KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Smart Distribution Systems - B-KUL-H00P3A
Forecasting Assignment

Carlos Menéndez Perdomo (r0979414)
Stefania Andreea Blasko (r0979717)
Master of Engineering: Energy
Academic Year 2023-2024

1 Introduction

The approach taken for this assignment has been similar to the one followed during the exercise sessions for Smart Distribution Systems.

As will be explained in the following chapter, the initial approach added a hyperparameter and architect tuner using the "KerasTuner"[1]. However, due to overfitting, complications getting the final results, and comparison with actual values for the predicted time interval (although the metrics themselves seemed to be good), this approach has been discarded for a more manual and maybe naive methodology. However, one of the architectures obtained has been used as a template for the one used.

On a side note, although the results may be not good enough, the insights and programming knowledge gained have been invaluable.

2 Methodology

The steps on the exercise sessions have been followed to a certain extent. First, a clean up and validation of the data has revealed that the column of `"Price_BE"` is actually not in a numerical format, and NaN values for the same column appear on the time interval to be predicted. This last part is to be expected but the neural network will have trouble with them if included in the training set.

A plot of the original prices against time yields the following Figure 1.

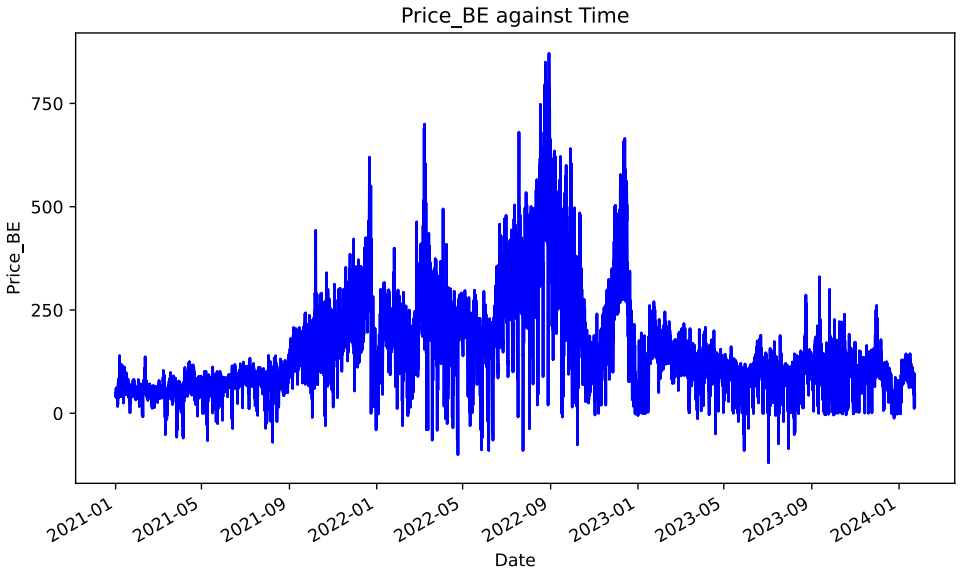

Figure 1: Price for the dates in the .CSV

Correlation checks are performed but yield no significant results.

Initially, a tuner was set for hyper-parameter and architecture selection, as explained be-

fore. However, it has been used only as a template for the final model.

Once the data has been validated and curated to ensure that the values had the correct format and that no NaN values would be fed to the model, the selection of features, target, training, test and validation sets are selected.

The features selected are all the parameters included in the original csv, but the column `"Price_BE"` has been shifted by a week. This is done to emulate the interval of prices with NaN values that need to be predicted.

The split in training, test and validation sets is performed with the help of the built-in tools from sklearn [2].

A sequential model is used, shown in the following Figure 2

```
model = Sequential([
    Dense(128, input_shape=X_train_scaled.shape[1:]),
    LeakyReLU(alpha=0.5),
    Dense(64),
    LeakyReLU(alpha=0.5),
    Dense(32),
    LeakyReLU(alpha=0.5),
    Dense(1, activation='linear')
])
```

Figure 2: Architecture for the model

This architecture is loosely adapted from one of the models proposed by the method of random search in the KerasTuner. Although the suggested model was not used, the number of layers and the initial number of neurons were taken from this approach. The code referring to the model search is in the following Figure 3.

```
def build_model(hp):
    model = keras.Sequential()
    model.add(layers.Flatten(input_shape=(X_train_scaled.shape[1],)))
    # Tune the number of layers.
    for i in range(hp.Int("num_layers", 1, 3)):
        model.add(
            layers.Dense(
                # Tune number of units separately.
                units=hp.Int(f"units_{i}", min_value=32, max_value=512, step=32),
                activation=hp.Choice("activation", ["leaky_relu","relu","tanh","sigmoid"]),
                #input_shape=(X_train_scaled.shape[1],)
            )
        )
    model.compile(
        optimizer = RMSprop(learning_rate=1e-5, rho=0.9, epsilon=1e-07),
        loss="val_loss",
        metrics=["accuracy"],
    )
    return model
```

Figure 3: Model search

After some trial and error, the mean square error and mean absolute error seem to not change much after 150 epochs, so this is the value chosen.

Different alternatives using "ReLU", "tanh" and "sigmoid" activation functions have been tried, but have been ultimately been discarded in favour of LeakyReLU. This decision has been made due to ReLU mapping negative inputs to zero [3], which is undesirable if there are negative prices. The alpha value for LeakyReLU has been adjusted manually, with not much difference in results.

3 Results

A plot of the comparison of the predicted price and the original for the test set is shown in Figure 4

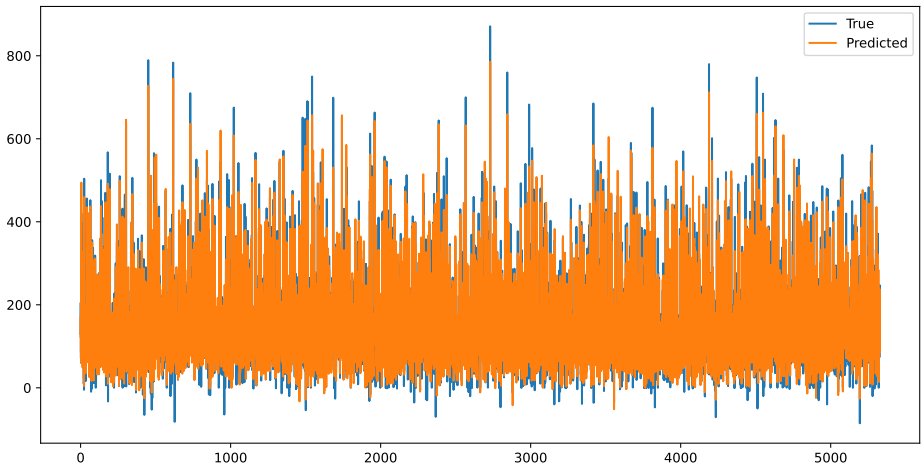

Figure 4: Test set price and predicted price

The metrics for this model fit are presented in Table 1

MSE @ 150 epochs	1142.6567
MAE	22.66

Table 1: Metrics for the model fit

Additionally, the results are compared with the actual day ahead prices in Belgium for the predicted interval [4]. This is shown in Figure 5. In this case the MSE is 498 for this short interval.

After several trials and different architectures and parameters, the neural network does not seem to be able to bias towards negative and/or values close to zero, even though that was the intention of using LeakyReLU. However, it is somewhat accurate in predicting peaks, or at least their placement during a typical day.

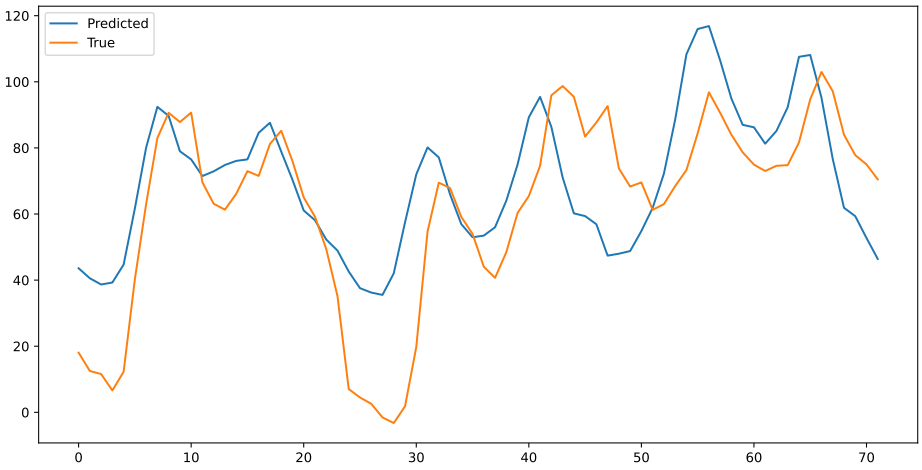

Figure 5: Actual price and predicted price for the requested interval of time.

Bibliography

[1] Tom O'Malley et al. *KerasTuner*. https://github.com/keras-team/keras-tuner. 2019.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html. Accessed: 2024-05-12.

[3] Xuan Qi et al. "Comparative Analysis of the Linear Regions in ReLU and LeakyReLU Networks". eng. In: *Communications in Computer and Information Science*. Vol. 1962. Communications in Computer and Information Science. Singapore: Springer Nature Singapore, 2024, pp. 528–539. ISBN: 981998131X.

[4] *Energy-Charts Electricity production and spot prices in Belgium*. https://energy-charts.info/charts/price_spot_market/chart.htm?l=en&c=BE&stacking=stacked_absolute_area&legendItems=010001&week=04. Accessed: 2024-05-12.