Import Packages

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import seaborn as sns
from math import sqrt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from keras.models import Sequential, load_model
from keras.layers import Dense, LSTM, Dropout, BatchNormalization
from sklearn.metrics import mean_squared_error as mse
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.losses import MeanSquaredError
from tensorflow.keras.metrics import RootMeanSquaredError
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146:
UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this
version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and
<{np_maxversion}"
```

Load Data

```
df_train = pd.read_csv('/kaggle/input/lstm-datasets-multivariate-
univariate/LSTM-Multivariate_pollution.csv')
df_train
```

```
                      date  pollution  dew  temp   press wnd_dir
wnd_spd  \
0      2010-01-02 00:00:00      129.0  -16  -4.0  1020.0      SE
1.79
1      2010-01-02 01:00:00      148.0  -15  -4.0  1020.0      SE
2.68
2      2010-01-02 02:00:00      159.0  -11  -5.0  1021.0      SE
3.57
3      2010-01-02 03:00:00      181.0   -7  -5.0  1022.0      SE
5.36
4      2010-01-02 04:00:00      138.0   -7  -5.0  1022.0      SE
6.25
...                    ...        ...  ...   ...     ...     ...
...
43795  2014-12-31 19:00:00        8.0  -23  -2.0  1034.0      NW
231.97
43796  2014-12-31 20:00:00       10.0  -22  -3.0  1034.0      NW
```

```
237.78
43797  2014-12-31 21:00:00       10.0  -22  -3.0  1034.0      NW
242.70
43798  2014-12-31 22:00:00        8.0  -22  -4.0  1034.0      NW
246.72
43799  2014-12-31 23:00:00       12.0  -21  -3.0  1034.0      NW
249.85

       snow  rain
0         0     0
1         0     0
2         0     0
3         1     0
4         2     0
...     ...   ...
43795     0     0
43796     0     0
43797     0     0
43798     0     0
43799     0     0

[43800 rows x 9 columns]
```

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43800 entries, 0 to 43799
Data columns (total 9 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   date       43800 non-null  object
 1   pollution  43800 non-null  float64
 2   dew        43800 non-null  int64
 3   temp       43800 non-null  float64
 4   press      43800 non-null  float64
 5   wnd_dir    43800 non-null  object
 6   wnd_spd    43800 non-null  float64
 7   snow       43800 non-null  int64
 8   rain       43800 non-null  int64
dtypes: float64(4), int64(3), object(2)
memory usage: 3.0+ MB
```

```
df_test = pd.read_csv("/kaggle/input/lstm-datasets-multivariate-univariate/pollution_test_data1.csv")
df_test
```

```
     dew  temp  press wnd_dir  wnd_spd  snow  rain  pollution
0    -16     4   1027      SE     3.58     0     0        128
1    -17     5   1027      SE     7.60     0     0         77
2    -16     4   1027      SE     9.39     0     0         65
```

```
3    -16     1   1028      cv     0.89     0     0          79
4    -14     0   1028      NE     1.79     0     0          93
..   ...   ...    ...     ...      ...   ...   ...         ...
341  -23    -2   1034      NW   231.97     0     0           8
342  -22    -3   1034      NW   237.78     0     0          10
343  -22    -3   1034      NW   242.70     0     0          10
344  -22    -4   1034      NW   246.72     0     0           8
345  -21    -3   1034      NW   249.85     0     0          12

[346 rows x 8 columns]
```

```python
# Checking null values
print(df_train.isnull().sum() , "\n---------------- \n" ,
df_test.isnull().sum() )
```

```
date         0
pollution    0
dew          0
temp         0
press        0
wnd_dir      0
wnd_spd      0
snow         0
rain         0
dtype: int64
----------------
 dew          0
temp         0
press        0
wnd_dir      0
wnd_spd      0
snow         0
rain         0
pollution    0
dtype: int64
```

```python
df_train.describe()
```

```
          pollution           dew          temp         press
wnd_spd  \
count  43800.000000  43800.000000  43800.000000  43800.000000
43800.000000
mean      94.013516      1.828516     12.459041   1016.447306
23.894307
std       92.252276     14.429326     12.193384     10.271411
50.022729
min        0.000000    -40.000000    -19.000000    991.000000
0.450000
25%       24.000000    -10.000000      2.000000   1008.000000
1.790000
```

```
50%       68.000000      2.000000     14.000000   1016.000000
5.370000
75%      132.250000     15.000000     23.000000   1025.000000
21.910000
max      994.000000     28.000000     42.000000   1046.000000
585.600000

               snow          rain
count  43800.000000  43800.000000
mean       0.052763      0.195023
std        0.760582      1.416247
min        0.000000      0.000000
25%        0.000000      0.000000
50%        0.000000      0.000000
75%        0.000000      0.000000
max       27.000000     36.000000
```

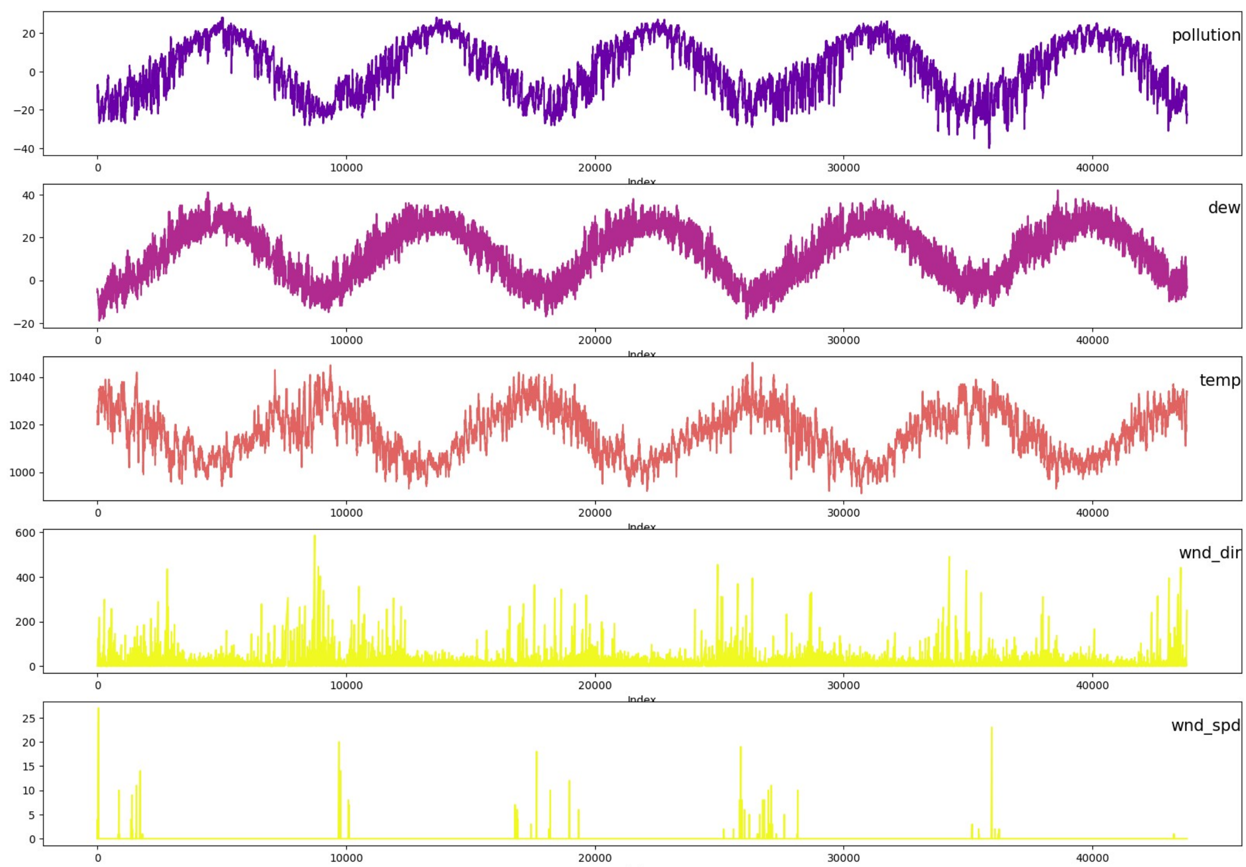
Data vislualization and Feature scaling

```python
df_train_scaled = df_train.copy()
df_test_scaled = df_test.copy()

# Define the mapping dictionary
mapping = {'NE': 0, 'SE': 1, 'NW': 2, 'cv': 3}

# Replace the string values with numerical values
df_train_scaled['wnd_dir'] = df_train_scaled['wnd_dir'].map(mapping)
df_test_scaled['wnd_dir'] = df_test_scaled['wnd_dir'].map(mapping)

df_train_scaled['date'] = pd.to_datetime(df_train_scaled['date'])
# Resetting the index
df_train_scaled.set_index('date', inplace=True)
df_train_scaled.head()
```

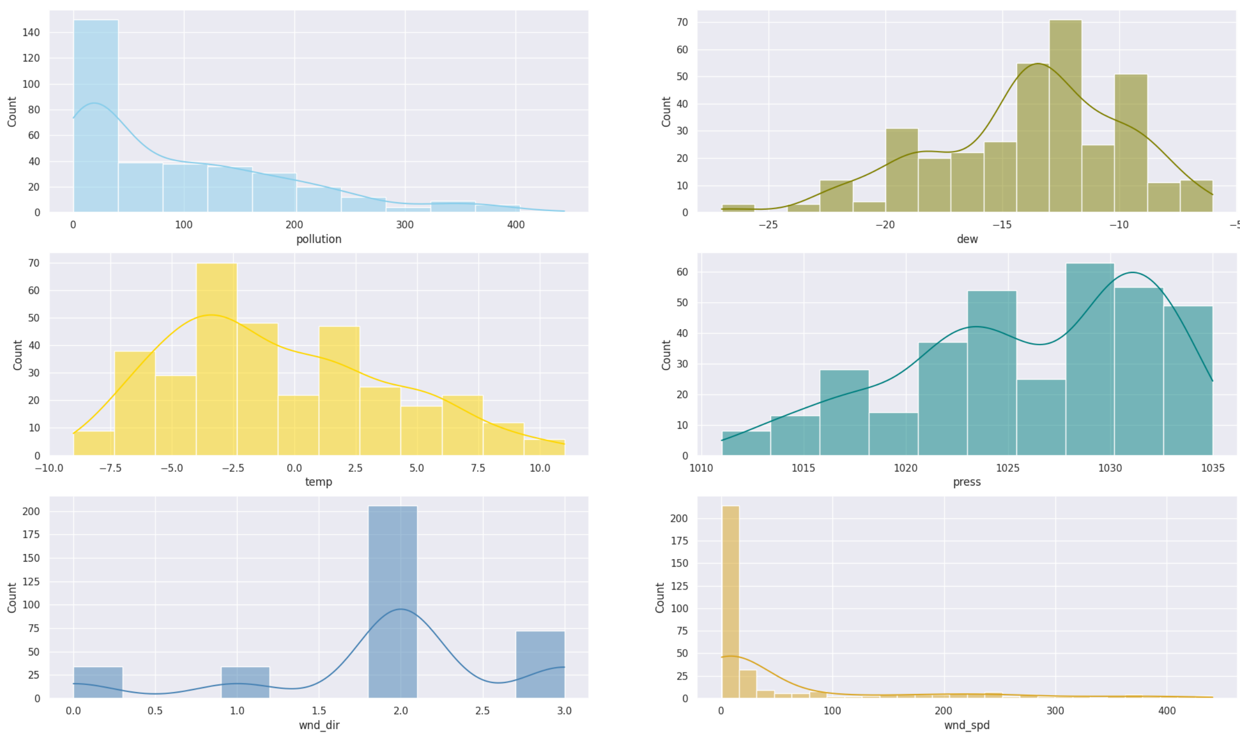
```
                     pollution  dew  temp   press  wnd_dir  wnd_spd
snow  \
date

2010-01-02 00:00:00      129.0  -16  -4.0  1020.0        1     1.79
0
2010-01-02 01:00:00      148.0  -15  -4.0  1020.0        1     2.68
0
2010-01-02 02:00:00      159.0  -11  -5.0  1021.0        1     3.57
0
2010-01-02 03:00:00      181.0   -7  -5.0  1022.0        1     5.36
1
2010-01-02 04:00:00      138.0   -7  -5.0  1022.0        1     6.25
2

                     rain
```

```
date
2010-01-02 00:00:00     0
2010-01-02 01:00:00     0
2010-01-02 02:00:00     0
2010-01-02 03:00:00     0
2010-01-02 04:00:00     0
```

```python
values = df_train_scaled.values

# specify columns to plot
groups = [1, 2, 3, 5, 6]
i = 1

# plot each column
plt.figure(figsize=(20,14))
for group in groups:
    plt.subplot(len(groups), 1, i)
    plt.plot(values[:, group], color=cm.plasma(group/len(groups)))
    plt.xlabel('Index')
    plt.title(df_train.columns[group], y=0.75, loc='right', fontsize = 15)
    i += 1
plt.show()
```



```python
sns.set(style="darkgrid")

fig, axs = plt.subplots(3,2, figsize=(24,14))

sns.histplot(data=df_test_scaled, x="pollution", kde=True,
color="skyblue", ax=axs[0, 0])
sns.histplot(data=df_test_scaled, x="dew", kde=True, color="olive",
ax=axs[0, 1])
sns.histplot(data=df_test_scaled, x="temp", kde=True, color="gold",
ax=axs[1, 0])
sns.histplot(data=df_test_scaled, x="press", kde=True, color="teal",
ax=axs[1, 1])
sns.histplot(data=df_test_scaled, x="wnd_dir", kde=True,
color="steelblue", ax=axs[2, 0])
sns.histplot(data=df_test_scaled, x="wnd_spd", kde=True,
color="goldenrod", ax=axs[2, 1])

plt.show()
```




```python
scaler = MinMaxScaler()

# Define the columns to scale
columns = (['pollution', 'dew', 'temp', 'press', "wnd_dir", 'wnd_spd',
        'snow', 'rain'])

df_test_scaled = df_test_scaled[columns]
```

```
# Scale the selected columns to the range 0-1
df_train_scaled[columns] =
scaler.fit_transform(df_train_scaled[columns])
df_test_scaled[columns] = scaler.transform(df_test_scaled[columns])

# Show the scaled data
df_train_scaled.head()
```

```
                     pollution       dew      temp     press   wnd_dir
\
date

2010-01-02 00:00:00   0.129779  0.352941  0.245902  0.527273  0.333333

2010-01-02 01:00:00   0.148893  0.367647  0.245902  0.527273  0.333333

2010-01-02 02:00:00   0.159960  0.426471  0.229508  0.545455  0.333333

2010-01-02 03:00:00   0.182093  0.485294  0.229508  0.563636  0.333333

2010-01-02 04:00:00   0.138833  0.485294  0.229508  0.563636  0.333333


                      wnd_spd      snow  rain
date
2010-01-02 00:00:00  0.002290  0.000000   0.0
2010-01-02 01:00:00  0.003811  0.000000   0.0
2010-01-02 02:00:00  0.005332  0.000000   0.0
2010-01-02 03:00:00  0.008391  0.037037   0.0
2010-01-02 04:00:00  0.009912  0.074074   0.0
```

```
df_test_scaled.head()
```

```
   pollution       dew      temp     press   wnd_dir   wnd_spd  snow
rain
0   0.128773  0.352941  0.377049  0.654545  0.333333  0.005349   0.0
0.0
1   0.077465  0.338235  0.393443  0.654545  0.333333  0.012219   0.0
0.0
2   0.065392  0.352941  0.377049  0.654545  0.333333  0.015278   0.0
0.0
3   0.079477  0.352941  0.327869  0.672727  1.000000  0.000752   0.0
0.0
4   0.093561  0.382353  0.311475  0.672727  0.000000  0.002290   0.0
0.0
```

Split the data into training and test sets

```
df_train_scaled = np.array(df_train_scaled)
df_test_scaled = np.array(df_test_scaled)
```

```
X = []
y = []
n_future = 1
n_past = 11

#  Train Sets
for i in range(n_past, len(df_train_scaled) - n_future+1):
    X.append(df_train_scaled[i - n_past:i,
1:df_train_scaled.shape[1]])
    y.append(df_train_scaled[i + n_future - 1:i + n_future, 0])
X_train, y_train = np.array(X), np.array(y)

#  Test Sets

X = []
y = []
for i in range(n_past, len(df_test_scaled) - n_future+1):
    X.append(df_test_scaled[i - n_past:i, 1:df_test_scaled.shape[1]])
    y.append(df_test_scaled[i + n_future - 1:i + n_future, 0])
X_test, y_test = np.array(X), np.array(y)

print('X_train shape : {}   y_train shape : {} \n'
      'X_test shape : {}      y_test shape : {}
'.format(X_train.shape, y_train.shape, X_test.shape, y_test.shape))

X_train shape : (43789, 11, 7)   y_train shape : (43789, 1)
X_test shape : (335, 11, 7)      y_test shape : (335, 1)
```

Create a LSTM model

```
# design network

model = Sequential()
model.add(LSTM(32, input_shape=(X_train.shape[1], X_train.shape[2]),
return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(16, return_sequences=False))
model.add(Dense(y_train.shape[1]))

# Compile the model
model.compile(loss='mse', optimizer=Adam(learning_rate=0.001),
metrics=[RootMeanSquaredError()])

# Define callbacks for avoiding overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)
checkpoint = ModelCheckpoint('best_model.h5', monitor='val_loss',
save_best_only=True)
```

```
model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 11, 32)            5120

 dropout (Dropout)           (None, 11, 32)            0

 lstm_1 (LSTM)               (None, 16)                3136

 dense (Dense)               (None, 1)                 17

=================================================================
Total params: 8,273
Trainable params: 8,273
Non-trainable params: 0
_________________________________________________________________
```

```
# fit network
history = model.fit(X_train, y_train, epochs=150, batch_size=32,
validation_split=0.1, callbacks=[early_stopping, checkpoint],
shuffle=False)

Epoch 1/150
1232/1232 [==============================] - 15s 6ms/step - loss:
0.0074 - root_mean_squared_error: 0.0861 - val_loss: 0.0115 -
val_root_mean_squared_error: 0.1071
Epoch 2/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0067 - root_mean_squared_error: 0.0819 - val_loss: 0.0109 -
val_root_mean_squared_error: 0.1043
Epoch 3/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0065 - root_mean_squared_error: 0.0804 - val_loss: 0.0105 -
val_root_mean_squared_error: 0.1026
Epoch 4/150
1232/1232 [==============================] - 7s 6ms/step - loss:
0.0063 - root_mean_squared_error: 0.0794 - val_loss: 0.0109 -
val_root_mean_squared_error: 0.1043
Epoch 5/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0062 - root_mean_squared_error: 0.0785 - val_loss: 0.0110 -
val_root_mean_squared_error: 0.1049
Epoch 6/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0061 - root_mean_squared_error: 0.0778 - val_loss: 0.0099 -
val_root_mean_squared_error: 0.0993
```

```
Epoch 7/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0059 - root_mean_squared_error: 0.0767 - val_loss: 0.0098 -
val_root_mean_squared_error: 0.0989
Epoch 8/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0058 - root_mean_squared_error: 0.0759 - val_loss: 0.0107 -
val_root_mean_squared_error: 0.1036
Epoch 9/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0057 - root_mean_squared_error: 0.0757 - val_loss: 0.0093 -
val_root_mean_squared_error: 0.0964
Epoch 10/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0057 - root_mean_squared_error: 0.0755 - val_loss: 0.0084 -
val_root_mean_squared_error: 0.0914
Epoch 11/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0057 - root_mean_squared_error: 0.0753 - val_loss: 0.0076 -
val_root_mean_squared_error: 0.0872
Epoch 12/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0056 - root_mean_squared_error: 0.0750 - val_loss: 0.0070 -
val_root_mean_squared_error: 0.0839
Epoch 13/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0056 - root_mean_squared_error: 0.0749 - val_loss: 0.0065 -
val_root_mean_squared_error: 0.0808
Epoch 14/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0056 - root_mean_squared_error: 0.0746 - val_loss: 0.0063 -
val_root_mean_squared_error: 0.0791
Epoch 15/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0056 - root_mean_squared_error: 0.0745 - val_loss: 0.0063 -
val_root_mean_squared_error: 0.0792
Epoch 16/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0055 - root_mean_squared_error: 0.0743 - val_loss: 0.0060 -
val_root_mean_squared_error: 0.0774
Epoch 17/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0055 - root_mean_squared_error: 0.0741 - val_loss: 0.0059 -
val_root_mean_squared_error: 0.0769
Epoch 18/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0055 - root_mean_squared_error: 0.0741 - val_loss: 0.0057 -
val_root_mean_squared_error: 0.0757
Epoch 19/150
```

```
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0055 - root_mean_squared_error: 0.0740 - val_loss: 0.0057 -
val_root_mean_squared_error: 0.0753
Epoch 20/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0054 - root_mean_squared_error: 0.0738 - val_loss: 0.0056 -
val_root_mean_squared_error: 0.0750
Epoch 21/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0054 - root_mean_squared_error: 0.0738 - val_loss: 0.0055 -
val_root_mean_squared_error: 0.0740
Epoch 22/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0054 - root_mean_squared_error: 0.0737 - val_loss: 0.0054 -
val_root_mean_squared_error: 0.0735
Epoch 23/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0054 - root_mean_squared_error: 0.0736 - val_loss: 0.0054 -
val_root_mean_squared_error: 0.0732
Epoch 24/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0054 - root_mean_squared_error: 0.0736 - val_loss: 0.0053 -
val_root_mean_squared_error: 0.0727
Epoch 25/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0054 - root_mean_squared_error: 0.0735 - val_loss: 0.0053 -
val_root_mean_squared_error: 0.0727
Epoch 26/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0054 - root_mean_squared_error: 0.0734 - val_loss: 0.0051 -
val_root_mean_squared_error: 0.0715
Epoch 27/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0054 - root_mean_squared_error: 0.0732 - val_loss: 0.0051 -
val_root_mean_squared_error: 0.0716
Epoch 28/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0054 - root_mean_squared_error: 0.0732 - val_loss: 0.0051 -
val_root_mean_squared_error: 0.0713
Epoch 29/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0731 - val_loss: 0.0050 -
val_root_mean_squared_error: 0.0711
Epoch 30/150
1232/1232 [==============================] - 7s 6ms/step - loss:
0.0053 - root_mean_squared_error: 0.0731 - val_loss: 0.0050 -
val_root_mean_squared_error: 0.0709
Epoch 31/150
1232/1232 [==============================] - 7s 5ms/step - loss:
```

```
0.0053 - root_mean_squared_error: 0.0731 - val_loss: 0.0050 -
val_root_mean_squared_error: 0.0708
Epoch 32/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0730 - val_loss: 0.0050 -
val_root_mean_squared_error: 0.0704
Epoch 33/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0729 - val_loss: 0.0050 -
val_root_mean_squared_error: 0.0706
Epoch 34/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0729 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0700
Epoch 35/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0728 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0698
Epoch 36/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0727 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0701
Epoch 37/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0728 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0699
Epoch 38/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0727 - val_loss: 0.0048 -
val_root_mean_squared_error: 0.0696
Epoch 39/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0725 - val_loss: 0.0048 -
val_root_mean_squared_error: 0.0691
Epoch 40/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0726 - val_loss: 0.0048 -
val_root_mean_squared_error: 0.0690
Epoch 41/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0725 - val_loss: 0.0048 -
val_root_mean_squared_error: 0.0694
Epoch 42/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0725 - val_loss: 0.0047 -
val_root_mean_squared_error: 0.0689
Epoch 43/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0052 - root_mean_squared_error: 0.0724 - val_loss: 0.0047 -
```

```
val_root_mean_squared_error: 0.0689
Epoch 44/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0052 - root_mean_squared_error: 0.0724 - val_loss: 0.0047 -
val_root_mean_squared_error: 0.0686
Epoch 45/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0052 - root_mean_squared_error: 0.0723 - val_loss: 0.0047 -
val_root_mean_squared_error: 0.0686
Epoch 46/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0052 - root_mean_squared_error: 0.0723 - val_loss: 0.0046 -
val_root_mean_squared_error: 0.0681
Epoch 47/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0052 - root_mean_squared_error: 0.0722 - val_loss: 0.0046 -
val_root_mean_squared_error: 0.0681
Epoch 48/150
1232/1232 [==============================] - 7s 6ms/step - loss:
0.0052 - root_mean_squared_error: 0.0721 - val_loss: 0.0046 -
val_root_mean_squared_error: 0.0681
Epoch 49/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0052 - root_mean_squared_error: 0.0722 - val_loss: 0.0046 -
val_root_mean_squared_error: 0.0679
Epoch 50/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0052 - root_mean_squared_error: 0.0720 - val_loss: 0.0046 -
val_root_mean_squared_error: 0.0677
Epoch 51/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0052 - root_mean_squared_error: 0.0719 - val_loss: 0.0046 -
val_root_mean_squared_error: 0.0676
Epoch 52/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0052 - root_mean_squared_error: 0.0719 - val_loss: 0.0046 -
val_root_mean_squared_error: 0.0676
Epoch 53/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0052 - root_mean_squared_error: 0.0718 - val_loss: 0.0045 -
val_root_mean_squared_error: 0.0673
Epoch 54/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0051 - root_mean_squared_error: 0.0716 - val_loss: 0.0045 -
val_root_mean_squared_error: 0.0675
Epoch 55/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0051 - root_mean_squared_error: 0.0716 - val_loss: 0.0045 -
val_root_mean_squared_error: 0.0673
```

```
Epoch 56/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0051 - root_mean_squared_error: 0.0717 - val_loss: 0.0045 -
val_root_mean_squared_error: 0.0673
Epoch 57/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0051 - root_mean_squared_error: 0.0715 - val_loss: 0.0045 -
val_root_mean_squared_error: 0.0672
Epoch 58/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0051 - root_mean_squared_error: 0.0714 - val_loss: 0.0045 -
val_root_mean_squared_error: 0.0674
Epoch 59/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0051 - root_mean_squared_error: 0.0714 - val_loss: 0.0045 -
val_root_mean_squared_error: 0.0672
Epoch 60/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0051 - root_mean_squared_error: 0.0714 - val_loss: 0.0045 -
val_root_mean_squared_error: 0.0672
Epoch 61/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0051 - root_mean_squared_error: 0.0714 - val_loss: 0.0045 -
val_root_mean_squared_error: 0.0668
Epoch 62/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0051 - root_mean_squared_error: 0.0712 - val_loss: 0.0045 -
val_root_mean_squared_error: 0.0672
Epoch 63/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0051 - root_mean_squared_error: 0.0713 - val_loss: 0.0045 -
val_root_mean_squared_error: 0.0668
Epoch 64/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0050 - root_mean_squared_error: 0.0710 - val_loss: 0.0045 -
val_root_mean_squared_error: 0.0671
Epoch 65/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0051 - root_mean_squared_error: 0.0711 - val_loss: 0.0045 -
val_root_mean_squared_error: 0.0669
Epoch 66/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0050 - root_mean_squared_error: 0.0709 - val_loss: 0.0045 -
val_root_mean_squared_error: 0.0668
Epoch 67/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0050 - root_mean_squared_error: 0.0708 - val_loss: 0.0044 -
val_root_mean_squared_error: 0.0667
Epoch 68/150
```

```
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0050 - root_mean_squared_error: 0.0709 - val_loss: 0.0044 -
val_root_mean_squared_error: 0.0667
Epoch 69/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0050 - root_mean_squared_error: 0.0708 - val_loss: 0.0044 -
val_root_mean_squared_error: 0.0665
Epoch 70/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0050 - root_mean_squared_error: 0.0709 - val_loss: 0.0044 -
val_root_mean_squared_error: 0.0664
Epoch 71/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0050 - root_mean_squared_error: 0.0708 - val_loss: 0.0044 -
val_root_mean_squared_error: 0.0663
Epoch 72/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0050 - root_mean_squared_error: 0.0707 - val_loss: 0.0044 -
val_root_mean_squared_error: 0.0664
Epoch 73/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0050 - root_mean_squared_error: 0.0706 - val_loss: 0.0044 -
val_root_mean_squared_error: 0.0664
Epoch 74/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0050 - root_mean_squared_error: 0.0704 - val_loss: 0.0044 -
val_root_mean_squared_error: 0.0662
Epoch 75/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0049 - root_mean_squared_error: 0.0704 - val_loss: 0.0044 -
val_root_mean_squared_error: 0.0661
Epoch 76/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0050 - root_mean_squared_error: 0.0704 - val_loss: 0.0044 -
val_root_mean_squared_error: 0.0661
Epoch 77/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0049 - root_mean_squared_error: 0.0703 - val_loss: 0.0044 -
val_root_mean_squared_error: 0.0660
Epoch 78/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0049 - root_mean_squared_error: 0.0703 - val_loss: 0.0044 -
val_root_mean_squared_error: 0.0660
Epoch 79/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0049 - root_mean_squared_error: 0.0701 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0659
Epoch 80/150
1232/1232 [==============================] - 6s 5ms/step - loss:
```

```
0.0049 - root_mean_squared_error: 0.0702 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0658
Epoch 81/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0049 - root_mean_squared_error: 0.0700 - val_loss: 0.0044 -
val_root_mean_squared_error: 0.0660
Epoch 82/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0049 - root_mean_squared_error: 0.0700 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0659
Epoch 83/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0049 - root_mean_squared_error: 0.0700 - val_loss: 0.0044 -
val_root_mean_squared_error: 0.0660
Epoch 84/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0049 - root_mean_squared_error: 0.0698 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0659
Epoch 85/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0049 - root_mean_squared_error: 0.0697 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0657
Epoch 86/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0048 - root_mean_squared_error: 0.0696 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0655
Epoch 87/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0049 - root_mean_squared_error: 0.0697 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0654
Epoch 88/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0048 - root_mean_squared_error: 0.0696 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0654
Epoch 89/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0048 - root_mean_squared_error: 0.0694 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0656
Epoch 90/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0048 - root_mean_squared_error: 0.0694 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0654
Epoch 91/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0048 - root_mean_squared_error: 0.0693 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0655
Epoch 92/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0048 - root_mean_squared_error: 0.0694 - val_loss: 0.0043 -
```

```
val_root_mean_squared_error: 0.0653
Epoch 93/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0048 - root_mean_squared_error: 0.0692 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0655
Epoch 94/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0048 - root_mean_squared_error: 0.0691 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0652
Epoch 95/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0047 - root_mean_squared_error: 0.0689 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0652
Epoch 96/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0047 - root_mean_squared_error: 0.0689 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0653
Epoch 97/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0047 - root_mean_squared_error: 0.0688 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0652
Epoch 98/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0047 - root_mean_squared_error: 0.0686 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0653
Epoch 99/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0047 - root_mean_squared_error: 0.0684 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0651
Epoch 100/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0047 - root_mean_squared_error: 0.0684 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0653
Epoch 101/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0047 - root_mean_squared_error: 0.0683 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0652
Epoch 102/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0047 - root_mean_squared_error: 0.0682 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0650
Epoch 103/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0046 - root_mean_squared_error: 0.0681 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0654
Epoch 104/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0046 - root_mean_squared_error: 0.0681 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0651
```

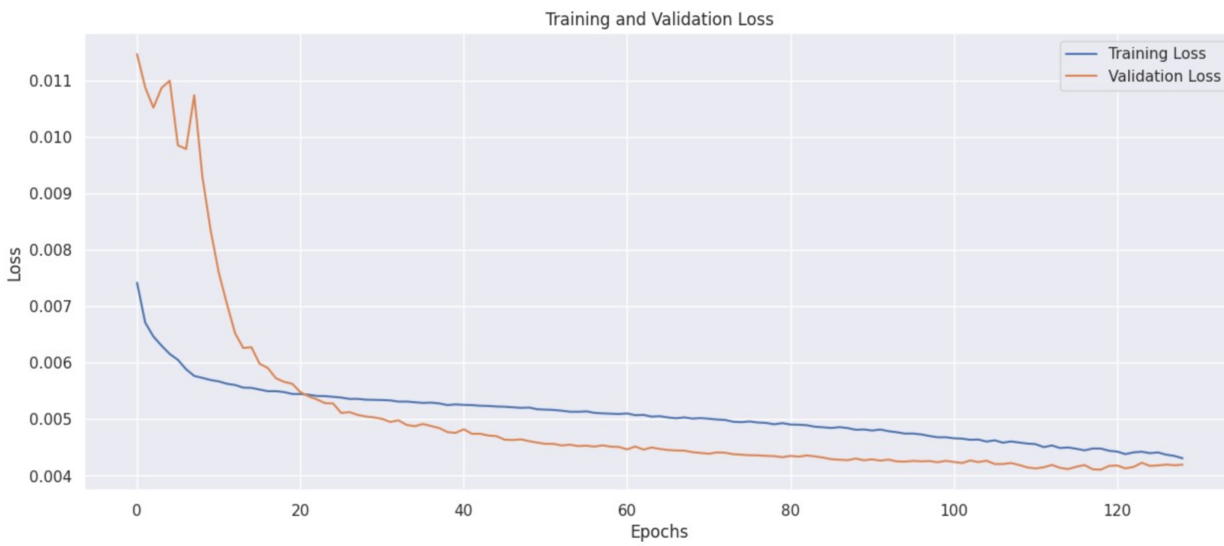
```
Epoch 105/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0046 - root_mean_squared_error: 0.0679 - val_loss: 0.0043 -
val_root_mean_squared_error: 0.0653
Epoch 106/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0046 - root_mean_squared_error: 0.0680 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0649
Epoch 107/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0046 - root_mean_squared_error: 0.0677 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0649
Epoch 108/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0046 - root_mean_squared_error: 0.0679 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0650
Epoch 109/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0046 - root_mean_squared_error: 0.0677 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0647
Epoch 110/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0046 - root_mean_squared_error: 0.0676 - val_loss: 0.0041 -
val_root_mean_squared_error: 0.0644
Epoch 111/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0046 - root_mean_squared_error: 0.0675 - val_loss: 0.0041 -
val_root_mean_squared_error: 0.0643
Epoch 112/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0045 - root_mean_squared_error: 0.0671 - val_loss: 0.0041 -
val_root_mean_squared_error: 0.0644
Epoch 113/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0045 - root_mean_squared_error: 0.0673 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0647
Epoch 114/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0045 - root_mean_squared_error: 0.0670 - val_loss: 0.0041 -
val_root_mean_squared_error: 0.0643
Epoch 115/150
1232/1232 [==============================] - 7s 6ms/step - loss:
0.0045 - root_mean_squared_error: 0.0671 - val_loss: 0.0041 -
val_root_mean_squared_error: 0.0642
Epoch 116/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0045 - root_mean_squared_error: 0.0669 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0645
Epoch 117/150
```

```
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0045 - root_mean_squared_error: 0.0667 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0647
Epoch 118/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0045 - root_mean_squared_error: 0.0669 - val_loss: 0.0041 -
val_root_mean_squared_error: 0.0641
Epoch 119/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0045 - root_mean_squared_error: 0.0669 - val_loss: 0.0041 -
val_root_mean_squared_error: 0.0641
Epoch 120/150
1232/1232 [==============================] - 7s 6ms/step - loss:
0.0044 - root_mean_squared_error: 0.0667 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0646
Epoch 121/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0044 - root_mean_squared_error: 0.0665 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0647
Epoch 122/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0044 - root_mean_squared_error: 0.0662 - val_loss: 0.0041 -
val_root_mean_squared_error: 0.0643
Epoch 123/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0044 - root_mean_squared_error: 0.0664 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0645
Epoch 124/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0044 - root_mean_squared_error: 0.0665 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0650
Epoch 125/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0044 - root_mean_squared_error: 0.0663 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0646
Epoch 126/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0044 - root_mean_squared_error: 0.0664 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0647
Epoch 127/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0044 - root_mean_squared_error: 0.0661 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0648
Epoch 128/150
1232/1232 [==============================] - 6s 5ms/step - loss:
0.0044 - root_mean_squared_error: 0.0660 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0647
Epoch 129/150
1232/1232 [==============================] - 7s 5ms/step - loss:
```

```
0.0043 - root_mean_squared_error: 0.0656 - val_loss: 0.0042 -
val_root_mean_squared_error: 0.0648
```

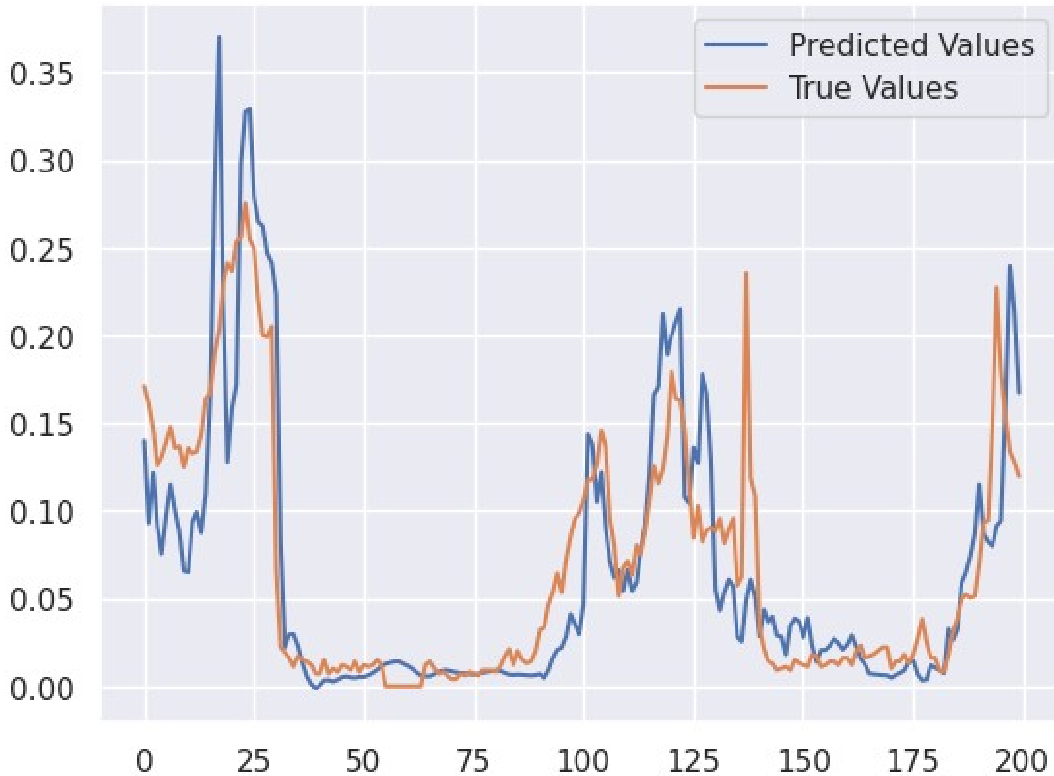
Evaluate the Model

```
# Load the best model
best_model = load_model('best_model.h5')

plt.figure(figsize=(15,6))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```




```
test_predictions = best_model.predict(X_test).flatten()
test_results = pd.DataFrame(data={'Train Predictions':
test_predictions,
                                  'Actual':y_test.flatten()})
test_results.head()

11/11 [==============================] - 1s 2ms/step

   Train Predictions    Actual
0           0.139874  0.171026
1           0.093021  0.160966
2           0.121613  0.146881
3           0.091181  0.125755
4           0.075569  0.130785
```

```
plt.plot(test_results['Train Predictions'][:200], label='Predicted
Values')
plt.plot(test_results['Actual'][:200], label='True Values')
plt.legend()
plt.show()
```




```
rmse = sqrt(mse(y_test, test_predictions))
print('Test RMSE: %.5f' % rmse)

Test RMSE: 0.06954
```

Experementing with variables

```
# design network

model_2 = Sequential()
model_2.add(LSTM(64, input_shape=(X_train.shape[1], X_train.shape[2]),
return_sequences=True))
model_2.add(Dropout(0.2))
model_2.add(LSTM(32, return_sequences=False))
model_2.add(Dense(y_train.shape[1]))

# Compile the model
model_2.compile(loss='mse', optimizer=Adam(learning_rate=0.0005),
```

```
metrics=[RootMeanSquaredError()])

# Define callbacks for avoiding overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)
checkpoint = ModelCheckpoint('best_model.h5', monitor='val_loss',
save_best_only=True)

model_2.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 lstm (LSTM)                 (None, 11, 32)            5120      
                                                                 
 dropout (Dropout)           (None, 11, 32)            0         
                                                                 
 lstm_1 (LSTM)               (None, 16)                3136      
                                                                 
 dense (Dense)               (None, 1)                 17        
                                                                 
=================================================================
Total params: 8,273
Trainable params: 8,273
Non-trainable params: 0
_________________________________________________________________
```

```
# fit network
history = model_2.fit(X_train, y_train, epochs=150, batch_size=32,
validation_split=0.1, callbacks=[early_stopping, checkpoint],
shuffle=False)
```

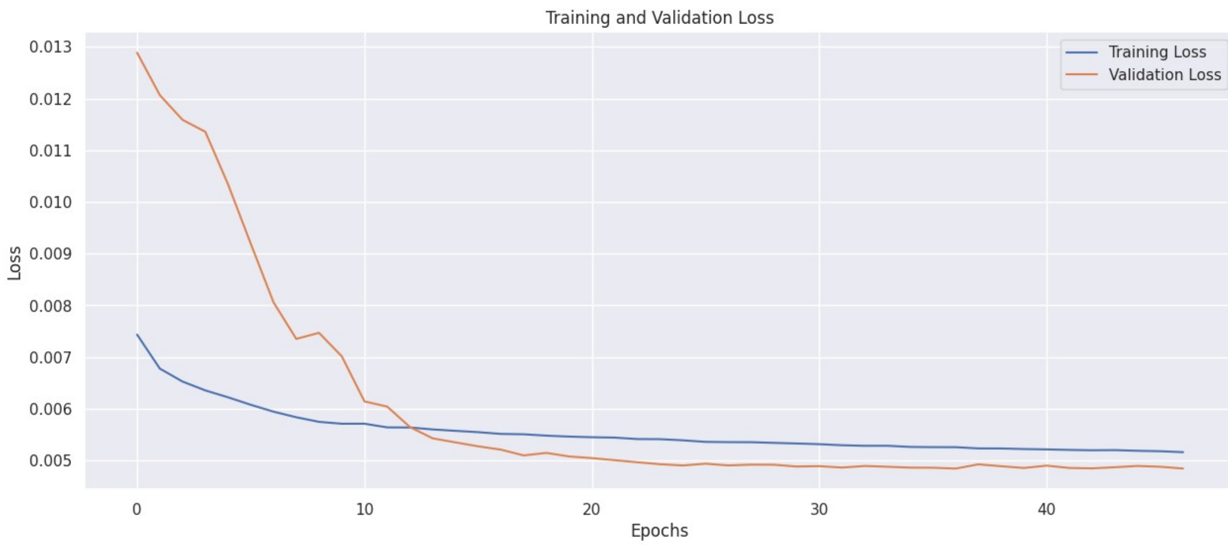
```
Epoch 1/150
1232/1232 [==============================] - 11s 6ms/step - loss:
0.0074 - root_mean_squared_error: 0.0862 - val_loss: 0.0129 -
val_root_mean_squared_error: 0.1135
Epoch 2/150
1232/1232 [==============================] - 7s 6ms/step - loss:
0.0068 - root_mean_squared_error: 0.0823 - val_loss: 0.0121 -
val_root_mean_squared_error: 0.1098
Epoch 3/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0065 - root_mean_squared_error: 0.0808 - val_loss: 0.0116 -
val_root_mean_squared_error: 0.1077
Epoch 4/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0064 - root_mean_squared_error: 0.0797 - val_loss: 0.0114 -
val_root_mean_squared_error: 0.1066
Epoch 5/150
```

```
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0062 - root_mean_squared_error: 0.0789 - val_loss: 0.0103 -
val_root_mean_squared_error: 0.1017
Epoch 6/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0061 - root_mean_squared_error: 0.0780 - val_loss: 0.0092 -
val_root_mean_squared_error: 0.0959
Epoch 7/150
1232/1232 [==============================] - 7s 6ms/step - loss:
0.0059 - root_mean_squared_error: 0.0771 - val_loss: 0.0081 -
val_root_mean_squared_error: 0.0898
Epoch 8/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0058 - root_mean_squared_error: 0.0764 - val_loss: 0.0074 -
val_root_mean_squared_error: 0.0857
Epoch 9/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0057 - root_mean_squared_error: 0.0758 - val_loss: 0.0075 -
val_root_mean_squared_error: 0.0864
Epoch 10/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0057 - root_mean_squared_error: 0.0756 - val_loss: 0.0070 -
val_root_mean_squared_error: 0.0838
Epoch 11/150
1232/1232 [==============================] - 7s 6ms/step - loss:
0.0057 - root_mean_squared_error: 0.0756 - val_loss: 0.0061 -
val_root_mean_squared_error: 0.0784
Epoch 12/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0056 - root_mean_squared_error: 0.0751 - val_loss: 0.0060 -
val_root_mean_squared_error: 0.0777
Epoch 13/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0056 - root_mean_squared_error: 0.0751 - val_loss: 0.0056 -
val_root_mean_squared_error: 0.0751
Epoch 14/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0056 - root_mean_squared_error: 0.0748 - val_loss: 0.0054 -
val_root_mean_squared_error: 0.0737
Epoch 15/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0056 - root_mean_squared_error: 0.0747 - val_loss: 0.0054 -
val_root_mean_squared_error: 0.0731
Epoch 16/150
1232/1232 [==============================] - 7s 6ms/step - loss:
0.0055 - root_mean_squared_error: 0.0745 - val_loss: 0.0053 -
val_root_mean_squared_error: 0.0726
Epoch 17/150
1232/1232 [==============================] - 7s 5ms/step - loss:
```

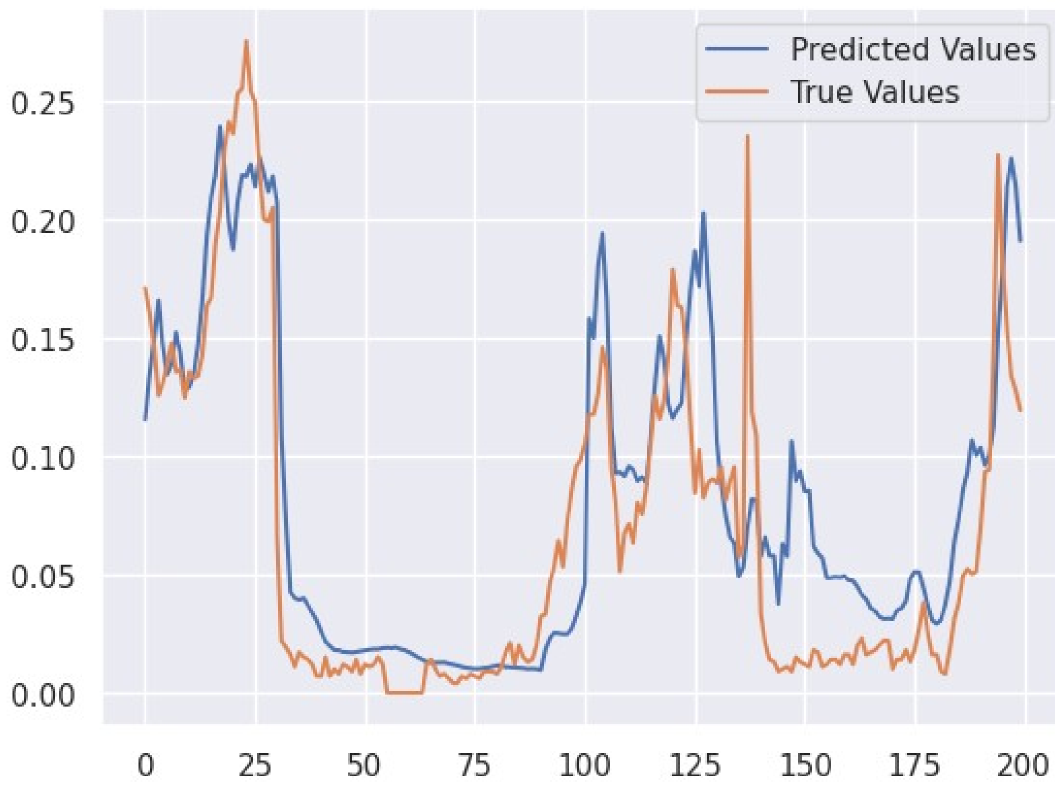
```
0.0055 - root_mean_squared_error: 0.0743 - val_loss: 0.0052 -
val_root_mean_squared_error: 0.0722
Epoch 18/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0055 - root_mean_squared_error: 0.0742 - val_loss: 0.0051 -
val_root_mean_squared_error: 0.0714
Epoch 19/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0055 - root_mean_squared_error: 0.0740 - val_loss: 0.0051 -
val_root_mean_squared_error: 0.0717
Epoch 20/150
1232/1232 [==============================] - 7s 6ms/step - loss:
0.0055 - root_mean_squared_error: 0.0739 - val_loss: 0.0051 -
val_root_mean_squared_error: 0.0713
Epoch 21/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0055 - root_mean_squared_error: 0.0738 - val_loss: 0.0050 -
val_root_mean_squared_error: 0.0710
Epoch 22/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0054 - root_mean_squared_error: 0.0738 - val_loss: 0.0050 -
val_root_mean_squared_error: 0.0708
Epoch 23/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0054 - root_mean_squared_error: 0.0736 - val_loss: 0.0050 -
val_root_mean_squared_error: 0.0705
Epoch 24/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0054 - root_mean_squared_error: 0.0736 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0702
Epoch 25/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0054 - root_mean_squared_error: 0.0734 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0700
Epoch 26/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0054 - root_mean_squared_error: 0.0732 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0703
Epoch 27/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0054 - root_mean_squared_error: 0.0732 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0700
Epoch 28/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0054 - root_mean_squared_error: 0.0732 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0701
Epoch 29/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0731 - val_loss: 0.0049 -
```

```
val_root_mean_squared_error: 0.0701
Epoch 30/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0730 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0699
Epoch 31/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0729 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0699
Epoch 32/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0728 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0697
Epoch 33/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0727 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0700
Epoch 34/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0727 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0698
Epoch 35/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0725 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0697
Epoch 36/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0725 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0697
Epoch 37/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0053 - root_mean_squared_error: 0.0725 - val_loss: 0.0048 -
val_root_mean_squared_error: 0.0696
Epoch 38/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0052 - root_mean_squared_error: 0.0723 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0702
Epoch 39/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0052 - root_mean_squared_error: 0.0723 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0699
Epoch 40/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0052 - root_mean_squared_error: 0.0723 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0697
Epoch 41/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0052 - root_mean_squared_error: 0.0722 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0700
```

```
Epoch 42/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0052 - root_mean_squared_error: 0.0721 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0697
Epoch 43/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0052 - root_mean_squared_error: 0.0721 - val_loss: 0.0048 -
val_root_mean_squared_error: 0.0696
Epoch 44/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0052 - root_mean_squared_error: 0.0721 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0698
Epoch 45/150
1232/1232 [==============================] - 7s 6ms/step - loss:
0.0052 - root_mean_squared_error: 0.0720 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0700
Epoch 46/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0052 - root_mean_squared_error: 0.0720 - val_loss: 0.0049 -
val_root_mean_squared_error: 0.0699
Epoch 47/150
1232/1232 [==============================] - 7s 5ms/step - loss:
0.0052 - root_mean_squared_error: 0.0718 - val_loss: 0.0048 -
val_root_mean_squared_error: 0.0696

# Load the best model
best_model = load_model('best_model.h5')

plt.figure(figsize=(15,6))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```




```
test_predictions = best_model.predict(X_test).flatten()
test_results = pd.DataFrame(data={'Train Predictions':
test_predictions,
                                  'Actual':y_test.flatten()})
test_results.head()

11/11 [==============================] - 1s 2ms/step

   Train Predictions    Actual
0           0.115494  0.171026
1           0.134319  0.160966
2           0.149019  0.146881
3           0.165917  0.125755
4           0.146514  0.130785

rmse = sqrt(mse(y_test, test_predictions))
print('Test RMSE: %.5f' % rmse)

Test RMSE: 0.06896

plt.plot(test_results['Train Predictions'][:200], label='Predicted
Values')
plt.plot(test_results['Actual'][:200], label='True Values')
plt.legend()
plt.show()
```



ARIMA

```
! pip install statsmodels
! pip install pandas
```

```
Requirement already satisfied: statsmodels in
/opt/conda/lib/python3.10/site-packages (0.14.0)
Requirement already satisfied: numpy>=1.18 in
/opt/conda/lib/python3.10/site-packages (from statsmodels) (1.23.5)
Requirement already satisfied: scipy!=1.9.2,>=1.4 in
/opt/conda/lib/python3.10/site-packages (from statsmodels) (1.11.2)
Requirement already satisfied: pandas>=1.0 in
/opt/conda/lib/python3.10/site-packages (from statsmodels) (2.0.2)
Requirement already satisfied: patsy>=0.5.2 in
/opt/conda/lib/python3.10/site-packages (from statsmodels) (0.5.3)
Requirement already satisfied: packaging>=21.3 in
/opt/conda/lib/python3.10/site-packages (from statsmodels) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/opt/conda/lib/python3.10/site-packages (from packaging>=21.3-
>statsmodels) (3.0.9)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.10/site-packages (from pandas>=1.0-
>statsmodels) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
```

ARIMA

```
! pip install statsmodels
! pip install pandas
```

```
Requirement already satisfied: statsmodels in
/opt/conda/lib/python3.10/site-packages (0.14.0)
Requirement already satisfied: numpy>=1.18 in
/opt/conda/lib/python3.10/site-packages (from statsmodels) (1.23.5)
Requirement already satisfied: scipy!=1.9.2,>=1.4 in
/opt/conda/lib/python3.10/site-packages (from statsmodels) (1.11.2)
Requirement already satisfied: pandas>=1.0 in
/opt/conda/lib/python3.10/site-packages (from statsmodels) (2.0.2)
Requirement already satisfied: patsy>=0.5.2 in
/opt/conda/lib/python3.10/site-packages (from statsmodels) (0.5.3)
Requirement already satisfied: packaging>=21.3 in
/opt/conda/lib/python3.10/site-packages (from statsmodels) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/opt/conda/lib/python3.10/site-packages (from packaging>=21.3-
>statsmodels) (3.0.9)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.10/site-packages (from pandas>=1.0-
>statsmodels) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/opt/conda/lib/python3.10/site-packages (from pandas>=1.0-
>statsmodels) (2023.3)
Requirement already satisfied: tzdata>=2022.1 in
/opt/conda/lib/python3.10/site-packages (from pandas>=1.0-
>statsmodels) (2023.3)
Requirement already satisfied: six in /opt/conda/lib/python3.10/site-
packages (from patsy>=0.5.2->statsmodels) (1.16.0)
Requirement already satisfied: pandas in
/opt/conda/lib/python3.10/site-packages (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.10/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/opt/conda/lib/python3.10/site-packages (from pandas) (2023.3)
Requirement already satisfied: tzdata>=2022.1 in
/opt/conda/lib/python3.10/site-packages (from pandas) (2023.3)
Requirement already satisfied: numpy>=1.21.0 in
/opt/conda/lib/python3.10/site-packages (from pandas) (1.23.5)
Requirement already satisfied: six>=1.5 in
/opt/conda/lib/python3.10/site-packages (from python-dateutil>=2.8.2-
>pandas) (1.16.0)
```

```python
import pandas as pd

# Assuming df is your DataFrame
df_arima = pd.read_csv('/kaggle/input/lstm-datasets-multivariate-
univariate/LSTM-Multivariate_pollution.csv')
```

```
df_arima['date'] = pd.to_datetime(df_arima['date'])
df_arima.set_index('date', inplace=True)

train_size = int(len(df_arima) * 0.8)
train, test = df_arima[:train_size], df_arima[train_size:]

from statsmodels.tsa.statespace.sarimax import SARIMAX

# Order and seasonal_order depend on the characteristics of your data
(you may need to tune them)
order = (1, 1, 1)
seasonal_order = (1, 1, 1, 30)  # Example: 24 for daily seasonality

model = SARIMAX(train['pollution'], exog=train[['dew', 'temp',
'press', 'wnd_spd', 'snow', 'rain']],
                order=order, seasonal_order=seasonal_order)
results = model.fit()
```

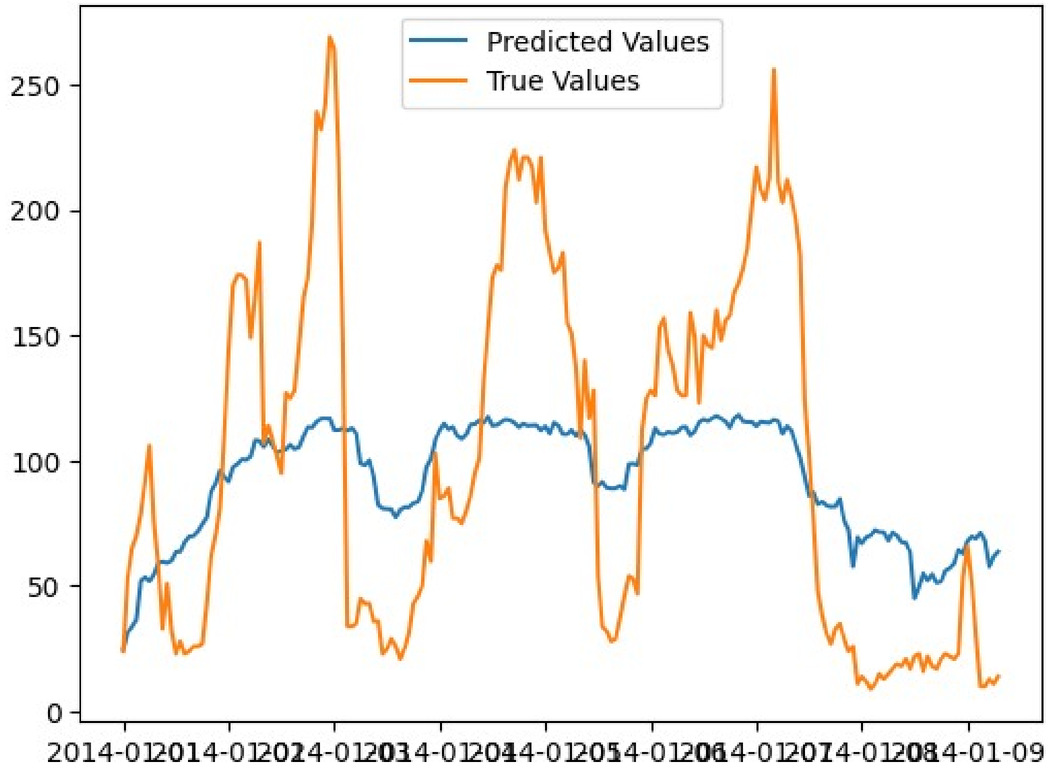
```
/opt/conda/lib/python3.10/site-packages/statsmodels/tsa/base/
tsa_model.py:473: ValueWarning: No frequency information was provided,
so inferred frequency H will be used.
  self._init_dates(dates, freq)
/opt/conda/lib/python3.10/site-packages/statsmodels/tsa/base/tsa_model
.py:473: ValueWarning: No frequency information was provided, so
inferred frequency H will be used.
  self._init_dates(dates, freq)

RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =           11     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  4.88599D+00    |proj g|=  6.74344D-02

 This problem is unconstrained.


At iterate    5    f=  4.84571D+00    |proj g|=  1.49889D-02

At iterate   10    f=  4.75210D+00    |proj g|=  6.87411D-03

At iterate   15    f=  4.75143D+00    |proj g|=  3.27156D-03

At iterate   20    f=  4.75085D+00    |proj g|=  6.12194D-03

At iterate   25    f=  4.74692D+00    |proj g|=  1.68066D-02

At iterate   30    f=  4.74019D+00    |proj g|=  2.02246D-03
```

```
At iterate   35    f=  4.73929D+00    |proj g|=  5.63497D-03

At iterate   40    f=  4.73884D+00    |proj g|=  1.72796D-03

At iterate   45    f=  4.73865D+00    |proj g|=  6.97092D-04

At iterate   50    f=  4.73860D+00    |proj g|=  2.13787D-04

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
   11     50     58      1     0     0   2.138D-04   4.739D+00
  F =   4.7385993161891964

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT

/opt/conda/lib/python3.10/site-packages/statsmodels/base/model.py:607:
ConvergenceWarning: Maximum Likelihood optimization failed to
converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

```
forecast = results.get_forecast(steps=len(test), exog=test[['dew',
'temp', 'press', 'wnd_spd', 'snow', 'rain']])
predicted_values = forecast.predicted_mean

from sklearn.metrics import mean_squared_error
import numpy as np

mse = mean_squared_error(test['pollution'], predicted_values)
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

```
Root Mean Squared Error (RMSE): 104.1033718150482
```

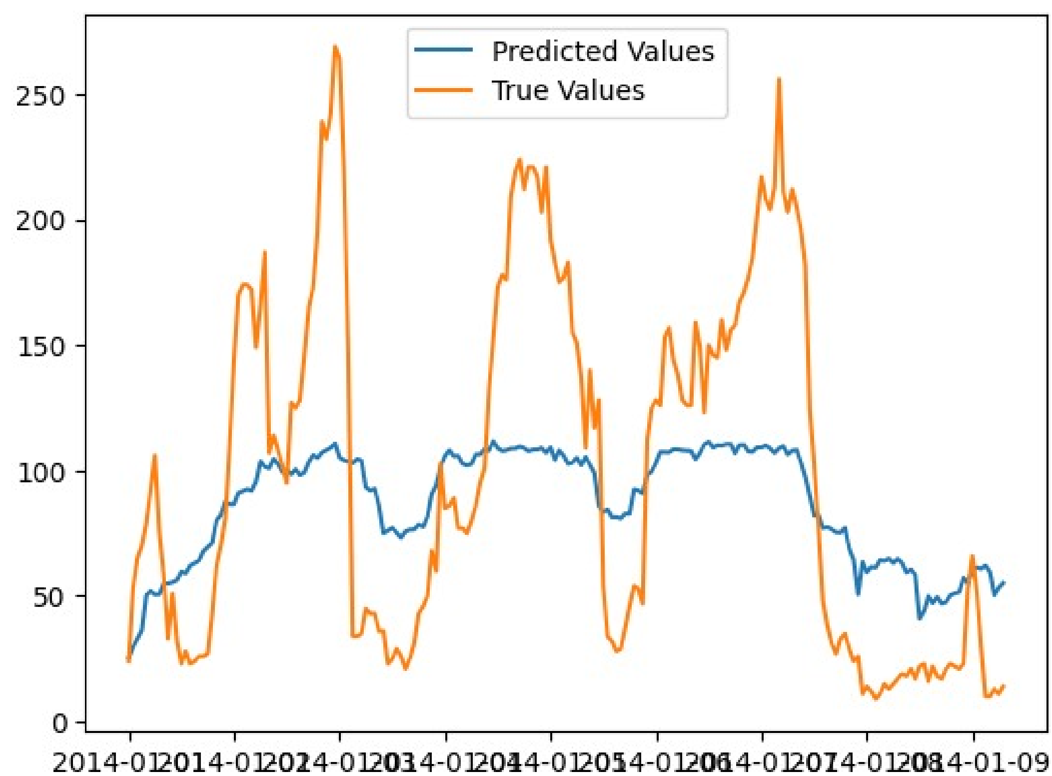
```
import matplotlib.pyplot as plt

plt.plot(predicted_values[:200], label='Predicted Values')
plt.plot(test['pollution'][:200], label='True Values')
plt.legend()
plt.show()
```




```
### experiment

from statsmodels.tsa.statespace.sarimax import SARIMAX

# Order and seasonal_order depend on the characteristics of your data
(you may need to tune them)
order = (1, 1, 1)
seasonal_order = (1, 1, 1, 7)  # Example: 24 for daily seasonality

model = SARIMAX(train['pollution'], exog=train[['dew', 'temp',
'press', 'wnd_spd', 'snow', 'rain']],
                order=order, seasonal_order=seasonal_order)
results = model.fit()

/opt/conda/lib/python3.10/site-packages/statsmodels/tsa/base/
tsa_model.py:473: ValueWarning: No frequency information was provided,
so inferred frequency H will be used.
  self._init_dates(dates, freq)
/opt/conda/lib/python3.10/site-packages/statsmodels/tsa/base/tsa_model
.py:473: ValueWarning: No frequency information was provided, so
inferred frequency H will be used.
  self._init_dates(dates, freq)
 This problem is unconstrained.

RUNNING THE L-BFGS-B CODE
```

```
           * * *

Machine precision = 2.220D-16
 N =           11     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  4.88985D+00    |proj g|=  7.45723D-02

At iterate    5    f=  4.84262D+00    |proj g|=  6.63395D-02

At iterate   10    f=  4.75308D+00    |proj g|=  1.93635D-03

At iterate   15    f=  4.75305D+00    |proj g|=  2.61986D-03

At iterate   20    f=  4.75206D+00    |proj g|=  1.65841D-02

At iterate   25    f=  4.74987D+00    |proj g|=  8.62406D-03

At iterate   30    f=  4.74539D+00    |proj g|=  1.66471D-02

At iterate   35    f=  4.74292D+00    |proj g|=  5.86799D-03

At iterate   40    f=  4.74055D+00    |proj g|=  3.56983D-03

At iterate   45    f=  4.74017D+00    |proj g|=  1.06264D-03

/opt/conda/lib/python3.10/site-packages/statsmodels/base/model.py:607:
ConvergenceWarning: Maximum Likelihood optimization failed to
converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "


At iterate   50    f=  4.73993D+00    |proj g|=  5.56225D-03

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
   11     50     56      1     0     0   5.562D-03   4.740D+00
  F =   4.7399333063875160
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT

forecast = results.get_forecast(steps=len(test), exog=test[['dew',
'temp', 'press', 'wnd_spd', 'snow', 'rain']])
predicted_values = forecast.predicted_mean

from sklearn.metrics import mean_squared_error
import numpy as np

mse = mean_squared_error(test['pollution'], predicted_values)
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error (RMSE): {rmse}")

Root Mean Squared Error (RMSE): 105.14277656039751

import matplotlib.pyplot as plt

plt.plot(predicted_values[:200], label='Predicted Values')
plt.plot(test['pollution'][:200], label='True Values')
plt.legend()
plt.show()
```



Temporal Convolutional Networks

```
! pip install keras-tcn
```

```
Collecting keras-tcn
  Downloading keras_tcn-3.5.0-py3-none-any.whl (13 kB)
Requirement already satisfied: numpy in
/opt/conda/lib/python3.10/site-packages (from keras-tcn) (1.23.5)
Requirement already satisfied: tensorflow in
/opt/conda/lib/python3.10/site-packages (from keras-tcn) (2.12.0)
Requirement already satisfied: tensorflow-addons in
/opt/conda/lib/python3.10/site-packages (from keras-tcn) (0.21.0)
Requirement already satisfied: absl-py>=1.0.0 in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(1.6.3)
Requirement already satisfied: flatbuffers>=2.0 in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(23.5.26)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(1.51.1)
Requirement already satisfied: h5py>=2.9.0 in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(3.9.0)
Requirement already satisfied: jax>=0.3.15 in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(0.4.13)
Requirement already satisfied: keras<2.13,>=2.12.0 in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(2.12.0)
Requirement already satisfied: libclang>=13.0.0 in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(16.0.0)
Requirement already satisfied: opt-einsum>=2.3.2 in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(3.3.0)
Requirement already satisfied: packaging in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(21.3)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!
```

```
=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(3.20.3)
Requirement already satisfied: setuptools in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(68.0.0)
Requirement already satisfied: six>=1.12.0 in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(1.16.0)
Requirement already satisfied: tensorboard<2.13,>=2.12 in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(2.12.3)
Requirement already satisfied: tensorflow-estimator<2.13,>=2.12.0
in /opt/conda/lib/python3.10/site-packages (from tensorflow->keras-
tcn) (2.12.0)
Requirement already satisfied: termcolor>=1.1.0 in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(2.3.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(4.6.3)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/opt/conda/lib/python3.10/site-packages (from tensorflow->keras-tcn)
(0.32.0)
Requirement already satisfied: typeguard<3.0.0,>=2.7 in
/opt/conda/lib/python3.10/site-packages (from tensorflow-addons-
>keras-tcn) (2.13.3)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/opt/conda/lib/python3.10/site-packages (from astunparse>=1.6.0-
>tensorflow->keras-tcn) (0.40.0)
Requirement already satisfied: ml-dtypes>=0.1.0 in
/opt/conda/lib/python3.10/site-packages (from jax>=0.3.15->tensorflow-
>keras-tcn) (0.2.0)
Requirement already satisfied: scipy>=1.7 in
/opt/conda/lib/python3.10/site-packages (from jax>=0.3.15->tensorflow-
>keras-tcn) (1.11.2)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/opt/conda/lib/python3.10/site-packages (from tensorboard<2.13,>=2.12-
>tensorflow->keras-tcn) (2.20.0)
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in
/opt/conda/lib/python3.10/site-packages (from tensorboard<2.13,>=2.12-
>tensorflow->keras-tcn) (1.0.0)
Requirement already satisfied: markdown>=2.6.8 in
/opt/conda/lib/python3.10/site-packages (from tensorboard<2.13,>=2.12-
>tensorflow->keras-tcn) (3.4.3)
Requirement already satisfied: requests<3,>=2.21.0 in
```

```
/opt/conda/lib/python3.10/site-packages (from tensorboard<2.13,>=2.12-
>tensorflow->keras-tcn) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0
in /opt/conda/lib/python3.10/site-packages (from
tensorboard<2.13,>=2.12->tensorflow->keras-tcn) (0.7.1)
Requirement already satisfied: werkzeug>=1.0.1 in
/opt/conda/lib/python3.10/site-packages (from tensorboard<2.13,>=2.12-
>tensorflow->keras-tcn) (2.3.7)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/opt/conda/lib/python3.10/site-packages (from packaging->tensorflow-
>keras-tcn) (3.0.9)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/opt/conda/lib/python3.10/site-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.13,>=2.12->tensorflow->keras-tcn) (4.2.4)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/opt/conda/lib/python3.10/site-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.13,>=2.12->tensorflow->keras-tcn) (0.2.7)
Requirement already satisfied: rsa<5,>=3.1.4 in
/opt/conda/lib/python3.10/site-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.13,>=2.12->tensorflow->keras-tcn) (4.9)
Requirement already satisfied: urllib3<2.0 in
/opt/conda/lib/python3.10/site-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.13,>=2.12->tensorflow->keras-tcn) (1.26.15)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/opt/conda/lib/python3.10/site-packages (from google-auth-
oauthlib<1.1,>=0.5->tensorboard<2.13,>=2.12->tensorflow->keras-tcn)
(1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/opt/conda/lib/python3.10/site-packages (from requests<3,>=2.21.0-
>tensorboard<2.13,>=2.12->tensorflow->keras-tcn) (3.1.0)
Requirement already satisfied: idna<4,>=2.5 in
/opt/conda/lib/python3.10/site-packages (from requests<3,>=2.21.0-
>tensorboard<2.13,>=2.12->tensorflow->keras-tcn) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.10/site-packages (from requests<3,>=2.21.0-
>tensorboard<2.13,>=2.12->tensorflow->keras-tcn) (2023.7.22)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/opt/conda/lib/python3.10/site-packages (from werkzeug>=1.0.1-
>tensorboard<2.13,>=2.12->tensorflow->keras-tcn) (2.1.3)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
/opt/conda/lib/python3.10/site-packages (from pyasn1-modules>=0.2.1-
>google-auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow->keras-
tcn) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in
/opt/conda/lib/python3.10/site-packages (from requests-
oauthlib>=0.7.0->google-auth-oauthlib<1.1,>=0.5-
>tensorboard<2.13,>=2.12->tensorflow->keras-tcn) (3.2.2)
Installing collected packages: keras-tcn
Successfully installed keras-tcn-3.5.0
```

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tcn import TCN

# Load your data
df = pd.read_csv('/kaggle/input/lstm-datasets-multivariate-
univariate/LSTM-Multivariate_pollution.csv')

# Feature scaling
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df[['pollution', 'dew', 'temp',
'press', 'wnd_spd', 'snow', 'rain']])

# Prepare data for TCN
X, y = [], []
for i in range(len(df) - 24):  # Adjust the look-back window size (24
hours in this example)
    X.append(scaled_data[i:(i+24), :])
    y.append(scaled_data[i+24, 0])  # Assuming 'pollution' is the
target variable

X, y = np.array(X), np.array(y)

model = Sequential([
    TCN(input_shape=(X.shape[1], X.shape[2])),
    Dense(units=1)
])

model.compile(optimizer='adam', loss='mean_squared_error')

model.fit(X, y, epochs=50, batch_size=32)

Epoch 1/50
1368/1368 [==============================] - 23s 8ms/step - loss:
0.0571
Epoch 2/50
1368/1368 [==============================] - 10s 7ms/step - loss:
0.0015
Epoch 3/50
1368/1368 [==============================] - 10s 7ms/step - loss:
0.0013
Epoch 4/50
1368/1368 [==============================] - 10s 8ms/step - loss:
0.0013
Epoch 5/50
1368/1368 [==============================] - 11s 8ms/step - loss:
0.0012
Epoch 6/50
```

```
1368/1368 [==============================] - 10s 7ms/step - loss:
0.0011
Epoch 7/50
1368/1368 [==============================] - 10s 7ms/step - loss:
9.2349e-04
Epoch 8/50
1368/1368 [==============================] - 10s 7ms/step - loss:
9.0949e-04
Epoch 9/50
1368/1368 [==============================] - 11s 8ms/step - loss:
8.6504e-04
Epoch 10/50
1368/1368 [==============================] - 11s 8ms/step - loss:
8.1356e-04
Epoch 11/50
1368/1368 [==============================] - 10s 8ms/step - loss:
7.9250e-04
Epoch 12/50
1368/1368 [==============================] - 10s 7ms/step - loss:
7.6921e-04
Epoch 13/50
1368/1368 [==============================] - 10s 7ms/step - loss:
7.5185e-04
Epoch 14/50
1368/1368 [==============================] - 10s 7ms/step - loss:
7.3601e-04
Epoch 15/50
1368/1368 [==============================] - 10s 7ms/step - loss:
7.2050e-04
Epoch 16/50
1368/1368 [==============================] - 10s 7ms/step - loss:
7.0817e-04
Epoch 17/50
1368/1368 [==============================] - 10s 7ms/step - loss:
7.0625e-04
Epoch 18/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.9781e-04
Epoch 19/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.8820e-04
Epoch 20/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.8039e-04
Epoch 21/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.7806e-04
Epoch 22/50
1368/1368 [==============================] - 10s 7ms/step - loss:
```

```
6.7349e-04
Epoch 23/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.6601e-04
Epoch 24/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.6400e-04
Epoch 25/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.5986e-04
Epoch 26/50
1368/1368 [==============================] - 10s 8ms/step - loss:
6.3882e-04
Epoch 27/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.4464e-04
Epoch 28/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.3846e-04
Epoch 29/50
1368/1368 [==============================] - 10s 8ms/step - loss:
6.3939e-04
Epoch 30/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.3633e-04
Epoch 31/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.3672e-04
Epoch 32/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.3061e-04
Epoch 33/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.3201e-04
Epoch 34/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.2916e-04
Epoch 35/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.3025e-04
Epoch 36/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.2190e-04
Epoch 37/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.2117e-04
Epoch 38/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.1875e-04
```

```
Epoch 39/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.2507e-04
Epoch 40/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.1726e-04
Epoch 41/50
1368/1368 [==============================] - 10s 8ms/step - loss:
6.1524e-04
Epoch 42/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.1613e-04
Epoch 43/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.1631e-04
Epoch 44/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.1415e-04
Epoch 45/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.0810e-04
Epoch 46/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.0630e-04
Epoch 47/50
1368/1368 [==============================] - 10s 7ms/step - loss:
6.0427e-04
Epoch 48/50
1368/1368 [==============================] - 10s 7ms/step - loss:
5.9589e-04
Epoch 49/50
1368/1368 [==============================] - 10s 7ms/step - loss:
5.9983e-04
Epoch 50/50
1368/1368 [==============================] - 10s 7ms/step - loss:
5.9156e-04

<keras.callbacks.History at 0x7a795484ad10>

# # Assume you have a test set X_test
# predicted_values = model.predict(X)

# # Inverse transform predictions to the original scale
# predicted_values =
scaler.inverse_transform(np.concatenate([np.zeros((24, 1)),
predicted_values]))

# Make predictions on the training data
predicted_values = model.predict(X)
```

```python
# Inverse transform predictions to the original scale
predicted_values =
scaler.inverse_transform(np.concatenate([np.zeros((predicted_values.sh
ape[0], 6)), predicted_values], axis=1))

# Inverse transform the true values to the original scale
true_values =
scaler.inverse_transform(np.concatenate([np.zeros((y.shape[0], 6)),
y.reshape(-1, 1)], axis=1))
```

```
1368/1368 [==============================] - 4s 3ms/step
```

```python
# Inverse transform the true values to the original scale
# true_values = scaler.inverse_transform(np.concatenate([np.zeros((24,
1)), y.reshape(-1, 1)]))
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score


# Calculate performance metrics
mse = mean_squared_error(true_values, predicted_values)
mae = mean_absolute_error(true_values, predicted_values)
r2 = r2_score(true_values, predicted_values)

print(f'Mean Squared Error (MSE): {mse:.5f}')
print(f'Mean Absolute Error (MAE): {mae:.5f}')
print(f'R-squared (R2): {r2:.5f}')
```

```
Mean Squared Error (MSE): 0.11026
Mean Absolute Error (MAE): 0.06790
R-squared (R2): 0.99013
```