Classification of tall tower meteorological variables and forecasting wind speeds in Columbia, Missouri

Sarah Balkissoon[1, 2], Neil Fox[1], Anthony Lupo[1], Sue Ellen Haupt [3], and Stephen G. Penny[4,2]

[1]Atmospheric Science Program, School of Natural Resources, University of Missouri, USA
[2] Cooperative Institute for Research in Environmental Sciences, Boulder, Colorado, USA
[3]Research Applications Lab, National Center for Atmospheric Research, Boulder, Colorado, USA
[4]Sofar Ocean, San Francisco, California, USA

June 20, 2023

Abstract

The wind speeds given in 10-minute intervals is forecast using multiple methods inclusive of persistence, statistical methods of ARIMA as well as artificial intelligence methods of Artificial Neural Networks. Tall tower meteorological variables in Columbia, Missouri are clustered using Self-Organizing Maps after the optimal number of clusters was determined using the Elbow and Silhouette methods among others. The optimal number of clusters, k was given as 4 for all methods. The data were then grouped into three Intervals which consisted of approximately 50 percent and over of vectors or rows from the data frame. These intervals were then used as training and testing for the forecast models of Long Short-Term Memory Networks with pressure and wind speeds as inputs as well as lagged wind speeds as inputs. Other models using these intervals in our analyses include Moving Autoregressive Integrated Moving Average (ARIMA) and persistence. From the results obtained from the ARIMA, the metric of the root mean square error (RMSE) ranged from approximately 0.6 to 1.0 $\mathrm{m\,s^{-1}}$ for forecast horizon 2 to 12 in increments of 2. Interval2 had the upper and lower values and thus showed most variability in errors because it encompassed most of spring, all of summer and the beginning of fall. The moving ARIMA showed lower errors than the LSTM with pressure and wind speeds inputs for all the intervals. This may be attributed to the difficulty in representing the system's non-linearity and high dimensionality by using just the wind speeds and pressure as inputs. The lagged co-ordinates of the wind speed was then examined and used as inputs for the LSTM. The metric used for the evaluation of prediction of the forecast horizons of 60, 120, 180, 240, 300 and 360 minutes or 1, 2, 3, 4, 5 and 6 hours ahead is the Normalized Root Mean Square Error (NRMSE). These models were compared to the benchmark model of persistence. It was determined that all of the models beat persistence and the LSTM with the lag series outperforms the LSTM with pressure and wind speed as inputs. The Moving ARIMA is now beaten by the lagged series LSTM in all intervals for at least 2 time forecast horizons of 60 and 120 minutes or 1 and 2 hours. It is thus shown that the Artificial Neural Network method with the lagged series inputs is the best performing model.

Keywords: Self-Organizing Maps (SOMs), Autoregressive Integrated Moving Average (ARIMA), Long Short-Term Memory (LSTM) Networks

1 Introduction

1.1 Wind speeds

Wind speeds closer to the ground are subjected to resistance and friction. Even though these winds are highly positively correlated with each other, the correlations grow weaker with height as noted in both this study and Cao et al.'s [9]. Due to local surface characteristics and large scale forcing mechanisms such as pressure and temperature differences, wind is one of the most difficult meteorological variables to forecast [28]. Also, since the atmosphere is highly nonlinear and high dimensional, it is especially difficult to forecast this variable in the much needed higher resolutions and longer time horizons [15]. The higher resolution shows more details of the faster variations in wind speeds caused by turbulence and other factors. The importance of such forecasts stems from the ability to aid in the scheduling, dispatching and adjusting electricity reservations [15]. In our work we are looking at short term forecasting at high resolution (10 minute wind speeds at hub height).

1.2 Forecasting of wind speeds

Due to the stochastic nature of wind speeds, forecasting this variable is important for its optimal integration into the power grids [27]. These short term forecasts, can be used by plant managers to adjust turbine components to achieve more efficiency. Another advantage of short term forecasts is the ability to make turbines operable closer to extreme weather events before shutting down. Daily short term forecasts are also important as they relate to the operability of the turbines in terms of their cut-in and cut-off wind speeds as they aid in the reduction of structural damage to infrastructure [9].

There are numerous methods that have been used to forecast wind speed values, some of which are illustrated in Figure 3. Please see the acronyms [1] associated with this chart. The methods incorporated in this paper are Artificial Intelligence (AI) methods which are compared to statistical models as well as the benchmark of persistence. The significant difference between these two methods is that statistical multiple linear regression is written in terms of a set of linear operators whilst Artificial Neural Networks(ANNs) are representative of a linear combination of simple non-linear functions [4]. A mapping is done from random input vectors to output vectors without the assumption that there is a fixed relationship between the two [19]. ANNs have the ability to learn from past data by recognizing patterns among the observations and using these to forecast into the future [19]. Research has indicated the superiority in prediction accuracy of ANNs to statistical regression especially as the non-linearity of the problem increases [4]. Previous studies, namely [2] and [3], done in Missouri, found the wind speeds to be chaotic in nature, hence motivating the choice of this method to address the complexity and non-linearity of the data.

1.3 Wind Power

From the relationship $P = \frac{1}{2}\rho AV^3$ where P is the available power at the turbine,ρ is the density of air, A is the area swept by the turbine and V is the wind speed, the two meteorological variables which determines the available turbine power are ρ and V. The latter variable has the greater influence as the power varies as the cube of V. The air density is dependent on pressure and temperature as seen from the following equation [21]

$$\rho = D\left(\frac{273.15}{T}\right)\left[\frac{B - 0.3783e}{760}\right] \tag{1}$$

where D is $1.168kg/m^3$ - the density of dry air at standard atmospheric temperature ($25°C$) and pressure ($100kPa$) and B is the barometric pressure in torr, e is the moist air vapour pressure in torr. Hence, as seen in subsequent sections of the methods, these two meteorological parameters, will be considered, together with wind speeds, when determining the inputs to the Neural Network.

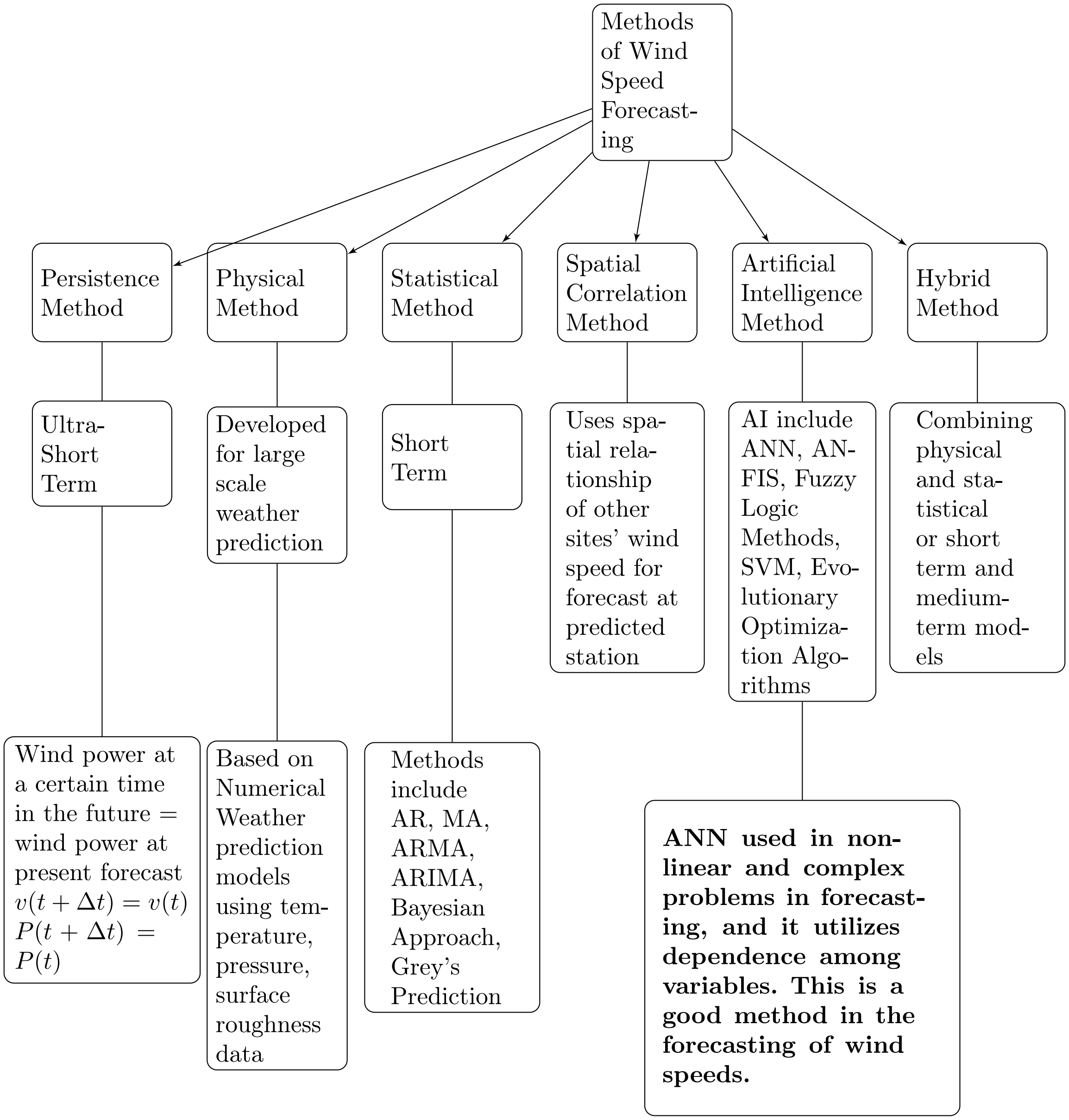
[1] AI-Artificial Intelligence, ANN-Artificial Neural Networks, ANFIS-Adaptive Neuro-Fuzzy Interface System, SVM-Support Vector Machine, AR-Auto-Regressive, MA-Moving Average, ARMA-Auto-Regressive Moving Average, ARIMA-Auto-Regressive Integrated Moving Average

1.4 Literature Review

From [10], comparative analysis of ARIMA and LSTM models in predicting hourly wind speeds indicated the RMSE to be less than of the ARIMA method. An analysis of the existing literature and the studies done by [10] showed that for smaller datasets, ARIMA performed better whilst for larger datasets, deep learning techniques such as LSTM outperforms the statistical methods such as ARIMA. Another study done by [33] forecasts 10-minute wind speeds as done in our study, but uses fuzzy set theory to conduct attribute reduction of the factors affecting the wind speeds instead of the magnitude correlation plot. This determined what were the inputs into the LSTM model. This simplified input improves upon the accuracy as well as the speed of the model [33]. In [14], dimensionality reduction of the meteorological data that affects the wind speed is also conducted. They instead utilizes principal component analysis (PCA). It was found that this case had the most improvements in terms of errors when compared to models' inputs of the historical wind speed data and the other exogenous variables. In reference [13], the authors compared the results of two models, LSTM and Support-Vector Machine (SVM) in the prediction of wind speeds. It was determined that of the two algorithms the LSTM incurred the lower RMSE than that of the SVM due to the LSTM's ability to remember patterns for a longer duration. [1] showed the superiority in model performance of hybrid simulations. This study, using the metric of RMSE, determined that the LSTM-ARIMA model had less forecasting errors when compared to the LSTM and SVM models. Another paper that incorporates clustering to select the training samples before feeding them to the LSTM is [31]. However, this is done for day-ahead forecasting. They utilized a density based spatial clustering (DBSCAN), deep feature extraction and LSTM forecasting in their study. Their proposed model out performed the benchmark methods of random forest (RF), least square support vector regression (LSSVR) and back propagation neural network (BPNN), by at least 17% [31]. In another comparative study of forecasting hourly wind speeds for a year, Neural Networks was compared with the statistical method of ARMA in which the ARMA was outperformed by all the other methods; the feed forward neural network (FNN), recurrent neural network (RNN), LSTM and the gated recurrent unit (GRU)[34]. The ANN which are utilized for the purpose of short term forecasting of wind speeds usually perform better than the time series methods with a few exceptions [34]. The authors considered the variables that were also investigated in this study in the prediction of the target variable; these are wind direction, wind speed with one time step lag, pressure and temperature. After various permutations of the inputs, it was found that the wind speed with the one time lag had the largest correlation with the wind speed and as such it was deemed the most important feature [34]. In [12],the variations of inputs into a LSTM as well as a 1D-CNN were also investigated.

The objective of this paper is to apply multiple forecasting techniques of tall tower data in Missouri; persistence as a benchmark, statistical methods of ARIMA and ANN techniques in clustering and subsequently forecasting using LSTM. The novelty of this work includes the usage of the competitive learning Neural Network, SOMs, in the clustering of the data with similar patterns which are then the inputs into the LSTM. This feature of data mining, clustering of data, allow for the preprocessing of the data and thus the accurate development of forecasting models [13] [31]. The forecasting of tall tower data in Columbia, MO for this scale is carried out; short term forecasts at high resolution is conducted. The proposed methodology is tested using real -world tall tower data and it outperforms other known prediction methods. The subsequent structure of the paper is as follows. Section 2 gives the data used in this study, section 3 introduces the concepts of the methods incorporated in this paper, section 4 discusses the results whilst sections 5 gives the future analyses that can be conducted and the conclusion of the paper collectively.

Figure 1: Methods for wind speed forecasting


Methods of Wind Speed Forecasting
Persistence Method
Physical Method
Statistical Method
Spatial Correlation Method
Artificial Intelligence Method
Hybrid Method
Ultra-Short Term
Developed for large scale weather prediction
Short Term
Uses spatial relationship of other sites' wind speed for forecast at predicted station
AI include ANN, ANFIS, Fuzzy Logic Methods, SVM, Evolutionary Optimization Algorithms
Combining physical and statistical or short term and medium-term models
Wind power at a certain time in the future = wind power at present forecast $v(t+\Delta t) = v(t)$ $P(t+\Delta t) = P(t)$
Based on Numerical Weather prediction models using temperature, pressure, surface roughness data
Methods include AR, MA, ARMA, ARIMA, Bayesian Approach, Grey's Prediction
ANN used in non-linear and complex problems in forecasting, and it utilizes dependence among variables. This is a good method in the forecasting of wind speeds.

2 Data

Columbia, Missouri is located in 038°53.270′N latitude and 092°15.820′W longitude and has a site elevation of 255m as seen in Figure 2. Ten-minute tall tower wind speed, wind direction and temperature data in 2009 from this region were used in our study [11]. The respective units are $\mathrm{m\,s^{-1}}$, degrees and degrees Celsius, respectively. The anemometer orientations were 120°and 300°for the tall tower height of 68 m. Channels 1 and 2 represent the respective wind speed times series. The larger of the wind speed values at each time step were taken and labelled as Max1. The wind direction time series at this height level was given from channel 7 and sine of these angles was labelled as Direction1. The temperature time series from 2 m logger height were also utilized in our analyses, taken from channel 11, it is referred to as Temp. Hourly maximum pressure data was taken from University of Missouri, Extension's Missouri Historical Agricultural Weather Database. Each hourly pressure value was repeated five times. This time series, labelled as Pressure, along with Max1, Direction1 and Temp were combined and used in all of the analyses for Columbia68 as detailed in the Methods section below.

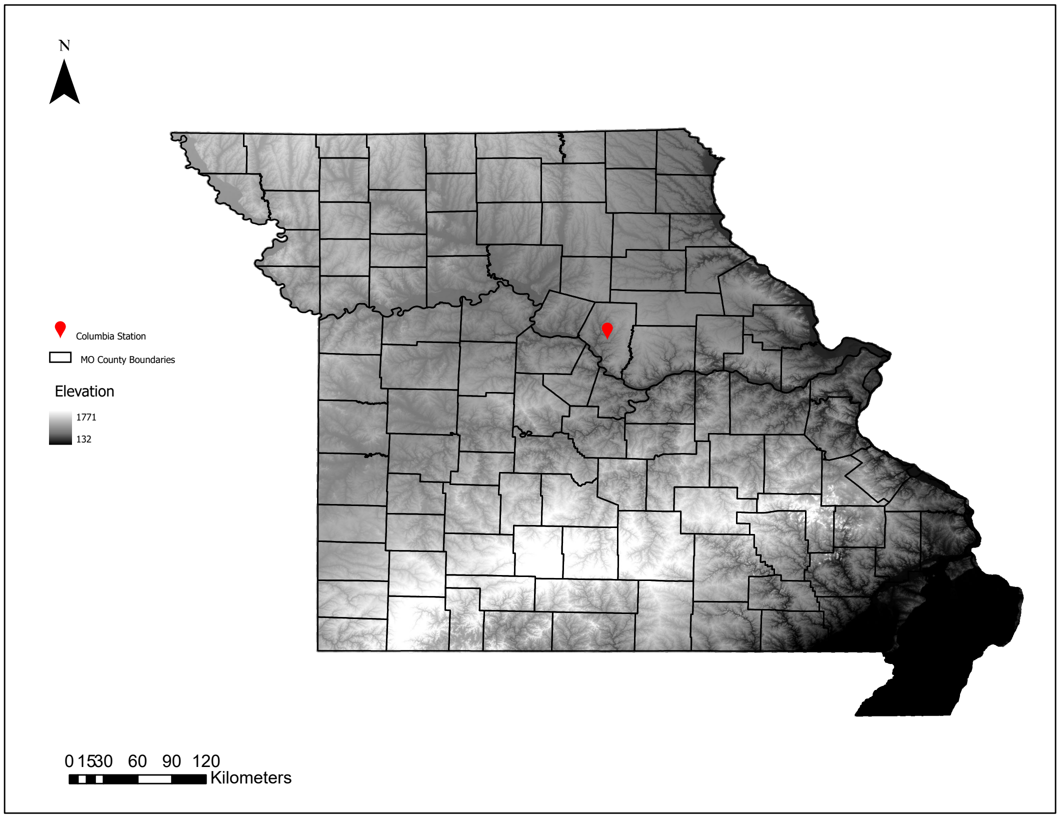


Figure 2: Tall tower location

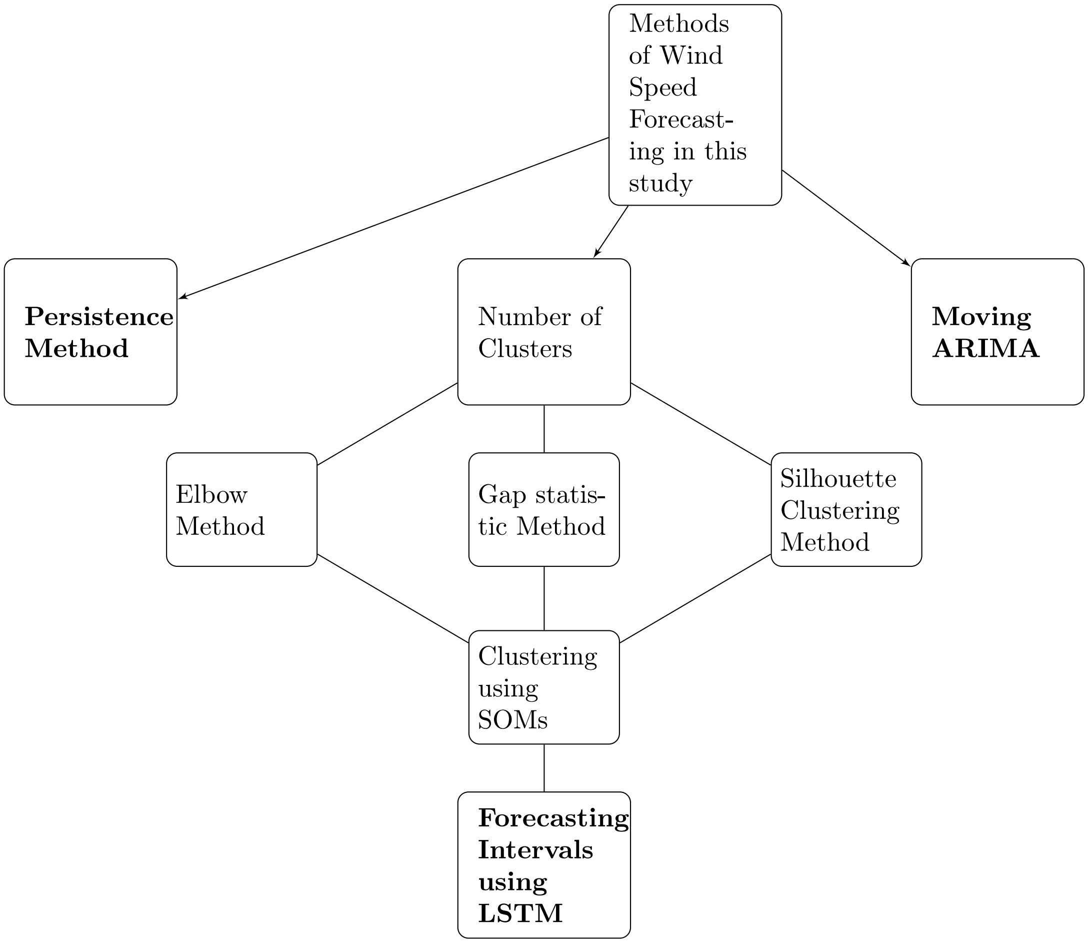
3 Methods

Artificial Neural Networks (ANN) have to date been applied to a multitude of fields in solving complex problems. These data-driven models are utilized especially as physically-based mathematical models are difficult to construct given the high non-linearity of natural systems [20]. As defined by Basheer and Hajmeer [4] and Ramasamy et al. [25] , ANNs can be considered as a system of densely interconnected processing elements, also called artificial neurons or nodes, which have the ability to conduct parallel computations of input data .Complex relationships are derived from the input and the output. The input variables are multiplied by weights and biases are added to these products. These are then passed through transfer functions for the generation of the outputs [25].

ANNs, being abstractions of biological systems, have the advantages of processing data that are highly nonlinear. These robust systems have the ability to learn and generalize imprecise and fuzzy data [4]. Data are allowed to be processed faster and have a better fit amidst inaccuracies from measurement errors. The system in its learning is self updating and has the ability to unlearn data as well [4].

There are many applications of ANNs, which include modelling, classification, pattern recognition and multivariate data analysis problems [4] [25]. Here, we will focus our attention on the clustering of the data into various clusters and then on subsequent modelling and forecasting. Clustering as described by [4], is formed by investigating the similarities and differences of the inputs based on their inter-correlations. Kohonen networks or Self-Organizing Feature Maps (SOMs), the unsupervised learning ANN where the actual values are not required for the training set, are used in this study. Forecasting is also done by training the ANN using a training set of historical data. A Recurrent Neural Network (RNN) is utilized especially for its dynamic memory capabilities where the outputs of neurons are fed as inputs to the same neurons or other neurons in the preceding layers [4]. A general overview of the methods incorporated in this study is depicted in the flow diagram below. More details of these methods are given in the following sub-sections.

Figure 3: Methods for wind speed forecasting utilized in this study


Methods of Wind Speed Forecasting in this study
Persistence Method
Number of Clusters
Moving ARIMA
Elbow Method
Gap statistic Method
Silhouette Clustering Method
Clustering using SOMs
Forecasting Intervals using LSTM

3.1 Methods determining the number of clusters

Clustering analyses as mentioned in [22], are statistical methods which are used to partition multivariate data into subsets. There are numerous methods that can be used to determine the optimal number of clusters to classify the data into relatively homogeneous groups, such as the Elbow Method, Silhouette Analysis and Gap Statistic. These methods, which are used in this study, are outlined below. We have considered in the analyses this multivariate data set (of length n) of variables wind speed, direction, temperature and pressure. We denote these points as x_i for $i = 1, \ldots, n$.

The Elbow Method is determined by plotting the within-cluster sum of squares (WCSS) against the number of clusters, say k. The $WCSS(k)$ gives the sum of the squared distances between each data point, say x_i, in all clusters and their associated centroids denoted as $\bar{x_j}$ (which is the geometric center or the arithmetic mean position of all the points in the plane figure). This can be written as follows.

$$WCSS(k) = \sum_{j=1}^{k} \sum_{x_i \varepsilon clusterj} \|x_i - \bar{x_j}\|^2 \tag{2}$$

The changes in $WCSS$ with a range of k determines the optimal number of clusters in accordance with the Elbow method. The value in which k elbows or the point where the rate of decrease in $WCSS$ is relatively minimal when compared to its previous k values.

The Silhouette Clustering method was also used in our analysis. This method examines the within cluster-consistency by comparing how similar objects from a cluster are to another. Its value, $S(i)$ range from -1 to 1 where the lower end of the interval indicates that the configuration has too much or few clusters. The closer this value is to 1 however, is indicative of an object that is well matched to its cluster or poorly matched to the other clusters.

The mean similarity of point i and all other points in the same cluster, C_i, is given by equation 3 where $|C_i|$ denotes the number of elements in C_i and $d(i,j)$ give the distances between data points i and j in cluster C_i. In this average the distances, $d(i,i)$ are not considered hence the consideration of $|C_i| - 1$ in the formulation below.

$$a_i = \frac{1}{|C_i| - 1} \sum_{j \varepsilon C_i, i \neq j} d(i,j) \tag{3}$$

The smallest mean dissimilarity of point i and all the other points of another cluster, C_k, is given by b_i in equation 4.

$$b_i = min_{k \neq i} \frac{1}{|C_k|} \sum_{j \varepsilon C_k} d(i,j) \tag{4}$$

This is the second best fit cluster for point i based on the distance metric. The Silhouette value for point i, we define as S_i, is given in terms of a_i and b_i as seen in equation 5.

$$S_i = \begin{cases} \frac{(b_i)-(a_i)}{max\{a_i, b_i\}} & \text{if } |C_i| > 1 \\ 0 & \text{if } |C_i| = 1 \end{cases} \tag{5}$$

This can be further simplified as seen below, depending on the inequality relations between the mean similarity and dissimilarity.

$$S_i = \begin{cases} 1 - \frac{a_i}{b_i} & \text{if } a_i < b_i \\ 0 & \text{if } a_i = b_i \\ \frac{b_i}{a_i} - 1 & \text{if } a_i > b_i \end{cases} \tag{6}$$

The Gap Statistic, another consideration used in this paper, is outlined as follows. As previously denoted, let C_i be the i^{th} Cluster and $|C_i|$ be representative of the number of elements in this cluster. Let the pairwise distances between elements say i and j in C_i, d_i, be given by equation 7.

$$d_i = \sum_{i,j \varepsilon C_i} d(i,j) \tag{7}$$

For a given number of clusters k, the within cluster distance for that particular partitioning P_k, is given by equation 8. A better classification is indicative of a smaller W_k value.

$$W_k = \sum_{i=1}^{k} \frac{1}{2\,|C_i|} d_i \tag{8}$$

Considering the data in which the 'true' number of clusters is given by G, W_k should drop as k increases until it reaches G where it will decrease at a much slower rate. Thus, there will be an 'elbow' point in W_k; this k value corresponds to the optimal number of clusters. The Gap Method is used to compare the original data with the expected curve, $E_n^* \{log(W_k)\}$ where E_n^* gives the expectation of sample n from the reference distribution. The Gap Statistic is the value of k which maximizes $Gap_n(k)$ or from equation 9, the cluster value where W_k is at the furthest distance from the expected curve [32].

$$Gap_n(k) = E_n^* \{log(W_k)\} - log(W_k) \tag{9}$$

3.2 Self Organizing Maps (SOM)

A Self Organizing Map (SOM) is an unsupervised clustering method as there is no additional information being supplied to the model by a 'supervisor' [16]. In this model, high-dimensional data sets are reduced to the two-dimensional map in which nodes with most similarity are nearest to each other and vice-versa [23]. It does this dimensionality reduction via the usage of cluster centers which can then be interpreted as an 'abstract representation' of any given vector from that particular cluster [16]. It preserves topology where vectors that are near in input space are also mapped to nearby neurons in the SOM [22][23]. This resulting map is a projection of a multidimensional space rather than a geographical space [23]. There are two modes of operation, training which builds the map using input examples through a method called vector quantization and mapping which classifies new input vectors [7].

This Kohonen Neural Network is used in many applications [30] such as Pearce et al.'s air quality classifications [23] and in geoscience for the extraction of climate and atmospheric circulation patterns [16]. Previous studies using SOMs also include Berkovic's [5] determination of the wind regimes, choosing from various map sizes, the number of nodes in the rows and columns. However in our study, since we utilized SOMs for the purposes for clustering our data to be later incorporated in our forecasting algorithm, we defined our map size based on the formulation written in [29]. The number of neurons, M of the map is determined from the number of observations, N. It is given by the following expression [7].

$$M \approx 5\sqrt{N} \tag{10}$$

According to [6], the methodology of the SOM can be achieved via the processes of competition- where the Best Match Unit (BMU) is identified, cooperation -where the topological neighbourhood of the 'excited' neurons are identified and finally adaptation -where BMU and excited neurons are updated in accordance to the input vector.

In more detail the methodology of the SOM is as follows [29].

1. The weight vector of each of the neurons in the map is initialized randomly.

2. The training observed data, say x_t, is 'passed' to the map as an input vector and Euclidean distance between the all the neurons and this vector is calculated. The neuron with the smallest distance is termed the Best Matching Unit or (BMU). For each input observation, the BMU is identified. We denoted this unit as c henceforth.

3. A neighbourhood of c is selected and using a neighbourhood function given by h_{ci}, the weighted vectors of the neighbouring neurons, i are updated.

$$h_{ci}(t) = a(t)e^{-\frac{\|r_c - r_i\|^2}{2R^2(t)}} \tag{11}$$

$$W_i(t+1) = W_i(t) + h_{ci}(t)\left[x_t - W_i(t)\right] \tag{12}$$

Where, from equation 11, h_{ci} is the neighbouring function and t is an index of iteration, $a(t)$ is the learning rate, r_c is vector of c, r_i is the vector of the neuron i and R is the radius around c. This function is a monotonically decreasing function of t as the learning rate decreases with the iterations during the training process and the radii around c decreases with t. This process ensures that neurons i closest to c are being adjusted the most.

The neurons are updated in accordance to equation 12 where $W_i(t+1)$ and $W_i(t)$ represent the weighted vector of neuron i at the $t+1$ and t indices of iterations respectively, h_{ci} is the neighbourhood function above and x_t is the observed input vector.

4. This process is repeated in the iterative training until the clusters are identified based on their distances.

The data described in Section 4.2, were normalized between 0 and 1 by subtracting from each element in that particular column of the data frame, its mean. These values are then divided by the standard deviation of the column to give the z or standard scores. This standardizing of the variables was done using the scale command. These analyses were done in R studio [30]. The SOM grid was then created using the relation of 10 where $N = 52,560$ data observations for each variable. The grid size used was 41 by 28 of hexagonal nodes corresponding to the factor pair of 1,148. This value was used instead of the calculated numeric of 1,146 because it had more factor pairs.

The following is a list of the metrics to be plotted and their description will be shown in the results section.

1. Node Count- This map gives the number of samples that are mapped to each of the nodes of the map. This value should be relatively uniform throughout the SOM. Large values in some areas of the map is indicative for the need of a larger map whilst empty nodes indicates that a smaller map may be more appropriate. Generally, it is used to determine high density areas in the map where ideally there should be a homogeneous distribution [24].

2. Neighbourhood Distance or U-Matrix- This map gives the distance between each node and its neigbouring neurons. It represents the Euclidean distance amongst the codebook vectors of the respective neighbourhoods [24]. Larger distances indicates dissimilarities and thus cluster boundaries as nodes from the same cluster have the tendency to be closer.

3. Heat Maps - These maps separately give the distribution of each of the parameters throughout the map. These are done for the four variables, both scaled and unscaled.

4. Clustering of the codebook vectors - This map consists of the codebook vectors which is the data structure that carries the neuron's weight vector in a 2D grid. The number of clusters or groups is input as well as the specification to add the cluster boundaries.

After the clusters are identified, the cluster associated with each of the $x(t)$ vectors was determined. Continuous intervals of the clustered 2009 Columbia, MO data set, representing approximately 50% and over of data points in that particular cluster, were established. There were three intervals in which the majority of the vectors or rows from the data frame belonged to two of the identified four clusters, denoted Cluster1 to Cluster4 (we will explain more of this in the subsequent results section). For example, Interval1 ranged from 1 to 16,000 rows in which Cluster3 consisted of 50.2% of the vectors. Interval2 which started at 16,001 and ended 40,500 inclusively, comprised 78.92% of rows from Cluster2. Interval3 included vectors from 40,501 to 52,500 in which Cluster3 represented approximately 48% of this interval. It should be noted that there were predominately two clusters which we will also show in subsequent results of the clustering of the codebook vectors. Another note to mention is that the entire time series of length was not used. Instead, 52,500 rows were utilized in our analyses. There were 16,000, 24,500, and 12,000 points in Cluster1, Cluster2 and Cluster3 respectively.

These intervals are then separately trained and tested in time series forecasting using the Recurrent Neural Network explained in the subsequent subsection.

3.3 Recurrent Neural Networks (RNN), Long Short- Term Memory Networks (LSTM)

Recurrent Neural Networks (RNN) allow information to persist via one or more hidden states and loops that pass information from one step to another of the network. However, for this, there exists the vanishing gradient problem as the gradients asymptotically reduce to 0 from the repeated multiplication of weights for various time steps. Long Short-Term Memory networks (LSTMs) are a special type of RNN that can learn these long-term dependencies. The LSTM has memory blocks called cells where information is stored in the cell state, c_t and the hidden state, h_t. A diagrammatic representation of the architecture of such memory blocks or cells is seen in Figure 4. Information is regulated by gates by optionally allowing certain data through using sigmoid and tanh activation functions. The output of the sigmoid function is a number between 0 and 1, where 0 and 1 mean no and all information goes to the cell state, respectively. Generally notated, the inputs to the gates are the output hidden state from the previous step , h_{t-1}, and the output cell state from the previous step, c_{t-1} and current input, x_t, which are pointwise multiplied by weight matrices, W, and then added to a bias, b.

There are three major gates: the forget, the input, and the output gates.

1. The forget gate: As seen in Figure 4, the input of this gate is x_t and h_{t-1} for that time step. These inputs are multiplied by weight matricies and added to a bias. This value is then inputted to the sigmoid function and a vector is outputted which corresponds to each value in the cell state, c_{t-1}. Please refer to equation 13. This vector output is multiplied to the cell state. If a 0 is output from the sigmoid function for a particular value, the forget gate wants the cell state to disregard that information whilst if 1 is the sigmoid output, the forget gate wants the cell state to remember this data.

$$f_t = \sigma \left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{13}$$

2. The input gate: The gate determines the information being stored in the cell state. The sigmoid layer decides the data to be updated and the tanh layer, whose output values ranges from -1 to 1, creates a vector of possible values that could be added to the cell state. Please refer to equations 14 and 15.

$$i_t = \sigma \left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{14}$$

$$g_t = \tanh \left(W_g \cdot [h_{t-1}, x_t] + b_g\right) \tag{15}$$

The old cell state, c_{t-1} is then used to update the new cell state c_t. This is done representatively by equation 16. The old state is multiplied by f_t to forget the information decided upon earlier and then it is added to the product of i_t and g_t which is indicative of the new possible values scaled to the update amount decided upon for each value. Note that $*$ is representative of the Hadamard or entrywise product.

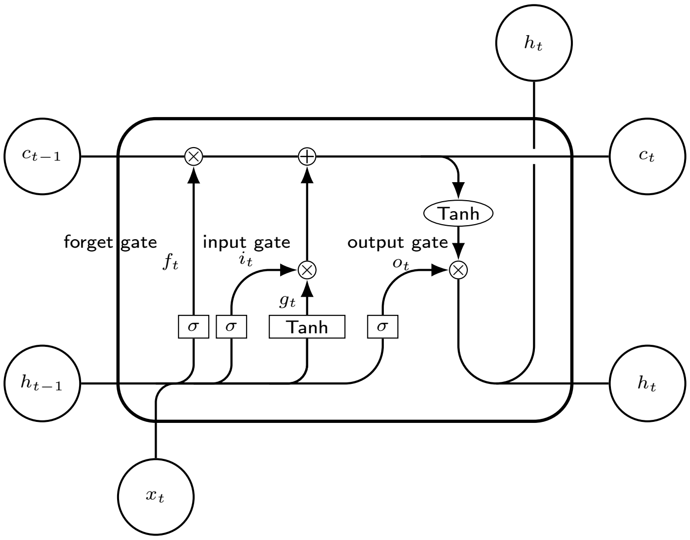
$$c_t = (f_t * c_{t-1}) + (i_t * g_t) \tag{16}$$

3. The output gate: A vector is created from scaling the values in the cell state using a tanh function. The sigmoid function is once again used as a filter to regulate what is to be outputted from the vector mentioned previously. This can be represented by equation 17. This is sent as the output and as the hidden state of the next cell.

$$o_t = \sigma \left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \tag{17}$$

$$h_t = o_t * \tanh \left(c_t\right) \tag{18}$$

Figure 4: LSTM Architecture



3.4 Moving AutoRegressive Integrated Moving Average Method (ARIMA)

A moving AutoRegressive Integrated Moving Average Method (ARIMA) is used as another model in our analysis. This is a statistical method which uses the relationship within the time series data in its construction. Data cannot be white noise, that is, purely random with mean = 0 and standard deviation being a constant as forecasting into the future would not be possible. If this condition is met, AutoRegressive, AR(p), Moving Average, MA(q) and AutoRegressive Moving Average, ARMA(p,q) methods can be utilized. If the data are not stationary (that is not constant mean and variances), differencing needs to be performed. An AutoRegressive Integrated Moving Average, ARIMA (p,d,q) can be used where the Integrating part represents the d or the differencing factor.

The AR method, a time series model, is regressed from its previous values up to an order determined by the p parameter. This can be seen mathematically from equation 19. The Partial Autocorrelation function (PACF) determines how many lags are to be incorporated in the AR method; large PACF values gives the order of the model. For lag p, the relationship between x_t and x_{t-p} is determined, filtering all the intermediate linear influence from $x_{t-1}, x_{t-2}, \ldots, x_{t-(p-1)}$.

$$x_t = \beta_0 + \beta_1 x_{t-1} + \beta_2 x_{t-2} + \ldots + \beta_p x_{t-p} + \varepsilon_t \tag{19}$$

Where x_t and x_{t-1}, ...,x_{t-p} are the current and previous values respectively and β_0 is a constant term and β_1,...,β_p are the coefficient representing what part of x_{t-1},..., x_{t-p} are relevant in explaining the current value etc.

The MA model is written in terms of a linear combination of past error. It gives the extent the series is related to its past errors. Generally it can written as equation 20. The Autocorrelation function determines the number of lags for the MA model. It is given by the lag value which is statistically different from 0 and above the error band, followed by consecutive insignificant ACF values for subsequent lags.

$$x_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q} \tag{20}$$

Where x_t is the current value, ε_t and $\varepsilon_{t-1}, \ldots, \varepsilon_{t-q}$ are errors from the current and previous predictions respectively and $\theta_1, \ldots, \theta_q$ represent the corresponding part which is relevant in explaining the current value.

The ARMA method is the linear combination of the linear models, AR and MA as such they too are linear models. This method thus, takes into account past values and errors in its formulation. Generally it can be written as equation 21.

$$x_t = \beta_0 + \beta_1 x_{t-1} + \beta_2 x_{t-2} + \ldots + \beta_p x_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q} \tag{21}$$

The differencing parameter,d is introduced in the ARIMA models to remove trends and seasonality. The first order difference is given by $\Delta_1 x_t = x_t - x_{t-1}$. This and higher orders can be written in terms of B, the backward shift operator where $Bx_t = x_{t-1}$ and $B\left(Bx_t\right) = B\left(x_{t-1}\right) = x_{t-2}$. Generally for shifting an observation some m periods, $B^m x_t = x_{t-m}$. Thus, the first and second differences in terms of operator B are $\Delta_1 x_t = x_t - x_{t-1} = x_t - Bx_t = \left(1-B\right) x_t$ and $\Delta_2 x_t = \Delta_1 x_t - \Delta_1 x_{t-1} = \left(1-B\right)^2 x_t$ respectively. The second difference can be shown to be via expansion, $x_{t-2} - 2x_{t-1} + x_t$.

To determine the number of differencing to use we examine the autocorrelations. If the series has positive autocorrelations out to a large number of lags then the series may need differencing. If for lag 1, the autocorrelation is zero or negative then the series does not need higher order differencing. However, if for lag 1 the autocorrelation is less than or equal to -0.5, then the series may be over-differenced. A model with no differencing implies that the series is stationary whilst the assumptions are made that for the first and second differencing of the series, the original series has a constant average trend and has time varying trends respectively. An ARIMA(1,1,0), ARIMA(0,1,1) and ARIMA(1,1,1) can be written mathematically as equations 22a and 22b, 23a and 23b, 24a and 24b respectively.

$$\Delta_1 x_t = \beta_0 + \beta_1 \Delta_1 x_{t-1} \tag{22a}$$

$$\Rightarrow x_t = \beta_0 + x_{t-1} + \beta_1 \left(x_{t-1} - x_{t-2}\right) \tag{22b}$$

$$\Delta x_t = c + \Theta_1 \varepsilon_{t-1} \tag{23a}$$

$$\Rightarrow x_t = c + x_{t-1} + \Theta_1 \varepsilon_{t-1} \tag{23b}$$

$$\Delta x_t = \beta_0 + \beta_1 \Delta_1 x_{t-1} + \Theta_1 \varepsilon_{t-1} \tag{24a}$$

$$\Rightarrow x_t = \beta_0 + x_{t-1} + \beta_1 \left(x_{t-1} - x_{t-2}\right) + \Theta_1 \varepsilon_{t-1} \tag{24b}$$

3.5 Model Configuration

Model: LSTM (pressure and wind speeds as inputs) The Pytorch structure of the codes for this model was motivated/developed by [17].

- The data were loaded, preprocessed (by taking the larger wind speed of the orientations at each time step) and plotted.

- The target variable was specified as wind speed along with the forecast lead (how much we are forecasting ahead, h). The target was specified as the lag/shift of the wind speed by the forecast lead. The features were given as wind speed and pressure. The data were then split into the training (75%) and testing (25%) sets from the observations. The train and test data were then standardized where the values are not restricted to a particular bounding range like normalization.

- A sequence of observations from the train and test set were constructed. This sequence was given as a block of data from some i^{th} row - sequence length through row i. For i less than the sequence length, the 1^{st} row was padded by repeating it as many times deemed necessary. Thus, the outputs have the number of rows in the block equal to the sequence length.

- These sequences from data set was set in Pytorch's dataloader to select minibatches. However, in our model the batch sizes selected were the entire respective train and test data sets for the Intervals. Thus we had two features (columns), fifty sequence length (rows) and one batch the length of the train and test sets.

- A shallow regression LSTM model was then utilized with one hidden layer of 100 hidden units. The loss function is used to calculate the error or the difference between the predicted and the actual values. The loss function chosen was Mean Square Error (MSE). The optimizer is used to make changes to the weights; it does this to try to lower the model loss function. The optimizer chosen was the Adaptive Moment Estimation (Adam) algorithm with a learning rate of 0.01. An epoch is the number of times the algorithm traverses the training data. The model was trained using 20 epochs and was then evaluated.

4 Results

From Figure 5, we can see that the elbow occurs at 4, indicative that this is the optimal k. In figure 6, 4 has the largest $S(i)$ value indicating that for $k = 4$, the objects are well matched to their respective clusters. Similarly, from Figure 7, the value which maximizes $Gap_n(k)$ is $k = 4$. From the analysis of multiple methods, the bar chart in Figure 8 indicates that most of the methods result in an optimal k of 4. This is an important consideration, as mentioned in Pearce et al. [23], because a grid with too few classes losses important information via generalizations whilst too many classes will result in loss of statistical power as there will exist smaller within class sample sizes.

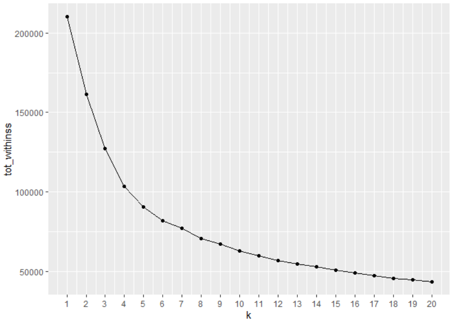


Figure 5: Elbow Method showing the optimal number of clusters of the data to be 4

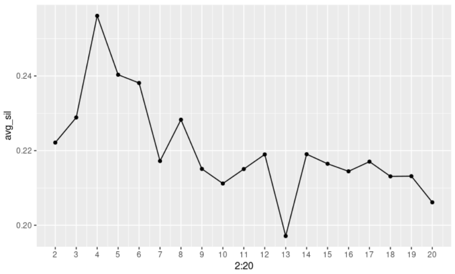


Figure 6: Silhouette Clustering Method showing the optimal number of clusters of the data to be 4

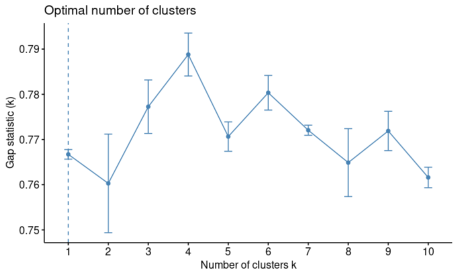


Figure 7: Gap Statistic showing the optimal number of clusters of the data to be 4

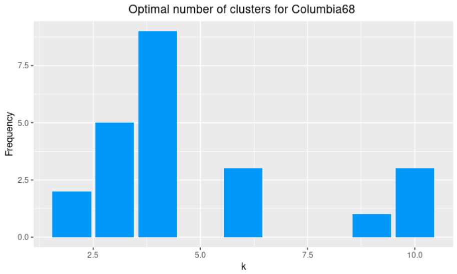


Figure 8: Methods determining optimal k show that optimally, from most methods, that $k = 4$

The grid was a hexagonal structure consisting of 1148 nodes. This structure consisted of no x and y axes but rather nodal positions, which were numbered as bottom left having the least value, whose node numbering increases from left to right [18]. As mentioned in [23], limitations of SOMs include its grid having a finite structure, which imposes restrictions on the map in the provision of precise information on clustering dissimilarity. Another restriction is using set of numbers to define the grid that in turn generalize its shape, be it a rectangle or a square [23].

From the results of the SOMs, the node count plot can be seen in Figure 9. Since the distribution of the counts is relatively uniform throughout the domain of the SOM, the map size is appropriate. Figure 10 shows the neighbourhood distance in which cluster boundaries can be identified via large nodal distances. From this map, it is evident that there exist areas where there are greater distances representative of the upper end of the scale and the lighter colours. This is seen for example in the north eastern portion of the map. From the clustering of the codebook vectors in Figure 15, we do note that this is separated as part of a cluster. This is contained in a smaller cluster whilst there are two major clusters where the adjacent nodes are grouped in the same cluster. This grid also shows, for each node, all the variables (as colour coded) in various sector representations. The radii of the sectors varies with respect to its variable [18]. The unscaled heat map for all of the variables used in this study are seen from Figures 11 to 14.

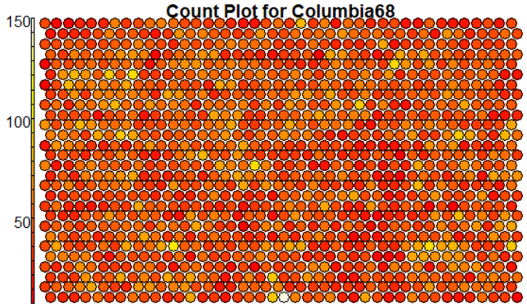


Figure 9: Node Count Plot showing the homogeneous distribution of the samples on the map

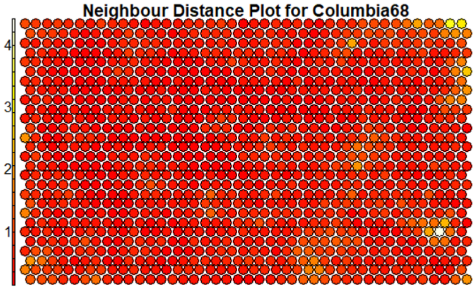


Figure 10: Neighbourhood Distance or U-Matrix showing the distance between each node and its neighbouring neurons

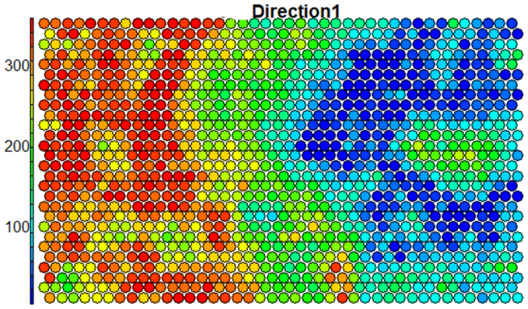


Figure 11: Heat Map showing the unscaled distribution of the wind direction values throughout the map

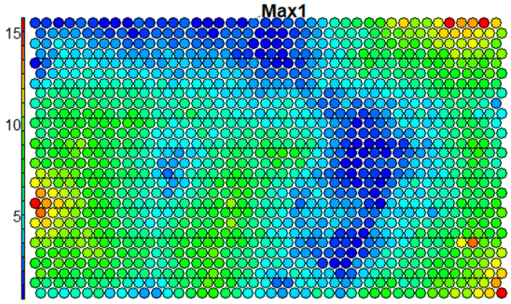


Figure 12: Heat Map showing the unscaled distribution of the wind speed values throughout the map

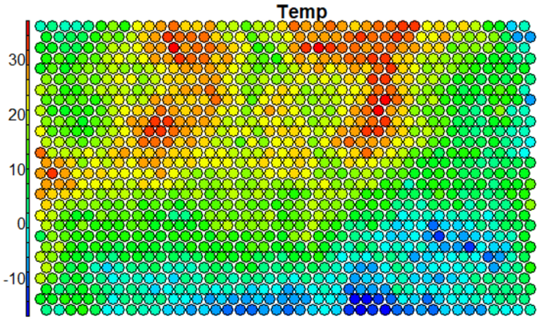


Figure 13: Heat Map showing the unscaled distribution of the temperature values throughout the map

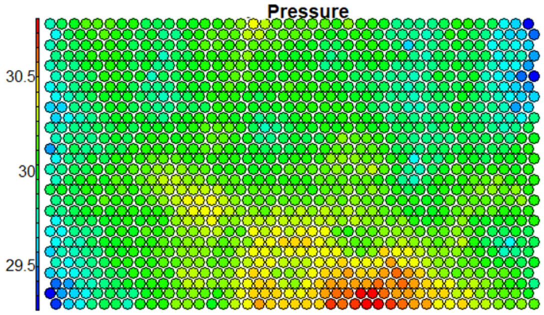


Figure 14: Heat Map showing the unscaled distribution of the pressure values throughout the map

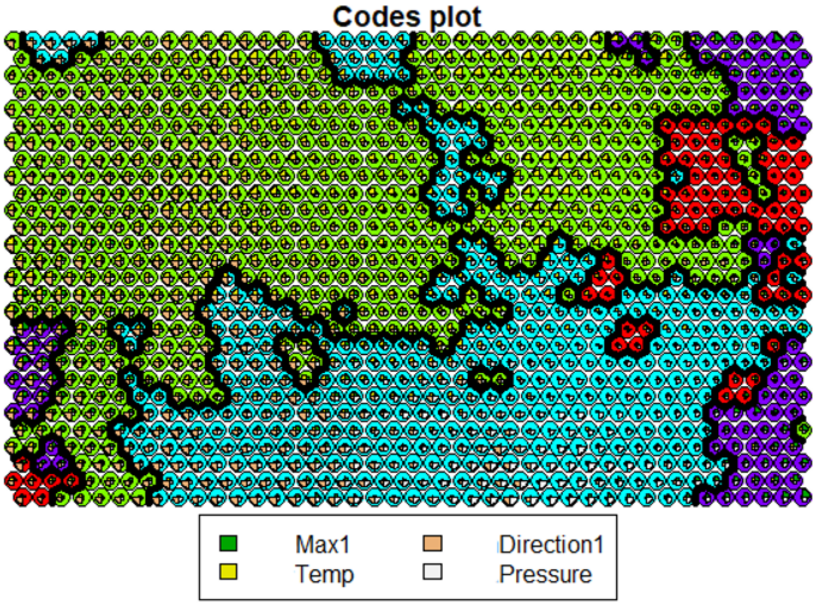


Figure 15: Clustering of codebook vectors into the optimal 4 clusters identified showing there are 2 dominant clusters

From the clusters of the SOMs, continuous intervals belonging to a particular cluster were identified. These intervals are representative of approximately 50% and more of the rows from the data frame belonging to a particular cluster where Interval1, Interval2 and Interval3 belonging to Cluster3, Cluster2, and Cluster3 ranged from 1 to 16,000, 16,001 to 40,500 and 40,501 to 52,500 rows respectively. The three intervals identified by our clustering are graphed in Figures 16, 17 and 18 where both the test and training sets are visualized. These clusters from the SOM were utilized to optimize model performance in forecasting as done in Browell et al.'s article [6]. The forecast horizon is from 20 minutes to 2 hours. As mentioned in [6], for these time scales which are used to balance the power systems by operators, statistical methodologies inclusive of ARIMA are superior to that of results obtained from Numerical Weather Predictions (NWP). This can be attributed to its low computational cost and ease of including of new data [6].

From table 1, the RMSE and the MAE for these intervals and various time steps, h, using the moving ARIMA model, can be seen. These values ranged from approximately 0.6 to 1.0 $\mathrm{m\,s^{-1}}$. These results are somewhat comparable to that of [6]. Browell et al. [6] used vector autoregression in the spatial consideration of multiple locations and for this model they obtained RMSEs of 0.96, 1.55, 2.00 $\mathrm{m\,s^{-1}}$ for one, three and six hours ahead. Another study by [27] using both hourly and

10 minute data in which 39 and 173 points were forecasted respectively for each data set, have RMSEs of 1.27 $\mathrm{m\,s^{-1}}$ for the hourly dataset and 0.96 $\mathrm{m\,s^{-1}}$ for the 10 minute dataset. For our time step or forecast horizon of h equal to 6 (one hour ahead), for Interval1 and Interval3 this value was approximately 0.8 $\mathrm{m\,s^{-1}}$ whilst for Interval2, it was an estimated 0.1 $\mathrm{m\,s^{-1}}$ less that the other two intervals.

In our analyses, the upper and lower values from this range resulted from the run of Interval2. This is expected as this interval encompassed most of the spring, all of the summer and the beginning of the fall. As such it is expected that the model shows the most variability in errors for this interval. It is expected as well that this interval has the lowest errors as it has highest learning ability of the neurons due to its largest training set [20]. This can be seen graphically in Figure 19. From Table 2, we see that the results were comparable to that of the intervals defined by the SOMs. We note also that spring has the largest RMSE from the moving ARIMA as expected due to the prevalence of convective storms. The moving ARIMA was also trained using three quarters of the entire data set, despite having this advantage of more information variability in training/learning phase, these results did not deviate significantly from the interval and the seasonal analyses.

ANN are powerful and are frequently used in time series forecasting due to their high parallelism, among other characteristics [26]. However, the ARIMA model is widely used and has given more accurate results for very short term forecasts [26].

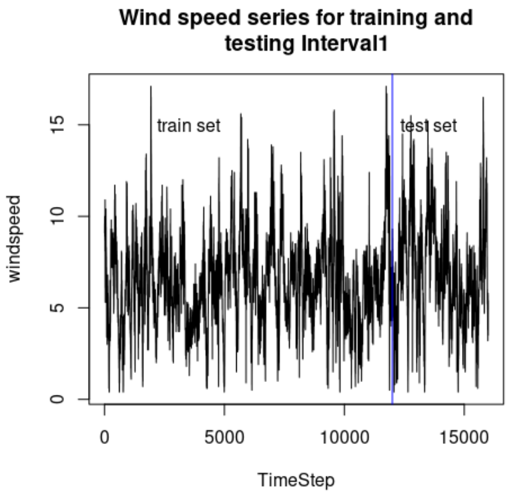


Figure 16: Interval 1 used in model runs consisting of rows 1 - 16,000 of the data

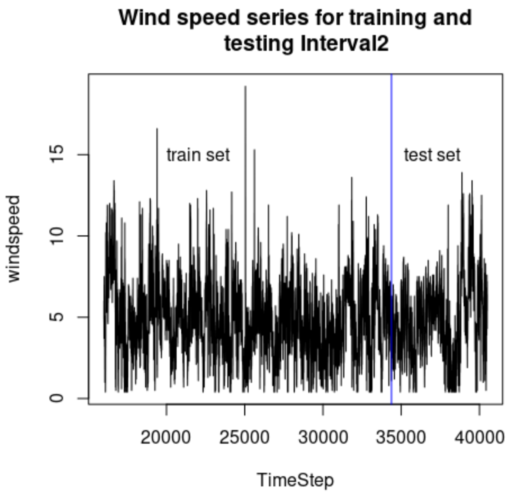


Figure 17: Interval 2 used in model runs consisting of rows 16,001 - 40,500 of the data

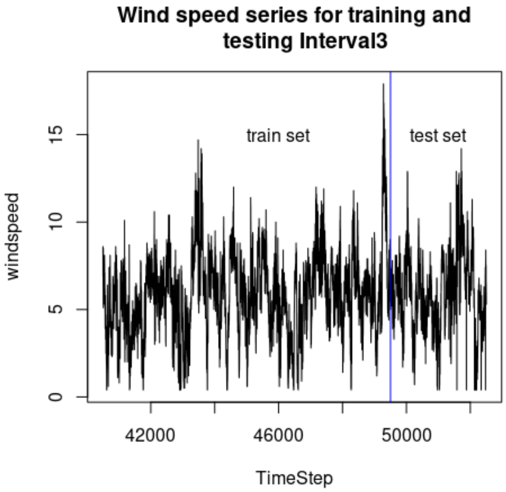


Figure 18: Interval 3 used in the model runs consisting of rows 40,501 - 52,500 of the data

Table 1: Moving ARIMA Results for the Intervals- RMSE determined for 20 to 120 minutes forecasts

h	**Interval1**	**Interval1**	**Interval2**	**Interval2**	**Interval3**	**Interval3**
(10-mins)	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*
2	0.7624309	0.5587051	0.6152723	0.4480265	0.738208	0.4980114
4	0.7817789	0.5783413	0.665906	0.5017876	0.756581	0.5186229
6	0.8008848	0.5957457	0.7360105	0.5666321	0.7666935	0.5284694
8	0.8154731	0.6082994	0.8106469	0.6321702	0.7712227	0.5328341
10	0.8257323	0.6171721	0.8831231	0.6938833	0.7727915	0.5341433
12	0.832471	0.622839	0.9509406	0.7507482	0.7735839	0.5348155

Table 2: Moving ARIMA Results for 2009 data set and the seasons- RMSE determined for 20 to 120 minutes forecasts

h (10-mins)	2009 data set- RMSE	Spring- RMSE	Summer- RMSE	Fall- RMSE
2	0.6943994	0.763676	0.5139139	0.6625789
4	0.711852	0.7787825	0.5504232	0.68182
6	0.730347	0.7998983	0.5724217	0.7019841
8	0.7445687	0.8185765	0.5821634	0.7278857
10	0.7543516	0.8330311	0.5854124	0.7558021
12	0.7608936	0.8438839	0.5858682	0.7844518

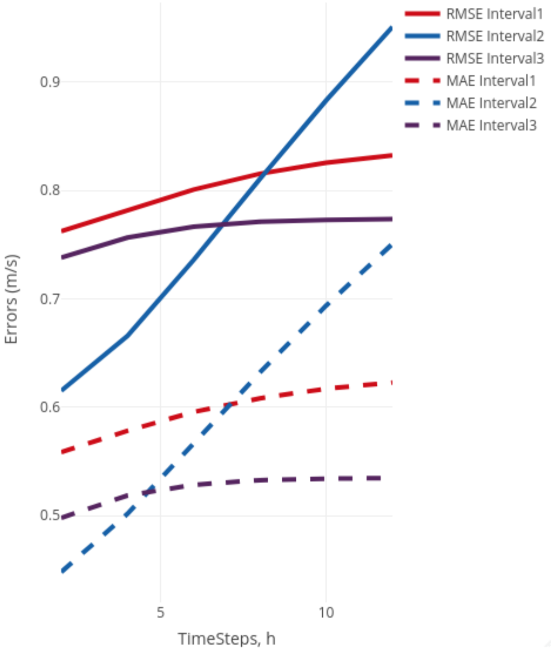


Figure 19: ARIMA Errors for the Intervals- RMSE for 20 to 120 minutes forecasts shows that Interval2 which consists of most of spring, all of summer and the beginning of fall, has the largest range of errors

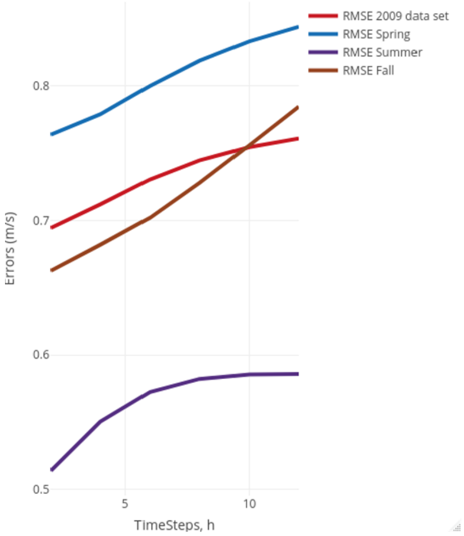


Figure 20: ARIMA Errors for 2009 data set and the seasons- RMSE for 20 to 120 minutes forecasts shows that summer and spring have the least and most errors respectively

The LSTM methodology was applied for the wind speed and pressure time series. Pressure was chosen because it had the greatest magnitude correlation with wind speed when compared with the other meteorological variables of wind direction and temperature. The RMSE results can be seen from Table 3. The test forecasted series for the various intervals, together with the actual series, can be seen in Figures 21 to 23. From the results obtained, ARIMA incurs smaller RMSE than the LSTM model for all intervals. Though there have been studies for which ARIMA outperforms ANN and SVM as mentioned in [27], there have been RNN methods used in wind speed forecasting which performs better that ARIMA. In [27], the errors are approximately 11 to 14 percent less in the RNN model compared to their ARIMA method. Another study, [9], univariate ARIMA saw higher errors than univariate RNN. The same result was observed when comparing multivariate ARIMA with a multivariate RNN.

The reason for such results can be attributed to the difficulty of representing the high dimensional and non-linear system using the one-dimensional wind speed time series [15]. As such the series is lagged using the time delay τ and the embedding dimension d for each of the intervals and these lagged co-ordinates were input to the LSTM model. The τ value was determined to be 3 using Auto Mutual Information (AMI) with the exception of Interval2 whose value was given by 2. The d value was determined to be 6 using Cao Algorithm for all intervals. The τ value was taken at the first local minimum for the AMI and the d value, as when $E1(d)$ attains saturation. Please refer to [3] for more information on the methodologies of these parameters as well as Figures 24 and 25. Another study that uses the lags of the series in the training of the ANN as input variables was [8]. It was determined in their study that the best model was the simplest consisting of two layers and two input and one output neurons [8].

The results obtained can be seen in Figures 26 to 28 for Interval1 to Interval3 respectively. The persistence model for each interval was constructed by calculating the average for every multiple of the 6^{th} hour and recording these as the values of persistence for the next consecutive 6 hours or 36 time steps. The time forecast horizon, h looked at for this analysis are 60, 120, 180, 240, 300 and 360 minutes. The models under comparison are the LSTM with lagged wind speeds as inputs, the Moving ARIMA, the LSTM with pressure and wind speeds as inputs and persistence. The Normalized Root Mean Squared Error (NRMSE) metric for model evaluation was determined for all of the models. For values of this metric exceeding 1 or 100% implies that the forecast is no better than the mean of the data after this run.

It can be noted that all models performed better than the persistence model which stayed consistently between 0.8 and approximately 1.0 for the three intervals. The h value of 360 minutes for Interval1 and Interval2 have values which are over 1.0 or representative of a forecast no better than the mean. The LSTM with the lagged wind speeds as inputs, denoted as Lagseries, outperformed the LSTM with the pressure and wind speeds as inputs, denoted as Pressureandwind, for all of the intervals. The Moving ARIMA method is now beaten by the lagged LSTMs for up to the 180^{th} minute time step in Interval2 and up to the 120^{th} time forecast horizon in Interval1 and Interval3. The second interval as mentioned previously has most of spring which have convective storm events, so it is expected that if any interval is to do best in the non-linear model of the LSTM when compared to the linear model of the ARIMA model, it would have been Interval2. The NRMSE of the LSTM Pressureandwind tends to one faster than the LSTM Lagseries for all of the intervals though up to the h value of 360 minutes, they do not exceed 1. As expected when the entire test set was forecasted for the models (h= test set), the NRMSE for most of the intervals exceeded 1; for the other cases, they were 0.97 and 0.98.

The tabulated results of the RMSE values for each of these models can be seen in Table 4. The forecasted and the actual series for h = 360 minutes for LagSeries1 to LagSeries3 can be seen in Figures 29 to 31 respectively. Similarly, these plots for the ARIMA1 to ARIMA3 test set can be seen in subsequent figures whilst correspondingly the error defined as the difference between the actual test data and the predicted test data can be viewed in Figures 35 to 37. It can be noted, especially for the Moving ARIMA results, there was a significant match between the predicted and the actual series. The differences in the actual test data and the predicted test data were varying about the zero marker thus indicating that the trends were well captured by the model.

Table 3: LSTM (with pressure and wind as inputs) RMSE values indicate that the Moving ARIMA beats the LSTM for forecast times of 20 to 120 minutes

h (10-mins)	**Interval1- RMSE**	**Interval2- RMSE**	**Interval3- RMSE**
2	0.940142796	0.975498677	0.834158971
4	1.155461471	1.219467954	0.927660403
6	1.296958276	1.257914418	0.992396502
8	1.444971761	1.35340903	1.026433109
10	1.451004208	1.432323828	1.137951695
12	2.767638696	1.502783097	1.14651633

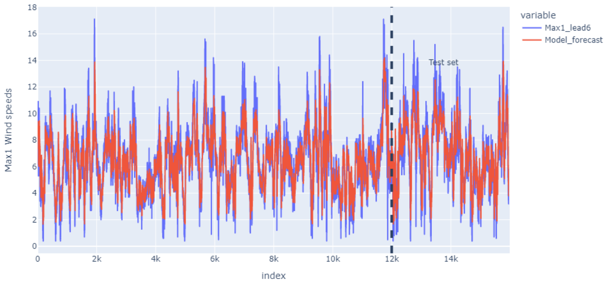


Figure 21: LSTM Interval1 showing the model forecast of forecast time 60 minutes or 1 hour and the actual wind speed values, $Max1$

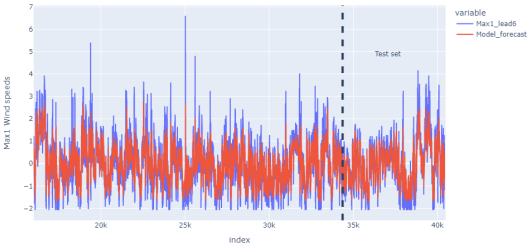


Figure 22: LSTM Interval2 showing the model forecast of forecast time 60 minutes or 1 hour and the actual wind speed values, $Max1$

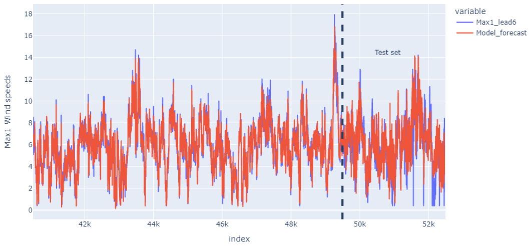


Figure 23: LSTM Interval3 showing the model forecast of forecast time 60 minutes or 1 hour and the actual wind speed values, $Max1$

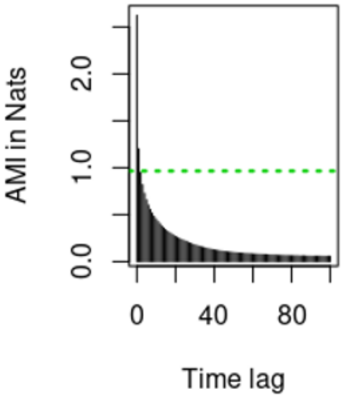


Figure 24: Tau or time delay for Interval2 is given by 2 using the Auto Mutual Information (AMI)

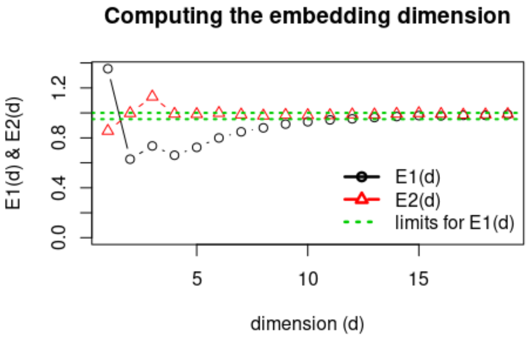


Figure 25: Embedding dimension for Interval2 is determined to be 6 from Cao Algorithm

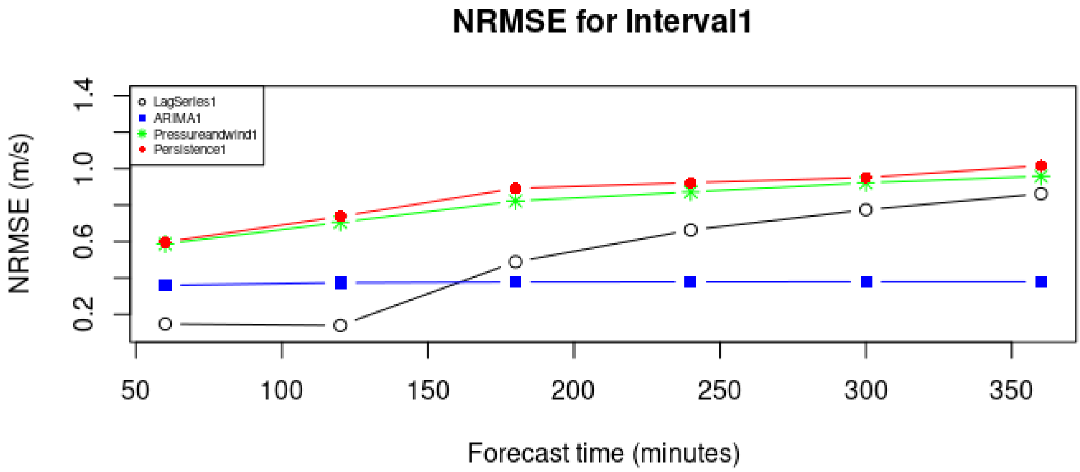


Figure 26: NRMSE for Interval1 using Models- LagSeries1 which is the LSTM with lagged wind speeds as inputs, ARIMA1 which is the Moving ARIMA model, Pressureandwind1 which is the LSTM with pressure and wind speeds as inputs and Persistence1 which is the Persistence model. The LagSeries1 improves upon Pressureandwind1 and beats the Moving ARIMA for forecast times of 60 and 120 minutes

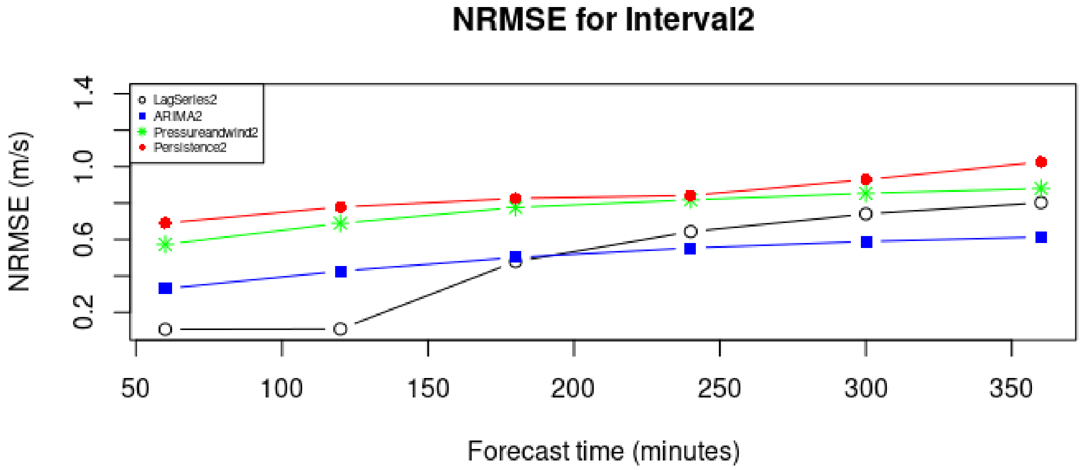


Figure 27: NRMSE for Interval2 using Models- LagSeries2 which is the LSTM with lagged wind speeds as inputs, ARIMA2 which is the Moving ARIMA model, Pressureandwind2 which is the LSTM with pressure and wind speeds as inputs and Persistence2 which is the Persistence model. The LagSeries2 improves upon Pressureandwind2 and beats the Moving ARIMA for forecast times of 60, 120 and 180 minutes

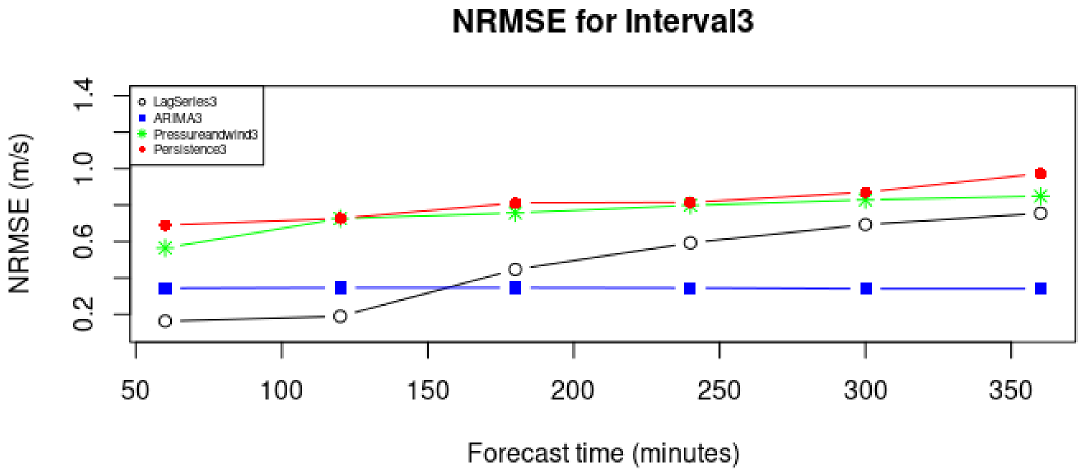


Figure 28: NRMSE for Interval3 using Models- LagSeries3 which is the LSTM with lagged wind speeds as inputs, ARIMA3 which is the Moving ARIMA model, Pressureandwind3 which is the LSTM with pressure and wind speeds as inputs and Persistence3 which is the Persistence model. The LagSeries3 improves upon Pressureandwind3 and beats the Moving ARIMA for forecast times of 60 and 120 minutes

Table 4: Models' RMSE Results for the Intervals showing that the best performing model, up to 180 minutes or 3 hours, is LagSeries

Interval	**h (mins)**	**LagSeries**	**ARIMA**	**Pressureandwind**	**Persistence**
Interval1	60	0.372907494	0.8008848	1.307214932	1.331339773
	120	0.352250278	0.832471	1.572144369	1.643609466
	180	1.087379921	0.8412903	1.831567231	1.987401039
	240	1.477212704	0.8436806	1.943241466	2.055514112
	300	1.724707138	0.8443778	2.05444378	2.119308164
	360	1.920543992	0.8444589	2.13412727	2.263135526
Interval2	60	0.238592146	0.7360105	1.276806808	1.538227218
	120	0.242534049	0.9509406	1.53582464	1.734079072
	180	1.067046011	1.117465	1.72837262	1.83946146
	240	1.432849673	1.233804	1.822299134	1.875548504
	300	1.649253577	1.312496	1.9014607	2.069048477
	360	1.786658699	1.365862	1.961462493	2.284102311
Interval3	60	0.376359712	0.7666935	1.257796263	1.536880065
	120	0.436734176	0.7735839	1.617069345	1.619311738
	180	1.028754688	0.7734066	1.685952464	1.80858766
	240	1.36522671	0.7698904	1.778578867	1.814853169
	300	1.596357379	0.7602228	1.84599341	1.939363159
	360	1.739204251	0.7602228	1.891493873	2.168188039

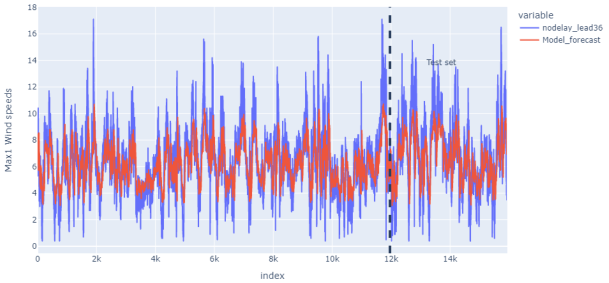


Figure 29: LagSeries1 LSTM showing the model forecast of prediction time, 6 hours and the actual wind speed values, $Max1$

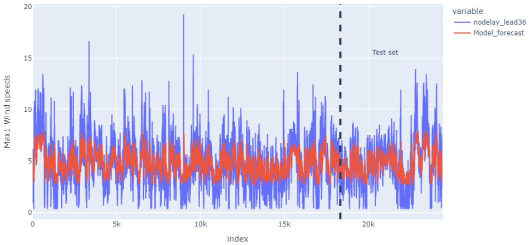


Figure 30: Lagseries2 LSTM showing the model forecast of prediction time, 6 hours and the actual wind speed values, $Max1$

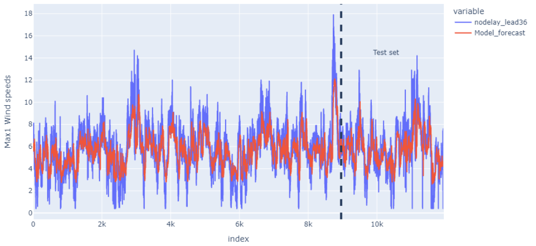


Figure 31: LagSeries3 LSTM showing the model forecast of prediction time, 6 hours and the actual wind speed values, $Max1$

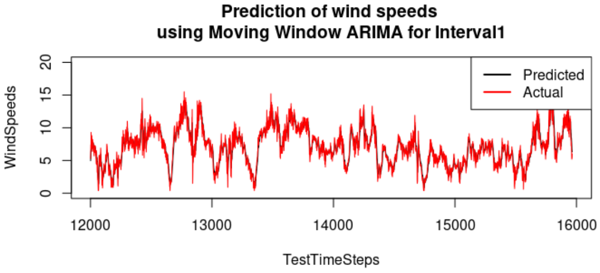


Figure 32: Predictions of Moving ARIMA1 for forecast time, 6 hours and the actual wind speed values

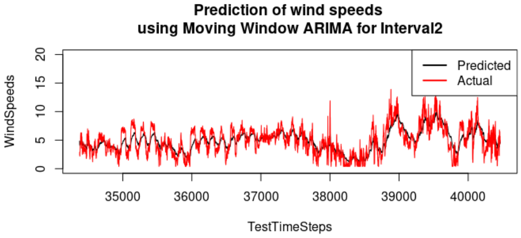


Figure 33: Predictions of Moving ARIMA2 for forecast time, 6 hours and the actual wind speed values

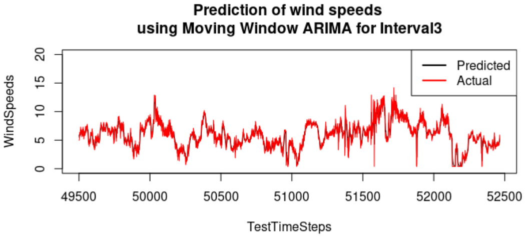


Figure 34: Predictions of Moving ARIMA3 for forecast time, 6 hours and the actual wind speed values

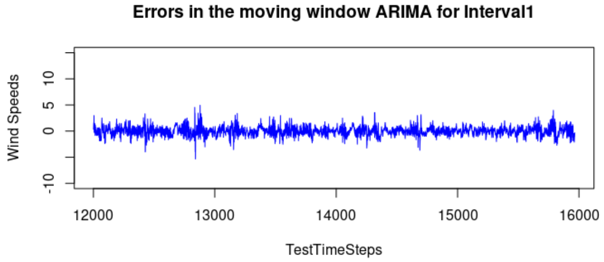


Figure 35: Errors of Moving ARIMA1 for forecasting time, 6 hours, of the test time series

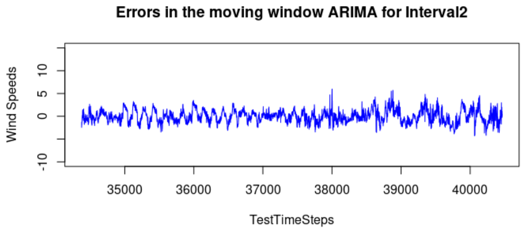


Figure 36: Errors of Moving ARIMA2 for forecasting time, 6 hours, of the test time series

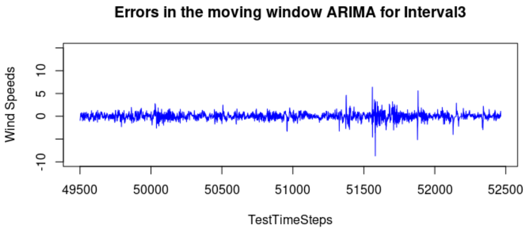


Figure 37: Errors for Moving ARIMA3 for forecasting time, 6 hours, of the test time series

5 Future Work, Additional Analyses and Conclusion

In the model runs, the forecast variable - in our case wind speed - can be further processed to determine if there are any patterns in the wind speed forecast values (in terms of its accuracy) when its actual values is less than or greater than some x value or the difference between consecutive actual values rates are higher than some y value.

Yearly analysis can be done to see if there are the same number of clusters and accuracy in forecasting (seasonal analysis - using yearly data) is similar.

The optimal number of clusters was determined to be 4 using the Elbow and Silhouette methods among others. SOMs were then used to cluster the data after which three continuous intervals belonging to a particular cluster, which represented approximately 50% and over of the input vectors or rows from the data frame were identified. These intervals were then inputs for the LSTMs with inputs pressure and wind speeds, the lagged series LSTMs with embedding dimension d and time delay τ, the Moving Window ARIMA and persistence models. It was determined that the Moving ARIMA model is outperformed by the lagged LSTM for at most 180 minutes from the runs of the defined intervals. The lagged series improved upon the LSTM with the wind speed and pressure series. All of these models however, performed better than the benchmark of persistence for all time steps.

References

[1] V. Bali, A. Kumar, and S. Gangwar. A novel approach for wind speed forecasting using lstm-arima deep learning models. International Journal of Agricultural and Environmental Information Systems (IJAEIS), 11(3):13–30, 2020.

[2] S. Balkissoon, N. Fox, and A. Lupo. Fractal characteristics of tall tower wind speeds in missouri. Renewable Energy, 154:1346–1356, 2020.

[3] S. Balkissoon, N. Fox, A. Lupo, S. E. Haupt, Y. C. Li, P. Market, and S. Walsh. Determining chaotic characteristics and forecasting tall tower wind speeds in missouri using empirical dynamical modeling (edm). Renewable Energy, 170:1292–1307, 2021.

[4] I. A. Basheer and M. Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. Journal of microbiological methods, 43(1):3–31, 2000.

[5] S. Berkovic. Winter wind regimes over israel using self-organizing maps. Journal of Applied Meteorology and Climatology, 56(10):2671–2691, 2017.

[6] J. Browell, D. R. Drew, and K. Philippopoulos. Improved very short-term spatio-temporal wind forecasting using atmospheric regimes. Wind Energy, 21(11):968–979, 2018.

[7] J. C. Burguillo. Using self-organizing maps with complex network topologies and coalitions for time series prediction. Soft Computing, 18(4):695–705, 2014.

[8] E. Cadenas and W. Rivera. Short term wind speed forecasting in la venta, oaxaca, méxico, using artificial neural networks. Renewable Energy, 34(1):274–278, 2009.

[9] Q. Cao, B. T. Ewing, and M. A. Thompson. Forecasting wind speed with recurrent neural networks. European Journal of Operational Research, 221(1):148–154, 2012.

[10] M. Elsaraiti and A. Merabet. A comparative analysis of the arima and lstm predictive models and their effectiveness for predicting wind speed. Energies, 14(20):6782, 2021.

[11] N. I. Fox. A tall tower study of missouri winds. Renewable Energy, 36(1):330–337, 2011.

[12] R. Fukuoka, H. Suzuki, T. Kitajima, A. Kuwahara, and T. Yasuno. Wind speed prediction model using lstm and 1d-cnn. Journal of Signal Processing, 22(4):207–210, 2018.

[13] S. Gangwar, V. Bali, and A. Kumar. Comparative analysis of wind speed forecasting using lstm and svm. EAI Endorsed Transactions on Scalable Information Systems, 7(25):e1–e1, 2020.

[14] D. Geng, H. Zhang, and H. Wu. Short-term wind speed prediction based on principal component analysis and lstm. Applied Sciences, 10(13):4416, 2020.

[15] R. Hu, W. Hu, N. Gökmen, P. Li, Q. Huang, and Z. Chen. High resolution wind speed forecasting based on wavelet decomposed phase space reconstruction and self-organizing map. Renewable Energy, 140:17–31, 2019.

[16] H. Kalinić, F. Matić, H. Mihanović, I. Vilibić, N. Žagar, B. Jasenko, and M. Tudor. Comparison of two meteorological models using self-organizing maps. In OCEANS 2015-Genova, pages 1–6. IEEE, 2015.

[17] B. Kent. How to use pytorch lstms for time series regression. https://www.crosstab.io/articles/time-series-pytorch-lstm.

[18] S. Lakshminarayanan. Application of self-organizing maps on time series data for identifying interpretable driving manoeuvres. European transport research review, 12(1):1–11, 2020.

[19] G. Li and J. Shi. On comparing three artificial neural networks for wind speed forecasting. Applied Energy, 87(7):2313–2320, 2010.

[20] G.-F. Lin, T.-C. Wang, and L.-H. Chen. A forecasting approach combining self-organizing map with support vector regression for reservoir inflow during typhoon periods. Advances in Meteorology, 2016, 2016.

[21] M. C. Mabel and E. Fernandez. Analysis of wind power generation and prediction using ann: A case study. Renewable energy, 33(5):986–992, 2008.

[22] V. Nourani, A. H. Baghanam, V. Vousoughi, and M. Alami. Classification of groundwater level data using som to develop ann-based forecasting model. Int J Soft Comput Eng, 2(1):2231–07, 2012.

[23] J. L. Pearce, L. A. Waller, H. H. Chang, M. Klein, J. A. Mulholland, J. A. Sarnat, S. E. Sarnat, M. J. Strickland, and P. E. Tolbert. Using self-organizing maps to develop ambient air quality classifications: a time series example. Environmental Health, 13(1):1–14, 2014.

[24] R. Rakotomalala. Tanagra data mining. Version, 1:39, 2005.

[25] P. Ramasamy, S. Chandel, and A. K. Yadav. Wind speed prediction in the mountainous region of india using an artificial neural network model. Renewable Energy, 80:338–347, 2015.

[26] N. Ramesh Babu and P. Arulmozhivarman. Forecasting of wind speed using artificial neural networks. Int. Rev. Mod. Sim, 5(5), 2012.

[27] K. Sandhu, A. R. Nair, et al. A comparative study of arima and rnn for short term wind speed forecasting. In 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pages 1–7. IEEE, 2019.

[28] A. Sfetsos. A comparison of various forecasting techniques applied to mean hourly wind speed time series. Renewable energy, 21(1):23–35, 2000.

[29] J. Tian, M. H. Azarian, and M. Pecht. Anomaly detection using self-organizing maps-based k-nearest neighbor algorithm. In PHM Society European Conference, volume 2, 2014.

[30] R. Wehrens, L. M. Buydens, et al. Self-and super-organizing maps in r: the kohonen package. Journal of Statistical Software, 21(5):1–19, 2007.

[31] Y.-X. Wu, Q.-B. Wu, and J.-Q. Zhu. Data-driven wind speed forecasting using deep feature extraction and lstm. IET Renewable Power Generation, 13(12):2062–2069, 2019.

[32] M. Yan and K. Ye. Determining the number of clusters using the weighted gap statistic. Biometrics, 63(4):1031–1037, 2007.

[33] W. Yao, P. Huang, and Z. Jia. Multidimensional lstm networks to predict wind speed. In 2018 37th Chinese Control Conference (CCC), pages 7493–7497. IEEE, 2018.

[34] U. Zaman, H. Teimourzadeh, E. H. Sangani, X. Liang, and C. Y. Chung. Wind speed forecasting using arma and neural network models. In 2021 IEEE Electrical Power and Energy Conference (EPEC), pages 243–248. IEEE, 2021.