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 5. Evaluate the models and compare their respective scores like R2, RMSE, etc.

```
#Importing the required libraries

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

#import dependencies

import pandas as pd

import numpy as np

#import plotly.express as px

#load the data into a dataframe

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

#df1 = pd.read_csv('uber.csv')

#check the first 5 rows

df.head()

#print Dataset

print("Original Dataset")

print(df)

#Data Preprocessing

#drop the unnecessary columns

#df = df.drop(columns=(['Unnamed: 0', 'key','pickup_datetime','pickup_longitude', 'pickup_latitude',
'dropoff_longitude', 'dropoff_latitude']))

df = df.drop(['Unnamed: 0', 'key'], axis= 1) #To drop unnamed column as it isn't required

print("Dataset after dropping the unnecessary columns")

print(df)

print(df.dtypes) #To get the type of each column

print(df.shape) #To get the total (Rows,Columns)

print(df.describe()) #To get statistics of each columns

# Filling Missing Values

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

```
df['dropoff_latitude'].fillna(value=df['dropoff_latitude'].mean(),inplace = True)
df['dropoff_longitude'].fillna(value=df['dropoff_longitude'].median(),inplace = True)
df.isnull().sum()
print(df.dtypes)
#Column pickup_datetime is in wrong format (Object). Convert it to DateTime Format
df.pickup_datetime = pd.to_datetime(df.pickup_datetime, errors='coerce')
print(df.dtypes)
df= df.assign(hour = df.pickup_datetime.dt.hour,
        day= df.pickup_datetime.dt.day,
        month = df.pickup_datetime.dt.month,
        year = df.pickup_datetime.dt.year,
        dayofweek = df.pickup_datetime.dt.dayofweek)
print(df)
# drop the column 'pickup_daetime' using drop()
# 'axis = 1' drops the specified column
df = df.drop('pickup_datetime',axis=1)
print(df)
#Checking outliers and filling them
#plt.plot(kind = "box",subplots = True,layout = (7,2),figsize=(15,20)) #Boxplot to check the outliers
#Using the InterQuartile Range to fill the values
def remove_outlier(df1 , col):
  Q1 = df1[col].quantile(0.25)
  Q3 = df1[col].quantile(0.75)
  IQR = Q3 - Q1
  lower_whisker = Q1-1.5*IQR
  upper_whisker = Q3+1.5*IQR
  df[col] = np.clip(df1[col] , lower_whisker , upper_whisker)
  return df1
def treat_outliers_all(df1 , col_list):
  for c in col_list:
    df1 = remove_outlier(df , c)
  return df1
df = treat_outliers_all(df , df.iloc[: , 0::])
print("Outliers")
```

```
print(df)

#pip install haversine

import haversine as hs  #Calculate the distance using Haversine to calculate the distance between to points.
Can't use Eucladian as it is for flat surface.

travel_dist = []

for pos in range(len(df['pickup_longitude'])):

        long1,lati1,long2,lati2 =
[df['pickup_longitude'][pos],df['pickup_latitude'][pos],df['dropoff_longitude'][pos],df['dropoff_latitude'][pos]]

        loc1=(lati1,long1)

        loc2=(lati2,long2)

        c = hs.haversine(loc1,loc2)

        travel_dist.append(c)

#print(travel_dist)

df['dist_travel_km'] = travel_dist

print("Distance Calculated")

print(df)

#Function to find the correlation

corr = df.corr()

print("Correlation");

print(corr);

#Dividing the dataset into feature and target values

x =
df[['pickup_longitude','pickup_latitude','dropoff_longitude','dropoff_latitude','passenger_count','hour','day','mont
h','year','dayofweek','dist_travel_km']]

y = df['fare_amount']

#Dividing the dataset into training and testing dataset

from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(x,y,test_size = 0.33)

#Linear Regression

from sklearn.linear_model import LinearRegression

regression = LinearRegression()

regression.fit(X_train,y_train)

#To find the linear intercept

print("#To find the linear intercept");

print(regression.intercept_)

#To find the linear coeeficient
```

```
print("#To find the linear coeeficient");
print(regression.coef_);
#To predict the target values
prediction = regression.predict(X_test)
print("Prediction")
print(prediction)
print("Y Test ")
print(y_test)
#Metrics Evaluation using R2, Mean Squared Error, Root Mean Sqared Error
from sklearn.metrics import r2_score
print("R2 Score")
print(r2_score(y_test,prediction))
print("MSE")
from sklearn.metrics import mean_squared_error
MSE = mean_squared_error(y_test,prediction)
print(MSE)
print("RMSE")
RMSE = np.sqrt(MSE)
print(RMSE)
#Random Forest Regression
from sklearn.ensemble import RandomForestRegressor
#Here n_estimators means number of trees you want to build before making the prediction
rf = RandomForestRegressor(n_estimators=100)
rf.fit(X_train,y_train)
y_pred = rf.predict(X_test)
print(y_pred)
#Metrics evaluatin for Random Forest
R2_Random = r2_score(y_test,y_pred)
print("R2 Random")
print(R2_Random)
print("MSE_Random")
MSE_Random = mean_squared_error(y_test,y_pred)
print(MSE_Random)
print("RMSE")
```

RMSE_Random = np.sqrt(MSE_Random)

print(RMSE_Random)

Output

1. Pre-process the dataset

```
Original Dataset
        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]
```

```
Dataset after dropping the unnecessary columns
        fare_amount          pickup_datetime  pickup_longitude  \
0               7.5  2015-05-07 19:52:06 UTC        -73.999817
1               7.7  2009-07-17 20:04:56 UTC        -73.994355
2              12.9  2009-08-24 21:45:00 UTC        -74.005043
3               5.3  2009-06-26 08:22:21 UTC        -73.976124
4              16.0  2014-08-28 17:47:00 UTC        -73.925023
...             ...                      ...               ...
199995          3.0  2012-10-28 10:49:00 UTC        -73.987042
199996          7.5  2014-03-14 01:09:00 UTC        -73.984722
199997         30.9  2009-06-29 00:42:00 UTC        -73.986017
199998         14.5  2015-05-20 14:56:25 UTC        -73.997124
199999         14.1  2010-05-15 04:08:00 UTC        -73.984395

        pickup_latitude  dropoff_longitude  dropoff_latitude  passenger_cou
0             40.738354         -73.999512         40.723217
1             40.728225         -73.994710         40.750325
2             40.740770         -73.962565         40.772647
3             40.790844         -73.965316         40.803349
4             40.744085         -73.973082         40.761247
...                 ...                ...               ...
199995        40.739367         -73.986525         40.740297
199996        40.736837         -74.006672         40.739620
199997        40.756487         -73.858957         40.692588
199998        40.725452         -73.983215         40.695415
199999        40.720077         -73.985508         40.768793

[200000 rows x 7 columns]
```

2. Identify outliers

```
Outliers
        fare_amount  pickup_longitude  pickup_latitude  dropoff_longitude  \
0              7.50        -73.999817        40.738354         -73.999512
1              7.70        -73.994355        40.728225         -73.994710
2             12.90        -74.005043        40.740770         -73.962565
3              5.30        -73.976124        40.790844         -73.965316
4             16.00        -73.929786        40.744085         -73.973082
...             ...               ...              ...                ...
199995         3.00        -73.987042        40.739367         -73.986525
199996         7.50        -73.984722        40.736837         -74.006672
199997        22.25        -73.986017        40.756487         -73.922036
199998        14.50        -73.997124        40.725452         -73.983215
199999        14.10        -73.984395        40.720077         -73.985508

        dropoff_latitude  passenger_count  hour  day  month  year  dayofweek
0              40.723217              1.0    19    7      5  2015          3
1              40.750325              1.0    20   17      7  2009          4
2              40.772647              1.0    21   24      8  2009          0
3              40.803349              3.0     8   26      6  2009          4
4              40.761247              3.5    17   28      8  2014          3
...                  ...              ...   ...  ...    ...   ...        ...
199995         40.740297              1.0    10   28     10  2012          6
199996         40.739620              1.0     1   14      3  2014          4
199997         40.692588              2.0     0   29      6  2009          0
199998         40.695415              1.0    14   20      5  2015          2
199999         40.768793              1.0     4   15      5  2010          5

[200000 rows x 11 columns]
```

3.Check the correlation

```
Correlation
                  fare_amount  pickup_longitude  pickup_latitude  \
fare_amount          1.000000          0.154069        -0.110842
pickup_longitude     0.154069          1.000000         0.259497
pickup_latitude     -0.110842          0.259497         1.000000
dropoff_longitude    0.218675          0.425619         0.048889
dropoff_latitude    -0.125898          0.073290         0.515714
passenger_count      0.015778         -0.013213        -0.012889
hour                -0.023623          0.011579         0.029681
day                  0.004534         -0.003204        -0.001553
month                0.030817          0.001169         0.001562
year                 0.141277          0.010198        -0.014243
dayofweek            0.013652         -0.024652        -0.042310
dist_travel_km       0.786385          0.048446        -0.073362

                  dropoff_longitude  dropoff_latitude  passenger_count
fare_amount                0.218675         -0.125898         0.015778
pickup_longitude           0.425619          0.073290        -0.013213
pickup_latitude            0.048889          0.515714        -0.012889
dropoff_longitude          1.000000          0.245667        -0.009303
dropoff_latitude           0.245667          1.000000        -0.006308
passenger_count           -0.009303         -0.006308         1.000000
hour                      -0.046558          0.019783         0.020274
day                       -0.004007         -0.003479         0.002712
month                      0.002391         -0.001193         0.010351
year                       0.011346         -0.009603        -0.009749
dayofweek                 -0.003336         -0.031919         0.048550
dist_travel_km             0.155191         -0.052701         0.009884

                       hour       day     month      year  dayofweek  \
fare_amount       -0.023623  0.004534  0.030817  0.141277   0.013652
pickup_longitude   0.011579 -0.003204  0.001169  0.010198  -0.024652
pickup_latitude    0.029681 -0.001553  0.001562 -0.014243  -0.042310
dropoff_longitude -0.046558 -0.004007  0.002391  0.011346  -0.003336
dropoff_latitude   0.019783 -0.003479 -0.001193 -0.009603  -0.031919
passenger_count    0.020274  0.002712  0.010351 -0.009749   0.048550
hour               1.000000  0.004677 -0.003926  0.002156  -0.086947
day                0.004677  1.000000 -0.017360 -0.012170   0.005617
month             -0.003926 -0.017360  1.000000 -0.115859  -0.008786
year               0.002156 -0.012170 -0.115859  1.000000   0.006113
dayofweek         -0.086947  0.005617 -0.008786  0.006113   1.000000
dist_travel_km    -0.035708  0.001709  0.010050  0.022294   0.030382
```


```
                       hour       day     month      year  dayofweek
fare_amount       -0.023623  0.004534  0.030817  0.141277   0.013652
pickup_longitude   0.011579 -0.003204  0.001169  0.010198  -0.024652
pickup_latitude    0.029681 -0.001553  0.001562 -0.014243  -0.042310
dropoff_longitude -0.046558 -0.004007  0.002391  0.011346  -0.003336
dropoff_latitude   0.019783 -0.003479 -0.001193 -0.009603  -0.031919
passenger_count    0.020274  0.002712  0.010351 -0.009749   0.048550
hour               1.000000  0.004677 -0.003926  0.002156  -0.086947
day                0.004677  1.000000 -0.017360 -0.012170   0.005617
month             -0.003926 -0.017360  1.000000 -0.115859  -0.008786
year               0.002156 -0.012170 -0.115859  1.000000   0.006113
dayofweek         -0.086947  0.005617 -0.008786  0.006113   1.000000
dist_travel_km    -0.035708  0.001709  0.010050  0.022294   0.030382
```


```
                   dist_travel_km
fare_amount              0.786385
pickup_longitude         0.048446
pickup_latitude         -0.073362
dropoff_longitude        0.155191
dropoff_latitude        -0.052701
passenger_count          0.009884
hour                    -0.035708
day                      0.001709
month                    0.010050
year                     0.022294
dayofweek                0.030382
dist_travel_km           1.000000
```

4. Implement linear regression

```
from sklearn.linear_model import LinearRegression
regression = LinearRegression()
regression.fit(X_train,y_train)
```

```
LinearRegression
LinearRegression()
```

```
print("#To find the linear intercept");
print(regression.intercept_)
```

```
#To find the linear intercept
3635.5856985214964
```

```
prediction = regression.predict(X_test)
print("Prediction")
print(prediction)
```

```
Prediction
[10.94536423  8.06827567 10.78924417 ... 10.79770802 13.21194787
  9.29401109]
```

```
print("Y Test ")
print(y_test)
```

```
Y Test
137384     8.10
62615      7.50
3098      11.00
10235      7.70
187118     2.50
          ...
125942    22.25
162666     3.50
26469     10.00
104910     9.00
100909     8.10
Name: fare_amount, Length: 66000, dtype: float64
```

5. Evaluate the models and compare their respective scores like R2, RMSE, etc.

```
R2 Score
0.6673368068443333
```

```python
from sklearn.metrics import mean_squared_error
MSE = mean_squared_error(y_test,prediction)
print("MSE")
print(MSE)
print("RMSE")
RMSE = np.sqrt(MSE)
print(RMSE)
```

```
MSE
9.802420191767698
RMSE
3.1308816955879535
```

4.Random forest

```python
y_pred = rf.predict(x_test)
print(y_pred)
```

```
[ 9.448    6.775   11.375   ... 10.355   13.7675   7.396 ]
```

5. Evaluate the models and compare their respective scores like R2, RMSE, etc.

```python
#Metrics evaluatin for Random Forest
R2_Random = r2_score(y_test,y_pred)
print("R2 Random")
print(R2_Random)
print("MSE_Random")
MSE_Random = mean_squared_error(y_test,y_pred)
print(MSE_Random)

print("RMSE")

RMSE_Random = np.sqrt(MSE_Random)

print(RMSE_Random)
```

```
R2 Random
0.7982958868316123
MSE_Random
5.94351437839771
RMSE
2.4379323982419425
```