import pandas as pd uber ride

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, mean\_absolute\_error from math import radians, sin, cos, sqrt, atan2

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

**----------------------------------------------------------------------------------------------------------------------------------------------------**

df = pd.read\_csv("uber.csv")

**----------------------------------------------------------------------------------------------------------------------------------------------------**

df.head()

**----------------------------------------------------------------------------------------------------------------------------------------------------**

print("Missing values per column:\n", df.isnull().sum())

**---------------------------------------------------------------------------------------------------------------------------------------------------**

df.dropna(inplace=True)

**----------------------------------------------------------------------------------------------------------------------------------------------------**

df.isnull().sum()

**----------------------------------------------------------------------------------------------------------------------------------------------------**

df['pickup\_datetime'] = pd.to\_datetime(df['pickup\_datetime'], errors='coerce') df.dropna(subset=['pickup\_datetime'], inplace=True)

**---------------------------------------------------------------------------------------------------------------------------------------------------**

df.info()

**---------------------------------------------------------------------------------------------------------------------------------------------------**

df.head()

**----------------------------------------------------------------------------------------------------------------------------------------------------**

def haversine\_distance(lat1, lon1, lat2, lon2):

R = 6371 # Earth radius in kilometers

lat1, lon1, lat2, lon2 = map(radians, [lat1, lon1, lat2, lon2]) dlat = lat2 - lat1

dlon = lon2 - lon1

a = sin(dlat / 2)\*\*2 + cos(lat1) \* cos(lat2) \* sin(dlon / 2)\*\*2 c = 2 \* atan2(sqrt(a), sqrt(1 - a))

return R \* c

**------------------------------------------------------------------------------------------------------------------------------------------------**

df['distance\_km'] = df.apply(lambda row: haversine\_distance(row['pickup\_latitude'], row['pickup\_longitude'],

row['dropoff\_latitude'], row['dropoff\_longitude']), axis=1)

**-----------------------------------------------------------------------------------------------------------------------------------------------**

df = df[['distance\_km', 'fare\_amount', 'passenger\_count']] df = df[(df['distance\_km'] > 0) & (df['fare\_amount'] > 0)]

**--------------------------------------------------------------------------------------------------------------------------------------------------**

df.head()

**--------------------------------------------------------------------------------------------------------------------------------------------------**

plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)

sns.boxplot(data=df, x='fare\_amount') plt.title("Fare Amount Outliers") plt.subplot(1, 2, 2)

sns.boxplot(data=df, x='distance\_km') plt.title("Distance Outliers") plt.show()

**---------------------------------------------------------------------------------------------------------------------------------------------------**

# Calculate Q1, Q3, and IQR for 'fare\_amount' and 'distance\_km' Q1 = df[['fare\_amount', 'distance\_km']].quantile(0.25)

Q3 = df[['fare\_amount', 'distance\_km']].quantile(0.75)

IQR = Q3 - Q1

# Remove rows with outliers based on the IQR method

df = df[~((df[['fare\_amount', 'distance\_km']] < (Q1 - 1.5 \* IQR)) | (df[['fare\_amount', 'distance\_km']] > (Q3 + 1.5 \* IQR))).any(axis=1)]

**-----------------------------------------------------------------------------------------------------------------------------------------------**

plt.figure(figsize=(8, 6))

sns.heatmap(df.corr(), annot=True, cmap='coolwarm') plt.title("Feature Correlation")

plt.show()

**------------------------------------------------------------------------------------------------------------------------------------------------**

X = df[['distance\_km', 'passenger\_count']] y = df['fare\_amount']

**------------------------------------------------------------------------------------------------------------------------------------------------**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

**---- --------------------------------------------------------------------------------------------------------------------------------------------**

lin\_reg = LinearRegression() lin\_reg.fit(X\_train, y\_train) y\_pred\_lr = lin\_reg.predict(X\_test)

**------------------------------------------------------------------------------------------------------------------------------------------------**

rf\_reg = RandomForestRegressor(n\_estimators=100, random\_state=42) rf\_reg.fit(X\_train, y\_train)

y\_pred\_rf = rf\_reg.predict(X\_test)

**-------------------------------------------------------------------------------------------------------------------------------------------------**

def print\_metrics(y\_true, y\_pred, model\_name):

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

rmse = root\_mean\_squared\_error(y\_true, y\_pred) # Use new function for RMSE mae = mean\_absolute\_error(y\_true, y\_pred)

print(f"{model\_name} Performance:") print(f"R-squared: {r2:.2f}")

print(f"RMSE: {rmse:.2f}")

print(f"MAE: {mae:.2f}\n")

**--------------------------------------------------------------------------------------------------------------------------------------**

print\_metrics(y\_test, y\_pred\_lr, "Linear Regression")

**----------------------------------------------------------------------------------------------------------------------------------------**

print\_metrics(y\_test, y\_pred\_rf, "Random Forest Regression")