Euribor (January 1994 - October 2024) Univariate DL TSA

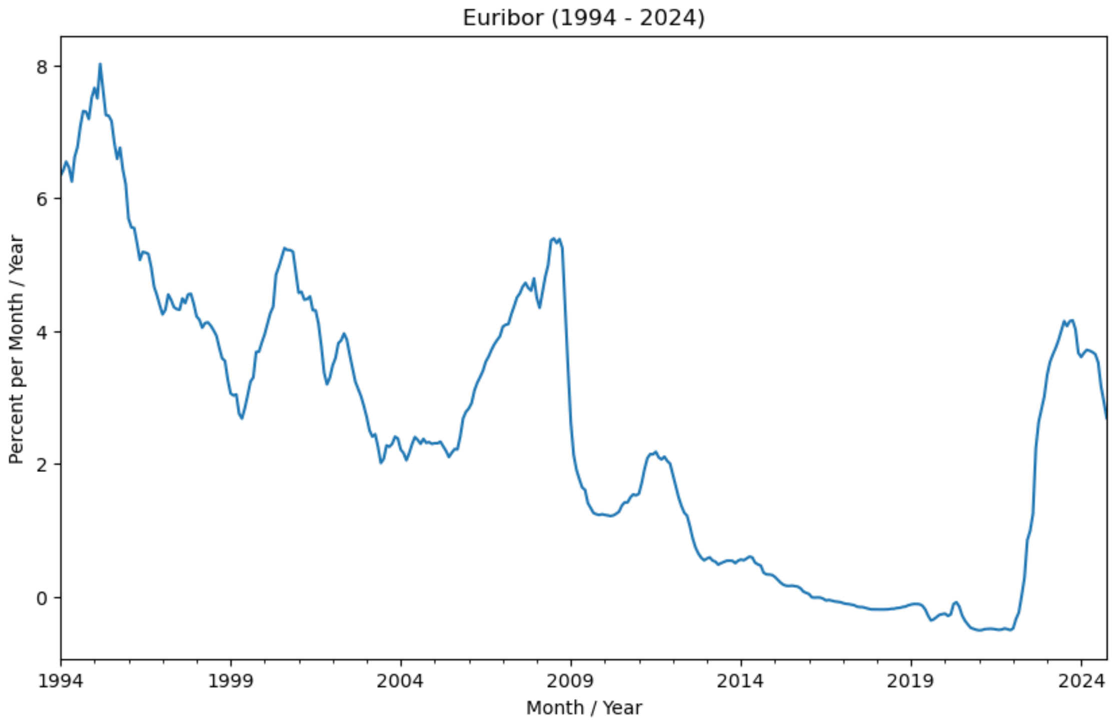
November 20, 2024

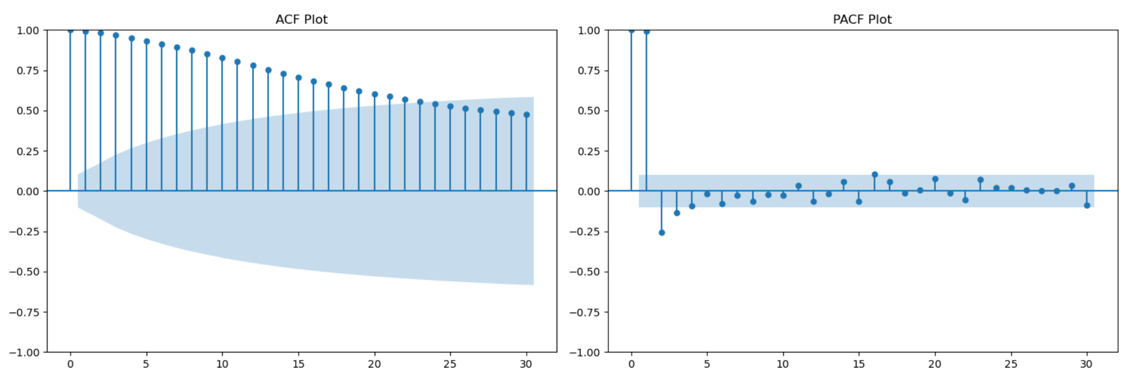
[1]:
```
#Long Short-Term Memory Model

import os
import random
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Input
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import mean_squared_error

random.seed(42)

new_directory = 'C:/Users/artem/Desktop'

os.chdir(new_directory)

euribor_data = pd.read_csv('ECB Data Portal_20241114135109.csv')

euribor_data.rename(columns={'Euribor 1-year - Historical close, average of␣
 ↪observations through period (FM.M.U2.EUR.RT.MM.EURIBOR1YD_.HSTA)':␣
 ↪'Euribor'}, inplace=True)

columns_to_remove = ['DATE', 'TIME PERIOD']

euribor_data = euribor_data.drop(columns=columns_to_remove)

dates = pd.date_range(start='1994', periods=len(euribor_data), freq='M')

euribor_data.index = dates

plt.figure(figsize=(10, 6))
euribor_data['Euribor'].plot()
```

```
plt.title('Euribor (1994 - 2024)')
plt.xlabel('Month / Year')
plt.ylabel('Percent per Month / Year')
plt.show()

def acf_pacf_fig(series, both=True, lag=30):
    fig, axes = plt.subplots(1, 2 if both else 1, figsize=(15, 5))

    if both:
        plot_acf(series, lags=lag, ax=axes[0])
        plot_pacf(series, lags=lag, ax=axes[1])
        axes[0].set_title('ACF Plot')
        axes[1].set_title('PACF Plot')
    else:
        plot_acf(series, lags=lag, ax=axes[0])
        axes[0].set_title('ACF Plot')

    plt.tight_layout()
    plt.show()

acf_pacf_fig(euribor_data['Euribor'], both=True, lag=30)

train = euribor_data.iloc[:300]
valid_start = euribor_data.index[300]
test_start = euribor_data.index[324]

valid = euribor_data[(euribor_data.index >= valid_start) & (euribor_data.index
 ↪< test_start)]
test = euribor_data[test_start:]

scaler = MinMaxScaler()
train.loc[:, 'Euribor'] = scaler.fit_transform(train[['Euribor']])
valid.loc[:, 'Euribor'] = scaler.transform(valid[['Euribor']])
test.loc[:, 'Euribor'] = scaler.transform(test[['Euribor']])

T = 12
HORIZON = 3

def prepare_shifted_data(data, T, horizon):
    data_shifted = data.copy()

    y_columns = []
    for h in range(1, horizon + 1):
        data_shifted[f'y_t+{h}'] = data_shifted['Euribor'].shift(-h)
        y_columns.append(f'y_t+{h}')

    for t in range(1, T + 1):
```

```
        data_shifted[f'Euribor_t-{t}'] = data_shifted['Euribor'].shift(t)

    data_shifted.dropna(inplace=True)

    y = np.array(data_shifted[y_columns])
    X = np.array(data_shifted[[f'Euribor_t-{t}' for t in range(1, T + 1)]])

    return X.reshape(X.shape[0], T, 1), y

X_train, y_train = prepare_shifted_data(train, T, HORIZON)
X_valid, y_valid = prepare_shifted_data(valid, T, HORIZON)
X_test, y_test = prepare_shifted_data(test, T, HORIZON)

print(f"X_train shape: {X_train.shape}, y_train shape: {y_train.shape}")

latent_dim = 6
batch_size = 16
epochs = 50

model = Sequential()

model.add(Input(shape=(X_train.shape[1], X_train.shape[2])))

model.add(LSTM(latent_dim, return_sequences=True))

model.add(LSTM(units=64, return_sequences=False))

model.add(Dense(HORIZON))

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

LSTM_earlystop = EarlyStopping(monitor='val_loss', patience=5)

model_fit = model.fit(
    X_train, y_train,
    batch_size=batch_size,
    epochs=epochs,
    validation_data=(X_valid, y_valid),
    callbacks=[LSTM_earlystop],
    verbose=1
)

preds = model.predict(X_test)

evdta = pd.DataFrame(preds, columns=[f't+{t}' for t in range(1, 4)])

predicted_values = evdta[['t+1', 't+2', 't+3']].values
```

```python
scaled_predicted_values = scaler.inverse_transform(predicted_values)

evdta[['t+1', 't+2', 't+3']] = scaled_predicted_values

evdta['Average Prediction'] = evdta[['t+1', 't+2', 't+3']].mean(axis=1)

evdta.columns = [f'November 2024' if col == 't+1' else
                 f'December 2024' if col == 't+2' else
                 f'January 2025' if col == 't+3' else
                 'Average Prediction' if col == 'Average Prediction' else col
                 for col in evdta.columns]

evdta = evdta.drop(columns=['actual'], errors='ignore')

print(evdta[['November 2024', 'December 2024', 'January 2025', 'Average Prediction']])

y_test_flattened = y_test.reshape(-1, 1)

avg_nov_2024 = evdta['November 2024'].mean()
avg_dec_2024 = evdta['December 2024'].mean()
avg_jan_2025 = evdta['January 2025'].mean()

print(f"Average Prediction for November 2024 (t+1): {avg_nov_2024}")
print(f"Average Prediction for December 2024 (t+2): {avg_dec_2024}")
print(f"Average Prediction for January 2025 (t+3): {avg_jan_2025}")

mse_avg = mean_squared_error(y_test_flattened[:len(evdta)], evdta['Average Prediction'])
mse_t1 = mean_squared_error(y_test_flattened[:len(evdta)], evdta['November 2024'])
mse_t2 = mean_squared_error(y_test_flattened[:len(evdta)], evdta['December 2024'])
mse_t3 = mean_squared_error(y_test_flattened[:len(evdta)], evdta['January 2025'])

rmse_avg = np.sqrt(mse_avg)
rmse_t1 = np.sqrt(mse_t1)
rmse_t2 = np.sqrt(mse_t2)
rmse_t3 = np.sqrt(mse_t3)

print(f"Root Mean Squared Error for November 2024 (t+1): {rmse_t1}")
print(f"Root Mean Squared Error for December 2024 (t+2): {rmse_t2}")
print(f"Root Mean Squared Error for January 2025 (t+3): {rmse_t3}")
print(f"Root Mean Squared Error for Average Prediction: {rmse_avg}")
```






```
X_train shape: (285, 12, 1), y_train shape: (285, 3)
Epoch 1/50
18/18            6s 38ms/step -
loss: 0.1186 - val_loss: 0.0454
Epoch 2/50
18/18            0s 7ms/step - loss:
0.0214 - val_loss: 0.0159
Epoch 3/50
18/18            0s 7ms/step - loss:
0.0120 - val_loss: 0.0064
Epoch 4/50
```

```
18/18            0s 7ms/step - loss:
0.0076 - val_loss: 0.0018
Epoch 5/50
18/18            0s 8ms/step - loss:
0.0065 - val_loss: 7.7963e-04
Epoch 6/50
18/18            0s 10ms/step -
loss: 0.0070 - val_loss: 7.2944e-04
Epoch 7/50
18/18            0s 7ms/step - loss:
0.0077 - val_loss: 8.7984e-04
Epoch 8/50
18/18            0s 7ms/step - loss:
0.0073 - val_loss: 5.4222e-04
Epoch 9/50
18/18            0s 21ms/step -
loss: 0.0069 - val_loss: 6.8102e-04
Epoch 10/50
18/18            0s 12ms/step -
loss: 0.0065 - val_loss: 4.8734e-04
Epoch 11/50
18/18            0s 9ms/step - loss:
0.0056 - val_loss: 4.7078e-04
Epoch 12/50
18/18            0s 10ms/step -
loss: 0.0068 - val_loss: 5.3361e-04
Epoch 13/50
18/18            0s 10ms/step -
loss: 0.0063 - val_loss: 4.5304e-04
Epoch 14/50
18/18            0s 10ms/step -
loss: 0.0062 - val_loss: 3.9673e-04
Epoch 15/50
18/18            0s 9ms/step - loss:
0.0060 - val_loss: 4.3649e-04
Epoch 16/50
18/18            0s 11ms/step -
loss: 0.0068 - val_loss: 4.0960e-04
Epoch 17/50
18/18            0s 9ms/step - loss:
0.0061 - val_loss: 4.1501e-04
Epoch 18/50
18/18            0s 11ms/step -
loss: 0.0054 - val_loss: 4.2010e-04
Epoch 19/50
18/18            0s 14ms/step -
loss: 0.0045 - val_loss: 4.4987e-04
1/1            0s 486ms/step
```

```
    November 2024  December 2024  January 2025  Average Prediction
0       -0.390768      -0.492114     -0.543599           -0.475493
1       -0.389007      -0.490106     -0.541366           -0.473493
2       -0.367369      -0.466700     -0.513609           -0.449226
3       -0.333657      -0.430218     -0.470464           -0.411446
4       -0.266700      -0.357849     -0.385228           -0.336592
5       -0.165177      -0.248301     -0.257169           -0.223549
6        0.014270      -0.056222     -0.034188           -0.025380
7        0.201180       0.143381      0.193654            0.179405
8        0.412638       0.368099      0.446346            0.409028
9        0.772453       0.742941      0.865622            0.793672
10       1.172292       1.154593      1.316570            1.214485
11       1.566791       1.557704      1.747267            1.623920
12       1.945648       1.942766      2.148731            2.012382
13       2.327644       2.328063      2.543222            2.399643
14       2.681913       2.683908      2.900262            2.755361
15       2.989194       2.992518      3.203084            3.061599
16       3.248507       3.253194      3.453704            3.318468
17       3.469337       3.475132      3.663762            3.536077
18       3.663300       3.669397      3.846273            3.726323
19       3.835344       3.840982      4.006635            3.894320
20       3.948987       3.955858      4.109219            4.004688
21       4.045738       4.052542      4.197060            4.098446
22       4.119317       4.126641      4.262957            4.169638
23       4.146219       4.156830      4.283431            4.195493
24       4.093022       4.112257      4.225563            4.143614
25       4.028049       4.055340      4.157753            4.080381
26       3.978958       4.011707      4.107177            4.032614
27       3.942882       3.978914      4.070634            3.997477
28       3.908357       3.946501      4.036530            3.963796
29       3.875349       3.914528      4.004820            3.931566
30       3.842630       3.882115      3.974246            3.899664
Average Prediction for November 2024 (t+1): 2.268301248550415
Average Prediction for December 2024 (t+2): 2.2517552375793457
Average Prediction for January 2025 (t+3): 2.3554482460021973
Root Mean Squared Error for November 2024 (t+1): 2.6393624413216643
Root Mean Squared Error for December 2024 (t+2): 2.6537827374118343
Root Mean Squared Error for January 2025 (t+3): 2.771034934148278
Root Mean Squared Error for Average Prediction: 2.687817033564091
```

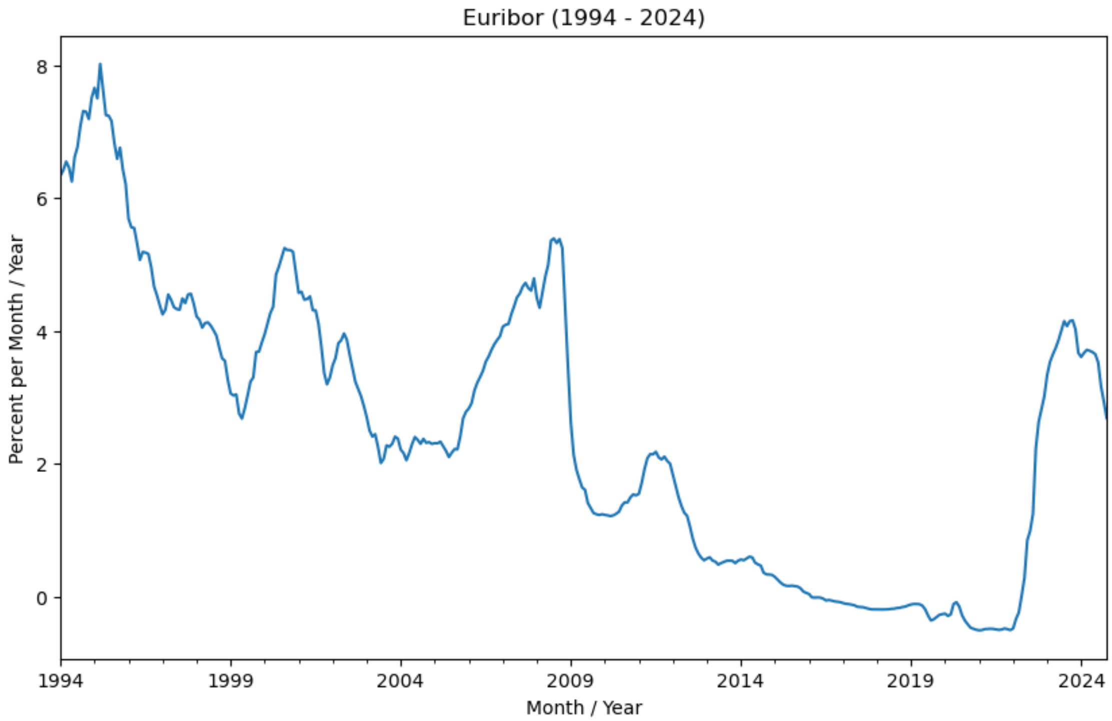
[2]:
```
#Gated Recurrent Units Model

import os
import random
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

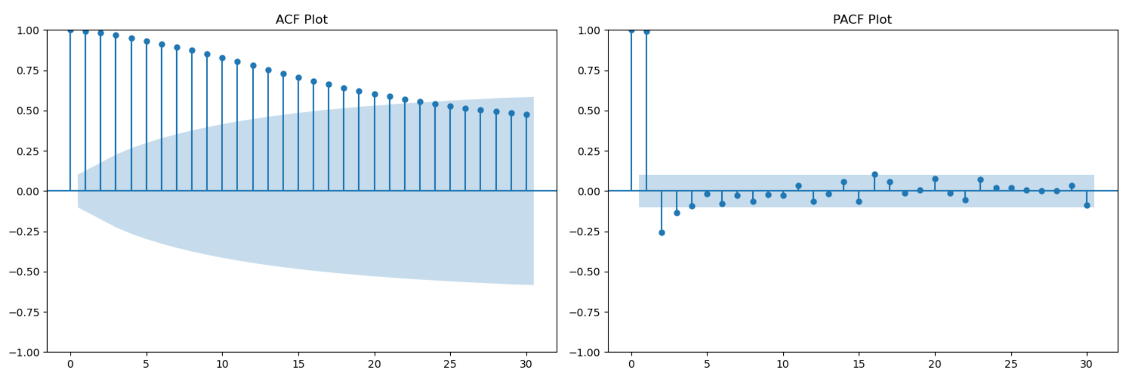
```python
import statsmodels.api as sm
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GRU, Dense, Input
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import mean_squared_error

random.seed(42)

new_directory = 'C:/Users/artem/Desktop'

os.chdir(new_directory)

euribor_data = pd.read_csv('ECB Data Portal_20241114135109.csv')

euribor_data.rename(columns={'Euribor 1-year - Historical close, average of␣
 ↪observations through period (FM.M.U2.EUR.RT.MM.EURIBOR1YD_.HSTA)':␣
 ↪'Euribor'}, inplace=True)

columns_to_remove = ['DATE', 'TIME PERIOD']

euribor_data = euribor_data.drop(columns=columns_to_remove)

dates = pd.date_range(start='1994', periods=len(euribor_data), freq='M')

euribor_data.index = dates

plt.figure(figsize=(10, 6))
euribor_data['Euribor'].plot()
plt.title('Euribor (1994 - 2024)')
plt.xlabel('Month / Year')
plt.ylabel('Percent per Month / Year')
plt.show()

def acf_pacf_fig(series, both=True, lag=30):
    fig, axes = plt.subplots(1, 2 if both else 1, figsize=(15, 5))

    if both:
        plot_acf(series, lags=lag, ax=axes[0])
        plot_pacf(series, lags=lag, ax=axes[1])
        axes[0].set_title('ACF Plot')
        axes[1].set_title('PACF Plot')
    else:
        plot_acf(series, lags=lag, ax=axes[0])
        axes[0].set_title('ACF Plot')
```

```
    plt.tight_layout()
    plt.show()

acf_pacf_fig(euribor_data['Euribor'], both=True, lag=30)

train = euribor_data.iloc[:300]
valid_start = euribor_data.index[300]
test_start = euribor_data.index[324]

valid = euribor_data[(euribor_data.index >= valid_start) & (euribor_data.index
 ↪< test_start)]
test = euribor_data[test_start:]

scaler = MinMaxScaler()
train.loc[:, 'Euribor'] = scaler.fit_transform(train[['Euribor']])
valid.loc[:, 'Euribor'] = scaler.transform(valid[['Euribor']])
test.loc[:, 'Euribor'] = scaler.transform(test[['Euribor']])

T = 12
HORIZON = 3

def prepare_shifted_data(data, T, horizon):
    data_shifted = data.copy()

    y_columns = []
    for h in range(1, horizon + 1):
        data_shifted[f'y_t+{h}'] = data_shifted['Euribor'].shift(-h)
        y_columns.append(f'y_t+{h}')

    for t in range(1, T + 1):
        data_shifted[f'Euribor_t-{t}'] = data_shifted['Euribor'].shift(t)

    data_shifted.dropna(inplace=True)

    y = np.array(data_shifted[y_columns])
    X = np.array(data_shifted[[f'Euribor_t-{t}' for t in range(1, T + 1)]])

    return X.reshape(X.shape[0], T, 1), y

X_train, y_train = prepare_shifted_data(train, T, HORIZON)
X_valid, y_valid = prepare_shifted_data(valid, T, HORIZON)
X_test, y_test = prepare_shifted_data(test, T, HORIZON)

print(f"X_train shape: {X_train.shape}, y_train shape: {y_train.shape}")

latent_dim = 6
batch_size = 16
```

```
epochs = 50

model = Sequential()

model.add(Input(shape=(X_train.shape[1], X_train.shape[2])))

model.add(GRU(latent_dim, return_sequences=True))

model.add(GRU(units=64, return_sequences=False))

model.add(Dense(HORIZON))

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

GRU_earlystop = EarlyStopping(monitor='val_loss', patience=5)

model_fit = model.fit(
    X_train, y_train,
    batch_size=batch_size,
    epochs=epochs,
    validation_data=(X_valid, y_valid),
    callbacks=[GRU_earlystop],
    verbose=1
)

preds = model.predict(X_test)

evdta = pd.DataFrame(preds, columns=[f't+{t}' for t in range(1, 4)])

predicted_values = evdta[['t+1', 't+2', 't+3']].values
scaled_predicted_values = scaler.inverse_transform(predicted_values)

evdta[['t+1', 't+2', 't+3']] = scaled_predicted_values

evdta['Average Prediction'] = evdta[['t+1', 't+2', 't+3']].mean(axis=1)

evdta.columns = [f'November 2024' if col == 't+1' else
                 f'December 2024' if col == 't+2' else
                 f'January 2025' if col == 't+3' else
                 'Average Prediction' if col == 'Average Prediction' else col
                 for col in evdta.columns]

evdta = evdta.drop(columns=['actual'], errors='ignore')

print(evdta[['November 2024', 'December 2024', 'January 2025', 'Average␣
 ↪Prediction']])
```

```
y_test_flattened = y_test.reshape(-1, 1)

avg_nov_2024 = evdta['November 2024'].mean()
avg_dec_2024 = evdta['December 2024'].mean()
avg_jan_2025 = evdta['January 2025'].mean()

print(f"Average Prediction for November 2024 (t+1): {avg_nov_2024}")
print(f"Average Prediction for December 2024 (t+2): {avg_dec_2024}")
print(f"Average Prediction for January 2025 (t+3): {avg_jan_2025}")

mse_avg = mean_squared_error(y_test_flattened[:len(evdta)], evdta['Average Prediction'])
mse_t1 = mean_squared_error(y_test_flattened[:len(evdta)], evdta['November 2024'])
mse_t2 = mean_squared_error(y_test_flattened[:len(evdta)], evdta['December 2024'])
mse_t3 = mean_squared_error(y_test_flattened[:len(evdta)], evdta['January 2025'])

rmse_avg = np.sqrt(mse_avg)
rmse_t1 = np.sqrt(mse_t1)
rmse_t2 = np.sqrt(mse_t2)
rmse_t3 = np.sqrt(mse_t3)

print(f"Root Mean Squared Error for November 2024 (t+1): {rmse_t1}")
print(f"Root Mean Squared Error for December 2024 (t+2): {rmse_t2}")
print(f"Root Mean Squared Error for January 2025 (t+3): {rmse_t3}")
print(f"Root Mean Squared Error for Average Prediction: {rmse_avg}")
```






```
X_train shape: (285, 12, 1), y_train shape: (285, 3)
Epoch 1/50
18/18              6s 41ms/step -
loss: 0.1222 - val_loss: 0.0391
Epoch 2/50
18/18              0s 10ms/step -
loss: 0.0227 - val_loss: 0.0165
Epoch 3/50
18/18              0s 11ms/step -
loss: 0.0150 - val_loss: 0.0076
Epoch 4/50
```

```
18/18            0s 10ms/step -
loss: 0.0106 - val_loss: 9.0887e-04
Epoch 5/50
18/18            0s 9ms/step - loss:
0.0119 - val_loss: 3.6561e-04
Epoch 6/50
18/18            0s 16ms/step -
loss: 0.0115 - val_loss: 3.0521e-04
Epoch 7/50
18/18            0s 9ms/step - loss:
0.0088 - val_loss: 3.4999e-04
Epoch 8/50
18/18            0s 11ms/step -
loss: 0.0085 - val_loss: 3.3910e-04
Epoch 9/50
18/18            0s 9ms/step - loss:
0.0085 - val_loss: 5.0409e-04
Epoch 10/50
18/18            0s 12ms/step -
loss: 0.0078 - val_loss: 4.3261e-04
Epoch 11/50
18/18            0s 16ms/step -
loss: 0.0081 - val_loss: 7.7090e-04
1/1            1s 551ms/step
    November 2024  December 2024  January 2025  Average Prediction
0       -0.745658      -0.730921     -0.721642           -0.732740
1       -0.744282      -0.729317     -0.719936           -0.731178
2       -0.734205      -0.719783     -0.709674           -0.721221
3       -0.716583      -0.703471     -0.692342           -0.704132
4       -0.679347      -0.668983     -0.656007           -0.668112
5       -0.617557      -0.611570     -0.596003           -0.608377
6       -0.506603      -0.507993     -0.488355           -0.500984
7       -0.369560      -0.379378     -0.355968           -0.368302
8       -0.195975      -0.215553     -0.189010           -0.200180
9        0.072771       0.039125      0.068481            0.060126
10       0.401500       0.353028      0.381571            0.378700
11       0.771339       0.709771      0.732881            0.737997
12       1.169519       1.097743      1.111009            1.126091
13       1.591260       1.511859      1.511304            1.538141
14       2.011276       1.927144      1.909200            1.949207
15       2.406398       2.320913      2.285037            2.337449
16       2.762033       2.677352      2.624202            2.687862
17       3.072652       2.990239      2.921999            2.994963
18       3.339995       3.260576      3.178306            3.259625
19       3.560955       3.486130      3.397637            3.481574
20       3.720880       3.649873      3.561971            3.644241
21       3.850365       3.779558      3.687975            3.772633
22       3.955470       3.884954      3.789582            3.876668
```

```
23      4.026760      3.957835      3.861346            3.948647
24      4.048717      3.983481      3.888854            3.973684
25      4.047154      3.986150      3.893045            3.975450
26      4.034515      3.978023      3.886957            3.966499
27      4.011814      3.959972      3.872726            3.948170
28      3.977286      3.929852      3.848495            3.918545
29      3.934641      3.890621      3.816166            3.880476
30      3.888546      3.846554      3.778188            3.837763
Average Prediction for November 2024 (t+1): 1.9143894910812378
Average Prediction for December 2024 (t+2): 1.8694769144058228
Average Prediction for January 2025 (t+3): 1.8347738981246948
Root Mean Squared Error for November 2024 (t+1): 2.492480994575092
Root Mean Squared Error for December 2024 (t+2): 2.442061149004987
Root Mean Squared Error for January 2025 (t+3): 2.38390893399653
Root Mean Squared Error for Average Prediction: 2.439454710689772
```

Author: Artem Urlapov Sedova (Universidad Autónoma de Madrid)