Improving forecast accuracy of wind power output using multi-input LSTM model.

Phillemon Ntona Senoamadi (phillemon@aims.ac.za)
African Institute for Mathematical Sciences (AIMS)

Supervised by: Dr. Nicolene Botha
CSIR, South Africa
Co-Supervised by: Dr. Bubacarr Bah
AIMS, South Africa
Co-Supervised by: Dr. Vukosi Marivate
University of Pretoria, South Africa

12 October 2018

Submitted in partial fulfillment of a structured masters degree at AIMS South Africa

Abstract

Accurate wind power forecasting can help operators to estimate how much profit to make depending on where and when in order to produce more electricity. Precise prediction allows operators to achieve suitable trading performances. In this study, data sets obtained from two wind stations i.e. Memel (Free State) and Jozini (Kwa-Zulu-natal), are used for the prediction of the mean wind speed. Moreover, the mean wind speed from one of the station is used to increase the prediction of the other station. The long short term memory (LSTM) model is used for the prediction which is benchmarked by the persistence forecast. The LSTM model was able to beat the benchmark model root mean square results. The applied forecasting models focuses on the first, twelfth and twenty fourth hour.

Keywords: long short term memory (LSTM), persistence forecast and root mean square.

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

Phillemon Ntona Senoamadi, 12 October 2018

Contents

1. Introduction

1.1 Wind Energy and Other Sources

Energy is very important with regards to the development and economic growth of the world. Energy can be classified as either non-renewable or renewable. Non-renewable sources of energy are those that can be depleted by repeated use such as fossil fuels. According to the article by Höök and Tang (2013) fossil fuels such as coal and oil are facing major depletion year by year, which has been identified as a future challenge. In addition they have a negative impact on the environment such as air pollution. They are also detrimental to the health of humans and animals.

Renewable energy sources are those that cannot be depleted by repeated use, such as solar, wind and geothermal energy. The study of Höök and Tang (2013) shows how important renewable sources of energy are compared to non-renewable source of energy. From the national renewable energy conference held in 2008 by Department of Energy, it has been motivated that 15% of electricity will come from renewable energy source by the year 2020 (Marquard, 2008). According to the study of Panwar et al. (2011) renewable sources of energy that are mostly used in households have the potential to produce energy with almost zero emission of air pollution and green house gases.

In this research project the focus will be on wind energy. Wind energy is advantageous due to the fact that is renewable and it is not in a category of sources that are facing depletion. It has no negative impact to the environment in terms of pollution and is thus one of the recommended sources of energy. Wind energy is generated using wind turbines that covert mechanical energy to electrical energy. According to Van der Walt and Botha (2016) power output generated by wind turbines is highly correlated to wind speed. Figure 1.1 shows the wind farm turbines that generates electricity.

Figure 1.1: Wind Farm Turbines, (Blue World Carbon ,2013)

According to the study of (Lee and Wang, 2008) wind power generation systems uses three wind turbine generators (WTGs), a diesel engine generator, two fuel cells (FCs), and a photovoltaic system (PV) while, the energy storage subsystems uses a battery energy storage system. However, there is a problem regarding the storage of wind energy power due to the lack of available space in order to build the huge storage facilities.

1.2 Importance of the Study

The power generation of electricity by turbines is in proportion with the wind speed. Forecasting wind speed it is crucial for construction of projects and maintenance planning (Shafiee, 2015). Precise wind speed forecast will help operators to estimate how much profit to make depending on where and when to produce more electricity. By making proper usage of wind prediction, we can determine the cost effective use of wind energy as an activity component of the energy system (Jones et al., 2005). Precise prediction allows operators to achieve suitable trading performances on an area which commercial dealings of electricity are conducted.

1.3 Relevant methods to the study

Wind speed data used in this project was in the form of Time Series (TS). Wind TS data can sometimes be noisy and it is non-linear. To perform prediction on non-linear and noisy data, Machine learning (ML) approaches such as Neural Network (NN), Recurrent Neural Network (RNN) techniques and Long Short Term Memory (LSTM) we can be used. Traditional methods and statistical methods such as regression, autoregressive moving averages (ARIMA) models and conditional heteroscedastic (CH) models have shown some limitation in wind power forecast (Deljac et al., 2011). However using NN's techniques to forecast wind speed has shown a great improvement over traditional and statistics methods (Ritchie et al., 2003). According to Moniz and Krueger (2018) Long short term memory (LSTM) models is known for its capability of learning long term dependencies, LSTM's uses multi-gated units in its architecture to encounter related problems like vanishing and exploding gradient problems.

The aim of the study is forecast the wind speed at 1, 12th and 24th hour ahead using LSTM. More features are added in order to increase the accuracy of the LSTM using features from another station. The features used are from the Station that is close to the Station we want to increase forecast accuracy of wind power. According to the study by Filonov et al. (2016) LSTM prediction model has higher prediction accuracy and greater potential in forecasting time series data. In this research the LSTM will be benchmarked against persistence model. Persistence model uses the value previous hour and use it as aprediction at current hour.

1.4 Data Source and Preparation

The data set was extracted from the Wind Atlas of South Africa (WASA) (WASAData, 2010).The data was taken from two stations in South Africa, namely Memel (station 13) Free State and Jozini (station 14) Kwa-Zulu-Natal. The data was recorded at a space interval of 10 minutes, however, we had to sampled the data to 1 hour time interval and the prediction its still accurate with increased time interval. The data was sampled to 1 hour interval because of the cost of computation due to the large data set. The data set used 335600 measurements recorded at 10 minutes interval and was sampled to 17300 hours. In the process of data generation there is sometimes problems with sensor failure which lead to missing data (Wei et al., 2010). The data used had the space of at most an hour with missing values and an interpolation method was used to fill those missing values. The data is recorded in terms of wind direction (WD in ^{0}TN), wind speed (WS in m/s), air temperature (TAir in ^{0}C), Barometric Pressure (PBaro in Hpa) and Relative Humidity (RH in $\%$). The data was separated in terms of mean wind speed (WS), minimum WS, maximum WS and standard deviation WS. The data was recorded at

a hub height of 10m, 20m, 40m, 60m and 62m. In this study we focus on predicting mean wind speed at a height of 40m and 60m.

1.5 Project Organization

This section describes the structure and order of this project. The remainder of this project is as follows; Chapter 2, presents related methods under Machine Leaning (ML) and statistical methods used for time series data prediction. Chapter 3, presents the methods used namely; Long Short Term Memory (LSTM), Persistence model and data processing. Chapter 4, consists of findings, visualisations and discussion. Chapter 5 concludes the study and presents prospects for further research.

2. Related Methods

This chapter is divided in to four sections. Section 2.1 presents three machine learning tasks namely supervised, unsupervised and semi-supervised. Section 2.2 presents machine learning (ML) algorithm that can be used to predict and classify data. Section 2.3 presents activation function that are used to train models in ML. Section 2.4 presents different optimisation algorithm that are used to train ML models.

2.1 Learning Tasks

2.1.1 Supervised Learning. Given a set of data input values x_i for i=1, 2, . . . , n where n is the number of training examples. In supervised learning we train the algorithm on both input value x_i and output values y_i. The algorithm will deduce the mapping from input to output e.g $y = f(x)$. The main goal in supervised learning is to create a function that is good enough to be able to learn and forecast outputs given new or unseen input values (Chapelle et al., 2009). Some classification algorithms includes Support Vector Machine (SVM) and K-Nearest Neighbours (KNN). While Regression algorithms includes Linear regression, Non-linear regression and Neural Network (NN).

2.1.2 Unsupervised Learning. In Unsupervised learning we have only the input values but no outputs. The main goal about unsupervised learning is to understand the structure of data or distribution in order to learn more on how the data is structured. Unsupervised learning is classified into association and clustering. In clustering we try to group data into similar groups or of same data points (Lloyd et al., 2013). In association we want to discover a manner that illustrate large portions of data, in such a way that the behaviour of data in A also tends to be like in B for some similar ways. Some examples behind unsupervised learning algorithms are k-means for clustering problems and mixture models.

2.1.3 Semi Supervised Learning. Semi-supervised (SS) learning can be either Supervised Learning or Unsupervised learning. In SS learning we have a huge amount of dataset with few number of labelled dataset (Chapelle et al., 2009). Some guessing techniques such as KNN can be used to associate the suitable values. Semi-Supervised Learning is time consuming and can be very expensive sometimes. we can also make use of unsupervised learning techniques to discover and learn the structure in the input or the distribution of the dataset. According to Sathya and Abraham (2013) unsupervised learning model identify the pattern class information heuristically and reinforcement get more knowledge through trial and error interactions with its environment.

2.2 Related Approaches and Algorithms

2.2.1 Artificial Neural Network. Artificial Neural Networks (ANN's) are the implementation of the neuronal structure of a brain in terms of a software. In a human brain, neural network (NN) is a huge interconnected network of neurons. This simply means that the output of a neuron in a brain can be the input of huge amount of neurons. According to Lee et al. (2016), ANNs are used to create models of a system state using non-linear combinations of the input variables. In this study we are dealing with TS data which is non-linear and according to the study by Ranasinghe et al. (2017) ANNs have been extensively used in modelling a wide range of problems with the privilege of being able to deal

with non-linearity and have demonstrated extremely reliable predictive capability. The differentiation structure of a NN network of human and ANN is given in Figure 2.1.

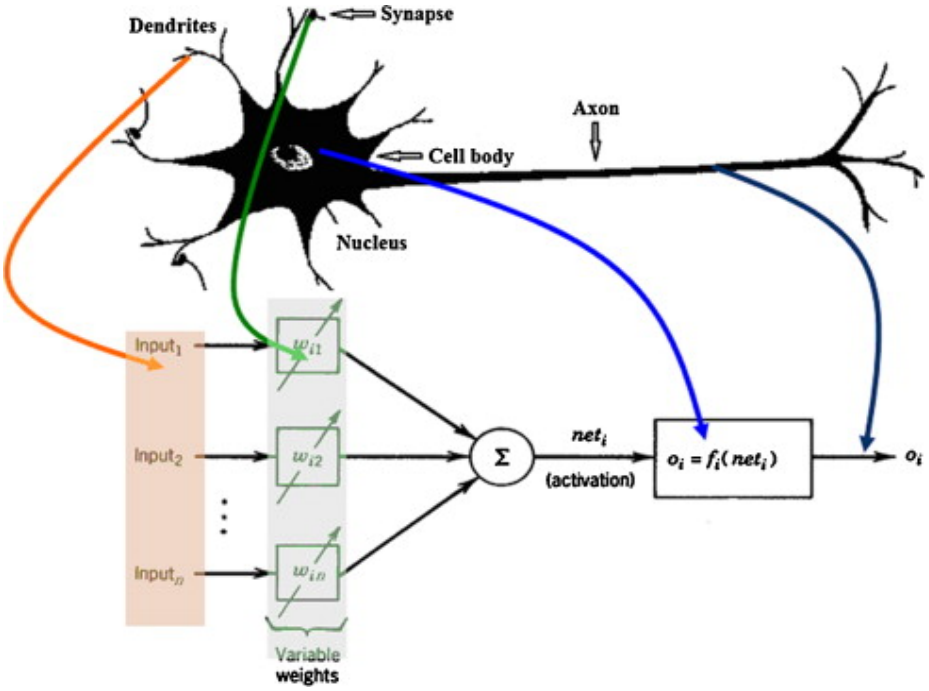


Figure 2.1: BNN vs ANN, (Srivastava T,2018).

In Figure 2.1 we have both brain neural network (BNN) and ANN that use the same architecture of having an activation unit. The dendrites in BNN works as inputs in ANN and synapse in BNN works the same as the weights in ANN. According to Patterson and Gibson (2017) Neural networks are a computational model that have most properties of a human brain in which many plain units are collaborating together in parallel with no consolidation control unit. Updating the weights is the primary way the NN learns new information.

2.2.2 Recurrent Neural Network. Recurrent Neural Network (RNN) is a class of NN that takes advantage of the sequence or succeeding nature of the input. Those inputs can be speech, text or TS data. According to the study by Patterson and Gibson (2017) an input can be of any form where the actual instance of an element in the sequence is dependent on the elements that appeared before it. RNN's are very flexible and have been used to solve problems such as speech recognition, language modelling, machine learning translation, sentiment analysis and image captioning (Patterson and Gibson, 2017). The idea behind the recurrent NN, it tries to forecast the future by taking into account the past or history. RNN is capable of using long term history of sequence, but in practice we can use few steps.The typical structure of RNN is given below in 2.2.

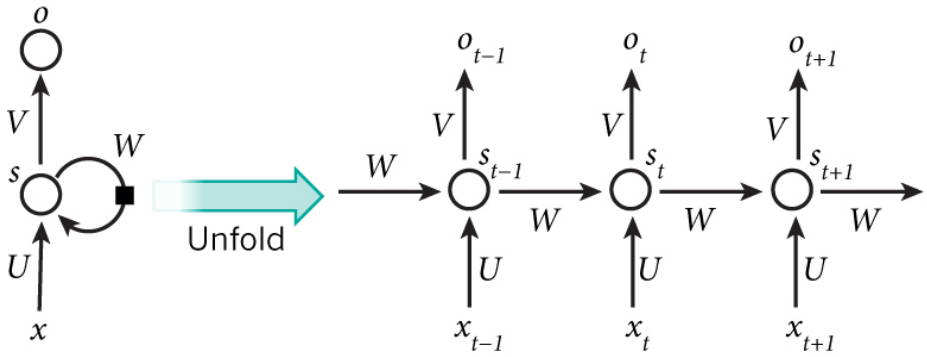


Figure 2.2: RNN, (Britz .D,2015).

From Figure 2.2 we have folded and unfolded recurrent NN. Table 2.1 below explains the variables of the unfolded RNN in Figure (2.2).

Table 2.1: RNN

t	time at each step
x_t	input at current step.
x_{t-i}	input at previous time step.
x_{t+i}	input at coming steps.
W	weights matrix connecting states.
V	weights matrix between the states and output
o	outputs at each time step
U	weights between the states and output.
S_t,S_{t-i} and S_{t+i}	hidden states at current, previous or next state.

$$S_t = W.S_{t-i} + Ux_t, \tag{2.2.1}$$

$$O_t = V.S_t, \tag{2.2.2}$$

From Figure 2.2 RNN at a time t is given by the value of the hidden vector S_t. To get S_t we sum the product of the (weight matrix W and S_{t-1} from previous states) and (the weight matrix U and x_t) at current state (see Figure (2.2.1)).The output O_t it is the product of S_t and the weight matrix V. S_t and O_t are passed through some activation function for non linearity. According to Gulli and Pal (2017) the choice of $tanh$ over other non-linearities has to do with its second derivative decaying very slowly to zero. This keeps the gradients in the linear region of the activation function and helps combat the vanishing gradient problem.

2.2.3 Vanishing/Exploding Gradient Problem. One of the problems with training a NN, especially deep neural network (DNN) is that we encounter a vanishing and exploding gradient problems (Kim et al., 2016). When training a very deep network the derivative or slope can sometimes get very big or small. Options to overcome vanishing and exploding problem in Table 2.2.

Table 2.2: Vanishing and Exploding Gradient, (Kim et al., 2016)

Exploding Gradient Solution	Vanishing Gradient Solutions
Truncated Backpropagation	Weight Initialisation
Penalties	Echo State Networks
Gradient Clipping	LSTM

2.2.4 Regression Cost function. Suppose we have training examples $(x^1, y^1), \ldots, (x^m, y^m)$, where m is the number of training examples. The decision to approximate the value y, represented as h as multiple linear regression function of x given in (2.2.3).

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + +\theta_m x_m, \tag{2.2.3}$$

In a summation form can be given below in 2.2.4:

$$h_\theta(x) = \sum_{i=0}^{m} \theta_i x_i = \theta^T x, \tag{2.2.4}$$

(2.2.4) it is used to define the cost function in (2.2.5).

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2, \tag{2.2.5}$$

Least Mean Square Algorithm starts with an initial guess for θ and repeatedly change θ to make $J(\theta)$ smaller until, it converges to the local minimum of our function $J(\theta)$.

2.2.5 Back Propagation Algorithm. In a feed forward NN we mostly get the loss or error high. With back propagation (BP) we update the weights in order to minimize the error to get the optimal results. According to the study by Humpert (1994) the modified BP in a NN improves the learning speed. Hence the study of Humpert (1994) shows how important to the increase of learning rate. In practice the learning rate is always between $0 < \eta < 1$ to ensure that sequential steps in the weight space do not exceed or overshoot the minimum of the error surface (Magoulas et al., 1999). Feed forward NN that is differentiable allows BP. The function it is usually the summed squared of the errors (SEE) between the desired output and predicted (Guo, 2013). The loss function is defined in (2.2.6).

$$SEE = L = \frac{1}{2} \sum_{i}^{n} \sum_{j}^{m} (y_{ij} - \hat{y}_{ij})^2, \tag{2.2.6}$$

In (2.2.6), y is the desired output, n the number of training examples and m is the number of predicted values. $\hat{y}$ it is the predicted value. BP is performed through gradient decent (GD). Using GD the weights change must be proportionate to the loss function in terms of the specific weights. Table 2.3 gives the description of each variable used in Subsection 2.2.5.

Variable	Description
w and v	weights at specific layers
ΔW	weight change
η	learning rate
L	loss function
σ	vector at each node
nt, s	linear combination of inputs

Table 2.3: BP Parameters

$$\Delta W = -\eta \frac{\partial L}{\partial w}, \tag{2.2.7}$$

With a large number of layers cost function can be very hard due to the successive arrangement of NN structure. Computation of the weights in all layer are implemented by using the chain rule. The hard part about BP it is because of non-linearity of the activation functions, activation functions are explained in Subsection 2.3. So we can make things simpler from (2.2.8) below by partitioning the linear and non linearity in the gradients.

$$\Delta W = -\eta \frac{\partial L}{\partial nt} \frac{\partial nt}{\partial w}, \tag{2.2.8}$$

$$\frac{\partial nt}{\partial w}, \tag{2.2.9}$$

$$\frac{\partial L}{\partial nt}, \tag{2.2.10}$$

In (2.2.8) nt represent the linear combinations of inputs and (2.2.9) will be easy to be derived. Then we need to work with (2.2.10) and set $\delta = -\frac{\partial L}{\partial nt}$ to be a vector at each node in our NN. The output nodes are given by:

$$\delta_{ij} = \frac{\partial L}{\partial y_{ij}} \frac{\partial y_{ij}}{\partial w}, \tag{2.2.11}$$

$$\delta_{ij} = (y_{ij} - \hat{y}_{ij}) g^{'}(nt_{ij}), \tag{2.2.12}$$

The hidden nodes will be given by:

$$\delta_{ik} = -\sum_{j}^{m} \frac{\partial L}{\partial y_{ij}} \frac{\partial y_{ij}}{\partial nt_{ij}} \frac{\partial nt_{ij}}{\partial s_{ik}} \frac{\partial s_{ik}}{\partial nt_{ik}}, \tag{2.2.13}$$

$$\delta_{ik} = \sum_{j}^{m} \delta_{ij} w_{ij} f^{'}(nt_{ik}), \tag{2.2.14}$$

Therefore it is evident that the weighted sum propagate the sum of the errors. The change in weight can be given by:

$$\Delta w_{jk} = \eta \sum_{i}^{n} \delta_{ij} s_{ik} \tag{2.2.15}$$

$$\Delta v_{ks} = \eta \sum_{i}^{n} \delta_{ik} s_{is}, \tag{2.2.16}$$

for hidden layer the output weights W in (2.2.15) and hidden weights V in (2.2.16).

Finally the output layer recurrent weights change is given below in (2.2.17):

$$\Delta w_{ks} = \eta \sum_{i}^{n} \delta_{ik} s_{iq}, \tag{2.2.17}$$

2.2.6 Back Propagation Through Time. Back Propagation through time (BBTT) is one of the methods that are used to train RNN. The unfolded RNN in Figure 2.2 is used to perform a forward pass. According to the study by Bersini and Gorrini (1997) the BPTT algorithm is less time consuming and it can be simplified by applying Lagrangian calculus. In BPTT we compute gradients through the usual BP and update shared weights. According to the study by Werbos (1990) BPTT has a challenge with the high cost of updating single parameter. Hence, it makes it not easy to use big number of iterations (Werbos, 1990). From (2.2.1) in Subsection 2.2.2 and change the notation of O_t to $\hat{y}$ as the output and let y be the actual value. The cross loss entropy function is given below in 2.2.20:

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t, \tag{2.2.18}$$

$$E(y, \hat{y}) = \sum_{t} E_t(y_t, \hat{y}_t), \tag{2.2.19}$$

$$= -y_t \log \hat{y}_t, \tag{2.2.20}$$

Equation 2.2.20 it is the total sum of the error per each time sped between outputs y_t and the actual values y_t. Our main goal is to update the gradients with respect to the weights given in Figure 2.2 using stochastic gradient decent (SGD) discussed in Subsection 2.4.1.

Now, summing up the gradients we obtain 2.2.21.

$$\frac{\partial E}{\partial W} = \sum_{t} \frac{\partial E_t}{\partial W}, \tag{2.2.21}$$

Using the chain rule to calculate the errors, using E_3 from Figure 2.3 through the entire illustration to make things simpler, we have:

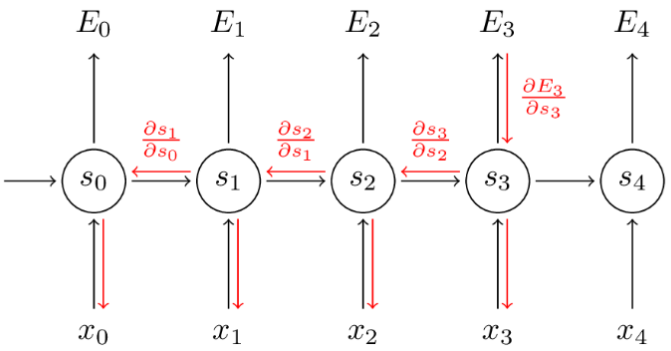


Figure 2.3: BPTT Illustration network, (Britz D ,2015).

$$\frac{\partial E_3}{\partial V} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial V}, \tag{2.2.22}$$

$$\frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial z_3}\frac{\partial z_3}{\partial V}, \tag{2.2.23}$$

$$(\hat{y}_3 - y_3) \otimes s_3, \tag{2.2.24}$$

where V it is the Weight in Figure 2.2, $\hat{y}_3$ it is the output in unit 3 and $z_3 = Vs_3$. s_3 is give in 2.2.1. By applying the chain rule twice to $\frac{\partial E_3}{\partial W}$ we get 2.2.25.

$$\frac{\partial E_3}{\partial V} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial \hat{s}_3}{\partial W} \tag{2.2.25}$$

The above process is repeated with all E_t by back propagating gradients from $t = 3$ till $t = 1$. Hence its called this method BPTT (Britz D ,2015).

2.2.7 Support Vector Machines. Support Vector Machines (SVM) is a machine learning algorithm that is versatile in the way that it can be used for both regression and classification. Nevertheless is used by a large in classification problems (Joachims, 1998). Support Vector machine creates a border line that separate or classify by finding the hyperplane that can show the best distinction between two classes in an optimal way, see the Figure 2.4 below:

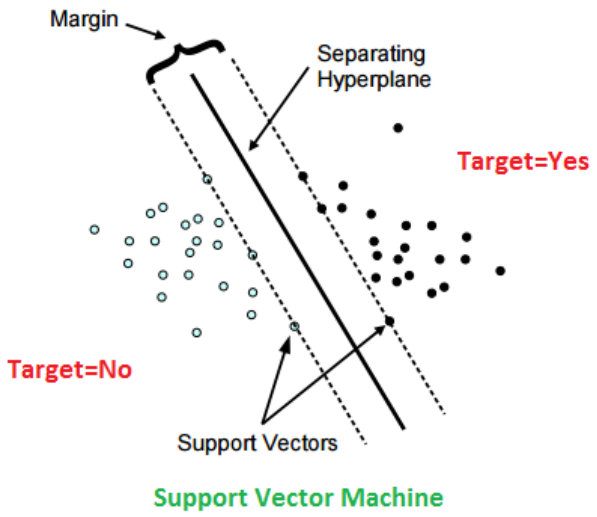


Figure 2.4: Support Vector Machines, (Dnl Institute,2015).

Apart from SVM, some machine learning algorithms that can work like SVMs are Decision Trees, Random Forest and NN. In SVM we select or identify the good hyper-planes to classify all the objects in our dataset (Joachims, 1998). For example if choose two classes, we select the best hyper-plane that can classify both in the best way. Next we maximize the metric norm between the points that are close to each other and also separating two classes. Then we draw the vertical lines on the on those maximised distance data points in parallel to the best hyperplane chosen. In Figure 2.4 it is evident those two vertical lines named support vectors. SVM has a technique called the kernel trick (Chen et al., 2017). The Kernel Trick is a technique in machine learning to avoid some intensive computation in some algorithms, which makes some computation goes from infeasible to feasible. According to the study by Chen et al. (2017) the kernel trick can be used to generalize the linear ranking problem to a non-linear case.

2.2.8 K-Nearest Neighbours. KNN is a versatile supervised machine learning algorithm that is used for classification and regression. It is sometimes called lazy learning and also non parametric method (Zhang and Zhou, 2007). KNN consists of training examples with labelled and is one of the simplest machine learning algorithm. The reason why KNN falls under supervised learning it is because it uses the class labels of the training data. In KNN we choose, k, the number of nearest neighbours (NNB). There are several ways to choose the value of k, it depends on the data and whether the problem is classification or regression. k can be chosen as the square root of n samples or be chosen as odd numbers to avoid bias. There are different types of metric spaces used in KNN, but Euclidean metric space is used often in KNN. From the study of Hu et al. (2016) euclidean and Minkowsky distance functions perform the worst over some other mixed type of datasets. This gives us a reason that using different metric spaces depend on the data used.

Given A=$(x_1,x_2,\ldots, x_n)$ and B = $(y_1, y_2, \ldots,y_n)$, where n is the dimensionality of the feature space. To calculate the distance between A and B, the Euclidean distance formula is generally given by:

$$D(A,B) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}, \tag{2.2.26}$$

Other distance metric measure to mension but few we have Hamming, Manhattan distance. According to Hu et al. (2016) the lazy learning algorithm directly searches through all the training examples by calculating the distances between the testing and the training examples in order to identify its NNB and produce the classified output. In regression we sum the distance between the averages.

2.3 Activation Functions

To understand the use of activation functions (AF) we have to understand the structure of ANN. NN are based on how a neurons works in a human brain. Figure 2.5 below shows a ANN that works like a human neuron that get electrically excitable cell that receives, processes, and transmits information through electrical and chemical signals (Glorot and Bengio, 2010). The AFs main duty is to decide on weather a neuron should be activated or not. AF in NN are used to determine the firing of neurons in a NN. AF acts in such a way that it can learn nonlinear pattens of data (Glorot and Bengio, 2010). AFs are non linear and differentiable. Their differentiability helps into BP errors to improve efficiency or optimize the weights. Consider a neuron model below:

$$\mu_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.3.1}$$

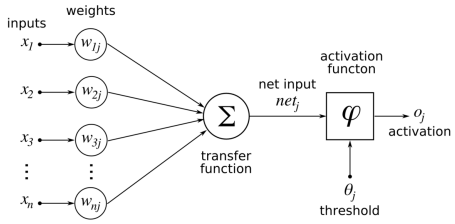


Figure 2.5: brain neuron, (Zhang .H ,2017).

In Figure 2.5 we have n inputs x_1,x_2, x_3,..., x_n which are inputs that will be multiplied with weights w_{k1},w_{k2}, w_{k3},...,w_{kn} called synaptic weights and the summing functions $\sum$ that sums the products of x_j and w_{kj}. Where k is the k^{th} neuron in a single layer. b_k is the bias, the bias is an external parameter that is used in an artificial neurons. Again in Figure 2.5 we have ϕ that is our AF. Consider the mathematical form of a neuron below:

$$\mu_k = \sum_{j=1}^{m} w_k j x_j, \tag{2.3.2}$$

$$y_k = \phi(\mu_k + b_k), \tag{2.3.3}$$

In (2.3.2) we have x_j which are the input or inputs values to our neuron model. w_k are the weights, where k is the k^{th} neuron in a single layer of a NN. In (2.3.3) we have ϕ which is the AF that. y_k is the output signal fired by our AF and b_k is our bias. AFs that are most used are define in Subsections 2.3.1, 2.3.2 and 2.3.3

2.3.1 Sigmoid function.

$$\phi(v) = \frac{1}{1 + e^{-v}} \tag{2.3.4}$$

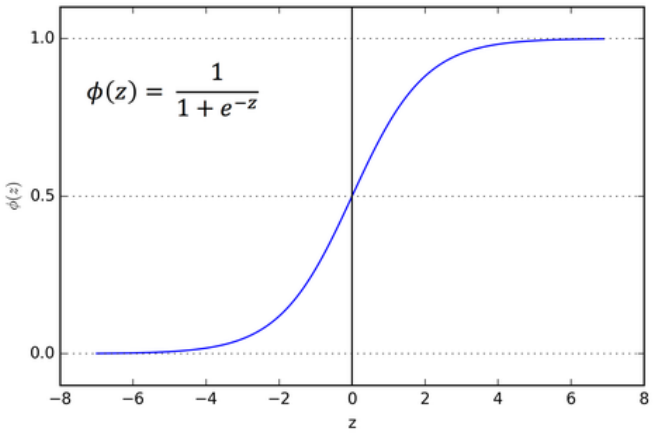


Figure 2.6: Sigmoid activation function, (Sharma .S ,2017).

The sigmoid function, described in 2.3.4, is frequently used in NNs. It has a characteristic of a an S or curved shape. It is the special case of a statistical function called logistics function (Glorot and Bengio, 2010). Sigmoid function takes values of a range between $-\infty$ to $+\infty$ and squeeze them and return the values in the range between 0 and 1. Using (2.3.4) it is on point that, if we input ϕ to be zero, as $\phi(0)$, it is evident in Figure 2.6 that the output will be 0.5 because of the S shaped curve intersect with the y axis at z=0.

2.3.2 Tanh function. . Consider the $tanh$ function below (Chris A ,2017): .

$$\phi(v) = \frac{e^{v} - e^{-v}}{e^{v} + e^{-v}} \tag{2.3.5}$$

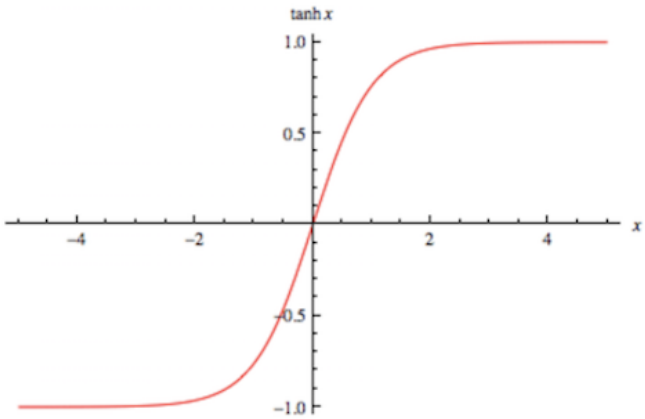


Figure 2.7: Tanh activation Function Figure, (Chris A ,2017).

In Figure (2.3.5) we have a tach function which in mathematics it is also knows as hyperbolic function. The $tanh$ function it is the rescalling of the logistic or sigmoid funtion in such a special case that the output lies between -1 and 1 whilist in sigmoid lies between 0 and 1. See Figure 2.3.5 and 2.3.4 for more clarification. According to the study by Zhang and Zhou (2006) the $tanh$ function is continuous and differentiable at all points, in addition we can easily BP the errors.

2.3.3 ReLu function.

$$\phi(v) = \max(0, v) \tag{2.3.6}$$

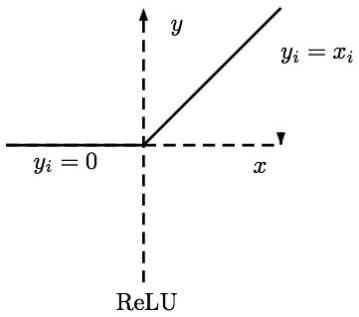


Figure 2.8: ReLu activation function, (Walia A ,2017).

ReLu- rectified linear unit functions has become very popular in the past couple of years. It was recently proved that it had an improvement in convergence from $\tanh$ function (Maas et al., 2013). It is evident from Figure (2.3.6) that the function return 0 or its input value v. The relu function has limitations is that a it can only be used within the layers of a network for good results, hence for this reason we can use softmax function. The softmax function is the extension of the sigmoid function for more than two values (Bouchard, 2007). The ReLU function avoids and rectifies the vanishing gradient problem.

2.4 ML Optimization Algorithms

Optimization algorithms are used in NN models to produce better results faster by modifying or updating the model parameters such as weights and biases. Optimization algorithms are used in training phase to minimize or maximize the value of an objective function with respect to the parameter of that functions (Bottou, 2010).

2.4.1 Stochastic Gradient Decent. Stochastic Gradient Decent (SGD) minimize and optimize differentiable objective functions in terms of iteration. According to the study by Bottou (2010) optimization algorithms such as SGD gives an amazing performance for large data set problems. Minimizing objective functions, for large data set, gradient decent become more computationally expensive procedure.

Given a loss function as $J(\theta_0, \theta_1)$, (2.4.1) shows the gradient decent algorithm. The iteration is repeated until convergence.

$$\theta_j := \theta_j - \eta \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1), \tag{2.4.1}$$

where $j = 0$ and $j = 1$ get updated simultaneously per iteration. η is our learning rate.

From (2.4.1) if η is too small, gradient descent can be slow. however, if η is too large, gradient decent can overshoot the minimum and it might not converge or diverge. As we approach the local minimum, gradient descent will automatically take smaller steps. Hence, there is no need to change the learning rate η over time.

2.4.2 Batch Gradient Descent. Using batch gradient decent. In order to minimize the loss function $J(\theta_0, \theta_1)$, the average can be calculated from n training set. where n is the whole data set (Konečnỳ et al., 2016). If the data set is too much, thousands and million, batch gradient decent can be costly since we need to evaluate the entire training dataset at a time to the local minimum of our function.

the batch gradient decent is given below in (2.4.2):

$$\theta_j := \theta_j + \sum_{i=1}^{n} (y^i - h_\theta(x^i))x^{i_j}, \tag{2.4.2}$$

2.4.3 Mini Batch Gradient Descent. Batch gradient descent uses all m examples in each iteration, while stochastic gradient uses 1 example in each iteration. Hence, in mini batch gradient descent(MBGD) we use b examples per iteration (Konečnỳ et al., 2016), where b is the mini batch size.

Givem b examples $(x^i, y^i),\ldots, (x^{i+b}, y^{i+b})$ then using (2.4.3) below:

$$\theta_j := \theta_j - \eta \frac{1}{b} \sum_{k=i}^{i+b} (h_\theta(x^k) - y^k)x_j^k, \tag{2.4.3}$$

where $i := i + b$, the function will converge after running through the necessary number of iterations.

2.4.4 Adam optimization algorithm. Adam optimizer it is an optimizer that minimises or maximises an objective function in a high computational speed. the name Adam is derived from adaptive moment estimation. Adam is a SGD that is used mostly because of better adoption in deep learning application and natural language processing (NLP) (Kingma and Ba, 2014).

To implement Adam optimization algorithm :

Initializing momentum updates are $V_{dw} = 0$ and $V_{db} = 0$. Initializing RMSprop as $S_{db} = 0$ and $S_{dw} = 0$, where RMSprop is an unpublished, adaptive learning rate method proposed by Geoff Hinton (Tieleman and Hinton, 2012).

Subscripts dw and db are computed using current mini-batch Gradient decent. Below we have a momentum update with hyper-parameter β_1.

$$V_{dw} = \beta_1 V_{dw} + (1 - \beta_1)dw, \tag{2.4.4}$$
$$V_{db} = \beta_1 V_{db} + (1 - \beta_1)db, \tag{2.4.5}$$
$$\tag{2.4.6}$$

Below we have RMSprop update with hyperparameter β_2

$$S_{dw} = \beta_1 V_{dw} + (1-\beta_1)dw^2, \tag{2.4.7}$$
$$V_{db} = \beta_2 S_{db} + (1-\beta_2)db, \tag{2.4.8}$$
$$\tag{2.4.9}$$

where dw is the first moment and dw^2 is the second moment.

In the typical implementation of Adam, bias correction is implemented. The bias correction implementation is given in the equations below:

$$V_{dw}^{corrected} = \frac{V_{dw}}{1-R^t}, \tag{2.4.10}$$
$$V_{db}^{corrected} = \frac{V_{db}}{1-R^t}, \tag{2.4.11}$$

$$S_{dw}^{corrected} = \frac{S_{dw}}{1-R^t}, \tag{2.4.12}$$
$$S_{db}^{corrected} = \frac{S_{db}}{1-R^t}, \tag{2.4.13}$$

Finally we perform the updates in (2.4.14) and (2.4.14):

$$w := w - \alpha \frac{V_{dw}^{corrected}}{\sqrt{S_{dw}^{corrected}}+\epsilon}, \tag{2.4.14}$$
$$b := W - \alpha \frac{V_{db}^{corrected}}{\sqrt{S_{db}^{corrected}}+\epsilon}, \tag{2.4.15}$$

Even though, Adam has a large number of parameters, the learning rate hyperparemeter is still important. Table 2.4 below shows the default setted parameters for Adam:

α	Need to be tuned.
β_1	Common choice is 0.90, it is the weighted average. dw
β_2	Commonly 0.99, weighted moving average of dw^2
ϵ	Most of people use 10^{-8}

Table 2.4: Adam parameters, (Kingma and Ba, 2014).

3. Methods Used

3.1 Introduction

This chapter is divided in to four sections. Section 3.3 presents the description of the data in terms of statistical summary, it also give the method of interpolation used in this study and method used to normalise data. Section 3.3 presents the model used in this study, model architecture and finally how data is portioned in terms of percentages. In Section 3.4 we have persistence forecast model that is used to benchmark long short term memory (LSTM). Section 3.5 presents evaluation measurements used for both models.

3.2 Data Description and Processing

3.2.1 Statistical Description. The wind speed data used in this study was extracted from the wind atlas of South Africa (WASA) (WASAData, 2010). Station 13 is at memel, free state province in South Africa and Station 14 is at jozini, Kwa-Zulu-Natal in South Africa. Tables 3.1 and 3.2 compare statistical summary of the wind speed data set in terms of mean wind speed (WS), standard deviation WS, minimum WS, median WS and maximum wind WS mean. Tables 3.1 and 3.2 shows the descriptive summary of wind speed at height 40m, 60m and 62m for station 13 and 14.

height(m)	data count	mean WS	std WS	min WS	median WS	max WS
60m	17300	4.83	2.65	0.23	4.47	19.67
62m	17300	4.86	2.67	0.22	4.51	19.69
40m	17300	4.39	2.52	0.20	3.95	19.60m

Table 3.1: Station 13 data Summary

height(m)	data count	mean WS	std WS	min WS	median WS	max WS
60m	17300	7.27	3.51	0.23	6.74	28.17
62m	17300	7.35	3.52	0.23	6.80	28.16
40m	17300	6.91	3.31	0.23	6.38	27.48

Table 3.2: Station 14 data Summary

3.2.2 Interpolation. During data generation measurements at the stations sometimes problems are encountered in data recording. Due to this, while gathering data for analysis, we encounter missing values due to sensor failures (Wei et al., 2010). The data used had missing values with an hour or less. So, due to consistency and to be able to get meaningful results we had to use interpolation method called cubic spline (Chawla and Subramanian, 1988). According to Chawla and Subramanian (1988) cubic spline method it is regarded as one of the best method to use on non-linear data. In most cases TS wind data contains noise. Cubic Spline Interpolation normally use the following in (3.2.1).

$$y = Ay_1 + By_{i+1} + Cy_j^{''} + Dy_{j+1}^{''}, \tag{3.2.1}$$

where $A = \frac{x_{j+1}-x}{x_{j+1}-x_j}$, $B = 1 - A$, $C = \frac{1}{6}(A^3 - A)(x_{j+1} - x_j)^2$ and $D = \frac{1}{6}(B^3 - B)(x_{j+1} - x_j)^2$

3.2.3 Data Normalisation and Scaling. Training Neural networks and Long short term memory the day need to be normalised and scaled completely. The data was normalised using the scikit-learn object MinMaxScaler. Normalization it happens when rescaling of the data from the actual range so that all values are between the range of 0 and 1. Normalisation is necessary to remove noise and normalise strokes (Huang et al., 2007). Normalisation formulae is given below in (3.2.2).

$$y = \frac{x - min}{max - min}, \tag{3.2.2}$$

where x, is a data point from the TS dataset and $(min \ \& \ max)$ are minimum and maximum.

3.3 LSTM Model

3.3.1 LSTM Compared to RNN. This section explains a simple understandable LSTM. Long Short Term Memory is special case of a NN specially designed to prevent the NN output from decaying or exploding (Malhotra et al., 2015). LSTM provides a better performance as compared to RNN. LSTM are known to be able to overcome the problem of long term dependency and works better in cases where the space or gap between the information needed and the place it is wanted its small. According to the study by Zafeiriou et al. (2016) RNNs and LSTMs have the form of a repeating module and among those modules there is a very simple structure, its simplicity in the module will be of a single sigmoid or tanh layer. The single RNN layer is given below in Figure 3.1.

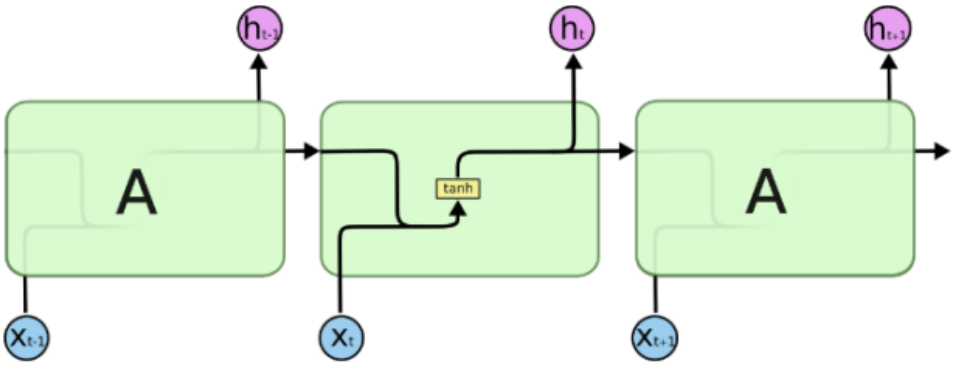


Figure 3.1: module in RNN single layer, (Kina .O, 2015)

LSTMs are build in the form of repeating module but with internal structure that is different from the RNN. LSTM is different from a RNN in such a way that there are four interacting layers in one module. The structure of LSTMs is given in Figure 3.2.

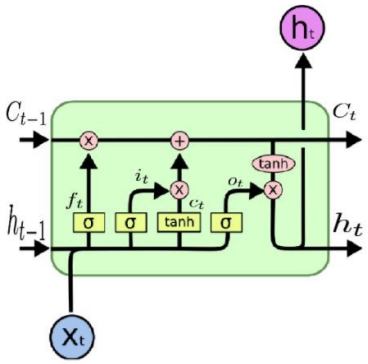


Figure 3.2: Long Short Term Memory, (Kina .O, 2015)

3.3.2 LSTM Architecture. Figure 3.2 shows the LSTM structure that is made up of blocks of memory cells. LSTM take care of vanishing and exploading gradient problems. The memory blocks are made up of three gates called forget, input and output gates. All these gates have their own set of weights. There are two states that connect blocks in an LSTM structure namely cell C and hidden state h. LSTM in Figure 3.2 consist of three gates namely input, output and forget gate. From 3.3.1 we have a forget gate f_t that help us to determines on which input feature is important or not. Forget gate combines the previous feature output h_{t-1} and the current input x_t in the hidden state to figure out which one is important for the state.

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{3.3.1}$$

From 3.3.2, i_t is a gating signal that depends on the input h_{t-1} that is from the hidden layer as previous output and is multiplied by the weight matrix i, W_i together with x_t and passed through the activation function sigmoid σ.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{3.3.2}$$

In (3.3.3) we have the new candidate $\tilde{C}_t$ with two inputs layer combined together with a weight matrix c, W_c and they are passed through hyperbolic $tanh$. $\tilde{C}_t$ is called a temporary state or new candidate state that also depends on the input and the previous output.

$$\tilde{C}_t = \tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{3.3.3}$$

From Figure 3.2 we have C_t, which in (3.3.4) is the previous state multiplied by f_t which is the gating mechanism and C_{t-1} is from the previous cell state block. Then we add that with the product of i_t and $\tilde{C}_t$.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3.3.4}$$

Figure 3.2 presents the hyperbolic $tanh$ function which basically process the inputs and squeeze the values between -1 and 1 and hence, multiply by the output sigmoid σ.

From (3.3.6), we have the output gating which is h_t, it is considered at the current state and is passed through σ. Hence, the output o_t in (3.3.5) multiplied by the $tanh$ of the current state C_t.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{3.3.5}$$

$$h_t = o_t \tanh(C_t) \tag{3.3.6}$$

The equations above presents the process behind LSTM step by step. There are different ways of building LSTMs. With a slightly difference from the one explained above. According to the study by Sak et al. (2014) the upgraded classic LSTM structure contains peep-hole connections from its internal cells to the gates in the same cell to learn precise timing of the outputs.

3.3.3 LSTM Model Fit. To be able to feed Time Series (TS) data in the LSTM model we had to convert the data in to a supervise learning format where $y = f(x)$. Shift function is used to help in transforming TS data to a supervised learning problem, made simply with the pandas shift function (Graves, 2012). Loading a data frame using pandas libraries, the shift function was used to create multiple columns. The columns will appear in the form of output and inputs. Time frame may be consisting of at least 1 feature(s).

Then data sets were first split in to training, validation and test data for both stations. The data split are shown in Table 3.3.

Table 3.3: Data Split

Type	%
Training	80
Validation	10
Test	10

After performing the data split in Table 3.3, we had to reshape our data in to different dimensions. Our input data set was reshaped to three dimensions and output data in to one dimension. The input data was reshaped in to number of samples, time-steps and number of features. Hence our output was univariate. The outline of layers used and activation functions are given in Figure 3.3.

Figure 3.3 show the layers used in our LSTM architecture. The input layer has 64 units of blocks and a drop-out of 20%. Hidden layers has 32 units each and drop-out of 30% and 20% respectively. Drop-out can be described regularization method where input and recurrent connections to LSTM units are have a chance to be neglected from the activation function and weight updates when training the model or network. The activation function used in our dense layer is called sigmoid function and is explained in section 2.3.1. The Optimiser used is Adam and is explained in subsection 2.4.4.

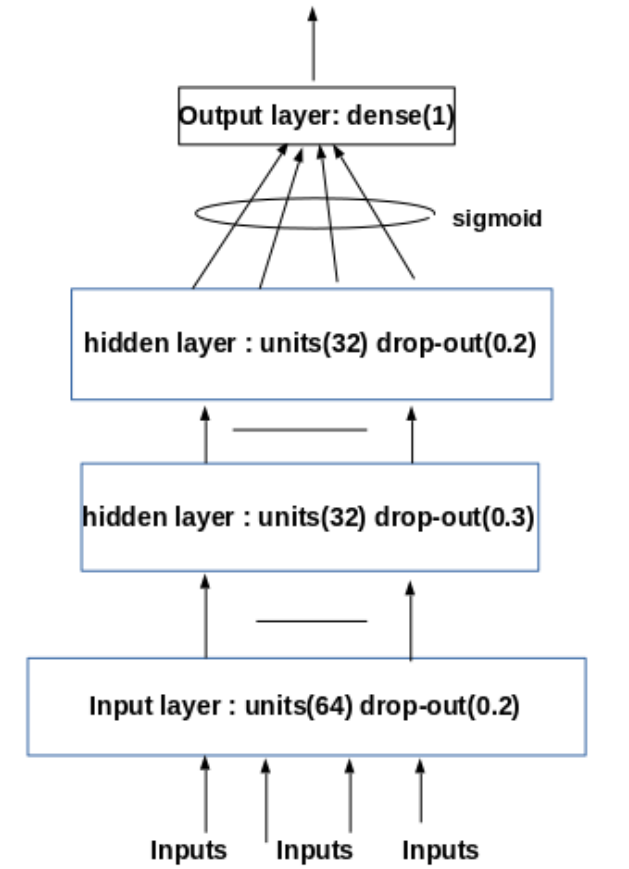


Figure 3.3: LSTM Architecture used

3.4 Persistence Model

In this study a persistence model was used as a benchmark model. A Persistence Algorithm it is also known as the (naive) forecast. Persistence Algorithm uses the dataset values of the previous time step $t-1$ to forecast the outcome of the next step t. According to the study by Cheng et al. (2017) a model which is achieving good model efficiency may actually be of lower quality component to the baseline model (or persistence) forecasting, if the flow series has a high lag-1 autocorrelation coefficient. From the study of Van der Walt and Botha (2016) article used persistence forecast or naive base model where (the previous measurement is used to estimate the next) to nearly exact the wind speed as an initial forecast benchmark.

Given the sequence x_t, for t=1,2,3,. . . ,n. predicting x_t using previous time step x_{t-1} is given in (3.4.1):

$$\hat{x_t} = x_{t-i}, \tag{3.4.1}$$

where i is the number of time steps or the look back. Evaluation of the persistence forecast for regression on TS data we use Root Mean Square Error (RMSE), which is explained in the Section 3.5. According

to the study by Moriasi et al. (2007) the mean absolute error (MAE), Mean Square Error (MSE), and root RMSE are error indices were commonly used in model evaluation.

3.5 Evaluation Measurements Used

This study uses a LSTM model to forecast the wind speed mean and Persistence Forecast Model to function as a benchmark for forecasting accuracy. According to the study by Ismail et al. (2011) statistical evaluation metrics for evaluation used was MAE and RMSE.

3.5.1 Mean Absolute Error (MSE). MAE measures the relative average of the errors in a set of forecasts, with no intend of taking in to account on their direction. It is the mean over the test sample of the absolute discrepancy between prediction and real values observation where all individual differences have same weight (Goharnejad et al., 2014).

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_i - \hat{y}_i| \tag{3.5.1}$$

From Figure (3.5.1) y_i is the real values observation and $\hat{y}_i$ are the predicted values.

3.5.2 Root mean square(RMSE). RMSE measures the average magnitude of the error. According to Willmott (1982) RMSE is the square root of the average of squared differences between prediction and actual observation.

$$MSE = \frac{1}{n}\sum_{j=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.5.2}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.5.3}$$

From (3.5.2) and (3.5.3), y_i is the real values observation and $\hat{y}_i$ are the predicted values. RMSE gives a relatively high weight to large errors compare to MAE due to taking the square root of the errors. From the study of Willmott (1982) RMSE and MSE are natural measures to use in many forecast error evaluations that use regression-based and statistical methods.

4. Findings and Discussions

This chapter is divided into four sections. In section 4.1, 4.2 and 4.3, the results are presented. The results from these sections are discussed in section 4.4. Section 4.2 shows the RMSE's results obtained using our benchmarked model LSTM, in addition it shows the figures produced for Station 13. In section 4.1, persistence model results in terms of Root Mean Square Error (RMSE's) are presented. Furthermore, the persistence RMSE's figures produced in section 4.3 shows the improvement of the results using an increased number of epochs per time step ahead. In section 4.4, the results obtained in this study by comparing the LSTM model RMSE's results to the persistence results are discussed.

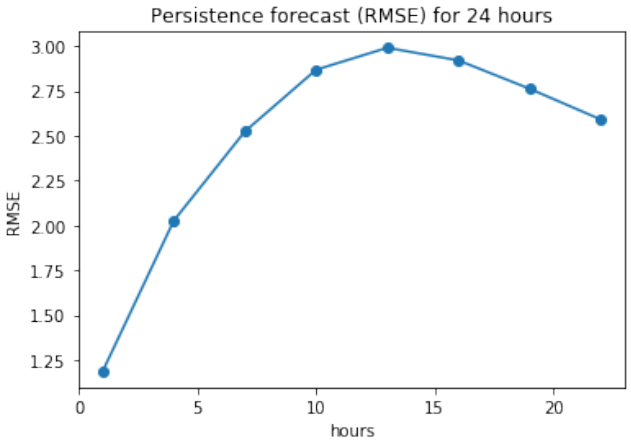
4.1 Persistence Forecast

Persistence forecast is used as a benchmark model to evaluate the LSTM-RMSE results for Station 13. Table 4.1 presents the RMSE results obtained from the persistence forecast model. According to the study by Van der Walt and Botha (2016), the RMSE for wind speed using persistence forecast RMSE's reaches the maximum at the 12th hour and reaches the local minimum at 24th hour due to the 24 hour cyclical factors. It is evident from Table 4.1 that the RMSE error at 1 hour has a lower RMSE for both heights, that is 60m and 40m. The RMSE for height 60m at 24th hour is lower than the RMSE at height 60m for 12th hour ahead by 21.03%. Similarly, for height 40m, the RMSE for 24th hour ahead is lower than that of the 12th hour by 20.64%. Hence, this is due to the 24 hour cyclic behaviour of wind speed (Van der Walt and Botha, 2016).

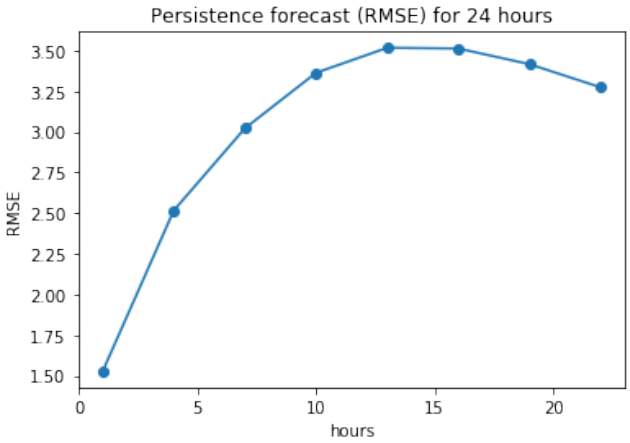
Table 4.1: Persistence Model RMSE's for Station 13

Station Number	Height (m)	1 hour ahead	12 hour ahead	24 hour ahead
13	60m	1.61	3.69	3.46
13	40m	1.53	3.52	3.24

Figure 4.1 shows the persistence forecast RMSE results for the next 24 hour using heights 60m and 40m. Figure 4.1a illustrates the 24 hour cyclic behaviour of wind speed at height 60m and Figure 4.1b shows the 24 hour cyclic behaviour of wind speed at height 40m. Figure 4.1 RMSE results are supported by the RMSE results shown in Table 4.1.



(a) Persistence forecast results for the next 24 hours at height 60m



(b) Persistence forecast results for the next 24 hours at height 40m

Figure 4.1: Persistence (RMSE) plots for Station 13 at height 60m and 40m

4.2 LSTM Results

The architecture of the model used is explained well in Section 3.3. Table 4.2 is divided in to Station number, height of the windmill at each Station and RMSE's at each time step ahead (1, 12 and 24) hours. In this Section, we are trying to improve the accuracy of wind speed at Station 13 using wind speed features at Station 14. The mean wind speed mean used was at height 40m, 60m and 62m. Looking at the RMSE's on Table 4.2 it is evident that, error decrease as we add the features from another Station. The RMSE's results in Table 4.2 will be compared with the Persistence RMSE's results in Table 4.1 later in Subsection 4.4. The plots in Figures 4.2, 4.3 and 4.4 shows the Wind speed forecast at 1, 12 and 24 hour respectively from Station 13.

Table 4.2: LSTM RMSE's for Station 13

Station Number	Height (m)	1 hour ahead	12 hour ahead	24 hour ahead
13	60m	0.63	0.70	0.72
13 and 14	60m and 60m	0.62	0.69	0.70
13 and 14	60m and 62	0.69	0.687	0.69
13	40m	0.63	0.65	0.63
13 and 14	40m and 40m	0.63	0.63	0.62

Figure 4.2 shows the mean wind speed prediction of the LSTM model for the next hour. Figure 4.2 is plotted in terms of persistence forecast for wind mean speed and the LSTM at 1 hour ahead. In Figure 4.2 the actual mean wind speed at height 60m is compared to the foretasted wind speed at height 60m and the wind speed at height 60m with an added mean wind speed feature height from Station 13 at height 60m. Results that support Figure 4.3 results and the architecture used to obtain the figure above are explained more in Subsection 4.4.

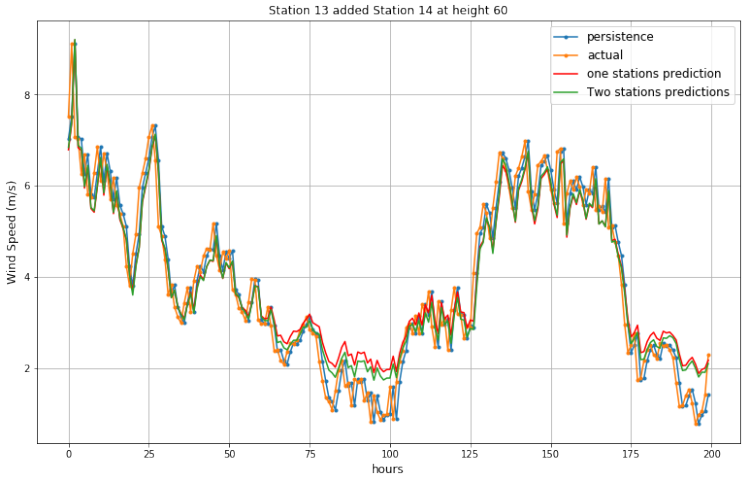


Figure 4.2: The LSTM mean wind speed forecast at height 60m for the next hour.

Figure 4.3 shows the mean wind speed prediction of the LSTM model for the next 12 hour. Figure 4.3 is plotted in terms of persistence forecast for wind mean speed and the LSTM at 12 hour ahead. In Figure 4.3 the actual mean wind speed at height 60m is compared to the forecasted wind speed at height 60m and the wind speed at height 60m with an added mean wind speed feature height from Station 14 at height 60m. Figure 4.3 shows the dependency of wind speed with respect to hours.

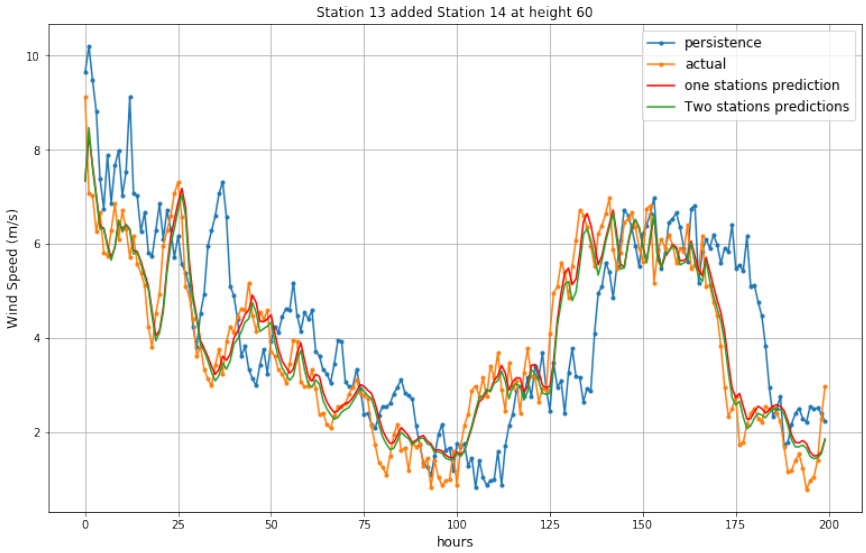


Figure 4.3: The LSTM mean wind speed forecast at height 60m for the next 12th hour.

Figure 4.4 shows the mean wind speed prediction of the LSTM model for the next 24th hour. Figure 4.4 is plotted in terms of persistence forecast for wind mean speed and the LSTM at 24 hour ahead. Figure 4.3 shows the dependency of wind speed with respect to hours. In Figure 4.4 the actual mean wind speed at height 60m is compared to the foretasted wind speed at height 60m and the wind speed at height 60m with an added mean wind speed feature from Station 13 at height 60m.

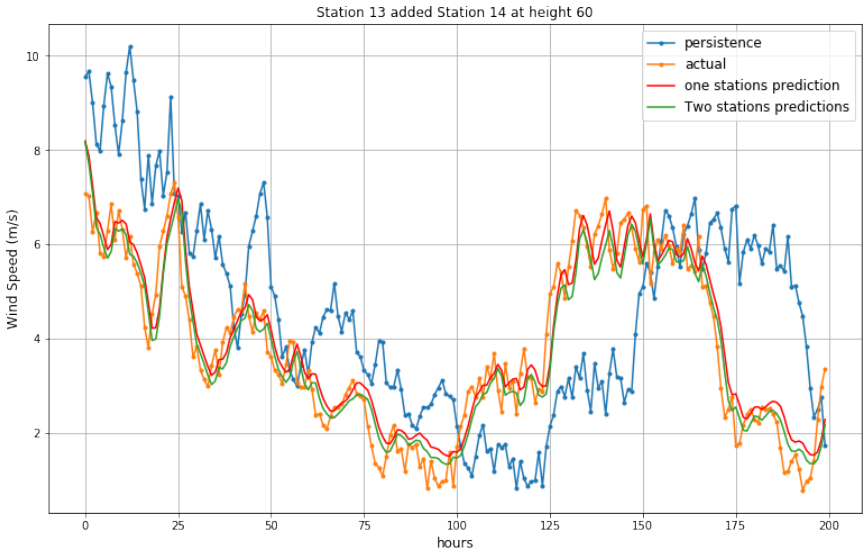


Figure 4.4: The LSTM mean wind speed forecast at height 60m for the next 24 hour.

4.3 Improve LSTM Model by Increasing Number of Epochs

Table 4.3 shows the RMSE's results from Station 13 mean wind speed, as well as the added mean wind speed features from Station 14 at height 40m, 60m and 62m. The RMSE's results are obtained using LSTM and they will be compared with persistence forecast results given in Table 4.1. The RMSE's results are calculated for 1, 12 and 24 hour ahead for both Stations. Mostly, training the LSTM sometime goes under fit or over fit because of the number of epochs. By increasing the models epochs, we are giving the model the time to learn the data for better results. The LSTM model has been trained using 50 and 100 epochs which is recorded in Table 4.3. The data split was in the form of training, validation and test data sets. The model has been validated using the loss function. The loss function which is also knows as cost function is explained in 2.2.5. The loss function plots are given in Figures 4.5 and 4.6.

Table 4.3: LSTM results with increased number epochs

		50 Epochs			100 Epochs		
Station Number	Height (m)	1 hour	12 hour	24 hour	1 hour	12hour	24 hour
13	60m	0.63	0.7	0.72	0.68	0.68	0.59
13 and 14	60m and 60m	0.74	0.69	0.70	0.68	0.69	0.58
13 and 14	60m and 62	0.63	0.68	0.69	0.65	0.68	0.57
13	40m	0.63	0.65	0.63	0.62	0.63	0.62
13 and 14	40m and 14	0.63	0.63	0.63	0.63	0.62	0.62

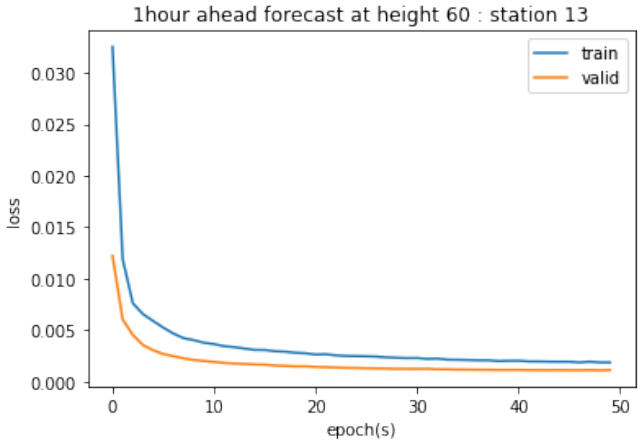
4.4 Discussions

4.4.1 Persistence Forecast. Persistence model use a value from the previous time step to predict the current time step. Persistence model results are shown in Table 4.1. The evaluation measurement used is called Root Mean Square Error (RMSE). For all Stations the RMSE increases by at most 30% at each horizon hour. As mentioned in Section 4.1, the RMSE for height 60m at the 24th hour is lower than RMSE at height 60m at 12th hour by 21.03%. Similarly at height 40m, the RMSE for 24th hour is lower than of 12th hour by 20.644%. This is due to a 24 hour cyclic behaviour of wind speed. The Persistence results in Table 4.1 will be used as evaluation comparison measure for the LSTM model results in Table 4.3. The LSTM RMSE mean wind speed at all height in Figure 4.3 are all less than RMSE of the persistence at all height. hence, it simply means the LSTM model performs well (Soman et al., 2010).

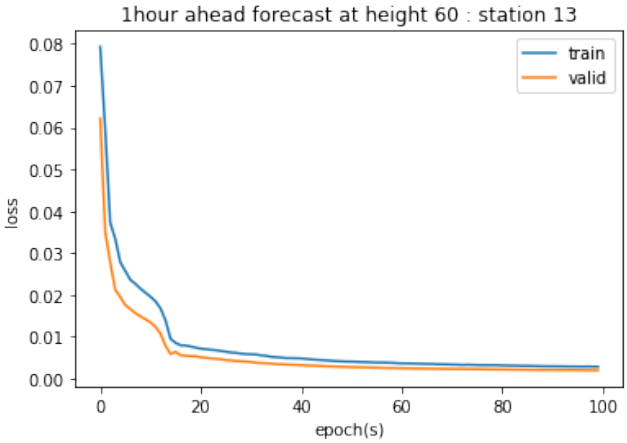
4.4.2 Loss Function. The data split was in the form of training, validation and test data, whereby we use training data to train the model and use validation data to validate the model, by checking over-fitting and under-fitting. The loss is calculated on training and validation. The optimization used is called the Adam and it is explained in Chapter 2, Subsection 2.4.4. Adam optimizer use default parameters to learn. Figure 4.5 shows that the loss decrease as the number of epochs increase. Figure 4.5 shows the two plots with 50 and 100 epochs for the next 1 hour and 12 hour ahead. Table 4.3 shows that, the error decrease as the number of epochs are increased from 50 to 100. The approach used to choose the number of epochs, can be called early stopping. Which is the number of epochs where the validation loss interests with training loss.

4.4.3 LSTM Model. The RMSE's results in Table 4.2 for height 60m and 40m were able to beat baseline model RMSE results in Table 4.1 for all specified time step. Hence this shows that our model perform good since, for better model, then RNSE results for our LSTM model must always be smaller than that of our baseline model. The error decreased as we add features from Station 14 for all time steps (1, 12 and 24 hour ahead). Figure 4.2, 4.4 and 4.4 shows the plots of the results shown in 4.2. The decrease of error for all time horizons produces good plots, that really show a good fit of data for the LSTM data. looking at Station 13 results in Table 4.3 we can spot the effect of number of epochs as we train the model by the RMSE test results adding more epochs to the model can cause over-fitting or under-fitting of the model. val loss and training loss plots can be seen from Figure 4.5.

The type of LSTM model used is a sequential model. The model is called sequential model because it consists of linear stack of layers. The data was farmed from time series to supervised learning. The data input to the LSTM had to be of three dimensions. The model architecture used has four layers, namely input layer with 64 units and dro-out of 20%, 2 layers each consists of 34 units and dropout of 30% and 20 % and finally the dense layer which has an activation function sigmoid. Words such as dro-pout, epoch, layer and units are shown well explained in Chapter 2 and shown in Figure 3.3

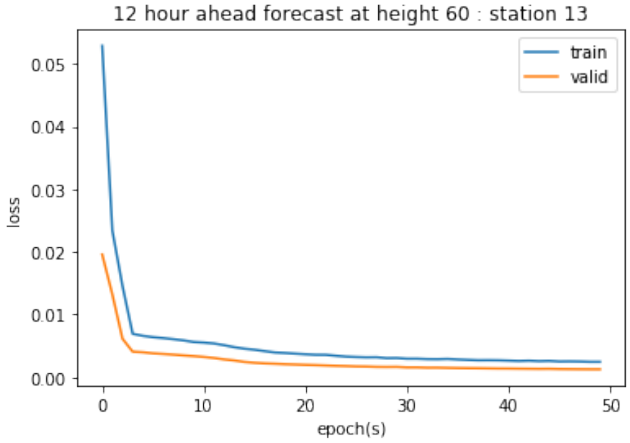


(a) Station 13: 50 Epochs at height 60m

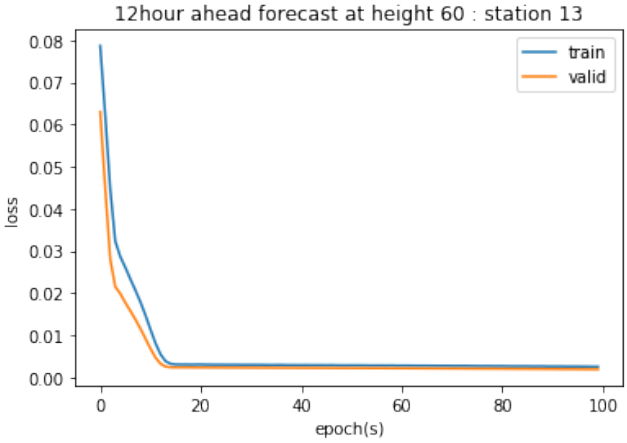


(b) Station 13: 100 Epochs at height 60m

Figure 4.5: 1 hour ahead



(a) Station 13: 50 Epochs at height 60m



(b) Station 13: 100 Epochs at height 60m

Figure 4.6: 12 hour ahead

5. Conclusion and Future Work

5.1 Conclusion

This project focus on improving mean wind speed forecast accuracy on one station by adding mean wind speed features from another station. The stations used are from two neighbouring provinces in South Africa. The reason for the choice of stations is similarity in the wind speed behaviour due to cyclical behaviour of wind speed on daily basis. The increase accuracy forecasting mean wind speed will help to plan for the approximate time needed to generation electricity power from the turbines.

To accomplish the aim of the study, we employed two models, persistence model and long Short term memory (LSTM). The LSTM model was benchmarked using persistence forecast model (baseline model). Both models were used to forecast at 1, 2 and 24 hour ahead. The height used were 40m and 60m. The evaluation measurements used in this study is called Root Mean Square Error (RMSE). The RMSE's for LSTM model were all lower than that of the persistence Forecast model. Hence, the LSTM results outperformed the persistence forecasts, which simply means that the LSTM model performed good. However, the persistence forecast for mean wind speed at height 60m and 40m were showing the 24 hourly cyclic behaviour of wind speed.

The study revealed that indeed adding mean wind speed features from another station can highly increase the accuracy. Choosing the right number of epochs and the right optimization helped us to improve the model's accuracy, since our loss function plots were all not over-fitting or under-fitting.

5.2 Future Work

In future work, the study might include comparing bidirectional LSTM and unidirectional LSTM with multiple inputs and compare the results by using at least two stations. Persistence forecast can still be used to benchmark the models. Hence, RMSE can still be used as a evaluation measurements.

Acknowledgements

I would start by thanking GOD for taking me this far with studies and granting me the opportunity to explore AIMS.

Will also like to thank Prof Neil Turok, Prof Barry, Jeff Sanders, Jan Groenwald, Igsaan Kamalie and all the facilitators who came from overseas and local to teach us.

I would like to thank my supervisor Nicolene Botha for being patient in me from the start till the end of this project. I will also also like to thank my co-supervisors Dr Vukosi Marivate and Dr Bubacarr Bha. Will like to thank Nolu and Lebeko for the support and tutoring.

A special thank to my parents Mrs Hildah Senoamadi and Mr Simon Senoamadi for giving me life and raising me to be the man i am today. I would also like to thank my friends Zakiena, Daniel, TJ, Sne, Mvubu, Kaode, Jordan, Tumisang for helping be to put this project together and my siblings and Jerminah for always being there for me.

References

Annampedu, V. and Wagh, M. D. Decomposition of threshold functions into bounded fan-in threshold functions. *Information and Computation*, 227:84–101, 2013.

Bersini, H. and Gorrini, V. A simplification of the backpropagation-through-time algorithm for optimal neurocontrol. *IEEE transactions on neural networks*, 8(2):437–441, 1997.

Blue World Carbon ,2013. South african grid connected wind farm programme. Analytics Vidhya, http://www.blueworldcarbon.com/south-african-grid-connected-wind-farm-programme-2/, Accessed September 2018.

Botha, N. and Van der Walt, C. M. Forecasting wind speed using support vector regression and feature selection. In *Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech), 2017*, pages 181–186. IEEE, 2017.

Bottou, L. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

Bouchard, G. Efficient bounds for the softmax function, applications to inference in hybrid models. 2007.

Britz D ,2015. Recurrent Neural Networks – backpropagation through time and vanishing gradients. Analytics Vidah Blog, http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/, October, 2015.

Britz .D,2015. Recurrent neural networks – introduction to rnns. Word Press, http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/, September ,2015.

Chapelle, O., Scholkopf, B., and Zien, A. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

Chawla, M. and Subramanian, R. A new fourth-order cubic spline method for second-order nonlinear two-point boundary-value problems. *Journal of Computational and Applied Mathematics*, 23(1):1–10, 1988.

Chen, K., Li, R., Dou, Y., Liang, Z., and Lv, Q. Ranking support vector machine with kernel approximation. *Computational intelligence and neuroscience*, 2017, 2017.

Cheng, K.-S., Lien, Y.-T., Wu, Y.-C., and Su, Y.-F. On the criteria of model performance evaluation for real-time flood forecasting. *Stochastic Environmental Research and Risk Assessment*, 31(5):1123–1146, 2017.

Chris A ,2017. From perceptron to deep neural nets. Medium, Article, https://becominghuman.ai/from-perceptron-to-deep-neural-nets-504b8ff616e, December, 2017.

Dayton, C. M. Logistic regression analysis. *Stat*, pages 474–574, 1992.

Deljac, Ž., Kunštić, M., and Spahija, B. A comparison of traditional forecasting methods for short-term and long-term prediction of faults in the broadband networks. In *MIPRO, 2011 Proceedings of the 34th International Convention*, pages 517–522. IEEE, 2011.

DnI Institute,2015. Building predictive model using svm and r. Generate Press, http://dni-institute.in/blogs/building-predictive-model-using-svm-and-r/, September, 2015.

Filonov, P., Lavrentyev, A., and Vorontsov, A. Multivariate industrial time series with cyber-attack simulation: Fault detection using an lstm-based predictive data model. *arXiv preprint arXiv:1612.06676*, 2016.

Foley, A. M., Leahy, P. G., Marvuglia, A., and McKeogh, E. J. Current methods and advances in forecasting of wind power generation. *Renewable Energy*, 37(1):1–8, 2012.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

Goharnejad, A., Zarei, A., and Tahmasebi, P. Comparing multiple regression, principal componant analysis, partial least square regression and ridge regression in predicting rangeland biomass in the semi steppe rangeland of iran. *Environment and Natural Resources Journal*, 12(1):1–21, 2014.

Graves, A. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*, pages 5–13. Springer, 2012.

Gulli, A. and Pal, S. *Deep Learning with Keras*. Packt Publishing Ltd, 2017.

Guo, J. Backpropagation through time. *Unpubl. ms., Harbin Institute of Technology*, 2013.

Höök, M. and Tang, X. Depletion of fossil fuels and anthropogenic climate change—a review. *Energy Policy*, 52:797–809, 2013.

Hu, L.-Y., Huang, M.-W., Ke, S.-W., and Tsai, C.-F. The distance function effect on k-nearest neighbor classification for medical datasets. *SpringerPlus*, 5(1):1304, 2016.

Huang, B. Q., Zhang, Y., and Kechadi, M. T. Preprocessing techniques for online handwriting recognition. In *Intelligent Systems Design and Applications, 2007. ISDA 2007. Seventh International Conference on*, pages 793–800. IEEE, 2007.

Humpert, B. K. Improving back propagation with a new error function. *Neural Networks*, 7(8):1191–1192, 1994.

Ismail, S., Shabri, A., and Samsudin, R. A hybrid model of self-organizing maps (som) and least square support vector machine (lssvm) for time-series forecasting. *Expert Systems with Applications*, 38(8): 10574–10578, 2011.

Joachims, T. Making large-scale svm learning practical. Technical report, Technical Report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund, 1998.

Jones, L., Zavadil, R., Grant, W., et al. The future of wind forecasting and utility operations. *IEEE Power and Energy Magazine*, 3(6):57–64, 2005.

Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):881–892, 2002.

Kim, J., Kwon Lee, J., and Mu Lee, K. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1637–1645, 2016.

Kina .O, 2015. Understanding lstm. git hub blog, http://colah.github.io/posts/2015-08-Understanding-LSTMs/, August, 2015.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Konečnỳ, J., Liu, J., Richtárik, P., and Takáč, M. Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):242–255, 2016.

Lee, D.-J. and Wang, L. Small-signal stability analysis of an autonomous hybrid renewable energy power generation/energy storage system part i: Time-domain simulations. *IEEE Transactions on Energy Conversion*, 23(1):311–320, 2008.

Lee, K. Y., Chung, N., and Hwang, S. Application of an artificial neural network (ann) model for predicting mosquito abundances in urban areas. *Ecological informatics*, 36:172–180, 2016.

Lloyd, S., Mohseni, M., and Rebentrost, P. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411*, 2013.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

Magoulas, G. D., Vrahatis, M. N., and Androulakis, G. S. Improving the convergence of the backpropagation algorithm using learning rate adaptation methods. *Neural Computation*, 11(7):1769–1796, 1999.

Malhotra, P., Vig, L., Shroff, G., and Agarwal, P. Long short term memory networks for anomaly detection in time series. In *Proceedings*, page 89. Presses universitaires de Louvain, 2015.

Manneni V ,2016. The evolution and core concepts of deep learning and neural networks. Analytics Vidah Blog, https://www.analyticsvidhya.com/blog/2016/08/evolution-core-concepts-deep-learning-neural-networks/, August, 2016.

Marquard, 2008. Costing a 2020 target of 15 Blue World Carbon, https://open.uct.ac.za/bitstream/item/19453/Marquard_Costing_a_2020_target_15_2008.pdf, Accessed September 2018.

Menard, S. *Applied logistic regression analysis*, volume 106. Sage, 2002.

Moniz, J. R. A. and Krueger, D. Nested lstms. *arXiv preprint arXiv:1801.10308*, 2018.

Moriasi, D. N., Arnold, J. G., Van Liew, M. W., Bingner, R. L., Harmel, R. D., and Veith, T. L. Model evaluation guidelines for systematic quantification of accuracy in watershed simulations. *Transactions of the ASABE*, 50(3):885–900, 2007.

Newsroom AI. The future is renewable energy. Wikipedia, the Free Encyclopedia, https://www.iol.co.za/business-report/energy/the-future-is-renewable-energy-16773019, Accessed October 2018.

Olah, C. Understanding lstm networks. *GITHUB blog, posted on August*, 27:2015, 2015.

Panwar, N., Kaushik, S., and Kothari, S. Role of renewable energy sources in environmental protection: a review. *Renewable and Sustainable Energy Reviews*, 15(3):1513–1524, 2011.

Patterson, J. and Gibson, A. *Deep Learning: A Practitioner's Approach*. " O'Reilly Media, Inc.", 2017.

Pham, D. T., Dimov, S. S., and Nguyen, C. D. Selection of k in k-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 219(1): 103–119, 2005.

Ranasinghe, R., Jaksa, M., Kuo, Y., and Nejad, F. P. Application of artificial neural networks for predicting the impact of rolling dynamic compaction using dynamic cone penetrometer test results. *Journal of Rock Mechanics and Geotechnical Engineering*, 9(2):340–349, 2017.

Ritchie, M. D., White, B. C., Parker, J. S., Hahn, L. W., and Moore, J. H. Optimizationof neural network architecture using genetic programming improvesdetection and modeling of gene-gene interactions in studies of humandiseases. *BMC bioinformatics*, 4(1):28, 2003.

Sak, H., Senior, A., and Beaufays, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association*, 2014.

Sathya, R. and Abraham, A. Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence*, 2(2):34–38, 2013.

Shafiee, M. Maintenance logistics organization for offshore wind energy: Current progress and future perspectives. *Renewable Energy*, 77:182–193, 2015.

Sharma .S ,2017. Activation functions: Neural networks. Towards Data science, Article, https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6, September, 2017.

Soman, S. S., Zareipour, H., Malik, O., and Mandal, P. A review of wind power and wind speed forecasting methods with different time horizons. In *North American power symposium (NAPS), 2010*, pages 1–8. IEEE, 2010.

Sorrell, S., Speirs, J., Bentley, R., Brandt, A., and Miller, R. Global oil depletion: A review of the evidence. *Energy Policy*, 38(9):5290–5295, 2010.

Srivastava T,2018. A must-read introduction to sequence modelling (with use cases). Wikipedia, the Free Encyclopedia, https://www.analyticsvidhya.com/blog/2018/04/sequence-modelling-an-introduction-with-practical-use-cases/, April 2018.

Tieleman, T. and Hinton, G. Rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning. *COURSERA Neural Networks Mach. Learn*, 2012.

Van der Walt, C. M. and Botha, N. A comparison of regression algorithms for wind speed forecasting at alexander bay. In *Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech), 2016*, pages 1–5. IEEE, 2016.

Walia A ,2017. Activation functions and it's types-which is better? Towards data Science, Article, https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f, May, 2017.

WASAData, 2010. Wasa meteorological data downloads. Webmaster, http://wasadata.csir.co.za/wasa1/WASAData, June, 2010.

Wei, X., Verhaegen, M., and Van Engelen, T. Sensor fault detection and isolation for wind turbines based on subspace identification and kalman filter techniques. *International Journal of Adaptive Control and Signal Processing*, 24(8):687–707, 2010.

Werbos, P. J. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

Wikipedia. Machine learning. Wikipedia, the Free Encyclopedia, https://en.wikipedia.org/wiki/Machine_learning, Accessed September 2018.

Willmott, C. J. Some comments on the evaluation of model performance. *Bulletin of the American Meteorological Society*, 63(11):1309–1313, 1982.

Wolpert, D. H. and Macready, W. G. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

Xiaoyun, Q., Xiaoning, K., Chao, Z., Shuai, J., and Xiuda, M. Short-term prediction of wind power based on deep long short-term memory. In *Power and Energy Engineering Conference (APPEEC), 2016 IEEE PES Asia-Pacific*, pages 1148–1152. IEEE, 2016.

Xu, B., Wang, N., Chen, T., and Li, M. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

Zafeiriou, S., Papaioannou, A., Kotsia, I., Nicolaou, M., and Zhao, G. Facial affect"in-the-wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 36–47, 2016.

Zhang, M.-L. and Zhou, Z.-H. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.

Zhang, M.-L. and Zhou, Z.-H. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.

Zhang, Q., Wang, H., Dong, J., Zhong, G., and Sun, X. Prediction of sea surface temperature using long short-term memory. *IEEE Geoscience and Remote Sensing Letters*, 14(10):1745–1749, 2017.

Zhang .H ,2017. Building predictive model using svm and r. Isaac Changhau, https://isaacchanghau.github.io/post/activation_functions/, May, 2017.