Assignment 1

Title of the Assignment:

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.

Dataset 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.

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

Objective of the Assignment:

Students should be able to preprocess dataset and identify outliers, to check correlation and implement linear regression and random forest regression models. Evaluate them with respective scores like R2, RMSE etc.

Prerequisite:

1. Basic knowledge of Python
2. Concept of preprocessing data
3. Basic knowledge of Data Science and Big Data Analytics.

Contents of the Theory:

1. Data Preprocessing
2. Linear regression
3. Random forest regression models
4. Box Plot
5. Outliers

6. Haversine
7. Mathplotlib
8. Mean Squared Error

Data Preprocessing:

Data preprocessing is a process of preparing the raw data and making it suitable for amachine learning model. It is the rst and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean itand put in a formatted way. So for this, we use data preprocessing task.

Why do we need Data Preprocessing?

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and e ciency of a machine learning model.

It involves below steps:

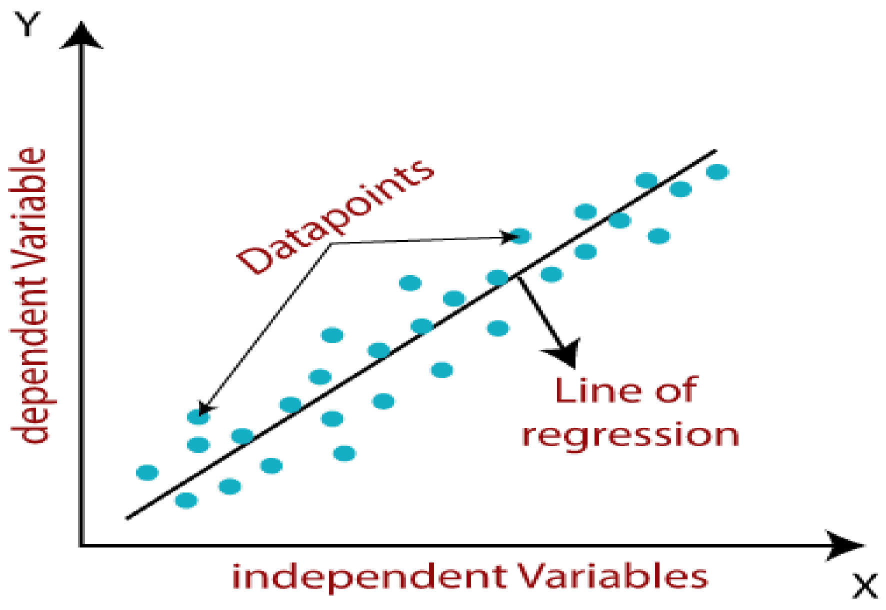
- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scaling

Linear Regression:

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such assales, salary, age, product price,etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it nds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables. Consider the below image:



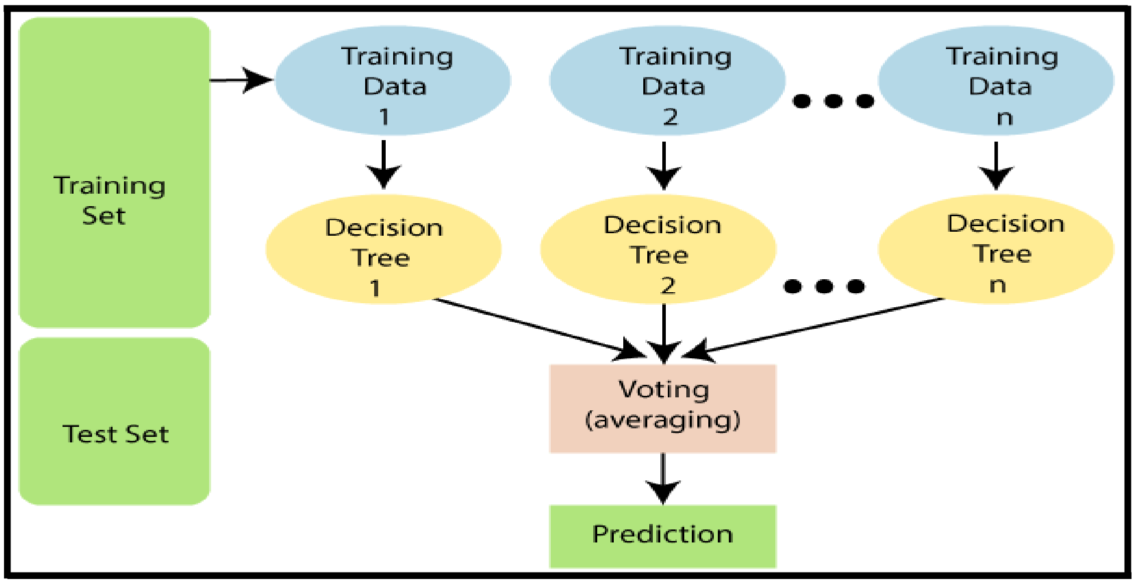
Random Forest Regression Models:

Random Forest is a popular machine learning algorithm that belongs to the supervisedlearning technique. It can be used for both Classi cation and Regression problems in ML. Itis based on the concept ofensemble learning,which is a process of*combining multiple classi ers to solve a complex problem and to improve the performance of the model.*

As the name suggests,*"Random Forest is a classi er that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."*Instead of relying on one decision tree, the random forest takes the

prediction from each tree and based on the majority votes of predictions, and it predicts the nal output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of over tting.



Boxplot:

Boxplots are a measure of how well data is distributed across a data set. This divides thedata set into three quartiles. This graph represents the minimum, maximum, average, rst quartile, and the third quartile in the data set. Boxplot is also useful in comparing the distribution of data in a data set by drawing a boxplot for each of them.

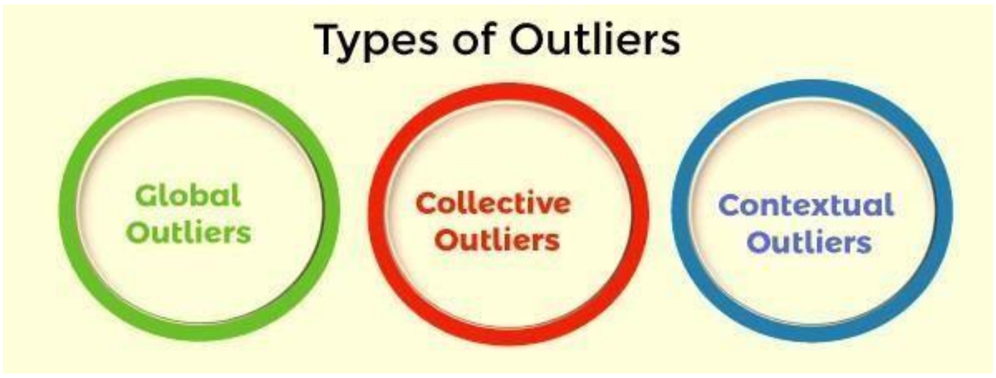
R provides a boxplot() function to create a boxplot. There is the following syntax of boxplot()function: boxplot(x, data, notch, varwidth, names, main)Here,

1.	X	It is a vector or a formula.
2.	Data	It is the data frame.
3.	Notch	It is a logical value set as true to draw a notch.
4.	varwidth	It is also a logical value set as true to draw the width of the box same as the sample size.
5.	Names	It is the group of labels that will be printed under each boxplot.
6.	Main	It is used to give a title to the graph.

Outliers:

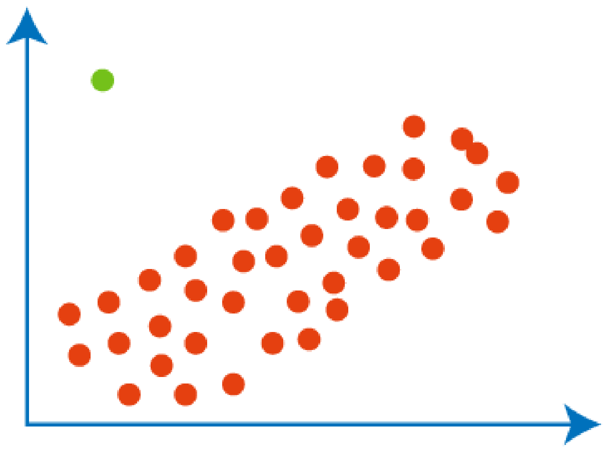
As the name suggests, "outliers" refer to the data points that exist outside of what is to be expected. The major thing about the outliers is what you do with them. If you are going to analyze any task to analyze data sets, you will always have some assumptions based on how this data is generated. If you nd some data points that are likely to contain some form of error, then these are de nitely outliers, and

depending on the context, you want to overcome those errors. The data mining process involves the analysis and prediction of data that the data holds. In 1969, Grubbs introduced the rst de nition of outliers.



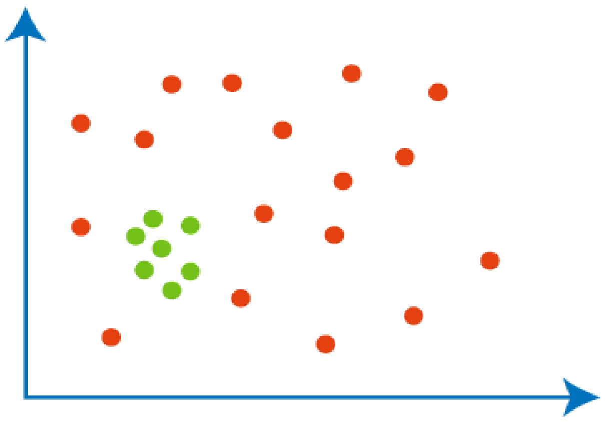
Global Outliers

Global outliers are also called point outliers. Global outliers are taken as the simplest form of outliers. When data points deviate from all the rest of the data points in a given data set, it is known as the global outlier. In most cases, all the outlier detection procedures are targeted to determine the global outliers. The green data point is the global outlier



Collective Outliers

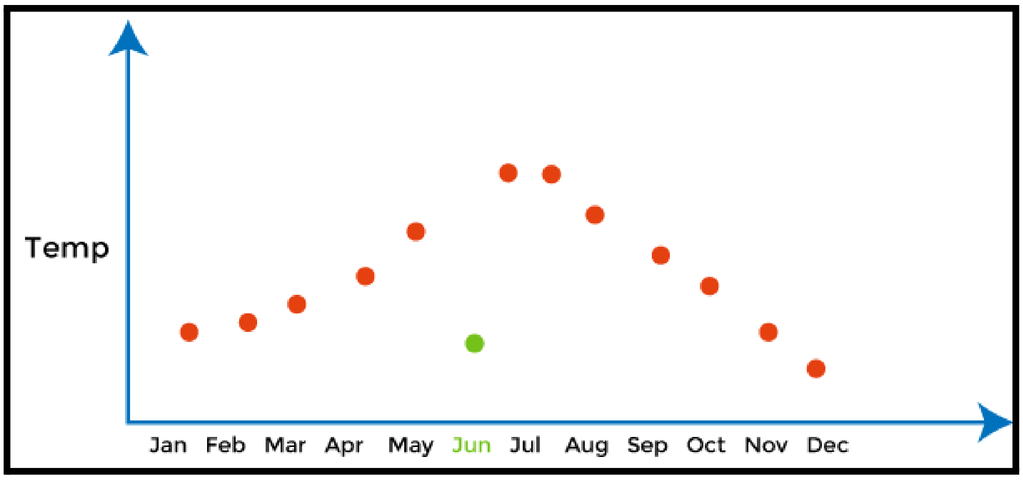
In a given set of data, when a group of data points deviates from the rest of the data set is called collective outliers. Here, the particular set of data objects may not be outliers, but when you consider the data objects as a whole, they may behave as outliers. To identify the types of different outliers, you need to go through background information about the relationship between the behavior of outliers shown by different data objects. For example, in an Intrusion Detection System, the DOS package from one system to another is taken as normal behavior. Therefore, if this happens with the various computer simultaneously, it is considered abnormal



behavior, and as a whole, they are called collective outliers. The green data points as a whole represent the collective outlier.

Contextual Outliers

As the name suggests, "Contextual" means this outlier introduced within a context. For example, in the speech recognition technique, the single background noise. Contextual outliers are also known as Conditional outliers. These types of outliers happen if a data object deviates from the other data points because of any speci c condition in a given data set. As we know, there are two types of attributes of objects of data: contextual attributes and behavioral attributes. Contextual outlier analysis enables the users to examine outliers in different contexts and conditions, which can be useful in various applications. For example, A temperature reading of 45 degrees Celsius may behave as an outlier in a rainy season. Still, it will behave like a normal data point in the context of a summer season. In thegiven diagram, a green dot representing the low-temperature value in June is a contextual outlier since the same value in December is not an outlier.



Haversine:

The Haversine formula calculates the shortest distance between two points on a sphere using their latitudes and longitudes measured along the surface. It is important for use in navigation.

Matplotlib:

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

Mean Squared Error;

The Mean Squared Error (MSE)or Mean Squared Deviation (MSD)of an estimator measures the average of error squares i.e. the average squared difference between the estimated values and true value. It is a risk function, corresponding to the expected value of the squared error loss. It is always non – negative and values close to zero are better. TheMSE is the second moment of the error (about the origin) and thus incorporates both the variance of the estimator and its bias.

Code :- https://www.kaggle.com/code/proxzima/uber-fare-price-prediction

Conclusion:

In this way we have explored Concept correlation and implement linear regression and random forest regression models.

In [8]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
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, mean_absolute_error
from haversine import haversine

df = pd.read_csv('uber.csv')
```

In [9]:
```
print("Original Dataset")
print(df.head())
```

```
Original Dataset
   Unnamed: 0                            key  fare_amount  \
0    24238194    2015-05-07 19:52:06.0000003          7.5
1    27835199    2009-07-17 20:04:56.0000002          7.7
2    44984355   2009-08-24 21:45:00.00000061         12.9
3    25894730    2009-06-26 08:22:21.0000001          5.3
4    17610152  2014-08-28 17:47:00.000000188         16.0

           pickup_datetime  pickup_longitude  pickup_latitude  \
0  2015-05-07 19:52:06 UTC        -73.999817        40.738354
1  2009-07-17 20:04:56 UTC        -73.994355        40.728225
2  2009-08-24 21:45:00 UTC        -74.005043        40.740770
3  2009-06-26 08:22:21 UTC        -73.976124        40.790844
4  2014-08-28 17:47:00 UTC        -73.925023        40.744085

   dropoff_longitude  dropoff_latitude  passenger_count
0         -73.999512         40.723217                1
1         -73.994710         40.750325                1
2         -73.962565         40.772647                1
3         -73.965316         40.803349                3
4         -73.973082         40.761247                5
```

In [10]:
```
df = df.drop(['Unnamed: 0', 'key'], axis=1)
print("Dataset after dropping the unnecessary columns")
print(df)
```

```
Dataset after dropping the unnecessary columns
        fare_amount          pickup_datetime  pickup_longitude  \
0               7.5  2015-05-07 19:52:06 UTC        -73.999817
1               7.7  2009-07-17 20:04:56 UTC        -73.994355
2              12.9  2009-08-24 21:45:00 UTC        -74.005043
3               5.3  2009-06-26 08:22:21 UTC        -73.976124
4              16.0  2014-08-28 17:47:00 UTC        -73.925023
...             ...                      ...               ...
199995          3.0  2012-10-28 10:49:00 UTC        -73.987042
199996          7.5  2014-03-14 01:09:00 UTC        -73.984722
199997         30.9  2009-06-29 00:42:00 UTC        -73.986017
199998         14.5  2015-05-20 14:56:25 UTC        -73.997124
199999         14.1  2010-05-15 04:08:00 UTC        -73.984395

        pickup_latitude  dropoff_longitude  dropoff_latitude  passenger_count
0             40.738354         -73.999512         40.723217                1
1             40.728225         -73.994710         40.750325                1
2             40.740770         -73.962565         40.772647                1
3             40.790844         -73.965316         40.803349                3
4             40.744085         -73.973082         40.761247                5
...                 ...                ...               ...              ...
199995        40.739367         -73.986525         40.740297                1
199996        40.736837         -74.006672         40.739620                1
199997        40.756487         -73.858957         40.692588                2
199998        40.725452         -73.983215         40.695415                1
199999        40.720077         -73.985508         40.768793                1

[200000 rows x 7 columns]
```

In [11]:
```
print(df.dtypes)
print(df.shape)
print(df.describe())
```

```
fare_amount          float64
pickup_datetime       object
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count        int64
dtype: object
(200000, 7)
         fare_amount  pickup_longitude  pickup_latitude  dropoff_longitude  \
count  200000.000000     200000.000000    200000.000000      199999.000000   
mean       11.359955        -72.527638        39.935885         -72.525292   
std         9.901776         11.437787         7.720539          13.117408   
min       -52.000000      -1340.648410       -74.015515       -3356.666300   
25%         6.000000        -73.992065        40.734796         -73.991407   
50%         8.500000        -73.981823        40.752592         -73.980093   
75%        12.500000        -73.967154        40.767158         -73.963658   
max       499.000000         57.418457      1644.421482        1153.572603   

       dropoff_latitude  passenger_count  
count     199999.000000    200000.000000  
mean          39.923890         1.684535  
std            6.794829         1.385997  
min         -881.985513         0.000000  
25%           40.733823         1.000000  
50%           40.753042         1.000000  
75%           40.768001         2.000000  
max          872.697628       208.000000  
```

In [12]:
```
print(df.isnull().sum())
```

```
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    1
dropoff_latitude     1
passenger_count      0
dtype: int64
```

In [13]:
```
df['dropoff_latitude'] = df['dropoff_latitude'].fillna(df['dropoff_latitude'].mean(
df['dropoff_longitude'] = df['dropoff_longitude'].fillna(df['dropoff_longitude'].me
```

In [14]:
```
print(df.isnull().sum())
print(df.dtypes)
```

```
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
dtype: int64
fare_amount          float64
pickup_datetime       object
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count        int64
dtype: object
```

In [15]:
```
df['pickup_datetime'] = pd.to_datetime(df['pickup_datetime'], errors='coerce')

df['hour'] = df['pickup_datetime'].dt.hour
df['day'] = df['pickup_datetime'].dt.day
df['month'] = df['pickup_datetime'].dt.month
df['year'] = df['pickup_datetime'].dt.year
df['dayofweek'] = df['pickup_datetime'].dt.dayofweek
```

In [16]:
```
df = df.drop('pickup_datetime', axis=1)
print(df)
```

```
        fare_amount  pickup_longitude  pickup_latitude  dropoff_longitude  \
0               7.5        -73.999817        40.738354         -73.999512   
1               7.7        -73.994355        40.728225         -73.994710   
2              12.9        -74.005043        40.740770         -73.962565   
3               5.3        -73.976124        40.790844         -73.965316   
4              16.0        -73.925023        40.744085         -73.973082   
...             ...               ...              ...                ...   
199995          3.0        -73.987042        40.739367         -73.986525   
199996          7.5        -73.984722        40.736837         -74.006672   
199997         30.9        -73.986017        40.756487         -73.858957   
199998         14.5        -73.997124        40.725452         -73.983215   
199999         14.1        -73.984395        40.720077         -73.985508   

        dropoff_latitude  passenger_count  hour  day  month  year  dayofweek  
0              40.723217                1    19    7      5  2015          3  
1              40.750325                1    20   17      7  2009          4  
2              40.772647                1    21   24      8  2009          0  
3              40.803349                3     8   26      6  2009          4  
4              40.761247                5    17   28      8  2014          3  
...                  ...              ...   ...  ...    ...   ...        ...  
199995         40.740297                1    10   28     10  2012          6  
199996         40.739620                1     1   14      3  2014          4  
199997         40.692588                2     0   29      6  2009          0  
199998         40.695415                1    14   20      5  2015          2  
199999         40.768793                1     4   15      5  2010          5  

[200000 rows x 11 columns]
```

In [ ]:

In [17]:
```
def remove_outlier(dff, col):
    Q1 = dff[col].quantile(0.25)
    Q3 = dff[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    dff[col] = np.clip(dff[col], lower, upper)
    return dff

def treat_outliers_all(dff, col_list):
    for col in col_list:
        dff = remove_outlier(dff, col)
    return dff

df = treat_outliers_all(df, ['fare_amount', 'pickup_latitude', 'pickup_longitude',
print(df)
```

```
        fare_amount  pickup_longitude  pickup_latitude  dropoff_longitude  \
0              7.50        -73.999817        40.738354         -73.999512
1              7.70        -73.994355        40.728225         -73.994710
2             12.90        -74.005043        40.740770         -73.962565
3              5.30        -73.976124        40.790844         -73.965316
4             16.00        -73.929786        40.744085         -73.973082
...             ...               ...              ...                ...
199995         3.00        -73.987042        40.739367         -73.986525
199996         7.50        -73.984722        40.736837         -74.006672
199997        22.25        -73.986017        40.756487         -73.922036
199998        14.50        -73.997124        40.725452         -73.983215
199999        14.10        -73.984395        40.720077         -73.985508

        dropoff_latitude  passenger_count  hour  day  month  year  dayofweek
0              40.723217                1    19    7      5  2015          3
1              40.750325                1    20   17      7  2009          4
2              40.772647                1    21   24      8  2009          0
3              40.803349                3     8   26      6  2009          4
4              40.761247                5    17   28      8  2014          3
...                  ...              ...   ...  ...    ...   ...        ...
199995         40.740297                1    10   28     10  2012          6
199996         40.739620                1     1   14      3  2014          4
199997         40.692588                2     0   29      6  2009          0
199998         40.695415                1    14   20      5  2015          2
199999         40.768793                1     4   15      5  2010          5

[200000 rows x 11 columns]
```

In [ ]:

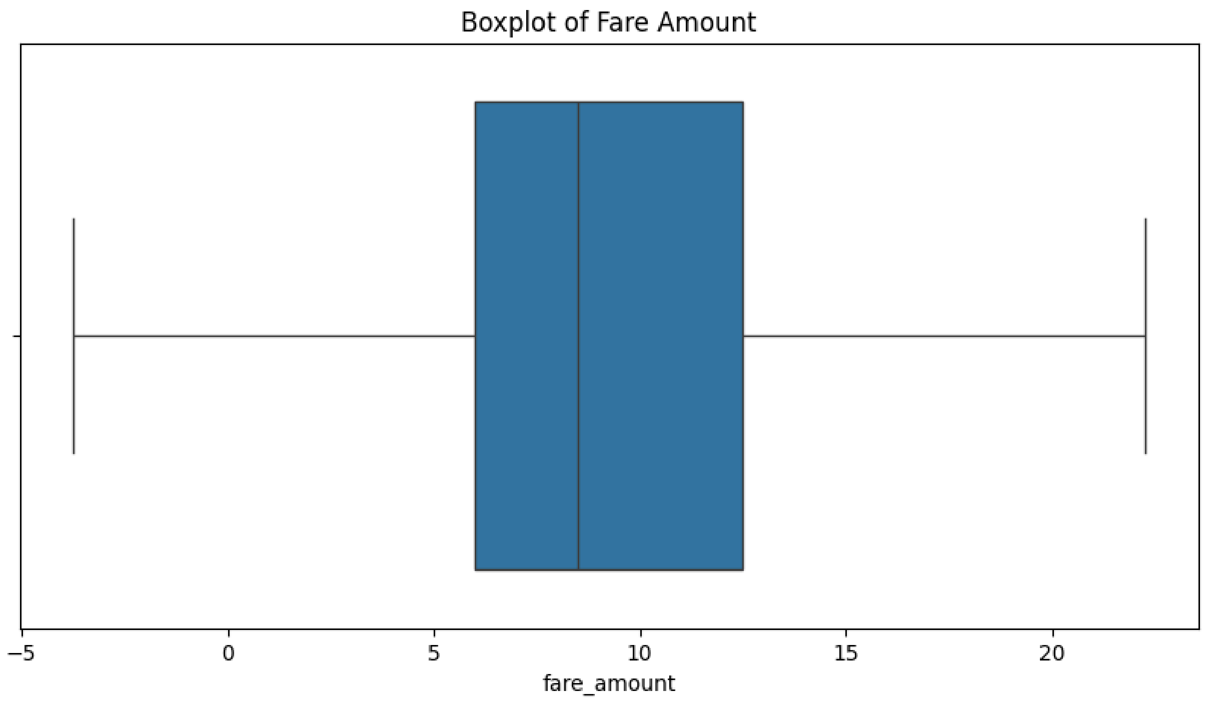
In [18]:
```
travel_dist = []
for pos in range(len(df)):
    long1 = df['pickup_longitude'].iloc[pos]
    lat1 = df['pickup_latitude'].iloc[pos]
    long2 = df['dropoff_longitude'].iloc[pos]
    lat2 = df['dropoff_latitude'].iloc[pos]
    loc1 = (lat1, long1)
    loc2 = (lat2, long2)
    d = haversine(loc1, loc2)
    travel_dist.append(d)

df['distance_km'] = travel_dist
print("Distance Calculated")
```

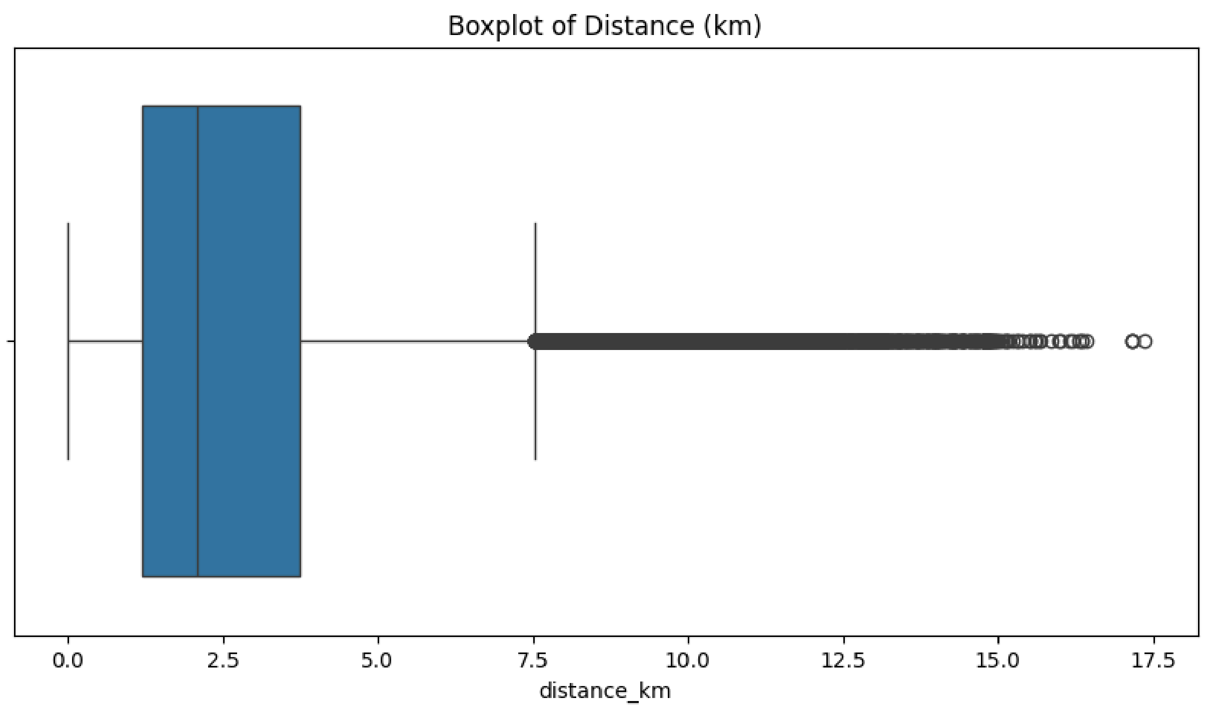
```
Distance Calculated
```

In [19]:
```
X = df.drop('fare_amount', axis=1)
y = df['fare_amount']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [20]:
```
plt.figure(figsize=(10, 5))
sns.boxplot(x=df['fare_amount'])
plt.title('Boxplot of Fare Amount')
plt.show()
```



In [21]:
```
plt.figure(figsize=(10, 5))
sns.boxplot(x=df['distance_km'])
plt.title('Boxplot of Distance (km)')
plt.show()
```



In [22]:
```
print("Correlation:")
print(df.corr())
```

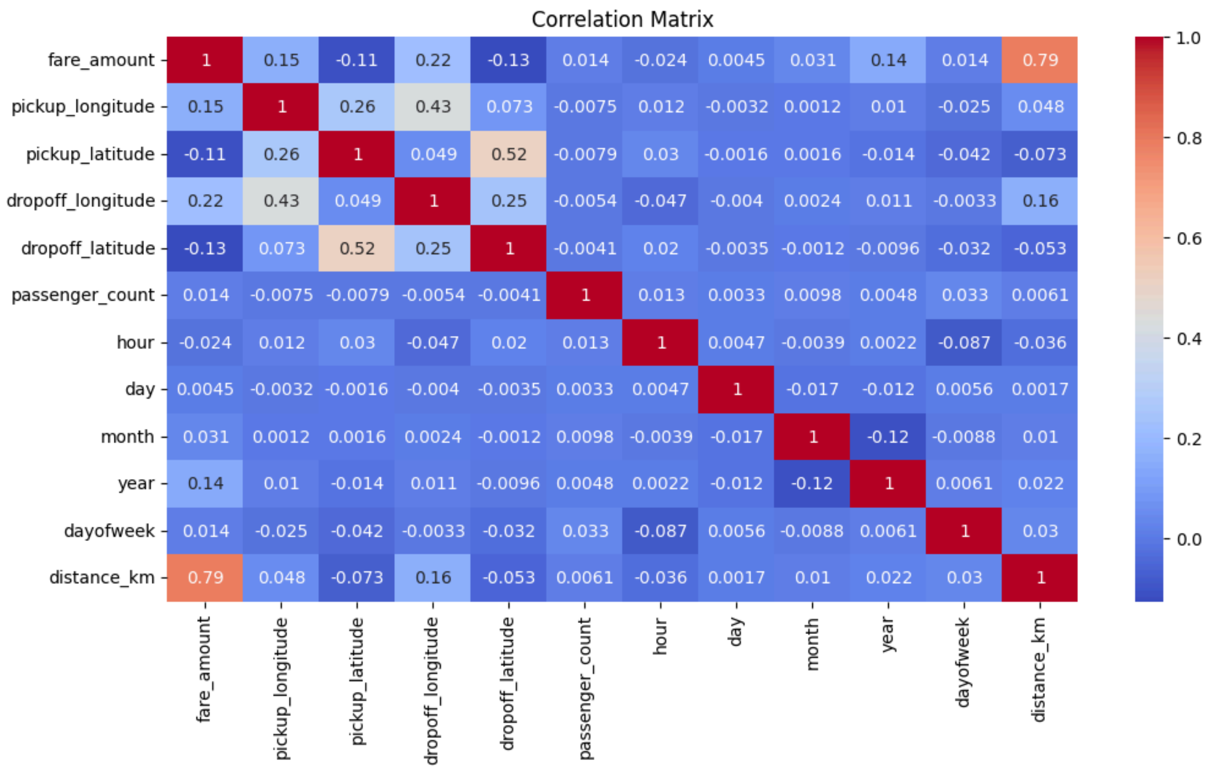
```
Correlation:
                   fare_amount  pickup_longitude  pickup_latitude  \
fare_amount           1.000000          0.154069        -0.110842   
pickup_longitude      0.154069          1.000000         0.259497   
pickup_latitude      -0.110842          0.259497         1.000000   
dropoff_longitude     0.218675          0.425619         0.048889   
dropoff_latitude     -0.125898          0.073290         0.515714   
passenger_count       0.013624         -0.007495        -0.007854   
hour                 -0.023623          0.011579         0.029681   
day                   0.004534         -0.003204        -0.001553   
month                 0.030817          0.001169         0.001562   
year                  0.141277          0.010198        -0.014243   
dayofweek             0.013652         -0.024652        -0.042310   
distance_km           0.786385          0.048446        -0.073362   

                   dropoff_longitude  dropoff_latitude  passenger_count  \
fare_amount                 0.218675         -0.125898         0.013624   
pickup_longitude            0.425619          0.073290        -0.007495   
pickup_latitude             0.048889          0.515714        -0.007854   
dropoff_longitude           1.000000          0.245667        -0.005377   
dropoff_latitude            0.245667          1.000000        -0.004087   
passenger_count            -0.005377         -0.004087         1.000000   
hour                       -0.046558          0.019783         0.013196   
day                        -0.004007         -0.003479         0.003252   
month                       0.002391         -0.001193         0.009773   
year                        0.011346         -0.009603         0.004798   
dayofweek                  -0.003336         -0.031919         0.033196   
distance_km                 0.155191         -0.052701         0.006117   

                       hour       day     month      year  dayofweek  \
fare_amount       -0.023623  0.004534  0.030817  0.141277   0.013652   
pickup_longitude   0.011579 -0.003204  0.001169  0.010198  -0.024652   
pickup_latitude    0.029681 -0.001553  0.001562 -0.014243  -0.042310   
dropoff_longitude -0.046558 -0.004007  0.002391  0.011346  -0.003336   
dropoff_latitude   0.019783 -0.003479 -0.001193 -0.009603  -0.031919   
passenger_count    0.013196  0.003252  0.009773  0.004798   0.033196   
hour               1.000000  0.004677 -0.003926  0.002156  -0.086947   
day                0.004677  1.000000 -0.017360 -0.012170   0.005617   
month             -0.003926 -0.017360  1.000000 -0.115859  -0.008786   
year               0.002156 -0.012170 -0.115859  1.000000   0.006113   
dayofweek         -0.086947  0.005617 -0.008786  0.006113   1.000000   
distance_km       -0.035708  0.001709  0.010050  0.022294   0.030382   

                   distance_km  
fare_amount           0.786385  
pickup_longitude      0.048446  
pickup_latitude      -0.073362  
dropoff_longitude     0.155191  
dropoff_latitude     -0.052701  
passenger_count       0.006117  
hour                 -0.035708  
day                   0.001709  
month                 0.010050  
year                  0.022294  
dayofweek             0.030382  
distance_km           1.000000  
```

```
In [23]: plt.figure(figsize=(12,6))
         sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
         plt.title("Correlation Matrix")
         plt.show()
```



```
In [ ]:
```

Assignment Questions:

1. What is Data Preprocessing?

- **Data Preprocessing** is the process of cleaning, transforming, and organizing raw data into a format that can be easily and effectively used by machine learning models. It is a crucial step in the ML pipeline.

Common steps include:

- **Handling missing values** (e.g., filling or removing)
- **Removing duplicates**
- **Encoding categorical variables** (Label Encoding, One-Hot Encoding)
- **Feature scaling** (Normalization, Standardization)
- **Outlier detection and removal**
- **Splitting data** into training and testing sets

2. Define Outliers

- **Outliers** are data points that significantly differ from other observations in the dataset. They may result from variability in the data, errors in data collection, or rare events.

Example:
In the dataset [2, 3, 4, 5, 100], the value **100** is an outlier.

Why outliers matter:

- Can skew model results
- Affect mean and standard deviation
- May indicate data quality issues or important rare events

3. What is Linear Regression?

- **Linear Regression** is a supervised machine learning algorithm used for predicting a **continuous** target variable based on one or more input features.

Formula (for one feature):

y=mx+cy = mx + cy=mx+c

Where:

- y is the predicted value
- m is the slope (coefficient)
- x is the input feature
- c is the intercept

Goal: Find the line (or hyperplane) that best fits the data by minimizing the error between predicted and actual values (using **least squares method**).

4. What is Random Forest Algorithm?

- **Random Forest** is an ensemble learning method that builds multiple **decision trees** and merges them to get a more accurate and stable prediction.

Key Features:

- Works for both classification and regression
- Reduces overfitting compared to single decision trees
- Uses **bagging** (Bootstrap Aggregating) technique

How it works:

- Creates multiple decision trees on random subsets of data and features

- Aggregates results (e.g., majority vote for classification or average for regression)

5. Explain: pandas, numpy in Machine Learning

✓ **Pandas:**

- A powerful **Python library** for data manipulation and analysis.
- Works with structured data like **tables (DataFrames)**.
- Provides tools to **read/write CSV, Excel, JSON**, filter rows/columns, group data, handle missing values, etc.

Example use in ML:

- Loading and exploring datasets
- Cleaning and preparing features before model training

✓ **Numpy:**

- Short for **Numerical Python**
- Supports **high-performance mathematical operations** on large arrays and matrices
- Provides functionalities for **linear algebra, statistics**, and random number generation

Example use in ML:

- Handling numerical data arrays
- Performing fast matrix operations for algorithms like linear regression, PCA, etc.