Alberto Marinelli (638283)

Midterm 1 (assignment 1)
Autoregressive analysis

To study the effect of **non-stationarity**, we will add a linear trend to the “Appliances” column of the dataset, which measures the energy consumption of appliances across a period of 4.5 months.

1) First, preprocess the dataset to remove any trend (if necessary)
2) Perform an autoregressive analysis on the clean time series
3) Add a linear trend to the time series
4) Perform the autoregressive analysis on the new time series

Show the results of the analysis with and without the linear trend, discussing your design choices and the results.
To perform the autoregressive analysis, fit an autoregressive model on the first 3 months of data and estimate performance on the remaining 1.5 months. Remember to update the autoregressive model as you progress through the 1.5 testing months. For instance, if you have trained the model until time T, use it to predict at time T+1. Then to predict at time T+2 retrain the model using data until time T+1. And so on. You might also try and experimenting with less "computationally heavy" retraining schedule (e.g. retrain only "when necessary"). You can use the autoregressive model of your choice (AR, ARMA, ...).

Hint: in Python, use the ARIMA class of the statsmodels library (set order=(3,0,0) for an AR of order 3); in Matlab you can use the ar function to fit the model and the forecast function to test.

Checks for Stationarity

Use the **Augmented Dickey-Fuller test** to provide a quick check and confirmatory evidence that the time series is stationary or non-stationary.

This type of statistical test is based on the null hypothesis that suggests the time series has a unit root, meaning it is non-stationary (has some time-dependent structure). The alternate hypothesis (rejecting the null hypothesis) is that the time series is stationary.

This result are interpreted using the **p-value** from the test. A p-value below a threshold suggests we reject the null hypothesis (stationary), otherwise a p-value above the threshold suggests we fail to reject the null hypothesis (non-stationary).

p-value > 0.05: Fail to reject the null hypothesis (H0), the data has a unit root and is non-stationary.

p-value <= 0.05: Reject the null hypothesis (H0), the data does not have a unit root and is stationary.

The p-value of the test applied on **our time series** is 0.0 that means that it **is stationary**.

[Code]

```
adf = adfuller(data)
adf, pvalue, critical_values = adf[0], adf[1], adf[4]
print(f"ADF: {adf}\np-value: {pvalue}")
print("Critical values:")
for k, v in critical_values.items():
    print(f"{k}: {v}")
```

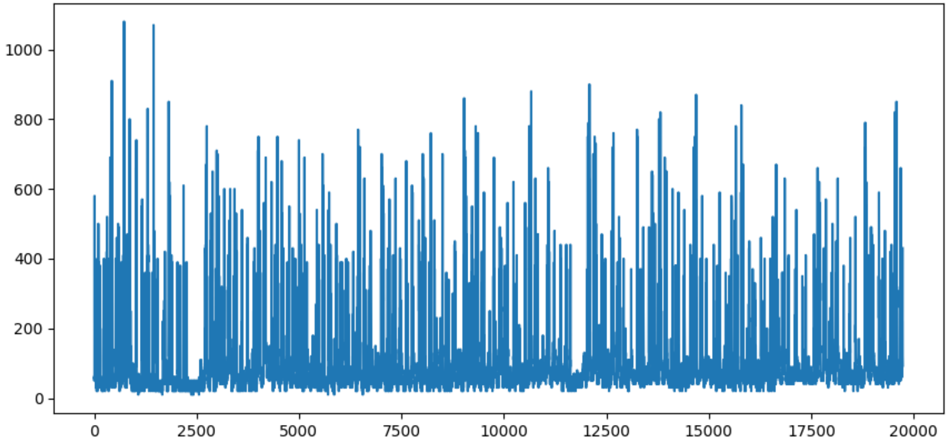
[Out]

```
ADF: -21.616378198036106
p-value: 0.0
Critical values:
1%: -3.430681600227742
5%: -2.8616865555329394
10%: -2.566848007525354
```

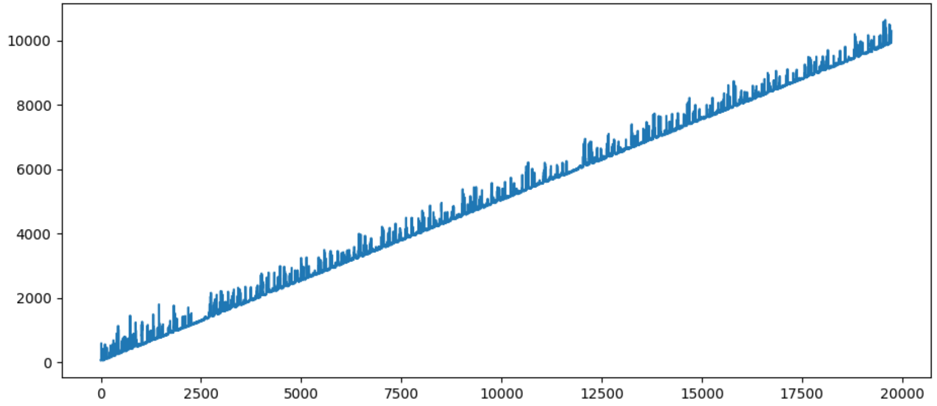
Generate a trend in the original time series

The goal of this operation is to add a time dependency in the time series to **make it non-stationary**.

[Original time series]



[Time series after adding a growing trend]



[Code]

```
total_duration = len(data)
step = 1
time = np.arange(0, total_duration, step)

k0= 0.01
k1= 0.5

series_trend = k0 + k1 * time
past_data = data["Appliances"]
data["Appliances"]= past_data + series_trend
```

Checks for Stationarity on the new time series

The p-value of the Augmented Dickey-Fuller test applied on the **new time series** is 0.93 that means that it **is not stationary**.

[Out]

```
ADF: -0.21007184178712407
p-value: 0.9373103837320303
Critical values:
1%: -3.430682139184172
5%: -2.86168679371923
10%: -2.566848134307755
```

Plot autocorrelation and partial autocorrelation

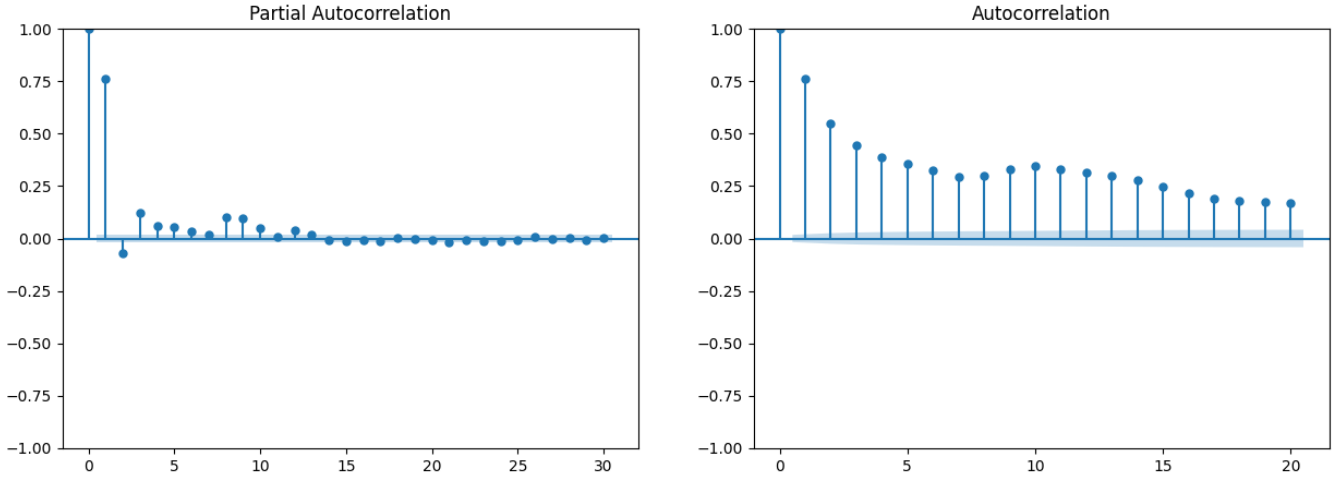
To understand visually which lags have more influence on the current value y_t

ACF plot shows the autocorrelations which measure the relationship between y_t and y_{t-k} for different values of k.

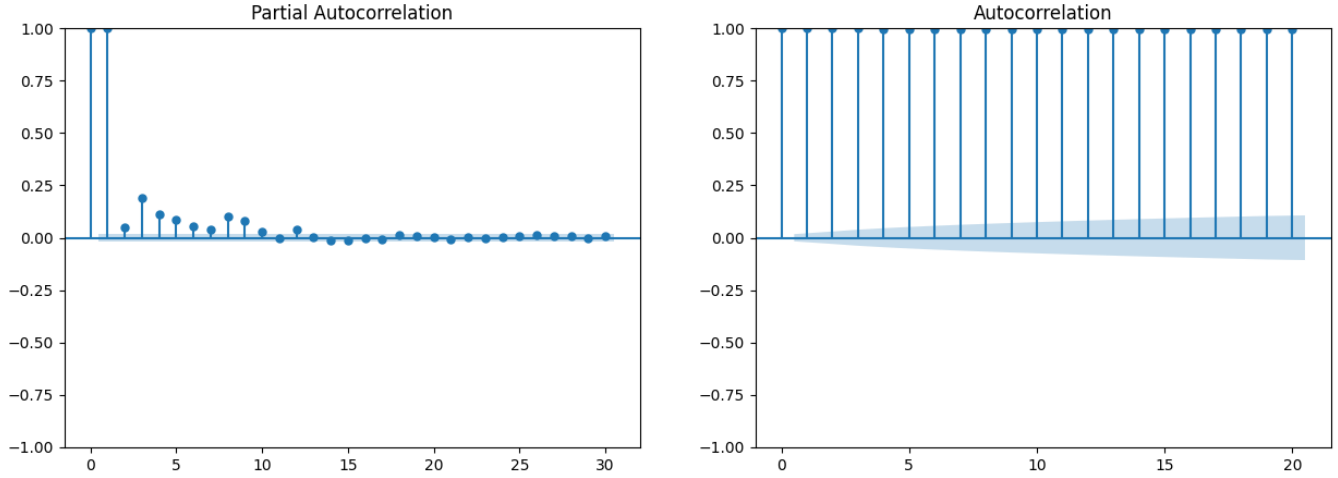
Partial autocorrelations measure the relationship between y_t and y_{t-k} after removing the effects of lags 1, 2, 3, …, k.

The ACF plots are also useful for identifying non-stationary time series. For a **stationary time series [1]**, the ACF will **drop to zero** relatively quickly, while the ACF of **non-stationary data [2] decreases slowly**.

[1] [Original time series]



[2] [Time series after adding a growing trend]



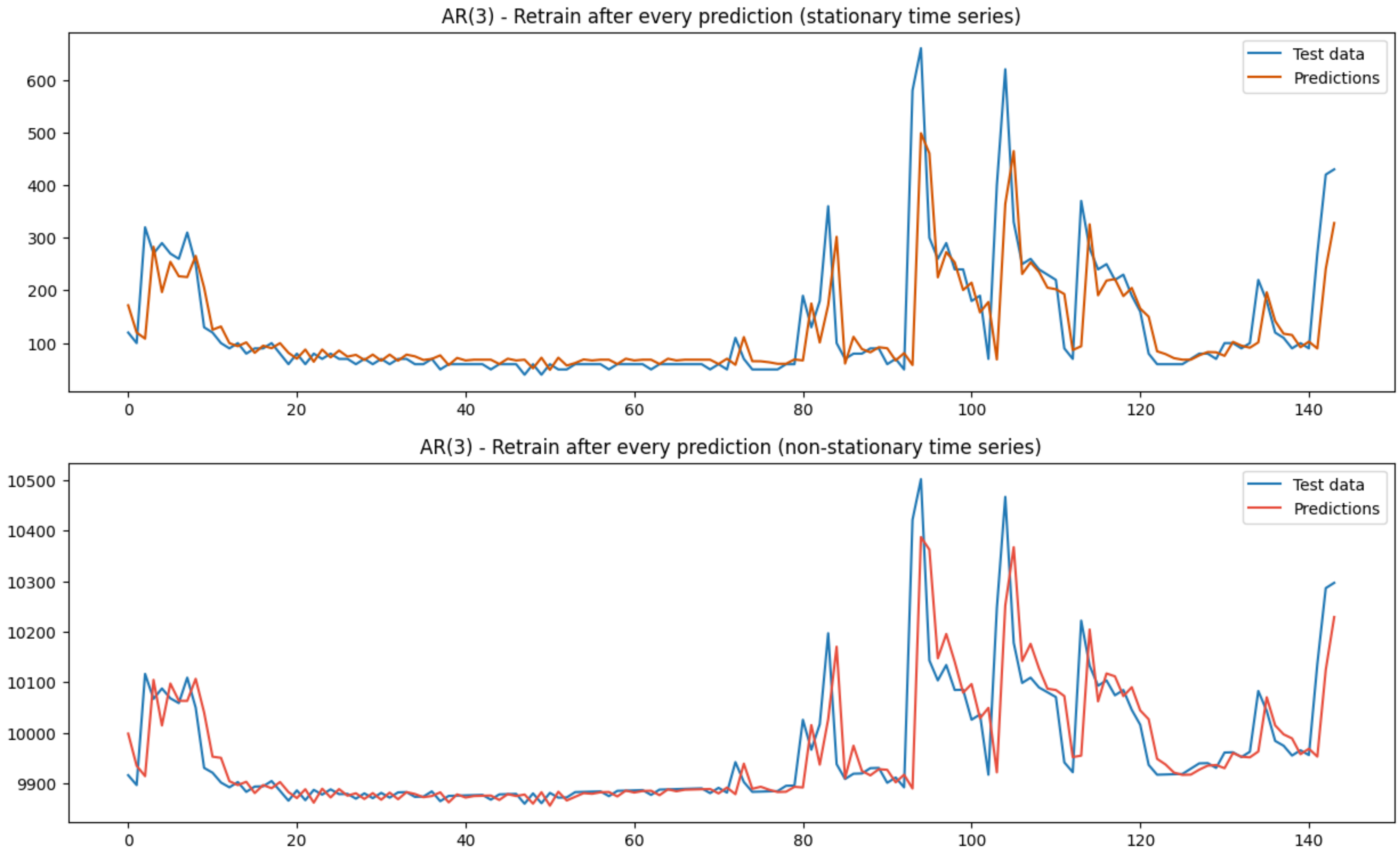
Training

The model used is **AR** with **order 3**. This models is trained with the training data (first 3 months) and tested on the test data (last 1.5 month), instead the **retraining schedule** is to **retrain after every prediction on the test set**.

[Code]

```
def predict_and_retrain(order_ar, order_ma, tr_set, ts_set, retrain, err_thresh):

    # Instantiate ARIMA statsmodels
    model = ARIMA(endog=tr_set, order=(order_ar, 0, order_ma)) #It works like ARMA when d = 0
    res = model.fit() #Fit the model

    idx_retrain = 0
    count_no_retrain = 1
    predictions_arr = []

    # Start forecasting on the test set and retrain when needed
    for i in tqdm(range(len(ts_set))):
        predictions_arr.append(res.forecast(steps=count_no_retrain)[-1])
        err = abs(ts_set[i] - predictions_arr[-1])
        if retrain and err > err_thresh:
            idx_last_retrain = i
            count_no_retrain = 1
            tr_set = np.concatenate((tr_set, ts_set[idx_retrain: i + 1])) #Adding to the tr_set new observed data
            model = ARIMA(endog=ts_set, order=(order_ar, 0, order_ma))
            res = model.fit() #Retrain
        else:
            count_no_retrain += 1

    # MAE - compute mean absolute error
    mae = np.mean(np.abs(np.subtract(ts_set, predictions_arr)))
    out = {'mae': mae, 'predictions': predictions_arr}
    filename = str(order_ar) + "_" + str(order_ma) + "_" + str(err_thresh) + "_retrain" if retrain else "" + ".json"
    with open(filename, 'w') as outf:
        json.dump(out, outf, indent='\t')
```

Results


AR(3) - Retrain after every prediction (stationary time series)
Test data
Predictions
600
500
400
300
200
100
0
20
40
60
80
100
120
140
AR(3) - Retrain after every prediction (non-stationary time series)
Test data
Predictions
10500
10400
10300
10200
10100
10000
9900
0
20
40
60
80
100
120
140

Conclusions

1) Tools like **PACF and ACF look different** when non-stationarity exists. It becomes difficult to determine the order of AR and the MA terms.

2) We restrict ourself to the stationary region as on the non-stationary one **ARMA processes become explosive** (they go to infinity)

3) The **variance increases to infinity** as the number of observation increases that implies a **bad asymptotic result**

4) If processes were not stationary, we would have a hard time estimating the **mean**, **variance** and **autocorrelation** of the process because they would **change at every time step**.

Thanks for attention!

Alberto Marinelli