POLITECNICO DI MILANO
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MSc. Computer Science and Engineering
Academic Year 2023/2024

POLITECNICO
MILANO 1863

Team CAIPIRINHA
Homework 2

Authors:
Domingos Savio Pinheiro do Nascimento Junior (10880091)
domingossavio.pinheiro@mail.polimi.it

Felipe Azank dos Santos (10919711)
felipe.azank@mail.polimi.it

Felipe Bagni (10912321)
felipe.bagni@polimi.it

Professors:
Matteo Matteuci
Giacomo Boracchi

Dec 2023

1 Introduction

The following report serves as a brief explanation of what was done by the group in regards to completing the second homework competition, which consisted on an Time Series forecast with a fixed window and with different categories.

2 Exploratory Data Analysis and Data Cleaning

Given the dataset for the Time series problem, an exploratory data analysis is done. At first, the overall data was explored. At first, it was possible to notice that the dataset is composed by 48000 time series with many variable lengths. This showed to be a problem given that many decisions were drafted in order to deal with it.

2.1 Data Inspection

After loading the data into a jupyter notebook, the first step taken was to check the distribution of the time series per category. In order to understand closely how they differ from each other.

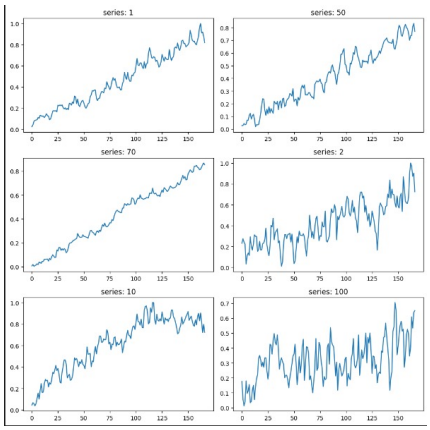

Figure 1: Category A - less samples then the other ones but with longer time series

Category	Mean Length	Median Length	Std Dev Length
A	277.09	287.0	109.26
B	164.78	156.0	116.07
C	206.83	201.0	146.18
D	216.01	237.0	148.93
E	162.10	118.0	127.79
F	193.54	171.0	153.08

Table 1: Length statistics by category

As seen in the table, each categories presents similar mean length values, however, with high standard deviation. This shows that the values have a big variability in size. Which should be considered when developing models that require a fixed "look back" length study.

After some time analysing these time series distributions (and also doing it visually), it was possible to realize that category A represented a more consistent trend, while category F showed a quite erratic behaviour (even making the team wonder if random data was imputed on it).

By the end of the analysis, it was set that all categories would be pre-processed by the same steps and eventually (if decided) the division between them wound take place when entering the modelling stage, given that, with regards to main estimations, the categories would look the same. As seen below

Category	Mean of Values	Std Dev of Values
A	0.47	0.23
B	0.40	0.22
C	0.45	0.23
D	0.44	0.23
E	0.44	0.24
F	0.45	0.24

Table 2: Values statistics by category

3 Methodology

3.1 Baseline models

In order to start the modeling studies, a progressive approach was used in order to understand better the difficulty and complexity of this time series task. The first models applied on the data were simple and well known classical time series prediction models that did not evolve around the Deep Learning Framework. In order to make these models, the Dart Library [1] was used, given that it has some nice built-in functions.

During development, many classical models were trained, like: Naive Seasonal Forecast, ARIMA, SARIMA (and many others from the SARIMAX family) and decomposition approaches such as the Theta model.

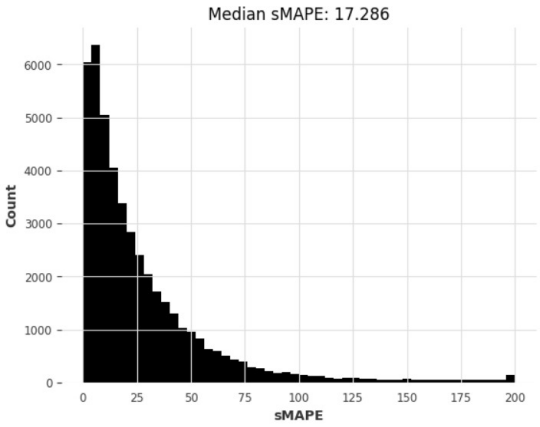

Figure 2: sMAPE (Symmetric Mean Absolute Percentage Error) of the Theta model

Given that this is not the priority of the work being

done (as well as because these models would lose for simple LSTMs as we will see later), it was decided not to go further on this report with regards to the classical approaches.

3.2 Data Preparation for the Neural Netwoks

3.2.1 Data Preprocessing

Before the creation of all the architectures, a simple preprocessing of the clean data was made in order to prepare the input of the networks.

In order to avoid data leakage, the first process was to split the data into 2 parts: 20% of the samples for validation and Early Stopping mechanism, and 80% for training. The split was made in a non stratified manner, given that, at first, we decided to build models that were agnostic to the different categories.

The next procedure was the imputation of some possible null values inside the time series. In order to solve that, we used a simple moving average with equal weight and a window of 18 previous values. After that, the data was then scaled using Robust and MinMax Scalers, even though we face a problem with values between 0 and 1 (which also looks like something made on purpose by the AN2DL team. In the end, in order to reduce the amount of time series with few examples. The values inside were filtered so that only time series with more than 20% of missing values (i.e having more than 550 samples).

3.2.2 Sequence format for the Neural Networks

Finally, as a last step of the process, the sequence creation function was made-up in order to make it feasible to be inserted in a Deep Learning Framework. In order to do that, we prepared the data so that we produce the input (X) composed by 200 time series values with the output (y) composed at first by 9 values and later by 18 values. It's important to realize that, at the end, the amount of training "rows" is much larger than the amount of series, given that we can extract a big number of time series from it.

3.3 Simple LSTM Approach

The first proposed model to understand the process of neural network was a simple double layered LSTM. The use of long memory and short memory in the layers cells presents a good alternative to the previous solutions (like RNNs) given that dividing these paths is fundamental to decrease the chances and impact of Vanishing and Exploding Gradient. Given that, the simple model had the following architecture

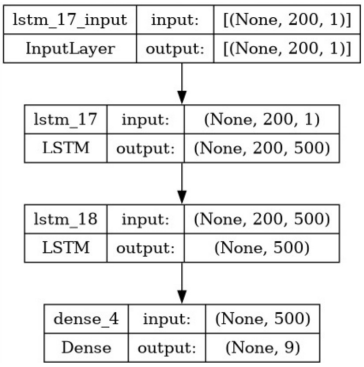

Figure 3: First Model Architecture

After testing, it was possible to see that a network with this amount of parameters was leading to an overfit. To avoid this, a few layers of dropout were added, reducing the variance by "ensembling" the network.

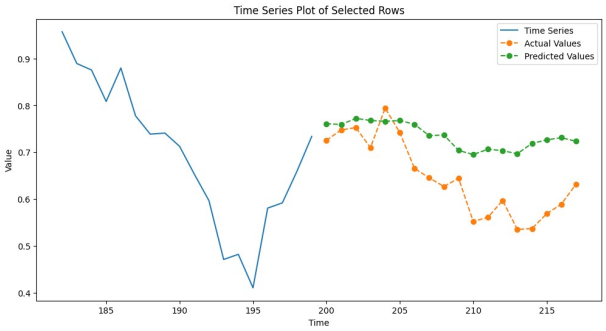

Figure 4: First Model Results example of prediction (MSE = 0.0124)

As we can see, the model indeed has learned something regarding the patters from the process, however, it still struggles on obtaining a deeper understanding of the patterns and specially with regards to understanding more quickly the change in pattern. Given that, we explore a new models in order to make the models better.

3.4 Simple LSTM with Hyper Parameter Tuning

Now that we created a simple model and seen the performance of the task, we can now run and hyper parameter grid in order to find a better suited model.

During this development, many different grids and many different architectures and hyper parameters were tested, they mostly involved the number of LSTM cells and the learning rate.

```
Trial 10 Complete [00h 59m 20s]
val_mean_squared_error: 0.008696356788277626

Best val_mean_squared_error So Far: 0.008380582245687643
Total elapsed time: 09h 58m 36s
Model: "sequential"

Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 200, 140)          79520

lstm_1 (LSTM)                (None, 140)               157360

dense (Dense)                (None, 9)                 1269

=================================================================
Total params: 238149 (930.27 KB)
Trainable params: 238149 (930.27 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 5: Hyper parameter tuning with Keras Tuner example

This process was extremely important given that it could save a lot of time as well as helping to control the overfitting made by many LSTM cells.

In the end, it was possible to realize that the model indeed showed a better performance. However, the results still didn't showed a considerable fit to the time series data yet.

3.5 ConvLSTM and N-Beats

After some research on new architectures for more sophisticated time series predictions. Some new types of studies were found, the main ones were: ConvLSTMs and N-Beats architecture.

3.5.1 N-Beats

The N-Beats architecture [2] is an univariate time series architecture firstly proposed in 2019 (but still presenting updates until this last semester). This architecture is based on very deep stack of FC Layers along side with many backward and forward residual links.

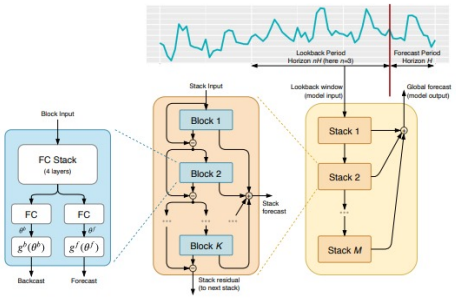

Figure 6: N-Beats scheme taken from the original paper

In order to apply this network (differently from the procedures made for the LSTMs), the Dart Library was again used given that they indeed have a PyTorch application of this architecture. However, the final result can be seen below, and, even thought the results show properity and, with the right hyper parameter tuning, could lead to results better than the classical optimized approach. The processing power required by this networks would surpass all known free tier GPUs.

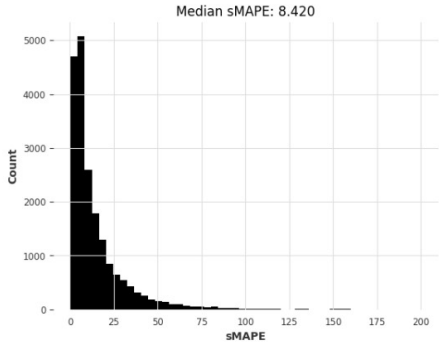

Figure 7: sMAPE of the N-Beats solution

3.5.2 ConvLSTM

It's worth noting that a ConvLSTM architecture was also tested and, even thought it's development is focused on capturing spatial features, it's also possible to reshape it for univariate features in order to try extracting time patterns by changing the kernel dimension to fit this uni variate data. The results didn't surpassed the simple LSTM implementation, so are not worth mentioning.

4 Conclusion

From the results obtained, it is possible to see that Hyper parameter tuning is a must in every single Data Science project (both in classic Machine Learning pipelines and Deep Learning ones). It's also worth mentioning that the LSTMs proved to be a great achievement regarding time series prediction, given that it successfully surpasses well know and consecrated solutions from the field (such as SARIMAX).

It's possible to conclude that the categories can have a good influence in this process and should be better considered and perhaps be included in the model as one hot encoded features, however, due to time constraints and lack of organization of the time, it was not possible to explore these features further than the process of separating the data and training different models.

Finally, it is also important to note that the solution development is a iterative process that requires to control the variance-bias trade-off by searching methods to control overfitting (something that happens with considerable frequency on large neural networks). During this project, this tradeoff was explored by using Dropout layers and Number of neurons control.

5 Contributions

All members were involved during all steps of the project, as we worked together. Notably:

Felipe Azank dos Santos: Data Preprocessing, Data Analysis, Architectures development and report

Felipe Bagni: Exploratory Data Analysis, finding the problematic data, LSTM architectures and hyperparameter tuning.

Domingos Savio: LSTM tuning and ConvNet development

References

[1] J. Herzen, F. LÃ¤ssig, S. G. Piazzetta, T. Neuer, L. Tafti, G. Raille, T. V. Pottelbergh, M. Pasieka, A. Skrodzki, N. Huguenin, M. Dumonal, J. KoÅ›cisz, D. Bader, F. Gusset, M. Benheddi, C. Williamson, M. Kosinski, M. Petrik, and G. Grosch. Darts: User-friendly modern machine learning for time series. *Journal of Machine Learning Research*, 23(124):1–6, 2022. URL `http://jmlr.org/papers/v23/21-1177.html`.

[2] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=r1ecqn4YwB`.