Politecnico di Torino

ICT for Smart Mobility
Lab Report
Lab 2

Prof. Marco Mellia
Prof. Luca Vassio

Group 11

Seyed Mohammad Mehdi Hosseini - S301769
Seyedeh Sadaf Ekram - S301178
Elaheh Lotfimahyari - S301125

December 11, 2022

Contents

3.1 Rental Data Extraction and Inspection of Possible Missing Data 1
3.2 Stationarity and Periodicity Evaluation 1
3.3 Auto Correlation Function (ACF) and Partial Auto Correlation Function (PACF) . . 2
3.4 ARMA Model Training 2
3.5 Hyperparameters Tuning with Grid Search 3
3.5.1 Find the Optimal p and q 3
3.5.2 Find the Optimal Learning Strategy and Training Sample Size 4
3.5.3 Optimal Hyperparameters 5
3.6 [Optional]: Effect of Time Horizons on Prediction Performance 5

3.1 Rental Data Extraction and Inspection of Possible Missing Data

In this lab, we aimed to to experiment with predictions, as well as analyze how error changes with hyperparameters in ARIMA Models with respect to a time series of data, consisting rentals of each hour of one month for Seattle (Car2Go), Milano (Car2Go), Milano (Enjoy) cities and mobility services. When using the ARIMA model, it's important that there should be no missing data in the series. To find the best month to use for further analysis, we've chosen 4 months of October, November and December pf 2017, and January of 2018 to step forward. The November has a whole week of missing data which is not convenient, and there were some problems with December and January data stationarity which will be precisely explained in 3.2. Eventually, we considered filtered data from the October (no outliers and fake rentals which have been well-discussed in the previous lab). Although the October was the best month among other above-mentioned months, but there are still numerous missing data to handle in all three cities. Our strategy for the missing data imputation is to replace the missing with the a week shifted value at the next week, plus some noise. Note that, if the missing data appears at the last week of the month, we replace it with the a week shifted value at the previous week, plus some noise. This strategy is mainly based on the weekly periodic behavior in the whole data. The results and the missing dates are shown in figure 3.1.

3.2 Stationarity and Periodicity Evaluation

In order to benefit from using ARIMA model, we also have to satisfy the stationarity conditions. In December and January, there is no periodic trend and the series are non-stationary due to the Christmas and new year holidays. It's the most important point to not consider these months for the ARIMA model. We analyzed rolling statistics of series with a window size of one week (168 hours) to make sure that the properties are constant over time. Figure 3.1 verifies that our series is almost stationary considering all the above-mentioned facts. Augmented Dickey Fuller test (ADF Test) is also a common statistical test used to test whether a given time series is stationary or not. It is one of the most commonly used statistical test when it comes to analyzing the stationary of a series. The results related to each city, has been recorded in Table 3.1. Thanks to the stationary properties of the October month, we don't need to consider differencing in our model so we can set the parameter d in $ARIMA(p, d, q)$ equal to zero all the time. Therefore, our model will be simplified to ARMA.

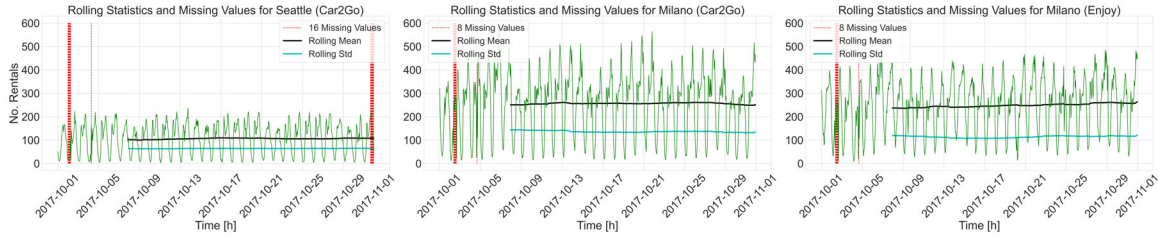

Figure 3.1: Rolling statistics with the window size of a week (168h) and missing values

City	ADF Test $< 5 \times 10^{-3}$
Seattle(Car2Go)	1.51×10^{-7}
Milano (Car2Go)	2.69×10^{-5}
Milano (Enjoy)	3.43×10^{-5}

Table 3.1: Augmented Dickey Fuller (ADF) Test results

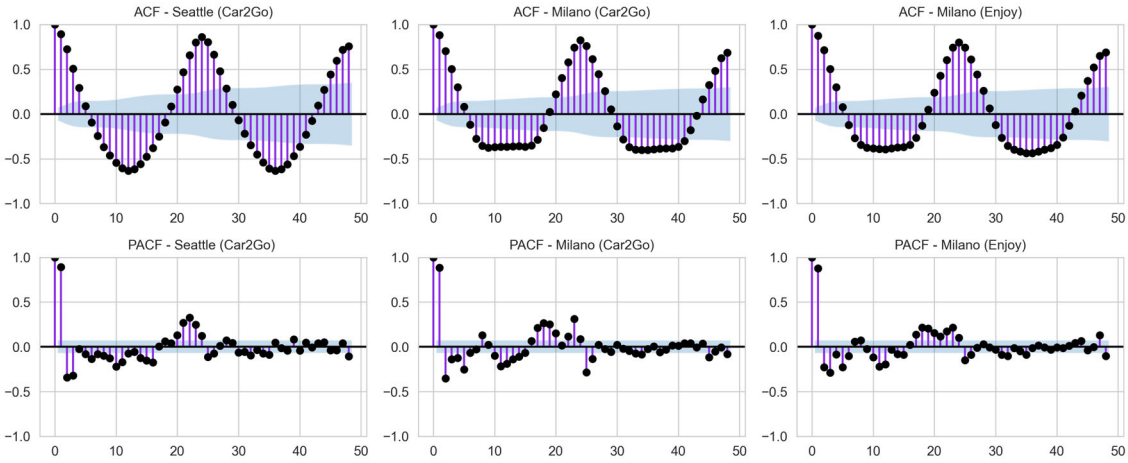

Figure 3.2: ACF and PACF over a 48 hours period

3.3 Auto Correlation Function (ACF) and Partial Auto Correlation Function (PACF)

ARMA model itself consists of two models, Auto-Regressive (AR) and Moving Average (MA) and each of the models require a hyperparameter. These parameters respectively correspond to the number of previous samples (p) and previous errors (q) that affect the current realization of the observed random variable. AR property demonstrates that for $h > p$ the Partial Auto-correlation Function (PACF) is equal (or nearly equal) to 0 and MA property demonstrates that the Auto-correlation Function (ACF) is equal (or nearly equal) to 0 for $h > q$.

Based on Figure 3.2, we considered $p = 3$ and $q = 4$ for Seattle (Car2Go), $p = 2$ and $q = 4$ for Milano (Car2GO), and $p = 3$ and $q = 4$ for Milano (Enjoy) as the boundary hyperparameters.

3.4 ARMA Model Training

ARMA model was trained based on hyperparameters(p and q) chosen in the previous part. We use 504 hours(3 weeks) for training and 168 hours(1 week) for testing. For each sample, a new model was trained with the expanding window learning strategy.
Figure 3.3 shows the predictions, real rentals, and the element-wise error between them.
We evaluate our models by four types of errors listed in table 3.2. Among all of these measurements, Mean Absolute Percentage Error (MAPE) gives a clear and understandable accuracy measurement of our model.
Based on Table 3.2, we can see that the minimum Mean Square Error (MSE) and Mean Absolute Error (MAE) are for Seattle (Car2Go); however, MAPE is minimum for Milano (Enjoy). Because MSE and MAE are proportional to the number of rentals. We already know that there are less rentals

for Seattle (Car2Go), so we would have less MSE and MAE respectively but MAPE is independent of the number of rentals and it's more convenient to decide about the model performance based on the percentage error. Moreover, the results of different p values for Milano (Enjoy) could be observed in Figure 3.4. By increasing the value of p, the model become more complex and as well, it tends to be more fluctuated during the prediction and it leads to much more errors than the other possible values.

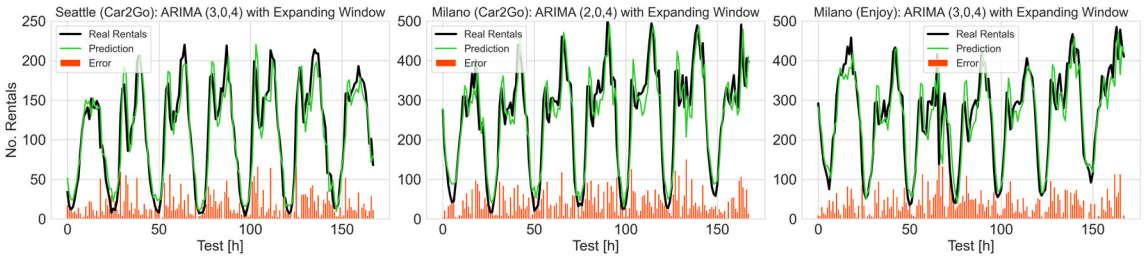

Figure 3.3: ARMA Models predictions for each city

City	p	d	q	MSE	MAE	$MAPE$	R^2
Seattle(Car2Go)	3	0	4	542.26	17.73	0.317	0.869
Milano(Car2Go)	2	0	4	2617.05	40.93	0.269	0.850
Milano(Enjoy)	3	0	4	2712.90	39.58	0.199	0.802

Table 3.2: ARMA Models error results with expanding window strategy

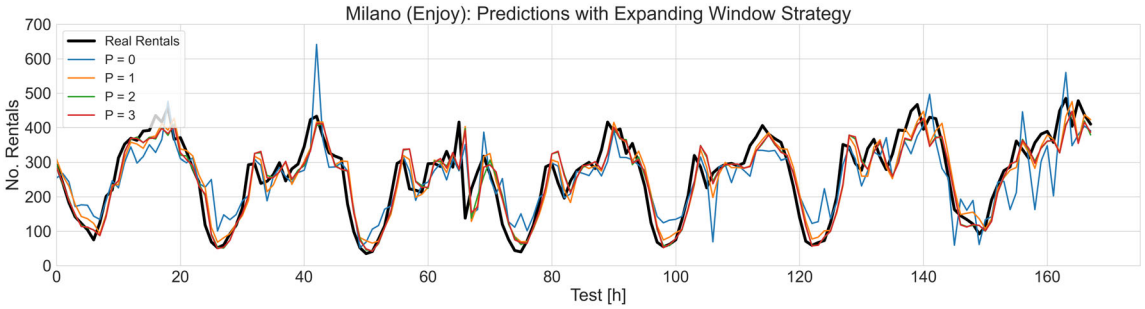

Figure 3.4: ARMA Model predictions for Milano (Enjoy) with different p values and expanding window strategy, $(d = 0, q = 2)$

3.5 Hyperparameters Tuning with Grid Search

3.5.1 Find the Optimal p and q

We performed a grid search with fixed number of training and testing samples (504 hours and 168 hours respectively). The learning strategy was expanding window and its performance was evaluated by the MAPE error for a variation of $p = [0, 3]$ and $q = [0, 4]$ for Seattle (Car2Go) and Milano (Enjoy), and $p = [0, 2]$ and $q = [0, 4]]$ for Milano (Car2Go). We considered these ranges according to what we discussed in the previous section and Figure 3.2.
The Grid Search took 3075.23 seconds to be executed by "Intel(R) Core(TM) i7-9750HF CPU @ 2.60GHz" processor unit and 16 GB DDR4 dedicated RAM. According to the results mentioned on

Figure 3.5, for both Milano (Car2go) and Milano (Enjoy), $(p = 2, q = 1)$ are the optimum hyperparameters. For Seattle (Car2Go), both $(p = 3, q = 2)$ and $(p = 2, q = 2)$ pairs' results are almost identical; however, it's much more convenient to choose the simpler model which is $(p = 2, q = 2)$.

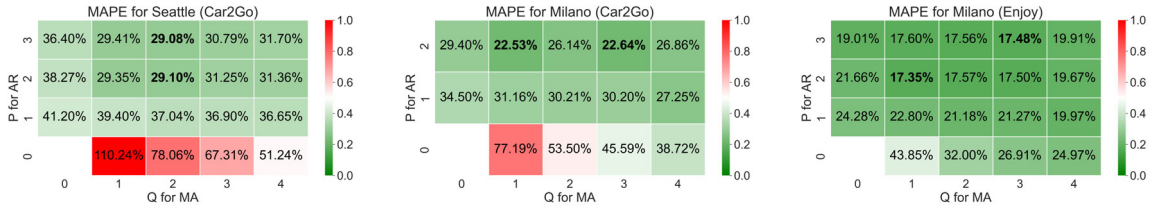

Figure 3.5: Grid Search for optimal hyperparameters (p and q) for each city

3.5.2 Find the Optimal Learning Strategy and Training Sample Size

Considering the Optimal hyperparameters from previous part, and fixed test sample size (168 hours). Another grid search has been performed to find the optimal value for training sample size (N) and also to find the best learning strategy between Expanding Window and Sliding Window Strategies. We trained the ARMA models with training sample size ranging from one to three weeks (each two days) with both Expanding Window and Sliding Window strategies. Note that, the test sample size is still 1 week (168 hours). It took 2194.77 seconds to be executed by the same processor mentioned above, See 3.5.1. Figure 3.6 illustrates grid search results. For all of the cities, there is a little difference between Sliding and Expanding Window Strategies but we can conclude that by increasing the number of training samples, we can observe that MAPE decreases since the model can remarkably generalize the new data and consequently, it can predict the future values with less errors. In addition, we can observe that most of the minimum values in the results are in case of Expanding Window Strategy. This learning strategy enables a prediction model to be trained on not only the initial training set, but also the test values already predicted. Particularly, the training window expands as predictions are made. On the contrary, with the sliding window method, only the previous N values are used for training the model, where N represents the size of the training window. With all that being said, the Sliding Window Strategy results were satisfying on Milano (Enjoy) data.
Table 3.3 demonstrates which number of training samples for both strategy could cause minimum MAPE.

City	p	d	q	Sample Size	Learning Strategy	$MAPE$
Seattle (Car2Go)	2	0	2	17	Sliding Window	0.290
Seattle (Car2Go)	2	0	2	9	Expanding Window	0.288
Milano (Car2Go)	2	0	1	17	Sliding Window	0.238
Milano (Car2Go)	2	0	1	21	Expanding Window	0.225
Milano (Enjoy)	2	0	1	11	Sliding Window	0.169
Milano (Enjoy)	2	0	1	21	Expanding Window	0.174

Table 3.3: ARMA Models error results with different learning strategies and sample sizes

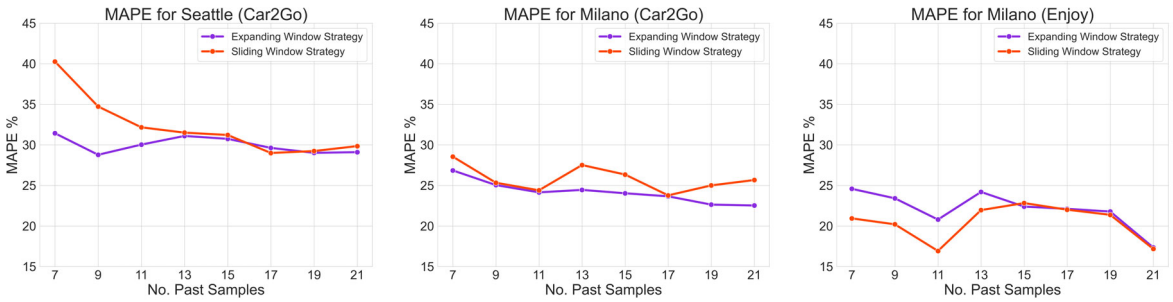

Figure 3.6: MAPE - Sliding vs Expanding Window Strategy

3.5.3 Optimal Hyperparameters

Bringing all these together, we can conclude that in general, Expanding Window Strategy outperforms the Sliding Window in most of the cases. Moreover, Keeping smaller p and q values can make the ARMA model almost performs finer than the large values since it's more robust in short-term forecasting.

Figure 3.7 shows the predictions based on optimal hyperparameters found in previous sections.

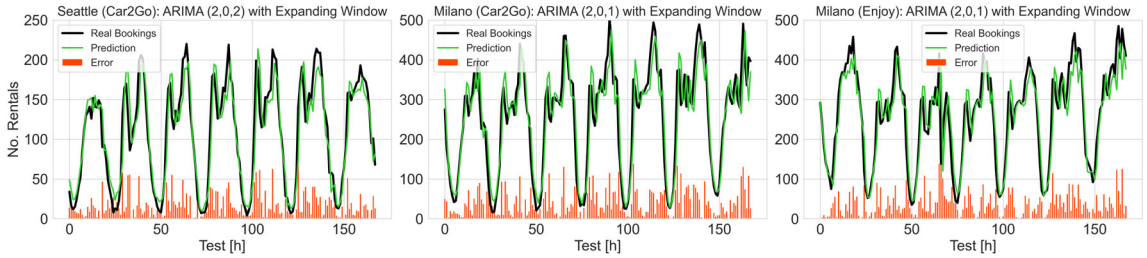

Figure 3.7: Final ARMA models based on optimal hyperparameters

3.6 [Optional]: Effect of Time Horizons on Prediction Performance

As we discussed in the previous sections, during the prediction process, an ARMA model was trained for each timestep ahead. Consequently, each ARMA model trained has been only produced one predicted value. This Task explores the possibility of increasing the number of predictable timesteps for each model. According to the ARMA model, when the time horizon is only set to 1, the predicted result is similar to the original time series (just same as what we did in the Task 7); nevertheless, the accuracy of the prediction decreases significantly as the forecast horizon increases, due to ARIMA's limitations as a long-term forecasting model. According to the Figure 3.8, which is the results related to Seattle (Car2Go), the predictions for high horizons will lead to high MAPEs which is not considered even as normal accuracy. Figure 3.9 also as a verification for Seattle (Car2Go), can show how badly the model is in forecasting the next horizons (range of [4,24]) and long-term values. As there were 24 horizons in the initial Figure, there are only 8 horizons shown in the Figure 3.9 for the reading simplicity. It should be noted that, this process took 35407.13 seconds to execute.

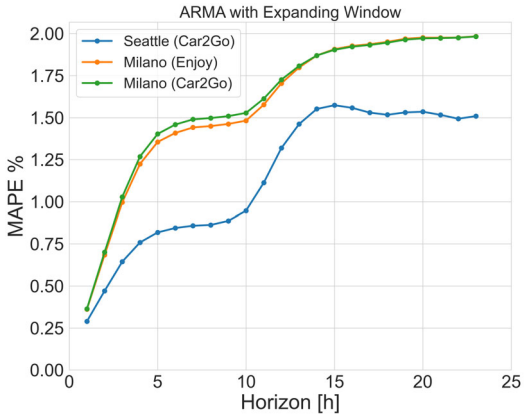

Figure 3.8: ARMA Model with different horizon values, ($p = 2, q = 2$ with expanding window strategy for Seattle (Car2Go)

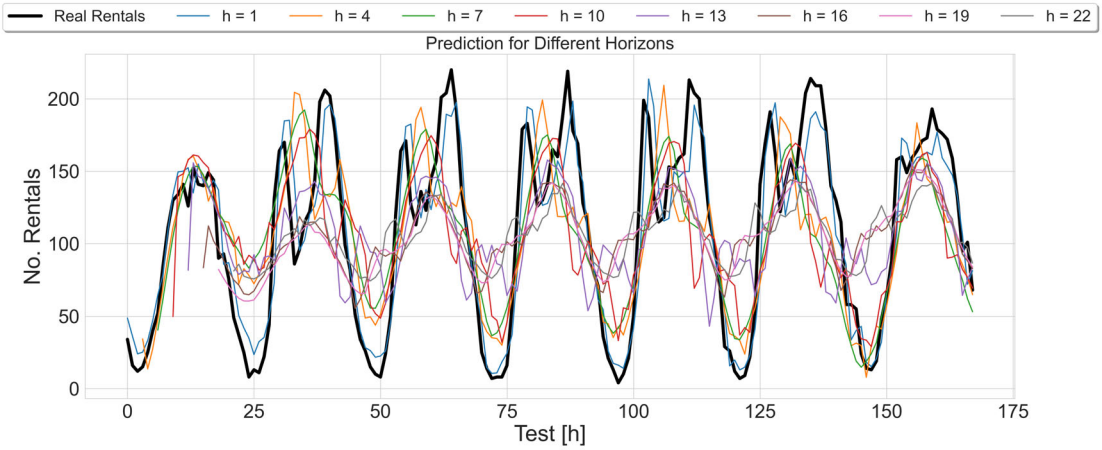

Figure 3.9: Predictions for different horizon values, ($p = 2, q = 2$ with expanding window strategy for Seattle (Car2Go)