Forecasting Electricity Demand based on Weather Data

Claude Shyaka
ID: 801326243
cshyaka@uncc.edu
GitHub repo: https://github.com/claudeshyaka/ml-final-project

Abstract

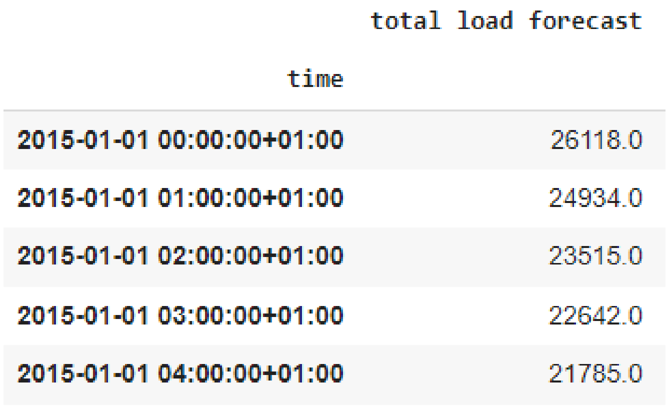
In modern economies, electricity is used in most aspects of everyday life, from lighting homes to making and driving Teslas. However, in some cases, unforeseen events such as extreme weather can significantly impact the supply of this vital product. Therefore, developing machine learning models that could help predict the impact of such events on the electrical grid could revolutionize the way electricity is produced and distributed. In this project, machine learning and neural network models were developed to predict electric loads based on weather patterns.

1. Introduction

Machine learning and artificial neural networks have enhanced our abilities to derive insightful information from data. In this report, weather patterns data from various cities in Spain are explored and used to develop supervised models to predict electric loads for that particular region. That is a linear regression model, a support vector regression (SVR) model, two fully connected neural networks, and Long-short term memory (LSTM) networks were developed and trained with the weather patterns data (see Pytorch on LSTM documentation for more details)[2]. Specifically, the linear regression model was trained with an L2 regularization parameter, the support vector regression model was trained with a radial basis function (RBF) kernel, and the fully connected neural network models had one and five hidden layers, respectively with a Tanh activation function, and were trained for 500 and 1000 epochs, respectively.
Results from these models showed that more sophisticated models would be required to gain more insight from the data, therefore, LSTM models were developed and trained for 10 epochs. Also, note that the model evaluation metric throughout this project is the mean squared error. The upcoming sections discuss the data preprocessing, model parameter tuning as well as analysis of results for the different models described above.

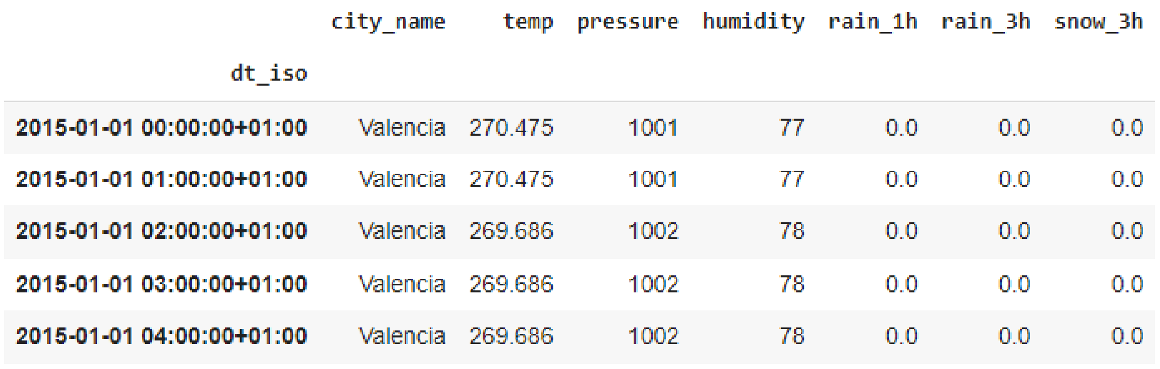
2. Data

Data used in this project was obtained from Kaggle[4]. As mentioned in the introduction, the weather patterns data used in this project is for five different cities in Spain (Madrid, Barcelona, Bilbao, Valencia, and Seville), and the electric load data is for all five cities combined. In addition, the weather patterns dataset contained the following 16 columns and 178396 instances and the electric load dataset contained 35064 instances and 28 columns. Please see the data-processing Jupyter notebook in the Github repo for more details[1]. After a preview of the datasets, the following filter was used for the weather patterns dataset: time, city_name, temperature, pressure, humidity, rain_1h, rainh_3h, and swon_3h; and for the electric load dataset, the following filter was used: time, total load actual. Figure 1(a) shows a preview of the filtered electric load dataset and figure 1(b) shows a preview of the filtered weather patterns dataset.



time	total load forecast
2015-01-01 00:00:00+01:00	26118.0
2015-01-01 01:00:00+01:00	24934.0
2015-01-01 02:00:00+01:00	23515.0
2015-01-01 03:00:00+01:00	22642.0
2015-01-01 04:00:00+01:00	21785.0

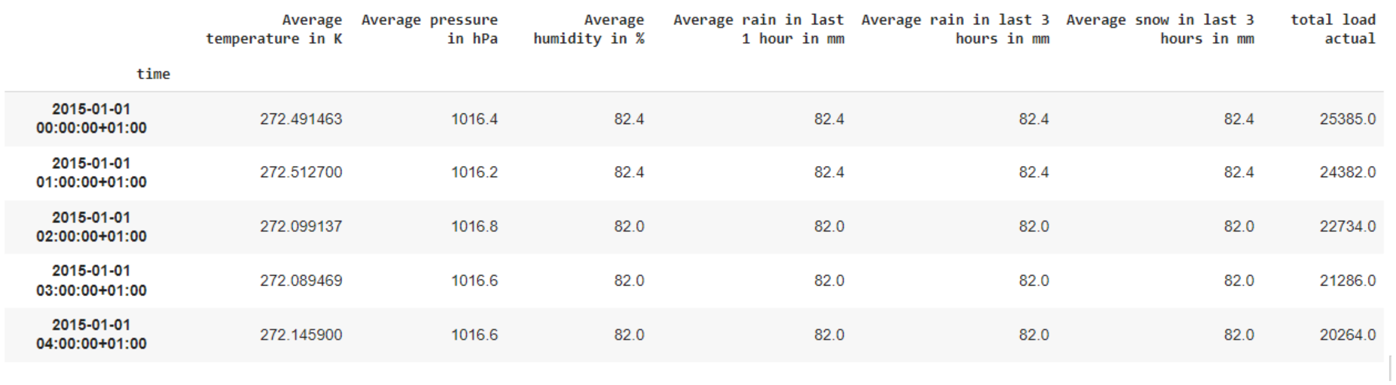
Figure 1 (a): Preview of electric load dataset



dt_iso	city_name	temp	pressure	humidity	rain_1h	rain_3h	snow_3h
2015-01-01 00:00:00+01:00	Valencia	270.475	1001	77	0.0	0.0	0.0
2015-01-01 01:00:00+01:00	Valencia	270.475	1001	77	0.0	0.0	0.0
2015-01-01 02:00:00+01:00	Valencia	269.686	1002	78	0.0	0.0	0.0
2015-01-01 03:00:00+01:00	Valencia	269.686	1002	78	0.0	0.0	0.0
2015-01-01 04:00:00+01:00	Valencia	269.686	1002	78	0.0	0.0	0.0

Figure 1 (b): Preview of weather forecast dataset

To obtain a data format that was suitable for training the models, a series of data transformation steps were applied to each dataset. First, data points in the weather pattern dataset were records of each city and for every hour from 01-01-2015 to 12-31-2018, therefore, the data was first split into records for each individual city and then aggregated to find the mean measurements for all five cities. In addition, missing values were filled in with linear interpolation. Next, data points in the electric load dataset contained some missing values, thus, the linear interpolation method was used to fill in these missing values. Finally, the two datasets were intersected into a single dataset based on their timestamp columns. Figure 2 shows a preview of the data used in training and validation. This dataset contains 38568 sample points of which 30855 sample points (80%) were used for training and 7713 sample points (20%) were used for validation.



time	Average temperature in K	Average pressure in hPa	Average humidity in %	Average rain in last 1 hour in mm	Average rain in last 3 hours in mm	Average snow in last 3 hours in mm	total load actual
2015-01-01 00:00:00+01:00	272.491463	1016.4	82.4	82.4	82.4	82.4	25385.0
2015-01-01 01:00:00+01:00	272.512700	1016.2	82.4	82.4	82.4	82.4	24382.0
2015-01-01 02:00:00+01:00	272.099137	1016.8	82.0	82.0	82.0	82.0	22734.0
2015-01-01 03:00:00+01:00	272.089469	1016.6	82.0	82.0	82.0	82.0	21286.0
2015-01-01 04:00:00+01:00	272.145900	1016.6	82.0	82.0	82.0	82.0	20264.0

Moving on to the models and parameter tuning. First, a linear regression model with an L2 regularization parameter was trained. This is the ridge regression model from the Sckit-learn machine learning library. Using grid search cross-validation tools from Sckit-learn, the model's best value for the alpha hyper-parameter was obtained as 1.0. Second, a support vector regression model with an RBF kernel was trained and using parameter tuning, the best values for gamma, and C hyper-parameters were reported as 10 and 1000, respectively. Third, two fully connected neural networks were developed and trained for 500 and 1000 epochs, respectively. The first neural network was tuned with 3 hidden layers and 280 nodes and trained with a scholastic gradient descent (SGD) optimizer, and the second neural network was configured with 5 hidden layers and 681 nodes, and also trained with an SDG optimizer. Finally, an LSTM network was developed and trained for 10 epochs. First, the LSTM network was configured with 1 layer and 64 hidden units and trained with an Adam optimizer. And finally, the network was re-configured with 3 layers, 64 hidden units, and retrained with the Adam optimizer.

3. Results and Analysis

First, the linear regression model reported a mean squared error of 0.0356 and the training time was 0.409 seconds. Now, these results suggest a fair performance of the linear model, however, keep in mind that this model did not take into account the time component of the data, which leads to conclude that these results, though they might be promising, do not capture the full aspect of the data. Second, the support vector regression (SVR) reported a mean squared error of 0.0340 and completed training in 16397.253 seconds which converts to approximately 4.5 hours. Similarly, the SVR model presents promising results, however, they are not representative of the complete dataset. Next, the first fully connected neural network reported a training loss of 0.8969 and a validation loss of 0.8176 after 500 epochs, and the second neural network reported a training loss of 0.8919 and a validation loss of 0.8158 after 1000 epochs. Both networks had suboptimal results compared to the SVR and linear regression models. Figure 3(a) shows the loss curves of the first fully connected neural network and figure 3(b) shows the loss curve of the second fully connected neural network.

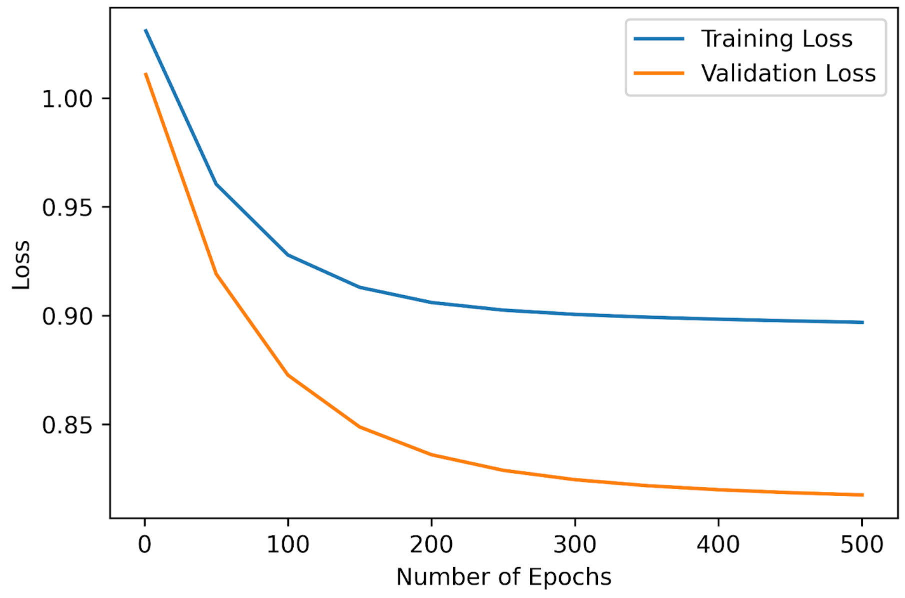


Figure 3(a): Loss curve of the fully connected network with 3 hidden layers and 280 nodes.

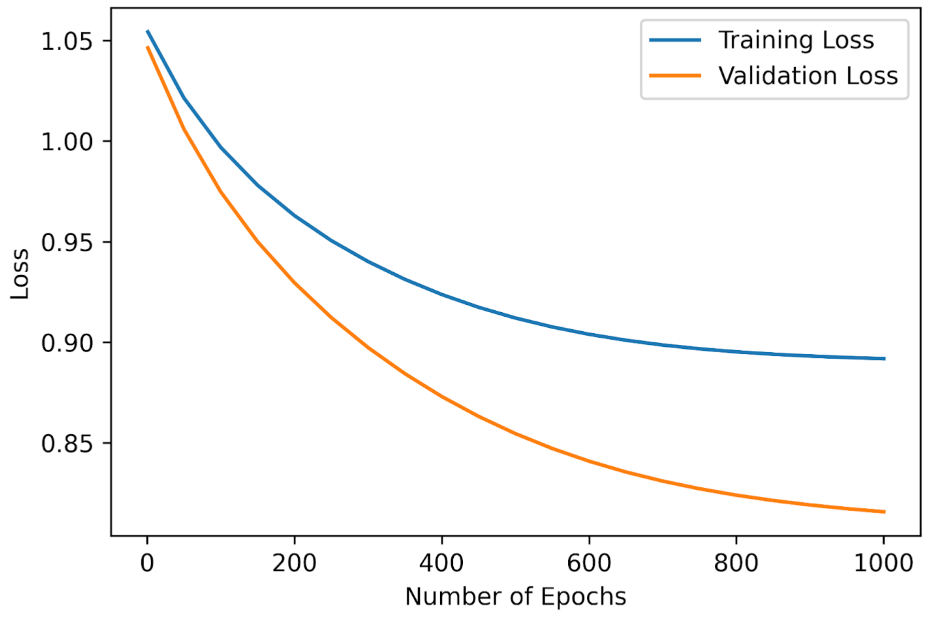


Figure 3(b): Loss curve of the fully connected network with 5 hidden layers and 681 nodes.

Finally, the LSTM network configured with 1 layer, 64 hidden units, and an Adam optimizer reported a training loss of 0.4639 and a validation loss of 0.5457 after 10 epochs, and the LSTM network configured with a 3 layer, 64 hidden units, and the Adam optimizer reported a training loss of 0.4636 and a validation loss of 0.5339. Figure 4 (a) shows a plot of the predicted and actual electric load values over time for the LSTM trained with the Adam optimizer and 1 layer, and Figure 4 (b) shows a plot of the LSTM trained with the Adam optimizer and 3 layers, in addition, for both figures the black dotted line indicates the beginning of the validation data points.

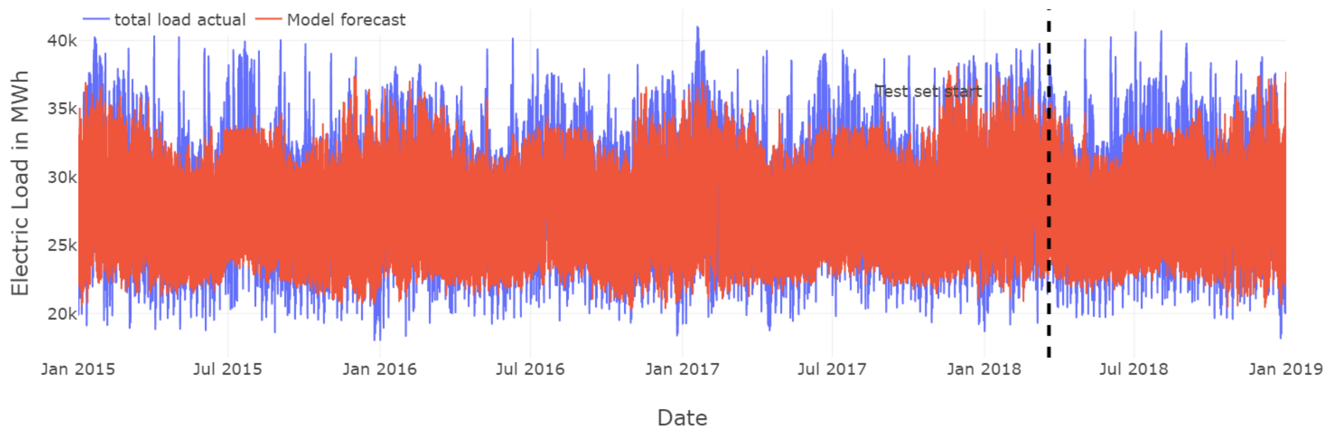


Figure 4(a): Predicted and actual electric load values over time for the LSTM trained with the Adam optimizer and 1 layer.

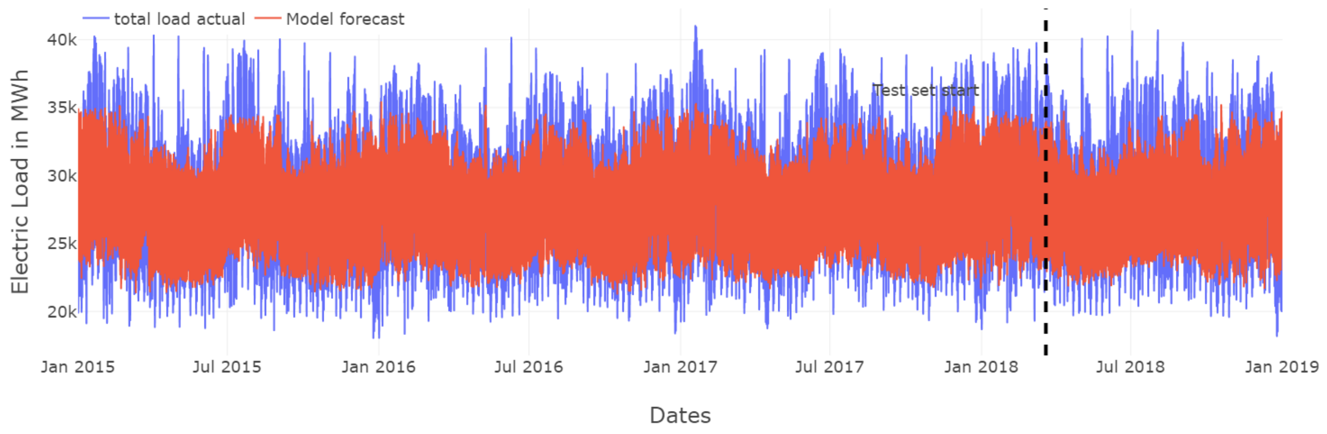


Figure 4(a): Predicted and actual electric load values over time for the LSTM with 3 layers.

4. Conclusion

The goal of this project was to implement a machine learning model that could predict electric loads based on weather forecasted conditions. To accomplish this task, multiple models were developed and contrasted including a linear regression model, a support vector regression model, a fully connected neural network model, and a long short-term memory (LSTM) network model. In the end, The LSTM model performed better since it captured the full aspect of the data and presented more results that more or less correspond to realistic predictions. Looking ahead, more research into electricity consumption would need to be done to get a clear picture of all the contributing factors and then develop a dataset that would be representative of these factors.

References

[1] Project repository, https://github.com/claudeshyaka/ml-final-project
[2] Long short-term memory (LSTM) RNN,
https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html
[3] How to use PyTorch LSTMs for time series regression,
https://www.crosstab.io/articles/time-series-pytorch-lstm/
[4] Hourly energy demand generation and weather,
https://www.kaggle.com/datasets/nicholasjhana/energy-consumption-generation-prices-and-weather?select=weather_features.csv
[5] Pytorch LSTMs for time-series data,
https://towardsdatascience.com/pytorch-lstms-for-time-series-data-cd16190929d7