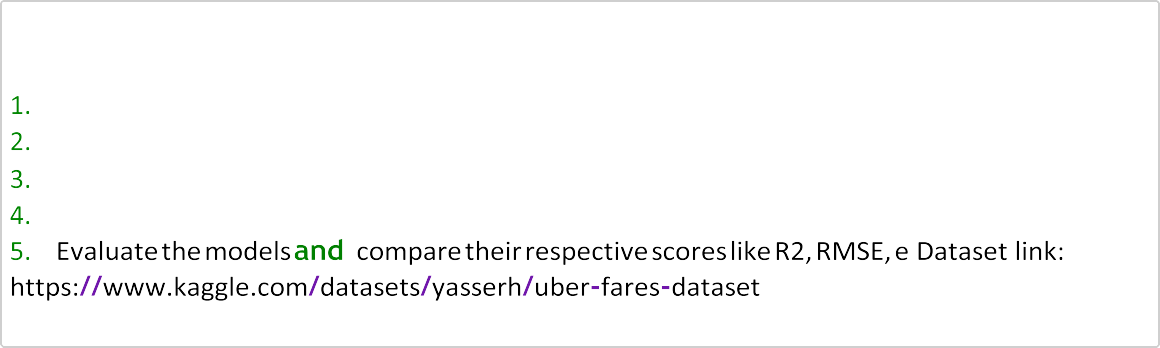
In [ ]:

In [ ]:



Practical No : 1

Predict the price of the Uber ride **from** a given pickup point to the agreed Perform following tasks: Pre**-**process the dataset.

Identify outliers. Check the correlation.

Implement linear regression **and** random forest regression models.

In [1]:



**from** sklearn.model\_selection **import** train\_test\_split

In [2]:

df **=** pd.read\_csv('uber.csv') 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

#Preprocess the data

In [3]:

df.shape

Out[3]: (200000, 9)

In [4]:

df.head()

Out[4]: **Unnamed: 0**

**key fare\_amount pickup\_datetime pickup\_longitude pickup\_la**

17:47:00.000000188 17:47:00 UTC

|  |  |
| --- | --- |
| **0** 24238194 19:52: 2015-05-07 7.5 192015-05-07 -73.999817  06.0000003 :52:06 UTC | 40.7 |
| **1** 27835199 20:04: 2009-07-17 7.7 202009-07-17 -73.994355  56.0000002 :04:56 UTC | 40.7 |
| **2** 44984355 21:45:0 2009-08-24 12.9 2 2009-08-24 -74.005043  0.00000061 1:45:00 UTC | 40.7 |
| **3** 25894730 08:22: 2009-06-26 5.3 082009-06-26 -73.976124  21.0000001 :22:21 UTC | 40.7 |
| **4** 17610152 2014-08-28 16.0 2014-08-28 -73.925023 | 40.7 |



In [5]:

df.isnull()

Out[5]:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Unnamed:**  **0** | | **key** | **fare\_amount** | **pickup\_datetime** | **pickup\_longitude** | **pickup\_latitude** |
| **0** | False | False | False | False | False | False |
| **1** | False | False | False | False | False | False |
| **2** | False | False | False | False | False | False |
| **3** | False | False | False | False | False | False |
| **4** | False | False | False | False | False | False |
| **...** | ... | ... | ... | ... | ... | ... |
| **199995** | False | False | False | False | False | False |
| **199996** | False | False | False | False | False | False |
| **199997** | False | False | False | False | False | False |
| **199998** | False | False | False | False | False | False |
| **199999** | False | False | False | False | False | False |

200000 rows × 9 columns



In [6]:

df.drop(columns**=**["Unnamed: 0", "key"], inplace**=True**) df.head()

Out[6]: **fare\_amount pickup\_datetime pickup\_longitude pickup\_latitude dropoff\_longitude dropo**

|  |  |  |  |
| --- | --- | --- | --- |
| 05-07 -73.999817  6 UTC | 40.738354 | -73.999512 |  |
| 07-17 -73.994355 | 40.728225 | -73.994710 |  |
| 08-24 -74.005043 | 40.740770 | -73.962565 |  |
| 06-26  -73.976124 | 40.790844 | -73.965316 |  |

|  |  |  |
| --- | --- | --- |
|  | **0** | 7.5 |
| **1** | 7.7 |
| **2** | 12.9 |
| **3** | 5.3 |
|  | **4** | 16.0 |

2015-

19:52:0

2009-

20:04:56 UTC

2009-

21:45:00 UTC

2009-

08:22:21 UTC

2014-08-28

17:47:00 UTC

-73.925023 40.744085 -73.973082



|  |  |  |
| --- | --- | --- |
| In [7]: | df.isnull().sum() |  |
| Out[7]: | fare\_amount | 0 |
|  | pickup\_datetime | 0 |
|  | pickup\_longitude | 0 |
|  | pickup\_latitude | 0 |
|  | dropoff\_longitude | 1 |
|  | dropoff\_latitude | 1 |
|  | passenger\_count | 0 |
|  | dtype: int64 |  |

In [8]:

df['dropoff\_latitude'].fillna(value**=**df['dropoff\_latitude'].mean(),

inplace **= True**) df['dropoff\_longitude'].fillna(value**=**df['dropoff\_longitude'].median(),

inplace **= True**)

In [9]:

df.dtypes

Out[9]: fare\_amount float64 pickup\_datetime object pickup\_longitude float64 pickup\_latitude float64

dropoff\_longitude float64

dropoff\_latitude float64 passenger\_count int64 dtype: object

In [10]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *# From # But* | *the above output, we 'pickup\_datetime'is a* | *see that the data type of date time stamp variable,* | *'pickup\_datetime' which is wrongly* | *is inte* |
| C |  |  |  | C |

In [11]:

df.pickup\_datetime **=** pd.to\_datetime(df.pickup\_datetime) df.dtypes

|  |  |  |
| --- | --- | --- |
| Out[11]: fare\_amount  pickup\_datetime pickup\_longitude | datetime64[ns, | float64  UTC]  float64 |
| pickup\_latitude dropoff\_longitude dropoff\_latitude passenger\_count  dtype: object |  | float64 float64 float64  int64 |

In [12]:

In [13]:

In [14]:

Out[14]:

df **=** df.assign(hour **=** df.pickup\_datetime.dt.hour,

day **=** df.pickup\_datetime.dt.day,

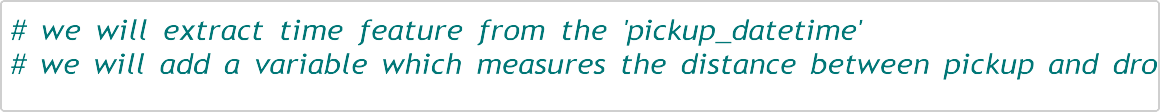
month **=** df.pickup\_datetime.dt.month, year **=**

df.pickup\_datetime.dt.year,

dayofweek **=** df.pickup\_datetime.dt.dayofweek)

df

|  |  |  |
| --- | --- | --- |
| **fare\_amount pickup\_datetime pickup\_longitude** | **pickup\_latitude** | **dropoff\_longitude** |
| **0** 7.5 2015-05-07 -73.999817  19:52:06+00:00 | 40.738354 | -73.999512 |
| **1** 7.7 2009-07-17 -73.994355  20:04:56+00:00 | 40.728225 | -73.994710 |
| **2** 12.9 2009-08-24 -74.005043  21:45:00+00:00 | 40.740770 | -73.962565 |
| **3** 5.3 2009-06-26 -73.976124  08:22:21+00:00 | 40.790844 | -73.965316 |
| **4** 16.0 2014-08-28 -73.925023  17:47:00+00:00 | 40.744085 | -73.973082 |
| **...** ... ... ... | ... | ... |
| **199995** 3.0 2012-10-28 -73.987042  10:49:00+00:00 | 40.739367 | -73.986525 |
| **199996** 7.5 2014-03-14 -73.984722  01:09:00+00:00 | 40.736837 | -74.006672 |
| **199997** 30.9 2009-06-29 -73.986017  00:42:00+00:00 | 40.756487 | -73.858957 |
| **199998** 14.5 2015-05-20 -73.997124  14:56:25+00:00 | 40.725452 | -73.983215 |
| **199999** 14.1 2010-05-15 -73.984395 | 40.720077 | -73.985508 |

04:08:00+00:00

200000 rows × 12 columns



In [15]:

df **=** df.drop(["pickup\_datetime"], axis **=**1) df

Out[15]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **fare\_amount** | **pickup\_longitude** | **pickup\_latitude** | **dropoff\_longitude** | **dropoff\_latitude** |
| **0** | 7.5 | -73.999817 | 40.738354 | -73.999512 | 40.723217 |
| **1** | 7.7 | -73.994355 | 40.728225 | -73.994710 | 40.750325 |
| **2** | 12.9 | -74.005043 | 40.740770 | -73.962565 | 40.772647 |
| **3** | 5.3 | -73.976124 | 40.790844 | -73.965316 | 40.803349 |
| **4** | 16.0 | -73.925023 | 40.744085 | -73.973082 | 40.761247 |
| **...** | ... | ... | ... | ... | ... |
| **199995** | 3.0 | -73.987042 | 40.739367 | -73.986525 | 40.740297 |
| **199996** | 7.5 | -73.984722 | 40.736837 | -74.006672 | 40.739620 |
| **199997** | 30.9 | -73.986017 | 40.756487 | -73.858957 | 40.692588 |
| **199998** | 14.5 | -73.997124 | 40.725452 | -73.983215 | 40.695415 |
| **199999** | 14.1 | -73.984395 | 40.720077 | -73.985508 | 40.768793 |

200000 rows × 11 columns



In [16]:

*# function to calculate the travel distance from the longitudes and latitud*

**from** math **import \***

**def** distance\_formula(longitude1, latitude1, longitude2, latitude2): travel\_dist **=** []

**for** pos **in** range (len(longitude1)):

lon1, lan1, lon2, lan2 **=** map(radians, [longitude1[pos],

latitude1[pos], longitude2[pos], latitu

dist\_lon**=** lon2 **-** lon1 dist\_lan **=** lan2 **-** lan1

a **=** sin(dist\_lan**/**2)**\*\***2 **+** cos(lan1) **\*** cos(lan2) **\*** sin(dist\_lon**/**2)**\*\***2

*#radius of earth = 6371* c **=** 2 **\*** asin(sqrt(a)) **\*** 6371 travel\_dist.append(c)

**return** travel\_dist

In [17]:

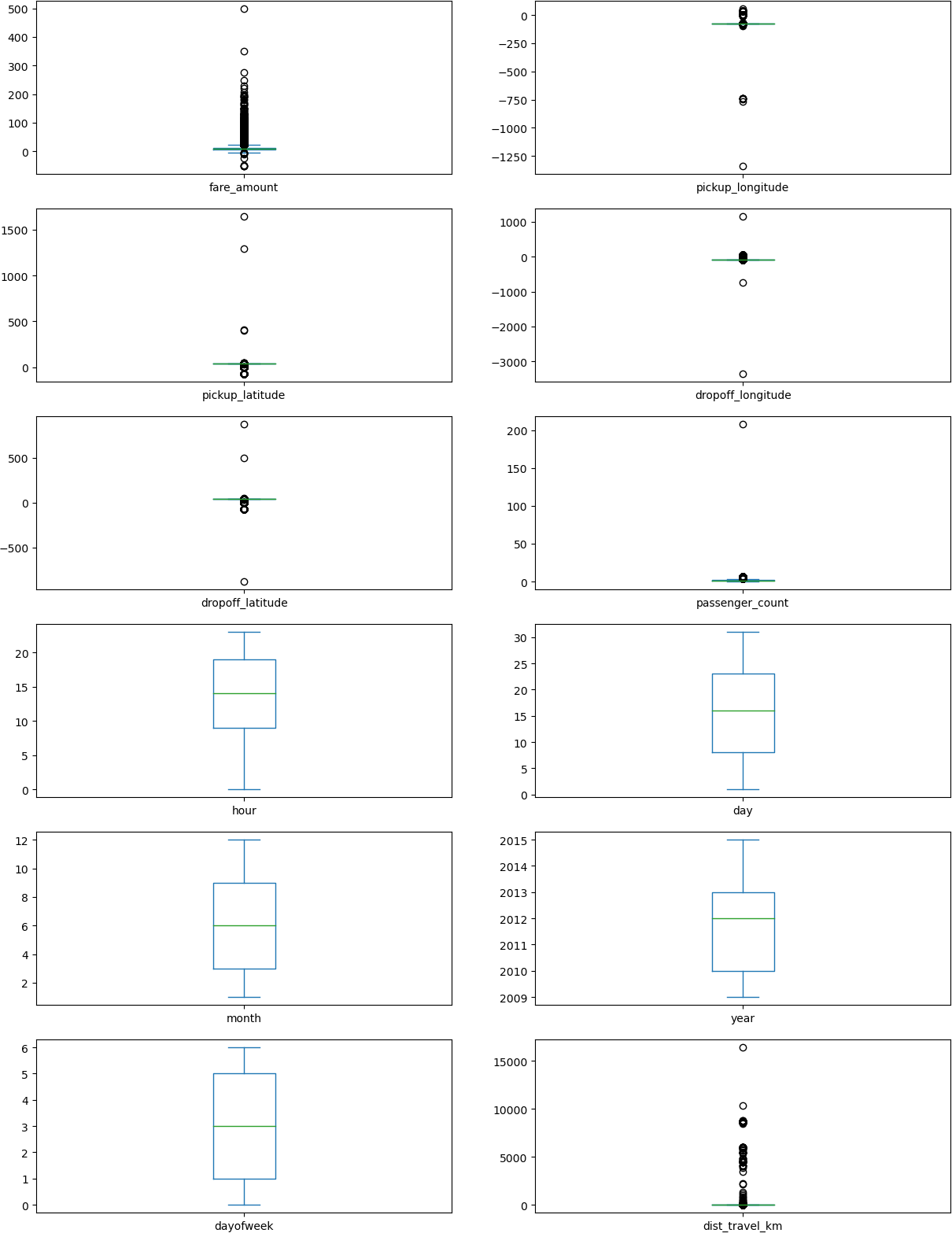
df['dist\_travel\_km'] **=** distance\_formula(df.pickup\_longitude.to\_numpy(),

df.pickup\_latitude.to\_numpy(), df.dropoff\_longitude.to\_numpy(), df.dropoff\_latitude.to\_numpy())

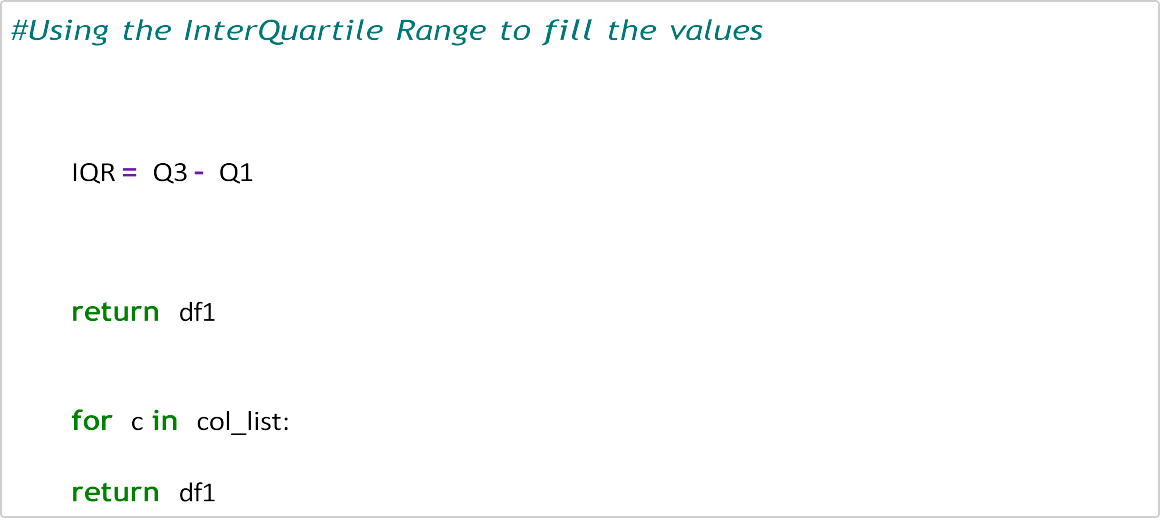
# Identify Outliers

In [18]: df.plot(kind **=** "box",subplots **= True**,layout **=** (6,2),figsize**=**(15,20)) *#Boxpl*

plt.show()



In [19]:



**def** remove\_outlier(df1 , col):

Q1 **=** df1[col].quantile(0.25) Q3 **=**

df1[col].quantile(0.75)

lower\_whisker **=** Q1**-**1.5**\***IQR upper\_whisker **=** Q3**+**1.5**\***IQR

df[col] **=** np.clip(df1[col] , lower\_whisker , upper\_whisker)

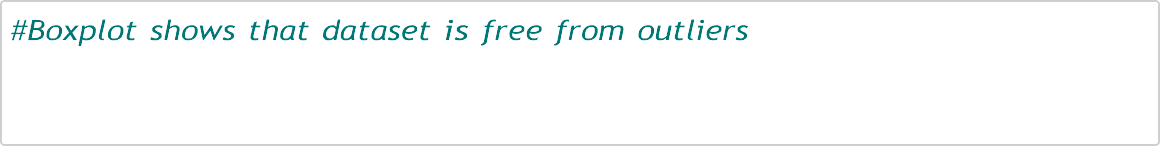
**def** treat\_outliers\_all(df1 , col\_list):

df1 **=** remove\_outlier(df , c)

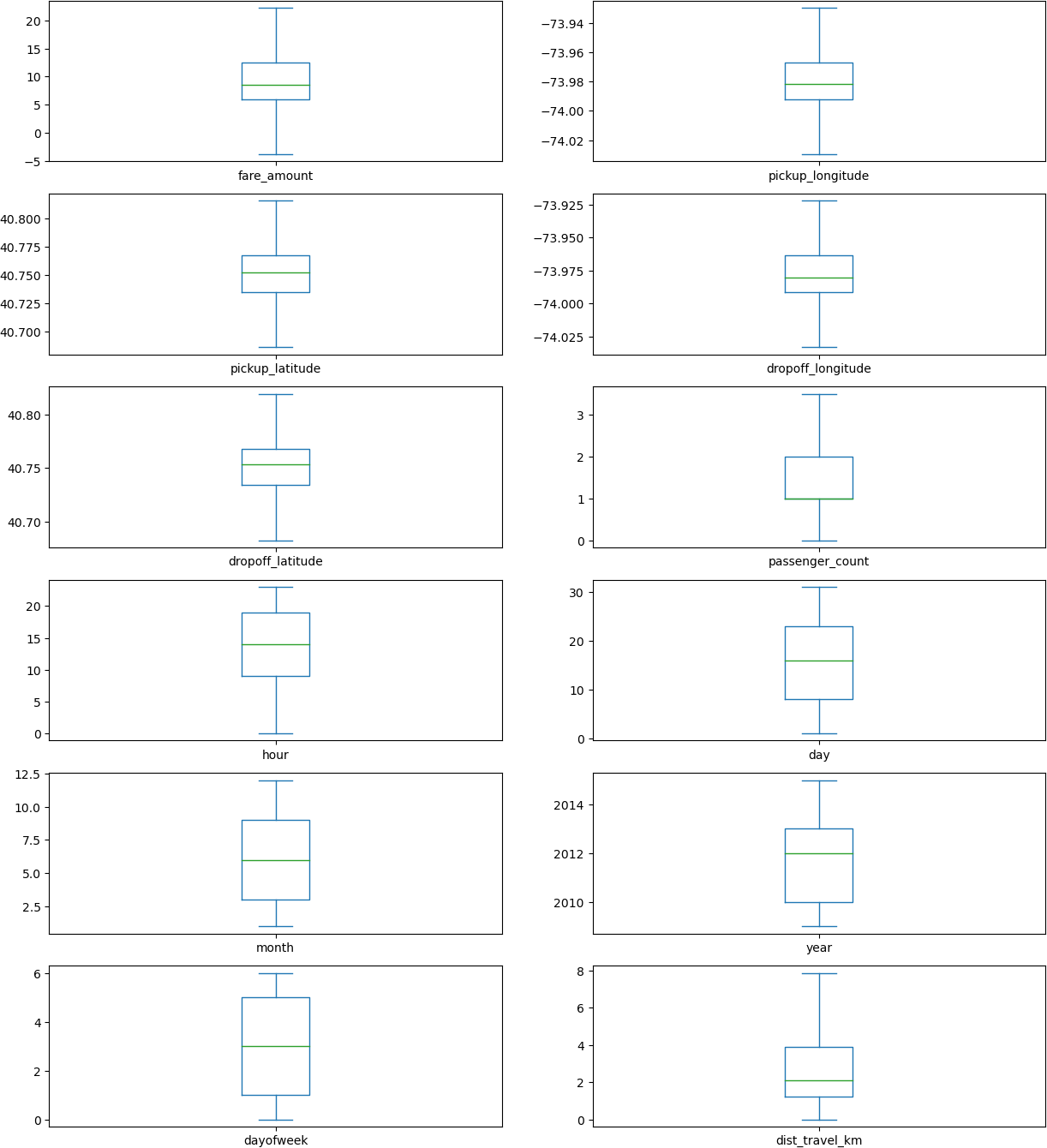
In [20]:

df **=** treat\_outliers\_all(df , df.iloc[: , 0::])

In [21]:



df.plot(kind **=** "box",subplots **= True**,layout **=** (7,2),figsize**=**(15,20)) plt.show()



Check the correlation

In [22]:



corr **=** df.corr() corr

Out[22]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **fare\_amount** | | **pickup\_longitude** | **pickup\_latitude** | **dropoff\_longitude** | **dropoff** |
| **fare\_amount** | 1.000000 | 0.154069 | -0.110842 | 0.218675 | - |
| **pickup\_longitude** | 0.154069 | 1.000000 | 0.259497 | 0.425619 |  |
| **pickup\_latitude** | -0.110842 | 0.259497 | 1.000000 | 0.048889 |  |
| **dropoff\_longitude** | 0.218675 | 0.425619 | 0.048889 | 1.000000 |  |
| **dropoff\_latitude** | -0.125898 | 0.073290 | 0.515714 | 0.245667 |  |
| **passenger\_count** | 0.015778 | -0.013213 | -0.012889 | -0.009303 | - |
| **hour** | -0.023623 | 0.011579 | 0.029681 | -0.046558 |  |
| **day** | 0.004534 | -0.003204 | -0.001553 | -0.004007 | - |
| **month** | 0.030817 | 0.001169 | 0.001562 | 0.002391 | - |
| **year** | 0.141277 | 0.010198 | -0.014243 | 0.011346 | - |
| **dayofweek** | 0.013652 | -0.024652 | -0.042310 | -0.003336 | - |
| **dist\_travel\_km** | 0.844374 | 0.098094 | -0.046812 | 0.186531 | - |

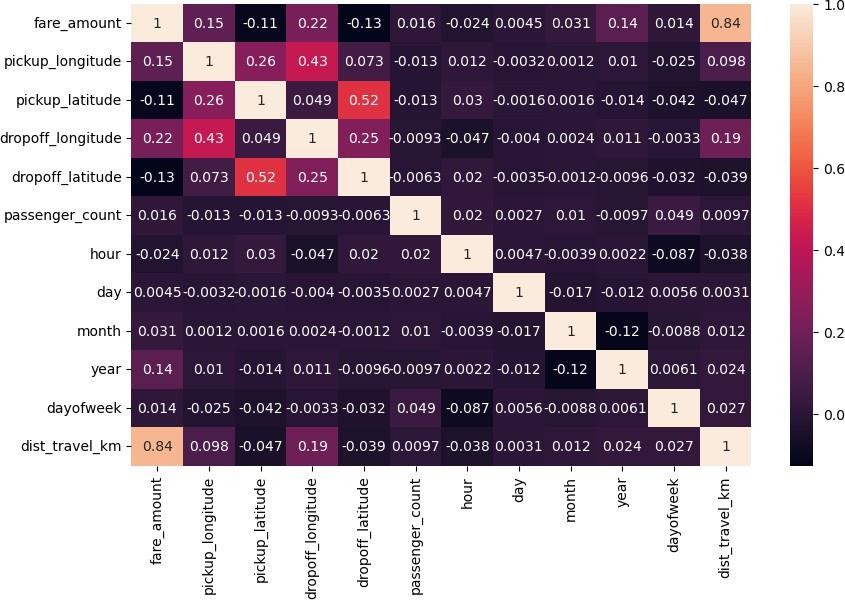


In [23]:

fig,axis **=** plt.subplots(figsize **=** (10,6))

sns.heatmap(df.corr(),annot **= True**) *#Correlation Heatmap (Light values mean*

Out[23]: <Axes: >



Implement linear regression and random forest regression models.

In [25]:

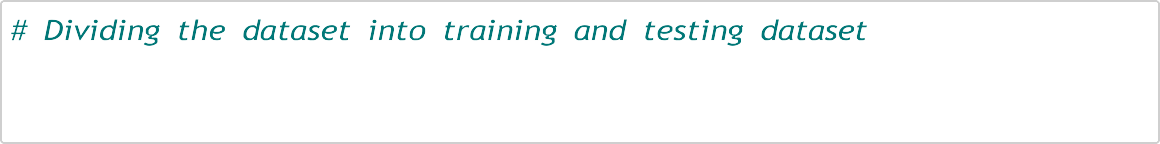


df\_x **=** df[['pickup\_longitude','pickup\_latitude','dropoff\_longitude'

,'dropoff\_latitude','passenger\_count','hour','day','month', 'year','dayofweek','dist\_travel\_km']]

df\_y **=** df['fare\_amount']

In [26]:



x\_train, x\_test, y\_train, y\_test **=** train\_test\_split(df\_x, df\_y,

test\_size**=**0.2, random\_state**=**1)

In [27]:

df

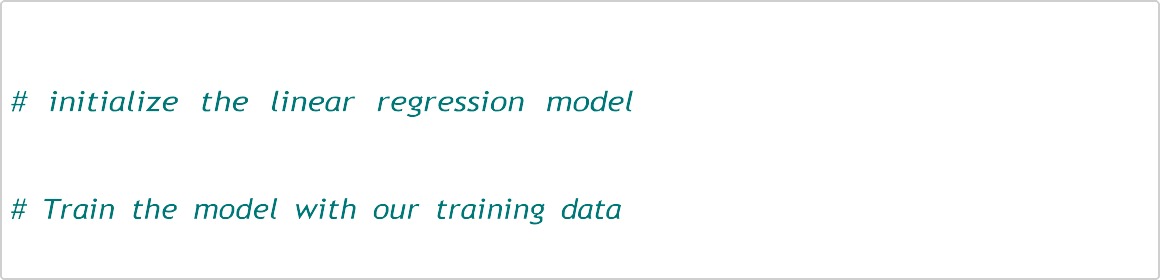
Out[27]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **fare\_amount** | **pickup\_longitude** | **pickup\_latitude** | **dropoff\_longitude** | **dropoff\_latitude** |
| **0** | 7.50 | -73.999817 | 40.738354 | -73.999512 | 40.723217 |
| **1** | 7.70 | -73.994355 | 40.728225 | -73.994710 | 40.750325 |
| **2** | 12.90 | -74.005043 | 40.740770 | -73.962565 | 40.772647 |
| **3** | 5.30 | -73.976124 | 40.790844 | -73.965316 | 40.803349 |
| **4** | 16.00 | -73.929786 | 40.744085 | -73.973082 | 40.761247 |
| **...** | ... | ... | ... | ... | ... |
| **199995** | 3.00 | -73.987042 | 40.739367 | -73.986525 | 40.740297 |
| **199996** | 7.50 | -73.984722 | 40.736837 | -74.006672 | 40.739620 |
| **199997** | 22.25 | -73.986017 | 40.756487 | -73.922036 | 40.692588 |
| **199998** | 14.50 | -73.997124 | 40.725452 | -73.983215 | 40.695415 |
| **199999** | 14.10 | -73.984395 | 40.720077 | -73.985508 | 40.768793 |

200000 rows × 12 columns



In [28]:



**from** sklearn.linear\_model **import** LinearRegression

reg **=** LinearRegression()

reg.fit(x\_train, y\_train)

Out[28]: 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.**

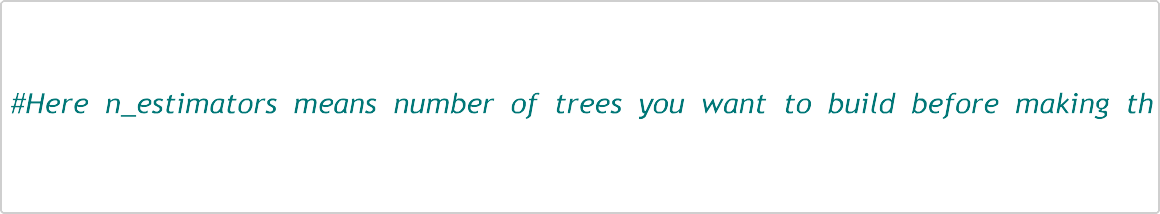
In [29]:

y\_pred\_lin **=** reg.predict(x\_test) print(y\_pred\_lin)

[ 6.27615184 5.09986098 9.43641238 ... 11.07663949 12.15392248

11.41496075]

In [30]:



**from** sklearn.ensemble **import** RandomForestRegressor

rf **=** RandomForestRegressor(n\_estimators**=**100) rf.fit(x\_train,y\_train)

Out[30]: 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.**

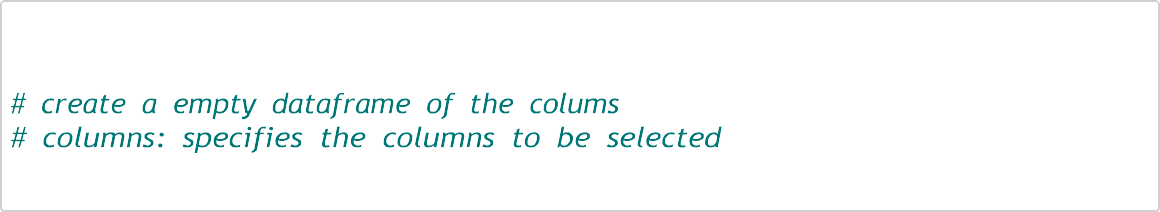
In [31]:

y\_pred\_rf **=** rf.predict(x\_test) print(y\_pred\_rf)

[ 4.8275 6.758 9.145 ... 11.255 11.064 13.5 ]

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

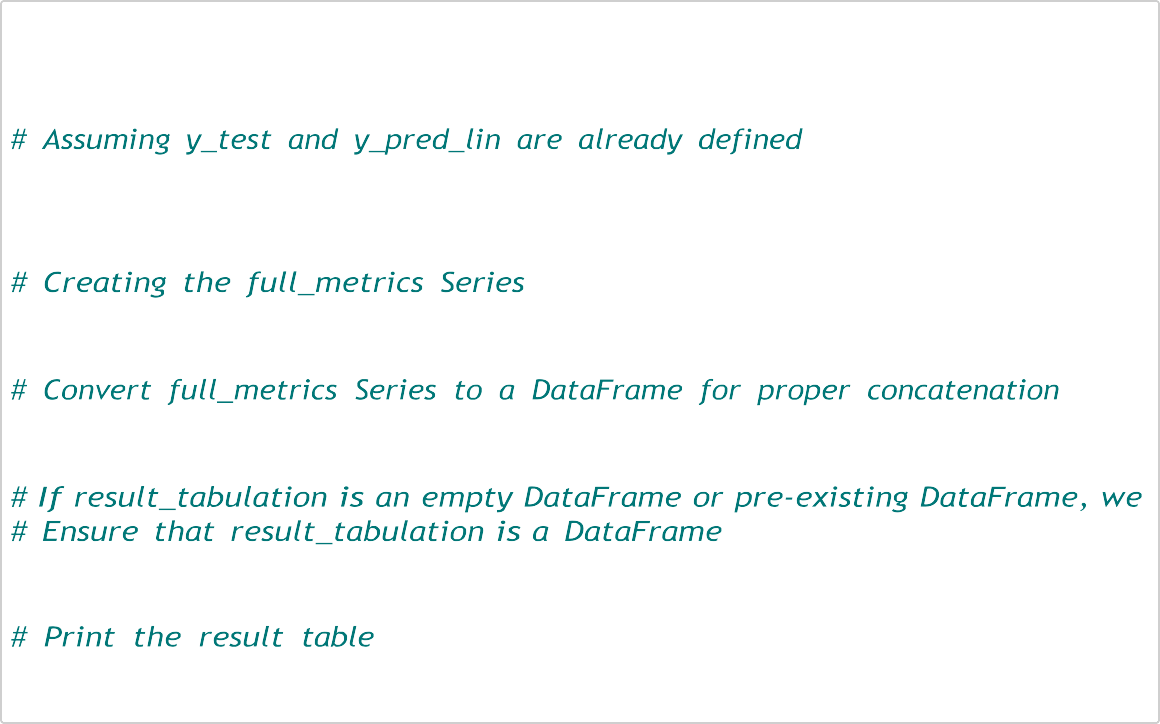
In [32]:



cols **=** ['Model', 'RMSE', 'R-Squared']

result\_tabulation **=** pd.DataFrame(columns **=** cols)

In [37]:



**from** sklearn **import** metrics

**from** sklearn.metrics **import** r2\_score

reg\_RMSE **=** np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred\_lin)) reg\_squared **=**

r2\_score(y\_test, y\_pred\_lin)

full\_metrics **=** pd.Series({'Model': "Linear Regression", 'RMSE': reg\_RMSE, '

full\_metrics\_df **=** full\_metrics.to\_frame().T

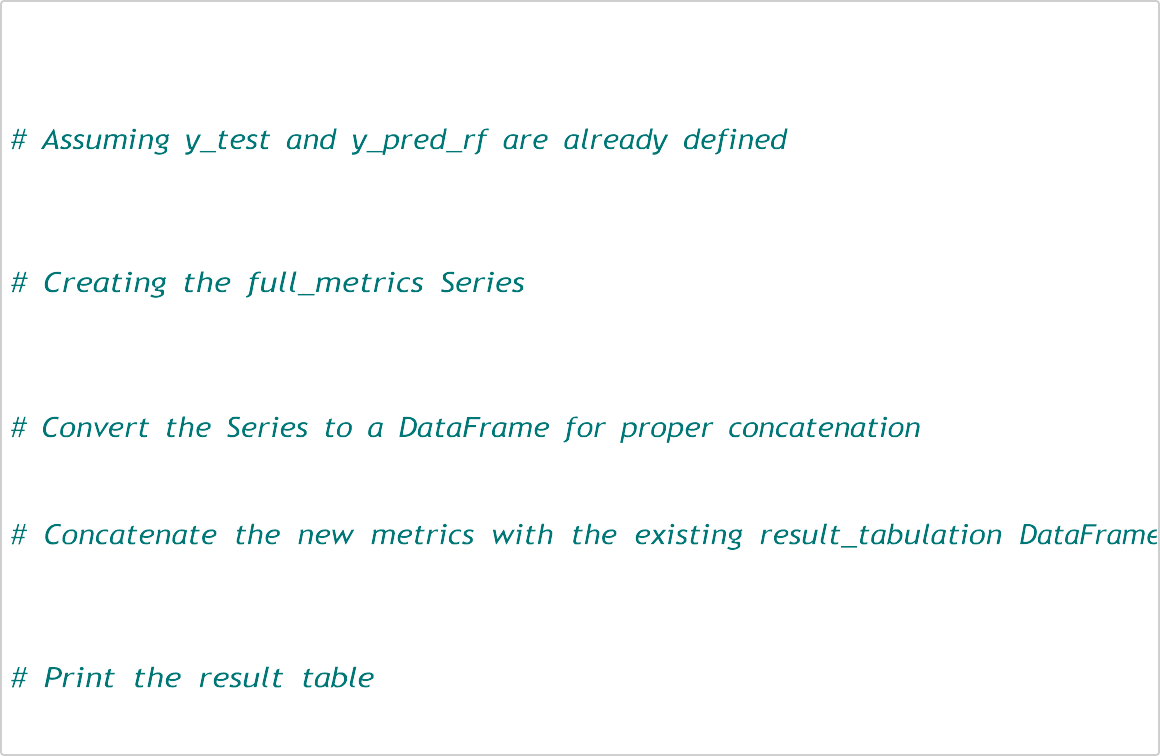
result\_tabulation **=** pd.concat([result\_tabulation, full\_metrics\_df], ignore\_

print(result\_tabulation)

Model RMSE R-Squared

0 Linear Regression 2.703957 0.753906

In [38]:



**from** sklearn **import** metrics

**from** sklearn.metrics **import** r2\_score

rf\_RMSE **=** np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred\_rf)) rf\_squared **=**

r2\_score(y\_test, y\_pred\_rf)

full\_metrics **=** pd.Series({'Model': "Random Forest", 'RMSE': rf\_RMSE,

'R-Squared': rf\_squared})

full\_metrics\_df **=** full\_metrics.to\_frame().T

result\_tabulation **=** pd.concat([result\_tabulation, full\_metrics\_df],

ignore\_index**=True**)

print(result\_tabulation)

Model RMSE R-Squared

0 Linear Regression 2.703957 0.753906

1 Random Forest 2.362658 0.81211