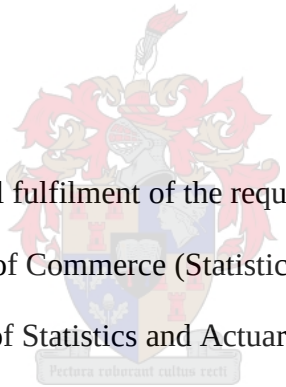
# Short-Term Wind Speed Prediction using Various Forecasting Methods

by

Maria N Nambandi



Thesis presented in partial fulfilment of the requirements for the degree of

Master of Commerce (Statistics) in the

Department of Statistics and Actuarial Science at

Stellenbosch University

Supervisor: Dr F Kamper

March 2020

# Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2020

# Abstract

There is a significant challenge in finding ways to enhance energy security and decrease greenhouse gas emissions emanating from the consumption of non-renewable resources for energy. The release of greenhouse gases causes global warming and is considered not clean. Compared to current conventional sources of energy, such as fossil resources, renewable energy sources have become more attractive for electricity production as it has been identified as clean with a closed carbon dioxide cycle. Thus, the $CO_2$ produced during processing is reabsorbed by plants for food production.

A significant development in the electricity industry in recent years has been the fast growth of wind power. The wind power generated from the wind depends on meteorological conditions such as wind speed and wind direction. These meteorological conditions are considered stochastic in nature, especially wind speed, and attempts to accurately forecast future values are therefore considered important in power generation.

There are various studies in the literature which make use of statistical techniques to predict wind speed data. In this thesis, the short-term prediction of hourly wind speeds at 60m hub height for two sites, Jozini and Memel in South Africa, over a 1 to 24-hour forecast horizon is considered. The potential short-term wind speed at a site was predicted using statistical forecasting techniques such as traditional time series models (ARMA, ARIMA, seasonal ARIMA and regression using Fourier terms with ARMA errors), multilayer perceptron (MLP) neural networks and long short term memory (LSTM) recurrent neural networks. These predictions are relevant for planning purposes to ensure that the necessary base load on the electricity grid is established at all times. All forecasting techniques were applied to forecast wind speeds for each forecast horizon and site. Different LSTM and MLP configurations were created using a different number of hidden layers, a different number of hidden nodes in each layer, different learning rates, and different activation functions. The forecast performances of each configuration were compared to the persistence forecast (benchmark model). Root mean square errors (RMSE) and mean absolute percentage errors (MAPE) were used to select the configuration that best predicted the test data.

Our empirical results show that the three different statistical techniques considered achieved similar results for each site and all the forecast horizons. Seasonal ARIMA models were used because there was a clear indication that the wind speeds data for Jozini and Memel are seasonal, with daily and annual regular cycles respectively. For the Memel site, a more accurate model was obtained through the use of regression with ARMA errors, where the Fourier term corresponding to annual seasonality was used as a regressor. Overall, the persistence forecast was the least accurate model to predict wind speeds at 60m height for all sites. LSTM configurations and regression using Fourier terms with

ARMA errors achieved similar results with a slight improvement for the latter. Neural networks achieved comparable results with traditional time series models, thus, suggesting that the behaviour of wind speeds for each site is not overly complicated and simple forecasting techniques can be used for modelling.

The predicted values obtained using the most accurate model, and the actual values for each site were plotted. The results showed that regression with Fourier terms and ARMA errors method could accurately predict the oscillations of the wind speed series with high accuracy, and it predicted most of the sudden peaks in the series.

The analysis reported in this work provides much insight into wind speed forecasting for researchers who might apply statistical forecasting techniques on wind data in the future.

# Opsomming

Daar is ‘n groot uitdaging om maniere te vind om energiesekuriteit te bevorder en kweekhuisgasse, wat veroorsaak word deur die verbruik van nie-hernubare energiebronne, te verminder. Die vrylating van kweekhuisgasse veroorsaak aardverwarming en word nie as skoon beskou nie. In vergelyking met die konvensionele bronne van energie, soos fossielbronne, word hernubare energie as meer aantreklik beskou aangesien dit as skoon geidentifiseer is en omdat dit oor ‘n geslote koolstofdioksied siklus beskik. Dus word die $CO_2$ wat deur hierdie proses gegenereer word deur plante geabsorbeer vir voedselproduksie.

‘n Belangrike ontwikkeling in die elektrisiteitsindustrie die afgelope paar jare is die vinnige groei van windenergie. Die energie wat uit wind gegenereer word is afhanklik van meteorologiese toestande soos windspoed en windrigting. Hierdie meteorologiese toestande, veral windspoed, is stogasties in natuur en die vooruitskatting van waarnemings word dus as belangrik beskou in kragopwekking.

Daar is verskeie studies in the literatuur wat gebruik maak van statistiese tegnekie om windsnelhede te voorspel. In hierdie tesis word korttermyn vooruitskatting van uurlikse windspoedmetings by ‘n 60m naaf hoogte vir twee liggings, Jozini en Memel in Suid-Afrika, oor ‘n 1 tot 24 uur vooruitskattingshorison oorweeg. Die korttermyn windspoed by ‘n ligging was voorspel deur gebruik te maak van statistiese vooruitskattingstegnieke soos tradisionele tydreeks modelle (ARMA, ARIMA, seisoenale ARIMA en regressie met Fourier terme en ARMA foute), veelvoudige vlak “perceptron” (MLP) neurale netwerke en lang- korttermyn geheue (LSTM) herhalende neurale netwerke. Hierdie voorspellings is relevant vir beplanningsdoeleindes om te verseker dat die nodige basislading op die elektrisiteitsnetwerk verseker word. Alle vooruitskattings tegnieke was gebruik om windsnelhede te voorspel vir elke kombinasie van vooruitskattingshorison en ligging. Verskeie LSTM en MLP samestellings was geskep deur verskillende hoeveelhede verskuilde vlakke, hoeveelhede verskuilde nodusse, leerkoerse en aktiveringsfunksies te oorweeg. Die vooruitskattingskwaliteit van elke samestelling was vergelyk met die volhardingsvooruitskatting (maatstaf model). Die wortel gemiddelde kwadraat fout (RMSE) en die gemiddelde absolute persentasie fout (MAPE) was gebruik om die samestellings te kies wat die toetsdata die beste voorspel.

Ons empiriese resultate toon aan dat die drie verskillende statistiese tegnieke wat oorweeg was soortgelyke resultate lewer vir elke kombinasie van ligging en vooruitskattingshorison. Seisonale ARIMA modelle was gebruik omdat daar ‘n duidelike aanduiding is dat die windsnelhede vir Jozini en Memel seisonaal is, met ‘n daaglikse en jaarlikse siklus onderskeidelik. 'n Meer akkurate model

was verkry vir die Memel ligging deur 'n regressie met ARMA foute model te pas waar die Fourier term met 'n jaarlikse periode ooreenstem. In die algemeen was die volhardingsvooruitskatting die minste akkuraat om windsnelhede by by ‘n 60m hoogte te voorspel. LSTM samestellings en regressie met Fourier term en ARMA foute het soortgelyke resultate gelewer, met die laasgenoemde wat ‘n effense verbetering getoon het. Neurale netwerke en tradisionele tydreeksmodelle het soortgelyke resultate gelewer wat aandui dat die gedrag van windsnelhede nie oormatig kompleks is nie en gemodelleer kan word deur eenvoudige vooruiskattingstegnieke.

Vir die toetsdata was die voorspelde waarnemings van die mees akkurate model saam met die werklike waarnemings gestip. Die resultate toon aan dat regressie met Fourier terme en ARMA foute die ossillasies in die windsnelhede akkuraat kan voorspel. Hierdie model was ook in staat om meeste van die skielike pieke in die tydreekse te herken.

Die analise wat in hierdie werkstuk uitgevoer is bied insig in windspoedvooruitskatting vir navorsers wat statistiese tegniek op winddata in die toekoms wil toepas.

# Acknowledgements

I want to thank the following people for their contribution toward the success of my thesis. Without them, this thesis will not have been possible.

- God, for giving me strength and wisdom to carry out this study of analysing wind speed data.
- My supervisor, Dr. F Kamper, for his valuable guidance and inputs throughout this thesis. Thank you for your patience and dedication to the Department of Statistics and Actuarial Science at Stellenbosch University.
- Thirdly, I give thanks to my entire family and friends for sacrifices and support throughout my postgraduate studies for three years. Greatest gratitude to my two grandmothers, Pokati Zakari and Theopolina Nantanga, for all the prayers, financial support, and love. To Rebekka Enkali, Namene Timoteus and Clement Owusu Prempeh, thank you for the counselling and advice, I highly appreciate it.
- All the lecturers in the Department of Statistics and Actuarial Science for your course presentations. The statistical techniques you have taught me have helped me. More thanks to Professor SJ Steel, who supervised my honours project. His supervision was enlightening. I have learnt a lot from him.
- Finally, to Mr. Lee Robinson at INUKA fragrances, for his understanding and helping hand to stand in for me when I had a meeting with my supervisor.

# Table of Contents

# List of Figures

# List of Tables

# List of Acronyms

| | |
|---|---|
| **ACF** | Autocorrelation Function |
| **AIC** | Akaike Information Criterion |
| **ANN** | Artificial Neural Network |
| **AR** | Autoregressive |
| **ARIMA** | Autoregressive Integrated Moving Average |
| **ARMA** | Autoregressive Moving Average |
| **BIC** | Bayesian Information Criterion |
| **CSIR** | Council for Scientific and Industrial Research |
| **DoE** | Department of Energy |
| **DTU** | Technical University of Denmark |
| **IRP** | Integrated Resource Plan |
| **KAMM** | Karlsruhe Atmospheric Mesoscale Model |
| **LSTM** | Long-Short Term Memory |
| **MA** | Moving Average |
| **MAPE** | Mean Absolute Percentage Errors |
| **MLP** | Multilayer Perceptron |
| **NCAR** | National Centers for Atmospheric Prediction |
| **NCEP** | National Center for Atmospheric Research |
| **NDP** | National Development Plan |
| **PACF** | Partial Autocorrelation Function |
| **RMSE** | Root Mean Squares Errors |
| **RNN** | Recurrent Neural Network |
| **SARIMA** | Seasonal Autoregressive Integrated Moving Average |
| **SARMA** | Seasonal Autoregressive Moving Average |
| **SVR** | Support Vector Regression |
| **UNDP** | United Nations Development Programme |
| **WASA** | Wind Atlas of South Africa |
| **WRF** | Weather Research and Forecasting |

# CHAPTER 1

# INTRODUCTION

## 1.1 Research Overview

There is a significant challenge on how to enhance energy security and decrease greenhouse gas emissions associated with energy consumption from non-renewable resources. The releases of greenhouse gases cause many environmental issues such as global warming (Lawan *et al.*, 2014). Another problem is that even though non-renewable energy sources are accessible in most parts of the world, these sources will become more costly because of limited resources. Korkmaz *et al.* (2018) mentioned that due to the challenges with non-renewable resources like coal, renewable energy became more attractive for electricity production as it is considered a clean and more efficient source of energy that will sustain and maintain the environment.

Wind power is the fastest-growing type of renewable energy and has been in use for several years. The amount of power that can be extracted from the wind depends on its speed. Figure 1.1 shows the wind turbine power curve, viz. how the power output from a wind turbine in a wind farm varies with wind speed. The higher the wind speed, the more power can be harnessed to generate electricity on a large scale until a rated wind speed, and then after that, it levels off at the rated power. When the wind speed exceeds the cut-out wind speed, the wind turbine shuts down and stops generating any power for safety reasons.

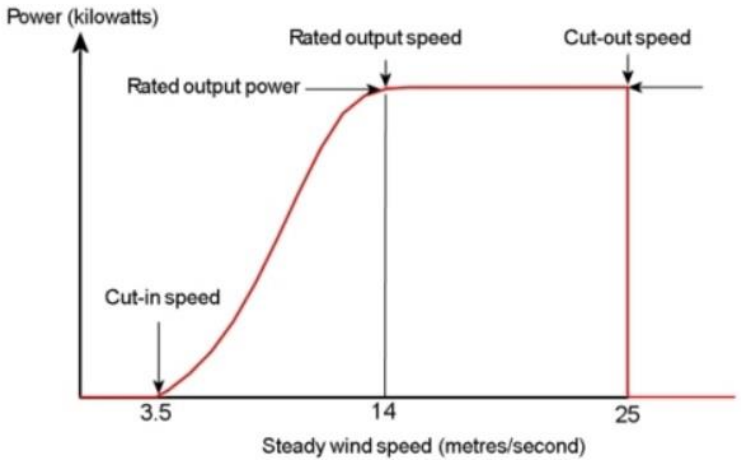



**Fig. 1.1: Typical wind turbine power curve (WindPower Program, 2019).**

Wind speeds are stochastic in nature, and forecasts of future values are considered important in power generation. In this thesis, we focus on the short-term forecasting of wind speeds for two wind sites situated in the large region of the Northern Cape and Eastern Cape in South Africa.

The meteorological data for both sites under study was obtained through WASA project and is available on their website (WASA, 2019). According to Mortensen *et al*. (2014), the project started in 2009 as an initiative of the South African Government (Department of Energy), United Nations Development Programme (UNDP), and other partners. The project aims to generate mesoscale wind information for areas with the most potential for wind power generation, and it currently consists of three stages, with stage 1 and 2 completed.

The two sites considered in this study for wind speed forecasting were identified in phase 2; Jozini located at KwaZulu-Natal and Memel at Free State. The weather information contains wind speed, wind direction, air temperature, barometric pressure, and relative humidity, all measured over 10-minute intervals. A full description of the sites and meteorological data will be discussed at a later stage.

## 1.2 Research Motivation

In South Africa presently, there is a critical shortage of power generation and reserves, leading to long load shedding periods to preserve the stability of the national grid whenever unforeseen capacity generation losses occur (Joubert, 2017). Power plants using coal or nuclear energy take a longer time to generate power compared to renewable energy sources such as wind. The process of generating power using wind is more efficient. According to Freitas *et al*. (2018), governments and utility companies around the world face pressure to find a solution to these challenges. Renewable energy, such as wind energy was proposed for generating power. This renewable energy has grown significantly owing to its benefits for large-scale power generation.

Wind power is nearly emission-free; turbines do not emit standard pollutants and use very little water which makes it cost-effective. Even though wind power is cost-effective, Korkmaz *et al*. (2018) argued that wind power is considered an unreliable source of energy that is not capable of meeting the future needs of the national electricity grid alone. The reason is because wind power generation depends on weather conditions like wind speed, which have irregular patterns. Due to this issue, forecasting wind speed with high precision has been an important obstacle to improve the quality of wind power, as the unstable behaviour of wind speed makes it hard to predict.

Consistent, accurate forecasts of wind speed are essential in order to generate more reliable wind power, to promote wind systems regulation and enhance industry operational efficiency through more reliable decision-making. Also, Zucatelli *et al.* (2019) mentioned that operators cannot control wind power, in the same manner as other resources, because it is a non-disposable source of energy and predicting future values is crucial. According to Soman *et al.* (2010), the relationship between power output from the turbine in a farm and wind speed is cubic (nonlinear), and any forecast error from wind speed forecast will effectively result in a large cubic error in wind power. Fig 1.2 shows the nonlinear relationship between wind speed and generated wind power of a wind turbine.

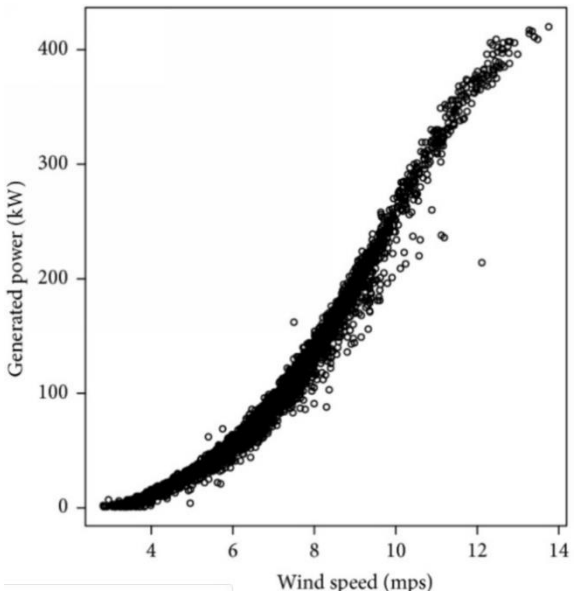



**Fig. 1.2: Nonlinear relationship between wind speed and wind power** (Zolna *et al.*, 2015)**.**

Since the innovation of generating power from wind, wind speed forecasting has been an essential and challenging study. Over the past couple of years, different wind speed forecasting techniques have been recorded in literature. Botha and Van Der Walt (2018) have proven that forecasting techniques to be considered depends on the nature of the required prediction and that different models are appropriate for different kinds of circumstances. Some forecasting techniques are accurate for short-term and others for long-term forecasting.

Short-term forecasting will be considered in this thesis to predict the wind speed for two sites over a 1 to 24-hour time horizon using statistical forecasting techniques. It is essential to predict the potential wind power at a site to ensure that the necessary base load on the electricity grid is secured at all times and to know when to drive wind turbines.

## 1.3 Research Objectives

Wind power generation requires wind speed knowledge for a particular site/farm, and it is essential to predict the future value of wind speed using present and past wind speed data. The focus of this thesis is to study the performance of three statistical techniques: Recurrent neural networks (LSTM), feed-forward neural networks (Multilayer perceptron) and time series models (Autoregressive integrated moving average models (ARIMA), Seasonal ARIMA models, ARMA models, seasonal ARMA models and regression using Fourier terms and ARMA errors), for short-term forecasting of wind speed for two sites in South Africa using observational wind data collected over a period of 3 years (beginning of 2016 to the end of 2018). The forecasts obtained from these models will be compared to the persistence forecasts which act as a baseline model.

The study will be performed as experimental work to guide wind speed forecasting researchers who might apply the above approaches for wind speed predictions at different sites for different forecast horizons.

The objectives of the study are as follows:

- To forecast 1 to 24 hours ahead wind speeds at a 60 m hub height for each site using observational wind data obtained through the Wind Atlas of South Africa (WASA) project by applying the persistence model, ARIMA and SARIMA models, regression using Fourier terms with ARMA errors, multilayer perceptron (MLP) neural networks and LSTM recurrent neural networks (RNNs).
- To compare the predictive capability of the techniques using different forecasting horizons. The reason for this is to investigate if the performance of the wind speed forecasts obtained using each technique depends on the forecast horizon.
- To compare the results obtained by each technique with the persistence forecasts at the different sites in order to determine an accurate model for reliable wind speed prediction for each site.
- To identify the strengths and weaknesses of each approach applied at each site.
- Lastly, to give directions for further research on wind speed forecasting at different locations.

## 1.4 Research Methodology

To achieve the objectives of the study, the following steps will be followed:

- Conduct a literature review on wind speed forecasting techniques used in the past and do a review on renewable energy in South Africa.
- Obtain the observational atmospheric data for all sites under consideration from the Wind Atlas for South Africa (WASA) website.
- Calculate the periodogram of each time series site to identify the periodicity. The periodic components are needed for SARIMA models.
- Use persistence models separately for all sites by forecasting 1-24 hours ahead.
- Use ARIMA models, SARIMA models, ARMA models, and SARMA models and select the most accurate models separately for all sites by forecasting 1-24 hours ahead.
- Use regression using Fourier terms with ARMA errors and select the most accurate models separately for all sites by forecasting 1-24 hours ahead.
- Use RNNs and select the most accurate models separately for all sites by forecasting 1-24 hours ahead.
- Use MLPs and select the most accurate models separately for all sites by forecasting 1-24 hours ahead.
- Make a comparison between the performance of the different models and determine the best model for all the sites separately.

The forecasting techniques used in this study were chosen based on past literature that indicated the ability of these techniques to predict wind speed with better results compared to other forecasting techniques. To select the best forecasting technique for each site, different configuration models were created for each forecasting technique and compared based on forecasting errors.

## 1.5 Structure of the thesis

The remaining chapters of this thesis will be organised into appropriate subdivisions and are as follows:

Chapter 2: A brief overview of the development and generation of renewable energy in South Africa, as well as Wind Atlas of South Africa (WASA) project, is provided. Next, the literature on wind speed forecasting techniques is reviewed.

Chapter 3: The research methodologies are discussed. The architecture of wind speed forecasting techniques employed in this study and functionalities of each technique are explained.

Chapter 4: Involves the description of the data used in the study, the selection of appropriate packages for the models and pre-processing of the data.

Chapter 5: Contains the results of an empirical analysis of wind speed data implemented in R.

Chapter 6: Conclusion, possible further studies, and recommendations are provided.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1 Overview

There has been an increased interest in wind speed or power forecasting research over the years and numerous reviews on wind speed forecasting techniques (De Freitas et al., 2018; Sharma and Singh, 2018). Wind speed forecasts are considered important for the renewable energy sector as power output from turbines depends on the wind speed, which changes over time. A summary of renewable energy in South Africa, the WASA project and the relevant literature to wind forecast using different forecasting techniques are discussed in this chapter.

## 2.2 Renewable Energy in South Africa

Renewable electricity generation capacity has considerably risen, but this capacity is not sufficient to substitute fossil-fuel energy. Better renewable energy technologies must be developed to replace fossil fuels fully.

Renewable energy is often called clean energy as it is generated from natural resources or methods that are often refilled. Wright and Grab (2017) highlighted that solar and wind power generation are the fastest growing forms of renewable energy, with wind power overgrowing at rates more than 15% per annum. In recent years, wind power generations have been integrated into the national electricity grid of South Africa and serve as an alternative source to cater for the limited reserve of fossil fuels.

South Africa is a developing country with limited ability to satisfy energy demand due to population and industrial growth. An alternative is needed to substitute the traditional manner of producing electricity, and to integrate renewable energy such as wind energy, in electrical systems for power generation. The department of energy suggested building new power plants through the 2011 Integrated Resource Plan (IRP) for the period 2010 to 2030, and it was proposed that the plan should be revised by the department frequently. According to Department of Energy (2011), the objective of IRP is to determine how to meet future demand in terms of capacity generation, type, timing, and to minimise the cost of electricity.

In 2011, the South African government suggested the purchase of wind power from private electricity producers to diversify and expand the national energy generation in the country. Also, this can provide clean energy with cheaper tariffs. At the end of 2013, South Africa reported its first electricity generated by the wind from private power producers integrated into the national grid. By the end of 2015, this generation had achieved 2012 MW, which is equivalent to 2.66% of the domestic energy capacity installed. The predicted outlook would see wind generation installed up to 4360 MW (about 5% of domestic energy capacity) by 2030 (Wright and Grab, 2017). Wind as an energy source is only practical in regions with powerful and constant winds. South Africa has good wind potential, particularly along the Western Cape, Northern Cape and Eastern Cape coastal regions.

## 2.3 Wind Atlas for South Africa (WASA) Project

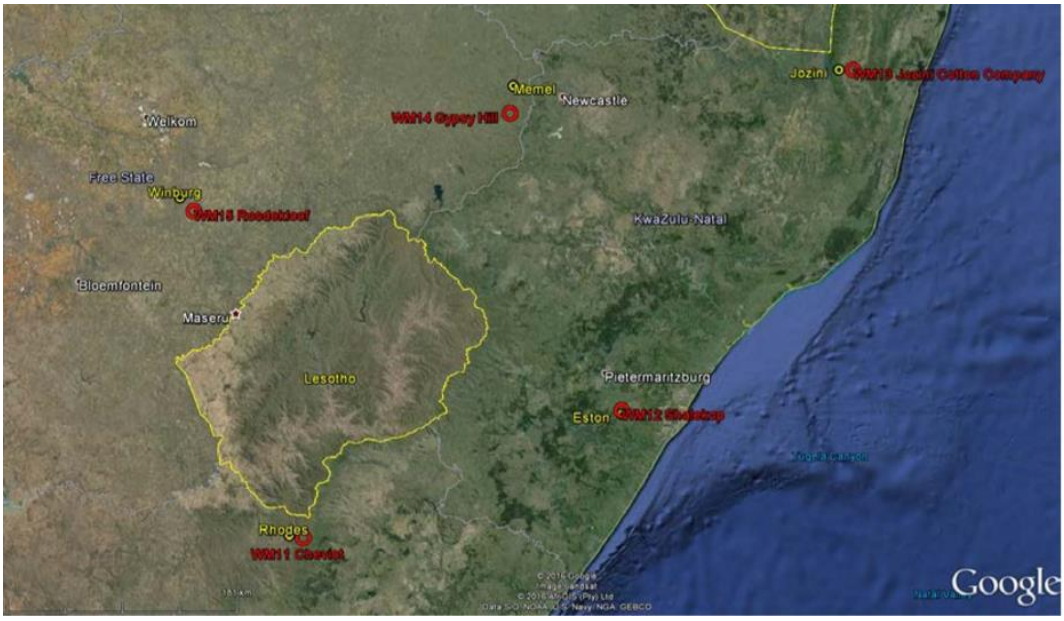
The department of energy started the WASA project in 2008 with the United Nations Development Programme (UNDP), the government of Denmark and other partners. The aim is to update the existing wind atlas with recent wind measurement technologies intended to avoid obstacles in the landscape; such as buildings and trees (Otto, 2015). The project was carried out in conjunction with the Renewable Energy Independent Power Producers Programme (REIPPP). The REIPPP was designed by the department of energy to respond to the call by the National Development Plan (NDP) and Integrated Resource Plan (IRP) 2010 for renewable energy sources.

Refer to Independent Power Producer Office (2014) for a detailed description of the following summary. The NDP aims to eliminate poverty by 2030 and for South Africa to invest in a strong network of economic infrastructure. The NDP plan also includes an additional 10 000 MW electricity capacity by 2019 compared to the 44 000 KW baseline for 2010. The Integrated Resource Plan (IRP) was created for South Africa to meet the electricity demand by 2030 (over 20-year horizon) and 17 800 MW of electricity will be from renewable energy sources; 5 000 MW by 2019 and an additional 2 000 MW operational by 2020.

The WASA project covers a large region of the Northern Cape, Eastern Cape and Western Cape in South Africa and consists of three phases with phase 1 and 2 completed. According to Prinsloo *et al.*( 2014), in Phase 1, 10 sites with 60-m mast instrumentation were inspected by the Council for Scientific and Industrial Research (CSIR) and DTU Wind Energy in 2011. The ten sites are: Alexander Bay, Calvinia, Sutherland and Noupoort in the Northern Cape; Vredendal, Vredenburg, Napier and Beaufort West in the Western Cape as well as Humansdorp and Butterworth in the Eastern Cape.

In Phase 2, site inspections were carried out by the CSIR and DTU Wind Energy in 2016 and 5 sites were recorded. The five sites are: Rhodes located in the Eastern Cape; two sites in KwaZulu-Natal (Eston and Jozini) and two sites in the Free State (Memel and Winburg) (Prinsloo *et al.*, 2017). All sites inspected in both phases are shown in the maps below.


(a)




(b)

**Fig. 2.1:** (a) Overview map of the southernmost part of South Africa, showing the location of the ten meteorological masts identified in phase 1, (b) five mast locations in the north-eastern part of South Africa recorded in Phase 2 (Image reference: Prinsloo *et al.*, 2017).

Information in the following 2 paragraphs was taken from a paper written by (Lennard *et al.*, 2015). The WASA project produced both observational and numerical wind atlases for South Africa. Observational wind atlases are produced using microscale modelling techniques by integrating measured time series wind speed and direction, as well as features of the local topography. In the absence of high-quality measured information, numerical wind atlas methods are implemented using macroscale global weather datasets such as the NCEP/NCAR worldwide reanalysis data set with local typographic features to generate mesoscale wind information covering a wide region. In layman's terms, an observational atlas involves observed wind speeds while a numerical atlas involves simulated wind speeds.

The WASA numerical wind atlas contains generalised wind data sets for tens of thousands of model grid points covering the whole of South Africa. Two numerical wind atlases were produced in the WASA project using two different methods, the Karlsruhe Atmospheric Mesoscale Model (KAMM) in 2012 and the Weather Research and Forecasting (WRF) model in 2014. The KAMM method produced 15 000 data points on a 5 km x 5 km spatial resolution and the WRF method resulted in 40 000 data points on a 3 km x 3 km spatial resolution. Also, the project created time series data using the WRF model with a spatial resolution of 27 km x 31 km covering the whole project area.

The time series data consists of hourly wind speed and wind direction for Western Cape and parts of Northern and Eastern Cape for the period 01-09-1990 to 31-12-2012.

All meteorological data are available on the WASA website for download (WASA, 2019).

## 2.4 Overview of Wind Speed/Power Forecasting and Time Horizon

Variable renewable energy generation, especially from wind energy sources, introduces uncertainties in the operation of power systems. Research undertaken by Sørensen *et al.* (2018) quantify how wind power development can influence the use of short-term automatic reserves in the South African power system. They found that 5% of wind power penetration will boost the use of short-term automatic reserves by around 2% in 2025 (Ejnar *et al.*, 2017). Accurate wind forecasting is regarded as an effective instrument to reduce many issues; such as uncertainty of variable generation, competitive market designs, real-time grid activities, quality of power, stability and reliability of energy system (Soman *et al.*, 2010). Correct forecasting can assist in developing a well-functioning hour ahead or a day ahead markets in a competitive electricity market.

Olaofe (2013) argues that there are two ways of predicting a future event. It can be done either by developing a forecast model or by an inferred study of historical measurement of wind site/farm measurements over a period of time. Wind power output from a turbine depends on the wind speed around that farm, and accurate wind speed forecasting is as good as accurate wind power forecasting. Errors in wind speed forecasting would lead to extensively propagated errors in the expected power output. Several wind speed models/techniques have been developed for various types of situations, as some forecasting techniques are better for short-term forecasting while others are better for long-term forecasting.

According to (Soman *et al.*, 2010), there are four forecast time horizons, namely, very short-term, short-term, medium-term and long-term forecasting. Very-short-term forecasting involves predicting values from few seconds to half an hour (few minutes), short-term forecasting is from half an hour to 6 hours ahead, medium-term forecasting is from 6 hours to a day ahead, and long-term forecasting is from a day ahead to a week or more. Each forecasting time horizon has its purpose and importance in power generation. Long-term forecasting supports decisions made in the electricity market and also to minimise cost in the planning of maintenance. Medium-term forecasting is used in making decisions about the shutdown of wind turbines (Soman *et al.*, 2010). Short-term forecasting helps with load dispatching decisions, whether to decrease or increase power to meet customers' needs within a short time. Very short-term forecasting is perfect for wind turbine configuration and clarification of additional market data (Lawan *et al.*, 2014).

The accuracy of various forecasting model differs with the quality of available meteorological data and the intended forecast time horizons. Due to variations in the weather pattern across a geographical location, there has been no standardised forecast model that can be used at any specific wind farm for wind speed and power forecasting. However, several forecasting techniques were reported in the literature over the past few years and are grouped as follows: Persistence techniques; Physical systems techniques; Statistical forecasting techniques and Hybrid techniques (Botha and Van Der Walt, 2018).

### 2.4.1 Persistence Techniques

The persistence technique is the simplest model used for forecasting, and it is often referred to as a naïve predictor. It is more accurate than other techniques/approaches in very-short term forecasting (Wu and Hong, 2007). This model is not only the simplest way to forecast but also the most economical even though the performance accuracy of the model drops as the forecasting time horizon increases. It is useful as a baseline model for comparing more advanced methods. This model is used

in this study to compare with SARIMA models and neural networks and is discussed in more detail in Chapter 3.

### 2.4.2 Physical Systems Technique

These models are built on detailed physical descriptions of the geographical locations of interest. They consist of certain physical equations to transform a large amount of meteorological information from a particular time to the predicted wind speed at a site (Azad *et al*., 2014). An example is the Numerical Weather Prediction (NWP) model. The NWP model works by solving complicated mathematical equations using present atmospheric information/data, such as temperature and pressure, to predict the future state of atmospheric conditions. The NWP model generally offers wind speed predictions with spatial resolution for a grid of neighbouring points around the wind generators.

The physical model uses a meso or micro-scale model that interpolates these wind speed predictions to the level of the wind generator (Giebel and Al., 2011). These models are preferred when wind speed prediction is needed at the start of a new wind farm operation.

Physical models, however, are computer-intensive because they are complex and require supercomputers to operate successfully. Therefore, they are recommended for long-term prediction (Shiyan et al., 2009). According to Wu and Hong (2007), physical models are unsuitable for short-term prediction because of difficulties in acquiring data for the model and the complicated computation.

### 2.4.3 Statistical Forecasting Techniques

This technique is the most commonly used predictive model for the prediction of future events based on historical events or measurements. Physical systems use complex mathematical equations for forecasts, but statistical methods describe the connection between input and output data for pattern detection in the data by analysing a huge training dataset (Freitas *et al.*, 2018).

Statistical techniques can be used to address issues in engineering, economics, finance and natural sciences that have a huge amount of data where observations are interdependent (Chang, 2014). Statistical techniques are effective for very short-term and short-term forecasts but cannot be used alone for long-term predictions. Other methods, such as numerical weather forecasts should be taken into account for long-term forecasting (Azad *et al*., 2014).

Statistical techniques are grouped into two categories, viz., time series models and artificial intelligence models. In this thesis, statistical forecasting techniques are used to forecast wind speed data and will be discussed in Chapter 3.

### 2.4.4 Hybrid Techniques

The hybrid technique is a method of combining different techniques such as; physical techniques with statistical forecasting techniques, time series models with artificial neural networks or short-term with long-term models. Usually, a hybrid forecast model is used to enhance forecasting performance, mainly when a single forecast model is poorly performing (Chang, 2014). The hybrid structure of the NWP model and ANNs are more accurate for medium and long-term forecasting while ANNs combined with Fuzzy logic models are recommended for very short-term forecasting (Soman *et al.*, 2010).

Figure 2 shows a summary of wind speed forecasting techniques used in the past and the focus of this thesis.

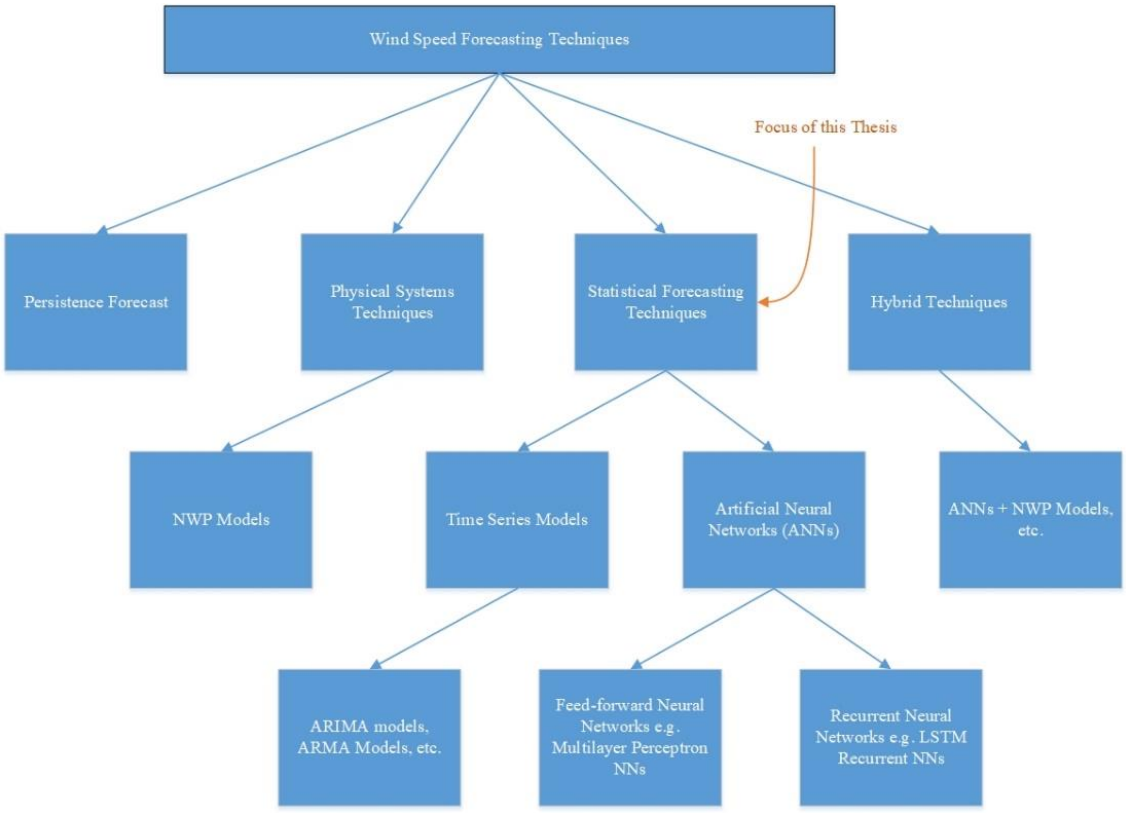



**Fig. 2.2: Summary of wind speed forecasting techniques**

## 2.5 Review of Statistical Forecasting for Wind speeds

A review of recent studies on wind speed and wind power forecasting is introduced in the following sections.

### 2.5.1 Review of Studies in South Africa

Most recent studies on short-term forecasting have revealed that conventional statistical methods are effective for short term prediction. Ayodele *et al*. (2019) compared persistence, second-order Markov chain (SOMC), autoregressive moving average (ARMA) and Weibull models using 10-minute average wind speed data collated at Alexander Bay. They tested the models at different time horizons (very-short term, short-term, medium-term and long-term horizons) to identify the most accurate wind speed prediction model for each of the time horizons. Three statistical measures were employed to compare the models; the mean value, the root mean square error (RMSE) and the mean absolute percentage error (MAPE). A time interval of 30 minutes was used for very short-term forecasting, 6 hours for short-term, 1 day for medium-term and 7 days for long term forecasting. ARMA models were reported to predict wind speed at very short-term and long-term forecasting with higher accuracy, giving the lowest error values. SOMC was accurate for short-term and medium-term forecasting, and persistence results appear to be the least accurate for all the time horizons. An explanation for this lies in the fact that persistence model is based on the assumption that future values of wind speeds will be the same as the present values and with highly variable time series, the model performs poorly. However, the performance accuracy of the model is expected to improve if a time horizon of less than 10-minutes is considered.

Machine learning models are built through algorithms which finds patterns in the training data. These models have been successfully used in the past for wind speed forecasting and in most cases, outperform traditional statistical methods due to the nature of their flexibility. Support Vector Regression (SVR) and feature selection were used in a study by Botha and Van Der Walt (2018) to forecast wind speed using meteorological data collected at Alexander Bay in South Africa over a three-year period. Training of the SVR algorithm was done using past wind speeds. Additional features such as wind speeds at the same location but at five different hub heights, wind direction, air pressure, temperature and other weather data were used to improve the forecasting accuracy of the model. The main objective of the paper was to explore the influence of feature selection on the short-term forecasting accuracy of a 1 to 24 hour ahead wind speed prediction at a 60m hub height using support vector regression. The authors created several SVR models with a combination of variables,

and these models were compared to the base case model (SVR model with wind speed at 60m). The root mean square error (RMSE) was calculated for each model and the results show that by systematically choosing and combining appropriate input features, the relative forecast performance of a short-term SVR model can be improved by up to 11.12%. The most accurate model was reported to be SVR model with wind speed at 60m (base case) and additional features such as air temperature and barometric pressure.

Another paper making use of machine learning was written by Van Der Walt and Botha (2017), where they used nonlinear support vector regression (SVR) with a radial basis function kernel and Bayesian ridge regression (BRR) to forecast wind speed 1 to 24 hours ahead using wind speed time series at Alexander Bay in South Africa. The data includes the wind speeds at 60m hub height collected over a period from 1 January 2011 to 31 December 2013 with 10-minute resolution. The authors compared the SVR and BRR methods with the ordinary least squares (OLS) model (traditional statistical method). Before computing the forecasts using the algorithms, they generated input features for the algorithms using a 24-hour moving time window approach. The persistence model was used as a benchmark model, and they calculated 1 to 24 ahead forecasts using this naive model. This study illustrated that as the forecast window increases, the error rate of the persistence forecast increases until a maximum error rate is reached, and after that, it reduces. They later extended the window of the forecast from 24 hours ahead to 72 hours and the experimental results indicated a cyclic nature of 24 hours, which showed a strong relationship between wind speed and time of day. Forecasts from the three algorithms were compared to the persistence forecasts using the root mean square error (RMSE). They used a grid search method to optimise the SVR hyperparameters by making use of a validation set. The SVR (RBF) model was found to be more accurate (with low RMSE values for all prediction time horizons) than OLS and BRR. The experimental results showed that as the time horizon increases, the performance of all models decreases. Also, the study examined adding wind speed information at 40m and 62m hub heights to the input features (wind speed at 60m) to see if there will be any improvement on the chosen model.

The conclusion made was the same as Botha and Van Der Walt (2018). There was a consistent improvement in the model performance after adding the variables. It is interesting to notice that when appropriate variables are added to the model, there is an improvement in the forecast model.

### 2.5.2 Review of Other Studies

Conventional ARMA models and Fourier based-ARMA models were compared in a study to predict daily wind speed data of three locations; Senai, Bayan Lepas and Subang at Peninsular Malaysia (Jamaludin *et al.*, 2016). The three locations were chosen from 10 different wind speed stations all over Peninsular Malaysia using the ordinary Kriging spatial interpolation technique. The data was obtained from the Malaysian meteorological department, which contains daily wind speed data for a period from 1st January 1985 to 31st December 2001. A Fourier transformation was applied to the time series to decompose it into a sum sinusoidal functions to express it as a new sequence in the frequency domain. The Akaike Information Criterion (AIC) was used for ARMA model selection, and Ljung Box test was used to test for model adequacy at 0.05 significant level. The daily wind speed was predicted for 355 days ahead and the root mean square error (RMSE) together with mean absolute percentage error (MAPE) were used to compare the models at each station. Based on the results obtained, it was found that Fourier based-ARMA model was the best model for daily wind speed forecasting at all stations compared to the conventional ARMA model. The results highlighted the higher performance accuracy of predicting using frequency-domain data over time-domain data.

Artificial Intelligence techniques are considered to be accurate for forecasting compared to traditional time series models because of their ability to handle noisy data. Artificial neural networks (ANN) was employed by Zucatelli *et al.* (2019) in Uruguay to forecast wind speed 1, 3, 6, 9 and 12 hours ahead (short-term forecasting) at four different anemometric heights of 101.8, 81.8, 25.7 and 10.0m during a period of one year. Their study aimed to identify the most efficient ANN configuration from different proposed ANN configurations using MLPs with the Levenberg-Marquardt backpropagation training algorithm. First, they used the proposed ANN models to do a one-step-ahead forecast using meteorological data collated at each height and compare the results using RMSE, MAE, MSE and MAPE as performance measure metrics. The authors reported that the lowest MAPE value was 15.840% at the height of 101.8 m, the highest height considered. The authors used the best model obtained at this height to predict wind speed 3, 6, 9 and 12 hours ahead. Their results showed that 1 and 3 hours ahead wind speed forecasts were mostly accurate with the lowest RMSE values. Also, the analysis indicated that as the forecast time horizon increases, the accuracy of the model decreases. They concluded that ANNs were adequate for wind speed forecasting at different heights and that ANN models can produce accurate short-term forecasts with low computational cost and this will help governments to manage the national energy supply (Zucatelli *et al.*, 2019).

A different approach was proposed by López *et al.* (2018), a hybrid model with Long Short-Term Memory (LSTM) plus an Echo State Network (ESN) to forecast wind power 1 to 48 hours ahead using historical wind power data and meteorological data provided by a Numerical Weather Prediction (NWP) system developed at the Technical University of Denmark (DTU). This model uses LSTM memory blocks as units in the hidden layer with an architecture of ESN type to impose some restrictions on the layers. The proposed model was trained using the two-stage process, firstly the hidden layer was trained using a gradient descent approach, and lastly, the output layer weights are adapted by quantile regression. They compared the proposed model against a persistence model, the forecasts generated by the Wind Power Prediction Tool (WPPT) and some variants of the LSTM+ ESN framework. The MSE, MAE, MAPE and SDE were used as performance measures, and the proposed model attained better global performance in all the metrics.

In another paper by Lakshmi and Sujatha (2016), it was shown that feed-forward neural networks could be used as a practical tool to forecast wind speeds. They used monthly wind speed data collected at two stations located in Mersin and Silifke districts in Turkey for a period from January 1975 to December 2006. The authors used MATLAB to apply feed-forward neural networks with backpropagation to the data. The input variables used to design the network were monthly averaged relative humidity, monthly averaged wind speed, monthly averaged atmospheric pressure and monthly averaged atmospheric temperature. The authors also experimented with adding appropriate variables such as minimum, maximum and average values of temperature, humidity and air pressure to the model to achieve better performance. They used a neural network consisting of two-hidden layers with a logarithmic sigmoid function in each layer and an output layer with linear activation function to predict monthly average wind speed. The experimental results showed that ANN is an efficient tool for estimating wind speed values.

Cali and Sharma (2019) investigated the use of LSTM-RNN for short-term wind power forecasting, to forecast wind power 1 to 24 hours ahead. The study used historical generated wind power and Numerical Weather Prediction (NWP) data for a wind farm in Spain (Sotavento). The farm consists of 24 wind turbines and the data used was for 1 year (1$^{st}$ January 2016 to 31$^{st}$ December 2016) with an hourly resolution. They mentioned that NWP data consists of surface pressure, temperature, wind speed (at 10, 35, 100 and 170 M) and wind direction at 35M and 170 M. The authors make use of sensitivity analysis which is a variable selection technique to select input variables from the NWP data. They used normalised mean absolute error (nMAE), and normalised root mean square error (nRMSE) to evaluate the models. The authors believe that the future value of wind power depends on atmospheric variables but also the previous value of the wind power generated and that RNNs will be the best model to incorporate the short-term temporal dependency in wind power. To forecast 1 to

24 hours using the model, they first calculated the autocorrelation function of the data. This approach was used to find the number of lags required for the model, and it was found to be 24. The authors created 9 different forecasting models using different combinations of input variables and forecast one-day ahead using 9 months of data as training and 3 months of the data as testing. The 9$^{th}$ model contained all the variables from NWP data and based on the experimental results of forecasting 1 day-ahead; it performed poorly with the highest nRMSE value of 12.32% which means that more variables do not necessarily mean a more accurate model. The most accurate model was found to contain temperature, wind speed at all heights and wind direction at all heights. They used this model to forecast 1 to 24 hours, and the results showed that the model performed with high accuracy for 1 hour ahead forecast (with lowest nRMSE of 4.23% and 3.01% nMAE). The results also indicated that as the forecast horizon increases, nRMSE and nMAE values increased. We see that the performance accuracy of wind power forecasts depends on the forecast horizon.

### 2.5.3 Conclusion

Important points to note from all these studies is the application of various forecasting methods for wind speed/power forecasting by different researchers. All techniques used in these studies can accurately forecast wind speed or wind power, but some forecasting techniques outperform others in different situations. A considerable amount of research was conducted on machine learning and artificial-intelligence methods to predict wind speed, and they usually outperform conventional statistical forecasting models. Wind speed forecasting at different time horizons was successfully implemented, and the conclusion from most of these studies is common, that the performance of the model depends on the forecasting horizon. This thesis will further contribute in this direction by making use of LSTMs, MLPs and traditional time series models (ARIMA models, SARIMA models, ARMA models, SARMA models and regression using Fourier terms with ARMA errors) to forecast at different forecasting horizons and find the most accurate model for wind speed forecasting at two sites.

# CHAPTER 3

# STATISTICAL FORECASTING TECHNIQUES FOR WIND SPEED FORECASTING

Statistical forecasting techniques have been widely used in the renewable sector because of its capacity to accurately forecast and provide timely predictions. Artificial neural networks (ANNs) and time series models are subclasses of statistical forecasting methods. In this thesis, artificial neural network methods, namely; MLPs and RNNs will be used together with time series models, namely; ARIMA models, seasonal ARIMA models and regression using Fourier terms with ARMA errors. Two wind speed time series (Jozini and Memel) are considered for forecasting. In this chapter, the key concepts and architectures involving the forecasting methods used are introduced to give a better understanding of the procedure applied in the empirical study reported in Chapter 5.

## 3.1 Forecasting

An observed time series can be defined as a sequence of random variables, $Y_1, Y_2, Y_3, \ldots\ldots\ Y_n,$ where the random variable $Y_1$ denotes the value of the series at the first time point, $Y_2$ denotes the value for the second time point, $Y_3$ denotes the value for the third time point, and so on until time $n$ , where $n$ is discrete (Shumway and Stoffer, 2012). It means that the time series consists of n sequential observations taken from an underlying stochastic process.

Predicting or forecasting a random variable in a time series means that we use past random variables to obtain information about the random variable we want to forecast. If we are at time $n$ , and want to predict $Y_{n+i}$ , we consider the observations $Y_1, Y_2, \ldots, Y_{n-1}$ for $i > 0$. This means we want to find a function $\hat{Y}_n(i)$ in terms of the past observations which gives us good information about $Y_{n+i}$.

Different techniques have been proposed to forecast future values as a function of past observations. These techniques are appropriate to use when the data is available and when the pattern in the data is expected to continue into the future. We make use of statistical forecasting techniques in this thesis and the following sections introduce the key concepts of these techniques.

## 3.2 Persistence Forecast

The persistence forecast is based on the assumption that at some forecast time, the wind speed will be the same as when the prediction was made. When the measured wind speed at time $n$ is $Y_n$ then the future wind speed at time $n + i$ is forecasted as:

$$\hat{Y}_n(i) = Y_n \tag{3.1}$$

For some $i > 0$. The persistence forecast is the simplest method used for forecasting and is often referred to as a naïve predictor. The method works well when the atmospheric patterns change very little, but is not the best model to use when the atmospheric conditions change considerably from day to day. However, it is more accurate than other techniques/approaches in very-short term forecasting (Wu and Hong, 2007). This method is not only the simplest way to forecast but also the most economical, even though the performance accuracy of the model drops as the forecasting horizon increases. It is useful as a baseline model for comparing more advanced methods.

## 3.3 Time Series Models

Time series models are applied based on training a model with historical measurement data of that specific location/site to predict future events. The purpose of time series techniques is to develop a forecast model that can adjust the prediction parameters to minimise the forecast error between the forecasted and actual values (Olaofe, 2013). Time series models are easy to train and less expensive to develop than other models because they do not require data other than historical data of the variable of interest (Wu *et al.*, 2007). However, the forecast error increases as the forecast time increases. Time series models are preferred for short-term forecasting.

Time series models used by researchers in forecasting time series data include moving average (MA); autoregressive (AR); autoregressive moving average (ARMA); autoregressive integrated moving average (ARIMA), seasonal autoregressive integrated moving average (SARIMA) models and seasonal autoregressive moving average (SARMA) models. These models are accurate for short-term wind speeds and power predictions of a wind turbine on a farm. The SARIMA models and SARMA models are effective when the time series to be analysed have seasonal patterns. These models will be utilised in this study and are discussed in the following section.

### 3.3.1 MA Models

MA models forecast the next value of the time series as a linear combination of the residual errors from the previous forecasts and a constant term. A typical representation of the MA model is:

$$Y_t = \mu + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

$$= \mu + \sum_{j=1}^{q} \theta_i \varepsilon_{t-j} + \varepsilon_t. \quad (3.2)$$

The model is called a moving average model of order $q$, where $\varepsilon_t, \varepsilon_{t-1}, \varepsilon_{t-2}, \varepsilon_{t-q}$ are white noise errors at time $t, t-1, t-2, \ldots, t-q$ ; $\theta_1, \theta_2, \ldots, \theta_q$ are parameters of the model to be estimated, and $\mu$ is the mean of the time series. The errors are called white noise if they are purely random with zero mean $(E(\varepsilon_t) = 0)$, constant variance $(V(\varepsilon_t) = \sigma^2)$ and are uncorrelated $(E(\varepsilon_t, \varepsilon_{t-k}) = 0)$ for $k \neq 0$.

The MA model is considered more accurate for short-term forecasting, and when there is no trend or seasonality in the data. The moving average model performs poorly when the time series is noisy and seasonal because the residual errors of the previous forecasts are propagated to the future values (predictions). When the time series is nonstationary, the MA model requires a smoothing approach to remove the trends and irregularities in the time series.

### 3.3.2 AR Models

AR models forecast the next value of the univariate time series as a linear combination of the past observations and a random error. The AR($p$) model can be expressed mathematically as:

$$(Y_t - \mu) = \sum_{i=1}^{p} \phi_i (Y_{t-i} - \mu) + \varepsilon_t, \quad (3.3)$$

where $\phi_i$ ( $i = 1,2, \ldots p$) are the autoregressive model parameters to be estimated, $\varepsilon_t$ is the white noise error at time $t$ and $\mu$ is the mean of the time series (Cryer and Chan, 2008). Using this model, we assume that for every $t$, $\varepsilon_t$ is independent of the past values ($Y_{t-1}, Y_{t-2}, \ldots Y_{t-p}$). This model is called an AR model of order $p$. The model uses $p$ past values of the time series as predictors. Adhikari *et al.*, (2013) mentioned that it is simpler to fit an AR model to the time series than to fit an MA model because white noise errors in the MA model are not easily predictable.

### 3.3.3 ARMA Models

The ARMA model is a combination of AR and MA models. The ARMA model is a stationary time series model developed by Box and Jenkins, and it was combined to form a general model of $p$-order autoregressive and $q$-order moving average processes. The ARMA$(p, q)$ model can be mathematically expressed as:

$$(Y_t - \mu) = \sum_{i=1}^{p} \phi_i (Y_{t-i} - \mu) + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t \ , \tag{3.4}$$

where $Y_t$ is the value of the variable observed at time $t$ , $\phi_i, i = 1,2, \dots, p$ are the autoregressive parameters, $\theta_j, j = 1,2, \dots, q$ are the moving average parameters, $\mu$ is the mean of the time series and $\varepsilon_t$ is white noise error at time $t$ . The model can be written in backshift notation as:

$$\phi(B)(Y_t - \mu) = \theta(B)\varepsilon_t \tag{3.5}$$

where $\varepsilon_t$ is a white noise process, $B$ is a backshift operator on time series (lag operator) defined as $BY_t = Y_{t-1}$, $\phi(B) = 1 - \phi_1 B^1 - \cdots - \phi_P B^p$ and $\theta(B) = 1 + \theta_1 B^1 + \cdots + \theta_q B^q$ are used to represent the autoregressive operator and moving average operator of the process in backshift, respectively. The ARMA model is suitable for short-term forecasting, but forecast accuracy performance decreases as the forecasting time horizon increases. This model performs well when the time series of interest is stationary and performs poorly for nonstationary time series. Milligan *et al.*, (2004) applied a class of ARMA models to predict 1-hour ahead wind speed and wind power output for wind farms in Minnesota and along the Washington-Oregon border. The results of the ARMA models were compared to persistence forecasts, and the authors found that the ARMA model can provide significant improvement in wind forecasts compared to persistence forecasts.

### 3.3.4 SARMA Models

Most socio-economic and environmental time series exhibit various types of periodic seasonal and cyclic impacts. For example, they are affected by annual seasonal variations or quasi-periodic business cycle, with regular periodicity (Chen and Perchonok, 2008). ARMA models perform poor on such series because they are local models. Box and Jenkins introduced SARMA models which is an extension of an ARMA model with a seasonal component to deal with seasonality. The model is termed as SARMA $(p, q)$ x $(P, Q)_s$ where $p$, $q$, $P$ and $Q$ denote the AR order, MA order, seasonal AR order and seasonal MA order respectively. The general SARMA $(p, q)$ x $(P, Q)_s$ model can be written as (Dwivedi *et al.*, 2017):

$$\Phi(B^s)\,\phi(B)(Y_t - \mu) = \Theta(B^s)\theta(B)\varepsilon_t \ , \tag{3.6}$$

where $s$ is the period, $\Phi(B^s) = 1 - \Phi_1 B^s - \cdots - \Phi_P B^{PS}$ and $\Theta(B^s) = 1 + \Theta_1 B^s + \cdots + \Theta_Q B^{QS}$.

### 3.3.5 ARIMA Models

One of the typical assumptions about a time series is that they are stationary, that is, they evolve randomly around a steady mean over time, reflecting some stable balance. In real-life problems, most time series have some form of nonstationary. A time series is stationary if the mean or variance between periods is constant, and the covariance value between two periods depends only on the distance (Junior *et al*., 2014).

If the mean differs with time or variance varies with time, a time series is non-stationary. In other words, the time series mean or variance are unstable over time. Stationary models such as ARMA models perform poor when used to model and forecast nonstationary time series data. In real-life problems, many time series, such as those related to natural science, show non-stationary behaviour and a general class of time series models developed by Box and Jenkins called ARIMA can be used to forecast nonstationary time series.

An ARIMA model is an extension of the simpler ARMA model including integration ($I$) to make the time series stationary when necessary. Time series data which contain seasonal patterns or trends are non-stationary. The trend in the non-stationary time series can be removed by differencing the time series a finite number of times until it becomes stationary. An ARIMA $(p, d, q)$ model arises when the differenced time series $Z_t = (1 - B)^d Y_t$ follows an ARMA$(p, q)$ process. An ARIMA $(p, d, q)$ can be expressed as:

$$\phi(B)(Z_t - \mu) = \theta(B)\varepsilon_t,$$

or:
$$\phi(B)((1 - B)^d Y_t - \mu) = \theta(B)\varepsilon_t \ , \tag{3.7}$$

where $\phi(B)$ is the autoregressive operator, $\theta(B)$ is a moving average operator and $\mu$ represent a drift term. The parameter $p$ in the notation is the number of autoregressive terms, $d$ is the degree of differencing, and $q$ is the number of moving average forecast error terms. If $d$ is zero in the notation, then the ARIMA model becomes an ARMA model.

### 3.3.6 SARIMA Models

Most time series are seasonal in nature and ARIMA models cannot model the seasonality in the time series. Box and Jenkins introduced a generalised model to deal with seasonal data. The model is referred to as seasonal ARIMA model. It is an extension of ARIMA with a seasonal component to deal with seasonality. The model is generally termed as SARIMA $(p,d,q)$ x $(P,D,Q)_s$ where $p,d,q$ are the same parameters as in the ARIMA model. The $P,Q$ are the same as in the case of the SARMA model, while $D$ is the number of seasonal differences required to make the process stationary. A seasonal difference is defined as a difference between a value and a value with lag that is a multiple of $s$. Suppose a monthly data have s=12, the seasonal differences are $(1-B^{12})Y_t = Y_t - Y_{t-12}$. If the differenced time series $Z_t = (1-B^d)(1-B^s)^D Y_t$ is assumed to follow a SARMA $(p,q)$ x $(P,Q)_s$ model, we can write the general SARIMA $(p,d,q)$ x $(P,D,Q)_s$ model as:

$$\Phi(B^s)\,\phi(B)(Z_t - \mu) = \Theta(B^s)\theta(B)\varepsilon_t$$

Or:
$$\Phi(B^s)\,\phi(B)((1-B)^d(1-B^s)^D Y_t - \mu) = \Theta(B^s)\theta(B)\varepsilon_t \tag{3.8}$$

where $\varepsilon_t$ is white noise errors at time $t$, $\mu$ is the mean of the differenced process $(1-B^d)(1-B^s)^D Y_t$ and $B^s$ is the backshift operator at seasonal lag (Siregar, 2018).

### 3.3.7 Estimation and Model Identification

In this section, we discuss the estimation and identification of an appropriate ARIMA $(p,d,q)$ model. The methods discussed here can be extended to other models; such as SARIMA models and regression with ARMA errors. We will work with Equation (3.7) under the assumption that the drift term is zero ( $\mu$= 0). Thus, our goal is to estimate the parameters in

$$\phi(B)(1-B)^d Y_t = \theta(B)\varepsilon_t\,, \tag{3.9}$$

from an observed time series $Y_1, Y_2, \ldots, Y_n$.

#### 3.3.7.1 Determining the Appropriate Number of Differences

Let $Z_t = (1-B)^d Y_t$, then the process defined by $\phi(B)Z_t = \theta(B)\varepsilon_t$ must be stationary. It can be shown that $Z_t$ is stationary if and only if the roots of the auxiliary equation $\phi\left(\frac{1}{B}\right) = 0$ all have modulo less than one.

Consider the new auxiliary equation $\tilde{\phi}\left(\frac{1}{B}\right) = 0$ where $\tilde{\phi}(B) = \phi(B)(1-B)^d$. This equation will have $d$ unit roots and $p$ roots with modulus less than one. It means that the appropriate number of differences to take is equal to the number of unit roots in $\tilde{\phi}\left(\frac{1}{B}\right) = 0$. A unit root test is used to determine if there is a unit root present in $\tilde{\phi}\left(\frac{1}{B}\right) = 0$. If there is sufficient evidence to suggest the presence of a unit root, a difference is taken. The test is then repeated on the differenced time series and a new difference is taken every time until the presence of a unit root is rejected.

The above description is an elementary illustration of how to test for unit roots, and more complicated variations exist (viz. variations that allow for trends and drifts, etc.). There are popular unit root tests such as the KPSS test (Hadri and Rao, 2009), the augmented Dickey-Fuller test (Mushtaq, 2011) and the Phillips-Perron test (Abdul-Rahim *et al.*, 2015).

The *auto.arima*() function in the forecast package in R (Hyndman and Khandakar, 2008) uses the KPSS test as a default for estimating the number of differences. In our empirical study, we show that the KPSS test recommends one difference to be taken. It is, therefore surprising to see that taking a difference actually had a detrimental effect on the performance of the model. In this context, we also note that *auto.arima*() can be forced to consider stationary models.

### 3.3.7.2 Maximum Likelihood Estimation

Once we have obtained the appropriate number of differences, we can use $Z_t = (1-B)^d Y_t$ to find MLEs for $\boldsymbol{\theta}, \boldsymbol{\phi}$ and $\sigma^2$. The key assumption is that $\varepsilon_t \sim \text{iid } N(0, \sigma^2)$. Note that because $Z_t$ is stationary we can write:

$$Z_t = \varepsilon_t + \sum_{j=1}^{\infty} \psi_j \varepsilon_{t-j}. \tag{3.10}$$

Hence, $Z_t$ is a linear combination of normally distributed random variables and will therefore also follow a normal distribution with mean 0. One can show that the random vector $\boldsymbol{Z} = (Z_1, Z_2, \ldots, Z_n)$ follows a multivariate normal distribution. Hence the log-likelihood of $Z_1, Z_2, \ldots, Z_n$ can be written as:

$$l(\boldsymbol{\Theta}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\det\left(\sum(\boldsymbol{\Theta})\right) - \frac{1}{2}\boldsymbol{Z}'[\sum(\boldsymbol{\Theta})]^{-1}\boldsymbol{Z}, \tag{3.11}$$

where $\boldsymbol{\Theta} = (\boldsymbol{\theta}, \boldsymbol{\phi}, \sigma^2)$ and $\sum(\boldsymbol{\Theta})$ is the covariance of $\boldsymbol{Z}$ which is an implicit function of $\boldsymbol{\Theta}$. We can find our MLE by computing:

$$\widehat{\boldsymbol{\Theta}} = \underset{\boldsymbol{\Theta}}{\text{argmax}} \{l(\boldsymbol{\Theta})\}. \quad (3.12)$$

Note that the covariance matrix, $\sum(\boldsymbol{\Theta})$ can be obtained by solving difference equations, although computing the likelihood this way is inefficient. The above considerations are meant to be illustrative and more complicated methods are preferred in practice. For details on how maximum likelihood estimation is performed in R, refer to Durbin and Koopman (2001).

#### 3.3.7.3 Box and Jenkins Model Identification Approach

Box and Jenkins introduced a practical approach to build ARIMA types models. This approach consists of 3 iterative steps, namely, model identification, parameter estimation and diagnostic testing to determine the best parsimonious model in a particular class of ARIMA models (Adhikari *et al*., 2013):

1. **Model Identification**: This step includes the determination of appropriate values for parameters $p$ and $q$ and the degree of differentiation for the time series to be stationary. The original time series graph or the differenced time series is used in conjunction with their estimated autocorrelation and partial autocorrelation functions as tools to identify the model parameters. If both autocorrelation and partial autocorrelation of the time series decay exponentially, then an ARMA $(p, q)$ model should be used. If the autocorrelation cuts after lag $p$ and the partial autocorrelation decay, then an MA $(q)$ model is used. If the autocorrelation decay and the partial autocorrelation cuts after lag $p$, then an AR $(p)$ model is used.
2. **Parameter Estimation:** At this step, the model parameters identified in step 1 are estimated using maximum likelihood or least squares.
3. **Diagnostic Testing**: Once the model has been identified, and its parameters estimated, diagnostic check is used to identify the model deficiencies and indicate appropriate improvements (Cadenas *et al.*, 2016). The model residuals and autocorrelation are examined.

The model is a good fit to the time series data if the residuals are white noise and have few significant autocorrelations. According to (Adhikari *et al*., 2013) these three steps are repeated several times until a satisfactory model is finally selected based on the lowest AIC or BIC values and then, the model chosen can be used for forecasting time series.

### 3.3.7.4 The AIC and BIC

In this thesis, we take an alternative approach to the Box and Jenkins methodology. The approach is implemented by the *auto.arima* () function in R. This function attempts to find the model with the lowest Akaike Information Criterion (AIC)/ Bayesian Information Criterion (BIC) from a pre-chosen set of candidate models. The AIC and BIC are applicable when maximum likelihood is used. These models penalise the maximised log-likelihood based on the number of parameters used in its estimation. The AIC and BIC are defined by:

$$\text{AIC} = -2\log l(\widehat{\boldsymbol{\Theta}}) + 2k \tag{3.13}$$

$$\text{BIC} = -2\log l(\widehat{\boldsymbol{\Theta}}) + k\,log\,n. \tag{3.14}$$

where $\widehat{\boldsymbol{\Theta}}$ is the maximum likelihood estimate (MLE) of the model parameters, $\log l(\widehat{\boldsymbol{\Theta}})$ is the value of the log-likelihood evaluated at the MLE and $k$ is the number of parameters used in the estimation. Note that the BIC penalises models more heavily than the AIC and hence prefers simpler models. (Brewer *et al.*, 2016).

According to Shumway and Stoffer (2012) various simulation study have verified that BIC tends to be superior to AIC in large samples and AIC tends to be superior in smaller samples where the relative number of parameters is large. This thesis makes use of BIC because the time series for the sites under analysis is large.

### 3.3.7.5 Forecasting

This section we briefly explain the forecasting procedure for ARMA models by assuming that the ARMA model is invertible. The ARMA model is invertible if it can be written as an AR model. Suppose a zero-mean ARMA model is:

$$\phi(B)Y_t = \theta(B)\varepsilon_t. \tag{3.15}$$

When performing forecasting, we are trying to find:

$$\hat{Y}_n(l) = E[Y_{n+l}|Y_1, Y_2, \dots, Y_n]. \tag{3.16}$$

Under the assumption of invertibility, we have:

$$Y_t = \sum_{j=1}^{\infty} \omega_j Y_{t-j} + \varepsilon_t. \tag{3.17}$$

If $n$ is large, we can use the approximation

$$Y_{n+1} \approx \sum_{j=1}^{n} \omega_j Y_{n+1-j} + \varepsilon_{n+1}\,, \tag{3.18}$$

in order to obtain:

$$\hat{Y}_n(1) = E[Y_{n+1}|Y_1, Y_2, \ldots, Y_n] \approx \sum_{j=1}^{n} \omega_j Y_{n+1-j}\,. \tag{3.19}$$

Consider, for example, finding $\hat{Y}_n(2)$. We can use $Y_{n+2} \approx \sum_{j=1}^{n+1} \omega_j Y_{n+2-j} + \varepsilon_{n+2} = \omega_1 Y_{n+1} + \sum_{j=2}^{n+1} \omega_j Y_{n+2-j}$ and hence:

$$\hat{Y}_n(2) \approx \omega_1 E[Y_{n+1}|Y_1, Y_2, \ldots, Y_n] + \sum_{j=1}^{n} \omega_{j+1} Y_{n+1-j}$$

$$= \omega_j \hat{Y}_n(1) + \sum_{j=1}^{n} \omega_{j+1} Y_{n+1-j}\,.$$

This procedure is repeated to obtain approximations for any $\hat{Y}_n(l)$. Again, these considerations are meant to be illustrative. For exact details on the implementation of the forecasting of ARMA models in R, refer to Durbin and Koopman (2001).

### 3.3.8 Periodogram using the Fast Fourier Transform

Before applying models that support modelling of the seasonal component of a time series, the periodic behaviour in a series needs to be identified ahead of time. Refer to Cryer & Chan (2008) for a detailed description of the summary below. Important frequencies (or periods) in a time series can be identified by making use of a periodogram. Consider the stochastic process defined by:

$$Y_t = Rcos(2\pi f t + \phi) + \varepsilon_t\,, \tag{3.20}$$

where $R$, $f$, $\phi$, denote the amplitude, frequency and phase of the curve and $\varepsilon_t$ follows a white noise process. Let $k$ be a positive integer, then:

$$Y_{t+\frac{k}{j}} = Rcos\left(2\pi f\left(t + \frac{k}{f}\right) + \phi\right) + \varepsilon_{t+\frac{k}{f}}$$

$$= Rcos(2\pi f t + 2\pi k + \phi) + \varepsilon_{t+\frac{k}{f}}$$

Since $cos(2\pi + \phi) = cos(\phi)$

$$Y_{t+\frac{k}{j}} = Rcos(2\pi f t + \phi) + \varepsilon_{t+\frac{k}{f}}$$

$$= Y_t + \left(\varepsilon_{t+\frac{k}{f}} - \varepsilon_t\right).$$

Hence, we have $E\left(Y_{t+\frac{k}{j}}\right) = E(Y_t)$ and the stochastic process will tend to repeat itself in increments of $\frac{1}{f}$ . The quantity $\frac{1}{f}$ is known as the period of the stochastic process, which is the reciprocal of the frequency. The parameters $R$ and $\phi$ in equation (3.20) are unknown and appear in the equation in a nonlinear way which makes (3.20) difficult for estimation. Trigonometric identities can be used to rewrite equation (3.20) and make it suitable for estimation.

**Fourier Transform**

Note the trigonometric identity:

$$Rcos(2\pi ft + \phi) = Acos(2\pi ft) + Bsin(2\pi ft)\,, \tag{3.21}$$

where $A = Rcos(\phi)$ and $B = -Rsin(\phi)$ . Therefore, we can rewrite equation (3.20):

$$\begin{aligned} Y_t &= Rcos(2\pi ft + \phi) + \varepsilon_t \\ &= R\sin(\phi)\,cos(2\pi ft) + R\cos(\phi)\,sin(2\pi ft) \\ &= Acos(2\pi ft) + Bsin(2\pi ft) + \varepsilon_t\,. \end{aligned} \tag{3.22}$$

Equation (3.22) is a signal made up of a sum of sine and cosine functions with the same frequency and different amplitudes. The main idea is to estimate the amplitude $R$ and the phase $\phi$ of the functions, and this can be estimated by making use of $A$ and $B$ in equation (3.22). The amplitude $R$ and phase $\phi$ can be calculated as:

$$R = \sqrt{A^2 + B^2} \quad \text{and } \phi = \operatorname{atan}\left(-\frac{B}{A}\right).$$

Suppose we have observed the time series $Y_1, Y_2, \ldots, Y_n$ . Our goal is to determine its frequency components by finding the best combinations of sines and cosines of different frequencies and amplitudes that will sum up to approximate the time series. It can be shown that

$$Y_t = A_0 + \sum_{j=1}^{k}\left[A_j \cos\left(2\pi f_j t\right) + B_j \sin\left(2\pi f_j t\right)\right], \tag{3.23}$$

for $t = 1,2,3, \ldots, n$. If $n$ is even, then $k = \frac{n}{2}$ otherwise $\frac{n-1}{2}$. The coefficients in (3.23) are called Fourier coefficients and they can be computed as:

$$A_0 = \bar{Y}$$

$$A_j = \frac{2}{n}\sum_{t=1}^{n} Y_t \cos\left(2\pi f_j t\right) : j = 1,2,3,\dots,k-1$$

$$B_j = \frac{2}{n}\sum_{t=1}^{n} Y_t \sin\left(2\pi f_j t\right) : j = 1,2,3,\dots,k-1$$

The coefficient at $j = k$ depends on $n$. If $n$ is odd, we have $A_k = \frac{2}{n}\sum_{t=1}^{n} Y_t \cos(2\pi f_k t)$ and $B_k = \frac{2}{n}\sum_{t=1}^{n} Y_t \sin(2\pi f_k t)$. Otherwise:

$$A_k = \frac{1}{n}\sum_{t=1}^{n}(-1)^t Y_t$$

$$B_k = 0$$

The representation of time series in the form of Equation (3.23) is called a representation in the frequency domain. For a long time series, the calculation of these coefficients can be intensive. However, the fast Fourier transform (FFT) can be used to perform these calculations efficiently.

**The Periodogram**

The Fourier transform discussed in the previous section can be used to identify the period of a time series. The periodogram at the frequency $f_j$ is defined as

$$I(f_j) = \frac{n}{2}\left(A_j^2 + B_j^2\right). \tag{3.24}$$

For $j = 1,2,3,\dots,k$. When the sample size $n$ of a time series is even, at $j = k$, we have:

$$I(f_k) = I\left(\frac{1}{2}\right) = nA_k^2. \tag{3.25}$$

We can interpret the periodogram values as measuring the relative strength of the sine-cosine pairs at different frequencies. If $I(f_k)$ is large, relative to the other periodogram values, then $\frac{1}{f_j}$ should be considered for the period of the time series. The periodogram is a plot of $f_j$ on the x-axis against $I(f_j)$ on the y-axis. Figure 3.1 displays an example of a periodogram to show how the plot is interpreted.

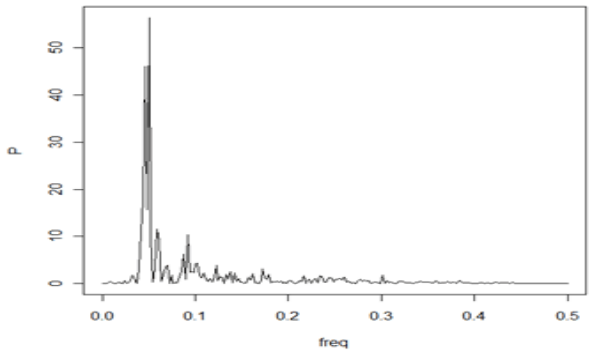



**Fig. 3.1: An example of a periodogram**

The periodogram is for a semi-annual data taken from https://online.stat.psu.edu/stat510/lesson/6/6.1. Looking at the plot, the large periodogram value seems to occur between frequencies 0.0 and 0.1 (in the middle). This indicates that the dominant peak occurs around frequency, $f$ = 0.05 and corresponds to the period of 1/0.05 =20 times. The data is semi-annual which means the dominant cycle seems to be every 10 years.

### 3.3.9 Regression using Fourier Terms with ARMA Errors

High-frequency data (daily or shorter time intervals time series) tend to have a large and/or non-integer seasonal period which makes it hard to fit a traditional time series such as SARIMA models. The SARIMA models can be applied to the data using the *Arima* function from the forecast package in R, which require the seasonal period to be an integer and not longer than 350 (Hyndman *et al*., 2013). To forecast high-frequency data with accuracy, a Fourier series approach defined in section 3.3.8 can be used to model the seasonal pattern of the time series using Fourier terms of the seasonal period.

Consider the following linear regression model

$$Y_t = \boldsymbol{\beta}' \boldsymbol{X}_t + v_t \,. \tag{3.26}$$

For $t = 1,2,3, \dots, n$. Where, $\boldsymbol{\beta}$ represents a vector of coefficients, $\boldsymbol{X}_t$ denotes a vector of covariates and $v_t$ denotes the residual of the regression. It is typically assumed that the residuals, $v_t$ arise identically and independently from a specified underlying distribution (usually normal with zero mean).

In regression with ARMA errors, we relax this assumption by allowing the sequence $\{v_t : t = 1,2,3, \dots, n\}$ to have autocorrelation. In particular, we assume that an appropriate ARMA process can capture this autocorrelation structure.

**Regression using Fourier Terms**

We now introduce an alternative method of modelling seasonality in a time series. This method is based on the regression model with ARMA errors, where Fourier terms are used as regressors. The key property is that any periodic function with a period of $m$ can be approximated by:

$$Y_t = a_0 + \sum_{k=1}^{K} \left[ a_k \cos\left(\frac{2\pi kt}{m}\right) + b_k \sin\left(\frac{2\pi kt}{m}\right) \right], \tag{3.27}$$

where a set of constants $a_0$, $a_k$ , $b_k$ for $k = 1,2,3, \ldots . K$ are called Fourier coefficients. Each $k$ represents a Fourier term, sometimes called a harmonic, and $K$ is the number of Fourier terms for the seasonal period $m$ (Hyndman *et al.*, 2013). This approximation can be made arbitrarily well by making $K$ sufficiently large. The regression using Fourier terms with ARMA errors method can be represented as:

$$Y_t = \sum_{k=1}^{K} \left[ a_k \cos\left(\frac{2\pi kt}{m}\right) + b_k \sin\left(\frac{2\pi kt}{m}\right) \right] + v_t \ , \tag{3.28}$$

where $v_t$ denotes an ARMA process. The idea is relatively straight-forward:

1. Identify the period of the time series using a periodogram analysis.
2. Specify the number of harmonics $K$, and include the pairs $\left\{\cos\left(\frac{2\pi kt}{m}\right) \ , \ \sin\left(\frac{2\pi kt}{m}\right)\right\}$ for $k = 1,2 \ ,3, \ldots , K$ as covariates in the linear regression with ARMA errors model.

Note that in R, the $Arima()$ and $auto.arima$ () functions can perform linear regression with ARMA errors. By assuming $v_t$ to have an autocorrelation structure captured by a ARMA process with Gaussian white-noise, maximum likelihood estimation is performed, and the AIC or BIC can be used to select the number of harmonics.

This approach can easily be extended to time series with multiple seasonality, simply by adding more Fourier terms for each period in the regression. In this thesis, we only considered modelling a single period.

## 3.4 Artificial Neural Networks (ANNs)

ANNs emerged from the idea of imitating the functioning of the human brain to solve complex problems. An ANN is described as a computing system consisting of a set of simple highly interconnected processing components which process data through its dynamic response to external inputs. The use of time series methods in forecasts perform poorly with increasing time steps, and this has been resolved by making use of ANNs. ANNs have existed since the 1950s, but in the past

years, the methods have been made more dominant due to improvements in computer architecture and performance (Ciaburro *et al.*, 1996). Neural networks are mathematical models with approximation functions, and only numerical data can be used in these models.

ANNs attempt to recognise patterns and regularities in the input data, learn from the experience and then generalise outcomes. This enables neural network models to perform well, even without the researcher's understanding of the problem (De Alencar *et al.*, 2017).

Suppose we have a vector of inputs $\boldsymbol{x}$ and a vector of outputs $\boldsymbol{y}$, then ANNs can be defined as a function, $f(.)$ which maps the inputs to outputs, represented as $f: \boldsymbol{x} \rightarrow \boldsymbol{y}$. ANNs can be used whenever a relationship between input variables and output variables exists. However, ANNs represent black boxes, as we do not know how each input variable is affecting the output variables.

In many fields such as finance, economics and others, ANNs were implemented, particularly for prediction and classification (Kamruzzaman *et al.*, 2006).

### 3.4.1 The architecture of a Neural Network

Any neural network consists of a set of neurons (nodes) connected by edges. Neurons are basic processing elements that are positioned within three consecutive layers, namely the input layer, the hidden layer(s) and the output layer. Note that there can be any number of input neurons, hidden neurons, hidden layers and output neurons in a particular neural network and each layer makes independent computations (Beccali and Brano, 2014). A neuron conducts a mathematical operation to produce one output from a set of inputs, and this is similar to the biological neuron structure. The operation is displayed in Figure 3.2. A neuron can receive input and send out output to the next neuron. Consider two neurons, $i$ and $j$. Each neuron contains an activation function $\sigma$ and each edge from neuron $i$ to neuron $j$ is associated with a weight $w_{ij}$. The output from neuron $i$ is the input at neuron $j$.

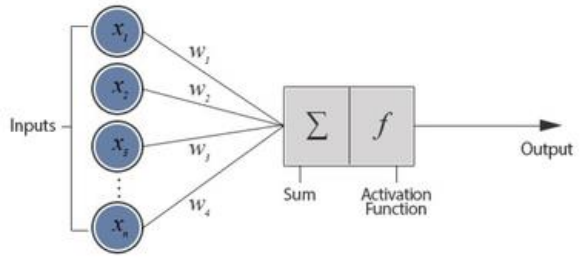



**Fig. 3.2: Computation performed by a NN neuron**

The interaction within any neural network is based on the layered structure. There are two popular ANNs; feed-forward neural networks (FFNNs) and recurrent neural networks (RNNs), illustrated in Figure 3.3 (Pekel and Kara, 2017).

a) FFNNs are the first and most simple neural network design. The data is processed across layers without any back loops (Hewamalage *et al.*, 2019). The connections between neurons are in one direction, and there are no connections between the neurons in the same layer. MLP and radial basis function (RBF) NNs are FFNNs and have been used in various applications, including wind speed prediction (Beccali and Brano, 2014).

b) RNNs are similar to FFNNs, only that the output of the input signals can be fed back to a neuron in the input layer or hidden layer where it has already been processed. RNNs have the ability, through training, to store the learned sequences or patterns. These networks perform well with time series or pattern recognition problems, which require internal memory to strengthen the learning process.

Examples of RNNs include single layer RNNs and long short-term memory (LSTM) RNNs (Pekel and Kara, 2017).

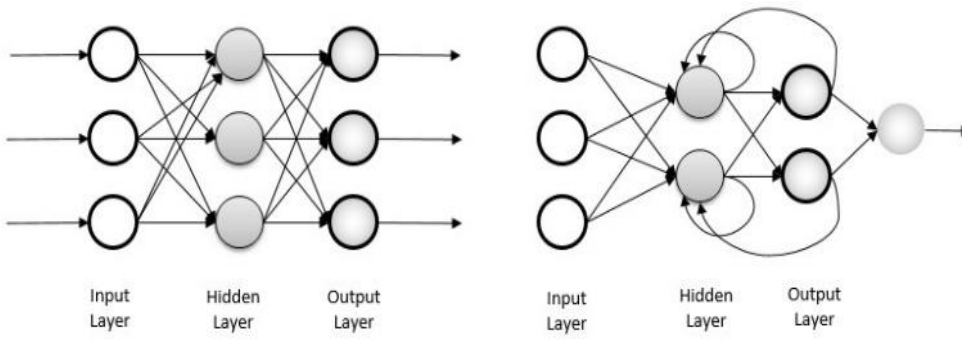



**Fig. 3.3: Architecture of neural networks. Left: FFNN. Right: RNN by** (Pekel and Kara, 2017)

Important parameters need to be considered when training any neural network, such as the number of hidden layers in the network, the number of neurons per layer, the weights connecting the neurons, the learning rate, activation functions and epochs.

#### 3.4.1.1 Activation Functions

The activation function can be defined as a mathematical function that transforms the input into an output. The activation function determines the amplitude of the outputs based on the weighted net inputs for the neuron and the bias factor applied (Nwankpa *et al.*, 2018a). An appropriate activation function modifies the learned patterns in the data by the network. There are various activation functions used in neural networks. Activation functions can be linear or non-linear, and it depends on the objective of the layer in the neural network it represents. Linear activation functions, non-linear activation functions and rectified linear unit (ReLU) activation function will be briefly discussed.

##### 3.4.1.1.1 Linear Activation Function

This function transforms the input directly proportional to the output. A linear function is a polynomial of order one. This activation function is frequently used for multilayer neural networks together with non-linear activation functions. The linear function is usually used for linear transformation for the output layer when regression is relevant, and a non-linear activation function is generally used for the hidden layers.

The linear activation function can be represented as:

$$\sigma(x) = x,$$

where $x$ is the input.

##### 3.4.1.1.2 Non-linear Activation Function

ANNs are used to solve nonlinear and complex problems. Non-linear activation functions are needed to model the nonlinearity between the input and output vectors. These functions are more complex than linear functions. Commonly used nonlinear activation functions for neural networks are the sigmoid function and the hyperbolic tangent function. In multilayer neural networks, these functions are used to perform nonlinear transformations of the input vector using a defined algorithm. The sigmoid function is used in neural networks where the desired output value is in the range 0 to 1, making the model logistic in nature. This function is sometimes referred to as a logistic function (Nwankpa *et al.*, 2018b). The sigmoid function is expressed as:

$$\sigma(x) = \frac{1}{1+ e^{-x}} ,$$

where $x$ is the input vector.

The hyperbolic tangent function (tanh) is another nonlinear activation function. This function is used in multilayer networks where the desired output value is in the range -1 to 1. The tanh function can be expressed as:

$$\sigma(x) = \tanh(x) = \frac{1-\exp(-\theta x)}{1+\exp(-\theta x)} \ .$$

The tanh function became the preferred function to the sigmoid because it gives better training performance for multilayer networks. The tanh function produces zero centred output (-1 to 1) which aids the back-propagation process (Nwankpa *et al.*, 2018b).

#### 3.4.1.1.3 Rectified Linear Unit (ReLU) Activation Function

The ReLU activation function is a piecewise linear function and has been used in various neural networks. The following expression defines the ReLU function:

$$\sigma(x) = \begin{cases} 0; & x < 0 \\ x; & x \geq 0 \end{cases}.$$

This function outputs the input directly if it is positive, and when it is negative, it outputs zero. ReLU has an advantage over other functions such as sigmoid and tanh. The derivative of the sigmoid function is $\sigma'(x) = \sigma(x)\big(1 - \sigma(x)\big)$. This function approaches zero quickly for large $x$, and this can cause the vanishing gradient problem during training. The vanishing gradient problem means that weights of earlier layers are not updated, and hence do not learn. When the gradients are multiplied by a small learning rate, this problem is amplified. The ReLu function has been used as a default activation function for many neural networks such as MLPs and convolutional neural networks. The ReLU activation function can be used for hidden layers while a linear function is used for the output layer in deep neural networks for regression problems.

The following sections discussed the key concepts of each neural network used in this thesis.

### 3.4.2 Multilayer Perceptron (MLP)

MLPs are the most commonly used ANNs in forecasting problems (Beccali and Brano, 2014), (Zucatelli *et al.*, 2019). Figure 3.4 is a typical example of a simple neural network with one layer of units between the input layer and the output layer.

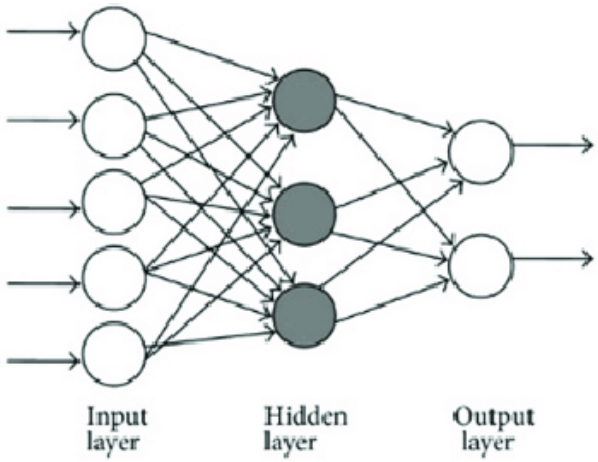



**Fig. 3.4: Simple multilayer perceptron** (Zadeh, 2017)

Consider a single hidden layer MLP as depicted in Figure 3.4. Assume inputs $x_1, x_2, \ldots, x_p$ and outputs $y_1, \ldots, y_r$. Each input node corresponds to a single input variable and let $h_1, \ldots, h_q$ denote the outputs from the nodes in the hidden layer. The number of nodes in the input and output layers corresponds to the size of the input and the desired output, respectively. Each node in the hidden layer receives its own bias $b_j$ and a set of weights $w_{jt}$ for $t = 1,2, \ldots, p$ . These weights control the process in each node and are estimated from the data through training. The weights are tuned through training of the neural networks in such a way that when the network is given a set of inputs data, the network predicted output is close as possible to the desired outcome. This means we seek an optimal set of weights that minimises the error in the network.

First, a linear combination of the inputs is formed:

$$L_j = \sum_{t=1}^{p} w_{jt} x_t.$$

This linear combination $L_j$, is a dot product of the inputs and a set of weights $w_{jt}$ , sum up over $t = 1,2, \ldots, p$.

The output of the $jth$ node in the hidden layer is obtained by passing this linear combination through an activation function with a bias:

$$h_j = \sigma(b_j + L_j).$$

It is essential to take note that several activation functions can be used, depending on the nature of the desired output and the inputs. This process is repeated for all nodes in the hidden layer. Associated with $\hat{y}_j$ are the weights $v_{jt}$ for $t = 1,2, \ldots\ldots, q$. The output of the neural network is:

$$\hat{y}_j = s(c_j + \sum_{t=1}^{q} v_{jt} h_t), \tag{3.29}$$

for $j = 1,2, \ldots\ldots, r$, where $c_j$ denotes a bias associated with the output layer and $h_t$ is the outputs from the nodes in the hidden layers. The activation function $s(.)$ differs from $\sigma(.)$, and its specification depends on the context of the problem, for example, the identity operator for regression and the sigmoid activation function for binary classification. Here $\sigma(.)$ denotes the activation function applied to obtain the outputs of the hidden nodes, and $s(.)$ represents the activation function applied to get the output of the neural network.

This process can be view as a function $\hat{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\Theta})$ where $\boldsymbol{\Theta} = \{\boldsymbol{b}, \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{V}\}$ are the parameters to be estimated using the training data $(\boldsymbol{y}_i, \boldsymbol{x}_i): i = 1,2, \ldots\ldots, n$. Here $\boldsymbol{W}$ and $\boldsymbol{V}$ represent the weight matrices, $\boldsymbol{b}$ and $\boldsymbol{c}$ are the bias vectors associated with the inputs and hidden nodes output received in the output layer, respectively. To facilitate this training, there is a need to specify a loss function $\boldsymbol{L}(\boldsymbol{y}, \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\Theta}))$ which measures the loss for trying to estimate $\boldsymbol{y}$ through the neural network. We attempt to find:

$$\widehat{\boldsymbol{\Theta}} = \underset{\boldsymbol{\Theta}}{\operatorname{argmin}} \{\sum_{i=1}^{n} L(\boldsymbol{y}_i, \boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{\Theta}))\} \tag{3.30}$$

To solve this optimisation problem is typically difficult, owing to a large number of parameters and the non-convexity of the objective. Practically, batch gradient descent is usually applied to the objective in attempt to find a solution.

It is possible to differentiate the objective with respect to its parameters through the use of a process known as backpropagation, i.e. we can evaluate $\frac{\partial L(\boldsymbol{y}, \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\Theta}))}{\partial \boldsymbol{\Theta}}$. Backpropagation is a process that can be used to evaluate the gradients of functions and is useful in the evaluation of the gradients of an MLP (Goodfellow *et al.*, 2016, p. 208). For an explanation of backpropagation, refer to Goodfellow *et al.* (2016, p. 200).

The method of batch gradient descent is applied as follows:

(a) Split the training data into (approximately) equal-size groups: $G_1, G_2, \ldots, G_K$.

(b) Initialise the parameters as $\boldsymbol{\Theta}^{(0)}$ and choose the maximum number of epochs $m^*$.

(c) For $m = 1,2, \ldots, m^*$ do:

i. Set $\boldsymbol{\Theta}^{(m)} = \boldsymbol{\Theta}^{(m-1)}$

ii. For $k = 1,2, \ldots, K$ do

- Find $D_k\left(\boldsymbol{\Theta}^{(m)}\right) = \left[\frac{\partial}{\partial \boldsymbol{\Theta}} \sum_{i \in G_k} L(\boldsymbol{y}_i, \boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{\Theta}))\right]_{\boldsymbol{\Theta}=\boldsymbol{\Theta}^{(m)}}$
- Update $\boldsymbol{\Theta}^{(m)} \leftarrow \boldsymbol{\Theta}^{(m)} - \alpha D_k(\boldsymbol{\Theta}^{(m)})$

(d) Output $\boldsymbol{\Theta}^{(m^*)}$.

In the above procedure, one pass through all the batches is called an epoch. The parameter $\alpha$ is called the learning rate of the procedure. Note that the above procedure is a fundamental formulation of batch gradient descent. Many more sophisticated learning procedures exist, such as the Adam optimiser (Kingma and Ba, 2014; Reddi *et al.*, 2018), which will be used in this thesis. The above procedure can also be generalised to allow for the training of MLP with an arbitrary number of hidden layers. The parameters are updated at every epoch until the model converges. This reduces the overall error at the output layer.

### 3.4.3 Recurrent Neural Networks (RNNs)

MLPs lack the ability to address future inputs based on the previous inputs because hidden layers in the network are independent of one another, and the network does not memorise the previous output (DataFair, 2019). RNNs include edges that span adjacent time steps which are called recurrent edges. These edges can form cycles and introduce the notion of time to the model (Pekel and Kara, 2017). Recurrent neural networks are used for sequential data and can be used for predicting time series values (Hewamalage *et al.*, 2019). Below is the simple architecture.

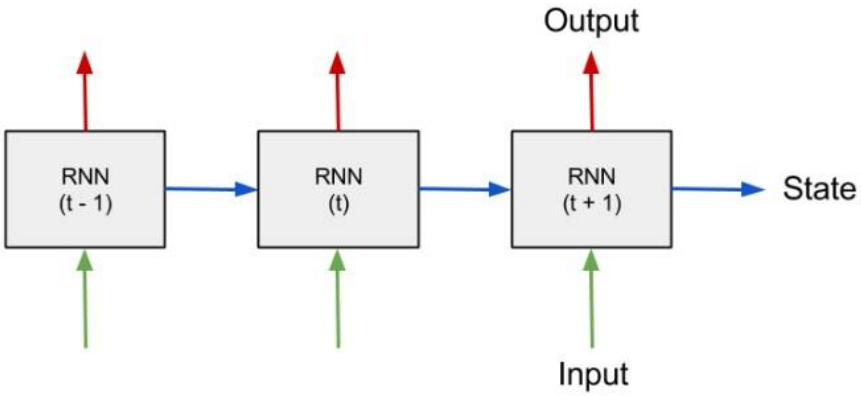



**Fig. 3.5: Architecture of an RNN by DataFair (2019)**

The input to RNN at every time step is the current value at that time as well as a state vector representing what the network has seen in previous time steps as indicated in Figure 3.5. The input at time step $t$ depends on an output from time step $t-1$ (Dontas, 2010). The state vector is the RNN's memory. For time series forecasting scenario, the value of the next timestep may depend on more than one previous timesteps. How far the model wants to look in the past is determined by analysing the autocorrelation function of the data.

There are various structures of RNNs, and we will first introduce the simple main structure before we discuss LSTM, which has been used in this thesis to analyse the time series. We will consider many-to-one RNNs where we have a set of $p$-dimensional vectors $\mathbb{Z} = \{z_1, z_2, \dots, z_T\}$, with a sequential structure, to be used to predict a target $\boldsymbol{y}: r \times 1$.

### 3.4.3.1 Vanilla RNNs

The idea behind RNNs is to associate a state vector $\boldsymbol{h}_t : q \times 1$ with each time point in the sequential structure. These state vectors are built up sequentially over time. Let $\boldsymbol{h}_0$ be an initial state and $\boldsymbol{\sigma}()$: $\mathfrak{R}^q \rightarrow \mathfrak{R}^q$ be a vector function where the chosen activation function (usually the hyperbolic function) is applied element-wise to its input vector. In a vanilla RNN, the states are updated through the following equation

$$\boldsymbol{h}_t = \boldsymbol{\sigma}(\boldsymbol{b} + \boldsymbol{U}\boldsymbol{h}_{t-1} + \boldsymbol{V}\boldsymbol{z}_t), \tag{3.31}$$

where $\boldsymbol{z}_t$ represents the input vector, $\boldsymbol{h}_{t-1}$ is the previously hidden state vector, $\boldsymbol{b}: q \times 1$ denote the bias vector and $\boldsymbol{U}: q \times q$ and $\boldsymbol{V}: q \times p$ are the weight matrices of the RNN. The weighting matrices $\boldsymbol{V}$ and $\boldsymbol{U}$ denotes the set of weights associated with the inputs and hidden state, respectively. If $\boldsymbol{U} =$

$\mathbf{0}$, then it will be a feed-forward network at each time step (Dontas, 2010b). We compute $\boldsymbol{h}_t: t = 1,2,3, \dots, T$ and pass the final hidden state through an MLP:

$$\hat{\boldsymbol{y}} = MLP(\boldsymbol{h}_T;\ \boldsymbol{\Theta}) = RNN(\mathbb{Z}; \boldsymbol{\Omega}), \tag{3.32}$$

where $\hat{\boldsymbol{y}}$ denotes the predicted output and $\boldsymbol{\Theta}$ and $\boldsymbol{\Omega} = \{\boldsymbol{\Theta}, \boldsymbol{b},\ \boldsymbol{U}, \boldsymbol{V}\}$ contain the parameters of the MLP and RNN, respectively. It is important to note that the set of weights in $\boldsymbol{U}$ and $\boldsymbol{V}$ determine the importance of each input vector to the target or desired output. If the weights associated with the input vector are large, it is an indication of greater contribution toward the desired outcome.

When training an RNN, we represent the training data by $\{\mathbb{Z}_i,\ \boldsymbol{y}_i\} : i = 1, 2, \dots, n$ and a loss function by $\boldsymbol{L}(\boldsymbol{y}, \boldsymbol{f}(\mathbb{Z};\ \boldsymbol{\Omega})))$. The parameters in $\boldsymbol{\Omega}$ are unknown and need to be estimated through training of the network. To find appropriate parameters, we attempt to minimise the empirical risk

$$R(\boldsymbol{\Omega}) = \sum_{i=1}^{n} L(\boldsymbol{y}_i, RNN(\mathbb{Z}_i; \boldsymbol{\Omega})), \tag{3.33}$$

through the use of batch gradient descent. The calculations of the derivatives of an RNN is discussed by Goodfellow *et al. (*2016, p. 378).

This structure of the RNN is not capable of carrying long-term dependencies to the future because it suffers from the vanishing gradient problem. The propagated gradients tend to vanish when the sequences are long and causing the weights not to be updated adequately.

Before we move on to the idea of LSTMs, we introduce the concept of a cell state to give a clear differentiation between the two structures. For vanilla RNNs, we interpret the cell state as $\boldsymbol{c}_t = \boldsymbol{b} + \boldsymbol{U}\boldsymbol{h}_{t-1} + \boldsymbol{V}\boldsymbol{z}_t$ and the hidden state as $\boldsymbol{h}_t = \boldsymbol{\sigma}(\boldsymbol{c}_t)$. The hidden state receives the output of the cell state as input, and an activation function $\boldsymbol{\sigma}()$, usually a tanh function is applied. We also note that we can express equation (3.31) as:

$$\boldsymbol{h}_t = \boldsymbol{\sigma}\left(\boldsymbol{W}\begin{bmatrix}\boldsymbol{h}_{t-1}\\ \boldsymbol{z}_t\end{bmatrix} + \boldsymbol{b}\right), \tag{3.34}$$

where $\boldsymbol{W} = [\boldsymbol{U} \quad \boldsymbol{V}]$.

### 3.4.3.2 LSTM - RNNs

Every RNN is a combination of numerous RNN units, and there are two popular and effective RNN units used for sequential data (Hewamalage *et al.*, 2019): long-short-term memory (LSTM) and gated recurrent unit (GRU). LSTM became popular because of its capability to capture long term dependencies. LSTM is a gated memory unit with three gates (input gate, output gate, and forget gate)

that monitor the memory data. These gates are basic logistic functions of the weighted sums. GRU is a simplified LSTM version. They have the same role in the network, and the only difference is that GRU has two gates, with no output gate (Zhang, 2012). This study will make use of LSTMs and Figure 3.6 shows the architecture of LSTM cells.

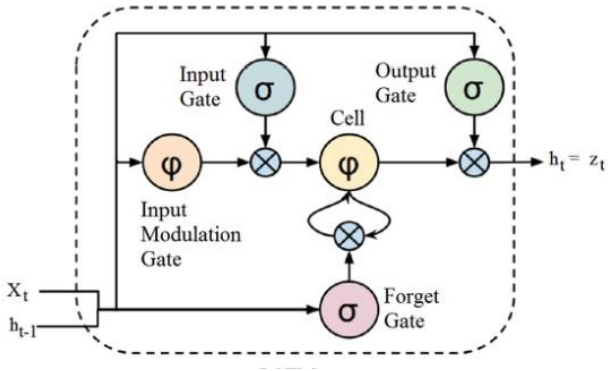



**Fig. 3.6: Architecture of LSTM cell (Kang, 2017)**

The standard/vanilla RNNs have typical nodes in the hidden layer, and each typical node is replaced with a memory cell in the LSTM structure. A memory cell is a complex unit built from simpler nodes in a specific connectivity pattern (Lipton *et al*., 2015). The memory cell enables the network to remember for a long time, and the hidden state is for the short-term memory component.

There are different types of activation functions in the LSTM cell, and we will define them differently because each activation function plays a different role in the cell. Let $\boldsymbol{\sigma}_s(.)$ denotes the sigmoid function and its associated vector function by $\boldsymbol{\sigma}_s(.)$: $\mathfrak{R}^q \rightarrow \mathfrak{R}^q$. Similarly, we denote the hyperbolic tangent function by $\boldsymbol{\sigma}_h(.)$: $\mathfrak{R}^q \rightarrow \mathfrak{R}^q$ .

To explain the different roles played by the activation functions, we start by considering the final step in the LSTM cell, which is the hidden state:

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \boldsymbol{\sigma}_h(\boldsymbol{c}_t), \tag{3.35}$$

where $\boldsymbol{o}_t = \boldsymbol{\sigma}_s\left(\boldsymbol{W}_0 \begin{bmatrix} \boldsymbol{h}_{t-1} \\ \boldsymbol{z}_t \end{bmatrix} + \boldsymbol{b}_0\right)$ represents the output gate, $\boldsymbol{c}_t$ denote the cell state and $\odot$ stands for element-wise multiplication. Each component of the output gate $\boldsymbol{o}_t$ is between 0 and 1. The output gate decide which information of $\boldsymbol{\sigma}_h(\boldsymbol{c}_t)$ is kept or lost. A value of $\boldsymbol{o}_t$ that is close to one can be interpreted as the information will be kept and a value close to zero is otherwise.

The generation of the cell state $\boldsymbol{c}_t$ is slightly more complicated than in the vanilla RNN case. We generate a new candidate for the cell state at time $t$:

$$\tilde{\boldsymbol{c}}_t = \boldsymbol{\sigma}_h\left(\boldsymbol{W}_c \begin{bmatrix} \boldsymbol{h}_{t-1} \\ \boldsymbol{z}_t \end{bmatrix} + \boldsymbol{b}_c\right), \tag{3.36}$$

This is the input modulation gate in Figure 3.5. It has a tangent activation function which takes activation from the current input $\boldsymbol{z}_t$ as well as from the hidden layer at the previous time step $\boldsymbol{h}_{t-1}$. Note that this is the hidden state obtained in a vanilla RNN.

The next step is to decide which of the values of the old and new cell state to keep. For this purpose, the input gate and the forget gates are computed as:

$$\boldsymbol{g}_t = \boldsymbol{\sigma}_s\left(\boldsymbol{W}_g \begin{bmatrix} \boldsymbol{h}_{t-1} \\ \boldsymbol{z}_t \end{bmatrix} + \boldsymbol{b}_g\right), \tag{3.37}$$

$$\boldsymbol{f}_t = \boldsymbol{\sigma}_s\left(\boldsymbol{W}_f \begin{bmatrix} \boldsymbol{h}_{t-1} \\ \boldsymbol{z}_t \end{bmatrix} + \boldsymbol{b}_f\right), \tag{3.38}$$

where $\boldsymbol{g}_t$ denote the input gate and $\boldsymbol{f}_t$ denote the forget gate. The forget gate controls which information from the previous cell state to include in the new cell state, while the input gate decides which information to use from the candidate cell state. These gates manage the cell state and at time $t$, which is computed as:

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{g}_t \odot \tilde{\boldsymbol{c}}_t, \tag{3.39}$$

where $\boldsymbol{c}_{t-1}$ is the previous cell state.

Equation (3.37) for the input gate is then substituted into equation (3.34), the hidden state expressed in the vanilla RNN section to obtain the new hidden state. As in Equation (3.32), the final hidden state is then passed through an MLP to obtain the prediction:

$$\hat{\boldsymbol{y}} = MLP(\boldsymbol{h}_T;\ \boldsymbol{\Theta}) = LSTM(\mathbb{Z}\,;\ \boldsymbol{\Psi}), \tag{3.40}$$

where $\boldsymbol{\Psi} = \{\boldsymbol{W}_0, \boldsymbol{W}_c, \boldsymbol{W}_g, \boldsymbol{W}_f, \boldsymbol{b}_0, \boldsymbol{b}_c, \boldsymbol{b}_g, \boldsymbol{b}_f, \boldsymbol{\Theta}\}$ are parameters to be estimated using the training data through the empirical risk as in Equation (3.33):

$$R(\boldsymbol{\Psi}) = \sum_{i=1}^{n} L(\boldsymbol{y}_i, LSTM(\mathbb{Z}_i; \boldsymbol{\Psi})). \tag{3.41}$$

Again, the derivatives can be calculated using backpropagation and update the parameters using batch gradient descent. The optimization of LSTMs is discussed by Goodfellow et al. (2016, p. 378). For an alternative description of LSTMs, refer to Nguyen (2018).

Time series has various events occurring one after the other, and understanding each event requires remembering the previous events. Due to the memory cell in the RNNs, it performs tasks that MLPs cannot. In this study, RNNs and MLPs will be compared. The MLP with a linear activation function (purelin) in the output layer and non-linear activation function in the hidden layer to introduce

nonlinearity in the model will be used for wind speed forecasting. Note that the training of RNN could be challenging with a large number of inputs and hidden layers, but it tends to perform better with highly varying noisy data compared to MLPs.

### 3.4.4 Issues with Training Neural Networks

Some issues need to be considered when training a neural network and will be discussed in this section. We will also discuss a method of reading in a time series in an appropriate format for MLPs and RNNs.

#### 3.4.4.1 The number of Hidden Layers and Neurons

There are only one input layer and one output layer in a typical neural network, but it can have multiple hidden layers. Before creating the neural network configuration, the number of hidden layers and the number of neurons in each layer need to be determined. The size of the input layer and the output layer is determined by the number of inputs to the network and the desired outputs, respectively. Choosing the number of hidden layers or the size of the hidden layer to produce accurate results is the most challenging step in modelling. Increasing the number of hidden layers in the network increases the complexity of the network and training this network can take time to converge. When the size of the hidden layer in the network is small, the network might not be able to learn the patterns in the data and produce poor results. When the size of the hidden layer is too large, the network tends to overfit the data. Overfitting means the network performs well on the training data but produce large testing errors.

Different authors have proposed different rules for choosing the size of the hidden layer. Heaton (2008) mentioned that the optimal size of the hidden layer is between the size of the outputs and the inputs. In this thesis, we model using different sizes of the hidden layers and choose the size of the hidden layer based on the neural network validation errors.

#### 3.4.4.2 The Weight Initialisation

There are weights between nodes in a neural network, and the weighted sum of the inputs are the input to the activation function. Through the training of a neural network, weight initialisation is

important for the network to learn well. Training a network using backpropagation can find several local minima, and proper weights initialisation helps the network to find a good minimum (Raut *et al.* 2018). If the weights are not properly initialised, it increases the number of iterations needed, causing the network to take long to converge. When searching for appropriate weights, each neuron weights should be unique to avoid neurons to compute the same outputs. We want to initialise the weights, not too small and not too large. If we initialise the weights to be too large, the neuron activation outputs tend to be overly large (explode) and when the weights are too small, the neuron outputs vanish entirely.

Different weight initialisation algorithms have been proposed, such as the Glorot uniform initialiser (also called Xavier uniform initialiser) and random initialisation. With random initialisation, the weights are drawn from a normal distribution with mean zero and standard deviation one. Xavier uniform initialiser draws samples from a uniform distribution within sqrt (6/ (number of input units + number of output units)) (Glorot and Bengio, 2010). The Xavier uniform initialiser was used for this thesis as it was set as a default initialisation algorithm in *keras* package.

#### 3.4.4.3 The Learning Rate

The learning rate controls how much to change the model in response to the errors obtained each time weights are updated during training. This hyperparameter is a positive value between 0 and 1. If the learning rate is small, it means that from one training cycle to the next, the change in the weight vector is small and when the learning rate is large, the changes in the weight vector is large. Choosing the learning rate is hard because small learning rate may lead to vanishing gradients such that network cannot learn (huge bias) and large learning rate causes loss to diverge or oscillate (huge variance). When searching for appropriate learning rate, Brownlee (2019) mentioned that a small learning rate requires more training cycles compare to a large learning rate.

For this thesis, different learning rate values were used in all the configurations (MLPs and RNNs) created to give satisfactory results. We will now discuss how to convert a univariate time series into an appropriate format for MLPs and RNNs.

#### 3.4.4.4 Time Series Data for MLPs

Suppose we have a time series $Y_1, Y_2, \ldots, Y_n$ and we want to process this in some way for an MLP. The MLP takes arguments $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, where $\boldsymbol{x}_i$ is the inputs and $\boldsymbol{y}_i$ is the outputs. A time series in its raw format is not appropriate. Typically, a univariate time series is converted to an MLP format by using

a sliding window technique. First, a lookback ($b$) and look forward ($f$) period is specified. In essence, we are trying to forecast $f$ units into a future by using the past $b$ observations.

In this thesis, the first window moves from time point 1 to time point $f+b$. The first input vector is $\boldsymbol{x}_1 = (Y_1, Y_2, \dots, Y_b)$ and the first output is $\boldsymbol{y}_1 = (Y_{b+1}, Y_{b+2}, \dots, Y_{b+f})$. We then slide the window forward one unit in time, viz. time point 2 to time point $f+b+1$. Therefore, the next input and output are $\boldsymbol{x}_2 = (Y_2, Y_3, \dots, Y_{b+1})$ and $\boldsymbol{y}_2 = (Y_{b+2}, Y_{b+3}, \dots, Y_{b+f+1})$ respectively. This process is repeated until no data is left to facilitate another slide in the window.

As an example, for more clarification, consider $Y_1, Y_2, \dots, Y_{10}$ with $b = 6$ and $f = 2$. The training data for the MLP will be:

$$\boldsymbol{X}_{train} = \begin{bmatrix} Y_1 & Y_2 & Y_3 & Y_4 & Y_5 & Y_6 \\ Y_2 & Y_3 & Y_4 & Y_5 & Y_6 & Y_7 \\ Y_3 & Y_4 & Y_5 & Y_6 & Y_7 & Y_8 \end{bmatrix}$$

$$\boldsymbol{Y}_{train} = \begin{bmatrix} Y_7 & Y_8 \\ Y_8 & Y_9 \\ Y_9 & Y_{10} \end{bmatrix}.$$

### 3.4.4.5 Time Series Data for RNNs

Converting a univariate time series into an appropriate format for an RNN is a bit more complicated than for an MLP, because of the sequential structure of the inputs. As mentioned before, the input for an RNN is a set of p-dimensional vectors $\mathbb{Z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \dots, \boldsymbol{z}_T\}$. There are many ways to read in a time series for an RNN, but we will only consider one method. As for an MLP, a lookback period ($b$) and a look forward period ($f$) need to be specified, and also the dimension of the sequential vectors ($p$).

Again, we adopt a sliding window approach in this thesis, and our first window was from time point 1 to time point $f+b$. The first output is again $\boldsymbol{y}_1 = (Y_{b+1}, Y_{b+2}, \dots, Y_{b+f})$ but the first input is of the form $\mathbb{Z}_1 = \{\boldsymbol{z}_{11}, \boldsymbol{z}_{12}, \dots, \boldsymbol{z}_{1T}\}$, with $T = {}^{b}/_{p}$ . We divide the input vector $(Y_1, Y_2, \dots, Y_b)$ into $p$ non-overlapping parts and construct our input as

$$\boldsymbol{z}_{1j} = \left(Y_{1+(j-1)p}, Y_{2+(j-1)p}, \dots, Y_{jp}\right)', \tag{3.42}$$

for $j = 1,2, \dots, \frac{b}{p}$. We then repeat this process by sliding the window with one observation at a time until there are no data left.

Usually, the inputs of RNN are represented in the form of a three-dimensional tensor with dimensions representing (sample, time steps, input units). In this thesis, time steps = $\frac{b}{p}$ and input units = $p$ . Consider the previous example in the MLP section, where we choose the input units, $p$ = 2. Our training inputs will be a tensor with dimensions (3,3,2) and are as follows:

$$\boldsymbol{X}_{train}[1,,] = \begin{bmatrix} Y_1 & Y_2 \\ Y_3 & Y_4 \\ Y_5 & Y_6 \end{bmatrix}.$$

$$\boldsymbol{X}_{train}[2,,] = \begin{bmatrix} Y_2 & Y_3 \\ Y_4 & Y_5 \\ Y_6 & Y_7 \end{bmatrix}.$$

$$\boldsymbol{X}_{train}[3,,] = \begin{bmatrix} Y_3 & Y_4 \\ Y_5 & Y_6 \\ Y_7 & Y_8 \end{bmatrix}$$

$$\boldsymbol{Y}_{train} = \begin{bmatrix} Y_7 & Y_8 \\ Y_8 & Y_9 \\ Y_9 & Y_{10} \end{bmatrix}.$$

### 3.4.4.6 Using Validation Data to Avoid Overfitting

One of the major concerns when training a neural network is overfitting. Even a moderately sized neural network can have a substantial number of parameters, which means that it is very easy to produce artificially small training errors.

Several approaches can be used to combat overfitting when training a neural network. Due to the large size of the time series under consideration, we decided to use a validation set to determine an appropriate neural network. The time series is divided into three components; viz. a training set, a validation set and a test set. These components were processed into an appropriate format for neural networks using the procedures described in the previous sections.

The training set was used to produce updates for the weights and biases using batch gradient descent. When an epoch is completed, the new estimates are evaluated using the validation set. This evaluation is in terms of the root mean squared error (we discuss this measure in the following section). The chosen weights and biases are those associated with the epoch with the smallest validation error.

## 3.5 Forecast Performance Metrics

It is essential to compare the performance of the models in terms of prediction accuracy. There is no unique metric to evaluate different models as a universal standard (Korkmaz *et al.*, 2018). It is, therefore, necessary to assess the performance of models using different metrics. Two metrics are used in this thesis: root mean square error (RMSE) and mean absolute percentage error (MAPE).

i. Root mean squared error (RMSE):

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}\,.$$

The RMSE is an extension of MSE which uses the squared difference between the predicted value $\hat{Y}_i$ and actual value $Y_i$. RMSE takes the root of MSE. It is more appropriate to use the RMSE for model comparison since the MSE is more sensitive towards outliers.

ii. Mean Absolute Percentage Error (MAPE):

$$\frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i-\hat{Y}_i}{Y_i}\right| \times 100.$$

The MAPE takes the difference between the predicted value $\hat{Y}_i$ and actual value $Y_i$, scales the error by dividing it with the actual value $Y_i$. This measure of error is expressed as a percentage. According to (Adhikari *et al.*, 2013), this metric is independent of the scale of the data but affected by data transformation. The RMSE and MAPE values should be small to have the best forecast model.

## 3.6 Conclusion

Statistical forecasting techniques used in the past for wind speed forecasting were discussed in this chapter to give a better understanding of the procedures applied in the empirical study reported in Chapter 5. Some factors need to be considered when training a neural network, and these factors were also discussed in this chapter. We also discussed a method of reading in a time series in an appropriate format for MLPs and RNNs. The forecasting techniques discussed in this chapter will be used to analyse the wind speed for the two sites described in the following chapter. These forecasting techniques will be compared using RMSE and MAPE.

# CHAPTER 4

# DATA DESCRIPTION AND EXPLORATION

## 4.1 Data Description

The data used in the thesis was obtained from the WASA website. This thesis involves forecasting wind speed for two sites, Jozini and Memel in South Africa. These two measurement sites were inspected in phase 2 of the WASA project. For these sites, meteorological data is available for the period 2015 to 2019. However, due to a large number of missing values, it was decided to only focus on the data from the beginning of 2016 to the end of 2018 (this corresponds to 157824 data points per site). The time series are averaged wind speeds at 60 m hub height with 10 minutes between measurements. The table below summarises the mast and sites characteristics.

**Table 4.1: Sites Description**

| Sites | Longitude | Latitude | Anemometer height (level 60 m) | Province |
|---|---|---|---|---|
| Jozini | 32.16636 | -27.42605 | 60.84 | KwaZulu-Natal |
| Memel | 29.54348 | -27.88169 | 60.71 | Free State |

The positions of the meteorological masts were determined using GPS receivers, and longitude and latitude are the coordinates for the masts obtained during the site inspection trips. The heights in the table refer to the height of the cup anemometer rotor plane above the top of the terrain surface. The province is the region where the site is located.

## 4.2 Data Transformation

The original data used consists of ten-minute averaged wind speeds measurements for Memel and Jozini sites. The data set from Jozini site have 22 missing values in June 2018. The data set from Memel site have 2 missing values. The missing values in all the data sets are relatively small compared to the total number of data points. First, the decision was made to impute the missing values with the previous most recent known value.

The focus of the thesis is to forecast 1 to 24 hours ahead, in hourly intervals. Therefore, there was a need to pre-process the data into the desired time interval before training the models and forecasting. For each site, the data was converted to hourly averaged data by taking the mean hourly wind speed (taking the mean of 6 consecutive 10-minute averaged wind speeds). After converting the data, each site contains 26305 hourly averaged observations for 2016 to 2018. This means the data for each site was processed to have one observation for each hour, e.g. there are 24 observations for 24 hours.

Following the hourly averaging, the data were divided into a training, testing and validation set. The data points between 2016 and 2017 was used as the training set to obtain the model parameters, half of the data points for 2018 (January 2018 to June 2018) were used as the validation set and the last half (July 2018 to December 2018) as the testing set to compare with the forecasted values. The testing set did not participate in the model training process but was used for performance evaluation.

In this thesis, we make use of traditional time series models and ANNs. Before applying the ANNs, the data were scaled. Neural networks work best when the input data are scaled (Wanjohi, 2018). Scaling makes the training faster and allows for more accurate results. We make use of the min-max normalisation method to scale the input and target data to be in a range of the activation function (Wanjohi, 2018). The scaling parameters were determined based only on the training data and usually requires the minimum and maximum of the training data to be computed. These scaling parameters were then applied to the validation and testing set. The reason for this is to ensure that the range values of the testing data do not affect the model.

Different activation functions were used for modelling. For the sigmoid activation function, the data were scaled to be in the range of 0 and 1 while for the hyperbolic tangent activation function, the data were scaled to be in the range -1 to 1. For the ReLu activation function, the data is already on an appropriate scale and no scaling was done. After prediction, the predicted values were transformed back to the original scale in order to evaluate the model performance.

Normalisation was used when applying neural networks only. There was no need to scale the data when applying traditional time series models. Another note is that prediction using traditional time series models does not require a validation set because the models were selected based on BIC; therefore, a decision was made to train the models using the combination of validation set and training set.

The final step of the data transformation was to convert the time series into a suitable format for the neural networks. This process was discussed in Section 3.4.4.4 for MLPs and Section 3.4.4.5 for RNNs. A decision was made to have a lookback period of 31 days (this corresponds to 744 observations).

Because we focus on short-term wind speed forecasting, our look forward period was 24 hours (24 observations). An overview of the data transformation process is given in Figure 4.1.

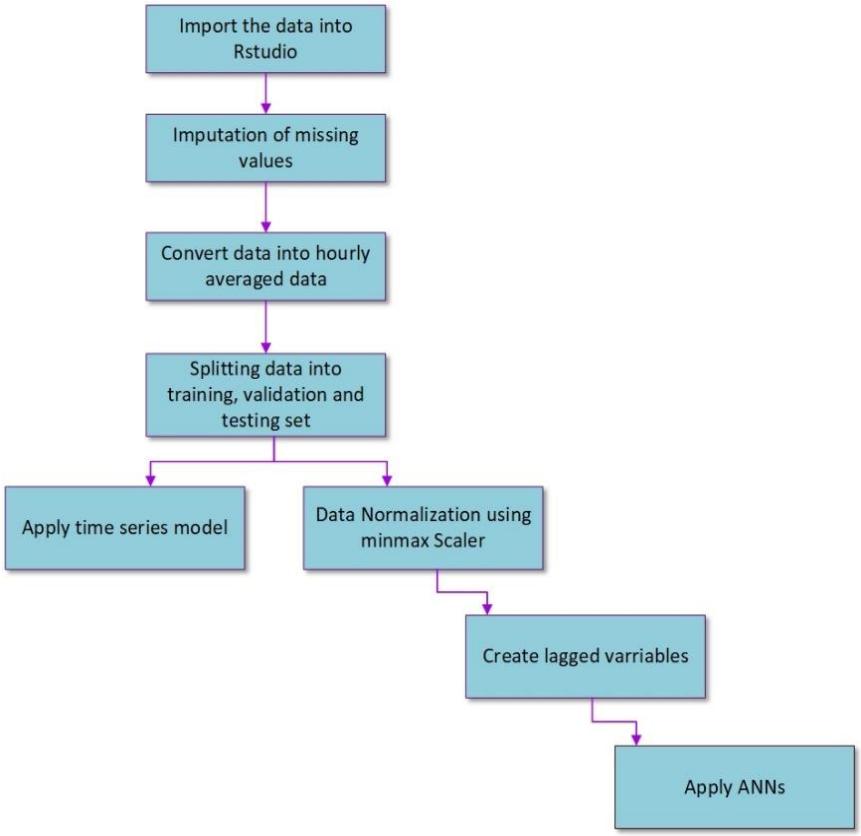



**Fig. 4.1: Flowchart of the data transformation process.**

## 4.3 Software Application and Packages

RStudio software was used for all modelling work, including data transformation and statistical analysis. RStudio was chosen because of the suitable packages and available built-in functions for modelling. Two packages were used, *Keras R* from CRAN and the *forecast* package (Hyndman and Khandakar, 2008).

*Keras R* was used for the fitting and analysis of each site data using MLPs and LSTMs and the *forecast* package in R was used for traditional time series models. The following section gives a brief presentation of the packages used.

### 4.3.1 Introduction to *Keras R*

*Keras R* package provides an R interface to Keras. Keras is a high-level neural network API library written in Python. It was developed and maintained by a Google engineer called Francois Chollet, to enable fast experimentation. Keras was built in such a way that it is user-friendly to make it easy and quick to develop and test neural networks or deep learning models. Another key feature for Keras is the built-in support for RNNs, MLPs and conventional neural networks (Chollet, 2015).

*Keras R* interface uses TensorFlow backend engine by default. Therefore, both Keras library and TensorFlow backend need to be installed in RStudio using the *install_keras ()* function. A useful website for the application of Keras in R can be found at https://keras.rstudio.com.

Before making use of *Keras R* package, it is crucial to understand how the package works. Below we introduce the basics of Keras in five steps:

1. After installation, load Keras library from Comprehensive R Archive Network (CRAN) using the *library (*Keras*)*
2. Build a model by assembling layers using the *model* and *sequential* APIs. This creates a model with a linear stack of layers. You can add as many layers as you want to the model using the pipe operator (%>%). The fully connected layers are created using the *layer_dense ()* function with an *input_shape* argument in the first layer to specify the shape of the input data. The shape of the input data depends on the type of neural network. The input format for the MLP should have a shape (length of a numeric vector), and RNNs should have a shape (number of sequences, dimension of sequence). Note that in each layer, you can specify the number of neurons, the activation function, the weight initialisation algorithm and the weight regularisation. By default, no activation or regularisation is applied, but Glorot uniform initialiser is set as a default to initialise weights.

3. The next step is to configure the model built in step 2 for training using the *compile* method. The *compile* takes three arguments, namely; *loss, optimiser and metrics* to specify the loss function to be minimised, the training algorithm and the measure of accuracy respectively.

4. After compiling the model, train the model by fitting to the training data using the *fit* () method. The data needs to be prepared in a specific format that Keras accept. The input and output variables need to be created for the training, validation and testing set. For LSTM, the input needs to be in a 3-dimensional array of the form (number of observations, number of sequences, dimension of sequence). The *fit* () method takes the following arguments (objects);

the *epochs*, *batch_size* and *validation_data* to specify the number of training cycles, the size of each training batch and validation data to monitor the performance of the model respectively.

5. The final step is to evaluate the model and perform predictions using the *evaluate* () and *predict* () method.

In this thesis, we use the previous 744 observations at a time as the training input. For training MLP, the input was a vector of length 744 (31 days), and the input format for LSTM was a matrix of shape (31 x 24 hours). In other words, the LSTM model looked back 31 days, and at each time step, a sequence with dimension 24 hours was used. A function with a for loop was created to reshape the inputs for LSTM, and it is denoted as *toRNN* in the R code in Appendix A.

### 4.3.2 Introduction to *Forecast* Package in R

The *forecast* package was developed and is maintained by Professor Rob Hyndman. The package is available from CRAN and contains methods and functions for displaying and analysing univariate time series. There are various univariate forecasting methods implemented in the package such as automatic ARIMA models and exponential smoothing. In this thesis, this package was used for automatic ARIMA modelling and shortly, we discuss the implementation of these models in steps to introduce the features in the package:

1. First, load the package into RStudio using *library*(forecast).
2. To implement ARIMA models, the *auto.arima* () function is used. The function automatically conducts a search over a number of candidate models and returns an object of class ARIMA with the lowest BIC or AIC. The *auto.arima* () takes various arguments, and below we define the most important arguments to be specified when applying the different time series models used in the analysis:
    - The first argument, *y*: specifies the univariate time series under study.
    - *stationary*: If it is specified as true, it restricts the *auto.arima* () function to only search over stationary models.
    - *seasonal*: If is true, it limits the *auto.arima* () function to only search over seasonal models.
    - *ic*: this specifies the criterion to be used in the model selection, AIC or BIC
    - *xreg*: A vector or matrix of external regressors. The length of the vector or number of rows in the matrix should be the same as *y*.

3. After model selection, the *forecast ()* function is used to forecast the future values of the time series. The first argument *obj*, specify the time series or the model for which forecasts are needed and the second argument *h* represent the number of periods for forecasting. The function returns information such as the original time series, point forecasts, the forecasting method used, residuals from the fitted model and others. The outputs from the *forecast ()* can be viewed using the *summary* () function.

Note that, the *forecast* package also contains *Arima* () function to fit the ARIMA types models to the time series, but the order of the model must be manually specified in the function. Before fitting time series models, all data were converted to a time series object using *ts* () function with a period specified by an argument *frequency*. The *ts* () function is another feature in the *forecast* package. In this thesis, the *auto.arima* () function was used to implement the 5 time series models; ARIMA models, SARIMA models, SARMA models, ARMA models and regression using Fourier terms with ARMA errors. The *auto.arima* () function returned the model with the lowest BIC values. Applying ARMA models and SARMA models to each site data, stationarity was enforced in the model by specifying *stationary*= TRUE in the *auto.arima* () function.

## 4.4 Data Exploration

Data exploration was performed using the training data only. The training data for the Jozini and Memel sites are displayed in Figure 4.2. The wind speeds at 60m height for all sites are not very stable, and there are sudden peaks which can make wind speed prediction difficult (see Figure 4.2). The mean wind speed for both sites appears to be stable over time, while the variability does not seem to be stable. We also see regularly repeating patterns in both series that seems to repeat every year. These repeating patterns or cycles are easily visible, especially in the Memel time series. The repeating pattern indicates that the time series may have seasonal patterns. The Jozini series tend to exhibit another hidden cycle which is not easily visible in the series. Figure 4.3 and 4.4 show the autocorrelation function (ACF) and partial autocorrelation function (PACF) plot for Memel and Jozini respectively, and both ACF plots show a periodic component. This periodical behaviour is of interest when training models. The periodicities of each site will be identified using a periodogram analysis.

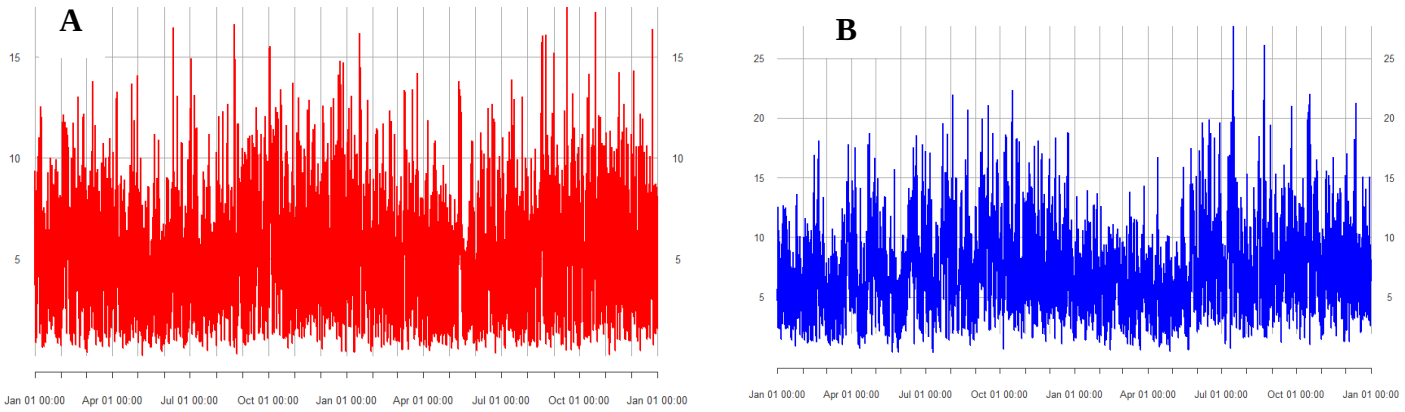



**Fig. 4.2: Hourly averages wind speed (m/s); A = Training data for Jozini site and B = Training data for Memel site**

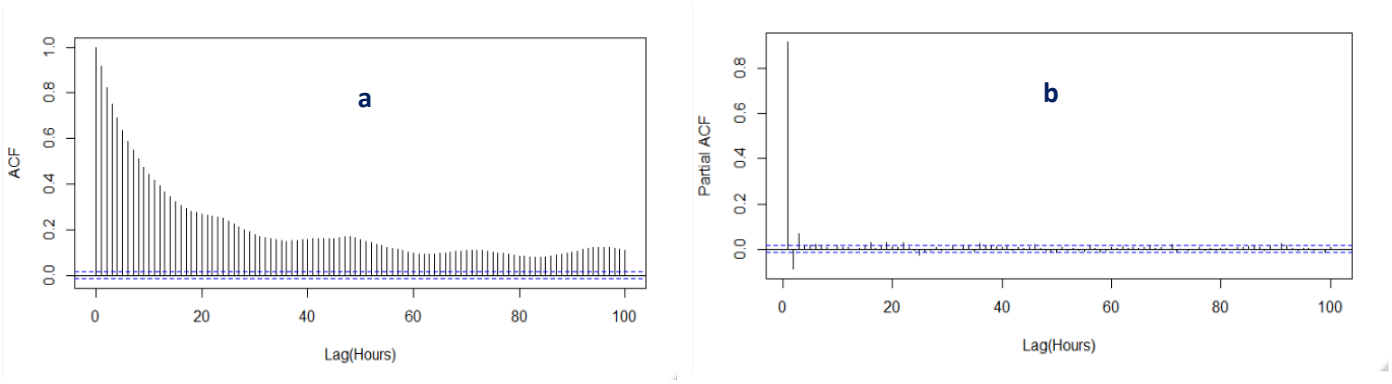



**Fig 4.3: ACF (a) and PACF (b) plot for Memel site**

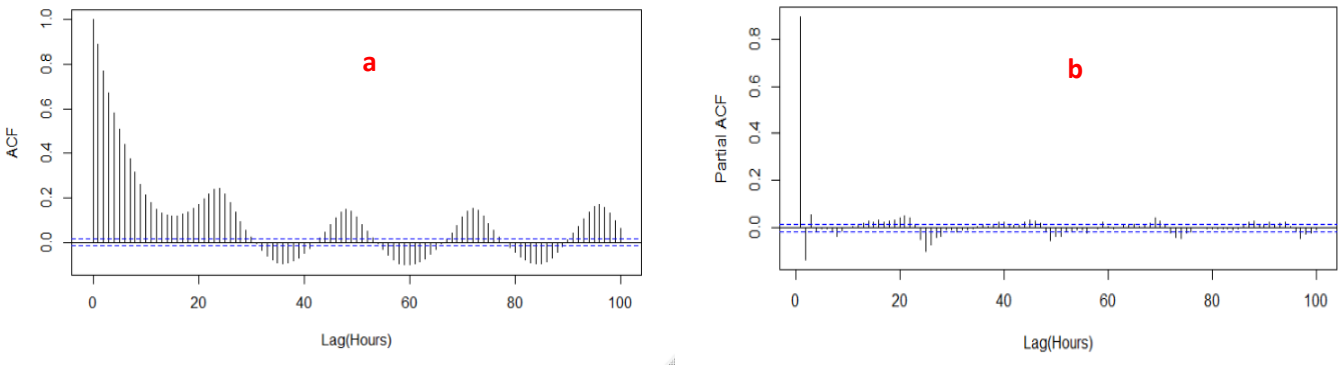



**Fig 4.4: ACF (a) and PACF (b) plot for Jozini site**

Table 4.2 gives the descriptive statistics of the hourly averaged wind speed data for both sites in order to visualise the nature of the data.

**Table 4.2: Statistical Summary of the Wind speed data**

| Site | Min (m/s) | Max (m/s) | Median (m/s) | Standard Deviation (m/s) | Mean (m/s) | Kurtosis (m/s) | Skewness |
|---|---|---|---|---|---|---|---|
| **Jozini** | 0.2159 | 17.4832 | 4.7888 | 2.6933 | 5.1124 | 3.2449 | 0.6752 |
| **Memel** | 0.3514 | 27.6848 | 6.7021 | 3.3989 | 7.2774 | 4.2667 | 0.9592 |

The data for all sites are not normally distributed as indicated by the values of the kurtosis in Table 4.2 (these values are more than 3). This can be confirmed by the histograms in Figure 4.5. The distribution of the data is not close to bell-shaped but skewed. The coefficient of the skewness for all the data are positive, and not approximately zero, which indicates that the distributions of the data are skewed to the right. The histograms show that the data are not normally distributed, but skewed to the right, and something like a Weibull distribution might be more appropriate. The Weibull distribution is typically used to model the distribution of wind speeds (Carrillo *et al.*, 2014).

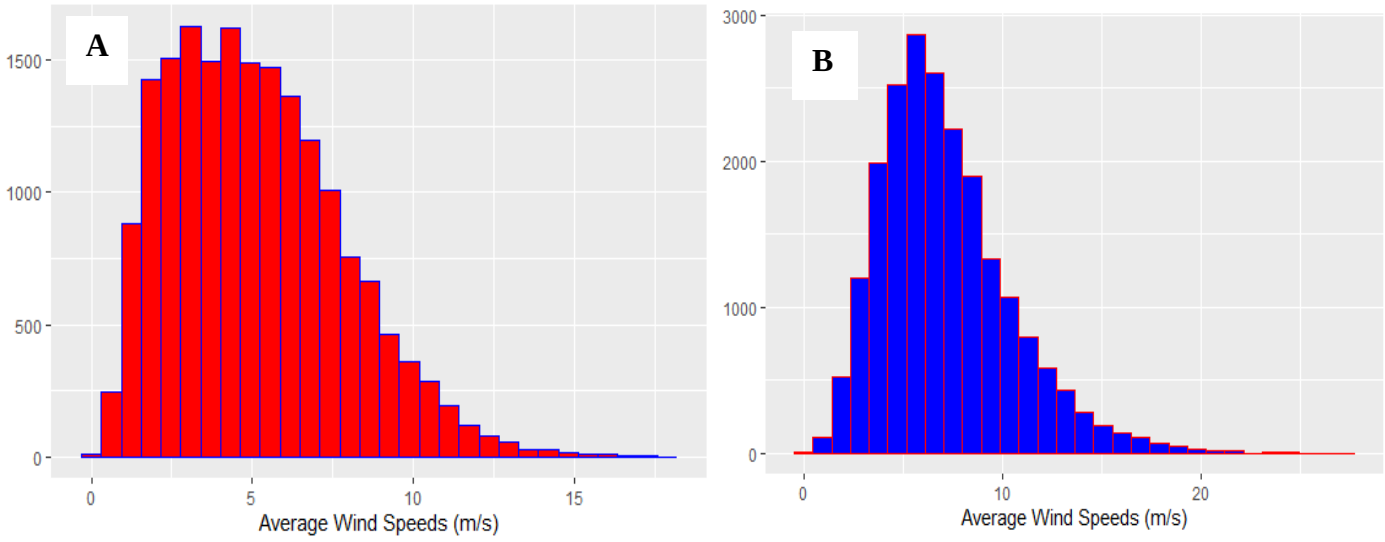



**Fig. 4.5: Histogram of hourly average wind speeds; A = Jozini and B = Memel**

Figure 4.6 shows the boxplots of the time series to give an indication of how the data sets are spread and there seem to be possible outliers in both data sets. The boxplots reconfirm that the distribution of the data for both sites is not normal.

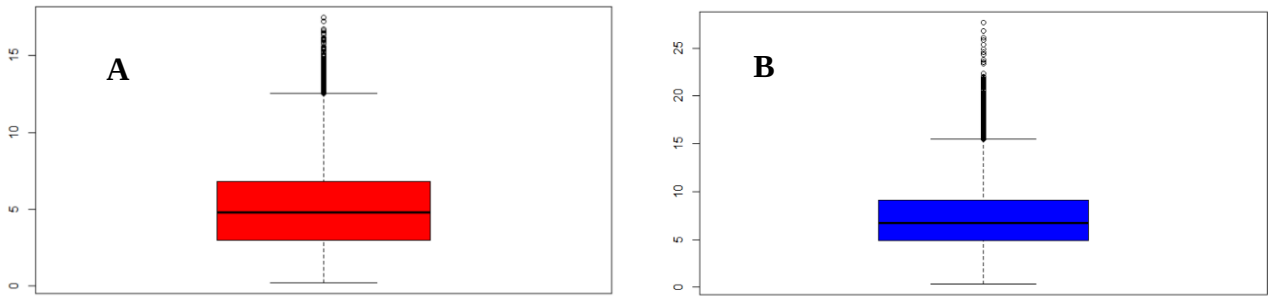



**Fig. 4.6: Boxplots to identify outliers; A = Jozini and B = Memel**

As previously mentioned, Figures 4.2 – 4.4 suggest that the data have seasonal patterns, and it is advisable to determine whether the series to be studied has characteristics of seasonality. This is relevant because seasonal variation components need to be included in the SARIMA models if they do exist. A periodogram was used to identify the dominant periods of the time series. The periodograms for the Jozini and Memel sites are shown in Figure 4.7.

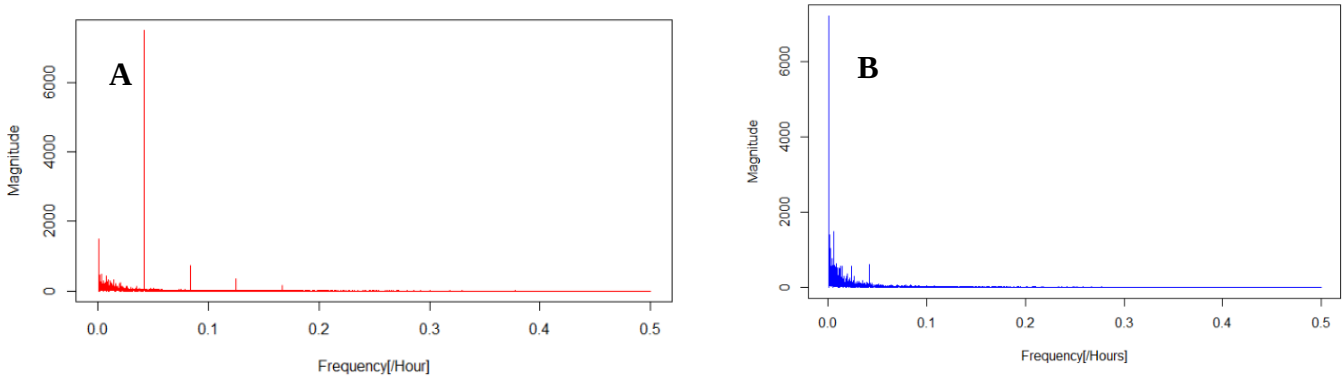



**Fig. 4.7: Periodogram; A = Jozini and B = Memel**

The periodogram shows prominent spikes at the low frequencies for both sites. There are two prominent spikes for Jozini corresponds to a frequency of 0.0416650827 and a frequency of 0.0001140468. These frequencies have a period of 1/0.0416650827 = 24 hours and 8768.3333 hours (yearly). Jozini shows strong daily cycles, which means there is a repeating pattern after 24 hours.

The highest peak value of Memel corresponds to a frequency of 0.0001140468, with a period of 1/0.0001140468 = 8768.3333 hours. The Memel site shows strong yearly cycles.

## 4.5 Conclusion

The data to be used in the experimental analysis was discussed and transformed into a useful form using RStudio. The packages to be used for modelling were discussed to give an idea of how the different forecasting techniques will be implemented in Chapter 5. In this chapter, data exploration was performed on the training data. Here it was shown that the Jozini site exhibits daily seasonality, while the Memel site exhibits annual seasonality. Hence, to model the seasonality in the data, seasonal time series models will be applied to forecast the averaged wind speeds.

# CHAPTER 5

# EMPIRICAL STUDY

We present the implementation of the five traditional time series models, MLPs and LSTMs in the prediction of hourly wind speeds at 60m hub height for Jozini and Memel site. The analysis was performed using RStudio. The code is given in Appendix A.

## 5.1 Traditional Time Series Models

The data was imported into RStudio using the procedure discussed in Chapter 4. The *forecast* package was used for training and forecasting. To begin with forecasting, the *auto.arima* () function from the *forecast* package was used to find the best ARIMA type models with the lowest BIC values.

The seasonal period for Jozini and Memel sites identified was 24 hours and 8768 hours, respectively. A decision was made to set both time series frequencies to 24 hours to fit SARIMA and SARMA models. The reason for using a 24 hour period for Memel is because the strong seasonality is long (yearly), and *auto.arima* () function works best when the frequency of the time series is small. Also, a typical period for wind speeds is 24 hour.

The regression with ARMA errors allows more flexibility for modelling the period. For these models, a daily period was used for the Jozini site, while an annual period was used for the Memel site. The number of Fourier terms for each model was chosen by minimising the BIC values. The *auto.arima*() function was applied to a number of Fourier terms ranging from 1 to 5 and the model with the smallest BIC value was chosen for each site. Table 5.1 Summarises the traditional time series models chosen by *auto.arima* () function.

**Table 5.1: Traditional Time Series Models**

| Model | Jozini Site | Memel Site |
|---|---|---|
| ARIMA | ARIMA(0,1,5) | ARIMA(0,1,5) |
| ARMA | ARIMA(1,0,2) | ARIMA(2,0,2) |
| SARIMA | ARIMA(5,1,0)(2,0,0)[24] | ARIMA(5,1,0)(1,0,0)[24] |
| Regression using Fourier terms with ARMA errors | Regression with ARMA(2,0,2) errors | Regression with ARMA(4,0,2) errors |

The 5 models listed in Table 5.1 was used to predict wind speeds 1 to 24 hours ahead. The results obtained was compared to the persistence forecasts, which acts as a baseline model.

The statistical results for the models were evaluated using RMSE and MAPE to select the model that best predicts the wind speed at all forecast horizon for each site. The RMSE and MAPE values for the five models and persistence forecast are displayed in Figure 5.1 and Figure 5.2, respectively.

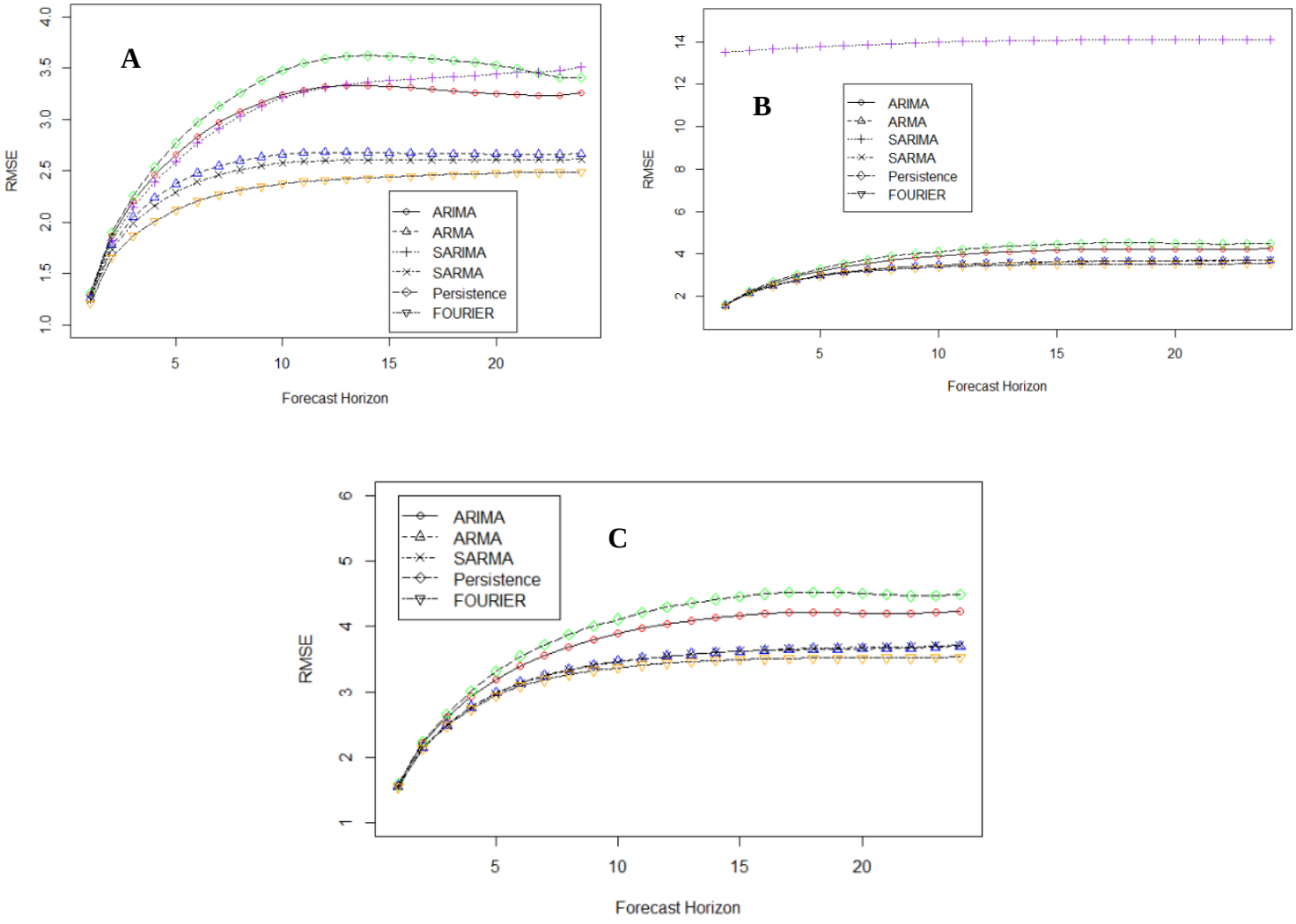



**Fig.5.1: Forecast performance of traditional time series models using RMSE; A = Jozini, B = Memel and C= Memel without SARIMA**

Figures 5.1 and 5.2 shows that all the time series models are effective to predict the hourly wind speed at a 60m hub height. For Jozini, we see that all the models outperform the persistence forecast. For this site, we see that the regression using Fourier terms with ARMA errors, denoted as FOURIER in the Figures outperforms all the other models with respect to both RMSE and MAPE.

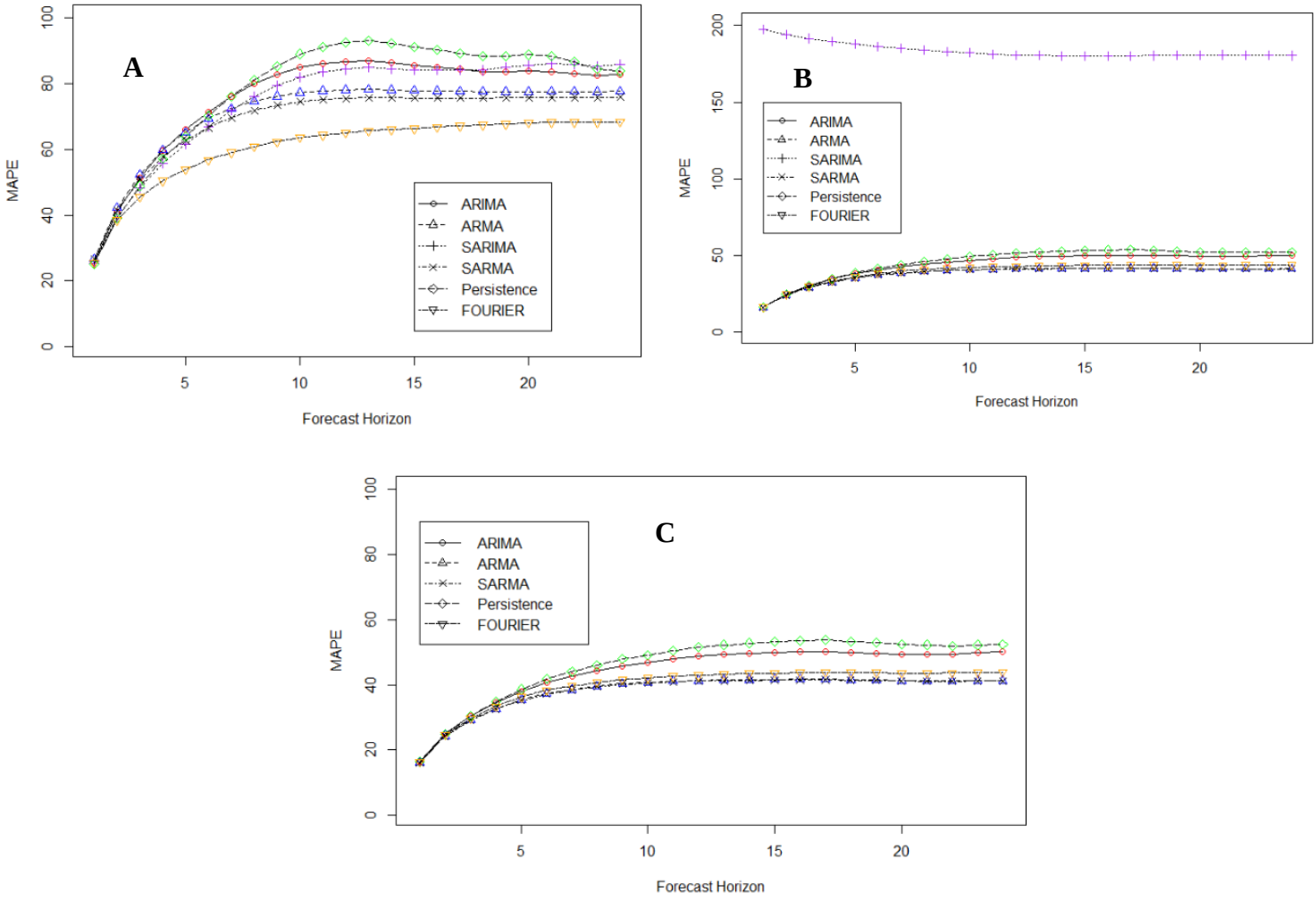



**Fig 5.2: Forecast performance of traditional time series models using MAPE; A = Jozini, B = Memel and C= Memel without SARIMA**

For Memel, the persistence forecasts outperform SARIMA model, and this could be attributed to the fact that the site series might not have daily seasonality (24 hours), which was set when the data was converted to a time series object. It was not clear to see which model performed best because of high values for SARIMA. Figures 5.1(C) and 5.2 (C) shows all models except SARIMA. It can be seen that, the regression using Fourier series with ARMA errors yields the most accurate results with respect to RMSE. The SARMA and ARMA models yield the best results with respect to MAPE. Figure 5.1 and Figure 5.2 also reveal that the performance accuracy for both traditional time series models decreases as the forecast horizon increases.

## 5.2 Fitting Neural Networks and Forecasting

Various MLP and LSTM configurations were constructed and applied to each site, and then forecasting was conducted. Some important factors were considered when creating different configurations; the number of hidden layers, the number of hidden nodes, batch-size (the sub-sequences selected at random from the training data), the activation function, the number of epochs and the learning rate. For both configurations, the mean squared error was used as a loss function, and Adam optimiser was used to minimise the function. The linear activation function was used in the output layer for all configurations.

### 5.2.1 Multilayer Perceptron Neural Networks

Thirteen different MLP configurations were created, and the summary of these configurations is presented in Table 5.2 in Appendix B. For all configurations, lagged values of 744 hours were used as inputs. The statistical results obtained by different configurations were evaluated using RMSE and MAPE to select the configuration that best predicts the wind speed at each forecast horizon for each site. First, the best 5 configurations with the lowest test RMSE and MAPE values for each site were chosen, and these are presented in Table 5.3 and 5.4 below.

**Table 5.3: MLP most 5 accurate configurations for Jozini**

| Model | Activation function | Number of Hidden Layers | Number of Hidden Nodes | Batch Size | Number of Epochs | Learning rate |
|---|---|---|---|---|---|---|
| FNN1-Mod1 | ReLu | 1 | 48 | 1000 | 500 | 0.001 |
| FNN2-Mod1 | Sigmoid | 1 | 48 | 1000 | 500 | 0.001 |
| FNN3-Mod1 | Sigmoid | 1 | 12 | 1000 | 500 | 0.001 |
| FNN6-Mod1 | Sigmoid | 1 | 96 | 500 | 1000 | 0.001 |
| FNN4-Mod1 | Sigmoid | 1 | 24 | 500 | 500 | 0.001 |

**Table 5.4: MLP most 5 accurate configurations for Memel**

| Model | Activation Function | Number of Hidden layer | Number of Hidden nodes | Batch Size | Number of Epochs | Learning Rate |
|---|---|---|---|---|---|---|
| FNN1-Mod1 | ReLu | 1 | 48 | 1000 | 500 | 0.001 |
| FNN2-Mod1 | Sigmoid | 1 | 48 | 1000 | 500 | 0.001 |
| FNN3-Mod1 | Sigmoid | 1 | 12 | 1000 | 500 | 0.001 |
| FNN2-Mod2 | ReLu | 2 | 20 & 8 | 1000 | 800 | 0.001 |
| FNN1-Mod2 | ReLu,ReLu | 2 | 48 & 12 | 1000 | 500 | 0.001 |

Figures 5.3 and 5.4 display the best five MLP configurations based on RMSE and MAPE, respectively.

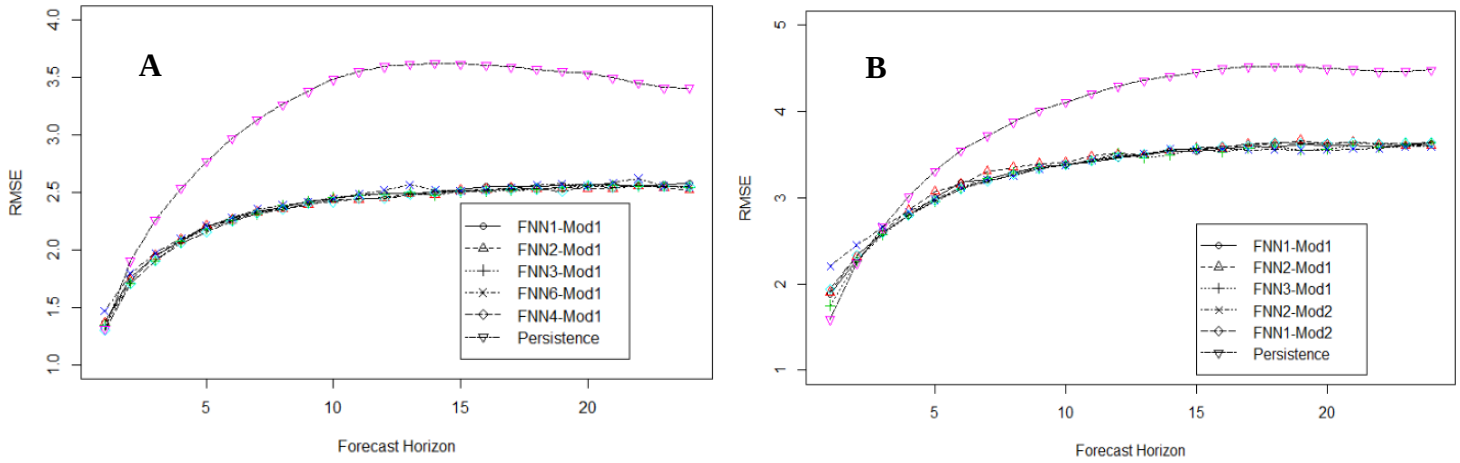



**Fig 5.3: Forecast performance of five MLPs using RMSE; A = Jozini, B = Memel**

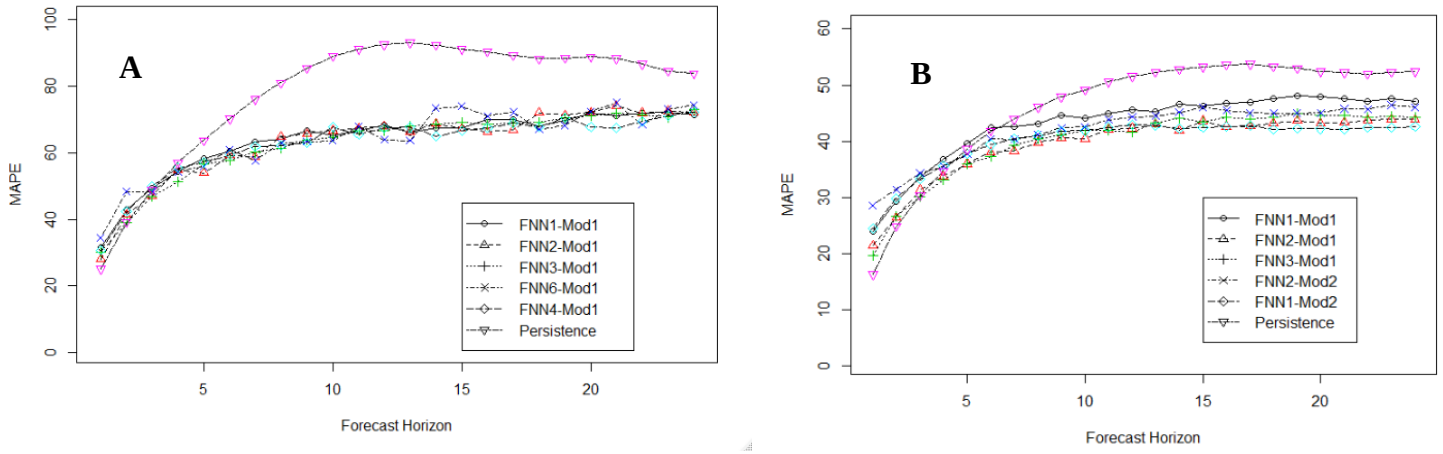



**Fig 5.4: Forecast performance of five MLPs using MAPE; A = Jozini, B = Memel**

Figures 5.3 and 5.4 shows that all five MLP configurations significantly outperform the persistence forecast for all sites. For Jozini, the different configurations yield similar results, and it is not clear, which is the best. The RMSE and MAPE values generated by both configurations are presented in Tables 5.5 and 5.6 in Appendix B, respectively for a closer comparison. Overall, a 1-layer MLP network with 24 nodes and a sigmoid function yields the most accurate results with respect to both RMSE and MAPE. This model is denoted as FNN4-Mod1 in Figures 5.3 and 5.4. For the Memel site,

it is clear that a 2-layer MLP with 20 and 8 hidden nodes in each layer and a sigmoid activation function yields the best result with respect to RMSE (Figure 5.3(B)). The configuration is presented as FNN2-Mod2. Figure 5.4(B) shows that a 1-layer MLP network with 48 nodes and a sigmoid activation function yields the most accurate results with respect to MAPE. The configuration is presented as FNN2-Mod1.

### 5.2.2 LSTM Neural Networks

The LSTMs takes time to run, and because of computational and time constraints, six different configurations were applied to each site. These configurations consist of a 1-layer LSTM followed by a dense layer (s) with different activation functions. The best 5 configurations with the lowest RMSE and MAPE values were chosen, and the summary of the configurations are presented in Table 5.7 below. Figures 5.5 and 5.6 displayed the forecast performance of the 5 configurations with respect to RMSE and MAPE for each site.

**Table 5.7: LSTM most 5 accurate configurations for all sites**

| Model | Activation Function | Number of Hidden Layers | Number of Hidden Nodes | Batch Size | Number of Epochs | Learning Rate |
|---|---|---|---|---|---|---|
| RNN1-Mod2 | Tanh, ReLu | 2 | 72 & 48 | 1000 | 100 | 0.001 |
| RNN2-Mod2 | Sigmoid | 1 | 72 | 1000 | 200 | 0.001 |
| RNN2-Mod3 | Sigmoid, ReLu | 2 | 20 & 12 | 1000 | 500 | 0.001 |
| RNN1-Mod4 | Tanh,ReLu,ReLu | 3 | 72 ,48 &36 | 2000 | 100 | 0.001 |
| RNN1-Mod3 | Tanh, ReLu | 2 | 20 &12 | 1000 | 500 | 0.01 |

Figures 5.5 and 5.6 show that the different LSTM configurations outperform the persistence forecast for all sites. For each site, the different configurations perform similar for all forecast horizons, but some configurations perform slightly better than others. For the Jozini site, a one-layer LSTM configuration with 72 nodes and a sigmoid activation function perform slightly better with respect to RMSE. The configuration is displayed as RNN2-Mod2 in Figure 5.5 (A). A configuration with one-layer LSTM with 72 nodes and additional 2- layer MLP with 48 and 36 nodes in each layer yields the most accurate results with respect to MAPE. The configuration is displayed as RNN1-Mod4 in Figure 5.6 (A).

For the Memel site, a one-layer LSTM configuration with 72 nodes and a sigmoid activation function yields the most accurate results with respect to both RMSE and MAPE. The configuration is displayed as RNN2-Mod2 in Figures 5.5 (B) and 5.6 (B).

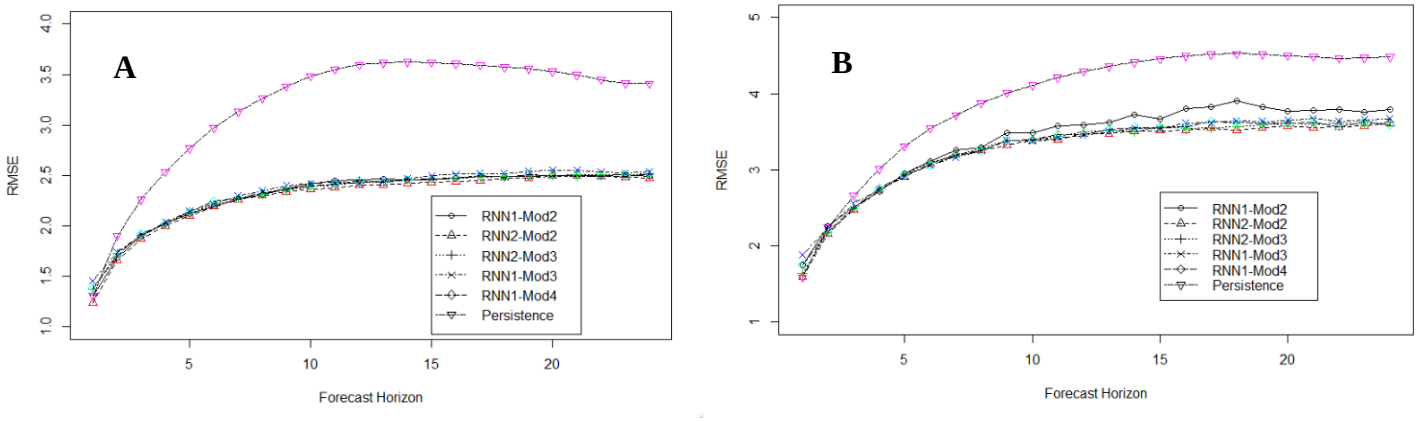



**Fig 5.5: Forecast performance of five LSTM using RMSE; A = Jozini, B = Memel**

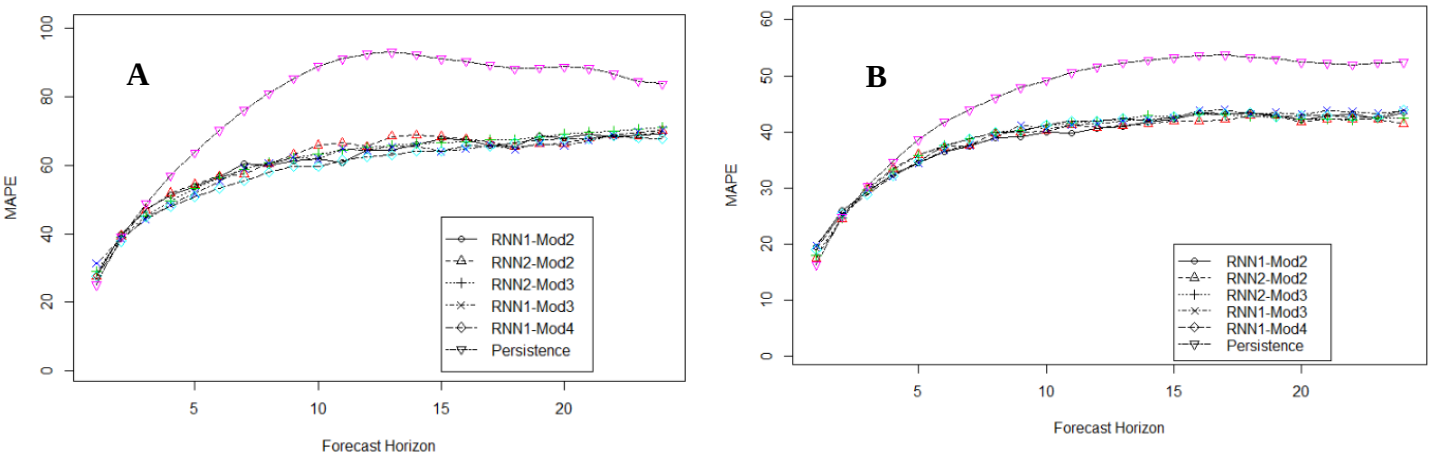



**Fig 5.6: Forecast performance of five LSTM using MAPE; A = Jozini, B = Memel**

## 5.3 Overall Performance of the Techniques

The overall performance of the three techniques (traditional time series models, MLPs and LSTMs) is compared. First, we compare the different forecasting techniques based on the RMSE and lastly on the MAPE.

**RMSE**

Figures 5.7 and 5.8 show the forecast performance of the most accurate models for the three techniques for each site separately using RMSE. For both sites, the performance of the three forecasting techniques is similar for each forecast horizon, with the lowest RMSE when the forecast horizon is short.

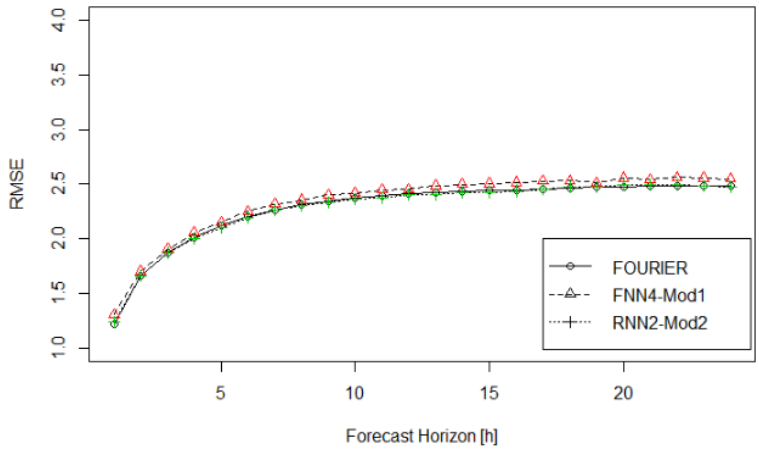



**Fig 5.7: RMSE of all 3 forecasting techniques used for Jozini site**

The results in Figure 5.7 show the most accurate models of the three techniques used to predict the hourly wind speeds of Jozini site for each of the 24-forecasting horizon. Of the three techniques, it can be seen that LSTM and regression using Fourier terms with ARMA errors yield pretty similar results for all forecast horizon. The RMSE values of the three models are summarised and presented in Table 5.8 in Appendix B for a closer comparison. The results indicate that regression using 4 Fourier terms and ARMA (2,0,2) errors predicts the wind speed at 60 m height better than a one-layer LSTM with 72 nodes and a sigmoid activation function.

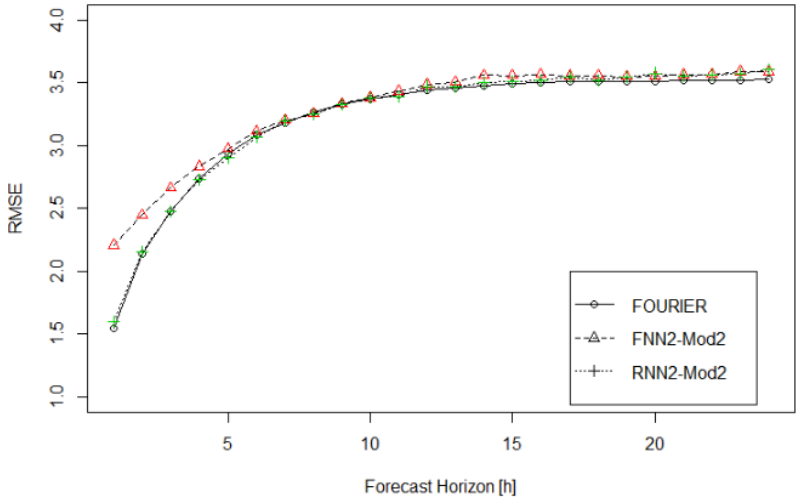



**Fig 5.8: RMSE of all 3 forecasting techniques used for Memel site**

The results in Figure 5.8 shows the most accurate models of the three techniques used to predict the hourly wind speeds of Memel site for each of the 24-forecasting horizon. Of the three techniques, the MLP performs the worst and a one-layer LSTM of 72 nodes with a sigmoid activation function and regression using 1 Fourier term and ARMA (4,0,2) errors yield similar results. The RMSE values of the three models are summarised and presented in Table 5.9 in Appendix B. The RMSE values indicate that regression with 1 Fourier term and ARMA (4,0,2) errors predict the wind speed at 60 m height better than a one-layer LSTM of 72 nodes with a sigmoid activation function.

Overall, the regression using Fourier terms with ARMA errors slightly performed better than LSTM configurations with respect to RMSE for all 24hours forecast horizons for each site.

## MAPE

Figures 5.9 and 5.10 show the forecast performance of the most accurate models for the three techniques based on MAPE for Jozini and Memel, respectively. The performance of the three forecasting techniques is similar, especially for Memel site. The MAPE values generated by the models are summarised and presented in Table 5.10 and 5.11 in Appendix B for more comparison.

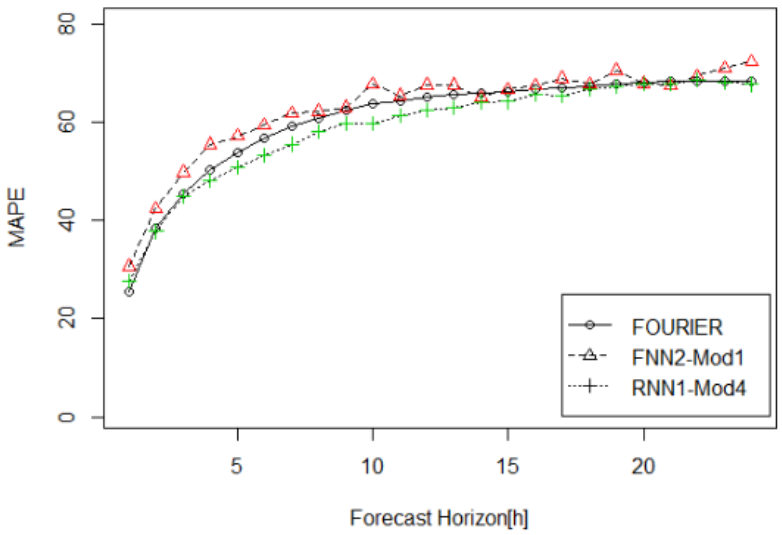



**Fig 5.9: MAPE of all 3 forecasting techniques used for Jozini site**

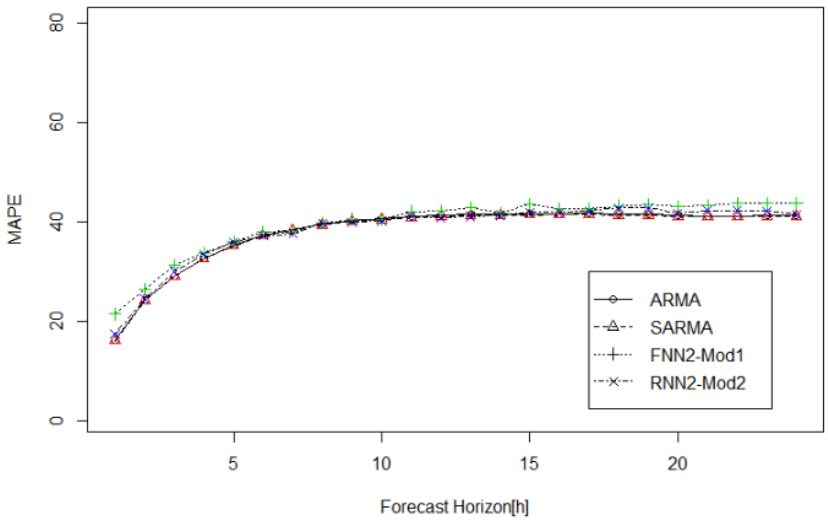



**Fig 5.10: MAPE of all 3 forecasting techniques used for Memel site**

Figure 5.9 shows that of the three techniques, LSTM yields the most accurate results for Jozini. A configuration with one-layer LSTM with 72 nodes and additional 2- layer MLP with 48 and 36 nodes in each layer generated the lowest MAPE values compared to other models. For the Memel site, it can be seen in Figure 5.10 that traditional time series models (SARMA and ARMA) obtained the lowest MAPE values compared to LSTM and MLP configurations. The SARMA and ARMA models yield similar results, and it is not clear in the plot which model performs the best. The MAPE values presented in Table 5.11 indicate that SARMA model slightly outperforms the ARMA model in predicting the wind speed at 60 m height for Memel site.

Overall, the performance accuracy for all techniques decreases as the forecast horizon increases, thereby obtaining more accurate results when forecasting 1 hour ahead. The regression using Fourier terms with ARMA errors performed the best with respect to RMSE for all 24 hours forecast horizons for both Jozini and Memel. Based on MAPE, the LSTM performed the best when forecasting wind speeds for Jozini and traditional time series models (SARMA or ARMA) performed the best for Memel site. Interestingly, MLPs obtained inconsistent forecasts and was outperformed by the traditional time series models with respect to both evaluation metrics.

For both sites, regression using Fourier terms with ARMA errors yields the best results based on RMSE, and we decided to plot the actual values and predicted values of the model. Figures 5.11 and 5.12 show the actual and the predicted values of the model for 1 hour, 6 hours and 24 hours forecast ahead for Jozini and Memel respectively.

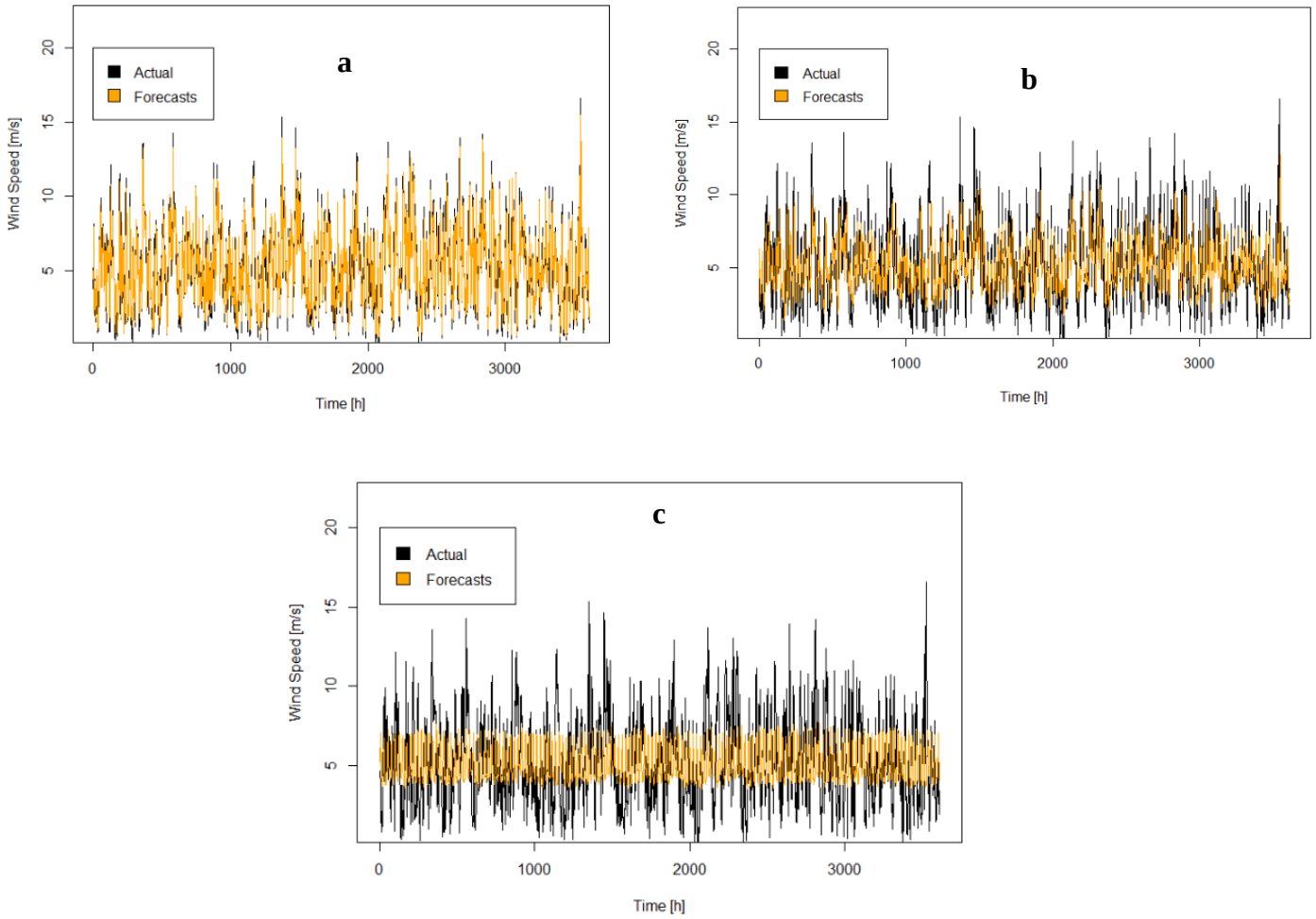
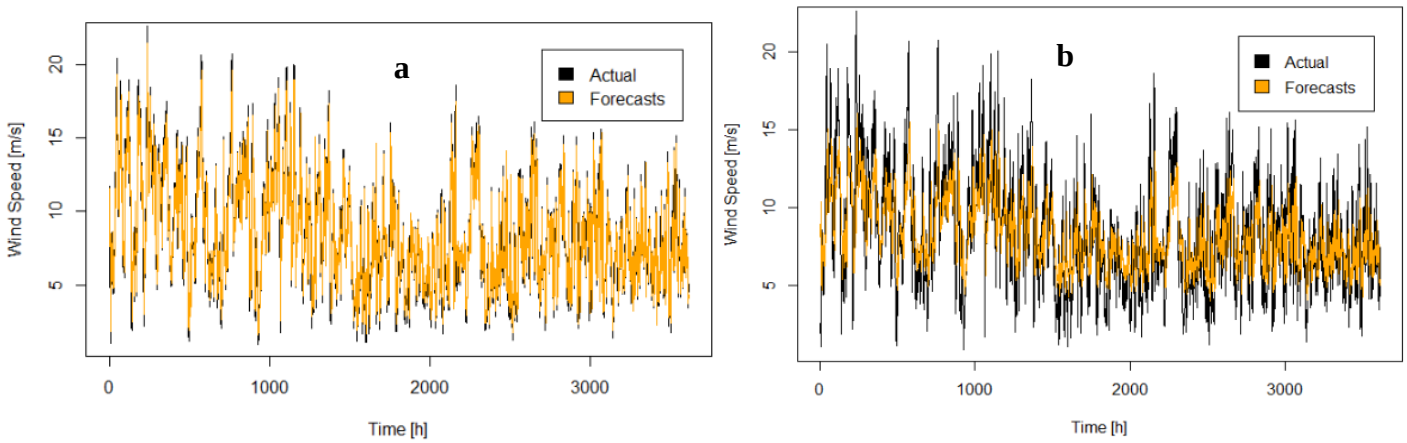



**Fig 5.11: Predicted values and the Actual values for Jozini site. a = 1 hour forecasts, b = 6 hours forecasts and c = 24 hours forecasts.**



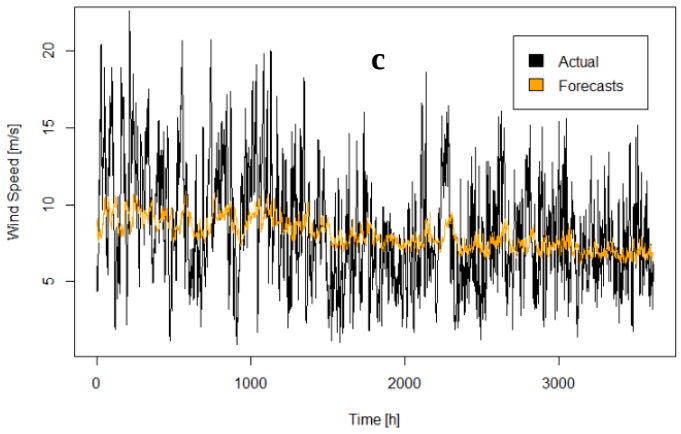



**Fig 5.12: Predicted values and the Actual values for Memel site. a = 1hour forecasts, b = 2hours forecasts and c = 24hours forecasts.**

Figures 5.11 and 5.12 show the output variable of the test set as actual values, *Y_test* in the R code in Appendix A and not the whole test set. The results showed that regression using Fourier terms with ARMA errors was able to predict the overall oscillations of the hourly wind speeds for each site with high accuracy. For each site, it can be seen that the model performs the best when forecasting 1 hour ahead and worse for 24 hours. The 24-hour step forecasts have higher RMSE values. Figures 5.11 (a) and 5.12 (a) show the predictions of 1 hour ahead for Jozini and Memel respectively. It can be seen that the model predicted the peaks in the time series accurately compares to other forecast horizons. The performance accuracy of the model decreases as the forecast horizon increases for all site. The figures revealed that when the forecast horizon is long (24 hours), the model cannot predict the sudden peaks in the time series. Therefore, to accurately predict the wind speeds, short-term forecasting is essential.

# CHAPTER 6

# CONCLUSION AND FURTHER RESEARCH

## 6.1 Main Findings

This thesis has investigated the use of statistical forecasting techniques in the prediction of hourly wind speeds at 60m hub height for two sites, Jozini and Memel. The forecast was 1 to 24 hours ahead. The forecasting techniques used were five traditional time series models, MLPs and LSTMs. In Chapter 2, a review of the literature surrounding forecasting techniques used to predict wind speeds was conducted. There has been a large amount of research into the use of statistical forecasting techniques such as MLPs in the analysis of wind speeds data. The advantage of using neural networks is the flexibility of networks to model complex data compared to the traditional time series models.

In Chapter 3, the key concepts and architectures involving the four techniques were introduced to give a better understanding of the procedure used in the empirical study in Chapter 5. Chapter 4 covered the description and exploration of the data used in the analysis. In Chapter 5, an empirical study comparing the performance of all the methods under consideration was conducted. Five time series models (ARIMA, SARIMA, ARMA, SARMA and regression using Fourier terms with ARMA errors) were applied to each site and forecasting was performed. Various MLPs and LSTMs configurations were created using different number of hidden layers, different number of hidden nodes, different activation functions and learning rates. The lagged values of the wind speed time series were used as inputs to the neural network models. A periodogram analysis was performed to identify the seasonality for each site. These seasonal components were used to construct Fourier terms for use in regression with ARMA errors.

In this thesis, the forecast performance to predict hourly wind speeds at 60 m hub height from 1 to 24 hours ahead obtained from time series models and artificial neural networks were compared to determine which technique perform better for each site. These models were compared to the persistence forecast, and generally, the results showed that persistence forecast was the least accurate model to forecast wind speeds for all forecasting horizons. The results showed that the majority of forecasting techniques considered achieved similar results. This is particularly true for the LSTMs and the regression using Fourier terms with ARMA errors. This suggests that simpler methods are sufficient for modelling the behaviour of wind speeds at these particular sites.

Overall, regression using Fourier terms with ARMA errors performed better based on RMSE for all forecast horizon compared to other techniques. The most accurate for Jozini site is a regression using

4 Fourier terms (corresponding to a period of 24) and an ARMA(2,0,2) process to model the remaining information in the residuals. For Memel, the most accurate model is a regression using 1 Fourier term (to model the annual seasonality) and an ARMA(4,0,2) process. The RMSE values generated by this model were slightly lower compared to other models. The model predictions for 1 hour, 6 hours and 24 hours ahead for each site were plotted, and it showed that the models could predict the oscillations of the wind speeds for each site with high accuracy, particularly for lower forecasting horizons. Forecasting 1 hour ahead generated lower errors compared to other forecast horizons and most of the sudden peaks in the time series were predicted.

## 6.2 Further Research

The use of other features (such as wind direction, air temperature, barometric pressure and relative humidity) has been used in the literature to aid in the forecasting of wind speeds (Botha and Van Der Walt, 2018). In this thesis, we only considered forecasting in a univariate context, and other features were ignored. The models used in this thesis can be extended to include additional features such as temperature, wind speeds measured at different hub heights and wind direction. Incorporation of additional features in the model could result in a more accurate model that can predict the peaks in the data. Certain features (such as wind direction which is measured in degrees) enter the forecasting of future wind speeds in a non-linear way, and this means that non-linear models, such as MLPs and LSTMs, may offer a convenient way of incorporating these features.

One of the advantages of using regression with Fourier terms and ARMA errors is that it is easy to include multiple seasonalities in these models. In this thesis, the Fourier terms were used to model a single seasonality. Since these models performed relatively well in our empirical study, a natural avenue for further research would be to identify and model multiple seasonalities.

In our empirical study, multiple configurations of MLPs and LSTMs were investigated and evaluated. In all of these configurations, the number of lagged variables was kept fixed at 744 (mostly for computational reasons). The effect of varying the number of lagged variables on the performance of the neural networks was not considered and should be investigated in further research.

In this thesis, the focus was placed on traditional time series models and neural networks. Many other models could be applied to the problem of forecasting wind speeds at Jozini and Memel. An alternative to the MLPs would be the use of support vector regression (SVR). SVR is a non-linear model and can also be used to model additional features. Another alternative is the use of autoregressive Markov switching processes. In essence, these are AR models where the AR

coefficients varies according to some unknown hidden state. It should, however, be noted that the non-linear methods considered in this thesis did not offer substantial improvement in testing performance. We are therefore sceptical if SVR and Markov switching processes will offer significant improvement in a univariate context.

We used hourly averaged wind speeds in our empirical study, although the raw data has a 10-minute resolution. This was mostly done for computational reasons. It is possible that this averaging lead to loss of information. A further avenue of research is to apply the models considered to the raw 10-minute averaged wind speeds.

The focus of our empirical study was on the Jozini and Memel sites. There are data available for several other sites in South Africa. It would be interesting to see if the findings of this thesis generalise to other sites in South Africa. We also note that some of these sites (such as Alexander Bay) have more data available. One possible reason for the "poor" behaviour of the neural networks at Jozini and Memel might be that there is not enough data available to estimate the non-linearities appropriately.

Many other approaches might improve forecasting accuracy. In the literature review, statistical forecasting was compared to the physical systems approach. It was also mentioned that hybrid approaches (combining models) have been used in the literature. These approaches were not considered in this thesis. Another option is to investigate the use of ensemble methods. If the best model differs with the forecasting horizon, this would be a sensible approach.

# REFERENCES

Abdul-Rahim, A.S., Noraida, A.W., Ye, C., Shan, C.J. & Lui, S.C.T. (2015) 'Reexamines the emission-growth-energy nexus in Malaysia : Does trade matters ?. *International Conference on Entrepreneurship, Business, and Social Sciences*. August 2015. Yogyakarta, Indonesia.

Adhikari, R. & Agrawal, R.K. 2013. *An Introductory Study on Time Series Modeling and Forecasting.*United Kingdom: LAP Lambert Academic Publishing.

Alsharif, M. H., Younes, M. K. & Kim, J. 2019. Time series ARIMA model for prediction of daily and monthly average global solar radiation: The case study of Seoul, South Korea. South Korea: *Symmetry*,11(2):1–17.

Ayodele, T. R., Olarewaju, R. & Munda, J. L. 2019. Comparison of Different Wind Speed Prediction Models for Wind Power Application. *2019 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa, SAUPEC/RobMech/PRASA* . IEEE. Bloemfrontein, South Africa.

Azad, H. B., Mekhilef, S. & Ganapathy, V. G. 2014. Long-Term Wind Speed Forecasting and General Pattern Recognition using Neural Networks. *IEEE Transaction on Sustainable Energy*, 5(2):546–553.

Barbosa de Alencar, D., De Mattos Affonso, C., Limão de Oliveira, R.C., Moya Rodríguez, J.L., Leite, J.C. & Reston Filho, J.C. 2017. Different Models for Forecasting Wind Power Generation: Case Study. Brazil: *Energies*, 10(12).

Beccali, M., Culotta, S & Lo Brano, V. 2012. Short term wind speed prediction using Multi Layer Perceptron. Available at: https://www.researchgate.net/publication/233751408_Short_term_wind_speed_prediction_using_Multi_Layer_Perceptron.

Botha, N. & Van Der Walt, C. 2018. Forecasting wind speed using support vector regression and feature selection. *2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference, PRASA-RobMech 2017*. IEEE. Bloemfontein: 181–186.

Brewer, M. J., Butler, A. & Cooksley, S. L. 2016. The relative performance of AIC, AICC and BIC in the presence of unobserved heterogeneity. *Methods in Ecology and Evolution*, 7(6): 679–692.

Cadenas, E., Rivera, W., Campos-Amezcua, R. & Heard, C. 2016. Wind speed prediction using a univariate ARIMA model and a multivariate NARX model. *Energies*, 9(2): 1–15.

Cali, U. & Sharma, V. 2019. Short-term wind power forecasting using long-short term memory based recurrent neural network model and variable selection. *International Journal of smart Grid and Clean Energy*, 8(2):103–110.

Carrillo, C. et al. 2014. An approach to determine the Weibull parameters for wind energy analysis: The case of Galicia (Spain). *Energies*, 7(4):676–2700.

Chang, W. 2014. A Literature Review of Wind Forecasting Methods. *Journal of Power and Energy Engineering*, 02(04):161–168.

Chollet, F. 2015. Keras. Available at: https://github.com/fchollet/keras.

Cryer, J.D. & Chan, K. 2008. *Time series Analysis: With Applications in R*. 2nd edition. New York: Springer Science and Business Media.

De Freitas, N. C. A., Silva, M. P. S. and Sakamoto, M. S. 2018. Wind Speed Forecasting : A Review .*International Journal of Engineering Research and Application*, 8(1):4-9.

Department of Energy. 2011. Integrated Resource Plan for Electricity 2010-2030. Final Report dated 25 March 2011. Available at: www.energy.gov.na.

Dontas, G. 2010. Feed-forward and recurrent neural networks. pp. 1–5. [online]. Available at: https://stats.stackexchange.com/a/2218/126819.

Durbin, J. & Koopman, S. J. 2001. *Time Series Analysis by State Space Methods.* Oxford University Press.

Dwivedi, D.K., Sharma, G.R. & Wandre, S.S. 2017. Forecasting Mean Temperature using Sarima Model for Junagadh City of Gujarat. *International Journal of Agricultural Science and Research*, 7(4): 183–194.

Ekpenyong, E. J. 2016. Short-Term Forecasting of Nigeria Inflation Rates Using Seasonal ARIMA Model. *Science Journal of Applied Mathematics and Statistics*, 4(3): 101-107.

Giebel, G., Brownsword, R., Kariniotakis, G., Denhard, M. & Draxl, C. 2011. The State of the Art in Short-Term Prediction of Wind Power A Literature Overview, 2nd Edition [Electronic]. Available at:https://www.researchgate.net/publication/283641726_The_State_of_the_Art_in_Short-Term_Prediction_of_Wind_Power_A_Literature_Overview_2nd_Edition/citation/download.

Glorot, X. & Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research*, 9: 249–256.

Goodfellow, I., Bengio, Y., Courville, A. & Bengio, Y. 2016. *Deep Learning*. MIT Press. Cambridge.

Hadri, K. & Rao, Y. 2009. KPSS test and model misspecifications. *Applied Economics Letters*, 16(12):1187–1190. Avalibale at: https://doi.org/10.1080/13504850701367239

Heaton, J. 2008. *Introduction to Neural networks with java*. 2nd edition. Heaton Research, Inc.

Hewamalage, H., Bergmeir, C. & Bandara, K. 2019. Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions. September [Electronic]. Available at: http://arxiv.org/abs/1909.00590.

Hyndman, R. J. & Athanasopoulos, G. 2013. *Forecasting: Principles and Practice*. *2nd ed.* Melbourne, Australia: OTexts. Available at: https://OTexts.org/fpp2/.

Hyndman, R. J. & Khandakar, Y. 2008. Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software*, 26(3): 1–22. Available at: http://www.jstatsoft.org/article/view/v027i03.

Independent Power Producer Office. 2014. Outlining the Renewable Energy Independent Power Producer Procurement Programme (REIPPPP) Empowerment Imperative. Available at: www.energy.gov.na.

Jamaludin, A. R., Yusof, F., Kane, I.L. & Norrulasikin, S.M. 2016. A comparative study between conventional ARMA and Fourier ARMA in modeling and forecasting wind speed data. *AIP Conference Proceedings*, 1750(1) [Electronic]. Available at: https://doi.org/10.1063/1.4954627 [21 June 2016].

Joubert, C. J. 2017. Geographical location optimisation of wind and solar photovoltaic power capacity in South Africa using mean-variance portfolio theory and time series clustering. Master of Engineering dissertation. Stellenbosch: Stellenbosch University.

Kamruzzaman, J., Begg, R. K. & Sarker, R. A. 2006. *Artificial neural networks in finance and manufacturing*, *Artificial Neural Networks in Finance and Manufacturing*. United States of America: Idea Group Publishing.

Kang, E. 2017. Long short-term memory (LSTM): Concept. Available at: https://medium.com/@kangeugine/long-short-term-memory-lstm-concept-cb3283934359

Kingma, D.P. & Ba, J. 2014. Adam: A Method for Stochastic Optimization. arXiv. 1412.6980. Available at: https://arxiv.org/pdf/1412.6980v8.pdf.

Korkmaz, E., Izgi, E. & Tutun, S. 2018. Forecasting of short-term wind speed at different heights using a comparative forecasting approach. *Turkish Journal of Electrical Engineering and Computer Sciences*, 26(5):2553–2569.

Lakshmi, B.D. & Sujatha, K. 2016. Artificial neural networks for wind speed prediction. *International Journal of Control Theory and Applications*, 9(4):179–185.

Lawan, S. M., Abidin, W.A.W.Z., Chai,W.Y., Baharun,A. & Masri, T. 2014. Different Models of Wind Speed Prediction; A Comprehensive Review. *International Journal of Scientific & Engineering Research*, 5(1):1760–1768. Available at: http://www.ijser.org/researchpaper/Different-Models-of-Wind-Speed-Prediction-A-Comprehensive-Review.pdf.

Lennard, C., Hahmann, A. N., Badger, J., Mortensen, N. G. & Argent, B. 2015. Development of a Numerical Wind Atlas for South Africa. *Energy Procedia*, 76:128–137.

Leung.K. & Chew.L. 2019. Learning Paradigms in neural Networks. *Medium, The Startup Blog* [web log post]. Available at :https://medium.com/swlh/learning-paradigms-in-neural-networks-30854975aa8d.

Lipton, Z. C., Berkowitz, J. & Elkan, C. 2015. A Critical Review of Recurrent Neural Networks for Sequence Learning [Electronic]. Available at: http://arxiv.org/abs/1506.00019.

López, E., Valle, C., Allende, H., Gill, E. & Madsen, H. 2018. Wind power forecasting based on Echo State Networks and Long Short-Term Memory. *Energies*, 11(3): 1–22.

Milligan, M., Schwartz, M.N. & Wan, Y. 2004. Statistical Wind Power Forecasting for U.S. Wind Farms. Paper delivered at the 17th Conference on Probability and Statistics in the Atmospheric Sciences, 11-15 January 2004, Washington. Available at: http://www.nrel.gov/docs/fy04osti/35087.

Moon, T. & Liu, E. 2016. Using Feedforward and Recurrent Neural Networks to Predict a Blogger's Age. Available at: https://www.semanticscholar.org/paper/Using-Feedforward-and-Recurrent-Neural-Networks-to-Moon/a88a4d1a9b773e88d087a1d6735b36bcfcc51d60.

Mortensen, N. G., Hansen, J. C. and Kelly, M. C., Prinsloo, E., Mabille, E. & Szewczuk, S. 2014. *Wind Atlas for South Africa ( WASA ) Western Cape and parts of Northern and Eastern Cape Station and Site Description Report*'.

Mushtaq, R. 2011. Augmented Dickey Fuller Test. Available at: http://dx.doi.org/10.2139/ssrn.1911068.

Nguyen, M. 2018. Illustrated guide to LSTM's and GRU's: A step by step explanation. *Towards data science.* 24 September 2018. Available at: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

Nwankpa, C., Ijomah, W., Gachagan, A. & Marshall, S. 2018.'Activation Functions: Comparison of trends in Practice and Research for Deep Learning. pp. 1–20. Available at: http://arxiv.org/abs/1811.03378.

Olaofe, Z. O. 2013. Wind Energy Generation and Forecasts: A case Study of Darling and Vredenburg Sites. Unpublished master's thesis. Cape Town: University of Cape Town.

Otto, A. 2015. *The Wind Atlas for South Africa Project*. South Africa: South African National Energy Development Institute. Available at: http://www.wasaproject.info/docs/WASABooklet.pdf.

Pekel, E. and Kara, S. S. 2017. A Comprehensive Review for Artifical Neural Network Application To Public Transportation. *Sigma Journal of Engineering and Natural Sciences*, 35(1):157–179.

Petneházi, Gábor. 2018. Recurrent Neural Networks for Time Series Forecasting. Available: https://www.researchgate.net/publication/330102696_Recurrent_Neural_Networks_for_Time_Series_Forecasting.

Prinsloo, E., Mabille, E., Haasbroek, S., Mortensen, N. G. & Hansen, J. C. 2017. *Wind Atlas for South Africa ( WASA ) Station and Site Description Report.*

Reddi, S. J., Kale, S. & Kumar, S. 2018. On the Convergence of Adam and Beyond. International Conference on Learning Representations. Available at: https://openreview.net/pdf?id=ryQu7f-RZ.

Rotela Junior, P., Salomon, F. L. R. and de Oliveira Pamplona, E. 2014. ARIMA: An Applied Time Series Forecasting Model for the Bovespa Stock Index. *Scientific Research*: *Applied Mathematics*, 05(21):3383–3391 [Electronic]. Available at: http://dx.doi.org/10.4236/am.2014.521315.

Sharma, R. & Singh, D. 2018. A Review of Wind Power and Wind Speed Forecasting. *Journal of Engineering Research and Application*, 8(7): 1–09. Available at: www.ijera.com.

Soman, S. S., Zareipour, H., Malik, O.P. & Mandal, P. 2010. A review of wind power and wind speed forecasting methods with different time horizons. *North American Power Symposium 2010, NAPS 2010*, (June 2014).

Sørensen, P.E., Litong-Palima, A., Hahmann, A.N., Heunis, S., Ntusi, M. & Hanse, J.C. 2017.Wind power variability and power system reserves in South Africa. *Journal of Energy in Southern Africa*, 29(1):59-71

Van Der Walt, C. M. and Botha, N. 2017. A comparison of regression algorithms for wind speed forecasting at Alexander Bay. *2016 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference, PRASA-RobMech 2016*. *IEEE*. Stellenbosch, South Africa.

Wanjohi, R. 2018. Time series Forecasting using LSTM in R. LinkedIn [web log post]. Available at: http://rwanjohi.rbind.io/2018/04/05/time-series-forecasting-using-lstm-in-r/.

WASA .2019. *Wind Atlas for South Africa (WASA) download site*, viewed 13 May 2019. Available at: http://wasadata.csir.co.za/wasa1/WASAData.

What are recurrent neural networks? An ultimate guide for nebies!. 2019, July 24. *Data Flair Blog* [web log post].Available: https://data-flair.training/blogs/recurrent-neural-networks/.

Wright, M. A. and Grab, S. W. 2017. Wind speed characteristics and implications for: Wind power generation: Cape regions, South Africa. *South African Journal of Science*, 113(7–8): 35–43.

Wu, Y. K. and Hong, J. S. 2007. A literature review of wind forecasting technology in the world. *IEEE Lausanne Power Tech 2007*, pp. 504–509.

Zadeh, M. A. & Mahmoudi, J. 2017. Discrimination between Earthquakes and Explosion Using MLP and RBF Neural Networks. *Biostatistics and Biometrics Open Access Journal*, 2(4).

Zhang, G. P. (2012) 'Neural networks for time-series forecasting', *Handbook of Natural Computing*, 1–4, pp. 461–477. doi: 10.1007/978-3-540-92910-9_14.

Zolna, K., Dao, P.B., Staszewski, W.J. & Barszcz, T. 2015. Nonlinear cointegration approach for condition monitoring of wind turbines. *Mathematical Problems in Engineering*. Available: http://dx.doi.org/10.1155/2015/978156.

Zucatelli, P. J., Nascimento, E.G.S., Aylas, G.Y.R., Souza, N.B.P., Kitagawa, Y.K.L., Santos, A.A.B., Arce, A.M.G. & Moreira, D.M. 2019. Short-term wind speed forecasting in Uruguay using computational intelligence. *Heliyon*. Elsevier Ltd, 5(5).

# APPENDIX A

**Rstudio Code**

**# Packages used**

```
setwd("D:\\Data")
library(xts)
library(dplyr)
library("readxl")
library(ggplot2)
library(forecast)
library(tseries)
library(keras)
```

### **preparing the data for the sites.**

######**Read in the wind speed csv file**##################################################

```
#i is the site number under analysis
#Memel =14 and Jozini =13
i=14
name <- paste("site_",i,"_data.xlsx",sep = "")
site1.obs <- read_excel(name,col_names = TRUE)
#################################################################################################
#Original data set contains 157 824 observations in 10 minutes interval
#The data are for 2016,2017 and 2018
```

**#convert the data to hourly averages**

```
site1.obs <- site1.obs[,-2]
hourly.aver <- aggregate(site1.obs[,-1],list(hour=cut(site1.obs$date_time,breaks = "hour")),mean)
#########################################################################
# There are 26305 hourly observations
```

**#split the data into training, validation and  test set**

```
N = nrow(hourly.aver)
#The training set consists of 17545 observations for 2016 and 2017
train_data = hourly.aver[1:17545, 2]
# The first 4380 observations from 2018 is for validation set
validation_data = hourly.aver[(17545+1):21925,2 ]
#the second 4380 observations from 2018 is for testing.
```

```
test_data  = hourly.aver[(21925+1):N,  2]
#checking for missing values
sum(is.na(validation_data))
sum(is.na(train_data))
sum(is.na(test_data))
##############################################################################
#the validation set contains some missing values ( Jozini).
# the training set contains some missing values ( Memel).
#a simple imputation method was used, where we impute with the most recent known value
# imputation function
##############################################################################
impute.fun <- function (x)
{
  while(sum(is.na(x)) > 0)          {
    ind <- which(is.na(x))
    x[ind] <- x[ind - 1]
  }
  return(x)
}
##############################################################################
###Impute the validation set for Jozini
#validation_data<- impute.fun(validation_data)
###Impute the training set for Memel
train_data <- impute.fun(train_data)
##############################################################################
#the next step is to convert the time series into a suitable format for neural networks.
##############################################################################
lag.data <- function (z,Lb=744,Lf=24)
{
  K <- (length(z)-Lb-Lf+1)
   xmat <- matrix(0,nrow=K,ncol=Lb)
  ymat <- matrix(0,nrow=K,ncol=Lf)
```

```
  for(i in 1:K)    {
    xmat[i,] <- z[(i):(i+ Lb - 1)]
    ymat[i,] <- z[(i+Lb):(i + Lb + Lf -1)]
  }
 return(list("x"=xmat,"y"=ymat))
}
#############################################################################
#Specify a maximum look back period. Looking back a
#maximum of 31*24 = 744 hours.
#############################################################################
obj.train <- lag.data(train_data,Lb=744)
obj.val <- lag.data(validation_data,Lb=744)
obj.test <- lag.data(test_data,Lb=744)
X_train <- obj.train$x
Y_train <- obj.train$y
X_val <- obj.val$x
Y_val <- obj.val$y
X_test <- obj.test$x
Y_test <- obj.test$y
# All our neural network models was trained using these objects
#############################################################################
#Training the traditional time series models.
#note that we are using the auto.arima function, which selects a model based on the lowest BIC value
#############################################################################
sarima_mod1 <- auto.arima(c(train_data,validation_data),ic="bic")
sarima_mod2 <- auto.arima(c(train_data,validation_data),ic="bic",stationary=TRUE)
ys <- c(train_data,validation_data)
ys <- ts(ys,frequency=24)
sarima_mod3 <- auto.arima(ys,ic="bic")
sarima_mod4 <- auto.arima(ys,ic="bic",stationary=TRUE)
#############################################################################
#we use the following function to evaluate the models
#############################################################################
```

```
predict.sarima <- function (mod)
{
  preds <- matrix(0,nrow(X_test),24)
    for(i in 1:nrow(X_test))          {
    obj <- Arima(X_test[i,],model=mod)
    preds[i,] <- as.vector(forecast(obj,h=24)$mean)
    cat(i,"\n")
  }
  return(preds)
}
```

```
################################################################################
```

**#generate predictions on the test set**

```
#the function terminates when i = 3613
################################################################################
sarima_mod1_pred <- predict.sarima(sarima_mod1)
sarima_mod2_pred <- predict.sarima(sarima_mod2)
sarima_mod3_pred <- predict.sarima(sarima_mod3)
sarima_mod4_pred <- predict.sarima(sarima_mod4)
```

```
################################################################################
```

**#evaluate the quality of the predictions**

**#accuracy measure is RMSE**

```
################################################################################
sarima_mod1_per <- apply((sarima_mod1_pred - Y_test)^2,2,function(x) {sqrt(mean(x))})
sarima_mod2_per <- apply((sarima_mod2_pred - Y_test)^2,2,function(x) {sqrt(mean(x))})
sarima_mod3_per <- apply((sarima_mod3_pred - Y_test)^2,2,function(x) {sqrt(mean(x))})
sarima_mod4_per <- apply((sarima_mod4_pred - Y_test)^2,2,function(x) {sqrt(mean(x))})
```

```
################################################################################
```

**#accuracy measure is MAPE**

```
################################################################################
sarima_mod1_permape <- apply(abs(((Y_test - sarima_mod1_pred)/Y_test)),2,function(x) {(mean(x))*100})
sarima_mod2_permape <- apply(abs(((Y_test - sarima_mod2_pred)/Y_test)),2,function(x) {(mean(x))*100})
sarima_mod3_permape <- apply(abs(((Y_test - sarima_mod3_pred)/Y_test)),2,function(x) {(mean(x))*100})
sarima_mod4_permape <- apply(abs(((Y_test - sarima_mod4_pred)/Y_test)),2,function(x) {(mean(x))*100})
```

```
##############################################################################
# fit a persistence model
#observation to forecast the next 24.
##############################################################################
persistence_pred <- X_test[,rep(ncol(X_test),24)]
#RMSE
persistence_per <- apply((persistence_pred - Y_test)^2,2,function(x) {sqrt(mean(x))})
#MAPE
persistence_permape <- apply(abs(((Y_test - persistence_pred)/Y_test)),2,function(x) {(mean(x))*100})
##############################################################################
#Fitting regression using Fourier terms with ARMA errors
#############For Memel###################
ys <- ts(c(train_data,validation_data),frequency=24*365)
ys.total <- ts(c(train_data,validation_data,test_data),frequency=24*365)

#obtain the fourier terms for both objects based on the lowest BIC values
# For the ys time series (training data)
bestfit.ys <- list(bic=Inf)
for(i in 1:5)
{
  fit.ys <- auto.arima(ys, xreg=fourier(ys, K=i), seasonal=FALSE,stationary = TRUE)
  if(fit.ys$bic < bestfit.ys$bic)
    bestfit.ys <- fit.ys
  else break;
  print(i)
}
######################
#For the whole time series
bestfit.ys.total <- list(bic=Inf)
for(i in 1:5)
{
  fit.ys.total <- auto.arima(ys.total, xreg=fourier(ys.total, K=i), seasonal=FALSE,stationary = TRUE)
  if(fit.ys.total$bic < bestfit.ys.total$bic)
```

```
   bestfit.ys.total <- fit.ys.total
  else break;
  print(i)
}
#############################################################################
fourier.ys <- fourier(ys,K=1) # Memel
fourier.ys.total <- fourier(ys.total,K=1) # Memel
#############################################################################
#next we train our model using auto.arima()
fourier_mod <- auto.arima(ys,seasonal=FALSE,stationary=TRUE,xreg=fourier.ys)
#############################################################################
#the next step is to find the fourier components for the test set
#############################################################################
fourier_test <- fourier.ys.total[21926:26305,]
fourier.data <- function (z,fourier,Lb=744,Lf=24)
{
  K <- (length(z)-Lb-Lf+1)
  #our output is a list.
  out <- vector(K,mode="list")
  for(i in 1:K)    {
    #each element of the list contains three components.
    #(i) past 744 observations.
    #(ii) xreg for the past 744 observations
    #(iii) xreg for the next 24 observations to be used for forecasting
    out[[i]]$series <- z[(i):(i+ Lb - 1)]
    out[[i]]$fourier_inp <- fourier[((i):(i+ Lb - 1)),]
    out[[i]]$fourier_for <- fourier[((i+ Lb):(i + Lb + Lf - 1)),]
  }
  return(out)
}
#############################################################################
#Prepare the test data for the fourier model
X_test_fourier <- fourier.data(test_data,fourier_test)
```

```
predict.fourier <- function (mod)
{
  preds <- matrix(0,nrow(X_test),24)
   for(i in 1:length(X_test_fourier))         {
    ll <- X_test_fourier[[i]]
    obj <- Arima(ll$series,model=mod,xreg=ll$fourier_inp)
    preds[i,] <- as.vector(forecast(obj,h=24,xreg=ll$fourier_for)$mean)
    cat(i,"\n")
  }
 return(preds)
}
fourier_preds <- predict.fourier(fourier_mod)
#RMSE
fourier_per <- apply((fourier_preds - Y_test)^2,2,function(x) {sqrt(mean(x))})
#MAPE
fourier_permape <- apply(abs(((Y_test - fourier_preds)/Y_test)),2,function(x) {(mean(x))*100})
##########################################################################################
Function to plot the RMSE from the traditional time series models ##################
fig2 <- function ()
{
  ylim <- c(1,15)
  plot(1:24,sarima_mod1_per,type="l",ylim=ylim,xlab="Forecast Horizon",ylab="RMSE",main="Forecast
Performance of Traditional Time Series Models")
  points(1:24,sarima_mod1_per,col="red")
   lines(1:24,sarima_mod2_per,lty=2)
  points(1:24,sarima_mod2_per,pch=2,col="blue")
   lines(1:24,sarima_mod3_per,lty=3)
  points(1:24,sarima_mod3_per,pch=3,col="purple")
  lines(1:24,sarima_mod4_per,lty=4)
  points(1:24,sarima_mod4_per,pch=4,col="black")
   lines(1:24,persistence_per,lty=5)
  points(1:24,persistence_per,pch=5,col="green")
   lines(1:24,fourier_per,lty=6)
  points(1:24,fourier_per,pch=6,col="orange")
```

```
legend(x=6,y=12,legend=c("ARIMA","ARMA","SARIMA","SARMA","Persistence","FOURIER"),lty=1:6,
pch=1:6)

}

################################################################################
```

**##The following function rescale the input data for neural networks**

***#The min and max values of the training data are the scaler***

```
scale_data = function(train,validation, test, feature_range = c(0, 1)) {

  x = train

  fr_min = feature_range[1]

  fr_max = feature_range[2]

  std_train = ((x - min(x)) / (max(x) - min(x)))

  std_test  = ((test - min(x)) / (max(x) - min(x)))

  std_val  = ((validation - min(x)) / (max(x) - min(x)))

   scaled_train = std_train *(fr_max -fr_min) + fr_min

  scaled_test = std_test *(fr_max -fr_min) + fr_min

  scaled_val = std_val *(fr_max -fr_min) + fr_min

  return( list(scaled_train = as.vector(scaled_train),scaled_val = as.vector(scaled_val) ,scaled_test =
as.vector(scaled_test) ,scaler= c(min =min(x), max = max(x))) )

}

Scaled <- scale_data(train =train_data,validation =  validation_data,test =  test_data, feature_range = c(0, 1))

scaled_train <- Scaled$scaled_train

scaled_test <- Scaled$scaled_test

scaled_val <- Scaled$scaled_val

################################################################################
```

**#Fitting neural Network**

```
#first we randomly rearrange the training data to avoid it getting stuck in loops.

#################################################################

library(keras)

set.seed(5)

ind <- sample(1:nrow(X_train),nrow(X_train),replace=FALSE)

X_train <- X_train[ind,]

Y_train <- Y_train[ind,]

#this model fits a MLP with a single hidden layer

#with 12 nodes. The model looks back 31 days (744 hours).
```

```
model <- keras_model_sequential()
model %>%
  layer_dense(units = 12, activation = 'sigmoid',input_shape=744) %>%
  layer_dense(units = 24, activation = 'linear')

#lr stands for learning rate
model %>% compile(
  loss="mse",
  optimizer = optimizer_adam(lr=0.001)
)
#this step tells R to store the model with the smallest validation error
cp_callback <- callback_model_checkpoint(filepath="fnn3_mod1.h5",save_best_only=TRUE)

history <- model %>% fit(
  X_train, Y_train, batch_size = 1000, epochs = 500
  , validation_data=list(X_val,Y_val)
  ,callbacks=list(cp_callback))

#load the model with the smallest validation error and predict
mod_best <- load_model_hdf5("fnn2_mod1.h5")
mod_best %>% summary()
preds <- predict(mod_best,X_test)
######################################################################
#The following code revert the predicted values to the original scale
scaler = Scaled$scaler
invert_scaling = function(scaled, scaler, feature_range = c(0, 1)){
  min = scaler[1]
  max = scaler[2]
  t = length(scaled)
  mins = feature_range[1]
  maxs = feature_range[2]
  inverted_dfs = numeric(t)
   for( i in 1:t){
```

```
   X = (scaled[i]- mins)/(maxs - mins)
   rawValues = X *(max - min) + min
   inverted_dfs[i] <- rawValues
  }
 return(inverted_dfs)
}
preds1<-invert_scaling(preds, scaler,  c(0, 1))
#############################################################
#Evaluating the model in the original scale
obj.test <- lag.data(test_data,Lb=744)
Y_test <- obj.test$y
#RMSE
fnn2_mod1_per <- apply((preds1 - Y_test)^2,2,function(x) {sqrt(mean(x))})
#MAPE
fnn2_mod1_permape <- apply(abs(((Y_test - preds1)/Y_test)),2,function(x) {(mean(x))*100})
#############################################################
#function for models comparison based on MAPE
#########################################################################################
fig2_MAPE <- function ()
{
 ylim <- c(1,100)
 plot(1:24,sarima_mod1_permape,type="l",ylim=ylim,xlab="Forecast
Horizon",ylab="MAPE",main="Forecast Performance of Traditional Time Series Models")
points(1:24,sarima_mod1_permape,col="red")
 lines(1:24,sarima_mod2_permape,lty=2)
 points(1:24,sarima_mod2_permape,pch=2,col="blue")
  lines(1:24,sarima_mod3_permape,lty=3)
 points(1:24,sarima_mod3_permape,pch=3,col="purple")
 lines(1:24,sarima_mod4_permape,lty=4)
 points(1:24,sarima_mod4_permape,pch=4,col="black")
 lines(1:24,persistence_permape,lty=5)
 points(1:24,persistence_permape,pch=5,col="green")
 lines(1:24,fourier_permape,lty=6)
 points(1:24,fourier_permape,pch=6,col="orange")
```

```
legend(x=15,y=50,legend=c("ARIMA","ARMA","SARIMA","SARMA","Persistence","FOURIER"),lty=1:
6,pch=1:6)

}
```

**# training rnns.**

```
#RNNs require the inputs to be of a slightly different form, i.e.

#a 3-dimensional array indicating c(observation,number_of_sequences,dimension_of_sequences)

###############################################################################################

library(keras)

set.seed(5)

toRNN <- function (X)

{

  N <- nrow(X)

  out <- array(dim=c(N,31,24))

   for(i in 1:N)    {

    out[i,,] <- matrix(X[i,],nrow=31,ncol=24,byrow=TRUE)

  }

   return(out)

}

###############################################################################################
##########
```

***#convert the different inputs***

```
#note that the outputs remain unchanged

###############################################################################################
##########

X_train_rnn <- toRNN(X_train)

X_val_rnn <- toRNN(X_val)

X_test_rnn <- toRNN(X_test)

###############################################################################################
##########

#the above code states that we go back 744 observations in time (31 days)

#this means we have 31 sequences each of dimension 24 as inputs.

#####################################################################

#the lstm cell generates hidden states of dimension 72

#the final hidden state is passed to a MLP with one hidden layer having 12 nodes.

#the model looks back 31 days (744 hours).
```

```
#the input contains 31 sequences of dimension 24.
#the first sequence is the average windspeeds of 31 days ago
#the second sequence of 30 days ago
#and so on...

model <- keras_model_sequential()
model %>%
  layer_lstm(units = 20, activation = 'tanh',input_shape=c(31,24)) %>%
  layer_dense(units = 12, activation = 'relu') %>%
  layer_dense(units = 24, activation = 'linear')

model %>% compile(
  loss="mse",
  optimizer = optimizer_adam(lr=0.01)
)
#this step tells R to store the model with the smallest validation error
cp_callback <- callback_model_checkpoint(filepath="Jrnn1_mod3.h5",save_best_only=TRUE)
history <- model %>% fit(
  X_train_rnn, Y_train, batch_size = 1000, epochs = 500
  , validation_data=list(X_val_rnn,Y_val)
  ,callbacks=list(cp_callback))

#load the model with the smallest validation error and predict
mod_best <- load_model_hdf5("Jrnn1_mod3.h5")
mod_best %>% summary()
preds <- predict(mod_best,X_test_rnn)
```

---

```
########Plotting the ACF and PACF for the time series
#######Memel
acf.Memel <- plot(acf(train_data,lag.max = 100, main="ACF for Memel"),xlab = "Lag(Hours)")
Pacf.Memel <- plot(pacf(train_data,lag.max = 100, main="PACF for Memel"),xlab = "Lag(Hours)")

#####Jozini
acf.Jozini <- plot(acf(train_data,lag.max = 100, main="ACF for Jozini"),xlab = "Lag(Hours)")
```

```
Pacf.Jozini <- plot(pacf(train_data,lag.max = 100, main="PACF for Jozini"),xlab = "Lag(Hours)")

# plotting sites time series
#to make it an xts object as a time-based object
#Plot for the training data
Jozini.ts <- xts(train_data[,-1],as.POSIXct(train_data$hour,tz="UTC", format="%Y-%m-%d %H:%M:%S"))
plot(Jozini.ts,col = "red")
#Plot for 2 days
plot(Jozini.ts[1:48,],col = "red")

#Function to compute the frequency of the time series
#read in the time series for one site at a time
N1 = length(train_data)
x <- train_data
FF = abs(fft(x)/sqrt(N1))^2
P = FF[1:(N1/2)+1] # Only need the first (n/2)+1 values of the FFT result excluding DC-offset.
f = (1:(N1/2))/N1 # this creates harmonic frequencies from 0 to .5 in steps of 1/N.

#Data frame with frequency and spec ordered from largest to smallest
new.df <- data.frame(x = f, y = P,period=1/f)
new <- new.df[order(new.df$y,decreasing = TRUE),]
new[1:3,]

#####Plotting the periodogram

plot(f,P,type = "l", xlab = "Frequency[/Hour]",ylab = "Magnitude",col="blue") #Memel
plot(f,P,type = "l", xlab = "Frequency[/Hour]",ylab = "Magnitude",col="red") #Jozini
```

# APPENDIX B

**Table 5.2: The summary of thirteen MLP configurations**

| Configuration | Activation Function | Number of Hidden Layer | Number of Hidden Nodes | Batch-Size | Number of Epochs | Learning Rate |
|---|---|---|---|---|---|---|
| FNN1-Mod1 | ReLu | 1 | 48 | 1000 | 500 | 0.001 |
| 2 | ReLU | 1 | 48 | 500 | 500 | 0.01 |
| FNN2-Mod1 | Sigmoid | 1 | 48 | 1000 | 500 | 0.001 |
| 4 | ReLu, ReLu | 2 | 48,24 | 1000 | 500 | 0.001 |
| FNN1-Mod2 | ReLu,ReLu | 2 | 48,12 | 1000 | 500 | 0.01 |
| FNN3-Mod1 | Sigmoid | 1 | 12 | 1000 | 500 | 0.001 |
| FNN4-Mod1 | Sigmoid | 1 | 24 | 500 | 500 | 0.001 |
| FNN6-Mod1 | Sigmoid | 1 | 96 | 500 | 1000 | 0.001 |
| 9 | ReLu, ReLu | 2 | 96,48 | 1000 | 200 | 0.001 |
| 10 | ReLu | 1 | 10 | 1000 | 200 | 0.001 |
| 11 | Sigmoid | 1 | 10 | 1000 | 800 | 0.001 |
| FNN2-Mod2 | ReLu | 2 | 20,8 | 1000 | 800 | 0.001 |

**Table 5.5: Summary of root mean square errors (RMSE) of MLPs for Jozini**

| Forecast Horizon | FNN1-Mod1 | FNN2-Mod1 | FNN3-Mod1 | FNN6-Mod1 | FNN4-Mod1 |
|---|---|---|---|---|---|
| 1 | 1.370296 | 1.362108 | 1.339887 | 1.466029 | 1.303320 |
| 2 | 1.755291 | 1.731936 | 1.708837 | 1.793926 | 1.701920 |
| 3 | 1.947529 | 1.945950 | 1.906359 | 1.969613 | 1.904274 |
| 4 | 2.078656 | 2.084415 | 2.072320 | 2.096582 | 2.056550 |
| 5 | 2.196721 | 2.209748 | 2.181771 | 2.209820 | 2.150985 |
| 6 | 2.273850 | 2.257783 | 2.258228 | 2.276008 | 2.248816 |
| 7 | 2.331589 | 2.327703 | 2.311835 | 2.354898 | 2.317978 |
| 8 | 2.368521 | 2.361936 | 2.384887 | 2.387674 | 2.354939 |
| 9 | 2.422622 | 2.389861 | 2.417034 | 2.416263 | 2.401306 |
| 10 | 2.447780 | 2.439545 | 2.453392 | 2.443244 | 2.417227 |
| 11 | 2.472166 | 2.434808 | 2.470987 | 2.481393 | 2.447353 |
| 12 | 2.488428 | 2.454381 | 2.479956 | 2.519960 | 2.448884 |
| 13 | 2.488784 | 2.484815 | 2.496049 | 2.566227 | 2.484129 |
| 14 | 2.502752 | 2.484118 | 2.475917 | 2.519343 | 2.498442 |
| 15 | 2.529305 | 2.515571 | 2.503846 | 2.508776 | 2.505500 |
| 16 | 2.551392 | 2.527187 | 2.506290 | 2.523355 | 2.513870 |
| 22 | 2.551684 | 2.535906 | 2.513458 | 2.536919 | 2.527777 |
| 17 | 2.553671 | 2.531160 | 2.528633 | 2.562347 | 2.529389 |
| 18 | 2.553808 | 2.541014 | 2.539111 | 2.575414 | 2.511298 |
| 19 | 2.562675 | 2.535121 | 2.560170 | 2.552855 | 2.556202 |
| 21 | 2.564706 | 2.531644 | 2.557824 | 2.583770 | 2.543087 |
| 23 | 2.571907 | 2.555520 | 2.551562 | 2.619982 | 2.561820 |
| 20 | 2.571998 | 2.541579 | 2.552542 | 2.550152 | 2.554227 |
| 24 | 2.579304 | 2.519716 | 2.544285 | 2.550500 | 2.547930 |

**Table 5.6: Summary of mean absolute percentage errors (MAPE) of MLPs for Jozini**

| Forecast Horizon | FNN1-Mod1 | FNN2-Mod1 | FNN3-Mod1 | FNN6-Mod1 | FNN4-Mod1 |
|---|---|---|---|---|---|
| 1 | 31.54254 | 28.00878 | 30.19055 | 34.51693 | 30.64350 |
| 2 | 42.89560 | 41.40901 | 39.00886 | 48.29336 | 42.46749 |
| 3 | 49.25738 | 47.07170 | 46.90056 | 48.35108 | 49.84768 |
| 4 | 54.34001 | 54.88972 | 51.28380 | 54.29837 | 55.48089 |
| 5 | 58.29604 | 53.92148 | 56.54325 | 55.93838 | 57.16661 |
| 6 | 60.44563 | 59.21092 | 57.68792 | 60.97547 | 59.44985 |
| 7 | 63.35046 | 58.74051 | 60.11086 | 57.71821 | 61.81360 |
| 8 | 64.12264 | 64.80255 | 61.25492 | 62.91201 | 62.16911 |
| 9 | 66.75247 | 65.71223 | 63.81475 | 63.39423 | 62.98428 |
| 10 | 65.26653 | 66.71926 | 64.83206 | 63.69953 | 67.73441 |
| 11 | 66.54190 | 67.35694 | 66.75124 | 68.01270 | 65.47686 |
| 12 | 68.26619 | 67.98723 | 66.67649 | 63.90347 | 67.58261 |
| 13 | 66.18966 | 66.44888 | 68.09999 | 63.70960 | 67.64969 |
| 14 | 67.59137 | 68.70834 | 68.58497 | 73.38469 | 65.06997 |
| 15 | 67.50228 | 67.41682 | 69.10272 | 74.01764 | 66.66414 |
| 16 | 70.11458 | 66.29925 | 68.47212 | 70.99466 | 67.49273 |
| 17 | 70.10977 | 66.87328 | 68.87903 | 72.21875 | 69.00616 |
| 18 | 67.93732 | 72.07616 | 69.13258 | 67.03870 | 67.78941 |
| 19 | 69.71263 | 71.32224 | 70.43837 | 68.25316 | 70.61292 |
| 20 | 71.88853 | 72.17657 | 70.99796 | 72.45818 | 67.86275 |
| 21 | 71.05943 | 74.14644 | 71.83273 | 75.03485 | 67.49856 |
| 22 | 71.88945 | 71.98043 | 71.25475 | 68.44995 | 69.39139 |
| 23 | 72.16893 | 72.55320 | 70.70561 | 73.11365 | 71.05378 |
| 24 | 71.41388 | 72.25235 | 73.06222 | 74.37197 | 72.31352 |

**Table 5.8: Summary of root mean square errors (RMSE) for Jozini**

| Forecast Horizon [Hour] | Regression with Fourier Terms and ARMA errors | Multilayer Perceptron NN | LSTM Recurrent NN |
|---|---|---|---|
| 1 | 1.213021 | 1.362108 | 1.303320 |
| 2 | 1.656811 | 1.731936 | 1.701920 |
| 3 | 1.868571 | 1.945950 | 1.904274 |
| 4 | 2.009565 | 2.084415 | 2.056550 |
| 5 | 2.118290 | 2.209748 | 2.150985 |
| 6 | 2.205937 | 2.257783 | 2.248816 |
| 7 | 2.266165 | 2.327703 | 2.317978 |
| 8 | 2.309766 | 2.361936 | 2.354939 |
| 9 | 2.345613 | 2.389861 | 2.401306 |
| 10 | 2.376939 | 2.439545 | 2.417227 |
| 11 | 2.396876 | 2.434808 | 2.447353 |
| 12 | 2.411204 | 2.454381 | 2.448884 |
| 13 | 2.420864 | 2.484815 | 2.484129 |
| 14 | 2.430265 | 2.484118 | 2.498442 |
| 15 | 2.439478 | 2.515571 | 2.505500 |

| | | | |
|---|---|---|---|
| 16 | 2.448188 | 2.527187 | 2.513870 |
| 17 | 2.456982 | 2.535906 | 2.527777 |
| 18 | 2.464888 | 2.531160 | 2.529389 |
| 19 | 2.472220 | 2.541014 | 2.511298 |
| 20 | 2.478837 | 2.535121 | 2.556202 |
| 21 | 2.483458 | 2.531644 | 2.543087 |
| 22 | 2.485512 | 2.555520 | 2.561820 |
| 23 | 2.486328 | 2.541579 | 2.554227 |
| 24 | 2.488128 | 2.519716 | 2.547930 |

**Table 5.9: Summary of root mean square errors (RMSE) for Memel**

| Forecast Horizon [Hour] | Regression with Fourier Terms and ARMA errors | Multilayer Perceptron NN | LSTM Recurrent NN |
|---|---|---|---|
| 1 | 1.548436 | 1.798351 | 1.595884 |
| 2 | 2.135867 | 2.242004 | 2.155479 |
| 3 | 2.475798 | 2.549235 | 2.476316 |
| 4 | 2.734818 | 2.742086 | 2.730466 |
| 5 | 2.939262 | 2.930246 | 2.902876 |
| 6 | 3.088013 | 3.084947 | 3.072685 |
| 7 | 3.184435 | 3.219381 | 3.202562 |
| 8 | 3.265429 | 3.286624 | 3.253108 |
| 9 | 3.328123 | 3.348049 | 3.330670 |
| 10 | 3.369481 | 3.359573 | 3.385391 |
| 11 | 3.410098 | 3.421557 | 3.393345 |
| 12 | 3.441119 | 3.442369 | 3.473430 |
| 13 | 3.463239 | 3.572983 | 3.470330 |
| 14 | 3.480985 | 3.495321 | 3.507214 |
| 15 | 3.494036 | 3.501156 | 3.513419 |
| 16 | 3.505440 | 3.562810 | 3.519423 |
| 17 | 3.511986 | 3.519785 | 3.547166 |
| 18 | 3.515152 | 3.543162 | 3.521231 |
| 19 | 3.516022 | 3.530154 | 3.548701 |
| 20 | 3.515679 | 3.545272 | 3.577732 |
| 21 | 3.517887 | 3.554597 | 3.556766 |
| 22 | 3.518786 | 3.553533 | 3.566377 |
| 23 | 3.525255 | 3.565752 | 3.572938 |
| 24 | 3.532350 | 3.580831 | 3.610466 |

**Table 5.10: Summary of mean absolute percentage errors (MAPE) for Jozini**

| Forecast Horizon [Hour] | Regression with Fourier Terms and ARMA errors | Multilayer Perceptron | LSTM Recurrent |
|---|---|---|---|
| 4 | 50.42605 | 55.48089 | 48.09997 |
| 5 | 53.89857 | 57.16661 | 50.88854 |
| 6 | 56.78376 | 59.44985 | 53.22796 |

| | | | |
|---|---|---|---|
| 7 | 59.10231 | 61.81360 | 55.55286 |
| 8 | 60.85877 | 62.16911 | 58.04883 |
| 9 | 62.37556 | 62.98428 | 59.73294 |
| 10 | 63.66740 | 67.73441 | 59.70921 |
| 11 | 64.43688 | 65.47686 | 61.48080 |
| 12 | 65.05240 | 67.58261 | 62.48946 |
| 13 | 65.68542 | 67.64969 | 63.10082 |
| 14 | 66.03267 | 65.06997 | 64.19429 |
| 15 | 66.32901 | 66.66414 | 64.22565 |
| 16 | 66.83141 | 67.49273 | 65.74843 |
| 17 | 67.15791 | 69.00616 | 65.48791 |
| 18 | 67.41226 | 67.78941 | 66.75118 |
| 19 | 67.75349 | 70.61292 | 67.35481 |
| 20 | 68.10669 | 67.86275 | 67.83689 |
| 21 | 68.29348 | 67.49856 | 67.76280 |
| 22 | 68.32344 | 69.39139 | 68.58083 |
| 23 | 68.29628 | 71.05378 | 68.10315 |
| 24 | 68.36009 | 72.31352 | 67.73531 |

**Table 5.11: Summary of mean absolute percentage errors (MAPE) for Memel**

| Forecast Horizon [Hour] | ARMA | SARMA | Multilayer Perceptron | LSTM Recurrent |
|---|---|---|---|---|
| 1 | 16.11186 | 16.10194 | 21.45556 | 17.45851 |
| 2 | 24.29087 | 24.23347 | 26.46928 | 24.47542 |
| 3 | 29.17250 | 29.08848 | 31.29988 | 30.00887 |
| 4 | 32.70056 | 32.59519 | 33.74725 | 33.51905 |
| 5 | 35.41090 | 35.21735 | 36.01648 | 35.99633 |
| 6 | 37.39883 | 37.20731 | 38.07194 | 37.24824 |
| 7 | 38.54994 | 38.35718 | 38.21869 | 37.66903 |
| 8 | 39.56527 | 39.42045 | 39.77784 | 39.71901 |
| 9 | 40.35929 | 40.21239 | 40.58335 | 39.95861 |
| 10 | 40.67221 | 40.52737 | 40.48163 | 40.19662 |
| 11 | 41.09821 | 40.92690 | 42.14552 | 41.20806 |
| 12 | 41.21127 | 41.13761 | 42.29863 | 40.82851 |
| 13 | 41.23402 | 41.26949 | 43.00706 | 41.15627 |
| 14 | 41.26191 | 41.39321 | 41.87715 | 41.38847 |
| 15 | 41.29698 | 41.43811 | 43.63464 | 41.94245 |
| 16 | 41.30594 | 41.51243 | 42.64127 | 41.96687 |
| 17 | 41.34545 | 41.54091 | 42.68905 | 42.17901 |
| 18 | 41.48363 | 41.39262 | 43.24234 | 42.85020 |
| 19 | 41.50207 | 41.32689 | 43.67699 | 42.87538 |
| 20 | 41.57055 | 41.14995 | 43.23381 | 41.73199 |
| 21 | 41.58559 | 41.10368 | 43.33875 | 42.31340 |
| 22 | 41.63341 | 41.08324 | 43.82844 | 42.29099 |
| 23 | 41.69652 | 41.14799 | 43.88061 | 42.23015 |
| 24 | 41.71965 | 41.18893 | 43.90151 | 41.48879 |