In [ ]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from geopy.distance import geodesic
```

In [2]:
```
# --- 1. Load the dataset ---
# Dataset link: https://www.kaggle.com/datasets/yasserh/uber-fares-dataset
data = pd.read_csv("uber.csv")

print("Original Dataset Shape:", data.shape)
print(data.head())
```

```
Original Dataset Shape: (200000, 9)
   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 [3]:
```
# --- 2. Basic cleaning ---
data = data.dropna(subset=["pickup_longitude", "pickup_latitude",
                           "dropoff_longitude", "dropoff_latitude", "fare_amount
```

In [4]:
```
# Remove invalid or unrealistic values
data = data[(data["fare_amount"] > 0) & (data["fare_amount"] < 500)]
data = data[(data["pickup_longitude"].between(-180, 180)) &
            (data["dropoff_longitude"].between(-180, 180)) &
            (data["pickup_latitude"].between(-90, 90)) &
            (data["dropoff_latitude"].between(-90, 90))]
```

In [5]:
```
# --- 3. Feature Engineering ---
# Calculate distance between pickup and dropoff points using geodesic (km)
def calculate_distance(row):
    pickup = (row["pickup_latitude"], row["pickup_longitude"])
    dropoff = (row["dropoff_latitude"], row["dropoff_longitude"])
    return geodesic(pickup, dropoff).km

data["distance_km"] = data.apply(calculate_distance, axis=1)
data = data[data["distance_km"] < 100]  # remove extreme long trips
```

In [6]:
```
# --- 4. Prepare data ---
X = data[["distance_km"]]
y = data["fare_amount"]

# Split into train-test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print("\nData Ready for Training:")
print("Training Samples:", len(X_train))
print("Testing Samples:", len(X_test))
```

```
Data Ready for Training:
Training Samples: 159605
Testing Samples: 39902
```

In [7]:
```
# --- 5. Train Linear Regression ---
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_pred = lr_model.predict(X_test)
```

In [8]:
```
# --- 6. Train Random Forest Regressor ---
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)
```

In [9]:
```
# --- 7. Evaluate Models ---
print("\n--- Model Evaluation ---")

lr_r2 = r2_score(y_test, lr_pred)
lr_rmse = np.sqrt(mean_squared_error(y_test, lr_pred))
print(f"Linear Regression -> R2: {lr_r2:.4f}, RMSE: {lr_rmse:.4f}")

rf_r2 = r2_score(y_test, rf_pred)
rf_rmse = np.sqrt(mean_squared_error(y_test, rf_pred))
print(f"Random Forest     -> R2: {rf_r2:.4f}, RMSE: {rf_rmse:.4f}")
```

```
--- Model Evaluation ---
Linear Regression -> R2: 0.6078, RMSE: 6.3253
Random Forest     -> R2: 0.5969, RMSE: 6.4124
```

In [10]:
```
# --- 8. Compare Models ---
if rf_r2 > lr_r2:
    print("\nConclusion: ✅ Random Forest performs better.")
else:
    print("\nConclusion: ✅ Linear Regression performs better.")
```

```
Conclusion: ✅ Linear Regression performs better.
```

In [11]:
```
# --- 9. Show sample predictions ---
results = pd.DataFrame({
    "Actual": y_test.values[:10],
    "LR_Predicted": lr_pred[:10],
    "RF_Predicted": rf_pred[:10]
})
print("\nSample Predictions:")
print(results)
```

```
Sample Predictions:
   Actual  LR_Predicted  RF_Predicted
0    11.3      4.331917     11.535748
1     9.3      8.224941      7.377000
2    45.0     50.388414     56.449600
3    17.5     14.334062     12.444000
4     5.0      6.568458      4.129000
5    12.1     15.512883     15.350000
6     5.7      6.008335      4.219000
7     5.7      6.753152      5.447000
8     7.3      4.331917     11.535748
9    10.9      8.451748      8.583000
```

In [ ]: