**Assignment No.: ML 1**

**Title:** Predict the price of the Uber ride from a given pickup point to the agreed drop-off location. Perform following tasks:

1. Pre-process the dataset.

2. Identify outliers.

3. Check the correlation.

4. Implement linear regression and random forest regression models.

5. Evaluate the models and compare their respective scores like R2, RMSE, etc.

**Objective of the Assignment:** The objective of this assignment is to predict the price of an Uber ride given the pickup point and the agreed drop-off location. To achieve this, we will follow a structured approach that includes pre-processing the dataset, identifying outliers, checking the correlation between features, and implementing both linear regression and random forest regression models. Finally, we will evaluate and compare the models' performance using metrics such as R² and RMSE.

**Prerequisites:** Before embarking on this assignment, you should have:

* A basic understanding of Python programming.
* Knowledge of data manipulation and analysis using libraries like Pandas and NumPy.
* Familiarity with data visualization techniques using libraries like Matplotlib and Seaborn.
* An understanding of machine learning concepts and algorithms, particularly linear regression and random forest regression.
* Basic knowledge of statistical measures and evaluation metrics such as R² and RMSE.

**Contents for Theory:**

1. **Introduction to Uber Fare Prediction**

* Overview of the problem: Predicting Uber fares involves estimating the cost of a ride based on various factors such as the pickup and drop-off locations, distance traveled, time of day, and traffic conditions. Accurate fare predictions can enhance the user experience by providing transparency and helping users make informed decisions about their travel plans.
* Importance of fare prediction in ride-sharing services: Fare prediction is crucial for both ride-sharing companies and customers. For companies, it helps in optimizing pricing strategies and improving service efficiency. For customers, accurate fare estimates prevent price surprises and build trust in the service. It also aids in demand forecasting and resource allocation.

1. **Data Pre-processing**

* Handling missing values: Missing values in a dataset can skew analysis and model performance. Common techniques for handling missing values include removing rows with missing data, filling missing values with mean/median/mode, or using more advanced imputation methods.
* Converting data types: Data type conversion ensures that all features are in the appropriate format for analysis. For example, date-time fields need to be converted to datetime objects, and categorical variables may need to be encoded.
* Feature engineering (e.g., extracting date-time features): Feature engineering involves creating new features from existing data to improve model performance. For Uber fare prediction, useful features might include the time of day, day of the week, and month, which can be extracted from the timestamp of the ride.
* Data normalization and scaling: Normalization and scaling are techniques used to standardize the range of independent variables. This is especially important for algorithms that rely on distance calculations, like linear regression, to ensure that all features contribute equally to the model.

1. **Outlier Detection and Treatment**

* Definition of outliers: Outliers are data points that significantly differ from other observations. They can arise due to errors in data collection, recording errors, or genuine anomalies.
* Methods to identify outliers (e.g., Z-score, IQR):
  + - Z-score: Measures how many standard deviations a data point is from the mean.
    - Interquartile Range (IQR): Identifies outliers by considering the range between the first and third quartiles.Strategies to handle outliers (e.g., removal, transformation)

#### Strategies to Handle Outliers

* **Removal:** Deleting outliers from the dataset.
* **Transformation:** Applying mathematical transformations (e.g., log transformation) to reduce the impact of outliers.

1. **Correlation Analysis**

#### Understanding Correlation:

Correlation measures the relationship between two variables. It indicates how changes in one variable are associated with changes in another.

#### Methods to Compute Correlation

* **Pearson Correlation**: Measures linear relationships.
* **Spearman Correlation**: Measures monotonic relationships.

#### Interpreting Correlation Results

A correlation coefficient close to +1 or -1 indicates a strong relationship, while a coefficient close to 0 indicates no relationship.

#### Identifying Multicollinearity

Multicollinearity occurs when two or more independent variables are highly correlated. It can inflate the variance of coefficient estimates and make the model unstable.

1. **Linear Regression**

#### Introduction to Linear Regression

Linear regression is a statistical method that models the relationship between a dependent variable and one or more independent variables by fitting a linear equation.

#### Assumptions of Linear Regression

* + - Linearity: The relationship between the independent and dependent variables is linear.
    - Independence: Observations are independent.
    - Homoscedasticity: Constant variance of residuals.
    - Normality: Residuals should be normally distributed.

#### Implementing Linear Regression in Python

Using libraries such as Scikit-learn, one can easily implement linear regression to predict Uber fares.

#### Evaluating Model Performance (R², RMSE)

* + - **R² (Coefficient of Determination):** Indicates the proportion of the variance in the dependent variable that is predictable from the independent variables.
    - **RMSE (Root Mean Squared Error):** Measures the average magnitude of the errors between predicted and actual values.

1. **Random Forest Regression**

* Introduction to random forest regression:

Random forest regression is an ensemble learning method that operates by constructing multiple decision trees and outputting the mean prediction of the individual trees. It reduces overfitting and improves accuracy.

* Advantages and disadvantages of random forest regression

**Advantages:** Handles large datasets well, reduces overfitting, handles non-linear relationships.

**Disadvantages:** Can be computationally intensive, less interpretable compared to linear regression.

* Implementing random forest regression in Python

Using libraries such as Scikit-learn, one can implement random forest regression for fare prediction.

* Evaluating model performance (R², RMSE)

Evaluation metrics are similar to linear regression, with R² and RMSE used to assess the model's predictive performance.

1. **Model Comparison and Conclusion**

* Comparing the performance of linear regression and random forest regression models

Evaluate and compare the models based on R², RMSE, and other relevant metrics to determine which model performs better for Uber fare prediction.

* Interpretation of results

Analyze the results to understand which factors most influence fare prediction and the relative performance of each model.

* Conclusion and potential improvements

Summarize findings, highlight the best-performing model, and suggest potential improvements for future work, such as incorporating additional features or using more advanced machine learning techniques.

Implementation Algorithm:

Step 1: Import necessary libraries

|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  import seaborn as sns  from sklearn.model\_selection import train\_test\_split  from sklearn.linear\_model import LinearRegression  from sklearn.ensemble import RandomForestRegressor  from sklearn.metrics import r2\_score, mean\_squared\_error  from scipy import stats  # Load the dataset  df = pd.read\_csv('uber\_fares.csv') |

Step 2: Pre-process the dataset

* + - Check for missing values
    - Drop rows with missing target values
    - Convert pickup and dropoff datetime to datetime objects
    - Extract useful features from datetime
    - Drop unnecessary columns

|  |
| --- |
| # Display the first few rows of the dataset  print(df.head())  # Check for missing values  print(df.isnull().sum())  # Convert pickup\_datetime to datetime objects  df['pickup\_datetime'] = pd.to\_datetime(df['pickup\_datetime'])  # Impute missing values for numeric columns (example using mean imputation)  numeric\_columns = df.select\_dtypes(include=[np.number]).columns  imputer = SimpleImputer(strategy='mean')  df[numeric\_columns] = imputer.fit\_transform(df[numeric\_columns])  # Drop rows with missing target values  df.dropna(subset=['fare\_amount'], inplace=True)  # Extract useful features from datetime  df['pickup\_year'] = df['pickup\_datetime'].dt.year  df['pickup\_month'] = df['pickup\_datetime'].dt.month  df['pickup\_day'] = df['pickup\_datetime'].dt.day  df['pickup\_hour'] = df['pickup\_datetime'].dt.hour  # Drop unnecessary columns  df.drop(columns=['pickup\_datetime', 'key'], inplace=True) |

Step 3: Check the correlation, Plot the heatmap

|  |
| --- |
| # Calculate the correlation matrix  corr\_matrix = df.corr()  # Plot the heatmap  plt.figure(figsize=(10, 8))  sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm')  plt.show() |

Step 4: Implement linear regression and random forest regression models

Split the dataset into features and target variable

Split the data into training and testing set

Linear Regression

Random Forest Regression

|  |
| --- |
| # Split the dataset into features and target variable  X = df.drop(columns=['fare\_amount'])  y = df['fare\_amount']  # Split the data into training and testing sets  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # Linear Regression  lr\_model = LinearRegression()  lr\_model.fit(X\_train, y\_train)  y\_pred\_lr = lr\_model.predict(X\_test)  # Random Forest Regression  rf\_model = RandomForestRegressor(n\_estimators=100, random\_state=42)  rf\_model.fit(X\_train, y\_train)  y\_pred\_rf = rf\_model.predict(X\_test) |

Step 5: Evaluate the models and compare their respective scores like R², RMSE, etc.

|  |
| --- |
| # Evaluate Linear Regression  r2\_lr = r2\_score(y\_test, y\_pred\_lr)  rmse\_lr = np.sqrt(mean\_squared\_error(y\_test, y\_pred\_lr))  # Evaluate Random Forest Regression  r2\_rf = r2\_score(y\_test, y\_pred\_rf)  rmse\_rf = np.sqrt(mean\_squared\_error(y\_test, y\_pred\_rf))  # Print the evaluation metrics  print(f"Linear Regression - R²: {r2\_lr}, RMSE: {rmse\_lr}")  print(f"Random Forest Regression - R²: {r2\_rf}, RMSE: {rmse\_rf}") |