R CODE:

# Install necessary libraries

!pip install pandas matplotlib geopy scikit-learn

# Import necessary libraries

import pandas as pd

import matplotlib.pyplot as plt

from geopy.distance import geodesic

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

import numpy as np

# Mount the Google Drive (optional, if you have the file in Google Drive)

from google.colab import drive

drive.mount('/content/drive')

file\_path = '/content/drive/MyDrive/NYC Taxi Data.csv' # Update this path

df = pd.read\_csv(file\_path)

# Function to calculate distance between pickup and dropoff points (in kilometers)

def calculate\_distance(row):

pickup\_coords = (row['pickup\_latitude'], row['pickup\_longitude'])

dropoff\_coords = (row['dropoff\_latitude'], row['dropoff\_longitude'])

return geodesic(pickup\_coords, dropoff\_coords).kilometers

# Calculate the distance in kilometers for each trip

df['distance\_km'] = df.apply(calculate\_distance, axis=1) # Changed 'data' to 'df'

# Convert pickup\_datetime to datetime format and extract useful time features

df['pickup\_datetime'] = pd.to\_datetime(df['pickup\_datetime']) # Changed 'data' to 'df'

df['pickup\_hour'] = df['pickup\_datetime'].dt.hour # Changed 'data' to 'df'

df['pickup\_day'] = df['pickup\_datetime'].dt.day # Changed 'data' to 'df'

df['pickup\_month'] = df['pickup\_datetime'].dt.month # Changed 'data' to 'df'

df['pickup\_dayofweek'] = df['pickup\_datetime'].dt.dayofweek # Changed 'data' to 'df'

# Features to use for prediction

features = ['distance\_km', 'passenger\_count', 'pickup\_hour', 'pickup\_day', 'pickup\_month', 'pickup\_dayofweek']

# Define X (independent variables) and y (dependent variable)

X = df[features] # Changed 'data' to 'df'

y = df['trip\_duration'] # Changed 'data' to 'df'

# Split the data into training and testing sets (80% training, 20% testing)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create and train the Linear Regression model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test)

# Evaluate the model's performance

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f"Mean Squared Error: {mse}")

print(f"R-squared Score: {r2}")

# Scatter plot of actual vs predicted trip durations with a line of equality

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1) # 1 row, 2 columns, subplot 1

plt.scatter(y\_test, y\_pred, alpha=0.5, label='Predictions')

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], color='red', linestyle='--', label='Perfect Prediction')

plt.xlabel('Actual Trip Duration')

plt.ylabel('Predicted Trip Duration')

plt.title('Actual vs Predicted Trip Duration')

plt.legend()

plt.grid(True)

# Scatter plot of trip\_duration vs distance\_km with the regression line

plt.subplot(1, 2, 2) # 1 row, 2 columns, subplot 2

X\_distance = df[['distance\_km']] # Changed 'data' to 'df'

y\_duration = df['trip\_duration'] # Changed 'data' to 'df'

model\_distance = LinearRegression()

model\_distance.fit(X\_distance, y\_duration)

x\_range = np.linspace(X\_distance['distance\_km'].min(), X\_distance['distance\_km'].max(), 100).reshape(-1, 1)

y\_pred\_line = model\_distance.predict(x\_range)

plt.scatter(df['distance\_km'], df['trip\_duration'], alpha=0.5, label='Data') # Changed 'data' to 'df'

plt.plot(x\_range, y\_pred\_line, color='red', linewidth=2, label='Regression Line')

plt.xlabel('Distance (km)')

plt.ylabel('Trip Duration (seconds)')

plt.title('Trip Duration vs Distance')

plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()