Assignment NO.1 Solutions

Data Mining | Fall 1401 | Dr.Rahmani

Student name : **Amin Fathi**

Student id : **400722102**

ابتدا مجموعه داده به صورت یک DataFrame خوانده می‌شود. برای انجام پیش‌پردازش‌های لازم، اولین گام بررسی مقادیر null است. با توجه به اینکه در ستون مسافت تجمیعی مقدار null وجود دارد و همچنین این مسئله یک مسئله‌ی پیش‌بینی مسافت طی شده است که مسافت طی شده در هر روز تصادفی بوده و روند خاصی را طی نمی‌کند، از این رو مقدار مشخصی را به‌جای null جایگزین نکرده (مثلا میانگین) و سطر حاوی null را به‌صورت کامل از مجموعه داده حذف می‌کنیم. این کار باعث کاهش خطا در روند پیش‌بینی می‌شود. علاوه بر این، ستون time که یک object شامل روز، ماه، زمان و ساعت بوده را نیز در ستون‌های مجزا از همدیگر جدا می‌کنیم.

```
data = pd.read_csv('/content/ARIMA-dataset.csv')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47091 entries, 0 to 47090
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   time       47091 non-null  object 
 1   lat        47091 non-null  float64
 2   lon        47091 non-null  float64
 3   dev_acc_d  45527 non-null  float64
dtypes: float64(3), object(1)
memory usage: 1.4+ MB
```

```
[4]  data['dev_acc_d'].isnull().values.any()
```

```
True
```

```
[5]  index = data['dev_acc_d'].index[data['dev_acc_d'].apply(np.isnan)]
     data.drop(index, axis=0, inplace=True)
```

```
[6]  data['time'] = pd.to_datetime(data['time'], format="%Y-%m-%d %H:%M:%S.%f")
     data['year'] = data['time'].dt.year
     data['month'] = data['time'].dt.month
     data['day'] = data['time'].dt.day
     data['hour'] = data['time'].dt.hour
     data['week_day'] = data['time'].dt.dayofweek
```

برای محاسبه‌ی مسافت طی شده در هر روز، ابتدا هر دو عنصر ستون dev_acc_d از هم کم شده و مسافت‌های طی شده در یک (اما در ساعت‌های مختلف) را با هم جمع کرده تا مسافت کلی یک روز مشخص شود.

```
data['dist_per_day'] = data['dev_acc_d'].diff()
```

```
data['dist_per_day'][0] = 0
```

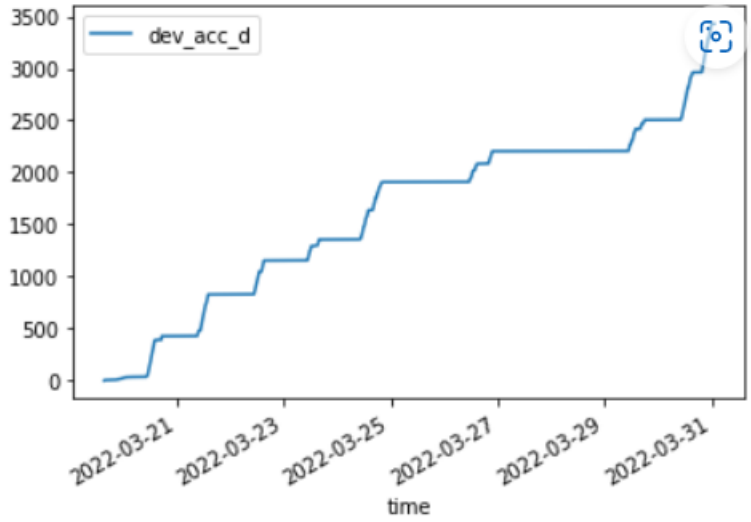
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

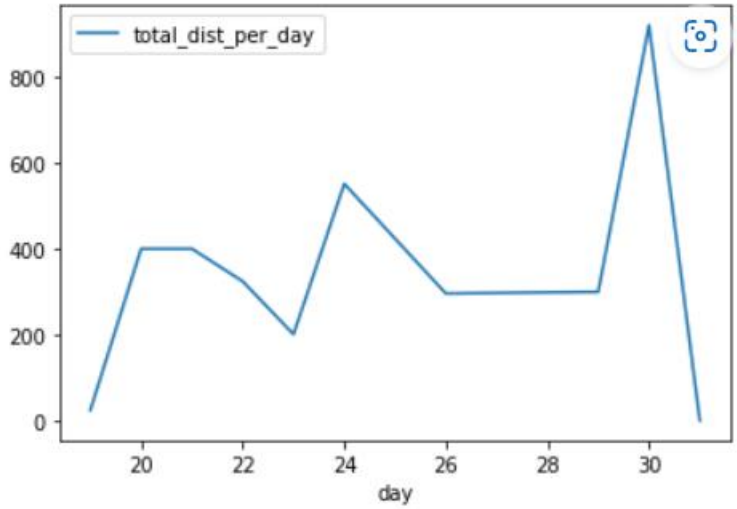
```
data2 = data.groupby('day').agg('sum')['dist_per_day'].to_frame(name='total_dist_per_day').reset_index()
```

نمودار مسافت تجمعی و مسافت طی شده در هر روز به ترتیب در زیر نشان داده شده است.

```
data.plot(x = 'time', y = 'dev_acc_d')
pt.show()
```




```
[11] data2.plot(x = 'day', y = 'total_dist_per_day')
     pt.show()
```



برای تخمین پارامترهای p، q و d، ابتدا به نمودار مجموعه داده نگاه می‌کنیم تا stationary بودن یا نبودن آن و همچنین نیاز به استفاده از عملیات differencing مشخص شود.

```
[12] data3 = data2.set_index('day')
```

```
[13] from statsmodels.tsa.stattools import adfuller
     X = data3.total_dist_per_day.values
     result = adfuller(X)
     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))
```

```
ADF Statistic: 0.481267
p-value: 0.984292
Critical Values:
        1%: -5.354
        5%: -3.646
        10%: -2.901
```

به دلیل کمبود تعداد داده‌ها، از تست ADF استفاده شده و متوجه stationary نبودن مجموعه داده می‌شود. به همین دلیل، نیاز به استفاده از عملیات differencing است. پس از differencing، مجموعه داده stationary شده و مجدداً تست ADF انجام می‌شود و مقدار d برابر با ۱ تخمین زده می‌شود.

```
[14] first_diffs = data3.total_dist_per_day.values[1:] - data3.total_dist_per_day.values[:-1]
     first_diffs = np.concatenate([first_diffs, [0]])
```

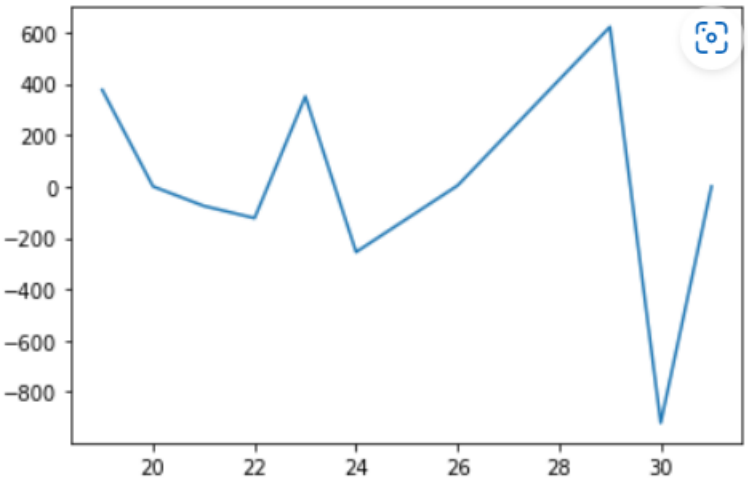
```
[15] data3['FirstDifference'] = first_diffs
```

```
X = data3.FirstDifference.values
result = adfuller(X)
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))
```

```
ADF Statistic: -4.541499
p-value: 0.000165
Critical Values:
        1%: -4.665
        5%: -3.367
        10%: -2.803
```

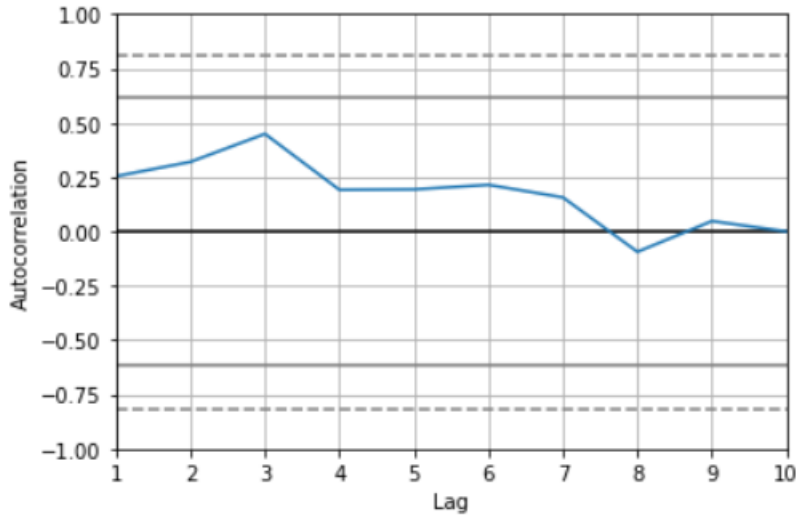
نمودار داده‌ها بعد از عملیات differencing در شکل زیر نشان داده شده است.

```
[17] pt.plot(data3.FirstDifference)
     pt.show()
```



از نمودار autocorrelation نیز برای تخمین p و q استفاده می‌کنیم.

```
[18] from pandas.plotting import autocorrelation_plot
     autocorrelation_plot(data2)
     pt.show()
```



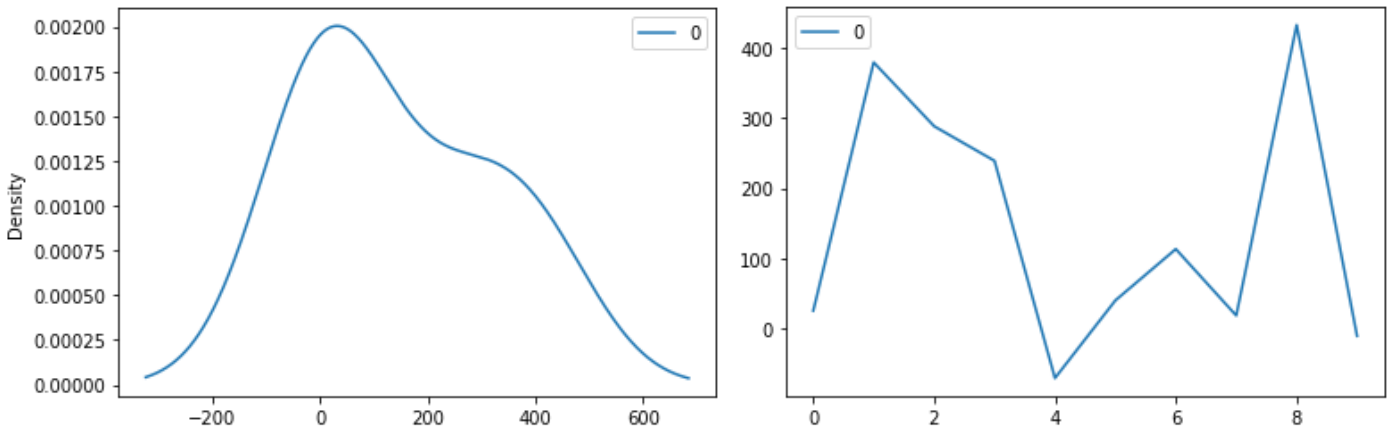
تا lag = 7، correlation مثبت است که این میزان برای lag = 3 قابل‌توجه‌تر است. در نتیجه با فرض q = 0، مقدار p = 3 تخمین زده می‌شود.

در ادامه مدل ARIMA آموزش داده می‌شود.

```
#fit model
model = ARIMA(data2.total_dist_per_day, order=(3,1,0))
model_fit = model.fit()
# summary of fit model
print(model_fit.summary())
# line plot of residuals
residuals = DataFrame(model_fit.resid)
residuals.plot()
pt.show()
# density plot of residuals
residuals.plot(kind='kde')
pt.show()
# summary stats of residuals
print(residuals.describe())
```

```
                               SARIMAX Results                                
==============================================================================
Dep. Variable:     total_dist_per_day   No. Observations:                   10
Model:                 ARIMA(3, 1, 0)   Log Likelihood                -61.293
Date:                Fri, 18 Nov 2022   AIC                            130.585
Time:                        19:49:07   BIC                            131.374
Sample:                             0   HQIC                           128.883
                                  - 10                                         
Covariance Type:                  opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -1.6502      1.122     -1.471      0.141      -3.849       0.549
ar.L2         -1.4103      1.697     -0.831      0.406      -4.737       1.916
ar.L3         -0.4722      1.425     -0.331      0.740      -3.264       2.320
sigma2      3.234e+04   2.64e+04      1.223      0.221   -1.95e+04    8.42e+04
===================================================================================
Ljung-Box (L1) (Q):                   1.88   Jarque-Bera (JB):                 1.35
Prob(Q):                              0.17   Prob(JB):                         0.51
Heteroskedasticity (H):               2.09   Skew:                             0.93
Prob(H) (two-sided):                  0.56   Kurtosis:                         3.36
===================================================================================
```

نمودار خطی خطاهای باقی‌مانده و نمودار چگالی مقادیر خطای باقی‌مانده در شکل زیر نشان داده شده است.




```
count   10.000000
mean   145.410727
std    176.573925
min    -71.007038
25%     19.989016
50%     76.484070
75%    276.024340
max    432.760363
```

با توجه به وجود میانگین غیرصفر در باقی‌مانده، متوجه شدیم که باياس در پیش‌بینی وجود دارد.

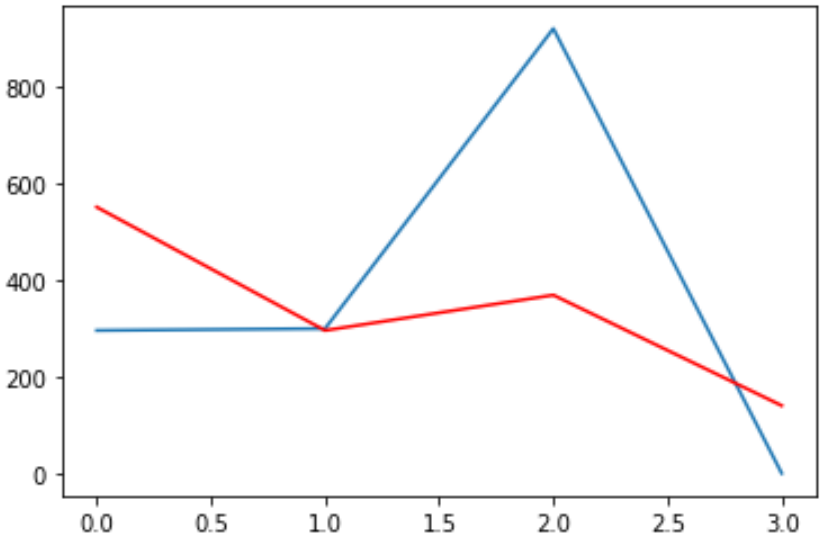
برای آموزش مدل، داده‌ها به دو قسمت train و test تقسیم می‌شوند.

```
# split into train and test sets
X = data3.total_dist_per_day.values
size = int(len(X) * 0.66)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()
# walk-forward validation
for t in range(len(test)):
    model = ARIMA(history, order=(3,1,0))
    model_fit = model.fit()
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    obs = test[t]
    history.append(obs)
    print('predicted=%f, expected=%f' % (yhat, obs))
# evaluate forecasts
rmse = sqrt(mean_squared_error(test, predictions))
print('Test RMSE: %.3f' % rmse)
# plot forecasts against actual outcomes
pyplot.plot(test)
pyplot.plot(predictions, color='red')
pyplot.show()
```

نتایج حاصل در زیر نشان داده شده است.

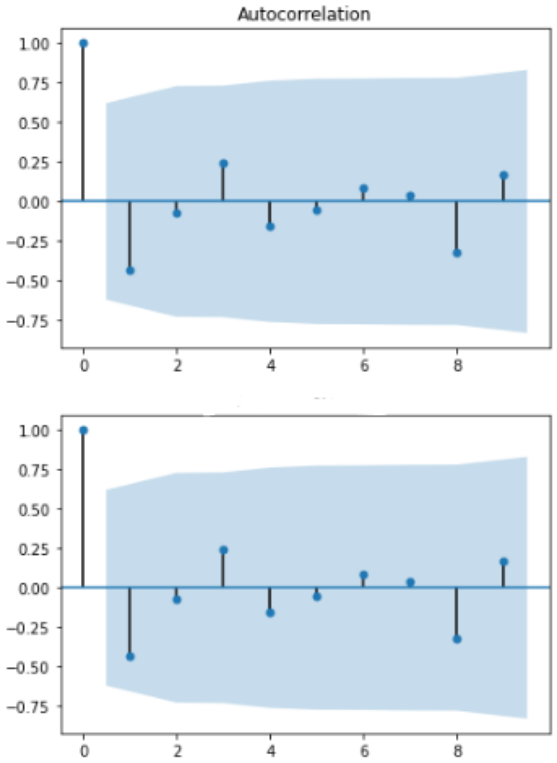
```
predicted=552.444285, expected=297.042210
predicted=297.042046, expected=300.806880
predicted=370.255855, expected=922.060740
predicted=141.479102, expected=1.269300
Test RMSE: 312.006
```

نمودار نتایج حاصل نیز در ادامه نشان داده شده است.



با استفاده از کتابخانه‌های statmodels و plot_acf و plot_pacf نمودارها رسم شده و سعی در محاسبه‌ی مقادیر p، q و d داریم.

```
from statsmodels.tsa.stattools import acf, pacf
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
plot_acf(data3.FirstDifference)
```

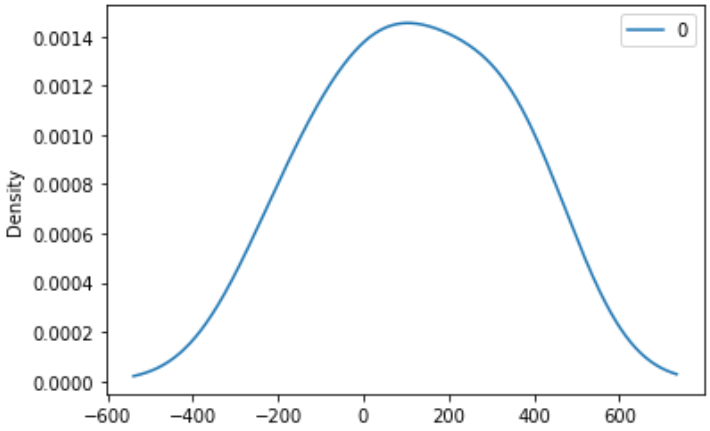


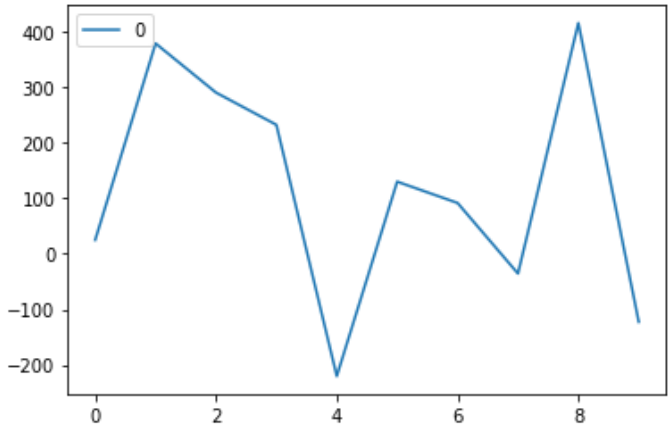
اگر به نمودار توجه شود، مشاهده می‌شود که با توجه به مقداری که در هر Lag داریم، دو lag از مقدار حد تجاوز کرده و در نتیجه $p = 2$ است.

در انتها بار دیگر مدل را با پارامترهای جدید آموزش می‌دهیم و نتایج حاصل به صورت زیر است.

```
#fit model
model = ARIMA(data2.total_dist_per_day, order=(2,1,0))
model_fit = model.fit()
# summary of fit model
print(model_fit.summary())
# line plot of residuals
residuals = DataFrame(model_fit.resid)
residuals.plot()
pt.show()
# density plot of residuals
residuals.plot(kind='kde')
pt.show()
# summary stats of residuals
print(residuals.describe())
```

```
==============================================================================
Dep. Variable:     total_dist_per_day   No. Observations:                   10
Model:                 ARIMA(2, 1, 0)   Log Likelihood                -61.999
Date:                Fri, 18 Nov 2022   AIC                            129.998
Time:                        19:49:24   BIC                            130.589
Sample:                             0   HQIC                           128.721
                                  - 10
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -1.2798      0.461     -2.778      0.005      -2.183      -0.377
ar.L2         -0.8208      0.355     -2.313      0.021      -1.516      -0.125
sigma2      4.123e+04   2.84e+04      1.451      0.147   -1.45e+04    9.69e+04
===================================================================================
Ljung-Box (L1) (Q):                   3.13   Jarque-Bera (JB):                 0.16
Prob(Q):                              0.08   Prob(JB):                         0.92
Heteroskedasticity (H):               1.72   Skew:                            -0.08
Prob(H) (two-sided):                  0.67   Kurtosis:                         2.38
===================================================================================
```






```
count    10.000000
mean    118.854204
std     212.408462
min    -220.472179
25%     -20.299050
50%     111.063308
75%     276.428450
max     416.443815
```

```
# split into train and test sets
X = data3.total_dist_per_day.values
size = int(len(X) * 0.66)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()
# walk-forward validation
for t in range(len(test)):
    model = ARIMA(history, order=(2,1,0))
    model_fit = model.fit()
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    obs = test[t]
    history.append(obs)
    print('predicted=%f, expected=%f' % (yhat, obs))
# evaluate forecasts
rmse = sqrt(mean_squared_error(test, predictions))
print('Test RMSE: %.3f' % rmse)
# plot forecasts against actual outcomes
pyplot.plot(test)
pyplot.plot(predictions, color='red')
pyplot.show()
```

```
predicted=399.387133, expected=297.042210
predicted=321.974869, expected=300.806880
predicted=452.191453, expected=922.060740
predicted=347.322271, expected=1.269300
Test RMSE: 296.417
```