SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

TAXI PRICE PREDICTION

CA4 PROJECT REPORT

Submitted by

RISHITHA THOKA – E0322026

AFRIN BANU – E0322050

In partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

(Artificial Intelligence & Data Analytics)

Sri Ramachandra Faculty of Engineering and Technology

Sri Ramachandra Institute of Higher Education and Research, Porur,

Chennai -600116

OCTOBER, 2024

SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

TAXI PRICE PREDICTION

CA4 PROJECT REPORT

Submitted by

RISHITHA THOKA – E0322026

AFRIN BANU – E0322050

In partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

(Artificial Intelligence & Data Analytics)

Sri Ramachandra Faculty of Engineering and Technology

Sri Ramachandra Institute of Higher Education and Research, Porur,

Chennai -600116

OCTOBER, 2024

TABLE OF CONTENTS

ABSTRACT

The project aims to build a predictive model for estimating taxi trip prices using regression-based machine learning algorithms and comprehensive data preprocessing techniques. The dataset, consisting of features like trip distance, passenger count, time of day, and traffic conditions, underwent imputation to address missing values and encoding for categorical variables. Several regression models—Linear Regression, Ridge Regression with regularization, Polynomial Regression, and Decision Tree were implemented to analyse the pricing patterns and predict trip prices accurately.

Evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), R-squared (R^2), and accuracy were utilized to assess the models' performance. The report outlines the comparative analysis of these models, highlighting their strengths and limitations in predicting taxi pricing based on the given features.

CHAPTER 1
INTRODUCTION

In the modern transportation sector, dynamic pricing models have become essential for efficient and user-friendly taxi services. Predicting taxi trip prices is a complex process influenced by various factors such as trip distance, time of day, traffic conditions, passenger count, and weather. Accurately forecasting prices based on these variables can enhance customer experience, improve operational efficiency, and enable competitive advantages for service providers.

This project focuses on developing a machine learning pipeline to predict taxi trip prices by leveraging a structured dataset with detailed trip attributes. Comprehensive preprocessing steps, including handling missing values, encoding categorical variables, and scaling features, form the foundation of the analysis. Various regression techniques were explored, including:

- **Linear Regression**: To establish a baseline model and evaluate the linear relationship between features and the target variable.
- **Ridge Regression**: To improve generalization and address overfitting through L2 regularization.
- **Polynomial Regression**: To capture and model non-linear interactions within the data.
- **Decision Tree Regression**: To explore hierarchical decision boundaries and account for complex relationships among variables.

Visualizations were integral to understanding data patterns and enhancing the interpretability of results. Key visualizations include:

- **Correlation Heatmaps**: To identify relationships between numerical features.
- **Scatter Plots**: To analyze the distribution of target variables against key features.

- **Bar Charts**: For examining the influence of categorical variables on pricing.
- **Residual Plots**: To evaluate model performance and identify areas of improvement.

The models were assessed using robust evaluation metrics, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), R-squared (R^2), and accuracy, enabling a comparative analysis of predictive performance. This project provides insights into factors influencing taxi prices and demonstrates the power of machine learning for predictive analytics in transportation.

1.1. TECHNIQUES INVOLVED

Data Preprocessing:

- Handling missing values using mean, median, and mode imputation for numerical and categorical data.
- Feature scaling to standardize numerical values for model consistency.
- Encoding categorical variables using label encoding to convert them into machine-readable formats.

Feature Engineering:

- Selection of relevant variables influencing taxi pricing, such as trip distance, time of day, traffic conditions, and passenger count.
- Polynomial transformation for modeling non-linear relationships.

Machine Learning Models:

- Linear Regression: To model the relationship between input variables and the target variable.
- Ridge Regression: For improved generalization and handling multicollinearity using L2 regularization.
- Polynomial Regression: To capture non-linear patterns between features and trip pricing.

- Decision Tree Regression: For understanding and modeling complex decision structures in pricing.

Model Evaluation:

- Assessment metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), R-squared (R^2), and accuracy for evaluating model performance.
- Residual analysis for identifying inconsistencies and improving model predictions.

Data Visualization:

- Heatmaps: To explore correlations between features and the target variable.
- Scatter Plots: For studying the distribution and relationships of numerical variables.
- Bar Graphs: To analyze the impact of categorical features such as traffic conditions and time of day.
- Residual Plots: For understanding errors in model predictions.

1.2. TECHNOLOGY INVOLVED

1. **Programming Language**
    - **Python:** Primary programming language used for data analysis and machine learning.
2. **Libraries and Frameworks**
    - **Pandas and NumPy:** For data manipulation and preprocessing.
    - **Scikit-learn:** For implementing regression models, preprocessing tools, and evaluation metrics.

- **Matplotlib and Seaborn:** For visualizing data insights and model evaluations.

3. **Development Environment**

   - **Jupyter Notebook:** Interactive environment for writing code, analyzing data, and visualizing results.

4. **Tools for Deployment and Experimentation**

   - **Google Colab:** Cloud-based platform for running Python scripts and training machine learning models.

These combined techniques and technologies ensure an end-to-end workflow for effective taxi price prediction, delivering practical solutions to pricing challenges.

CHAPTER 2
LITERATURE REVIEW

- **PAPER 1**

TITLE: Data Analysis and Fair Price Prediction Using Machine Learning Algorithms

SUMMARY: The paper explores the application of machine learning algorithms for predicting fair market prices, with a focus on data analysis techniques. The researchers likely examine various ML models and their effectiveness in price prediction, comparing traditional approaches with more advanced algorithms. The study probably includes data preprocessing methods, feature selection, and model evaluation metrics to determine the most accurate price prediction framework.

- **PAPER 2**

TITLE: Predicting demand for air taxi urban aviation services using machine learning algorithms

SUMMARY: The paper on predicting demand for air taxi urban aviation services using machine learning algorithms explores how AI can be used to forecast the demand for air taxis in urban environments. By integrating data such as population density, urban mobility patterns, transportation infrastructure, weather, and economic activity, the paper applies machine learning models like regression, classification, and time-series analysis to predict the demand for air taxi services. The study aims to assist in optimizing the operational planning, fleet management, and pricing strategies for air taxi services, offering a scalable solution to urban transportation challenges.

- **PAPER 3**

TITLE: New York City taxi trip duration prediction using MLP and XGBoost

SUMMARY: The paper focuses on leveraging machine learning models, specifically Multi-Layer Perceptron (MLP) and XGBoost, to predict taxi trip durations in New York City. The authors use a rich dataset of trip details, including pickup and drop-off locations, timestamps, weather conditions, and traffic data, to train the models. The MLP model, a type of neural network, learns complex patterns from the data, while XGBoost, a gradient boosting algorithm, enhances predictive accuracy by combining multiple weak predictive models. The study evaluates the performance of both models, demonstrating that XGBoost outperforms MLP, delivering highly accurate and reliable predictions for estimating taxi trip durations.

- **PAPER 4**

TITLE: A Novel Approach for Fare Prediction Using Machine Learning Techniques.

SUMMARY: The paper introduces an innovative method for predicting transportation fares by leveraging machine learning algorithms. The study employs various data sources, such as historical fare data, location, time, and traffic conditions, to build accurate predictive models. It compares multiple machine learning techniques, including regression models, decision trees, and neural networks, to forecast fares with high accuracy. The model's effectiveness is validated using real-world datasets, demonstrating improved performance over traditional fare prediction systems. This approach aims to optimize pricing strategies for both riders and service providers, contributing to efficient resource allocation in the transportation sector.

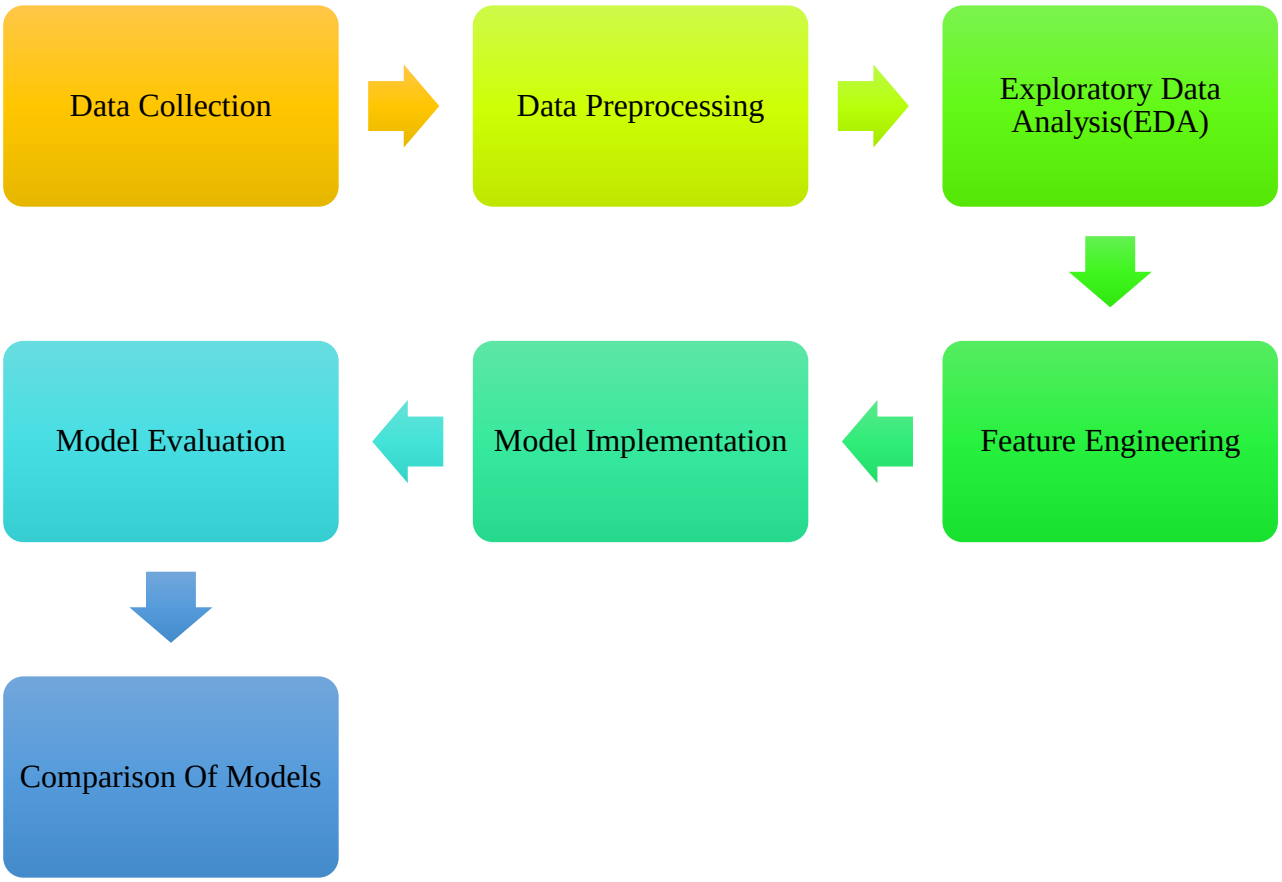
- **PAPER 5**

TITLE: A stepwise interpretable machine learning framework using linear regression (LR) and long short-term memory (LSTM): City-wide demand-side prediction of yellow taxi and for-hire vehicle (FHV) service

SUMMARY: The paper presents a stepwise interpretable machine learning framework that combines linear regression (LR) and long short-term memory (LSTM) networks to predict city-wide demand for yellow taxis and for-hire vehicle (FHV) services. The framework integrates LR for initial feature selection and explanation, followed by LSTM models for time-series forecasting to capture spatial and temporal dependencies in ride-demand patterns. This hybrid approach provides interpretable insights into factors driving demand, while improving prediction accuracy for both short-term and long-term taxi service requirements. The model is tested on real-world data, showing its potential for enhancing transportation planning and optimization.

CHAPTER 3

PROPOSED METHODOLOGY



- **Data Collection:** Load and explore the dataset to understand the structure, missing values, and distribution of variables.
- **Data Preprocessing:**
  1. Handle missing values using imputation techniques (mean, median, or mode).
  2. Encode categorical features into numerical form using techniques like Label Encoding or One-Hot Encoding.
  3. Scale features to standardize input data.

Figure 3.1-Methodology

- **Exploratory Data Analysis (EDA):** Use visualizations such as scatter plots, bar charts, and heatmaps to uncover relationships between features and the target variable.
- **Feature Engineering:** Select the most relevant variables and create polynomial features for non-linear relationships.
- **Model Implementation:** Train various regression models (Linear, Ridge, Polynomial, and Decision Tree).
- **Model Evaluation:** Evaluate the models using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), R-squared (R^2), and Accuracy.
- **Comparison of Models:** Analyze and compare the performance of the models based on evaluation metrics.

CHAPTER 4

IMPLEMENTATION

The implementation of the taxi price prediction project involves a structured workflow, combining data preparation, model training, evaluation, and insights generation. Below are the detailed steps broken down by phase:

1. Data Loading and Exploration

- Import required libraries: pandas, numpy, matplotlib, seaborn, and scikit-learn.
- Load the dataset into a Pandas DataFrame.
- Inspect the data using .info(), .describe(), and .head() to understand the structure, feature types, and summary statistics.

2. Data Preprocessing

- **Handle missing values using imputation techniques:**
    - For numerical variables: Fill with the mean or median.
    - For categorical variables: Fill with the mode or a new category.
- **Encode categorical features:**
    - Apply label encoding or one-hot encoding to convert textual data into numerical representations.
- **Feature scaling:**
    - Standardize features (StandardScaler) to ensure consistent input ranges for model training.

3. Exploratory Data Analysis (EDA)

- Visualize relationships between features and target variables:
    - Create scatter plots for numerical variables (e.g., trip distance vs. price).
    - Use correlation heatmaps to identify significant relationships.
    - Plot bar charts and histograms for categorical features to analyze their distribution.
- Identify potential outliers and skewed distributions for key variables.

4. Feature Engineering

- Select relevant features based on domain knowledge and statistical significance.
- Generate polynomial features for non-linear interactions using PolynomialFeatures from scikit-learn.
- Drop irrelevant or highly correlated features to reduce redundancy and improve model performance.

5. Model Implementation

- Split the data into training and testing sets using train_test_split from scikit-learn.
- Train the following models:
    - Linear Regression: As a baseline model.
    - Ridge Regression: Using Ridge from scikit-learn to apply L2 regularization.
    - Polynomial Regression: Transform input features and train using LinearRegression on the polynomial data.

- Decision Tree Regressor: To handle non-linear relationships using DecisionTreeRegressor.

- Fit each model to the training dataset and evaluate on the test set.

6. Model Evaluation

- Calculate evaluation metrics such as:
  - Mean Absolute Error (MAE): Average of absolute prediction errors.
  - Mean Squared Error (MSE): Average of squared prediction errors.
  - R-squared (R^2): Proportion of variance in the target variable explained by the model.
  - Accuracy: Percentage of correct predictions, where applicable.
- Use residual plots to visualize and diagnose prediction errors.

7. Comparison of Models

- Tabulate and compare the performance metrics across all models.
- Highlight the strengths and limitations of each model based on evaluation results.

8. Results and Visualization

- Present final insights with visualizations:
  - Model performance comparison using bar graphs.
  - Feature importance plots for Decision Trees.
- Summarize the best-performing model and its impact on prediction tasks.

CHAPTER 5
RESULTS AND DISCUSSIONS

The analysis of four different regression models reveals that the Decision Tree model demonstrated the best overall performance, with an R^2 value of 0.815903 and the highest accuracy of 81.59%. The Polynomial Regression followed as the second-best performer with an R^2 value of 0.904765 and an accuracy of 80.47%. Both Linear Regression and Ridge Regression showed similar performance metrics, with accuracies of approximately 76.7% and 76.9% respectively.

The Decision Tree's lower Mean Squared Error (MSE) of 0.098249 and Mean Absolute Error (MAE) of 0.231444 further supports its superior predictive capability compared to the other models. However, it's worth noting that the Polynomial Regression achieved the highest R^2 value, suggesting it captures the underlying relationships in the data well, despite having slightly lower accuracy than the Decision Tree.

APPENDICES

APPENDIX-1: CODE COMPILER

```
import pandas as pd
file_path = '/content/taxi_trip_pricing.csv'
data = pd.read_csv(file_path)

data.head()
data.describe()
data.isnull().sum()
data['Trip_Distance_km'].fillna(data['Trip_Distance_km'].mean(), inplace=True)
data['Passenger_Count'].fillna(data['Passenger_Count'].median(), inplace=True)
data['Base_Fare'].fillna(data['Base_Fare'].mean(), inplace=True)
data['Per_Km_Rate'].fillna(data['Per_Km_Rate'].mean(), inplace=True)
data['Per_Minute_Rate'].fillna(data['Per_Minute_Rate'].mean(), inplace=True)
data['Trip_Duration_Minutes'].fillna(data['Trip_Duration_Minutes'].median(),
inplace=True)
data['Trip_Price'].fillna(data['Trip_Price'].mean(), inplace=True)
for column in ['Time_of_Day', 'Day_of_Week', 'Traffic_Conditions', 'Weather']:
    data[column].fillna(data[column].mode()[0], inplace=True)
from sklearn.preprocessing import LabelEncoder
categorical_columns = ['Time_of_Day', 'Day_of_Week', 'Traffic_Conditions', 'Weather']

# Initialize the LabelEncoder
label_encoders = {}
for col in categorical_columns:
    label_encoders[col] = LabelEncoder()
    data[col] = label_encoders[col].fit_transform(data[col])
from sklearn.preprocessing import StandardScaler
numerical_columns = [
```

```
    'Trip_Distance_km', 'Passenger_Count', 'Base_Fare', 'Per_Km_Rate',
    'Per_Minute_Rate', 'Trip_Duration_Minutes', 'Trip_Price'
]

scaler = StandardScaler()
data_standardized = data.copy()
data_standardized[numerical_columns] = scaler.fit_transform(data[numerical_columns])

data_standardized.head()

import matplotlib.pyplot as plt
import seaborn as sns

correlation_matrix = data_standardized.corr()
plt.figure(figsize=(12, 8))

sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm', cbar=True,
square=True, linewidths=0.5)
plt.title("Correlation Matrix Heatmap", fontsize=16)
plt.show()

# Distribution of Trip Prices after standardization
sns.histplot(data_standardized['Trip_Price'], kde=True, bins=30, color="blue")
plt.title("Distribution of Trip Prices (Standardized)", fontsize=14)
plt.xlabel("Standardized Trip Price")
plt.ylabel("Frequency")

#Box plot of Trip Duration vs. Traffic Conditions
sns.boxplot(
    x='Traffic_Conditions',
    y='Trip_Duration_Minutes',
```

```
    data=data,
    palette="Set3",
    order=data['Traffic_Conditions'].value_counts().index,
)
plt.title("Trip Duration Across Traffic Conditions", fontsize=14)
plt.xlabel("Traffic Conditions")
plt.ylabel("Trip Duration (Minutes)")

#low,med,high
#Passenger Count vs. Trip Distance Scatter Plot
sns.scatterplot(
    x='Passenger_Count',
    y='Trip_Distance_km',
    data=data,
    hue='Time_of_Day',
    palette="coolwarm",
    alpha=0.8
)
plt.title("Passenger Count vs. Trip Distance", fontsize=14)
plt.xlabel("Passenger Count")
plt.ylabel("Trip Distance (km)")

#morn,afternoon,eve,night
#Count of Trips by Weather Condition
sns.countplot(
    x='Weather',
    data=data,
    order=data['Weather'].value_counts().index,
    palette="viridis"
)
```

```
plt.title("Number of Trips by Weather Condition", fontsize=14)
plt.xlabel("Weather Condition")
plt.ylabel("Count")

plt.tight_layout()
plt.show()
#clear,rain,snow
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Define features (X) and target (y)
X = data_standardized.drop(['Trip_Price'], axis=1)  # Drop the target column
y = data_standardized['Trip_Price']  # Target variable

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the Linear Regression model
linear_model = LinearRegression()

# Train the model
linear_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = linear_model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
```

```
r2 = r2_score(y_test, y_pred)
accuracy = r2 * 100

# Print evaluation metrics
print("Linear Regression Model Evaluation:")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"R² Score: {r2:.2f}")
print(f"Model Accuracy (R² as percentage): {accuracy:.2f}%")

# Display Actual vs Predicted Prices
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.7, color='blue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', linewidth=2)
plt.title("Actual vs Predicted Trip Prices", fontsize=16)
plt.xlabel("Actual Prices", fontsize=14)
plt.ylabel("Predicted Prices", fontsize=14)
plt.show()
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
ridge_model = Ridge(alpha=4.0, random_state=42)
# Train Ridge
ridge_model.fit(X_train, y_train)
ridge_pred = ridge_model.predict(X_test)
# Evaluate Ridge
ridge_mse = mean_squared_error(y_test, ridge_pred)
ridge_mae = mean_absolute_error(y_test, ridge_pred)
ridge_r2 = r2_score(y_test, ridge_pred)
```

```
ridge_accuracy = ridge_r2 * 100

print("\nRidge Regression Model Evaluation:")
print(f"Mean Squared Error (MSE): {ridge_mse:.2f}")
print(f"Mean Absolute Error (MAE): {ridge_mae:.2f}")
print(f"R² Score: {ridge_r2:.2f}")
print(f"Model Accuracy (R² as percentage): {ridge_accuracy:.2f}%")

# Ridge: Actual vs Predicted
plt.scatter(y_test, ridge_pred, alpha=0.7, color='purple')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', linewidth=2)
plt.title("Ridge: Actual vs Predicted Trip Prices", fontsize=14)
plt.xlabel("Actual Prices", fontsize=12)
plt.ylabel("Predicted Prices", fontsize=12)
plt.show()

from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np

# Transform the features to a polynomial space (degree=2 as an example)
poly = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly.fit_transform(X)  # Transform the features

# Split polynomial features into training and testing sets
X_poly_train, X_poly_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2,
random_state=42)

# Train a linear regression model on the polynomial features
poly_model = LinearRegression()
```

```
poly_model.fit(X_poly_train, y_train)

# Make predictions
y_poly_pred = poly_model.predict(X_poly_test)

# Evaluate the model
poly_mse = mean_squared_error(y_test, y_poly_pred)
poly_mae = mean_absolute_error(y_test, y_poly_pred)
poly_r2 = r2_score(y_test, y_poly_pred)
poly_accuracy = poly_r2 * 100  # Accuracy as R² percentage

# Print evaluation metrics
print("Polynomial Regression Model Evaluation:")
print(f"Mean Squared Error (MSE): {poly_mse:.2f}")
print(f"Mean Absolute Error (MAE): {poly_mae:.2f}")
print(f"R² Score: {poly_r2:.2f}")
print(f"Model Accuracy (R² as percentage): {poly_accuracy:.2f}%")

# Visualization of Actual vs Predicted Trip Prices
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_poly_pred, alpha=0.7, color='blue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', linewidth=2)
plt.title("Polynomial Regression: Actual vs Predicted Trip Prices", fontsize=16)
plt.xlabel("Actual Prices", fontsize=14)
plt.ylabel("Predicted Prices", fontsize=14)
plt.show()

from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

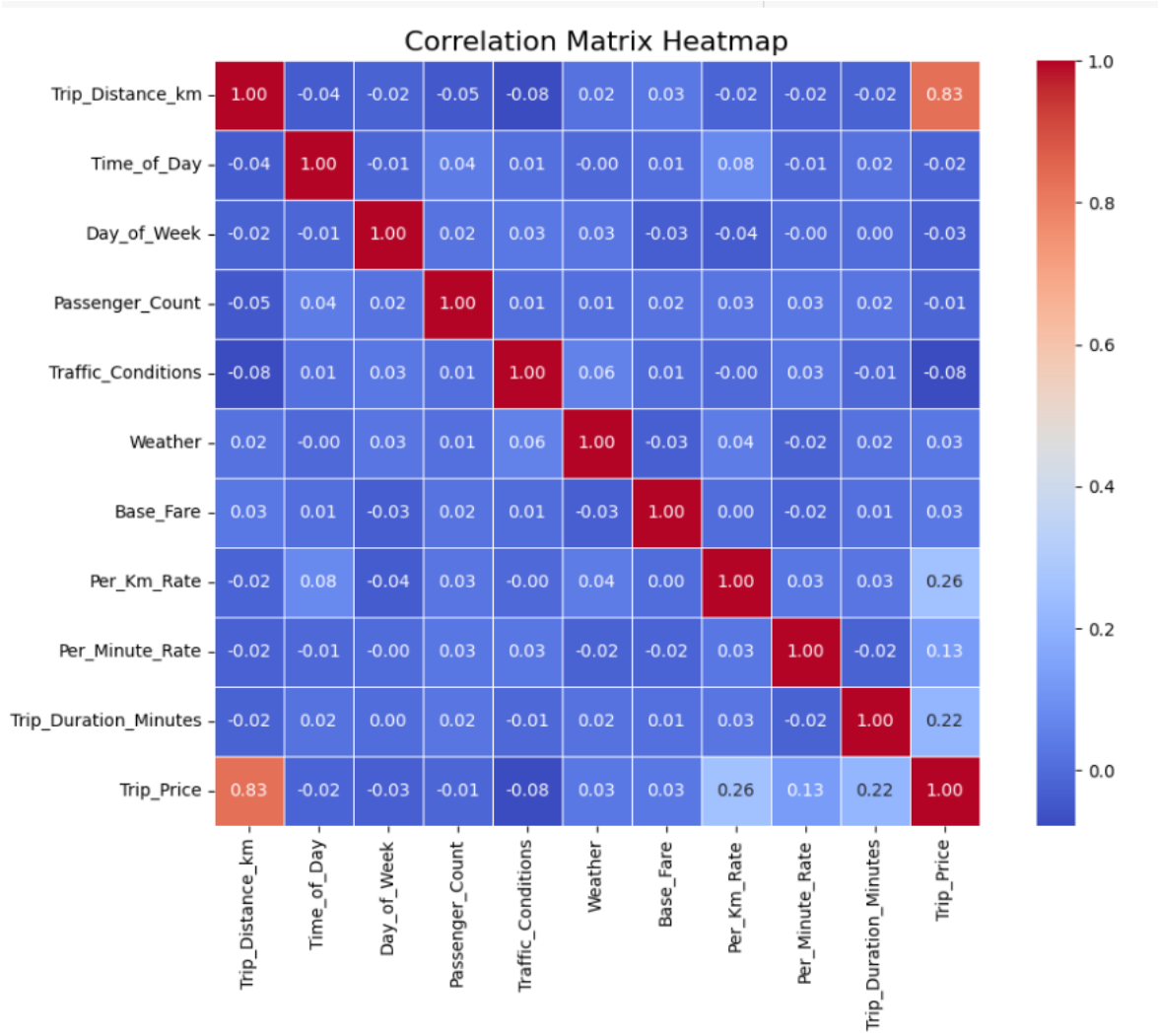
```
# Initialize the Decision Tree model
decision_tree_model = DecisionTreeRegressor(random_state=42)

# Train the model
decision_tree_model.fit(X_train, y_train)

# Make predictions on the test set
y_tree_pred = decision_tree_model.predict(X_test)

# Evaluate the model
tree_mse = mean_squared_error(y_test, y_tree_pred)
tree_mae = mean_absolute_error(y_test, y_tree_pred)
tree_r2 = r2_score(y_test, y_tree_pred)
tree_accuracy = tree_r2 * 100  # Accuracy as R² percentage

# Print evaluation metrics
print("Decision Tree Regression Model Evaluation:")
print(f"Mean Squared Error (MSE): {tree_mse:.2f}")
print(f"Mean Absolute Error (MAE): {tree_mae:.2f}")
print(f"R² Score: {tree_r2:.2f}")
print(f"Model Accuracy (R² as percentage): {tree_accuracy:.2f}%")

# Visualization of Actual vs Predicted Prices
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_tree_pred, alpha=0.7, color='green')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', linewidth=2)
plt.title("Decision Tree Regression: Actual vs Predicted Trip Prices", fontsize=16)
plt.xlabel("Actual Prices", fontsize=14)
```

```
plt.ylabel("Predicted Prices", fontsize=14)
plt.show()


import pandas as pd
import matplotlib.pyplot as plt


# Store evaluation metrics for all models
model_results = {
    'Model': ['Linear Regression', 'Polynomial Regression', 'Decision Tree', 'Ridge Regression'],
    'MSE': [mse, poly_mse, tree_mse,  ridge_mse],
    'MAE': [mae, poly_mae, tree_mae,  ridge_mae],
    'R²': [r2, poly_r2, tree_r2,  ridge_r2],
    'Accuracy (%)': [accuracy, poly_accuracy, tree_accuracy, ridge_accuracy]
}


# Convert to DataFrame
results_df = pd.DataFrame(model_results)


# Display the table
print("Model Evaluation Summary:")
print(results_df)


# Visualize the metrics
plt.figure(figsize=(16, 8))


# Accuracy Comparison
plt.subplot(1, 2, 1)
plt.bar(results_df['Model'], results_df['Accuracy (%)'], color=['blue', 'green', 'orange', 'purple'], alpha=0.7)
plt.title('Model Accuracy Comparison', fontsize=16)
```

```
plt.xlabel('Models', fontsize=12)
plt.ylabel('Accuracy (%)', fontsize=12)
plt.ylim(0, 100)

# MSE Comparison
plt.subplot(1, 2, 2)
plt.bar(results_df['Model'], results_df['MSE'], color=['blue', 'green', 'orange', 'purple'], alpha=0.7)
plt.title('Model MSE Comparison', fontsize=16)
plt.xlabel('Models', fontsize=12)
plt.ylabel('Mean Squared Error (MSE)', fontsize=12)

plt.tight_layout()
plt.show()

# Scatter Plots of Actual vs Predicted for Comparison
plt.figure(figsize=(20, 10))

# Linear Regression
plt.subplot(2, 2, 1)
plt.scatter(y_test, y_pred, alpha=0.7, color='blue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', linewidth=2)
plt.title("Linear Regression: Actual vs Predicted", fontsize=14)
plt.xlabel("Actual Prices", fontsize=12)
plt.ylabel("Predicted Prices", fontsize=12)

# Polynomial Regression
plt.subplot(2, 2, 2)
plt.scatter(y_test, y_poly_pred, alpha=0.7, color='green')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', linewidth=2)
plt.title("Polynomial Regression: Actual vs Predicted", fontsize=14)
```

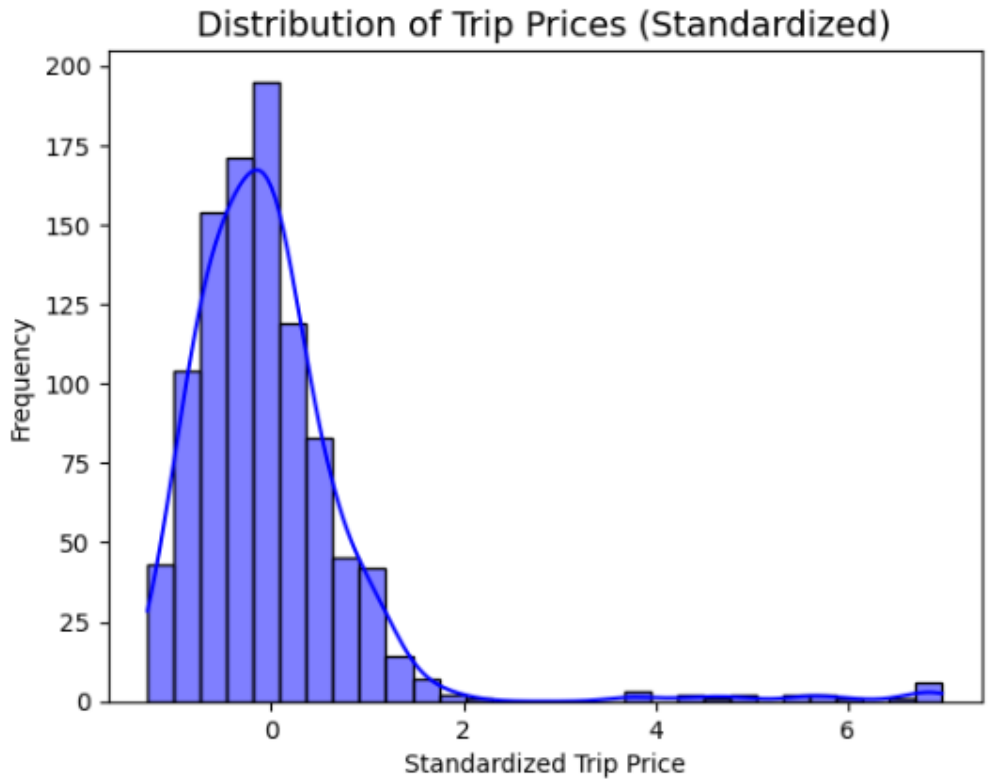
```
plt.xlabel("Actual Prices", fontsize=12)
plt.ylabel("Predicted Prices", fontsize=12)

# Decision Tree
plt.subplot(2, 2, 3)
plt.scatter(y_test, y_tree_pred, alpha=0.7, color='orange')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', linewidth=2)
plt.title("Decision Tree: Actual vs Predicted", fontsize=14)
plt.xlabel("Actual Prices", fontsize=12)
plt.ylabel("Predicted Prices", fontsize=12)

# Ridge Regression
plt.subplot(2, 2, 4)
plt.scatter(y_test, ridge_pred, alpha=0.7, color='purple')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', linewidth=2)
plt.title("Ridge Regression: Actual vs Predicted", fontsize=14)
plt.xlabel("Actual Prices", fontsize=12)
plt.ylabel("Predicted Prices", fontsize=12)

plt.tight_layout()
plt.show()
```

APPENDIX-2: SCREENSHOTS



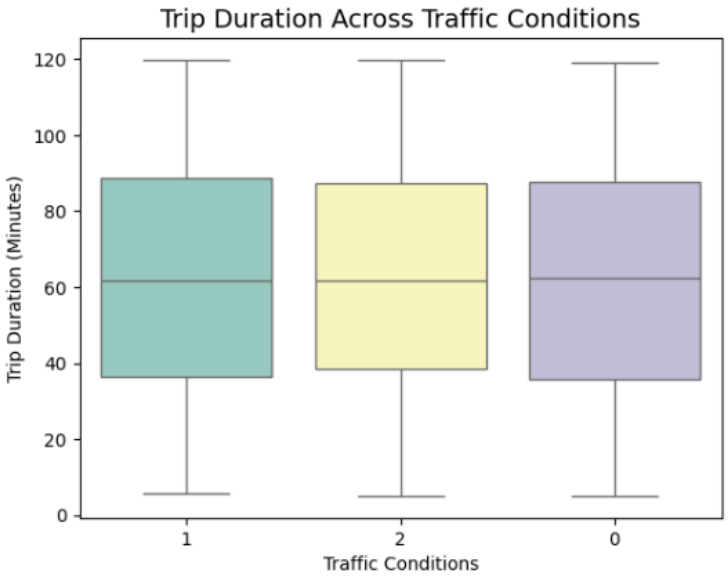
A1- Correlation Matrix Heatmap

```
Text(0, 0.5, 'Frequency')
```

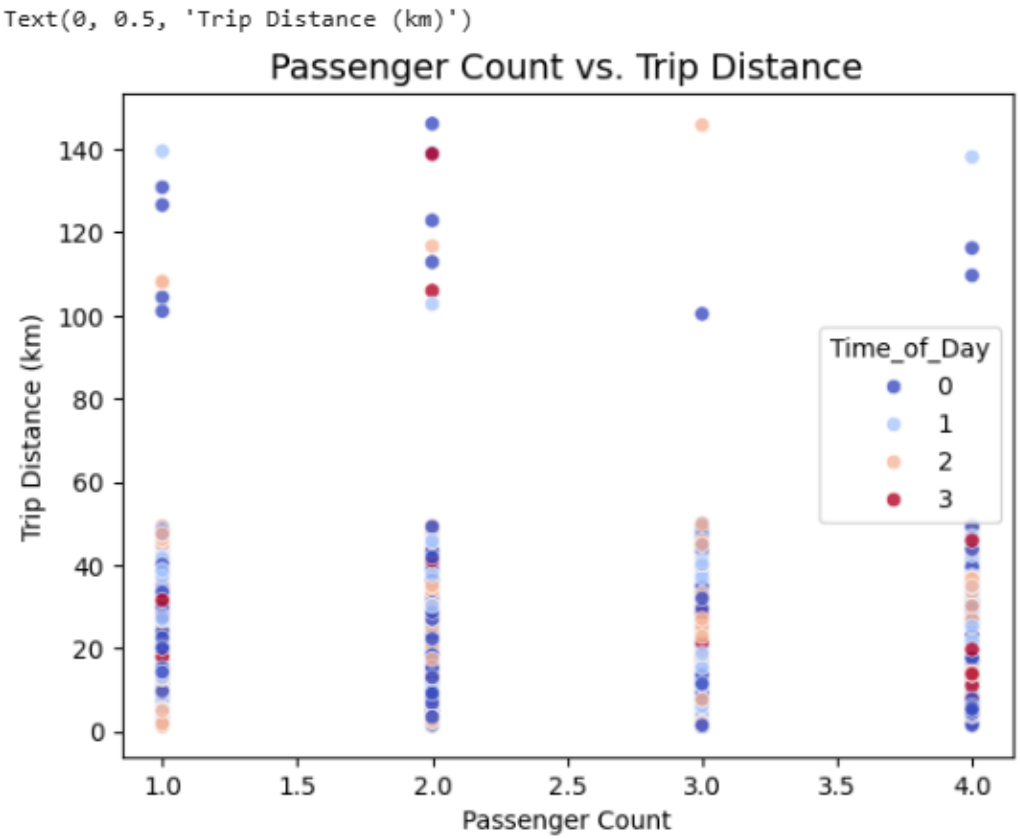


A2- Distribution of trip prices

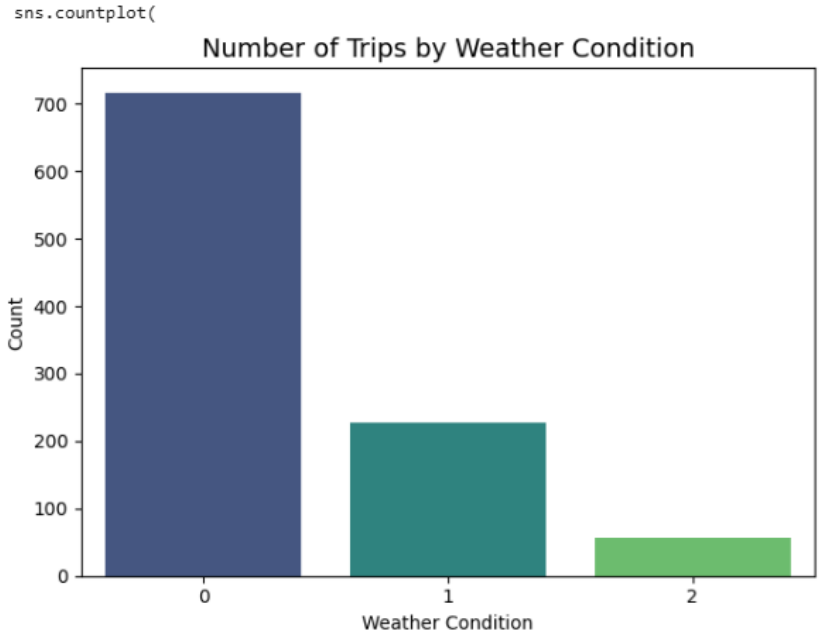
```
sns.boxplot(
Text(0, 0.5, 'Trip Duration (Minutes)')
```



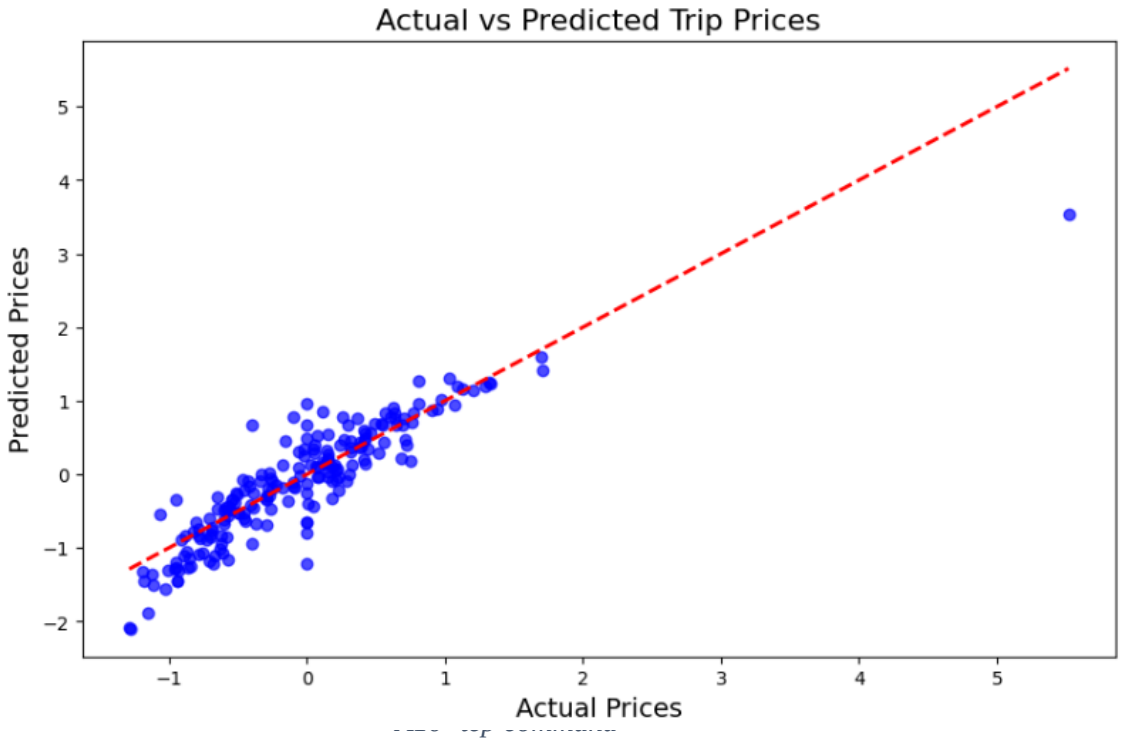
A3- Trip duration across traffic conditions



A4- Passenger Count VS Trip Distance

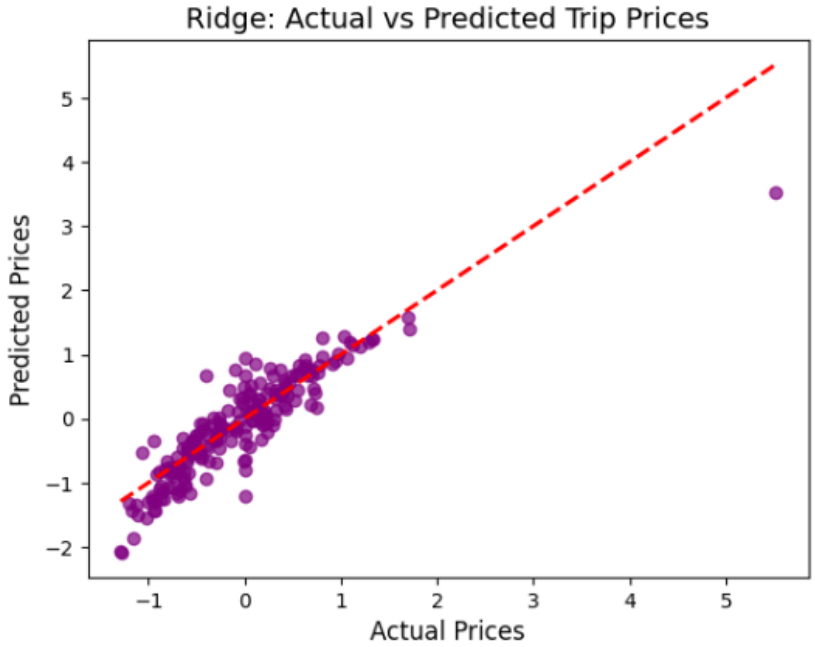


A5- Number of trips by weather condition



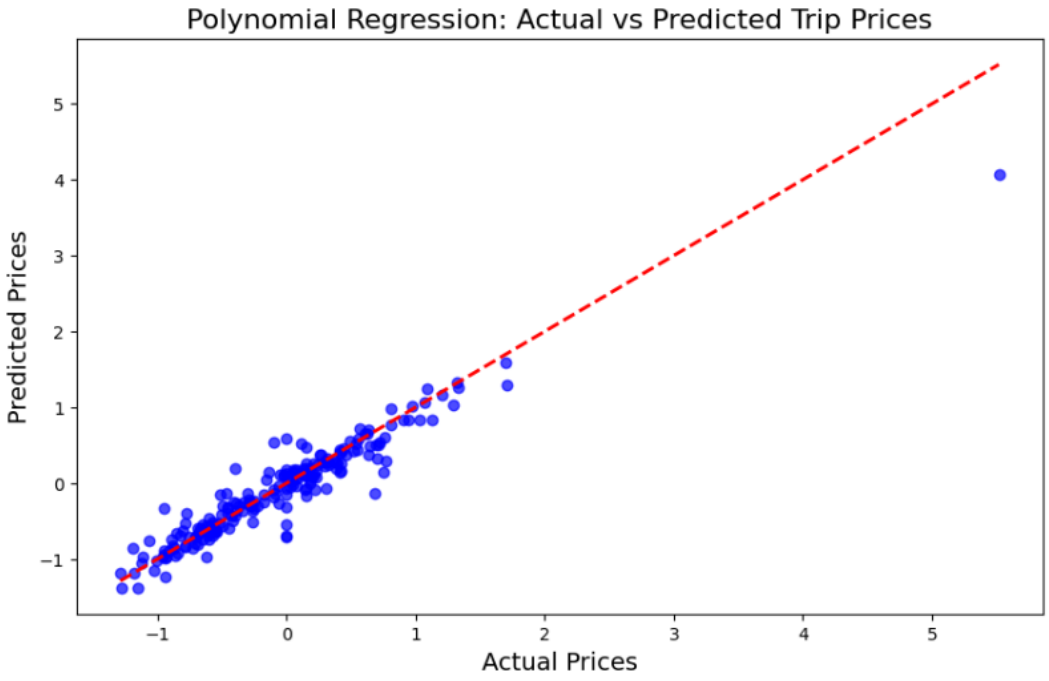
A6- Linear regression

```
Ridge Regression Model Evaluation:
Mean Squared Error (MSE): 0.12
Mean Absolute Error (MAE): 0.25
R² Score: 0.77
Model Accuracy (R² as percentage): 76.90%
```



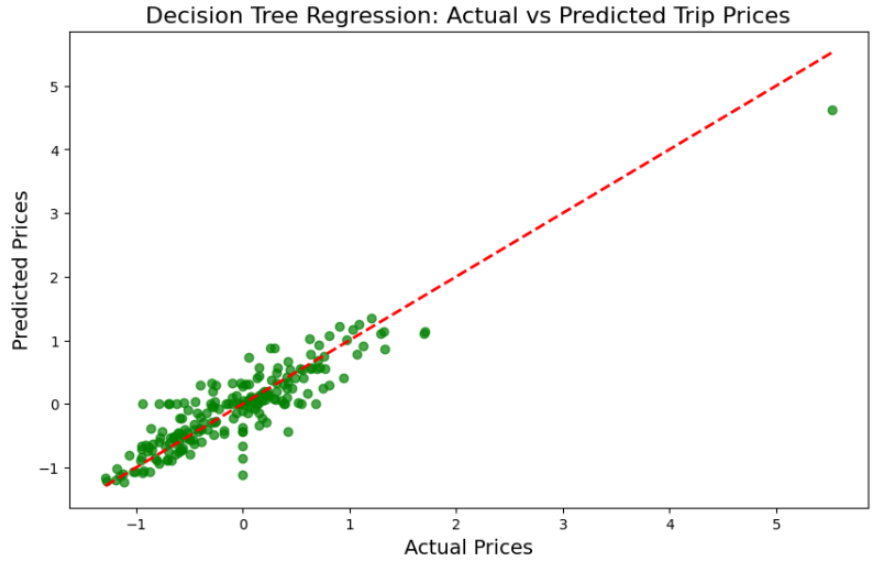
A7- Ridge regularization

```
Polynomial Regression Model Evaluation:
Mean Squared Error (MSE): 0.05
Mean Absolute Error (MAE): 0.14
R² Score: 0.90
Model Accuracy (R² as percentage): 90.48%
```



A8- Polynomial Regression

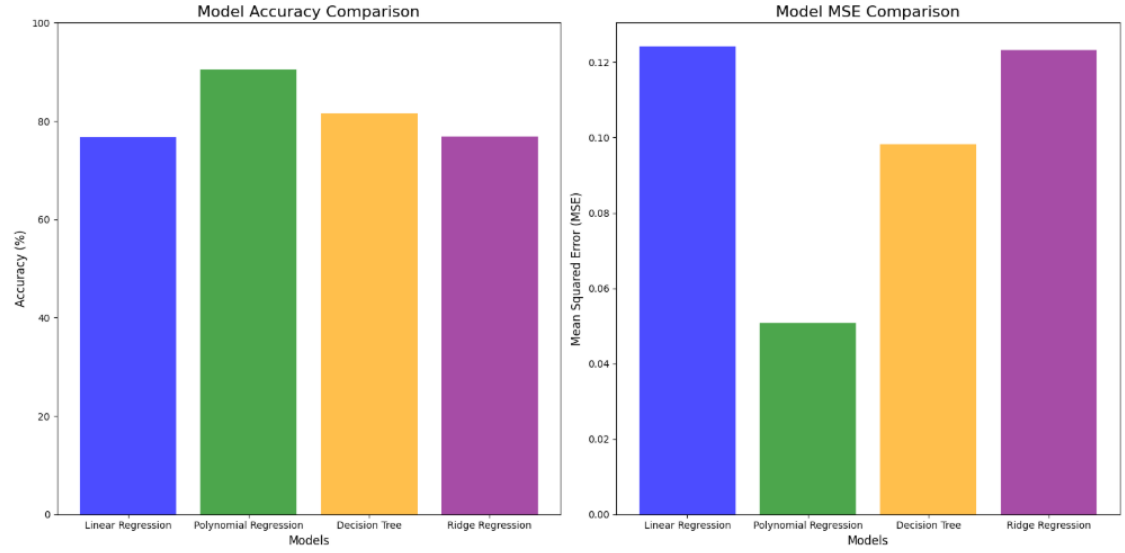
```
Decision Tree Regression Model Evaluation:
Mean Squared Error (MSE): 0.10
Mean Absolute Error (MAE): 0.23
R² Score: 0.82
Model Accuracy (R² as percentage): 81.59%
```



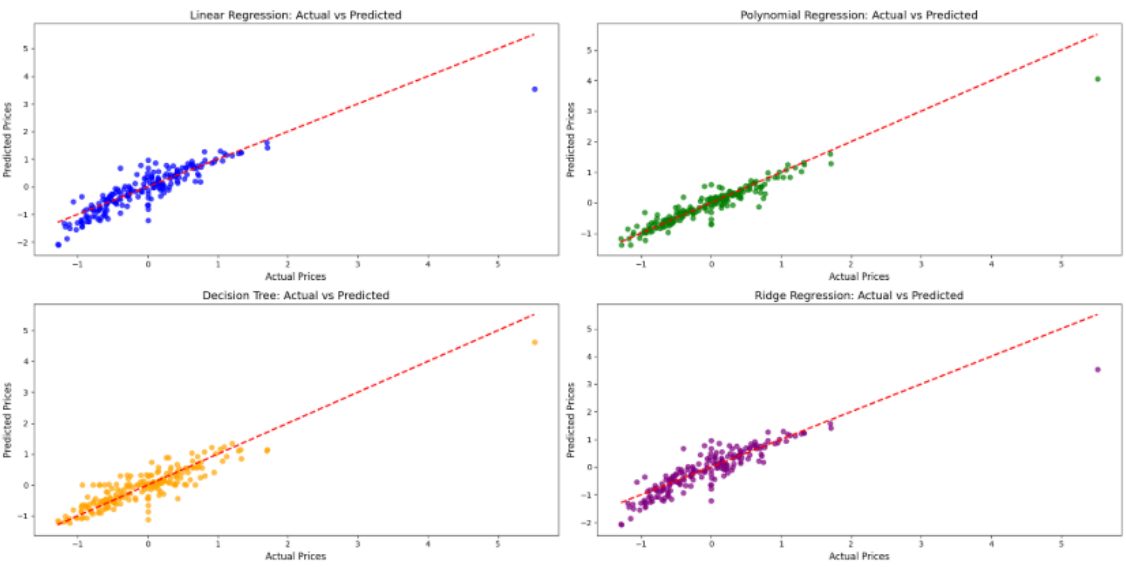
A9- Decision Tree

Model Evaluation Summary:

	Model	MSE	MAE	R^2	Accuracy (%)
0	Linear Regression	0.124267	0.250913	0.767151	76.715109
1	Polynomial Regression	0.050825	0.142595	0.904765	90.476491
2	Decision Tree	0.098249	0.231444	0.815903	81.590344
3	Ridge Regression	0.123276	0.249474	0.769008	76.900797



A10- Model Evaluation Summary



A11- Ml Models

REFERENCES

Journal References

1. Poongodi, M., Malviya, M., Kumar, C., Hamdi, M., Vijayakumar, V., Nebhen, J. and Alyamani, H., 2022. New York City taxi trip duration prediction using MLP and XGBoost. *International Journal of System Assurance Engineering and Management*, pp.1-12.
2. VinayKumar, K. and Santosh, N.C., 2024. Data Analysis and Fair Price Prediction Using Machine Learning Algorithms. *Journal of Computer Allied Intelligence (JCAI, ISSN: 2584-2676)*, *2*(1), pp.26-45.
3. Rajendran, S., Srinivas, S. and Grimshaw, T., 2021. Predicting demand for air taxi urban aviation services using machine learning algorithms. *Journal of Air Transport Management*, *92*, p.102043.
4. AMADXARIF, Z. and OTTER, N., 2023. PREDICTING NEW YORK CITY TAXI FARES WITH SUPERVISED MACHINE LEARNING.
5. Khandelwal, K., Sawarkar, A. and Hira, S., 2021. A Novel Approach for Fare Prediction Using Machine Learning Techniques. *International Journal of Next-Generation Computing*, *12*(5).

Web References

1. (13) (PDF) A Review on Linear Regression Comprehensive in Machine Learning
2. A Comprehensive Study of Regression Analysis and the Existing Techniques | IEEE Conference Publication | IEEE Xplore