PRACTICAL NO.:01

Name: Imam Mahamad Shaikh

Batch:C2

Roll no.:B23025

Problem Statement:

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.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

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

```
        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
...            ...                             ...          ...
199995    42598914   2012-10-28 10:49:00.00000053          3.0
199996    16382965    2014-03-14 01:09:00.0000008          7.5
199997    27804658   2009-06-29 00:42:00.00000078         30.9
199998    20259894    2015-05-20 14:56:25.0000004         14.5
199999    11951496   2010-05-15 04:08:00.00000076         14.1
```

```
                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
...                         ...               ...              ...
199995  2012-10-28 10:49:00 UTC        -73.987042        40.739367
199996  2014-03-14 01:09:00 UTC        -73.984722        40.736837
199997  2009-06-29 00:42:00 UTC        -73.986017        40.756487
199998  2015-05-20 14:56:25 UTC        -73.997124        40.725452
199999  2010-05-15 04:08:00 UTC        -73.984395        40.720077

        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
...                   ...               ...              ...
199995         -73.986525         40.740297                1
199996         -74.006672         40.739620                1
199997         -73.858957         40.692588                2
199998         -73.983215         40.695415                1
199999         -73.985508         40.768793                1

[200000 rows x 9 columns]

df.head()

   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
```

```
df.tail()
```

```
        Unnamed: 0                            key  fare_amount  \
199995    42598914  2012-10-28 10:49:00.00000053          3.0
199996    16382965   2014-03-14 01:09:00.0000008          7.5
199997    27804658  2009-06-29 00:42:00.00000078         30.9
199998    20259894   2015-05-20 14:56:25.0000004         14.5
199999    11951496  2010-05-15 04:08:00.00000076         14.1

                pickup_datetime  pickup_longitude  pickup_latitude  \
199995  2012-10-28 10:49:00 UTC        -73.987042        40.739367
199996  2014-03-14 01:09:00 UTC        -73.984722        40.736837
199997  2009-06-29 00:42:00 UTC        -73.986017        40.756487
199998  2015-05-20 14:56:25 UTC        -73.997124        40.725452
199999  2010-05-15 04:08:00 UTC        -73.984395        40.720077

        dropoff_longitude  dropoff_latitude  passenger_count
199995         -73.986525         40.740297                1
199996         -74.006672         40.739620                1
199997         -73.858957         40.692588                2
199998         -73.983215         40.695415                1
199999         -73.985508         40.768793                1
```

```
df.dtypes
```

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

```
df.describe()
```

```
         Unnamed: 0    fare_amount  pickup_longitude  pickup_latitude
\
count  2.000000e+05  200000.000000     200000.000000    200000.000000

mean   2.771250e+07      11.359955        -72.527638        39.935885

std    1.601382e+07       9.901776         11.437787         7.720539

min    1.000000e+00     -52.000000      -1340.648410       -74.015515

25%    1.382535e+07       6.000000        -73.992065        40.734796

50%    2.774550e+07       8.500000        -73.981823        40.752592
```

```
75%    4.155530e+07      12.500000        -73.967154        40.767158

max    5.542357e+07     499.000000         57.418457      1644.421482


       dropoff_longitude  dropoff_latitude  passenger_count
count      199999.000000     199999.000000    200000.000000
mean          -72.525292         39.923890         1.684535
std            13.117408          6.794829         1.385997
min         -3356.666300       -881.985513         0.000000
25%           -73.991407         40.733823         1.000000
50%           -73.980093         40.753042         1.000000
75%           -73.963658         40.768001         2.000000
max          1153.572603        872.697628       208.000000
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   Unnamed: 0         200000 non-null  int64
 1   key                200000 non-null  object
 2   fare_amount        200000 non-null  float64
 3   pickup_datetime    200000 non-null  object
 4   pickup_longitude   200000 non-null  float64
 5   pickup_latitude    200000 non-null  float64
 6   dropoff_longitude  199999 non-null  float64
 7   dropoff_latitude   199999 non-null  float64
 8   passenger_count    200000 non-null  int64
dtypes: float64(5), int64(2), object(2)
memory usage: 13.7+ MB
```

```
df.isnull()
```

```
        Unnamed: 0    key  fare_amount  pickup_datetime
pickup_longitude  \
0            False  False        False            False
False
1            False  False        False            False
False
2            False  False        False            False
False
3            False  False        False            False
False
4            False  False        False            False
False
...            ...    ...          ...              ...
...
```

```
199995       False  False        False             False
False
199996       False  False        False             False
False
199997       False  False        False             False
False
199998       False  False        False             False
False
199999       False  False        False             False
False

        pickup_latitude  dropoff_longitude  dropoff_latitude
passenger_count
0                 False              False             False
False
1                 False              False             False
False
2                 False              False             False
False
3                 False              False             False
False
4                 False              False             False
False
...                 ...                ...               ...
...
199995            False              False             False
False
199996            False              False             False
False
199997            False              False             False
False
199998            False              False             False
False
199999            False              False             False
False

[200000 rows x 9 columns]
```

```
df.isnull().sum()
```

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

```
df.shape
```

```
(200000, 9)
```

```
df.dropna(inplace=True)
```

```
df.isnull().sum()
```

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

```
df["pickup_datetime"] = pd.to_datetime(df["pickup_datetime"],
errors="coerce")
```

```
df["hour"] = df["pickup_datetime"].dt.hour
df["weekday"] = df["pickup_datetime"].dt.weekday
df["month"] = df["pickup_datetime"].dt.month
```

```
# distances in KM
def haversine(lon1, lat1, lon2, lat2):
    lon1, lat1, lon2, lat2 = map(np.radians, [lon1, lat1, lon2, lat2])
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = np.sin(dlat/2)**2 + np.cos(lat1) * np.cos(lat2) *
np.sin(dlon/2)**2
    return 6371 * (2 * np.arcsin(np.sqrt(a)))  # km
```

```
df["distance"] = haversine(df["pickup_longitude"],
df["pickup_latitude"],
                           df["dropoff_longitude"],
df["dropoff_latitude"])
```

```
df
```

```
        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
...            ...                             ...          ...
199995    42598914   2012-10-28 10:49:00.00000053          3.0
199996    16382965    2014-03-14 01:09:00.0000008          7.5
```

```
199997    27804658   2009-06-29 00:42:00.00000078          30.9
199998    20259894    2015-05-20 14:56:25.0000004          14.5
199999    11951496   2010-05-15 04:08:00.00000076          14.1

                  pickup_datetime  pickup_longitude  pickup_latitude  \
0      2015-05-07 19:52:06+00:00        -73.999817         40.738354
1      2009-07-17 20:04:56+00:00        -73.994355         40.728225
2      2009-08-24 21:45:00+00:00        -74.005043         40.740770
3      2009-06-26 08:22:21+00:00        -73.976124         40.790844
4      2014-08-28 17:47:00+00:00        -73.925023         40.744085
...                          ...               ...               ...
199995 2012-10-28 10:49:00+00:00        -73.987042         40.739367
199996 2014-03-14 01:09:00+00:00        -73.984722         40.736837
199997 2009-06-29 00:42:00+00:00        -73.986017         40.756487
199998 2015-05-20 14:56:25+00:00        -73.997124         40.725452
199999 2010-05-15 04:08:00+00:00        -73.984395         40.720077

        dropoff_longitude  dropoff_latitude  passenger_count  hour
weekday  \
0              -73.999512         40.723217                1    19
3
1              -73.994710         40.750325                1    20
4
2              -73.962565         40.772647                1    21
0
3              -73.965316         40.803349                3     8
4
4              -73.973082         40.761247                5    17
3
...                   ...               ...              ...   ...
...
199995         -73.986525         40.740297                1    10
6
199996         -74.006672         40.739620                1     1
4
199997         -73.858957         40.692588                2     0
0
199998         -73.983215         40.695415                1    14
2
199999         -73.985508         40.768793                1     4
5

        month   distance
0           5   1.683323
1           7   2.457590
2           8   5.036377
3           6   1.661683
4           8   4.475450
...       ...        ...
199995     10   0.112210
```

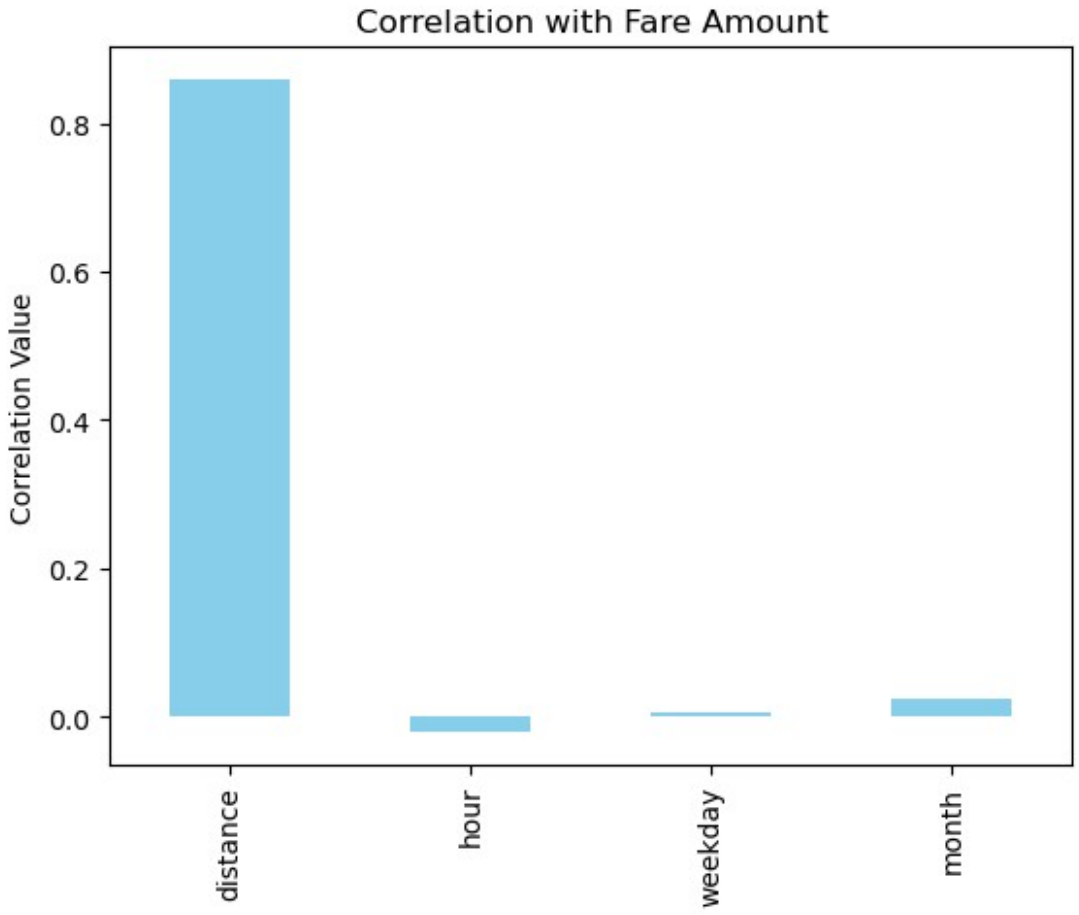
```
199996      3   1.875050
199997      6  12.850319
199998      5   3.539715
199999      5   5.417783

[199999 rows x 13 columns]
```

```python
# Outlier handling
df = df[(df["fare_amount"] > 0) & (df["fare_amount"] < 100)]
df = df[(df["distance"] > 0) & (df["distance"] < 100)]

corr = df[["fare_amount","distance","hour","weekday","month"]].corr()
["fare_amount"]

corr.drop("fare_amount").plot(kind="bar", color="skyblue")
plt.title("Correlation with Fare Amount")
plt.ylabel("Correlation Value")
plt.show()
```



```
# 5. Features & Target
X = df[["distance","hour","weekday","month","passenger_count"]]
y = df["fare_amount"]

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor

#LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
pred_lr = lr.predict(X_test)

pred_lr

array([ 5.80204908,  8.40852548,  8.54483812, ...,  9.47122511,
       26.02043998,  5.69969744])

rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
pred_rf = rf.predict(X_test)

pred_rf

array([ 6.043,  9.023,  9.299, ...,  9.286, 26.837,  6.523])

from sklearn.metrics import r2_score, mean_squared_error,
mean_absolute_error

results = []
def evaluate(y_test, y_pred, name):
    r2 = r2_score(y_test, y_pred)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    mae = mean_absolute_error(y_test, y_pred)
    results.append([name, r2, rmse, mae])
    print(f"{name} -> R2: {r2:.3f}, RMSE: {rmse:.2f}, MAE: {mae:.2f}")

evaluate(y_test, pred_lr, "Linear Regression")

Linear Regression -> R2: 0.712, RMSE: 5.05, MAE: 2.38

evaluate(y_test, pred_rf, "Random Forest")

Random Forest -> R2: 0.787, RMSE: 4.34, MAE: 2.39

results = np.array(results, dtype=object)

# Plot bar chart for comparison
metrics = ["R2", "RMSE", "MAE"]
```

```
x = np.arange(len(metrics))  # [0,1,2]
width = 0.35

plt.figure(figsize=(8,5))
plt.bar(x - width/2, results[0,1:].astype(float), width,
label=results[0,0])
plt.bar(x + width/2, results[1,1:].astype(float), width,
label=results[1,0])

plt.xticks(x, metrics)
plt.ylabel("Score / Error")
plt.title("Model Performance Comparison")
plt.legend()
plt.show()
```