An Exploration of 2020 Vanderbilt Chiller Data

CS 3266: Topics in Big Data

Dylan TerMolen, Christopher Hoogenboom, Karl Schreiner

May 2021

1. Introduction

We hypothesized that the two chillers in the Engineering and Science Building would have different weather dependent efficiencies. We set out to create a predictive model for both chillers in order to recommend load distribution across chillers for max efficiency.

In section 2.1, we outline the acquisition and joining of chiller data with weather data from Vanderbilt University's WeatherSTEM station. In section 2.2 we explore data cleaning and formatting. In section 3 we begin a cursory exploration of the data in order to discern if relationships between weather and energy efficiency are present. This allows us to determine if creating a predictive model with our current data is a feasible plan of action. In section 4 we detail the creation of the neural networks. In section 5 we elucidate our findings and elaborate on its implications.

2. Data Preparation

2.1 Building the Dataset

We were given data in five minute increments for all of 2020 for two air conditioning chillers which both cool the Engineering Science Building. Importantly, these chillers are never run at the same time. This means that it will be feasible to calculate the efficiency of these chillers independently. The dataset consisted of 2 CSV files with 16 columns each with these 16 columns being the same for each chiller. Figure 2.1 illustrates the flow of the data for each chiller and the relationships between all of the columns in the dataset. The table below outlines the critical system parameters that are described in the following analyses .

System Parameter	Description
PowChi	The power that is being consumed by the chiller
TempAmbient	Average air temperature surrounding the chiller
RunChi	Value indicating the ON/OFF status off the chiller
Humidity	Amount of water vapor in the air measured as percent compared to max water vapor possible
Precip	Amount of rainfall measured in inches representing depth
Dew Point	Temperature at which the air must be cooled to become saturated with water vapor measured in Fahrenheit
Pressure	Indication of air pressure at the time measure

	in inches

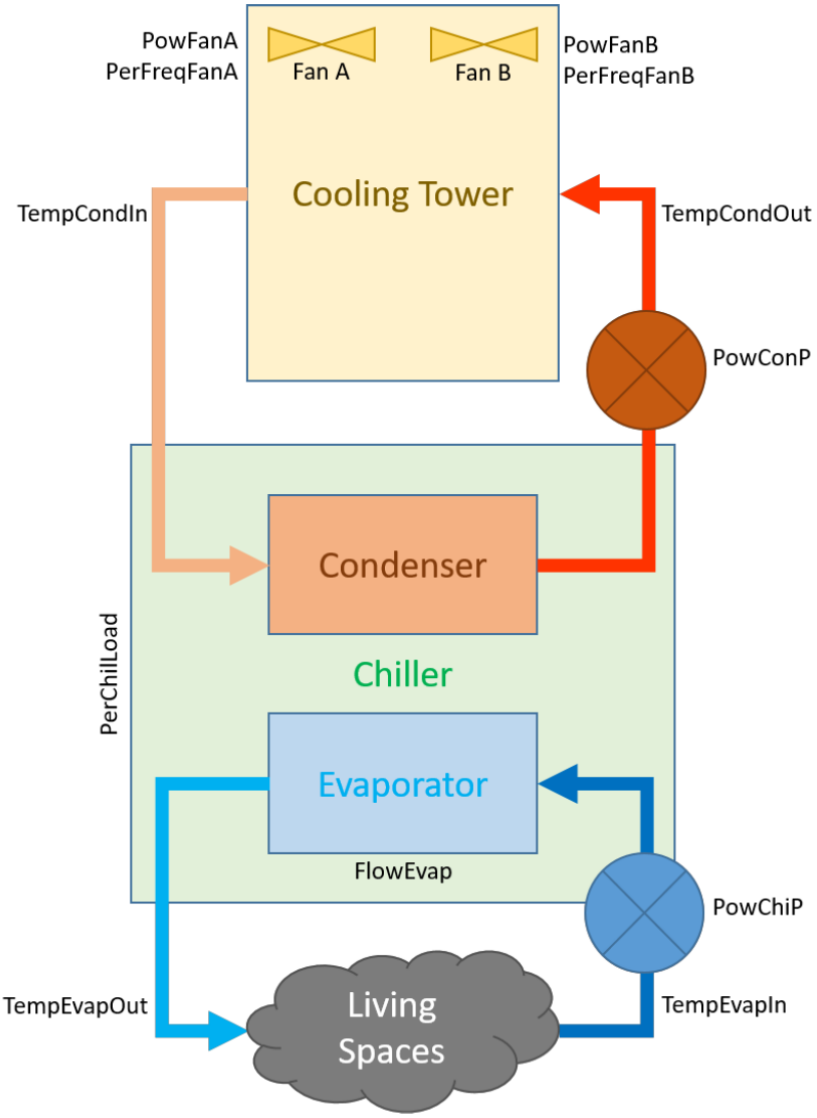

Figure 2.1 Chiller Data Schematic

We wanted to incorporate weather data for the area around the Engineering Science Building that houses the chillers in order to build a predictive model of power consumption by a given chiller. To do that, we performed a scrape of Wunderground's historical weather data for the Vanderbilt University WeatherSTEM station from their website (https://www.wunderground.com/history/daily/us/tn/nashville/KBNA). We used the Selenium WebDriver API for the web scrape and Pandas for the data storage.

Using Pandas we were able to clean up the weather data that was scraped from the internet and join this data with the chiller data. This is done so that the chiller data could be analyzed in relation to the weather across time. In order to clean up the weather data we wanted to ensure that columns such as Temperature were analyzed as numbers as opposed to strings.

Initially, the data was pulled in such a manner that the all of the data was stored as “32 °F”. This is an issue because this storage format is inefficient and makes it difficult for developers to easily perform operations on these columns. This storage format is inefficient because there is a lot of information reuse and the numbers are not being stored as efficiently as possible. We know that all of the data is in Fahrenheit so it does not make sense to store “°F” in each column. This was solved by parsing the columns by splitting them on the space in the string. This allowed us to parse the necessary columns from strings to either floats or integers. The columns were then renamed from Temperature to Temperature(°F) and the documentation of the dataset was changed to reflect this update. Furthermore, this was not an efficient way to store all of the data because of how integers and strings are stored in memory. Most temperature data in Nashville will fall between 0 and 255 degrees which would allow us to store our temperature in a 1 byte location in memory. In contrast, a single character in ASCII takes up 1 byte. This means that our temperature data would most likely occupy 3 bytes in memory as we need to utilize 2 bytes to store the numbers and 1 byte to store the null terminator of the string. Overall, by cleaning up the dataset we can optimize the data analysis process for the computer itself and the end user of the dataset.

Some cleaning of the chiller data was required as well. Tonnage, a measure of how much cooling occurred, is not always reported, and at other times is reported as 0. We removed these entries as we were only interested in efficiency when the chiller was running. We also removed entries when “RunChi” was set to false as this means that the chiller was not on. Occasionally we would get a “PowChi” (power consumption) reading that was almost an order of magnitude higher than any of its even remotely close neighbors. We decided that this was probably due to sensor malfunction and decided to omit all entries where “PowChi” was greater than 400. After this formatting we then created a new data column titled “efficiency ratio” which is calculated by dividing tonnage by energy consumption. This resulted in a ratio of relative efficiency and allowed us to easily make comparisons of efficiency between hot days where the chiller is run near full capacity and cooler days when the chiller is barely being utilized. Again there were a couple of extreme outliers, and in retrospect this was probably because we fixed “PowChi” without fixing and examining “tonnage”. However, this was easily fixed by throwing out all instances where “relative efficiency” was greater than 10.

After cleaning up the individual datasets we needed to join the two datasets together. The biggest problem that we needed to solve regarding the dataset join was that the two timeseries datasets had different time intervals for the data. The chiller data was taken every 5 minutes whereas the weather dataset was pulled randomly and only contained about 1 datapoint for every hours. This meant that we could not simply join the dataset on the timestamp of the datapoints as we would like to. An initial idea to alleviate this issue was to interpolate the weather datapoints for the times that we did not have information for. This was certainly possible to perform, but the randomness in the time intervals between the datapoints made this more difficult to implement. We ultimately decided to utilize the pandas.merge_asof function in order to join the chiller data on the weather datapoint that was closest in time to the chiller datapoint time. The merge_asof function made this incredibly easy to perform as it gave us a plethora of options to choose from in terms of how we wanted to merge the data. We could join the chiller datapoints by choosing the closest weather datapoint with a timestamp less than the chiller timestamp, greater than the chiller timestamp, or whichever timestamp was closest to the chiller timestamp. We decided to

join using the closest weather datapoint timestamp because this was the simplest option and provided us the greatest flexibility going forward. In the future, we may want to use different weather data that provides us with higher precision data. If we decided to do this then the merge would be no different than its current form. The merging and cleaning of the two individual datasets provided the initial foundation for the future analyses performed on the data throughout the project.

3.1 Exploration of the Data

We theorized that external weather factors would have a large impact on chiller efficiency. Ideally, we would discern differing relationships between Chiller1 and Chiller2. This would mean that we could design an artificial neural network for each chiller and recommend a specific chiller be used based on weather conditions in order to maximize energy efficiency. If, on the other hand, we were not able to discern a weather relationship we would need to dramatically redesign our project as there would be no way to create a useful predictive model! Furthermore, if we discerned a relationship between weather and energy efficiency but this relationship was identical between chillers, we would have to redesign our project as we would not be able to recommend one chiller over another for a specific weather pattern as they would both be equally efficient in all weather conditions. We used numpy to find linear regressions and matplotlib to visualize the data. Since this is big data, significant overplotting occurs when plotting all data points. Therefore, we used the Panda's .sample(x) method to randomly poll the data for visualization purposes, though computation was performed on the entire dataset.

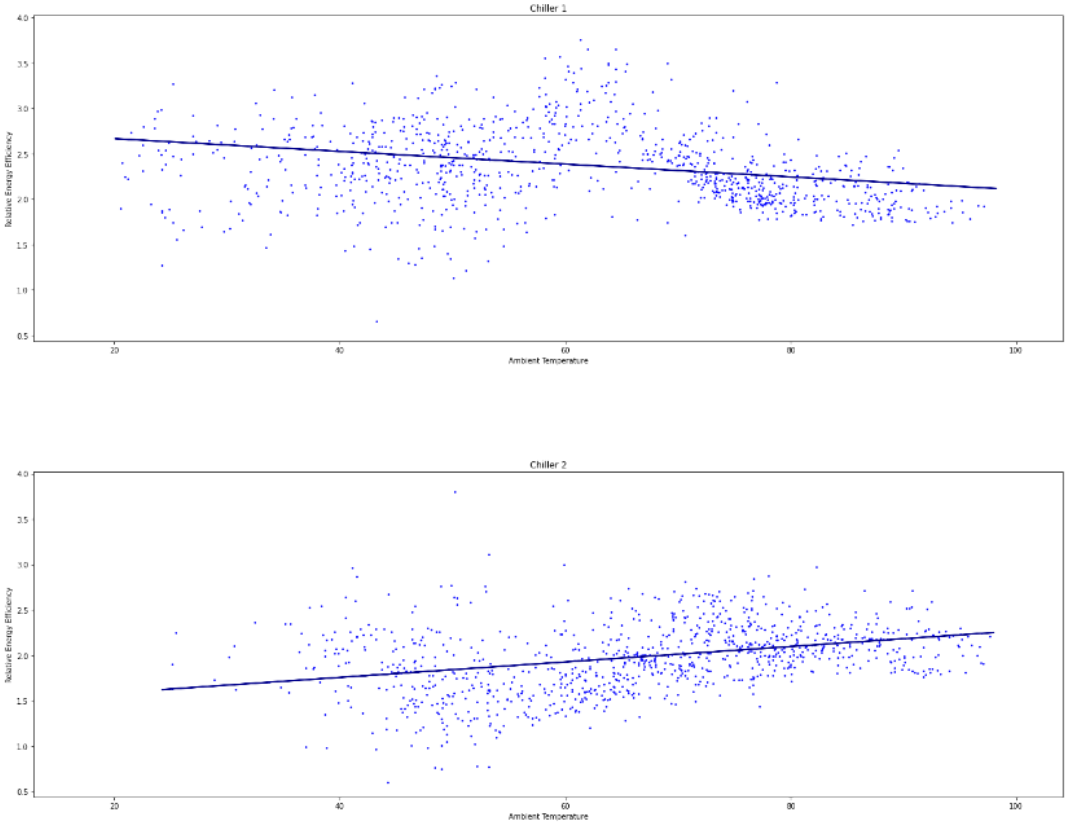

Fig 3.1

Figure 3.1 demonstrates the relationship between outdoor ambient temperature and relative efficiency.. We plotted ambient outdoor temperature on the X axis and Relative Energy Efficiency on the Y axis. This plot provides several insights. There appears to be a negative correlation between temp and efficiency for Chiller 1, but a positive correlation between temperature and energy efficiency. This means that at higher temperatures, Chiller 1 will likely be more efficient however at lower temperatures, Chiller 2 will likely be more efficient. We also should notice that when temperature is low, there is much more variation between points, however, when temperature is high, the points are relatively tightly clustered. This means that ambient temperature alone may not be an ideal indicator for efficiency at low temperatures as there is simply too much variation.

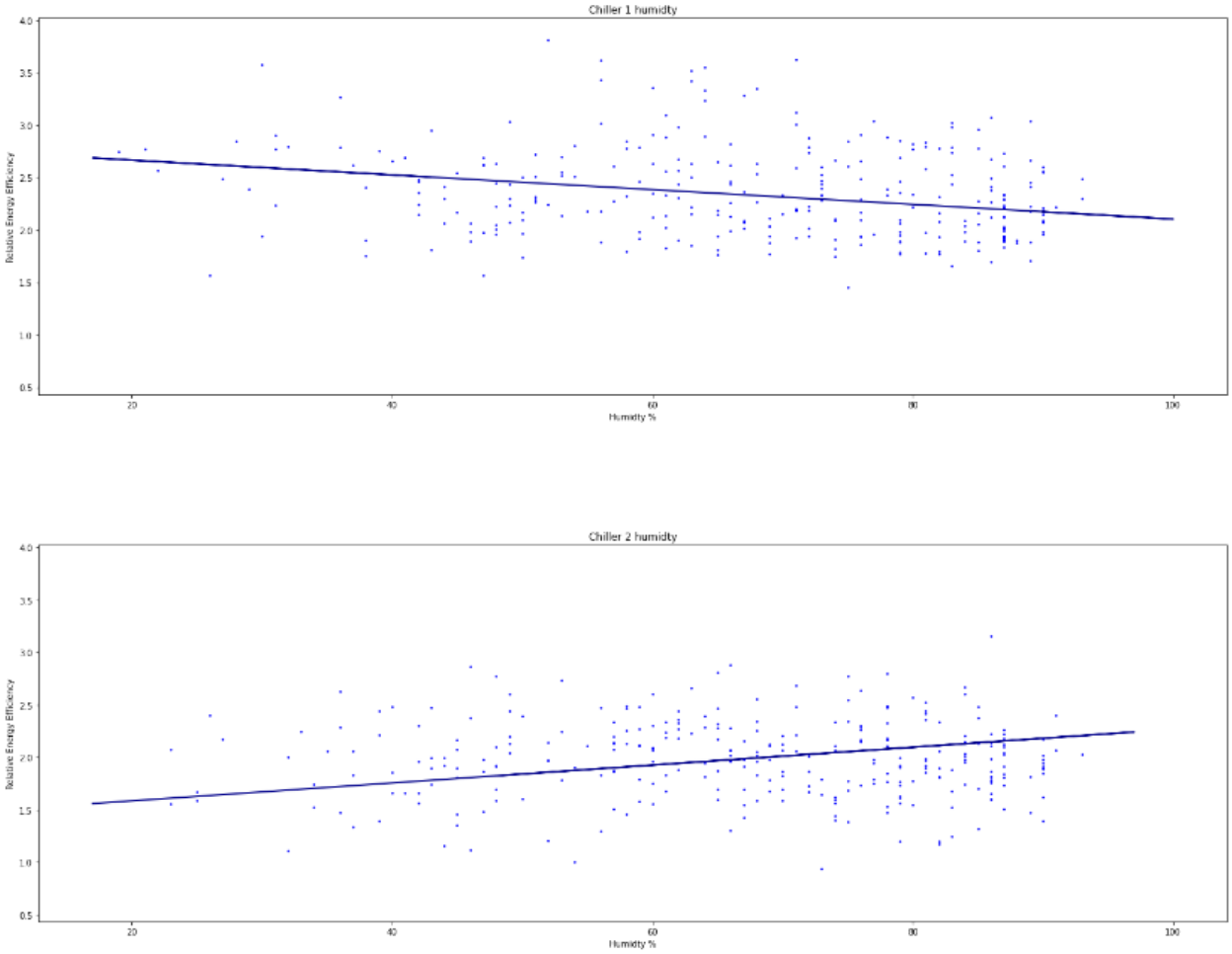

Fig 3.2

Figure 3.2 investigated the relationship between energy consumption and outdoor humidity levels. This graph is hard to read as the external humidity data we collected is binned into integer percentages, which results in hard to read bands of data points. However, we can still run a linear regression on the data and realize that there are trends between humidity and efficiency and importantly, these trends are different between chillers.

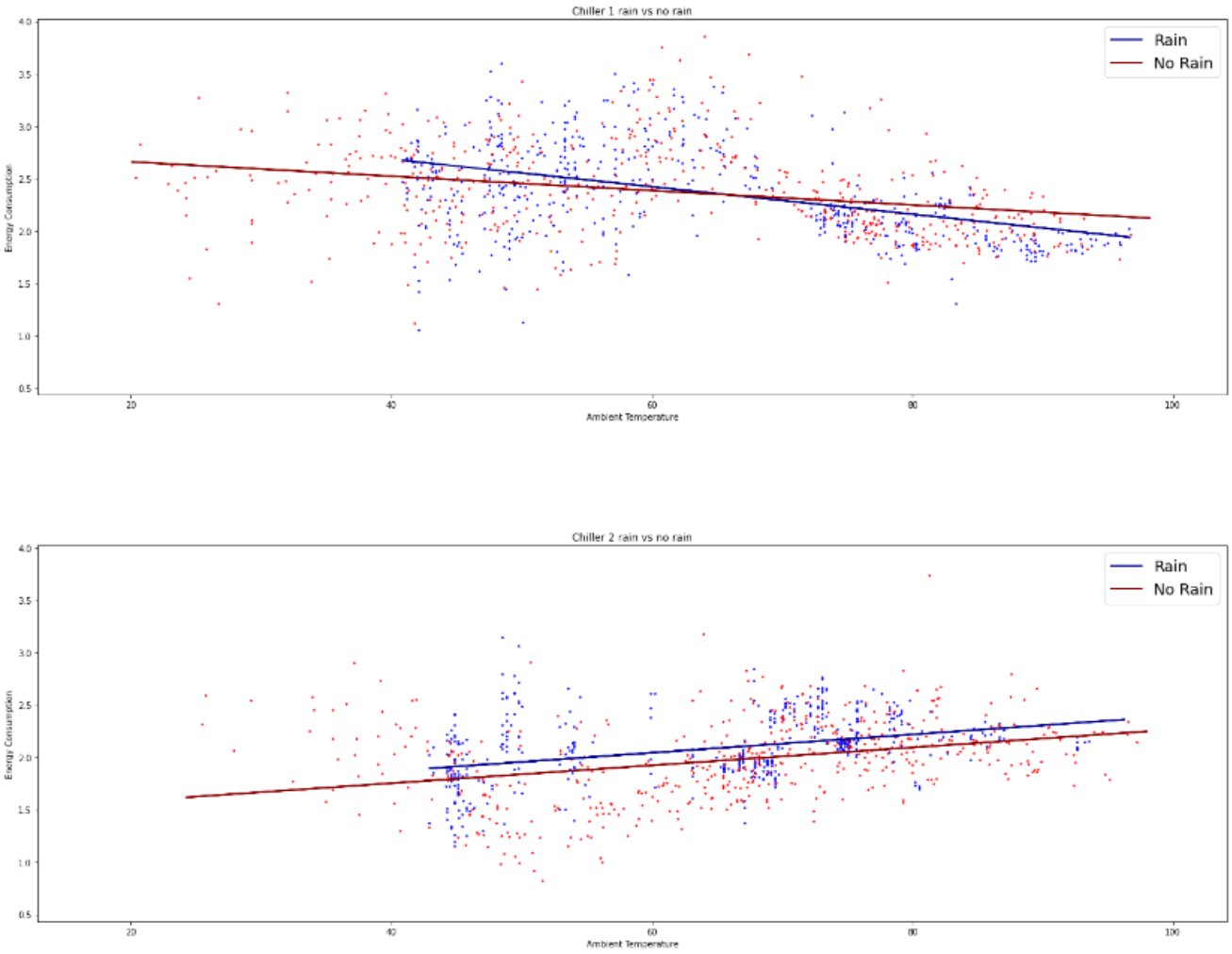

Fig 3.3

Figure 3.3 explores multi-variable relationships. We divided the chiller data based on the categorical variable of precipitation. We then plotted both of these subsets of data onto the same graph, coloring rainy data points blue and dry data points red. Again, we see opposite trends in efficiency between the two chillers. For chiller 2, the slope of both lines is identical and rain always results in a slightly more efficient running. For chiller 1, when it is cold outside, chiller 1

is more efficient when it is raining, but this relationship swaps when it becomes hot outside. Importantly, if we look at the chiller 2 no rain data, we notice that when it is cold most of the data points are over the regression line, but when it is warm most of the data points are actually under the line. This implies a relationship which is not linear further supporting the creation of a neural network to discern exact trends. These initial results strongly supported our plan to build a neural network as we were able to discern relationships between external weather factors and relative humidity but no single weather factor was a completely accurate indicator of said effeciency.

3.2 Feature Set Construction

To determine which data to use as inputs to our predictive models we examined the correlations between each column of our dataset and the target variables, which are power consumption (PowChi) and total cooling achieved by chiller in tons (tonnage). To make the data more interpretable to humans we generated the heatmap in figure 3.2.1 using the heatmapz Python package. The heatmap in figure 3.2.1 is for chiller1, and we have a very similar heatmap for chiller2 that did not need to be included. Darker, larger, bluer squares indicate that the variables are strongly positively correlated and darker, larger, redder squares indicate that the variables are strongly negatively correlated, while weaker colored smaller squares indicate little or no correlation. There is a red box around the columns with our target variables. As you can see, they are similarly correlated to the rest of our dataset and strongly correlated to each other.

Of the variables correlated with our targets, we decided to use TempAmbient (the outside temperature), Humidity(%), Precip.(in), Dew Point(F), and Pressure(in), as these are the variables in our dataset that can be used as predictors of chiller load (as opposed to the data that is a by-product of chiller load). We also used categorical variables not present in the heat map, which are the hour of the day, the day of the week, and the weather conditions for the day. In order for our models to be able to interpret the categorical variables we encoded them as one-hot vectors. Day of the week became a vector length 7, hour of the day became a vector length 24, and weather condition became a vector length 32. We appended each of these vectors to the existing features to finish building the feature set. Because we have 50379 rows of data, our feature set has dimension 50379, 68. A description of the numerical features is in figure 3.2.2.

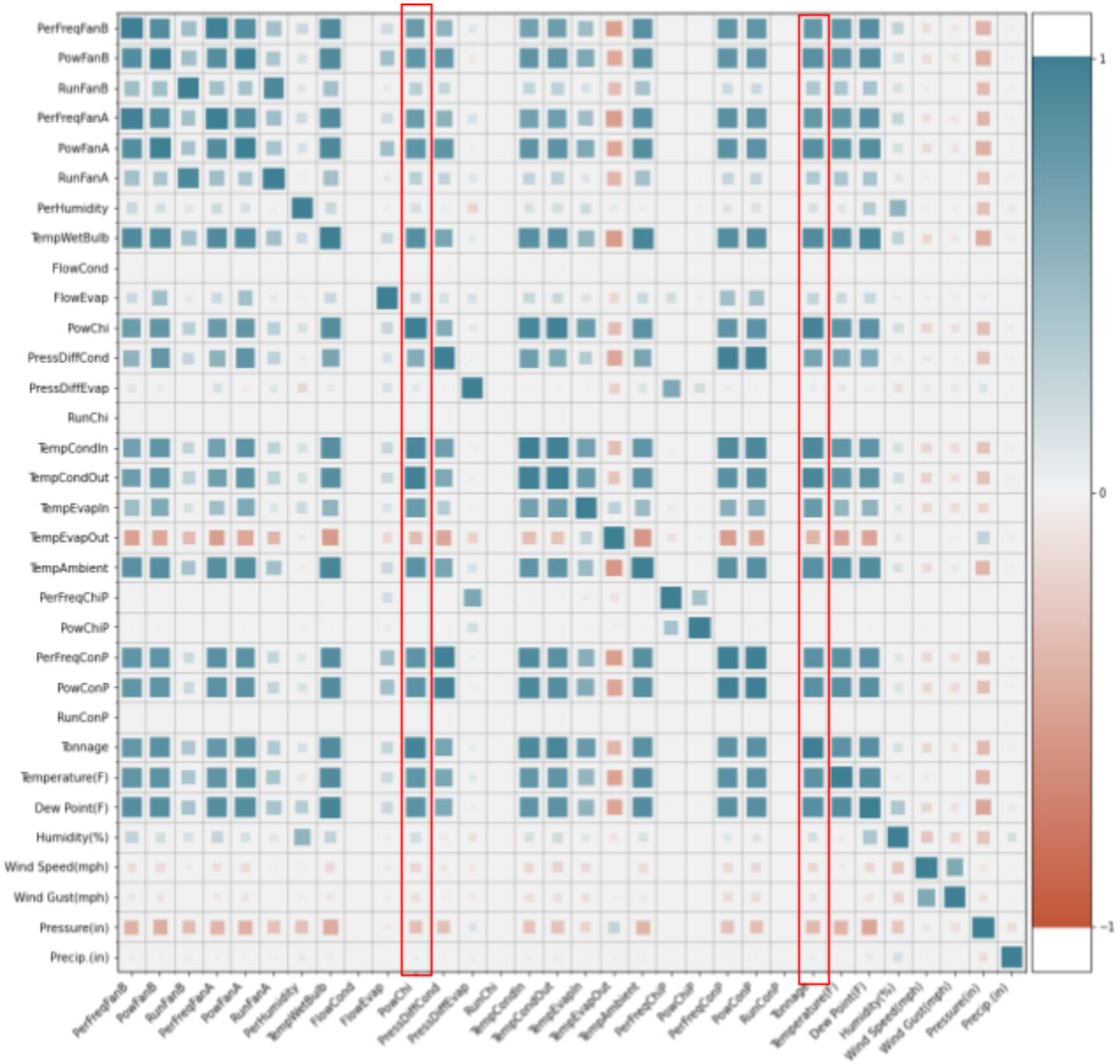

Figure 3.2.1

	TempAmbient	Humidity(%)	Precip.(in)	Dew Point(F)	Pressure(in)
count	50379	50379	50379	50379	50379
mean	60.4581	66.804	0.0060243	47.235	29.4358
std	17.5111	18.01321969	0.04095	18.5566	0.17867
min	20.06987	17	0	2	28.76
25%	46.0631	53	0	31	29.33
50%	59.1444	70	0	47	29.42

75%	75.6851	82	0	66	29.52
max	98.239	100	1.3	76	29.99

Figure 3.2.2

4. Machine Learning Models

We are interested in using different machine learning models to try to predict both power consumption and cooling achieved. In particular, we used artificial neural networks (ANNs) of fully connected linear layers and a specific recurrent neural network called Long Short-Term Memory networks (LSTMs).

4.1 Methods

4.1.1 Artificial neural networks (ANNs)

ANNs consist of a sequence of mathematical equations. Each layer of an ANN has a set sized set of inputs and a set sized set of outputs. A layer consists of parallel nodes each responsible for one of the outputs of the layer. Each node, or neuron, takes in the whole set of inputs after multiplying each one by a weight. Then the weighted inputs are summed together, processed through an activation function, and then the node outputs. The inputs to the first layer are the feature set, and the inputs to each subsequent layer are the outputs of the previous layer. The output of the last layer is the network's prediction of the target data. Figure 4.1.1.2 contains a diagram of an ANN.

In order for an ANN to predict the target data, the weights between nodes need to be properly adjusted. We do this by training the network on example inputs and outputs. To train a network, we process each training example through the network, then use a loss function to produce a metric for the closeness of the networks prediction, and finally use the gradient of the loss function to adjust the weights of the model in a process called gradient descent. If enough training data is processed, then the loss function ideally should converge to a minimum value indicating that the network is making the best predictions possible.

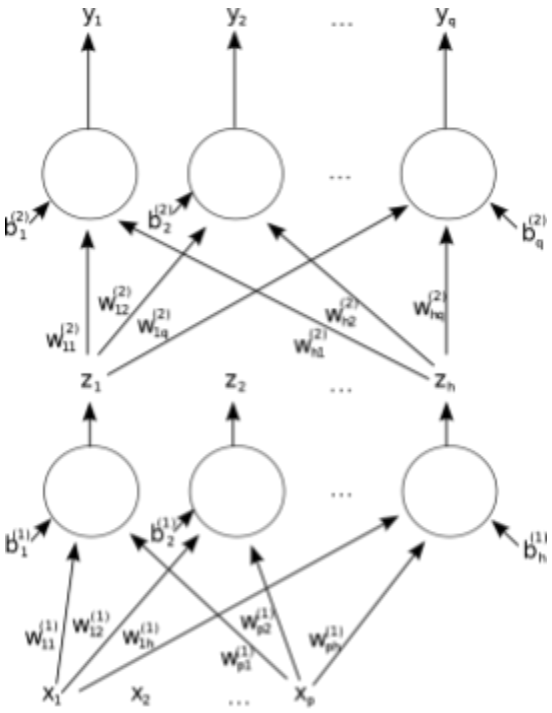

Figure 4.1.1.1

4.1.2 Long Short-Term Memory networks (LSTMs)

Recurrent Neural Networks (RNNs) are similar to ANNs, however RNNs have the ability to maintain memory based on recent inputs. Thus they can be more powerful than ANNs when fed sequential data. Long Short-Term Memory networks (LSTMs) are a type of RNN that can preserve both long term and short term memory, making them wonderfully powerful for taking on problems with sequential data.

LSTMs work by keeping a hidden state and 3 different gates that determine what data is kept and what data is forgotten. The input gate determines which values get to be stored in long term memory, the forget gate determines what data is forgotten from long term memory, and the output gate produces the output and the new hidden state. Figure 4.1.2.1 is a diagram of an LSTM cell.

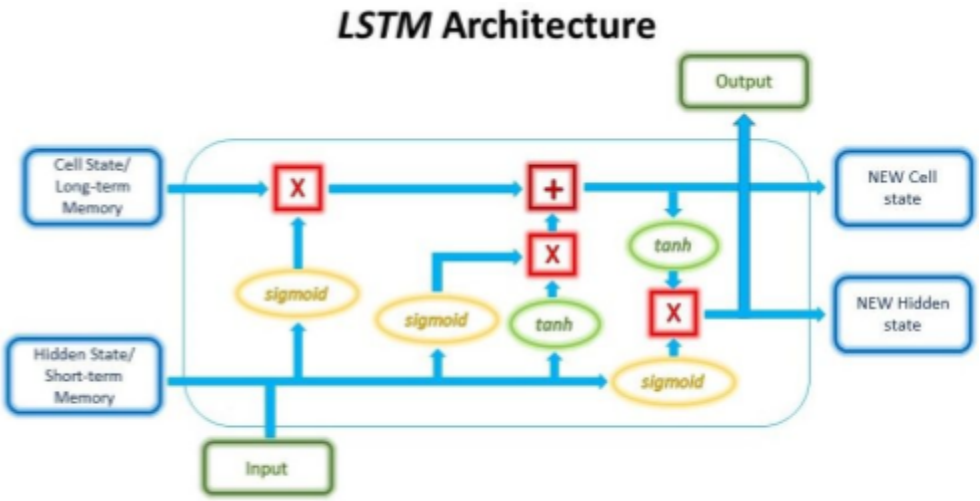

Figure 4.1.2.1

4.2 Models

4.2.1 ANN Modelling

The same structure ANN was used for both chiller1 and chiller2, as well as for both power consumption and total cooling, however each network was trained on its data separately, meaning that we built 4 different ANNs each with the same structure and trained them on each respective input and target set. Each ANN consisted of 4 fully connected linear layers with ReLU activation functions and a dropout layer for regularization to prevent overfitting with a dropout rate of 0.25. The input layer has dimension 68, the first hidden layer has dimension 102, the second hidden layer has dimension 68, and the output layer has dimension 1. Figure 3.2.1.1 is a diagram of the ANN structure.

The ReLU activation function is a commonly used function for a variety of neural network tasks and works very well for regression problems like predicting power consumption and total cooling. It takes the output of a node and forces negative outputs to 0 and passes positive outputs.

The choice of loss function for training is one of the more impactful design choices on the performance of a neural network. We trained our networks using mean absolute loss as the loss function, as it is good for minimizing the impact of extreme outliers on weight adjustment.

We also used an optimizer to try to minimize many common problems of gradient descent like the local minima problem and exploding/vanishing gradient problems. The optimizer we used when training our models is called Adam, which uses momentum, adaptive learning rate, and other techniques to adjust the network weights as accurately as possible.

Neural network performance is normally measured by both the loss and the accuracy of the network when predicting target variables on a test set. We used relative error as our accuracy metric as this is a regression problem. We trained our networks for 200 epochs with a learning rate of 0.001. The validation loss and accuracy of our best networks are shown in figure 3.2.1.2.

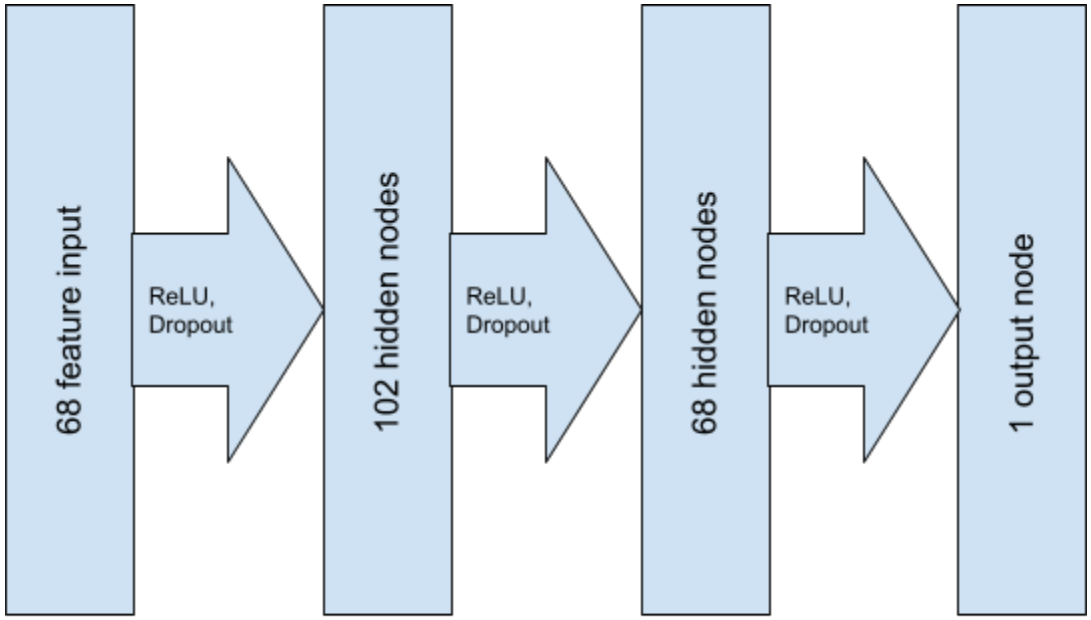

Figure 4.2.1.1

	ANN for chiller1 power consumption	ANN for chiller2 power consumption	ANN for chiller1 total cooling	ANN for chiller2 total cooling
Mean Absolute Error	10.0873	14.279	25.5379	26.8565
Average relative Error	9.8670%	12.243%	12.812%	13.459%

Figure 4.2.1.2

4.2.2 LSTM Modelling

Similar to our approach to modelling using ANNs, we decided to use the same LSTM network structure for each of our 4 desired models. Each LSTM network consisted of an LSTM layer with input size 68, 32 hidden states, and 1 layer, which fed its outputs to a hidden linear layer with 128 nodes, which then fed the 1 node output layer. Figure 4.2.2.1 contains a diagram of the structure of the network.

The network also contained dropout layers for regularization with dropout rate 0.3. We used ReLU again as the activation function between layers, and Adam as the optimizer. For the LSTM we chose to use mean squared error as the loss function.

Once again, we use loss and average relative error as the metric for the goodness of our model, however because we used different loss functions for the ANNs and LSTMs so we cannot compare loss values across the different types of networks. However, the average relative error is a good metric for comparing the LSTMs to the ANNs. The performance of the LSTMs is displayed in figure 4.2.2.2.

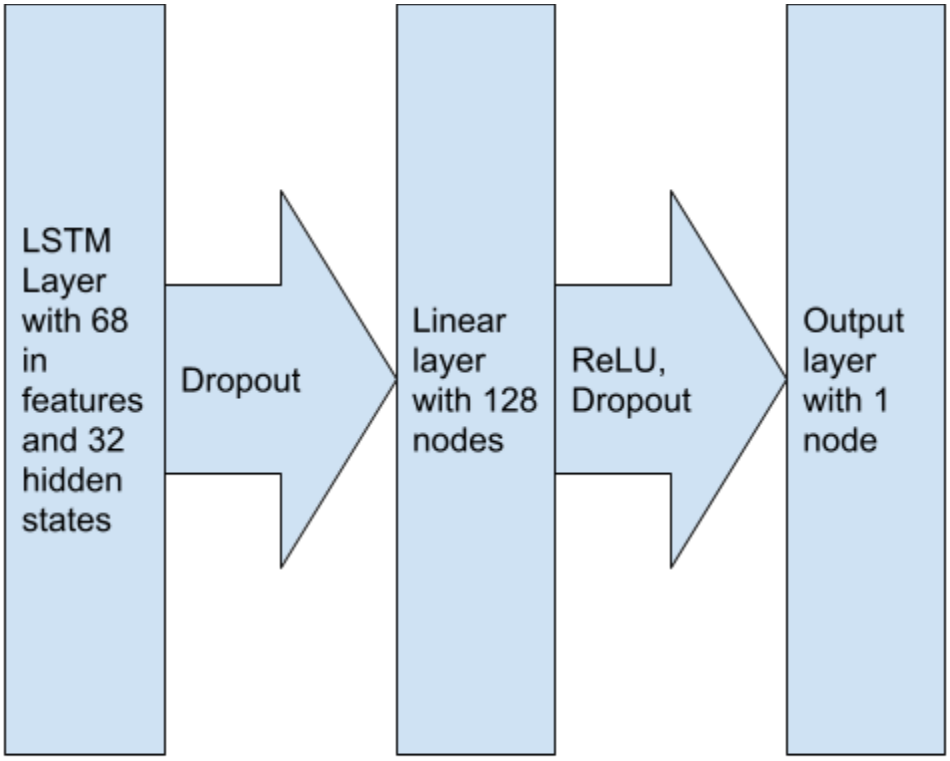

Figure 4.2.2.1

	LSTM for chiller1 power consumption	LSTM for chiller2 power consumption	LSTM for chiller1 total cooling	LSTM for chiller2 total cooling
Mean Squared Error	173.0764	181.2234	422.5720	439.6538
Average Relative Error	11.193%	12.864%	13.798%	13.484%

Figure 4.2.2.2

5. Conclusion

5.1 Method Comparison

Both the ANNs and the LSTMs performed well on this dataset. An average relative error of ~10% is very promising, though with more time to tweak network parameters we could achieve even higher degrees of accuracy. While the performances are comparable, it appears that the ANN performs better on this task than the LSTM. This is surprising, however it might be explained by the fact that Chris, the model builder for this project, has a lot more experience working with ANNs and with LSTMs.

While the accuracy of both methods is comparable, the resources necessary for training is not. The ANNs normally trained in around 5-10 minutes, while the LSTMs took closer to 45-60

minutes to train. As the size of the training data increases it seems that this disparity would only grow. Given the slightly higher accuracy and much more efficient training we conclude that ANNs are the better modelling technique in this case than LSTMs, but recognize that a lot more work can be done on the LSTMs for improvement.

5.2 Implications of a good model

We built these models in order to predict both the power consumption and total cooling performed by a chiller on a given day. This type of model is useful for things like increasing the power efficiency of an individual chiller, distributing chiller load to maximize efficiency, comparing chillers to look for possible improvements, among others. On a larger scale, similar models might be used by power companies to anticipate power load for a given day or other similar tasks.

5.3 Further work

In the future we would like to explore different ways of computing the most accurate relative efficiency. Our present model predicts energy consumption and tonnage separately and then this information is manually converted to find the relative efficiency. We believe this is the most accurate way to predict relative energy efficiency. However, to prove that this is the most efficient method we would also have to train models using our relative efficiency data column and compare their errors.

We could also use other machine learning algorithms to try to build more predictive models, as we have by no means come close to a perfect model yet. Spending more time on fine tuning the LSTM models would also be helpful to increasing accuracy, as we were restricted by the long training time of the LSTM models.

Lastly, it would be very interesting to build a classification neural network for determining when each chiller should be active or inactive depending on the efficiency of each. A classification network of this kind could be immediately implemented in the automated chiller system to maximize efficiency.