In [1]:

```
import os
import pandas as pd
import numpy as np

from sklearn.preprocessing import MinMaxScaler
import joblib
import seaborn as sns
sns.set(color_codes=True)
import matplotlib.pyplot as plt
%pylab inline

from numpy.random import seed
seed(1)
from tensorflow import random
random.set_seed(1)
import tensorflow as tf

from keras.layers import Input, Dropout, Dense, LSTM, TimeDistributed, RepeatVector
from keras.models import Model
from keras import regularizers
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:

```
seed(10)
random.set_seed(10)
```

In [3]:

```
data = pd.read_csv('pollutionData158324.csv', header=0, infer_datetime_format=True, parse_d
data.head(5)
```

Out[3]:

timestamp	ozone	particullate_matter	carbon_monoxide	sulfure_dioxide	nitrogen_dioxide	long
2014-08-01 00:05:00	101	94	49	44	87	10.1(
2014-08-01 00:10:00	106	97	48	47	86	10.1(
2014-08-01 00:15:00	107	95	49	42	85	10.1(
2014-08-01 00:20:00	103	90	51	44	87	10.1(
2014-08-01 00:25:00	105	94	49	39	82	10.1(

In [4]:

```
data= data.drop('longitude', axis =1)
data= data.drop('latitude', axis =1)
```

In [5]:

```
data.head(5)
```

Out[5]:

timestamp	ozone	particullate_matter	carbon_monoxide	sulfure_dioxide	nitrogen_dioxide
2014-08-01 00:05:00	101	94	49	44	87
2014-08-01 00:10:00	106	97	48	47	86
2014-08-01 00:15:00	107	95	49	42	85
2014-08-01 00:20:00	103	90	51	44	87
2014-08-01 00:25:00	105	94	49	39	82

In [6]:

```
data.shape
```

Out[6]:

```
(17568, 5)
```

In [7]:

```
train_size = int(len(data) * 0.8)
test_size = len(data) - train_size
data_train, data_test = data.iloc[0:train_size], data.iloc[train_size:len(data)]
print("Training set shape : ", data_train.shape)
print("Testing set shape : ", data_test.shape)
```

```
Training set shape :  (14054, 5)
Testing set shape :  (3514, 5)
```

In [8]:

```
# zamanları frenaks cınsıne donusturduk
data_train_fft= np.fft.fft(data_train)
data_test_fft= np.fft.fft(data_test)
```

In [9]:

```
# normalizasyon

scaler=MinMaxScaler()
X_train = scaler.fit_transform(data_train)
X_test = scaler.transform(data_test)
scaler_filename = "scaler_data"
joblib.dump(scaler, scaler_filename)
```

Out[9]:

```
['scaler_data']
```

In [10]:

```
 # model için shapleri yeniden düzenleme
X_train= X_train.reshape(X_train.shape[0], 1,X_train.shape[1])
print("Train data büyüklüğü : ", X_train.shape)
X_test= X_test.reshape(X_test.shape[0], 1,X_test.shape[1])
print("Test data büyüklüğü : ", X_test.shape)
```

```
Train data büyüklüğü :  (14054, 1, 5)
Test data büyüklüğü :  (3514, 1, 5)
```

In [11]:

```
# model kurulumu
def autoencoder_model(X):
    inputs = Input(shape=(X.shape[1], X.shape[2]))
    L1 = LSTM(16, activation='relu', return_sequences=True,
              kernel_regularizer=regularizers.l2(0.00))(inputs)
    L2 = LSTM(5, activation='relu', return_sequences=False)(L1)
    L3 = RepeatVector(X.shape[1])(L2)
    L4 = LSTM(5, activation='relu', return_sequences=True)(L3)
    L5 = LSTM(16, activation='relu', return_sequences=True)(L4)
    output = TimeDistributed(Dense(X.shape[2]))(L5)
    model = Model(inputs=inputs, outputs=output)
    return model
```

In [12]:

```
model = autoencoder_model(X_train)
model.compile(optimizer='adam', loss='mae')
model.summary()
```

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 1, 5)]            0
_________________________________________________________________
lstm (LSTM)                  (None, 1, 16)             1408
_________________________________________________________________
lstm_1 (LSTM)                (None, 5)                 440
_________________________________________________________________
repeat_vector (RepeatVector) (None, 1, 5)              0
_________________________________________________________________
lstm_2 (LSTM)                (None, 1, 5)              220
_________________________________________________________________
lstm_3 (LSTM)                (None, 1, 16)             1408
_________________________________________________________________
time_distributed (TimeDistri (None, 1, 5)              85
=================================================================
Total params: 3,561
Trainable params: 3,561
Non-trainable params: 0
_________________________________________________________________
```

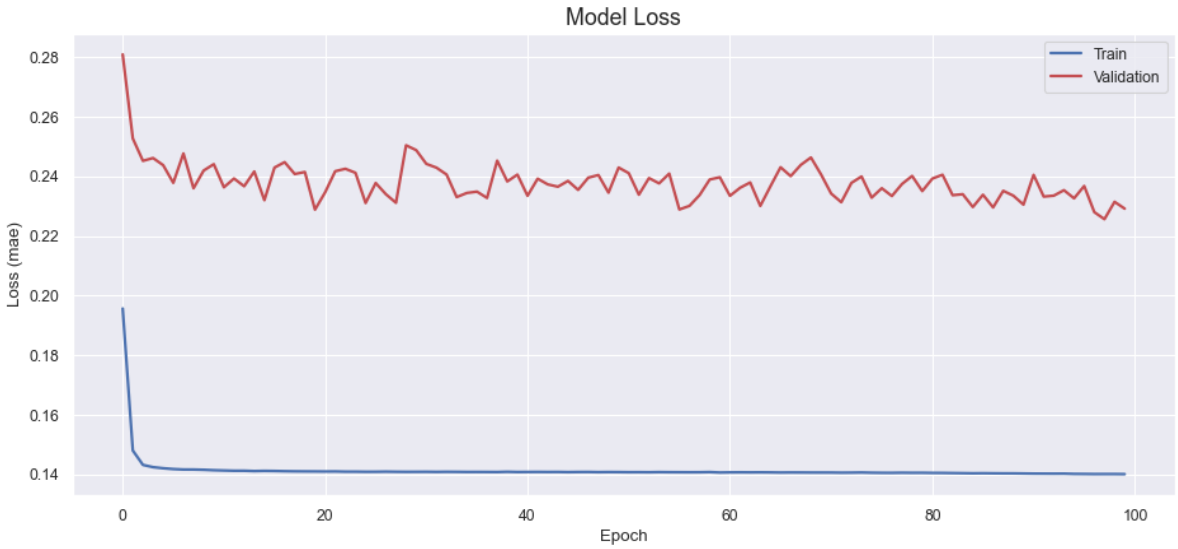
In [13]:

```
nb_epochs = 100
batch_size = 10
history = model.fit(X_train, X_train, epochs=nb_epochs, batch_size=batch_size,
                    validation_split=0.05).history
```

```
val_loss: 0.2353
Epoch 95/100
1336/1336 [==============================] - 4s 3ms/step - loss: 0.1402 -
val_loss: 0.2326
Epoch 96/100
1336/1336 [==============================] - 4s 3ms/step - loss: 0.1402 -
val_loss: 0.2368
Epoch 97/100
1336/1336 [==============================] - 4s 3ms/step - loss: 0.1402 -
val_loss: 0.2280
Epoch 98/100
1336/1336 [==============================] - 4s 3ms/step - loss: 0.1402 -
val_loss: 0.2256
Epoch 99/100
1336/1336 [==============================] - 4s 3ms/step - loss: 0.1402 -
val_loss: 0.2315
Epoch 100/100
1336/1336 [==============================] - 4s 3ms/step - loss: 0.1402 -
val_loss: 0.2292
```

In [14]:

```
fig, ax = plt.subplots(figsize=(14,6), dpi=80)
ax.plot(history['loss'], 'b', label='Train', linewidth=2)
ax.plot(history['val_loss'], 'r', label='Validation', linewidth=2)
ax.set_title('Model Loss', fontsize=16)
ax.set_ylabel('Loss (mae)')
ax.set_xlabel('Epoch')
ax.legend(loc='upper right')
plt.show()
```



In [15]:

```
X_test
```

Out[15]:

```
array([[[0.765     , 0.165     , 0.86432161, 0.565     , 0.76649746]],

       [[0.77      , 0.17      , 0.88944724, 0.58      , 0.78680203]],

       [[0.75      , 0.19      , 0.89447236, 0.605     , 0.79187817]],

       ...,

       [[0.215     , 0.635     , 0.22613065, 0.9       , 0.54314721]],

       [[0.225     , 0.655     , 0.20100503, 0.92      , 0.55329949]],

       [[0.23      , 0.65      , 0.19095477, 0.945     , 0.54822335]]])
```

In [16]:

```
X_train
```

Out[16]:

```
array([[[0.43      , 0.395     , 0.17085427, 0.145     , 0.35025381]],

       [[0.455     , 0.41      , 0.16582915, 0.16      , 0.34517766]],

       [[0.46      , 0.4       , 0.17085427, 0.135     , 0.34010152]],

       ...,

       [[0.76      , 0.16      , 0.85929648, 0.6       , 0.79695431]],

       [[0.76      , 0.135     , 0.88442211, 0.575     , 0.78172589]],

       [[0.745     , 0.15      , 0.86934673, 0.59      , 0.79187817]]])
```

In [17]:

```
X_pred = model.predict(X_train)
X_pred = X_pred.reshape(X_pred.shape[0], X_pred.shape[2])
X_pred = pd.DataFrame(X_pred, columns= data_train.columns)
X_pred.index = data_train.index
```

In [18]:

```
scored = pd.DataFrame(index=data_train.index)
Xtrain = X_train.reshape(X_pred.shape[0], X_pred.shape[1])
```

In [19]:

```
X_pred = model.predict(X_test)
X_pred = X_pred.reshape(X_pred.shape[0], X_pred.shape[2])
X_pred = pd.DataFrame(X_pred, columns=data_test.columns)
X_pred.index = data_test.index
```

In [20]:

```
X_pred.head(5)
```

Out[20]:

timestamp	ozone	particullate_matter	carbon_monoxide	sulfure_dioxide	nitrogen_dioxide
2014-09-18 19:15:00	0.349957	0.467971	0.387402	0.638032	0.811599
2014-09-18 19:20:00	0.361077	0.465588	0.392639	0.659522	0.807454
2014-09-18 19:25:00	0.375879	0.461637	0.400223	0.685817	0.802057
2014-09-18 19:30:00	0.388475	0.457647	0.407185	0.706337	0.797581
2014-09-18 19:35:00	0.382151	0.459718	0.403634	0.696235	0.799815

In [21]:

```
Xtest = X_test.reshape(X_test.shape[0], X_test.shape[2])
```

In [22]:

```
Xtest
```

Out[22]:

```
array([[0.765     , 0.165     , 0.86432161, 0.565     , 0.76649746],
       [0.77      , 0.17      , 0.88944724, 0.58      , 0.78680203],
       [0.75      , 0.19      , 0.89447236, 0.605     , 0.79187817],
       ...,
       [0.215     , 0.635     , 0.22613065, 0.9       , 0.54314721],
       [0.225     , 0.655     , 0.20100503, 0.92      , 0.55329949],
       [0.23      , 0.65      , 0.19095477, 0.945     , 0.54822335]])
```

In [23]:

```
XtestDataframe=pd.DataFrame(Xtest,columns=['ozone','particullate_matter','carbon_monoxide',
```

In [24]:

```
XtestDataframe
```

Out[24]:

	ozone	particullate_matter	carbon_monoxide	sulfure_dioxide	nitrogen_dioxide
0	0.765	0.165	0.864322	0.565	0.766497
1	0.770	0.170	0.889447	0.580	0.786802
2	0.750	0.190	0.894472	0.605	0.791878
3	0.765	0.205	0.904523	0.620	0.771574
4	0.750	0.225	0.904523	0.615	0.791878
...	...	...	...	...	...
3509	0.190	0.600	0.206030	0.890	0.553299
3510	0.190	0.610	0.201005	0.875	0.543147
3511	0.215	0.635	0.226131	0.900	0.543147
3512	0.225	0.655	0.201005	0.920	0.553299

In [25]:

```
ozonepredict = pd.DataFrame(X_pred.ozone)
ozonepredict
```

Out[25]:

	ozone
timestamp	
2014-09-18 19:15:00	0.349957
2014-09-18 19:20:00	0.361077
2014-09-18 19:25:00	0.375879
2014-09-18 19:30:00	0.388475
2014-09-18 19:35:00	0.382151
...	...
2014-09-30 23:40:00	0.478224
2014-09-30 23:45:00	0.467474
2014-09-30 23:50:00	0.486916
2014-09-30 23:55:00	0.499728
2014-10-01 00:00:00	0.518357

3514 rows × 1 columns

In [26]:

```
ozonepredict['Loss_mae'] = abs(X_pred.ozone.values - XtestDataframe.ozone.values)
ozonepredict['Threshold'] = 0.3
ozonepredict['Anomaly'] = ozonepredict['Loss_mae'] > ozonepredict ['Threshold']
```

In [27]:

```
ozonepredict.head(10)
```

Out[27]:

timestamp	ozone	Loss_mae	Threshold	Anomaly
2014-09-18 19:15:00	0.349957	0.415043	0.3	True
2014-09-18 19:20:00	0.361077	0.408923	0.3	True
2014-09-18 19:25:00	0.375879	0.374121	0.3	True
2014-09-18 19:30:00	0.388475	0.376525	0.3	True
2014-09-18 19:35:00	0.382151	0.367849	0.3	True
2014-09-18 19:40:00	0.365350	0.384650	0.3	True
2014-09-18 19:45:00	0.366192	0.383808	0.3	True
2014-09-18 19:50:00	0.360838	0.369162	0.3	True
2014-09-18 19:55:00	0.346110	0.393890	0.3	True
2014-09-18 20:00:00	0.336923	0.388077	0.3	True

In [28]:

```
anomalies_ozone = ozonepredict[ozonepredict.Anomaly == True]
print(anomalies_ozone.shape)
anomalies_ozone.head()
```

```
(730, 4)
```

Out[28]:

timestamp	ozone	Loss_mae	Threshold	Anomaly
2014-09-18 19:15:00	0.349957	0.415043	0.3	True
2014-09-18 19:20:00	0.361077	0.408923	0.3	True
2014-09-18 19:25:00	0.375879	0.374121	0.3	True
2014-09-18 19:30:00	0.388475	0.376525	0.3	True
2014-09-18 19:35:00	0.382151	0.367849	0.3	True

In [29]:

```
particullate_matter_predict = pd.DataFrame(X_pred.particullate_matter)
particullate_matter_predict
```

Out[29]:

timestamp	particullate_matter
2014-09-18 19:15:00	0.467971
2014-09-18 19:20:00	0.465588
2014-09-18 19:25:00	0.461637
2014-09-18 19:30:00	0.457647
2014-09-18 19:35:00	0.459718
...	...
2014-09-30 23:40:00	0.416988
2014-09-30 23:45:00	0.422735
2014-09-30 23:50:00	0.412210
2014-09-30 23:55:00	0.404968
2014-10-01 00:00:00	0.394061

3514 rows × 1 columns

In [30]:

```
particullate_matter_predict['Loss_mae'] = abs(X_pred.particullate_matter.values - XtestData
particullate_matter_predict['Threshold'] = 0.3
particullate_matter_predict['Anomaly'] = particullate_matter_predict['Loss_mae'] > particul
```

In [31]:

```
particullate_matter_predict
```

Out[31]:

timestamp	particullate_matter	Loss_mae	Threshold	Anomaly
2014-09-18 19:15:00	0.467971	0.302971	0.3	True
2014-09-18 19:20:00	0.465588	0.295588	0.3	False
2014-09-18 19:25:00	0.461637	0.271637	0.3	False
2014-09-18 19:30:00	0.457647	0.252647	0.3	False
2014-09-18 19:35:00	0.459718	0.234718	0.3	False
...	...	...	...	...
2014-09-30 23:40:00	0.416988	0.183012	0.3	False
2014-09-30 23:45:00	0.422735	0.187265	0.3	False
2014-09-30 23:50:00	0.412210	0.222790	0.3	False
2014-09-30 23:55:00	0.404968	0.250032	0.3	False
2014-10-01 00:00:00	0.394061	0.255939	0.3	False

3514 rows × 4 columns

In [32]:

```
anomalies_particullate_matter = particullate_matter_predict[particullate_matter_predict.Ano
print(anomalies_particullate_matter.shape)
anomalies_particullate_matter.head()
```

```
(756, 4)
```

Out[32]:

timestamp	particullate_matter	Loss_mae	Threshold	Anomaly
2014-09-18 19:15:00	0.467971	0.302971	0.3	True
2014-09-18 20:30:00	0.469249	0.324249	0.3	True
2014-09-18 20:40:00	0.468385	0.313385	0.3	True
2014-09-18 20:45:00	0.467039	0.332039	0.3	True
2014-09-18 20:50:00	0.468030	0.348030	0.3	True

In [33]:

```
carbon_monoxide_predict = pd.DataFrame(X_pred.carbon_monoxide)
carbon_monoxide_predict
```

Out[33]:

timestamp	carbon_monoxide
2014-09-18 19:15:00	0.387402
2014-09-18 19:20:00	0.392639
2014-09-18 19:25:00	0.400223
2014-09-18 19:30:00	0.407185
2014-09-18 19:35:00	0.403634
...	...
2014-09-30 23:40:00	0.466799
2014-09-30 23:45:00	0.458957
2014-09-30 23:50:00	0.473242
2014-09-30 23:55:00	0.482887
2014-10-01 00:00:00	0.497189

3514 rows × 1 columns

In [34]:

```
carbon_monoxide_predict['Loss_mae'] = abs(X_pred.carbon_monoxide.values - XtestDataframe.ca
carbon_monoxide_predict['Threshold'] = 0.3
carbon_monoxide_predict['Anomaly'] = carbon_monoxide_predict['Loss_mae'] > carbon_monoxide_
```

In [35]:

```
carbon_monoxide_predict.head(10)
```

Out[35]:

timestamp	carbon_monoxide	Loss_mae	Threshold	Anomaly
2014-09-18 19:15:00	0.387402	0.476920	0.3	True
2014-09-18 19:20:00	0.392639	0.496809	0.3	True
2014-09-18 19:25:00	0.400223	0.494250	0.3	True
2014-09-18 19:30:00	0.407185	0.497337	0.3	True
2014-09-18 19:35:00	0.403634	0.500889	0.3	True
2014-09-18 19:40:00	0.394758	0.524840	0.3	True
2014-09-18 19:45:00	0.395183	0.539491	0.3	True
2014-09-18 19:50:00	0.392522	0.542151	0.3	True
2014-09-18 19:55:00	0.385687	0.559037	0.3	True
2014-09-18 20:00:00	0.381798	0.572976	0.3	True

In [36]:

```
anomalies_carbon_monoxide = carbon_monoxide_predict[carbon_monoxide_predict.Anomaly == True
print(anomalies_carbon_monoxide.shape)
anomalies_carbon_monoxide.head()
```

```
(689, 4)
```

Out[36]:

timestamp	carbon_monoxide	Loss_mae	Threshold	Anomaly
2014-09-18 19:15:00	0.387402	0.476920	0.3	True
2014-09-18 19:20:00	0.392639	0.496809	0.3	True
2014-09-18 19:25:00	0.400223	0.494250	0.3	True
2014-09-18 19:30:00	0.407185	0.497337	0.3	True
2014-09-18 19:35:00	0.403634	0.500889	0.3	True

In [37]:

```
sulfure_dioxide_predict = pd.DataFrame(X_pred.sulfure_dioxide)
sulfure_dioxide_predict
```

Out[37]:

timestamp	sulfure_dioxide
2014-09-18 19:15:00	0.638032
2014-09-18 19:20:00	0.659522
2014-09-18 19:25:00	0.685817
2014-09-18 19:30:00	0.706337
2014-09-18 19:35:00	0.696235
...	...
2014-09-30 23:40:00	0.816562
2014-09-30 23:45:00	0.805936
2014-09-30 23:50:00	0.824765
2014-09-30 23:55:00	0.836273
2014-10-01 00:00:00	0.851887

3514 rows × 1 columns

In [38]:

```
sulfure_dioxide_predict['Loss_mae'] = abs(X_pred.sulfure_dioxide.values - XtestDataframe.su
sulfure_dioxide_predict['Threshold'] = 0.3
sulfure_dioxide_predict['Anomaly'] = sulfure_dioxide_predict['Loss_mae'] > sulfure_dioxide_
```

In [39]:

```
sulfure_dioxide_predict.head(10)
```

Out[39]:

timestamp	sulfure_dioxide	Loss_mae	Threshold	Anomaly
2014-09-18 19:15:00	0.638032	0.073032	0.3	False
2014-09-18 19:20:00	0.659522	0.079522	0.3	False
2014-09-18 19:25:00	0.685817	0.080817	0.3	False
2014-09-18 19:30:00	0.706337	0.086337	0.3	False
2014-09-18 19:35:00	0.696235	0.081235	0.3	False
2014-09-18 19:40:00	0.667373	0.077373	0.3	False
2014-09-18 19:45:00	0.668893	0.078893	0.3	False
2014-09-18 19:50:00	0.659078	0.074078	0.3	False
2014-09-18 19:55:00	0.630219	0.070219	0.3	False
2014-09-18 20:00:00	0.610715	0.065715	0.3	False

In [40]:

```
anomalies_sulfure_dioxide = sulfure_dioxide_predict[sulfure_dioxide_predict.Anomaly == True
print(anomalies_sulfure_dioxide.shape)
anomalies_sulfure_dioxide.head()
```

```
(0, 4)
```

Out[40]:

timestamp	sulfure_dioxide	Loss_mae	Threshold	Anomaly

In [41]:

```
nitrogen_dioxide_predict = pd.DataFrame(X_pred.nitrogen_dioxide)
nitrogen_dioxide_predict
```

Out[41]:

timestamp	nitrogen_dioxide
2014-09-18 19:15:00	0.811599
2014-09-18 19:20:00	0.807454
2014-09-18 19:25:00	0.802057
2014-09-18 19:30:00	0.797581
2014-09-18 19:35:00	0.799815
...	...
2014-09-30 23:40:00	0.768571
2014-09-30 23:45:00	0.771818
2014-09-30 23:50:00	0.765982
2014-09-30 23:55:00	0.762220
2014-10-01 00:00:00	0.756855

3514 rows × 1 columns

In [42]:

```
nitrogen_dioxide_predict['Loss_mae'] = abs(X_pred.nitrogen_dioxide.values - XtestDataframe.
nitrogen_dioxide_predict['Threshold'] = 0.3
nitrogen_dioxide_predict['Anomaly'] = nitrogen_dioxide_predict['Loss_mae'] > nitrogen_dioxi
```

In [43]:

```
nitrogen_dioxide_predict.head(10)
```

Out[43]:

timestamp	nitrogen_dioxide	Loss_mae	Threshold	Anomaly
2014-09-18 19:15:00	0.811599	0.045101	0.3	False
2014-09-18 19:20:00	0.807454	0.020651	0.3	False
2014-09-18 19:25:00	0.802057	0.010178	0.3	False
2014-09-18 19:30:00	0.797581	0.026008	0.3	False
2014-09-18 19:35:00	0.799815	0.007936	0.3	False
2014-09-18 19:40:00	0.805881	0.008926	0.3	False
2014-09-18 19:45:00	0.805572	0.023846	0.3	False
2014-09-18 19:50:00	0.807542	0.005511	0.3	False
2014-09-18 19:55:00	0.813049	0.011018	0.3	False
2014-09-18 20:00:00	0.816539	0.004356	0.3	False

In [44]:

```
anomalies_nitrogen_dioxide = nitrogen_dioxide_predict[nitrogen_dioxide_predict.Anomaly == T
print(anomalies_nitrogen_dioxide.shape)
anomalies_nitrogen_dioxide.head()
```

```
(1001, 4)
```

Out[44]:

timestamp	nitrogen_dioxide	Loss_mae	Threshold	Anomaly
2014-09-23 13:45:00	0.736436	0.310040	0.3	True
2014-09-23 13:50:00	0.731587	0.300115	0.3	True
2014-09-23 13:55:00	0.739876	0.308404	0.3	True
2014-09-26 18:50:00	0.826556	0.313865	0.3	True
2014-09-26 18:55:00	0.826524	0.323986	0.3	True

In [45]:

```
plt.figure(figsize=(15,10))
plt.plot(anomalies_ozone.ozone, color='black', marker="o", linestyle='-')
plt.plot(X_pred.ozone, color='orange', marker='.', linestyle="dashed")

plt.title('ozone için Aykırı Değer Grafiği ')
plt.xlabel('zaman serisi', fontweight='bold')
plt.ylabel('Mutlak Ortala Hata Değerleri', fontweight='bold')
```

Out[45]:

```
Text(0, 0.5, 'Mutlak Ortala Hata Değerleri')
```

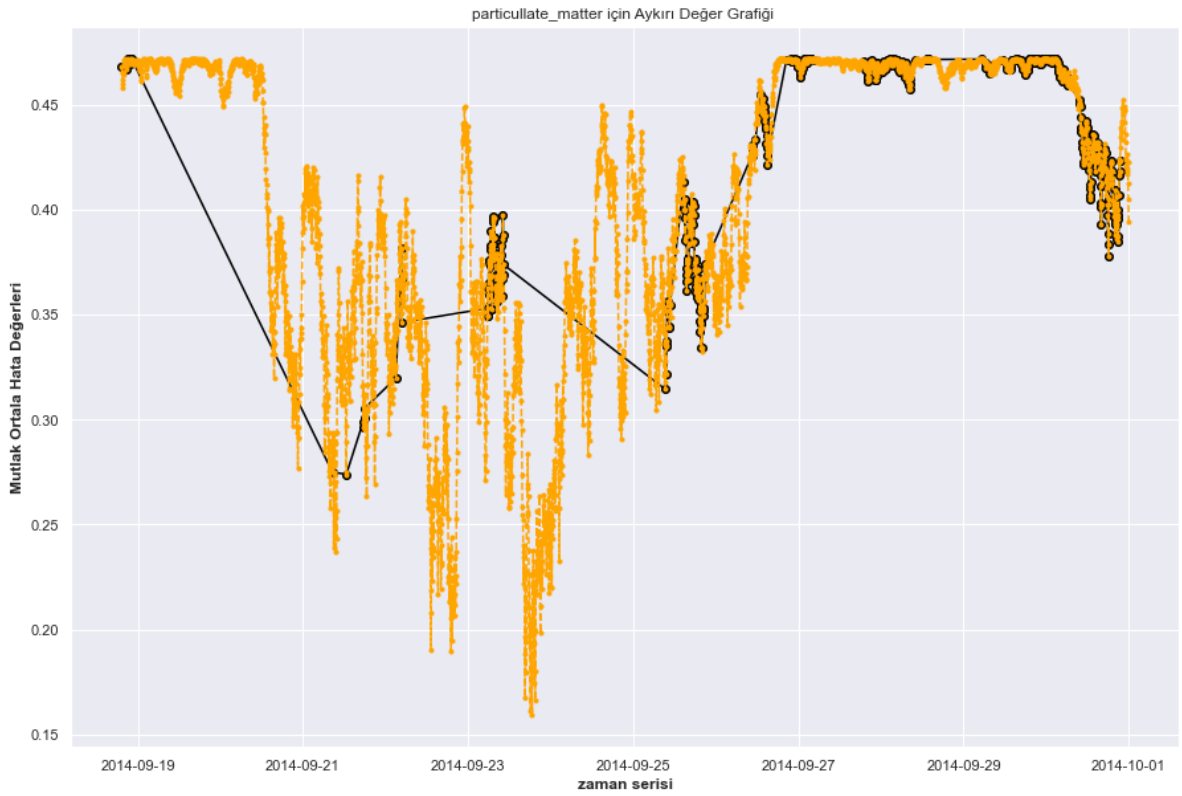
In [46]:

```
plt.figure(figsize=(15,10))
plt.plot(anomalies_particullate_matter.particullate_matter, color='black', marker="o", line
plt.plot(X_pred.particullate_matter, color='orange', marker='.', linestyle="dashed")

plt.title('particullate_matter için Aykırı Değer Grafiği ')
plt.xlabel('zaman serisi', fontweight='bold')
plt.ylabel('Mutlak Ortala Hata Değerleri', fontweight='bold')
```

Out[46]:

```
Text(0, 0.5, 'Mutlak Ortala Hata Değerleri')
```



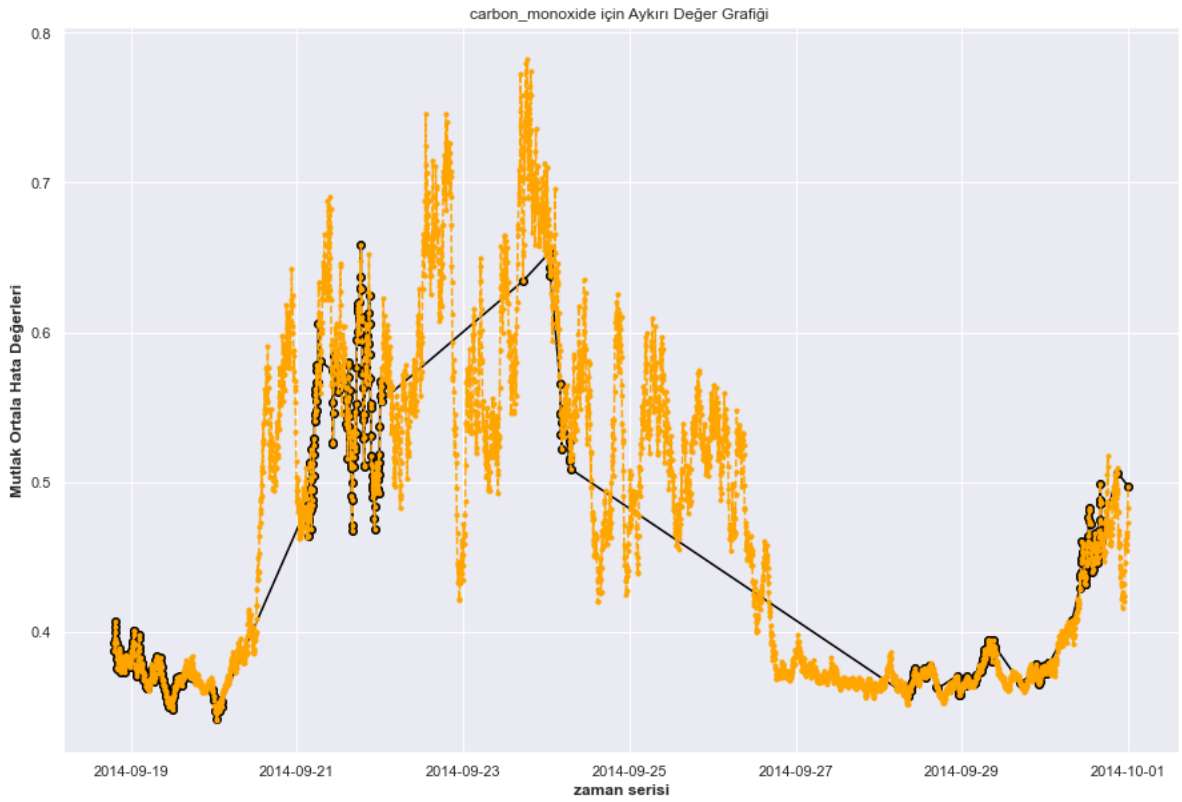
In [47]:

```
plt.figure(figsize=(15,10))
plt.plot(anomalies_carbon_monoxide.carbon_monoxide, color='black', marker="o", linestyle='-
plt.plot(X_pred.carbon_monoxide, color='orange', marker='.', linestyle="dashed")

plt.title('carbon_monoxide için Aykırı Değer Grafiği ')
plt.xlabel('zaman serisi', fontweight='bold')
plt.ylabel('Mutlak Ortala Hata Değerleri', fontweight='bold')
```

Out[47]:

```
Text(0, 0.5, 'Mutlak Ortala Hata Değerleri')
```



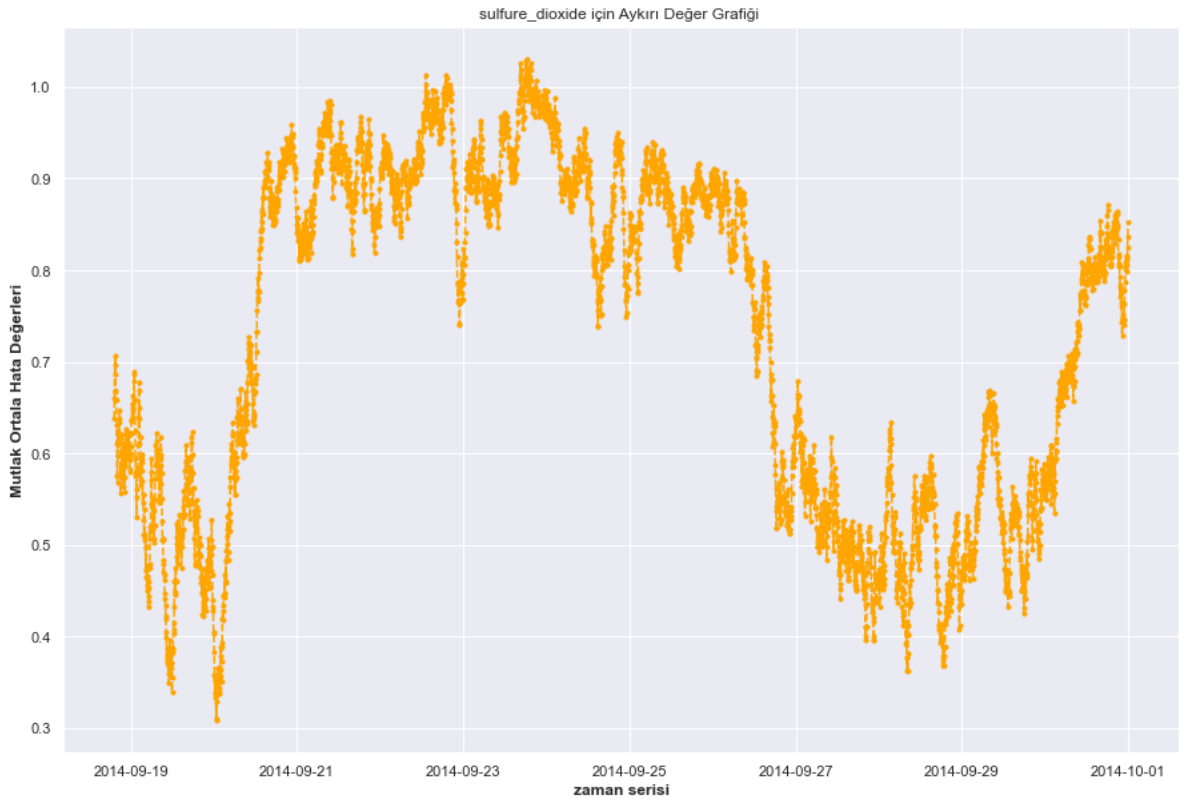
In [48]:

```
plt.figure(figsize=(15,10))
plt.plot(anomalies_sulfure_dioxide.sulfure_dioxide, color='black', marker="o", linestyle='-
plt.plot(X_pred.sulfure_dioxide, color='orange', marker='.', linestyle="dashed")

plt.title('sulfure_dioxide için Aykırı Değer Grafiği ')
plt.xlabel('zaman serisi', fontweight='bold')
plt.ylabel('Mutlak Ortala Hata Değerleri', fontweight='bold')
```

Out[48]:

```
Text(0, 0.5, 'Mutlak Ortala Hata Değerleri')
```



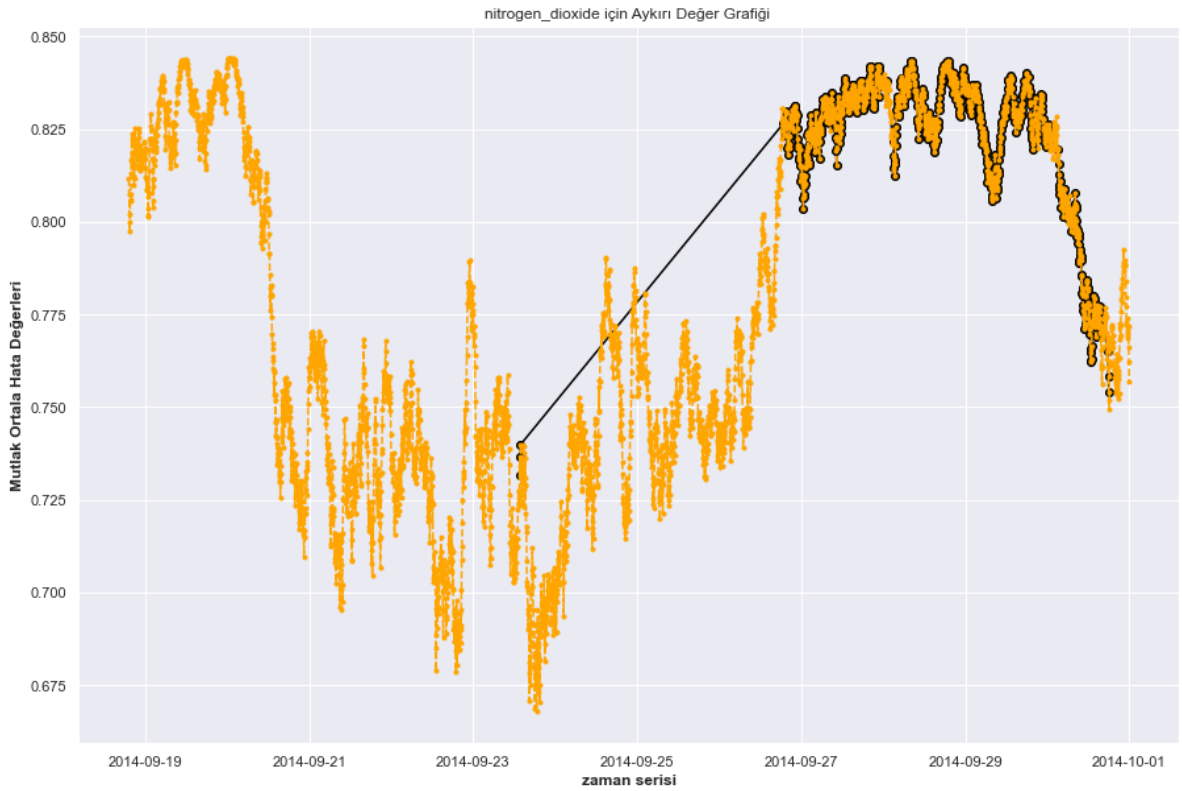
In [49]:

```
plt.figure(figsize=(15,10))
plt.plot(anomalies_nitrogen_dioxide.nitrogen_dioxide, color='black', marker="o", linestyle=
plt.plot(X_pred.nitrogen_dioxide, color='orange', marker='.', linestyle="dashed")

plt.title('nitrogen_dioxide için Aykırı Değer Grafiği ')
plt.xlabel('zaman serisi', fontweight='bold')
plt.ylabel('Mutlak Ortala Hata Değerleri', fontweight='bold')
```

Out[49]:

```
Text(0, 0.5, 'Mutlak Ortala Hata Değerleri')
```



In [ ]: