RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

TAXI FARE PREDICTION USING XGBOOST AND RANDOM FOREST

SUBMITTED BY

HARIHAR T (231801048)

GUGAN M (231801046)

AI23331 - FUNDAMENTALS OF MACHINE LEARNING

Department of Artificial Intelligence and Data Science

Rajalakshmi Engineering College, Thandalam

Nov 2024

RAJALAKSHMI
ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

NAME .....GUGAN M..... ACADEMIC

YEAR.....2024-2025.....SEMESTER...III..... BRANCH .....AI&DS.....

UNIVERSITY REGISTER No. **2116231801046**

Certified that this is the bonafide record of work done by the above students in the Mini Project titled "**TAXI FARE PREDICTION USING XGBOOST AND RANDOM FOREST** " in the subject **AI23331 – FUNDAMENTALS OF MACHINE LEARNING** during the year **2024 - 2025.**

Signature of Faculty – in – Charge

Submitted for the Practical Examination held on -------23.11.2024----------

Internal Examiner **External Examiner**

TABLE OF CONTENTS

ABSTRACT

This project aims to develop a predictive model for estimating taxi fares based on various factors such as distance, route complexity, traffic conditions, weather severity, and demand levels. The dataset used in this study includes features related to the taxi ride, such as the time of day, day of the week, driver availability, event indicators, and trip duration, with the target variable being the base fare of the ride. Data preprocessing involved handling missing values, standardizing numerical features, and performing one-hot encoding on categorical variables to prepare the dataset for model training.

Exploratory Data Analysis (EDA) was conducted to identify patterns and relationships between the features and the base fare. Correlation analysis and data visualization techniques were used to uncover key factors influencing fare prediction, such as distance, route complexity, and traffic delays.

For the predictive modeling, an ensemble approach was employed, combining XGBoost and Random Forest models with a final linear regression estimator in a stacking regressor framework. The model was trained on a subset of the data and evaluated using common regression metrics, including Mean Absolute Error (MAE), Mean Squared Error (MSE), and R^2 score. The results indicated that the model performed reasonably well in predicting taxi fares, with residuals analysis showing no significant bias in predictions.

This project demonstrates the effectiveness of machine learning techniques in predicting taxi fares, providing a tool that can assist both passengers and taxi service providers in estimating costs more accurately. Future work could focus on hyperparameter optimization, incorporation of real-time data (e.g., live traffic), and improvements in model generalization for more robust predictions.

Introduction

1.1 General

Taxi fare prediction is a complex yet essential task that combines data science, machine learning, and transportation analytics. The goal is to estimate the fare of a taxi ride based on measurable and quantifiable factors such as trip distance, traffic conditions, weather, and demand. Traditional methods of fare estimation often rely on static rate cards or simple distance-based calculations, which may not account for dynamic factors like traffic delays or special events. With advancements in machine learning and deep learning, it has become possible to develop more accurate, real-time fare prediction systems that offer consistency and scalability.

Machine learning techniques, particularly ensemble and deep learning models, are capable of capturing complex relationships in large datasets and are widely applied to regression problems. Predicting taxi fares involves training models on datasets that include various trip-related features such as distance, route complexity, traffic delays, time of day, and other contextual factors.

1.2 Need for Study

Accurate taxi fare prediction is critical for multiple stakeholders in the transportation ecosystem. For passengers, it provides transparency and helps in budgeting; for service providers, it aids in optimizing pricing strategies and resource allocation. Traditional fare estimation methods often fail to account for real-time variations in traffic, demand, and other external factors, leading to inaccurate predictions and dissatisfaction among users.

An automated system powered by machine learning can enhance fare estimation accuracy by considering a wide range of contextual variables. This reduces dependency on static or manual methods, improves customer trust, and increases operational efficiency for service providers.

1.3 Overview of the Project

This project aims to develop a machine learning-based model to predict taxi fares using a dataset that includes features such as trip distance, route complexity, traffic delay, weather severity, and time-based indicators. The approach followed in this project involves the following key steps:

1. **Data Preprocessing**: Cleaning and preparing the dataset by handling missing values, standardizing numerical features, and encoding categorical variables.
2. **Feature Engineering**: Identifying and selecting key factors influencing fare estimation.
3. **Model Training and Evaluation**: Training an ensemble model using machine learning techniques like XGBoost, Random Forest, and Stacking Regressor, followed by performance evaluation using regression metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R^2 score.
4. **Result Analysis**: Analyzing the model's performance, visualizing residuals, and identifying areas for improvement.

1.4 Objective of the Study

The primary objectives of this study are:

1. To design and implement a machine learning model capable of predicting taxi fares based on trip-related features.
2. To evaluate the model’s performance using regression metrics like MAE, MSE, and R^2 score.
3. To analyze the influence of various factors, such as traffic delays and weather, on fare prediction.
4. To demonstrate how machine learning techniques can improve real-world applications in the transportation industry.

ALGORITHM USED

Ensemble Learning

Ensemble learning is a robust machine learning technique that combines predictions from multiple models to improve overall performance. In this project, ensemble learning is employed to predict taxi fares based on features such as trip distance, traffic delays, weather conditions, and demand levels. The ensemble model combines the strengths of multiple algorithms to enhance prediction accuracy and reduce errors in fare estimation.

How Ensemble Learning Works

Ensemble models leverage the predictive capabilities of multiple base learners to make more accurate and reliable predictions. In this project, a stacking regressor is used, combining two powerful regression models—XGBoost and Random Forest—with a linear regression model as the final estimator.

The stacking regressor works as follows:

1. **Base Learners**: Individual models, such as XGBoost and Random Forest, are trained on the dataset and make initial predictions.
2. **Meta-Learner**: A final model (linear regression) takes the outputs of the base learners as inputs and learns to combine them optimally to improve predictions.

The general equation for the stacking model is:

y^=g(h1(X),h2(X),…,hn(X))\hat{y} = g(h_1(X), h_2(X), \dots, h_n(X))y^=g(h1(X),h2(X),…,hn(X))

Where:

- y^\hat{y}y^ is the predicted taxi fare.
- h1(X),h2(X),…,hn(X)h_1(X), h_2(X), \dots, h_n(X)h1(X),h2(X),…,hn(X) are the predictions from the base models.
- ggg is the meta-learner that combines the base model predictions.

By combining the strengths of XGBoost (handling nonlinear relationships) and Random Forest (robust to overfitting), the stacking model achieves better performance.

Model Training

1. Data Preparation

The dataset includes features such as trip distance, route complexity, traffic delay, weather severity, and time-related variables like day of the week and time of day. Data preprocessing involves:

- Handling missing values to ensure data quality.
- Normalizing numerical features to standardize their scale.
- Encoding categorical variables (e.g., time of day) using one-hot encoding.
- Splitting the dataset into training and testing sets for model evaluation.

The features (X) are used as inputs to the model, while the target variable (y) represents the base fare of the taxi ride.

2. Model Architecture

The ensemble model consists of:

- **XGBoost Regressor**: A gradient-boosted decision tree model, effective for capturing complex patterns in data.
- **Random Forest Regressor**: An ensemble of decision trees,

known for its robustness and accuracy in regression tasks.

- **Linear Regression**: A simple but effective model used as the meta-learner to combine predictions from XGBoost and Random Forest.

The stacking regressor uses the predictions from XGBoost and Random Forest as inputs to train the linear regression model, resulting in a more accurate final prediction.

3. Training the Model

The training process involves:

1. Training the base learners (XGBoost and Random Forest) on the training dataset.
2. Using the predictions from the base learners to train the meta-learner (Linear Regression).
3. Optimizing the model parameters to minimize the Mean Squared Error (MSE) and improve prediction accuracy.

Evaluation

The performance of the ensemble model is evaluated using the following metrics:

- **Mean Absolute Error (MAE)**: Measures the average magnitude of errors between predicted and actual fares, providing an easily interpretable metric in fare units.
- **Mean Squared Error (MSE)**: Penalizes larger errors more heavily, helping assess the overall model accuracy.
- **R^2 Score**: Indicates how well the model explains the variance in taxi fares. A value close to 1 implies a highly accurate model.

Additional Analysis:

- **Residual Analysis**: Evaluates the differences between

actual and predicted fares to detect biases or inconsistencies in the model.

- **Feature Importance**: Assesses the contribution of each feature (e.g., distance, traffic delay) to the predictions, offering insights into the factors influencing taxi fares.

By leveraging ensemble learning, this project demonstrates how advanced machine learning techniques can enhance the accuracy of fare predictions

SYSTEM ARCHITECTURE

2. System Requirements

2.1 Hardware Requirements

The following hardware is required to run the machine learning models for taxi fare prediction:

1. **CPU**: Any modern multi-core processor (Intel i5 or higher recommended).
2. **RAM**: A minimum of 8 GB of RAM (16 GB or more is recommended for faster processing and model training).
3. **GPU**: An NVIDIA GPU (with CUDA support) is optional but recommended for faster training, especially when working with large datasets or deep learning models.
4. **Storage**: A minimum of 10 GB of free disk space to store datasets, model files, and logs.
5. **Internet Access**: Required to download datasets, pre-trained models, and libraries.

2.2 Software Requirements

The software requirements for the taxi fare prediction project include:

1. **Operating System**: Compatible with any modern OS such as Windows, macOS, or Linux.
2. **Python**: Python 3.6 or higher is required. The project was developed using Python 3.8.
3. **Libraries**:
    - **Scikit-learn**: For data preprocessing, model training, and evaluation.

- **XGBoost**: For training the gradient boosting model.
- **Random Forest**: Included as part of the Scikit-learn library.
- **Pandas**: For data manipulation and preprocessing.
- **NumPy**: For numerical computations.
- **Matplotlib/Seaborn**: For visualizing data and analysis results.
- **Joblib**: For saving and loading trained models.

4. **IDE**:
   Any Python Integrated Development Environment (IDE) or code editor such as Visual Studio Code, PyCharm, or Jupyter Notebook.

5. **Optional Tools**:

   - **Anaconda**: For managing Python environments and dependencies.
   - **Google Colab**: For running the code with access to free GPU resources.

This combination of hardware and software ensures efficient execution of the machine learning pipeline, from data preprocessing to model training and evaluation.

SYSTEM OVERVIEW

3.1 SYSTEM ARCHITECTURE

3.1 Module

3.2 Module 1: Data Collection

The New York City (NYC) Taxi Fare Prediction Dataset was sourced from Kaggle. The dataset contains millions of records with features such as pickup and drop-off locations, passenger count, distance traveled, and time of day. The target variable is the taxi fare amount, a continuous value representing the cost of a taxi ride in U.S. dollars.

Steps for Data Collection:

1. **Download the dataset from the Kaggle platform.**
2. **Load the dataset into Python using the Pandas library.**
3. **Perform data cleaning, including handling missing values, identifying and removing outliers, and ensuring consistency in the data.**

3.3 Module

2: Model Development

This module involves building predictive models using Random Forest (RF) and XGBoost algorithms to estimate taxi fare based on the input features. These ensemble-based models are known for their robustness and ability to handle complex relationships within the data.

Model Architecture:

1. **Input Features:**

- **Latitude and longitude of pickup and drop-off locations.**
- **Passenger count.**
- **Date and time of the trip (used to extract features like hour, day, and weekday).**
- **Distance traveled (calculated using Haversine or Manhattan distance).**

2. **Model 1:**

   Random Forest

   - **Ensemble method that uses multiple decision trees to generate predictions.**
   - **Hyperparameters such as the number of trees and maximum depth are tuned for optimal performance.**

3. **Model 2:**

   XGBoost

   - **Gradient boosting algorithm optimized for speed and performance.**
   - **Important hyperparameters include the learning rate, max depth, and the number of estimators.**

4. **Training and Evaluation:**

   - **The models are trained on the cleaned dataset using an 80/20 train-test split.**
   - **Cross-validation is used to ensure the model generalizes well.**

System Architecture for Taxi Fare Prediction Using Random Forest and XGBoost

1. Data Collection Layer

- **Data Source:**
  The NYC Taxi Fare dataset contains features such as pickup and drop-off coordinates, time of trip, and passenger count.
- **Data Input:**
  The input data includes geographical coordinates, temporal details, and contextual information to predict the fare amount.

2. Data Preprocessing Layer

- **Data Cleaning:**
  - **Handle missing values (e.g., missing coordinates or fares).**
  - **Remove outliers, such as trips with excessively high or low fares or unrealistic distances.**
  - **Ensure valid geographical bounds for latitude and longitude.**
- **Feature Engineering:**
  - **Calculate trip distance using Haversine formula.**
  - **Extract temporal features like hour, weekday, and season from the timestamp.**
  - **Create new features such as airport proximity, which can significantly influence fares.**
- **Data Splitting:**
  - **The dataset is split into training (80%) and testing (20%) sets to evaluate model performance.**

3. Model Training Layer

- **Random Forest Model:**
  - **A robust ensemble-based method using multiple decision trees.**
  - **Handles non-linear relationships effectively and performs well with minimal preprocessing.**
- **XGBoost Model:**
  - **An optimized gradient boosting algorithm known for handling sparse data and delivering high accuracy.**
  - **Hyperparameters such as learning rate, max depth, and regularization parameters are tuned using grid search or random search techniques.**
- **Model Fitting:**
  - **Both models are trained using mean squared error (MSE) as the loss function, minimizing the difference between predicted and actual fare amounts.**

4. Model Evaluation Layer

- **Performance Metrics:**
  - **Root Mean Squared Error (RMSE): Measures the average deviation of predicted fares from actual fares.**
  - **R-squared (R^2): Indicates the proportion of variance in the fare amount explained by the model.**
  - **Mean Absolute Error (MAE): Measures the average absolute difference between predicted and actual fares.**

- **Cross-validation:**
    - **The models are validated using k-fold cross-validation to assess their ability to generalize to unseen data.**

5. Prediction Layer

- **User Input:**

  New data (e.g., pickup and drop-off locations, passenger count, and time) is provided through a user interface or API.

- **Prediction:**

  The trained models (Random Forest and XGBoost) use the input features to predict the taxi fare amount.
    - **The output is a continuous value representing the estimated taxi fare.**
    - **Both models may be compared, and the best-performing one is selected for deployment.**

Tools And Requirment

1. Python

Python is the core programming language for your project. It's a versatile and widely-used language in machine learning and data science due to its simplicity and vast ecosystem of libraries.

2. Pandas

Pandas is used for data manipulation and preprocessing tasks. It helps in reading the dataset, handling missing values, encoding categorical features, and performing other data transformations necessary before feeding the data into a deep learning model.

3. NumPy

NumPy is essential for numerical computations. It handles large arrays and matrices and provides functions for mathematical operations required for data processing, such as feature scaling and normalization.

4. TensorFlow / Keras

TensorFlow, along with its high-level API **Keras**, is the deep learning framework used to build and train the neural network model. TensorFlow provides efficient tools for model construction, training, and evaluation, and Keras simplifies the creation of neural networks.

5. Scikit-learn

Scikit-learn is used for tasks like data splitting (train-test split) and evaluating model performance with metrics like accuracy and confusion matrix. It also helps with preprocessing tasks and comparing the deep learning model to simpler models (e.g., logistic regression).

Algorithm for Taxi Fare Prediction Using Machine Learning

Step 1: Data Collection

- **Source the Dataset:** Obtain the NYC Taxi Fare dataset from a reliable platform such as Kaggle or an open government dataset.
- **Dataset Contents:** Ensure the dataset includes features like pickup and drop-off coordinates, date and time, passenger count, and the target variable (fare amount).

Step 2: Data Preprocessing

- **Handle Missing Data:**
  - Fill missing numerical values (e.g., passenger count) with the median or mean.
  - Remove rows with missing or incorrect fare amounts (e.g., negative values).
- **Remove Outliers:**
  - Filter out unrealistic fares or distances (e.g., trips with zero distance but high fare).
- **Feature Engineering:**
  - Calculate trip distance using latitude and longitude (Haversine or Manhattan formula).
  - Extract additional features like day of the week, hour, and month from the trip timestamp.
  - Flag proximity to key landmarks (e.g., airports).
- **Scale Features:** Normalize numerical columns (e.g., distance, passenger count) to bring them to a similar scale.

Step 3: Feature Selection

- Identify and retain the most relevant features impacting taxi fare, such as distance, passenger count, time of day, and proximity to high-demand areas.

Step 4: Train-Test Split

- Split the dataset into two parts:
    - **Training Set:** 80% of the data for model training.
    - **Testing Set:** 20% of the data for model evaluation.

Step 5: Model Construction

- Build machine learning models using Random Forest and XGBoost algorithms:
    - **Random Forest:** A tree-based ensemble model that uses multiple decision trees to improve accuracy and robustness.
    - **XGBoost:** A gradient boosting model optimized for speed and performance, handling sparse and complex data effectively.
    - Set the loss function to **Mean Squared Error (MSE)** for regression tasks.

Step 6: Model Training

- Train both models on the training dataset.
- Use techniques like cross-validation to avoid overfitting.
- Tune hyperparameters such as the number of estimators, learning rate, and maximum depth to improve model performance.

Step 7: Model Evaluation

- Evaluate the trained models on the test dataset using these metrics:
    - **Root Mean Squared Error (RMSE):** Quantifies the average deviation of predicted fares from actual fares.
    - **R-squared (R^2):** Measures the proportion of variance in fare explained by the model.
    - **Mean Absolute Error (MAE):** Calculates the average absolute difference between predicted and actual fares.
- Compare the performance of Random Forest and XGBoost.

Step 8: Visualization

- **Plot Results:**
    - Graph the relationship between predicted and actual fares to visualize performance.
    - Create feature importance plots to highlight the most significant predictors.

    - Generate residual plots to identify any patterns in prediction errors.
- **Comparison Visualization:**
    - Display comparative bar charts for RMSE, MAE, and R^2 across the two models.

Step 9: Model Improvement (Optional)

- If performance is not satisfactory:
    - Add new features (e.g., weather conditions, traffic data).
    - Fine-tune hyperparameters further or use advanced optimization techniques.
    - Experiment with ensemble techniques like stacking or blending.

Step 10: Conclusion

- Summarize the model's performance, including accuracy metrics and feature importance.
- Highlight strengths, such as model robustness or speed.
- Suggest areas for improvement (e.g., integrating real-time data or using neural networks for further optimization).

RESULTS AND DISCUSSION

The taxi fare prediction models using Random Forest and XGBoost demonstrated strong performance in predicting fares based on trip characteristics such as distance, time, passenger count, and geographic features. Both models provided accurate predictions, with XGBoost slightly outperforming Random Forest in terms of lower Root Mean Squared Error (RMSE) and higher R-squared (R^2) values.

A detailed analysis of feature importance revealed that trip distance, time of day, and pickup location were the most significant predictors of taxi fare. This aligns with domain knowledge, as fare calculations typically depend heavily on distance traveled and timing (e.g., peak vs. off-peak hours). Other features, such as passenger count and drop-off proximity to airports or landmarks, had a relatively smaller impact but still contributed to the prediction accuracy.

The results indicate that both Random Forest and XGBoost are effective at capturing the nonlinear relationships between input features and taxi fares. However, the evaluation metrics and residual plots showed that the models struggled with extreme fares, such as outliers from unusually short or long trips. While the models performed well on the test set, there is potential for improvement. Incorporating real-time data, such as traffic conditions and weather, could improve predictive accuracy further. Additionally, using ensemble methods or blending predictions from multiple models might address some limitations in handling outliers.

In conclusion, both models demonstrated strong performance in predicting taxi fares, providing a robust foundation for deployment in applications like ride-sharing fare estimations or automated billing systems.

CONCLUSION

The taxi fare prediction project successfully demonstrated the effectiveness of machine learning models—Random Forest and XGBoost—in estimating fares based on trip characteristics. By leveraging features such as trip distance, time, and geographic coordinates, the models achieved high accuracy and provided reliable predictions.

Key features like trip distance and time proved to be critical determinants of fare, validating their inclusion in the predictive framework. The advanced XGBoost model, with its capability to handle complex data relationships and outliers, slightly outperformed Random Forest, confirming its suitability for regression tasks involving structured datasets.

Despite the success of the models, there remains room for enhancement. Future iterations could benefit from:

1. **Hyperparameter Tuning:** Fine-tuning model parameters to optimize performance.
2. **Feature Enrichment:** Adding external data, such as real-time traffic or weather conditions, to better capture external influences on fare.
3. **Advanced Architectures:** Exploring deep learning models or ensemble techniques to further improve prediction accuracy and handle edge cases effectively.

Overall, this project underscores the importance of feature engineering, robust model selection, and evaluation in developing practical solutions for real-world applications. The models developed here provide a scalable and efficient framework for taxi fare prediction, with potential applications in transportation analytics, ride-sharing platforms, and urban planning.

6.1 SOURCE CODE

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.ensemble import StackingRegressor, RandomForestRegressor
from sklearn.linear_model import LinearRegression
from xgboost import XGBRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import joblib  # For saving the model
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
data = pd.read_csv('/content/generated_taxi_fare_dataset.csv')

# Step 1: Data Preprocessing
data.dropna(inplace=True)

# One-Hot Encoding for Categorical Variables
data = pd.get_dummies(data, columns=['time_of_day'], drop_first=True)

# Normalize/Standardize Numerical Features
scaler = StandardScaler()
numerical_features = [
    'distance_km', 'route_complexity', 'traffic_delay_min',
    'weather_severity', 'day_of_week', 'demand_level',
    'driver_availability', 'event_indicator', 'trip_duration'
]
data[numerical_features] = scaler.fit_transform(data[numerical_features])

# Feature and Target Separation
X = data.drop(columns=['base_fare'])
y = data['base_fare']

# Step 2: Split Dataset
train_test_data = X.iloc[:-50]  # First 9950 rows for training/testing
train_test_target = y.iloc[:-50]
input_data = X.iloc[-50:]  # Last 50 rows for prediction
input_target = y.iloc[-50:]

X_train, X_test, y_train, y_test = train_test_split(train_test_data, train_test_target, test_size=0.2,
random_state=42)
```

```
# Step 3: Model Definition
# XGBoost Model
xgb_model = XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=6,
random_state=42)
xgb_model.fit(X_train, y_train)

# Random Forest Model
rf_model = RandomForestRegressor(n_estimators=100, max_depth=6, random_state=42)
rf_model.fit(X_train, y_train)

# Ensemble Model
ensemble_model = StackingRegressor(
    estimators=[('xgb', xgb_model), ('rf', rf_model)],
    final_estimator=LinearRegression()
)
ensemble_model.fit(X_train, y_train)

# Save Models
joblib.dump(ensemble_model, 'ensemble_model.pkl')
joblib.dump(scaler, 'scaler.pkl')

# Step 4: Evaluate the Model
y_pred_test = ensemble_model.predict(X_test)

# Metrics
mae = mean_absolute_error(y_test, y_pred_test)
mse = mean_squared_error(y_test, y_pred_test)
r2 = r2_score(y_test, y_pred_test)

print(f"Mean Absolute Error: {mae}")
print(f"Mean Squared Error: {mse}")
print(f"R² Score: {r2}")

# Step 5: Residual Analysis
residuals = y_test - y_pred_test

# Residuals Histogram
plt.figure(figsize=(8, 6))
sns.histplot(residuals, kde=True, bins=30, color='blue')
plt.title('Residuals Distribution')
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.show()

# Residuals vs Predicted
plt.figure(figsize=(8, 6))
plt.scatter(y_pred_test, residuals, alpha=0.5, color='orange')
plt.axhline(0, color='red', linestyle='--', linewidth=2)
plt.xlabel('Predicted Fares')
```

```
plt.ylabel('Residuals')
plt.title('Residuals vs Predicted Fares')
plt.show()

# Calculate the correlation matrix
correlation_matrix = data.corr()

# Plot the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, fmt='.2f', cmap='coolwarm', cbar=True)
plt.title('Feature Correlation Matrix')
plt.show()

# Step 7: Actual vs Predicted
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred_test, alpha=0.5, label='Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', label='Perfect Fit')
plt.xlabel('Actual Fares')
plt.ylabel('Predicted Fares')
plt.title('Actual vs Predicted Fares')
plt.legend()
plt.show()

# Step 8: Predict on Input Data
input_predictions = ensemble_model.predict(input_data)

# Save Input Data and Predictions
input_data.to_csv('input_data.csv', index=False)
pd.DataFrame(input_predictions, columns=['Predicted_Fare']).to_csv('predicted_fares.csv',
index=False)
```

REFERENCES

- UCI Machine Learning Repository,Taxi Fare Prediction dataset
- **Keras Documentation**: https://keras.io/
- **TensorFlow Documentation**: https://www.tensorflow.org/