LIGHTNING PREDICTION FOR SPACE LAUNCH USING MACHINE LEARNING BASED OFF OF ELECTRIC FIELD MILLS AND LIGHTNING DETECTION AND RANGING DATA

THESIS

Anson Cheng, 2nd Lt, USAF

AFIT-ENS-MS-20-M-138

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-MS-20-M-138

LIGHTNING PREDICTION FOR SPACE LAUNCH USING MACHINE LEARNING BASED OFF OF ELECTRIC FIELD MILLS AND LIGHTNING DETECTION AND RANGING DATA

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Anson Cheng,

2nd Lt, USAF

March 26, 2020

AFIT-ENS-MS-20-M-138

LIGHTNING PREDICTION FOR SPACE LAUNCH USING MACHINE LEARNING BASED OFF OF ELECTRIC FIELD MILLS AND LIGHTNING DETECTION AND RANGING DATA

Anson Cheng,

2nd Lt, USAF

Committee Membership:

Lt Col A. Geyer, Ph.D.
Advisor

Raymond R. Hill, Ph. D.
Reader

Abstract

Kennedy Space Center and Cape Canaveral Air Station, FL, where the Air Force conducts space launches, are in an area of frequent lightning strikes, which is main obstacle in meeting launch goals. The 45th Weather Squadron (45th WS) ensures that any weather safety requirements are met during pre-launch and actual space launch. Using only summer months from three years' worth of lightning detection and ranging (LDAR) and electric field mill (EFM) data from KSC, several feedforward neural networks are constructed. Separate models are built for each EFM and trained by adjusting parameters to forecast lightning 30 minutes out in the surrounding area of each field mill.

Acknowledgments

To my advisor, thank you for the guidance through this research process. I would also like to thank my family and friends for their love and support, as I'm just as surprised as you all that I've made it this far.

Table of Contents

Page

Abstract……..iv
Acknowledgements……v
Table of Contents……vi
List of Figures viii
List of Tables ix
I. Introduction 1
1.1 Problem Statement 1
1.2 Research Questions 1
1.3 Organization of Thesis 2
II. Literature Review 3
2.1 Introduction 3
2.2 Past Research for the 45th WS 3
2.3 Work done on Lightning Prediction 4
2.4 Current Approach 5
2.5 Unbalanced Data 6
2.6 Multinomial Propensity Scores 8
2.7 Principal Components Analysis (PCA) 8
2.8 Artificial Neural Networks 9
2.9 Neural Network Improvements 11
2.10 Design of Experiments 11
2.11 Loss Functions 12
2.12 R Programming 13
2.13 Conclusion 14
III. Methoodology 15
3.1 Introduction 15
3.2 Data Processing and Cleaning 15
3.3 Data Classification and Balancing 19

3.4 Determining Model Adequacy 20
3.5 Hyperparameter Tuning 21
3.6 Data Dimensionality Reduction 22
3.7 Experimental Design 22
3.8 Conclusion 24
IV. Discussion 25
4.1 Introduction 25
4.2 Initial Results 25
4.3 Parameter Analysis 32
4.4 Model Adequacy 34
4.5 Conclusion 37
V. Conclusion and Moving Forward 39
5.1 Introduction 39
5.2 Overview of Results 39
5.3 Future Work 39
5.4 Conclusion 41
Appendix 42
Bibliography 50

LIST OF FIGURES

Page

1. Neural Network Performance over epochs for Design abc for EFM 18…….…… 27
2. Neural Network Performance over epochs for Design 1 for EFM 34 27
3. Neural Network Performance over epochs for Design ac for EFM 18.....28
4. Neural Network Performance over epochs for Design 1 for EFM 22 29
5. Neural Network Performance over epochs for Design abc for EFM 15.....30
6. Neural Network Performance over epochs for Design abc for EFM 30.....30
7. Parameter Estimates of Neural Network Performance on Loss.....33
8. Parameter Estimates of Neural Network Performance on Ratio 34
9. Quantile Plot for Loss Prediction.....35
10. Quantile Plot for Ratio Prediction.....35
11. Predicted Loss vs Studentized Residual Plot for Loss Prediction 36
12. Predicted Loss vs Studentized Residual Plot for Ratio Prediction 37

LIST OF TABLES

Page

1. Neural Network Parameters…….. 23
2. Fractional Factorial Design for Neural Networks.....24
3. Initial Results from Neural Networks 25
4. Best and Worst Averages by EFM 31
5. Best and Worst Averages by Design 31

LIGHTNING PREDICITON FOR SPACE LAUNCH USING MACHINE LEARNING BASED OFF OF ELECTRIC FIELD MILLS AND LIGHTNING DETECTION AND RANGING DATA

I. Introduction

The 45th Weather Squadron assists Cape Canaveral Air Force Station (CCAFS), National Aeronautics and Space Administration (NASA), Kennedy Space Center (KSC), and Patrick Air Force Base in carrying out space missions. The United States space program and its ability to stay on launch schedule is highly sensitive to meeting the weather criteria for launch [4]. Currently, lightning is the leading cause of scrubs and delays in space launches. Thousands of people and billions of dollars in funding are poured into preparing the rockets, payloads, and launchpads for space launch so there are both safety and financial concerns when lightning is present in the launch area [3].

1.1 Problem Statement

The 45th Weather Squadron seeks to continue to build upon previous research on lightning prediction around CCAFS. Steps to improve lightning prediction will lead to less scrubbed and delayed launches.

1.2 Research Questions

To address the problem, two questions are examined:

1. Can a feedforward neural network per electric field mill handle the noise in the data and better predict lightning onset?
2. How well can a neural network predict lightning onset?

1.3 Organization of Thesis

The following chapter is the literature view that examines past studies on lightning prediction as well as methods that can be used in this study. Chapter 3 provides methodology, which explains the data preparation required to feed the models and what type of models were picked. Then, the findings from the analysis are presented. Finally, lessons learned, and potential future work are discussed in Chapter 5.

II. Literature Review

2.1 Introduction

This chapter provides a brief overview of the literature related to the application of multivariate techniques in lightning prediction to include a review of the research already conducted for the 45th Weather Squadron regarding lightning prediction. Then, the current model used at CCAFS is examined. Finally, the concept of artificial neural networks (ANNs) and the specifics to possible avenues in this research is discussed.

2.2 Past Research for the 45th WS

There is past work on ANNs to make lightning predictions. Hill [1] created a convolutional recurrent neural network (CRNN) as the electric field mill (EFM) data that was provided was time-series and had locations on it. While the results from the Hill model had a lower probability of a false lightning detection than other similar studies, many parameters that could have been included in the neural network were left out.

The approach Hill [1] took is good at predicting for a specific window but the issue is that computationally, it took a long time. The neural networks did improve on the current lightning prediction rates but with the full EFM data from all 31 stations being pulled instead in hour increments, this approach is not viable given how long it will take to train and run the neural nets.

Speranza [2] implemented a Long-Short Term Memory neural network on the same data in Hill [1]. However, a different approach is implemented compared to this model in identifying which EFM data is used when lightning is detected. Lightning detection is provided in the LDAR dataset, which provides the location and time of a lightning strike

from the detection site. Speranza [2] created an imaginary box, 41.7 kilometers wide and 87 kilometers long around the Kennedy Space Center. Only lightning detected in the box was pulled from the LDAR dataset. Instead of pulling from a box as the criteria for including lightning detections, each EFM is checked to see if there was lightning within its 5 nautical mile radius. Instead of looking at all the EFMs at the time of a strike in a box, each EFM is examined individually. Speranza [2] also examined METAR data, which encompasses weather information such as surface wind, visibility, time, and temperature, which is no longer being used [35].

2.3 Work done on Lightning Prediction

Many studies examining EFM readings are from regions that experience a high rate of lightning such as India, Colombia, Brazil, and South Korea. The studies of interest were the ones that applied multivariate techniques to forecast lightning.

The most recent of these studies was from South Korea in which an ANN was developed to improve on the current lightning warning system. Like Florida lightning data, the most lightning strikes that the study recorded was from the months of May to August. September is of interest in this study, so September is also included. A general lightning threshold was determined to be between 20 kV/m and 50kV/m, which was where the readings ranged during the summer months [10]. From there, the change rate of the electric field within 5 seconds was recorded, along with the temperature, and humidity 2 minutes before every lightning strike that occurred.

An ANN was then built taking in the change rate of the electric field, temperature, and humidity. Reading in over 6 months of data, the eventual prediction accuracy turned

out to be 93.9% [10]. This study shows that EFM readings are viable in prediction lightning strikes.

In a recent study, Mostajabi et al. [35] explore predicting lightning 10-30 minutes ahead of time inside a 30-kilometer radius. A four-parameter model was created, based on air pressure, air temperature, relative humidity, and wind speed. Then, classification-based machine learning techniques such as decision trees were applied to predict lightning onset [35]. While the study is recent, the results are not comparable to Speranza's research. The data was collected in the mountains and classification techniques were used. This is vastly different from the conditions the 45th WS faces. The 45th WS is concerned about the 5-mile radii around all 30+ EFM's around Cape Canaveral. Also, different variables affect lightning onset as Cape Canaveral is sea level and by the sea, compared to Switzerland, which is high elevation and in the mountains.

2.4 Current Approach

The current method of issuing a lightning warning in Cape Canaveral is utilizing lightning circles defined around the area. A warning is issued when lightning is imminent or occurring in one of the warning circles defined around Cape Canaveral. The new circles of detection, which were converted from 13 to 10 in 2014 have a radii of either 5 nautical miles or 6 nautical miles. The reduction of circles helps combat the amount of overlap in the previous system. However, an issue with the new system is a small increase in over-warning is offset by the benefits [5].

According the William Roeder, a meteorologist in the 45th Weather Squadron, lightning cessation has occurred when there has been no lightning within the last 15

minutes. Recognizing redevelopment of a storm then becomes the decision of a trained forecaster [36].

2.5 Unbalanced Data

The dataset that in this research consists of two variables per EFM. One variable is the voltage reading during a minute in time and the other variable is the minutes until a lightning strike occurs. As lightning is constantly occurring during the May to September window, finding a minute in time where no lightning is occurring within 5 nautical miles of an EFM is a rare event. This leads to an unbalanced data set, where there are unequal instances per class: 0 for a lightning strike and a number greater than 0 lightning onset. If an unbalanced data set is used to train a machine learning algorithm, the unbalanced data set will bias the algorithm to predict for the more prevalent class, which is lightning in this case [19].

There are two popular methods applied to unbalanced datasets. One is to under-sample portions of the data to reduce the size of the abundant class. Another is oversampling to balance the under-represented class. Buda [20] performed a study to see how oversampling and under-sampling affects class imbalance. Using convolutional neural networks and with a receiver operation characteristic curve as the evaluation metric, the study found that oversampling seemed to be the dominant method to balancing a data set compared to under-sampling. Oversampling was able to eliminate the balance, but under-sampling only addresses the class imbalance to an extent. Zhou and Liu [21] also examined oversampling and under-sampling to deal with class imbalance. They found that while class

balancing techniques worked well for two-class data sets, they were ineffective and sometimes caused a negative effect on multiclass data sets.

Two often used methods of oversampling are Bootstrap Random Over-Sampling Examples (ROSE) and Synthetic Minority Over-sampling Technique (SMOTE). ROSE handles binary classification problems that have imbalanced classes. The minority class is balanced against the majority class by artificially generated balanced samples according to a smoothed bootstrap approach as suggested by Menardi and Torelli [22]. The ROSE process has three steps. The first is that data of the majority class is resampled using a bootstrap resampling technique to remove observations of the majority class to 50%. Then, the same method is used to resample the minority class data until the minority class also has an overall ratio of 50%. Afterwards, new synthetic data similar, but different from the observations are generated. The three steps are repeated until a new synthetic training sample of about equal size to the original data set is created, with about an equal representation from the two classes [23].

The second method, SMOTE, is a similar approach. SMOTE generates synthetic data based on the feature space. Chawla [24] and his team perform an experiment comparing SMOTE data performance against replicated over-sampling data. They found that the minority class had a higher prediction rate with SMOTE when using decision trees. This technique oversamples the minority class and generates synthetic examples that fall along the line segments joining the k minority class nearest neighbors. Afterwards, the oversampled minority class and the undersampled majority class data are combined.

2.6 Multinomial Propensity Scores

Propensity scores, which have become popular in medical studies, are a way to achieve exchangeability between different groups in an observational study (usually treatment versus no treatment). The way propensity scores work is that a set of covariates from a dataset are first selected. Next, a logistic regression is performed, and each observation/subject is given a propensity score. Then the observations/subjects are matched based on whether they received a treatment or not and their respective propensity score. The balance of the covariates is checked once the treatment and non-treatment groups are matched. The effect of the treatment is calculated [25]. This approach handles unbalanced response datasets with more than two response classes.

2.7 Principal Components Analysis

Two significant issues when working with larger data sets are the curse of dimensionality and highly correlated variables. The more variables a dataset has, the harder it can be to draw informative conclusions. A way to combat high dimensionality or highly correlated variables in data sets is to reduce the dimension using principal component analysis (PCA). PCA creates principal components in terms of a new set of uncorrelated variables. These fewer components sufficiently explain the variation in the original dataset. When PCA is performed, a covariance matrix, $\sum$, between the principal components is created. Each eigenvalue of $\sum$ is proportional to the portion of the variance associated with that eigenvector. The sum of all the eigenvalues is the multivariate variability. Usually, the first few principal components will explain most of the variation in the original data set [32].

While PCA can simplify high-dimensional and correlated data while retaining patterns in the data set, it is up to the user to determine the number of principal components and corresponding amount of variation explained to use. The most common method is picking enough components to explain around 70 to 90 percent of the variation. The issue with this is the number of components the user decides is highly subjective and may heavily affect the results. Another way is to examine the eigenvalues of each principal component and exclude components where the eigenvalue is less than the average. As the average eigenvalue is equal to the average variance of the original variables, this method keeps the components that have more variance than the average for the observed variables [32].

2.8 Artificial Neural Networks

An ANN consists of an input layer of nodes, one or more hidden layers of nodes and then a final layer of output nodes. These nodes try to reflect the connections neurons have in a human brain with each layer acting as a black box, taking data from the input layers and regressing them in each node. Every layer has a different weight based on its relative importance [6]. Apart from the layers, input data, a loss function and the optimizer are necessary to train and eventually obtain results from a neural network. The loss function outputs a loss value, which is just a measurement of how well a neural network is predicting and the optimizer, which aims to minimize the loss value, then takes the loss value and updates the weights in the neural network [7].

Chollet [7] discusses how to process data for the neural networks. The data itself is noisy but deep neural networks can generalize after being training on noisy data and can accommodate a wide range of noise levels. Although logistic regression is a possible

method for binary data, an ANN's ability to deal with large and noisy datasets makes it the better choice [13].

The most basic type of neural network is a multi-layer, feed-forward neural network. Feed-forward means the information in the neural network only flows in one direction, from the input layer to the input layers. This type of neural network is trained using a back-propagation learning algorithm. Back-propagation fine tunes the weights to minimize the error rates in a previous epoch. To predict a continuous value using a feed-forward neural network, there is one node for the output layer whereas a classification problem will have as many output nodes as there are classes to predict [31].

Recurrent neural networks, specifically Long Short-Term Memory (LSTM) neural networks, perform well with time-series data [8]. Unlike traditional neural networks, recurrent networks can use its reasoning from previous events to inform later events. Recurrent Neural Networks (RNNs) have a loop that allows data to persist [7]. The issue with RNNs is that the gradient loss function decays exponentially. This leads to RNNs having an increasingly harder time predicting longer temporal trends. LSTMs do not have this issue. LSTM neural networks can remember information for longer periods of time than regular RNNs, helping overcome long-term dependency [9]. Unlike traditional neural networks, every LSTM layer should also have a dropout layer to help prevent overfitting. Dropout improves generalization in the neural nets by randomly ignoring some neurons when the nets are being trained, thereby reducing the affect certain weights will have on the results [14].

2.9 Neural Network Improvements

There are several ways to improve neural network performance. One method is batch normalization. Batch normalization normalizes the output of a layer by taking the output, subtracting by the batch mean, and then dividing by the batch standard deviation. Implementing batch normalization allows each layer to learn more independently compared to without batch normalization [33].

After a neural network is done training, the network weights can sometimes be large, indicating the possibility of overfitting. A way to combat overfitting in neural networks is to include weight regularization among the layers. This technique keeps the weights in each layer small, reducing overfitting in the model [34].

Dropout is another technique to combat overfitting and can improve neural network performance. When dropout is applied to a layer, output features are randomly dropped out at the layer [7]. Doing so will reduce the chances of overfitting from the neural network.

2.10 Design of Experiments

There is no specific way to build an ANN given the number of hyperparameters that must be chosen and tuned. A hyperparameter is a parameter that is set before the learning process [15]. In a study to predict the thickness of the chromium layer in a hard chromium plating process, Lasheras [16] uses the design of experiments theory to optimize the ANN built. Doing so allowed for optimal experiments that could maximize how well the different models performed and minimized the number of experiments necessary to train and validate the models.

Design of experiments was applied in this study to tune the hyperparameters in the ANNs created. Neural networks have hyperparameters such as layers, neurons, dropout, weight regularization, and epochs. Picking each factor with a high and low, yields a 2^k factorial design to sample the test space [17].

With each neural network design, there are about 30 neural networks built and trained as each EFM is trained separately. This means a 2^2 full factorial design would require 120 or more neural networks. Given that there are many parameters in a neural network, the factorial design is much larger, creating an exponentially increasing number of neural networks required.

Fractional factorial designs can reduce the number of factor combinations required while still identifying factors that have a large effect. These designs work based on three ideas. The first is the sparsity of effects principle, where main effects and lower ordered interaction variables dominate a system. Second is the projection property, where fractional factorial designs can be projected into stronger designs in the subset of significant factors. Finally, sequential experimentation, where it is possible to combine fractional factorials to construct sequentially larger designs to estimate more effects. The main idea of fractional factorials is to take a subset of the full factorial design, eliminating possibly redundant runs in a full factorial design [17].

2.11 Loss Functions

For algorithms that perform classification, the results are presented is a confusion matrix. The way a neural network produces its classifications is using a classification loss function. The most common function for classification is the cross-entropy loss/negative

log likelihood function. The log portion of the function is what classifies a label, with the loss function penalizing predictions that are confident but misclassify [26].

Since the dataset used is a time series, meaning that RNN's are used, there is the option for a neural network to predict a continuous value. The two main loss functions applied to a continuous prediction is the mean square error and the mean absolute error. The benefit of using mean square error as a loss function is that it penalizes outliers heavily compared to mean absolute error, which is more robust to outliers since it does not square the errors [27].

In a study done to forecast floods using time series data, a LSTM neural network was built with a mean square error loss function and Nash-Sutcliffe model efficiency coefficient as a measurement of model performance. The Nash-Sutcliffe efficiency is measured by determining the relative magnitude of residual variance to the measured data variance [28][29]. However, there have also been studies that have not required the Nash-Sutcliffe efficiency to determine model performance. In a study to forecast electric loads in smart grids, LSTM model parameters were tuned based on mean absolute error and root mean square error to produce better predictions [30].

2.12 R Programming

R is used to build the data set, neural net and perform the analysis. This is an open source programming language used for statistical computing and graphics. The number of packages available to perform all the tasks needed in this make it an ideal choice for this research.

2.13 Conclusion

There are previous several efforts into understanding lightning storms around the world with research continuing to this day. Hill [1] and Speranza [2] in particular, focused on understanding and predicting lightning in Cape Canaveral. Considering the lessons learned from the past and the analytical work that can be applied to this problem assisted with finding a new way approach to predicting lightning.

III. Methodology

3.1 Introduction

This chapter discusses analytical methodology. First, the process to obtain and clean the data is explained. Then dataset construction is discussed. Finally, the neural network structure and test design structures are examined.

3.2 Data Processing and Cleaning

The same data text files used in Speranza [2] were used here. Consequently, similar code was used to extract to data files into R. In Speranza's code, the EFM data read was converted to hourly averages. In this case, every EFM observation, down to the millisecond, was converted to a Rdata file. The code for this is in Appendix A. The EFM data uses day of the year instead of dates, unlike the LDAR data which is in epoch time. Both data sets have dates and times as a string instead of actual dates and times. Following the as.Date walkthrough from UC Berkley, [11] the EFM dates formatted as day of the year and string is converted to an actual date. Once day of the year is converted to a date, the lubridate package in R calculates the epoch time [12].

Using the tidyverse package in R, average voltage readings per minute in each EFM were calculated [18]. There were minutes of data missing that were imputed by taking the average of the most recent reading before and after the gaps. The dataset then consisted of time and date of the reading, the voltage reading itself, and the EFM station that had the reading. Since epoch time is easier numerically when it comes to calculating differences in time, all the time/dates in the EFM data were converted to the number of minutes that have passed between 1 January 2013 and the observation. This allows for calculating minute

differentials easily. This is similar to how the LDAR time data was originally recorded as epoch time, which is the number of seconds that had passed since 1 January 1970. In this case, a similar "epoch time" was calculated but downsized to reduce the numerical calculations

The LDAR data was also read in from a text file to a Rdata file and the code is in Appendix B. The data consisted of the time a lightning strike occurred, the associated "epoch time", number of meters from the center of KSC in terms of X (East/West) and Y (North/South), as well as Z, which was altitude. The altitude portion of the data was dropped due to the assumption that curvature of the earth was negligible. Epoch time was used as the time component as a numerical value was easier to work with and sort. From there, a conversion on the X and Y values were performed to find the actual longitude and latitude of where all lightning strikes occurred.

After the EFM and LDAR dataset were converted to accurate coordinates and "epoch time", a new dataset was created to identify lightning strikes that occurred within the range of each EFM. Using the LDAR data and going through each EFM and the nautical miles from the EFM to each strike in the LDAR dataset was calculated. Any value greater than 5 nautical miles was dropped. This created a dataset per field mill that contained a column of epoch time of strike and another that was nautical miles from EFM that strike occurred. As Roeder [36] stated, 15-minute gaps between lightning strikes are of interest. Therefore, a new column was created providing the difference of a current row's epoch time and the previous row's epoch time. This dataset gave the number of minutes that had passed since the last strike after a new strike occurs. From there, the dataset was filtered

down more, where any observations with differences less than 15 minutes were removed. The remaining dataset per EFM was the epoch time of the beginning of a "storm" and the minutes since the last storm. A storm is defined as starting if lightning strikes after there has not been a lightning strike for more than 15 minutes. The last strike is defined as a lightning strike occurring, then no other strike occurs for 15 minutes.

With the start and stop times of storms that each field mill encountered, the lightning onset was then calculated. Each minute and 30 minutes into the future of the "epoch time" was examined. Then, if there was a storm occurring, lightning onset was calculated. If there was a storm occurring 30 minutes in the future, the time is set to 0, if not, the time until the start of the next storm in minutes is recorded. This is done to see whether 30 minutes from a given time has a storm occurred. For seconds with no recorded voltage readings, the average of the observation the second before and after were taken. Since the EFM data was read in from text files, and processed in five-day intervals, there was usually some data missing at the beginning and end. This missing data was imputed by taking the most recent voltage readings and replicating the most recent observation until there was 7200 minutes worth of data (5 days in minutes). After this, a new data set was created with columns for "epoch time", EFM readings and lightning onset. The data set shows at any given epoch time, the current voltage readings at all EFM's and the time, 30 minutes from the epoch time, until a storm is within 5 nautical miles of each field mill with a 0 if there is an ongoing storm. Some EFM's were not working at all so the eventual data set consisted of only 26 field mills out of the 34 installed at Cape Canaveral. With one

variable for “epoch time” (minutes since 1 January 2013), EFM voltage reading, and time until a storm per reading, there are 52 variables in the dataset.

There were missing sections in the data where no readings were obtained by the field mills. These were generally minimal and were imputed by taking the averages between the voltage readings before the gap began and after the gap ended. The missing voltage readings caused some onset values to not count down to 0 correctly. Identifying gaps and imputing them made sure any onset values counted down to 0.

The datasets were then combined. All the data in the five-day intervals were read in, meaning that there were data sets that had more EFM readings than others. Some of these field mills were only up for a few days compared to the majority that were reading data constantly. Any EFM only having a few days worth of data was removed. The remaining field mill voltage readings and lightning onset values per field mills that had data available through all the five-day intervals. Once the common variables were identified, the columns were aligned in chronological order using rbind from the plyr package in R. This creates the dataset containing all 26 field mill voltage readings and lightning onset readings per field mill that were contained throughout all the five-day interval data sets. A total of 84 five-day intervals were read in, from the summer months of May through September for the years 2013, 2014, and 2015. This data set consisted of 604,800 rows of one-minute observations and 52 variables. The code to build the dataset is included in Appendix C.

3.3 Data Classification and Balancing

Onset variables are of interest, but the final dataset was unbalanced. About 97% of the onset data were 0's, indicating that a storm is occurring 30 minutes from that point in time. A binary classification variable indicating the presence of a storm is insufficient given the distribution of available onset times in the data set. The onset values range from 1 minute to over 2 hours, with different frequencies. To examine the unbalance in different classes, whichever onset variable was the predictor variable was classified. Four classes, (A, B, C, D), were created, based on the time frame of interest to the 45th WS. Any value of 0 was its own class (A), any value from 1-15 minutes was another class (B), 16-30 the third class (C), and anything greater than half an hour was the last class (D). Minutes 1-15 were their own class as that is usually the time limit before a lightning strike is considered as part of a different storm. Minutes 16-30 were another class since predicting lightning onset 30 minutes beforehand is of interest. Anything after that was its own class. The code that performed this task is included in Appendix D.

An issue that Speranza [2] ran into during his research is that his models were dependent on time of day, as lightning is dependent on time of day. As SMOTE synthetically and randomly generates points to balance a data set, epoch time in the data set was dropped for two reasons. The first is that future models built would not become dependent on the time of day. The second is SMOTE does not balance data with equal time steps. Since time-series models such as LSTM's have been applied to the dataset, with average results, a regular neural network is applied instead [23].

The classifications helped balance the data set. The UBL [37] package in R applied the synthesis minority over-sampling technique to balance the data set. $K-nearest$ neighbors was used with $k=5$ and Euclidian distance to find the nearest neighbors. The majority class, A, was under sampled while the minority class, B,C,D, were oversampled and new synthetic points were generated to balance the classes.

Once balanced, the classification variable was dropped from the data set. This process was performed for each EFM, creating 26 different balanced data sets.

3.4 Determining Model Adequacy

The predictions from neural networks are continuous values. There are two direct ways to determine model adequacy. The first is the loss in the model. This is calculated based on the training set and validation set and is an interpretation of how well the model does in the two data sets. The loss function used to determine the loss per model was mean-squared error (MSE). The second measure of the model is how well it can predict the validation set. This is measured using mean-absolute error (MAE). The mean of all the absolute values of the difference between the model's prediction and the validation set's dependent variable are taken. Both reflect how well the model predicts lightning onset, where a lower value indicates a better fit and prediction [26].

A final way to determine model adequacy is examining the ratio of the MAE and the average of the predicted lightning onset values. This determines model adequacy. While models may have similar MAE's, different predicted lightning onset values can help determine which model is better. The same goes for models with similar average predicted lightning onset values and different MAE's. For example, two models have a MAE of 5

minutes, but model A has a predicted lightning onset of 5 and model B has a predicted lightning onset of 30 minutes. Model B is the better model as a 5-minute deviation where lightning onset is usually 5 minutes is a much larger deviation than 5 minutes when the average is 30 minutes. Therefore, the ratio of each EFM ‘s MAE to its predicted lightning onset helps differentiate the models’ performances, where a lower ratio is better.

3.5 Hyperparameter Tuning

Initial hyperparameter tuning was performed, using the keras package in R, to determine what hyperparameter settings to use for the feedforward neural networks. Keras is an application programming interface for neural networks [39]. The first EFM’s balanced data set was the test sets. The four variables found to change the neural networks’ performances were: epoch time, number of layers in a neural network, presence/absence of weight regularization and batch normalization, and neuron count. The ranges for epoch time and number of layers were found for when neural network performance plateaued or decreased. The epoch time range was 35-85 and number of layers was 4 or 5. Batch normalization per layer and weight regularization always improved the neural networks’ performance. However, weight regularization only worked depending on what weight limits were set. The pattern in weight regularization that seemed to produce better results was the first layer having the highest weight and then decreasing the weight amount per layer. For neuron count per layer, the trend was to decrease neuron count per layer and neural network performance seemed to performance best when the input layer had around 200 neurons.

3.6 Data Dimensionality Reduction

The data set consists of 52 variables, 26 which are the voltage readings from the EFMs and 26 which are the lightning onset times. Given the proximity in distance of the EFMs, storms would pass over the field mills in some successive pattern, meaning the variables are correlated.

PCA was performed on the balanced data, using the factoextra package in R, for the first EFM to reduce the dimensionality of the data and to improve model performance [38]. The number of components picked explained over 90% of the variation in the data. Each model built during the hyperparameter tuning phase was also fed with the PCA data. The PCA data yielded higher loss values, MAEs, and ratios each time. While PCA did reduce the dimensionality of the data, it did not improve model performance. So, the regular balanced dataset was used moving forward and PCA was not included in the process as it produced worse results than non-PCA datasets.

3.7 Experimental Design

While optimal parameter ranges were found during the hyperparameter tuning phase, an infinite number of combinations between the 4 factors (epoch time, layers, neuron count, weight regularization) still exist. A design of experiment was created to test these parameters. Each parameter was transformed into a two-level factor as shown in Table 1.

Table 1. Neural Network Parameters

Factor	**Neuron Reduction Per Layer**	**Layers**	**Weight Regularization**	**Epoch Time**
High	0.75	5	Descending	85
Low	0.5	4	Constant	35

The neuron reduction per layer is the percentage of neuron count in the next layer with the first layer always starting at 200 neurons. For weight regularization, a decreasing weight meant that the last layer before the output node would have a weight regularization of 0.001 and each layer before that would just add a 0.001. An example is if there were 3 layers before an output node, the corresponding weight regularizations per layer from the input would be 0.003, 0.002, 0.001.

Given four factor two-level factors, a 2^4 full factorial design was created. However, each combination of the four factors requires 26 separate runs for each EFM. If a full factorial design was used, a total of 416 neural networks must be created and trained. A two-level fractional factorial design helped to reduce the number of neural nets required while still being able to identify possible parameters that have a large effect on neural network performance. The design is in Table 2.

Table 2. Fractional Factorial Design for the Neural Networks

Run (Design)	Neuron Reduction Per Layer	Layers	Weight Regularization	Epoch Time
1 (1)	0.5	4	Constant	35
2 (a)	0.75	4	Constant	85
3 (b)	0.5	5	Constant	85
4 (ab)	0.75	5	Constant	35
5 (c)	0.5	4	Descending	85
6 (ac)	0.75	4	Descending	35
7 (bc)	0.5	5	Descending	35
8 (abc)	0.75	5	Descending	85

3.8 Conclusion

The original datasets had readings down to 0.005 of a second. The EFM and LDAR data were reconstructed to reflect lightning onset in one-minute intervals. From there, 26 separate datasets were built to balance the lightning onset values. By performing hyperparameter tuning and creating a one-half fractional factorial design, different neural network structures and their performance were examined. Chapter IV investigates how the different models performed in all the metrics, what parameters affected performance, and how the analysis address the problem statements.

IV. Discussion

4.1 Introduction

This chapter discusses the results and analysis of the research. First, initial results from the neural networks' performances will be examined. Then, models are fit to examine the significance of parameters and how they affect prediction accuracy.

4.2 Initial Results

The loss, mean absolute error, mean of the predicted onset values, and the ratio of MAE to the mean of the predicted onset values, were collected after each ANN trained and predicted on the datasets. Mean of the predicted onset values was recorded to calculate the ratio so there were only three true measures of performance. All performance metrics were recorded for every ANN configuration for every EFM. The best and worst performance combinations are summarized in Table 3.

Table 3. Initial Results from Neural Networks

	Performance	Design	Electric Field Mill
Greatest Loss	1448.51 MSE	abc	18
Lowest Loss	61.77 MSE	1	34
Greatest MAE	12.80 minutes	ac	18
Lowest MAE	2.49 minutes	1	22
Greatest Ratio	0.33	abc	15
Lowest Ratio	0.09	abc	30

A closer comparison between the best and worst performance metrics is seen in the plotted model performance over epochs. Figures 1 and 2 represent the loss function and metric performance over each epoch for the models with the best and worst loss values. The plots show that early on, the models fit the validation set well with slight fluctuations and then plateau out rather quickly. In the ANN with design setting abc for EFM 18, the model's performance seems to get a bit worse over the last few epochs and loss starts to increase again. For the best performing model, the neural network with design 1 for EFM 34, the model's loss starts at a much lower value than the worst performing value. The loss increases over the next few epochs before decreasing and plateauing out. Between the two graphs, the main differentiation in how the models fit the validation set was the starting loss value. The plots then level out rather quickly, indicating the model was no longer improving over each epoch.

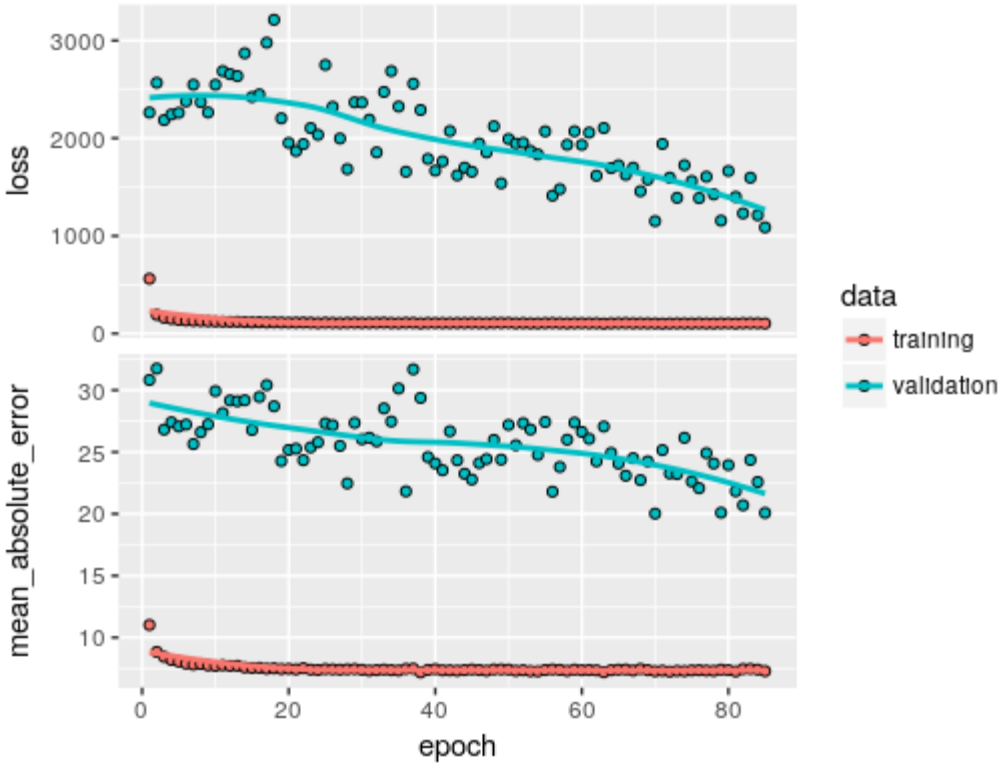

Figure 1. Neural Network performance over epochs for Design abc for EFM 18

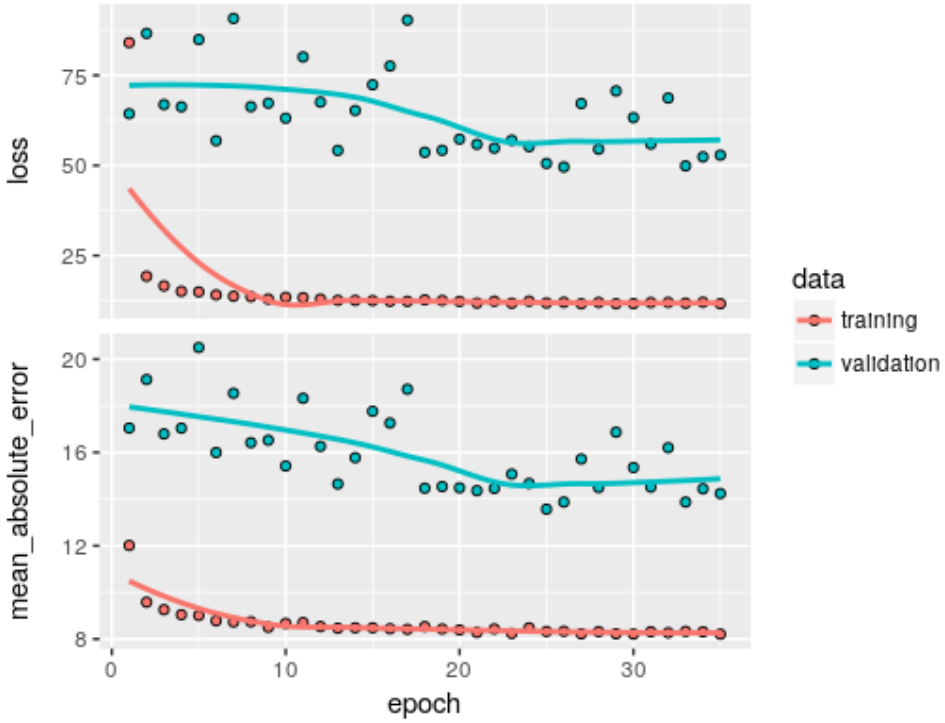

Figure 2. Neural Network performance over epochs for Design 1 for EFM 34

The next graphical comparison, in Figures 3 and 4, is between the models with the best and worst mean absolute errors. Figure 3 contains the worst and Figure 4 contains the best. A similar trend is observed here where the model performance plateaus near the end of the epochs. The main difference here are the models' beginning MAE. It is possible that had there been an ANN configuration that had more epochs for design 1, a lower MAE could have been observed but it is not conclusive as some models fluctuated over epochs, which occurs in the best and worst ratio models.

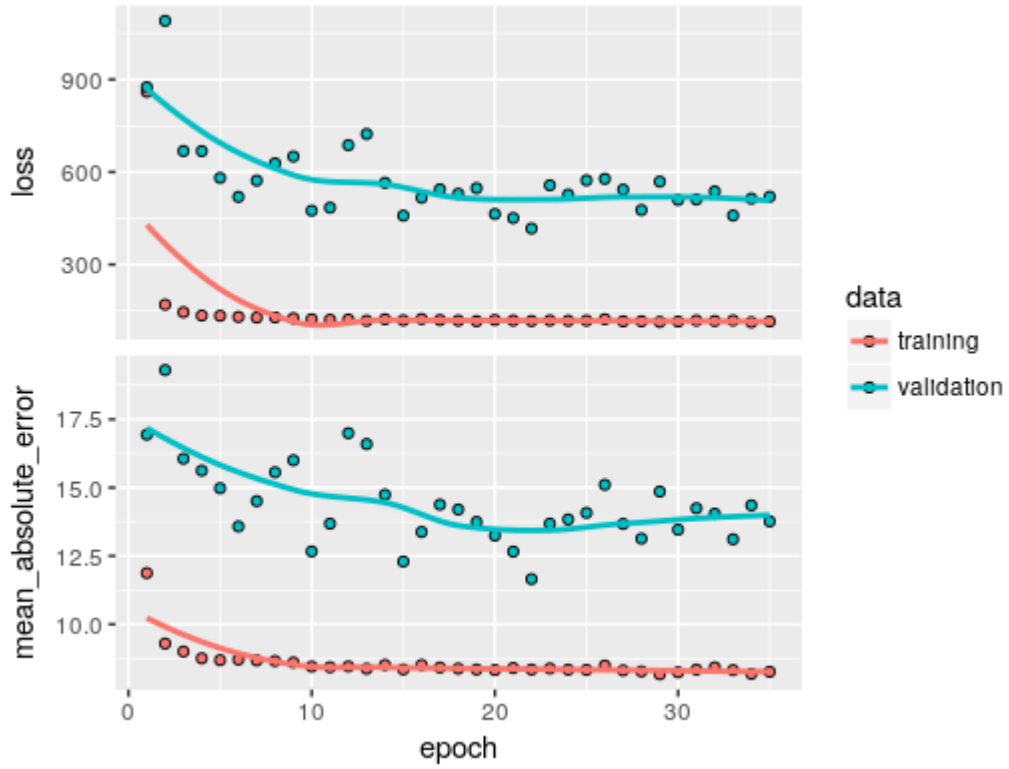

Figure 3. Neural Network performance over epochs for Design ac for EFM 18

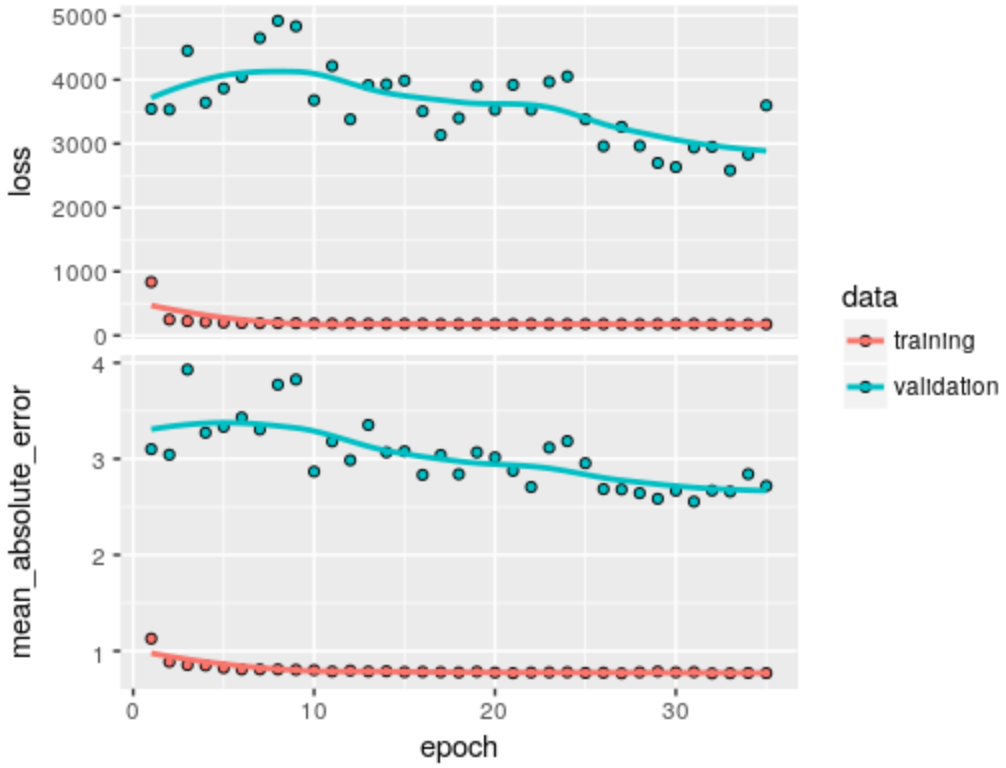

Figure 4. Neural Network performance over epochs for Design 1 for EFM 22

In Figures 5 and 6 are the best and worst models for the calculated ratio. A different trend appears here compared to the previous 4 plots. Instead of a plateau after a certain number of epochs, the models' performances fluctuate over epoch. Like Figures 1 through 4, the starting loss and MAE values, the starting values are where the values differ greatly as the models generally plateau quickly.

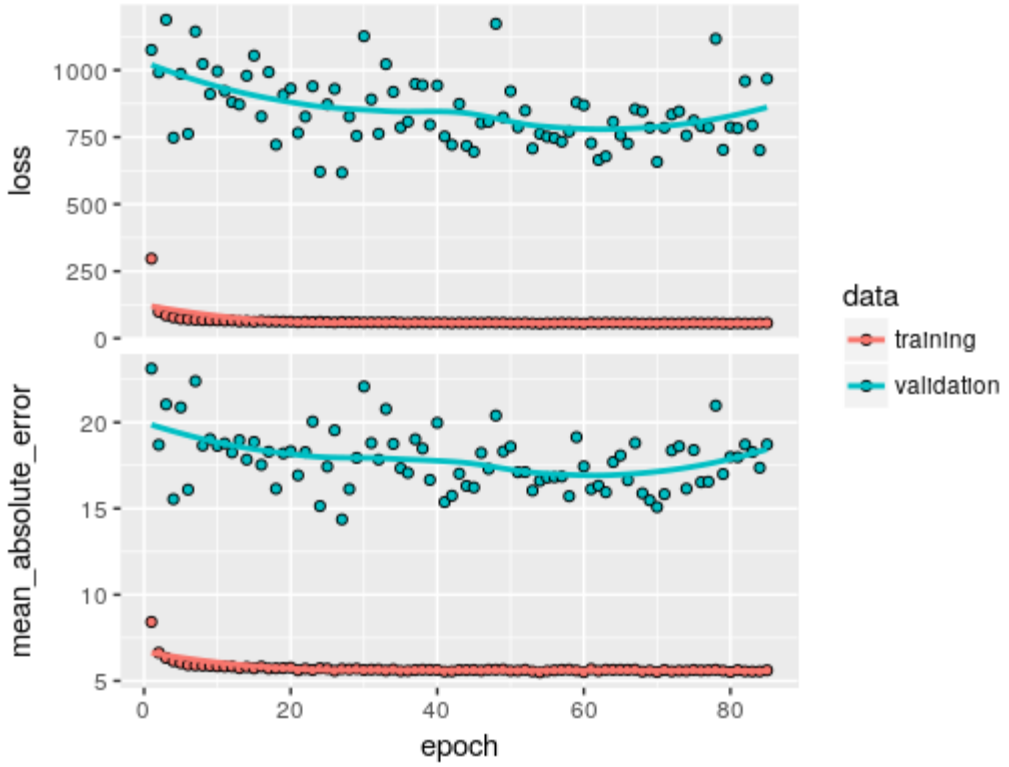

Figure 5. Neural Network performance over epochs for Design abc for EFM 15

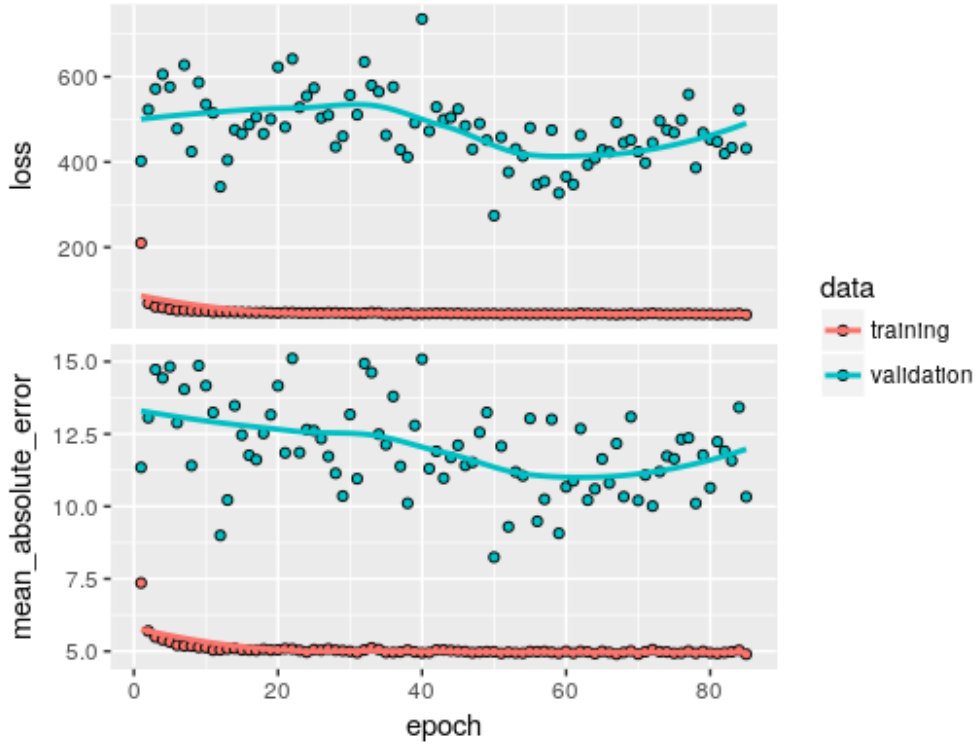

Figure 6. Neural Network performance over epochs for Design abc for EFM 30

The averages for the three metrics was calculated across the 8 different neural network configurations and between each EFM.

In Tables 4 and 5 are the results for the minimum and maximum values for the averages of the three results of interest and which design/ EFM the value is associated with. It is noticeable that the minimum and maximum values have greater differences between EFMs than the neural network configurations.

Table 4. Best and Worst Averages by EFM

Maximum Average Ratio	0.296	EFM 28
Minimum Average Ratio	0.117	EFM 34
Maximum Average MAE	10.408 minutes	EFM 18
Minimum Average MAE	3.019 minutes	EFM 34
Maximum Average Loss	1306.144 MSE	EFM 18
Minimum Average Loss	89.949 MSE	EFM 29

Table 5. Best and Worst Averages by Design

Maximum Average Ratio	0.203	Design bc
Minimum Average Ratio	0.182	Design a
Maximum Average MAE	7.001 minutes	Design c
Minimum Average MAE	6.096 minutes	Design abc
Maximum Average Loss	448.836 MSE	Design bc
Minimum Average Loss	376.598 MSE	Design a

While the overall ANN performance across all EFMs does not vary greatly with a range of (0.182, 0.203) MAE to average prediction ratio, there is greater variation if only one field mill happened to be the point of interest over another. Of all the designs set up, Design a achieved this ratio with a MAE around 6.4 minutes and an average predicted value of 34.67 minutes. In comparison, design bc's MAE is around 6.7 minutes and has an average predicted value of 32.9 minutes. Comparing the best and worst designs by predicted ratio, the MAE's are similar, with the difference being in the average predicted onset values creating that .02 difference in the ratios.

Comparing the average ratio of 0.296 found in EFM 28 to EFM 34's ratio, with a ratio of 0.117, the prediction accuracies between the two vary by about 18 percent. Similar trends can be seen in the average MAE and average loss performances where performance between neural networks are generally similar but can vary vastly between specific field mills.

4.3 Parameter Analysis

Two models were built to analyze the adjusted parameters. The models used each of the 208 models built as an observation and predicted for loss and ratio, nominal factors for each of the four parameters, using EFM as a blocking factor, and only going as high as the second order due to sparsity of effects.

In Figure 7, the first model, was fit to predict loss using the program JMP. There were no second order effects that were found to be statistically significant at the 95% confidence level. Summary of fit, analysis of variance and parameter estimates are also in Figure 7.

Summary of Fit

RSquare	0.921445
RSquare Adj	0.908646
Root Mean Square Error	97.75084
Mean of Response	416.7254
Observations (or Sum Wgts)	208

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	29	19950566	687951	71.9973
Error	178	1700830	9555	Prob > F
C. Total	207	21651396		<.0001*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	416.72538	6.777801	61.48	<.0001*
Reduction Rate[0.5]	-0.412351	6.777801	-0.06	0.9516
Layers[4]	-18.56377	6.777801	-2.74	0.0068*
Regularization[0]	-12.54756	6.777801	-1.85	0.0658
Epochs[35]	5.4157523	6.777801	0.80	0.4253

Figure 7. Parameter Estimates of Neural Network Performance on Loss

The adjusted R squared value indicates that the model explained 90.8% of the variation in the dependent variable, loss. From the analysis of variance section, the model's F Ratio shows the model is significant. Of the four parameters estimated, only the parameter layers found statistically significant. If the variable reduction was dropped, regularization would have a slightly lower p-value but was still not statistically significant. However, no main effects were removed so that the overall performance from the parameters could be examined. Since the input variables were nominal, the estimates represent the improvement between levels each variable has on the dependent variable. For the significant variable, layers, the estimate shows that using 4 layers will decrease loss by 18.56377 MSE compared to a 5-layer neural network.

In Figure 8, the second model, which predicted the ratio of mean absolute error to the average of predicted lightning onset, had similar results with the parameter estimates. In this case, an interaction variable between layers and epochs was included as it increased the adjusted R squared value. Here, the model only explains 65.3% of the variation in the dependent variable. None of the parameters were significant at the 95% confidence level but the model has an F Ratio of 14.0087, meaning the model is significant.

Summary of Fit

RSquare	0.703648
RSquare Adj	0.653419
Root Mean Square Error	0.029136
Mean of Response	0.19274
Observations (or Sum Wgts)	208

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	30	0.35675563	0.011892	14.0087
Error	177	0.15025316	0.000849	Prob > F
C. Total	207	0.50700878		<.0001*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	0.1927399	0.00202	95.41	<.0001*
Reduction Rate[0.5]	0.0032956	0.00202	1.63	0.1046
Layers[4]	-0.001327	0.00202	-0.66	0.5123
Regularization[0]	-0.002352	0.00202	-1.16	0.2460
Epochs[35]	0.0030213	0.00202	1.50	0.1365
Layers[4]*Epochs[35]	-0.003414	0.00202	-1.69	0.0928

Figure 8. Parameter Estimates of Neural Network Performance on Ratio

4.4 Model Adequacy

Residual analysis was performed to check for model adequacy. For both models, two graphs are looked at to check for normality in the errors and constant variance. A normal quantile plot of the studentized residuals is used to check if the errors are normally distributed. The two plots, one for each of the two models are in Figures 9 and 10.

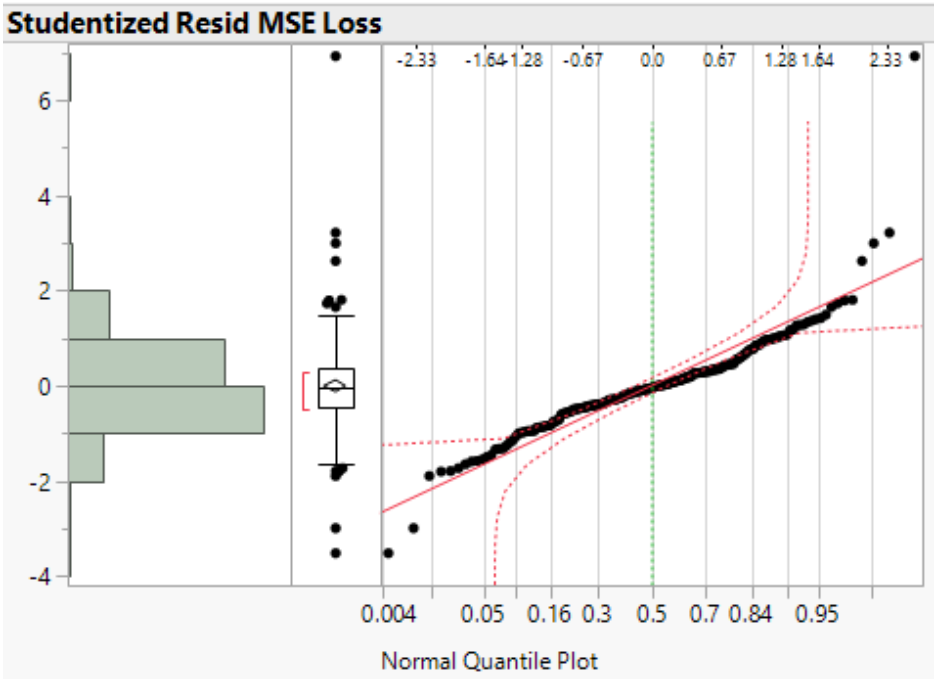

Figure 9. Quantile Plot for Loss Prediction

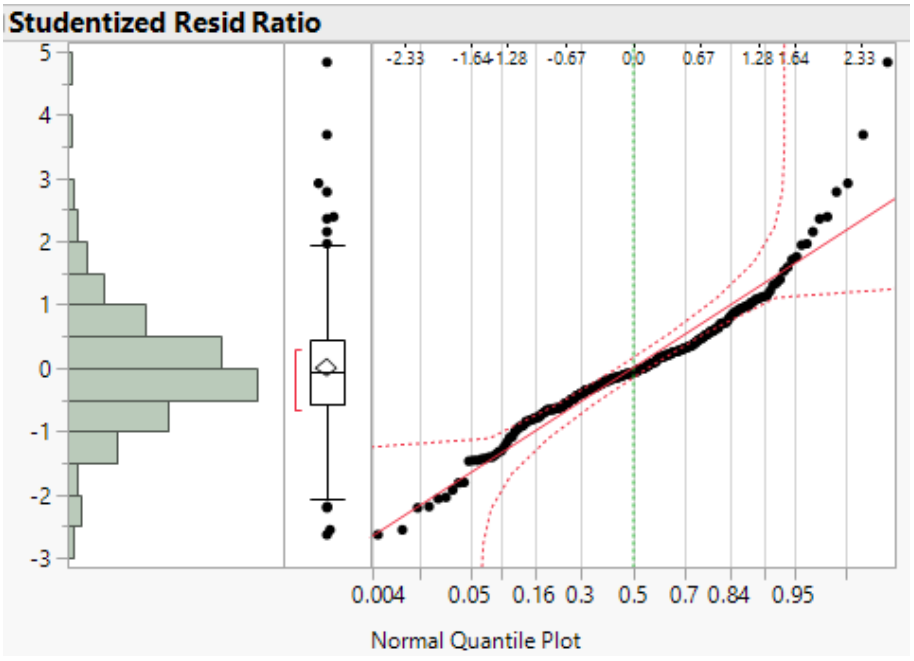

Figure 10. Quantile Plot for Ratio Prediction

The normal quantile for the loss model seems to show normally distributed studentized residual. However, there are slight tails at the end of the plot but overall, the histogram and probability plot seem to indicate that the errors in the loss model are normally distributed. The second figure is the histogram and normal quantile plot of the ratio model's studentized residuals. There is a heavier tail on the end in this plot. A look over the studentized residuals for the ratio plot show some with values greater than 3, indicating outliers as the possible reason for the tails. Looking at the overall plot and the histogram, the studentized residuals seem to be normally distributed even with heavy tail on one end.

The plots of the studentized residuals against predicted values is used to check for constant variance and nonlinearity in the model. These two plots are in Figures 11 and 12.

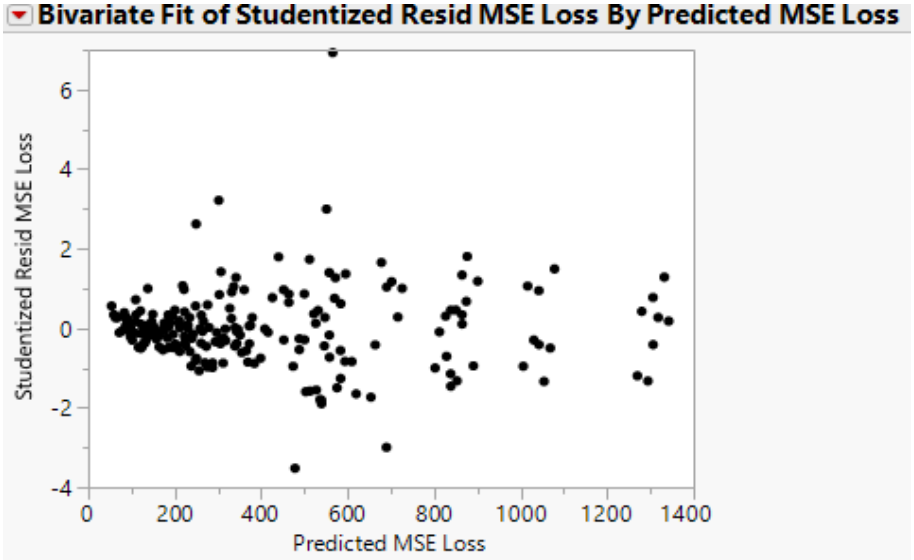

Figure 11. Predicted Loss vs Studentized Residual Plot for Loss Prediction

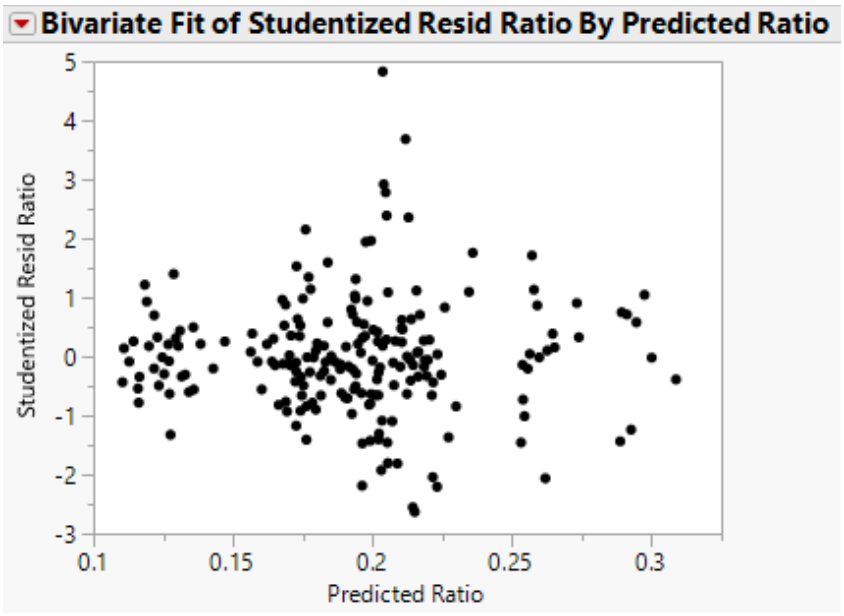

Figure 12. Predicted Loss vs Studentized Residual Plot for Ratio Prediction

A look at the plots indicate that there is generally no pattern. There is no funnel, double bow, or nonlinear pattern that can be distinguished to indicate the possibly of nonconstant variance in the models. Some of the points that seem to create a possible pattern are points with a studentized residual close to or greater than the absolute value of 3, indicating they are outliers.

4.5 Conclusion

The performance of the ANNs is demonstrated in the mean absolute error and ratios that were produced. The overall mean absolute error across all designs and EFMs was 6.57 minutes. The ratio between mean absolute error to predicted lightning onset value was 0.193. This showed a potential that the neural networks can predict lightning onset to a certain degree. Parameter analysis showed that while the least square models were significant in predicting lightning onset and loss, the parameters themselves were not

significant, making it difficult to tell which level of the 2-level parameters led to better performance.

V. Conclusion and Moving Forward

5.1 Introduction

This chapter summarizes the results of the study, issues with the work done, and discusses potential follow-on work for the project.

5.2 Overview of Results

The study examines the two research questions presented in Chapter 1.

1. Can a feedforward neural network per electric field mill handle the noise in the data and better predict lightning onset?
2. How well can a neural network predict lightning onset?

Insights were gained from the neural network performance based off mean absolute error and the error ratio as well as parameter analysis done on predicting loss and lightning onset. Unlike previous studies, neural networks were built to predict a continuous value rather than previous studies that forecasted lightning as a binary outcome. Without a binary outcome and confusion matrix to determine level of performance in a neural network, new measures were created to determine prediction accuracy.

While some insights were gained, other questions arose throughout the analysis and afterwards.

5.3 Future Work

Although 31 EFMs existed, 5 of them either worked for short periods of time or not at all. Due to inconsistent measurements, only 26 of the field mills could be used in the analysis. Given that the field mills are located at areas deemed important for gathering voltage data, having all field mills functioning could prove useful. Another issue the EFM

data presented was the number of observations. Originally, each 0.005 second reading was used as a separate observation but cleaning a data set with over 650 million lines for every five days of data, with parallel processing, took about a week. This was unfeasible given time constraints. However, instead of compressing the data set down into minute observations, one second intervals can also be used. While this may not improve any prediction results given how little voltage changes within one second and one minute, it would allow for the neural networks to differentiate at what point in time lightning storms start in comparison to voltage readings.

Other ideas for future work would require more data. The first is that there are many more factors that go into a lightning storm than just voltage readings. Other studies examined in Chapter II indicate that weather data on temperature, humidity, and wind speed can also improve algorithm predictions on lightning strikes. This would require new data to be measured in similar intervals as the EFM and LDAR data. Another suggestion is differentiating the lightning strikes as those moving in from the ocean and those that are not. This was suggested by the meteorologist in the 45th WS to identify light storm movements.

As for future methods to apply to this type of data set, there is one that should be explored. One of the challenges was that by balancing the data set, observations no longer had equal time steps between each other so a LSTM neural network could not be used. Further research can be done into finding out how to balance a data set while maintaining equal time steps so it may be put into a recurrent-type algorithm.

5.4 Conclusion

This research set out to examine the EFM and LDAR data in hopes of predicting lightning onset. With all the work done using these two datasets, follow-on research should require new datasets to develop better models and hopefully, better results.

Appendices

Appendix A

```
library(data.table)

## Warning: package 'data.table' was built under R version 3.5.3

library(plyr)

## Warning: package 'plyr' was built under R version 3.5.3

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 3.5.3

## -- Attaching packages ---------------------------------------------
-- tidyverse 1.2.1 --

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## Warning: package 'ggplot2' was built under R version 3.5.3

## Warning: package 'tibble' was built under R version 3.5.3

## Warning: package 'tidyr' was built under R version 3.5.3

## Warning: package 'readr' was built under R version 3.5.3

## Warning: package 'purrr' was built under R version 3.5.3

## Warning: package 'dplyr' was built under R version 3.5.3

## Warning: package 'stringr' was built under R version 3.5.3

## Warning: package 'forcats' was built under R version 3.5.3

## -- Conflicts ------------------------------------------------ tid
yverse_conflicts() --
## x dplyr::arrange()   masks plyr::arrange()
## x dplyr::between()   masks data.table::between()
## x purrr::compact()   masks plyr::compact()
## x dplyr::count()     masks plyr::count()
## x dplyr::failwith()  masks plyr::failwith()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::id()        masks plyr::id()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
```

```
## x dplyr::mutate()    masks plyr::mutate()
## x dplyr::rename()    masks plyr::rename()
## x dplyr::summarise() masks plyr::summarise()
## x dplyr::summarize() masks plyr::summarize()
## x purrr::transpose() masks data.table::transpose()

root = "/home/dom/Desktop/ChengThesis/ThesisData/Unprocessed EFM Data"
44irs._lvl1 <- list.dirs(root, recursive = F)
save.EFM_Data<-data.table(day = character(), time = character(), voltag
e = integer(), location=character(),timestamp=character())

for(I in 44irs._lvl1) {

  44irs._lvl2 <- list.dirs(I, recursive = F)

for(j in 44irs._lvl2) {

  44irs._lvl3 <- list.dirs(j, recursive = F)

for(k in 44irs._lvl3) {
  EFM_Data<-data.table(day = character(), time = character(), voltage =
integer(), location=character(),timestamp=character())
  data.files <- list.files(k,
                           pattern = '(.txt|.raw)',
                           full.names = T)
  if(file.info(data.files[1])$size != 0){
  EFM_Data[,c(1:3)] <- data.table::fread(data.files[1])
  EFM_Data$location <- substr(basename(data.files[1]), start = 1, stop
= 5)
  EFM_Data$timestamp <- basename(dirname(data.files[1]))
  }
  #This will pull in the data from one timestamp
  #will create a data table with date,time,voltage,location,and timesta
mp (in 30min increments)
  for(l in 2:length(data.files)) {

   if(file.info(data.files[l])$size == 0) next
  this.data <- try(fread(data.files[l]), silent = T)
   this.data$location <- substr(basename(data.files[l]), start = 1, sto
p = 5)
   this.data$timestamp <-EFM_Data$timestamp

    EFM_Data <- rbind(EFM_Data,this.data)

   }

  setkeyv(EFM_Data, c('location', 'timestamp'))
```

```
  temp.data <- ddply(EFM_Data,.(location,timestamp))
  names(temp.data)[1] <- "day"
  names(temp.data)[2] <- "time"
  names(temp.data)[3] <- "voltage"
  save.EFM_Data<-rbind(save.EFM_Data,temp.data)

  ###################################
  #move onto next timestamp in the day
}
  #move onto next day in series of 5 days
}
  #Move onto next group of dates.

}
```

Appendix B

```
library(data.table)
library(plyr)
library(foreach)
#clears all the variables being used
DateData<-data.table(Year = numeric(),Month = numeric(),Day = numeric(),Hour = numeric(),Strike= numeric())
tempDate<-data.table(Year = numeric(),Month = numeric(),Day = numeric(),Hour = numeric(),Strike= numeric())
StrikeDateData = NULL
#change your root to what you are working in.
root = "/home/dom/Desktop/ChengThesis/ThesisData/LDAR"
dirs_lvl1 <- list.dirs(root, recursive = F)
test=NULL

#for to go through each year of data
for (j in 1:length(dirs_lvl1)){
  #gets the list of all txt files in each year
  data.files <- list.files(dirs_lvl1[j],
                           pattern = '(.txt|.RAW|.raw)'
                          )
  foreach(l = 5:9,.export = 'fread') %do% {

    if(file.info(data.files[l])$size == 0){
      test = NULL
    }
    else {
      #read in the data
      test<- try(fread(data.files[l],fill=TRUE),silent = TRUE)
      test<-na.omit(test)

      #get the month, day, year, and hour for the data point.
      Month<-as.numeric(substring(test[1,1],1,2))
      Day<-as.numeric(substring(test[1,1],4,5))
      Year<-as.numeric(substring(test[1,1],7,10))
      Hour<-as.numeric(substring(test[1,1],12,13))
      tempDate<-data.table(Year = Year, Month = Month, Day = Day, Hour = Hour, Strike=1)
      StrikeDateData<-rbind(StrikeDateData,test)
      test = NULL
      return(tempDate)
    }

  }
  #stores the data. Make sure to check and save in increments
  #StrikeDateData<-rbind(StrikeDateData, DateData)
  DateDate = NULL

}
```

Appendix C

```
      #This will pull in the data from one timestamp
      #will create a data table with date,time,voltage,location,and timestamp (in 30min increments)
      for(l in 1:length(data.files)) {

        if(file.info(data.files[l])$size == 0) next
        this.data <- try(fread(data.files[l]), silent = T)
        names(this.data)[1] <- "day"
        names(this.data)[2] <- "time"
        names(this.data)[3] <- "voltage"
        this.data$location <- substr(basename(data.files[l]), start = 1, stop = 5)
        this.data$timestamp <-EFM_Data$timestamp

        EFM_Data <- rbind(EFM_Data,this.data)
      }

      setkeyv(EFM_Data, c('location', 'timestamp'))
      temp.data <- ddply(EFM_Data,.(location,timestamp))
      names(temp.data)[1] <- "day"
      names(temp.data)[2] <- "time"
      names(temp.data)[3] <- "voltage"
      save.EFM_Data<-rbind(save.EFM_Data,temp.data)

      ################################
      #move onto next timestamp in the day
    }
    #move onto next day in series of 5 days
  }
  #Move onto next group of dates.
}
save(save.EFM_Data, file = "efmtest85.RData")


#converting day number to a date
rm(list=ls())
gc()
load("~/efmtest85.RData")
library(lubridate)
year <- as.numeric(substring(save.EFM_Data[1:156729712,1],1,4))
day <- as.numeric(substring(save.EFM_Data[1:156729712,1],5,7))
df <- data.frame(cbind(year,day))
df$origin <-as.Date(paste0(df$year, "-01-01"),tz="UTC")-days(1)
df_new <- as.Date(df$day,origin = df$origin, tz ="UTC")
epoch_time1<-as.data.frame(difftime(as.POSIXct(paste(df_new,save.EFM_Data[1:156729712,2]),format = "%Y-%m-%d %H:%M:%S"),
                                    as.Date("2013-01-01"), units= "mins"),stringsAsFactors=FALSE)
```

```
#converting meters from KSCS to lat long
StrikeDateData[,2] <- sapply(StrikeDateData[,2],as.integer)
StrikeDateData2 <-  StrikeDateData

 latitude.conv <- function(x){28.538486 + ((x/1000)/6378) *(180/pi)
 }
  longitude.conv <- function(y){-80.642633 + ((y/1000)/6378) *(180/pi)/cos(28.538486*pi/180)
  }

StrikeDateData2[,2] <- lapply(StrikeDateData[,3],latitude.conv)
StrikeDateData2[,3] <- lapply(StrikeDateData[,2],longitude.conv)

StrikeDateData2 <- data.frame(cbind(StrikeDateData2[,2:3],StrikeDateData[,5]))
save(StrikeDateData2, file = "LDAR_Final.RData")

###Read in data

rm(list = ls(all.names = TRUE))
gc()
library(data.table)
library(plyr)
library(tidyverse)
root = "/home/dom/Desktop/ChengThesis/ThesisData/EFMData"
dirs_lvl1 <- list.dirs(root, recursive = F)
save.EFM_Data<-data.table(day = character(), time = character(), voltage = integer(), location=character(),timestamp=character())

for(i in dirs_lvl1[85]) {
  dirs_lvl2 <- list.dirs(i, recursive = F)
  for(j in dirs_lvl2) {
    dirs_lvl3 <- list.dirs(j, recursive = F)
    for(k in dirs_lvl3) {
      EFM_Data<-data.table(day = character(), time = character(), voltage = integer(), location=character(),timestamp=character())
      data.files <- list.files(k,
                               pattern = '(.txt|.RAW)',
                               full.names = T)

      for(l in 1:length(data.files)){
        if(file.info(data.files[1])$size != 0){
          EFM_Data[,c(1:3)] <- data.table::fread(data.files[1])
          EFM_Data$location <- substr(basename(data.files[1]), start = 1, stop = 5)
          EFM_Data$timestamp <- basename(dirname(data.files[1]))
        }
      }
```

```
year <- as.numeric(substring(save.EFM_Data[156729713:313459425,1],1,4))
day <- as.numeric(substring(save.EFM_Data[156729713:313459425,1],5,7))
df <- data.frame(cbind(year,day))
df$origin <-as.Date(paste0(df$year, "-01-01"),tz="UTC")-days(1)
df_new <- as.Date(df$day,origin = df$origin, tz ="UTC")
epoch_time2<-as.data.frame(difftime(as.POSIXct(paste(df_new,save.EFM_Data[156729713:313459425,2]),format = "%Y-%m-%d %H:%M:%S"),
                                    as.Date("2013-01-01"), units= "mins"),stringsAsFactors=FALSE)

#################
rows <- nrow(save.EFM_Data)
year <- as.numeric(substring(save.EFM_Data[313459426:rows,1],1,4))
day <- as.numeric(substring(save.EFM_Data[313459426:rows,1],5,7))
df <- data.frame(cbind(year,day))
df$origin <-as.Date(paste0(df$year, "-01-01"),tz="UTC")-days(1)
df_new <- as.Date(df$day,origin = df$origin, tz ="UTC")
epoch_time3<-as.data.frame(difftime(as.POSIXct(paste(df_new,save.EFM_Data[313459426:rows,2]),format = "%Y-%m-%d %H:%M:%S"),
                                    as.Date("2013-01-01"), units= "mins"),stringsAsFactors=FALSE)

rows <- nrow(epoch_time3)
final_epoch<-data.frame(a=c(epoch_time1[1:156729712,],epoch_time2[1:156729713,],epoch_time3[1:rows,]))

#################
year <- as.numeric(substring(save.EFM_Data[313459426:470189138,1],1,4))
day <- as.numeric(substring(save.EFM_Data[313459426:470189138,1],5,7))
df <- data.frame(cbind(year,day))
df$origin <-as.Date(paste0(df$year, "-01-01"),tz="UTC")-days(1)
df_new <- as.Date(df$day,origin = df$origin, tz ="UTC")
epoch_time3<-as.data.frame(difftime(as.POSIXct(paste(df_new,save.EFM_Data[313459426:470189138,2]),format = "%Y-%m-%d %H:%M:%S"),
                                    as.Date("2013-01-01"), units= "mins"),stringsAsFactors=FALSE)

rows <- nrow(save.EFM_Data)
year <- as.numeric(substring(save.EFM_Data[470189139:rows,1],1,4))
day <- as.numeric(substring(save.EFM_Data[470189139:rows,1],5,7))
df <- data.frame(cbind(year,day))
df$origin <-as.Date(paste0(df$year, "-01-01"),tz="UTC")-days(1)
df_new <- as.Date(df$day,origin = df$origin, tz ="UTC")
epoch_time4<-as.data.frame(difftime(as.POSIXct(paste(df_new,save.EFM_Data[470189139:rows,2]),format = "%Y-%m-%d %H:%M:%S"),
                                    as.Date("2013-01-01"), units= "mins"),stringsAsFactors=FALSE)

rows <- nrow(epoch_time4)
final_epoch<-data.frame(a=c(epoch_time1[1:156729712,],epoch_time2[1:156729713,],epoch_time3[1:156729713,],epoch_time4[1:rows,]))
```

```
final_efm<-data.frame(cbind(save.EFM_Data[,3],save.EFM_Data[,4]))
final_efm$epochtime <- as.data.frame(final_epoch)
final_efm$X1 = as.numeric(as.character(final_efm$X1))
final_efm$epochtime = as.numeric(floor(unlist(final_efm$epochtime)))


library(dplyr)
efm_avg = final_efm %>%
  group_by(X2,epochtime) %>%
  summarise(avg = mean(X1))
save(efm_avg, file = "efm_avg.RData")

rm(list=ls())
gc()

load("~/efm_avg.RData")
load("~/LDAR_Final.RData")
##########################################KSC01
library(data.table)
library(gmt)
#calculate distance between KSC01 and all lightning strikes in nautical miles
KSC01_geodist <- as.data.frame(geodist(StrikeDateData2[,1],StrikeDateData2[,2],28.7036,279.3314,units="nm"))
#add a second column that is the epoch time of all strikes and their respective nautical miles from KSC01
KSC01_geodist[,2] <- StrikeDateData2[,3]
KSC01_geodist[,2] <- (KSC01_geodist[,2]/60)-22616640
KSC01_geodist[,2] <- as.numeric(floor(KSC01_geodist[,2]))
#KSC01_geodist[,2] <- KSC01_geodist[order(KSC01_geodist[,2]),]
#subset data with only nautical miles less than or equal to 5
test<- subset(KSC01_geodist, KSC01_geodist[,1] <= 5)
#order the epoch times from beginning to end
# test <- test[order(test[,2]),]
# test[,2] <- test[,2]/60
# test[,2] <- test[,2]-22616640
# test[,2] <- as.numeric(floor(test$v2))

#take the differences between each lightning strike
test[,3] <-c(NA, diff(test$v2,lag=1,differences = 1))
#subset if the difference in lightning strikes is greater than or equal to 900 seconds (storms)
storms <- test[(test[,3]>=15),]
#create start stop times
endtime <- as.data.frame(storms[,2]-storms[,3])
endtime <- as.data.frame(endtime[-1,])
endtime <- rbind(endtime,tail(test,1)[,2])
starttime<-as.data.frame(storms[,2])
starttime[1,] <- test[1,2]
startstop <- cbind(starttime,endtime)
#remove duplicates
```

```
startstop <- startstop[!(startstop[,1]==startstop[,2]),]
#add the 30 minute window forecast
test<-test[,1:2]
data <- test[,2]
#loop through checking which interval of lightning strikes a reading occured in
startstop = as.matrix(startstop)


data = as.matrix(data)
data = cbind(data,0)
data <- as.data.frame(data)
library(dplyr)
data = data %>%
  group_by(V1) %>%
    summarise(V2 = mean(V2))
data = as.matrix(data)
for (i in 1:nrow(data)){
  check = F
  for (j in 1:nrow(startstop)){
    if(data[i]+30>=startstop[j,1] && data[i]+30<=startstop[j,2] && check == FALSE){
      check = TRUE
      data[i,2] = startstop[j,2]-data[i]
    }
  }
}
#create dataframe of 30 minute forecast
library(tidyverse)
library(dplyr)
data <- data.frame(data)
test <- distinct(test,V2,.keep_all=TRUE)
nm <- test[,1]
epochtime_strike<-test[,2]
time_to_end <- data[,2]
KSC01_clean <- as.data.frame(cbind(epochtime_strike,time_to_end))
KSC01_clean <-as.data.frame(KSC01_clean)
KSC01_cleanavg = KSC01_clean %>%
  group_by(epochtime_strike) %>%
  summarise(avg_time_end = mean(time_to_end))

seq_ksc = seq(min(KSC01_geodist$V2),max(KSC01_geodist$V2),1)
seq_ksc = tibble(seq_ksc)
KSC01_cleanavg$epochtime_strike = as.numeric(KSC01_cleanavg$epochtime_strike)
seq_ksc = seq_ksc %>% left_join(KSC01_cleanavg, by = c( "seq_ksc" = 'epochtime_strike' ))
#KSC01_cleanavg = KSC01_cleanavg %>% full_join(seq_ksc, by = c('epochtime_strike' = "seq_ksc"))
#seq_ksc[which(!is.na(seq_ksc$avg_time_end)),]
library(xts)
seq_ksc = seq_ksc %>% mutate(n=avg_time_end) %>% fill(n)
seq_ksc = as.matrix(seq_ksc)
for(i in 2:nrow(seq_ksc)){
  t=F
  if(is.na(seq_ksc[i,2])){
    seq_ksc[i,3] = seq_ksc[i-1,3]-1
  }
}
seq_ksc[,3][seq_ksc[,3]<0]=0
seq_ksc <- as.data.frame(seq_ksc)
library(imputeTS)
seq_ksc[,3] <- na_replace(seq_ksc[,3],0)
seq_ksc <- cbind(seq_ksc$seq_ksc,seq_ksc$n)
seq_ksc <- as.data.frame(seq_ksc)
KSC01 <- subset(efm_avg, efm_avg[,1]=='KSC01')
if (dim(KSC01)[1]==0){
  KSC01<-subset(efm_avg,efm_avg[,1]=='KSC02')
}


df = KSC01 %>%ungroup()
gaps = df$epochtime-lag(df$epochtime)
gaps = which(gaps!=1)
df1 = c()
for(i in gaps){
  df_temp = data.frame(
    df$X2[1],
    seq(df$epochtime[i-1]+1,df$epochtime[i]-1, by  = 1),
    mean(df$avg[i-1],df$avg[i])
  )
  colnames(df_temp) = colnames(df)

  df1 = rbind(df1,df_temp)
}
df = df %>% bind_rows(df1) %>% arrange(epochtime)

library(dplyr)
library(tibble)
df  = df %>% mutate(rowid = row_number())
check = 24*60*5
seq_check = seq(1,check,1)
```

```
if(nrow(df)<check){
  seq_to_add = which(!seq_check %in% df$rowid)
  seq_epoch_add = seq(1, length(which(!seq_check %in% df$rowid)),1)
  seq_epoch_add = seq_epoch_add + df$epochtime[length(df$epochtime)]
  df_add = data.frame(seq_to_add)
  df_add$X2 = df$X2[1]
  df_add$epochtime = seq_epoch_add
  df_add$avg = df$avg[length(df$avg)]
  colnames(df_add)[1] = 'rowid'
  df_add = df_add %>% select(X2,epochtime,avg,rowid)
  df = df %>% bind_rows(df_add)
}

finaldata = df[,2:3]
names(finaldata) = c("epochtime","KSC01_voltage")


##CHANGE
#+60
w = 145*60*5*24

seq_ksc = seq_ksc[-1:-w,]
finaldata$KSC01_onset <- seq_ksc[1:length(df$epochtime),2]
finaldata <- as.data.frame(finaldata)
```

Appendix D

```
library(dplyr)
column = data_all$KSC01_onset
column2=as.data.frame(column)
column2 = column2 %>% ## name of the dataframe
  mutate( newc = ## name of the new column to be added
             case_when(
               column == 0 ~ "A",
               column >0 & column <=15 ~ "B",
               column >15 & column <=30 ~ "C",
               column >30  ~ "D"
             )
  )

column2[,2] <- as.factor(column2[,2])
newc <- column2$newc
newc <- as.data.frame(newc)
epoch_time <- as.data.frame(epoch_time)
data_all <-cbind(data_all[,1:52],newc,epoch_time)

table(data_all$newc)
data_all<-data_all[,1:53]

library(UBL)
set.seed(123)
smoteif_data <- SmoteClassif(newc~.,data_all, C.perc="balance",k=15,repl=FALSE,dist="Euclidean",p=1)
table(smoteif_data$newc)
save(smoteif_data,file="smote_final.RData")

smoteif_data <- smoteif_data[order(smoteif_data[,54]),]
smoteif_data$newc <- NULL
```

Bibliography

1. D. Hill, *Lightning Prediction Using Artificial Neural Networks and Electric Field Mill Data*. PhD thesis, Air Force Institute of Technology, 2018.

2. D. Speranza, *Lightning Prediction Using Recurrent Neural Networks*. MS Thesis, Air Force Institute of Technology, 2019.

3. W.P. Roeder, L. L. Huddleston, W. H. Bauman II, and K. B. Doser, "Weather research requirements to improve space launch from Cape Canaveral Air Force Station and NASA Kennedy Space Center," *Space Traffic Management Conference.* 2014.

4. P. K. Medelius, W. P. Roeder, and J. T. Willingham, "Lightning Protection At The Kennedy Space Center And Cape Canaveral Air Force Station," *10th Conference on Aviation, Range, and Aerospace Meteorology*, pp. 175-177, 2002.

5. W. P. Roeder, M. McNamara, M. McAleenan, K. A. Winters, L. M. Maier, and L. L. Huddleston, "The 2014 upgrade to the lightning warning areas used by the 45th Weather Squadron," *18th Conference on Aviation, Range, and Aerospace Meteorology*, 2017.

6. S. C. Wang, "Artificial Neural Network," *The Springer International Series in Engineering and Computer Science*, vol. 743, pp. 81-90, 2003.

7. F. Chollat, J.J. Allaire, *Deep Learning with R*. Manning Publications Co., 2018.

8. J. T. Connor, R. D. Martin, L. E. Atlas, "Recurrent Neural Networks and Robust Time Series Prediction," *IEEE Transactions on Neural Networks*, vol. 5, pp. 240-253, 1994.

9. T. H. Trinh, A. M. Dai, M. Luong, Q. V. Le, "Learning Longer-term Dependencies in RNNs with Auxiliary Losses," 2018.

10. G. Wang, W. Kim, G. Kil, D. Park, S. Kim, "An Intelligent Lightning Warning System Based on Electromagnetic Field and Neural Network," *Energies*, vol. 12, pp. 1-11, 2019.

11. "Dates and Times in R." https://www.stat.berkeley.edu/~s133/dates.html, 2006.

12. "Package 'lubridate'." https://cran.r-project.org/web/packages/lubridate/lubridate.pdf, 2018.

13. D. Rolnick, A. Veit, S. Belongie, N. Shavit, "Deep Learning is Robust to Massive Label Noise," 2018.

14. G. Cheng, V. Peddinti, D. Povey, V.Manohar, S. Khundanpur, Y. Yan, "An exploration of dropout with LSTMs," 2017.

15. M. Mashiri and A. Farshbaf Geranmayeh, "Tuning the parameters of an artificial neural network using central composite design and genetic algorithm," *Scientia Iranica,* vol. 18, no. 6, pp. 1600-1608, 2011.

16. F. Lasheras, J. Vilan, P. Nieto, J. Diaz, "The use of design of experiments to improve a neural network model in order to predict the thickness of the chromium layer in a hard chromium plating process," *Mathematical and Computer Modeling,* vol. 52, no. 7-8, pp. 1169-1176, 2010.

17. D. C. Montgomery, *Design and Analysis of Experiments*. John Wiley & Sons, 2013.

18. "A Grammar of Data Manipulation." https://dplyr.tidyverse.org/index.html, 2019.

19. N. V. Chawla, "Data Mining for Imbalanced Datasets: An Overview," *Data Mining and Knowledge Discovery Handbook,* pp. 853-867, 2005.

20. M. Buda, A. Maki, M. A. Mazurowski, "A systemic study of the class imbalance problem in convolutional neural networks," *Neural Networks,* vol. 106, pp. 249-259, 2018.

21. X. Liu, Z. Zhou, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Transactions on Knowledge and Data Engineering,* vol. 18, pp. 63-77, 2005.

22. N. Lunardon, G. Menardi, N. Torelli, "ROSE: A Package for Binary Imbalanced Learning." https://journal.r-project.org/archive/2014/RJ-2014-008/RJ-2014-008.pdf, 2014.

23. C. Tantithamthavorn, A. E. Hassan, K. Matsumoto, "The Impact of Class Rebalancing Techniques on the Performance and Interpretation of Defect Prediction Models," *Transactions on Software Engineering,* pp. 1-20, 2018.

24. N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research,* vol. 16. pp. 321-357, 2002.

25. "Population Health Methods: Propensity Score." https://www.mailman.columbia.edu/research/population-health-methods/propensity-score, 2018.

26. M. R. Sabuncu, Z. Zhang, "Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels," *Proceedings of the 32nd International Conference on Neural Information Processing Systems,* pp. 8792-8802, 2018.

27. I. Goodfellow, Y. Bengiom A. Courville, *Deep Learning*. MIT Press, 2016.

28. X. Le, H. V. Ho, G. Lee, S. Jung, "Application of Long Short-Term Memory (STM) Neural Network for Flood Forecasting," *Water,* vol. 11, no. 7, 2019.

29. J. E. Nash, J.V. Sutcliffe, "River flow forecasting through conceptual models part I – A discussion of principles," *Journal of Hydrology,* vol. 10, no. 3, pp. 282-290, 1970.

30. S. Bouktif, A. Fiaz, A. Ouni, M. A. Serhani, "Optimal Deep Learning LSTM Model for Electric Load Forecasting using Feature Selection and Genetic Algorithm: Comparison with Machine Learning Approaches," *Energies,* vol. 11, no. 7. 2018.

31. D. Svozil, V. Kvasnicka, J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemometrics and Intelligent Laboratory Systems,* vol. 39. pp.43-62, 1997.

32. B. Everitt, T. Hothorn, *An Introduction to Applied Multivariate Analysis with R*. Springer, 2011.

33. G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, "Self-Normalizing Neural Networks," *Advances in Neural Information Processing Systems*, 2017.

34. T. Devries, G. Taylor, "Improved Regularization of Convolutional Neural Networks with Cutout," *CoRR,* vol. 08, 2017.

35. "Aviation Weather Center." https://www.aviationweather.gov/metar, 2020.

36. Roeder, William. Personal Interview. 2 Aug 2019.

37. "Package 'UBL'." https://cran.r-project.org/web/packages/UBL/UBL.pdf, 2017.

38. "Principal Component Methods in R: Practical Guide." http://www.sthda.com/english /articles/31-principal-component-methods-in-r-practical-guide/118-principal-component -analysis-in-r-prcomp-vs-princomp/, 2017.

39. "keras: R Interface to 'Keras'." https://cran.r-project.org/web/packages/keras/ index.html, 2019.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
26-03-2020	Master's Thesis	Oct 2018- - March 2020

4. TITLE AND SUBTITLE
LIGHTNING PREDICTION FOR SPACE LAUNCH USING MACHINE LEARNING BASED OFF OF ELECTRIC FIELD MILLS AND LIGHTNING DETECTION AND RANGING DATA

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Anson Cheng

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT-ENS-MS-20-M-138

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
45th Weather Squadron
Morrell Operations Center
Cape Canaveral, FL 32920
Comm: 321-853-8484 DSN: 467-8484
Email: 45wsdor@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)
45th Weather Squadron

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES
This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT
Last year in 2019, USAF had 48 projected launches out of CC, twice as much as the current standard, 24. Even with such lofty goals, the USAF fell short in one of its busiest years in space launches, not even reaching 24 launches. KSC and CCAFS are in an area of frequent lightning strikes, which is the leading cause for delayed and cancelled space launches and the main obstacle in meeting launch goals. The 45th WS ensures that any weather safety requirements are met during pre-launch and actual space launch. The 45th WS analyzes 31 EFMs data among other sources to predict lightning occurrence. This study hopes to build upon previous EFM studies while taking a different approach.

15. SUBJECT TERMS
Thesis,typesetting

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Lt. Col. A. Geyer, PHD
U	U	U	U	63	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636, x4630; anson.cheng@afit.edu