Machine Learning & Content Analytics

Project Report - 2020-2021 PT

Short Term Electrical Load Forecasting with Neural Networks

Pratikakis Emmanouil	p2822017
Dimopoulos Giorgos	p2822021

Contents

1. Introduction

This project concerns the development of a model used for predicting hourly values of electrical load based on neural networks. Load forecasting is used by utilities companies to meet the short term, medium term, or long-term demand. Nowadays every utility company should utilize some kind of forecasting tool in order to calculate possible imbalances in their demand, and their electric power generation must be balanced with their demand for electric power. Load timeseries have certain characteristics, such as load patterns tendencies due to usage and patterns that recur on certain days, weeks or seasons that should be taken into consideration when building a forecasting model.

1.1 Vision / Goals

The aim of this project is to provide a forecasting tool that can be used by utilities companies to predict their short-term demand. Short term refers to the day ahead electrical load, hence our goal is to build a model that can accurately predict load hourly values up to 24 hours into the future. Many companies already use statistical forecasting models for this purpose, and we are going to investigate whether deep learning models such as artificial neural networks can also accurately forecast electrical load timeseries. Since load consumption is affected by many exogenous factors such as extreme weather conditions or usage patterns such as holidays and/or weekends, we need a model that can capture all these relationships and provide adjusted forecasts in the short-term future.

2. Methodology

Our methodology is based on artificial neural networks (ANN) and recurrent neural networks (RNN). Also, Long short-term memory (LSTM) networks, a special kind of RNN that have the capabilities to learn the long-term dependencies, are being explored. In this project, we have used electrical load data with exogenous variables such as temperature, humidity, and wind speed to train these networks. In the following paragraphs we are going to describe the different models applied and the drawbacks and benefits of every model.

2.1 Dataset Overview

For the purpose of this project, we are going to use load consumption data that are publicly available from the ENTSO-E. ENTSO-E is the European Network of Transmission System Operators for Electricity and collects load consumption data from most European countries. The Greek transmission system operator (ΑΔΜΗΕ) publishes every day the load consumption in hourly intervals in MW for the entire Greek power system which are reported to ENTSO-E. These timeseries data can be downloaded from ENTSO-E Transparency Platform and are similar in terms of seasonality and peak hours to the consumption data each power company stores. We are also going to use additional data sources such as weather data for Athens and Thessaloniki since the temperature or extreme weather conditions may affect the electric load consumption.

2.1.1 Electric load data processing

The load consumption data that we downloaded from the ENTSO-E regard the dates from 1 June 2018 to 30 June 2021 and have the following format:

	Time	**Load**
0	01.06.2018 00:00 - 01.06.2018 01:00	5047.0
1	01.06.2018 01:00 - 01.06.2018 02:00	4696.0
2	01.06.2018 02:00 - 01.06.2018 03:00	4560.0
3	01.06.2018 03:00 - 01.06.2018 04:00	4482.0
4	01.06.2018 04:00 - 01.06.2018 05:00	4465.0
…	…	…

Table 2.1 Initial load consumption data

We begin our process by keeping the first part of the date before the dash ('-'). Then, we drop duplicate dates in case they exist. After that, we must covert the date to the format YYYY-MM-DD HH:MM:SS. Lastly, in case there is no 1-hour granularity, we resample per hour and forward fill the missing hours. In that way, we have transformed the load consumption data in the following way:

	Time	Load
0	2018-06-01 00:00:00	5047.0
1	2018-06-01 01:00:00	4696.0
2	2018-06-01 02:00:00	4560.0
3	2018-06-01 03:00:00	4482.0
4	2018-06-01 04:00:00	4465.0
…	…	…

Table 2.2 Transformed load consumption data

2.1.2 Weather data processing

The weather data were scraped from free meteo website by using a script that is built with selenium in python. We chose the two biggest cities of Greece, Athens and Thessaloniki. The reason is that the majority of the population lives in those two cities, hence they affect the electrical load the most. As before, the dates of our concern are from 1 June 2018 to 30 June 2021. The weather characteristics that have impact on the electric load are the temperature, the humidity and the wind speed. An obstacle that we faced was that for some dates there were not data for those characteristics. In order to overcome this, we filled those dates with the same data of the respective date from the previous year.

After scraping the weather data for Athens and Thessaloniki, we can transform them to a timeseries format with numeric values. Regarding the humidity, the temperature and the wind we removed the percentage ('%'), the Celsius ('°C') and the Beaufort ('Bf') symbols, respectively. The date was scrapped written in full words and in half hourly intervals. As before with the electric load, it was transformed to the format YYYY-MM-DD HH:MM:SS. Then, we grouped the data to hourly intervals by taking the mean of the values for each half hour. Lastly, there were missing data for some hours, so we resample per hour and forward fill those. The two files that have been created, one for each city, have the following layout:

	Temperature	Wind	Humidity	Date
0	19	1	66	2018-06-01 00:00:00
1	19	1	68	2018-06-01 01:00:00
2	19	1	64	2018-06-01 02:00:00
3	17	1,5	73	2018-06-01 03:00:00

4	16,5	1,5	75	2018-06-01 04:00:00
…	…	…	…	…

Table 2.3 Transformed weather data

2.1.3 Merge between electric load and weather data

As a first step, we join the data of Athens and Thessaloniki weather timeseries. Having the weather data in one file we concatenate them with the electric load data for all the years. Furthermore, we created two more variables, in order to assist better our models during the training phase. The one indicates the month of the year and the other flags the national holidays and weekends. As a result, we conclude in our final file:

	Date	**Load**	**Temperature Athens**	**Wind Athens**	**Humidity Athens**	**Temperature Thess**	**Wind Thess**	**Humidity Thess**	**Month**	**Holiday/ Weekend**
0	2018-06-01 00:00:00	5047.0	21.5	2.5	30.0	19.0	1.0	66.0	6	1
1	2018-06-01 01:00:00	4696.0	20.0	2.0	33.0	19.0	1.0	68.0	6	1
2	2018-06-01 02:00:00	4560.0	19.0	2.0	36.0	19.0	1.0	64.0	6	1
3	2018-06-01 03:00:00	4482.0	21.0	2.5	32.0	17.0	1.5	73.0	6	1
4	2018-06-01 04:00:00	4465.0	20.5	1.0	34.0	16.5	1.5	75.0	6	1
…	…	…	…	…	…	…	…	…	…	…

Table 2.4 Final data file

Before we feed our data to the neural network, we need to scale them to [0,1] range because, generally, it speeds up learning and leads to faster convergence. We did that via the

MinMaxScaler function, which uses the following mathematical formula:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

2.1.4 Exploratory data analysis (EDA)

Firstly, we explore the relationship between the electric load and the temperature of Athens. It can be clearly seen from the graph (§2.1) that at first there is a negative linear association among them that ranges from negative temperatures up to 18 °C. From there, the linear association changes to positive up until 40+ °C. This is expected since at very low or very high temperatures people use electrical devices to adjust the cold or the heat, respectively, resulting in the increase of the electricity consumption.

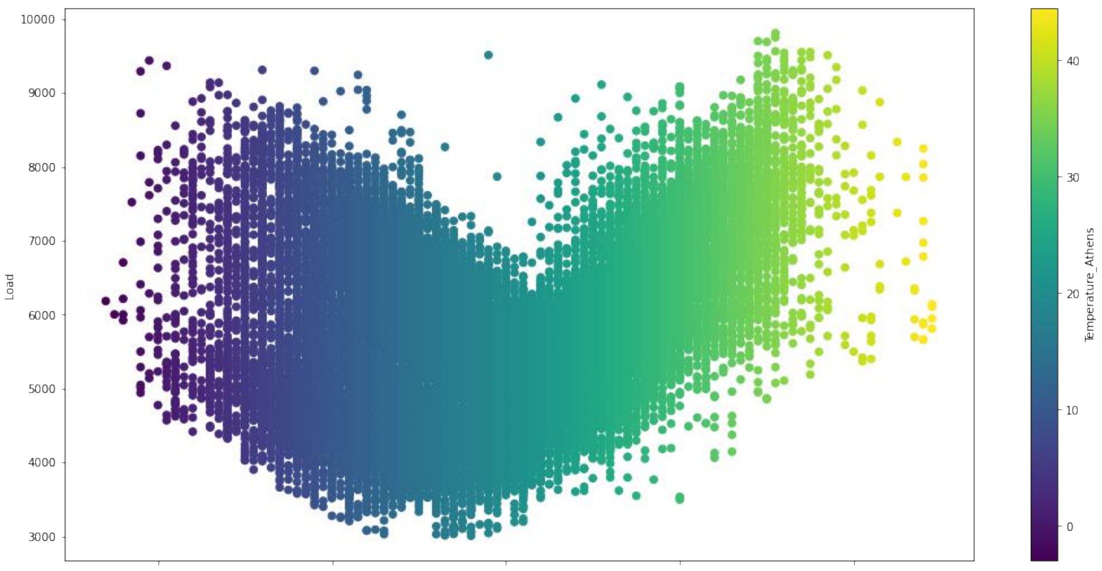


Figure 2.1 Paired association between electric load and temperature of Athens

The positive association is, also, been depicted in the figure (§2.2). There, we have separated in three different bins the temperature of Athens (Low (18-24°C), Medium (24-29°C), High (29+°C)) for the day of 1 June 2018. Almost all the low values of consumption are associated with low temperatures. On the other hand, great values of electric load belong to the medium and high bins, as expected.

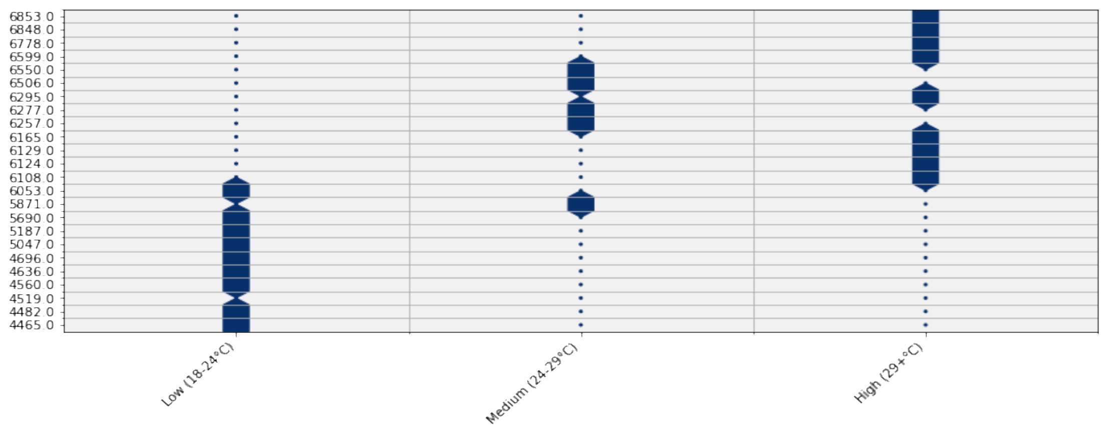
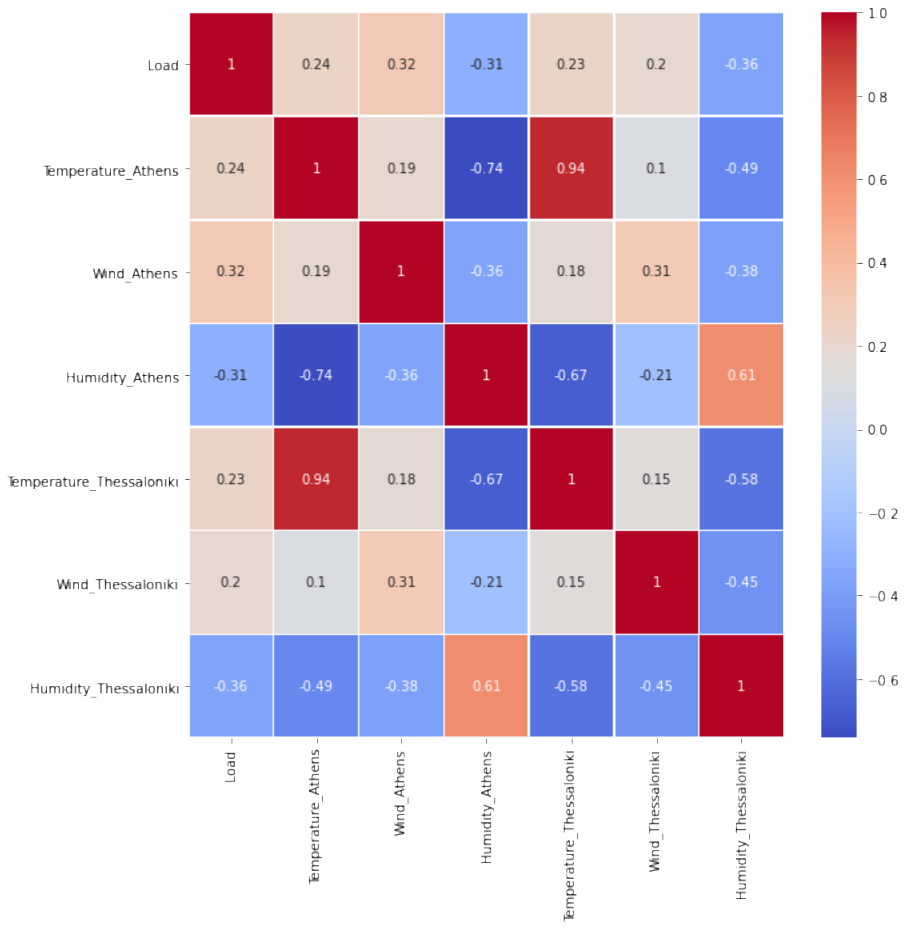


Figure 2.2 Electric load vs temperature of Athens divided in bins

Next, we find the pairwise correlation of all the variables (§2.3), by using the 'Pearson' correlation. The biggest one can be spot among the temperatures of Athens and Thessaloniki. Moreover, the humidity, in both cities, has negative correlation with the electric load.



	Load	Temperature_Athens	Wind_Athens	Humidity_Athens	Temperature_Thessaloniki	Wind_Thessaloniki	Humidity_Thessaloniki
Load	1	0.24	0.32	-0.31	0.23	0.2	-0.36
Temperature_Athens	0.24	1	0.19	-0.74	0.94	0.1	-0.49
Wind_Athens	0.32	0.19	1	-0.36	0.18	0.31	-0.38
Humidity_Athens	-0.31	-0.74	-0.36	1	-0.67	-0.21	0.61
Temperature_Thessaloniki	0.23	0.94	0.18	-0.67	1	0.15	-0.58
Wind_Thessaloniki	0.2	0.1	0.31	-0.21	0.15	1	-0.45
Humidity_Thessaloniki	-0.36	-0.49	-0.38	0.61	-0.58	-0.45	1

Figure 2.3 Pairwise correlations

Finally, on the following table (§2.5) we can see the Spearman's correlation coefficient between Load and all the other variables, along with p-value. The biggest correlation, in absolute value for the electric load, is the one with the humidity in Thessaloniki. It is, also, negative, meaning that when one of them tends to decrease the other increases.

	Spearman's correlation coefficient	**p-value**
Load and Temperature Thessaloniki	0.20793253392684044	1.05e-261
Load and Humidity Athens	-0.3048519936745509	0.0
Load and Wind Thessaloniki	0.19837633202037616	6.41e-238
Load and Wind Athens	0.3125897969613759	0.0
Load and Humidity Thessaloniki	-0.36464917630548216	0.0
Load and Temperature Athens	0.2380334013168545	0.0

Table 2.5 Spearman's correlation coefficient between load and all the other variables

2.2 Algorithms / Architectures

Initially we had to model the forecasting problem as a supervised learning problem in which given some input features x, our task would be to predict some output (target) variable y. Compared to other supervised learning problems, forecasting with supervised learning involves the time component that should be captured along with the exogenous variables' relationships.

In our case the target features or variable y are the future values of electrical load, while the input features are the n past values of electrical load and weather and calendar data. To this end, we need to define at this stage three key concepts that we used when building our models. The first is the output or **forecasting length**, which is the number of time steps we need our model to be able to predict. Since we are interested in short-term forecasting this output length is set to 24 values each corresponding to an hour of the day, hence our models will only be able to predict load values in MW for the day ahead. The second concept is the input or **past values length**, which is defined as the number of past timesteps our model will use as input to provide the forecasts. During the model's development stage, we applied different input length sizes ranging from 24 to 96 past values. Lastly, we use the **sliding / rolling window** method to produce forecasts. This method describes the way our data are used when training the model. Specifically in a sliding window forecast, we progressively use fixed length input arrays of n-past values of our input features to predict the next 24 future values. Since we are using neural network algorithms, we had to transform our data to [observations, input features], [observations, output features] format. In a sliding window forecast, each observation in the 2nd array holds the future 24 values of the corresponding observation in the 1st array. Each observation differs from the previous one by a fixed window size called the **sliding window size**. We applied two categories of models using 24h sliding window size and 1h sliding window size. The graph below depicts this process.

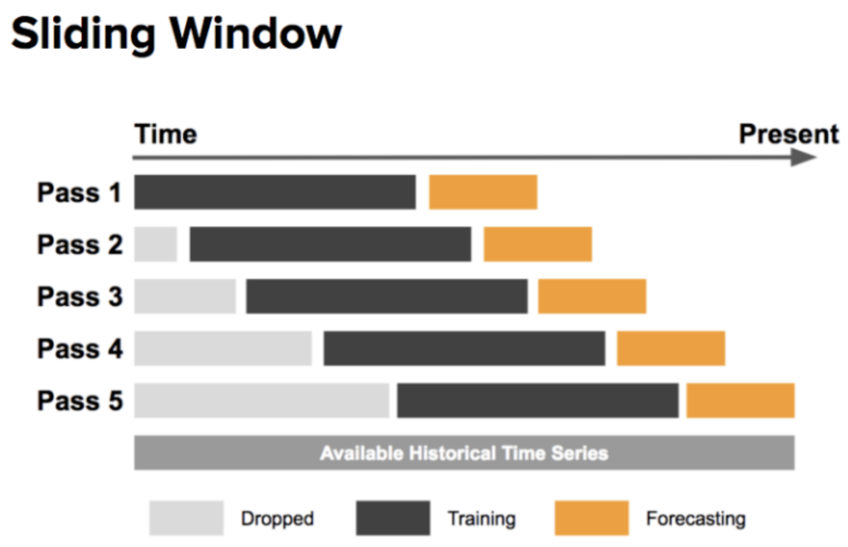


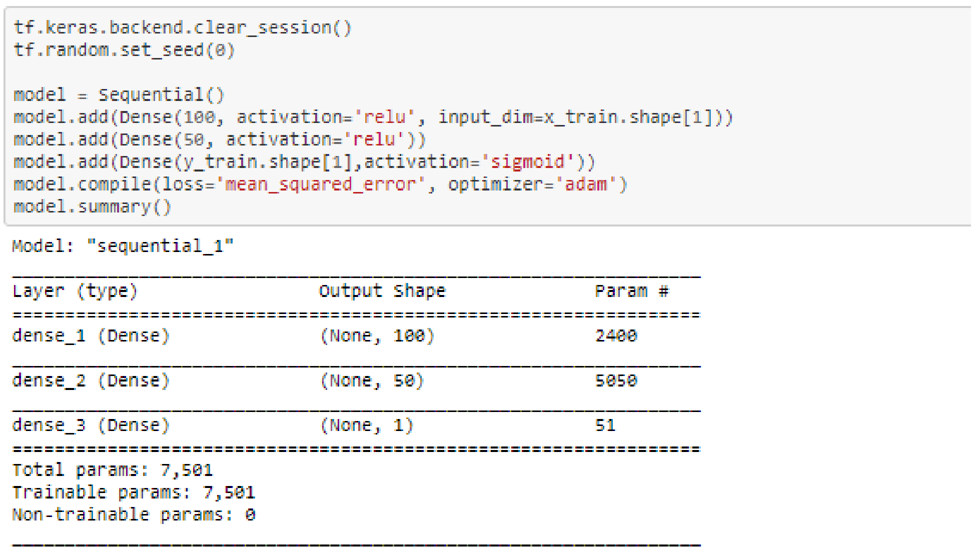
Figure 2.4 Sliding window process

Sliding Window Method

Source: https://eng.uber.com/forecasting-introduction/

2.2.1 Feed Forward Models

The first category of models we built were based on feed forward models using a sliding window of size 1. This means that every observation we have in our x and y arrays is shifted 1 hour ahead of the previous observation. The first model we created under this category uses 24h as the past values size, while the output length is 1h ahead.



```
tf.keras.backend.clear_session()
tf.random.set_seed(0)

model = Sequential()
model.add(Dense(100, activation='relu', input_dim=x_train.shape[1]))
model.add(Dense(50, activation='relu'))
model.add(Dense(y_train.shape[1],activation='sigmoid'))
model.compile(loss='mean_squared_error', optimizer='adam')
model.summary()
```

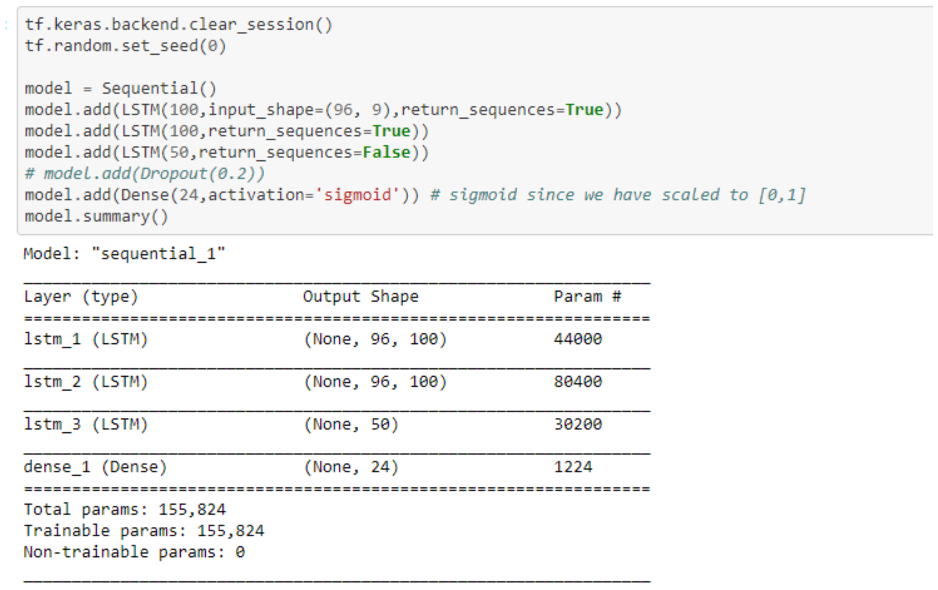
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_1 (Dense)              (None, 100)               2400      
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050      
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 51        
=================================================================
Total params: 7,501
Trainable params: 7,501
Non-trainable params: 0
_________________________________________________________________
```

Figure 2.5 Feed Forward Model with sliding window of size 1

Since we scaled our data to [0,1] scale as described in chapter 2.1 we used the sigmoid activation function in the last layer and the relu activation functions in other layers. The results are described in the next chapter. The drawback of this model is that it is necessary to have data of the last 24 hours so that we can predict a single value into the future. It is highly unlikely though, that we know the actual electrical load values in the past 24 hours (i.e. the time we choose to "hit the button" we need to know what the load consumption was 24h ago). To counter this issue, we are going to build the same model, but try a longer horizon of 48h. Now instead of predicting for the next 1 hour we are going to predict a single load value that is 48h ahead of our corresponding input values. With such a model, in a real case scenario we would use 24 hours of past data at most 2 days old to get a forecast for the next hour. This model’s results are also described in the next chapter.

2.2.1 Recurrent Neural Network Models

In this category of models, we used the sliding window method as described previously but instead of Dense layers, we used LSTM and GRU layers. Using LSTM or GRU layers requires our data to be in a three-dimensional shape [samples, timesteps, features]. The samples are the total observations we are going to use, the number of timesteps is the past values length and the features is the number of variables we are using as input features. This transformation is described in the notebook uploaded in our github repository. In this category we tried different input size lengths and sliding window lengths. The output length was set to 24h, hence this category of models uses n past values of all the available input features we described in chapter 2.1 to predict 24h of load values into the future. Below is a snapshot of one of these model's structure.



```
tf.keras.backend.clear_session()
tf.random.set_seed(0)

model = Sequential()
model.add(LSTM(100,input_shape=(96, 9),return_sequences=True))
model.add(LSTM(100,return_sequences=True))
model.add(LSTM(50,return_sequences=False))
# model.add(Dropout(0.2))
model.add(Dense(24,activation='sigmoid')) # sigmoid since we have scaled to [0,1]
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 96, 100)           44000
_________________________________________________________________
lstm_2 (LSTM)                (None, 96, 100)           80400
_________________________________________________________________
lstm_3 (LSTM)                (None, 50)                30200
_________________________________________________________________
dense_1 (Dense)              (None, 24)                1224
=================================================================
Total params: 155,824
Trainable params: 155,824
Non-trainable params: 0
_________________________________________________________________
```

Figure 2.6 LSTM model's structure

This specific model uses 9 input features and 96 past timesteps to predict 24 future values of electrical load.

We further investigated this category of models by creating a function which grid searches all combinations of parameters such as input size, batch size type of layer and rolling window size. The results from the grid search are described in chapter 3 as well as in our github repository.

3. Results

In this chapter we are going to present the results of the models' application described in the previous chapter as well as whether these models fulfill the business needs. In all our testing experiments we used a train test split. The metrics we defined to evaluate our models are forecasting accuracy metrics namely Root Mean Squared Error, Mean Absolute Error and Mean Absolute Percentage Error (%). These accuracy metrics were reported after using multiple different train test splits and averaging the results from all splits. We also took the execution time into consideration, as well as the Actual vs Predicted Graphs.

3.1 Feed Forward Model with Sliding Window Size 1

As described in chapter 2 this model uses the 24 past hours of load, weather and calendar data to predict 1 value of load for the next hour. The model's reported Root Mean Squared Error is:

Train Score: 240.16 RMSE

Test Score: 256.00 RMSE

We can also see the actual vs predicted plots for some random timestamps below:

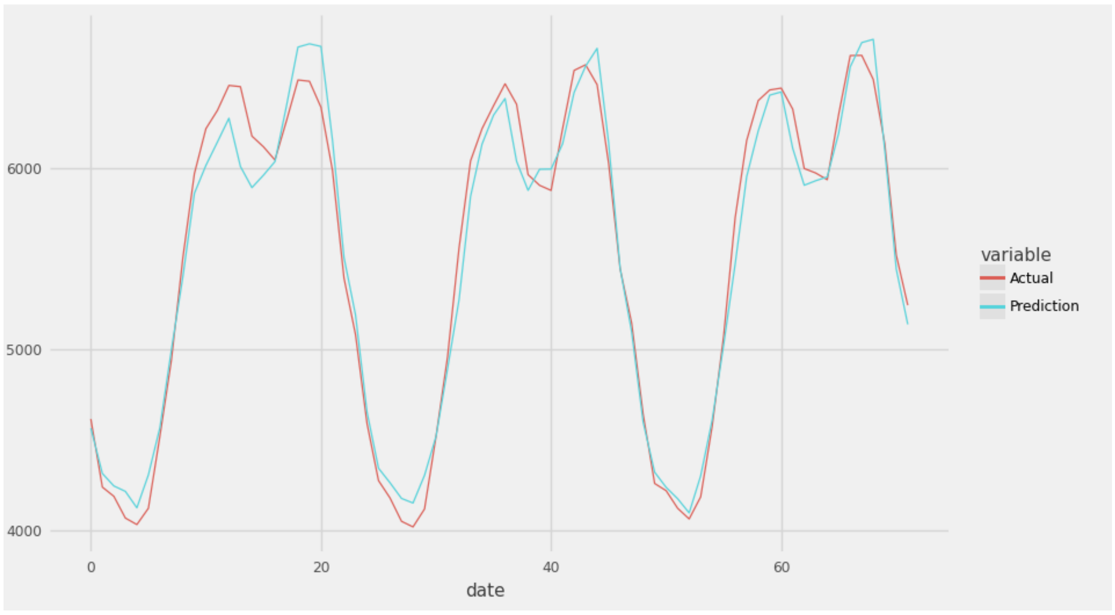


Figure 3.1 Actual vs Predicted electric load on random hours for FFM to forecast 1h

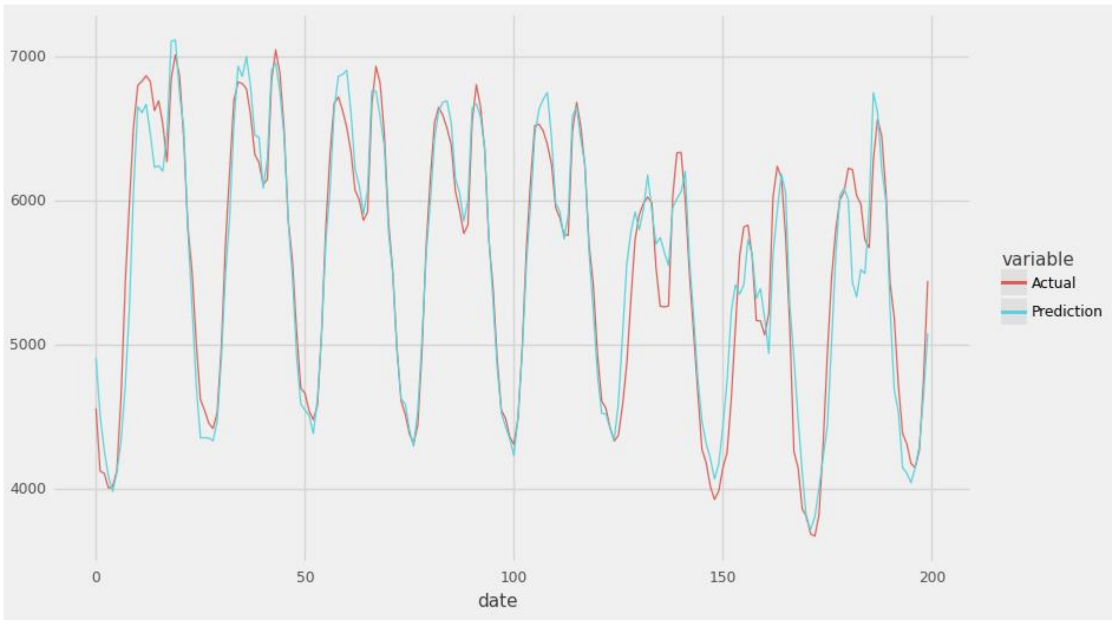


Figure 3.2 Actual vs Predicted electric load on random hours for FFM to forecast 1h

We notice that the accuracy in our predictions is acceptable. The predicted electrical load matches the actual load and we can see that our model is able to capture the seasonality and certain peaks of the actual load curve. As described in chapter 2 though, with this model structure we are only able to predict 1 future electrical load value while we need all exact previous 24 values as input. Hence this model does not meet our business requirements since utilities companies are not able to measure what their exact load consumption is 24 hours in the past.

3.2 Feed Forward Model with 48h Forecasting Horizon

We also try the same model but with a forecasting horizon of 48 hours, so in this model we can feed 24 past values that are 2 days old, so we can get a forecast for the next hour.

Train Score: 546.12 RMSE

Test Score: 565.41 RMSE

We can see that this model has almost double the RMSE compared to the previous one.

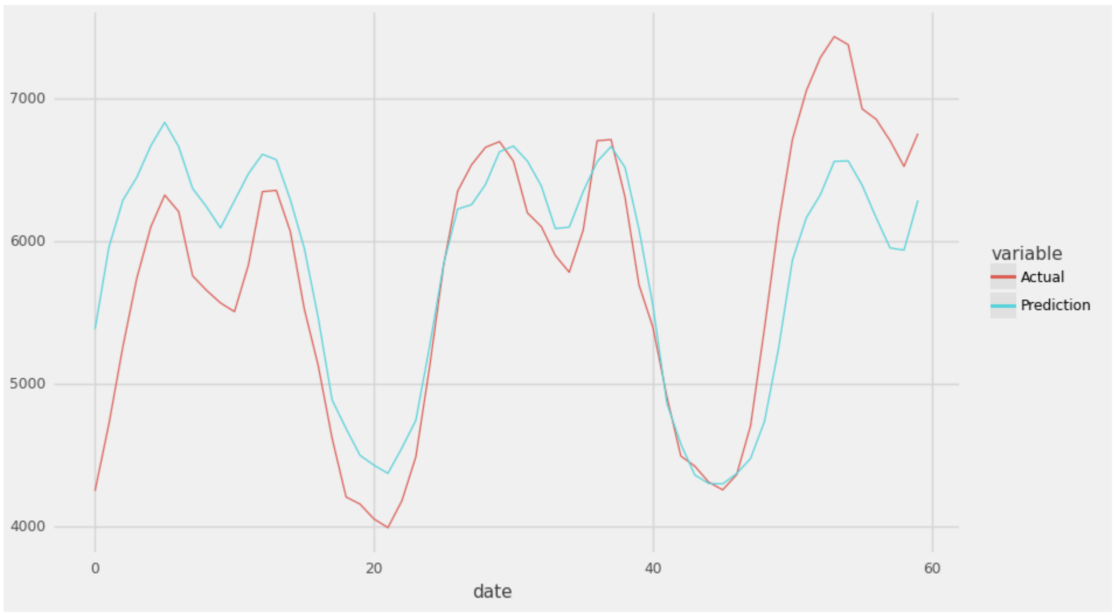
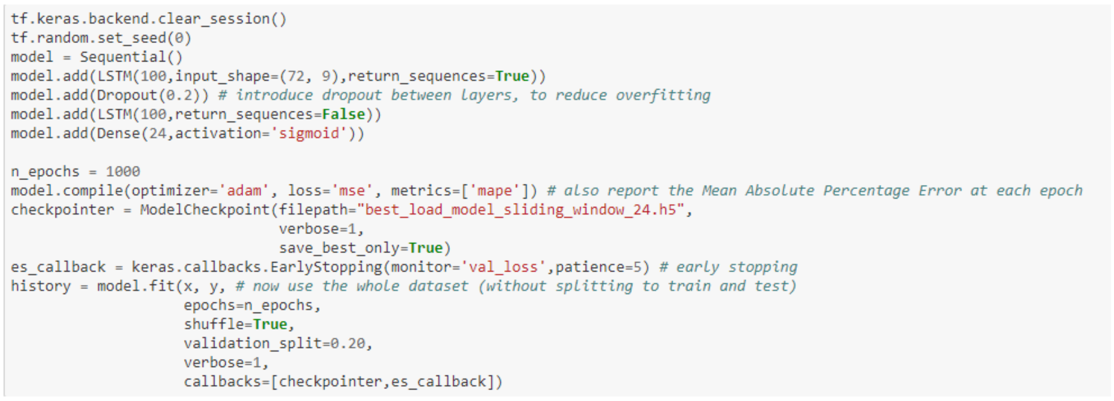


Figure 3.3 Actual vs Predicted electric load on random hours for FFM with 48h forecasting horizon

As expected, this model's accuracy is much less than the previous model. We can still see that the model has captured the electrical load seasonality, but it is not able to estimate all the peaks. In a real case scenario though, when using such models, we would have to do a long-range forecast since the last historical data we would have would be from at least 2 days ago while we need a 24h long forecast for tomorrow.

3.3 LSTM Model with Sliding Window Size 24 and Input Size 72

As described in our github repo ("*Trying best model on Validation Set*"), by applying the grid search function on different training and test sets, we ended up with a model built with LSTM layers and a sliding window of size 24, that uses the past 72 hours to predict the next 24.



```
tf.keras.backend.clear_session()
tf.random.set_seed(0)
model = Sequential()
model.add(LSTM(100,input_shape=(72, 9),return_sequences=True))
model.add(Dropout(0.2)) # introduce dropout between layers, to reduce overfitting
model.add(LSTM(100,return_sequences=False))
model.add(Dense(24,activation='sigmoid'))

n_epochs = 1000
model.compile(optimizer='adam', loss='mse', metrics=['mape']) # also report the Mean Absolute Percentage Error at each epoch
checkpointer = ModelCheckpoint(filepath="best_load_model_sliding_window_24.h5",
                               verbose=1,
                               save_best_only=True)
es_callback = keras.callbacks.EarlyStopping(monitor='val_loss',patience=5) # early stopping
history = model.fit(x, y, # now use the whole dataset (without splitting to train and test)
                    epochs=n_epochs,
                    shuffle=True,
                    validation_split=0.20,
                    verbose=1,
                    callbacks=[checkpointer,es_callback])
```

Figure 3.4 LSTM model after grid search function

This model had an average from all train test splits of:

Train Score: 306.65 RMSE

Test Score: 331.63 RMSE

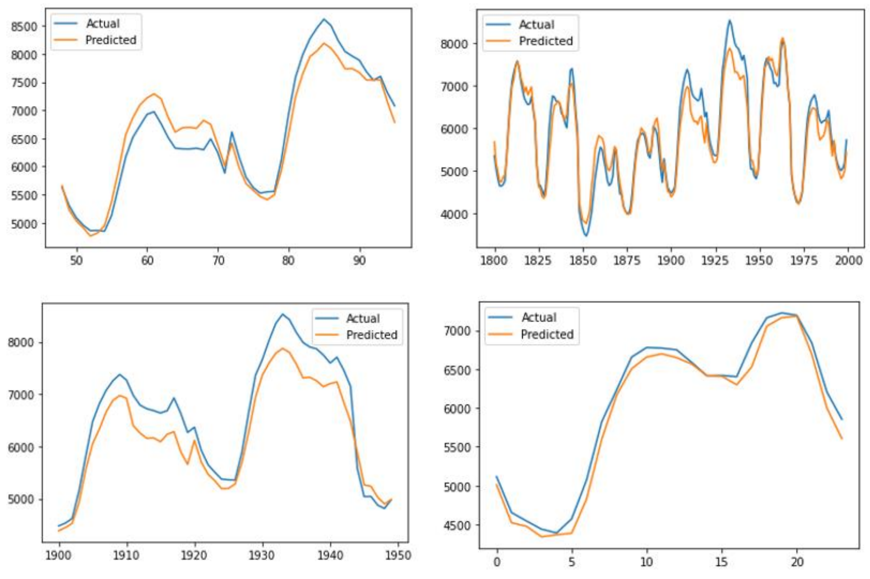


Figure 3.5 Actual vs Predicted electric load on random hours for LSTM model (slid.window 24)

We can see that the model has captured the trend and seasonality of the load timeseries and is also able to predict certain peaks of the timeseries. The Mean Absolute Percentage Error of this model is 6.9% and lastly, we can also check some residuals plots below:

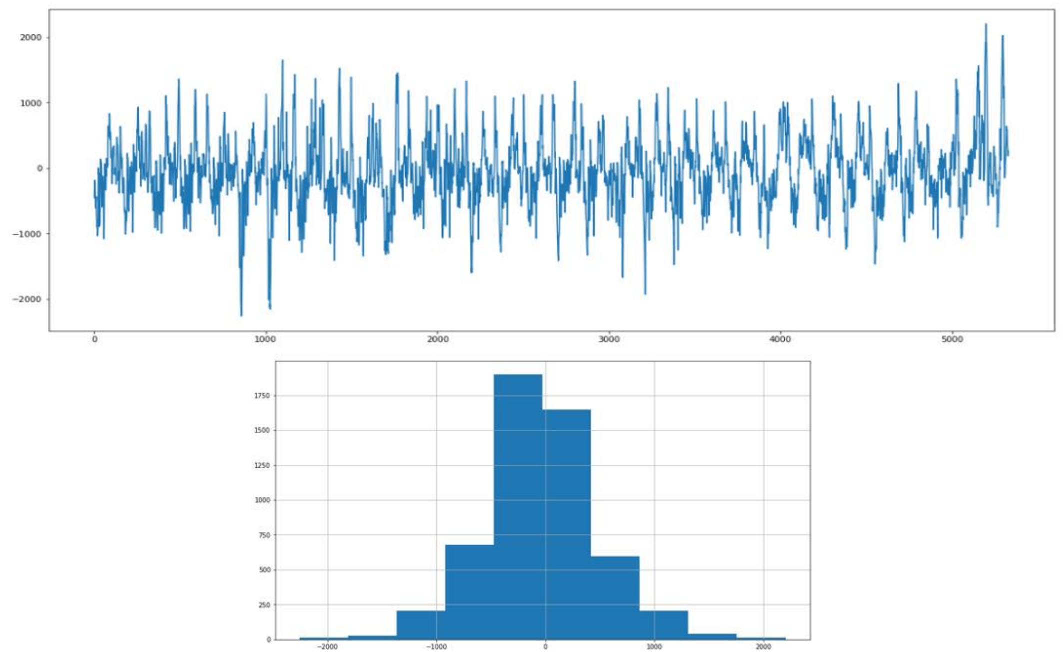


Figure 3.6 Residual plots of LSTM model (slid.window 24)

We can see that the residuals of this model appear to be normally distributed and centred around 0. There appears to be some seasonal component in the residuals, and this is an indicator that the model has not captured all peaks of the load timeseries, since the residuals are high and low at specific timestamps and not at random. This model fulfils our business needs since we can use it to predict the next 24 hours with a single model execution.

3.4 LSTM Model with Sliding Window Size 1 and Input Size 48

Finally, we apply a model with LSTM layers that uses a sliding window of size 1 and the past 48 hours, to predict the next 24 hours. This model gave us the best results in terms of predictive accuracy, although it had the highest execution time both in training and prediction stages.

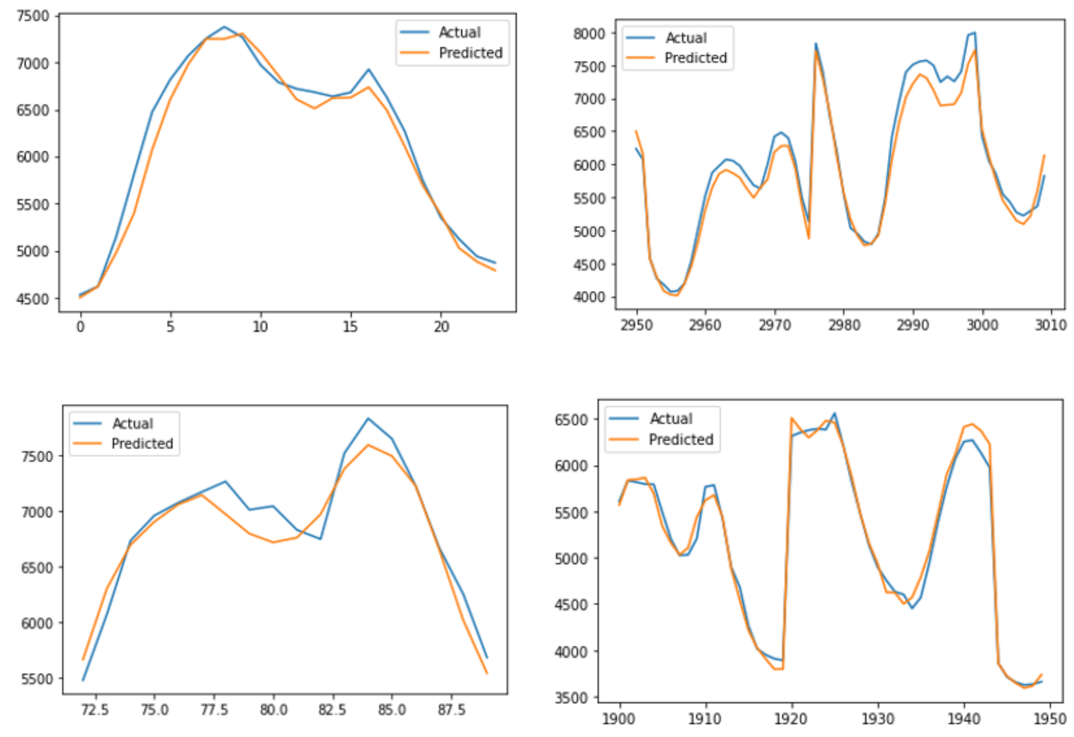


Figure 3.7 Actual vs Predicted electric load on random hours for LSTM model (slid.window 1)

The application of this model in five different train and test sets gave us the following error metrics:

Train Score: 227.42 RMSE

Test Score: 234.36 RMSE

Test Mean Absolute Error: 184.47 MAE

Mean Absolute Percentage Error: 3.21%

From all models we applied, we can conclude that incorporating weather data and calendar variables increases the predictive accuracy. LSTM layers produce more accurate forecasts and can adequately capture relationships between many different input variables and the output target. We can also conclude that setting a smaller sliding window size when training the models appears to have significant effect both in the increase of execution time but also in the decrease of the forecasting error.

4. Conclusion / Future Work

In this project, we dealt with the forecasting of short-term electric load using artificial neural networks (ANN) and recurrent neural networks (RNN), with timeseries data. We tested different input sizes and sliding windows and we were able to predict up to 24h ahead of time. The best predictive accuracy was achieved with Long short-term memory (LSTM) networks with sliding window size equals to one. The results are satisfying considering the fact that the model not only captured the trend but also most of the peaks of the electric load.

Future work can expand the research on medium-term load forecasting, with the goal to enforce a power system on the future energy demand. For example, forecasting monthly consumption with Multi-layer perceptron (MLP), which is a class of feedforward artificial neural network (ANN), or other deep learning techniques. Furthermore, the models can be better enhanced with more features in the input data. The load profiles during various holidays, such as Christmas or New Years, and Easter have specific shapes; thus, using a single categorical input for all holidays is insufficient. Therefore, more categorical variables can be created in order to indicate the occurrence of the three mentioned holidays. Lastly, other external factors that may affect the consumption of the electric load can be examined.

5. Appendix

Members/Roles

Manos Pratikakis: Provide domain knowledge, focus on model development and testing scenarios, apply different architectures to meet business needs, check if business requirements are met, cooperate in code development.

Giorgos Dimopoulos: Research on existing work, focus on data collection, quality and adequacy, data transformation for input to model, parameter tunning, cooperate in code development.

Time Plan

June	• Decide on the data sources to use & volume of data • Collect data and check data quality and format • Research on existing work

July	• Construct and apply different models and architectures (training, evaluation, parameter tunning) • Refine or enrich data, check for additional data that can be useful • Measure metrics such as forecasting error, complexity, execution time and shortlist best models
August	• Conclude to a final model • Write the report to summarize findings, methodology description and results • PPT presentation
September	Handover

Bibliography

[1] https://www.entsoe.eu/Technopedia/techsheets/enhanced-load-forecasting

[2] https://ieeexplore.ieee.org/abstract/document/9064654

[3] https://towardsdatascience.com/multiple-stock-prediction-using-deep-learning-network-d19a7acd8551

[4] https://towardsdatascience.com/preprocessing-time-series-data-for-supervised-learning-2e27493f44ae

[5]https://www.sciencedirect.com/science/article/abs/pii/S0306261919301217?dgcid=raven_sd_recommender_email

[6]https://www.sciencedirect.com/science/article/pii/S2405896318305652?dgcid=raven_sd_recommender_email

[7] https://machinelearningmastery.com/how-to-develop-lstm-models-for-multi-step-time-series-forecasting-of-household-power-consumption/

[8] https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

[9] https://machinelearningmastery.com/how-to-develop-lstm-models-for-multi-step-time-series-forecasting-of-household-power-consumption/