```
!pip install geopy
```

```
Requirement already satisfied: geopy in c:\users\shubham\anaconda3\
lib\site-packages (2.4.1)
Requirement already satisfied: geographiclib<3,>=1.52 in c:\users\
shubham\anaconda3\lib\site-packages (from geopy) (2.0)
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split

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
```

```
# 1. Pre-process the dataset
df.shape
```

```
(200000, 9)
```

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

   fare_amount          pickup_datetime  pickup_longitude
pickup_latitude  \
0          7.5  2015-05-07 19:52:06 UTC        -73.999817
40.738354
1          7.7  2009-07-17 20:04:56 UTC        -73.994355
40.728225
2         12.9  2009-08-24 21:45:00 UTC        -74.005043
40.740770
3          5.3  2009-06-26 08:22:21 UTC        -73.976124
40.790844
4         16.0  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.isnull()

        fare_amount  pickup_datetime  pickup_longitude
pickup_latitude  \
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

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

[200000 rows x 7 columns]
```

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

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

```
df['dropoff_latitude'].fillna(value = df['dropoff_latitude'].mean(),
inplace = True)
df['dropoff_longitude'].fillna(value =
df['dropoff_longitude'].median(), inplace = True)
```

```
C:\Users\Shubham\AppData\Local\Temp\ipykernel_15420\758970157.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
```

```
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df['dropoff_latitude'].fillna(value = df['dropoff_latitude'].mean(),
inplace = True)
C:\Users\Shubham\AppData\Local\Temp\ipykernel_15420\758970157.py:2:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df['dropoff_longitude'].fillna(value =
df['dropoff_longitude'].median(), inplace = True)

df.dtypes

fare_amount          float64
pickup_datetime       object
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count        int64
dtype: object

# From the above output, we see that the data type of
'pickup_datetime' is 'object
# But 'pickup_datetime'is a date time stamp variable, which is wrongly
interpreted as 'object', so we will convert this variable data type to
'datetime'.
df.pickup_datetime = pd.to_datetime(df.pickup_datetime)
df.dtypes

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
```

```
# We will extract time feature from the 'pickup_datetime'
# We will add a variable which measures the distance between pickup
and drop
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 = df.drop(["pickup_datetime"], axis = 1)
df
```

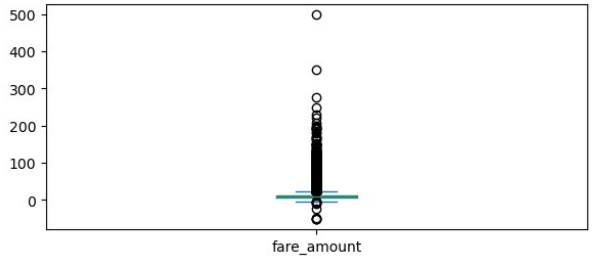
```
        fare_amount  pickup_longitude  pickup_latitude
dropoff_longitude  \
0               7.5        -73.999817        40.738354          -
73.999512
1               7.7        -73.994355        40.728225          -
73.994710
2              12.9        -74.005043        40.740770          -
73.962565
3               5.3        -73.976124        40.790844          -
73.965316
4              16.0        -73.925023        40.744085          -
73.973082
...             ...               ...               ...
...
199995          3.0        -73.987042        40.739367          -
73.986525
199996          7.5        -73.984722        40.736837          -
74.006672
199997         30.9        -73.986017        40.756487          -
73.858957
199998         14.5        -73.997124        40.725452          -
73.983215
199999         14.1        -73.984395        40.720077          -
73.985508

        dropoff_latitude  passenger_count  hour  day  month  year
dayofweek
0              40.723217                1    19    7      5  2015
3
1              40.750325                1    20   17      7  2009
4
2              40.772647                1    21   24      8  2009
0
3              40.803349                3     8   26      6  2009
4
4              40.761247                5    17   28      8  2014
3
...                  ...              ...   ...  ...    ...   ...
```

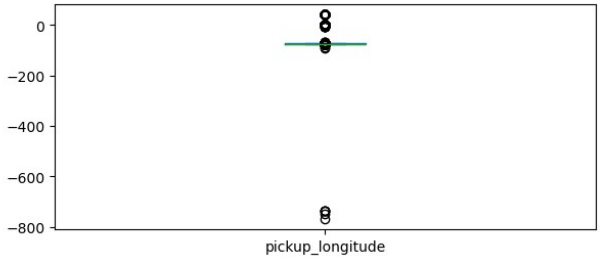
```
...
199995          40.740297                  1    10   28      10  2012
6
199996          40.739620                  1     1   14       3  2014
4
199997          40.692588                  2     0   29       6  2009
0
199998          40.695415                  1    14   20       5  2015
2
199999          40.768793                  1     4   15       5  2010
5

[200000 rows x 11 columns]
```

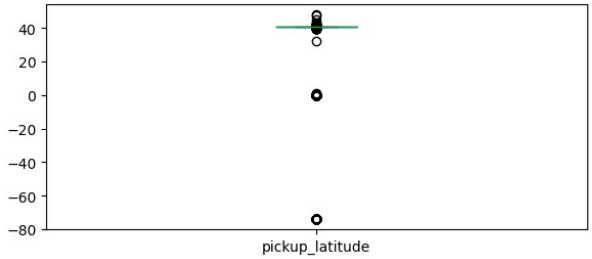
```
from geopy.distance import geodesic

# Filter out rows with valid latitude values
df = df[(df['pickup_latitude'] >= -90) & (df['pickup_latitude'] <= 90)
&
               (df['dropoff_latitude'] >= -90) &
(df['dropoff_latitude'] <= 90)]

df['dist_travel_km'] = df.apply(
    lambda row: geodesic((row['pickup_latitude'],
row['pickup_longitude']),
                         (row['dropoff_latitude'],
row['dropoff_longitude'])).kilometers, axis=1)
```

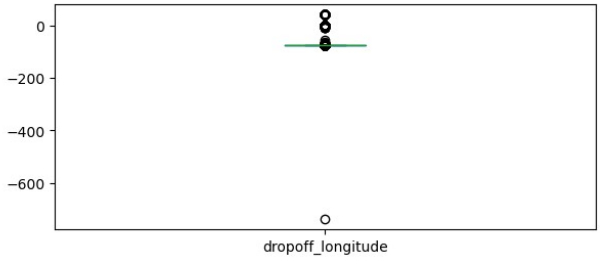
```
C:\Users\Shubham\AppData\Local\Temp\ipykernel_15420\1777527879.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df['dist_travel_km'] = df.apply(
```

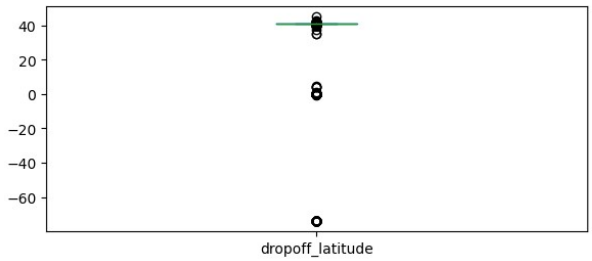
```
# 2. Identify outliers
df.plot(kind = "box", subplots = True, layout = (6, 2), figsize = (15,
20))
plt.show()
```

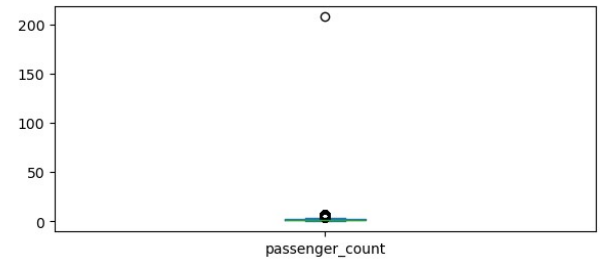

fare_amount

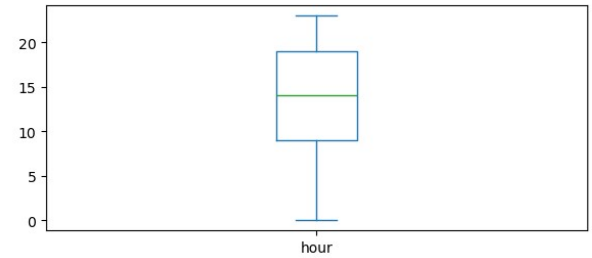

pickup_longitude

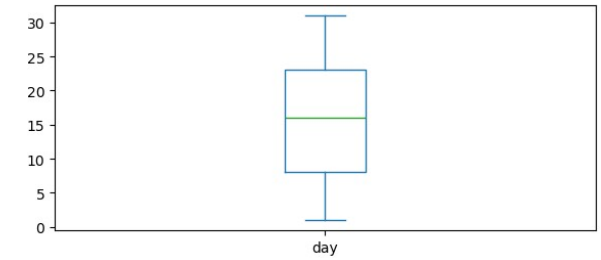

pickup_latitude

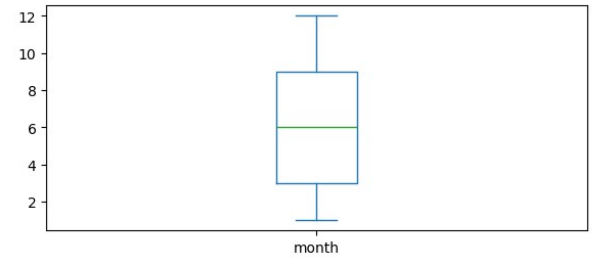

dropoff_longitude

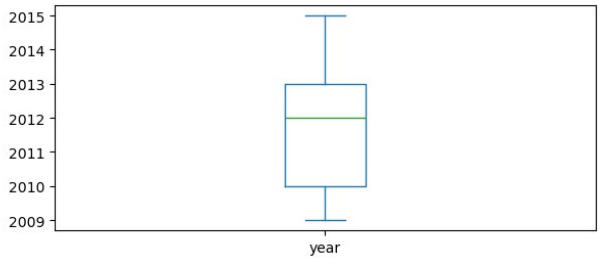

dropoff_latitude

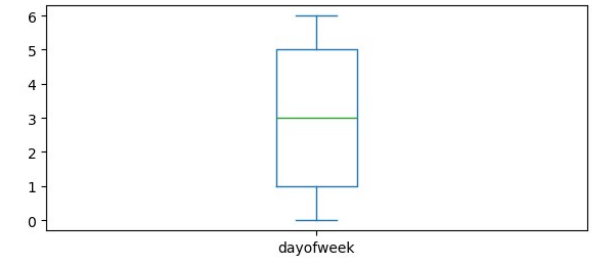

passenger_count

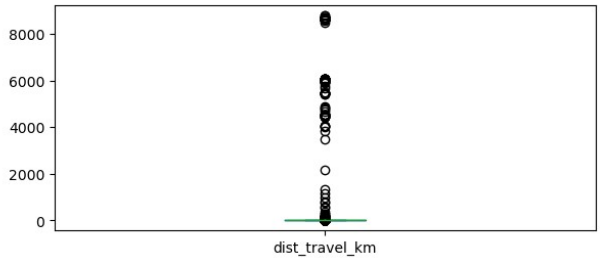

hour


day


month


year


dayofweek


dist_travel_km

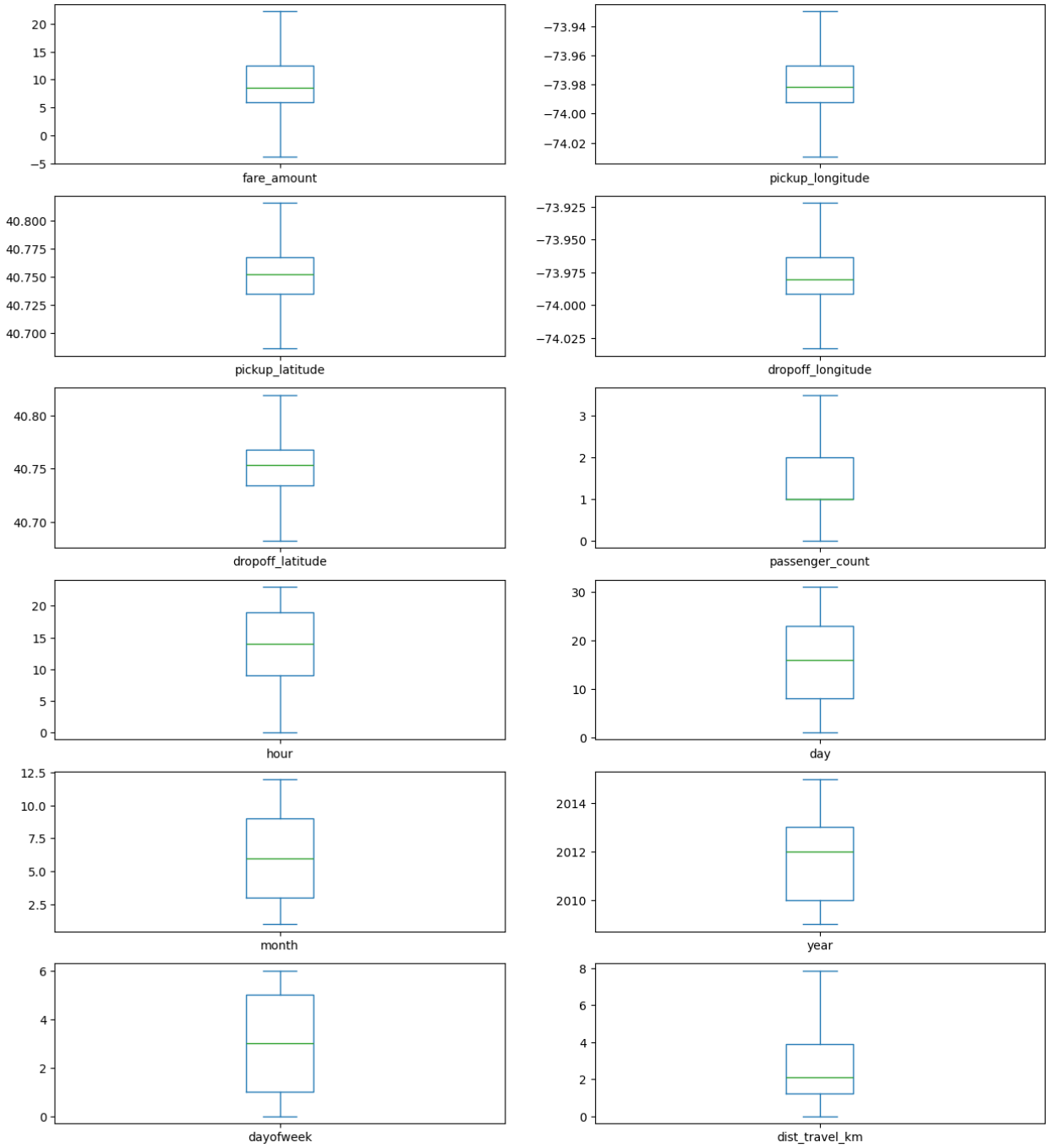
```python
# Using the Inter Quartile Range to fill the values
def remove_outlier(df1, col):
    Q1 = df1[col].quantile(0.25)
    Q3 = df1[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_whisker = Q1 - 1.5*IQR
    upper_whisker = Q3 + 1.5*IQR
    df[col] = np.clip(df1[col], lower_whisker, upper_whisker)
    return df1

def treat_outliers_all(df1, col_list):
    for c in col_list:
        df1 = remove_outlier(df, c)
    return df1

# df.iloc[:, 0 ::] is a method for selecting rows and columns in a
Pandas DataFrame. In this case, : in both positions selects all rows
and all columns in the DataFrame df. Essentially, it's selecting the
entire DataFrame.
df = treat_outliers_all(df, df.iloc[:, 0 : : ])
```

```
C:\Users\Shubham\AppData\Local\Temp\ipykernel_15420\3594884473.py:8:
SettingWithCopyWarning:
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/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df[col] = np.clip(df1[col], lower_whisker, upper_whisker)
C:\Users\Shubham\AppData\Local\Temp\ipykernel_15420\3594884473.py:8:
SettingWithCopyWarning:
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/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df[col] = np.clip(df1[col], lower_whisker, upper_whisker)
C:\Users\Shubham\AppData\Local\Temp\ipykernel_15420\3594884473.py:8:
SettingWithCopyWarning:
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/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df[col] = np.clip(df1[col], lower_whisker, upper_whisker)
C:\Users\Shubham\AppData\Local\Temp\ipykernel_15420\3594884473.py:8:
SettingWithCopyWarning:
```

```
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/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df[col] = np.clip(df1[col], lower_whisker, upper_whisker)
C:\Users\Shubham\AppData\Local\Temp\ipykernel_15420\3594884473.py:8:
SettingWithCopyWarning:
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/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df[col] = np.clip(df1[col], lower_whisker, upper_whisker)
C:\Users\Shubham\AppData\Local\Temp\ipykernel_15420\3594884473.py:8:
SettingWithCopyWarning:
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/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df[col] = np.clip(df1[col], lower_whisker, upper_whisker)
C:\Users\Shubham\AppData\Local\Temp\ipykernel_15420\3594884473.py:8:
SettingWithCopyWarning:
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/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df[col] = np.clip(df1[col], lower_whisker, upper_whisker)
C:\Users\Shubham\AppData\Local\Temp\ipykernel_15420\3594884473.py:8:
SettingWithCopyWarning:
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/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df[col] = np.clip(df1[col], lower_whisker, upper_whisker)
C:\Users\Shubham\AppData\Local\Temp\ipykernel_15420\3594884473.py:8:
SettingWithCopyWarning:
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/user_guide/indexing.html#
```

```
returning-a-view-versus-a-copy
  df[col] = np.clip(df1[col], lower_whisker, upper_whisker)
C:\Users\Shubham\AppData\Local\Temp\ipykernel_15420\3594884473.py:8:
SettingWithCopyWarning:
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/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df[col] = np.clip(df1[col], lower_whisker, upper_whisker)
C:\Users\Shubham\AppData\Local\Temp\ipykernel_15420\3594884473.py:8:
SettingWithCopyWarning:
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/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df[col] = np.clip(df1[col], lower_whisker, upper_whisker)
C:\Users\Shubham\AppData\Local\Temp\ipykernel_15420\3594884473.py:8:
SettingWithCopyWarning:
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/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df[col] = np.clip(df1[col], lower_whisker, upper_whisker)
```

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




```
# 3. Check the correlation
# .corr(): This is a Pandas method that is applied to the DataFrame
df. When you call df.corr(), it calculates the pairwise correlation
coefficients between all pairs of numerical columns in the DataFrame.
# Common correlation coefficients calculated by df.corr() include
Pearson's correlation coefficient, Kendall's tau, and Spearman's rank
correlation coefficient. By default, df.corr() computes Pearson's
correlation coefficient, which is the most widely used.
```

```
corr = df.corr()
corr
```

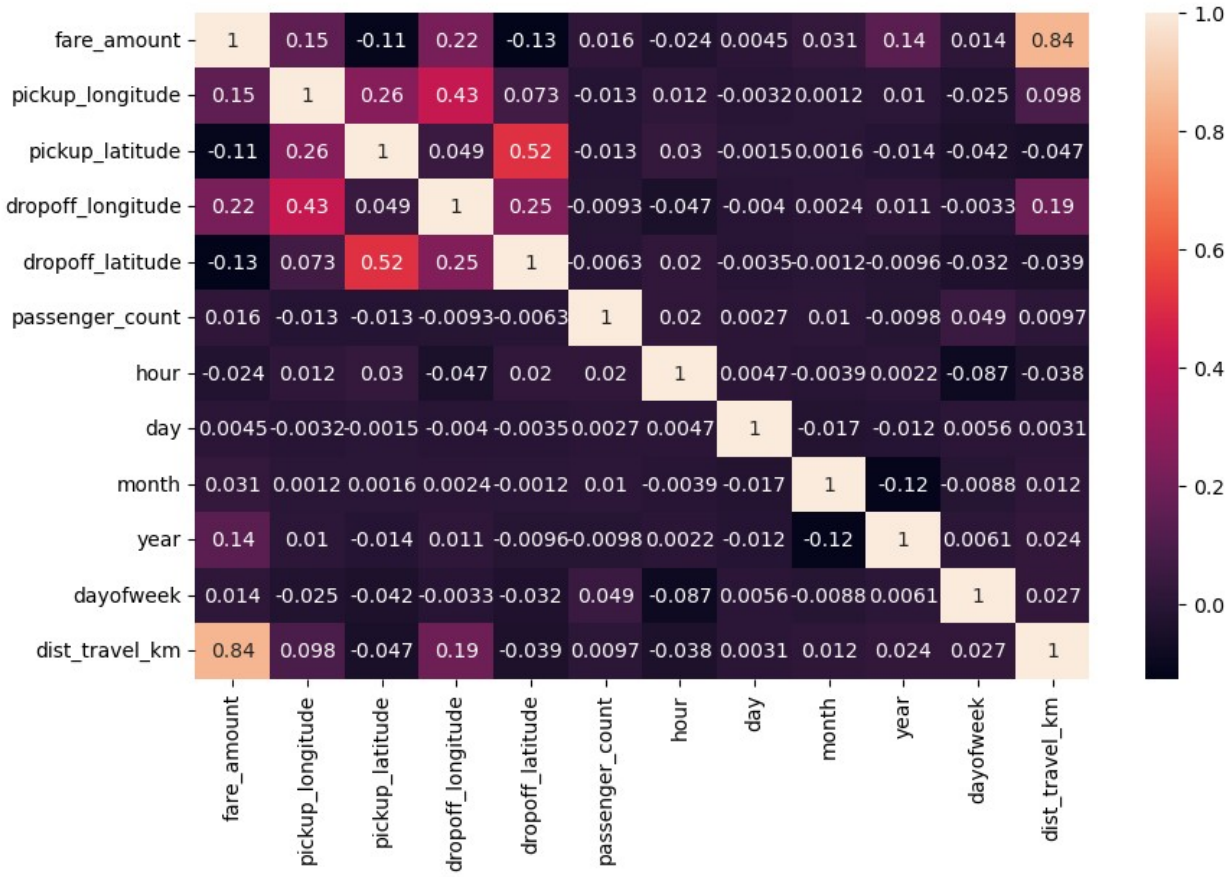
```
                   fare_amount  pickup_longitude  pickup_latitude  \
fare_amount           1.000000          0.154079        -0.110864
pickup_longitude      0.154079          1.000000         0.259502
pickup_latitude      -0.110864          0.259502         1.000000
dropoff_longitude     0.218684          0.425591         0.048902
dropoff_latitude     -0.125911          0.073380         0.515782
passenger_count       0.015775         -0.013190        -0.012842
hour                 -0.023615          0.011564         0.029738
day                   0.004536         -0.003215        -0.001526
month                 0.030816          0.001183         0.001550
year                  0.141282          0.010197        -0.014236
dayofweek             0.013650         -0.024633        -0.042328
dist_travel_km        0.844483          0.098242        -0.046853

                   dropoff_longitude  dropoff_latitude
passenger_count  \
fare_amount                 0.218684         -0.125911
0.015775
pickup_longitude            0.425591          0.073380        -
0.013190
pickup_latitude             0.048902          0.515782        -
0.012842
dropoff_longitude           1.000000          0.245751        -
0.009303
dropoff_latitude            0.245751          1.000000        -
0.006333
passenger_count            -0.009303         -0.006333
1.000000
hour                       -0.046590          0.019818
0.020266
day                        -0.004026         -0.003465
0.002701
month                       0.002408         -0.001192
0.010369
year                        0.011347         -0.009606        -
0.009752
dayofweek                  -0.003313         -0.031950
0.048550
dist_travel_km              0.186844         -0.038874
0.009700

                        hour       day     month      year
dayofweek  \
fare_amount       -0.023615  0.004536  0.030816  0.141282   0.013650

pickup_longitude   0.011564 -0.003215  0.001183  0.010197  -0.024633
```

```
pickup_latitude    0.029738 -0.001526  0.001550 -0.014236  -0.042328

dropoff_longitude -0.046590 -0.004026  0.002408  0.011347  -0.003313

dropoff_latitude   0.019818 -0.003465 -0.001192 -0.009606  -0.031950

passenger_count    0.020266  0.002701  0.010369 -0.009752   0.048550

hour               1.000000  0.004660 -0.003920  0.002154  -0.086934

day                0.004660  1.000000 -0.017350 -0.012172   0.005626

month             -0.003920 -0.017350  1.000000 -0.115857  -0.008783

year               0.002154 -0.012172 -0.115857  1.000000   0.006112

dayofweek         -0.086934  0.005626 -0.008783  0.006112   1.000000

dist_travel_km    -0.038370  0.003075  0.011644  0.024288   0.027034
```

```
                   dist_travel_km
fare_amount              0.844483
pickup_longitude         0.098242
pickup_latitude         -0.046853
dropoff_longitude        0.186844
dropoff_latitude        -0.038874
passenger_count          0.009700
hour                    -0.038370
day                      0.003075
month                    0.011644
year                     0.024288
dayofweek                0.027034
dist_travel_km           1.000000
```

```
fig, axis = plt.subplots(figsize = (10, 6))
sns.heatmap(df.corr(), annot = True) # Correlation Heatmap (Light
values means highly correlated)
```

```
<Axes: >
```




```
# 4. Implement linear regression and random forest regression models
# Dividing the dataset into feature and target values
df_x = df[['passenger_count', 'hour', 'day', 'month', 'year',
'dayofweek', 'dist_travel_km']]
df_y = df['fare_amount']

# Dividing the dataset into training and testing dataset
x_train, x_test, y_train, y_test = train_test_split(df_x, df_y,
test_size = 0.2)

from sklearn.linear_model import LinearRegression

# Initialize the linear regression model
reg = LinearRegression()

# Train the model with our training data
reg.fit(x_train, y_train)

LinearRegression()

y_pred_lin = reg.predict(x_test)
print(y_pred_lin)
```

```
[7.81922718 6.2107946  5.35920711 ... 9.43175974 8.88152522
8.27646638]

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 = rf.predict(x_test)
print(y_pred_rf)

# 5. Evaluate the models and compare their respective scores like R2,
RMSE, etc
cols = ['Model', 'RMSE', 'R-Squared']

# Create a empty dataframe of the colums
# Columns: specifies the columns to be selected
result_tabulation = pd.DataFrame(columns = cols)

import numpy as np
import pandas as pd
from sklearn import metrics
from sklearn.metrics import r2_score

# Initialize an empty list to store results
result_tabulation = []

# Linear Regression metrics
reg_RMSE = np.sqrt(metrics.mean_squared_error(y_test, y_pred_lin))
reg_squared = r2_score(y_test, y_pred_lin)

# Append Linear Regression results
result_tabulation.append({'Model': "Linear Regression", 'RMSE':
reg_RMSE, 'R-Squared': reg_squared})

# Random Forest metrics
rf_RMSE = np.sqrt(metrics.mean_squared_error(y_test, y_pred_rf))
rf_squared = r2_score(y_test, y_pred_rf)

# Append Random Forest results
result_tabulation.append({'Model': "Random Forest", 'RMSE': rf_RMSE,
'R-Squared': rf_squared})

# Convert the list of dictionaries into a DataFrame
result_tabulation_df = pd.DataFrame(result_tabulation)

# Print the result table
print(result_tabulation_df)
```