


# ISPR

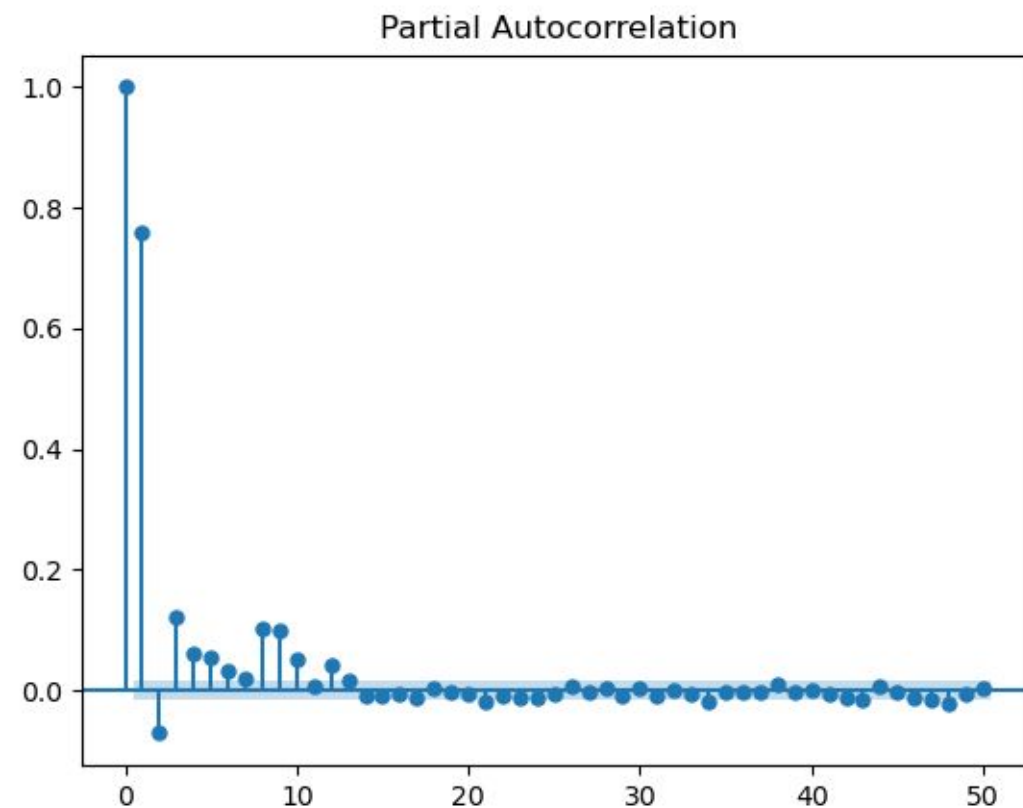
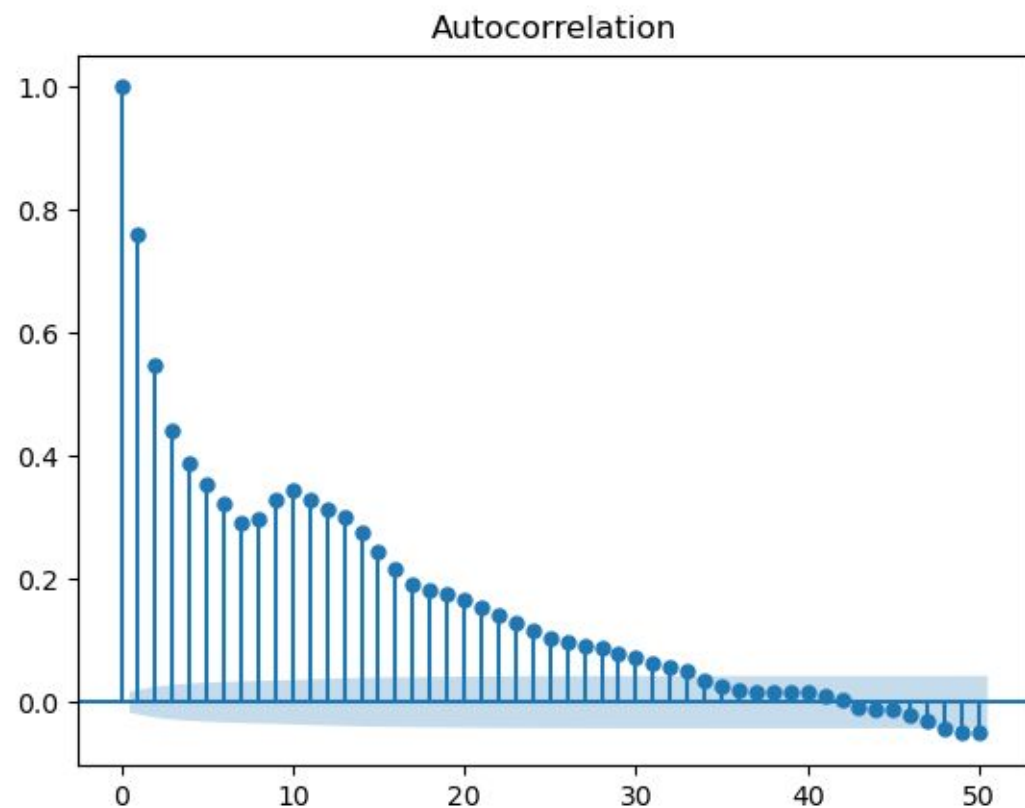
1° midterm

*assignment 1 – autoregressive analysis*



Frigo Giacomo (ID: 626201)  
g.frigo@studenti.unipi.it

# ACF & PACF



# code snippets

```
# (91 days) * 24 * (6 data per day)
train_limit = (91*24*6)+1
train_data = data[:train_limit]
test_data = data[train_limit:]
```

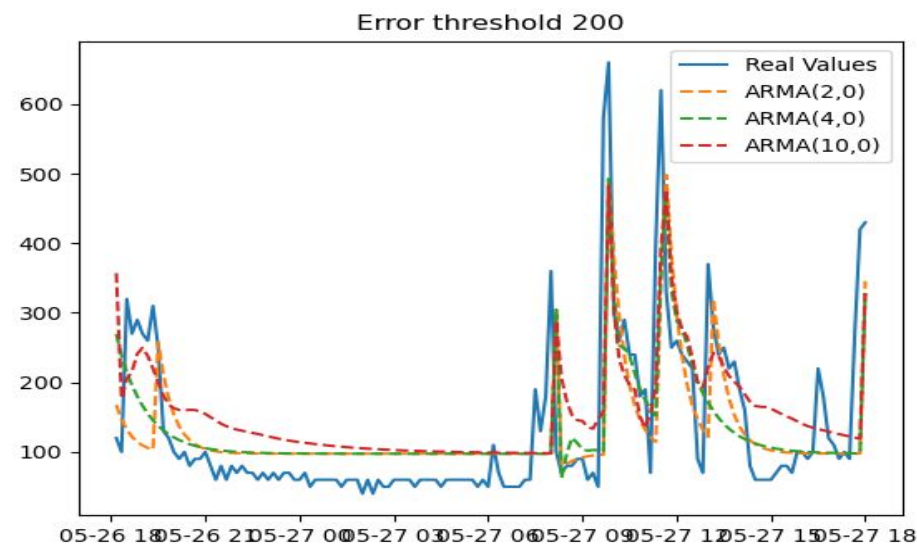
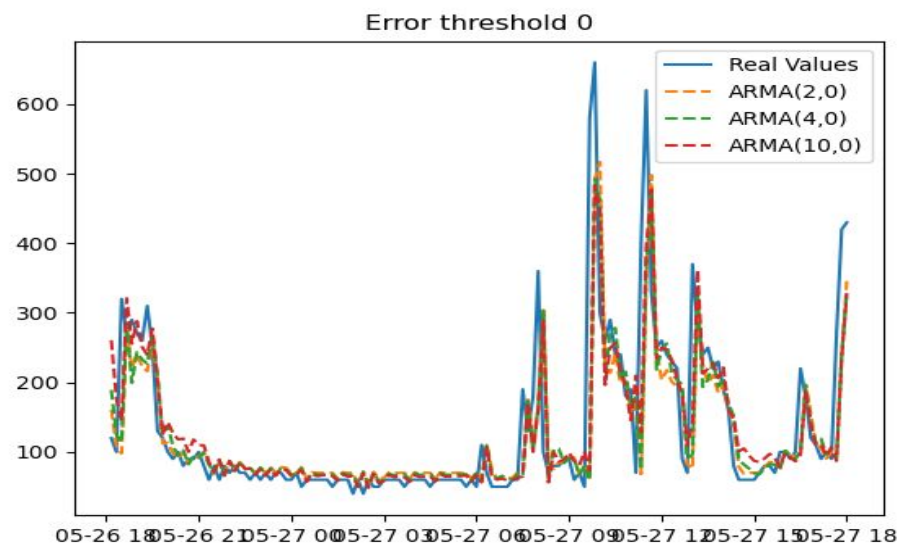
```
#ar model order
ar_order = 2
ma_order = 0

#create the ARIMA model
ar = ARIMA(train_data, order=(ar_order, 0, ma_order))

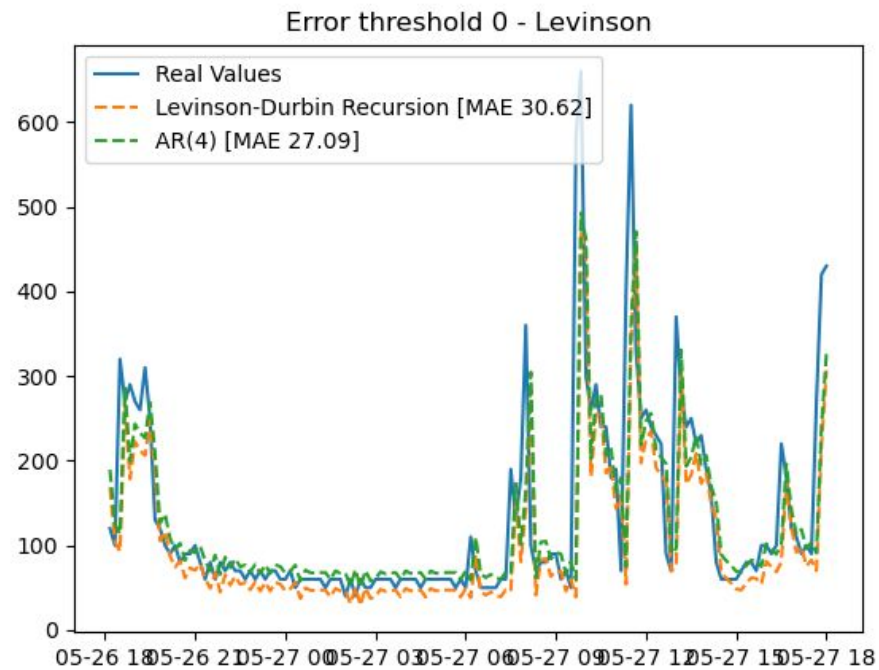
#fitting the model
model = ar.fit()
coeff = model.arparams
#print fitting results
print(model.summary())
```

```
for idx in range(train_limit, len(data)):
    #make steps predictions
    prediction = model.forecast(steps=steps)[-1]
    predictions.append(prediction)
    #calc the error
    error = abs(data[idx]-prediction)
    errors.append(error)
    #update train data (appending the real data we've just tried to forecast)
    train_data.append(data[idx])
    #if error large -> retrain the model
    if error > error_threshold:
        print_("Retraining at idx {} Prediction: {}, Real: {}".format(idx, prediction, data[idx]))
        #create once again the model (with new train data)
        ar = ARIMA(train_data, order=(ar_order, 0, ma_order))
        #fit the model
        model = ar.fit()
        #reset step to 1
        steps = 1
    else:
        steps += 1
```

# results



# Levinson Durbin Recursion



```
def train(train_data, k):  
    # nlags is k (the order)  
    # isacov need to be set to true if the first argument contains the autocovariance  
    sigmav, arcoefs, pacf, sigma, phi = levinson_durbin(train_data, nlags=k, isacov=False)  
    return arcoefs
```

```
for i in range(train_limit, len(data)):  
    estimation = 0  
    for idx, coeff in enumerate(coefficients):  
        estimation += data[i-(idx+1)] * coeff  
    estimations.append(estimation)  
    #calc the error  
    error = abs(data[i] - estimation)  
    errors.append(error)  
    #update train data  
    train_data.append(data[i])  
    #add the new data to the train data set  
    #and re train  
    coefficients = train(train_data, k)
```