POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Politecnico di Milano

054307 - Artificial Neural Networks and Deep Learning

2021

Homework 2

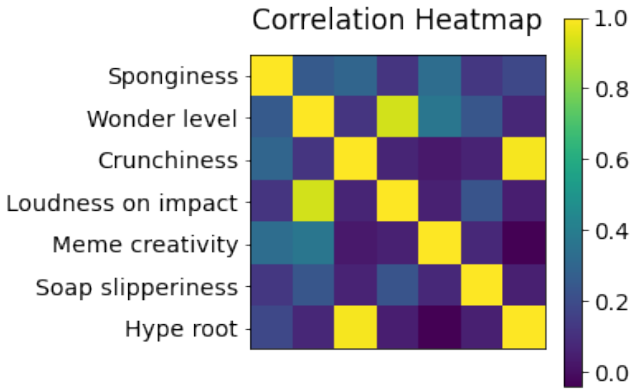
Student:	**Student ID**
Rasmus Tuxen	10844184
Davide Carnevali	10635813
Riccardo Morandi	10609940

Leaderboard Nickname:
DRTNet

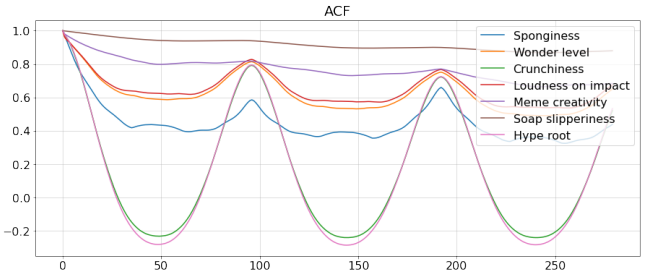
January 21, 2022

1 Data Exploration

We started by performing some exploratory data analysis. The original dataset was composed of 68528 entries each with 7 attributes, the fact that the labels of these attributes was cryptic forced us to use some statistics to better understand the data. we started with calculating the correlation matrix between the different attributes as seen in 1a, we found that the attributes `Crunchiness` and `Hype root` have an extremely high correlation of 0.987 and `Wonder level` and `Loudness of impact` have high correlation of 0.93. given the high correlation of `Crunchiness` and `Hype root` we looked for a possible deterministic law between the two such as a lag or a transformation without great success.
Looking at the data we noticed a periodicity, to validate our hypothesis we decided to use the autocorrelation function on the data and we noticed, as shown in 1b, that the data had a period of 96, this period was pretty pronounced in all of the attributes except for `sponginess` which presented a different behavior from the others; the existence of a period will be useful in the following for feature engineering and forecasting.
We also tried to examine the trend of the data plotting rolling statistics without particular successes.
On the other hand we noticed that in each attribute there were some intervals where the data was missing and was replaced with a constant value, and that within each attribute there were different type of patterns. Both of those features caused some difficulties in the training of the models.



(a) correlation matrix



(b) autocorrelation function

Figure 1

2 Data Preprocessing

Due to the periodic behavior found in the data, we decided to add a few extra features to the dataset which could help a network to identify where in the period the data was coming from. This is a common practice in time-series data.
Therefore two additional features `sin` and `cos` which, as the names imply, are the sine and cosine function with a period of 96. The purpose of these are such that the networks would be aware of the possibility of a certain repeated pattern in the data and would be able to better make predictions with this information.

This would mean that the networks would be given 9 features as input and only output the original 7 features.

All the data was normalized to be zero-mean and with unit variance. In figure 2 the resulting normalization of the data can be seen.
As the task of the project is to predict future values from the given dataset, it will be necessary to save the constants used in the normalization. This is done such that the predictions can first be made in normalized-space, where the models are trained, and then transformed back into the original data space to yield the lowest possible RMSE error.

2.1 Strategy for training and validation

The given dataset of 68528 observations was split with a train/validation ratio of 0.2, where all the data going to the validation set were taken from the end of the dataset. This is the general approach for time forecasting as it mimics the test scenario where the future is unknown.

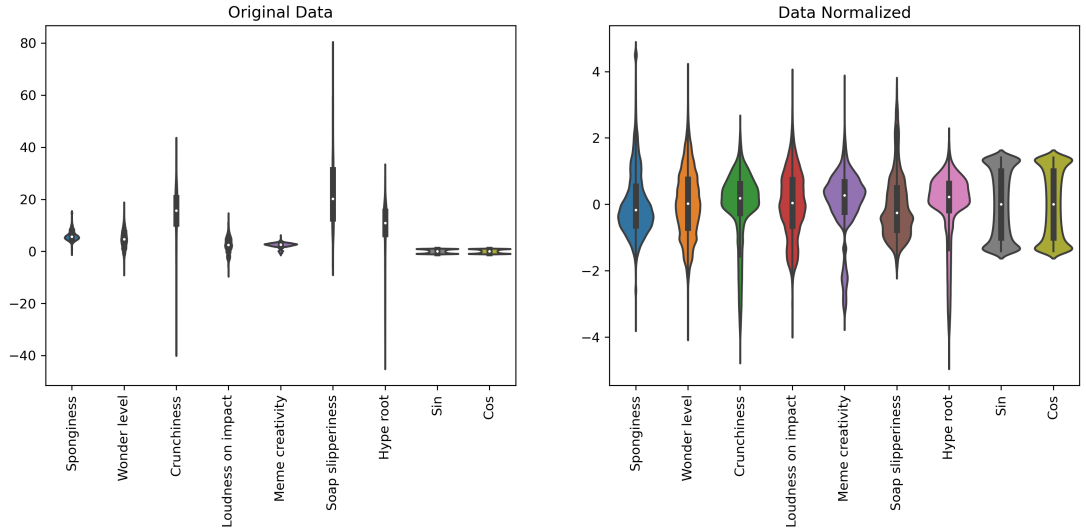


Figure 2: Visualization of data normalization

3 Experimentation with model architecture

As far as model architecture goes we experimented several different approaches.
The worst performing by far was the purely convolutional approach, we tried two different sub cases: the first was a conventional CNN formed of `conv1D` layers of small kernel size, the second was a custom network that had different braces of subsequent `conv1D` layers with different kernel sizes in order to capture the features of the data at different interval length, these branches were then combined with a `keras.layers.add` layer. Both of those sub-cases failed to capture adequately the patterns of the data and were therefore abandoned quickly.

A second approach was based on the `LSTM` modules directly implemented in keras, after a few experimentation we noticed that using bidirectional`LSTM` modules followed by one-dimensional convolutions worked best in this case.
We started with testing a convolutional `LSTM` with bidirectional layers. As expected, the net performed better if built and trained in an autoregressive way, using a window equal to 200, setting 10 as stride and predicting only one value each step. Adding another bidirectional layer didn't increase the results. The model obtained a test score of 5,5947.

3.1 Encoder-Decoder architecture

The main network we decided to focus on is an Encoder-Decoder, with `LSTM` layers, a `RepeatVector` layer, and a `TimeDistributed` layer to produce the output. The first trials, conversely of the other net, showed that instead of learning to forecast correctly the values of the future, the model was learning to produce the mean of the data provided as input. For this reason, we initially stuck to a direct forecasting technique, with a telescope region equal to the number of values to forecast. Using a model with two `LSTM` layers increased the performance, but using 3 or more worsened the results. At last, the final model implemented is an Encoder-Decoder (2 `LSTM` layers respectively) with skip connections. Since some values were left out before splitting in validation and training set, a final retraining adding also the test set increased the score from 4.9084 to 3.9647. The parameters passed to the `fit()` method are the validation size, EarlyStopping to minimize the validation loss, and `ReduceLROnPlateau` to reduce the learning rate in case of a slowdown of the learning process.

3.2 Forecasting window size

We tried different approaches for the size of the forecasting window: the first consisted in using the window of the previous 200 points to predict the single subsequent point in a purely autoregressive manner, the second consisted in predicting the whole period of 96 values in an attempt to make the model learn the patterns present in the sequence. A k-fold split was used both for model selection and training. Instead of predicting all the 864/1152 values we needed for the submission we decided to use a smaller size each step, and a blocking time series split loop was used to find out the best size. The result showed a valued around 100 performed slightly better than the others, and with the information obtained from the analysis of the data, we validated as final prediction window size 96. We also trained the final model with the time series split technique shown in 3, but unfortunately it didn't increase our best score.

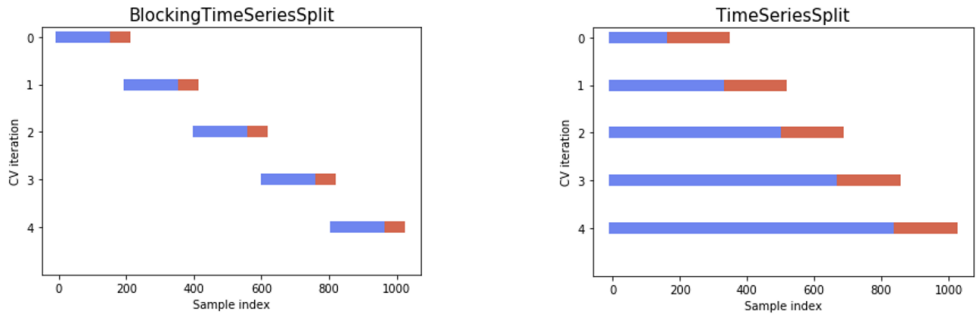


Figure 3: Cross-Validation strategies for Time Series forecasting

4 Conclusions

After the models had been trained it was time to perform autoregressive forecasting. In this process the model is given the last window from the validation set, makes a prediction, and then continuously uses its own previous predictions to make future predictions. In this way the model could in theory continue the time series indefinitely.

The autoregressive forecast predictions made by the best performing model on PHASE 1 can be seen in figure 4.

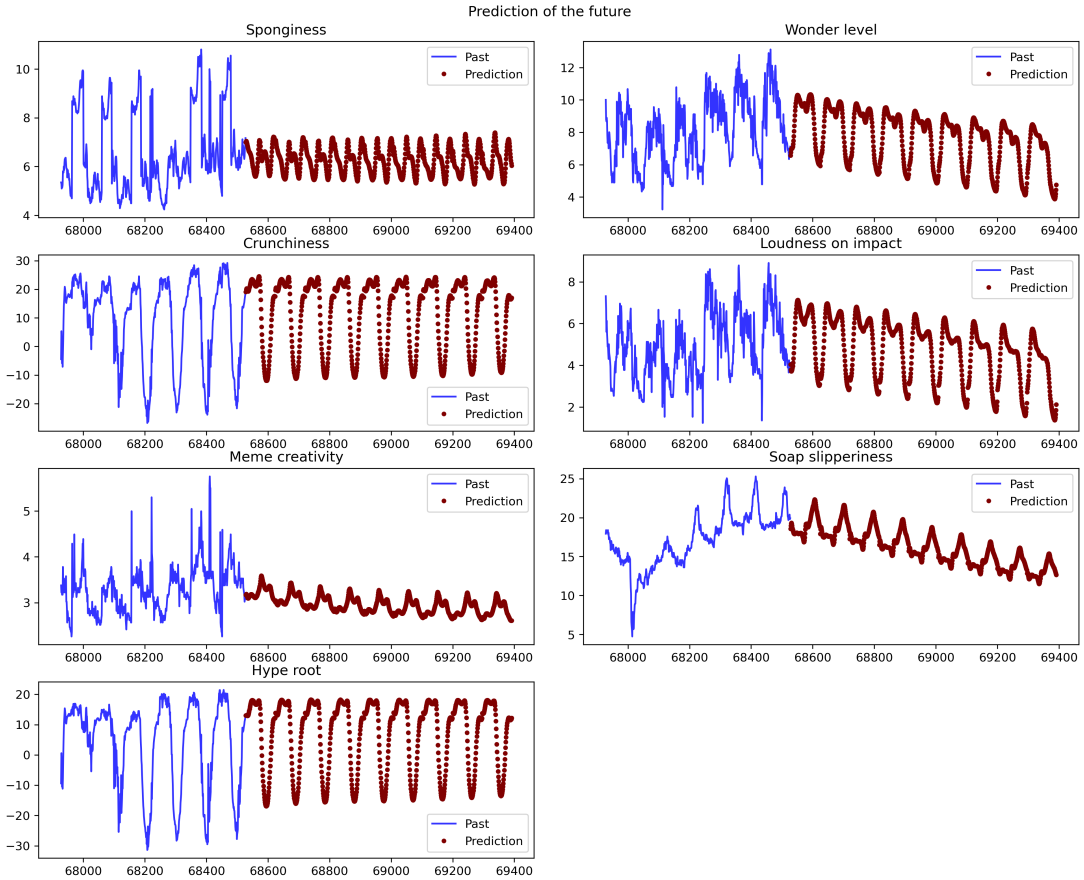


Figure 4: Visualization 864 predictions into the future corresponding to the submission to PHASE 1. For easier viewing the entire original dataset is not included, but only a part.