Euribor (January 1994 - October 2024) Univariate DL TSA

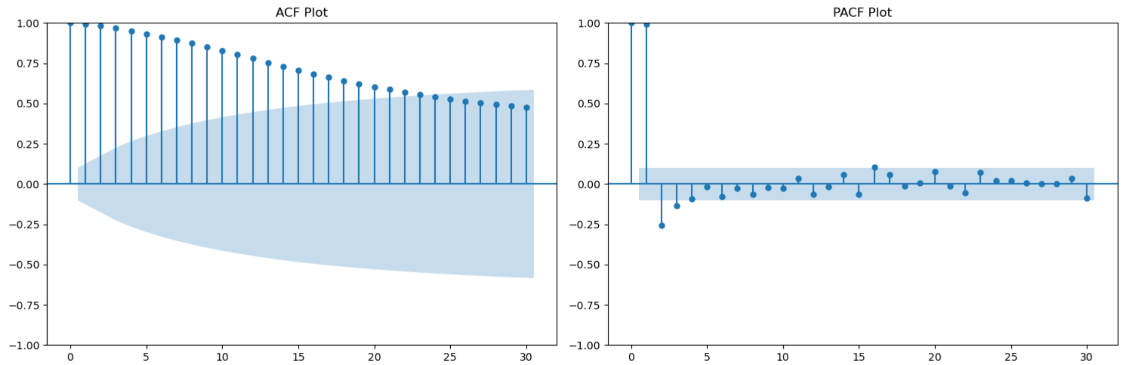
November 19, 2024

[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

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
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              4s 27ms/step -
loss: 0.1423 - val_loss: 0.0461
Epoch 2/50
18/18              0s 6ms/step - loss:
0.0229 - val_loss: 0.0151
Epoch 3/50
18/18              0s 6ms/step - loss:
0.0144 - val_loss: 0.0059
Epoch 4/50
18/18              0s 6ms/step - loss:
0.0091 - val_loss: 0.0013
Epoch 5/50
18/18              0s 6ms/step - loss:
0.0092 - val_loss: 0.0013
Epoch 6/50
18/18              0s 7ms/step - loss:
0.0065 - val_loss: 9.3834e-04
Epoch 7/50
18/18              0s 6ms/step - loss:
0.0058 - val_loss: 8.6217e-04
Epoch 8/50
18/18              0s 6ms/step - loss:
0.0083 - val_loss: 5.7020e-04
Epoch 9/50
18/18              0s 6ms/step - loss:
0.0056 - val_loss: 6.3694e-04
Epoch 10/50
18/18              0s 6ms/step - loss:
0.0056 - val_loss: 4.2841e-04
Epoch 11/50
18/18              0s 6ms/step - loss:
0.0053 - val_loss: 3.8109e-04
```

```
Epoch 12/50
18/18            0s 6ms/step - loss:
0.0065 - val_loss: 4.0378e-04
Epoch 13/50
18/18            0s 6ms/step - loss:
0.0060 - val_loss: 4.3089e-04
Epoch 14/50
18/18            0s 7ms/step - loss:
0.0069 - val_loss: 4.0723e-04
Epoch 15/50
18/18            0s 6ms/step - loss:
0.0059 - val_loss: 4.3359e-04
Epoch 16/50
18/18            0s 7ms/step - loss:
0.0071 - val_loss: 5.3090e-04
1/1          0s 292ms/step
    November 2024  December 2024  January 2025  Average Prediction
0       -0.513842      -0.596957     -0.657899           -0.589566
1       -0.512376      -0.595330     -0.656155           -0.587954
2       -0.494421      -0.576701     -0.634231           -0.568451
3       -0.465938      -0.547148     -0.599549           -0.537545
4       -0.409225      -0.488305     -0.530839           -0.476123
5       -0.322665      -0.398485     -0.426898           -0.382682
6       -0.170665      -0.241576     -0.246739           -0.219660
7       -0.009449      -0.075035     -0.059288           -0.047924
8        0.174426       0.114567      0.150393            0.146462
9        0.478820       0.423541      0.490288            0.464216
10       0.819592       0.766778      0.859006            0.815126
11       1.161768       1.109935      1.217030            1.162911
12       1.496709       1.444765      1.556278            1.499251
13       1.837602       1.783501      1.891763            1.837622
14       2.159513       2.102251      2.199465            2.153743
15       2.445696       2.385621      2.465174            2.432163
16       2.692814       2.630453      2.688586            2.670618
17       2.906277       2.841924      2.877263            2.875154
18       3.093936       3.027268      3.040529            3.053911
19       3.259900       3.190957      3.182710            3.211189
20       3.374998       3.305911      3.277369            3.319426
21       3.470929       3.400422      3.357051            3.409467
22       3.545433       3.474257      3.417847            3.479179
23       3.580452       3.511541      3.442323            3.511439
24       3.548896       3.487685      3.405252            3.480611
25       3.502805       3.449220      3.356881            3.436302
26       3.464830       3.416858      3.318235            3.399974
27       3.434076       3.389976      3.287984            3.370679
28       3.402551       3.361372      3.258308            3.340744
29       3.370859       3.331589      3.229773            3.310740
30       3.338750       3.300547      3.202035            3.280444
```

```
Average Prediction for November 2024 (t+1): 1.8600982427597046
Average Prediction for December 2024 (t+2): 1.7977873086929321
Average Prediction for January 2025 (t+3): 1.7858045101165771
Root Mean Squared Error for November 2024 (t+1): 2.219490154865728
Root Mean Squared Error for December 2024 (t+2): 2.1779990854982882
Root Mean Squared Error for January 2025 (t+3): 2.1589742859434256
Root Mean Squared Error for Average Prediction: 2.185119397986958
```

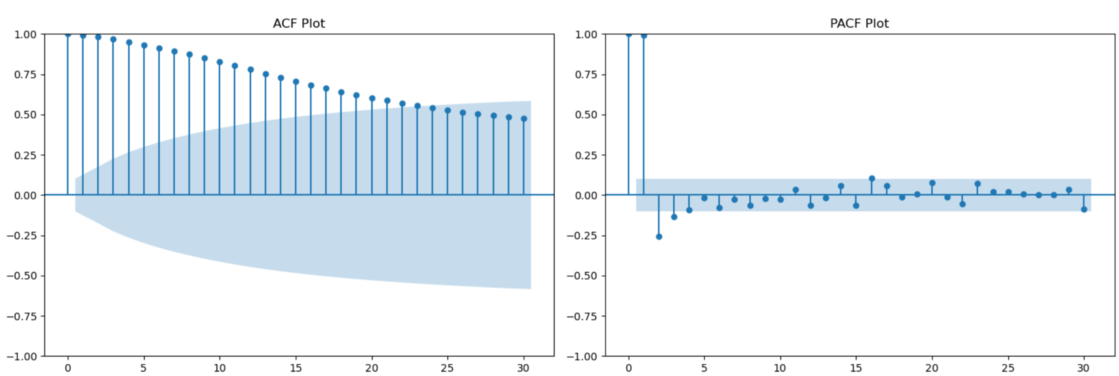
[2]:

```
#Gated Recurrent Units 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 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

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            4s 30ms/step -
loss: 0.1394 - val_loss: 0.0382
Epoch 2/50
18/18            0s 6ms/step - loss:
0.0229 - val_loss: 0.0187
Epoch 3/50
18/18            0s 6ms/step - loss:
0.0146 - val_loss: 0.0078
Epoch 4/50
18/18            0s 6ms/step - loss:
0.0124 - val_loss: 0.0013
Epoch 5/50
18/18            0s 6ms/step - loss:
0.0092 - val_loss: 3.6294e-04
Epoch 6/50
18/18            0s 6ms/step - loss:
0.0087 - val_loss: 3.0588e-04
Epoch 7/50
18/18            0s 6ms/step - loss:
0.0109 - val_loss: 6.6394e-04
Epoch 8/50
18/18            0s 8ms/step - loss:
0.0096 - val_loss: 4.2341e-04
Epoch 9/50
18/18            0s 7ms/step - loss:
0.0089 - val_loss: 3.3639e-04
Epoch 10/50
18/18            0s 6ms/step - loss:
0.0073 - val_loss: 8.0440e-04
Epoch 11/50
18/18            0s 7ms/step - loss:
0.0085 - val_loss: 4.6750e-04
1/1             0s 333ms/step
    November 2024  December 2024  January 2025  Average Prediction
0       -0.648621      -0.635227     -0.630596           -0.638148
1       -0.647411      -0.633726     -0.628999           -0.636712
2       -0.638996      -0.625769     -0.619968           -0.628244
3       -0.624120      -0.612127     -0.604648           -0.613632
4       -0.592462      -0.583064     -0.572357           -0.582628
5       -0.539480      -0.534185     -0.518635           -0.530767
6       -0.444075      -0.445588     -0.421973           -0.437212
7       -0.324724      -0.333861     -0.301648           -0.320078
8       -0.172195      -0.189908     -0.148590           -0.170231
```

```
9          0.063333        0.033595       0.087044              0.061324
10         0.353824        0.312231       0.376074              0.347376
11         0.683871        0.633360       0.703954              0.673728
12         1.042277        0.987171       1.060531              1.029993
13         1.424056        1.368472       1.440997              1.411175
14         1.806793        1.754894       1.822512              1.794733
15         2.169140        2.125386       2.186223              2.160250
16         2.497047        2.464001       2.517128              2.492725
17         2.784491        2.763579       2.809592              2.785887
18         3.032425        3.023797       3.062415              3.039546
19         3.236872        3.241931       3.279630              3.252811
20         3.385033        3.401459       3.443518              3.410004
21         3.505092        3.526976       3.568779              3.533615
22         3.602897        3.629452       3.670153              3.634167
23         3.670188        3.701962       3.742972              3.705041
24         3.693460        3.731358       3.773673              3.732830
25         3.694934        3.738529       3.781076              3.738180
26         3.685333        3.734404       3.777510              3.732416
27         3.665533        3.719832       3.765172              3.716846
28         3.634139        3.692933       3.742435              3.689836
29         3.594673        3.656388       3.711120              3.654060
30         3.551725        3.614386       3.673718              3.613276
Average Prediction for November 2024 (t+1): 1.746614694595337
Average Prediction for December 2024 (t+2): 1.7504078149795532
Average Prediction for January 2025 (t+3): 1.7918970584869385
Root Mean Squared Error for November 2024 (t+1): 2.2464573938800116
Root Mean Squared Error for December 2024 (t+2): 2.261140885208698
Root Mean Squared Error for January 2025 (t+3): 2.2964614155436798
Root Mean Squared Error for Average Prediction: 2.267953548481707
```

Author: Artem Urlapov Sedova (Universidad Autónoma de Madrid)