

Rainfall Forecasting for Agriculture Using DNN

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

Cibi AswanthV.S (37110166)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

March - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC | 12B Status by UGC | Approved by AICTE

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Cibi Aswanth V.S (Roll. No. 17SCS5042)** who carried out the project entitled "**Rainfall Forecasting for Agriculture Using DNN**" under my supervision from November 2020 to March 2021.

Internal Guide

Dr.K. Mohana prasad, M.TECH., Ph.D

Head of the Department

Dr.S.VIGNESHWARI, M.E., Ph.D

Dr. L. Lakshmanan, M.E., Ph.D

Submitted for Viva voce Examination held on

Internal Examiner

External Examiner

DECLARATION

I **Cibi AswanthV.S (Roll. No. 17SCS5042)** hereby declare that the Project Report entitled "**Rainfall Forecasting for Agriculture Using DNN**" done by me under the guidance of **Dr.K.Mohana Prasad**, at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **Board of Management** of **SATHYABAMA Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneshwari M.E., Ph.D.,** and **Dr. L. Lakshmanan M.E., Ph.D.,** Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.K.Mohana Prasad M.TECH., Ph.D,** for his valuable guidance, suggestions and constant encouragement paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The occurrence of rainfall is dependent on a myriad of atmospheric parameters. This paper proposes a systematic approach to analyze various parameters that affect precipitation in the atmosphere. Different ground-based weather features such as Temperature, Relative Humidity, Dew Point, Solar Radiation, PWV along with Seasonal and Diurnal variables are identified, and a detailed feature correlation study is presented. Input parameter like average temperature in month, wind velocity, humidity, and cloud cover was consider for predicting rainfall in non-monsoon session. The performance of the results was measure with MSE (mean squared error), correlation coefficient, coefficient of efficiency and MAE (mean absolute error).

V

Table of Contents

Chapter number:	Title	Pgno:
1	Introduction	
1.1	Overview	1
1.2	Scope of the Project	1
1.3	Domain Overview	1 - 2
1.4	Deep Learning	2 - 14
2	Literature Survey	
2.1	Literature review	15 - 18
3	System Analysis	
3.1	Existing System	19
3.2	Drawbacks of Existing System	20
3.3	Proposed System	21
3.4	Advantages of Proposed System	21
4	Hardware and Software specification	
4.1	Hardware Specification	22
4.2	Software Specification	22
4.3	Software Description	22
4.3.1	About Python	22 - 24
4.3.2	About IDE	25 - 29
5	System Design	

5.1 Architecture Diagram	30
6	System Implementation
6.1 List of Modules	31
6 1.1 Data Visualization	31
6 1.2 Preprocessing	32
6 1.3 Prediction	33
7	Conclusion and Future Work
7.1 conclusion	34
7.2 Future work	35
References	35 - 36
A) Screen Shots	36
B) Source Code	37 - 60

Chapter 1

Introduction

1.1 Overview

Thus, proposed scheme helpstoestablishtheeffectivenessoftheDL approachinaugmenting the quality of rainfall prediction, exploiting the potential of NN in simulating meteorological variables. The developedhybrid model is applied to twelve different locations in different climatic regimes in terms of daily precipitation characteristics.

1.2 Scope of the Project

The main contributions of this project therefore are:

- ▶ Data Analysis
- ▶ Dataset Preprocessing
- ▶ Training the Model
- ▶ Testing of Dataset

1.3 Domain Overview

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing. Deep learning is a machine learning technique that teaches computers to do whatcomes naturally to humans: learn by example. Deep learning is a key technology behind

1

driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labelled data and neural network architectures that contain many layers.

However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unravelling this wealth of information and are increasingly adapting to AI systems for automated support.

1.4 Deep Learning

About pandas:

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The DataFrame is one of these structures.

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.

Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas. Data in pandas is often used to feed statistical analysis in SciPy, plotting functions from Matplotlib, and machine learning algorithms in Scikit-learn.

Jupyter Notebooks offer a good environment for using pandas to do data exploration and modeling, but pandas can also be used in text editors just as easily.

Jupyter Notebooks give us the ability to execute code in a particular cell as opposed to running the entire file. This saves a lot of time when working with large datasets and complex transformations. Notebooks also provide an easy way to visualize pandas' DataFrames and plots. As a matter of fact, this article was created entirely in a Jupyter Notebook.

There are two types of data structures in pandas: Series and DataFrames.

1. Series: a pandas Series is a one dimensional data structure ("a one dimensional array") that can store values — and for every value it holds a unique index, too.
2. DataFrame: a pandas DataFrame is a two (or more) dimensional data structure — basically a table with rows and columns. The columns have names and the rows have indexes.

Those who are familiar with R know the data frame as a way to store data in rectangular grids that can easily be overviewed. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data

for a specific variable. This means that a data frame's rows do not need to contain, but can contain, the same type of values: they can be numeric, character, logical, etc.

3

Now, DataFrames in Python are very similar: they come with the Pandas library, and they are defined as two-dimensional labelled data structures with columns of potentially different types.

In general, you could say that the Pandas DataFrame consists of three main components: the data, the index, and the columns.

Firstly, the DataFrame can contain data that is:

- a Pandas DataFrame
- a Pandas Series: a one-dimensional labelled array capable of holding any data type with axis labels or index. An example of a Series object is one column from a DataFrame.
- a NumPy ndarray, which can be a record or structured
- a two-dimensional array
- dictionaries of one-dimensional array's, lists, dictionaries or Series.

Some of the key features of Python Pandas are as follows:

It provides DataFrame objects with default and customized indexing which is very fast and efficient.

- There are tools available for loading data of different file formats into in-memory data objects.
- It is easy to perform data alignment and integrated handling of missing data in

Python Pandas.

- It is very simple to perform pivoting and reshaping of data sets in Pandas.

4

- We can easily insert and delete columns from a data structure.
- Data aggregation and transformations can be done using group by.
- High-performance merging and joining of data can be done using Pandas.
- It also provides time series functionality.
- Inserting and deleting columns in data structures.
- Merging and joining data sets.
- Reshaping and pivoting data sets.
- Aligning data and dealing with missing data.
- Manipulating data using integrated indexing for DataFrame objects.
- Performing split-apply-combine on data sets using the group by engine.
- Manipulating high-dimensional data in a data structure with a lower dimension using hierarchical axis indexing.
- Subsetting, fancy indexing, and label-based slicing data sets that are large in size.
- Generating data range, converting frequency, date shifting, lagging, and other time-series functionality.
- Reading from files with CSV, XLSX, TXT, among other formats.
- Arranging data in an order ascending or descending.
- Filtering data around a condition.
- Analyzing time series.

- Iterating over a data set.

5

Numpy

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy.

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can't miss when you're learning data science, mainly because this library provides you with an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient.

NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays.

an array is basically nothing but pointers. It's a combination of a memory address, a data type, a shape, and strides:

6

- The data pointer indicates the memory address of the first byte in the array,
- The data type or dtype pointer describes the kind of elements that are contained within the array,
- The shape indicates the shape of the array, and
- The strides are the number of bytes that should be skipped in memory to go to the next element. If your strides are (10,1), you need to proceed one byte to get to the next column and 10 bytes to locate the next row.

Or, in other words, an array contains information about the raw data, how to locate an element and how to interpret an element.

With NumPy, we work with multidimensional arrays. We'll dive into all of the possible types of multidimensional arrays later on, but for now, we'll focus on 2-dimensional arrays. A 2-dimensional array is also known as a matrix, and is something you should be familiar with. In fact, it's just a different way of thinking about a list of lists. A matrix has rows and columns. By specifying a row number and a column number, we're able to extract an element from a matrix. We can create a NumPy array using the `numpy.array` function. If we pass in a list of lists, it will automatically create a NumPy array with the same number of rows and columns. Because we want all of the elements in the array to be float elements for easy computation, we'll leave off the header row, which contains strings. One of the limitations of NumPy is that all the elements in an array have to be of the same type, so if we include the header row, all the elements in the array will be read

in as strings. Because we want to be able to do computations like find the average quality of the wines, we need the elements to all be floats.

7

NumPy has several advantages over using core Python mathematical functions, a few of which are outlined here:

1. NumPy is extremely fast when compared to core Python thanks to its heavy use of C extensions.
2. Many advanced Python libraries, such as Scikit-Learn, SciPy, and Keras, make extensive use of the NumPy library. Therefore, if you plan to pursue a career in data science or machine learning, NumPy is a very good tool to master.

NumPy comes with a variety of built-in functionalities, which in core Python would take a fair bit of custom code.

Math plot lib

Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data visualization, and this property can be achieved by making use of Matplotlib.

Matplotlib makes use of many general-purpose GUI toolkits, such as wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications.

John D. Hunter was the person who originally wrote Matplotlib, and its lead developer was Michael Grootboom. One of the free and open-source Python library which is basically used for technical and scientific computing is Python SciPy. Matplotlib is

widely used in SciPy as most scientific calculations require plotting of graphs and diagrams.

8

Matplotlib is a plotting library like GNU plot. The main advantage towards GNU plot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well.

Another reason for the attractiveness of Matplotlib lies in the fact that it is widely considered to be a perfect alternative to MATLAB, if it is used in combination with Numpy and Scipy. Whereas MATLAB is expensive and closed source, Matplotlib is free and open source code. It is also object-oriented and can be used in an ObjectOriented way. Furthermore it can be used with general-purpose GUI toolkits like wxPython, Qt, and GTK+. There is also a procedural "pylab", which designed to closely resemble that of MATLAB. This can make it extremely easy for MATLAB users to migrate to matplotlib.

Matplotlib can be used to create publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Another characteristic of matplotlib is its steep learning curve, which means that users usually make rapid progress after having started. The official website has to say the following about this: "matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc, with just a few lines of code."

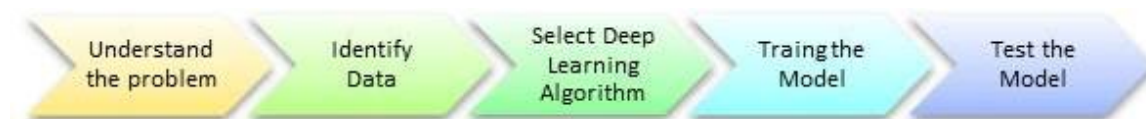
Deep Learning: Deep learning is a computer software that mimics the network of neurons in a brain. It is a subset of machine learning and is called deep learning because it makes use of deep neural networks.

9

Deep learning algorithms are constructed with connected layers.

- The first layer is called the Input Layer
- The last layer is called the Output Layer
- All layers in between are called Hidden Layers. The word deep means the network join neurons in more than two layers.

A deep neural network provides state-of-the-art accuracy in many tasks, from object detection to speech recognition. They can learn automatically, without predefined knowledge explicitly coded by the programmers.



To grasp the idea of deep learning, imagine a family, with an infant and parents. The toddler points objects with his little finger and always says the word 'cat.' As its parents are concerned about his education, they keep telling him 'Yes, that is a cat' or 'No, that is not a cat.' The infant persists in pointing objects but becomes more accurate with 'cats.' The little kid, deep down, does not know why he can say it is a cat or not. He has just learned how to hierarchies complex features coming up with a cat by looking at the

pet overall and continue to focus on details such as the tails or the nose before to make up his mind.

10

A neural network works quite the same. Each layer represents a deeper level of knowledge, i.e., the hierarchy of knowledge. A neural network with four layers will learn more complex feature than with that with two layers.

The learning occurs in two phases.

1. The first phase consists of applying a nonlinear transformation of the input and create a statistical model as output.
2. The second phase aims at improving the model with a mathematical method known as derivative.

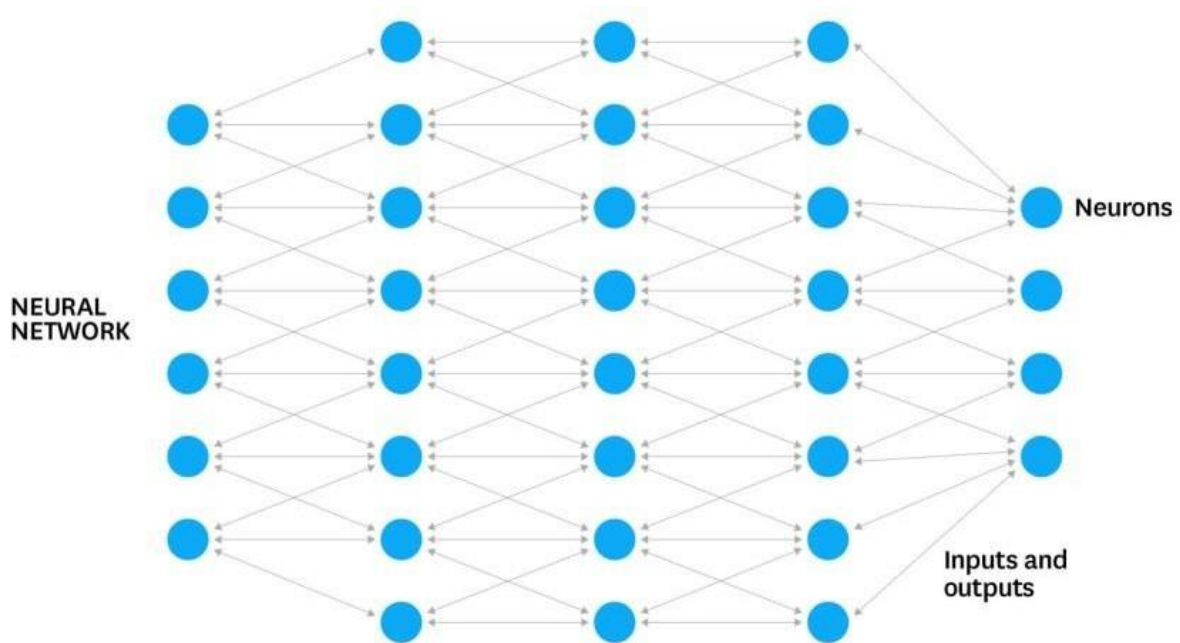
The neural network repeats these two phases hundreds to thousands of time until it has reached a tolerable level of accuracy. The repeat of this two-phase is called an iteration

Classification of Neural Networks

1. Shallow neural network: The Shallow neural network has only one hidden layer between the input and output.
2. Deep neural network: Deep neural networks have more than one layer. For instance, Google LeNet model for image recognition counts 22 layers.

The computational models in Deep Learning are loosely inspired by the human brain.

The multiple layers of training are called Artificial Neural Networks (ANN).



Neuron

Artificial Neural Networks contain layers of neurons. A neuron is a computational unit that calculates a piece of information based on weighted input parameters. Inputs accepted by the neuron are separately weighted.

Inputs are summed and passed through a non-linear function to produce output. Each layer of neurons detects some additional information, such as edges of things in a picture or tumors in a human body. Multiple layers of neurons can be used to detect additional information about input parameters.

Nodes

Artificial Neural Network is an interconnected group of nodes akin to the vast network of layers of neurons in a brain. Each circular node represents an artificial neuron and an arrow represents a connection from the output of one neuron to the input of another.

Inputs

12

Inputs are passed into the first layer. Individual neurons receive the inputs, with each of them receiving a specific value. After this, an output is produced based on these values.

Outputs

The outputs from the first layer are then passed into the second layer to be processed. This continues until the final output is produced. The assumption is that the correct output is predefined.

Each time data is passed through the network, the end result is compared with the correct one, and tweaks are made to their values until the network creates the correct final output each time.

Some of the commonly used neural networks are as follows:

1. Artificial Neural Network (ANN)
2. Convolutional Neural Network (CNN)
3. Recurrent Neural Network (RNN)
4. Deep Neural Network (DNN)

Artificial neural networks are one of the main tools used in machine learning. As the “neural” part of their name suggests, they are brain-inspired systems which are intended to replicate the way that we humans learn. Neural networks consist

of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use. They are excellent tools for finding patterns which are far too complex or numerous for a human programmer to extract and teach the machine to recognize.

13

Generally, the working of a human brain by making the right connections is the idea behind ANNs. That was limited to use of silicon and wires as living neurons and dendrites.

Here, neurons, part of human brain. That was composed of 86 billion nerve cells. Also, connected to other thousands of cells by Axons. Although, there are various inputs from sensory organs. That was accepted by dendrites. As a result, it creates electric impulses. That is used to travel through the Artificial neural network. Thus, to handle the different issues, neuron send a message to another neuron.

As a result, we can say that ANNs are composed of multiple nodes. That imitate biological neurons of the human brain. Although, we connect these neurons by links.

Also, they interact with each other. Although, nodes are used to take input data. Further, perform simple operations on the data. As a result, these operations are passed to other neurons. Also, output at each node is called its activation or node value.

Chapter – 2

Literature Survey

2.1 Literature Review

Literature Survey 1	
Title	Rainfall Rate Prediction for Propagation Applications: Model Performance at Regional Level Over Ireland
Authors	Lorenzo Luini , Luis Emiliani, Xavier Boulanger, Carlo Riva , and Nicolas Jeannin
Published Year	2017
Efficiency	✓ Optimum set of features are used ✓ This method significantly reduces the false alarm rates ✓ Reflects the local peculiarity of the rainfall process
Drawbacks	↓ Inaccurate estimations of the missing values ↓ High prediction complexity for large datasets ↓ Higher prediction complexity with higher dimensions

Description	Three global rainfall rate prediction methods are evaluated in their ability to estimate local precipitation statistics, which are the key to predict the impact of rain on the propagation of electromagnetic waves through the atmosphere. Specifically, the International Telecommunication Union-Radiocommunication Sector (ITU-R) P.837-6, model for rainfall statistics estimation (MORSE), and the ITU-R P.837-7 prediction methods are tested against long-term rainfall data collected in 19 sites in Ireland. The results indicate that the ITU-R P.837-7 prediction method delivers the best performance, and that both the ITU-R P.837-6 prediction method and MORSE exhibit a positive bias, likely due to the overestimation of the yearly rain amount in the maps used as input to such models. The results of the testing activity provide information on the accuracy of rainfall rate prediction methods at regional level, an important factor to consider given the direct link between the magnitude of rain-induced attenuation, and the operational frequency of wireless communication links. MORSE is a global method for the prediction of 1 min integrated $P(R)$ at different time scales, ranging from a monthly to a yearly basis. The prediction method requires as input the local convective (M_c) and total (M_t) rain amounts, cumulated over the period of interest, which can be extracted from gridded numerical weather prediction The availability of the rain gauges during the period of observation exceeds 97% in all stations. Table I and Fig. 1 provide an overview of station location and measurement duration, which ranges from four (2012–2015) to eight (2008–2015) years. This extensive set of data was used to calculate the long-term mean yearly $P(R)$s using the procedure described in detail in [15]. In aggregate, the measured data spans over 135 station years, making it unique for the purpose of communication system evaluation.
-------------	---

Literature Survey 2

Title	A Data-Driven Approach for Accurate Rainfall Prediction
Authors	Shilpa Manandhar , Student Member, IEEE, Soumyabrata Dev , Member, IEEE, Yee Hui Lee, Senior Member, IEEE, Yu Song Meng , Member, IEEE, and Stefan Winkler
Published Year	2019
Efficiency	✓ Detailed feature correlation study is performed. ✓ It uses the multiple linear regression instead of simple linear regression ✓ Calculates the time series and autocorrelations
Drawbacks	↓ Not Adaptable to Large Samples ↓ Tedious message updating ↓ Narrowly specialized knowledge

Description	This paper proposes a systematic approach to analyze various parameters that affect precipitation in the atmosphere. Different ground-based weather features such as Temperature, Relative Humidity, Dew Point, Solar Radiation, PWV along with Seasonal and Diurnal variables are identified, and a detailed feature correlation study is presented. While all features play a significant role in rainfall classification, only a few of them, such as PWV, Solar Radiation, Seasonal, and Diurnal features, stand out for rainfall prediction. Based on these findings, an optimum set of features are used in a data-driven machine learning algorithm for rainfall prediction. The experimental evaluation using a 4-year (2012–2015) database shows a true detection rate of 80.4%, a false alarm rate of 20.3%, and an overall accuracy of 79.6%. Compared to the existing literature, our method significantly reduces the false alarm rates. SVM is a parametric method that generates a hyperplane or a set of hyperplanes in the vector space by maximizing the margin between classifiers to the nearest neighbor data [29]. In this paper, we use SVM to classify rain and no-rain cases using different weather parameters as features. Consider a feature matrix X of dimension $n \times m$, where n is the number of features (weather variables such as T, DPT, RH, etc.) and m is the number of samples. Consider an output matrix Y, a column matrix of dimension $m \times 1$, where values can be either 1 or -1, indicating rain or no-rain, respectively. SVM is trained and tested using our weather database. The weather data have a temporal resolution of 1 min, and therefore, a 1-year database generally includes a total of 365×1440 data points. Out of these 525 600 data points, there are far fewer data points with rain than without, because rain is a relatively rare event. For example, in the year 2015 for NTUS station, there are 525 120 valid data points. Out of these, there are only 5017 data points with rain, referred to as minority cases, whereas there are 520 103 data points without rain (majority cases). The minority-to-majority ratio, here, is nearly 1:104, which poses the problem of a highly imbalanced data set. Training a model with such skewed data (skewed toward non-rain data points) would result in a biased model, which is dominated by the characteristics of the majority database.
-------------	---

Chapter – 3

System Analysis

3.1 Existing System

The main work of pre-processing includes min-max normalization and PCA. The initial data of this model include five high-altitude factors and eight earth surface factors. In meteorology, geopotential height is usually used instead of actual height, and an isobaric surface is used instead of horizontal height. Therefore, meteorological data is always in the format of the isobaric surface. For example, 500hPa is usually corresponding to the height of 5.5km. The rainfall system is often controlled by the weather system of 500hPa.

Temperature dew point difference is directly related to the humidity. Temperature dew point difference, temperature and actual height value affect the internal energy of rainfall system. Ground surface factor represents the local atmospheric condition in the area. The difference in surface factors in different regions leads to different rainfall.

Min-max Normalization. One of the most commonly used data normalization methods is Min- max normalization. It can standardize data between 0 and 1. Because different factors have different magnitude, it is necessary to pre-process the data. For a sequence to be processed, the maximum value of the sequence corresponds to 1, the minimum value corresponds to 0, and the rest values are proportionally transformed between 0 and 1.

3.2 Drawbacks of Existing System

- ↓ Can't learn relation between factors.
- ↓ Do not take account of the influence features.
- ↓ Maximizes the complexity of the problem.
- ↓ Prone to Errors.
- ↓ Scalability is an issue for those designs as the number of users and the amount of data grow.

3.3 Proposed System

Deep Neural Network models depend on past data to provide predictions it is immaterial whether the data is state data or district data as long as the data parameters or attributes are constant. Moreover, a state data is average of all districts data only. Hence, if a model shows good performance for data of Jagatsinghpur district of Odisha then definitely it will show equivalently good performance for entire state.

The variables are scaled between 0 and 1, using MinMax Scaler class from the sklearn preprocessing available in scipy library to avoid the problem of scaling. After scaling, the dataset is split into two parts, i.e. training and testing. A kfold cross validation is used to evaluate the prediction skill of the model. Accordingly, seven (k) approximately equal folds are considered and the model is repeatedly trained on the remaining six (k-1) folds, and its performance is measured on the remaining fold.

It is also a fully connected NN that receives the data in a one-dimensional vector form. Therefore, after the convolutional layer, a flattened layer is added. Next, a fully connected dense layer along with the activation functions are added. The fully connected layers have a more number of neurons than output layers. In this way, neural networks are allowed to think wider before they converged to the output layer. Output layer is also a fully connected dense layer responsible for giving the prediction. There are several parameters to be specified to fix the model architecture.

3.4 Advantages of Proposed System

- ✓ Outstanding Learning Capabilities
- ✓ Effectively improve prediction accuracy.
- ✓ Reduces the consumption of hardware resources
- ✓ Lowering the Complexity Threshold
- ✓ Low computational cost

Chapter – 4

Hardware and Software specification

4.1 Hardware Specification

Hardware Minimum Requirement

Disk Space 32 GB or more, 10 GB or more for Foundation Edition

Processor 1.4 GHz 64 bit

Memory 512 MB

Display (800 × 600) Capable video adapter and monitor

4.2 Software Specification

Backend Technologies

- ✓ Python
- ✓ Numpy
- ✓ Sci-learn
- ✓ Eclipse IDE

Frontend Technologies

- ✓ Web Technologies
- ✓ Bootstrap

4.3 Software Description

4.3.1 About Python:

Python is a free, open-source programming language. Therefore, all you have to do is install Python once, and you can start working with it. Not to mention that you can contribute your own code to the community. Python is also a cross-platform compatible language. So, what does this mean? Well, you can install and run Python on several operating systems. Whether you have a Windows, Mac or Linux, you can rest assured that Python will work on all these operating systems.

Python is also a great visualization tool. It provides libraries such as Matplotlib, seaborn and bokeh to create stunning visualizations.



In addition, Python is the most popular language for machine learning and deep learning. As a matter of fact, today, all top organizations are investing in Python to implement machine learning in the back-end.

Python Line Structure

Python coding style comprises physical lines as well as logical lines or statements. A physical line in a Python program is a sequence of characters, and the end of the line terminates the line sequence as opposed to some other languages, such as C and C++ where a semi-colon is used to mark the end of the statement. A logical line, on the other hand, is composed of one or more physical lines. The use of a semi-colon is not prohibited in Python, although it's not mandatory. The NEWLINE token denotes the end of the logical line. A logical line that only contains spaces, comments, or tabs are called blank lines and they are ignored by the interpreter.

As we saw that in Python, a new line simply means that a new statement has started. Although, Python does provide a way to split a statement into a multiline statement or to join multiple statements into one logical line. This can be helpful to increase the readability of the statement. Following are the two ways to split a line into two or more lines:

Explicit Line Joining

In explicit line joining, we use a backward slash to split a statement into a multiline statement.

Implicit Line Joining

Statements that reside inside [], {}, or () parentheses can be broken down into two or more physical lines without using a back slash.

Multiple Statements on a Single Line

In Python, it is possible to club multiple statements in the same line using a semi-colon; however, most programmers do not consider this to be a good practice as it reduces the readability of the code.

Whitespaces and Indentation

Unlike most of the programming languages, Python uses indentation to mark a block of code. According to Python coding style guideline or PEP8, we should keep an indent size of four. Most of the programming languages provide indentation for better code

formatting and don't enforce to have it. But in Python it is mandatory. This is why indentation is so crucial in Python.

22

Comments in any programming language are used to increase the readability of the code. Similarly, in Python, when the program starts getting complicated, one of the best ways to maintain the readability of the code is to use Python comments. It is considered a good practice to include documentations and notes in the python syntax since it makes the code way more readable and understandable to other programmers as well, which comes in handy when multiple programmers are simultaneously working on the same project. The code can only explain how it does something and not why it does that, but Python comments can do that. With Python comments, we can make documentations for various explanations in our code itself. Comments are nothing but tagged lines of codes which increase the readability of a code and make it self-explanatory. There are different ways of creating comments depending on the type of comment we want to include in our code. Following are different kinds of comments that can be included in our Python program:

1. Single Line Comments
2. Docstring Comments
3. Multiline Comments

Single line Python comments are marked with `#` character. These comments end at the end of the physical line, which means that all characters starting after the `#` character are part of the comment. Python has the documentation strings feature which is usually the first statement included in functions and modules. Rather

than being ignored by the Python Interpreter like regular comments, docstrings can actually be accessed at the run time using the dot operator.

23

It gives programmers an easy way of adding quick notes with every Python module, function, class, and method. To use this feature, we use triple quotes in the beginning of the documentation string or comment and the closing triple quotes at the end of the documentation comment. Docstrings can be one-liners as well as multi-liners.

Unlike some programming languages that support multiline comments, such as C, Java, and more, there is no specific feature for multiline comments in Python. But that does not mean that it is totally impossible to make multiline comments in Python. There are two ways we can include comments that can span across multiple lines in our Python code. Python Block Comments: We can use several single line comments for a whole block. This type of comment is usually created to explain the block of code that follows the Block comment. Python Block comment is the only way of writing a real comment that can span across multiple lines. It is supported and preferred by Python's PEP8 style guide since Block comments are ignored by Python interpreter or parser. However, nothing is stopping programmers from using the second 'non-real' way of writing multiline comments in Python which is explained below. Using Docstrings: Docstrings are largely used as multiline comments in Python by many programmers since it is the closest thing to having a multiline comment feature in Python. While it is not wrong to use docstrings when we need to make multiline comments, it is important to keep in mind that there is a significant difference between docstrings

and comments. Comments in Python are totally ignored by the Python Interpreter, while docstrings, when used inside the Python function, can be accessed at the run time.

4.3.2 About IDE

IDE – Integrated Development Environment. It is software suite. It contains different instruments for web application software engineers to compose and test applications.

Programming Web application software engineers utilize wide scope of devices habitually all through application source code creation, building and application testing. Programming Development apparatuses frequently incorporate word processors for source code creation, code libraries for reuse, compilers and test stages.

An Integrated Development Environment (IDE) is a product application that gives a programming domain to streamline creating and troubleshooting programming. Instead of playing out every one of the means required to make an executable program as disconnected individual assignments, it brings every one of the devices required into one application and workspace. Every one of the devices has a familiarity with the earth, and they cooperate to display a consistent improvement set for the designer.

An IDE typically has an editor and a compiler window. There, we can write our code, run it, and compile it. Hence, it becomes much easier to develop applications using an IDE.

The major reasons for using an IDE are given below:

- ▶ We can easily write our code in its text editor window.
- ▶ To store the resource files of a project, we have a project editor window.

- ▶ It makes debugging of the code easier.
- ▶ An IDE's autosuggest feature helps us in writing efficient code.
- ▶ We can import other projects for integrating with the existing one.

25

Without an IDE, a product web application software engineer must choose, send, incorporate and deal with these instruments independently. An IDE brings huge numbers of those improvement related instruments all together system, application or administration. The incorporated toolset is intended to streamline programming advancement and can distinguish and limit coding missteps and grammatical errors.

- ✓ Komodo
- ✓ NetBeans
- ✓ PyCharm
- ✓ Python for Visual Studio.NET
- ✓ PyDev
- ✓ Py Studio
- ✓ Li Eclipse (Eclipse)

LiEclipse is at the core for all Python based application development activities. It's been available for many years. LiEclipse is a set of plugins to enhance Eclipse. It improves the overall Eclipse experience. It is best IDE tool to develop Python Applications.

Advantages of LiClipse

- Perfect Debugger

26

- An enhanced ability to analyze the code for quality and security concerns
- Auto complete feature

Based on the various IDE that are described above, we decided to use LiClipse as IDE (Integrated Development Environment) for developing this e-Voting web based application.

Important Features That Make LiClipse our IDE of Choice

- Rich Reusable Libraries are available
- Model Driven Development
- Enterprise Python Tooling
- Powerful Debugging options

IDE is a program that is used to write applications (other programs) in a variety of programming languages. It allows you to complete many distinct program development steps in one place (program window). A program which allows you to do this is known as an integrated development environment, or IDE. The program is free and easy to import onto your home computer.

LiClipse is used for writing programs in Python. There are several steps that must be completed to set up LiClipse and further steps to start up a new Python project (set of Python components which work together to make up an application). PyCharm is an

Integrated Development Environment (IDE) that provides tools for writing efficient code. It supports various frameworks such as Django, Flask, and Pyramid. PyCharm is majorly used by developers for application development with Python language.

27

Let's suppose, we are making a project using Python programming language. We know that Python is extensively popular among the developer's community, as well as in industries, due to its following features:

- ▶ Simple to code: Python is a very simple language compared to other programming languages. Also, it is a developer-friendly language.
- ▶ Object-oriented language: Python is an object-oriented language that uses concepts of classes and objects.
- ▶ GUI programming support: We can create graphical user interfaces using Python Tkinter library. Besides, we can use modules such as PyQt5, PyQt4, and wxPython to create a GUI application.
- ▶ High-level language: Python is a high-level programming language. We do not need to take care of managing memory or the system architecture while writing Python codes.
- ▶ Portability: Python is a portable programming language, i.e., we can run Python codes on any platform. For example, we can run a code written for Windows in Linux, UNIX, and Mac, without changing it.
- ▶ Integrated language: Python is an integrated programming language. We can easily integrate C, C++, etc. with Python.

- ▶ Interpreted language: Python is an interpreted programming language as well. Python's source code is changed to bytecode. Due to this, it is directly executed without compiling.
- ▶ Large set of standard libraries: Python consists of a rich set of modules, packages, and libraries. It helps in various stages of application development.

28

"notebook" or "notebook documents" denote documents that contain both code and rich text elements, such as figures, links, equations, ... Because of the mix of code and text elements, these documents are the ideal place to bring together an analysis description, and its results, as well as, they can be executed perform the data analysis in real time.

The Jupyter Notebook App produces these documents.

As a server-client application, the Jupyter Notebook App allows you to edit and run your notebooks via a web browser. The application can be executed on a PC without Internet access, or it can be installed on a remote server, where you can access it through the Internet.

Its two main components are the kernels and a dashboard.

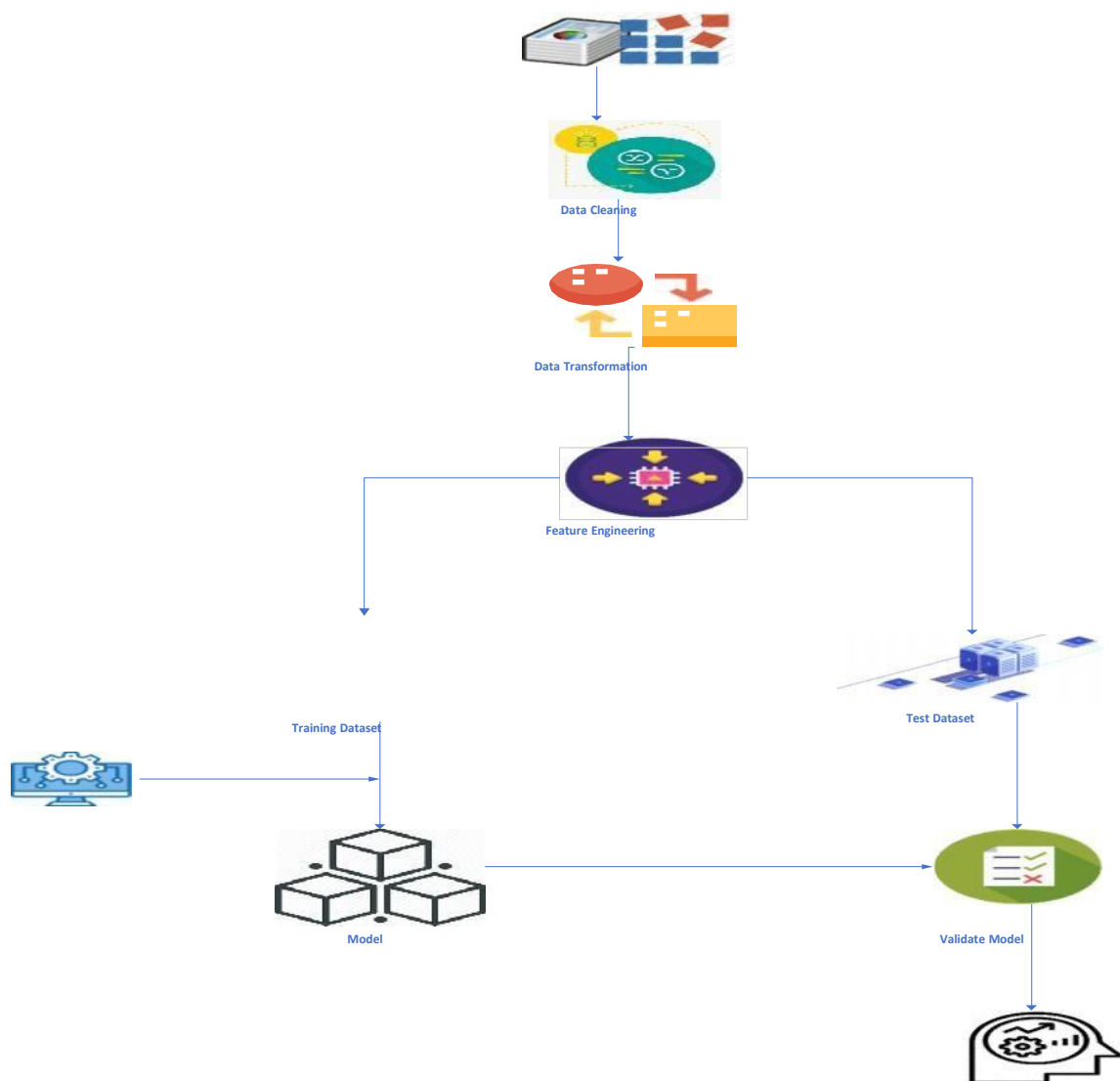
A kernel is a program that runs and introspects the user's code. The Jupyter Notebook App has a kernel for Python code, but there are also kernels available for other programming languages.

The dashboard of the application not only shows you the notebook documents that you have made and can reopen but can also be used to manage the kernels: you can which ones are running and shut them down if necessary.

Chapter – 5

System Design

5.1 Architecture Diagram



Chapter – 6

System Implementation

6.1 List of Modules

6.1.1 Data visualization

Data Visualization is common in day-to-day life. Various charts and graphs are used to illustrate the practical approach towards the classification of rainfall with the help of data visualization methods. Since it was impossible to analyze the large datasets earlier, the data visualization techniques have made it easier to plot the graphs for the better understanding of the weather. With the help of data visualization patterns such as the highest, lowest and average rainfall in the States/Union Territories the weather of India has been visualized.

Exploratory data analysis is generally cross-classified in two ways. First, each method is either non-graphical or graphical. And second, each method is either univariate or multivariate (usually just bivariate). Non-graphical methods generally involve calculation of summary statistics, while graphical methods obviously summarize the data in a diagrammatic or pictorial way. Univariate methods look at one variable (data column) at a time, while multivariate methods look at two or more variables at a time to explore relationships. Usually our multivariate EDA will be bivariate (looking at exactly two variables), but occasionally it will involve three or more variables. It is almost always a good idea to perform univariate EDA on each of the components of a multivariate EDA before performing the multivariate EDA.

6.1.2 Preprocessing

A pre-processing technique used to reduce the effects of minor observation errors. The sample is divided into intervals and replaced by categorical values. Indicator

variables: This technique converts categorical data into boolean values by creating indicator variables. If we have more than two values (n) we have to create $n-1$ columns. Centring & Scaling: We can centre the data of one feature by subtracting the mean to all values. To scale the data, we should divide the centred feature by the standard deviation.

Labels represents string labels of both ordinal and nominal features in relation to categorical feature set. Some labels may have order associated with them (ordinal features) while others may not have any orders associated with them (nominal features). It is an important part of data preprocessing to encode labels appropriately in numerical form in order to make sure that the learning algorithm interprets the features correctly. In the following section, you will see how you could use `LabelEncoder` class of `sklearn.Preprocessing` module to encode labels of categorical features. Label encoding refers to transforming word labels into a numerical form so that algorithms can understand how to operate on them.

6.1.3 Prediction

Firstly the input is given to the NN. The weights which are giving to neural networks are random numbers which are untrained. The value of weights which are giving to the NN must have values in the middle of -1 and +1. Weight training is used to decrease the error function in neural networks. By doing all this it gives the output. If the output is incorrect then adjust the weights on the NN. By doing so on up to the resemblance output is given by the neural network. Then stop adding the weights and consider the weights and check it for another dataset. The rainfall has been predicted using deep learning techniques. Two deep learning techniques which were used are Multilayer Perceptron and Auto-Encoders. Auto-Encoders are responsible in time series forecasting by performing the feature extraction.

32

The weights consider in the next data in the data goes wrong then the process repeats. Hidden layered neural networks are calculated by the sigmoid function. It is having one neuron in the output

layer. All these neurons are interrelated with each layer. In Backpropagation NN input signals are captured by input neurons and output signals are captured by outputneuron.

7.1 Conclusion

The correspondence between the observed and the predicted rainfall is better than the correspondence between the observed and the simulated rainfall magnitude. This can be considered as the potential of the hybrid NN in improving the rainfall prediction. We developed a different model: using a rainfall-flow pattern based on historical rainfall and flood flow data for real-time prediction of short-term flood stream-flow. Our models predict the flood process line in real-time using hydrological feature extraction and spatial-temporal metrics for similar rainfall flow patterns. The experimental results based on the datasets of various models in wet and drought-ridden watershed show that the proposed model offers considerable advantages in accurately predicting the peak time of rains in real time.

7.2 Future Work

The future research work includes Rainfall prediction algorithm and model design and Rainfall prediction practice based on the feature set, algorithm, and model.

Reference:

- ▶ F. Fotovatikhah, M. Herrera, S. Shamshirband, K.-W. Chau, S. F. Ardabili, and M. J. Piran, "Survey of computational intelligence as basis to big flood management: Challenges, research directions and future work," *Eng. Appl. Comput. Fluid Mech.*, vol. 12, no. 1, pp. 411–437, Jan. 2018.
 - ▶ J. Reynolds, S. Halldin, C. Xu, J. Seibert, and A. Kauffeldt, "Sub-daily runoff predictions using parameters calibrated on the basis of data with a daily temporal resolution," *J. Hydrol.*, vol. 550, pp. 399–411, Jul. 2017.
 - ▶ Z. M. Yaseen, A. El-Shafie, O. Jaafar, H. A. Afan, and K. N. Sayl, "Artificial intelligence based models for stream-flow forecasting: 2000–2015," *J. Hydrol.*, vol. 530, pp. 829–844, Nov. 2015.
- 34
- ▶ J. Feng, L. Yan, and T. Hang, "Stream-flow forecasting based on dynamic spatio-temporal attention," *IEEE Access*, vol. 7, pp. 134754–134762, 2019.

- ▶ C.P.ChenandC.-Y.Zhang, “Data-intensiveapplications,challenges,techniquesand technologies: A survey on big data,” *Inf. Sci.*, vol. 275, pp. 314–347, Aug.2014.
- ▶ H. Gao, W. Huang, and X. Yang, “Applying probabilistic model checking to path planning in an intelligent transportation system using mobility trajectories and their statistical data,” *Intell. Automat. Soft Comput.*, vol. 25, no. 3, pp. 547–559,2019.
- ▶ H. Gao, W. Huang, Y. Duan, X. Yang, and Q. Zou, “Research on costdriven services compositioninanuncertainenvironment,”*J.InternetTechnol.*,vol.20,no.3,pp.755–769, 2019.
- ▶ H. Gao, Y. Duan, L. Shao, and X. Sun, “Transformation-based processing of typed resources for multimedia sources in the iot environment,” *Wireless Netw.*, pp. 1–17, Nov.2019.
- ▶ H. Mislán, S. Hardwinarto, and M. A. Sumaryono, “Rainfall monthly prediction based on artificial neural network: A case study in Tenggarong Station, East Kalimantan- Indonesia,” *Procedia Comput. Sci.*, vol. 59, pp. 142–151, Jan.2015.
- ▶ Ddubey, “Artificial neural network models for rainfall prediction in pondicherry,” *Int. J. Comput. Appl.*, vol. 120, no. 3, pp. 30–35, Jun.2015.
- ▶ N. Sharma, M. Zakaullah, H. Tiwari, and D. Kumar, “Runoff and sediment yield modeling using ANN and support vector machines: A case study from Nepal watershed,” *Model. Earth Syst. Environ.*, vol. 1, no. 3, p. 23,2015
- ▶ S. Das, R. Chakraborty, and A. Maitra, “A random forest algorithm for nowcasting of intense precipitation events,” *Adv. Space Res.*, vol. 60, no. 6, pp. 1271–1282, Sep. 2017.
- ▶ Y. Zaman, “Machine learning model on rainfall—A predicted approach for Bangladesh,” *Ph.D. dissertation, United Int. Univ., Dhaka, Bangladesh*,2018.

Screen Shots:

```

File Edit View Insert Cell Kernel Help
+ + + + + Run
Out[43]: DecisionTreeClassifier(max_leaf_nodes=15, random_state=0)

In [44]: prediction = dtClassifier.predict(X_test)

In [45]: prediction[:100]

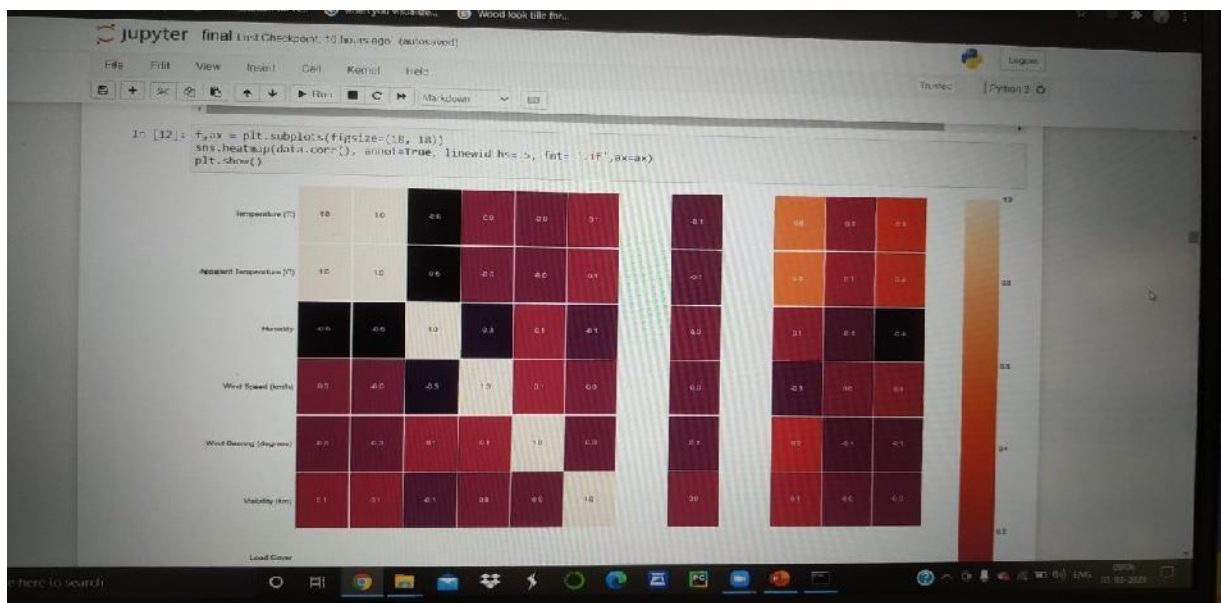
Out[45]: array([ 2., 13.,  2.,  2.,  6.,  6.,  2.,  1.,  2.,  2.,  2.,  6.,  2.,
  2., 17.,  2.,  2., 13.,  2.,  2.,  2.,  2., 17.,  2.,  2.,  2.,
 11.,  6.,  1., 17., 11.,  6.,  2.,  2., 17., 13.,  2.,  2., 17.,
  2.,  2., 13., 13., 17.,  2., 11.,  2., 12.,  2., 11.,  6.,  2.,
 17.,  2.,  2.,  6.,  1.,  3., 13., 17., 11.,  2., 11., 11.,  1.,
 12.,  2., 11., 28., 11.,  1., 13., 13.,  2., 17., 17.,  2.,  2.,
  2.,  1.,  6.,  1., 17.,  1.,  6.,  2.,  2., 17., 17., 17.,  2.,
 11., 13., 17., 17.,  2.,  1., 11.,  2.,  2.])

In [46]: y_test[:100]

Out[46]: 730    14.0
334      9.0
187      2.0
628      9.0
1179     6.0
...
1409     2.0
802      1.0
1095    11.0
545      6.0
1406     2.0
Name: DailySummaryCat, Length: 100, dtype: float64

In [47]: accuracy_score(y_true=y_test, y_pred=prediction)

```



Source Code:

```
import numpy as np

import pandas as pd

import tensorflow as tf

import glob

import folium #Interactive Maps viz

import seaborn as sns

import matplotlib.pyplot as plt

import matplotlib.dates as mdates

import math

from math import pi

import datetime as dt

import missingno as mn

from sklearn.impute import SimpleImputer as Imputer

from keras import backend as K

import keras

from keras.models import Sequential
```

```

from keras.layers import Dense, Dropout

from sklearn.model_selection import train_test_split

import datetime

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear_model import LogisticRegression

from tensorflow import keras

%matplotlib inline

plt.style.use('seaborn-whitegrid')


from mpl_toolkits.mplot3d import Axes3D

from sklearn.preprocessing import StandardScaler

from collections import Counter

import time

import datetime as dt

from datetime import datetime

import collections

```

```

import os # accessing directory structure

import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import seaborn as sns

from matplotlib.pyplot import rcParams

from sklearn import linear_model

from datetime import datetime

data = pd.read_csv('weatherHistory.csv', nrows=1464, parse_dates=[0])

data.head()

len(data)

data.tail(100)

data.dtypes

categorical = data.select_dtypes(include = ["object"]).keys()

print(categorical)

quantitative = data.select_dtypes(include = ["int64","float64"]).keys()

print(quantitative)

data['Formatted Date'].isnull().sum()

f,ax = plt.subplots(figsize=(18, 18))

sns.heatmap(data.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)plt.show()

```

```
data[quantitative].describe()
```

```
rcParams['figure.figsize'] = 9, 9
```

```
data[quantitative].hist()
```

```
data=data.drop('Loud Cover',axis=1)
```

```
pressure_median = data['Pressure (millibars)'].median()
```

```
def pressure(x):
```

```
    if x==0:
```

```
        return x + pressure_median
```

```
    else:
```

```
        return x
```

```
data["Pressure (millibars)"] = data.apply(lambda row:pressure(row["Pressure  
(millibars)"]), axis = 1)
```

```
rcParams['figure.figsize'] = 5, 3
```

```
data['Pressure (millibars)'].hist()
```

```
rcParams['figure.figsize'] = 8, 5sns.countplot(y=data['Summary'])
```

```

data['Daily Summary'].value_counts(dropna=False)

data['Summary'].value_counts(dropna=False)

data['Precip Type'].value_counts(dropna=False)

data.fillna(method='ffill', inplace=True)

sns.countplot(x=data['Precip Type'])

totalCells = np.product(data.shape)

missingCount = data.isnull().sum()

totalMissing = missingCount.sum()

print("The weather history dataset contains", round(((totalMissing/totalCells) * 100), 2),
"%", "missing values.")

plt.figure(figsize=(18,8)) # this creates a figure 8 inch wide, 4 inch high

sns.pairplot(data[['Apparent Temperature (C)', 'Temperature (C)', 'Humidity' ]])

plt.show()

plt.figure(figsize=(18,8))

sns.pairplot(data[['Wind Speed (km/h)', 'Wind Bearing (degrees)', 'Visibility (km)',
'Pressure (millibars)']])plt.show()

sns.violinplot(x="Precip Type", y="Temperature (C)", data=data, palette="YlGnBu");

sns.violinplot(x="Precip Type", y="Humidity", data=data, palette="YlGnBu");

```

```

from sklearn import linear_model

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error, r2_score


ls = linear_model.LinearRegression()

X = data["Humidity"].values.reshape(-1,1)

y = data["Temperature (C)"].values.reshape(-1,1)

X_train, X_test, y_train, y_test = train_test_split(X,

                                                    y,

                                                    test_size=0.33,

                                                    shuffle=True, random_state=0)

print("Linear Regression")

ls.fit(X_train, y_train)

print("alpha = ",ls.coef_[0])print("beta = ",ls.intercept_)

print("\n\nCalculating some regression quality metrics")

y_pred = ls.predict(X_test)

print("MSE = ",mean_squared_error(y_test, y_pred))

print("R2 = ",r2_score(y_test, y_pred))

```

```
def change_category_to_number(DailySummaryCat):  
  
    if DailySummaryCat=='Partly cloudy throughout the day.':  
  
        return 1  
  
    elif DailySummaryCat=='Mostly cloudy throughout the day.':  
  
        return 2  
  
    elif DailySummaryCat=='Foggy in the evening.':  
  
        return 3  
  
    elif DailySummaryCat=='Foggy overnight and breezy in the morning.':  
  
        return 4  
  
    elif DailySummaryCat=='Overcast throughout the day.':  
  
        return 5  
  
    elif DailySummaryCat=='Partly cloudy until night.':  
  
        return 6  
  
    elif DailySummaryCat=='Motly cloudy until night.':  
  
        return 7  
  
    elif DailySummaryCat=='Foggy starting overnight continuing until morning.':  
  
        return 8  
  
    elif DailySummaryCat=='Foggy in the morning.':return 9
```

```

elif DailySummaryCat=='Partly cloudy until evening.':

    return 10

elif DailySummaryCat=='Partly cloudy starting in the morning.':

    return 11

elif DailySummaryCat=='Mostly cloudy starting overnight continuing until night.':

    return 12

elif DailySummaryCat=='Partly cloudy starting in the afternoon.':

    return 13

elif DailySummaryCat=='Partly cloudy starting overnight.':

    return 14

elif DailySummaryCat=='Mostly cloudy starting overnight.':

    return 15

elif DailySummaryCat=='Mostly cloudy until night and breezy in the afternoon.':

    return 16

elif DailySummaryCat=='Mostly cloudy until evening.':

    return 17

elif DailySummaryCat=='Foggy throughout the day.':

    return 18
elif DailySummaryCat=='Partly cloudy starting in the morning.':

```

return 19

elif DailySummaryCat=='Partly cloudy starting in the morning continuing until evening.':

return 20

elif DailySummaryCat=='Foggy until morning.':

return 21

elif DailySummaryCat=='Partly cloudy starting in the morning continuing until night.':

return 22

elif DailySummaryCat=='Mostly cloudy starting in the morning.':

return 23

elif DailySummaryCat=='Foggy starting in the evening.':

return 24

elif DailySummaryCat=='Partly cloudy starting in the afternoon continuing until evening.':

return 25

elif DailySummaryCat=='Foggy overnight.':

return 26

elif DailySummaryCat=='Clear throughout the day.':return 27

elif DailySummaryCat=='Partly cloudy starting overnight continuing until night.':

return 28

elif DailySummaryCat=='Partly cloudy overnight.':

return 29

elif DailySummaryCat=='Partly cloudy starting overnight continuing until evening.':

return 30

elif DailySummaryCat=='Foggy until night.':

return 31

elif DailySummaryCat=='Partly cloudy in the morning.':

return 32

elif DailySummaryCat=='Foggy starting overnight continuing until afternoon.':

return 33

elif DailySummaryCat=='Foggy until afternoon.':

return 34

elif DailySummaryCat=='Breezy and mostly cloudy overnight.':

return 35

elif DailySummaryCat=='Partly cloudy overnight and breezy starting in the morning
continuing until afternoon.':return 36

```
elif DailySummaryCat=='Breezy in the morning and foggy in the evening.':
```

```
    return 37
```

```
elif DailySummaryCat=='Mostly cloudy until evening and breezy in the evening.':
```

```
    return 38
```

```
elif DailySummaryCat=='Mostly cloudy starting in the evening.':
```

```
    return 39
```

```
elif DailySummaryCat=='Mostly cloudy throughout the day and breezy starting  
overnight continuing until afternoon.':
```

```
    return 40
```

```
elif DailySummaryCat=='Breezy starting in the morning continuing until night.':
```

```
    return 41
```

```
elif DailySummaryCat=='Overcast throughout the day and breezy starting overnight  
continuing until morning.':
```

```
    return 42
```

```
elif DailySummaryCat=='Breezy starting overnight continuing until morning and foggy  
in the evening.':
```

```
    return 43
```

```
elif DailySummaryCat=='Light rain until morning.'return 44
```

elif DailySummaryCat=='Mostly cloudy until night and breezy starting in the afternoon continuing until night.':

return 45

elif DailySummaryCat=='Mostly cloudy starting in the morning continuing until afternoon.':

return 46

elif DailySummaryCat=='Breezy until afternoon and overcast throughout the day.':

return 47

elif DailySummaryCat=='Partly cloudy until evening and breezy in the afternoon.':

return 48

elif DailySummaryCat=='Breezy starting overnight continuing until morning and partly cloudy starting overnight continuing until evening.':

return 49

elif DailySummaryCat=='Light rain starting overnight.':

return 50

elif DailySummaryCat=='Partly cloudy starting overnight continuing until evening and breezy starting in the morning continuing until evening.':

return 51 elif DailySummaryCat=='Foggy starting in the morning continuing until evening and breezy in the evening.':

return 52

elif DailySummaryCat=='Partly cloudy throughout the day and breezy in the afternoon.':

return 53

elif DailySummaryCat=='Mostly cloudy starting overnight continuing until evening and breezy starting overnight continuing until morning.':

return 54

elif DailySummaryCat=='Partly cloudy starting overnight continuing until evening and breezy in the morning.':

return 55

elif DailySummaryCat=='Overcast throughout the day and breezy overnight.':

return 56

elif DailySummaryCat=='Light rain in the morning.':

return 57

elif DailySummaryCat=='Rain until morning.':

return 58

elif DailySummaryCat=='Breezy in the morning and mostly cloudy starting in the evening.':return 59elif DailySummaryCat=='Mostly cloudy starting in the morning and breezy overnight.':

return 60

elif DailySummaryCat=='Partly cloudy starting overnight and breezy starting in the morning continuing until afternoon.':

return 61

elif DailySummaryCat=='Partly cloudy starting in the morning and breezy starting in the afternoon continuing until evening.':

return 62

elif DailySummaryCat=='Partly cloudy starting in the morning continuing until evening and breezy in the afternoon.':

return 63

elif DailySummaryCat=='Foggy starting overnight continuing until morning and breezy in the afternoon.':

return 64

```
data['DailySummaryCat'] = data['Daily Summary'].apply(change_category_to_number)

data.fillna(method='ffill', inplace=True)

plt.figure(figsize=(18,8))

ax = sns.countplot(x=data['DailySummaryCat'])

ax.set_xticklabels(ax.get_xticklabels(), rotation=90, ha="right")

plt.tight_layout()

plt.show()

def change_category(Summary):

    if Summary=='Partly Cloudy':

        return 1

    elif Summary=='Mostly Cloudy':

        return 2

    elif Summary=='Foggy':

        return 3

    elif Summary=='Clear':

        return 4

    elif Summary=='Overcast':

        return 5 elif Summary=='Breezy and Overcast':
```

return 6

elif Summary=='Breezy and Partly Cloudy':

return 7

elif Summary=='Breezy and Mostly Cloudy':

return 8

elif Summary=='Dry and Partly Cloudy':

return 9

elif Summary=='Windy and Partly Cloudy':

return 10

elif Summary=='Light Rain':

return 11

elif Summary=='Breezy':

return 12

elif Summary=='Windy and Overcast':

return 13

elif Summary=='Humid and Mostly Cloudy':

return 14elif Summary=='Drizzle':

return 15

elif Summary=='Windy and Mostly Cloudy':

return 16

elif Summary=='Breezy and Foggy':

return 17

elif Summary=='Dry':

return 18

elif Summary=='Humid and Partly Cloudy':

return 19

elif Summary=='Dry and Mostly Cloudy':

return 20

elif Summary=='Rain':

return 21

elif Summary=='Windy':

return 22

elif Summary=='Humid and Overcast':

return 23

elif Summary=='Windy and Foggy':

return 24elif Summary=='Dangerously Windy and Partly Cloudy':

```
return 25
```

```
    elif Summary=='Windy and Dry':
```

```
        return 26
```

```
    elif Summary=='Breezy and Dry':
```

```
        return 27
```

```
data['SummaryCat'] = data['Summary'].apply(change_category)
```

```
plt.figure(figsize=(18,8))
```

```
ax = sns.countplot(x=data['SummaryCat'])
```

```
ax.set_xticklabels(ax.get_xticklabels(), rotation=90, ha="right")
```

```
plt.tight_layout()
```

```
plt.show()
```

```
def change_category(PrecipTypeCat):
```

```
    if PrecipTypeCat=='rain':
```

```
        return 1
```

```
    elif PrecipTypeCat=='snow':
```

```
        return 2
```

```
data['PrecipTypeCat'] = data['Precip Type'].apply(change_category)
```

```

plt.figure(figsize=(18,8))

ax = sns.countplot(x=data['PrecipTypeCat'])

ax.set_xticklabels(ax.get_xticklabels(), rotation=90, ha="right")

plt.tight_layout()

plt.show()

data.drop(['Formatted Date','Summary','Daily Summary','Precip
Type','Date'],axis=1,inplace=True)

data.head()

X = data.drop('DailySummaryCat', axis=1)

y = data['DailySummaryCat']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=128)

dtClassifier = DecisionTreeClassifier(max_leaf_nodes=15,random_state=0)

dtClassifier.fit(X_train, y_train)

prediction = dtClassifier.predict(X_test)

prediction[:100]

y_test[:100]

accuracy_score(y_true=y_test, y_pred=prediction)

lrClassifier = LogisticRegression()lrClassifier.fit(X_train,y_train)

```

```
prediction = lrClassifier.predict(X_test)
```

```
prediction[:3000]
```

```
y_test[:100]
```

```
accuracy_score(y_true=y_test, y_pred=prediction)
```

```
def facet_heatmap(data, color, **kws):
```

```
    values=data.columns.values[3]
```

```
    data = data.pivot(index='day', columns='hour', values=values)
```

```
    sns.heatmap(data, cmap='coolwarm', **kws)
```

```
#Joining heatmaps of every month in a year
```

```
def weather_calendar(year,weather): #Year= Any year in DataFrame. Weather=Any
quantitative variable
```

```
    datayear = data[data['year']==year][['month', 'day', 'hour', weather]]
```

```
    vmin=datayear[weather].min()
```

```
    vmax=datayear[weather].max()
```

```
    print(vmin,          vmax)with          sns.plotting_context(font_scale=12):g          =
```

```
sns.FacetGrid(datayear,col="month",          col_wrap=3)          g          =
```

```
g.map_dataframe(facet_heatmap,vmin=vmin, vmax=vmax)
```

```
g.set_axis_labels('Hour', 'Day')

plt.subplots_adjust(top=0.9)

g.fig.suptitle('%s Calendar. Year: %s.' %(weather, year), fontsize=18)

weather_calendar(2006, 'Temperature (C)')

weather_calendar(2006, 'Humidity')

data_neur=data

data_neur['T'] = data['Temperature (C)']

data_neur['TA'] = data['Apparent Temperature (C)']

data_neur['H'] = data['Humidity']

data_neur['WS'] = data['Wind Speed (km/h)']

data_neur['WB'] = data['Wind Bearing (degrees)']

data_neur['V'] = data['Visibility (km)']

data_neur['P'] = data['Pressure (millibars)']

data_neur['Y'] = data['year']

data_neur['M'] = data['month']

data_neur['D'] = data['day']

data_neur['HR'] = data['hour']
```

```
data_neur['SC'] = data['SummaryCat']
```

```
data_neur['PT'] = data['PrecipTypeCat']
```

```
data.drop(['hour','day','month','year','Pressure (millibars)', 'Temperature (C)', 'Apparent  
Temperature (C)', 'Humidity', 'Wind Speed (km/h)', 'Wind Bearing (degrees)', 'Visibility  
(km)', 'SummaryCat', 'PrecipTypeCat'], axis=1, inplace=True)
```

```
data_neur.head()
```

```
X = data_neur[[col for col in data_neur.columns if col != 'DailySummaryCat']]
```

```
y = data_neur['DailySummaryCat']
```

```
X_train, X_tmp, y_train, y_tmp = train_test_split(X, y, test_size=0.2, random_state=23)
```

```
X_test, X_val, y_test, y_val = train_test_split(X_tmp, y_tmp, test_size=0.5,  
random_state=23)
```

```
X_train.shape, X_test.shape, X_val.shape
```

```
print("Training instances {}, Training features {}".format(X_train.shape[0],  
X_train.shape[1]))
```

```
print("Validation instances {}, Validation features {}".format(X_val.shape[0],  
X_val.shape[1]))
```

```
print("Testing instances {}, Testing features  
{}".format(X_test.shape[0],X_test.shape[1]))
```

58

```
feature_cols = [tf.feature_column.numeric_column(col) for col in X.columns]
```

```
regressor = tf.estimator.DNNRegressor(feature_columns=feature_cols,
```

```
    hidden_units=[50, 50],
```

```
    model_dir='tf_wx_model')
```

```
def wx_input_fn(X, y=None, num_epochs=None, shuffle=True, batch_size=260): # 260
```

is used as we have approx 570 dataset for training

```
    return tf.compat.v1.estimator.inputs.pandas_input_fn(x=X,
```

```
        y=y,
```

```
        num_epochs=num_epochs,
```

```
        shuffle=shuffle,
```

```
        batch_size=batch_size)
```

```
evaluations = []
```

```
STEPS = 260
```

```
for i in range(100):
```

```
    regressor.train(input_fn=wx_input_fn(X_train, y=y_train), steps=STEPS)
```

```
    evaluation = regressor.evaluate(input_fn=wx_input_fn(X_val, y_val,
```

```
num_epochs=1,  
shuffle=False)steps=1)
```

59

```
evaluations.append(regressor.evaluate(input_fn=wx_input_fn(X_val,  
y_val,  
num_epochs=1,  
shuffle=False)))
```

```
plt.rcParams['figure.figsize'] = [14, 10]
```

```
loss_values = [ev['loss'] for ev in evaluations]
```

```
training_steps = [ev['global_step'] for ev in evaluations]
```

```
plt.scatter(x=training_steps, y=loss_values)
```

```
plt.xlabel('Training steps (Epochs = steps / 2)')
```

```
plt.ylabel('Loss (SSE)')
```

```
plt.show()
```

