**CCT College Dublin**

**Assessment Cover Page**

|  |  |
| --- | --- |
| **Module Title:** | Advanced Data Analytics  Big Data Storage and Processing |
| **Assessment Title:** | *Integrated CA1 Sem2 MSc in Data Analytics* |
| **Lecturer Name:** | David McQuaid  Muhammad Iqbal |
| **Student Full Name:** | Gerard Heraghty |
| **Student Number:** | sba23347 |
| **Assessment Due Date:** | 07/04/2024 |
| **Date of Submission:** | 03/04/2024 |

Gerard Heraghty

**Declaration**

|  |
| --- |
| By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution. |

Air Temperature Forecasting Using Neural Networks

Gerard Heraghty  
MSc Data Analytics  
CCTDublin, Ireland  
sba23347@student.cct.ie

GitHub Link: **https://github.com/GerardCCT/Msc\_DataAnalytics\_Sem2\_CA1**

***Abstract* —** **Accurately forecasting air temperature as a significant meteorological parameter is an interesting research topic as changes in it can affect our day to day lives. Given the wide array of advantages of neural networks at identifying hidden patterns in a dataset without any prior knowledges and their proven track record across time-series problems (e.g. stock market prediction), a fundamental research question that this paper will set out to answer is: how can neural networks be leveraged to forecast air temperature? In conducting this research through a review of past literature and performing a practical analysis on different neural network architectures, analyzing advantages and shortcomings of each network, this paper will aim to conclude on the optimal neural network based on relevant evaluation metrics, as well as analyzing how input data affects the model performances. With advancements in big data technologies, this research paper also sets out to answer the question: how can these technologies be utilized to efficiently store and process the massive amounts of weather data being collected? Reviewing the literature will help gain insight into how state of the art technologies such as Hadoop and Spark are being used to improve the storage and processing of this data. Using datasets sourced from Met Éireann, a demonstration and analysis of these technologies will be performed.**

# Introduction

In recent years, the fields of big data and weather forecasting have become more and more intertwined. Big data technologies provide useful tools for enhancing the precision and efficiency of applying forecasting models to weather data as a result of the development of the technology and proliferation of data sources (Latafat, 2023). Meteorological departments such as Met Éireann collect and analyze huge amounts of data from many sources including weather stations located nationwide. This data is used to create more precise weather models; however, difficulties are encountered in managing huge amounts of data and creating the appropriate analytics tools to make sense of it. The primary motivation behind this research paper is based on this big data problem with a focus on how advancements in neural network architectures to deal with time series data may be leveraged to forecast air temperature. Accurately forecasting the air temperature at any location at a particular time is an important research problem with applications spanning across multiple sectors such as agriculture and energy, and implications in areas such as transportation and public health.

# Literature review

The aim of this literature review is to summarize and critically assess existing literature relating to the use of big data tools and methods for storing and accessing weather data, and the application of different neural network architectures to weather data (specifically focusing on temperature forecasting).

## Storing and Processing Big Data

Big data not only refers to datasets that are large in volume, but also high in velocity, variety, value and veracity, making them difficult to handle using traditional tools and techniques. These innate characteristics are often referred to as the “5 Vs” and are what differentiates between big and small data analytics. (Ishwarappa & Anuradha 2015). With the evolution of technology and the increased multitudes of data flowing, challenges are being encountered in how this data is captured, stored and analyzed. Traditional methods such as relational databases, data marts and data warehouses are often unfeasible options when dealing with large amounts of unstructured and dynamic data. They also run into problems related to scalability, reliability and availability when dealing with larger amounts of data. (Elgendy and Elragal, 2014).

A wealth of weather data is collected and archived daily by various meteorological organizations around the world. Therefore, storage and processing of this big data for accurate weather predictions is a huge challenge. Riyaz et. al (2015) describes the use of MapReduce with Hadoop Distributed File System (HDFS) on 20GB of climatic data collected and made available by the National Climatic Data Centre (NCDC). In their implementation the input data is split into chunks of data influenced by the HDFS block size. Each split is sent to a Mapper function to find the average temperature for each weather station and the reduce function combines all the values to form the final result. Suryanarayana et al. (2019) and Fakherldin et al. (2019) present similar methods at dealing with large amounts of weather data in their application of MapReduce with HDFS to significantly speed up the processing of the very large datasets. Ramya et al. (2015) stores a vast array of unstructured weather data in HDFS and utilizes Hive to read the data. Hive is an SQL based tool that builds over Hadoop to process the data. It converts SQL queries into a series of MapReduce jobs for execution on the cluster. Jayanthi and Sumathi (2017) proposes the Apache spark implementation for weather data analysis. In this paper they suggest that due to spark being 100x faster than MapReduce for in memory data processing, drawbacks from low latency queries and computation time to run processing programs can be overcome by using spark.

## Air Temperature Prediction using Neural Networks

More traditionally, temperature forecasting from weather data was carried out mostly through numerical methods such as the Numerical Weather Prediction (NWP) which involved a set of equations that describe the atmosphere condition and statistical methods such as Autoregressive Integrated Moving Average (ARIMA). (Johnstone, 2021). However, with the relatively recent advances in the development of neural network architectures and computing power, deep learning has become a fundamental part of the new generation of time series forecasting.

Neural networks are machine learning models that make decisions in a similar manner to the human brain, by using processes that mimic the way biological neurons work together to identify phenomena, weigh options and arrive at conclusions. Due to their ability to handle intricate temporal dependencies and non-linear relationships, neural networks (often referred to as deep learning models) have proven to capture intricate patterns in complex datasets, improving time series forecasting significantly compared to traditional methods. (Lim and Zohren, 2021). The performance of neural networks varies depending not only on the architectures used but also the nature and number of input data.

One of the more basic types of neural networks is the feed forward neural network or multilayer perceptron (MLP). Data will flow in the forward direction only with this architecture, from input to output. The neurons are trained with the back propagation learning algorithm where the loss function is minimized by the optimizers (Abirami, 2020). Bilgil et al (2023) demonstrates the use of MLP to predict monthly air temperatures using historical temperature data with reasonable accuracy. As part of their study the model is improved significantly by providing additional input weather data (such as wind direction, rainfall cloud amount), therefore creating a multivariate model. Maqsood et al. (2004) and Kaur et al. (2011) both perform comparisons of MLP and Radial Basis Function Network (RBFN) to predict the hourly temperature 24 hours ahead. RBFN are similar to MLPs however they typically consist of 3 layers and their activation function is a radial basis function (e.g Gaussian). Both papers outline that although the accuracies of both models are similar, the RBFN is far less time consuming. This is due to the iterative optimization process involved in MLPs. Ustaoglu et al. (2008) goes a step further and compares these two models with a third ANN; Generalized Regression Neural Network (GRNN). GRNNs are particularly well-suited for regression tasks where the relationships between inputs and outputs is continuous and smooth. However, it is found not to outperform MLP and RBFN.

MLP networks do not posses’ memory of past inputs or outputs. They treat each input independently and do not consider the sequence or history of data. This limits their ability to predict what’s coming next. (Donges, 2024). This limitation is resolved by Recurrent Neural Networks (RNNs), where the output of the neurons in the current layer are used as feedback to the neurons of the previous layer. A large proportion of studies in the case of air temperature prediction use some form of RNNs which retain information from the previous time step through the recurrent connections.

Roy D S. (2020) proposes the use of a special type of RNN called Long Short-Term Memory (LSTM) networks in an attempt to deal with the long-term dependency problem faced by RNNs (vanishing gradient problem). Information flow in an LSTM is controlled by the cell state and three gates control whether or not to let the information through to the cell state. Although they work very well for making predictions based on time series data, they require more memory and can take longer to train due to more parameters. In this study a univariate model is performed on historical weather data from John F Kennedy Airport however a more complex multivariate model utilizing other input variables such as wind speed, precipitation, is noted to increase model accuracy. Park et al. (2019) suggests the use of a four-layer multivariate LSTM utilizing dropout to prevent overfitting due to the introduction of the additional weather features. Using multi-location weather station data, their model outperformed an MLP model also implemented.

Kreuzer et al (2020) performs a univariate LSTM network but presents an alternative method based on 2D-convolutional LSTM (convLSTM). This more complex model utilizes convolutional neural networks (CNNS) which are more commonly used with image data due to their ability to assign importance to various aspects/objects in the image enabling them to detect patterns. Kreuzer therefore leverages CNNs ability for feature extraction to be applied directly to the multivariate raw weather data before being passed through the LSTM. This technique gives more accurate and long-term temperature forecasts when compared to univariate LSTM.

# Methodology

## Data Collection and Storage

Weather data for this research paper was collected from the Irish meteorological service, Met Éireann, historical data portal (Met Éireann, 2024), which provides weather data for all weather stations across Ireland. Station data located at four Irish airports (Dublin, Cork, Shannon and Knock) was chosen as these stations all recorded the same number of weather parameters (i.e rain, temp, visibility, wind speed, sun) at hourly intervals and contained a sufficient amount of data to demonstrate the big data storage technology. The data was downloaded as a csv file of size 232MB.

Distributed, scalable, non-relational storage systems can store and process large quantities of complex data in real time to overcome the challenges brought on by the huge amount of weather data available (Beakta, 2015). One such solution, Apache Hadoop, was utilized as part of this research paper.

A key component of Hadoop is the Hadoop Distributed File System (HDFS). This primary storage system is designed to handle very large datasets, exhibit a very high level of fault tolerance and can be deployed on low-cost, commodity hardware. HDFS is implemented using a master-worker architecture, where each cluster has one master node and various worker nodes. The files are broken up into block sized chunks (128MB by default) which can be stored as independent units. This large block size is key to reducing the computational cost of seek operations. (Ashtari, 2022). The master node (NameNode) is the core of HDFS and is responsible for managing and storing the file system namespace as well as the block locations and permissions. The worker nodes (DataNodes) store blocks of a file, manage block creation or deletion and transmit heartbeat signals to the NameNode to help track the health of the HDFS.

Due to the size of the weather dataset used in this research paper, the file was stored in HDFS. HDFS is not geared up to efficiently accessing small files due to each small file consuming a block individually leading to excessive memory requirement, access and processing time (Aggarwal et al. 2022). HDFS was designed to work with a small number of large files. Therefore, it is appropriate to store weather data such as that used in this research paper in HDFS.

## Data Processing and Preparation

Although the MapReduce distributed execution framework is typically used within the Apache Hadoop ecosystem for processing files stored in HDFS, Apache Spark was utilized in this research project for all data processing. Apache Spark is an open-source data processing framework that makes use of distributed computing, in-memory stream and batch processing for the processing of big data. (Shaikh et al, 2019). Not only does Spark outperform MapReduce in terms of processing speed, but its ease of use and versatility made it a more suitable option.

Spark offers high-level developer friendly APIs such as its python API, PySpark, which was used for this research paper. Spark’s core data structure, the Resilient Distributed Dataset (RDD), is a fault-tolerant collection of data that can be operated on in parallel. To load the weather data from HDFS *spark.read.csv()* is used to load the data into a PySpark DataFrame. Through this method, the RDD is utilized under the hood, where they are only leveraged for distributed processing and not directly interacted with. PySpark DataFrames are much easier to use, allowing concise and intuitive operations for data processing as well as allowing read and write operations into various formats such as csv (Arora, 2023). The dataframes in pyspark are also lazily evaluated, which is a key difference to how pandas dataframes are evaluated. This means that transformations on the data are not immediately operated, instead spark builds a logical execution plan, until actions on the dataframe are explicitly called. This approach increases efficiency in processing the data while minimizing unnecessary computation and reducing memory usage (Vu, 2020).

To prepare the raw weather data for modelling using PySpark, data understanding, and transformations are performed on the data. This includes recasting column types to be in the correct format, dropping unnecessary columns, renaming columns to more appropriate names, imputing null values and checking for duplicates. This is all done using the *spark.sql* module which provides functions for structured data processing within PySpark. For modelling, only one weather station is used. Dublin Airport weather station was chosen for this and selected out of the dataframe. The dataframe was further filtered to only include the last 5 years of weather data. Most studies discussed in the literature review section only used a few years’ worth of weather data for predicting the air temperature. Including too much data will drastically increase the computation time of the subsequent neural networks. Therefore 5 years was decided as an appropriate amount of data. The resultant dataframe was then saved back to HDFS, where it can be extracted for modelling.

## Modelling

The approach taken with the modelling section of this research paper was to evaluate two types of neural networks, namely a Multilayer Perceptron (MLP) and a Long Short-Term Memory (LSTM) network, at predicting the air temperature using the processed weather data. These particular types of networks were chosen as they are both considered effective models for regression problems and they have varying complexity, which makes for interesting comparisons between the architectures and evaluation of results. Each type of neural network is tested with univariate and multivariate input data to analyze and conclude on the impact the input variables have on the model performance.

### Multilayer Perceptron (MLP)

#### MLP architecture

The MLP is a feed-forward artificial neural network that has been widely effectively in a variety of applications, such as image and audio recognition, natural language processing and time series prediction, more specifically air temperature predictions. (Bilgil et al. 2023) (Chattopadhyay, 2011).

The architecture consists of an input layer, one or more hidden layers and an output layer. As the name suggests, the input data is only processed in one direction sequentially through each layer of interconnected neurons, which are connected by adaptable weights (Sharma, 2024). The output of each neuron is a function of the weighted input, bias and activation function. As part of the training process, an optimizer algorithm is used to adjust the attributes (weights, bias) of the MLP such that the overall loss or error is minimized.

The architecture utilized as part of this research paper consisted of an MLP with 2 hidden layers, making it a 4-layer network. This set up was used as the general practice is to configure a shallow baseline network, consisting of one or two hidden layers.

#### Determining MLP hyperparameters

Choosing an activation is an important step in creating the creating a neural network, as without them the data would pass through the nodes and layers only going through linear functions, limiting their ability to learn complex relationships. ReLU (Rectified Linear Unit) is an effective activation function for regression problems that introduces non-linearity by outputting a value between 0 and infinity. It provides a good baseline activation for the hidden layers due to its efficiency and how it introduces sparsity (setting a subset of neurons to be inactive for a given input) which can help reduce overfitting (Rallabandi, 2023). The output layer activation is left as the default linear activation function to ensure the output is a numerical value that is not squashed or transformed, therefore allowing negative outputs. Adam (Adaptive Moment Estimation) is considered the most efficient optimizer and is utilized for all models implemented in this research paper. By computing the adaptive learning rates for each parameter Adam converges much faster than other algorithms. It performs well across various model architectures and input data. The mean squared error (MSE) loss is calculated as the average of the squared differences between the predicted and the actual values. It is regarded as the default loss to use for regression problems. Since the temperature data is continuous and normally distributed, MSE is considered an appropriate loss function.

The remaining hyperparameters that need to be defined are the number of neurons in each hidden layer, the epochs (number of iterations the network will be trained on) and the batch size (the number of training samples that are fed into the network at once). RandomizedSearchCV was used for tuning these hyperparameters as it is more efficient than GridSearchCV, particularly when the hyperparameter space is large, and easier to implement than more complex tuning algorithms such as Bayesian optimization. An alternative activation function to ReLu, known as Leaky ReLu was also included in this hyperparameter space. One limitation of ReLu for temperature prediction is that it flattens all negative values to 0, which could result in the discarding of potentially important information. Leaky ReLu allows a small gradient for negative values, so this may be a more suitable activation function for the hidden layers.

1. MLP Hyperparameters

| hyperparamter | value |
| --- | --- |
| Hidden layer 1 | 10 neurons |
| Hidden layer 2 | 30 neurons |
| Activation | Leaky ReLu |
| Optimizer | Adam |
| Loss | MSE |
| Epochs | 50 |
| Batch Size | 50 |

### Long Short Term Memory (LSTM) Network

#### MLP limitations and RNNs

Recurrent neural networks (RNN) are deep learning models that are used widely in problems which use sequential or time series data such as speech recognition, stock market prediction and weather forecasting. They were introduced to address the limitations of traditional neural networks, such as MLPs, when it comes to processing sequential data. MLPs process each input independently without consideration for temporal dependencies and sequential patterns, as well as failing to capture long-term dependencies. Even with time step input data, the MLP will consider each input distinctly and not as a continuation of previous time-step values. (Lipton, 2015). To capture the temporal dependency of the sequential data, RNNs have recurrent connections in which the output is transmitted back to the RNN neuron rather than just passing to the next node. These recurrent connections function as an internal memory allowing the network to capture the information from previous steps and utilize it in the current step. A problem with a simple RNN architecture is they often encounter the vanishing gradient problem during backpropagation learning. This occurs when the values of the gradient (learning rate) are too small, and the model stops learning or takes a very long time.

#### LSTM architecture

Long Short-Term Memory (LSTM) networks provide a solution to the shortcoming of simple RNNs (Hochreither and Schmidhuber, 1997).

The LSTM network architecture at a high level works similarly to an RNN. However, it has added complexity to resolve the limitation of the former. As well as recurrent connections, LSTM leverages gating mechanisms to control the flow of information in and out of the memory cell (where data across various time steps is held). The *Forget* gate decides which information to passes through to the cell state (long term memory) and what information is deemed irrelevant using the sigmoid activation. The hidden state (output at the previous point in time, short term memory) and the new input are fed into the *Forget* gate. The *Input* gate controls new information to flow into the cell state using a combination of tanh and sigmoid activation functions, updating the long-term memory. The *Output* gate decides the new hidden state using the newly updated cell state, the previous hidden state and the new input data. (Dolphin, 2020) In order to convert the hidden state to an output or prediction, a linear layer is required as the output layer in the LSTM architecture.

#### Model implementation

The hyperparameters required for the LSTM set up are largely the same as that used for the MLP. In this case, the activation function was left as the default hyperbolic tangent (*tanh*) and sigmoid as the recurrent activation. It is common practice to leave these as the activation functions as they introduce non-linearity. Experimentation was used as the approach for tuning hyperparameters as optimization algorithms such as RandomizedSearchCV previously utilized, were deemed impractical given the computation time. A network with 3 hidden LSTM layers each consisting of 45 neurons was chosen as an appropriate structure to balance model complexity with processing time. A single neuron output layer with a linear activation function was chosen due to the regression nature of the problem. The dropout rate hyperparameter is a regularization method that can improve the performance and accuracy of the LSTM by reducing overfitting (Salehin and Kang, 2023). It works by randomly dropping out a specified percentage of layer outputs during training. A common dropout amount used in research is 20%, therefore this value was implemented between each LSTM layer for this research paper. The model was trained on 50 epochs and a batch size of 100. Choosing these values again ensure good balance between model complexity and computation time.

1. LSTM Hyperparameters

| hyperparamter | value |
| --- | --- |
| All hidden layer | 45 neurons |
| Activation | tanh |
| Recurrent activation | sigmoid |
| Optimizer | Adam |
| Loss | MSE |
| Epochs | 50 |
| Batch Size | 100 |

### Univariate and Multivariate Input data

The performance of the types of neural networks is not only determined by the hyperparameters and architecture, but also the input data. For this reason, both univariate and multivariate models were performed as part of this research paper. In the univariate case, temperature was predicted based solely on the historical air temperature data. This historical temperature data is transformed into sequences of a specified number of timesteps in the past, to predict the current temperature. 48 was chosen as the number of timesteps for this research paper. Since the dataset is hourly data, 48 timesteps would mean that for every hour the neural network predicts, it will consider the previous 2 days would of hourly temperature data to determine the output.

In the case of the multivariate models, all weather readings (i.e. rain, wind, sun, visibility) available in the Met Éireann dataset as well as the historical temperatures were used to predict the temperature for the next hour. Similarly to the univariate models, the input data was transformed into 48 timesteps for each output to be predicted.

# Results and Discussion

To evaluate the performances of the neural network models employed as part of this research paper two evaluation metrics appropriate for regression problems were used. These were R-squared (r2) and mean squared error (mse). R2 is the proportion of variance explained by the model (i.e. goodness of fit). It ranges from 0 to 1, with 0 relating to a model that predicts 0% of the relationship between the input and output variables, while 1 relates to a model that predicts 100% of the output variables. The mean squared error (mse) is the average of the squared differences between the predicted and actual values. This metric gives more weight to larger differences and can be useful when there are unexpected values that need to be considered.

A comparison between the MLP and LSTM networks, for both univariate and multivariate input data using these evaluation metrics is illustrated in the table below.

1. Neural Network Evaluation Metric Comparison

| Neural Network | R2 Test | R2 Train | MSE Test |
| --- | --- | --- | --- |
| MLP Univariate | 0.957283 | 0.976991 | 1.054082 |
| MLP Multivariate | 0.970896 | 0.985283 | 0.718161 |
| LSTM Univariate | 0.977171 | 0.976899 | 0.563331 |
| LSTM Multivariate | 0.977127 | 0.980312 | 0.564415 |

It is seen from the above table that across both evaluation metrics the LSTM networks performed better than the less complex MLP networks. Considering the theory of LSTM networks and their ability to work better with time series data this result is not surprising. However, the multivariate MLP achieved a considerably good R-squared score on the test data and a low MSE, which highlights how even simple feed forward neural networks with appropriate hyperparameters can perform reasonably well on complex time series problems. The multivariate MLP appears to be suffering from slight overfitting due to the larger R2 score for the training data. The performance improvement offered by the multivariate MLP is consistent with the literature previously discussed.

The univariate LSTM is determined to be the best performing model with the lowest MSE and highest R2 score on the testing data. The similarity between the R2 train and test scores suggests that the model is generalizing very well to unseen data, which is a desirable outcome. Given the fact that the multivariate LSTM performs worse than the univariate model may suggest that adding the additional weather data may be creating a model that is too complex. This result appears to contradict a lot of the conclusions drawn about LSTM temperature prediction models discussed in the literature review, where multivariate models are seen to be the top performers.

Furter hyperparameter tuning such as changing the epochs, number of hidden layer neurons, as well as adding additional dropout may result in a better multivariate model. Although considered the best optimizer, a limitation of the *Adam* optimizer is its sensitivity to the choice of learning rate (i.e batch size) especially when using an increased number of parameters. Choosing a value that is too high may lead to overfitting, while too low may lead to very slow convergence. Therefore, choosing a lower batch size than used may improve the performance of the multivariate LSTM.

# Conclusions

In this paper a method for storing and processing large volumes of weather data is proposed and demonstrated. Using HDFS and PySpark (the python API for Apache Spark), the benefits of distributed storage and parallel processing are used in tandem to provide a framework for dealing with the ever-growing amount of weather data available. This framework proved to be very efficient at loading the entire data compared to traditional methods such as using pandas, demonstrating the improved seek times achieved using HDFS and PySpark Data manipulation operations on the PySpark dataframe were also improved through the parallel processing.

In a thorough review of the literature related to air temperature prediction using neural networks, it was decided to evaluate the performance of a simple MLP and a more complex LSTM network for both univariate and multivariate input data. Both LSTM models outperformed the MLP, demonstrating the importance of the temporal dependencies taken into account by LSTM architectures. Univariate LSTM was observed to be the best performing model overall.

A potential follow-up to this research may be to experiment with the time granularity which may help mitigate some noise within the input data, such as sudden weather changes. Using hybrid neural network models such as convLSTM may also assist in smoothing the input data to produces a model that generalizes better.

##### References

Latafat G. (2023) Weather forecasting using big data. Proceeding of the 7th Internation Scientific Conference 151, p. 629 – 635, doi:10.51582/interconf.19-20.04.2023.067

Accessed on 01/04/2023.

Riyaz P.A, Surekha M. V (2015). Leveraging Map Reduce With Hadoop for Weather Data Analytics. IOSR Journal of Computer Engineering 17(3), pp 06 – 12, doi: 10.9790/0661-17320612

Accessed on 21/03/2024.

Suryanarayana V. et al. (2019) Novel Weather Data Analysis Using Hadoop and MapReduce – A Case Study. 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), Coimbatore, India, 2019, p. 204-207, Doi: 10.1109/ICACCS.2019.8728444

Accessed on 21/03/2024.

Fakherldin M. A. (2019) Weather Data Analysis Using Hadoop: Application and Challenges. IOP Conf. Series: Materials Science and Engineering 551 (2019) 012044. doi: 10.1088/1757-899X/551/1/012044

Accessed on 21/03/2024.

Ishwarappa, Anuradha J. (2015). A brief Introduction on Big Data 5Vs Characteristics and Hadoop Technology. Procedia Computer Science, 48, p. 319-324. doi: https://doi.org/10.1016/j.procs.2015.04.188

Accessed on 21/03/2024.

Elgendy N, Elragal A (2014). Big data analytics: A literature review paper. P. Perner (Ed.): ICDM 2014, LNAI 8557, p. 214–227, doi: 10.1007/978-3-319-08976-8\_16

Accessed on 21/03/2024.

Beakta R (2015). Big Data and Hadoop: A review paper

Available at: https://www.researchgate.net/publication/281403776\_Big\_Data\_And\_Hadoop\_A\_Review\_Paper

Accessed on 21/03/2024.

Ashtari H (2022). What is Hadoop Distributed File System (HDFS)? Working, Architecture and Commands

Available at:

https://www.spiceworks.com/tech/big-data/articles/hadoop-distributed-file-system-hdfs/#:~:text=and%20Best%20Practices-,HDFS%20Architecture,based%20on%20their%20replication%20factor.

Accessed on 21/03/2024.

Johnstone C. (2021) Application of neural network in prediction of temperature. Neural Comput & Applic 33, p. 11487–11498. doi: https://doi.org/10.1007/s00521-020-05582-3

Accessed on 21/03/2024.

Lim B., Zohren S. (2021) Time series forecasting with deep learning: survey.

Available at: https://royalsocietypublishing.org/doi/10.1098/rsta.2020.0209

Accessed on 21/03/2024.

Ustaoglu B. et al. (2008) Forecast of daily mean, maximum and minimum temperature time series by three articifial neural network methods. Royal meteorological society, meteorol. Appl. 15: p. 431 – 445. Doi: 10.1002/met.83

Accessed: 21/03/2024

Roy D.S. (2020) Forecasting The Air Temperature at a Weather Station Using Deep Neural Networks. Procedia Computer Science. 178. p. 38 -46. doi: https://doi.org/10.1016/j.procs.2020.11.005

Accessed: 21/03/2024

Park I. et al (2019) Temperature Prediction Using the Missing Data Refinement Model Based on a Long Short-Term Memory Neural Network. Atmosphere 2019, 10(11), 718. Doi: https://doi.org/10.3390/atmos10110718

Accessed on 21/03/2024.

Abirami A. Chitra P (202)) Chapter 14 – Energy efficient edge based real-time healthcare support system. Advanced Computing 117. P. 339 -368. Doi: https://doi.org/10.1016/bs.adcom.2019.09.007

Accessed on 21/03/2024.

Bilgili M. et al. (2023) Artificial neural networks approach for monthly air temperature estimations and maps. Journal of Atmospheric and Solar-Terrestrial Physics, 242, doi: https://doi.org/10.1016/j.jastp.2022.10600

Accessed on 21/03/2024.

Maqsood I. et al. (2004). An ensemble of neural networks for weather forecasting. Neural Comput & Applic 13, p. 112–122.

doi: https://doi.org/10.1007/s00521-004-0413-4

Accessed on 21/03/2024.

Kaur A. et al (2011). Artificial neural networks in forecasting maximum and minimum relative humidity. IJCSNS International Journal of Computer Science and Network Security 11(5) p. 197 – 199.

Available at: http://paper.ijcsns.org/07\_book/201105/20110529.pdf

Accessed on 21/03/2024.

Donges N (2024). A complete guide to recurrent neural networks.

Available at: https://builtin.com/data-science/recurrent-neural-networks-and-lstm

Accessed on 21/03/2024.

Kreuzer D. et al (2020). Short-term temperature forecasts using convolution neural network – An application to different weather stations in Germany. Machine Learning with Applications, 2, science direct.

doi: https://doi.org/10.1016/j.mlwa.2020.100007

Accessed on 22/03/2024.

Met Éireann (2024). Historical Weather Station Data for Ireland.

Available at: https://www.met.ie/climate/available-data/historical-data

Accessed on 24/03/2024.

Aggarwal R et al. (2024). Small files problem in Hadoop: A systematic literature review. Journal of King Saud University - Computer and Information Sciences. 34(10), p. 8658-8674. Doi: https://doi.org/10.1016/j.jksuci.2021.09.007

Accessed on 30/03/2024.

Shaikh et al. (2019) Apache Spark: A Big Data Processing Engine.

Available at: https://www.researchgate.net/publication/339176824\_Apache\_Spark\_A\_Big\_Data\_Processing\_Engine

Accessed on 30/03/2024.

Arora L (2023). RDDs vs Dataframes vs Datasets- What is the Difference and Why Should Data Engineers Care?

Available at: https://www.analyticsvidhya.com/blog/2020/11/what-is-the-difference-between-rdds-dataframes-and-datasets/

Accessed on 30/03/2024.

Vu K. (2020) The Benefits and Examples of Using Apache Spark with PySpark

Available at: https://www.kdnuggets.com/2020/04/benefits-apache-spark-pyspark.html

Accessed on 30/03/2024.

Chattophadhyay et al (2011). Univariate modelling of monthly maximum temperature time series over northeast India.

RmetS Meteorological Applications, 18(1), p. 78-82. Doi: https://doi.org/10.1002/met.211

Accessed on: 30/03/2024.

Sharma P. (2024) Introduction to Feed-Forward Neural Network in Deep Learning

Available at: https://www.analyticsvidhya.com/blog/2022/03/basic-introduction-to-feed-forward-network-in-deep-learning/

Accessed on 30/03/2024.

Rallabandi S. (2023) Activation functions ReLU vs. Leaky ReLu

Available at: https://medium.com/@sreeku.ralla/activation-functions-relu-vs-leaky-relu-b8272dc0b1be

Accessed on 31/03/2024.

Lipton Z. (2015) A Critical Review of Recurrent Neural Networks for Sequence Learning

Available at: https://www.researchgate.net/publication/277603865\_A\_Critical\_Review\_of\_Recurrent\_Neural\_Networks\_for\_Sequence\_Learning

Accessed on 31/03/2024.

Hochreither S, Schmidhuber J (1997), Long Short-Term Memory. Neural Computation 9(8). Pp. 1735 – 1780. doi: 10.1162/neco.1997.9.8.1735

Accessed on 31/03/2024.

Dolphin R. (2020) LSTM Networks, A Detailed Explanation

Available at: https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9

Accessed on: 31/03/2024.

Salehin I, Kang D (2023) A Review on Dropout Regularization Approaches for Deep Neural Networks within the Scholarly Domain

Available at: https://www.mdpi.com/2079-9292/12/14/3106#:~:text=Dropout%20is%20an%20effective%20regularization,easily%20incorporated%20into%20further%20experiments.

Accessed on: 31/03/2024.