Energy Consumption Prediction using Machine Learning

Time Series Forecasting

1 Introduction

The electricity sector in Finland relies on nuclear power, the forest industry, black liquor and wood consumption, cogeneration, and electricity import from neighbouring countries (Electricity sector in Finland, 2022). In 2020, electricity production in Finland amounted to 66.6 TWh, of which 34.7 TWh was produced with renewable energy sources. This corresponds to 52 percent of Finland's electricity production (Statistics, 2021).

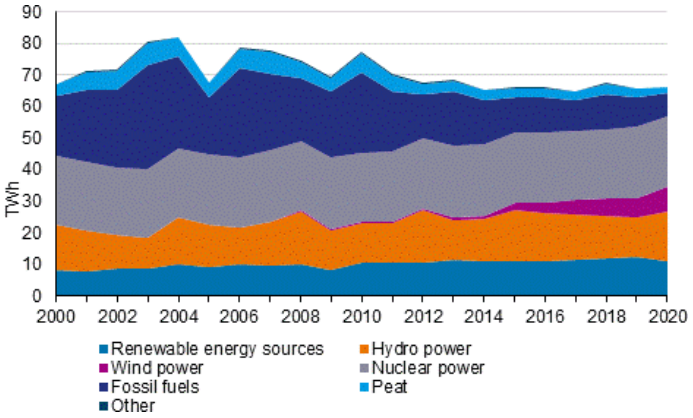


Figure 1. Electricity generation by energy source 2000- 2020 (Statistics, 2021)

The ability to forecast electricity load requirements is one of the most important aspects of the effective management of power systems. The quality of the forecasts directly impacts the economic viability and the reliability of every electricity company.

In recent years, artificial intelligence in general and machine learning in specific present solutions for a huge number of problems. One of the promising applications is predicting future energy consumption using historic time series by acquiring its imitation model.

This project aims to test whether it is possible to apply the famous deep learning model LSTM to the time series data, which contains the electricity consumption in Finland. The aim is to yield good enough results in forecasting energy consumption. The model will be trained using a dataset from Fingrid (Finland's transmission system operator), which contains 6-year of electrical consumption in Finland, where it is univariate time series as it contains one feature.

2 Machine Learning

2.1 Fundamentals

Machine learning is a relatively old field with classical methods and algorithms since the 1960s. This field of study considers a subfield of Artificial Intelligence. It's defined as the field of study that gives computers the ability to learn without being explicitly programmed and to learn from experience.

Machine learning classical algorithms include Naïve Bayes Classifier, Support Vector Machines, and more. These algorithms must be trained on large amounts of data.

Machine Learning models can be divided into the following four categories depending on the amount and type of supervision they need while training.

- **Supervised learning:** The training data is already supplied with predefined labels or outputs. Regression and Classification (Figure 2) are the most common problems that are solved by supervised learning.
- **Unsupervised learning:** Where the training dataset is unlabelled, where the model will learn the hidden patterns from the dataset by itself without any supervision. The techniques used in unsupervised learning are:

  Clustering, Association Rule Learning, and Dimensionality Reduction.
- **Semi-supervised learning** is the process of training the model with both labelled and unlabelled data. And this useful method used when extracting features is difficult from the data. (Salian, 2018)
- **Reinforcement learning:** In this model, the agent is developed to interact in a specific environment so that its performance for executing certain tasks improves from the interactions. In other words, the agent decides what to do to perform the given task, where data has no outputs.

Every machine learning algorithm learns the mapping from an input to an output. It is meant that the algorithm learns a function with a few sets of weights (Pai, 2020):

Input → f (W_1, W_2, W_3 …. W n) → Output ()

An algorithm also can learn the function that separates two classes. (Classification)

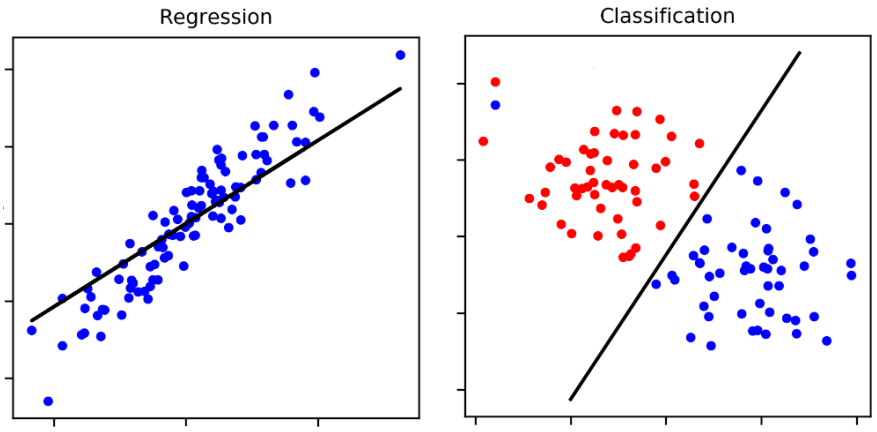


Figure 2. Supervised Learning Types (Agnihotri, n.d.)

2.2 Deep Learning

Unlike machine learning, deep learning is a young subfield of Artificial intelligence and a subfield of machine learning, and it is based on artificial neural networks.

Understanding the need for deep learning comes through these points:

- Feature Engineering: It's the key step in model building, where the required feature extracted for a problem from raw data and then select the important ones that improve the model performance (Figure 3). This process can be automated by using deep learning. (Pai, 2020)
- Decision boundary: Unlike machine learning, where algorithms learn decision boundaries for linear data, deep learning algorithms can learn decision boundaries for non-linear data (Figure 4).

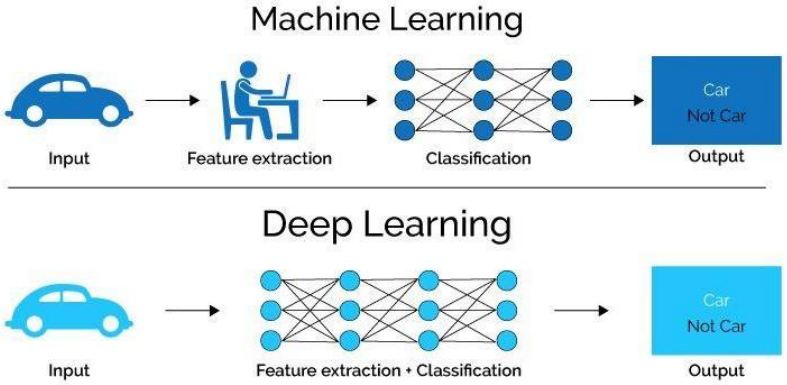


Figure 3. Feature Engineering Comparison (Pai, 2020)

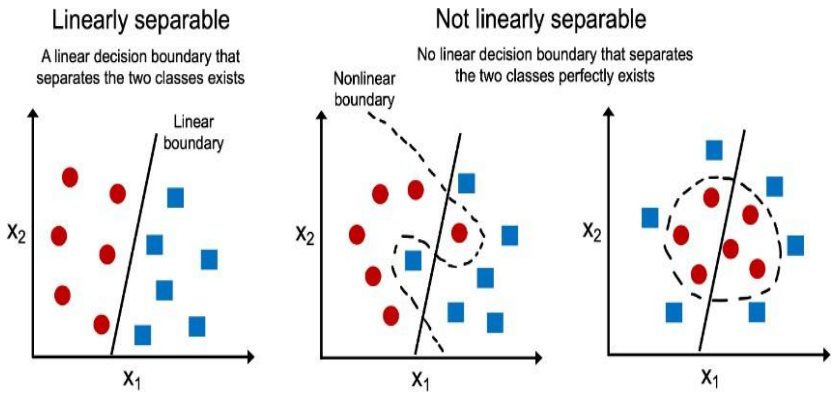


Figure 4. linear vs non-linear data (Kumar, 2022)

Most deep learning neural networks are Feed-Forward Neural Networks. It means that data only flow from input to output (Amey Thakur, 2021) such as ANN and CNN (Figure 5).

Artificial Neural Network (ANN) consists of three main types of layers (input, output, and hidden). Each layer consists of neurons (nodes), which are connected to neurons of the next layer. Each connection is associated with a numeric number called “weight” (Wang, 2003).

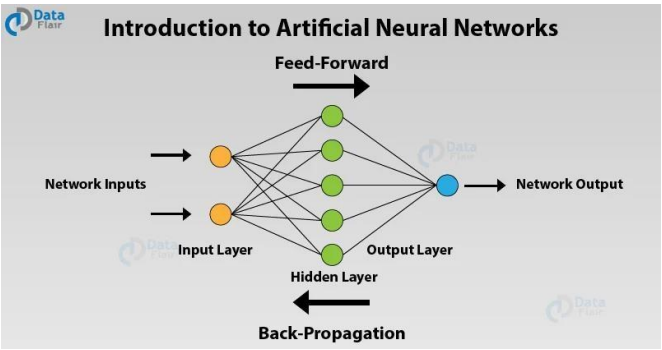


Figure 5. Artificial Neural Networks (Artificial Neural Networks for Machine Learning – Every aspect you need to know about, 2022)

Each neuron (node) in the ANN has some weight assigned to it and a transfer function is used to calculate the Weight sum of the inputs and the bias (Figure 6). After the transfer function calculates the sum, the activation function obtains the result until it received the output so it can then fire the appropriate result from the node. (Artificial Neural Networks for Machine Learning – Every aspect you need to know about, 2022)

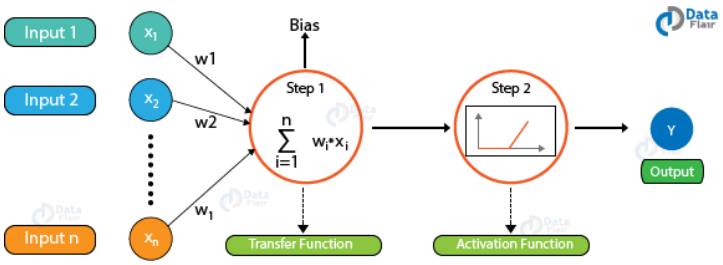


Figure 6. ANN Architecture (Artificial Neural Networks for Machine Learning – Every aspect you need to know about, 2022)

Convolutional Neural Network (CNN) was mainly created for processing data that has a grid pattern such as image and video recognition, recommendation systems, and image analysis and classification (Figure 7). These networks are composed of three types of layers (convolutional, Pooling, and fully connected layer). The first two layers (convolutional, pooling) perform feature extraction, whereas the third layer (fully connected) maps the extracted features into the final output. (Rikiya Yamashita, 2018)

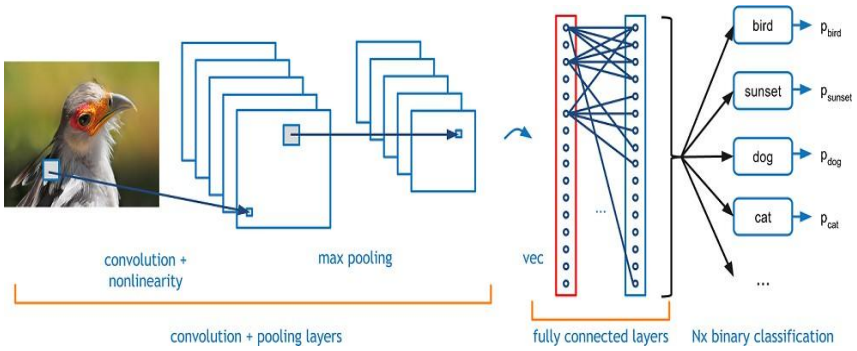


Figure 7. Convolutional Neural Network (Deshpande, 2016)

2.2.1 Recurrent Neural Networks (RNN)

Recurrent Neural Network is a class of Artificial neural networks where the connections between nodes can create a cycle, where the output from some nodes affects subsequent input to the same nodes. The RNN is different from other neural networks since it has a loop that allows information to be passed from one step of the network to the next (Figure 8).

One of the challenges with Recurrent Neural Networks is vanishing an exploding gradient problem which is a common problem in all types of neural networks. Recurrent Neural Networks have also a cyclic connection making them powerful for modelling sequences. capable of learning order dependence in sequence problems such as time series forecasting. (Sak, 2014)

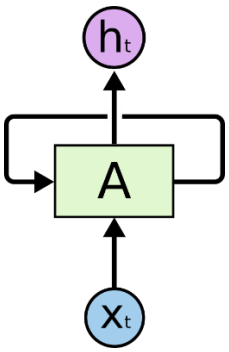


Figure 8. RNN loops (Olah, 2015)

2.2.2 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a special kind and powerful Recurrent Neural Network approach, which has capable of learning long-term dependencies. The LSTM model has achieved the best results for many problems on sequential data by remembering information for long periods (Figure 9).

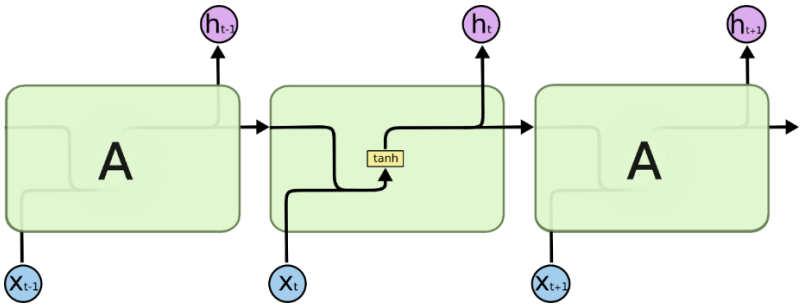


Figure 9. The repeating module in a standard RNN (Olah, 2015)

The main idea behind LSTM cells is to learn the important parts of the sequence and forget the less important ones by the so-called gates. An LSTM module has a cell state and three gates (input, output, forget) which provides them with the power to selectively learn, unlearn or retain information from each of the units. The cell state helps the information flow through the units without being altered by allowing only a few linear interactions. Each unit has an input, output, and forget gate which can add or remove the information to the cell state (Figure 10).

The forget gate decides which information from the previous cell state should be forgotten for which it uses a sigmoid function. The input gate controls the information flow to the current cell state using a point-wise multiplication operation of ‘sigmoid’ and ‘tanh’ respectively (Figure 10). Finally, the output gate decides which information should be passed on to the next hidden state (Time Series - LSTM Model, 2022). One of the purposes to de-

sign LSTMs was to address the vanishing and exploding gradient problems of conventional RNNs.

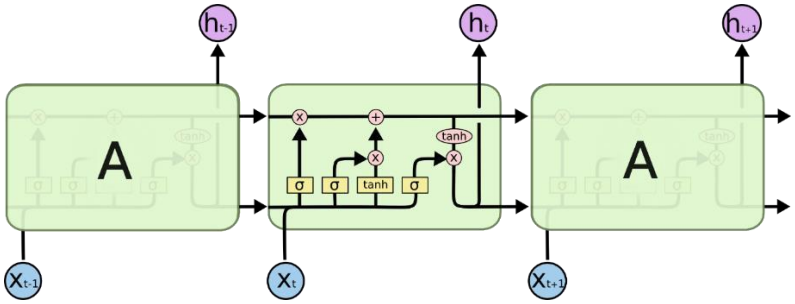


Figure 10. The repeating module in an LSTM (Olah, 2015)

2.3 Time Series Forecasting

2.3.1 Time Series

Time series data represents a series of data over a specified period. In time series, time is often the independent variable, and the goal is usually to make a forecast for the future.

Time series can be split according to time into:

- Continuous Time Series: Where Observations are measured in continuous time series for each one of time, such as temperature.
- Discrete Time Series: Observations are measured at discrete points in time, such as population change in a city or energy consumption. Discrete-time series observations are usually recorded at equal time intervals (daily, weekly, monthly, yearly).

The time series data can be split according to the dependent variables count into univariate, where a single variable is measured over time, and multivariate, where multiple variables are measured over time.

While time series analysis is all about understanding the dataset through extracting meaningful statistics and other characteristics of the data; forecasting is all about using a model to predict future values based on previously observed values. (Time series forecasting methods, n.d.)

2.3.2 Time Series Characteristics

Time Series has main three characteristics which can be modeled to obtain accurate forecasts.

- **Autocorrelation**: is intended to measure the relationship between a variable's present value and any past values that we may have access to.
- **Seasonality**: It is the presence of variations that occur at specific regular intervals of less than a year, such as weekly, monthly, or quarterly. (Wikipedia, 2022)
- **Stationarity:** When time series statistical properties do not change over time. In other words, it has constant mean and variance, and covariance is independent of time.

2.3.3 Time Series Forecast Methods

Time series forecasting is the use of historical information values and associated patterns to predict future ones. As well all forecasting methods are not guaranteed to succeed, so we use machine learning for this purpose. (What is time series data?, 2022)

Time series forecasting is also an important area of machine learning (ML), and it can be cast as a supervised learning problem. (Time series forecasting methods, n.d.)

Through the years, many studies evaluate the performance of classical and machine learning methods. Some of these classical methods are come to be famous such as:

- Autoregressive Moving Average (ARMA)
- Autoregressive Integrated Moving Average (ARIMA)
- Seasonal Autoregressive Integrated Moving-Average (SARIMA)

Also, machine learning has:

- Multi-Layer Perceptron (MLP)
- Bayesian Neural Networks (BNN)
- Generalized Regression Neural Networks (GRNN)

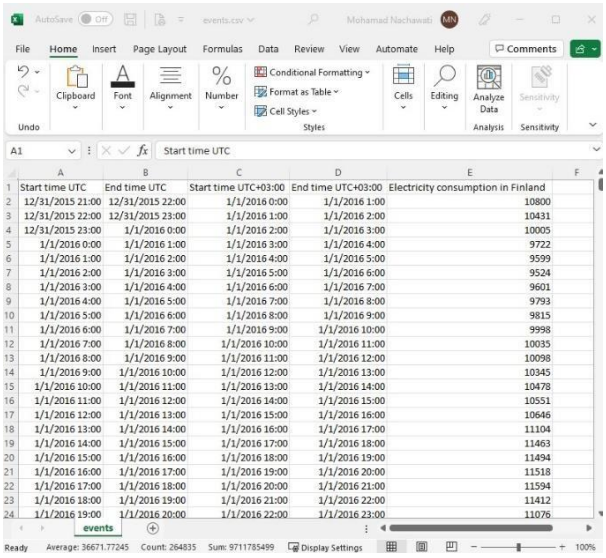
Two other modern algorithms are Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) (Brownlee, 2016).

3 Model Implementation

3.1 Data

This project tries to answer the question if it's possible to create an LSTM model to predict future energy consumption. The data was imported from Finland's transmission system operator as a CSV file and then exported to a GitHub repository (Figure 11). There was a total number of 52965 observations and 5 variables in this dataset and no missing values were found. The minimum load volume is 5341 MWh, and the maximum load volume is 15105 MWh along with an average volume of 9488.750519 MWh. There is no duplicate row in the dataset.

Two columns of data present start and end UTC, while the second two columns present start and end UTC + 3:00 time (Helsinki time). The last column contains electricity consumption in Finland.



	A	B	C	D	E
1	Start time UTC	End time UTC	Start time UTC+03:00	End time UTC+03:00	Electricity consumption in Finland
2	12/31/2015 21:00	12/31/2015 22:00	1/1/2016 0:00	1/1/2016 1:00	10800
3	12/31/2015 22:00	12/31/2015 23:00	1/1/2016 1:00	1/1/2016 2:00	10431
4	12/31/2015 23:00	1/1/2016 0:00	1/1/2016 2:00	1/1/2016 3:00	10005
5	1/1/2016 0:00	1/1/2016 1:00	1/1/2016 3:00	1/1/2016 4:00	9722
6	1/1/2016 1:00	1/1/2016 2:00	1/1/2016 4:00	1/1/2016 5:00	9599
7	1/1/2016 2:00	1/1/2016 3:00	1/1/2016 5:00	1/1/2016 6:00	9524
8	1/1/2016 3:00	1/1/2016 4:00	1/1/2016 6:00	1/1/2016 7:00	9601
9	1/1/2016 4:00	1/1/2016 5:00	1/1/2016 7:00	1/1/2016 8:00	9793
10	1/1/2016 5:00	1/1/2016 6:00	1/1/2016 8:00	1/1/2016 9:00	9815
11	1/1/2016 6:00	1/1/2016 7:00	1/1/2016 9:00	1/1/2016 10:00	9998
12	1/1/2016 7:00	1/1/2016 8:00	1/1/2016 10:00	1/1/2016 11:00	10035
13	1/1/2016 8:00	1/1/2016 9:00	1/1/2016 11:00	1/1/2016 12:00	10098
14	1/1/2016 9:00	1/1/2016 10:00	1/1/2016 12:00	1/1/2016 13:00	10345
15	1/1/2016 10:00	1/1/2016 11:00	1/1/2016 13:00	1/1/2016 14:00	10478
16	1/1/2016 11:00	1/1/2016 12:00	1/1/2016 14:00	1/1/2016 15:00	10551
17	1/1/2016 12:00	1/1/2016 13:00	1/1/2016 15:00	1/1/2016 16:00	10646
18	1/1/2016 13:00	1/1/2016 14:00	1/1/2016 16:00	1/1/2016 17:00	11104
19	1/1/2016 14:00	1/1/2016 15:00	1/1/2016 17:00	1/1/2016 18:00	11463
20	1/1/2016 15:00	1/1/2016 16:00	1/1/2016 18:00	1/1/2016 19:00	11494
21	1/1/2016 16:00	1/1/2016 17:00	1/1/2016 19:00	1/1/2016 20:00	11518
22	1/1/2016 17:00	1/1/2016 18:00	1/1/2016 20:00	1/1/2016 21:00	11594
23	1/1/2016 18:00	1/1/2016 19:00	1/1/2016 21:00	1/1/2016 22:00	11412
24	1/1/2016 19:00	1/1/2016 20:00	1/1/2016 22:00	1/1/2016 23:00	11076

Figure 11. Electricity consumption in Finland,

3.1.1 Data Exploration

Data exploration consider an important process to understand data with statistical and visualization methods. One of the advantages of this process is to identify patterns and problems in this dataset.

Return first 5 rows.

	Start time UTC	End time UTC	Start time UTC+03:00	End time UTC+03:00	Electricity consumption in Finland
0	2015-12-31 21:00:00	2015-12-31 22:00:00	2016-01-01 00:00:00	2016-01-01 01:00:00	10800.0
1	2015-12-31 22:00:00	2015-12-31 23:00:00	2016-01-01 01:00:00	2016-01-01 02:00:00	10431.0
2	2015-12-31 23:00:00	2016-01-01 00:00:00	2016-01-01 02:00:00	2016-01-01 03:00:00	10005.0
3	2016-01-01 00:00:00	2016-01-01 01:00:00	2016-01-01 03:00:00	2016-01-01 04:00:00	9722.0
4	2016-01-01 01:00:00	2016-01-01 02:00:00	2016-01-01 04:00:00	2016-01-01 05:00:00	9599.0

Figure 12. Data first five rows

Return last 5 rows.

	Start time UTC	End time UTC	Start time UTC+03:00	End time UTC+03:00	Electricity consumption in Finland
52961	2021-12-31 16:00:00	2021-12-31 17:00:00	2021-12-31 19:00:00	2021-12-31 20:00:00	11447.0
52962	2021-12-31 17:00:00	2021-12-31 18:00:00	2021-12-31 20:00:00	2021-12-31 21:00:00	11237.0
52963	2021-12-31 18:00:00	2021-12-31 19:00:00	2021-12-31 21:00:00	2021-12-31 22:00:00	10914.0
52964	2021-12-31 19:00:00	2021-12-31 20:00:00	2021-12-31 22:00:00	2021-12-31 23:00:00	10599.0
52965	2021-12-31 20:00:00	2021-12-31 21:00:00	2021-12-31 23:00:00	2022-01-01 00:00:00	10812.0

Figure 13. Data last five rows

It seems by viewing the data frame that some columns are not needed where it presents different times records. The most important is to use the last two columns where actual time with consumption is presented (Figure 12) and (Figure 13).

Outliers that have large or small values compared to most observations have a large impact on the model results. After exploring the data, the max and min values looked like reasonable values compared to the mean values, so outliers were not found in the data (Figure 14).

	Electricity consumption in Finland
count	52966.000000
mean	9488.750519
std	1576.241673
min	5341.000000
25%	8322.000000
50%	9277.000000
75%	10602.000000
max	15105.000000

Figure 14. Descriptive statistics

The data which contain null values will make the analysis more complicated and affect the result of the model. Null values were not found in this data (Figure 15) and the total Number of years in this dataset is 6 years between 2016 - 2021.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52966 entries, 0 to 52965
Data columns (total 5 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   Start time UTC                     52966 non-null  object
 1   End time UTC                       52966 non-null  object
 2   Start time UTC+03:00               52966 non-null  object
 3   End time UTC+03:00                 52966 non-null  object
 4   Electricity consumption in Finland 52966 non-null  float64
dtypes: float64(1), object(4)
memory usage: 2.0+ MB
```

Figure 15. Data Information

3.1.2 Data Pre-processing

Assuming the week starts on Monday and ends on Sunday. The closest Monday in the data set was on Monday 4/12/2016 and the closest end was on Sunday 26/2/2021. In this case, omitting was done to the first 71 rows and the last 121 ones (Figure 16).

The data is univariate time series, where there is a need for one column to present time and another one to present energy consumption. In addition, to create a new data frame containing just the two columns, there we renamed them (DateTime, and Consumption) beside extracting features from date objects such as Month, Year, Date, Time, Week, and Day, where it makes easy to analyze and visualize the data.

DateTime	Consumption	Month	Year	Date	Time	Week	Day
2016-01-01 01:00:00	10800.0	1	2016	2016-01-01	01:00:00	53	Friday
2016-01-01 02:00:00	10431.0	1	2016	2016-01-01	02:00:00	53	Friday
2016-01-01 03:00:00	10005.0	1	2016	2016-01-01	03:00:00	53	Friday
2016-01-01 04:00:00	9722.0	1	2016	2016-01-01	04:00:00	53	Friday
2016-01-01 05:00:00	9599.0	1	2016	2016-01-01	05:00:00	53	Friday

Figure 16. DateTime indexing and extracting features

3.1.3 Data Visualization

By looking for consumption according to years (Figure 17), it's seemed that there is seasonality in the data set, where the consumption increases in winter and decreases in summer. Also, in 2020 there is a decrease in energy consumption, and that could be related to the coronavirus epidemic.

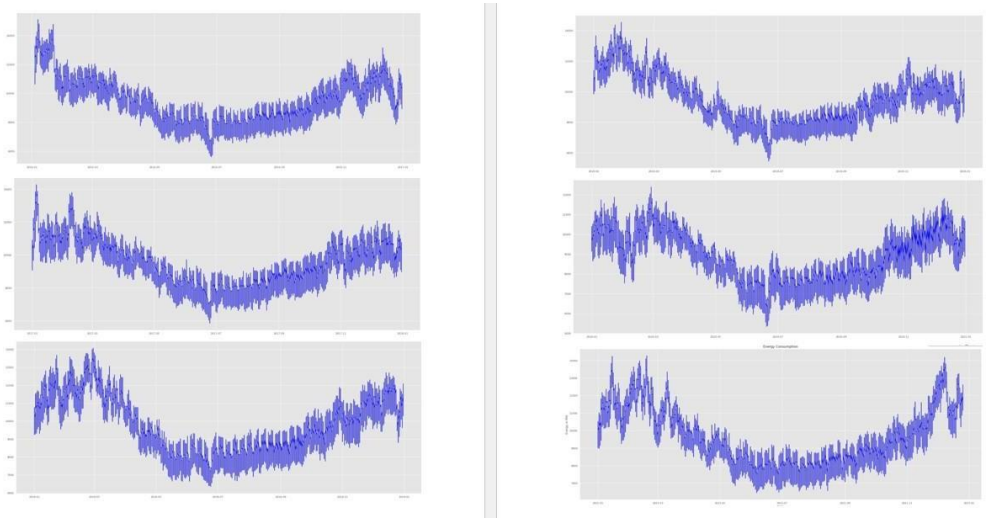


Figure 17. Energy consumption by the year

The distribution plot shows that the data has unimodal with right skew distribution and the observations are concentrated beside no outliers were found (Figure 18).

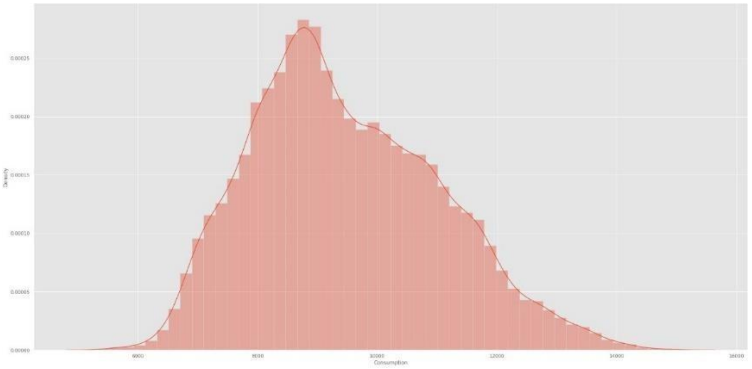


Figure 18. Distribution of energy consumption

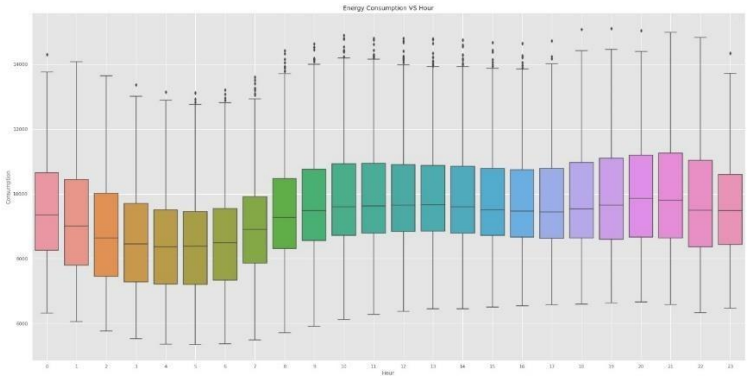


Figure 19. Energy consumption VS Hour

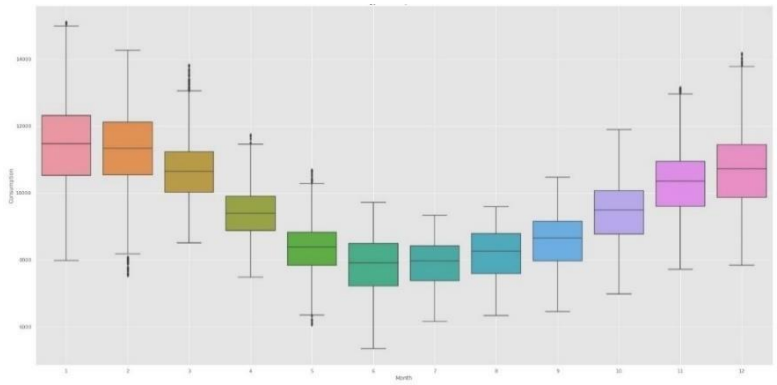


Figure 20. Energy consumption VS Month

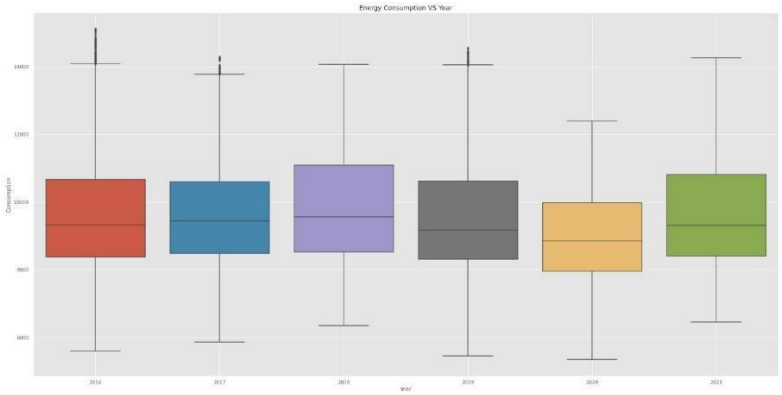


Figure 21. Energy Consumption VS Year

3.2 LSTM Model

3.2.1 Train, Validation, and Test Dataset

For predicting day consumption, data were down-sampled using resample function. This function changed the data from hourly frequency to daily frequency. This down-sampling was done by using the mean () method. The data rows were reduced from 52774 to 2184 rows (Figure 22).

DateTime	Consumption	Month	Year	Week
2016-01-04	12300.625000	1.0	2016.0	1.0
2016-01-05	12945.375000	1.0	2016.0	1.0
2016-01-06	13192.750000	1.0	2016.0	1.0
2016-01-07	14243.541667	1.0	2016.0	1.0
2016-01-08	14121.666667	1.0	2016.0	1.0

Figure 22. Daily Consumption Data

The LSTM model is sensitive to the scale of the input data, so it can be a good practice to rescale the data to the range of 0-to-1. This is called data normalization. In this case, the data was normalized using the MinMaxScaler function. MinMaxScaler scaled the output and the input in the range between 0 - 1 to match the scale of the LSTM layer. MinMaxScaler will subtract the minimum value of Consumption and then divide it by the range.

To train the model we split the data into a training set (80%) and a testing set (20%). Also, we split training data into 20% for the validation set and 80% for training. The validation set is used to validate the model performance during training.

The input layer of the LSTM model requires 3D input, Where the first dimension represents the sample size, and the second represents the number of time steps. And the third dimension represents the number of features used to train the model [number of samples, time steps, and number of features]. The dataset was reshaped to look like a 3-dimensional array, where our time steps were 100 and our feature is the Consumption (Figure 23).

```
X_train shape:  (1297, 100, 1)
X_test shape:  (336, 100, 1)
X_val shape:  (248, 100, 1)
```

Figure 22. Reshape Inputs to fit LSTM layers

3.2.2 Model Structure

Stacked LSTM will be built in this project, which contains four hidden LSTM layers one on top of another. The model building starts by creating a sequential model using Keras. Then we added the hidden LSTM layers to the model. Each layer had 50 units. A dropout layer was added to help in preventing overfitting. The last layer was the dense layer, which does the below operation on the input and returns the output (Figure 24).

To update network weights iterative based on training data we used the Adam optimization algorithm which was used famously instead of the classical gradient descent procedure. Also, we used root mean squared error (RMSE) to test the model performance. A smaller RMSE means that our model is performing better.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_4 (LSTM)               (None, 100, 50)           10400

 dropout (Dropout)           (None, 100, 50)           0

 lstm_5 (LSTM)               (None, 100, 50)           20200

 lstm_6 (LSTM)               (None, 100, 50)           20200

 lstm_7 (LSTM)               (None, 50)                20200

 dense_1 (Dense)             (None, 1)                 51

=================================================================
Total params: 71,051
Trainable params: 71,051
Non-trainable params: 0
_________________________________________________________________
```

Figure 24. LSTM Model Summary

3.2.3 Model Training

Training the LSTM model was done using the training set and the validation dataset for testing the results through the training process. The learning algorithm worked through the entire training dataset 60 times (Epoch), and the model weights were updated after each batch where the batch size is 20 (Figure 24).

```
Epoch 43/60
65/65 [==============================] - 12s 182ms/step - loss: 0.0028 - val_loss: 0.0016
Epoch 44/60
65/65 [==============================] - 12s 182ms/step - loss: 0.0024 - val_loss: 0.0016
Epoch 45/60
65/65 [==============================] - 12s 181ms/step - loss: 0.0022 - val_loss: 0.0017
Epoch 46/60
65/65 [==============================] - 13s 198ms/step - loss: 0.0023 - val_loss: 0.0015
Epoch 47/60
65/65 [==============================] - 12s 181ms/step - loss: 0.0022 - val_loss: 0.0017
Epoch 48/60
65/65 [==============================] - 13s 195ms/step - loss: 0.0023 - val_loss: 0.0020
Epoch 49/60
65/65 [==============================] - 12s 181ms/step - loss: 0.0018 - val_loss: 0.0015
Epoch 50/60
65/65 [==============================] - 13s 197ms/step - loss: 0.0017 - val_loss: 0.0016
Epoch 51/60
65/65 [==============================] - 12s 185ms/step - loss: 0.0020 - val_loss: 0.0012
Epoch 52/60
65/65 [==============================] - 12s 183ms/step - loss: 0.0017 - val_loss: 0.0011
Epoch 53/60
65/65 [==============================] - 12s 184ms/step - loss: 0.0015 - val_loss: 0.0010
Epoch 54/60
65/65 [==============================] - 13s 198ms/step - loss: 0.0016 - val_loss: 0.0013
Epoch 55/60
65/65 [==============================] - 12s 182ms/step - loss: 0.0015 - val_loss: 0.0011
Epoch 56/60
65/65 [==============================] - 12s 183ms/step - loss: 0.0015 - val_loss: 0.0010
Epoch 57/60
65/65 [==============================] - 13s 195ms/step - loss: 0.0016 - val_loss: 0.0010
Epoch 58/60
65/65 [==============================] - 12s 182ms/step - loss: 0.0014 - val_loss: 0.0012
Epoch 59/60
65/65 [==============================] - 12s 183ms/step - loss: 0.0015 - val_loss: 0.0013
Epoch 60/60
65/65 [==============================] - 12s 185ms/step - loss: 0.0016 - val_loss: 0.0014
```

Figure 25. Training LSTM Model

The check after training was to compare the training loss against the validation loss. The result shows that the two values were low, and no overfitting was detected.

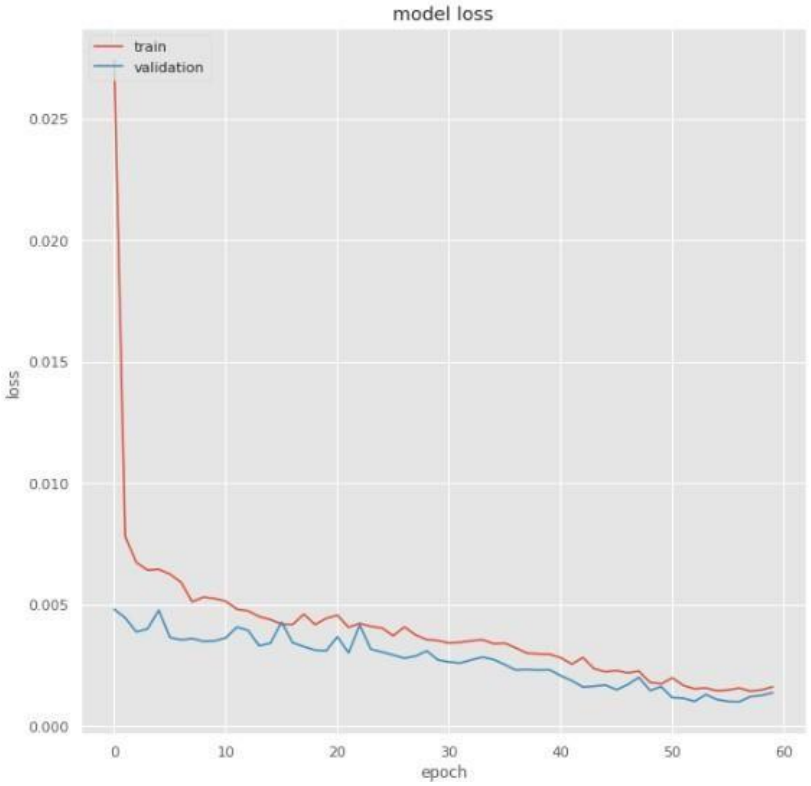


Figure 26. Comparison Curve of Training and validation Accuracy of 60 epochs

Examples of consumption predictions of the LSTM model were done using the training dataset (Figure 27), validation dataset (Figure 28), and test datasets (Figure 29).

41/41 [==============================] - 2s 47ms/step

	Train Predictions	Actuals
0	[9533.0693359375]	[9406.708333333334]
1	[9557.21875]	[9614.791666666666]
2	[9699.4619140625]	[9894.708333333334]
3	[9583.296875]	[8933.708333333334]
4	[8622.5341796875]	[8557.208333333334]
...	...	...
1292	[8619.515625]	[9259.666666666666]
1293	[9671.5712890625]	[10248.5]
1294	[10395.3466796875]	[10360.333333333334]
1295	[10408.70703125]	[10489.833333333334]
1296	[10580.2080078125]	[10204.5]

1297 rows × 2 columns

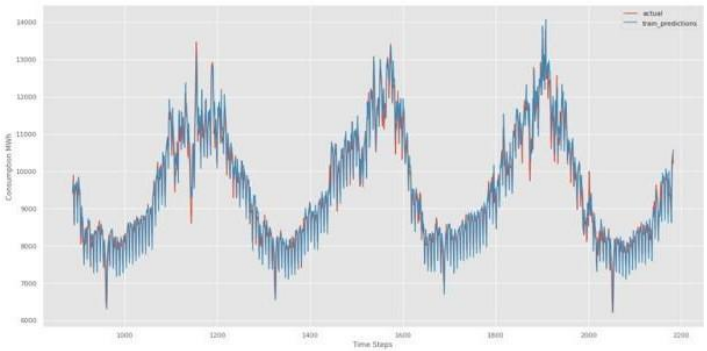


Figure 27. Predict consumption using training data

```
8/8 [==============================] - 1s 70ms/step
```

	Val Predictions	Actuals_val
0	[9564.7529296875]	[8931.875]
1	[9500.849609375]	[9112.916666666666]
2	[9322.599609375]	[10019.416666666666]
3	[9908.9287109375]	[10390.375]
4	[10409.23046875]	[10802.916666666666]
...	...	...
243	[8295.7900390625]	[8020.541666666667]
244	[7598.3251953125]	[7921.75]
245	[8153.66748046875]	[8705.041666666666]
246	[8841.4716796875]	[8824.708333333334]
247	[8815.4609375]	[9062.375]

248 rows × 2 columns

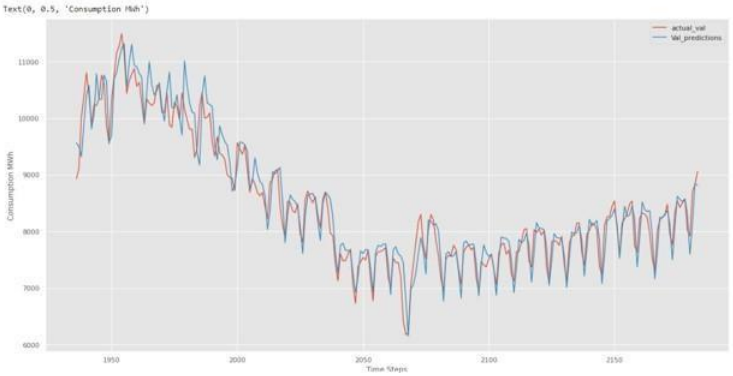


Figure 28. Predict consumption using validation data

```
11/11 [==============================] - 1s 71ms/step
```

	test Predictions	Actuals_test
0	[10396.9130859375]	[10032.541666666666]
1	[11196.3955078125]	[10732.125]
2	[11439.9912109375]	[10733.583333333334]
3	[10969.5]	[10971.875]
4	[11177.0654296875]	[11227.791666666666]
...	...	...
331	[12638.69921875]	[12540.25]
332	[12564.671875]	[12635.958333333334]
333	[12774.3154296875]	[11684.333333333334]
334	[11735.4423828125]	[11384.166666666666]
335	[11262.974609375]	[11581.625]

336 rows × 2 columns

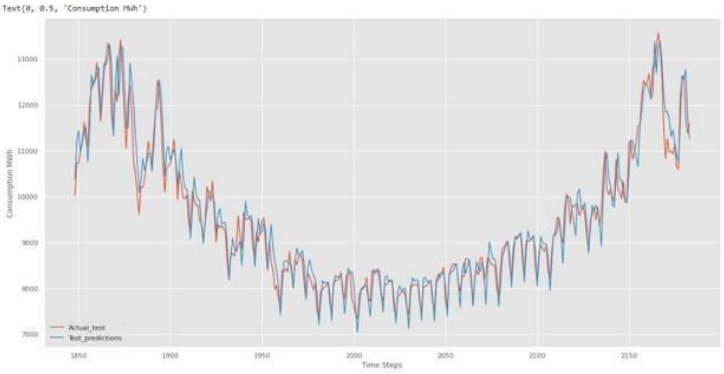


Figure 29. Predict consumption using test data

4 Conclusion

This project aimed to predict energy consumption in Finland using an LSTM model, which was trained on a dataset containing 6 years of electricity consumption in Finland.

The dataset was transformed from hourly time series into daily ones. It was split into samples, where each sample contained 100 time steps (day). The LSTM model was trained on the first 100 observations, and it predict observation 101 and so on.

The model was built successfully using four LSTM layers, one Dropout, and one output layer. This project used the Adam optimization algorithm and root mean squared error (RMSE) to test the model performance.

The result has shown good results when applying the stacked LSTM for short-term predictions. The model was tested first on training, validation, and test datasets. It is worth mentioning that this evaluation highly depends on the time steps that in this case was 100 days. With a different number, of considered timesteps, the results would differ.

References

Agnihotri, N. (n.d.). *Classification of machine learning algorithms*. Retrieved from engineersgarage.com: https://www.engineersgarage.com/machine-learning-algorithms-classification/

Amey Thakur, A. K. (2021, August). *Fundamentals of Neural Networks.* Retrieved from researchgate.net: https://www.researchgate.net/publication/353827517_Fundamentals_of_Neural_Networks

Artificial Neural Networks for Machine Learning – Every aspect you need to know about. (2022, 10 16). Retrieved from data-flair.training: https://data-flair.training/blogs/artificial-neural-networks-for-machine-learning/

Brownlee, J. (2016, August 6). *11 Classical Time Series Forecasting Methods in Python (Cheat Sheet)*. Retrieved from machinelearningmastery.com: https://machinelearningmastery.com/time-series-forecasting-methods-in-python-cheat-sheet/

Deshpande, A. (2016, July 20). *A Beginner's Guide To Understanding Convolutional Neural Networks*. Retrieved from adeshpande3.github.io: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Dogaru, L. (2020). *The Main Goals of the Fourth Industrial Revolution. Renewable Energy Prespectives.* Retrieved from ScienceDirect: https://www.sciencedirect.com/science/article/pii/S2351978920309367

Electricity sector in Finland. (2022, September 23). Retrieved from wikipedia.org: https://en.wikipedia.org/wiki/Electricity_sector_in_Finland

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* O'Reilly Media. Retrieved from https://books.google.fi/books?id=HnetDwAAQBAJ&lpg=PT9&ots=kPWvzExGx2&dq=Hands-on%20ML%20with%20Scikit-Learn%2C%20Keras%20and%20TensorFlow&lr&hl=ar&pg=PP1#v=onepage&q=Hands-on%20ML%20with%20Scikit-Learn,%20Keras%20and%20TensorFlow&f=false

Kumar, A. (2022, July 31). *Linear vs Non-linear Data: How to Know*. Retrieved from vitalflux.com: https://vitalflux.com/how-know-data-linear-non-linear/

Olah, C. (2015, August 27). *Understanding LSTM Networks*. Retrieved from colah.github.io: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Pai, A. (2020, 04 17). *CNN vs. RNN vs. ANN – Analyzing 3 Types of Neural Networks in Deep Learning*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/

Pecht, M. G. (2018). *Prognostics and Health Management of Electronics: Fundamentals, Machine Learning, and the Internet of Things.* USA: John Wiley and Sons Ltd. Retrieved from

https://lut.primo.exlibrisgroup.com/permalink/358FIN_LUT/1hujjmv/cdi_askewsholts_vlebooks_9781119515357

Peixeiro, M. (2019, August 7). *The Complete Guide to Time Series Analysis and Forecasting*. Retrieved from towardsdatascience.com: https://towardsdatascience.com/the-complete-guide-to-time-series-analysis-and-forecasting-70d476bfe775

Pickell, D. (2021, May 24). *Supervised vs Unsupervised Learning – What's the Difference?* Retrieved from www.g2.com: https://www.g2.com/articles/supervised-vs-unsupervised-learning

Rehurek, R. (2014, December). *Data Science with Python*. Retrieved from rasdimrehurek.com: https://radimrehurek.com/data_science_python/

Rikiya Yamashita, M. N. (2018, June 22). *Convolutional neural networks: an overview and application in radiology*. doi:https://doi.org/10.1007/s13244-018-0639-9

Sak, H. (2014, Feb 5). *Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition*. Retrieved from arxiv.org: https://arxiv.org/abs/1402.1128

Salian, I. (2018, August 2). *SuperVize Me: What's the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning?* Retrieved from nvidia.com: https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/

Statistics, F. (2021, 11 2). *Statistics on production of electricity and heat*. Retrieved from stat.fi: https://www.stat.fi/til/salatuo/2020/salatuo_2020_2021-11-02_tie_001_en.html

Time Series - LSTM Model. (2022, 10 16). Retrieved from tutorialspoint.com: https://www.tutorialspoint.com/time_series/time_series_lstm_model.htm

Time series forecasting methods. (n.d.). Retrieved from influxdata: https://www.influxdata.com/time-series-forecasting-methods/

Wang, S.-C. (2003). Artificial Neural Network. In S.-C. Wang, *Interdisciplinary Computing in Java Programming* (pp. 81-82). Boston: Springer Nature. doi:https://doi.org/10.1007/978-1-4615-0377-4_5

What is time series data? (2022, 10 16). Retrieved from influxdata: https://www.influxdata.com/what-is-time-series-data/

Wikipedia. (2022, 10 12). *Seasonality*. Retrieved from wikipedia.org: https://en.wikipedia.org/wiki/Seasonality

Python Code:

https://colab.research.google.com/drive/1CBaMYiT6BMl5CM_Jtj4vIxGJqWHqweQM?usp=sharing