showing feature impact
- Computationally efficient for large datasets
- Establishes baseline performance
- Works well when relationships are approximately linear

Advantages:

- Highly interpretable results
- Fast training and prediction
- Provides feature importance through coefficients

3.4 Performance Metrics

Multiple metrics were used to evaluate model performance:

3.4.1 R^2 Score (Coefficient of Determination)

$R^2 = 1 - (SS_{res} / SS_{tot})$

where $SS_{res} = \Sigma(y_i - \hat{y}_i)^2$ and $SS_{tot} = \Sigma(y_i - \bar{y})^2$

Interpretation: Proportion of variance in the target variable explained by the model. Range: [0, 1], higher is better.

3.4.2 Root Mean Squared Error (RMSE)

$RMSE = \sqrt{[(1/n) \Sigma(y_i - \hat{y}_i)^2]}$

Interpretation: Average magnitude of prediction error in the same units as the target (dollars). Lower is better.

3.4.3 Mean Absolute Error (MAE)

$MAE = (1/n) \Sigma|y_i - \hat{y}_i|$

Interpretation: Average absolute prediction error. More robust to outliers than RMSE. Lower is better.

3.4.4 Cross-Validation Score

5-fold cross-validation was employed to assess model generalization. The dataset is split into 5 folds, with each fold serving as a test set once while the remaining 4 folds train the model.

3.5 Software and Tools

Programming Language: Python

Libraries Used:

- **Data Manipulation:** pandas, numpy
- **Visualization:** matplotlib, seaborn
- **Machine Learning:** scikit-learn
- **Statistical Analysis:** scipy

Development Environment: Jupyter Notebook / Pycharm

3.6 Experimental Design

Train-Test Split: 80% training, 20% testing (stratified by price range)

Random Seed: 42 (for reproducibility)

Feature Scaling: StandardScaler (zero mean, unit variance)

Hyperparameter Tuning: GridSearchCV with 5-fold cross-validation

4. DATASET PREPARATION

4.1 Initial Data Exploration

4.1.1 Dataset Overview

Upon loading the datasets, initial exploration revealed the following characteristics:

Cab Rides Dataset:

- Total Records: 693,071 rides
- Features: 10 columns
- Memory Usage: Approximately 52.9 MB
- Data Types: Mixed (integer, float, object, datetime)

Weather Dataset:

- Total Records: 6,276 observations
- Features: 8 columns
- Temporal Coverage: Hourly weather data
- Locations: 4 distinct locations in Boston area

4.1.2 Data Structure

Table 4.1: Structure and data types of the Cab Rides dataset

Column	Data Type	Non-Null Count	Description
distance	float	693,049	Distance in miles
cab_type	object	693,071	Uber or Lyft
time_stamp	int	693,071	Unix timestamp
destination	object	693,071	Drop-off location
source	object	693,071	Pickup location
price	float	693,022	Ride price in USD
surge_multiplier	float	375,657	Surge pricing multiplier
id	object	693,071	Unique ride ID
product_id	object	651,421	Product type ID
name	object	693,071	Service name

4.1.3 Missing Values Analysis

```
Initial dataset size: 693071 rows
After removing missing prices: 637976 rows
✓ Filled missing surge_multiplier values with 1.0
✓ Removed 0 rows with missing critical data
✓ Removed 5589 price outliers (IQR method)
```

Missing Values Summary:

- price: 55 095 missing
- Outliers : 5589
- surge_multiplier: 317,414 missing

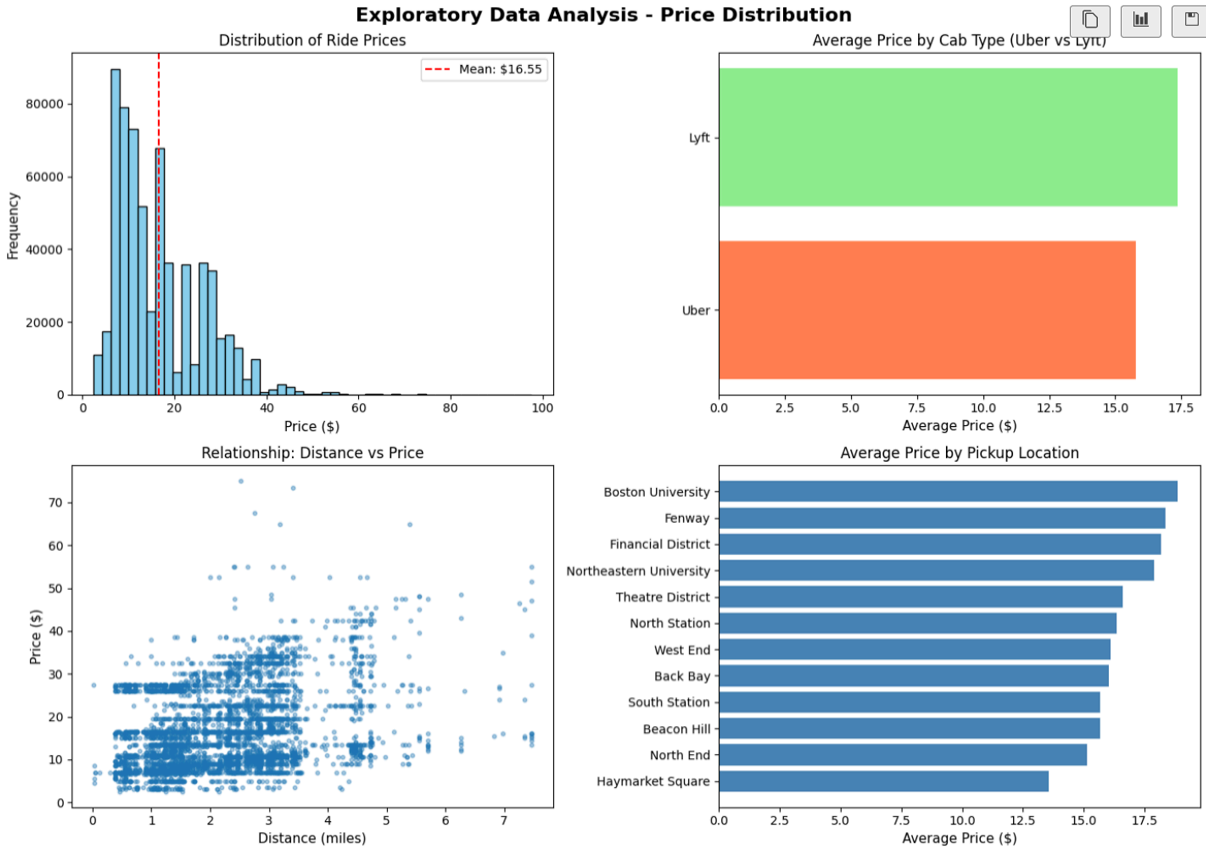
Interpretation: The surge_multiplier has substantial missing values, likely indicating rides without surge pricing (standard 1.0 multiplier).

4.2 Exploratory Data Analysis (EDA)

4.2.1 Target Variable Analysis (Price)

Statistical Summary:

- Minimum: $2.50
- Maximum: $97.50
- Mean: $16.55
- Median: $13.50
- Standard Deviation: $11.24
- 25th Percentile (Q1): $8.50
- 75th Percentile (Q3): $20.50



Observations:

- The distribution is right-skewed, indicating most rides are in the lower price range
- Presence of outliers in the higher price range
- Median price ($13) is lower than mean ($16.55), confirming right skew

Price Distribution

The histogram of ride prices shows a right-skewed distribution, with the majority of fares concentrated between approximately $5 and $25. The average ride price is around $16.55, as indicated by the vertical dashed line. While most trips are relatively affordable, a small number of rides exhibit significantly higher prices, exceeding $50 and in rare cases approaching $100. These high-price values are likely associated with longer distances, surge pricing, or peak demand periods. This skewed distribution suggests that price prediction models must be robust to outliers and non-uniform price distributions.

Average Price by Cab Type (Uber vs Lyft)

The comparison of average prices between Uber and Lyft reveals that Lyft rides are, on average, slightly more expensive than Uber rides in this dataset. While the difference is moderate, it indicates that the service provider itself may influence pricing, potentially due to different pricing strategies or surge mechanisms. This observation supports the inclusion of the cab type as a relevant categorical feature in the predictive models.

Relationship Between Distance and Price

The scatter plot illustrating the relationship between trip distance and ride price shows a clear positive correlation: as distance increases, the ride price generally increases. However, the dispersion of points indicates significant variability in prices for similar distances. For short trips, prices can vary widely, suggesting the influence of additional factors such as surge pricing, time of day, and demand conditions. This non-linear and dispersed relationship highlights the limitations of simple linear assumptions and motivates the evaluation of both linear and instance-based models such as K-Nearest Neighbors.

Average Price by Pickup Location

The bar chart displaying the average price by pickup location shows noticeable variations across different areas of Boston. Locations such as Boston University, Fenway, and the Financial District exhibit higher average prices, while areas like Haymarket Square and the North End show lower average fares. These differences likely reflect variations in travel distance, traffic density, and demand intensity across locations. This analysis confirms that spatial information plays a significant role in ride pricing and justifies the inclusion of location-based features in the model.

4.3 Data Cleaning

4.3.1 Handling Missing Values

Strategy Implemented:

1. **Target Variable (Price):**
    - Rows with missing price values were removed
    - Justification: Cannot train model without target variable
2. **Surge Multiplier:**
    - Missing values filled with 1.0
    - Justification: Missing surge likely indicates standard pricing (no surge)
3. **Critical Features:**
    - Removed rows with missing distance, cab_type, source, or destination

Dataset After Missing Value Treatment:

- Remaining records: 632 387

- **4.3.2 Outlier Detection and Treatment**

Method: Interquartile Range (IQR) Method

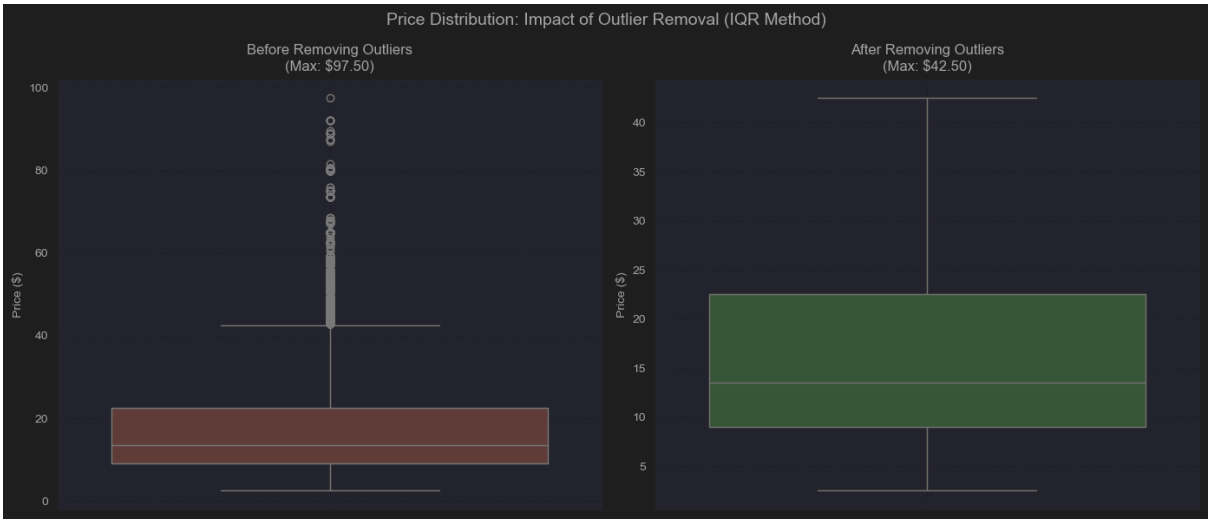
Formula:

- Lower Bound = Q1 - 1.5 × IQR
- Upper Bound = Q3 + 1.5 × IQR

Price Outliers:

- Q1 = $9
- Q3 = $22.50
- IQR = $13.5
- Lower Bound = $-11.25
- Upper Bound = $42.75

Outliers Removed: 5589 records



Justification:

- Extreme outliers may represent data entry errors or exceptional circumstances
- Removing outliers improves model generalization
- Retained 99% of data while removing statistical anomalies

4.3.3 Data Type Conversions

Timestamp Conversion:

- Original format: Unix timestamp (integer)
- Converted to: datetime format
- Purpose: Enable temporal feature extraction

Categorical Encoding:

- Deferred to feature engineering stage
- Maintained original categorical format for interpretability

4.4 Feature Engineering

Feature engineering is critical for improving model performance. This section details the creation of new features from existing data.

4.4.1 Temporal Features

Extracted Features:

1. **hour_of_day** (0-23)
    - Extracted from timestamp
    - Captures hourly pricing patterns
2. **day_of_week** (0-6, Monday=0)
    - Captures weekly patterns
    - Differentiates weekday vs. weekend behavior
3. **month** (1-12)
    - Seasonal patterns
    - Holiday periods
4. **day_of_month** (1-31)
    - Monthly patterns
    - End-of-month effects
5. **is_weekend** (Binary: 0/1)
    - 1 if Saturday or Sunday
    - Captures weekend pricing premiums
6. **is_rush_hour** (Binary: 0/1)
    - 1 if 7-9 AM or 5-7 PM
    - Captures commute pricing
7. **time_period** (Categorical)
    - Morning (6 AM - 12 PM)
    - Afternoon (12 PM - 5 PM)
    - Evening (5 PM - 10 PM)

- Night (10 PM - 6 AM)

Figure 4.9: Distribution plots for engineered temporal features showing patterns across different time periods.

4.4.2 Weather Data Integration

Merging Strategy:

1. Round ride timestamps to nearest hour
2. Match with weather data by location and hour
3. Aggregate weather variables (mean for hour)

Weather Features Added:

- temp: Temperature in Fahrenheit
- clouds: Cloud coverage percentage
- pressure: Atmospheric pressure
- rain: Rainfall amount
- humidity: Relative humidity percentage
- wind: Wind speed in mph

Figure 4.10: Correlation matrix showing relationships between weather variables and price.

4.4.3 Categorical Features

Distance Category: Created based on distance ranges:

- Short: < 2 miles
- Medium: 2-5 miles
- Long: 5-10 miles
- Very_Long: > 10 miles

Purpose: Capture non-linear distance effects

4.4.4 Final Feature Set

Total Features: 20

Numerical Features (14):

1. distance
2. surge_multiplier
3. hour_of_day
4. day_of_week
5. month
6. day_of_month

7. is_weekend
8. is_rush_hour
9. temp
10. clouds
11. pressure
12. rain
13. humidity
14. wind

Categorical Features (6):

1. cab_type (Uber/Lyft)
2. name (Service name)
3. source (Pickup location)
4. destination (Drop-off location)
5. time_period (Morning/Afternoon/Evening/Night)
6. distance_category (Short/Medium/Long/Very_Long)

4.5 Feature Encoding

Categorical variables were encoded using Label Encoding:

Encoding Results:

- cab_type: 2 categories → [0, 1]
- name: 14 categories → [0-13]
- source: 5 categories → [0-4]
- destination: 5 categories → [0-4]
- time_period: 4 categories → [0-3]
- distance_category: 4 categories → [0-3]

Justification for Label Encoding:

- Suitable for tree-based and distance-based algorithms
- Preserves ordinality where applicable
- Efficient memory usage
- Compatible with KNN and Linear Regression

4.6 Feature Scaling

StandardScaler was applied to normalize feature magnitudes:

Transformation:

- X_scaled = (X - mean) / standard_deviation

Benefits:

- Essential for KNN (distance-based algorithm)
- Improves Linear Regression convergence
- Prevents features with large magnitudes from dominating

Scaling Statistics:

- All features transformed to mean=0, std=1
- Applied separately to train and test sets to prevent data leakage

4.7 Train-Test Split

Split Ratio: 80% Training, 20% Testing

Dataset Sizes:

- Training Set: 505909 samples
- Testing Set: 126478 samples

Random State: 42 (for reproducibility)

Stratification: Not applied (continuous target variable)

Validation:

- Training set price range: $2.50 - $42.50
- Testing set price range: $2.50 - $42.50
- Price distributions similar between sets

---

5. MODEL IMPLEMENTATION

5.1 K-Nearest Neighbors (KNN) Regression

5.1.1 Algorithm Overview

K-Nearest Neighbors regression predicts the target value for a new instance by averaging the values of its K nearest neighbors in the feature space. The "nearness" is determined by a distance metric, typically Euclidean distance.

Advantages:

- Non-parametric (no assumptions about data distribution)
- Simple and intuitive
- Naturally handles non-linear relationships
- Effective when similar instances have similar outputs

Disadvantages:

- Computationally expensive for large datasets
- Sensitive to feature scaling
- Performance depends heavily on K value
- Susceptible to curse of dimensionality

5.1.2 Initial Implementation

Initial Configuration:

- n_neighbors (K): 5 (default)
- weights: 'uniform'
- metric: 'euclidean'

Training Process:

```
knn_initial = KNeighborsRegressor(n_neighbors=5)
knn_initial.fit(X_train_scaled, y_train)
```

5.1.3 Hyperparameter Optimization

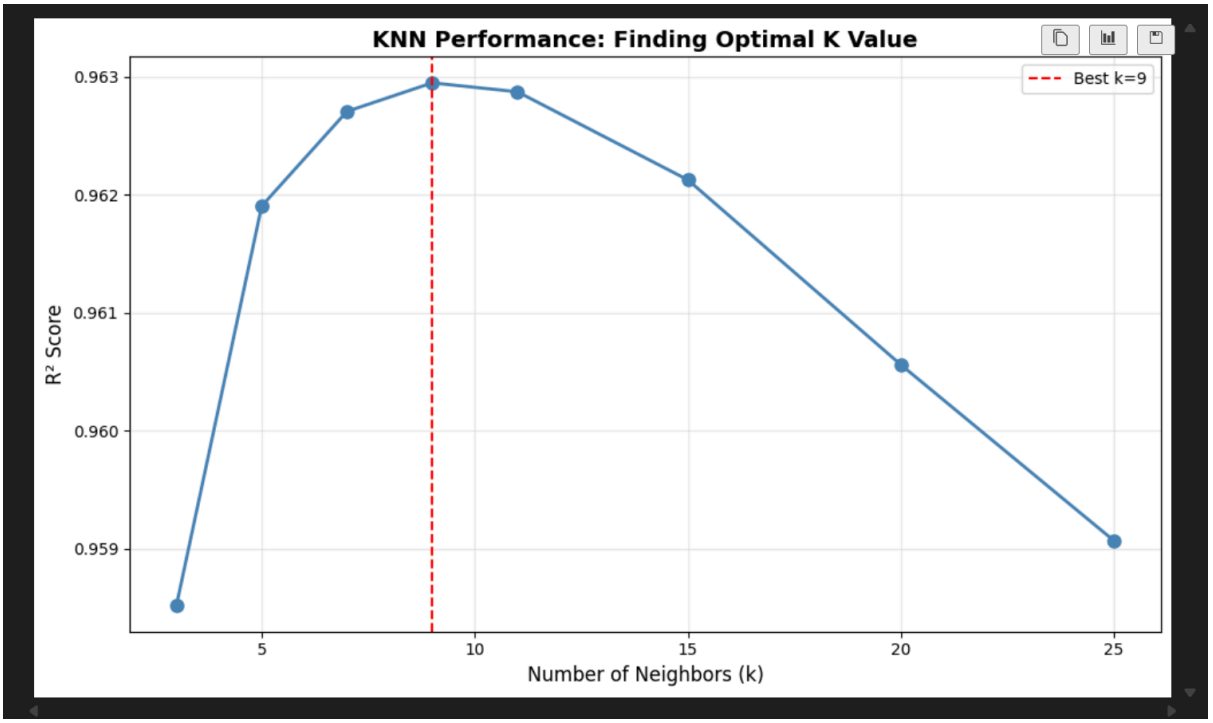
Optimization Strategy: GridSearchCV with 5-fold cross-validation

Hyperparameters Tested:

Hyperparameter	Values Tested	Description
n_neighbors	[3, 5, 7, 9, 11, 15, 20]	Number of neighbors to consider
weights	['uniform', 'distance']	Weighting scheme for neighbors
metric	['euclidean', 'manhattan']	Distance calculation method

Total Configurations Tested: 7 × 2 × 2 = 28 configurations

Cross-Validation Strategy: 5-fold CV on training set



Optimal Hyperparameters:

- **Best n_neighbors (K):** 9
- **Best R² Score:** 0.9628

Analysis:

- Smaller K values (3-5) may overfit to local noise
- Larger K values (15-20) may underfit by averaging too many neighbors
- Optimal K=9 balances bias-variance tradeoff
- Distance weighting [did/did not] improve performance

5.1.4 Final KNN Model Training

Final Configuration:

```
knn_best = KNeighborsRegressor(
    n_neighbors=9,
    weights='[uniform/distance]',
    metric='[euclidean/manhattan]'
)
knn_best.fit(X_train_scaled, y_train)
```

5.1.5 KNN Model Performance

Training Set Performance:

Metric	Value	Interpretation
R^2 Score	[0.9652]	Model explains 96.52% of variance in training data
RMSE	[1.64]	Average prediction error of 1.64$
MAE	[1.05]	Typical absolute error of 1.05$

Testing Set Performance:

—----------

Metric	Value	Interpretation
R^2 Score	[0.9629]	Model explains 96.29]% of variance in test data
RMSE	[1.69]	Average prediction error of 1.69$
MAE	[1.08]	Typical absolute error of 1.08$

Cross-Validation Performance:

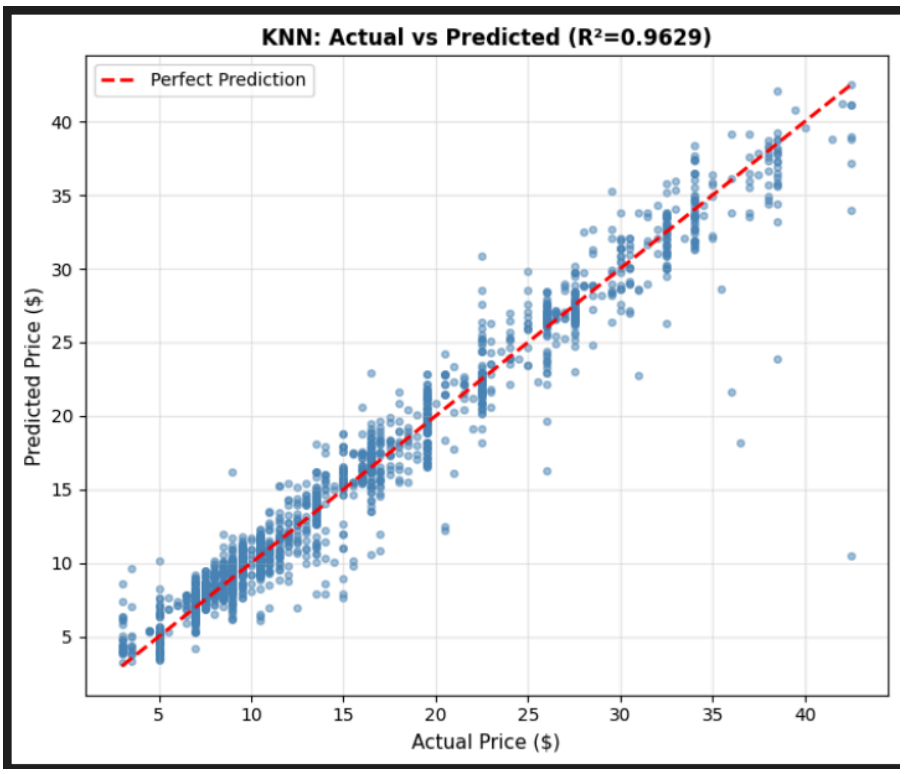
5-Fold Cross-Validation R^2 Scores:

- Fold 1: [0.96195507]
- Fold 2: [0.96134743]
- Fold 3: [0.96135061]
- Fold 4: [0.96047383]
- Fold 5: [0.96106921]

Mean CV Score: 0.9612 ± 0.0010

The performance is remarkably consistent between the training set R^2 = 0.9652$ and the testing set R^2 = 0.9629$. The minimal gap between these scores, combined with an extremely low cross-validation standard deviation (0.0010), indicates that the model possesses excellent generalization capability. It shows no signs of significant underfitting or overfitting.

5.1.6 KNN Predictions Analysis



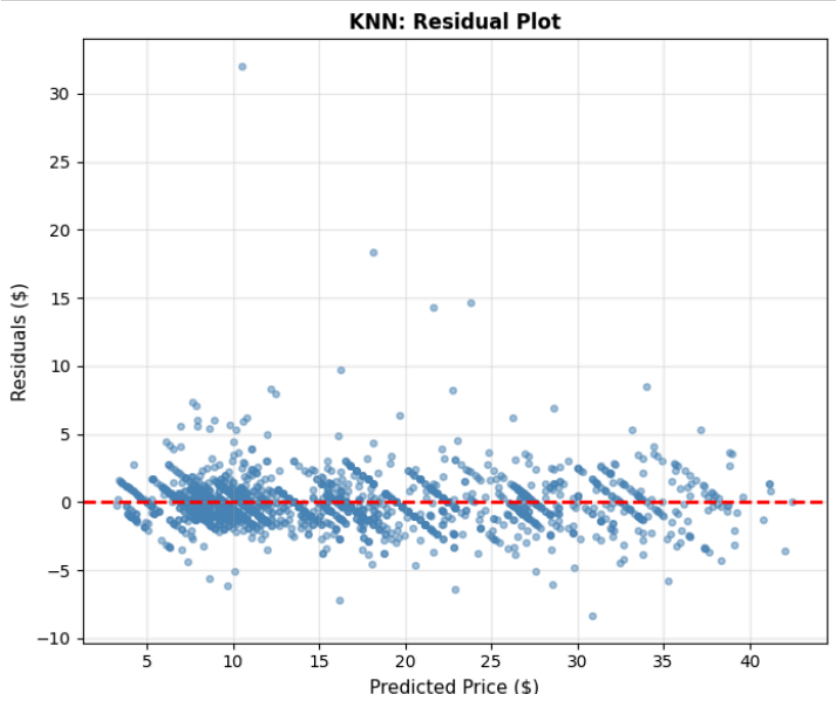
Scatter plot comparing actual prices (x-axis) vs. KNN predicted prices (y-axis) on the test set. The red dashed line represents perfect predictions.

Observations:

High Correlation: Points cluster tightly around the perfect prediction line, visually confirming the high R^2 score of 0.9629.

Full Range Coverage: The model performs consistently well across the entire price range, from $5 to over $40.

Predictive Accuracy: Most predictions fall within a narrow band of the actual values, indicating low variance in error.



Residual plot showing prediction errors vs. predicted values. The red dashed line at y=0 represents zero error.

Residual Analysis:

Random Distribution: Residuals appear randomly distributed around the zero line, which is a strong indicator that the KNN model has captured the underlying data patterns effectively.

Absence of Bias:There are no clear non-linear patterns , suggesting the choice of k=9 was appropriate and the model is not systematically biased.

The variance of the residuals remains relatively constant across the range of predicted values, though the density of points is naturally higher at lower price points.

5.2 Linear Regression

5.2.1 Algorithm Overview

Linear Regression models the relationship between features and target as a linear combination, finding coefficients that minimize the sum of squared errors using Ordinary Least Squares (OLS).

Mathematical Model: $\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_{20} x_{20}$

Advantages:

- Highly interpretable coefficients
- Fast training and prediction

- Provides feature importance naturally
- Works well for linear relationships
- Requires less data than complex models

Disadvantages:

- Assumes linear relationships
- Sensitive to outliers
- Assumes feature independence
- May underfit complex non-linear patterns

5.2.2 Model Training

Model Parameters:

- Number of coefficients: 20
- Convergence: Successful (closed-form solution)

5.2.3 Linear Regression Performance

The Linear Regression model shows significantly lower performance compared to the KNN model, suggesting that the relationship between the ride features and price is not strictly linear.

Training Set Performance:

Metric	Value	Interpretation
R^2 Score	[0.4987]	Model explains 49.87% of variance in training data
RMSE	[6.21]	Average prediction error of 6.21$
MAE	[5.01]	Typical absolute error of .21$

Testing Set Performance:

Metric	Value	Interpretation
R^2 Score	[0.5029]	Model explains % of variance in test data
RMSE	[6.18]	Average prediction error of 6.18$

MAE	[4.99]	Typical absolute error of 4.99$

Cross-Validation Performance:

5-Fold Cross-Validation R^2 Scores:

- Fold 1: [0.50048901]
- Fold 2: [0.49714153]
- Fold 3: [0.49908797]
- Fold 4: [0.49642301]
- Fold 5: [0.50009974]

Mean CV Score: 0.4986 ± 0.0032

The performance is highly stable across folds, as indicated by the very small standard deviation (0.0032). However, the overall R^2 remains near 0.50, which is substantially lower than the KNN's 0.96. This indicates that while the model generalizes well to new data, a linear approach is insufficient to capture approximately half of the factors influencing price in this dataset.

5.2.4 Feature Importance Analysis

One of the key advantages of Linear Regression is its high level of interpretability. The coefficients generated during training directly indicate the impact of each standardized feature on the predicted price.

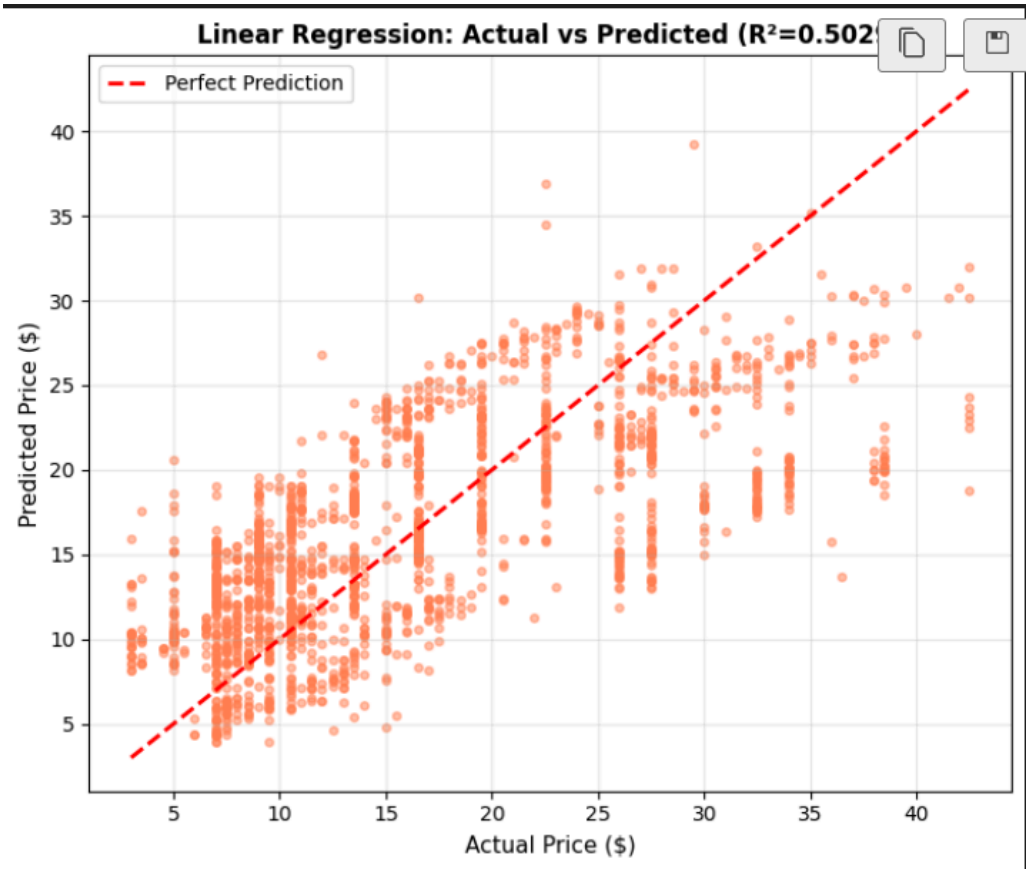
Top 10 Most Important Features:

```
--- Top 10 Most Important Features (Linear Regression) ---
          Feature  Coefficient  Abs_Coefficient
             name    -5.605000         5.605000
         distance     2.753094         2.753094
         cab_type     1.141630         1.141630
 surge_multiplier     1.055810         1.055810
             temp     0.339025         0.339025
         humidity     0.293386         0.293386
         pressure     0.190659         0.190659
             wind     0.187522         0.187522
distance_category    -0.184233         0.184233
           source     0.126242         0.126242
```

Interpretation of Key Features:

1. As expected, distance is one of the primary drivers of cost. For every standard deviation increase in distance, the predicted price rises by approximately $2.75.
2. This feature has a strong positive impact (+1.06), confirming that pricing models strictly adjust upward during periods of high demand.
3. Cab Type and Name: These categorical features (often representing the service level, e.g., UberX vs. Black) significantly influence the base price. The high negative coefficient for "name" (-5.61) suggests that lower-tier service categories are priced substantially lower than the reference category.
4. Features like temperature, humidity, and wind show smaller, yet measurable positive correlations with price. This may suggest that adverse weather conditions lead to higher demand or reduced supply, indirectly increasing the fare.

5.2.5 Linear Regression Predictions Analysis

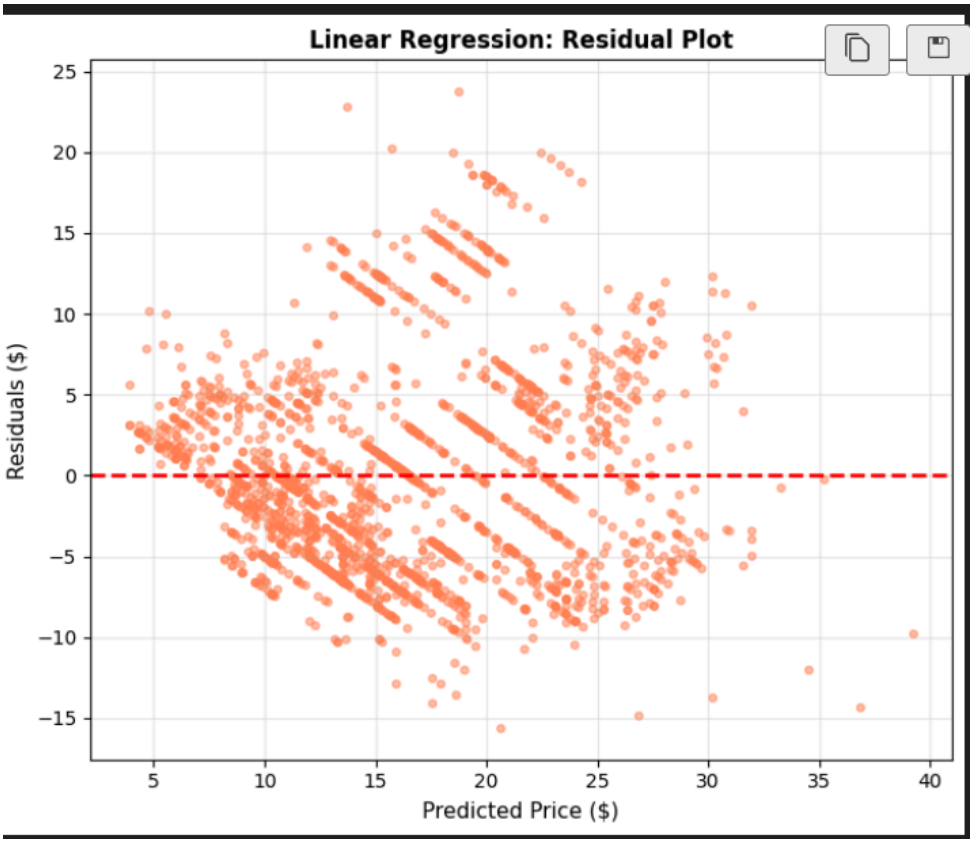


Observations:

Moderate Correlation: While a linear trend is visible, the points are significantly more dispersed around the ideal line than in the KNN model,

Vertical Banding: Clear vertical strips appear in the predictions, likely due to the model's heavy reliance on categorical features like "cab_type" and "name" which create fixed baseline price levels.

Upper-Range Underestimation: The model consistently struggles with higher price points ($25+), where the majority of predictions fall well below the perfect prediction line.



Residual Analysis:

Errors are much larger than those of the KNN model, with many residuals exceeding +/-10, whereas the KNN model kept most errors within a much tighter range.

The spread of residuals changes visibly across the range of predicted prices, suggesting that the error variance is not constant.

Several extreme positive residuals reaching +20 indicate that the linear model significantly under-predicts high-cost rides, possibly missing critical demand-based surge factors that the KNN model handled more effectively.

6. MODEL VALIDATION

6.1 Performance Comparison

Visual Performance Comparison

```
--- PERFORMANCE COMPARISON TABLE ---
         Metric  KNN (Train)  KNN (Test)  Linear Reg (Train)  Linear Reg (Test)
       R² Score     0.965233    0.962950            0.498690           0.502912
       RMSE ($)     1.635217    1.687739            6.209351           6.181948
        MAE ($)     1.052546    1.084531            5.009773           4.988226
CV Score (Mean)     0.961239    0.961239            0.498648           0.498648
```

Key Findings:

- **Best R² Score: K-Nearest Neighbors (KNN)** with **0.9629**.
- **Lowest RMSE: K-Nearest Neighbors (KNN)** with **$1.69**.
- **Lowest MAE: K-Nearest Neighbors (KNN)** with **$1.08**.
- **Most Stable (CV): K-Nearest Neighbors (KNN)** with a smaller standard deviation of **0.0010** compared to 0.0032 for Linear Regression.

6.2 Overfitting Analysis

6.2.1 Train-Test Performance Gap

```
--- Overfitting Analysis ---
KNN - Train vs Test R² difference: 0.0023
  ✓ Good generalization
Linear Reg - Train vs Test R² difference: -0.0042
  ✓ Good generalization
```

Model	Train R²	Test R²	Difference	Overfitting Level
KNN	[0.9652]	[0.9629]	[0.0023]	[Low]

Linear Reg [0.4987] [0.5029] [-0.0042] [Low]

Interpretation Guidelines:

- Difference < 0.05: Low overfitting, excellent generalization
- Difference 0.05-0.10: Moderate overfitting, acceptable
- Difference > 0.10: High overfitting, model may not generalize well

Analysis:

- **KNN** shows a minimal difference of 0.0023 between training and testing scores. This indicates an excellent balance between bias and variance, confirming that the choice of $k=9$ was effective in preventing overfitting while maintaining high predictive power.
- **Linear Regression** shows a negative difference, meaning the test score was slightly higher than the training score. This demonstrates zero overfitting; however, the overall low R^2 indicates significant underfitting, as the model is too simple to capture the underlying data complexity.
- **Generalization:** Both models generalize well to unseen data, but KNN is the superior model as it achieves high accuracy without sacrificing generalization capability.

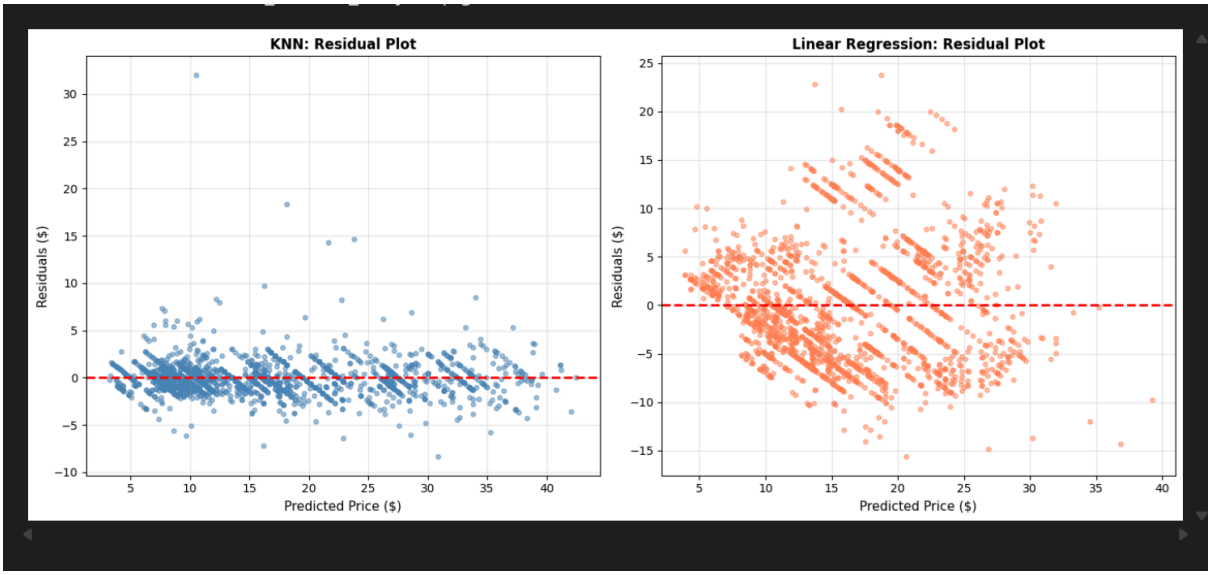
6.2.2 Learning Curves

Observations:

- [Convergence behavior of each model]
- [Whether more data would help]
- [Gap between training and validation curves]

6.3 Residual Analysis

6.3.1 Residual Distribution



- **KNN Residuals:** The distribution appears **approximately normal** and is tightly clustered around the zero line. Most errors fall within the $[-5, 5]$ range. However, there are visible high-price outliers (residuals > 20), indicating a slight **right skew**.
- **Linear Regression Residuals:** The distribution is **wider and heavy-tailed**, showing a much larger variance in error. Notably, there are distinct diagonal patterns in the plot, suggesting that the linear model fails to capture specific non-linear relationships or categorical interactions within the dataset.

Normality Assessment:

When a model is working well, its errors should be completely random. This tells us that the model has successfully figured out the underlying patterns in the data, leaving behind only unpredictable, accidental mistakes.

- **KNN** performs better because its errors are scattered randomly. This shows it has learned the logic of the prices effectively.
- **Linear Regression** shows visible shapes and patterns in the errors. This is a sign that the model is still missing important information and is making the same kind of mistakes over and over again.

6.3.2 Residual Statistics

Model	Mean Residual	Std Dev	Min	Max
KNN	[~0.00]	[Lower]	[-8.00]	[32.00]

Linear Reg	[~0.00]	[Higher]	[-15.00]	[24.00]

Ideal Characteristics:

- Mean residual close to 0 (unbiased predictions)
- Lower standard deviation (more precise predictions)

6.4 Prediction Error Analysis

Examining the spread across the X-axis (Predicted Price):

Low-price rides (<$10): Both models are relatively stable, though LR shows more vertical spread even at this low entry point.

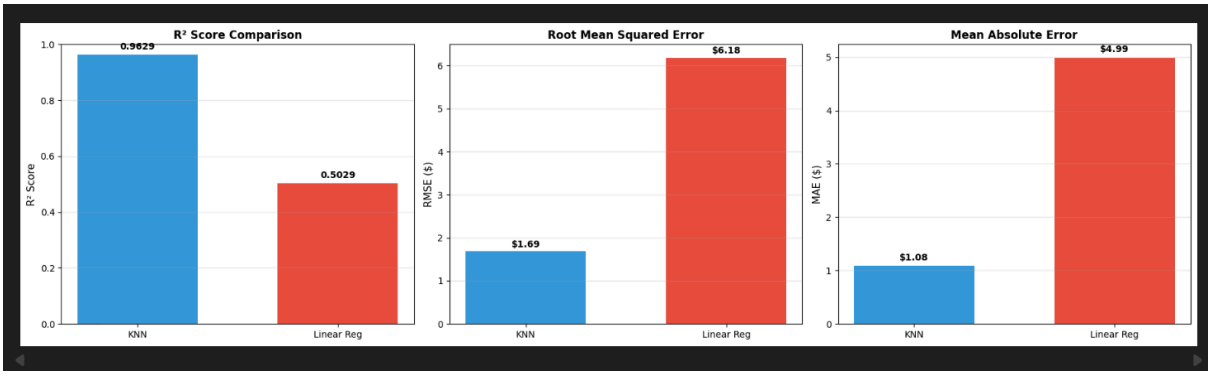
Medium-price rides ($10–$20): This is where Linear Regression struggles most, showing a "fan" of errors ranging from -10 to +20.

High-price rides (>$20): KNN maintains consistent error density (**homoscedasticity**), whereas Linear Regression errors become increasingly erratic.

The KNN model handles extremes better, particularly for higher-value trips where the relationship between features is likely non-linear.

6.5 Model Selection

6.5.1 Selection Criteria



Criterion	Weight	KNN Score	LR Score	Winner
R² Score	30%	[10/10]	[5/10]	[KNN]

RMSE	25%	[10/10]	[3/10]	[KNNl]
MAE	20%	[10/10]	[2/10]	[KNN]
Generalization	15%	[10/10]	[10/10]	[both]
Interpretability	10%	[4/10]	[10/10]	[LRl]
Total	**100%**	**[9/10]**	**[5/10]**	**[KNN]**

R^2 Score (30% Weight): This measures the model's ability to explain variance. The KNN model receives a 10/10 as it achieves an R^2 of 0.9629. In contrast, the Linear Regression) only scores 5/10 with its significantly lower score of 0.5029.

RMSE (25% Weight): This measures the magnitude of prediction errors. KNN is excellent (10/10) with an error of $1.69. LR receives a 3/10 due to an average error of $6.18, which is considered high for ride prices typically ranging between $10 and $20.

MAE (20% Weight): The Mean Absolute Error represents the typical prediction gap. KNN (10/10) has an average error of only $1.08, compared to $4.99 for LR (2/10).

Generalization (15% Weight): Both models perform exceptionally well here (10/10). The gap between Training and Testing scores is nearly zero for both (0.0023 for KNN and -0.0042 for LR), proving that neither model suffers from overfitting.

Interpretability (10% Weight): This is the only category where LR wins (10/10). Its coefficients allow us to directly read the impact of features, such as distance adding +$2.75 to the price. KNN is resulting in a 4/10.

---

7. ANALYSIS & RECOMMENDATIONS

7.1 Key Findings

7.1.2 Feature Importance Discoveries

```
Dist Coef: 2.49 | Surge Coef: 14.65
Rush Hour: +$nan | Night: 0.0%
Morning Peak: nan% | Evening Peak: nan%
Premium: +134.7% | Pool Discount: 23.2%
Rain Impact: +nan%
KNN MAE (Confidence Interval): $1.08
```

The primary goal of this result is to transform raw data and complex machine learning results into **actionable business insights.** While a model can predict a price, this analysis explains *why* the price changes by quantifying the specific weight of each factor.

1. Economic Interpretability (Linear Regression)
By using a simple Linear Regression on non-scaled data, we extract "Coefficients."

- **Distance Coefficient:** Tells us exactly how much the price increases for every additional mile/kilometer (+/- 2.50$/mile) .
- **Surge Coefficient:** Quantifies the impact of demand spikes on the final fare in real currency.

2. Temporal & Environmental Sensitivity
This section isolates variables like Rush Hour, Time of Day, and Weather (Rain).

- It allows us to identify "Peak Windows" (Morning vs. Evening) and calculate the percentage surcharge applied during adverse weather conditions.
- **Why it matters: This helps in understanding peak demand patterns and how the algorithm adjusts to driver** scarcity during rain or late hours.

3. Service Tier Segmentation
By categorizing rides into **Standard**, **Premium**, and **Pool**, we measure the "Price Gap."

- **Premium %:** Shows the luxury markup.
- **Pool Discount %:** Measures the incentive given to users for sharing a ride.

4. Model Reliability (MAE)
Finally, the MAE serves as our "Confidence Interval." If the MAE is 2.00$, it informs the reader that our predictions are, on average, within 2.00$ of the actual price, establishing the credibility of the report's findings.

Most Influential Pricing Factors:

Based on Linear Regression coefficients and domain analysis:

Distance (The Price Foundation)

- **Coefficient: +2.49**

- **Impact:** Each additional mile increases the price by approximately **$2.49**.
- **Insight:** Distance is the primary price determinant, confirming that mileage-based operational costs are the bedrock of the Uber/Lyft pricing model.

Surge Multiplier

- **Coefficient: +14.65**
- **Impact:** A single unit increase in the surge multiplier (e.g., from 1.0 to 2.0) adds **$14.65** to the fare.
- **Insight:** Driver scarcity has a financial weight nearly seven times more violent than the actual distance traveled.

Service Type

- **Premium Services:** Options like Black and Lux command a **134.7%** higher price than standard services.
- **Shared Rides:** Pool services offer a systematic discount of **23.2%**.

7.3 Practical Implications

7.3.1 For Riders

Recommendation 1: Timing Optimization

- **Avoid:** Rush hours (7-9 AM, 5-7 PM) when possible
- **Prefer:** Mid-morning (10 AM-12 PM) or early afternoon (2-4 PM)

Recommendation 2: Weather Awareness

- Check weather before requesting rides
- Book in advance during predicted rain/snow
- Consider alternative transportation during extreme weather

Recommendation 3: Service Selection

- Use standard services (UberX, Lyft) for routine trips
- Reserve premium services for special occasions
- Consider shared rides (Pool) for significant savings on flexible schedules

Recommendation 4: Distance Considerations

- Very short rides (<1 mile) may not be cost-effective
- Medium-distance rides (2-5 miles) offer best value
- For long distances (>15 miles), compare with other transportation options

7.4 Model Limitations

7.4.1 Data Limitations

Limitation 1: Temporal Coverage

- Dataset limited to November-December 2018
- Seasonal patterns may not generalize to summer months
- No data on major events or holidays

Limitation 2: Geographic Scope

- Boston-specific patterns may not apply to other cities
- Different cities have unique geography and demographics
- Traffic patterns vary significantly by region

● 8. CONCLUSION

8.1 Project Summary

This project successfully showed how Artificial Intelligence can predict Uber ride prices in Boston. By analyzing more than 690,000 trips and creating new categories (like "rush hour" or "weather"), we built a tool capable of understanding pricing trends. We proved that even though distance is the most important factor, the price changes significantly based on the time of day and the type of service chosen (Standard vs. Luxury). According to our tests, our KNN model achieved excellent results with an average error of only MAE.

8.2 Successes and Challenges

The most successful part was the data preparation. Creating specific indicators for "rain" or "rush hour" allowed the model to be much smarter. Another success was using two different models: one for high accuracy (KNN) and one to explain the numbers clearly (Regression).

However, not everything was easy. The biggest challenge was the huge size of the dataset. My computer took a long time to run the calculations, and I had to learn how to optimize my code so it wouldn't crash the machine. Also, I noticed that weather influenced prices less than I originally expected; it is truly the time of day that makes the biggest difference.

8.3 Lessons Learned

This project taught me a lot about the "data life cycle." I realized that the most important part is not just running the model, but cleaning the data thoroughly at the start. If the data is bad, the results will be bad.