ML LAB Assignment 1

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. Dataset link: https://www.kaggle.com/datasets/yasserh/uber-fares-dataset[link (https://www.kaggle.com/datasets/yasserh/uber-fares-dataset%5Blink) text](https://)

In [ ]:
```
# Importing necessary packages
import pandas as pd
import numpy as np

import matplotlib as plt
```

In [ ]:
```
import seaborn as sns # for data plots
```

In [ ]:
```
data=pd.read_csv("/content/uber.csv")
```

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

Out[4]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_lati
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.73
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.72
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.74
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.79
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.74

Type *Markdown* and LaTeX: α^2

In [ ]:
```
data.info()
# Gives total information about the dataframe
```

```
<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
```

In [ ]:
```
#dropping unnamed column as not required
data=data.drop(['Unnamed: 0'], axis=1)
```

In [ ]:
```
# also dropping column key as not need for training
data=data.drop(['key'], axis=1)
```

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

Out[10]:

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

In [ ]:
```
data.shape
```

Out[11]:
```
(35725, 7)
```

In [ ]:
```
data.dtypes
```

Out[12]:
```
fare_amount          float64
pickup_datetime       object
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count      float64
dtype: object
```

In [ ]:
```
# count of null values in columns
data.isnull().sum()
```

Out[14]:
```
fare_amount          0
pickup_datetime      0
pickup_longitude     1
pickup_latitude      1
dropoff_longitude    1
dropoff_latitude     1
passenger_count      1
dtype: int64
```

In [ ]:
```
# Filling the null values by mean of the column data
data['dropoff_longitude'].fillna(value=data['dropoff_longitude'].mean(), inp
data['dropoff_latitude'].fillna(value=data['dropoff_latitude'].median(), inp
```

In [ ]:
```
# Dropping the rows of dataset having null values in passenger_count
# pickup longitutde and pickup latitude
data = data.dropna(subset=['pickup_longitude', 'passenger_count'])
```

In [ ]:
```
# data now has no null values in any columns
data.isnull().sum()
```

Out[19]:
```
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
dtype: int64
```

In [ ]:
```
# converting datatype of columns from float to int64
data['passenger_count'] = data['passenger_count'].astype('int64')
```

In [ ]:
```
# Printing the datatypes of the columns
data.dtypes
```

Out[22]:
```
fare_amount          float64
pickup_datetime       object
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count        int64
dtype: object
```

In [ ]:
```
# convert datetime from object type to datetime type
data.pickup_datetime=pd.to_datetime(data.pickup_datetime, errors='coerce')
```

In [ ]:
```
data.dtypes
```

Out[25]:
```
fare_amount                      float64
pickup_datetime      datetime64[ns, UTC]
pickup_longitude                 float64
pickup_latitude                  float64
dropoff_longitude                float64
dropoff_latitude                 float64
passenger_count                    int64
dtype: object
```

In [ ]:
```
# creating new columns for hour,day,month,year,dayofweek from pickup_datetim
data= data.assign(hour = data.pickup_datetime.dt.hour,
             day= data.pickup_datetime.dt.day,
             month = data.pickup_datetime.dt.month,
             year = data.pickup_datetime.dt.year,
             dayofweek = data.pickup_datetime.dt.dayofweek)
```

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

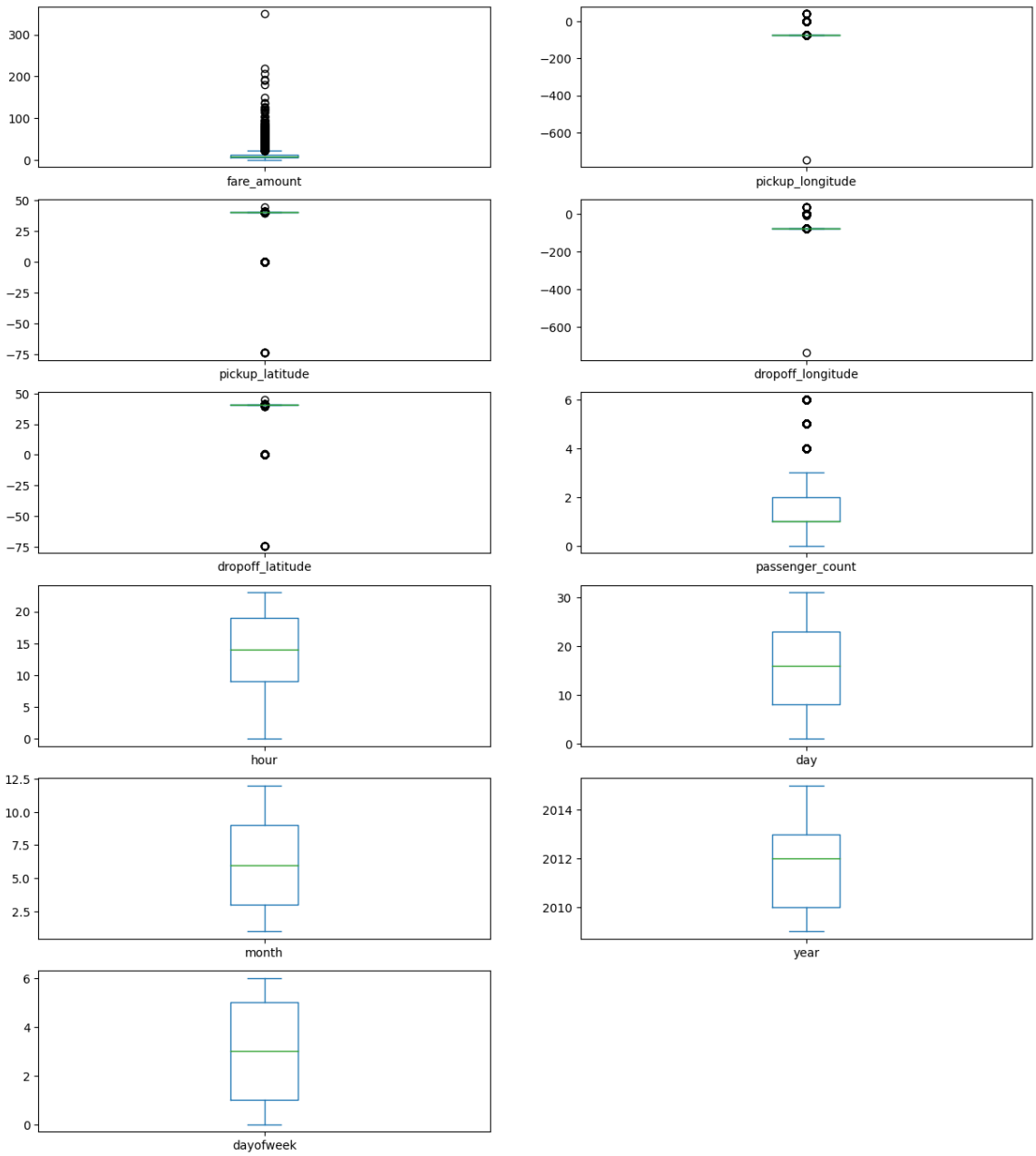
Out[27]:

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

In [ ]:
```
# dropping datetime as not needed further
data=data.drop(['pickup_datetime'], axis=1)
```

In [ ]:
```
# Plotting data for outliers
data.plot(kind = "box",subplots = True,layout = (7,2),figsize=(15,20))
```

Out[29]:
```
fare_amount            Axes(0.125,0.786098;0.352273x0.0939024)
pickup_longitude     Axes(0.547727,0.786098;0.352273x0.0939024)
pickup_latitude        Axes(0.125,0.673415;0.352273x0.0939024)
dropoff_longitude    Axes(0.547727,0.673415;0.352273x0.0939024)
dropoff_latitude       Axes(0.125,0.560732;0.352273x0.0939024)
passenger_count      Axes(0.547727,0.560732;0.352273x0.0939024)
hour                   Axes(0.125,0.448049;0.352273x0.0939024)
day                  Axes(0.547727,0.448049;0.352273x0.0939024)
month                  Axes(0.125,0.335366;0.352273x0.0939024)
year                 Axes(0.547727,0.335366;0.352273x0.0939024)
dayofweek              Axes(0.125,0.222683;0.352273x0.0939024)
dtype: object
```



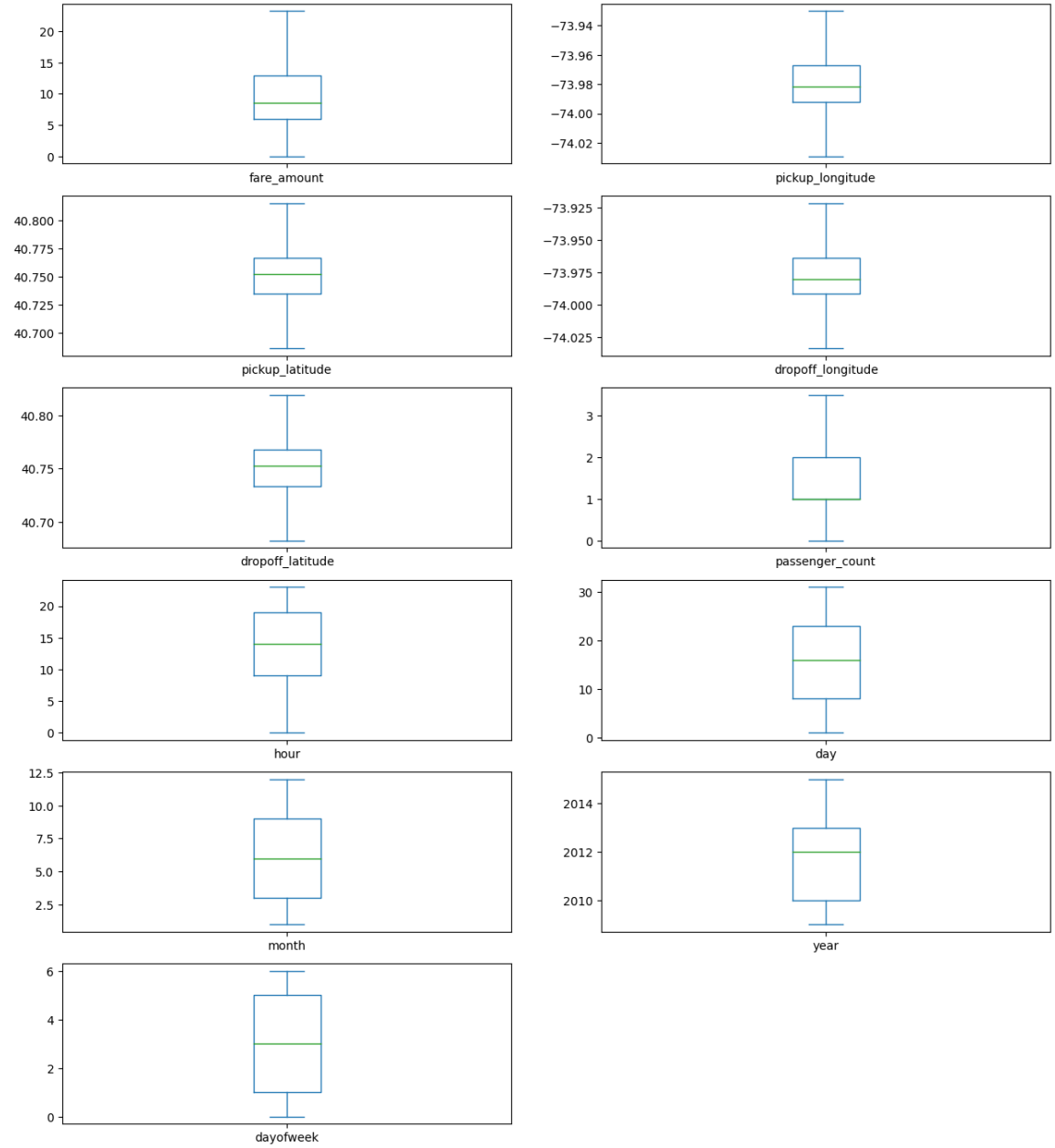
In [ ]:
```
#Using the InterQuartile Range to fill the values
def remove_outlier(data1 , col):
    Q1 = data1[col].quantile(0.25)
    Q3 = data1[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_whisker = Q1-1.5*IQR
    upper_whisker = Q3+1.5*IQR
    data[col] = np.clip(data1[col] , lower_whisker , upper_whisker)
    return data1

def treat_outliers_all(data1 , col_list):
    for c in col_list:
        data1 = remove_outlier(data , c)
    return data1
```

In [ ]:
```
# Calling treat_outliers to remove outliers
data=treat_outliers_all(data,data.iloc[: , 0::])
```

In [ ]:
```
# plots after removed outliers
data.plot(kind = "box",subplots = True,layout = (7,2),figsize=(15,20))
```

Out[32]:
```
fare_amount            Axes(0.125,0.786098;0.352273x0.0939024)
pickup_longitude     Axes(0.547727,0.786098;0.352273x0.0939024)
pickup_latitude        Axes(0.125,0.673415;0.352273x0.0939024)
dropoff_longitude    Axes(0.547727,0.673415;0.352273x0.0939024)
dropoff_latitude       Axes(0.125,0.560732;0.352273x0.0939024)
passenger_count      Axes(0.547727,0.560732;0.352273x0.0939024)
hour                   Axes(0.125,0.448049;0.352273x0.0939024)
day                  Axes(0.547727,0.448049;0.352273x0.0939024)
month                  Axes(0.125,0.335366;0.352273x0.0939024)
year                 Axes(0.547727,0.335366;0.352273x0.0939024)
dayofweek              Axes(0.125,0.222683;0.352273x0.0939024)
dtype: object
```



In [ ]:
```
!pip install haversine
```

```
Collecting haversine
  Downloading haversine-2.8.0-py2.py3-none-any.whl (7.7 kB)
Installing collected packages: haversine
Successfully installed haversine-2.8.0
```

In [ ]:
```
# for calculating distance between two coordinates
import haversine as hs
```

In [ ]:
```
# calculate and store the distance between two coordinates
travel_dist = []
for pos in range(len(data['pickup_longitude'])):
        long1,lati1,long2,lati2 = [data['pickup_longitude'][pos],data['picku
        loc1=(lati1,long1)
        loc2=(lati2,long2)
        c = hs.haversine(loc1,loc2)
        travel_dist.append(c)

print(travel_dist)
data['dist_travel_km'] = travel_dist
data.head()
```

```
[1.6833250775073447, 2.4575932783467835, 5.036384146783453, 1.6616857536
50294, 4.104580316730644, 0.0, 9.541187848508091, 0.808418691536387, 2.3
327142314177545, 4.889423641655177, 2.2508607308770285, 0.80841869153638
7, 0.3022521108558365, 3.5812557740132496, 1.3099517093917648, 1.7162797
92276335, 0.7299665570466272, 2.515953547298386, 1.790321726187665, 1.03
47050399795192, 2.4902472008677727, 0.9594701844599927, 1.26138976734378
17, 1.7517650017211177, 6.1932445014761095, 2.736192584061414, 0.7232537
124105735, 3.229443537425455, 1.4295172964395384, 2.233699311547041, 11.
086591403883585, 1.8950491608266506, 1.9049353402365328, 3.1821178242889
583, 4.485170626444128, 2.9230236888626995, 1.200213842045202, 2.6357908
07404098, 2.253613903554444, 9.208091771916012, 4.826533532872274, 1.250
2926870845612, 0.7984742276755328, 0.840396152683202, 0.3851992400170144
4, 2.434346176233359, 3.127909533264099, 3.7346507724526368, 0.0, 4.5043
60509008704, 1.5579008497303448, 4.923160610490316, 7.683158068167262,
0.6921558703072759, 4.523573083259081, 4.071896214031055, 1.174673708132
5782, 1.2923819784535335, 1.21232151965934, 0.8729716749950138, 2.107475
6058990856, 5.901643616945056, 0.6855640840334924, 5.634300221883286, 0.
7550126930090186, 0.808418691536387, 1.0321749030315608, 5.8471231457606
```

In [ ]:
```
# selecting columsn for x and y for model training
x = data[['pickup_longitude','pickup_latitude','dropoff_longitude','dropoff_
y = data['fare_amount']
```

In [ ]:
```
# importing for splitting data
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size = 0.33)
```

In [ ]:
```
# importing linearRegression
from sklearn.linear_model import LinearRegression
regression = LinearRegression()
```

In [ ]:
```
# fitting the model with x_train, y_train
regression.fit(X_train,y_train)
```

Out[40]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```

regression.intercept_ #To find the linear intercept
```

Out[41]:
```
4426.4725982283135
```

In [ ]:
```
regression.coef_ #To find the linear coeeficient
```

Out[42]:
```
array([ 2.99524239e+01, -1.01790899e+01,  2.37064937e+01, -2.02553543e+01,
        2.29021340e-02,  1.68030802e-02,  6.40743163e-03,  6.16932771e-02,
        3.91496627e-01, -3.99244822e-02,  1.88378578e+00])
```

In [ ]:
```
# predicting data for x_test
pred=regression.predict(X_test)
```

In [ ]:
```
pred
```

Out[45]:
```
array([ 9.43037453,  6.89882137,  7.77328839, ...,  7.98544231,
       14.20256744, 12.1397021 ])
```

In [ ]:
```
# importing for scores
from sklearn.metrics import r2_score
```

In [ ]:
```
r2_score(y_test,pred)
```

Out[47]:
```
0.6738514181247406
```

In [ ]:
```
# Mean Squared between predicted data and original data
MSE = mean_squared_error(y_test,pred)
```

In [ ]:
```
MSE
```

Out[49]:
```
10.250219126792926
```

In [ ]:
```
# root MSE
RMSE = np.sqrt(MSE)
```

In [ ]:
```
RMSE
```

Out[51]:
```
3.2015963403891075
```

In [ ]:
```
# importing random forest Regressor
from sklearn.ensemble import RandomForestRegressor
```

In [ ]:
```
rf = RandomForestRegressor(n_estimators=100)
```

In [ ]:
```
# Training the model
rf.fit(X_train,y_train)
```

Out[54]:
```
RandomForestRegressor()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```
# prediciting values again
pred = rf.predict(X_test)
```

In [ ]:
```
pred
```

Out[56]:
```
array([10.62  ,  4.296 ,  5.495 , ...,  7.5455, 13.872 , 12.1383])
```

In [ ]:
```
R=r2_score(y_test,pred)
```

In [ ]:
```
MSE = mean_squared_error(y_test,pred)
```

In [ ]:
```
MSE
```

Out[59]:
```
6.951710447309044
```

In [ ]:
```
RMSE = np.sqrt(MSE)
```

In [ ]:
```
RMSE
```

Out[61]:
```
2.636609650158522
```

In [ ]:
```

```