In [1]:
```
import pandas as pd
import os
import urllib
from sklearn.decomposition import PCA
from sklearn.decomposition import KernelPCA
import numpy as np
from scipy.signal import periodogram
import statsmodels.api as sm
import dask.dataframe as dd

from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.tsa.arima_model import ARIMA
from matplotlib import pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import PredefinedSplit
import warnings
warnings.filterwarnings('ignore')
```

Theoretic part

Multiple choice questions: please select all that applies and explain your answer.

Question 1 (Autocorrelation).

The autocorrelation plot of the daily time-series has local peaks at t=7,14,21,28 etc.. How would you interpret that?

A. The time-series reaches its maximum on the days 7,14,21,28...

B. The time-series reaches its minimum on the days 7,14,21,28...

C. The time-series is likely to have a periodic pattern with a period of 7 days

D. The time-series is likely to have 7 periods per day

E. The appropriate AR model for the time-series should have at least 7 terms.

Your answer:

C

Since the period has 7 cycles, strongly suggest that the periodic pattern with 1 week.

Question 2 (Stationarity).

Which of the following time-series models are always stationary:

A. Linear trend

B. MA(1) model

C. White noise

D. Random walk

E. ARMA(1,2) model

F. ARIMA(1,1,1) model

Your answer:

B

the q parameter in the MA(q) used for the lagged errors, the more errors the model has, the more complex the prediction has. So when q is as small as 1, the noise of the model is lower, and causing it station.

C

White noise has constant mean and variance, so it is station.

Question 3 (PCA).

Which of the following statements regarding the model dimensionality reduction through Principal Component Analysis (PCA) are true:

A. Leading principal components of the features are the most efficient for modeling the output variable.

B. Principal components of the standardized features are uncorrelated and this way less exposed to multicollinearity.

C. The model using principal components of the features can't overfit.

D. Feature selection based on the principal components of the features is often more efficient in preventing overfitting comparered the feature selection over the original features.

E. Principal components are harder to interpret compared to the original features making the PCA regresssion model less interpretable compared to the regression model using original features.

Your answer:

B

One of the key benefits of PCA is that it transforms the original features into a new set of variables (the principal components) that are orthogonal (uncorrelated) to each other. This characteristic is particularly useful in addressing multicollinearity among the features in a dataset.

E

Since principal components are linear combinations of the original features, they often do not have a direct or easily interpretable relationship to the original variables. This makes models using PCA less interpretable in terms of the original features.

Question 4 (MapReduce).

What is true about MapReduce:

A. MapReduce is a Python module enabling parallel computing

B. Using MapReduce approach makes the code more suitable for parallel computing.

C. MapReduce code always runs faster compared to the code using more traditional approaches, like loops or list comprehensions.

D. MapReduce code will always efficiently run on multiple cores of your CPU or multiple machines within your cluster if available.

E. Multiprocessing and PySpark efficient alternatives to MapReduce.

Your answer:

B

The MapReduce programming model is designed to process large data sets across a distributed cluster in a parallel manner. It breaks down the job into smaller chunks (Map phase) and then consolidates the results (Reduce phase), making the code inherently suitable for parallel computing.

Practice part: Taxi ridership from JFK to other taxi zones prediction.

This project is an example of applying PCA to predict hourly yellow taxi ridership at the taxi zone level. Modeling taxi ridership at a fine spatial and temporal granularity is challenging due to the low signal-to-noise ratio and high dimensionality. In this case, dimension reduction essential in feature engineering. This project has five steps: data downloading, data preprocessing, baseline modeling, feature engineering, and RandomForest modeling.

Let's start with data downloading.

1. Data downloading

Design a function to download yellow taxi data from 2017-01-01 to 2018-12-31 at https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page.

In [2]:
```
dataDir = 'taxidata'
if os.path.exists(dataDir):
    pass
else:
    os.mkdir(dataDir)
Years = [2017,2018]
Months = range(1,13)
VehicleTypes = ['yellow']

def getUrl(cabtype,year,month):
    baseUrl = 'https://d37ci6vzurychx.cloudfront.net/trip-data/'

    month = str(month).zfill(2)
    fileName = '%s_tripdata_%s-%s.parquet'%(cabtype,year,month)

    return baseUrl + fileName, fileName
```

In [3]:
```
for year in Years:
    for month in Months:
        for cabtype in VehicleTypes:
            url, fileName = getUrl(cabtype,year,month)

            print("Downloading: "+str(fileName))

            if fileName in os.listdir(dataDir):
                print("file exists")
                continue

            filePath = os.path.join(dataDir, fileName)
            try:
                urllib.request.urlretrieve(url, filePath)
            except:
                # if fails remove the incomplete file
                os.remove(filePath)
                try:
```

```
                # start again after a delay of 2 min
                time.sleep(60*2)
                urllib.request.urlretrieve(url, filePath)
            except:
                print("Download this file later!")
                pass
```

```
Downloading: yellow_tripdata_2017-01.parquet
file exists
Downloading: yellow_tripdata_2017-02.parquet
file exists
Downloading: yellow_tripdata_2017-03.parquet
file exists
Downloading: yellow_tripdata_2017-04.parquet
file exists
Downloading: yellow_tripdata_2017-05.parquet
file exists
Downloading: yellow_tripdata_2017-06.parquet
file exists
Downloading: yellow_tripdata_2017-07.parquet
file exists
Downloading: yellow_tripdata_2017-08.parquet
file exists
Downloading: yellow_tripdata_2017-09.parquet
file exists
Downloading: yellow_tripdata_2017-10.parquet
file exists
Downloading: yellow_tripdata_2017-11.parquet
file exists
Downloading: yellow_tripdata_2017-12.parquet
file exists
Downloading: yellow_tripdata_2018-01.parquet
file exists
Downloading: yellow_tripdata_2018-02.parquet
file exists
Downloading: yellow_tripdata_2018-03.parquet
file exists
Downloading: yellow_tripdata_2018-04.parquet
file exists
Downloading: yellow_tripdata_2018-05.parquet
file exists
Downloading: yellow_tripdata_2018-06.parquet
file exists
Downloading: yellow_tripdata_2018-07.parquet
file exists
Downloading: yellow_tripdata_2018-08.parquet
file exists
Downloading: yellow_tripdata_2018-09.parquet
file exists
Downloading: yellow_tripdata_2018-10.parquet
file exists
Downloading: yellow_tripdata_2018-11.parquet
file exists
Downloading: yellow_tripdata_2018-12.parquet
file exists
```

2. Data Preprocessing

Use dask to aggregate all months' records into one dataframe, and aggregate dataset by date and hour to get the ridership from JFK to each taxi zone each hour. The expected output has columns: date, hour, drop-off location 1, drop-off location 2, etc.

Hint:

1. JFK taxi zone id is 132.
2. time column should be the pickup time, and ridership is passenger count.
3. Try read_parquet("*.parquet") to read all parquet file in a folder
4. files in 2017 and 2018 have different columns; apply argument usecols to select desired columns.
5. using .compute() function to convert processed dask dataframe to pandas dataframe for further modeling.

2.1 Data loading

In [4]:
```
#your answer here
directory = 'taxidata'
df_list = []
desired_columns = ['tpep_pickup_datetime', 'tpep_dropoff_datetime','passenger_c
for filename in os.listdir(directory):
    if filename.endswith(".parquet"):
        filepath = os.path.join(directory, filename)
        df = pd.read_parquet(filepath, columns=desired_columns)
        df_list.append(df)
```

In [5]:
```
df_list
```

Out[5]:
```
[        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
 0        2018-09-01 00:01:35   2018-09-01 00:09:48              2.0
 1        2018-09-01 00:22:22   2018-09-01 00:28:55              1.0
 2        2018-09-01 00:38:10   2018-09-01 00:44:42              1.0
 3        2018-09-01 00:46:36   2018-09-01 00:54:49              1.0
 4        2018-09-01 00:59:46   2018-09-01 01:02:41              1.0
 ...                      ...                   ...              ...
 8049089  2018-09-30 23:04:00   2018-09-30 23:37:00              NaN
 8049090  2018-09-30 23:02:00   2018-09-30 23:31:00              NaN
 8049091  2018-09-30 23:21:00   2018-09-30 23:35:00              NaN
 8049092  2018-09-30 23:14:00   2018-09-30 23:36:00              NaN
 8049093  2018-09-30 23:17:00   2018-09-30 23:29:00              NaN

          trip_distance  PULocationID  DOLocationID
 0                 1.50           161           107
 1                 1.00           233           100
 2                 1.00           164           163
 3                 1.90            48           140
 4                 0.60           262           263
 ...                ...           ...           ...
 8049089          18.81           177            69
 8049090           6.27           225           226
 8049091           3.93           170           238
 8049092           8.04           226           159
 8049093           3.63             7           230

 [8049094 rows x 6 columns],
          tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
 0         2017-03-01 00:38:16   2017-03-01 00:59:21                1
 1         2017-03-01 00:25:01   2017-03-01 00:31:36                1
 2         2017-03-01 00:43:48   2017-03-01 00:44:17                1
 3         2017-03-01 00:47:17   2017-03-01 00:47:33                1
 4         2017-03-01 00:13:37   2017-03-01 00:13:46                1
 ...                       ...                   ...              ...
 10295436  2017-03-31 23:38:53   2017-04-01 00:05:17                1
 10295437  2017-03-31 23:06:11   2017-03-31 23:12:46                1
 10295438  2017-03-31 23:14:00   2017-03-31 23:14:11                1
 10295439  2017-03-31 23:22:28   2017-03-31 23:39:02                1
 10295440  2017-03-31 23:40:59   2017-03-31 23:55:37                1

           trip_distance  PULocationID  DOLocationID
 0                 10.50           231            42
 1                  1.40           239           262
 2                  0.00           145           145
 3                  0.00           145           145
 4                  0.00           145           145
 ...                 ...           ...           ...
 10295436           5.80           237           116
 10295437           1.05            48            50
 10295438           0.01            50           143
 10295439           1.44            48           100
 10295440           3.12           100           236

 [10295441 rows x 6 columns],
          tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
 0        2018-10-01 00:23:34   2018-10-01 00:44:50              1.0
 1        2018-10-01 00:40:05   2018-10-01 01:01:56              1.0
 2        2018-10-01 00:05:35   2018-10-01 00:19:38              1.0
 3        2018-10-01 00:42:56   2018-10-01 00:49:00              1.0
 4        2018-10-01 00:19:14   2018-10-01 00:31:54              1.0
```

```
...                    ...                   ...              ...
8834515  2018-10-31 23:36:00   2018-10-31 23:55:00              NaN
8834516  2018-10-31 23:09:00   2018-10-31 23:38:00              NaN
8834517  2018-10-31 23:35:16   2018-11-01 00:07:43              NaN
8834518  2018-10-31 23:01:00   2018-10-31 23:34:00              NaN
8834519  2018-10-31 23:07:53   2018-10-31 23:37:20              NaN

         trip_distance  PULocationID  DOLocationID
0                 6.20            68             7
1                12.60           132             9
2                 6.10            50           244
3                 1.30           151           239
4                 2.60           233           143
...                ...           ...           ...
8834515           9.38           235           246
8834516          13.29           225            53
8834517          19.58            79           222
8834518          11.43           158            69
8834519          12.84           140           102

[8834520 rows x 6 columns],
        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0        2017-02-01 00:19:20   2017-02-01 00:25:56                1
1        2017-02-01 00:19:55   2017-02-01 00:33:06                1
2        2017-02-01 00:01:15   2017-02-01 00:09:03                2
3        2017-02-01 00:06:36   2017-02-01 00:14:50                5
4        2017-02-01 00:07:53   2017-02-01 00:14:36                1
...                      ...                   ...              ...
9169770  2017-02-28 23:35:01   2017-02-28 23:56:25                2
9169771  2017-02-28 23:58:55   2017-03-01 00:17:41                1
9169772  2017-02-28 23:26:41   2017-02-28 23:26:45                1
9169773  2017-02-28 23:27:31   2017-02-28 23:27:33                1
9169774  2017-02-28 23:28:08   2017-02-28 23:28:10                1

         trip_distance  PULocationID  DOLocationID
0                 2.90            75           162
1                 4.90           246           166
2                 1.50           237           170
3                 1.51           137           236
4                 1.40           112           112
...                ...           ...           ...
9169770           5.40           114            43
9169771           5.80            43            79
9169772           0.00           264           193
9169773           0.00           264           193
9169774           0.00           264           193

[9169775 rows x 6 columns],
        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0        2017-12-01 00:12:00   2017-12-01 00:12:51                1
1        2017-12-01 00:13:37   2017-12-01 00:13:47                1
2        2017-12-01 00:14:15   2017-12-01 00:15:05                1
3        2017-12-01 00:15:33   2017-12-01 00:15:37                1
4        2017-12-01 00:50:03   2017-12-01 00:53:35                1
...                      ...                   ...              ...
9508496  2017-12-31 23:31:57   2017-12-31 23:55:59                1
9508497  2017-12-31 22:53:16   2017-12-31 22:57:03                1
9508498  2017-12-31 23:05:23   2017-12-31 23:14:30                1
9508499  2017-12-31 23:22:29   2017-12-31 23:31:47                1
9508500  2017-12-31 23:34:49   2017-12-31 23:40:29                1
```

```
         trip_distance  PULocationID  DOLocationID
0                 0.00           226           226
1                 0.00           226           226
2                 0.00           226           226
3                 0.00           226           226
4                 0.00           145           145
...                ...           ...           ...
9508496           9.80           186           127
9508497           0.63            74            74
9508498           3.05           236            42
9508499           1.76            74           152
9508500           0.91           152            41

[9508501 rows x 6 columns],
        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0        2018-11-01 00:51:36   2018-11-01 00:52:36              1.0
1        2018-11-01 00:07:47   2018-11-01 00:21:43              1.0
2        2018-11-01 00:24:27   2018-11-01 00:34:29              1.0
3        2018-11-01 00:35:27   2018-11-01 00:47:02              1.0
4        2018-11-01 00:16:46   2018-11-01 00:22:50              1.0
...                      ...                   ...              ...
8155444  2018-11-30 23:28:00   2018-12-01 00:00:00              NaN
8155445  2018-11-30 23:07:00   2018-11-30 23:51:00              NaN
8155446  2018-11-30 23:07:00   2018-11-30 23:27:00              NaN
8155447  2018-11-30 23:36:00   2018-12-01 00:09:00              NaN
8155448  2018-11-30 23:17:53   2018-11-30 23:45:06              NaN

         trip_distance  PULocationID  DOLocationID
0                 0.00           145           145
1                 2.30           142           164
2                 1.80           164            48
3                 2.30            48           107
4                 1.00           163           170
...                ...           ...           ...
8155444           9.95           151           196
8155445          15.80           107           191
8155446           9.40           226            42
8155447          10.52            68           169
8155448           7.75            25            21

[8155449 rows x 6 columns],
        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0        2018-01-01 00:21:05   2018-01-01 00:24:23                1
1        2018-01-01 00:44:55   2018-01-01 01:03:05                1
2        2018-01-01 00:08:26   2018-01-01 00:14:21                2
3        2018-01-01 00:20:22   2018-01-01 00:52:51                1
4        2018-01-01 00:09:18   2018-01-01 00:27:06                2
...                      ...                   ...              ...
8760682  2018-01-31 23:21:35   2018-01-31 23:34:20                2
8760683  2018-01-31 23:35:51   2018-01-31 23:38:57                1
8760684  2018-01-31 23:28:00   2018-01-31 23:37:09                1
8760685  2018-01-31 23:24:40   2018-01-31 23:25:28                1
8760686  2018-01-31 23:28:16   2018-01-31 23:28:38                1

         trip_distance  PULocationID  DOLocationID
0                 0.50            41            24
1                 2.70           239           140
2                 0.80           262           141
3                10.20           140           257
```

```
4                2.50          246           239
...               ...          ...           ...
8760682          2.80          158           163
8760683          0.60          163           162
8760684          2.95           74            69
8760685          0.00            7           193
8760686          0.00            7           193

[8760687 rows x 6 columns],
        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0        2018-08-01 00:44:35   2018-08-01 01:03:22              1.0   
1        2018-08-01 00:02:19   2018-08-01 00:02:31              1.0   
2        2018-08-01 00:13:25   2018-08-01 00:24:40              1.0   
3        2018-08-01 00:10:37   2018-08-01 00:49:10              1.0   
4        2018-08-01 00:02:18   2018-08-01 00:07:32              2.0   
...                      ...                   ...              ...   
7855035  2018-08-31 23:19:00   2018-08-31 23:38:00              NaN   
7855036  2018-08-31 23:46:00   2018-08-31 23:46:00              NaN   
7855037  2018-08-31 23:09:51   2018-08-31 23:14:33              NaN   
7855038  2018-08-31 23:30:00   2018-08-31 23:35:00              NaN   
7855039  2018-08-31 23:05:00   2018-08-31 23:31:00              NaN   

         trip_distance  PULocationID  DOLocationID  
0                 5.60           238            79  
1                 0.00           145           145  
2                 2.90           138             7  
3                 8.40           231             7  
4                 0.70            79           148  
...                ...           ...           ...  
7855035          10.25           183            75  
7855036           0.00           232           232  
7855037           0.54            79            79  
7855038           0.91            48           163  
7855039          12.60           246           213  

[7855040 rows x 6 columns],
        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0        2017-10-01 00:01:50   2017-10-01 00:14:13                1   
1        2017-10-01 00:02:43   2017-10-01 00:08:35                2   
2        2017-10-01 00:12:08   2017-10-01 00:25:49                3   
3        2017-10-01 00:00:25   2017-10-01 00:11:24                1   
4        2017-10-01 00:15:30   2017-10-01 00:25:11                1   
...                      ...                   ...              ...   
9768667  2017-10-31 23:05:37   2017-10-31 23:14:07                1   
9768668  2017-10-31 23:45:12   2017-10-31 23:50:51                1   
9768669  2017-10-31 23:33:17   2017-11-01 00:02:51                2   
9768670  2017-10-31 23:25:32   2017-10-31 23:31:22                1   
9768671  2017-10-31 23:34:55   2017-11-01 00:16:14                1   

         trip_distance  PULocationID  DOLocationID  
0                 2.00           142           233  
1                 2.30           142           166  
2                 2.80           151           262  
3                 1.97           100           229  
4                 2.17           141           142  
...                ...           ...           ...  
9768667           4.67            13           257  
9768668           1.54           181            33  
9768669           6.20           100           255  
9768670           0.82           264           264  
```

```
9768671           5.52          264           264

[9768672 rows x 6 columns],
        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0        2018-03-01 00:01:34   2018-03-01 00:01:43                1
1        2018-03-01 00:14:34   2018-03-01 00:28:13                1
2        2018-03-01 00:51:25   2018-03-01 00:59:54                1
3        2018-03-01 00:00:01   2018-03-01 00:00:17                1
4        2018-03-01 00:55:10   2018-03-01 00:56:36                1
...                      ...                   ...              ...
9431284  2018-03-31 23:34:47   2018-03-31 23:55:17                5
9431285  2018-03-31 23:02:38   2018-03-31 23:13:10                6
9431286  2018-03-31 23:15:58   2018-03-31 23:30:29                6
9431287  2018-03-31 23:05:37   2018-03-31 23:18:31                2
9431288  2018-03-31 23:37:11   2018-03-31 23:56:53                1

         trip_distance  PULocationID  DOLocationID
0                 0.00           145           145
1                 3.30           151           244
2                 2.70           238           152
3                 0.00           145           145
4                 3.70           145           145
...                ...           ...           ...
9431284           4.11           186           263
9431285           1.50           100           107
9431286           2.07           107           170
9431287           1.60           163           164
9431288           3.90           144           161

[9431289 rows x 6 columns],
        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0        2017-09-01 00:17:17   2017-09-01 00:18:49                1
1        2017-09-01 00:22:08   2017-09-01 00:25:22                2
2        2017-09-01 00:30:43   2017-09-01 00:33:47                1
3        2017-09-01 00:37:57   2017-09-01 00:42:24                1
4        2017-09-01 00:15:56   2017-09-01 00:28:28                1
...                      ...                   ...              ...
8945416  2017-09-30 23:03:35   2017-09-30 23:14:56                1
8945417  2017-09-30 23:15:53   2017-09-30 23:20:33                1
8945418  2017-09-30 23:05:20   2017-09-30 23:23:22                1
8945419  2017-09-30 23:28:39   2017-10-01 00:24:39                1
8945420  2017-09-30 23:36:10   2017-10-01 00:03:27                1

         trip_distance  PULocationID  DOLocationID
0                 0.40           161           161
1                 0.90           164           234
2                 0.52           193           193
3                 1.50           246            50
4                 1.30           143           143
...                ...           ...           ...
8945416           2.22            48           141
8945417           1.50           141           263
8945418           4.98           164            87
8945419          13.98            87           258
8945420           5.26           148           140

[8945421 rows x 6 columns],
        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0        2017-08-01 00:29:45   2017-08-01 00:29:51                1
1        2017-08-01 00:06:34   2017-08-01 00:19:23                2
```

```
2        2017-08-01 00:07:46   2017-08-01 00:14:51                 1
3        2017-08-01 00:16:03   2017-08-01 00:35:36                 1
4        2017-08-01 00:21:14   2017-08-01 00:22:40                 1
...                      ...                   ...               ...
8422148  2017-08-31 23:39:07   2017-09-01 00:08:22                 1
8422149  2017-08-31 23:04:26   2017-08-31 23:44:33                 1
8422150  2017-08-31 23:09:33   2017-08-31 23:15:32                 2
8422151  2017-08-31 23:37:20   2017-08-31 23:53:32                 1
8422152  2017-08-31 23:55:14   2017-09-01 00:02:22                 1

         trip_distance  PULocationID  DOLocationID
0                 0.00           145           145
1                 1.20           140           237
2                 3.80           223            70
3                 3.30            79           230
4                 0.50           162           237
...                ...           ...           ...
8422148          10.10           138            61
8422149          10.72            50           196
8422150           1.70            41           116
8422151           3.89            68           263
8422152           1.91           263           233

[8422153 rows x 6 columns],
        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0        2017-01-01 00:32:05   2017-01-01 00:37:48                 1
1        2017-01-01 00:43:25   2017-01-01 00:47:42                 2
2        2017-01-01 00:49:10   2017-01-01 00:53:53                 2
3        2017-01-01 00:36:42   2017-01-01 00:41:09                 1
4        2017-01-01 00:07:41   2017-01-01 00:18:16                 1
...                      ...                   ...               ...
9710815  2017-01-31 23:04:11   2017-01-31 23:10:56                 2
9710816  2017-01-31 23:32:46   2017-01-31 23:40:14                 4
9710817  2017-01-31 23:23:22   2017-01-31 23:38:38                 1
9710818  2017-01-31 23:48:10   2017-01-31 23:57:15                 1
9710819  2017-01-31 23:57:58   2017-02-01 00:11:16                 1

         trip_distance  PULocationID  DOLocationID
0                 1.20           140           236
1                 0.70           237           140
2                 0.80           140           237
3                 1.10            41            42
4                 3.00            48           263
...                ...           ...           ...
9710815           1.04           148            45
9710816           1.60           264           264
9710817           3.65           148            48
9710818           0.93           249            79
9710819           2.71            79           256

[9710820 rows x 6 columns],
        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0        2017-11-01 00:01:48   2017-11-01 00:03:47                 1
1        2017-11-01 00:18:22   2017-11-01 00:40:32                 1
2        2017-11-01 00:01:58   2017-11-01 00:15:57                 1
3        2017-11-01 00:18:53   2017-11-01 00:25:23                 1
4        2017-11-01 00:28:56   2017-11-01 00:38:22                 1
...                      ...                   ...               ...
9284798  2017-11-30 23:27:24   2017-11-30 23:48:15                 1
9284799  2017-11-30 23:59:05   2017-11-30 23:59:14                 1
```

```
9284800  2017-11-30 23:17:20   2017-11-30 23:39:33                1
9284801  2017-11-30 22:52:40   2017-11-30 23:27:26                1
9284802  2017-11-30 23:33:39   2017-11-30 23:42:54                1

         trip_distance  PULocationID  DOLocationID
0                 0.40           151           151
1                 4.80           142           144
2                 3.70           151           140
3                 1.90           140           233
4                 2.00           229            50
...                ...           ...           ...
9284798           3.16            90           141
9284799           0.00            25            25
9284800          10.28           161           127
9284801           5.80           113           181
9284802           1.68           181            25

[9284803 rows x 6 columns],
        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0        2018-12-01 00:28:22   2018-12-01 00:44:07              2.0
1        2018-12-01 00:52:29   2018-12-01 01:11:37              3.0
2        2018-12-01 00:12:52   2018-12-01 00:36:23              1.0
3        2018-12-01 00:35:08   2018-12-01 00:43:11              1.0
4        2018-12-01 00:21:54   2018-12-01 01:15:13              1.0
...                      ...                   ...              ...
8195670  2018-12-31 23:25:00   2018-12-31 23:40:00              NaN
8195671  2018-12-31 23:04:00   2018-12-31 23:30:00              NaN
8195672  2018-12-31 23:02:00   2018-12-31 23:35:00              NaN
8195673  2018-12-31 23:17:00   2018-12-31 23:43:00              NaN
8195674  2018-12-31 23:01:26   2018-12-31 23:43:42              NaN

         trip_distance  PULocationID  DOLocationID
0                 2.50           148           234
1                 2.30           170           144
2                 0.00           113           193
3                 3.90            95            92
4                12.80           163           228
...                ...           ...           ...
8195670           9.32            50           265
8195671           9.06           243           182
8195672          11.70           236           254
8195673           7.82           146            78
8195674           9.92            95            48

[8195675 rows x 6 columns],
        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0        2018-02-01 00:01:58   2018-02-01 00:04:03                1
1        2018-02-01 00:56:48   2018-02-01 00:57:42                1
2        2018-02-01 00:04:42   2018-02-01 00:19:32                1
3        2018-02-01 00:38:10   2018-02-01 00:40:16                1
4        2018-02-01 00:43:03   2018-02-01 00:59:26                2
...                      ...                   ...              ...
8492814  2018-02-28 23:21:56   2018-02-28 23:47:43                6
8492815  2018-02-28 23:54:07   2018-03-01 00:24:56                6
8492816  2018-02-28 23:17:42   2018-02-28 23:53:11                1
8492817  2018-03-01 13:39:17   2018-03-01 13:49:08                2
8492818  2018-02-28 23:03:18   2018-02-28 23:03:24                1

         trip_distance  PULocationID  DOLocationID
0                 0.00           145           145
```

```
1                 2.90          145           145
2                 5.80          236           119
3                 0.30           82            82
4                 2.60           82             7
...                ...          ...           ...
8492814           9.61          138           163
8492815           4.20          230           230
8492816           0.00          193           193
8492817           0.00          264           264
8492818           0.00          264           193

[8492819 rows x 6 columns],
        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0        2018-07-01 00:28:09   2018-07-01 00:28:51              1.0   
1        2018-07-01 00:29:27   2018-07-01 00:30:17              1.0   
2        2018-07-01 00:04:19   2018-07-01 00:08:29              2.0   
3        2018-07-01 00:14:26   2018-07-01 00:36:35              1.0   
4        2018-07-01 00:41:56   2018-07-01 00:50:54              1.0   
...                      ...                   ...              ...   
7851138  2018-07-31 19:02:00   2018-07-31 19:33:00              NaN   
7851139  2018-07-31 19:12:00   2018-07-31 19:13:00              NaN   
7851140  2018-07-31 20:57:00   2018-07-31 21:47:00              NaN   
7851141  2018-07-31 22:50:00   2018-07-31 22:50:00              NaN   
7851142  2018-07-31 23:02:00   2018-07-31 23:20:00              NaN   

         trip_distance  PULocationID  DOLocationID  
0                 5.30           145           145  
1                 5.30           145           145  
2                 0.70           211           144  
3                 4.80           144           142  
4                 1.80           142           141  
...                ...           ...           ...  
7851138          17.15           158           265  
7851139           0.00           132           132  
7851140          33.54            48           265  
7851141           0.00            79            79  
7851142           4.68           249           265  

[7851143 rows x 6 columns],
         tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0         2017-04-01 00:51:24   2017-04-01 00:51:49                1   
1         2017-04-01 00:41:17   2017-04-01 00:55:36                1   
2         2017-04-01 00:23:31   2017-04-01 00:35:17                1   
3         2017-04-01 00:05:31   2017-04-01 00:35:30                1   
4         2017-04-01 00:38:13   2017-04-01 00:54:48                2   
...                       ...                   ...              ...   
10047130  2017-04-30 23:42:01   2017-04-30 23:56:39                1   
10047131  2017-04-30 23:18:36   2017-04-30 23:24:55                1   
10047132  2017-04-30 23:33:25   2017-04-30 23:39:30                2   
10047133  2017-04-30 23:54:53   2017-04-30 23:56:16                1   
10047134  2017-04-30 23:58:57   2017-05-01 00:00:45                1   

          trip_distance  PULocationID  DOLocationID  
0                  0.00           145           145  
1                  3.40           249            87  
2                  2.50           163           263  
3                  4.50           163             7  
4                  4.90             7           262  
...                 ...           ...           ...  
10047130           3.39           186           263  
```

```
10047131          1.50          68          107
10047132          1.20         170          229
10047133          0.00           7            7
10047134          0.00         193          193

[10047135 rows x 6 columns],
        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0        2018-06-01 00:15:40   2018-06-01 00:16:46                1
1        2018-06-01 00:04:18   2018-06-01 00:09:18                1
2        2018-06-01 00:14:39   2018-06-01 00:29:46                1
3        2018-06-01 00:51:25   2018-06-01 00:51:29                3
4        2018-06-01 00:55:06   2018-06-01 00:55:10                1
...                      ...                   ...              ...
8714662  2018-06-30 23:09:48   2018-06-30 23:21:09                1
8714663  2018-06-30 23:39:24   2018-06-30 23:45:02                3
8714664  2018-06-30 23:24:13   2018-06-30 23:34:31                2
8714665  2018-06-30 23:46:15   2018-06-30 23:57:42                1
8714666  2018-06-30 23:43:59   2018-06-30 23:43:59                1

         trip_distance  PULocationID  DOLocationID
0                 0.00           145           145
1                 1.00           230           161
2                 3.30           100           263
3                 0.00           145           145
4                 0.00           145           145
...                ...           ...           ...
8714662           5.00           138            92
8714663           0.70           230           230
8714664           1.88           166           239
8714665           2.40           142            68
8714666           0.00           264             7

[8714667 rows x 6 columns],
         tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0         2017-05-01 00:02:54   2017-05-01 00:03:10                1
1         2017-05-01 00:03:52   2017-05-01 00:04:25                1
2         2017-05-01 00:00:10   2017-05-01 00:12:51                1
3         2017-05-01 00:48:58   2017-05-01 01:32:01                1
4         2017-05-01 00:27:37   2017-05-01 00:39:40                1
...                       ...                   ...              ...
10102122  2017-05-31 23:19:41   2017-05-31 23:31:11                1
10102123  2017-05-31 23:57:39   2017-05-31 23:59:06                1
10102124  2017-05-31 23:28:39   2017-05-31 23:44:40                1
10102125  2017-05-31 23:45:13   2017-05-31 23:49:07                1
10102126  2017-05-31 23:52:58   2017-06-01 00:07:15                1

          trip_distance  PULocationID  DOLocationID
0                  0.00           260           260
1                  0.00           145           145
2                  2.50            68            79
3                  7.20           230           160
4                  2.70           138           223
...                 ...           ...           ...
10102122           2.10           234            48
10102123           0.30           186           164
10102124           2.70           261           231
10102125           0.61           141           140
10102126           2.38           141           170

[10102127 rows x 6 columns],
```

```
        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0        2018-04-01 00:22:20   2018-04-01 00:22:26                1
1        2018-04-01 00:47:37   2018-04-01 01:08:42                1
2        2018-04-01 00:02:13   2018-04-01 00:17:52                2
3        2018-04-01 00:46:49   2018-04-01 00:52:05                1
4        2018-04-01 00:19:04   2018-04-01 00:19:09                1
...                      ...                   ...              ...
9306211  2018-04-30 23:15:20   2018-04-30 23:32:58                1
9306212  2018-04-30 23:02:02   2018-04-30 23:03:37                5
9306213  2018-04-30 23:38:18   2018-04-30 23:44:57                1
9306214  2018-04-30 23:07:08   2018-04-30 23:23:04                1
9306215  2018-04-30 23:26:50   2018-04-30 23:44:54                1

         trip_distance  PULocationID  DOLocationID
0                 0.00           145           145
1                 6.70           152            90
2                 4.10           239           158
3                 0.70            90           249
4                 0.00           145           145
...                ...           ...           ...
9306211           3.60           148           112
9306212           0.01           151           151
9306213           1.62           186           125
9306214           6.36           261           162
9306215           7.17           162            65

[9306216 rows x 6 columns],
        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0        2017-07-01 00:06:25   2017-07-01 00:10:50                1
1        2017-07-01 00:20:04   2017-07-01 00:21:38                2
2        2017-07-01 00:44:10   2017-07-01 00:59:29                1
3        2017-07-01 00:07:33   2017-07-01 00:31:30                1
4        2017-07-01 00:01:17   2017-07-01 00:16:18                1
...                      ...                   ...              ...
8588481  2017-07-31 23:57:40   2017-08-01 00:07:49                1
8588482  2017-07-31 23:04:41   2017-07-31 23:10:18                1
8588483  2017-07-31 23:35:47   2017-07-31 23:46:01                3
8588484  2017-07-31 23:50:49   2017-08-01 00:00:59                3
8588485  2017-07-31 23:44:14   2017-07-31 23:50:21                1

         trip_distance  PULocationID  DOLocationID
0                 1.20           249            90
1                 0.20           249           158
2                 4.30           100            45
3                 8.30           138           162
4                 1.90           107           158
...                ...           ...           ...
8588481           2.21           170           142
8588482           1.41           262           141
8588483           1.66           113           148
8588484           3.02           148            80
8588485           1.40           162           107

[8588486 rows x 6 columns],
        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0        2018-05-01 00:13:56   2018-05-01 00:22:46                1
1        2018-05-01 00:23:26   2018-05-01 00:29:56                1
2        2018-05-01 00:36:23   2018-05-01 00:48:26                2
3        2018-05-01 00:26:12   2018-05-01 00:27:05                1
4        2018-05-01 00:29:51   2018-05-01 00:30:02                1
```

```
...                       ...                     ...                ...
9224783  2018-05-31 23:25:13   2018-05-31 23:27:46                 2
9224784  2018-05-31 23:15:24   2018-05-31 23:19:39                 1
9224785  2018-05-31 23:46:26   2018-05-31 23:52:55                 1
9224786  2018-05-31 23:59:33   2018-06-01 00:11:58                 1
9224787  2018-05-31 23:27:40   2018-06-01 00:14:39                 1

         trip_distance  PULocationID  DOLocationID
0                 1.60           230            50
1                 1.70           263           239
2                 2.60           239           152
3                 0.00           145           145
4                 0.00           145           145
...                ...           ...           ...
9224783           0.70           140           262
9224784           0.91           263           237
9224785           1.29           230           237
9224786           2.42           163            90
9224787           9.10           142           158

[9224788 rows x 6 columns],
        tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0        2017-06-01 00:02:36   2017-06-01 00:10:02                1
1        2017-06-01 00:14:14   2017-06-01 00:16:50                1
2        2017-06-01 00:47:11   2017-06-01 00:57:47                1
3        2017-06-01 00:14:38   2017-06-01 00:19:49                1
4        2017-06-01 00:03:41   2017-06-01 00:57:09                1
...                      ...                   ...              ...
9656988  2017-06-30 23:07:15   2017-06-30 23:33:18                1
9656989  2017-06-30 23:35:12   2017-06-30 23:44:46                1
9656990  2017-06-30 23:59:15   2017-07-01 00:09:35                1
9656991  2017-06-30 23:12:25   2017-06-30 23:25:50                1
9656992  2017-06-30 23:34:04   2017-06-30 23:47:26                3

         trip_distance  PULocationID  DOLocationID
0                 1.80           161           263
1                 0.80           237           237
2                 1.70            48           233
3                 1.10           246           249
4                14.80           166            61
...                ...           ...           ...
9656988           6.76           232           238
9656989           4.51           238           244
9656990           2.49            42           238
9656991           2.50           161           141
9656992           3.60           264            41

[9656993 rows x 6 columns]]
```

In [6]:

```
combined_df = pd.concat(df_list, ignore_index=True)
combined_df
```

Out[6]:

	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	PUL
0	2018-09-01 00:01:35	2018-09-01 00:09:48	2.0	1.50	
1	2018-09-01 00:22:22	2018-09-01 00:28:55	1.0	1.00	
2	2018-09-01 00:38:10	2018-09-01 00:44:42	1.0	1.00	
3	2018-09-01 00:46:36	2018-09-01 00:54:49	1.0	1.90	
4	2018-09-01 00:59:46	2018-09-01 01:02:41	1.0	0.60	
...	...	...	...	...	
216371709	2017-06-30 23:07:15	2017-06-30 23:33:18	1.0	6.76	
216371710	2017-06-30 23:35:12	2017-06-30 23:44:46	1.0	4.51	
216371711	2017-06-30 23:59:15	2017-07-01 00:09:35	1.0	2.49	
216371712	2017-06-30 23:12:25	2017-06-30 23:25:50	1.0	2.50	
216371713	2017-06-30 23:34:04	2017-06-30 23:47:26	3.0	3.60	

216371714 rows × 6 columns

In [7]:
```
jfk_df = combined_df.loc[combined_df['PULocationID']==132]
jfk_df.size
```

Out[7]: 31822620

In [8]:
```
jfk_df['hours']=jfk_df['tpep_pickup_datetime'].dt.hour
jfk_df['days']=jfk_df['tpep_pickup_datetime'].dt.dayofyear
jfk_df['year']=jfk_df['tpep_pickup_datetime'].dt.year
jfk_df['date'] = jfk_df['tpep_pickup_datetime'].dt.date
```

In [9]:
```
jfk_df
```

Out[9]:

	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	PUL
25	2018-09-01 00:05:17	2018-09-01 00:28:45	5.0	16.55	
26	2018-09-01 00:41:28	2018-09-01 01:15:36	1.0	11.99	
130	2018-09-01 00:16:45	2018-09-01 00:23:58	1.0	4.00	
143	2018-09-01 00:15:27	2018-09-01 01:11:59	3.0	17.30	
144	2018-09-01 00:09:51	2018-09-01 00:33:27	1.0	8.47	
...	...	...	...	...	
216371511	2017-06-30 23:16:33	2017-07-01 00:14:36	1.0	29.20	
216371525	2017-06-30 23:44:28	2017-07-01 00:22:47	1.0	13.60	
216371577	2017-06-30 23:40:53	2017-07-01 00:09:37	1.0	15.33	
216371587	2017-06-30 23:47:12	2017-07-01 00:26:56	2.0	17.10	
216371689	2017-06-30 23:43:51	2017-06-30 23:57:27	3.0	4.90	

5303770 rows × 10 columns

2.2 Sanity check

Then, we need to do some basic sanity checks. It is possible that in a particular hour, completely dispatched no yellow taxis from JFK. Check does each day has 24-hour records and add missing records back to the dataframe. The final output should have 17520 rows ($365 \times 2 \times 24$)

In [10]:

```
#your answer here
grouped = jfk_df.loc[(jfk_df['year']== 2017) | (jfk_df['year'] == 2018)]

grouped = grouped.groupby(['date', 'hours', 'DOLocationID']).agg({'passenger_co

grouped
```

Out[10]:

date	hours	DOLocationID	passenger_count
2017-01-01	0	4	1.0
		7	2.0
		10	7.0
		12	1.0
		13	13.0
...	...	...	...
2018-12-31	23	260	1.0
		261	1.0
		262	8.0
		263	13.0
		265	25.0

1848572 rows × 1 columns

In [11]:

```
pivot = grouped.pivot_table(values='passenger_count',
                                      index=['date', 'hours'],
                                      columns='DOLocationID',
                                      aggfunc='sum',fill_value=0)
pivot=pivot.reset_index()
```

In [12]:

```
pivot['date'] = pd.to_datetime(pivot['date'])
```

In [13]:

```
date_range = pd.date_range(start="2017-01-01", end="2019-01-01", freq='H')

# Create a DataFrame with the date range
df = pd.DataFrame({
    'date': date_range,
    'hours': date_range.hour,  # Correctly extract hour from each datetime

})
# edit the formate to match with the result dataframe
df['date'] = pd.to_datetime(df['date'])
df['hours'] = df['hours'].astype('int64')
df['date'] = df['date'].dt.date
df['date'] = pd.to_datetime(df['date'])
for col in range(1, 266):  # Python range stops before the second number, so us
    df[f'DO{col}'] = 0.0
df
```

Out[13]:

	date	hours	DO1	DO2	DO3	DO4	DO5	DO6	DO7	DO8	...	DO256	DO257	DO25
0	2017-01-01	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.
1	2017-01-01	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.
2	2017-01-01	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.
3	2017-01-01	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.
4	2017-01-01	4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
17516	2018-12-31	20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.
17517	2018-12-31	21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.
17518	2018-12-31	22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.
17519	2018-12-31	23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.
17520	2019-01-01	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.

17521 rows × 267 columns

In [14]:

```
merged_df = pd.merge(df, pivot, on=['date', 'hours'], how='left')
merged_df = merged_df.fillna(0)
merged_df
```

Out[14]:

	date	hours	DO1	DO2	DO3	DO4	DO5	DO6	DO7	DO8	...	256	257	258	259
0	2017-01-01	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	3.0	0.0	0.0
1	2017-01-01	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	6.0	0.0	0.0	0.0
2	2017-01-01	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	2.0	0.0
3	2017-01-01	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	2.0	0.0
4	2017-01-01	4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17516	2018-12-31	20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	6.0	4.0	0.0	0.0
17517	2018-12-31	21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	4.0	5.0	5.0	0.0
17518	2018-12-31	22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	11.0	0.0	1.0
17519	2018-12-31	23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	11.0	0.0	0.0
17520	2019-01-01	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0

17521 rows × 530 columns

In [15]:

```
merged_df = merged_df.drop(merged_df.index[-1])
merged_df = merged_df.drop(merged_df.columns[2:267], axis=1)
merged_df
```

Out[15]:

	date	hours	1	2	3	4	5	6	7	8	...	256	257	258	259	260	261
0	2017-01-01	0	0.0	0.0	0.0	1.0	0.0	0.0	2.0	0.0	...	0.0	3.0	0.0	0.0	1.0	0.0
1	2017-01-01	1	0.0	0.0	0.0	2.0	0.0	0.0	5.0	0.0	...	6.0	0.0	0.0	0.0	0.0	5.0
2	2017-01-01	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	2.0	0.0	0.0	0.0
3	2017-01-01	3	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	1.0	0.0	2.0	0.0	0.0	0.0
4	2017-01-01	4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17515	2018-12-31	19	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	...	3.0	1.0	5.0	0.0	2.0	5.0
17516	2018-12-31	20	1.0	0.0	0.0	0.0	0.0	0.0	7.0	0.0	...	6.0	4.0	0.0	0.0	8.0	1.0
17517	2018-12-31	21	0.0	0.0	0.0	2.0	0.0	0.0	9.0	0.0	...	4.0	5.0	5.0	0.0	1.0	1.0
17518	2018-12-31	22	0.0	0.0	2.0	2.0	0.0	0.0	2.0	0.0	...	1.0	11.0	0.0	1.0	0.0	5.0
17519	2018-12-31	23	5.0	0.0	0.0	7.0	0.0	0.0	6.0	0.0	...	0.0	11.0	0.0	0.0	1.0	1.0

17520 rows × 265 columns

3. Time-series exploratory analysis

Apply exploratory analysis over the daily aggregated dataset at first.

3.1 aggregate the ridership from each dropoff location and sum it to get daily records.

In [16]:

```
#your answer here
# Assuming 'result' is your DataFrame and it has a column named 'date'
daily = merged_df.groupby(by=merged_df['date']).sum()
daily = daily.drop(columns='hours')
daily.reset_index()
```

Out[16]:

	date	1	2	3	4	5	6	7	8	9	...	256	257	258	259	260	2
0	2017-01-01	15.0	1.0	0.0	72.0	0.0	4.0	136.0	0.0	27.0	...	127.0	55.0	38.0	10.0	36.0	3
1	2017-01-02	13.0	0.0	4.0	74.0	0.0	0.0	192.0	0.0	11.0	...	135.0	35.0	41.0	8.0	50.0	6
2	2017-01-03	26.0	0.0	9.0	47.0	0.0	0.0	171.0	2.0	17.0	...	123.0	50.0	45.0	16.0	55.0	6
3	2017-01-04	21.0	0.0	3.0	43.0	0.0	5.0	142.0	0.0	3.0	...	125.0	36.0	32.0	9.0	12.0	6
4	2017-01-05	8.0	1.0	3.0	58.0	0.0	0.0	83.0	0.0	8.0	...	115.0	8.0	23.0	4.0	38.0	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
725	2018-12-27	8.0	0.0	2.0	33.0	0.0	0.0	116.0	1.0	5.0	...	109.0	21.0	14.0	5.0	45.0	10
726	2018-12-28	5.0	0.0	4.0	58.0	0.0	1.0	67.0	0.0	4.0	...	89.0	37.0	28.0	6.0	16.0	8
727	2018-12-29	8.0	0.0	3.0	21.0	0.0	1.0	99.0	0.0	11.0	...	109.0	33.0	27.0	6.0	26.0	6
728	2018-12-30	12.0	0.0	2.0	35.0	1.0	2.0	128.0	0.0	15.0	...	87.0	56.0	25.0	5.0	32.0	6
729	2018-12-31	10.0	0.0	2.0	29.0	0.0	0.0	106.0	0.0	17.0	...	66.0	49.0	26.0	13.0	36.0	3

730 rows × 264 columns

3.2 Period detection and report the strongest period length on the 2017 data.

Hint: using periodogram or acf plot.

In [17]:
```
grouped_2017 = merged_df.loc[merged_df['date'].dt.year==2017]
grouped_2017
```

Out[17]:

	date	hours	1	2	3	4	5	6	7	8	...	256	257	258	259	260	261
0	2017-01-01	0	0.0	0.0	0.0	1.0	0.0	0.0	2.0	0.0	...	0.0	3.0	0.0	0.0	1.0	0.0
1	2017-01-01	1	0.0	0.0	0.0	2.0	0.0	0.0	5.0	0.0	...	6.0	0.0	0.0	0.0	0.0	5.0
2	2017-01-01	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	2.0	0.0	0.0	0.0
3	2017-01-01	3	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	1.0	0.0	2.0	0.0	0.0	0.0
4	2017-01-01	4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8755	2017-12-31	19	1.0	0.0	1.0	0.0	0.0	0.0	3.0	0.0	...	6.0	2.0	0.0	0.0	1.0	5.0
8756	2017-12-31	20	0.0	0.0	4.0	2.0	0.0	0.0	14.0	0.0	...	5.0	4.0	3.0	0.0	9.0	1.0
8757	2017-12-31	21	1.0	0.0	0.0	1.0	1.0	0.0	2.0	0.0	...	3.0	2.0	3.0	0.0	0.0	4.0
8758	2017-12-31	22	1.0	0.0	0.0	1.0	0.0	0.0	7.0	0.0	...	2.0	0.0	0.0	0.0	0.0	2.0
8759	2017-12-31	23	0.0	0.0	1.0	1.0	0.0	0.0	6.0	0.0	...	3.0	9.0	0.0	0.0	3.0	2.0

8760 rows × 265 columns

In [18]:

```
d17= grouped_2017.groupby(by=grouped_2017['date']).sum()
d17 = d17.drop(columns='hours')
d17 = d17.reset_index()
d17
```

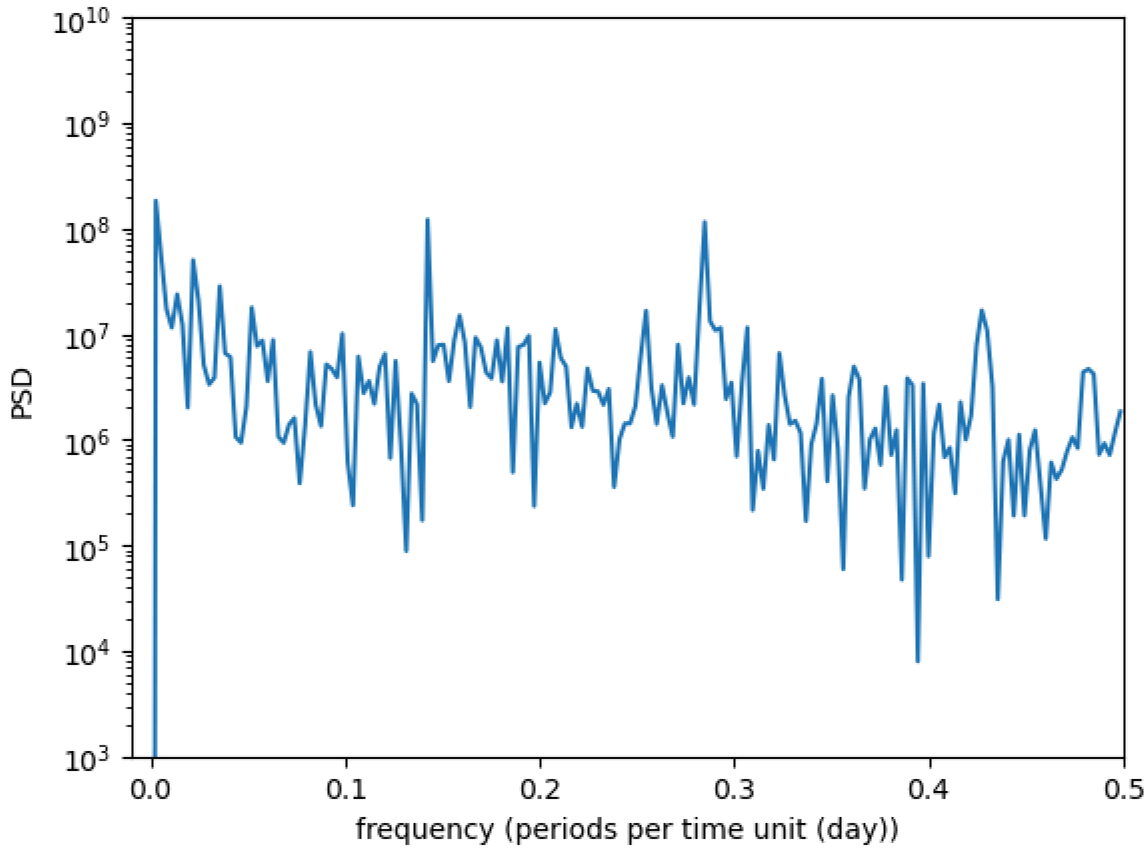
Out[18]:

	date	1	2	3	4	5	6	7	8	9	...	256	257	258	259	260	2
0	2017-01-01	15.0	1.0	0.0	72.0	0.0	4.0	136.0	0.0	27.0	...	127.0	55.0	38.0	10.0	36.0	3
1	2017-01-02	13.0	0.0	4.0	74.0	0.0	0.0	192.0	0.0	11.0	...	135.0	35.0	41.0	8.0	50.0	6
2	2017-01-03	26.0	0.0	9.0	47.0	0.0	0.0	171.0	2.0	17.0	...	123.0	50.0	45.0	16.0	55.0	6
3	2017-01-04	21.0	0.0	3.0	43.0	0.0	5.0	142.0	0.0	3.0	...	125.0	36.0	32.0	9.0	12.0	6
4	2017-01-05	8.0	1.0	3.0	58.0	0.0	0.0	83.0	0.0	8.0	...	115.0	8.0	23.0	4.0	38.0	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
360	2017-12-27	5.0	0.0	4.0	41.0	0.0	1.0	125.0	0.0	13.0	...	101.0	31.0	32.0	4.0	43.0	7
361	2017-12-28	13.0	1.0	3.0	30.0	0.0	0.0	79.0	0.0	5.0	...	89.0	33.0	20.0	9.0	45.0	7
362	2017-12-29	21.0	0.0	2.0	40.0	0.0	4.0	119.0	0.0	7.0	...	132.0	59.0	21.0	4.0	33.0	7
363	2017-12-30	8.0	0.0	10.0	37.0	0.0	0.0	119.0	0.0	5.0	...	103.0	54.0	37.0	6.0	28.0	6
364	2017-12-31	13.0	0.0	12.0	34.0	1.0	0.0	143.0	0.0	6.0	...	65.0	49.0	15.0	2.0	38.0	5

365 rows × 264 columns

In [19]:

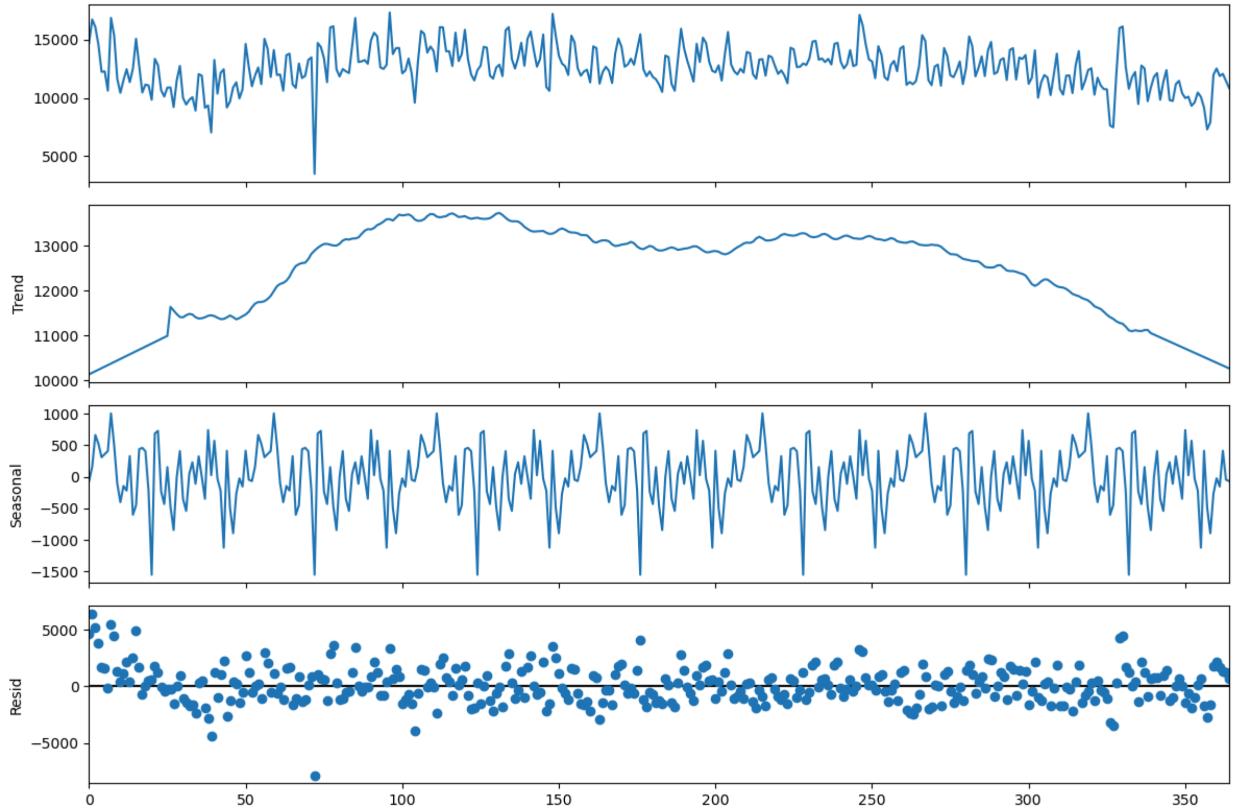
```
#your answer here
import scipy
f, PSD = scipy.signal.periodogram(d17.sum(axis=1))
plt.semilogy(f, PSD)
plt.xlabel('frequency (periods per time unit (day))')
plt.ylabel('PSD')
plt.xlim(-0.01,0.5)
plt.ylim(1e3,1e10)
#filter outputs - periods shorter than 2 years (approx 100 weeks)
PSD = PSD[f>0.01]
f = f[f>0.01]
plt.show()
print('Strongest period length = {}'.format(1/f[np.argmax(PSD)])) #report the
```




```
Strongest period length = 7.019230769230769
```

3.3 Trend, seasonality, noise decomposition (using additive model) on 2017 data, .

In [20]:
```
#your answer here
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = [12, 8]
daily17 = sm.tsa.seasonal_decompose(d17.sum(axis=1), model='additive', period =
fig = daily17.plot()
```



4. Predict the total daily ridership from JFK using ARIMA.

ARIMA is a common method to predict taxi ridership. Before we predict taxi zone level hourly ridership, let's try to predict the aggregated daily ridership using ARIMA.

4.1 Using adfuller test to test the stability of the aggregated dataset. If not stable, apply differencing method until the p-value from adfuller test is smaller than 0.05.

In [21]:
```
#your answer here
from statsmodels.tsa.stattools import adfuller

series=daily.sum(axis=1)
result = adfuller(series)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

    # first order differencing
series=daily.sum(axis=1).diff()
result = adfuller(series.dropna())
print('\n1st order differencing:')
print('  ADF Statistic: %f' % result[0])
print('  p-value: %f' % result[1])

# second order differencing
```

```
series=daily.sum(axis=1).diff().diff()
result = adfuller(series.dropna())
print('\n2nd order differencing:')
print('  ADF Statistic: %f' % result[0])
print('  p-value: %f' % result[1])
```

```
ADF Statistic: -2.153742
p-value: 0.223460
Critical Values:
	1%: -3.440
	5%: -2.866
	10%: -2.569

1st order differencing:
  ADF Statistic: -8.181998
  p-value: 0.000000

2nd order differencing:
  ADF Statistic: -12.229794
  p-value: 0.000000
```

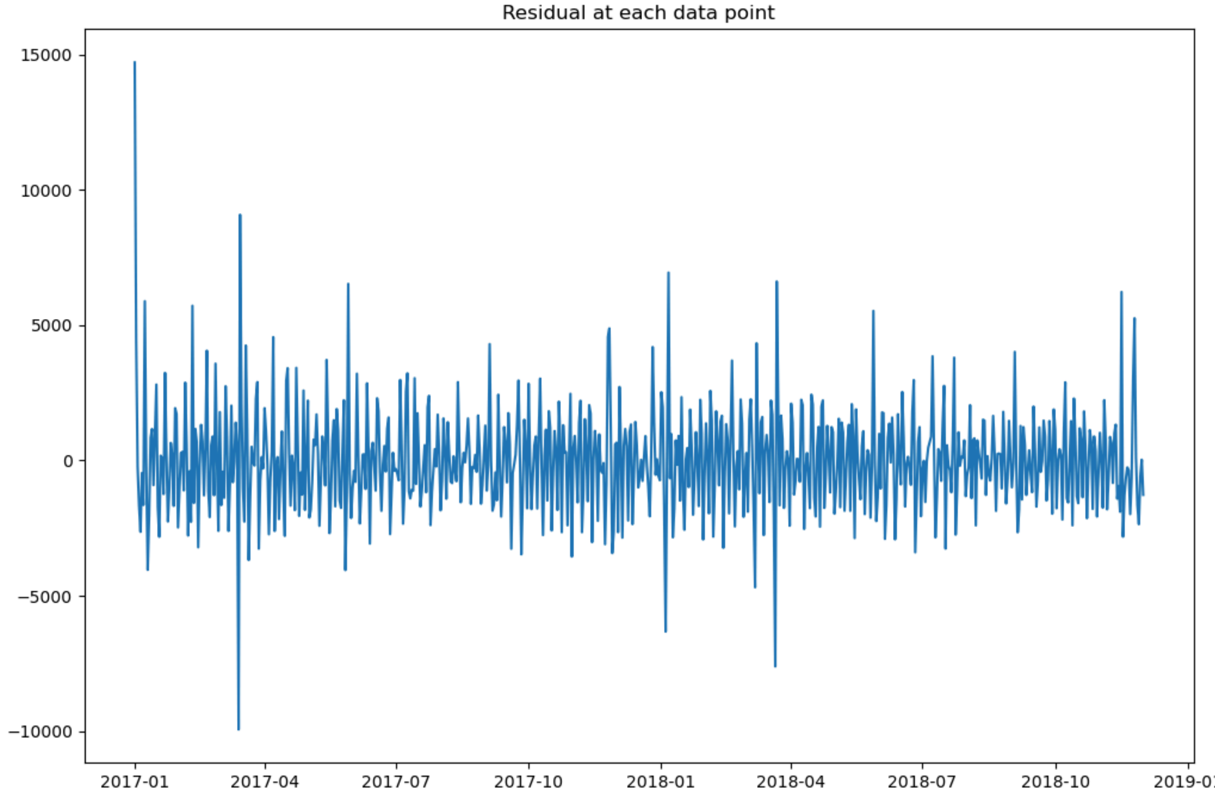
So we need the D=1 to get p-value smaller than 0.05

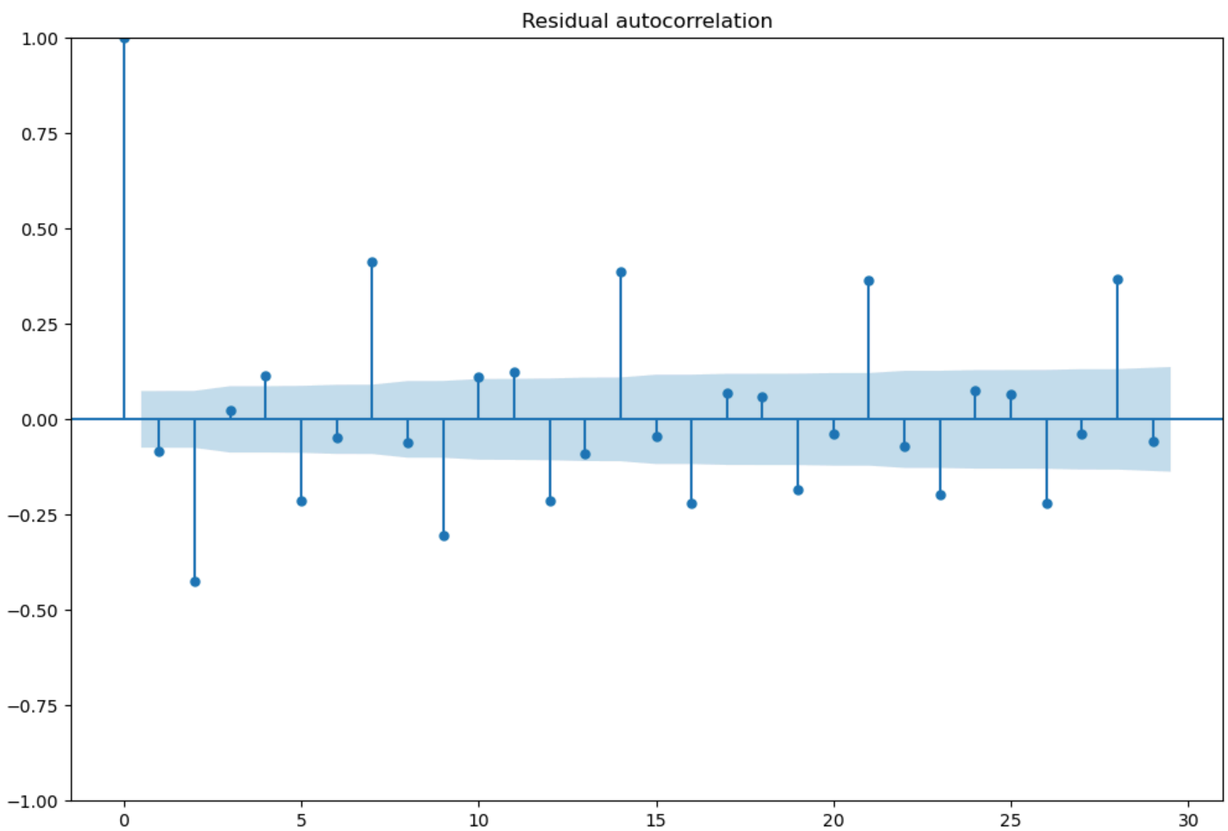
4.2 build an ARIMA model using terms [P=0, D=1, Q=1], training on the first 700 days, forecast on the last 31 days. Print ARIMA model results and plot in-sample and out-of-sample prediction in different colors.

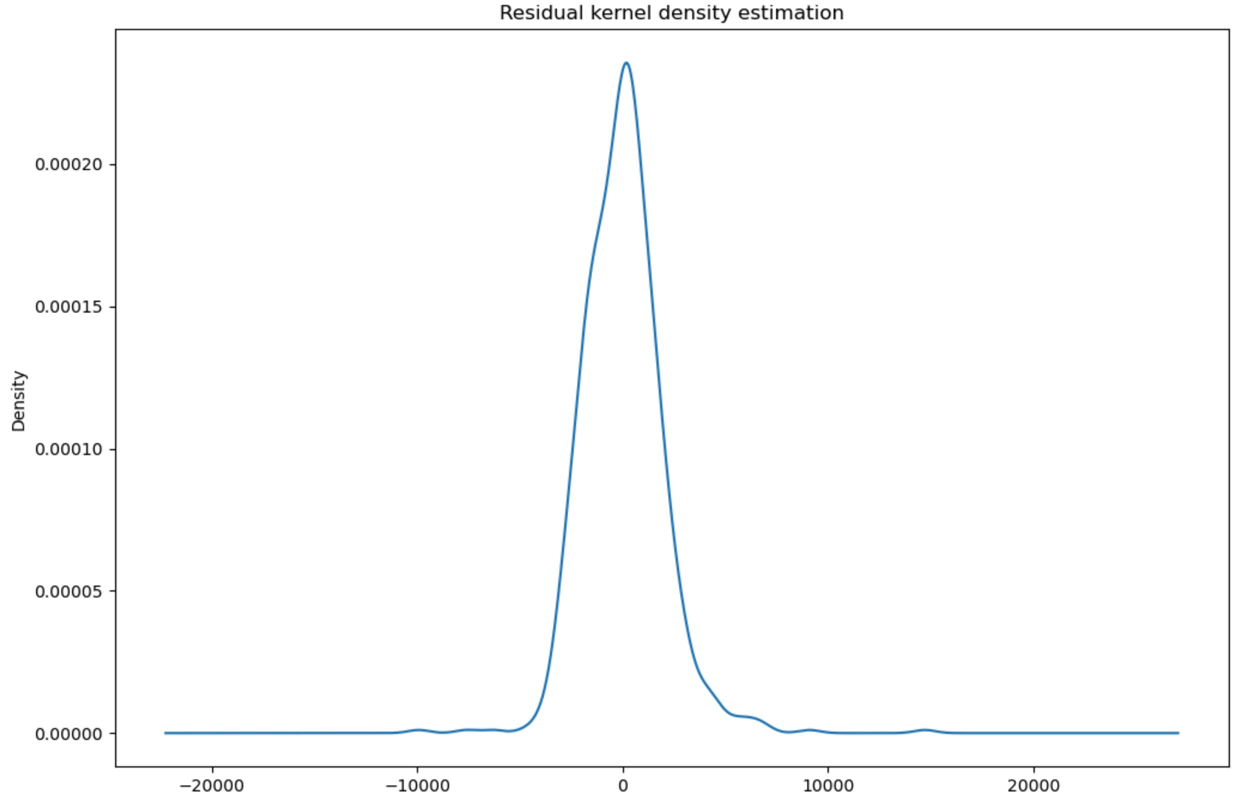
In [22]:
```
P=0
D=1
Q=1

# fit model
N = 700


#your answer here
series = daily.sum(axis=1)#.values

#model = sm.tsa.SARIMAX(series[:N], order=(Q,D,P))
model = sm.tsa.ARIMA(series[:N], order=(Q,D,P))
model_fit = model.fit()
print(model_fit.summary())

# plot residual errors
residuals = pd.DataFrame(model_fit.resid)
plt.plot(residuals)
plt.title('Residual at each data point')
plot_acf(residuals)
plt.title('Residual autocorrelation')
plt.show()
residuals.plot(kind='kde', legend=False)
plt.title('Residual kernel density estimation')
plt.show()
print(residuals.describe())
k2, p = scipy.stats.normaltest(residuals)
alpha = 0.1
```

```
print('p value is ',p[0])

print('null hypothesis: residuals come from a normal distribution')
if p < alpha:
    print("The null hypothesis can be rejected")
else:
    print("The null hypothesis cannot be rejected")

print("Ljung-Box:")
print(sm.stats.acorr_ljungbox(residuals))
```

```
                               SARIMAX Results                                
==============================================================================
Dep. Variable:                      y   No. Observations:                  700
Model:                 ARIMA(1, 1, 0)   Log Likelihood               -6251.498
Date:                Sat, 16 Mar 2024   AIC                          12506.996
Time:                        13:47:06   BIC                          12516.095
Sample:                    01-01-2017   HQIC                         12510.513
                         - 12-01-2018                                         
Covariance Type:                  opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.2136      0.025     -8.587      0.000      -0.262      -0.165
sigma2      3.409e+06   1.23e+05     27.620      0.000    3.17e+06    3.65e+06
==============================================================================
=====
Ljung-Box (L1) (Q):                  10.08   Jarque-Bera (JB):                2
20.75
Prob(Q):                              0.00   Prob(JB):                        
0.00
Heteroskedasticity (H):               0.60   Skew:                            
0.28
Prob(H) (two-sided):                  0.00   Kurtosis:                        
5.70
==============================================================================
=====

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex
-step).
```


Residual at each data point
15000
10000
5000
0
−5000
−10000
2017-01
2017-04
2017-07
2017-10
2018-01
2018-04
2018-07
2018-10
2019-01


Residual autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30

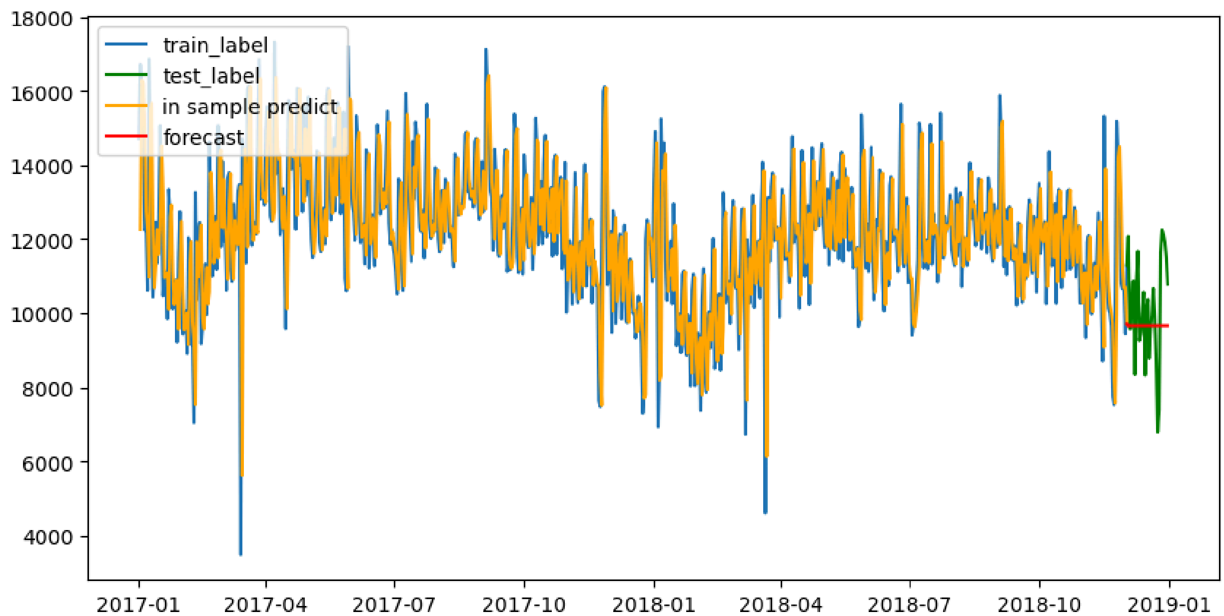



```
                  0
count    700.000000
mean      15.775769
std     1934.681591
min    -9938.582960
25%    -1263.293712
50%        9.805882
75%     1053.734335
max    14710.000000
p value is  3.2878940825612855e-37
null hypothesis: residuals come from a normal distribution
The null hypothesis can be rejected
Ljung-Box:
       lb_stat     lb_pvalue
1     5.048784  2.464323e-02
2   132.110656  2.053693e-29
3   132.512758  1.554284e-28
4   141.578676  1.296108e-29
5   173.663858  1.205690e-35
6   175.299494  3.377372e-35
7   295.979051  4.368333e-60
8   298.632974  8.046372e-60
9   364.719770  4.550781e-73
10  373.662225  3.760209e-74
```

In [23]:
```
# Forecast

fcast = model_fit.forecast(steps=len(series)-N)  # 95% conf
fc = model_fit.get_forecast(steps=len(series)-N).summary_frame()

fc_series = pd.Series(fc['mean'])
#mean_series = pd.Series(fc['mean'], index=range(N,len(series)))
lower_series = pd.Series(fc.mean_ci_lower, index=range(N,len(series)))
upper_series = pd.Series(fc.mean_ci_upper, index=range(N,len(series)))
```

```
plt.rcParams.update({'figure.figsize':(10,5)})
fig, ax = plt.subplots()
ax.plot(daily.index[:N+1],series[:N+1],label='train_label') # train
ax.plot(daily.index[N:],series[N:],color='green',label='test_label') # test
ax.plot(daily.iloc[1:N+1].index,model_fit.predict(start=1,end=N,dynamic=False,
        color='orange',label='in sample predict') # in-sample
ax.plot(fc_series, label='forecast', color='red') # forecast
ax.fill_between(daily.iloc[N:].index, lower_series, upper_series, color='k', a
ax.legend(loc='upper left')
```

Out[23]: <matplotlib.legend.Legend at 0x14e0dddd0>



Taxi zone level prediction

This project aims to predict hourly yellow taxi ridership volume from JFK to each taxi zone. The ARIMA experiment in section 3 forecasts the total ridership amount from JFK. However, based on the reported R^2, this model is not a good fit. ARIMA model has four main shortcomings: 1) they rely heavily on stationarity assumption which does not hold in real-world traffic systems 2) they do not consider spatial and structural dependencies that traffic networks exhibit and forecast each sensor as an individual time series 3) they are unable to model non-linear temporal dynamics 4) they suffer from the curse of dimensionality. Due to the limitation of ARIMA, we need to apply another method to predict taxi zone level ridership.

5. Feature engineering

Our workflow is first standardizing the dataset, then using PCA to compress the dataset. As we predict future ridership, PCA should be learned from historical data (2017) then apply to the following year (2018). Next, add lag features (PCA components) from the past 12 hours and apply a Random Forest regressor to predict each PCA component's values in the next hour. After we had the PCA component prediction, inverse PCA, and inverse standardization to retrieve taxi ridership prediction in its original scale and dimension, in other words, we are predicting the PCA components instead of taxi zone level ridership and then using the inverse PCA method to reconstruct

your answer here

ok

5.1. standardization.

The standardscaler stores information of this standardization process, including the mean and standard deviation values required when converting the prediction back to the raw scale. Split the whole dataset into two parts: 2017 and 2018, standardize each separately.

In [24]:
```
#your answer here
grouped_2017
```

Out[24]:

	date	hours	1	2	3	4	5	6	7	8	...	256	257	258	259	260	261
0	2017-01-01	0	0.0	0.0	0.0	1.0	0.0	0.0	2.0	0.0	...	0.0	3.0	0.0	0.0	1.0	0.0
1	2017-01-01	1	0.0	0.0	0.0	2.0	0.0	0.0	5.0	0.0	...	6.0	0.0	0.0	0.0	0.0	5.0
2	2017-01-01	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	2.0	0.0	0.0	0.0
3	2017-01-01	3	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	1.0	0.0	2.0	0.0	0.0	0.0
4	2017-01-01	4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8755	2017-12-31	19	1.0	0.0	1.0	0.0	0.0	0.0	3.0	0.0	...	6.0	2.0	0.0	0.0	1.0	5.0
8756	2017-12-31	20	0.0	0.0	4.0	2.0	0.0	0.0	14.0	0.0	...	5.0	4.0	3.0	0.0	9.0	1.0
8757	2017-12-31	21	1.0	0.0	0.0	1.0	1.0	0.0	2.0	0.0	...	3.0	2.0	3.0	0.0	0.0	4.0
8758	2017-12-31	22	1.0	0.0	0.0	1.0	0.0	0.0	7.0	0.0	...	2.0	0.0	0.0	0.0	0.0	2.0
8759	2017-12-31	23	0.0	0.0	1.0	1.0	0.0	0.0	6.0	0.0	...	3.0	9.0	0.0	0.0	3.0	2.0

8760 rows × 265 columns

In [25]:
```
grouped_2018 = merged_df.loc[merged_df['date'].dt.year==2018]
grouped_2018
```

Out[25]:

	date	hours	1	2	3	4	5	6	7	8	...	256	257	258	259	260	261
8760	2018-01-01	0	1.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	...	1.0	0.0	0.0	8.0	5.0	0.0
8761	2018-01-01	1	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	...	4.0	2.0	0.0	0.0	2.0	0.0
8762	2018-01-01	2	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	1.0	0.0	0.0	0.0	2.0	0.0
8763	2018-01-01	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	2.0	0.0	0.0	0.0
8764	2018-01-01	4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17515	2018-12-31	19	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	...	3.0	1.0	5.0	0.0	2.0	5.0
17516	2018-12-31	20	1.0	0.0	0.0	0.0	0.0	0.0	7.0	0.0	...	6.0	4.0	0.0	0.0	8.0	1.0
17517	2018-12-31	21	0.0	0.0	0.0	2.0	0.0	0.0	9.0	0.0	...	4.0	5.0	5.0	0.0	1.0	1.0
17518	2018-12-31	22	0.0	0.0	2.0	2.0	0.0	0.0	2.0	0.0	...	1.0	11.0	0.0	1.0	0.0	5.0
17519	2018-12-31	23	5.0	0.0	0.0	7.0	0.0	0.0	6.0	0.0	...	0.0	11.0	0.0	0.0	1.0	1.0

8760 rows × 265 columns

In [26]:

```python
grouped_2017.iloc[:,2:]
```

Out[26]:

	1	2	3	4	5	6	7	8	9	10	...	256	257	258	259	260	261	262
0	0.0	0.0	0.0	1.0	0.0	0.0	2.0	0.0	0.0	7.0	...	0.0	3.0	0.0	0.0	1.0	0.0	3.0
1	0.0	0.0	0.0	2.0	0.0	0.0	5.0	0.0	0.0	8.0	...	6.0	0.0	0.0	0.0	0.0	5.0	7.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	3.0	...	0.0	0.0	2.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	1.0	0.0	2.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8755	1.0	0.0	1.0	0.0	0.0	0.0	3.0	0.0	1.0	8.0	...	6.0	2.0	0.0	0.0	1.0	5.0	3.0
8756	0.0	0.0	4.0	2.0	0.0	0.0	14.0	0.0	0.0	12.0	...	5.0	4.0	3.0	0.0	9.0	1.0	5.0
8757	1.0	0.0	0.0	1.0	1.0	0.0	2.0	0.0	0.0	11.0	...	3.0	2.0	3.0	0.0	0.0	4.0	17.0
8758	1.0	0.0	0.0	1.0	0.0	0.0	7.0	0.0	1.0	24.0	...	2.0	0.0	0.0	0.0	0.0	2.0	11.0
8759	0.0	0.0	1.0	1.0	0.0	0.0	6.0	0.0	0.0	8.0	...	3.0	9.0	0.0	0.0	3.0	2.0	6.0

8760 rows × 263 columns

In [27]:
```
st_17 = StandardScaler().fit_transform(grouped_2017.iloc[:,2:])
st_18= StandardScaler().fit_transform(grouped_2018.iloc[:,2:])
st_18
```

Out[27]:
```
array([[ 0.31612173, -0.0286809 , -0.28145419, ..., -0.33416569,
        -0.68309719, -0.18582689],
       [-0.43454955, -0.0286809 ,  0.99668924, ..., -0.96396162,
        -0.68309719, -0.06607968],
       [-0.43454955, -0.0286809 , -0.28145419, ..., -0.96396162,
        -0.68309719, -1.26355178],
       ...,
       [-0.43454955, -0.0286809 , -0.28145419, ...,  1.76515406,
         1.71888228,  1.37088683],
       [-0.43454955, -0.0286809 ,  2.27483267, ...,  0.92542616,
        -0.68309719,  1.61038125],
       [ 3.31880687, -0.0286809 , -0.28145419, ...,  1.76515406,
        -0.68309719,  1.61038125]])
```

5.2. PCA

Train PCA on the standardized 2018 dataset. Set PCA components as 5, and gamma is None, use kernel 'linear'. Report the mean squared error between the standardized data and reconstructed data. Hint: fit the PCA on 2017 data and apply it to transform 2018 data. (5 pts)

In [28]:
```
#your answer here
kpca = KernelPCA(kernel='linear', gamma=None, n_components=5, fit_inverse_tran
kpca.fit(st_17)
```

Out[28]:
```
KernelPCA
KernelPCA(fit_inverse_transform=True, n_components=5)
```

In [29]:
```
transform_2018 = kpca.transform(st_18)
reconstruct_2018 = kpca.inverse_transform(transform_2018)
```

In [30]:
```
mean_squared_error(st_18, reconstruct_2018)
```

Out[30]:
```
0.8273212208215565
```

5.3 Add lag

add 12 lags of each component (pca_comps=5) (compressed 2018 data only). The expected output should have 65 dimensions. In the further modeling step, we will apply the 60 lag variables to predict the 5 components.

In [31]:
```
# Placeholder for demonstration: Converting the placeholder PCA transformed da
pca_components = pd.DataFrame(transform_2018)

# Add 12 lags for each PCA component
for component in pca_components.columns[:5]:  # Only the first 5 columns are a
    for lag in range(1, 13):
        pca_components[f'{component}_lag_{lag}'] = pca_components[component].sl
```

```
# Show the DataFrame
pca_components
```

Out[31]:

	0	1	2	3	4	0_lag_1	0_lag_2	0_lag_3
0	-2.720164	-4.312002	-0.554225	1.372271	0.879431	NaN	NaN	NaN
1	-4.799956	-3.612092	-1.400683	-0.201888	-2.080059	-2.720164	NaN	NaN
2	-8.787469	-0.853776	0.814226	-0.168481	-0.313817	-4.799956	-2.720164	NaN
3	-8.658432	-1.091070	0.500887	0.579890	0.425699	-8.787469	-4.799956	-2.720164
4	-7.905044	-1.693033	-0.752052	-0.783431	-0.666292	-8.658432	-8.787469	-4.799956
...	...	...	...	...	...	...	...	...
8755	7.454186	-4.539786	0.579454	0.888912	0.146922	-1.390830	5.546593	7.323307
8756	6.112123	-5.409046	3.829655	0.542472	1.232248	7.454186	-1.390830	5.546593
8757	8.619792	-8.059970	3.858406	0.955377	0.822999	6.112123	7.454186	-1.390830
8758	7.935406	-7.058935	0.783922	2.268200	-0.374162	8.619792	6.112123	7.454186
8759	3.239251	-6.747190	-1.182133	1.897086	0.040660	7.935406	8.619792	6.112123

8760 rows × 65 columns

In [32]:
```
pca_components.to_csv('filename.csv', index=False)
```

In [33]:
```
transform_2018
```

Out[33]:
```
array([[-2.72016412, -4.31200215, -0.55422469,  1.37227053,  0.87943082],
       [-4.7999561 , -3.61209236, -1.40068315, -0.2018875 , -2.08005864],
       [-8.7874694 , -0.85377619,  0.81422611, -0.16848084, -0.31381711],
       ...,
       [ 8.61979218, -8.05996965,  3.85840615,  0.95537746,  0.82299888],
       [ 7.9354065 , -7.05893474,  0.78392213,  2.26819964, -0.37416232],
       [ 3.23925122, -6.7471905 , -1.18213258,  1.89708569,  0.04066047]])
```

6. RandomForest modeling

We aim at predicting compressed daily ridership (5 PCA components values) from 12-hour lag variables. Parameter tuning is required in this section, including min_samples_split, min_samples_leaf, and n_estimators. First 80% days for training, test on the rest 20%. And in the training dataset, validate the model on the bottom 20%.

6.1 train test split

Please keep in mind that random train test split is not applicable in this case.

In [34]:
```
grouped_2018=grouped_2018.reset_index()
grouped_2018
```

Out[34]:

	index	date	hours	1	2	3	4	5	6	7	...	256	257	258	259	260	261
0	8760	2018-01-01	0	1.0	0.0	0.0	0.0	0.0	0.0	3.0	...	1.0	0.0	0.0	8.0	5.0	0.0
1	8761	2018-01-01	1	0.0	0.0	1.0	0.0	0.0	0.0	1.0	...	4.0	2.0	0.0	0.0	2.0	0.0
2	8762	2018-01-01	2	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	1.0	0.0	0.0	0.0	2.0	0.0
3	8763	2018-01-01	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	2.0	0.0	0.0	0.0
4	8764	2018-01-01	4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8755	17515	2018-12-31	19	0.0	0.0	0.0	0.0	0.0	0.0	4.0	...	3.0	1.0	5.0	0.0	2.0	5.0
8756	17516	2018-12-31	20	1.0	0.0	0.0	0.0	0.0	0.0	7.0	...	6.0	4.0	0.0	0.0	8.0	1.0
8757	17517	2018-12-31	21	0.0	0.0	0.0	2.0	0.0	0.0	9.0	...	4.0	5.0	5.0	0.0	1.0	1.0
8758	17518	2018-12-31	22	0.0	0.0	2.0	2.0	0.0	0.0	2.0	...	1.0	11.0	0.0	1.0	0.0	5.0
8759	17519	2018-12-31	23	5.0	0.0	0.0	7.0	0.0	0.0	6.0	...	0.0	11.0	0.0	0.0	1.0	1.0

8760 rows × 266 columns

In [35]:

```
pca_components_hour =pd.concat([pca_components, grouped_2018['date'],grouped_2(
pca_components_hour
```

Out[35]:

	0	1	2	3	4	0_lag_1	0_lag_2	0_lag_3
0	-2.720164	-4.312002	-0.554225	1.372271	0.879431	NaN	NaN	NaN
1	-4.799956	-3.612092	-1.400683	-0.201888	-2.080059	-2.720164	NaN	NaN
2	-8.787469	-0.853776	0.814226	-0.168481	-0.313817	-4.799956	-2.720164	NaN
3	-8.658432	-1.091070	0.500887	0.579890	0.425699	-8.787469	-4.799956	-2.720164
4	-7.905044	-1.693033	-0.752052	-0.783431	-0.666292	-8.658432	-8.787469	-4.799956
...	...	...	...	...	...	...	...	...
8755	7.454186	-4.539786	0.579454	0.888912	0.146922	-1.390830	5.546593	7.323307
8756	6.112123	-5.409046	3.829655	0.542472	1.232248	7.454186	-1.390830	5.546593
8757	8.619792	-8.059970	3.858406	0.955377	0.822999	6.112123	7.454186	-1.390830
8758	7.935406	-7.058935	0.783922	2.268200	-0.374162	8.619792	6.112123	7.454186
8759	3.239251	-6.747190	-1.182133	1.897086	0.040660	7.935406	8.619792	6.112123

8760 rows × 67 columns

In [ ]:

```
#Train test split by daily
```

In [36]:

```
pca_components_day = pca_components_hour.groupby(by='date').sum()
pca_components_day
```

Out[36]:

date	0	1	2	3	4	0_lag_1	0_lag_2	
2018-01-01	62.612008	-44.743105	56.586062	12.557598	15.442216	50.702225	32.286296	1
2018-01-02	84.746868	-36.094098	48.962733	-5.429675	6.205125	84.035248	87.515499	8
2018-01-03	36.192284	-33.511065	12.888260	-0.544993	1.960686	44.014725	51.802227	5
2018-01-04	-97.724889	-26.879326	-26.503198	-7.222360	-1.293627	-88.734585	-83.050344	-7
2018-01-05	-62.179944	-13.214626	-11.778107	-8.513691	8.028394	-71.182692	-75.780830	-7
...	...	...	...	...	...	...	...	
2018-12-27	20.203439	-14.794486	13.172488	-8.506571	-8.976869	18.597803	22.622436	2
2018-12-28	16.833980	-21.837957	9.738549	-13.645406	-11.357192	16.258343	20.525324	2
2018-12-29	14.601693	-17.897182	23.583924	-21.560275	-6.514834	20.235056	15.426699	1
2018-12-30	14.322982	-41.932620	30.016178	-1.490065	1.213039	9.204234	12.175358	1
2018-12-31	8.457661	-61.072147	35.693456	7.389496	14.416576	14.271713	11.427241	

365 rows × 66 columns

In [37]:
```
train_size = int(len(pca_components_day) * 0.8)
X = pca_components_day
X = X.iloc[:, 5:]
X_train = X.iloc[:train_size, :]
X_test = X.iloc[train_size:, :]
Y = pca_components_day.iloc[:, :5]
y_train = Y.iloc[:train_size, :]
y_test =Y .iloc[train_size:, :]
```

In [38]:
```
#your answer here
rf = RandomForestRegressor(n_estimators=50, min_samples_leaf=10, min_samples_s

# Fit the model on the training data
rf.fit(X_train, y_train)
```

Out[38]:
```
▾ RandomForestRegressor
RandomForestRegressor(min_samples_leaf=10, n_estimators=50)
```

In [39]:
```
y_pred = rf.predict(X_test)
y_pred
```

Out[39]:
```
array([[-9.02778188e+00,  1.04451065e+01,  6.33736429e+00,
        -2.38673860e+00, -3.16662986e+00],
       [ 2.77861125e+01,  6.51562148e+00,  1.58456016e+01,
        -1.55860500e+00, -1.17810381e+00],
       [ 3.86719418e+01,  6.73886438e+00,  2.28635067e+00,
        -1.16326741e+00, -2.92566006e+00],
       [-7.24392170e+00,  1.23544372e+01,  1.43239146e+00,
        -1.19054791e+00, -4.71156047e+00],
       [-1.04868544e+01,  1.05800233e+01,  6.80143025e+00,
         2.07550942e-01, -4.21131934e+00],
       [ 1.12520562e+01,  9.41663768e+00,  1.16099219e+01,
        -1.53076869e+00, -2.76137608e+00],
       [ 2.63117285e+01,  7.21191753e+00,  1.76516395e+01,
        -1.68691163e+00, -1.43078881e+00],
       [-1.68324697e+01,  6.00887243e+00,  9.88502972e+00,
        -6.21723141e+00,  4.63871733e-01],
       [ 8.50501078e+00,  1.05472371e+01,  7.42108582e+00,
        -1.16010478e+00, -3.55414681e+00],
       [ 2.23092537e+01,  4.22007876e+00,  8.81888672e-01,
        -6.27557973e+00, -1.47086787e-01],
       [-1.62670925e+01, -3.55899907e+00, -7.10704970e+00,
        -4.28295850e-01,  1.25954168e+00],
       [-2.67100704e+01,  2.91715588e+00,  4.66657911e+00,
         2.73269062e+00, -2.20148315e+00],
       [-1.42420225e+01,  6.03401527e+00, -7.63055319e-01,
         1.45404689e+00, -3.12583218e+00],
       [-1.42706374e+01,  7.39456162e+00,  9.62085311e+00,
         8.25728389e+00, -5.93396293e+00],
       [-3.06870811e+01, -1.50400710e+00,  1.00911714e+01,
        -2.80366436e+00,  2.55081470e+00],
       [-8.34875636e+00,  8.33573133e+00,  4.95160399e+00,
         5.14970136e+00, -5.33908531e+00],
       [ 7.68535335e+00,  5.52947363e+00, -6.03702022e+00,
        -2.55678855e+00, -1.06067291e+00],
       [-2.28943415e+01, -2.73835708e+00, -1.28679609e+00,
        -4.01897005e-02,  1.36278424e+00],
       [-2.72798878e+01,  2.86711058e+00,  8.37957567e+00,
        -1.39404125e+00, -2.77636507e-01],
       [-1.47627132e+01,  6.69392203e+00,  9.16530181e+00,
         6.22521681e+00, -4.70945043e+00],
       [-1.59023756e+01,  5.74753114e+00,  1.00427925e+01,
        -5.77443047e+00,  8.12964103e-01],
       [-1.56595857e+01,  2.68922867e+00,  8.91188236e+00,
        -7.80255463e+00,  2.49320951e+00],
       [-9.83217624e+00,  3.86804512e+00,  7.14471041e+00,
        -1.21064551e+00, -1.03217368e+00],
       [ 2.67013184e+01,  9.02713344e+00,  6.10503143e+00,
        -2.33619728e+00, -2.05998745e+00],
       [-1.11820284e+01,  3.19573932e+00, -5.17655563e+00,
         7.42933480e+00, -4.20018713e+00],
       [-1.96455671e+01,  5.16100154e+00,  4.98193280e+00,
         3.63411470e+00, -3.56641829e+00],
       [-7.12752867e+01, -4.26075033e+00,  5.66704055e+00,
         4.21957134e+00, -8.42169210e-02],
       [ 5.11677616e+01,  3.49937278e+00,  7.20233314e-01,
         6.81877682e-01, -3.33778335e+00],
       [ 1.33743800e+01,  2.48986240e+00,  1.34538175e+01,
        -5.00833926e+00,  6.01286881e-01],
       [-1.65447794e+01, -2.53789583e+00,  8.33106726e+00,
         7.39308577e+00, -9.17268883e-01],
```

```
[-2.22129820e+01, -5.85741498e+00,  4.54528047e+00,
  5.16150350e+00,  1.48369614e+00],
[-2.51918811e+01, -5.35507871e+00,  2.79591961e+00,
  7.52857621e+00, -1.52498703e-02],
[-5.06944021e+01, -2.73033700e+00,  5.69125604e+00,
  2.32650824e+00,  6.17590202e-01],
[-7.11713966e+01, -4.01786441e+00,  5.82800735e+00,
  4.54471442e+00, -3.32282709e-01],
[-7.11713966e+01, -4.01786441e+00,  5.82800735e+00,
  4.54471442e+00, -3.32282709e-01],
[-2.38760212e+01, -3.76495577e+00,  1.03781457e+01,
 -6.39594783e+00,  5.56275591e+00],
[ 5.87153417e+01, -1.55938332e+01,  1.88585088e+01,
  4.90976070e+00,  4.18553117e+00],
[ 5.05329261e+01,  2.23809620e+00,  9.34635798e+00,
  1.66001350e+00, -3.11917003e+00],
[ 3.02550221e+01, -2.38633383e+00, -8.83868360e+00,
  2.86827699e+00, -1.20171751e+00],
[-1.11018157e+01,  3.16053064e+00,  2.06554934e-01,
  9.94588109e+00, -5.35132949e+00],
[-2.59722185e+01,  3.67946279e+00,  9.14777553e+00,
  6.40205162e+00, -3.70657370e+00],
[-1.72456192e+01,  5.69066364e+00,  1.03920052e+01,
 -5.09223006e+00,  3.93451392e-01],
[-5.13262183e+01, -2.41979333e+00,  7.91830078e+00,
  4.30843130e-01,  1.31365137e+00],
[-1.73245104e+01,  5.61060021e+00,  7.89147324e+00,
 -5.14792996e+00,  5.03005729e-01],
[ 8.44291111e+00,  7.60929964e+00,  7.74607719e-02,
 -4.74154987e+00, -1.60287014e+00],
[-5.01816896e+01, -2.28737407e+00,  5.67446745e+00,
  1.44166623e+00,  6.98709180e-01],
[-6.44665654e+01, -2.93749608e+00,  6.46509773e+00,
  3.98499931e+00,  7.08635114e-02],
[-2.24466176e+01,  4.17181324e+00,  9.46949105e+00,
 -3.38244124e+00,  6.84601516e-02],
[-1.33002705e+01,  8.69711261e+00,  8.60384711e+00,
 -3.36626538e+00, -2.03119460e+00],
[-7.05828442e+01, -3.87958645e+00,  5.89458216e+00,
  4.47094234e+00, -2.58921436e-01],
[-2.26529871e+01,  3.78371283e+00,  9.15477517e+00,
 -1.35038173e+00, -5.78580940e-01],
[-6.81163872e+00,  9.37131110e+00,  6.04559492e-01,
  9.00348588e+00, -7.41445325e+00],
[-6.59515340e+01, -3.03013875e+00,  6.35266660e+00,
  4.17102463e+00,  1.06678116e-02],
[-6.44155603e+01, -3.08588306e+00,  6.12941112e+00,
  4.84536874e+00, -1.93453542e-01],
[-5.69963925e+01, -2.49876760e+00,  7.13237772e+00,
  4.93194469e+00, -4.53696703e-01],
[-2.33913178e+01,  1.66798110e+00,  1.12360441e+01,
 -3.90377153e+00,  2.11532821e+00],
[-7.12752867e+01, -4.26075033e+00,  5.66704055e+00,
  4.21957134e+00, -8.42169210e-02],
[-6.08421079e+01, -3.66903817e+00,  5.28352507e+00,
  4.86822693e+00, -1.72893514e-01],
[-1.89234409e+01, -3.26007260e+00, -4.36217164e+00,
  6.88489101e+00, -1.43224924e+00],
[-6.99033419e+01, -3.78144925e+00,  5.81343288e+00,
  5.67941704e+00, -6.48208419e-01],
```

```
       [-6.33051295e+01, -3.20747950e+00,  6.45007562e+00,
         3.66540782e+00,  4.02141538e-01],
       [-3.17119753e+01,  3.08239776e-01,  6.02439452e+00,
        -1.37316393e+00,  1.15616296e+00],
       [-1.78448673e+01, -9.78007907e-01, -4.10535513e+00,
        -5.73275601e+00,  2.93620129e+00],
       [-3.00957742e+01, -6.84905100e-01,  5.81114160e+00,
        -4.44224964e+00,  2.87955703e+00],
       [-7.12752867e+01, -4.26075033e+00,  5.66704055e+00,
         4.21957134e+00, -8.42169210e-02],
       [-7.11713966e+01, -4.01786441e+00,  5.82800735e+00,
         4.54471442e+00, -3.32282709e-01],
       [-7.11713966e+01, -4.01786441e+00,  5.82800735e+00,
         4.54471442e+00, -3.32282709e-01],
       [ 5.83877051e-01,  5.06821298e+00,  9.84308199e+00,
        -3.94653198e+00, -3.38981003e-01],
       [ 2.20567001e+01, -5.67115697e-01,  1.34789934e+01,
        -2.39852438e+00,  1.01773110e+00],
       [ 2.68105869e+01, -5.27603900e+00,  1.39349178e+01,
        -1.27313810e+00,  2.70553332e+00],
       [ 3.24707854e+01, -5.35564955e+00,  2.03848237e+01,
         1.31784139e+00,  2.50805772e+00],
       [ 2.23014193e+01, -4.02365932e+00,  1.70823509e+01,
        -2.78766422e-01,  1.85725785e+00],
       [ 2.68071151e+01, -5.48319617e+00,  1.78458541e+01,
         9.93962038e-01,  2.35542478e+00]])
```

In [40]:

```
y_test = np.array(y_test)
y_test
```

Out[40]:
```
array([[-1.44477741e+01,  8.85239245e+00,  6.60857916e+00,
        -2.03718957e+01, -9.15451211e+00],
       [ 3.38072659e+01,  6.97369108e+00,  1.63290479e+01,
        -6.95258656e+00, -5.15849112e+00],
       [ 3.35800333e+01,  8.15555495e+00, -6.01034607e+00,
        -2.22223603e+00, -8.03020081e+00],
       [-1.29783287e+01,  1.96959167e+01,  1.69311625e+00,
        -7.78363802e+00, -1.06841967e+01],
       [-1.41107690e+01,  2.25421435e+01,  5.94742058e+00,
        -6.30449296e-02, -7.16490355e+00],
       [ 1.14042773e+01,  2.83988479e+01,  1.91867250e+01,
         5.81192844e-01, -1.14936558e+01],
       [ 1.61751945e+01,  3.07758288e+01,  1.57854057e+01,
        -9.91301486e+00, -8.44936709e+00],
       [-2.01390666e+01,  1.36577090e+01,  1.09627459e+01,
        -1.96322922e+01, -5.81567926e+00],
       [ 9.27858818e+00,  1.49387968e+01,  1.27462299e+01,
        -2.03152889e+00, -6.33558046e+00],
       [ 1.81652514e+01, -7.87950821e-01, -3.69387087e+00,
        -1.06395871e+01, -6.11530218e+00],
       [-2.80614796e+01, -6.09427047e+00, -1.24965974e+01,
        -1.85424102e+00, -3.40170743e+00],
       [-3.76693009e+01,  2.45539241e+01, -4.57224005e+00,
         4.27519750e+00, -1.71150423e+01],
       [-1.98196983e+01,  4.28413667e+01, -5.38507431e+00,
         2.93765599e+00, -1.70217805e+01],
       [-2.05642078e+01,  2.81399241e+01,  1.09974498e+01,
         7.46240888e-01, -1.54673640e+01],
       [-4.61692868e+01, -5.19439278e+00,  8.94446314e+00,
        -1.32964520e+01, -9.22402215e-01],
       [-4.41322106e-01,  1.26266719e+01,  6.20274469e+00,
         2.36011995e+00, -3.86530497e+00],
       [ 5.34236596e+00,  9.10694518e+00, -1.06621842e+01,
        -4.84402527e+00, -7.88256080e+00],
       [-3.15047321e+01,  1.29229462e+00, -2.12437683e+00,
         3.08659645e+00, -2.02223294e+00],
       [-3.91416728e+01,  1.53919606e+01,  5.87606601e+00,
        -8.53025419e+00, -8.49479103e+00],
       [-1.98964460e+01,  2.60180540e+01,  1.48930189e+01,
         5.97075809e+00, -5.94885577e+00],
       [-9.32563610e+00,  2.29228538e+01,  2.38046332e+01,
        -6.29390873e+00, -2.74609467e+00],
       [-2.75448456e+01,  3.03832236e+00,  9.06149357e+00,
        -1.57928162e+01,  1.88608796e+00],
       [-1.12776125e+00,  6.49578582e+00,  1.91373811e+01,
        -3.31146840e+00, -7.65679913e+00],
       [ 1.89152056e+01,  1.50521067e+01,  2.39531276e+00,
         5.09613578e+00,  1.50195585e+00],
       [-1.75156926e+01,  1.02252664e+01, -1.33383522e+01,
         6.22512374e+00, -1.72229914e+00],
       [-2.87502105e+01,  1.24294745e+01,  3.08804344e-01,
         4.81564016e+00, -5.32208458e+00],
       [-6.96817705e+01,  2.30962556e+01, -8.43987008e+00,
        -3.06798530e+00,  9.04050212e+00],
       [ 7.49345985e+01,  1.06001596e+01,  1.21795874e+00,
         4.14880715e+00,  1.45502945e+01],
       [-9.12654909e+00, -1.49741604e+00,  2.04262135e+01,
        -8.53347075e+00,  6.56647951e+00],
       [-1.49402063e+01, -1.17677267e+01,  1.45397696e+01,
         6.28573812e+00, -1.21966831e+00],
```

```
[-2.63227951e+01, -1.39691904e+01,  2.56479886e+00,
  9.61881993e+00,  5.73128267e+00],
[-3.33012216e+01, -1.02192871e+01, -4.49626411e+00,
  8.81434781e+00, -1.71812139e+00],
[-4.31426238e+01,  4.38313081e+00, -2.66390776e+00,
 -2.72537364e+00, -7.81183704e+00],
[-8.53768754e+01,  5.32432305e+00, -9.33867011e+00,
 -3.79805984e+00, -1.22714644e+01],
[-8.84812634e+01, -3.77973935e+00,  7.74968131e-01,
  3.12783225e+00, -6.43713015e+00],
[-1.29420853e+01, -2.02691422e+01,  4.56698124e+01,
 -5.43244165e+00,  5.11945059e+00],
[ 9.04979454e+01, -1.26409422e+01,  7.53080681e+01,
  9.62446644e+00,  4.38506151e+00],
[ 5.97019084e+01,  6.10386039e+00,  1.02215276e+01,
  1.16343760e+01, -4.26809486e+00],
[ 2.93007891e+01, -1.30464086e+01, -8.24016558e+00,
  2.40543228e+01, -1.70448149e+01],
[-1.98854346e+01,  8.52488259e+00, -9.10064152e-01,
  8.35224601e+00, -1.15223603e+01],
[-2.89859645e+01,  3.45068068e+01,  1.01840004e+01,
  4.08314738e+00, -1.06133655e+01],
[-2.39096635e+01,  2.30138336e+01,  1.75130508e+01,
 -6.53319086e+00, -1.81199978e+00],
[-4.47801558e+01,  1.18791376e+01,  2.30607338e+01,
 -1.52653321e+01,  7.80188408e-01],
[-1.46131097e+01,  2.07180797e+01,  8.34724344e+00,
 -2.35517769e+00, -3.53759195e+00],
[ 1.46170011e+00,  2.30648476e+01, -9.96405971e+00,
 -9.69664479e+00, -1.23024959e+01],
[-4.78747557e+01,  1.39074144e+01, -3.04210163e+00,
 -1.13962474e+01, -9.96971438e+00],
[-4.98538923e+01,  2.83402111e+01,  1.04834821e+01,
 -7.31405489e+00, -8.30197104e+00],
[-2.52827349e+01,  3.72317594e+01,  8.84924310e+00,
 -2.84829493e+00, -6.52175819e+00],
[-1.88846368e+01,  2.67855190e+01,  2.37329154e+01,
 -1.13321758e+01, -8.75018339e+00],
[-6.97175337e+01,  4.43733234e+00,  1.55678836e+01,
 -1.33047486e+01,  2.51524622e+00],
[-2.06373217e+01,  1.61151597e+01,  1.24290181e+01,
  1.61776659e+00, -8.03835098e+00],
[-7.33274923e+00,  2.31000958e+01, -8.82496982e+00,
  8.55633605e+00, -9.86132489e+00],
[-5.44806140e+01,  1.16922455e+01, -5.25862808e+00,
 -2.21188798e+00, -8.11892142e+00],
[-5.14728868e+01,  2.01020580e+01,  2.62631252e+00,
  1.45703820e+00, -1.50272927e+01],
[-4.46681690e+01,  2.79118647e+01,  6.67270908e+00,
  9.06814389e+00, -6.87311859e+00],
[-2.55738891e+01,  1.85104215e+01,  1.90374584e+01,
 -9.47620122e-01,  7.30945554e+00],
[-6.99016583e+01,  5.73170859e+00,  1.80994839e+01,
 -1.13232023e+01,  7.93918580e+00],
[-3.46699677e+01, -3.22050333e+00, -2.66615045e+00,
  5.27696797e+00, -4.23234559e+00],
[-2.87390265e+01, -1.79773029e+00, -2.10428206e+01,
  9.44911655e+00, -3.64924003e+00],
[-6.36362521e+01,  1.47989273e+00, -1.13846712e+01,
  7.40666530e+00,  6.10161764e+00],
```

```
       [-4.72232485e+01,  6.42315927e+00, -4.24852017e+00,
        -1.30165471e+00,  4.99641710e+00],
       [-3.78197232e+01,  6.57934894e+00, -9.20094950e-02,
        -1.44526396e+00, -2.50100987e+00],
       [-2.77590637e+01,  2.14931125e+01, -1.12295128e+01,
        -1.42858446e+01,  2.78641531e+00],
       [-3.90284406e+01,  6.86091347e+00, -1.23541883e+00,
        -2.05430470e+01,  9.61788752e+00],
       [-6.79245953e+01,  5.43408870e+00, -5.49536812e+00,
        -4.69016121e+00, -3.17411573e+00],
       [-1.02268754e+02, -3.34927159e+00, -8.53939423e+00,
         4.70943991e+00, -3.53178145e+00],
       [-8.88345937e+01, -8.50630233e-01,  2.19008124e+00,
         9.32977329e+00, -1.08110838e+01],
       [ 3.20841817e+00,  9.78597526e-01,  1.74328508e+01,
        -9.06419044e+00, -2.74005391e+00],
       [ 2.02034387e+01, -1.47944860e+01,  1.31724880e+01,
        -8.50657080e+00, -8.97686887e+00],
       [ 1.68339799e+01, -2.18379571e+01,  9.73854931e+00,
        -1.36454055e+01, -1.13571922e+01],
       [ 1.46016931e+01, -1.78971820e+01,  2.35839237e+01,
        -2.15602752e+01, -6.51483383e+00],
       [ 1.43229824e+01, -4.19326201e+01,  3.00161777e+01,
        -1.49006498e+00,  1.21303948e+00],
       [ 8.45766075e+00, -6.10721467e+01,  3.56934558e+01,
         7.38949554e+00,  1.44165764e+01]])
```

In [41]:
```
from sklearn.metrics import r2_score

r2 = r2_score(y_test, y_pred)
```

In [42]:
```
r2
```

Out[42]:
```
0.3331655559345308
```

In [ ]:
```
# Train test split by hours
```

In [43]:
```
train_size = int(len(pca_components) * 0.8)
X = pca_components.iloc[13:,]
X = X.iloc[:, 5:]
X_train = X.iloc[:train_size, :]
X_test = X.iloc[train_size:, :]
Y = pca_components.iloc[13:,].iloc[:, :5]
y_train = Y.iloc[:train_size, :]
y_test =Y .iloc[train_size:, :]
```

6.2 model performance measurement

Use the RandomForest model with the provided parameters (min_samples_split: 2, min_samples_leaf: 10, and n_estimators equal to 50.) to predict the compressed daily ridership. Prediction results are PCA components instead of taxi zone level ridership. To reconstruct the data back to its original size and scale, we need to inverse PCA and inverse standardization. report the taxi zone level R^2 value.

In [44]:
```
#your answer here
rf = RandomForestRegressor(n_estimators=50, min_samples_leaf=10, min_samples_s

# Fit the model on the training data
rf.fit(X_train, y_train)
```

Out[44]:
```
RandomForestRegressor
RandomForestRegressor(min_samples_leaf=10, n_estimators=50)
```

In [45]:
```
y_pred = rf.predict(X_test)
y_pred
```

Out[45]:
```
array([[ 1.22742426,  2.44893175, -0.03127481,  0.02336949,  0.04823541],
       [ 4.04410323,  3.00525498, -0.77888339, -0.30251751,  0.13231062],
       [ 4.59584928,  2.40853006, -0.44148399, -0.20524202,  0.17279652],
       ...,
       [ 3.45775339, -2.98143902, -0.11226782, -0.05467581, -0.05032933],
       [ 4.67589206, -3.27699673, -0.12082895,  0.00972545, -0.19545674],
       [ 3.5982126 , -3.20680853, -0.56601796, -0.06323277, -0.32466589]])
```

In [46]:
```
y_test = np.array(y_test)
y_test
```

Out[46]:
```
array([[-1.304949  ,  2.98961347, -0.608941  , -0.09891158,  0.49470123],
       [ 2.95845202,  2.69846343,  0.3365104 , -0.65082308, -0.19725592],
       [ 2.90441484,  3.55936594,  0.32574949, -0.1090164 , -1.27846526],
       ...,
       [ 8.61979218, -8.05996965,  3.85840615,  0.95537746,  0.82299888],
       [ 7.9354065 , -7.05893474,  0.78392213,  2.26819964, -0.37416232],
       [ 3.23925122, -6.7471905 , -1.18213258,  1.89708569,  0.04066047]])
```

In [47]:
```
from sklearn.metrics import r2_score

r2 = r2_score(y_test, y_pred)
```

In [48]:
```
r2
```

Out[48]:
```
0.37744672969744325
```

In [ ]: