INSTITUTE FOR ADVANCED COMPUTING & SOFTWARE DEVELOPMENT, AKURDI, PUNE

Predict the Fare Amount of Future Rides Using Regression Analysis

PG - DBDA March 2024

Submitted By:

Group No: 07

Roll No.	Name.
243514	Dhruva Rakesh B
243552	Sumedh Bansod

Mr. Abhijit Nagargoje	**Mr. Rohit Puranik**
Project Guide	Centre Coordinator

ABSTRACT

Predicting the fare amount of future rides is a critical task for transportation and ride-hailing services, enabling more accurate pricing strategies and enhancing customer satisfaction. This study explores the application of various regression analysis techniques to predict ride fares based on historical trip data. By leveraging features such as pickup and drop-off locations, trip duration, time of day, and passenger count, the research aims to develop a model that accurately forecasts fare amounts. Several regression models, including linear regression, polynomial regression, decision trees, and ensemble methods like random forests and gradient boosting, are evaluated for their predictive performance. The study also examines the presence of non-linear relationships within the dataset, using visual inspections, residual analysis, and polynomial transformations. The results indicate that while simple linear models offer a baseline for prediction, more complex models like random forests and gradient boosting capture the non-linarites and interactions among features, leading to improved accuracy. The findings underscore the importance of model selection and feature engineering in developing robust fare prediction systems, offering valuable insights for future research and practical implementations in the transportation industry.

ACKNOWLEDGEMENT

I take this occasion to thank God, almighty for blessing us with his grace and taking our endeavour to a successful culmination. I extend my sincere and heartfelt thanks to our esteemed guide, **Mr. Abhijit Nagargoje** for providing me with the right guidance and advice at the crucial juncture sand for showing me the right way. I extend my sincere thanks to our respected Centre Co-Ordinator **Mr. Rohit Puranik**, for allowing us to use the facilities available. I would like to thank the other faculty members also, at this occasion. Last but not the least, I would like to thank my friends and family for the support and encouragement they have given me during the course of our work.

Dhruva Rakesh B (240341225014)

Sumedh Bansod (240341225050)

Table of Contents

Introduction

In the rapidly evolving urban transportation landscape, accurate fare prediction is crucial for both service providers and customers. Ride-hailing platforms like Uber, Lyft, and traditional taxis depend on fare prediction models to determine pricing, manage demand, and optimize resource allocation. Effective fare prediction enhances user experience through transparent pricing and helps providers maintain competitive pricing while maximizing profitability. This project focuses on predicting future ride fares using regression analysis—a statistical method that models the relationship between fare amounts and features such as pickup and dropoff locations, time of day, and passenger count. We will explore and compare several regression models, from linear regression, which assumes a straightforward relationship between features and fare, to more complex models like decision trees, random forests, and gradient boosting, which capture non-linear relationships and feature interactions. The study will also examine feature engineering, such as calculating the distance between pickup and dropoff points, and the importance of addressing data non-linearities. By developing and evaluating these models, the project aims to create a robust fare prediction system for ride-hailing applications, leading to more accurate and fair pricing, improved resource management, and enhanced customer satisfaction.

Objective

The task is to predict future ride fares using regression analysis, aiming to help ride-sharing companies optimize pricing strategies, offer transparency to riders, and ensure fair compensation for drivers. Fare amounts are influenced by factors such as distance, duration, traffic conditions, time of day, and demand. To address this, we will develop a regression model that predicts fare amounts based on these factors, utilizing a dataset with historical ride data, including fare amounts and relevant features. The project involves building and training a regression model on this historical data, evaluating it with appropriate metrics, and providing insights and recommendations based on the model's predictions.

Dataset Description:

The dataset was provided to us from Kaggle by Uber through Mentormind.

Kaggle is a leading platform for data science and machine learning, offering a vibrant community where users can participate in competitions, explore diverse datasets, and share code through notebooks. It provides a collaborative environment for solving real-world problems with data, allowing both beginners and experts to enhance their skills, engage in projects, and contribute to a global data science community. Kaggle also features educational resources, including tutorials and discussions, making it a comprehensive hub for learning and applying data science techniques.

Uber is a global technology company that provides a ride-hailing service through a mobile app, connecting passengers with drivers for on-demand transportation. Launched in 2009, Uber has expanded its offerings beyond rides to include food delivery through Uber Eats, and it has ventured into areas such as freight logistics and autonomous driving technology. By leveraging a user-friendly app and a network of drivers, Uber has revolutionized the way people commute, making transportation more accessible and efficient while operating in numerous cities around the world.

Description:

The project is about on world's largest taxi company Uber inc. In this project, we're looking to predict the fare for their future transactional cases. Uber delivers service to lakhs of customers daily. Now it becomes really important to manage their data properly to come up with new business ideas to get best results. Eventually, it becomes really important to estimate the fare prices accurately.

The dataset contains the following fields:

- key - a unique identifier for each trip
- fare_amount - the cost of each trip in USD
- pickup_datetime - date and time when the meter was engaged
- passenger_count - the number of passengers in the vehicle (driver entered value)
- pickup_longitude - the longitude where the meter was engaged
- pickup_latitude - the latitude where the meter was engaged
- dropoff_longitude - the longitude where the meter was disengaged
- dropoff_latitude - the latitude where the meter was disengaged

Platform Used:

Google Colab is a free, cloud-based platform that allows users to write and execute Python code in a collaborative, interactive environment. It provides an easy-to-use interface for running Jupyter notebooks, which can be used for data analysis, machine learning, and research, all within a browser. Colab integrates seamlessly with Google Drive for file storage and sharing, and it offers access to powerful computing resources, including GPUs and TPUs, making it a popular choice for both beginners and experienced data scientists to develop and share their code.

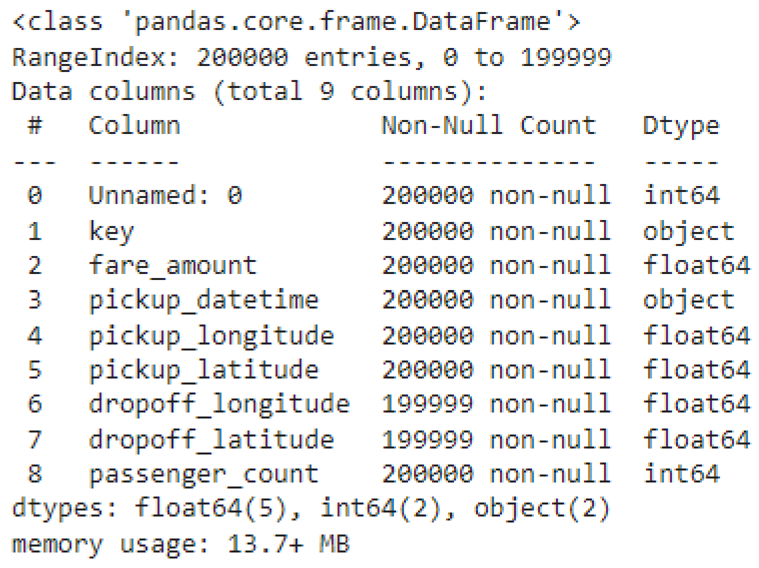
Data Preprocessing and Exploration

Load and Explore the Dataset

The first task is to load the dataset containing historical ride data. Take a look at the structure of the data, examine the features, and gain insights into the variables that may influence the fare amount. Perform basic exploratory data analysis (EDA) techniques to understand the distribution of the target variable (fare amount) and identify any patterns or outliers.

To work with files stored in Google Drive using Google Colab, first we will mount Google Drive to access the csv files. Next, we will define the path to the CSV file we want to work with and use Pandas to read the CSV file into a DataFrame. Create a copy of the DataFrame to ensure data integrity during analysis. Display the first few rows of the DataFrame to inspect its content, and generate a concise summary of the DataFrame to understand its structure and key statistics.

Summary of the dataframe:



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   Unnamed: 0         200000 non-null  int64  
 1   key                200000 non-null  object 
 2   fare_amount        200000 non-null  float64
 3   pickup_datetime    200000 non-null  object 
 4   pickup_longitude   200000 non-null  float64
 5   pickup_latitude    200000 non-null  float64
 6   dropoff_longitude  199999 non-null  float64
 7   dropoff_latitude   199999 non-null  float64
 8   passenger_count    200000 non-null  int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 13.7+ MB
```

Exploratory Data Analysis (EDA)

We will start by importing the necessary libraries and loading your dataset into a DataFrame. View the columns of the DataFrame and drop any invalid ones as needed. Check the dimensions of the DataFrame and determine the number of unique values for each column. Remove duplicate rows to clean your training dataset and identify and count missing values. For imputation, use the mean to fill in missing values for continuous data with a normal distribution and a small percentage of missing entries. If zeros are present, print rows with these values, and impute them with the median if the data contains outliers and the zeros are considered missing values. Convert the 'pickup_datetime' column from 'object' to 'datetime' to ensure proper handling. Finally, plot the distribution of the 'fare_amount' to visualize its spread.

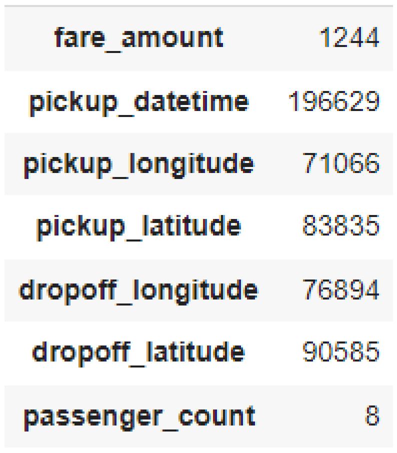
Columns of the dataframe:

```
Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count'],
      dtype='object')
```

Dimensions of the dataframe:

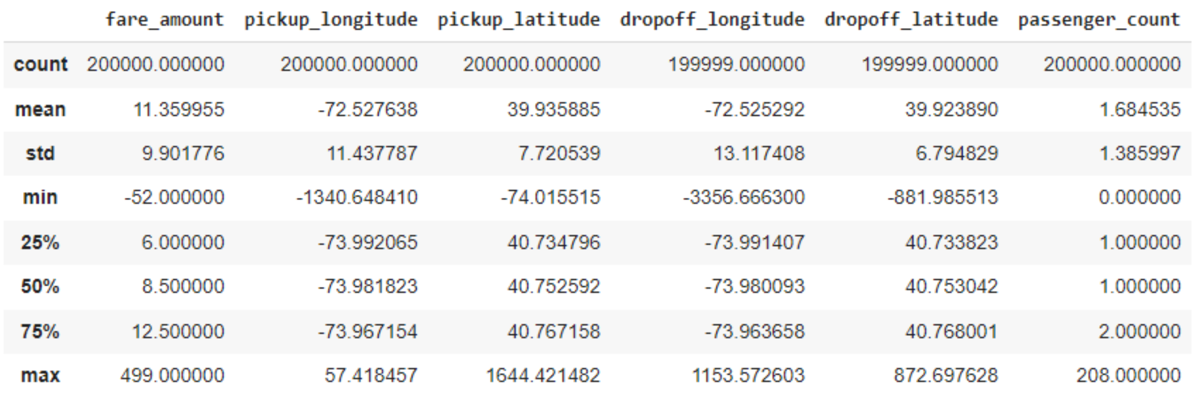
```
(200000, 7)
```

Unique values in a dataframe:



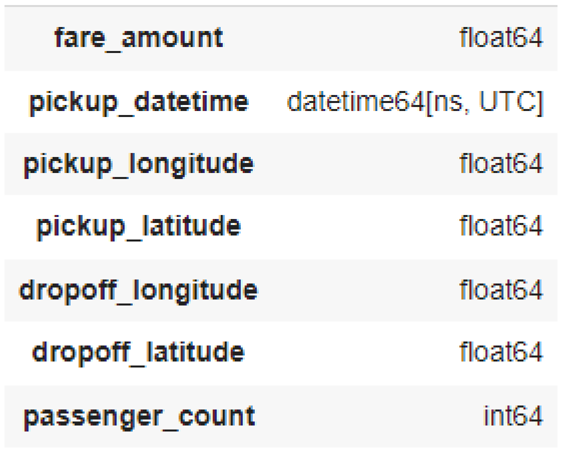
fare_amount	1244
pickup_datetime	196629
pickup_longitude	71066
pickup_latitude	83835
dropoff_longitude	76894
dropoff_latitude	90585
passenger_count	8

Statistical summary of the dataframe:



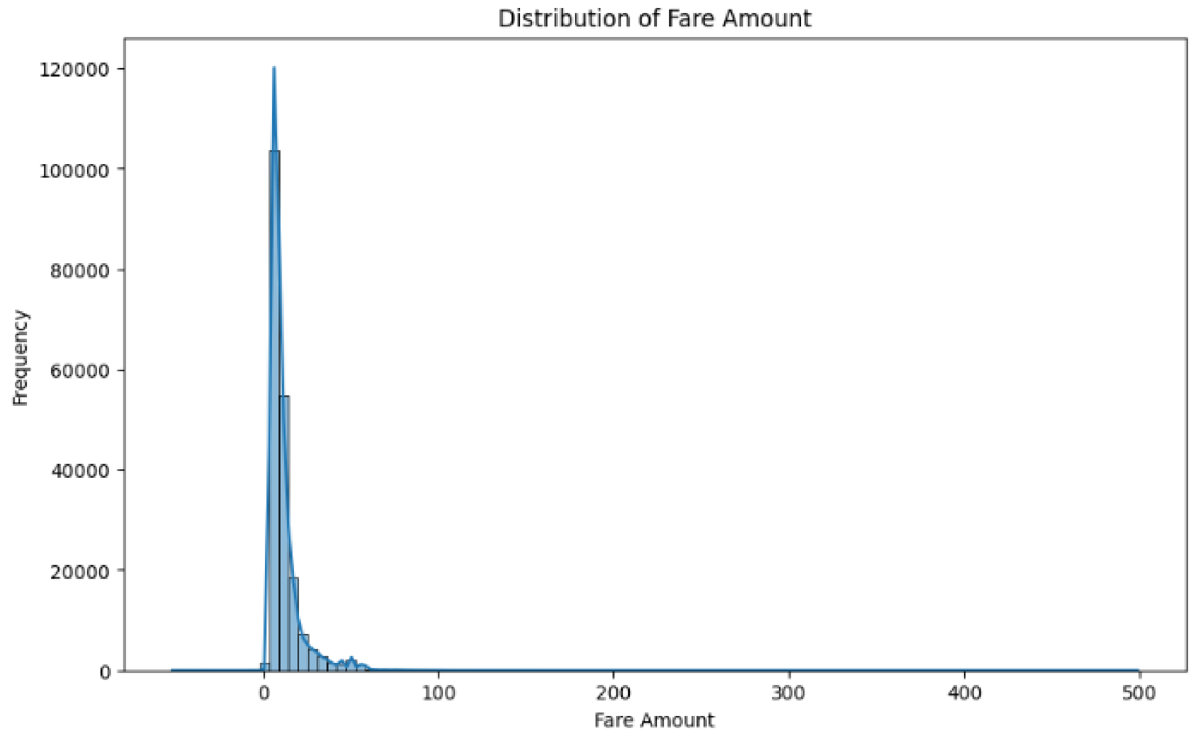
	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
count	200000.000000	200000.000000	200000.000000	199999.000000	199999.000000	200000.000000
mean	11.359955	-72.527638	39.935885	-72.525292	39.923890	1.684535
std	9.901776	11.437787	7.720539	13.117408	6.794829	1.385997
min	-52.000000	-1340.648410	-74.015515	-3356.666300	-881.985513	0.000000
25%	6.000000	-73.992065	40.734796	-73.991407	40.733823	1.000000
50%	8.500000	-73.981823	40.752592	-73.980093	40.753042	1.000000
75%	12.500000	-73.967154	40.767158	-73.963658	40.768001	2.000000
max	499.000000	57.418457	1644.421482	1153.572603	872.697628	208.000000

Updated datatypes of the dataframe:



fare_amount	float64
pickup_datetime	datetime64[ns, UTC]
pickup_longitude	float64
pickup_latitude	float64
dropoff_longitude	float64
dropoff_latitude	float64
passenger_count	int64

Distribution of fare_amount column:



Feature Extraction

Begin with feature extraction from the `pickup_datetime` column by deriving components such as Hour, Day, Month, Year, and Day of the week, which renders the original column redundant. Import the numpy library and define a function to calculate the Haversine distance between two GPS coordinates. Convert decimal degrees to radians and apply the Haversine formula to compute the distance. Additionally, extract a feature to indicate whether the date falls on a Weekend. Afterward, generate descriptive statistics for each column and transpose the results for a clear summary of the dataset's key metrics.

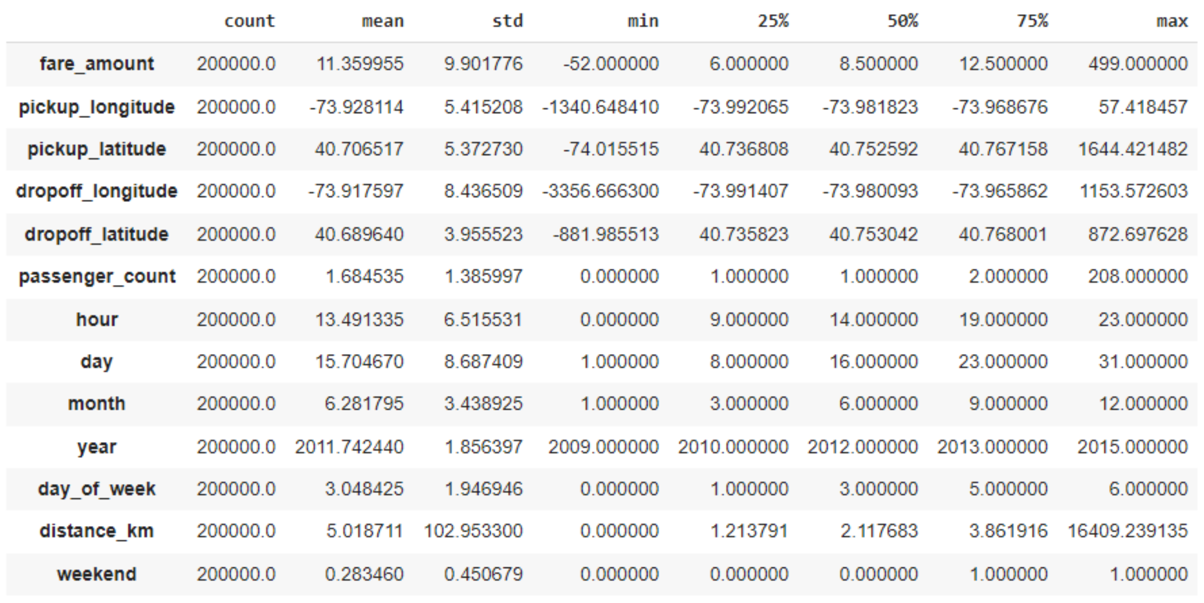
The Haversine formula calculates the distance between two points on the Earth's surface given their latitude and longitude. It accounts for the spherical shape of the Earth. Here’s the formula:

$$d = 2r \cdot \operatorname{asin}\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cdot\cos(\phi_2)\cdot\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

Where:

- d is the distance between the two points.
- r is the Earth's radius (mean radius = 6,371 km).
- ϕ1 and ϕ2 are the latitudes of the two points in radians.
- Δϕ is the difference in latitudes (i.e., ϕ2−ϕ1).
- Δλ is the difference in longitudes (i.e., λ2−λ1).

Descriptive statistics for each column with transpose:



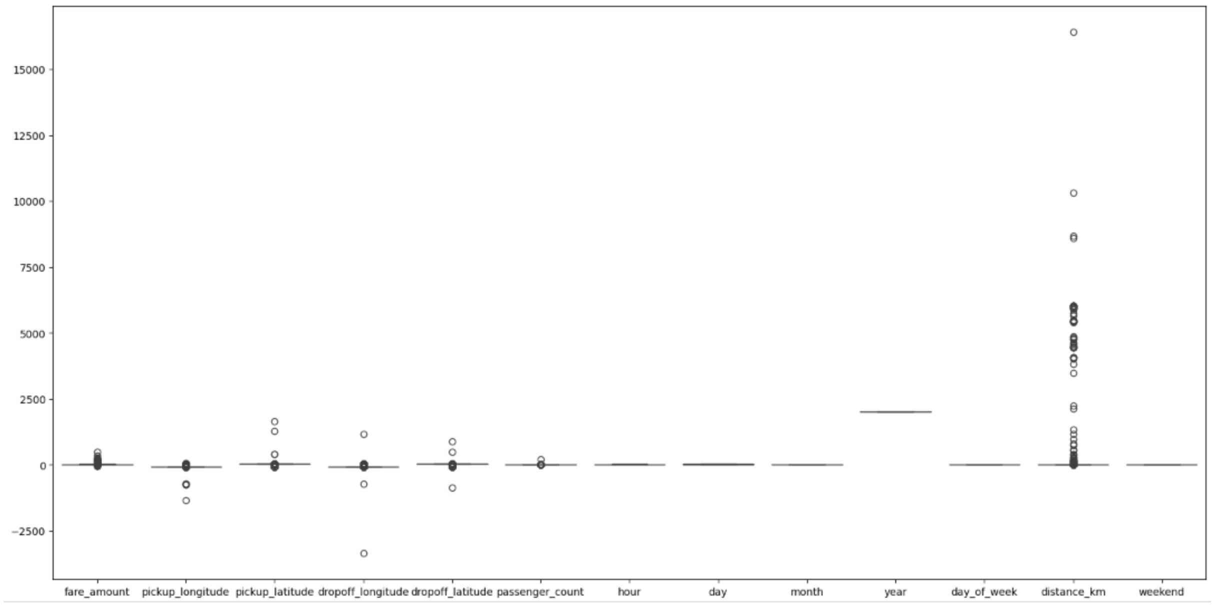
	count	mean	std	min	25%	50%	75%	max
fare_amount	200000.0	11.359955	9.901776	-52.000000	6.000000	8.500000	12.500000	499.000000
pickup_longitude	200000.0	-73.928114	5.415208	-1340.648410	-73.992065	-73.981823	-73.968676	57.418457
pickup_latitude	200000.0	40.706517	5.372730	-74.015515	40.736808	40.752592	40.767158	1644.421482
dropoff_longitude	200000.0	-73.917597	8.436509	-3356.666300	-73.991407	-73.980093	-73.965862	1153.572603
dropoff_latitude	200000.0	40.689640	3.955523	-881.985513	40.735823	40.753042	40.768001	872.697628
passenger_count	200000.0	1.684535	1.385997	0.000000	1.000000	1.000000	2.000000	208.000000
hour	200000.0	13.491335	6.515531	0.000000	9.000000	14.000000	19.000000	23.000000
day	200000.0	15.704670	8.687409	1.000000	8.000000	16.000000	23.000000	31.000000
month	200000.0	6.281795	3.438925	1.000000	3.000000	6.000000	9.000000	12.000000
year	200000.0	2011.742440	1.856397	2009.000000	2010.000000	2012.000000	2013.000000	2015.000000
day_of_week	200000.0	3.048425	1.946946	0.000000	1.000000	3.000000	5.000000	6.000000
distance_km	200000.0	5.018711	102.953300	0.000000	1.213791	2.117683	3.861916	16409.239135
weekend	200000.0	0.283460	0.450679	0.000000	0.000000	0.000000	1.000000	1.000000

Handling Outliers

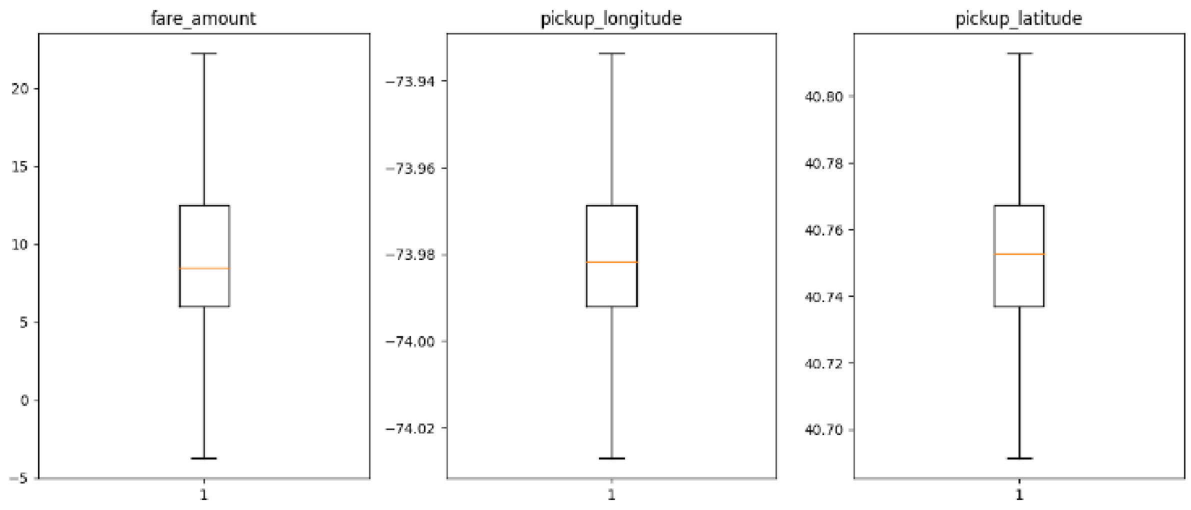
To address outliers in your dataset, start by identifying potential anomalies, such as amounts less than -52, trips with travel distances less than or equal to 0 or greater than 130 kilometers, and trips with latitude or longitude values outside their valid ranges (90 < latitude < -90 and 180 < longitude < -180). Begin by importing the necessary libraries and using a boxplot for initial visualization. Calculate quartiles and the interquartile range (IQR) to identify outliers. Impute these outliers by applying the lower and upper bounds for values that deviate significantly. Apply this imputation to the DataFrame and visualize the results using a boxplot.

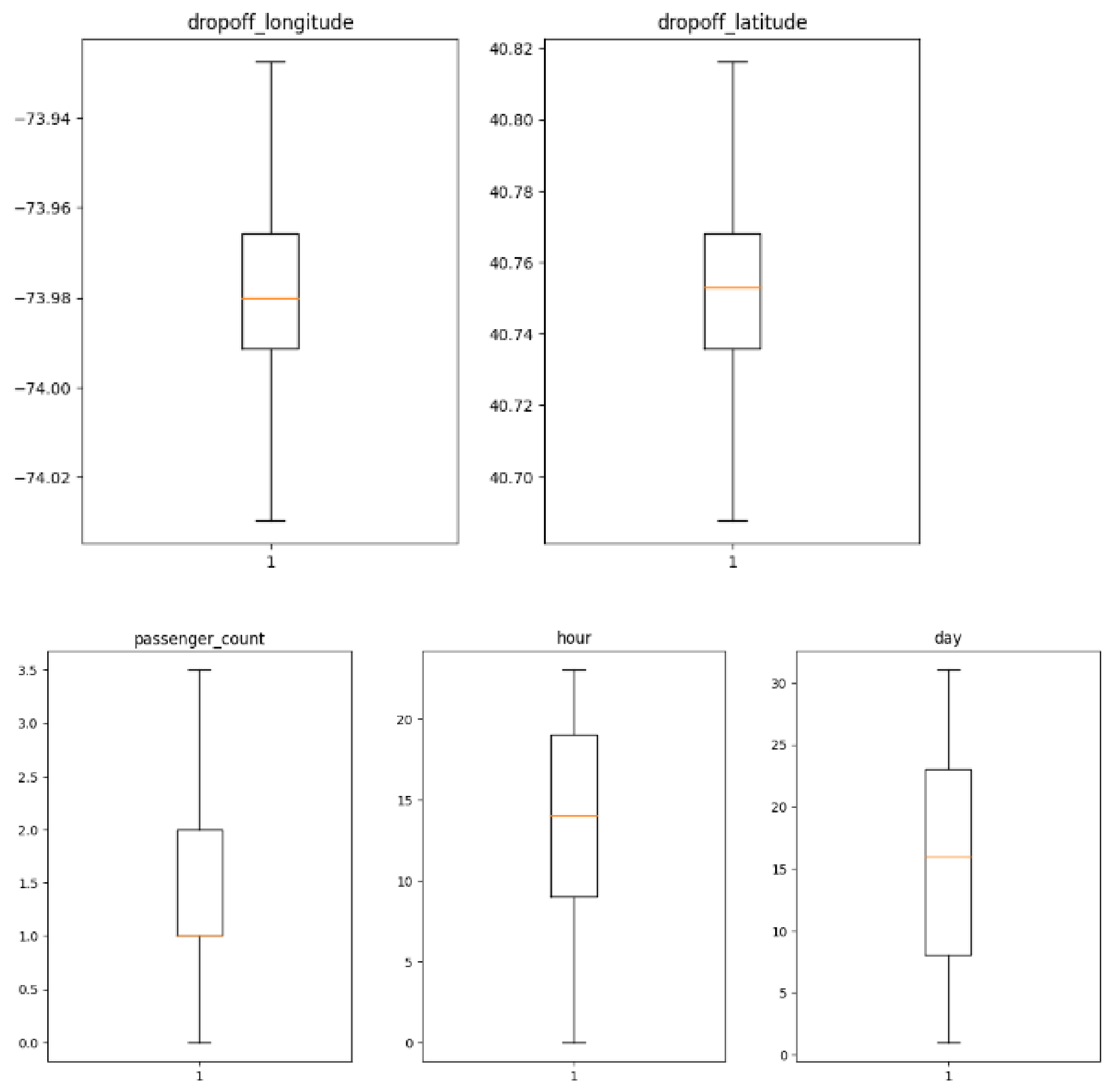
Ensure that fare amounts are non-negative and verify the DataFrame's dimensions after handling the outliers to ensure data integrity.

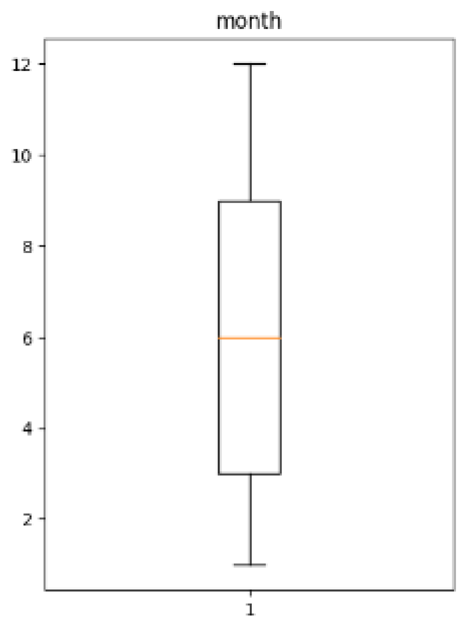
Boxplot of outliers:

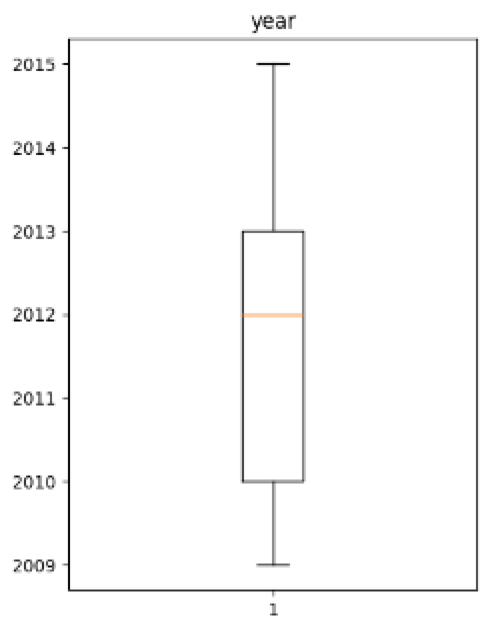


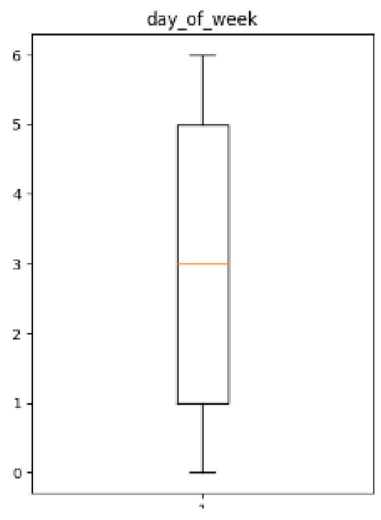
Boxplot of all the columns after imputing outliers:

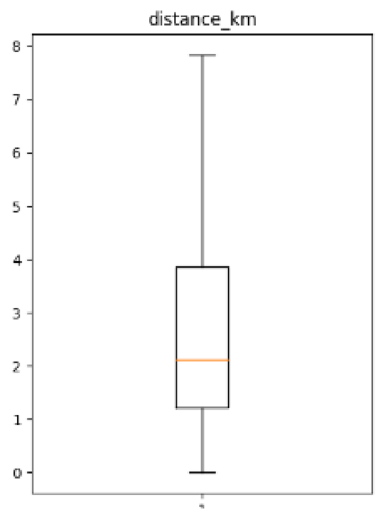


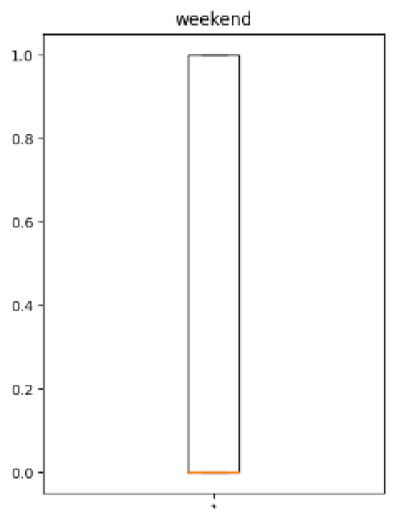

dropoff_longitude
dropoff_latitude
passenger_count
hour
day


month

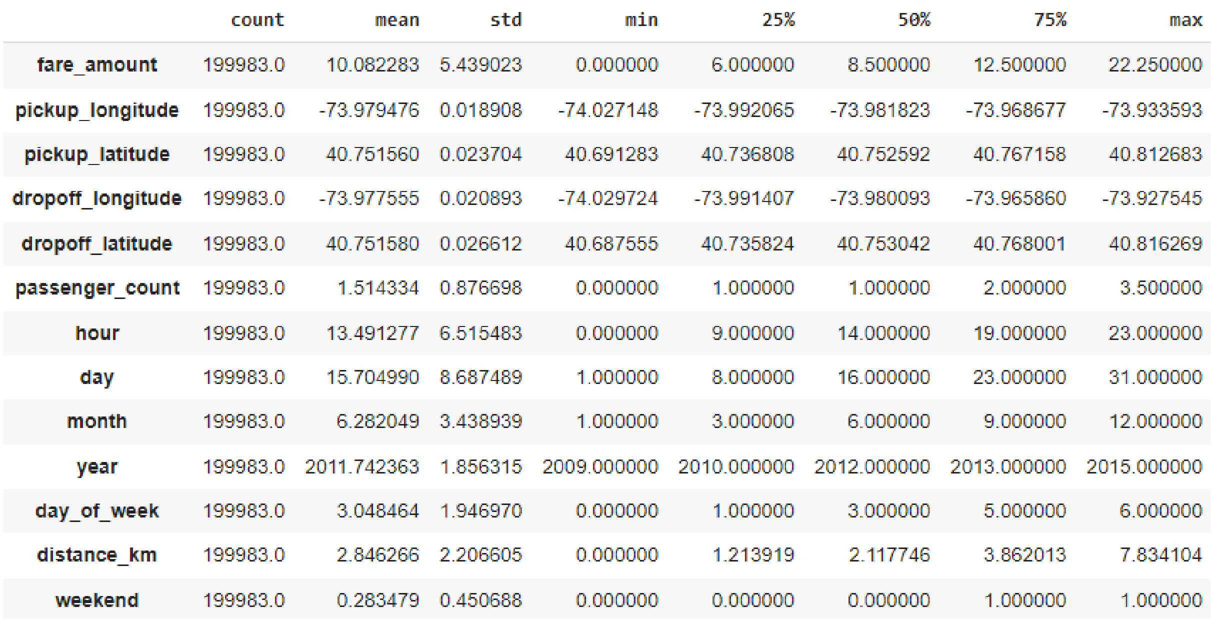

year


day_of_week


distance_km


weekend

Description of the dataframe after imputing the outliers:



	count	mean	std	min	25%	50%	75%	max
fare_amount	199983.0	10.082283	5.439023	0.000000	6.000000	8.500000	12.500000	22.250000
pickup_longitude	199983.0	-73.979476	0.018908	-74.027148	-73.992065	-73.981823	-73.968677	-73.933593
pickup_latitude	199983.0	40.751560	0.023704	40.691283	40.736808	40.752592	40.767158	40.812683
dropoff_longitude	199983.0	-73.977555	0.020893	-74.029724	-73.991407	-73.980093	-73.965860	-73.927545
dropoff_latitude	199983.0	40.751580	0.026612	40.687555	40.735824	40.753042	40.768001	40.816269
passenger_count	199983.0	1.514334	0.876698	0.000000	1.000000	1.000000	2.000000	3.500000
hour	199983.0	13.491277	6.515483	0.000000	9.000000	14.000000	19.000000	23.000000
day	199983.0	15.704990	8.687489	1.000000	8.000000	16.000000	23.000000	31.000000
month	199983.0	6.282049	3.438939	1.000000	3.000000	6.000000	9.000000	12.000000
year	199983.0	2011.742363	1.856315	2009.000000	2010.000000	2012.000000	2013.000000	2015.000000
day_of_week	199983.0	3.048464	1.946970	0.000000	1.000000	3.000000	5.000000	6.000000
distance_km	199983.0	2.846266	2.206605	0.000000	1.213919	2.117746	3.862013	7.834104
weekend	199983.0	0.283479	0.450688	0.000000	0.000000	0.000000	1.000000	1.000000

Correlation Analysis

Correlation analysis is a statistical technique used to determine the strength and direction of the relationship between two or more variables. Here's a brief overview of how to perform correlation analysis:

1. Understand Correlation:
   - Positive Correlation: As one variable increases, the other variable also increases.
   - Negative Correlation: As one variable increases, the other variable decreases.
   - No Correlation: No discernible relationship between the variables.
2. Calculate Correlation Coefficient:
   - The most commonly used coefficient is the Pearson correlation coefficient, which measures linear correlation between two variables. The formula is:

$$r = \frac{\text{cov}(X, Y)}{\sigma_X \cdot \sigma_Y}$$

Where:

- r is the Pearson correlation coefficient.
- cov(X,Y) is the covariance between variables X and Y.
- σX and σY are the standard deviations of X and Y, respectively.

A correlation matrix is a table used to show the correlation coefficients between many variables. Each cell in the matrix displays the correlation between two variables. Correlation coefficients range from -1 to 1, where:

- **1** indicates a perfect positive linear relationship,
- **-1** indicates a perfect negative linear relationship, and
- **0** indicates no linear relationship.

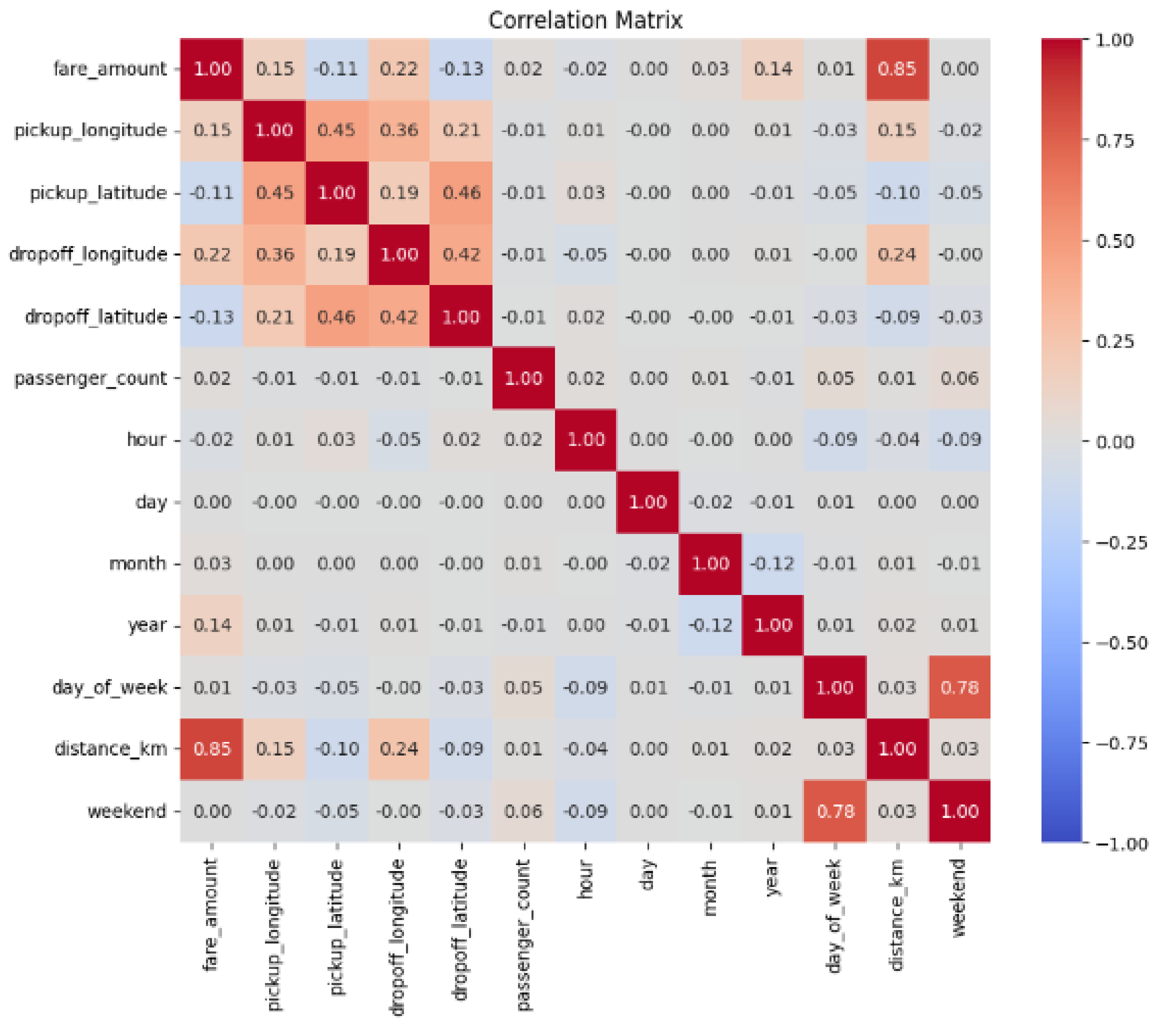
Strong Correlation:

- Positive Strong Correlation: When the correlation coefficient is close to 1 (e.g., 0.7 or higher), it indicates a strong positive linear relationship between the variables. This means that as one variable increases, the other variable tends to increase as well.
- Negative Strong Correlation: When the correlation coefficient is close to -1 (e.g., -0.7 or lower), it indicates a strong negative linear relationship. This means that as one variable increases, the other variable tends to decrease.

Weak Correlation:

- Positive Weak Correlation: When the correlation coefficient is close to 0 but positive (e.g., between 0 and 0.3), it indicates a weak positive linear relationship. There is a slight tendency for the variables to increase together, but the relationship is not strong.
- Negative Weak Correlation: When the correlation coefficient is close to 0 but negative (e.g., between -0.3 and 0), it indicates a weak negative linear relationship. There is a slight tendency for one variable to decrease as the other increases, but the relationship is not strong.

Correlation matrix:



correlations with the target variable (fare_amount)

```
fare_amount          1.000000
distance_km          0.848612
dropoff_longitude    0.219200
pickup_longitude     0.151777
year                 0.141429
month                0.030640
passenger_count      0.015882
day_of_week          0.013604
weekend              0.004517
day                  0.004443
hour                -0.023607
pickup_latitude     -0.112342
dropoff_latitude    -0.129299
Name: fare_amount, dtype: float64
```

Standardization

Standardization is a process used to transform variables into a common scale with a mean of zero and a standard deviation of one. This is particularly useful in statistical analyses and machine learning for the following reasons:

1. Normalization**:** It ensures that different variables contribute equally to the analysis, especially when they are on different scales or units.
2. Comparability**:** It allows for comparison of variables that have different units or ranges by putting them on the same scale.
3. Convergence**:** In machine learning algorithms, especially those involving distance metrics or gradient-based optimization (e.g., k-nearest neighbors, support vector machines, or gradient descent), standardization can improve convergence and performance.

Standardization is done using the Z-score formula:

$$z = \frac{(x - \mu)}{\sigma}$$

where:

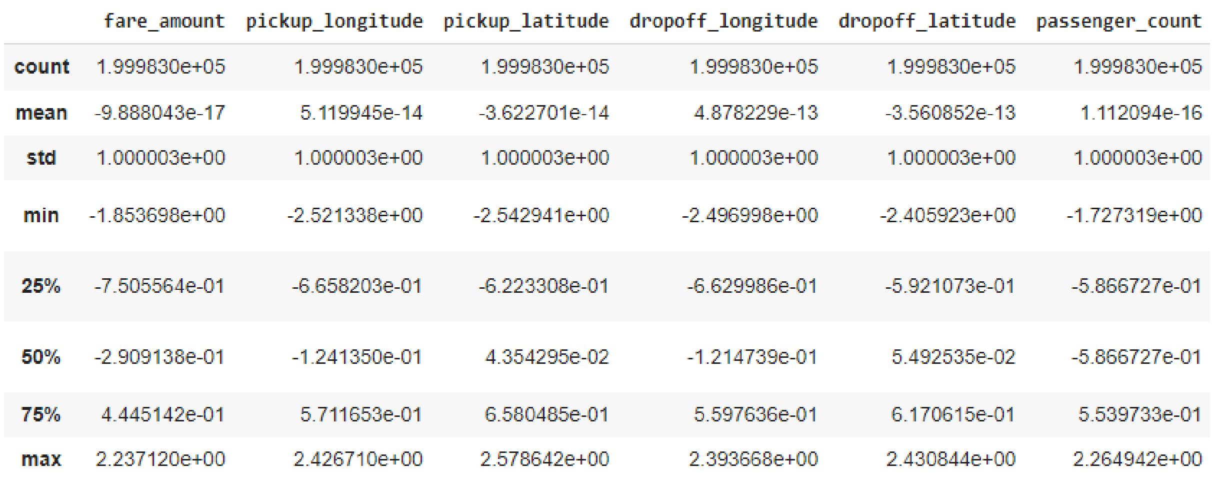
- x is the original value,
- μ is the mean of the variable,
- σ is the standard deviation of the variable,
- z is the standardized value.

Steps for Standardization

1. Calculate the Mean and Standard Deviation:
   - Compute the mean (μ) of the variable.
   - Compute the standard deviation (σ) of the variable.
2. Apply the Z-score Formula:

- For each value in the dataset, subtract the mean and then divide by the standard deviation.

After applying standard scalar



	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
count	1.999830e+05	1.999830e+05	1.999830e+05	1.999830e+05	1.999830e+05	1.999830e+05
mean	-9.888043e-17	5.119945e-14	-3.622701e-14	4.878229e-13	-3.560852e-13	1.112094e-16
std	1.000003e+00	1.000003e+00	1.000003e+00	1.000003e+00	1.000003e+00	1.000003e+00
min	-1.853698e+00	-2.521338e+00	-2.542941e+00	-2.496998e+00	-2.405923e+00	-1.727319e+00
25%	-7.505564e-01	-6.658203e-01	-6.223308e-01	-6.629986e-01	-5.921073e-01	-5.866727e-01
50%	-2.909138e-01	-1.241350e-01	4.354295e-02	-1.214739e-01	5.492535e-02	-5.866727e-01
75%	4.445142e-01	5.711653e-01	6.580485e-01	5.597636e-01	6.170615e-01	5.539733e-01
max	2.237120e+00	2.426710e+00	2.578642e+00	2.393668e+00	2.430844e+00	2.264942e+00

Split the Dataset

Splitting a dataset is a crucial step in machine learning and data analysis to ensure that your model is trained and evaluated properly. Typically, the dataset is divided into training and testing sets, and sometimes also a validation set. Here's a detailed guide on how to split a dataset:

Splitting a Dataset into Training and Testing Sets

Training Set: Used to train the machine learning model.

Testing Set: Used to evaluate the model's performance on unseen data.

A common split ratio is 70% for training and 30% for testing, but this can vary. Other typical splits are 80/20 or 90/10.

Stratified Splitting: For classification problems, ensure that the splits maintain the same proportion of classes as in the original dataset using stratify=y in train_test_split.

Shuffle and Random State: Ensure reproducibility and randomness in splits by setting the random_state parameter.

Scaling Before Splitting: If you plan to standardize or normalize your data, always fit the scaler on the training set only and then transform both training and testing (or validation) sets to avoid data leakage.

Feature Selection

Feature selection is a critical step in the machine learning pipeline that involves selecting the most relevant features (or variables) for model training. The goal is to improve model performance, reduce overfitting, and decrease computational cost by removing irrelevant or redundant features.

Why Feature Selection?

Improves Model Performance: Reduces overfitting by eliminating noisy or irrelevant features, which helps in creating a model that generalizes better.

Reduces Computational Complexity: Fewer features mean faster training and prediction times.

Simplifies Models: Makes the model easier to understand and interpret.

Feature selection is essential for developing effective machine learning models. It helps in improving model performance, reducing training times, and making models more interpretable. By applying these techniques, you can ensure that your model is trained on the most relevant features, leading to better insights and predictions.

Here, the Feature Selection is done using Recursive Decision Tree

Feature Ranking:

```
          Feature  Ranking
      distance_km        1
  dropoff_latitude       2
 dropoff_longitude       3
  pickup_longitude       4
   pickup_latitude       5
              year       6
               day       7
              hour       8
             month       9
       day_of_week      10
   passenger_count      11
           weekend      12
```

Regression Modeling

Regression is a type of statistical and machine learning technique used to model and analyse the relationships between a dependent (or target) variable and one or more independent (or predictor) variables. The goal of regression is to predict or estimate the value of the dependent variable based on the values of the independent variables.

Key Concepts in Regression

- Dependent Variable: The variable you are trying to predict or explain. It is also known as the target variable or response variable.
- Independent Variables: The variables that are used to predict or explain the dependent variable. They are also called predictors, features, or explanatory variables.
- Regression Line: In simple linear regression, this is a line that best fits the data points on a graph, showing the relationship between the dependent and independent variables.

Regression models are a core component of predictive analytics, used to model and analyse relationships between a dependent variable (target) and one or more independent variables (features). Here’s an overview of the various types of regression models:

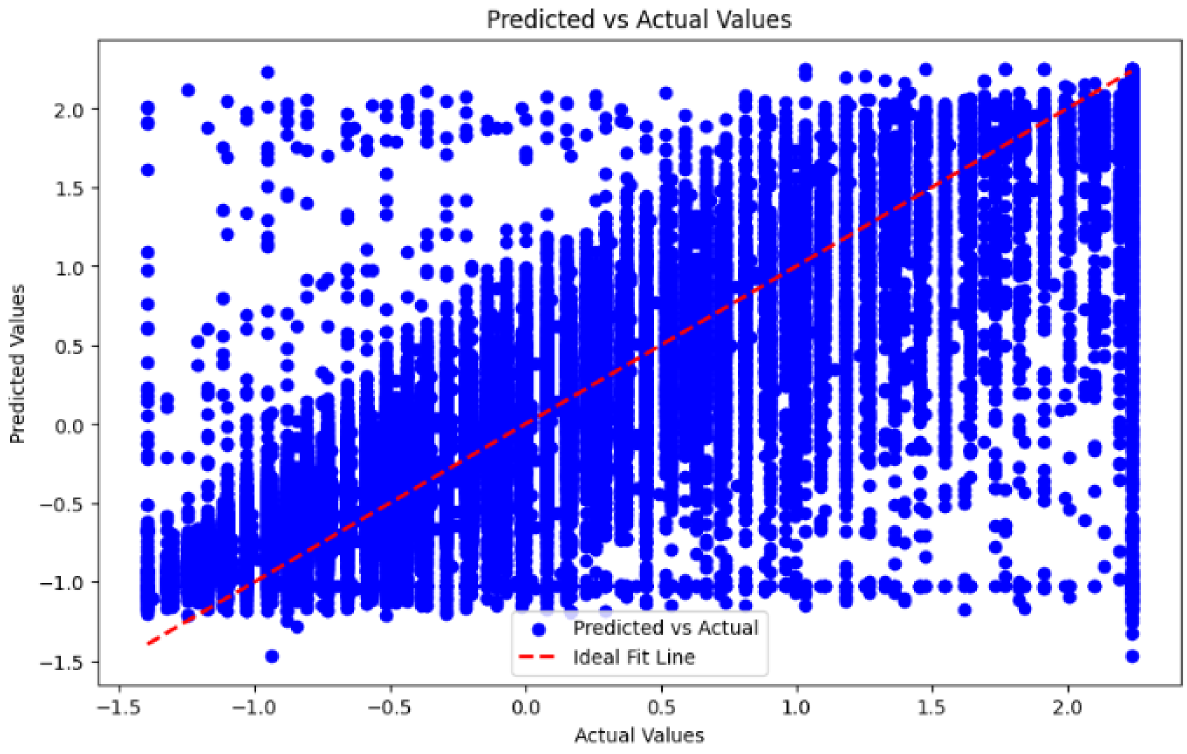
The different regression models used in the project are:

1. Linear Regression: Models the relationship between a dependent variable and multiple independent variables using a straight line, modelling the relationship with a hyperplane
2. Polynomial Regression: A form of regression where the relationship between the independent and dependent variables is modelled as an nth-degree polynomial, allowing for curved lines in the data.
3. Decision Tree Regression: Uses a tree-like model of decisions to predict the target variable; splits the data into subsets based on feature values, making it non-linear.

4. <u>Random Forest Regression</u>: An ensemble method that combines multiple decision trees to improve predictive performance and handle overfitting.
5. XGBoost regression utilizes the gradient boosting framework to build predictive models by iteratively improving upon residual errors with decision trees. It incorporates regularization techniques to prevent overfitting and enhance model generalization. Known for its speed and efficiency, XGBoost regression excels in handling large datasets and complex relationships, making it a popular choice in machine learning tasks.

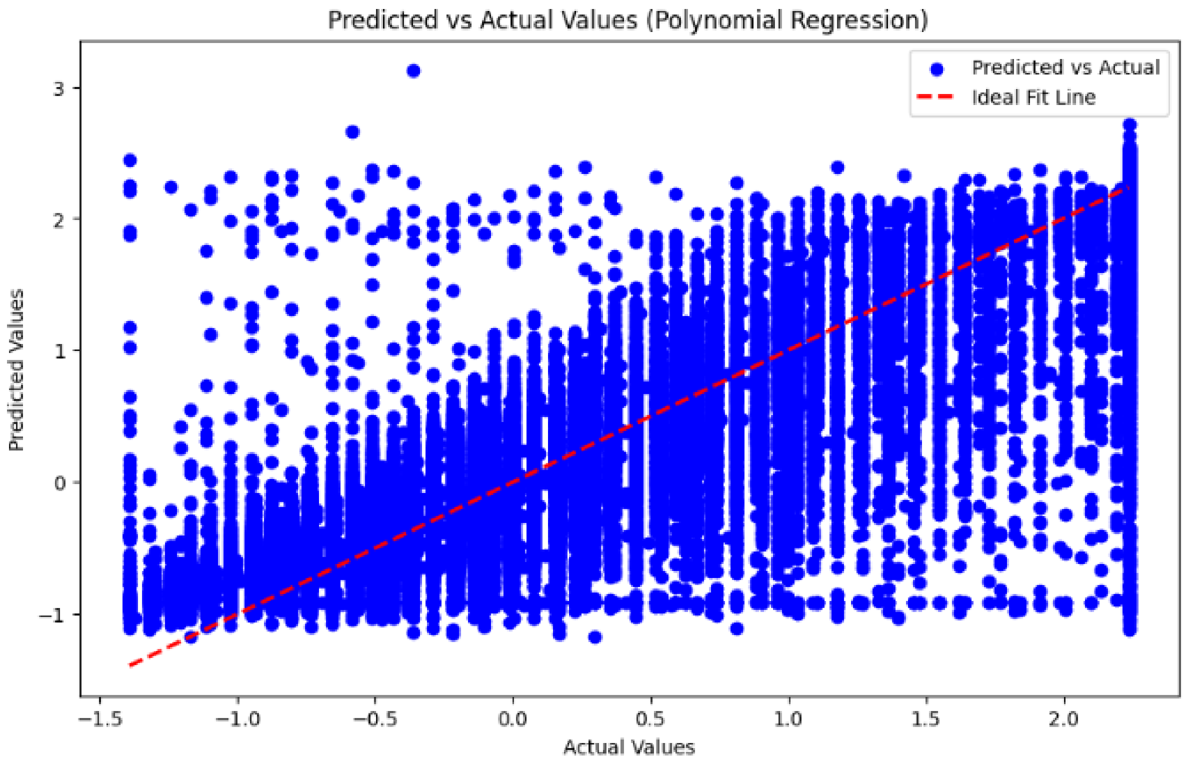
Linear Regression:

```
Root Mean Squared Error (RMSE): 0.5233882119978517
Mean Absolute Error (MAE): 0.3549751877989272
R-squared (R²): 0.7252535964004369
```



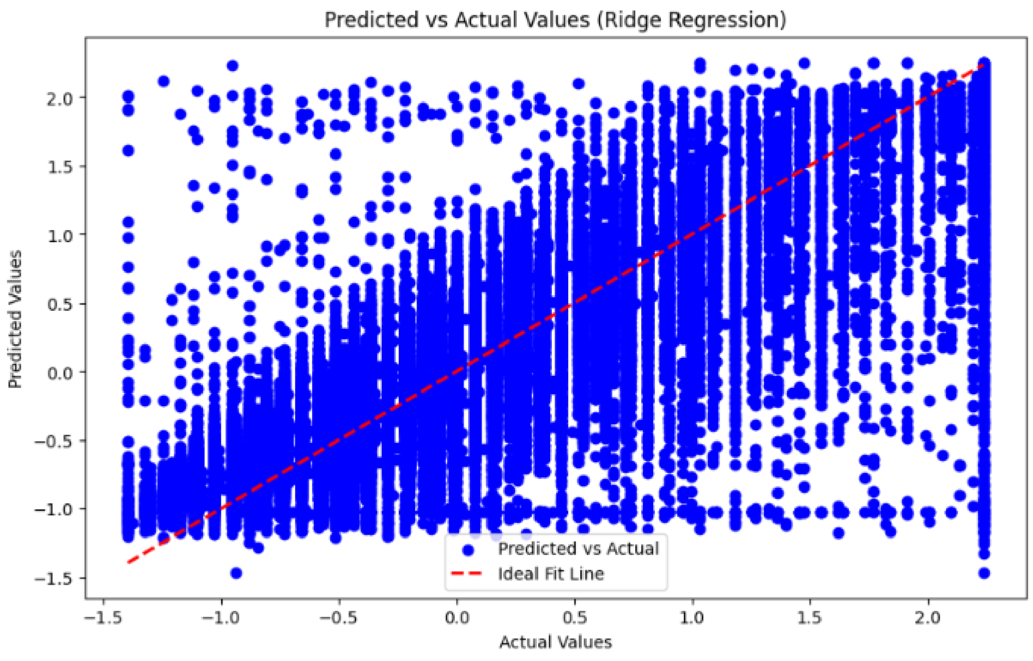
Polynomial Regression

```
Root Mean Squared Error (RMSE): 0.5164829374419744
Mean Absolute Error (MAE): 0.34704633265987656
R-squared (R²): 0.7324554555319458
```



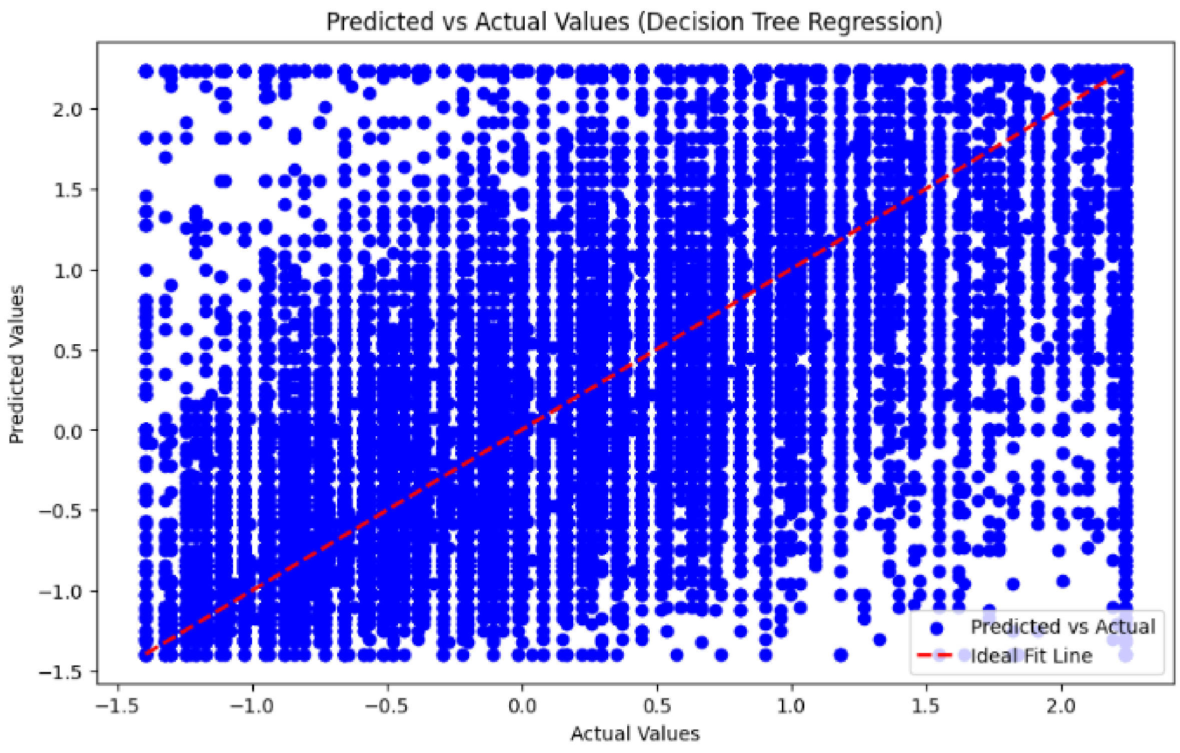
Ridge Regression

```
Root Mean Squared Error (RMSE): 0.5233882362724597
Mean Absolute Error (MAE): 0.3549763718138768
R-squared (R²): 0.725253570915104
```



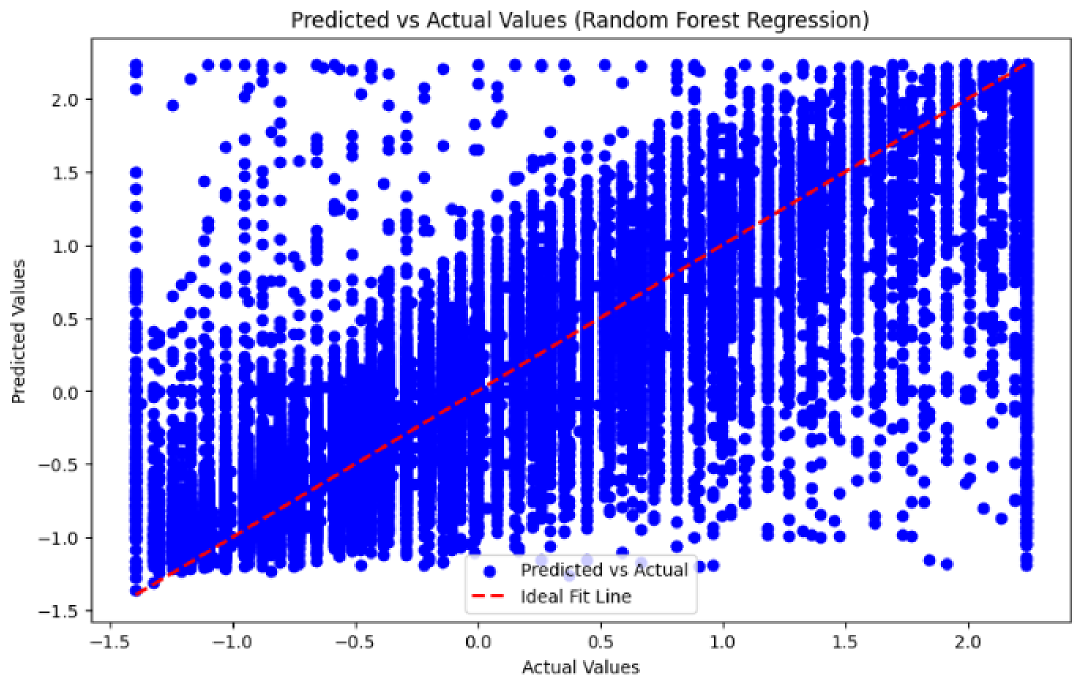
Decision Tree Regression

```
Root Mean Squared Error (RMSE): 0.6378233449396463
Mean Absolute Error (MAE): 0.4104904665294898
R-squared (R²): 0.5919767001966687
```



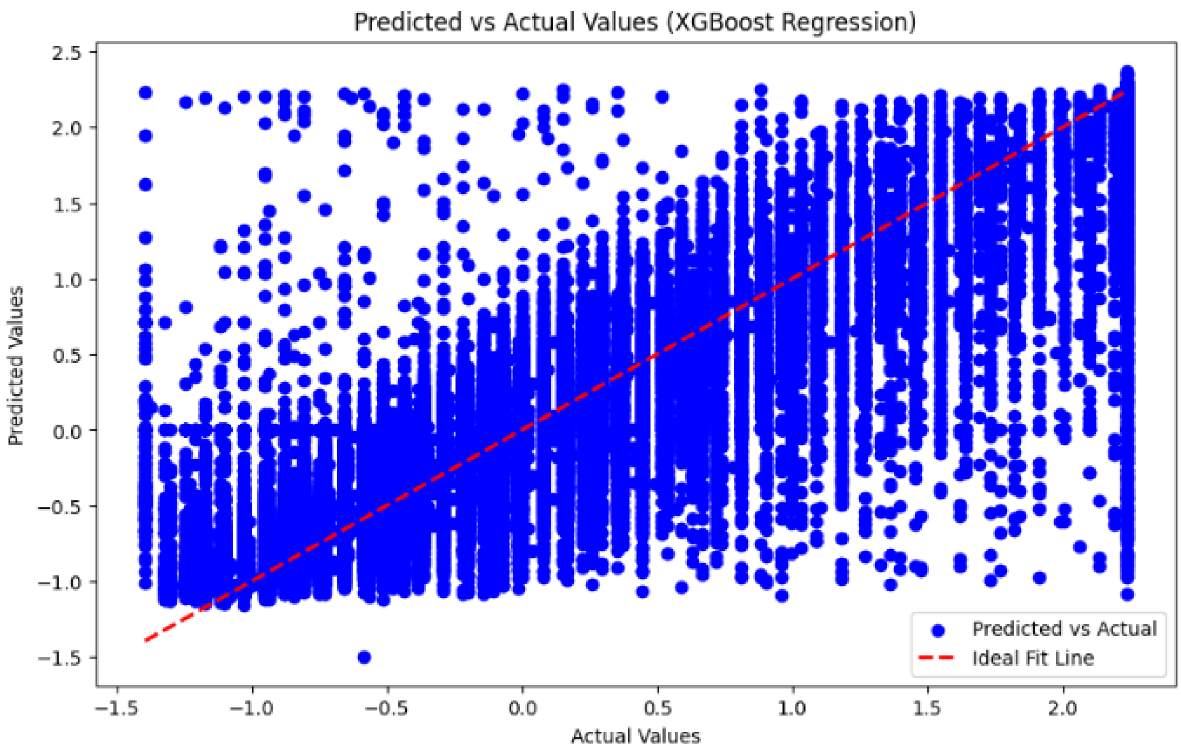
Random Forest Regression

```
Root Mean Squared Error (RMSE): 0.47115401959244585
Mean Absolute Error (MAE): 0.3060492091506313
R-squared (R²): 0.7773565395553911
```



XGBoost

```
Root Mean Squared Error (RMSE): 0.46353672025652876
Mean Absolute Error (MAE): 0.30447521927449456
R-squared (R²): 0.7844974420760744
```



Comparison of all the models:

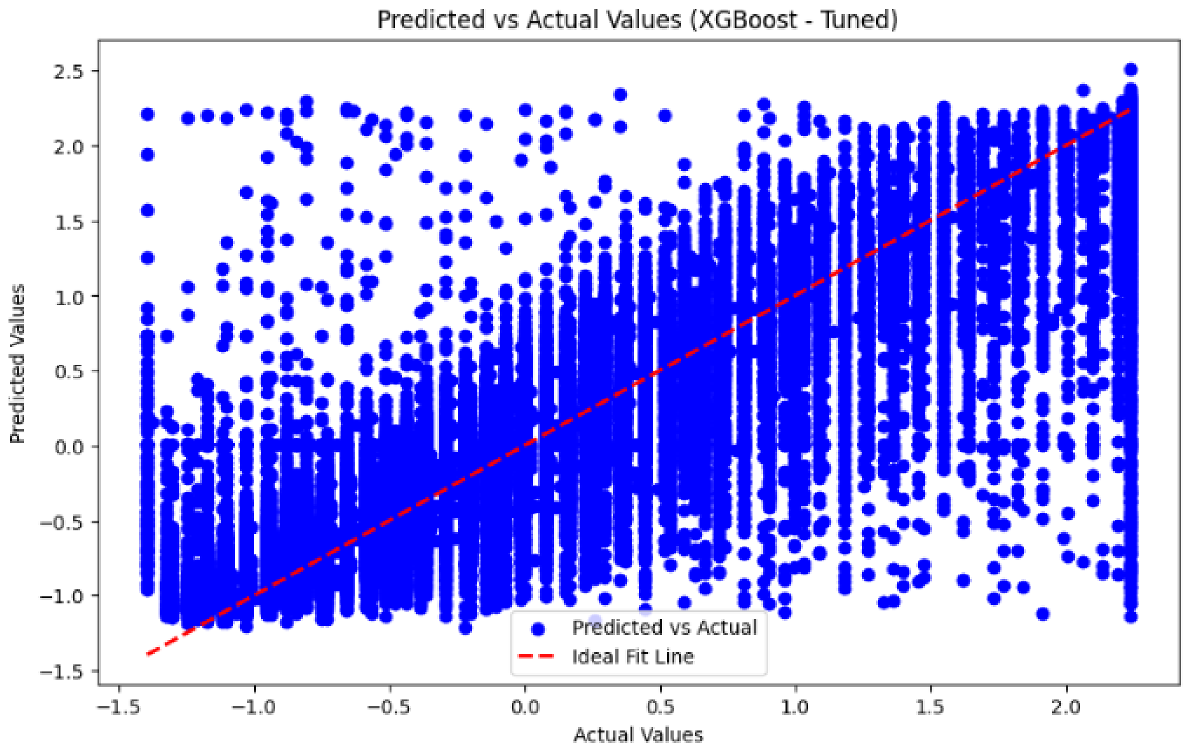
```
                                RMSE       MAE           R^2
Linear Regression           0.523388  0.354975  7.252536e-01
Polynomial Regression       0.516483  0.347046  7.324555e-01
Ridge Regression            0.523388  0.354976  7.252536e-01
Lasso Regression            0.998523  0.797344 -9.103263e-07
Elastic Net Regression      0.810968  0.645282  3.403836e-01
Decision Tree Regression    0.637823  0.410490  5.919767e-01
Random Forest Regression    0.471154  0.306049  7.773565e-01
Gradient Boosting Regression 0.479027  0.320195  7.698534e-01
XGBoost Regression          0.463537  0.304475  7.844974e-01
CatBoost Regression         0.471670  0.312942  7.768689e-01
AdaBoost Regression         0.754968  0.645598  4.283356e-01
```

Fine-tuning the Model

Fine-tuning the model involves optimizing its performance by adjusting hyperparameters and refining its structure based on validation data. This process typically includes techniques such as grid search or random search to explore different combinations of hyperparameters like learning rate, tree depth, and number of estimators, as well as feature engineering and selection to improve model accuracy. Additionally, cross-validation is often employed to ensure the model generalizes well to unseen data and avoids overfitting. Effective fine-tuning results in a model that not only performs well on the training data but also maintains high predictive accuracy and robustness when applied to new, real-world scenarios.

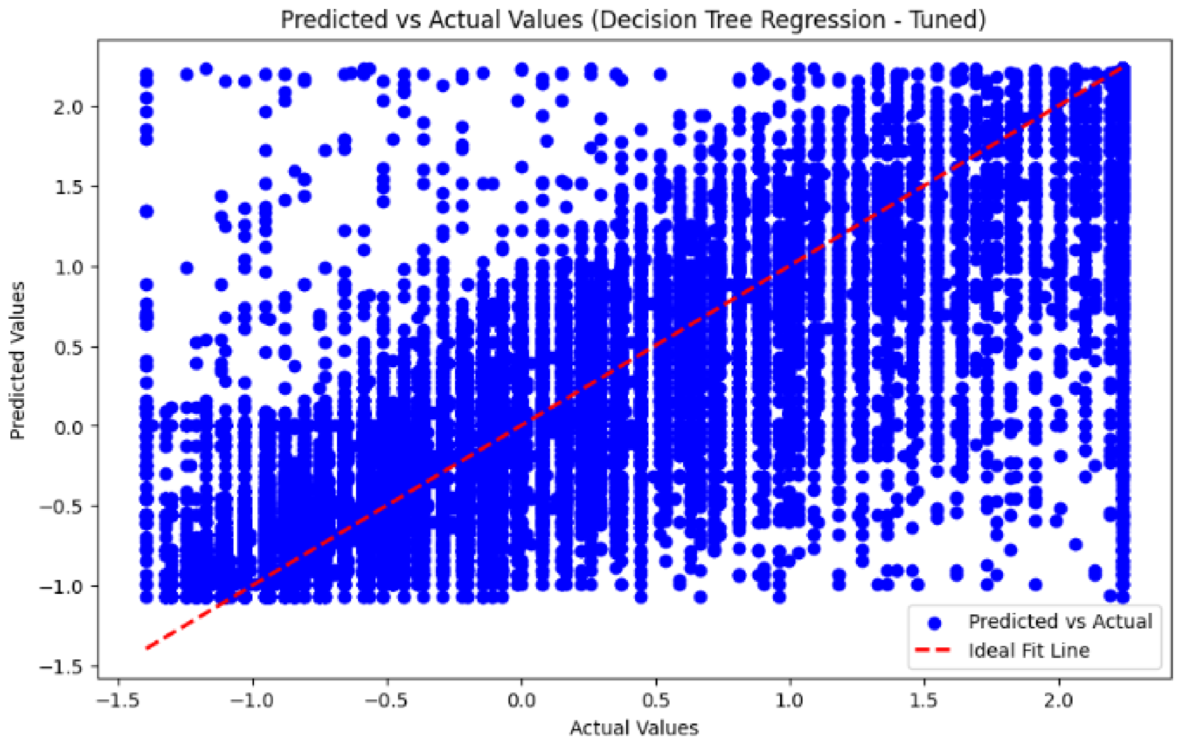
After fine tuning the XGBoost Model:

```
Fitting 3 folds for each of 27 candidates, totalling 81 fits
Best parameters: {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 200}
Root Mean Squared Error (RMSE): 0.45943403685036
Mean Absolute Error (MAE): 0.29861723033205795
R-squared (R²): 0.7882953120137891
```

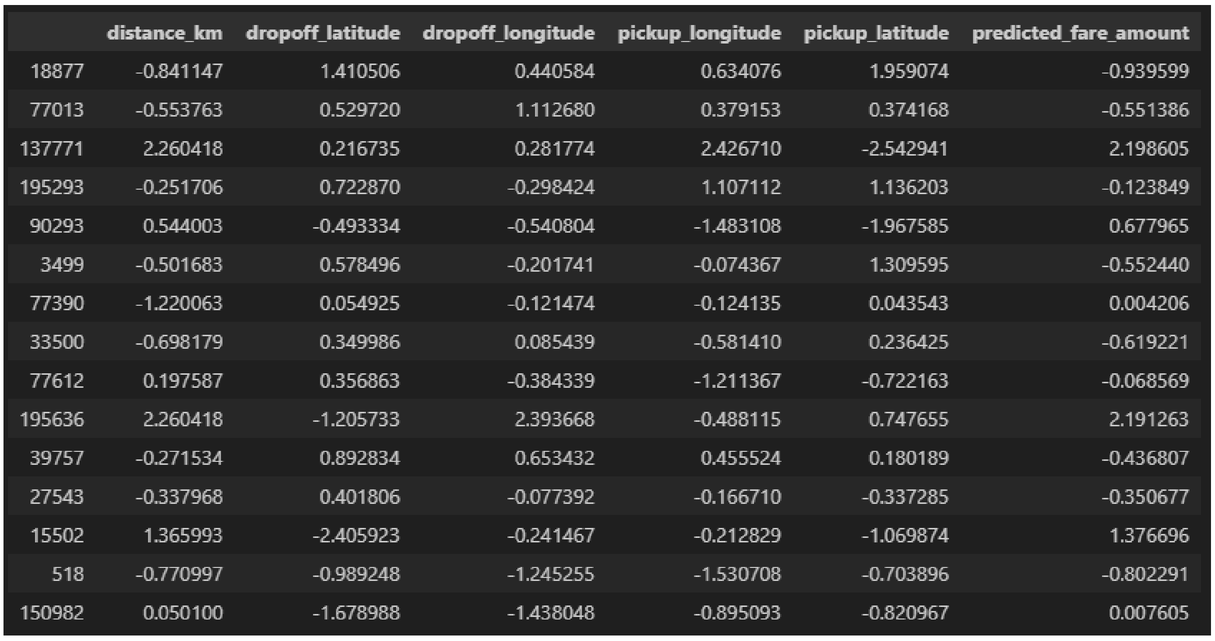


After fine-tuning the decision tree regression model:

```
Best parameters: {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 2}
Root Mean Squared Error (RMSE): 0.481437576346544
Mean Absolute Error (MAE): 0.31788717948321166
R-squared (R²): 0.7675315015912945
```



The model also predicts the fare amount for the new dataset:



	distance_km	dropoff_latitude	dropoff_longitude	pickup_longitude	pickup_latitude	predicted_fare_amount
18877	-0.841147	1.410506	0.440584	0.634076	1.959074	-0.939599
77013	-0.553763	0.529720	1.112680	0.379153	0.374168	-0.551386
137771	2.260418	0.216735	0.281774	2.426710	-2.542941	2.198605
195293	-0.251706	0.722870	-0.298424	1.107112	1.136203	-0.123849
90293	0.544003	-0.493334	-0.540804	-1.483108	-1.967585	0.677965
3499	-0.501683	0.578496	-0.201741	-0.074367	1.309595	-0.552440
77390	-1.220063	0.054925	-0.121474	-0.124135	0.043543	0.004206
33500	-0.698179	0.349986	0.085439	-0.581410	0.236425	-0.619221
77612	0.197587	0.356863	-0.384339	-1.211367	-0.722163	-0.068569
195636	2.260418	-1.205733	2.393668	-0.488115	0.747655	2.191263
39757	-0.271534	0.892834	0.653432	0.455524	0.180189	-0.436807
27543	-0.337968	0.401806	-0.077392	-0.166710	-0.337285	-0.350677
15502	1.365993	-2.405923	-0.241467	-0.212829	-1.069874	1.376696
518	-0.770997	-0.989248	-1.245255	-1.530708	-0.703896	-0.802291
150982	0.050100	-1.678988	-1.438048	-0.895093	-0.820967	0.007605

Conclusion

In conclusion, the project has demonstrated the power and flexibility of various machine learning models in handling complex, real-world data. The use of multiple regression techniques, including Linear Regression, Polynomial Regression, Ridge Regression, Decision Trees, Random Forest, and XGBoost has allowed for a comprehensive analysis and comparison of model performance.

Key performance metrics such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared values were calculated to evaluate each model's accuracy and reliability. The analysis revealed the strengths and weaknesses of different models, helping to identify the most suitable model which is XGBoost Model. After applying hyperparameter tuning using GridSearchCV the model is more optimized and also provided improved accuracy.

Overall, the analysis and results from this project lay a solid foundation for future research and development. By continuing to refine the models and incorporate new data sources and techniques, the project has the potential to significantly impact both the technical and practical aspects of fare prediction and pricing strategies in the transportation industry.

Future Scope

The project has significant potential for future development. One avenue for expansion involves the incorporation of more advanced models. For instance, deep learning techniques such as Neural Networks, especially Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) networks, could be explored for better accuracy in time-series predictions. Additionally, ensemble learning methods like stacking or blending different types of models (e.g., Random Forest combined with Gradient Boosting) could further enhance the model's predictive capabilities.

Another exciting direction is the development of a real-time fare prediction system. Integrating the model into a live system, where predictions are updated instantly based on new data—such as changes in traffic conditions or weather—would make the model more dynamic and applicable in real-world scenarios like ride-sharing platforms. Moreover, leveraging data from IoT devices, including GPS trackers and in-car sensors, could provide real-time inputs that enhance the accuracy and relevance of predictions.

Personalized fare prediction is another area of potential development. By analysing user behaviour, preferences, and historical data, the model could be adapted to offer customized fare estimates for individual users. This could also extend to dynamic pricing strategies, where the model predicts and recommends prices based on factors like peak hours or high-demand areas, thereby optimizing revenue for ride-sharing companies.

Overall, this project offers numerous opportunities for expansion and application across various domains, from real-time systems and personalized predictions to sustainability and economic analysis.

References

https://iacsd.mentormind.in/menternships/predict-the-fare-amount-of-future-rides-using-regression-analysis

https://www.kaggle.com/datasets/yasserh/uber-fares-dataset/data

https://drive.google.com/file/d/1Y0yK78NScNteDZu9q7lTRwXvzeF6FvX/view?usp=sharing

https://scikit-learn.org/stable/