In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
data=pd.read_excel(r"D:\Data\uber_rides_data.xlsx")
```

In [3]:
```
data.head(5)
```

Out[3]:

	ride_id	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dr
0	24238194	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512	
1	27835199	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710	
2	44984355	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962565	
3	25894730	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965316	
4	17610152	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973082	

In [5]:
```
#What is the shape of given dataset?

data.shape
```

Out[5]:
```
(200000, 8)
```

In [6]:
```
#How many integer columns(by default) are given in the dataset?

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 8 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   ride_id            200000 non-null  int64  
 1   fare_amount        200000 non-null  float64
 2   pickup_datetime    200000 non-null  object 
 3   pickup_longitude   200000 non-null  float64
 4   pickup_latitude    200000 non-null  float64
 5   dropoff_longitude  199999 non-null  float64
 6   dropoff_latitude   199999 non-null  float64
 7   passenger_count    200000 non-null  int64  
dtypes: float64(5), int64(2), object(1)
memory usage: 12.2+ MB
```

Only 2 columns namely 'ride_id' & 'passenger_count' were of 'int' datatype by default.

In [9]:
```
#How many missing values exists in 'dropoff_longitude' column?

data.isna().sum()
```

Out[9]:
```
ride_id              0
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    1
```

```
dropoff_latitude     1
passenger_count      0
dtype: int64
```

Only one missing value in dropoff_longitude column

What is the data type of ' pickup_datetime' feature in your data?

From the above results of data.info() function, it is clear that 'pickup_datetime' is of object datatype.

In [11]:
```
#Change pickup_datetime column from object datatype to datetime.

data['pickup_datetime'] = pd.to_datetime(data['pickup_datetime'])
```

In [12]:
```
# Remove the null values from the dataframe

df=data.dropna()
```

In [18]:
```
# What is the average fare amount?

avg_fare_amount = df['fare_amount'].mean()
print("The average fare amount during the Uber trips was: Rs.",avg_fare_amount)
```

```
The average fare amount during the Uber trips was: Rs. 11.359891549458371
```

In [ ]:
```
#How many rides have 0.0 haversine distance between pickup and dropoff location acco
```

In [28]:
```
#Calculate distance between each pickup and dropoff points using Haversine formula.

def haversine_distance(lat1, long1, lat2, long2):
    # the unit is in km
    #r = Average of earth radius
    lat1, long1, lat2, long2 = map(np.radians, (lat1, long1, lat2, long2))
    r = 6371
    lat = lat2 - lat1
    long = long2 - long1
    d = np.sin(lat * 0.5) ** 2 + np.cos(lat1) * np.cos(lat2) * np.sin(long * 0.5) **
    h = 2 * r * np.arcsin(np.sqrt(d))
    return h
```

In [29]:
```
def get_distance(df):
    df.loc[:, 'distance_haversine'] = haversine_distance(df['pickup_latitude'].value
                                                         df['pickup_longitude'].values,
                                                         df['dropoff_latitude'].values,
                                                         df['dropoff_longitude'].values


    return df
```

In [33]:
```
df1= get_distance(df)
```

```
C:\Users\affine\anaconda3\lib\site-packages\pandas\core\indexing.py:1745: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  isetter(ilocs[0], value)
```

In [40]:
```
#df1=df1.drop(['Distance'], axis=1)
```

In [41]:
```
df1.head(5)
```

Out[41]:

	ride_id	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dr
0	24238194	7.5	2015-05-07 19:52:06+00:00	-73.999817	40.738354	-73.999512	
1	27835199	7.7	2009-07-17 20:04:56+00:00	-73.994355	40.728225	-73.994710	
2	44984355	12.9	2009-08-24 21:45:00+00:00	-74.005043	40.740770	-73.962565	
3	25894730	5.3	2009-06-26 08:22:21+00:00	-73.976124	40.790844	-73.965316	
4	17610152	16.0	2014-08-28 17:47:00+00:00	-73.925023	40.744085	-73.973082	

In [35]:
```
# What is the median haversine distance between pickup and dropoff location accordin

median_distance_ = df1['distance_haversine'].median()
print("The median haversine distance between pickup and dropoff location is: ",media
```

```
The median haversine distance between pickup and dropoff location is:  2.12099239618
2902
```

In [43]:
```
# What is the maximum haversine distance between pickup and dropoff location accordi

max_haversine_distance_ = max(df1['distance_haversine'])
print("The maximum haversine distance between pickup and dropoff location is: ",max_
```

```
The maximum haversine distance between pickup and dropoff location is:  16409.239135
313168
```

In [45]:
```
# How many rides have 0.0 haversine distance between pickup and dropoff location acc

print(df1['distance_haversine'].value_counts()[0.0])
```

```
5632
```

In [46]:
```
# What is the mean 'fare_amount' for rides with 0 haversine distance?

df.loc[df1['distance_haversine'] == 0.0, 'fare_amount'].mean()
```

Out[46]:
```
11.585317826704578
```

Although idle, i.e the haversine distance is 0.0, the minimum fare is charged.

In [47]:
```
# What is the maximum fare amount for a ride?

max_fare_amount = max(df1['fare_amount'])
print("The maximum haversine distance between pickup and dropoff location is: ",max_
```

```
The maximum haversine distance between pickup and dropoff location is:  499.0
```

In [67]:
```
# What is the haversine distance between pickup and dropoff location for the costlie

#  row with the highest 'fare_amount'
costliest_ride = df1[df1['fare_amount'] == df1['fare_amount'].max()]

# Calculate the Haversine distance for the costliest ride
haversine_distance_costliest_ride = haversine(
```

```
    costliest_ride['pickup_latitude'].values[0],
    costliest_ride['pickup_longitude'].values[0],
    costliest_ride['dropoff_latitude'].values[0],
    costliest_ride['dropoff_longitude'].values[0]
)

print("The Haversine distance for the costliest ride is:", haversine_distance_costli
```

```
The Haversine distance for the costliest ride is: 0.0 kilometers
```

In [68]:
```
# Convert 'pickup_datetime' to a datetime object
df1['pickup_datetime'] = pd.to_datetime(df1['pickup_datetime'])

# Extract the year from 'pickup_datetime'
df1['year'] = df1['pickup_datetime'].dt.year

# Count the number of rides in the year 2014
rides_in_2014 = (df1['year'] == 2014).sum()

print("The number of rides recorded in the year 2014 is:", rides_in_2014)
```

```
The number of rides recorded in the year 2014 is: 29968
```

In [60]:
```
#Algorithm implementation

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

In [64]:
```
# Extract the day of the week (0 = Monday, 6 = Sunday)
df1['day_of_week'] = df1['pickup_datetime'].dt.dayofweek

# Filter data for September 2010
september_2010_data = df1[(df1['pickup_datetime'].dt.year == 2010) & (df1['pickup_da

# Group by day of the week and count rides
rides_by_day_of_week = september_2010_data.groupby('day_of_week').size()

#  day with the maximum number of rides
max_rides_day = rides_by_day_of_week.idxmax()  # This gives the index (0-6) of the d

# Convert the index to the corresponding day name
day_names = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Su
max_rides_day_name = day_names[max_rides_day]

print("The day of the week in September 2010 with the maximum rides recorded was:",
```

```
The day of the week in September 2010 with the maximum rides recorded was: Thursday
```

In [72]:
```
# Distribution of data
X = df1[['passenger_count', 'distance_haversine', 'day_of_week']]
y = df1['fare_amount']
```

In [74]:
```
# Split the data

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_stat
```

In [75]:
```
#Train different models


models = {
    'Linear Regression': LinearRegression(),
```

```
    'Decision Tree': DecisionTreeRegressor(),
    'Random Forest': RandomForestRegressor(),
    'Gradient Boosting': GradientBoostingRegressor()
}

results = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    adjusted_r2 = 1 - (1 - r2) * ((len(y_test) - 1) / (len(y_test) - len(X_test.colu
    results[name] = adjusted_r2

# Determine the algorithm with the least adjusted R-squared
min_adjusted_r2_algorithm = min(results, key=results.get)
print("Algorithm with the least adjusted R-squared:", min_adjusted_r2_algorithm)
```

```
Algorithm with the least adjusted R-squared: Linear Regression
```

In [76]:

```
print(results)
```

```
{'Linear Regression': 0.0003841322880681064, 'Decision Tree': 0.4718909595226325, 'R
andom Forest': 0.6300641618246313, 'Gradient Boosting': 0.6752474100357717}
```