Forecasting Electricity Load for Commercial Buildings

Matthew Barrett - 16383396
matthew.barrett1@ucdconnect.ie

Damian Dalton - damian.dalton@ucd.ie

Table of Contents

List of Important Abbreviations Used Within

ANN	*Artificial Neural Network*
SVM	*Support Vector Machine*
GA	*Genetic Algorithm*
IEMD	*Improved Empirical Mode Decomposition*
ARIMA	*Autoregressive Moving Average*
WNN	*Wavelet Neural Network*
GPR	*Gaussian Process Regression*
BDG	*Building Data Genome (Project)*
MARA	*Multivariate Auto regressive Algorithm*
RNN	*Recurrent Neural Network*
LSTM	*Long Short-Term Memory*
CNN	*Convolutional Neural Network*

Report Notes:

The following are new sections added as part of Final Report:

- Data and Context
- Core Contributions
- Evaluation

The following modifications were made to the interim report:

- Extension of literature review regarding Long Short Term Memory (LSTM) and Recurrent Neural Networks.
- Introduction of literature around Convolutional Neural Networks (CNN) and their use in time-series forecasting.

Abstract

This project endeavors to build a scalable machine-learning based solution which can be used to predict the electricity load for one or many buildings.

To accomplish this a solution consisting of two modules is required: a pre-processing module and a prediction model module. This paper will identify the most appropriate pre-processing module to prepare the data before it is fed into the prediction module, focusing on workflow software. A comparison of the pros and cons of different software will illustrate what piece of software is the most useful.

A similar approach will be applied to establish which machine learning model is most appropriate for prediction of load forecasting. The factors which will be taken into account when deciding on the most appropriate model are the complexity of the model, its efficiency, and its ability to accurately and consistently forecast electricity loads

Once a model has been established this will be tested on a dataset obtained from University College Dublin, to validate the accuracy of the paper's proposed solution.

Project Specification

This project will collect building energy sensor data, containing usage information as well as various metadata about the building such as occupancy and weather outside. This will be gathered primarily from the Building Data Genome Project. Literature and exploratory analysis of the available data and Machine Learning models will be conducted. The analysis will consider the quality of the data and the accuracy, consistency and efficiency of the machine learning models. A workflow based model execution engine for the forecast models will be designed and implemented. Utilising this, a case study based on University College Dublin campus data will be conducted, including an analysis of the results.

Beyond these primary goals, this project will aim to deploy the project in a cloud based environment and perform tests at scale. Efforts will also be made to design API ingestion software with an associated database for data collection and result analysis.

Finally where possible a web interface with results, metrics and analysis will be designed and deployed.

1. Introduction

This project aims to design and implement a fully scalable prediction tool for load forecasting in commercial buildings, which is not only capable of performing complete pre-processing of data but also can forecast electricity load accurately based upon this processed data. Many existing models can accomplish some of these feats but what sets this project apart is the intention to create a robust generalised tool which can be applied in any scenario without the necessity of fine-tuning, which can often be expensive, both financially and time-wise, and heavily reliant on individual experience gained through trial and error. A real-time data pre-processing model using workflow-based software is a viable solution to this issue.

Initially a literature review will be conducted to identify the most viable methods for the individual modules of the project, workflow software and different prediction models. Several workflow software models will be identified and so each will be compared, contrasted and ultimately the most appropriate model will be selected. Similarly, many papers focus on various models for forecasting electricity loads, comparing models such as Artificial Neural Networks, Support Vector Machines and regression models. An in-depth review of these papers will be conducted to identify what is the most logical choice.

Data will be collected which can be used to train the machine learning models. An understanding of the necessary transformations, such as normalization, standardization, etc., for the data will be sought, and this will form the basis of the workflow model. The prediction model will then be trained using this data, and a test can take place on a dataset obtained from University College Dublin. A review and analysis of the results will be insightful as to the accuracy of the model.

There are many driving factors which make accurate electricity load forecasting models a pertinent issue, the most obvious and pressing of which is climate change. Carbon emissions are one of the largest driving forces of climate change and with data being published **[1]** that shows CO2 rates are increasing, a solution is badly needed. Considering buildings accounted for an estimated 41.1% of primary energy and 74% of the electricity in the U.S. **[9]**, the necessity for accurate energy prediction models is obvious.

According to **[9]**, more than 25% of the 713 GW of the U.S electricity demand in 2010 could be dispatchable, meaning created and used on demand, rather than being created surplus to demand, if buildings could use advanced building energy systems. This is massively appealing, not only due to the fact that it will slow down climate change, which is projected to wreak irrevocable havoc, but also in part because of the financial incentive tied to lowering carbon emissions for governments.

The Effort Sharing Regulation adopted by the European Union in 2018 sets out emission targets for European member states to lower carbon emissions across the EU by 30% before 2030. Under this regulation, if targets are not met for one member state, the option is available to buy carbon credits from member states who have exceeded their goals. Therefore there is not only financial incentive for governments to reach their emission goals, so as to not have to pay for extra carbon credits, but also to surpass expectations and generate revenue through the selling of surplus carbon credits.

On a more local level, the introduction of accurate short term electricity forecasting will allow building operators to save money as they are no longer paying to offset wasted energy which could have been dispatched on-demand instead. Amidst the backdrop of climate change, coupled with the granular benefits of load prediction, it is easy to see why the pursuit of an accurate, scalable and complete energy prediction solution is desirable.

2. Related Work and Ideas

Section 1 - Pre-Processing Model

Dagster:

Dagster is a python-based workflow software for creating data pipelines. Data pipelines should be expressed as DAGs (directed acyclic graphs) of functional, idempotent computations where individual nodes in the graph consume their inputs, perform some computation, and yield outputs, either with no side effects or with clearly advertised side effects **[17]**. Any Python libraries such as Pandas, Spark or even calls out to SQL can be used. Testability is prioritised in Dagster which is important as tests are necessary for this project to measure accuracy of the data pre-processing.

Dagster can be deployed to any workflow engine and mentions Apache Airflow as one possibility. It can also be executed by itself but this is recommended for pipelines of moderate size. Each pipeline is composed of solids which are units of computation with defined inputs and outputs **[18]**. These solids are then linked together within the DAG using data dependencies, which allows for checks of the data as it is executing to ensure that the processing is going as planned.

When executed data is passed through the pipeline following a certain execution order. Run-time checks are completed by data dependencies specified by the designer. All data is then outputted at the end. In the context of this project that would mean the data is cleaned and transformed for use in the prediction model. A workflow is the most appropriate method in this case because it allows the execution of multiple tasks in real-time over a stream of data. Data from buildings/sensors will be received in real time and so a scalable real time solution such as a workflow data pipeline makes a lot of sense.

The advantage of Dagster specifically is its ability to perform real time data dependency checks. This is useful as data could be extremely varied depending on the source. Dagster's ability to check in real time would allow inconsistencies to be caught during processing and allow noise and outliers to be identified and removed. Dagster can be summarised as follows:

- Python-based workflow software for creating data pipelines.
- Easy integration with other workflow software such as Airflow.
- Real time data dependency checks and sufficient scalability.

Airflow:

Apache Airflow is used to programmatically author, schedule and monitor workflows **[19]**. Similar to Dagster it also works on the principle of creating Directed Acyclic Graphs or DAGs of tasks, through which data is processed. There is an extensive user interface which can make the task of analysing data which is being processed far more intuitive.

Airflow pipelines are created in code, specifically python which allows for dynamic pipeline generation. Consequently code can instantiate new pipelines on the fly. A key feature of Airflow is its modular architecture and use of a message queue which allows it to scale to infinity. As the scope of this project is to create a scalable solution, the ability to scale at such levels is a massive benefit.

Airflow has its shortcomings in the area of data streaming. On the Apache Airflow site it is explicitly stated that Airflow is not a data streaming solution as tasks do not move data from one to the other. The data pipeline necessary for this project will focus on taking an array of data from buildings and their sensors and performing a multitude of tasks on them. Feature selection, data cleaning and the removal of outliers/noise, aggregation, normalisation and data transformation are all tasks which may need to be done to incoming data. Reliance on metadata dependencies only to verify that processing is working as expected may prove difficult.

A strength of Airflow is its strict adherence to scheduling. This is useful for regular task execution, such as in the case of this project where data will need to be processed at regular intervals, if not in real time. However the inability to exchange data as discussed earlier means Airflow alone is not capable for this project. Airflow can be summarised as follows:

- Python-based software for authoring, scheduling and monitoring workflows.
- Massive scaling potential.
- Inability to stream data.

Toil:

Toil is described as a scalable, efficient, cross-platform pipeline management system written entirely in Python and designed around the principles of functional programming. Toil's main focus is to achieve a complete and robust experience of processing large-scale analyses across multiple environments **[20]**.

A key advantage of Toil over its competitors is fault tolerance, with integrated support for arbitrary worker and leader failure and with strong check-pointing that always allows resumption **[21]**. This could be a unique factor in this project. Large datasets, such as those from a set of buildings, being involved means computing cost can be incredibly high and so the distribution of the task across multiple machines is necessary. Strong fault tolerance ensures that no issues will arise from this.

Toil is also fully scalable and ready to be deployed into the cloud. Again, with distribution of the workload across multiple machines likely to be necessary, cloud solutions are incredibly appealing and so Toil's boast of processing upwards of 20,000 RNA-seq samples using a commercial cloud proves eye-catching. A significant upside of strong fault tolerance is that low-cost machines that can be terminated by the provider at short notice are available from commercial clouds. Utilising these machines can offset costs drastically and is made possible by the strong checkpointing and ability to resume of Toil. Toil can be summarised as follows:

- Cross platform workflow software.
- Strong fault-tolerance allowing distribution of tasks safely over multiple machines.

Section 2 - Prediction Model

Forecasting electricity load for buildings has been the subject of many papers **[3][6][7]**, several of which focus on individual models and discuss their advantages and shortcomings. According to a review paper **[12]**, 81% of studies focus on models for commercial and/or educational buildings, while 84% of the studies focus on short-term energy consumption prediction. Both of these are primary goals for this project. Comparisons of different state-of-the-art models are also offered, giving an overview of benefits and drawbacks if one is chosen over another. A common theme throughout most papers is discussions of issues faced in obtaining accuracy, no matter the model;

Short-term load forecasting models require a large amount of inputs. While this can be seen as an advantage, as it implies there is a good depth and diversity of data, it also raises a lot of issues in the fine-tuning of the model. This issue is described in the context of building energy performance simulation programs as involving thousands of input parameters which in turn creates a highly under-determined problem, meaning there are too many variables **[2]**. Some inputs are negligible while others should carry greater weight within the model. This means the process of calibration is extremely important.

Unfortunately the process of calibration is extremely poorly documented because where data is well calibrated the information is usually considered proprietary and confidential, meaning it cannot be made publicly available. The most commonly used optimization ‘engines’ are identified as GenOpt and Matlab optimization toolboxes, although these are generic and not specific to model requirements **[13]**. Much of the process of calibration relies on user knowledge or expert judgement as well as trial and error because Generally, this procedure is not well defined, in that the analyst decides on model changes based on personal judgement as opposed to quantifiable evidence **[2]**. This is also another major issue as it is heavily time-consuming for any new-comers to the field.

Excluding exogenous factors diminishes the accuracy of a short-term load forecasting model. Some models do not take into account these factors such as weather or climate. Such an example is a hybrid singular spectrum analysis method-autoregressive model as once the outside situation, such as political, economic, and climate condition, has a sudden change, this method may not work **[7]**. This would not be an issue in some applications but in the case of forecasting for a campus like University College Dublin, exogenous input is essential, as Ireland is known to have extreme weather events, and campus also regularly closes for holidays, etc.

The background research revealed 3-4 prediction models suitable for load forecasting prediction in buildings.

The hybrid forecasting model was revealed as a candidate for this project. This forecasting model would comprise of three modules; a pre-processing module, a forecasting module and an optimization module. The first step is to input lagged load data (data which has certain inputs aligned different to identify correlation) correlated with important exogenous variables such as meteorological data. This is fed to a feature selection technique which will remove irrelevant and/or redundant samples from the inputted data **[3].**

This processed data is used in the second module which is a sigmoid activation function. This is a logistic regression function which predicts categorical outcomes. This gives a value of probability between 0 and 1 and is then used in the training of the forecasting module, which is an Artificial Neural Network. It uses multivariate auto regressive algorithm (MARA) for training, because it can train the ANN faster than Levenberg-Marquardt algorithm and Gradient Descent Back Propagation algorithm. This is important to note for further usage as it clarifies which training algorithm for a neural network is best in this scenario. When the neural network is trained it begins to make day ahead load forecasts. Similar to what was mentioned previously operators of the neural network are expected to fine-tune some adjustable parameters to increase forecast accuracy. The final step of this model is to introduce feedback in the form of error correcting functions by comparing predicted data to actual data. This is extremely useful in improving the accuracy of day ahead load forecasting and a key takeaway from this paper.

This approach outlined in the project has large relevance to the task I am trying to accomplish as an Artificial Neural Network is also used. Despite being described as a hybrid model, much of the core of this model is the neural network and thus makes it is a viable option in the future for my project. The main disadvantage of the model is it would have to be fine-tuned but this it to be expected in any model. The model was tested on three smart-grids in the USA and was shown to achieve a forecast accuracy of 98.76%, which the authors claimed was relatively better than existing techniques.

Artificial Neural Network has been shown as a strategy for short term load forecasting when set up with past energy, Environmental and building/occupancy data **[4]**. To achieve a more balanced view comparison papers must be consulted. In such a paper, load forecasting techniques are arranged into four categories as follows;

- Statistical technique

- Artificial intelligent (AI) technique
- Knowledge based expert systems
- Hybrid techniques **[6]**

The preferred approach of this project falls into the second category of artificial intelligent techniques. Therefore a comparison to other state-of-the-art approaches here is crucial.

It is noted that artificial neural networks can be seen as competitive with other models as far as the forecasting of load profiles is concerned. This is due to many factors, much of which are discussed above but it was found in a study using building data from a local utility in Rio de Janeiro to predict 24 hours in advance, that large neural networks perform best, with the smallest mean absolute percentage error but also with a lesser spreading of errors.

ANNs have also been described as a primary position in a large number of applications ranging from load forecasting to retrofit potential estimation. However, it is notable that there are challenges to be faced when working with an ANN. Primarily, the choice of an appropriate architecture and learning rate are important to ensure results and as minimal error as possible.

Support Vector Machines are highlighted as competitor to ANNs in the field of forecasting electricity load, with similar levels of complexity and accuracy. SVMs do suffer from some disadvantages not seen in ANNs such as low running speed **[14]**. This is an issue, most notably in projects with large scope. Unfortunately this is a widespread issue with models which are accurate to a fair degree requiring a large computer memory and processing or computation time **[10]**. The upside of using SVMs is that they require few inputs, so this means the feature selection process is easier. They are also notable for having a relatively simple training process due to requiring a few inputs as mentioned earlier **[11]**.

Statistical regression models are a prominent option for forecasting electricity load. These models are extremely useful for evaluating the importance of potential inputs for models but struggle with short term predictions, with a relatively large amount of inaccuracy in this field.

A hybrid model is proposed **[5]** based on improved empirical mode decomposition (IEMD), autoregressive moving average (ARIMA) and wavelet neural network (WNN). The model is said to perform better in comparison to SVMs based upon case study data from America and Australia and can provide a robust, stable and accurate prediction result.

A method is introduced **[8]** based on Gaussian Process Regression (GPR) which also incorporates physical insights about load data characteristics. It achieved prediction accuracy of up to 94.38% and 99.26% for long- and short-term forecasting, respectively, although interestingly as training data and forecast length increased, so did prediction error.

A new framework based on Long Short Term Memory (LSTM) Network moving window-based technique is described by **[15]**. LSTMs are a form of Recurrent Neural Network, which excel at time-series forecasting due to **[22]** "maintaining a memory cell to determine which unimportant features should be forgot and which important features should be remembered during the learning process".

This approach is said to outperform regression models, Support Vector Machines and Artificial Neural Networks **[15]**. Recurrent Neural Networks can be used to efficiently predict electricity demand. There are numerous advantages such as handling non-linear complexities, minimum prediction errors and ease of generalization.

Research into Long Short Term Memory (LSTM) shows promise, with many studies being conducted to show the efficacy of the model in handling time-series **[15][22][23]**. One such study cites the use of Convolutional Neural Network (CNN) layers in conjunction with LSTM layers as a method of improving accuracy **[23]**. The ability of Recurrent Neural Networks, and more specifically LSTMs alongside CNNs to make proven accurate predictions using time-series data makes them a model for consideration.

3. Data Considerations

For the purposes of this project, large-scale data obtained from commercial buildings is required. Datasets of this kind are challenging to come across as many building managers have no reason to publish this kind of data, especially in a collaborative effort alongside other buildings.

Fortunately the Building Data Genome Project has collated such a dataset **[16]**. Data from 507 non-residential buildings was collected. The data includes hourly whole building electrical meter data for one year as well as meta-data such as area, weather or primary use type. The majority of the buildings in question are from universities or colleges. This is advantageous as this project is concerned with University College Dublin's data. Many of the buildings are based in New York, London, Phoenix and Chicago. A large geographical range of buildings is beneficial as the scope of this project is to create a universal scalable approach to load-forecasting.

The BDG dataset is sufficiently large for the scope of this project, with each of the 507 buildings, each generating 8760 data points over a period of time. 507 buildings should be an appropriate amount to offset concerns of overfitting.

As mentioned above, some meta-data is included alongside the primary data of electricity meters. For all samples there is included the gross floor area and the weather. For some but not all samples there is meta-data relating to heating type, main heating energy source, the number of floors, peak occupancy, energy rating, and year of construction.

Some data cleaning and re-organisation has already taken place on the dataset supplied by the Building Data Genome Project. Efforts have been made to convert all data to a standard format (Coordinated Universal Time), as much of the data is drawn from different sources. Seven campuses were selected for case studies, while more data was drawn from online repositories, although it is notable that the online repositories often offered less meta-data. Initially, outliers were removed and small gaps in data were filled. This was done using the LoadShape model, and its associated Python library. Other names for this model are the Time-of-week and Temperature or (TOWT) model or LBNL regression model. All timestamps were converted to Coordinated Universal Time (UTC) and all data was then collated into one comma-separated value (CSV) file, meaning the data is local static data which has been structured.

While most of the data is collected from the electricity meters on an hourly basis, in some cases there were samples which were collected sub-hourly. To remedy this, they were re-sampled to take the means of the values at every hour. For the purposes of this project, the changes made to the datasets are appropriate and well-documented. Therefore the dataset is suitable for this use case. The creators of the dataset have it under MIT License, meaning it can be used for commercial or private use.

In terms of anticipated data cleaning for the future, there is a possibility that no further data cleaning will be necessary. The removal of noise/outliers and filling of gaps in the data are two of the primary efforts for cleaning of data. Efforts could be made to sub-divide the dataset into samples which have excess meta-data and those that do not. This would allow comparison of the performance of a model trained on basic data and one with a multitude of inputs, although this is contingent on time.

More time will be spent on the data however in preparing it for training the machine-learning model chosen. To accomplish this a data-pipeline will be constructed to perform several transformative tasks on the data.

4. An Outline of My Approach

The approach to this project is contingent on how well implementation of certain aspects go, but the basic methodology will be as follows. Primarily, a data pipeline will be created through which data will be pre processed, one machine-learning prediction model will be chosen and implemented and its results will be tested.

For the data pre-processing it seems that Dagster is the most appropriate. Python is a powerful and universally accepted language making Dagster appealing in that sense. The greatest advantage Dagster offers in several different ways is its versatility. It can integrate with different data storage libraries such as Pandas and Spark and even different languages such as SQL. Another prominent option was Airflow but Dagster even has the ability to integrate with Airflow, again offering more versatility.

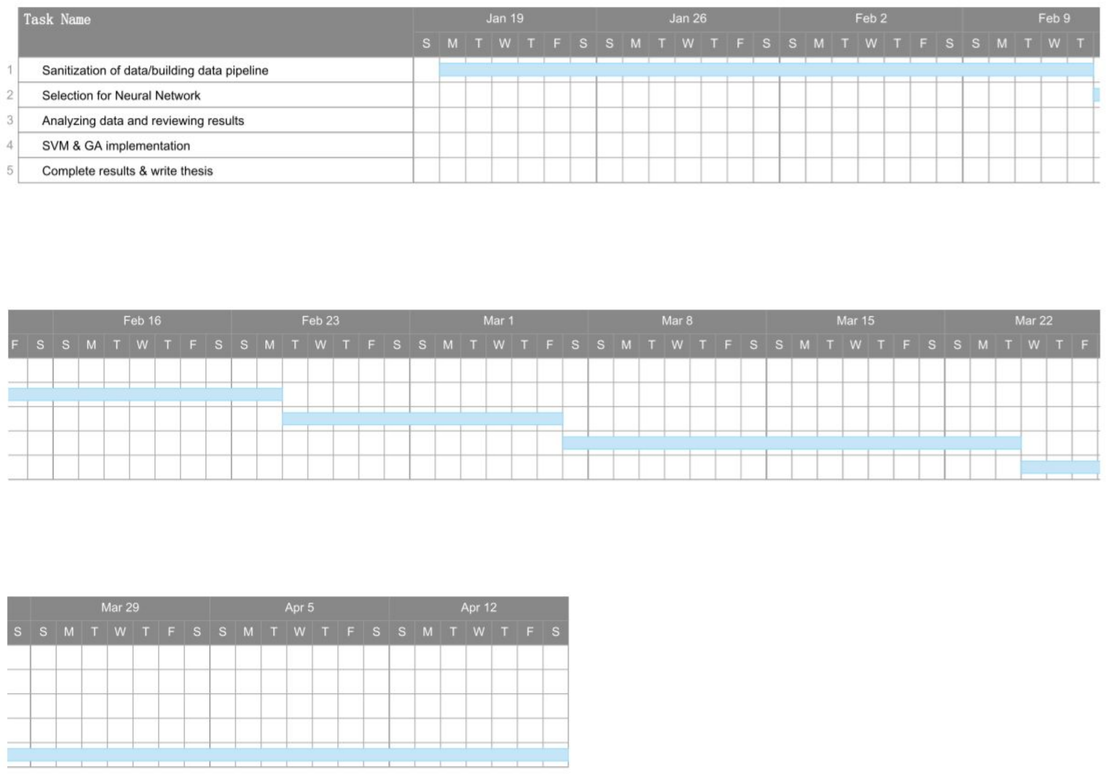
As for tasks to be carried out by the data pipeline, cleaning will be the first step. Although some cleaning has already been done on the dataset, a group of tasks to do so will be necessary to fit the scope of the project of creating a universal scalable solution. Most clearly, removal of gross outliers and filling of gaps in the data are two tasks which are necessary to clean the data. Following this, it is expected that all data will be normalized where necessary to get all values to between 0 and 1. After that standardization to rescale all the values so the mean is 0 and standard deviation is 1. From there data-specific adjustments will be made to ensure necessary features are represented, and after that labelling and splitting of the dataset.

The next step in my approach is the identification of the appropriate prediction model for the project. This is a more difficult choice to make as the pros and cons of each model are far more subtle. By far the most common and accepted approach is that of an Artificial Neural Network. Beyond its popularity, its relatively high running speed and complexity make it an ideal choice. In the case of choosing an ANN there are many factors to consider. There are many parameters in an ANN such as the algorithm, the activation function, the learning rate, the initial weights, the learning rate, the number of training patterns and the number of times the training data is split, to name a few.

The scope of this project is to create a universal scalable machine-learning solution for prediction of electricity load. A vast dataset exists upon which I can train and test my machine-learning model. As such it is difficult, if not impossible to give specific details of the parameters. I would like to train the model using a multivariate auto regressive algorithm (MARA) as this has been cited as relatively faster at training than other popular algorithms. A sigmoid activation function has also been shown to work well alongside this although many other options are available.

This project will aim to carry out a comparison of several models and their capacity as a prediction model for electricity load forecasting. The models that will be analysed are ANNs, Support Vector Machines (SVM) and Genetic Algorithms (GA), but these will all be contingent on time and the success of implementation of an ANN model first.

5. Project Work Plan



There are several key elements of this project which must be constructed to achieve the goal of the project. Primarily, constructing a data pipeline, configuring an ANN and finally testing the results. The construction of a data pipeline is the first task which must be carried out. To do this, a number of factors are involved. A familiarity with not only the data to be processed but also the software being used to process must be gained.

It is anticipated the familiarisation will take 3 days, beyond which an outline of the specific tasks which need to be carried out by the data pipeline can be drawn out. This is anticipated to take 7 more days after which 2 weeks will be spent on creating the data pipeline itself and on testing of the pipeline.

With the data pipeline in place the next step is to look to the prediction module of the forecasting solution. As has been stated before the primary goal is to implement an ANN initially. Similar to the data pipeline, 1 week will be taken to familiarise with the various parameters of the ANN and how they relate to the processed data from the pipeline. After this a process of training and testing the ANN will take place. This has been described as quite time costly in multiple papers so it is anticipated this will take about 3-4 days for this fine-tuning stage. After a desirable result has been achieved testing will be carried out on a new dataset supplied by University College Dublin. Data will be reviewed and analyzed and this is expected to take 1 week.

Pending the success of the implementation of the ANN, a similar process will be repeated for a Support Vector Machine and Genetic Algorithm, but this process is expected to be faster due to relative experience obtained from the ANN. These are expected to take 1 week each.

Review of the results and completion of reports is then expected to take 3 weeks after this.

6. Summary and Conclusions

In conclusion, the objectives of this project are clear; create a universal scalable machine-learning solution for prediction of electricity load. From my findings, it is clear that this area is well researched and as such many solutions exist at the moment. What separates my project from these is the emphasis being put on a universal solution. Models are usually only tested on a handful of case studies which means there is uncertainty about the ability to perform without context.

The scope of this project is without doubt ambitious but achievable due to the work done by others in the field to pave the way as well as the availability of a large collated dataset which will make the training and testing of the model far more advanced than others in the field. The Building Data Genome Project's considerable repository of building data will go a long way to making a reliable model.

The necessity and drive for this project are clear. Climate change is an issue which, if not addressed, will be disastrous for not only the planet but the human race. With buildings making up a massive part of carbon emissions, it only makes sense to me that tools to curb these emissions in whatever way possible should be a great priority. From a computer science perspective it is a great opportunity to showcase the power of machine-learning prediction models and their ability to solve real world problems.

7. Data and Context

The data used in this project was obtained from the Building Data Genome Project, a curated repository of 507 public buildings, made freely accessible online. For the purposes of this project the focus was on a single building, identified as "Prim_Class_Jaylin". From the available metadata, it is known this is a school building located in Europe.

The data used was static raw data. The raw data came in the form of three types of Comma Separated Value (CSV) files:

1. temp_open_utc.csv - this was the primary file which contained all metering data. Each building's metered data is represented in a column with each entry indexed by a timestamp.
2. meta_open.csv - this file contained the meta data of each building, including the weather file containing the building's weather data.
3. Assorted weather CSV files - there were 39 different weather files, containing weather information for different regions over time periods. The features included in each file were:
    a. Timestamp
    b. Conditions
    c. Date (UTC)
    d. Dew Point (C)
    e. Events
    f. Gust Speed (Km/h)
    g. Humidity
    h. Precipitation (mm)
    i. Sea Level Pressure (hPa)
    j. Temperature (C)
    k. Time (BST)
    l. Time (GMT)
    m. Visibility (Km)
    n. Wind Direction

o. Wind Speed (Km/h)
p. WindDir (Degrees)

Each building had roughly 8000 hours of data, mostly taken hourly. This repository was an excellent resource as most commercial building data is not released due to privacy concerns. A collection of 507 varied commercial buildings and their relevant meta-data proved invaluable. The data was collated by members of the Building Data Genome Project, from various online repositories and as a result of some site visits to colleges and universities worldwide.

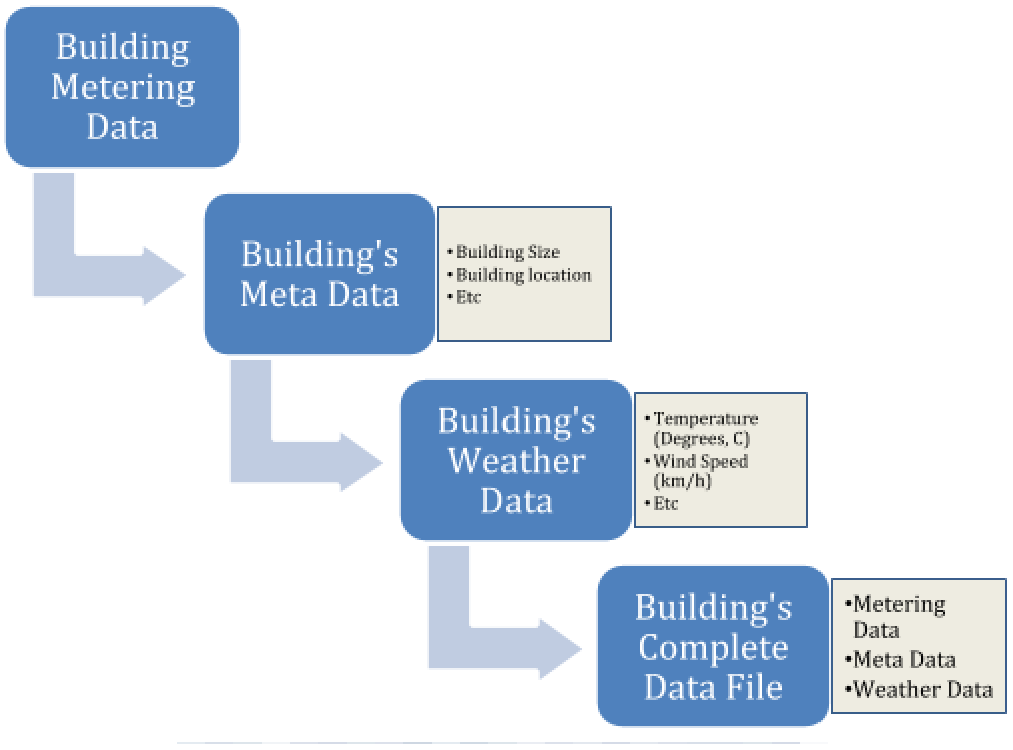
8. Core Contributions

The first stage of the project was preparing the data. The data was structured in three separate files, meaning collation of the files was necessary as well as sanitization (removal of outliers, filling in gaps), and finally normalization. The project initially had planned for the use of Dagster, however I found that in practice this was unnecessary for the intended usage. Due to extended familiarity with the Python Pandas Dataframe and Numpy libraries, this seemed to be a much more straightforward option, while also having the added benefit of decreasing the per-test runtime of the entire project. The data of each building must be put through a preparation process only once before being used for prediction. This, alongside the fact that the data was static, meant a streaming solution was unnecessary.

The objective of collating the data was to create a single building CSV file which would contain all data necessary to render a prediction. Usage data was essential as this formed the core of the predictions. Weather data would also prove very useful for making multivariate predictions, provided there was a correlation between the weather measurement and the usage data. Since each building's corresponding weather file was denoted in the meta-data file, all three different CSV files would have to be utilized in order to form a building master file from which predictions could be made.

Each file was read in individually. A function was created to handle the weather files separately as these would need the most preparation. There were several issues with the weather data. One column header, wind speed, contained a backslash which was causing trouble when it was being read in due to its use as an escape character. This meant the column had to be renamed with the backslash removed. Columns with the most relevance and consistent data were then selected, with irrelevant ones being discarded. The wind speed column again presented issues as there were some instances of the word “Calm”, despite the rest of the column being numerical data. Due to the regularity of recording it seemed acceptable to remove these instances and interpolate values using a mean of the values before and after. All weather timestamps were converted to UTC (Coordinated Universal Time) as this was what the usage data was recorded in. The usage data was recorded on the hour every hour, while weather data was recorded at 30 minute intervals, at 20 minutes past the hour and 50 minutes past the hour. This meant that the weather data had to be resampled into an hourly format for the purposes of collation. A check for missing or unusable values (known as NaNs, meaning Not a Number) is performed and any rows which are missing values are interpolated by taking an average of the values before and after each NaN. A Z-score outlier check is performed, using a high threshold to eliminate any values which are far from the mean. A high threshold was chosen because the data was mostly as expected, apart from a few extremely deviant outliers, meaning outlier removal was only focused on these values. Finally some simple functions were created to return the day of the week, day of the year and hour of the day. These were included to account for the possibility of strong temporal correlation with the usage data.

For the creation of building master files, each column of the usage data was iterated through. A data frame was created to contain the columns data. The building name was then matched to its meta-data entry and this was used to match it to its weather file. From there the name of the weather file was fed into the previously discussed function and the weather data was prepared and returned. The weather file was then merged with the usage DataFrame and this was finally outputted to a comma separated value (CSV) file, which would serve as the buildings master data file. A graphic representation of this process is seen in Figure 1.



Following this there is a second stage of data preparation. Where the first stage involved the collation of multiple files, this stage focuses on the preparation of a single file for use in making a prediction. A buildings master data file is read into a Pandas DataFrame. Each column's values are standardized using the mean and standard deviation of the training data. This is done to account for the fact that each of the different column's data is in varying domains and thus there would be no correlation if they were not standardized. In Figure 2 we can see the results of this. The shapes of the plots remain the same, while the values on the

Y-axis are all transformed into a uniform range.

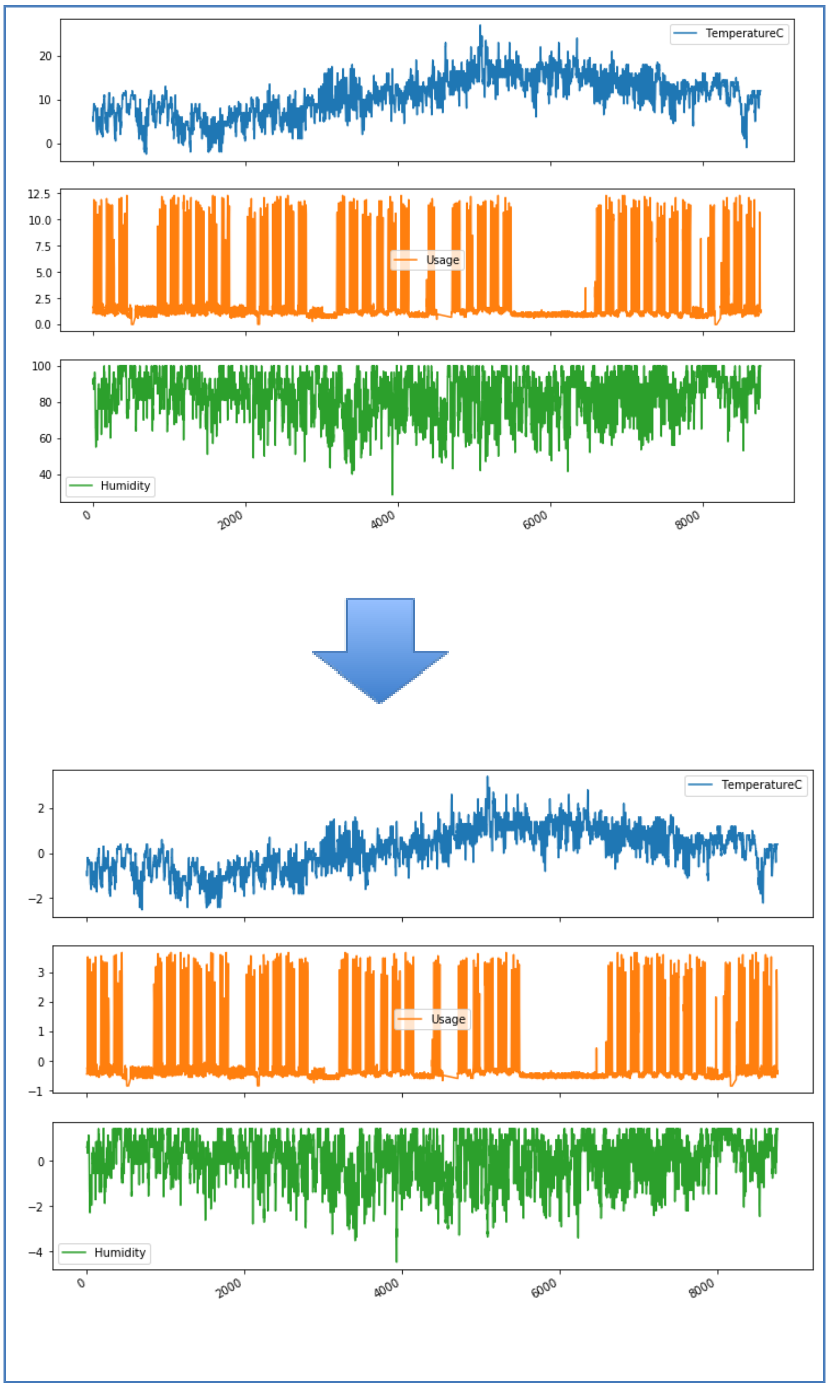


Figure 2 – Standardization of the data

Following this standardization, a test was performed to evaluate the pairwise correlation between different features, using the Pearson correlation. Each feature was paired with each other feature. If the features in a pair had similar trends, they would score higher, indicating correlation. In Figure 3 we can see the which features had the most correlation with the metering data feature across the entire dataset. Figure 4 using a time-lagged Pearson correlation calculation. This meant the correlation was calculated across a rolling 24 hour window and then collated.

Feature	**Correlation**
Day of the week	0.214753
Humidity	0.129895
Wind Speed (Km/h)	0.089089
Visibility (Km)	0.078085
Dew Point (C)	0.058925
Hour of the day	0.026198
Sea Level Pressure (h.Pa)	0.009954
Temperature (C)	0.004657

Figure 3 – Pearson Feature Correlation

Feature	**Correlation**
Humidity	0. 192860
Hour of the day	0. 121888
Temperature (C)	0. 101088
Day of the week	0. 087895
Wind Speed (Km/h)	0. 053891
Visibility (Km)	0. 052345
Dew Point (C)	0. 044371

Sea Level Pressure (h.Pa)	0. 002420

Figure 4 – Pearson Feature Correlation (Time-Lagged)

The key factor which we are trying to predict is future electricity metering data (referred to in the dataset as "Usage"). We can see the features with the largest correlation are the day of the week, the humidity, and the wind speed. These features are then selected to be used when the models are making predictions.

Specific preparations must be made for inputting the data to the first of the two models, the Support Vector Regression, which is a specific usage of a Support Vector Machine model, identified earlier in the literature review. The Support Vector Regression (SVR) takes input in the form of two separate feature sets: the dependent variable (value to be predicted) and the independent variables (the values which the prediction is based upon). The independent variables will be comprised of the best scoring features from above, the day of the week, the humidity, and the wind speed. Since the objective is to predict the building energy usage 24-hours ahead it is necessary to create a new feature containing the usage data shifted ahead 24-hour ahead for the dependent variable. This is accomplished again using Pandas DataFrames.

From there we begin splitting the data. An 80:20 ratio is selected for dividing the data into training data and testing data. Within each of these data is then divided up into batches, with each batch containing 256 instances. Each instance is a 24-hour window of the independent variables and the dependent variable. TensorFlow is then used to shuffle, batch and cache the dataset. The SVR is fitted to the training dataset and then used to predict 24-hour windows for the test dataset. The graph below shows a comparison of actual and predicted usage for one 24-hour window as well as the SVR's prediction across the entire validation dataset.

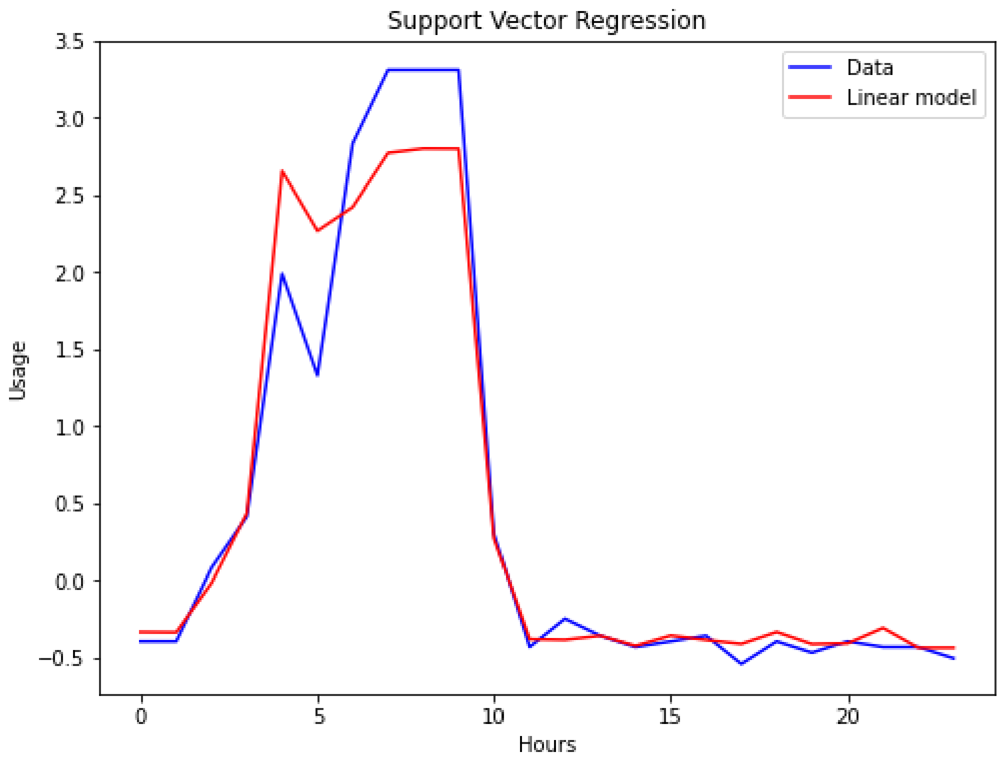


Figure 5 – Prediction of 24 Hour Window (SVR)

We can see in Figure 5 that the SVR is making reasonable guesses. A further investigation and comparison of the accuracy of the model will take place in the Evaluation section.

To compare the results from the SVR, the next step was to create a neural network which used the same structure of input and output data, 24 hours in and

24-hour prediction out. From the literature review, Long Short-Term Memory (LSTM) models were identified as the most promising model so this made for the best choice. is a type of artificial Recurrent Neural Network (RNN) which is known to be well-suited to making predictions based on time series data, as noted in the literature review. This is because there can be lags of unknown duration between important events in a time series, making LSTMs, which have feedback connections, an advantageous choice.

For this project's implementation of an RNN, the python library TensorFlow's API, Keras, was used. This high-level API allowed for the fast prototyping of different models, to find a suitable structure for predictions. Firstly the data was split into training data and validation data then shaped into an acceptable format for the RNN. A key stage of the process is the design of the neural network itself, and specifically what layers are going to be used. This involves knowledge of what different types of layers are available and how applicable they are to the scenario, but also a reasonable amount of trial and error to resolve what works best. A common technique used in RNNs when working with time-series is a combination of three different layer types: a convolutional neural network (CNN), a local max pooling (MaxPooling) and finally the LSTM.

Data is passed through a convolutional layer followed by a MaxPooling layer. This is repeated before the data is passed through 2 hidden LSTM layers, before being sent to a Dense layer which is fully connected and serves as the output. In my literature review a sigmoid activation function had been identified as the best option for the task of prediction. In practice a ReLU (Rectified Linear Units) function was more computationally efficient and empirically offered better results. This is evident in Figures 6 and 7, as we can see the training and validation loss displayed for both Sigmoid and ReLU. This measures the mean absolute error of predictions by the model as it is trained.

The role of the convolutional layers is to extract features from the time-series. They do so by iterating over the entire dataset, considering a specified number of data points at a time (defined by the filter size). Filter size of 10 and 20 were chosen through a process of trial and error. A kernel size of 3 was chosen for both, again through trial and error. A pool size of 2 and 2 strides were selected for the MaxPooling layers. For the first LSTM layer 200 units were chosen, while the second layer was assigned 25 units. These units were chosen as the result of fine-tuning through a process of trial and error.

An issue encountered was a problem with overfitting the model to the training data. Given the relatively small size of the dataset there was a tendency for validation loss and error to be far higher than its training counterparts. A proposed solution to this was adding Dropout layers. Layers with dropout rates of 0.1 were added after each run through the convolutional and max pooling combination and finally a layer with dropout rate of 0.3 was added before the fully connected layer. This means, in the case of the final layer for example, 0.3 of the units will be set to 0, to prevent overfitting. These values were also chosen through a process of trial and error.

Due to the time and resources that would be required to iterate over the entire dataset each time, the model was trained using 200 training steps and 50 validation steps per epoch over 10 epochs. As expected, the more iterations of the dataset (epochs) it is trained on, the lower the loss and thus the more accurate the results, seen in Figure 7.

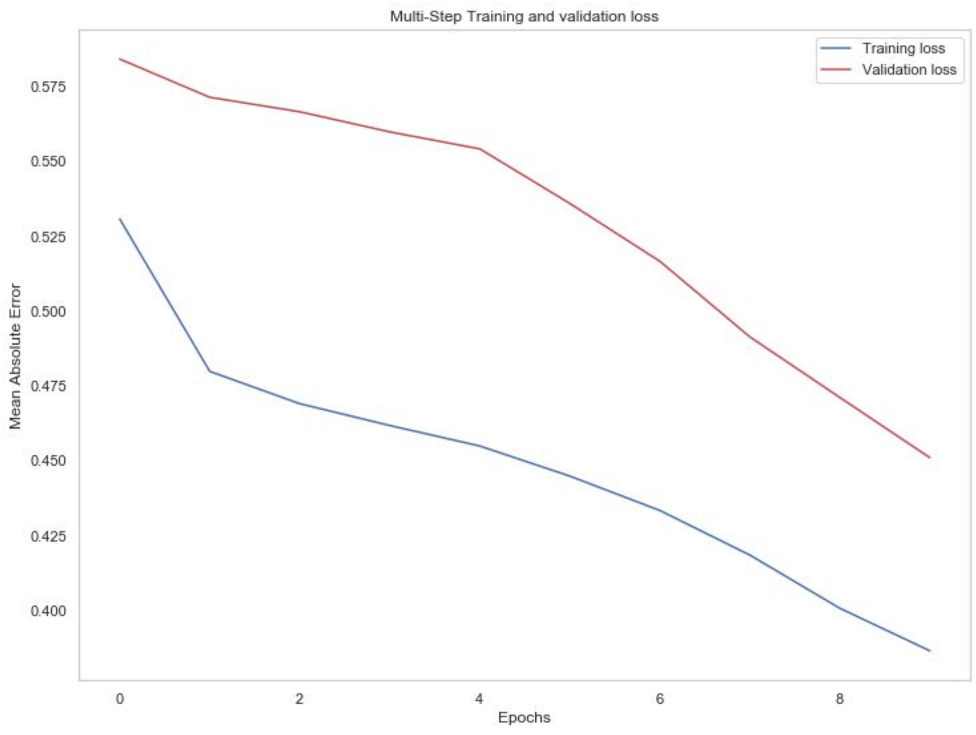


Figure 6 – Training Loss of LSTM Model (Sigmoid)

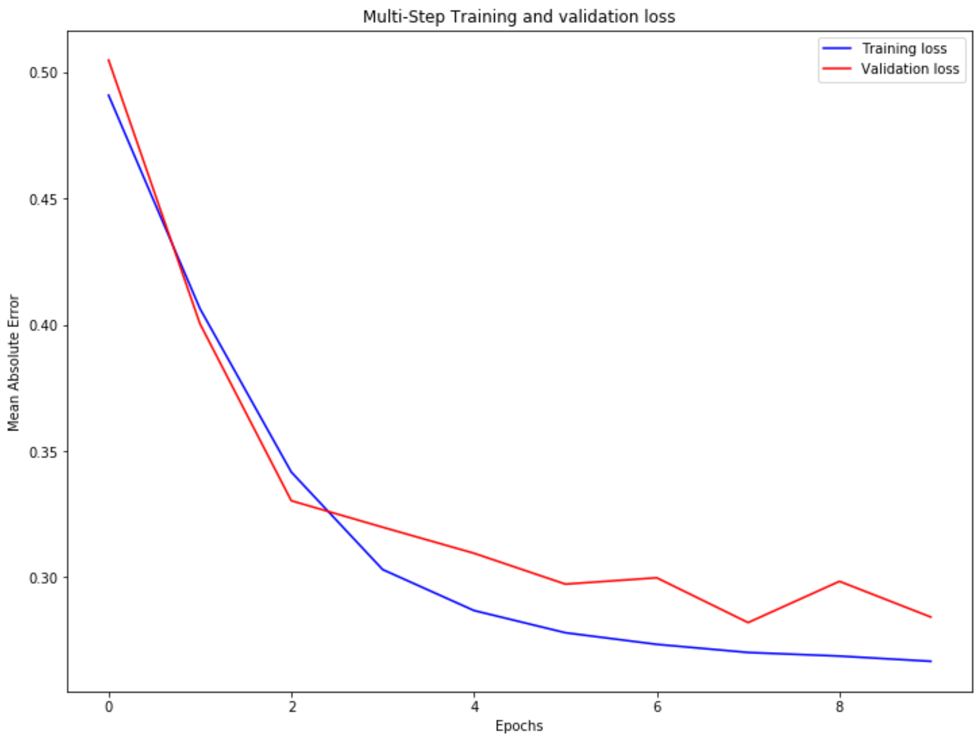


Figure 7 – Training Loss of LSTM Model (ReLU)

A similar model was trained using a larger window to determine what the optimal window size to train a recurrent neural network was. 48-hour and

72-hour windows were trialed as inputs. This meant the function which split the data had to be modified as well as the input layers of the RNN. These models were then trained and evaluated using similar methods and metrics to compare to the results of a 24-hour input window. These comparisons can be seen in the Evaluation section.

9. Evaluation

The basis upon which my evaluation will be formed is a comparison of the accuracy of the results achieved in predicting 24-hour windows by the different models I have implemented. The models will be trained using the same data and make predictions using the same unseen data. Comparing the error of these predictions using different metrics will give a clear picture as to which of these models performs the best at predicting energy usage using multivariate time-series data.

The two main types of models being evaluated are the Recurrent Neural Network (RNN) and Support Vector Regression (SVR). SVRs are commonplace methods of prediction which do an adequate job of predicting energy usage. It is expected however that the RNN will perform better. This was identified earlier in the literature review, not only that neural networks perform better but specifically recurrent neural networks will serve as one of the best prediction models due to the fact that they are able to recognize patterns over large inputs as a result of their feedback architecture.

The modelling for the project centered around data retrieved from a single building from the Building Data Genome Project. In all models this data was split into training and validation data. This validation data is unseen to the model and therefore can be used to evaluate each model's accuracy. Validation data is split into batches of 256 instances. Each instance contains 24 data points for each of the 3 dependent variables (Usage, Humidity and Wind Speed) as well as the independent variable to be predicted, usage 24 hours ahead. Each model is given the 256 instances as input and asked to predict the 24 hours ahead. These results are then compared against the actual 24 hour ahead usage values to determine the accuracy of the predictions.

For each instance, the Mean Average Error (MAE), Root Mean Squared Error (RMSE) and Symmetric Mean Absolute Percentage Error (MAPE) were calculated. A mean of these values was used as a metric of comparison. The results are shown in the graph below.

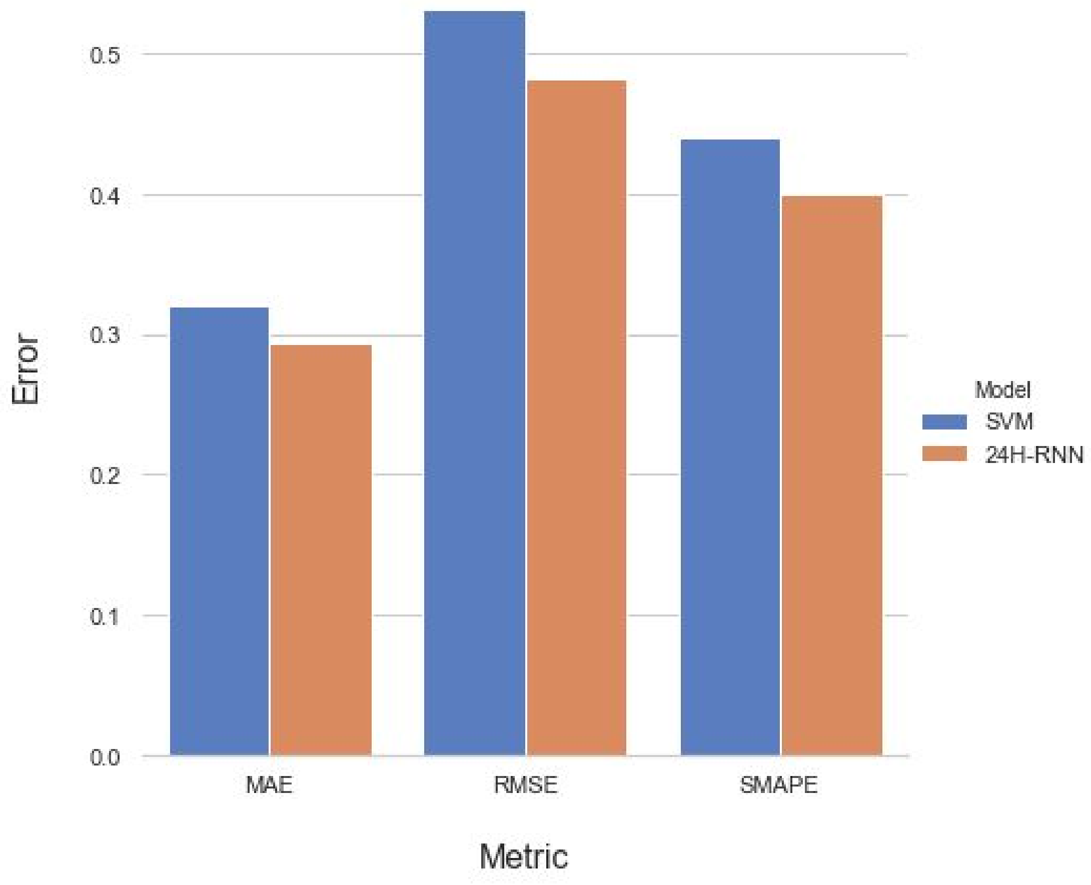
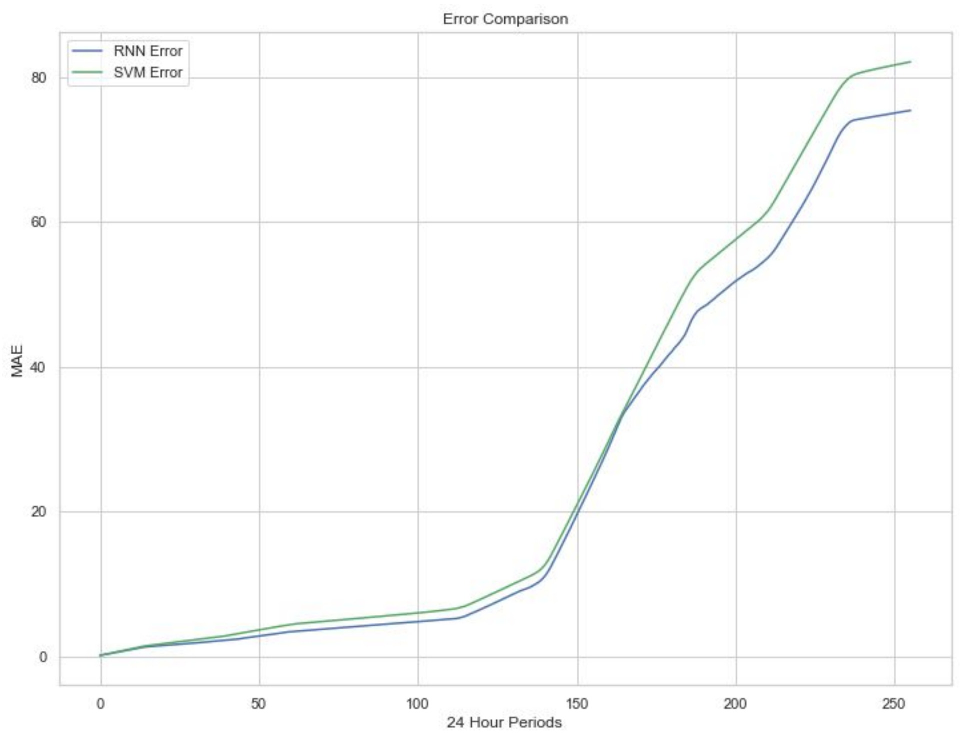


Figure 8 – Comparison of Error Between SVM and RNN Models

As we can see from Figure 8, the RNN consistently performs better than the SVR across all metrics. In MAE, RMSE and SMAPE, a lower value is desirable, and this is achieved by the RNN. MAE is a linear score, meaning all differences between the predicted result and actual result are weighted evenly, while RMSE places greater weighting on larger individual errors. The RMSE is larger than the MAE, indicating a large variance in the size of errors. SMAPE limits the effects of gross outliers when calculating the error in a symmetric fashion. We see its result are very similar to the other metrics, meaning there are few gross outliers present. These results were replicated across numerous runs. A representation of the cumulative MAE and RMSE, respectively, of both models as they predict the 256 instances is shown in Figure 9.



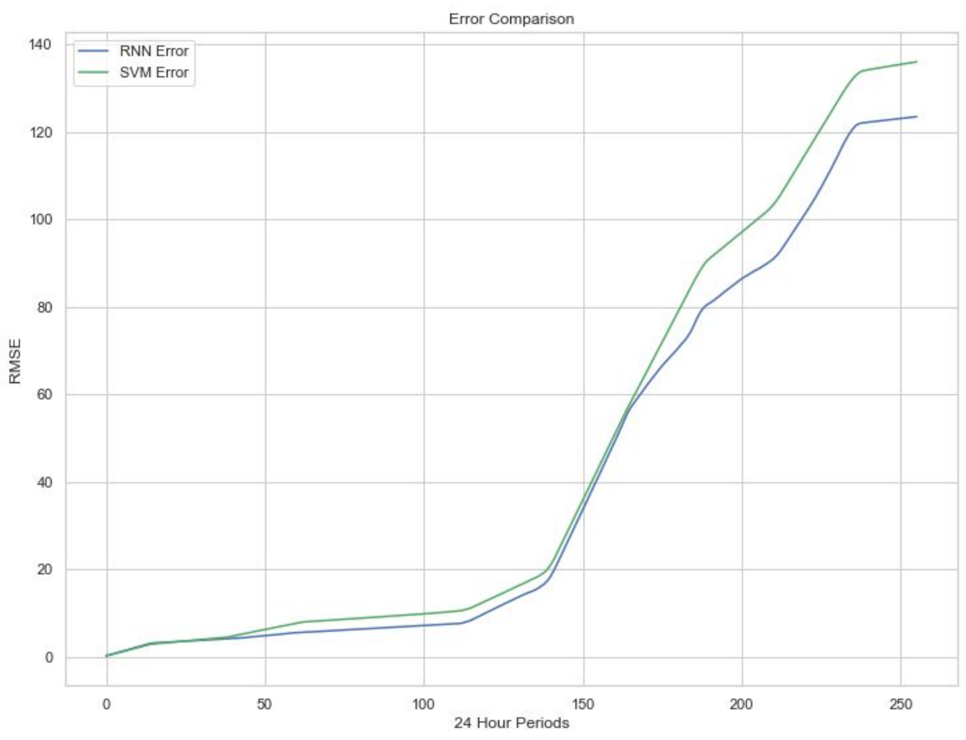


Figure 9 – Cumulative MAE and RMSE of SVM and RNN

A comparison of RNN models using different sizes of training windows revealed that the optimum training window size to predict a 24-hour period was 48 hours of input. The results of these comparisons are shown in Figure 10.

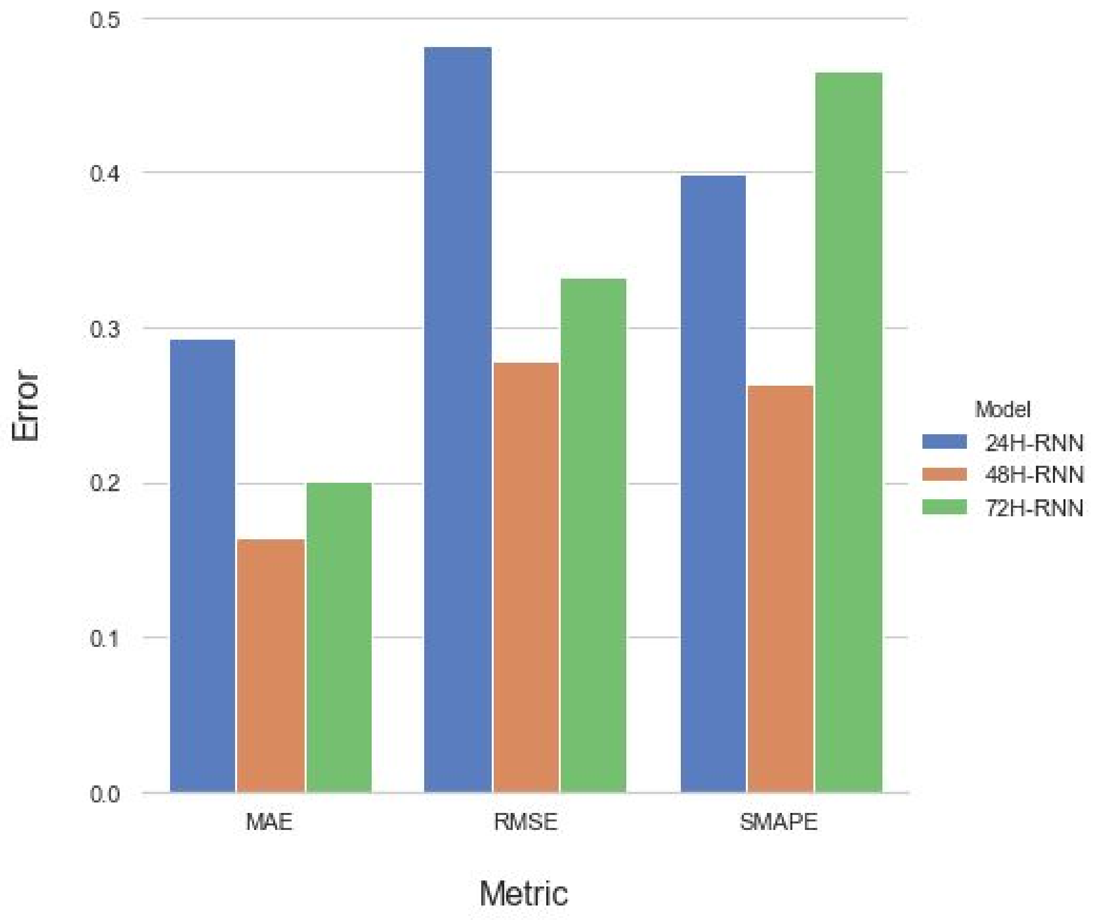
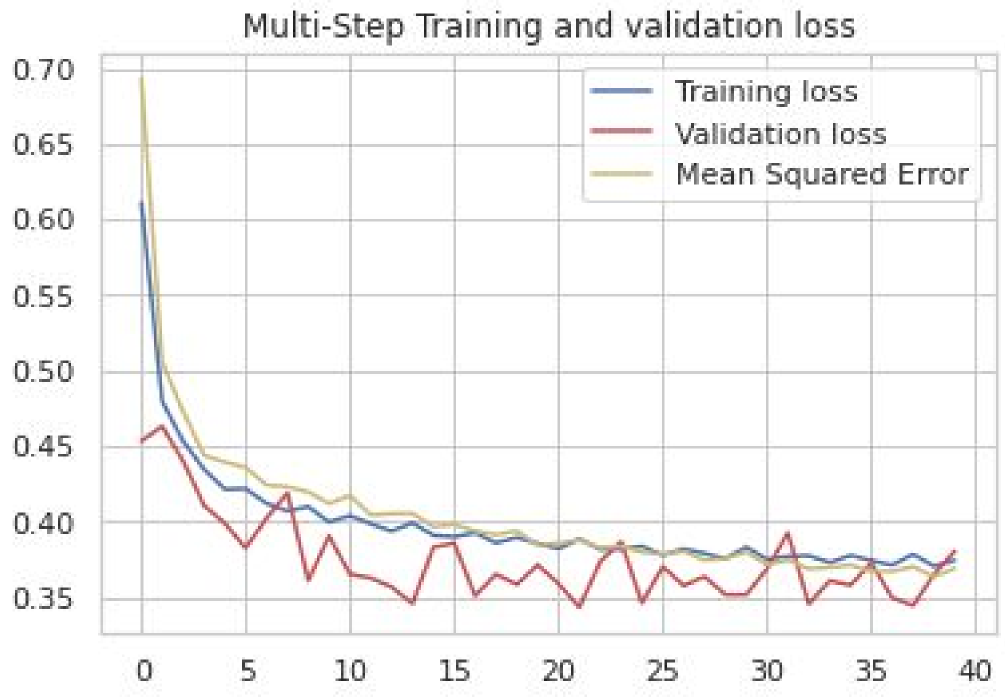


Figure 10 – Error Comparison for RNN Models

For the purposes of comparative evaluation, the models were tested using data from the University College Dublin (UCD) Veterinary building. Weather data collected from Dublin Airport was combined with the building energy usage data, and using the same process applied to the Prim_Class_Jaylin building, predictions were made. The results resemble what was gathered from the original building dataset. Since this dataset was much larger there was an opportunity to train the neural network further without encountering overfitting. The network was trained over 40 epochs with 500 training steps and 125 validation steps. The MAE loss for the training process is shown in Figure 11.

MAE



Epochs

Figure 11 – Training Loss for UCD RNN Model

The ability to train further does not appear to be overly helpful here as validation loss remains largely stagnant with minor increases and decreases with a small range. Evaluating the model's predictions of UCD Data, leads to similar results from the previous building as seen in Figure 12.

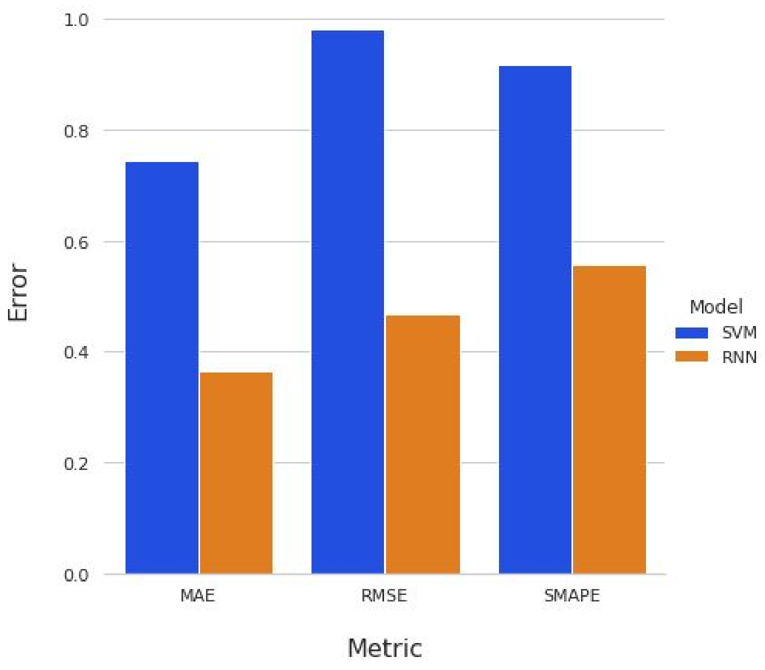


Figure 12 – Error Comparison for UCD predictions

These results show that the SVR trained on the same data perform far worse than the Recurrent Neural Network, with double the RMSE, and significantly higher values in the other metrics.

From my experiments, it is clear that Recurrent Neural Networks are superior to Support Vector Machines in the field of energy usage prediction. The stated aim of this project was to create a tool which could be used to predict energy usage in commercial buildings. From my literature review it became clear that the best deep learning model was a LSTM/RNN approach. This implementation and result set verifies not only that RNNs are capable of sufficiently predicting energy usage based on a multivariate dataset but also that they are empirically more accurate than a statistical counterpart such as a Support Vector Regression.

Acknowledgements

This report was supervised by Damian Dalton, and Fabiano Pallonetto of the UCD Energy Institute.

10. References

1. **Corinne Le Quéré et al.** (2018) Global Carbon Budget 2018, Earth Syst. Sci. Data, 10, 2141–2194, https://doi.org/10.5194/essd-10-2141-2018

2. **Coakley, D., Raftery, P. and Keane, M.** (2014). A review of methods to match building energy simulation models to measured data. *Renewable and Sustainable Energy Reviews*, 37, pp.123-141.

3. **Ahmad, A., Javaid, N., Mateen, A., Awais, M. and Khan, Z.** (2019). Short-Term Load Forecasting in Smart Grids: An Intelligent Modular Approach. *Energies*, 12(1), p.164.

4. **Kuster, C., Rezgui, Y. and Mourshed, M.** (2017). Electrical load forecasting models: A critical systematic review. *Sustainable Cities and Society*, 35, pp.257-270.

5. **Zhang, J., Wei, Y.M., Li, D., Tan, Z., Zhou, J.** (2018). Short term electricity load forecasting using a hybrid model. *Energy*, 158, pp.774-781.

6. **Srivastava, A. K., Pandey, A. S., Singh, D.** (2016). Short-term load forecasting methods: A review. *Emerging Trends in Electrical Electronics & Sustainable Energy Systems (ICETEESES)*, pp.130-138.

7. **Li, H., Cui, L. and Guo, S.** (2014). A Hybrid Short-Term Power Load Forecasting Model Based on the Singular Spectrum Analysis and Autoregressive Model. *Advances in Electrical Engineering*, 2014, pp.1-7.

8. **Prakash, A., Xu, S., Rajagopal, R. and Noh, H.** (2018). Robust Building Energy Load Forecasting Using Physically-Based Kernel Models. *Energies*, 11(4), p.862.

9. **Li, X. and Wen, J.** (2014). Review of building energy modeling for control and operation. *Renewable and Sustainable Energy Reviews*, 37, pp.517-537.

10. **Harish, V. and Kumar, A.** (2016). A review on modeling and simulation of building energy systems. *Renewable and Sustainable Energy Reviews*, 56, pp.1272-1292.

11. **Wei, Y., Zhang, X., Shi, Y., Xia, L., Pan, S., Wu, J., Han, M. and Zhao, X.** (2018). A review of data-driven approaches for prediction and classification of building energy consumption. Renewable and Sustainable Energy Reviews, 82, pp.1027-1047.

12. **Amasyali, K. and El-Gohary, N.** (2018). A review of data-driven building energy consumption prediction studies. Renewable and Sustainable Energy Reviews, 81, pp.1192-1205.

13. **Nguyen, A., Reiter, S. and Rigo, P.** (2014). A review on simulation-based optimization methods applied to building performance analysis. *Applied Energy*, 113, pp.1043-1058.

14. **Zhao, H. and Magoulès, F.** (2012). A review on the prediction of building energy consumption. *Renewable and Sustainable Energy Reviews*, 16(6), pp.3586-3592.

15. **Bedi, J. and Toshniwal, D.** (2019). Deep learning framework to forecast electricity demand. *Applied Energy*, 238, pp.1312-1326.

16. **Miller, Clayton & Meggers, Forrest.** (2017). The Building Data Genome Project: An open, public data set from non-residential building electrical meters. Energy Procedia. 122C. 439-444.

17. **Dagster Development Team.** (2019) Dagster Principles: https://dagster.readthedocs.io/en/0.6.3/sections/learn/principles.html

18. **Dagster Development Team.** (2019) Dagster Reference: https://dagster.readthedocs.io/en/0.6.3/sections/reference/reference.html

19. **Airflow Development Team.** (2019) Airflow Documentation: https://airflow.apache.org/

20. **Vivian, J., Rao, A., Nothaft, F. et al.** (2017) Toil enables reproducible, open source, big biomedical data analyses. Nat Biotechnol 35, 314–316 doi:10.1038/nbt.3772

21. **Toil Development Team.** (2019) Toil Documentation: http://toil.ucsc-cgl.org/

22. **Qing, X., & Niu, Y.** (2018). Hourly day-ahead solar irradiance prediction using weather forecasts by LSTM. Energy, 148, 461-468.

23. **Du, Q., Gu, W., Zhang, L., & Huang, S. L.** (2018, November). Attention-based LSTM-CNNs For Time-series Classification. In Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems (pp. 410-411).