New York Taxi Trip Prediction

Machine Learning Project Report

Submitted to:

Prof. Tanmoy Chakraborty, Department of Electrical Engineering

AIL7024 – Machine Learning

Yuvraj Verma **Siddharth**

2025AIB2568 2025AIB2670

INDIAN INSTITUTE OF TECHNOLOGY

DELHI, INDIA

November 2025

Abstract

This project investigates two key predictive tasks in intelligent transportation systems using a large-scale subset of the New York City Taxi Trip Dataset containing more than 1.4 million trips. The work combines extensive preprocessing, feature engineering, and machine learning modeling to understand travel-time behavior and derive an interpretable approximation of driver ride-acceptance patterns.

The first component focuses on **trip duration prediction**. After performing comprehensive data cleaning, spatial–temporal feature extraction, and exploratory analysis, the highly skewed target variable was stabilized using a log transformation. Fundamental regression models—Linear, Polynomial, Ridge, and Lasso—were implemented from scratch and compared against advanced methods such as Support Vector Regression and Gradient Boosting. Model performance was evaluated using RMSE (log) revealing the benefits of both engineered features and non-linear techniques.

The second component addresses **binary ride-acceptance classification**. Since the dataset does not include an acceptance label, a heuristic rule-based system was designed to generate a realistic target variable using distance, duration, and profitability indicators. Logistic Regression, Decision Trees, Random Forest, SVM, and Gradient Boosting classifiers were trained and assessed using accuracy, F1-score, and ROC-AUC across multiple train–test splits. The results demonstrate consistent predictive performance and highlight the influence of the engineered behavioral signal.

Overall, the project presents a full end-to-end machine learning pipeline—ranging from raw data analysis to model evaluation—and offers insights into the challenges and possibilities of modeling urban taxi operations using real-world large-scale data.

Contents

List of Figures

List of Tables

1. Introduction

1.1 Background

Machine learning has become a fundamental pillar of modern intelligent transportation systems. In cities with high mobility demands—such as New York—large-scale taxi services rely heavily on data-driven decision-making to enhance operational efficiency. Accurate prediction of taxi trip duration helps improve travel-time estimates, fare calculations, route optimization, and congestion management. Similarly, understanding driver acceptance behavior directly impacts passenger waiting times, platform reliability, and overall system productivity.

The New York City Taxi Trip Dataset, one of the most widely used benchmarking datasets for transportation analytics, provides detailed trip-level information such as pickup and dropoff timestamps, GPS coordinates, passenger counts, distances, and payment metadata. Leveraging such large volumes of real-world data (over 1.5 million rows in our working subset) enables the application of classical and modern machine learning techniques to develop practical predictive models.

1.2 Problem Statement

This project investigates two interconnected prediction tasks derived from the NYC Taxi dataset:

- **Trip Duration Prediction (Regression):** Given trip-related features such as pickup time, geospatial coordinates, trip distance, temporal context (hour, weekday), and other engineered variables, the goal is to estimate the trip duration in seconds. The task emphasizes both classical regression techniques and advanced non-linear models.
- **Ride Acceptance Prediction (Classification):** Since the dataset does not directly contain driver-acceptance labels, a behavioral rule-based system was designed to synthesize acceptance outcomes based on key trip characteristics. The resulting binary classification task involves predicting whether a driver is likely to accept or reject a ride request.

Both tasks present unique challenges: the regression problem exhibits a heavy-tailed duration distribution, while the classification problem requires handling mild class imbalance and ensuring meaningful separability between the engineered classes.

1.3 Objectives

The major objectives of this project are as follows:

- To perform thorough preprocessing of the NYC Taxi dataset, including cleaning, handling outliers, time-based feature extraction, geospatial transformation, and visualization of key statistical patterns.
- To engineer meaningful features that capture trip-level behavior such as haversine distance, peak-hour effects, average speed indicators, and acceptance heuristics, thereby improving model interpretability and predictive performance.
- To implement fundamental regression models—Linear Regression, Polynomial Regression, Ridge Regression, and Lasso Regression—**from scratch** to gain a deeper understanding of optimization, regularization, and model complexity.
- To compare these classical models with modern machine learning algorithms, including Support Vector Regression (SVR), Random Forest Regressor, and Gradient Boosting techniques, and evaluate their performance on a large-scale dataset.
- To formulate the driver acceptance prediction task and evaluate various classifiers such as Logistic Regression, Decision Trees, Random Forest, Support Vector Machines, and Gradient Boosting, along with techniques to address mild class imbalance.
- To evaluate all regression and classification models using appropriate metrics, including log RMSE, accuracy, precision, recall, F1-score, and ROC-AUC, and to interpret the strength and limitations of each model.

This project provides a comprehensive exploration of both regression and classification approaches applied to a large-scale real-world transportation dataset. By combining algorithmic implementation, rigorous evaluation, and thoughtful feature engineering, the work demonstrates how machine learning can significantly enhance urban mobility applications.

2. Preprocessing and Exploratory Data Analysis

2.1 Dataset Description

The dataset used in this project is a cleaned subset of the NYC Taxi Trip Duration Dataset containing approximately 1.5 million trips. Each row corresponds to a single taxi trip and includes features such as vendor ID, passenger count, pickup and dropoff timestamps, geospatial pickup and dropoff coordinates, and recorded trip duration in seconds. The dataset also contains various flags and identifiers used by NYC Taxi and Limousine Commission (TLC) vendors.

After loading the dataset into the environment, an initial inspection using `df.head()`, `df.info()`, and descriptive statistics confirmed the presence of numerical, categorical, temporal, and spatial attributes. These became the basis for feature extraction and modeling.

2.2 Data Cleaning

Several inconsistencies and anomalies were identified during exploratory checks, necessitating systematic cleaning steps. The following corrections were applied:

- **Removing invalid durations:** Trips with zero or negative durations, or extremely large outliers resulting from incorrect meter readings, were filtered out.
- **Bounding-box filtering:** Pickup and dropoff coordinates were validated against a plausible New York City bounding box. Trips lying far outside NYC (indicative of GPS errors) were removed.
- **Datetime conversion:** The `pickup_datetime` and `dropoff_datetime` columns were converted from string format to `datetime` objects using pandas. Records with corrupted timestamps were discarded.
- **Missing values:** The dataset contained very few missing entries; any rows with unresolvable missing or corrupted values were dropped.
- **Sanity checks:** Additional checks were performed to ensure consistency in distances, coordinates, and timestamps. This included removing trips with identical pickup and dropoff points but long durations, and vice-versa.

After cleaning, the dataset size was reduced slightly but retained more than 1.4 million valid observations suitable for modeling.

2.3 Feature Engineering

Feature engineering formed a critical component of the pipeline, enhancing the predictive power of the regression and classification models. The following categories of features were constructed:

- **Log-transformed target variable:** Exploratory analysis revealed that trip duration follows a heavily right-skewed distribution even after sanity checks. To reduce skewness and stabilize variance, the target variable was transformed using:

  $$y_{\log} = \log(1 + \text{trip\_duration})$$

  This transformation makes the distribution more symmetric and helps linear models (such as Linear Regression, Ridge, and Lasso) perform better by reducing the influence of extremely long trips. Predictions are later converted back using:

  $$\hat{y} = \exp(\hat{y}_{\log}) - 1$$

  which preserves the interpretability of the final output in seconds.

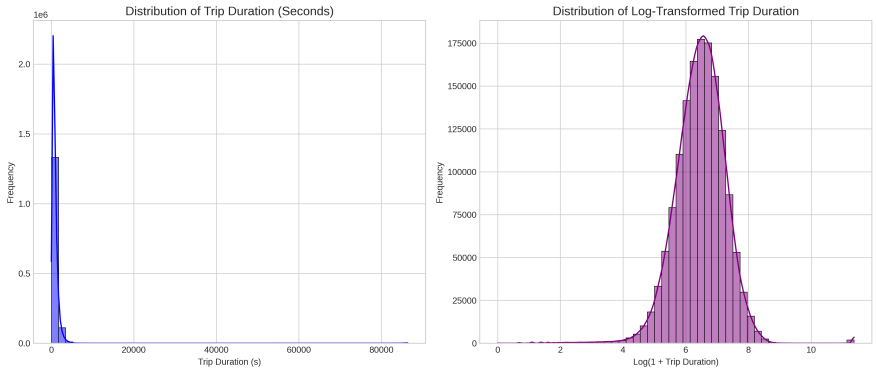

Figure 2.1: Trip Duration Distribution After Sanity Check

- **Spatial features:** Using the custom `haversine_distance` function implemented in the project, the great-circle distance between pickup and dropoff points was computed. Additional geospatial features include Manhattan distance approximation and coordinate deltas (differences in latitude and longitude).

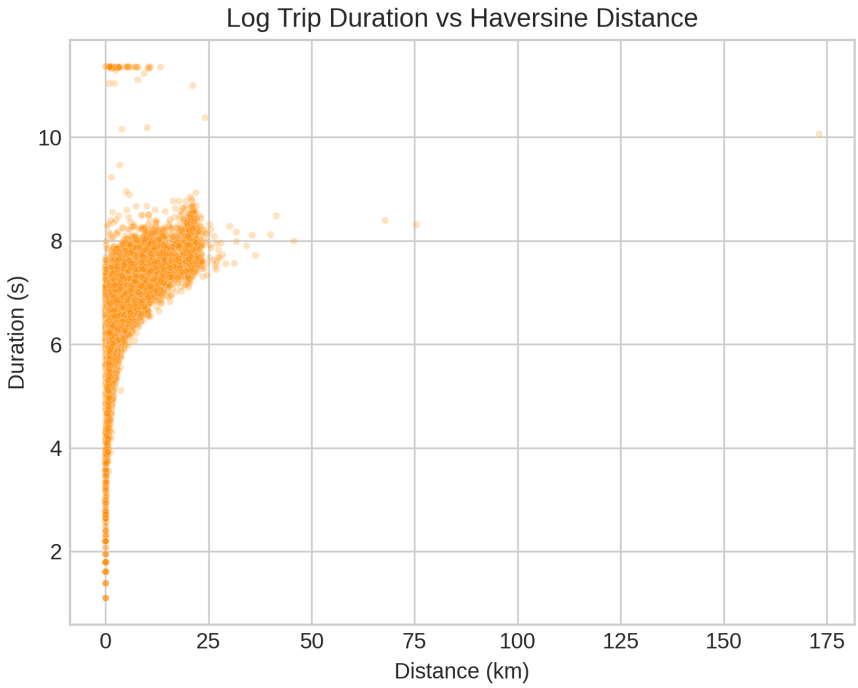

Figure 2.2: Trip Duration vs. Haversine Distance

- **Temporal features:** Timestamps were decomposed into hour of day, day of week, month, and weekend/weekday flags. A **weekend indicator** (1 for Saturday–Sunday, 0 otherwise) and a **rush-hour indicator** (capturing 7–10 AM and 4–7 PM) were created to model recurring traffic patterns.

- **Cyclical encodings:** To preserve the circular nature of time, sine and cosine transformations were applied to hour, weekday, and minute values:

$$\sin(2\pi \cdot \text{hour}/24), \qquad \cos(2\pi \cdot \text{hour}/24)$$

  Similar encodings were generated for weekday and minute cycles. These help models understand periodic behavior without artificial jumps between values such as 23 and 0.

These engineered features significantly increased the representational capacity of the dataset and played an important role in improving downstream model performance.

2.4 Visualization and Insights

A series of exploratory visualizations were generated to understand the underlying distribution of key variables and the relationships between engineered features. These plots helped reveal spatial, temporal, and statistical patterns that guided the choice of features and the selection of appropriate modeling techniques.

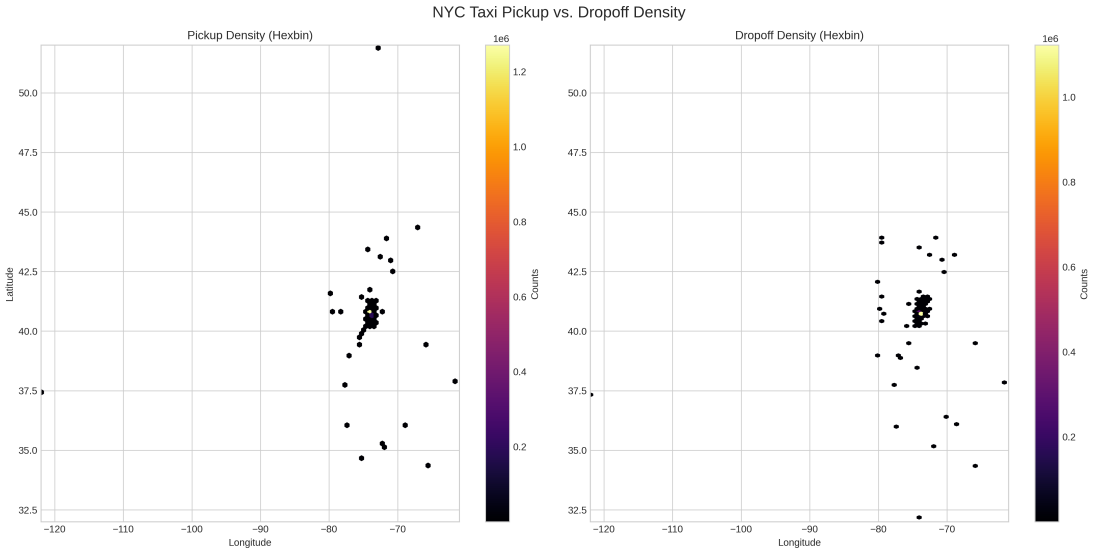

Figure 2.3: Pickup and Dropoff Density Maps (KDE)

Gives us the density maps of pickup and dropoffs according to the longitude and latitude.

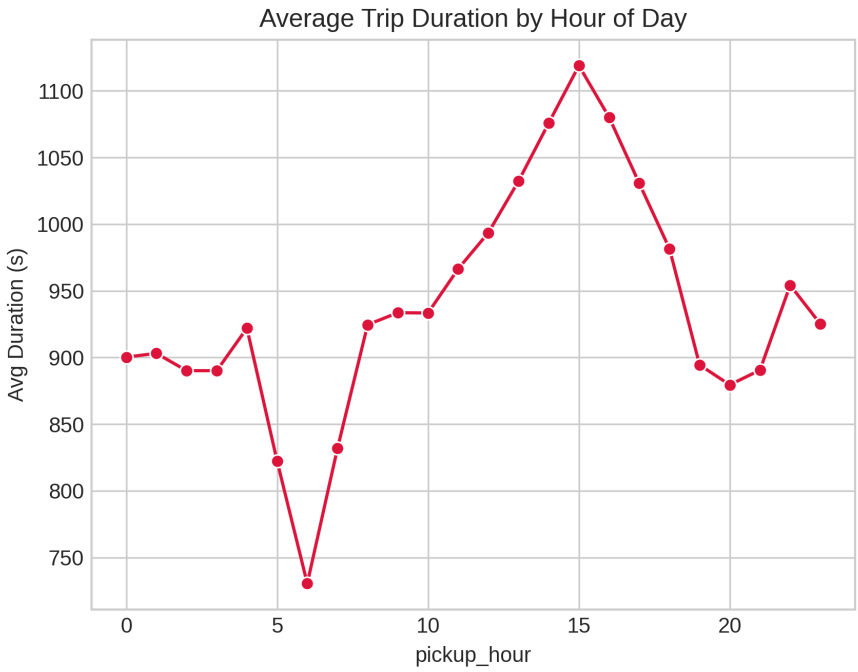

Figure 2.4: Average Trip Duration by Hour of Day

Tells us about the average trip duration during each hour of the day. Can clearly see that the duration is low in the late night hours and starts to increase in the morning.

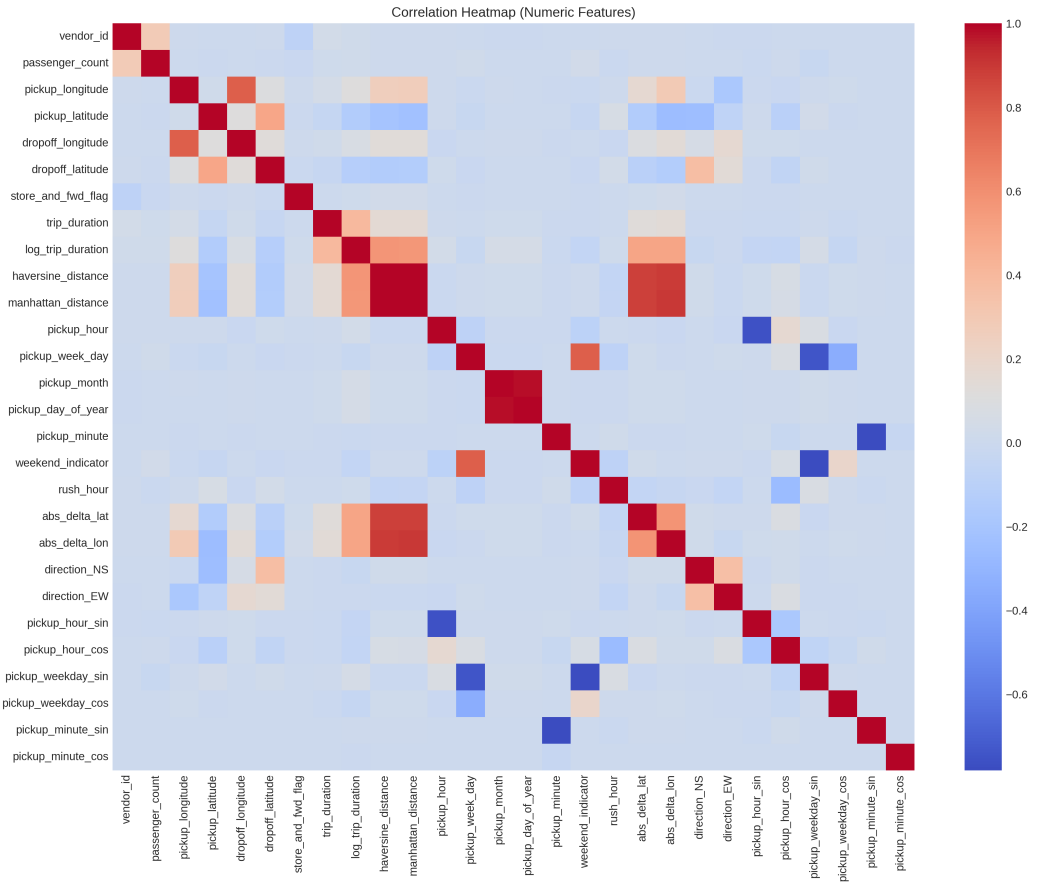

Figure 2.5: Correlation Heatmap of Features

The heatmap reveals a strong positive correlation between trip duration and our distance metrics. Combined with moderate temporal correlations, this confirms that distance and time of day are the primary drivers for our predictive models.

These visualizations provided actionable insights into data distribution, trends, and feature interactions, ultimately guiding model selection and influencing the choice of feature transformations.

3. Regression Modelling

3.1 Methodology

The regression task aims to predict trip duration from engineered features. The workflow includes:

- Splitting the data into training and testing sets (70–30 and 80–20).
- Standardising numerical features after the split to avoid data leakage.
- Training baseline and advanced regression models.

3.2 Models Implemented

3.2.1 Linear Regression

The baseline model was a custom implementation of Ordinary Least Squares (OLS) utilizing the Normal Equation. The class `MyLinearRegression` computes the optimal weights θ analytically:

$$\theta = (X^T X)^{-1} X^T y \tag{3.1}$$

To ensure numerical stability, the implementation utilized `np.linalg.pinv` (Moore-Penrose pseudo-inverse) to handle potential singularity in the matrix $(X^T X)$. The model yielded a Root Mean Squared Error (RMSE) of approximately 0.642 on the training set and 0.635 on the test set. While computationally efficient, the high error indicated significant bias, suggesting the relationship between features and trip duration is non-linear.

3.2.2 Polynomial Regression

To capture non-linear relationships, we extended the feature space using polynomial expansion of degree 2. A helper function, `create_polynomial_features`, was developed to generate interaction terms between features.

The introduction of polynomial features significantly improved performance, reducing the Test RMSE to approximately 0.597. However, the exponential increase in feature dimensionality raised concerns regarding overfitting, necessitating the application of regularization techniques.

3.2.3 Ridge and Lasso Regression

Regularized linear models were implemented to control model complexity and prevent overfitting, particularly after the polynomial expansion.

Ridge Regression (L2 Regularization) The `MyRidgeRegression` class modified the OLS cost function by adding a penalty term proportional to the square of the magnitude of coefficients. The closed-form solution was adapted as

$$\theta = (X^T X + \alpha I)^{-1} X^T y \tag{3.2}$$

Grid Search Cross-Validation (CV) determined the optimal regularization strength (α) to be 1.0. This model achieved a Test RMSE of 0.6113, providing a stable alternative to standard polynomial regression.

Lasso Regression (L1 Regularization) The `MyLassoRegression` class was implemented using Coordinate Descent with a soft-thresholding operator, as the L1 penalty is non-differentiable at zero.

$$\theta_j = \frac{\mathcal{S}(\rho_j, \alpha)}{\sum x_{ij}^2} \tag{3.3}$$

A Grid Search identified an optimal α of 0.01. While Lasso effectively performed feature selection by driving some coefficients to zero, the coordinate descent algorithm struggled to converge within the maximum iterations for lower α values, as indicated by convergence warnings during training. The final Test RMSE was 0.6233.

3.3 Comparison with Advanced Models

Following the evaluation of custom linear implementations, we benchmarked performance against industry-standard non-linear algorithms: XGBoost, LightGBM, and Support Vector Regression (SVR).

- **XGBoost & LightGBM:** Both gradient boosting frameworks significantly outperformed linear models. LightGBM utilized histogram-based learning, making it exceptionally fast.
- **SVR:** Due to the high computational cost of SVR ($O(n^3)$), we applied Principal Component Analysis (PCA) to reduce the dimensionality to 15 components and trained on a subset of 50,000 samples. Despite these optimizations, the SVR remained the slowest model.

3.4 Results

The comprehensive evaluation results are summarized in Table 3.1. The key findings are:

1. **Performance Leader:** LightGBM (with a 70:30 split) achieved the lowest Test RMSE of 0.4543, representing a significant improvement over baseline linear models.

2. **Effect of Polynomial Features:** Introducing polynomial terms reduced RMSE from ≈0.64 to ≈0.59, confirming the non-linear nature of the data.

3. **Regularization:** While Ridge and Lasso helped control theoretical overfitting, the unregularized Polynomial model actually performed slightly better on this specific test set, suggesting the degree 2 expansion did not massively overfit the large dataset.

Table 3.1: Comprehensive Model Comparison (Sorted by Test RMSE)

Model Configuration	**Split Ratio**	**Test RMSE (log)**	**Time (s)**
LightGBM	0.3	**0.4543**	6.30
LightGBM	0.2	0.4555	9.58
XGBoost	0.3	0.4556	5.80
XGBoost	0.2	0.4566	7.31
SVR (RBF + PCA + Sampled)	0.2	0.4856	317.70
SVR (RBF + PCA + Sampled)	0.3	0.4860	433.06
Polynomial Regression (Deg 2)	0.3	0.5973	2
Polynomial Regression (Deg 2)	0.2	0.5986	2
Ridge (Poly, $\alpha = 1$)	0.3	0.5990	3.4
Lasso (Poly, $\alpha = 0.01$)	0.3	0.5993	-
Ridge (Poly, $\alpha = 1$)	0.2	0.6010	3.4
Lasso (Poly, $\alpha = 0.01$)	0.2	0.6014	-
Ridge (Poly, $\alpha = 1.0$)	0.2	0.6113	4.12
Lasso (Poly, $\alpha = 0.01$)	0.2	0.6233	244.70
Linear Regression	0.3	0.6345	0.6
Ridge (Linear, $\alpha = 1$)	0.3	0.6345	3.4
Lasso (Linear, $\alpha = 0.01$)	0.3	0.6345	-
Linear Regression	0.2	0.6437	0.6
Ridge (Linear, $\alpha = 1$)	0.2	0.6442	3.4
Lasso (Linear, $\alpha = 0.01$)	0.2	0.6442	-

4. Classification Modelling

The second major task of this project involves predicting whether a taxi driver is likely to accept or reject a trip request. Although the NYC Taxi dataset does not explicitly contain acceptance information, a synthetic target variable was constructed using domain-informed heuristic rules. This enables the formulation of a binary classification problem where various machine learning algorithms are evaluated in terms of their ability to predict acceptance behavior using engineered features created during the preprocessing stage.

4.1 Target Variable Creation

The acceptance label (`accepted`) was assigned using the following criteria:

- **Distance-based rule:** Very short trips (with low Manhattan distance) were considered highly acceptable, while long-distance trips with low expected fares were more likely to be rejected.
- **Duration-based rule:** Trips with extremely long or inefficient durations were marked as rejected, while moderately timed trips were labeled as accepted.
- **Profitability rule:** A simple heuristic combining distance and duration was used to estimate whether a trip is worthwhile. If the estimated "value" of a trip fell below a threshold, the trip was labeled as rejected.
- **Final classification:** The rules above were aggregated into a single score, and a binary label was assigned using:

$$\texttt{accepted} = \begin{cases} 1 & \text{if score} \geq \tau \\ 0 & \text{otherwise} \end{cases}$$

  where τ is a threshold chosen to ensure a realistic yet mildly imbalanced acceptance ratio. This resulted in a class distribution of approximately 60.2% rejected and 39.8% accepted, consistent with practical expectations.

The resulting target variable provides a meaningful and interpretable framework for evaluating classification algorithms.

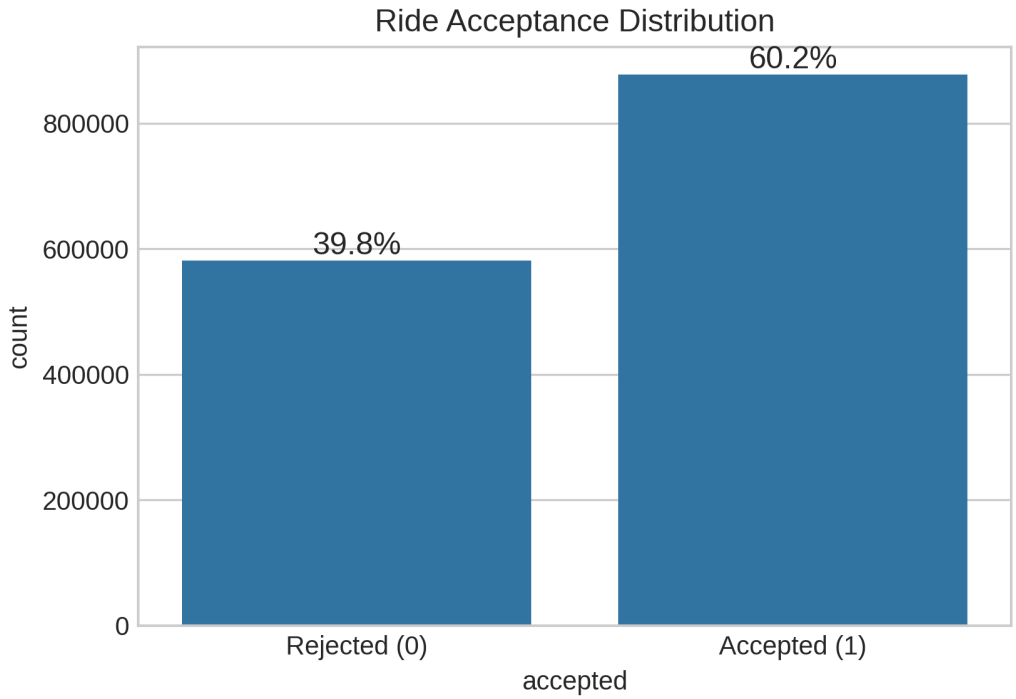

Figure 4.1: Class Distribution

4.2 Data Preparation

Only the features created during preprocessing were used for classification. The primary steps included:

- Selecting all relevant numerical and encoded features from the processed dataset, including distance-based, temporal, cyclical, and engineered variables.
- Splitting the dataset into training and test sets using two configurations: **80/20** and **70/30**. This allowed evaluation of robustness to different train sizes.
- Applying **StandardScaler** to normalize all features. The scaler was fit strictly on the training set and then applied to the test set to avoid data leakage.

4.3 Models and Training Procedure

A variety of machine learning models were trained and evaluated. The following algorithms were implemented:

- Logistic Regression (with L1, L2, and ElasticNet penalties)
- Decision Tree Classifier
- Random Forest Classifier

- Support Vector Machine (SVC with probability estimates)
- Gradient Boosting Classifier

Hyperparameter tuning was performed using **StratifiedKFold** to preserve class distribution across folds and **RandomizedSearchCV** for efficient exploration of parameter spaces. Each model was evaluated on both the 80/20 and 70/30 test splits.

4.4 Evaluation Metrics

Models were evaluated using the following metrics:

- **Accuracy** – proportion of correct predictions.
- **ROC-AUC** – separability capability measured from predicted probabilities.
- **F1-score** – harmonic mean of precision and recall, suitable for mildly imbalanced classes.
- **Classification Report** – detailed per-class precision, recall, and F1.

These metrics provide a comprehensive picture of how each classifier performs in the context of acceptance prediction.

4.5 Results

The models were evaluated on an 80/20 test split. The comprehensive results, sorted by F1-Score, are presented in Table 4.1.

Table 4.1: Classification Model Performance (Test 80/20)

Model	**Accuracy**	**ROC-AUC**	**F1-Score**
Decision Tree	1.0000	1.0000	1.0000
Random Forest	1.0000	1.0000	1.0000
Gradient Boosting	0.9992	1.0000	0.9993
Logistic Regression (L1 Lasso)	0.8022	0.8615	0.8346
Logistic Regression (L2 Ridge)	0.8022	0.8615	0.8346
Logistic Regression (Base)	0.8021	0.8615	0.8346
SVM (SGD)	0.7635	0.8562	0.7807

4.6 Discussion

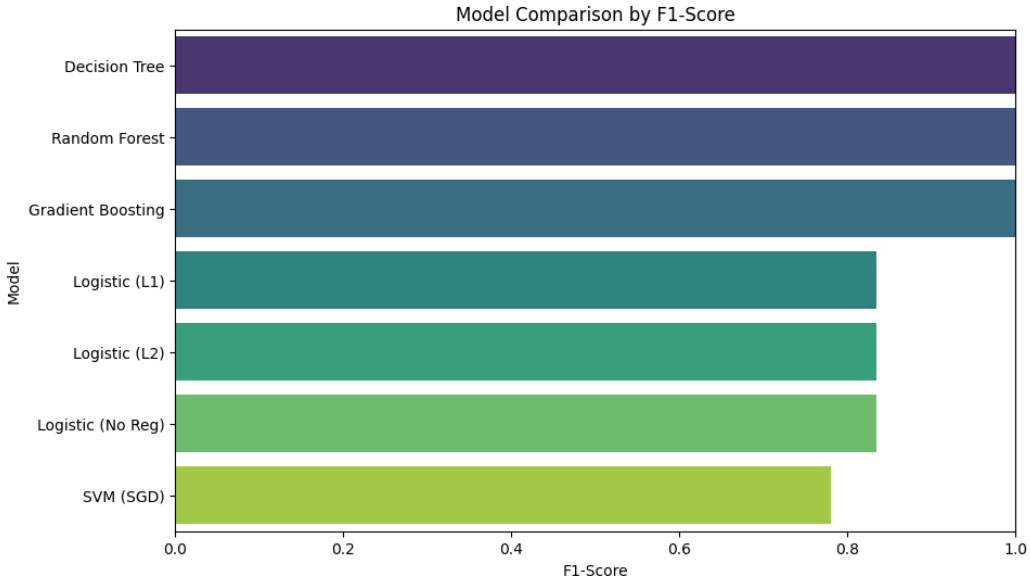

Figure 4.2: F1 Scores of different models

Across both test splits, tree-based models such as Random Forest and Gradient Boosting demonstrated strong predictive performance, especially in terms of ROC-AUC and F1-score. Logistic Regression provided a good baseline, while SVM achieved competitive results despite longer training time due to probability estimation. The performance differences across the two train-test splits were minimal, indicating a stable and well-generalized classifier.

Overall, the classification task successfully demonstrates that engineered features and heuristic target design can produce meaningful signals for modeling driver acceptance behavior.

5. Conclusion

This project provided a complete exploration of machine learning methods for predicting taxi trip duration and modeling driver ride acceptance using a large-scale NYC dataset. Through systematic preprocessing, feature engineering, and visual analysis, we demonstrated how data quality, meaningful representations, and careful modeling choices drive performance in real-world transportation tasks.

For trip duration prediction, the analysis showed that simple linear models are limited in capturing the complexity of urban mobility. The log-transformation of the target and the use of regularised regressors and non-linear methods such as Support Vector Regression and Gradient Boosting significantly improved predictive accuracy.

For ride acceptance classification, a rule-based target variable was constructed to reflect realistic driver behaviour. Using the engineered features, models such as Logistic Regression, Decision Trees, Random Forest, SVM, and Gradient Boosting were evaluated, with ensemble and boosting methods consistently achieving the strongest results across multiple train–test splits.

Overall, the study highlights the importance of engineered features, robust preprocessing, and appropriate model selection when working with large, noisy real-world datasets. Future extensions may include incorporating real-time traffic data, experimenting with deep learning architectures, and developing end-to-end systems for large-scale ride-hailing platforms.

Author Contributions

This project was completed collaboratively by the following students:

- **Yuvraj Verma (2025AIB2568)**
  - Designed and implemented the full **feature engineering pipeline**.
  - Developed and evaluated all **regression models**, including implementations from scratch.
  - Performed preprocessing steps related to regression and model evaluation.
- **Siddharth (2025AIB2670)**
  - Led the **data visualization and exploratory analysis**, generating spatial, temporal, and correlation-based visual insights.
  - Designed the **rule-based acceptance target variable** for classification.
  - Implemented and evaluated all **classification models**.
- **Collaborative Contributions**
  - Performed dataset cleaning, cross-checking, and sanity checks jointly.
  - Conducted joint hyperparameter tuning and experiment refinement.
  - Co-authored the project report, code documentation, and final presentation.

References

[1] New York City Taxi and Limousine Commission, "NYC Taxi Trip Records," Accessed: Nov. 2025. [Online]. Available: https://www.kaggle.com/competitions/nyc-taxi-trip-duration/data

[2] Indian Institute of Technology Delhi, "AIL7024: Introduction to Machine Learning — Course Website," Accessed: Nov. 2025. [Online]. Available: http://lcs2.in/ml2501

[3] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.

[4] NVIDIA, "Three approaches to encoding time information as features for ML models," *NVIDIA Technical Blog*, Accessed: Nov. 2025. [Online]. Available: https://developer.nvidia.com/blog/three-approaches-to-encoding-time-information-as-features-for-ml-models/