# WEATHER FORECASTING

## A PROJECT REPORT

*In partial fulfilment of the requirements for the award of the degree*

**BACHELOR OF TECHNOLOGY**

**IN**

**ELECTRICAL ENGINEERING**

*Under the guidance of*

**MAHENDRA DATTA**

**BY**

1. **AMEYA KUMAR NATH**
2. **SHRAMANA BOSE**
3. **RUPAK CHAKRABORTY**
4. **ANKIT SINHA**

**FUTURE INSTITUTE OF ENGINEERING AND MANAGEMENT**

**In association with**

**(ISO9001:2015)**

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

***(Note: All entries of the Performa of approval should be filled up with appropriate and complete information. Incomplete Performa of approval in any respect will be summarily rejected.)***

| | | |
|---|---|---|
| 1. | Title of the Project: | **WEATHER FORECASTING** |
| 2. | Project Members: | **1. AMEYA KUMAR NATH**<br>**2. SHRAMANA BOSE**<br>**3. RUPAK CHAKRABORTY**<br>**4. ANKIT SINHA** |
| 3. | Name of the guide: | **Mr. MAHENDRA DUTTA** |
| 4. | Address: | Ardent Computech Pvt. Ltd<br>(An ISO 9001:2015 Certified)<br>SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal, 700091 |

***Project Version Control History***

| Version | Primary Author | Description of Version | Date Completed |
|---|---|---|---|
| Final | **1. AMEYA KUMAR NATH**<br>**2. SHRAMANA BOSE**<br>**3. RUPAK CHAKRABORTY**<br>**4. ANKIT SINHA** | Project Report | 27TH JANUARY,2024 |

Signature of Team Member

Signature of Approver

Date:

Date:

For Office Use Only

**MR. MAHENDA DUTTA**

Project Proposal Evaluator

**Approved** **Not Approved**

# DECLARATION

We hereby declare that the project work being presented in the project proposal entitled **"WEATHER FORECASTING"** in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** at **ARDENT COMPUTECH PVT. LTD, SALTLAKE, KOLKATA, WEST BENGAL,** is an authentic work carried out under the guidance of **MR. MAHENDRA DUTTA**. The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date: 27TH JANUARY,2024

Name of the Student: **1. AMEYA KUMAR NATH**
**2. SHRAMANA BOSE**
**3. RUPAK CHAKRABORTY**
**4. ANKIT SINHA**

***Signature of the students:***

**Ardent ComputechPvt. Ltd (An ISO 9001:2015 Certified)**

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

# CERTIFICATE

This is to certify that this proposal of minor project entitled **"WEATHER FORECASTING"** is a record of bonafide work, carried out by ***AMEYA KUMAR NATH , SHRAMANA BOSE , RUPAK CHAKRABORTY , ANKIT SINHA* ,** under my guidance at **ARDENT COMPUTECH PVT LTD**. In my opinion, the report in its present form is in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** and as per regulations of the **ARDENT®.** To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report.

**Guide / Supervisor**

----------------------------------------------

**MR. MAHENDRA DUTTA**

Project Engineer

Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

## ACKNOWLEDGEMENT

Success of any project depends largely on the encouragement and guidelines of many others. We take this sincere opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project work.

We would like to show our greatest appreciation to ***Mr. MAHENDRA DUTTA***, Project Engineer at Ardent, Kolkata. We always feel motivated and encouraged every time by his valuable advice and constant inspiration, without his encouragement and guidance this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Ardent Computech Pvt. Ltd. for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

# CONTENTS

- Overview
- History of Python
- Environment Setup
- Basic Syntax
- Variable Types
- Functions
- Artificial Intelligence
    - Machine Learning
- Machine Learning
    - Supervised and Unsupervised Learning
    - NumPy
    - SciPy
    - Scikit-learn
    - Pandas
    - Regression Analysis
    - Matplotlib
    - Clustering
- Weather Forecasting

1. Abstract
2. Introduction
3. Literature Review
4. Methodology & Algorithms
5. Advantages & Disadvantages
6. Codes
7. Future Scope
8. Conclusion
9. Bibliography

## OVERVIEW

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and has fewer syntactical constructions than other languages.

**Python is interpreted**: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to Perl and PHP.

**Python is Interactive**: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

**Python is Object-Oriented**: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

**Python is a Beginner's Language**: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

# HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, UNIX shell, and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

## FEATURES OF PYTHON

**Easy-to-learn**: Python has few Keywords, simple structure and clearly defined syntax. This allows a student to pick up the language quickly.

**Easy-to-Read**: Python code is more clearly defined and visible to the eyes.

**Easy -to-Maintain:** Python's source code is fairly easy-to-maintain.

**A broad standard library**: Python's bulk of the library is very portable and cross platform compatible on UNIX, Windows, and Macintosh.

**Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

**Portable**: Python can run on the wide variety of hardware platforms and has the same interface on all platforms.

**Extendable**: You can add low level modules to the python interpreter. These modules enables programmers to add to or customize their tools to be more efficient.

**Databases:** Python provides interfaces to all major commercial databases.

**GUI Programming**: Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

**Scalable:** Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- It support functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte code for building large applications.
- It provides very high level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collections.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA and JAVA.

## ENVIRONMENT SETUP

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- UNIX (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2
- DOS (multiple versions)
- PalmOS
- Nokia mobile phones
- Windows CE
- Acorn/RISC OS

# BASIC SYNTAX OF PYTHON PROGRAM

Type the following text at the Python prompt and press the Enter –

```
>>> print "Hello, Python!"
```

*If you are running new version of Python, then you would need to use print statement with parenthesis* as in **print ("Hello, Python!");**.
However in Python version 2.4.3, this produces the following result –

```
Hello, Python!
```

## Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).
Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language.

## Python Keywords

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

**And, exec, not**
**Assert, finally, or**
**Break, for, pass**
**Class, from, print**
**continue, global, raise**
**def, if, return**
**del, import, try**
**elif, in, while**
**else, is, with**
**except, lambda, yield**

## Lines & Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced. The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True:
print "True"
else:
print "False"
```

## Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

```
$ python-h
usage: python [option]...[-c cmd|-m mod | file |-][arg]...

Options and arguments (and corresponding environment variables):

-c cmd: program passed in as string(terminates option list)

-d      : debug output from parser (also PYTHONDEBUG=x)

-E      : ignore environment variables (such as PYTHONPATH)

-h      : print this help message and exit [ etc.]
```

# VARIABLE TYPES

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

## Assigning Values to Variables

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

```
Counter =10             # An integer assignment
Weight =10.60           # A floating point
Name ="Ardent"           # A string
```

## Multiple Assignment

Python allows you to assign a single value to several variables simultaneously. For example −

```
a = b = c = 1
a,b,c = 1,2,"hello"
```

## Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has five standard data types −

-  String
-  List
-  Tuple
-  Dictionary
-  Number

## Data Type Conversion

Sometimes, you may need to perform conversions between the built-in types. To convert between types, you simply use the type name as a function.

There are several built-in functions to perform conversion from one data type to another.

| Sr.No. | Function & Description |
|---|---|
| 1 | **int(x [,base])**<br><br>Converts x to an integer. base specifies the base if x is a string |
| 2 | **long(x [,base] )**<br><br>Converts x to a long integer. base specifies the base if x is a string. |
| 3 | **float(x)**<br><br>Converts x to a floating-point number. |
| 4 | **complex(real [,imag])**<br><br>Creates a complex number. |
| 5 | **str(x)**<br><br>Converts object x to a string representation. |
| 6 | **repr(x)**<br><br>Converts object x to an expression string. |
| 7 | **eval(str)**<br><br>Evaluates a string and returns an object. |
| 8 | **tuple(s)**<br><br>Converts s to a tuple. |
| 9 | **list(s)**<br><br>Converts s to a list. |

# FUNCTIONS

## Defining a Function

-
  ```
  deffunction name( parameters ):
      "function_docstring"
      function suite
      return [expression]
  ```

## Pass by reference vs Pass by value

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example –

```
# Function definition is here

defchange me(mylist):
        "This changes a passed list into this function"
        mylist.append([1,2,3,4]);
        print"Values inside the function: ",mylist
        return

# Now you can call change me function

mylist=[10,20,30];
change me(mylist);
print" Values outside the function: ",mylist
```

Here, we are maintaining reference of the passed object and appending values in the same object. So, this would produce the following result −

```
Values inside the function: [10, 20, 30, [1, 2, 3, 4]]
Values outside the function: [10, 20, 30, [1, 2, 3, 4]]
```

## Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope . For Example-

```
total=0;                    # This is global variable.

# Function definition is here

defsum( arg1, arg2 ):

# Add both the parameters and return them."

total= arg1 + arg2; # Here total is local variable.
print"Inside the function local total: ", total
return total;

# Now you can call sum function

sum(10,20);
Print”Outside the function global total:", total
```

When the above code is executed, it produces the following result −

```
Inside the function local total: 30
Outside the function global total: 0
```

# MODULES

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here's an example of a simple module, support.py

```
defprint_func( par ):
        print"Hello : ", par
        return
```

## The *import* Statement

You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* has the following syntax –

```
Importmodule1 [, module2 […moduleN]
```

# PACKAGES

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-sub packages, and so on.

Consider a file *Pots.py* available in *Phone* directory. This file has following line of source code −

```
defPots ():
        print "I'm Pots Phone"
```

Similar way, we have another two files having different functions with the same name as above –

-  *Phone/Isdn.py* file having function Isdn ()
-  *Phone/G3.py* file having function G3 ()

Now, create one more file __init__.py in *Phone* directory −

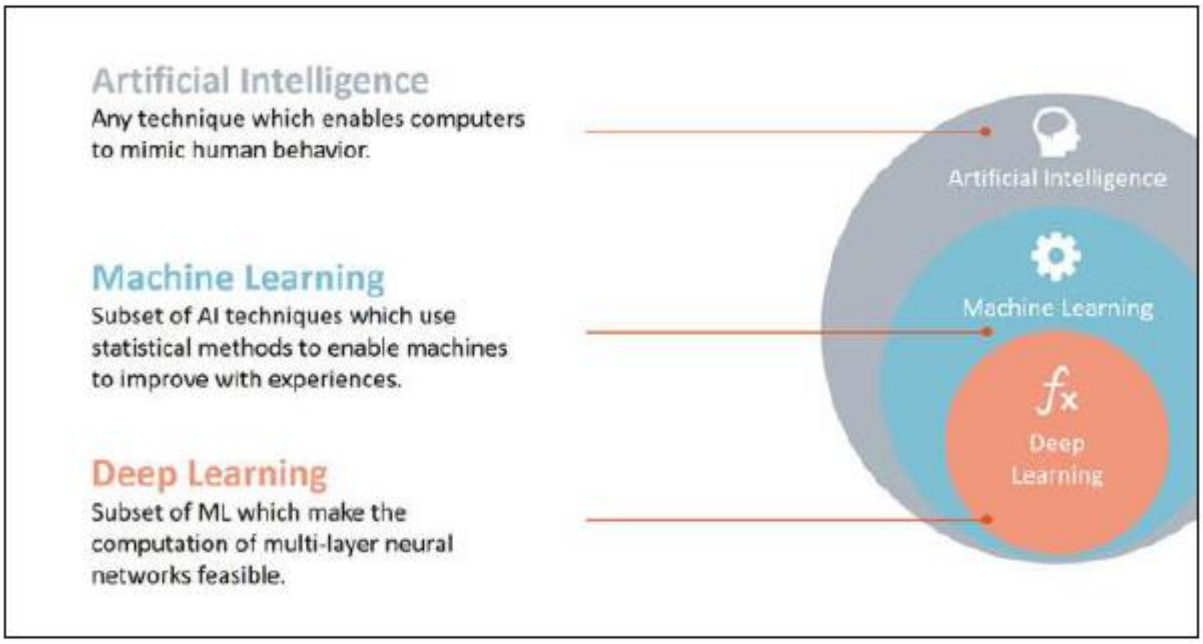
-  Phone/__init__.py

To make all of your functions available when you've imported Phone, you need to put explicit import statements in __init__.py as follows −

```
from Pots import Pots
from Isdn import Isdn
from G3 import
```

# ARTIFICIAL INTELLIGENCE

## Introduction


Artificial Intelligence
Any technique which enables computers to mimic human behavior.
Machine Learning
Subset of AI techniques which use statistical methods to enable machines to improve with experiences.
Deep Learning
Subset of ML which make the computation of multi-layer neural networks feasible.
Artificial Intelligence
Machine Learning
$f_x$
Deep Learning

According to the father of Artificial Intelligence, John McCarthy, it is *“The science and engineering of making intelligent machines, especially intelligent computer programs”.*

Artificial Intelligence is a way of **making a computer, a computer-controlled robot, or a software think intelligently**, in the similar manner the intelligent humans think.

AI is accomplished by studying how human brain thinks , and how humans learn, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems.

The development of AI started with the intention of creating similar intelligence in machines that we find and regard high in humans.

## Goals of AI

**To Create Expert Systems** − The systems which exhibit intelligent behaviour, learn, demonstrate, explain, and advice its users.
**To Implement Human Intelligence in Machines** − Creating systems that understand, think, learn, and behave like humans.

## Applications of AI

AI has been dominant in various fields such as:-

**Gaming** − AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.

**Natural Language Processing** − It is possible to interact with the computer that understands natural language spoken by humans.

**Expert Systems** − There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.

**Vision Systems** − These systems understand, interpret, and comprehend visual input on the computer.

For example: A spying aeroplane takes photographs, which are used to figure out spatial information
Or map of the areas.

Doctors use clinical expert system to diagnose the patient.

Police use computer software that can recognize the face of criminal with the stored
portrait made by forensic artist.

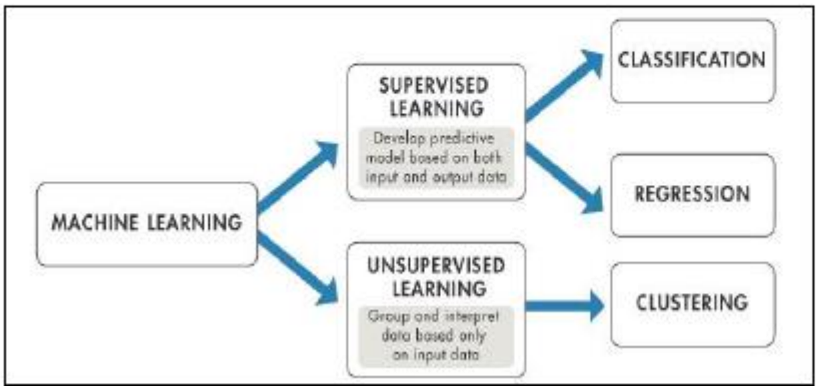
**Speech Recognition** − Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human’s noise due to cold, etc.

**Handwriting Recognition** − The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.

**Intelligent Robots** − Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.

# Application of AI

## MACHINE LEARNING


MACHINE LEARNING
SUPERVISED LEARNING
Develop predictive model based on both input and output data
UNSUPERVISED LEARNING
Group and interpret data based only on input data
CLASSIFICATION
REGRESSION
CLUSTERING

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.
Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data.

# INTRODUCTION TO MACHINE LEARNING

**Machine learning** is a field of computer science that gives computers the ability to learn without being explicitly programmed.

**Arthur Samuel**, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data

Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system:-

## SUPERVISED LEARNING

**Supervised learning** is the machine learning task of inferring a function from *labelled training data*.[1] The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value.

A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

## UNSUPERVISED LEARNING

**Unsupervised learning** is the machine learning task of inferring a function to describe hidden structure from "unlabelled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabelled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning and reinforcement learning.

A central case of unsupervised learning is the problem of density estimation in statistics, though unsupervised learning encompasses many other problems (and solutions) involving summarizing and explaining key features of the data.

## NUMPY

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin.

NumPy targets the Python reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

## NUMPY ARRAY

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

For example, the coordinates of a point in 3D space [1, 2, 1] is an array of rank 1, because it has one axis. That axis has a length of 3. In the example pictured below, the array has rank 2 (it is 2-dimensional). The first dimension (axis) has a length of 2, the second dimension has a length of 3.

```
[[1., 0., 0.],
 [ 0., 1., 2.]]
```

NumPy's array class is called *ndarray*. It is also known by the alias.

## SLICING NUMPY ARRAY

```
Import numpy as np

a = np.array ([[1, 2, 3],[3,4,5],[4,5,6]])

print 'Our array is:'
Print a
print '\n'


print 'The items in the second column are:'
print a[...,1]
print '\n'


print 'The items in the second row are:'
print a[1...]
print '\n'


print 'The items columns 1 onwards are:'
print a [...,1:]
```

**OUTPUT**

```
Our array is:
[[1 2 3]
[3 4 5]
[4 5 6]]
```

```
The items in the second column are:
[2 4 5]

The items in the second row are:
[3 4 5]

The items column 1 onwards are:
[[2 3]
[4 5]
[5 6]]
```

## SCIPY

modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy, and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as MATLAB, GNU Octave, and Scilab. The NumPy stack is also sometimes referred to as the SciPy stack.

### The SciPy Library/Package

The SciPy package of key algorithms and functions core to Python's scientific computing capabilities. Available sub-packages include:

- **constants:** physical constants and conversion factors (since version 0.7.0)
- **cluster:** hierarchical clustering, vector quantization, K-means
- **fftpack:** Discrete Fourier Transform algorithms
- **integrate:** numerical integration routines
- **interpolate:** interpolation tools
- **io:** data input and output
- **lib:** Python wrappers to external libraries
- **linalg:** linear algebra routines
- **misc:** miscellaneous utilities (e.g. image reading/writing)
- **ndimage:** various functions for multi-dimensional image processing
- **optimize:** optimization algorithms including linear programming
- **signal:** signal processing tools
- **sparse:** sparse matrix and related algorithms
- **spatial:** KD-trees, nearest neighbours, distance functions
- **special:** special functions
- **stats:** statistical functions
- **weave:** tool for writing C/C++ code as Python multiline strings

### Data Structures

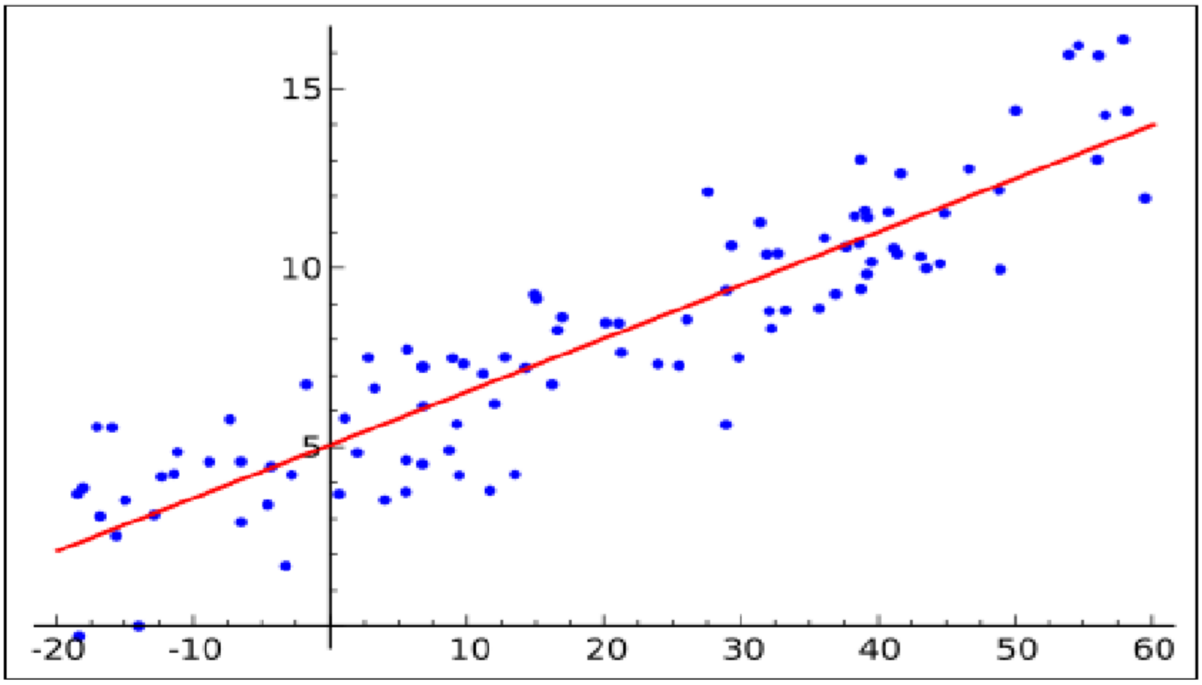
The basic data structure used by SciPy is a multidimensional array provided by the NumPy module. NumPy provides some functions for linear algebra, Fourier transforms and random number generation, but not with the generality of the equivalent functions in SciPy. NumPy can also be used as an efficient multi-dimensional container of data with arbitrary data-types. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. Older versions of SciPy used Numeric as an array type, which is now deprecated in favour of the newer NumPy array code.

## SCIKIT-LEARN

**Scikit-learn** is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

The scikit-learn project started as scikits.learn, a Google Summer of Code project by David Cournapeau. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy.[4] The original codebase was later rewritten by other developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel, all from INRIA took leadership of the project and made the first public release on February the 1st 2010[5]. Of the various scikits, scikit-learn as well as scikit-image were described as "well-maintained and popular" in November 2012.

## REGRESSION ANALYSIS



In statistical modelling, **regression analysis** is a set of statistical processes for estimating the relationships among variables. It includes many techniques for modelling and analysing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors'). More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed.

Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Regression analysis is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships. In restricted circumstances, regression analysis can be used to infer casual relationships between the independent and dependent variables. However this can lead to illusions or false relationships, so caution is advisable

## LINEAR REGRESSION

Linear regression is a linear approach for modelling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. The case of one explanatory variable is called *simple linear regression*. For more than one explanatory variable, the process is called *multiple linear regression*.

In linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called *linear models*.

## LOGISTIC REGRESSION

Logistic regression, or logit regression, or logit model [1] is a regression model where the dependent variable (DV) is categorical. This article covers the case of a binary dependent variable—that is, where the output can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Cases where the dependent variable has more than two outcome categories may be analysed in multinomial logistic regression, or, if the multiple categories are ordered, in ordinal logistic regression. In the terminology of economics, logistic regression is an example of a qualitative response/discrete choice model.

## POLYNOMIAL REGRESSION

Polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an $n^{th}$ degree polynomial in x.

Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y , denoted E( y | x ), and has been used to describe nonlinear phenomena such as the growth rate of tissues, the distribution of carbon isotopes in lake sediments, and the progression of disease epidemics.

Although *polynomial regression* fits a nonlinear model to the data, as a statistical estimation problem it is linear, in the sense that the regression function E(y | x) is linear in the unknown parameters that are estimated from the data.

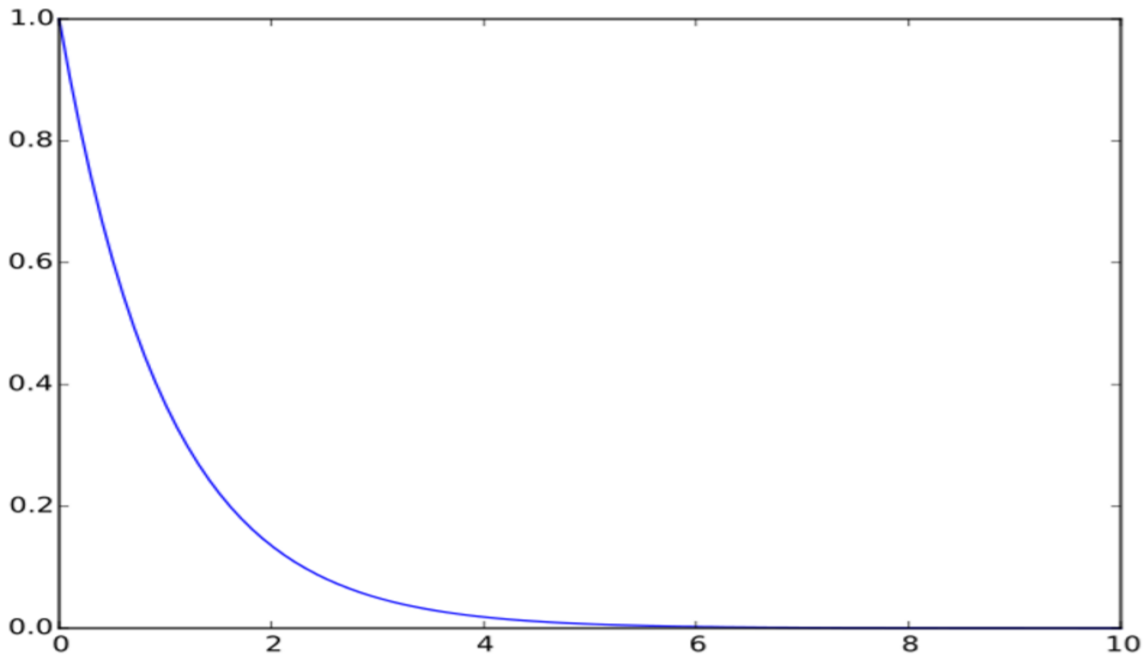
## MATPLOTLIB

**Matplotlib** is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter,wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged .SciPy makes use of matplotlib.

**EXAMPLE**

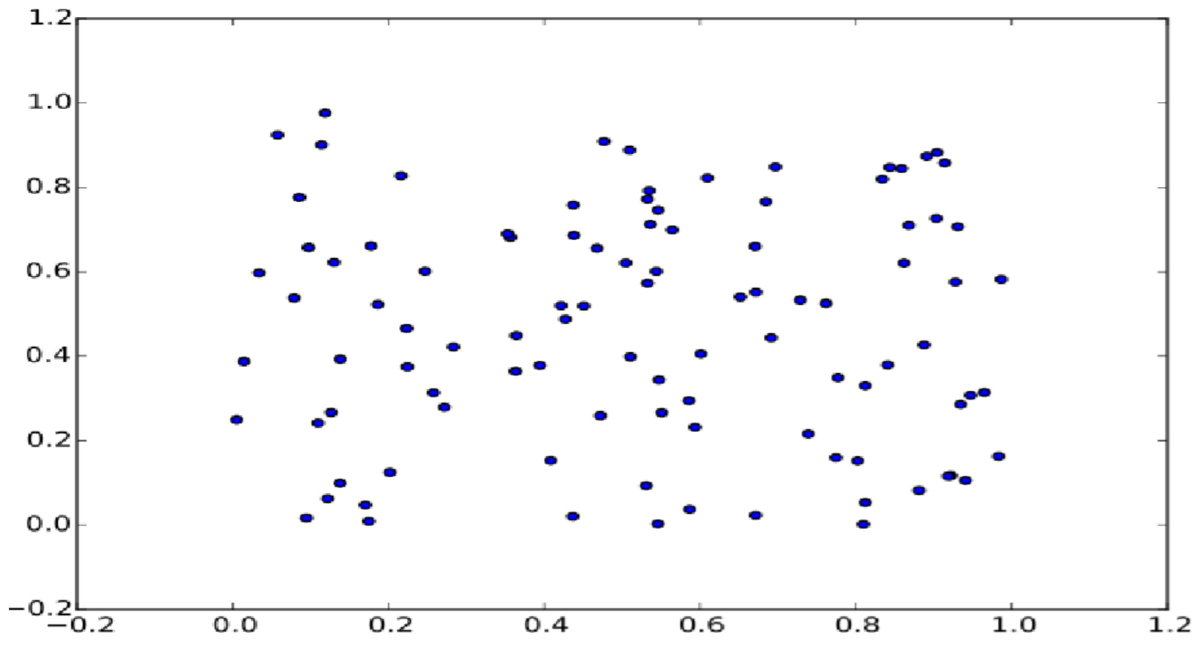
➢ **LINE PLOT**

```
>>>importmatplotlib.pyplotasplt
>>>importnumpyasnp
>>> a =np.linspace(0,10,100)
>>> b =np.exp(-a)
>>>plt.plot (a,b)
>>>plt.show ()
```



➢ **SCATTER PLOT**

```
>>>importmatplotlib.pyplotasplt
>>>fromnumpy.randomimport rand
>>> a =rand(100)
>>> b =rand(100)
>>>plt.scatter(a, b)
>>>plt.show ()
```



## PANDAS

In computer programming, **pandas** is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. "Panel data", an econometrics term for multidimensional, structured data sets.

**LIBRARY FEATURES**

- Data Frame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and sub setting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation.

## CLUSTERING

**Cluster analysis** or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

Cluster analysis itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with small distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. Clustering can therefore be formulated as a multi-objective optimization problem.

The appropriate clustering algorithm and parameter settings (including values such as the distance function to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results. Cluster analysis as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective optimization that involves trial and failure. It is often necessary to modify data pre-processing and model parameters until the result achieves the desired properties.

## ALGORITHM

- Data Collection
- Data Formatting
- Model Selection
- Training
- Testing

**Data Collection**: We have collected data sets of weather from online website. We have downloaded the .csv files in which information was present.

**Data Formatting**: The collected data is formatted into suitable data sets. We check the collinearity with mean temperature. The data sets which have collinearity nearer to 1.0 has been selected.

**Model Selection**: We have selected different models to minimize the error of the predicted value. The different models used are Linear Regression Model, RandomForest Classifier, Decision Tree,Support Vector Machine,KNN Classifier,.Gaussian NB.

**Training**: The data set was divided such that x_train is used to train the model with corresponding  x_test values and some y_train kept reserved for testing.

**Testing**: The model was tested with y_train and stored in y_predict. Both y_train and y_predict was compared.

# WEATHER FORECASTING

## ABSTRACT

Weather forecasts have grown increasingly significant in recent years since they can save us time, money, property, or even our lives. Despite the fact that India has a large number of weather stations, they are mainly located in inhabited regions such as cities, suburbs, or towns. This makes weather forecasting in isolated regions more imprecise, which can be inconvenient for individuals such as farmers who rely largely on weather reports in their daily work. In this paper, we are predicting the weather by analyzing features like temperature, apparent temperature, humidity, wind speed, wind bearing, visibility, cloud cover with Random Forest Classifier , Decision Tree ,KNN Classifier, Support Vector Machine(SVM), Linear regression, and Gaussian naive Bayes are examples of machine learning methods. Based on the results obtained a comparative study is done concerning the accuracy.

## INTRODUCTION

Historically, weather forecasting is done using physics-based models. These methods require high computational requirements and are sensitive to approximations of the physical laws on which they are based.

Applying machine learning to now casting, allows us to increase the accuracy and speed of making these predictions. Deep learning models can be built to find weather patterns of cloud behaviour by training it with satellite imagery. This means a model does not try to reproduce entire weather systems via simulations, but instead is trained to focus its compute power on seeing visual patterns from a mosaic of pixels. You can say this is similar to how our eyes work. Such advances have also enabled other projects such as Dynamic World, which helps granularly measure changes on the Earth.

Furthermore, pairing this technique with powerful data processing hardware like GPUs and TPUs, helps users get more accurate predictions at a fraction of the cost and time1.

**1.Precip Type:**

The main types of precipitation include drizzle, rain, sleet, snow, ice pellets, graupel, and hail. Precipitation happens when water vapour (reaching 100 percent relative humidity) saturates a portion of the atmosphere so that the water condenses and 'precipitates' or falls.

**2. Temperature:**

Air temperature is the state variable of the atmosphere and affects atmospheric and land surface processes . Forecasting air temperature is an important part of weather prediction because it is used to protect human lives and properties.

**3.Apparent Temperature:**

The heat index, also known as the apparent temperature, is what the temperature feels like to the human body when relative humidity is combined with the air temperature. his has important considerations for the human body's comfort. When the body gets too hot, it begins to perspire or sweat to cool itself off. If the perspiration is not able to evaporate, the body cannot regulate its temperature. Evaporation is a cooling process. When perspiration is evaporated off the body, it effectively reduces the body's temperature. When the atmospheric moisture content (i.e. relative humidity) is high, the rate of evaporation from the body decreases. In other words, the human body feels warmer in humid conditions. The opposite is true when the relative humidity decreases because the rate of perspiration increases. The body actually feels cooler in arid conditions. There is direct relationship between the air temperature and relative humidity and the heat index, meaning as the air temperature and relative humidity increase (decrease), the heat index increases (decreases).

**4. Humidity:**

The concentration of water vapour present in the air is known as humidity. The widely employed primary measurements of humidity are absolute, relative, and specific humidity. Humidity depends on the temperature and pressure of the system of interest. The same amount of water vapour results in higher relative humidity in cool air than warm air. Meteorologists typically describe water vapour in the atmosphere in two different ways: absolute humidity and relative humidity. Dew point is an absolute measure and helps us understand how muggy and humid it feels outside.

**5. Wind Speed:**

In meteorology, wind speed, or wind flow speed, is a fundamental atmospheric quantity caused by air moving from high to low pressure, usually due to changes in temperature. Wind speed is now commonly measured with an anemometer. Wind speed affects weather forecasting, aviation and maritime operations, construction projects, growth and metabolism rate of many plant species, and has countless other implications .Variations in large-scale wind circulation patterns are responsible for the daily weather we experience. Observations of wind speed and direction, along with other elements like temperature and moisture, are essential for estimating the state of the atmosphere at specific times and locations on Earth.

**6. Wind Bearing:**

Weather forecasts typically give the direction of the wind along with its speed, for example a "northerly wind at 15 km/h" is a wind blowing from the north at a speed of 15 km/h. If wind gusts are present, their speed may also be reported .The true direction from which the wind is blowing at a given location (i.e., wind blowing from the north to the south is a north wind). It is normally measured in tens of degrees from 10 degrees clockwise through 360 degrees. North is 360 degrees.

**7. Visibility:**

Visibility is a measure of the horizontal opacity of the atmosphere at the point of observation and is expressed in terms of the horizontal distance at which a person should be able to see and identify:  in the daytime, a prominent dark object against the sky at the horizon; at night, a known, preferably unfocused, moderately intense light source .Visibility distance is reduced by fog and heavy precipitation, as well as wind-blown snow, dust and smoke. Low visibility conditions cause increased speed variance, which increases crash risk. Each year, over 38,700 vehicle crashes occur in fog.

**8. Pressure:**

Atmospheric pressure is an indicator of weather. When a low-pressure system moves into an area, it usually leads to cloudiness, wind, and precipitation. High-pressure systems usually lead to fair, calm weather Changes in air pressure are caused by differences in air temperature above the earth, and the temperature of an air mass is determined by its location.

# LITERATURE REVIEW

There are many research papers that have been published related to predicting the weather. A paper was published on 'The Weather Forecast Using Data Mining Research Based on Cloud Computing' This paper proposes a modern method to develop a service oriented architecture for the weather information systems which forecast weather using these data mining techniques. This can be carried out by using Artificial Neural Network and Decision tree Algorithms and meteorological data collected in Specific time. Algorithm has presented the best results to generate classification rules for the mean weather variables. The results showed that these data mining techniques can be enough for weather forecasting. Another paper was published on 'Analysis on The Weather Forecasting and Techniques' where they decided that artificial neural network and concept of fuzzy logic provides a best solution and prediction comparatively . They decided to take temperature, humidity, pressure, wind and various other attributes into consideration. Another research paper titled 'Issues with weather prediction' discussed the major problems with weather prediction. Even the simplest weather prediction is not perfect. The one-day forecast typically falls within two degrees of the actual temperature. Although this accuracy isn't bad, as predictions are made for further in time. For example, in a place like New England where temperatures have a great variance the temperature prediction are more inaccurate than a place like the tropics. Another research paper titled 'Current weather prediction' used numerical methods to stimulate what is most likely going to happen based on known state of the atmosphere. For example, if a forecaster is looking at three different numerical models, and two model predict that a storm is going to hit a certain place, the forecaster would most likely predict that the storm is going to hit the area. These numerical models work well and are being tweaked all the time, but they still have errors because some of the equations used by the models aren't precise. The application of science and technology that predicts the state of atmosphere at any given particular time period is known as Weather forecasting. There is a many different methods to weather forecast. Weather forecast notices are important because they can be used to prevent destruction of life and environment. The weather forecasting methods used in the ancient time usually implied pattern recognition i.e., they usually rely on observing patterns of events. For example, it is found that the following day has brought fair weather; if the preceding day sunset is particularly red. However, all of the predictions prove not to be reliable.

Firstly, the data is trained. For training the data, we will take 15-20% of the data from the data set. For this prediction, we'll be using Linear regression algorithm and Naïve Bayesian classification algorithm. For the project, we'll be using python, NumPy , Jupiter Notebook, Spyder , Panda. The project is split into three separate Jupiter Notebooks: one to collect the weather data, inspect it, and clean it; a second to further refine the features and fit the data to a Linear Regression model , Random Forest Classifier, Decision Tree, Gaussian NB ,Support Vector Machine ,KNN classifier, model and a third to train and evaluate our output. The project simply uses temperature, apparent temperature , pressure ,wind speed ,wind visibility and humidity for training the data. Here these data are then trained using Linear Regression for the prediction.

## METHODOLOGY AND ALGORITHMS

The methodology used in this project for the prediction of weather and its classification comprises usage of machine learning algorithms- Random Forest Classifier, Decision Tree, KNN classifier, Support Vector Machine (SVM), Gaussian NB and Liner Regression respectively

Steps followed for implementation of this project:

1. ***Data collection****:* The data set used for this project was collected from the Kaggle site. This data set has comprised a total of 12 columns i.e., 12 weather attributes and 96454 data tuples. This dataset was processed to get the desired dataset.
2. ***Data Preprocessing*** *:* The data pre-processing was done on the raw dataset starting with feature selection. This feature selection was done manually based on the literature survey and research.
3. ***Training***: The data set was divided such that x_train is used to train the model with corresponding x_test values and some y_train kept reserved for testing.
4. ***Testing***: The model was tested with y_train and stored in y_predict. Both y_train and y_predict was compared.
5. ***Random Forest Classifier****:* Random Forest Algorithm widespread popularity stems from its user-friendly nature and adaptability, enabling it to tackle both classification and regression problems effectively. The algorithm's strength lies in its ability to handle complex datasets and mitigate overfitting, making it a valuable tool for various predictive tasks in machine learning .**In this project the accuracy for random forest classifier is 99 %.**
6. ***Decision Tree****:* A decision tree is a type of supervised learning algorithm that is commonly used in machine learning to model and predict outcomes based on input data .The decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems. **In this project the accuracy for Decision Tree is 99%.**
7. ***KNN Classifier****:* The K-Nearest Neighbors (KNN) algorithm is a supervised machine learning method employed to tackle classification and regression problems. finds intense application in

pattern recognition, data mining, and intrusion detection. **In this project the accuracy for KNN Classifier is 98 %.**

8. ***Support Vector Machine****:* Support Vector Machine is an algorithm widely used in classification objectives. In SVM hyperplanes are decision boundaries that help classify the data points and we are looking to maximize the margin between the data points and the hyperplane. SVM algorithms use a set of mathematical functions that are defined as the kernel. The kernel function used for the classification of this weather data is 'Polynomial kernel'. **In this project the accuracy of Support Vector Machine is 99%**.

9. ***Gaussian NB****:* Gaussian Naïve Bayes is the extension of naïve Bayes. While other functions are used to estimate data distribution, Gaussian or normal distribution is the simplest to implement as you will need to calculate the mean and standard deviation for the training data. **In this project the accuracy for Gaussian NB is 99%.**

10. ***Linear Regression****:* Linear regression is a type of supervised machine learning algorithm that computes the linear relationship between a dependent variable and one or more independent features .The variable we want to predict is called the dependent variable. The variable we are using to predict the other variable's value is called the independent variable.

# ADVANTAGES AND DISADVANTAGES

## ADVANTAGES:

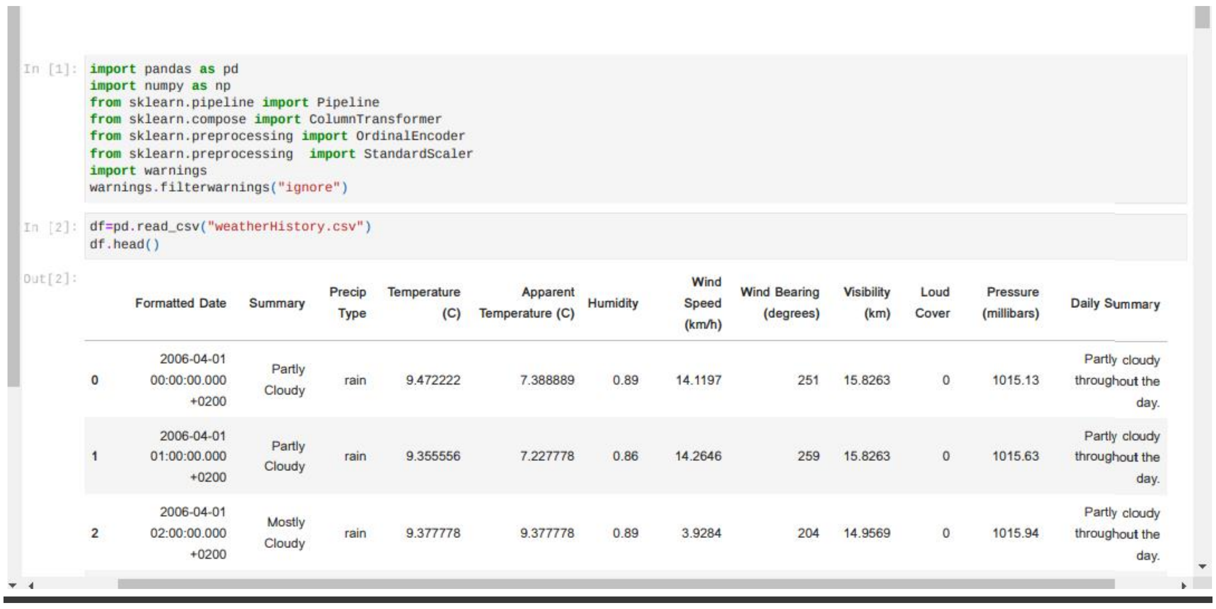
- Improved accuracy: Mathematical models can provide more accurate weather predictions than traditional methods, as they take into account a wide range of variables and can process large amounts of data quickly.
- Increased lead time: Mathematical models can provide weather predictions for several days or even weeks in advance, which can be useful for planning purposes.
- Better understanding of weather patterns: By analyzing data and creating mathematical models, scientists can gain a better understanding of weather patterns and how they are influenced by various factors.

## DISADVANTAGES:

- Complexity: Mathematical models can be very complex and require specialized knowledge to create and interpret.
- Limited accuracy: While mathematical models can provide more accurate weather predictions than traditional methods, they are not always 100% accurate and can still be affected by unforeseen events or errors in the data.
- Cost: Creating and maintaining mathematical models can be expensive, which can limit their accessibility and use in certain regions or by certain organizations.

Overall, while mathematical models have their advantages and disadvantages, they are an important tool for predicting weather and improving our understanding of the natural world.

## CODES

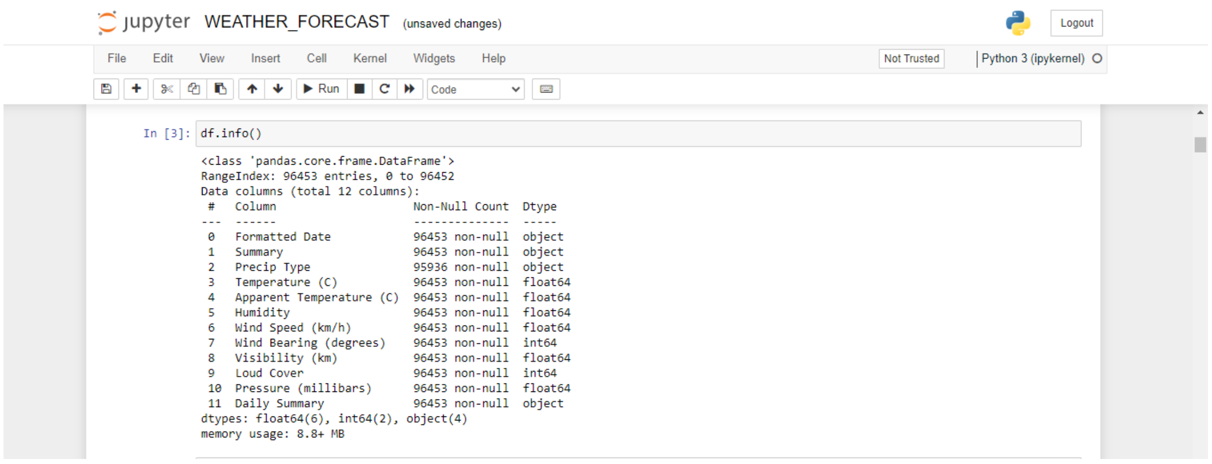


In [1]:
```python
import pandas as pd
import numpy as np
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing  import StandardScaler
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```python
df=pd.read_csv("weatherHistory.csv")
df.head()
```

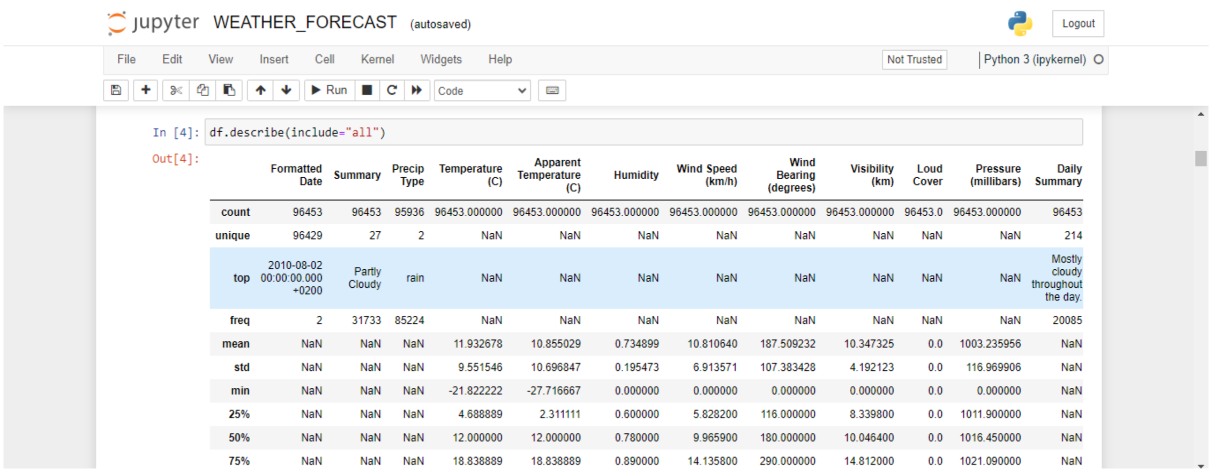
Out[2]:

| | Formatted Date | Summary | Precip Type | Temperature (C) | Apparent Temperature (C) | Humidity | Wind Speed (km/h) | Wind Bearing (degrees) | Visibility (km) | Loud Cover | Pressure (millibars) | Daily Summary |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2006-04-01 00:00:00.000 +0200 | Partly Cloudy | rain | 9.472222 | 7.388889 | 0.89 | 14.1197 | 251 | 15.8263 | 0 | 1015.13 | Partly cloudy throughout the day. |
| 1 | 2006-04-01 01:00:00.000 +0200 | Partly Cloudy | rain | 9.355556 | 7.227778 | 0.86 | 14.2646 | 259 | 15.8263 | 0 | 1015.63 | Partly cloudy throughout the day. |
| 2 | 2006-04-01 02:00:00.000 +0200 | Mostly Cloudy | rain | 9.377778 | 9.377778 | 0.89 | 3.9284 | 204 | 14.9569 | 0 | 1015.94 | Partly cloudy throughout the day. |


jupyter WEATHER_FORECAST (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

In [3]:
```python
df.info()
```

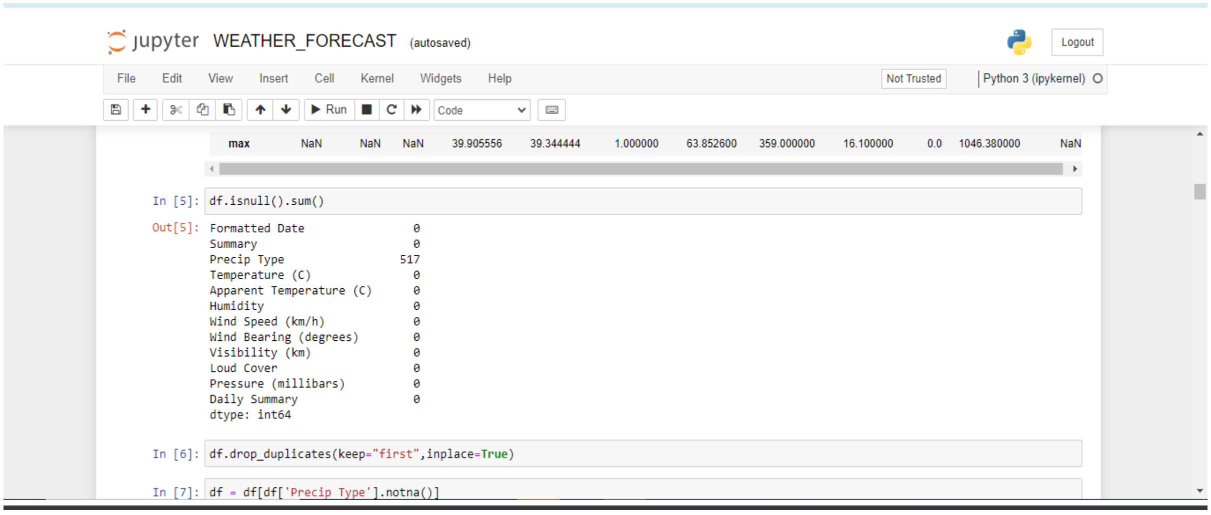
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96453 entries, 0 to 96452
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Formatted Date            96453 non-null  object 
 1   Summary                   96453 non-null  object 
 2   Precip Type               95936 non-null  object 
 3   Temperature (C)           96453 non-null  float64
 4   Apparent Temperature (C)  96453 non-null  float64
 5   Humidity                  96453 non-null  float64
 6   Wind Speed (km/h)         96453 non-null  float64
 7   Wind Bearing (degrees)    96453 non-null  int64  
 8   Visibility (km)           96453 non-null  float64
 9   Loud Cover                96453 non-null  int64  
 10  Pressure (millibars)      96453 non-null  float64
 11  Daily Summary             96453 non-null  object 
dtypes: float64(6), int64(2), object(4)
memory usage: 8.8+ MB
```


jupyter WEATHER_FORECAST (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

In [4]:
```python
df.describe(include="all")
```

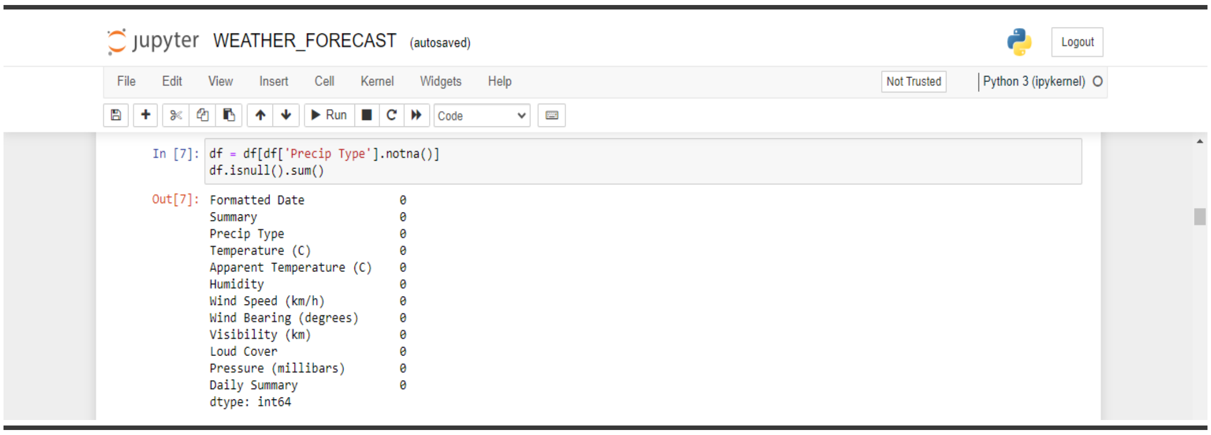
Out[4]:

| | Formatted Date | Summary | Precip Type | Temperature (C) | Apparent Temperature (C) | Humidity | Wind Speed (km/h) | Wind Bearing (degrees) | Visibility (km) | Loud Cover | Pressure (millibars) | Daily Summary |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 96453 | 96453 | 95936 | 96453.000000 | 96453.000000 | 96453.000000 | 96453.000000 | 96453.000000 | 96453.000000 | 96453.0 | 96453.000000 | 96453 |
| **unique** | 96429 | 27 | 2 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 214 |
| **top** | 2010-08-02 00:00:00.000 +0200 | Partly Cloudy | rain | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Mostly cloudy throughout the day. |
| **freq** | 2 | 31733 | 85224 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 20085 |
| **mean** | NaN | NaN | NaN | 11.932678 | 10.855029 | 0.734899 | 10.810640 | 187.509232 | 10.347325 | 0.0 | 1003.235956 | NaN |
| **std** | NaN | NaN | NaN | 9.551546 | 10.696847 | 0.195473 | 6.913571 | 107.383428 | 4.192123 | 0.0 | 116.969906 | NaN |
| **min** | NaN | NaN | NaN | -21.822222 | -27.716667 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.0 | 0.000000 | NaN |
| **25%** | NaN | NaN | NaN | 4.688889 | 2.311111 | 0.600000 | 5.828200 | 116.000000 | 8.339800 | 0.0 | 1011.900000 | NaN |
| **50%** | NaN | NaN | NaN | 12.000000 | 12.000000 | 0.780000 | 9.965900 | 180.000000 | 10.046400 | 0.0 | 1016.450000 | NaN |
| **75%** | NaN | NaN | NaN | 18.838889 | 18.838889 | 0.890000 | 14.135800 | 290.000000 | 14.812000 | 0.0 | 1021.090000 | NaN |

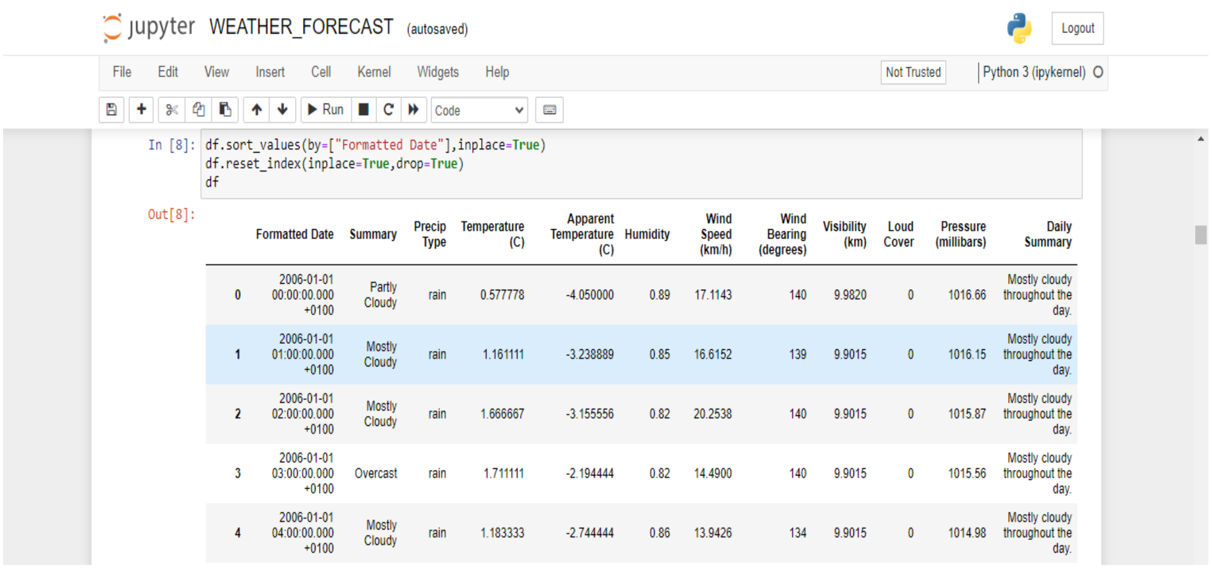

jupyter WEATHER_FORECAST (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
max     NaN   NaN   NaN   39.905556   39.344444   1.000000   63.852600   359.000000   16.100000   0.0   1046.380000   NaN
```

```
In [5]: df.isnull().sum()
Out[5]: Formatted Date                0
        Summary                       0
        Precip Type                 517
        Temperature (C)               0
        Apparent Temperature (C)      0
        Humidity                      0
        Wind Speed (km/h)             0
        Wind Bearing (degrees)        0
        Visibility (km)               0
        Loud Cover                    0
        Pressure (millibars)          0
        Daily Summary                 0
        dtype: int64

In [6]: df.drop_duplicates(keep="first",inplace=True)

In [7]: df = df[df['Precip Type'].notna()]
```

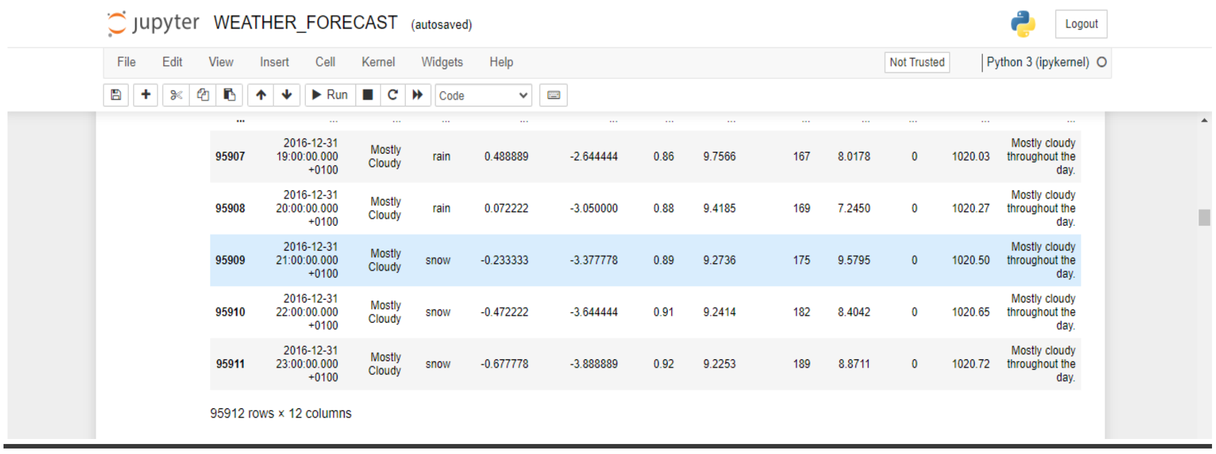

jupyter WEATHER_FORECAST (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [7]: df = df[df['Precip Type'].notna()]
        df.isnull().sum()
Out[7]: Formatted Date              0
        Summary                     0
        Precip Type                 0
        Temperature (C)             0
        Apparent Temperature (C)    0
        Humidity                    0
        Wind Speed (km/h)           0
        Wind Bearing (degrees)      0
        Visibility (km)             0
        Loud Cover                  0
        Pressure (millibars)        0
        Daily Summary               0
        dtype: int64
```


jupyter WEATHER_FORECAST (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

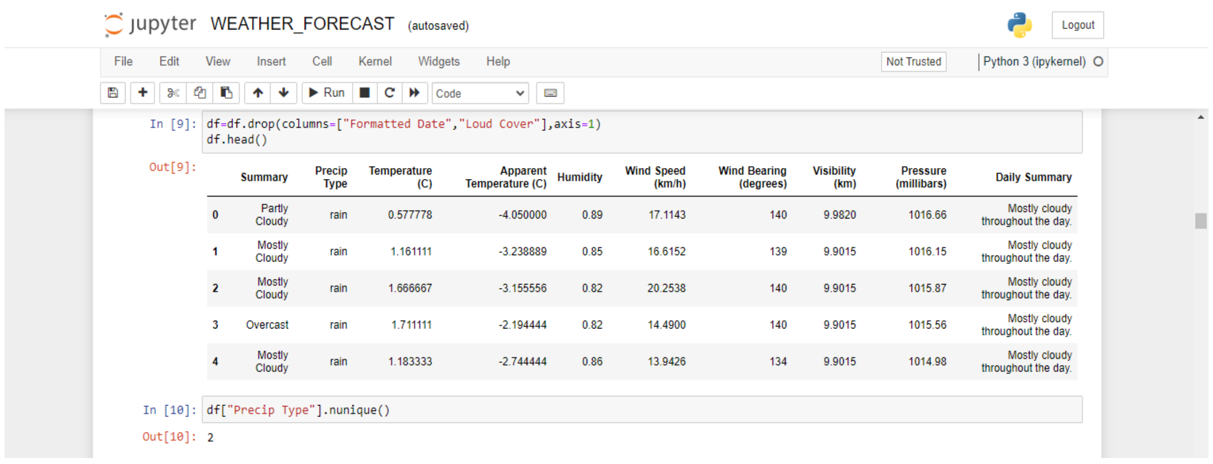
```
In [8]: df.sort_values(by=["Formatted Date"],inplace=True)
        df.reset_index(inplace=True,drop=True)
        df
Out[8]:
```

| | Formatted Date | Summary | Precip Type | Temperature (C) | Apparent Temperature (C) | Humidity | Wind Speed (km/h) | Wind Bearing (degrees) | Visibility (km) | Loud Cover | Pressure (millibars) | Daily Summary |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2006-01-01 00:00:00.000 +0100 | Partly Cloudy | rain | 0.577778 | -4.050000 | 0.89 | 17.1143 | 140 | 9.9820 | 0 | 1016.66 | Mostly cloudy throughout the day. |
| 1 | 2006-01-01 01:00:00.000 +0100 | Mostly Cloudy | rain | 1.161111 | -3.238889 | 0.85 | 16.6152 | 139 | 9.9015 | 0 | 1016.15 | Mostly cloudy throughout the day. |
| 2 | 2006-01-01 02:00:00.000 +0100 | Mostly Cloudy | rain | 1.666667 | -3.155556 | 0.82 | 20.2538 | 140 | 9.9015 | 0 | 1015.87 | Mostly cloudy throughout the day. |
| 3 | 2006-01-01 03:00:00.000 +0100 | Overcast | rain | 1.711111 | -2.194444 | 0.82 | 14.4900 | 140 | 9.9015 | 0 | 1015.56 | Mostly cloudy throughout the day. |
| 4 | 2006-01-01 04:00:00.000 +0100 | Mostly Cloudy | rain | 1.183333 | -2.744444 | 0.86 | 13.9426 | 134 | 9.9015 | 0 | 1014.98 | Mostly cloudy throughout the day. |


jupyter WEATHER_FORECAST (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 95907 | 2016-12-31 19:00:00.000 +0100 | Mostly Cloudy | rain | 0.488889 | -2.644444 | 0.86 | 9.7566 | 167 | 8.0178 | 0 | 1020.03 | Mostly cloudy throughout the day. |
| 95908 | 2016-12-31 20:00:00.000 +0100 | Mostly Cloudy | rain | 0.072222 | -3.050000 | 0.88 | 9.4185 | 169 | 7.2450 | 0 | 1020.27 | Mostly cloudy throughout the day. |
| 95909 | 2016-12-31 21:00:00.000 +0100 | Mostly Cloudy | snow | -0.233333 | -3.377778 | 0.89 | 9.2736 | 175 | 9.5795 | 0 | 1020.50 | Mostly cloudy throughout the day. |
| 95910 | 2016-12-31 22:00:00.000 +0100 | Mostly Cloudy | snow | -0.472222 | -3.644444 | 0.91 | 9.2414 | 182 | 8.4042 | 0 | 1020.65 | Mostly cloudy throughout the day. |
| 95911 | 2016-12-31 23:00:00.000 +0100 | Mostly Cloudy | snow | -0.677778 | -3.888889 | 0.92 | 9.2253 | 189 | 8.8711 | 0 | 1020.72 | Mostly cloudy throughout the day. |

95912 rows × 12 columns

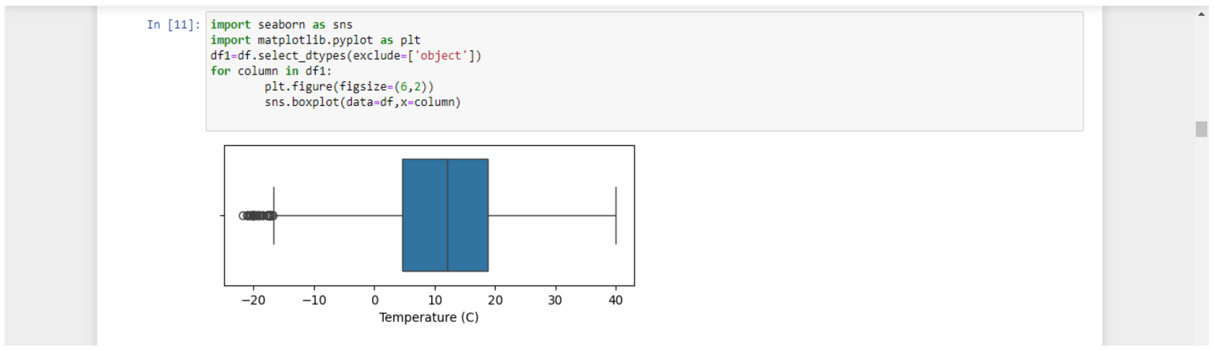

jupyter WEATHER_FORECAST (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [9]: df=df.drop(columns=["Formatted Date","Loud Cover"],axis=1)
        df.head()
```

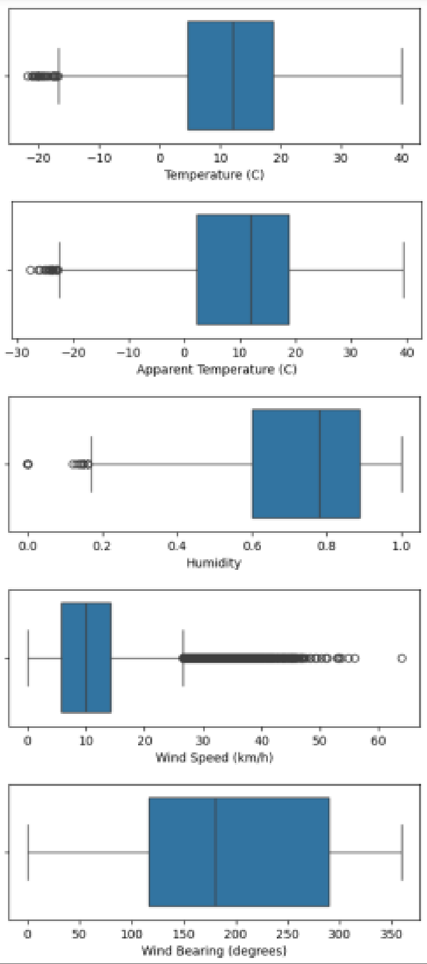
Out[9]:

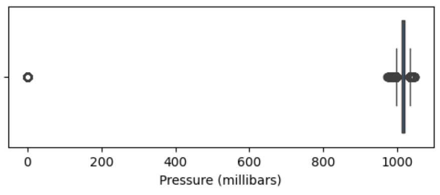
| | Summary | Precip Type | Temperature (C) | Apparent Temperature (C) | Humidity | Wind Speed (km/h) | Wind Bearing (degrees) | Visibility (km) | Pressure (millibars) | Daily Summary |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Partly Cloudy | rain | 0.577778 | -4.050000 | 0.89 | 17.1143 | 140 | 9.9820 | 1016.66 | Mostly cloudy throughout the day. |
| 1 | Mostly Cloudy | rain | 1.161111 | -3.238889 | 0.85 | 16.6152 | 139 | 9.9015 | 1016.15 | Mostly cloudy throughout the day. |
| 2 | Mostly Cloudy | rain | 1.666667 | -3.155556 | 0.82 | 20.2538 | 140 | 9.9015 | 1015.87 | Mostly cloudy throughout the day. |
| 3 | Overcast | rain | 1.711111 | -2.194444 | 0.82 | 14.4900 | 140 | 9.9015 | 1015.56 | Mostly cloudy throughout the day. |
| 4 | Mostly Cloudy | rain | 1.183333 | -2.744444 | 0.86 | 13.9426 | 134 | 9.9015 | 1014.98 | Mostly cloudy throughout the day. |

```
In [10]: df["Precip Type"].nunique()

Out[10]: 2
```

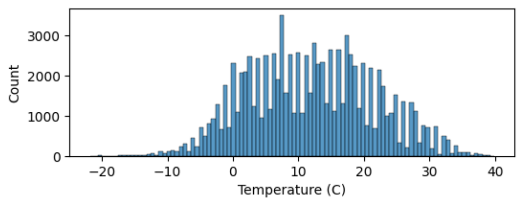


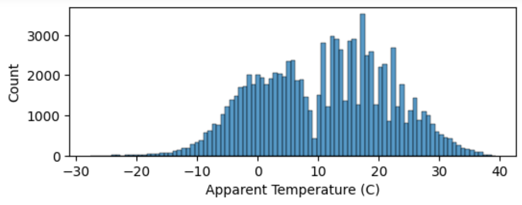
```
In [11]: import seaborn as sns
         import matplotlib.pyplot as plt
         df1=df.select_dtypes(exclude=['object'])
         for column in df1:
                plt.figure(figsize=(6,2))
                sns.boxplot(data=df,x=column)
```

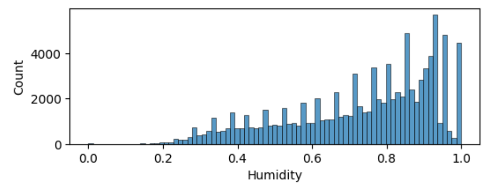


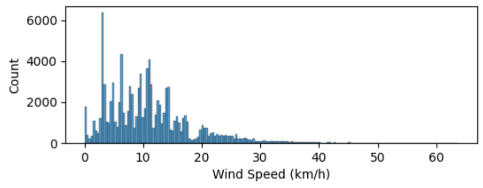


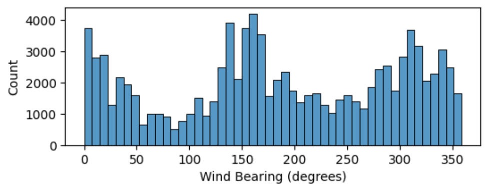
```
In [12]: for column in df1:
             plt.figure(figsize=(6,2))
             sns.histplot(data=df,x=column)
```

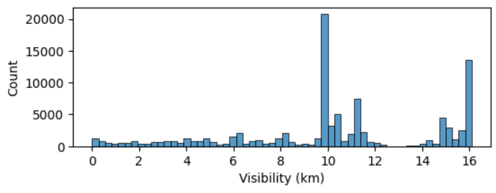


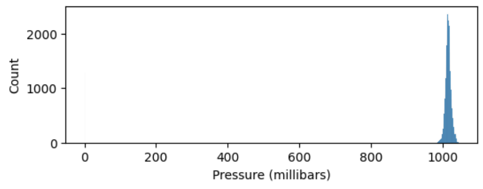




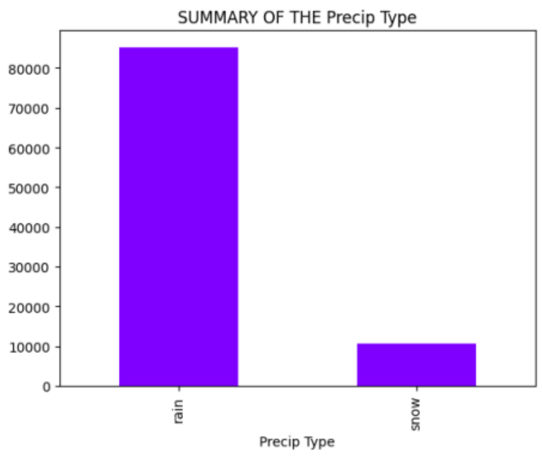




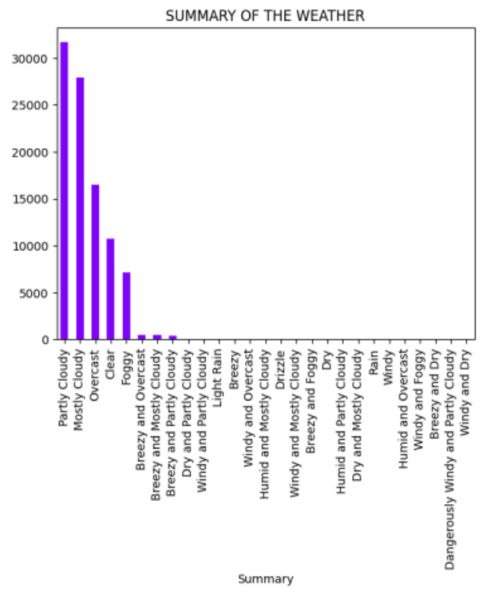




```
In [13]: df['Precip Type'].value_counts().plot(kind='bar', cmap='rainbow')
         plt.title('SUMMARY OF THE Precip Type')
         plt.show()
```



```
In [14]: df['Summary'].value_counts().plot(kind='bar', cmap='rainbow')
         plt.title('SUMMARY OF THE WEATHER')
         plt.show()
```

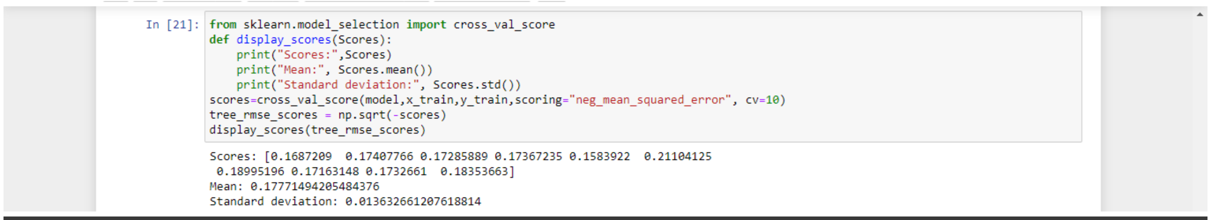


```
In [15]: df = df[df['Humidity'] != 0.0]
         df = df[df['Pressure (millibars)'] != 0]
         df.reset_index(inplace=True, drop=True)
         df.shape

Out[15]: (94602, 10)

In [16]: x = df.select_dtypes(include=[np.number])
         num_attribs =list(x)
         cat_attribs = list(df.select_dtypes(include=np.object_))
```

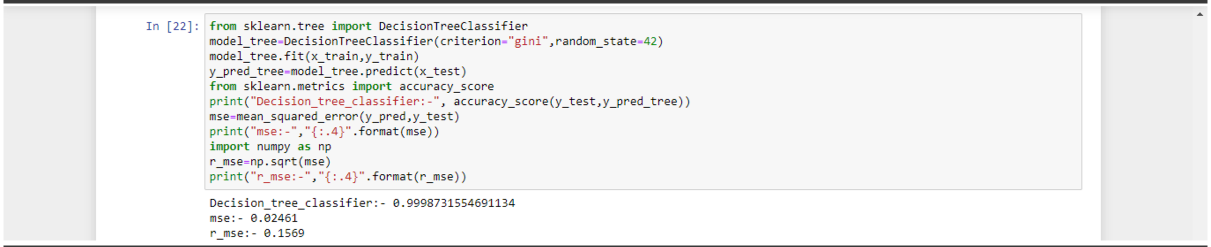
```
In [17]: num_pipeline = Pipeline([('std_scaler', StandardScaler())])
         full_pipeline = ColumnTransformer([("num", num_pipeline, num_attribs),("cat", OrdinalEncoder(), cat_attribs)])
         x=full_pipeline.fit_transform(df)
         x

Out[17]: array([[ -1.19099965,  -1.39641442,   0.79227398, ...,  19.        ,
                   0.        , 111.        ],
                [ -1.12996515,  -1.32059788,   0.58771043, ...,  17.        ,
                   0.        , 111.        ],
                [ -1.07706858,  -1.31280851,   0.43428777, ...,  17.        ,
                   0.        , 111.        ],
                ...,
                [ -1.27586667,  -1.33358016,   0.79227398, ...,  17.        ,
                   1.        , 111.        ],
                [ -1.30086175,  -1.35850615,   0.89455575, ...,  17.        ,
                   1.        , 111.        ],
                [ -1.32236915,  -1.38135497,   0.94569664, ...,  17.        ,
                   1.        , 111.        ]])
```

```
In [18]: y=x[:,7]
         y

Out[18]: array([19., 17., 17., ..., 17., 17., 17.])

In [19]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.25,random_state=0)
```

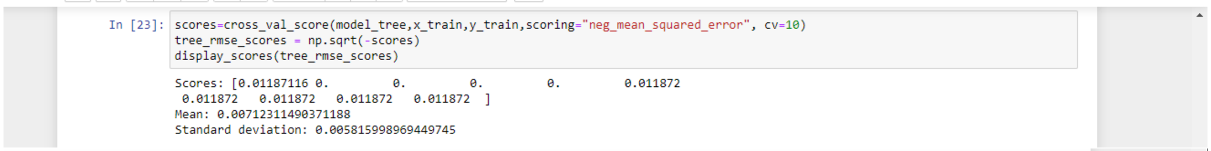
```
In [20]: from sklearn.neighbors import KNeighborsClassifier
         model=KNeighborsClassifier(weights="distance")
         model.fit(x_train,y_train)
         y_pred=model.predict(x_test)
         from sklearn.metrics import accuracy_score
         print("KNN CLASSIFIER:-",accuracy_score(y_test,y_pred))
         mse=mean_squared_error(y_pred,y_test)
         print("mse:-","{:.4}".format(mse))
         import numpy as np
         r_mse=np.sqrt(mse)
         print("r_mse:-","{:.4}".format(r_mse))

         KNN CLASSIFIER:- 0.98186123208321
         mse:- 0.02461
         r_mse:- 0.1569
```



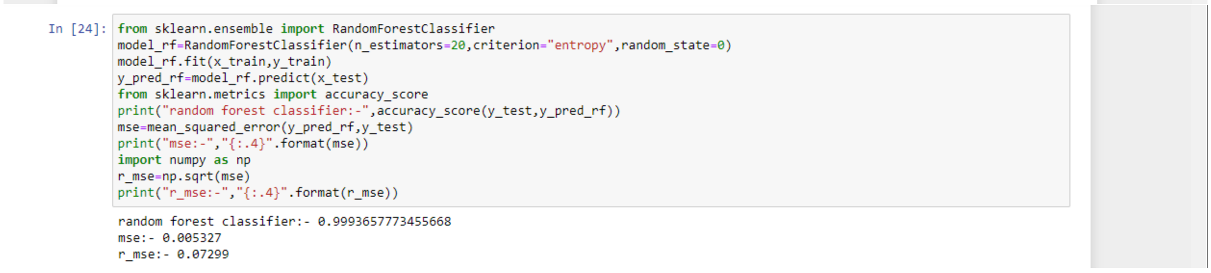
```
In [21]: from sklearn.model_selection import cross_val_score
         def display_scores(Scores):
             print("Scores:",Scores)
             print("Mean:", Scores.mean())
             print("Standard deviation:", Scores.std())
         scores=cross_val_score(model,x_train,y_train,scoring="neg_mean_squared_error", cv=10)
         tree_rmse_scores = np.sqrt(-scores)
         display_scores(tree_rmse_scores)

         Scores: [0.1687209  0.17407766 0.17285889 0.17367235 0.1583922  0.21104125
          0.18995196 0.17163148 0.1732661  0.18353663]
         Mean: 0.17771494205484376
         Standard deviation: 0.013632661207618814
```



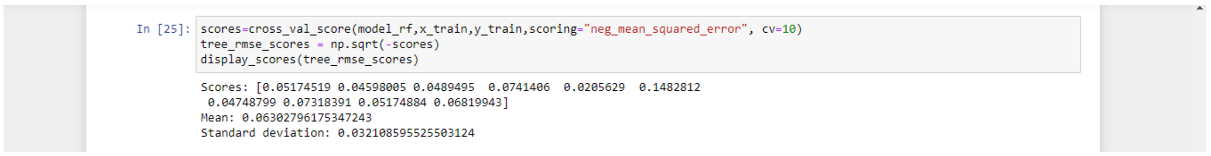
```
In [22]: from sklearn.tree import DecisionTreeClassifier
         model_tree=DecisionTreeClassifier(criterion="gini",random_state=42)
         model_tree.fit(x_train,y_train)
         y_pred_tree=model_tree.predict(x_test)
         from sklearn.metrics import accuracy_score
         print("Decision_tree_classifier:-", accuracy_score(y_test,y_pred_tree))
         mse=mean_squared_error(y_pred,y_test)
         print("mse:-","{:.4}".format(mse))
         import numpy as np
         r_mse=np.sqrt(mse)
         print("r_mse:-","{:.4}".format(r_mse))

         Decision_tree_classifier:- 0.9998731554691134
         mse:- 0.02461
         r_mse:- 0.1569
```



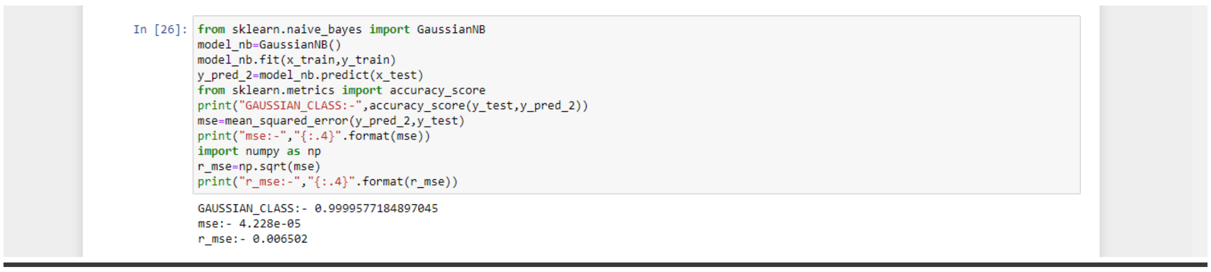
```
In [23]: scores=cross_val_score(model_tree,x_train,y_train,scoring="neg_mean_squared_error", cv=10)
         tree_rmse_scores = np.sqrt(-scores)
         display_scores(tree_rmse_scores)

         Scores: [0.01187116 0.         0.         0.         0.         0.011872
          0.011872   0.011872   0.011872   0.011872  ]
         Mean: 0.00712311490371188
         Standard deviation: 0.005815998969449745
```



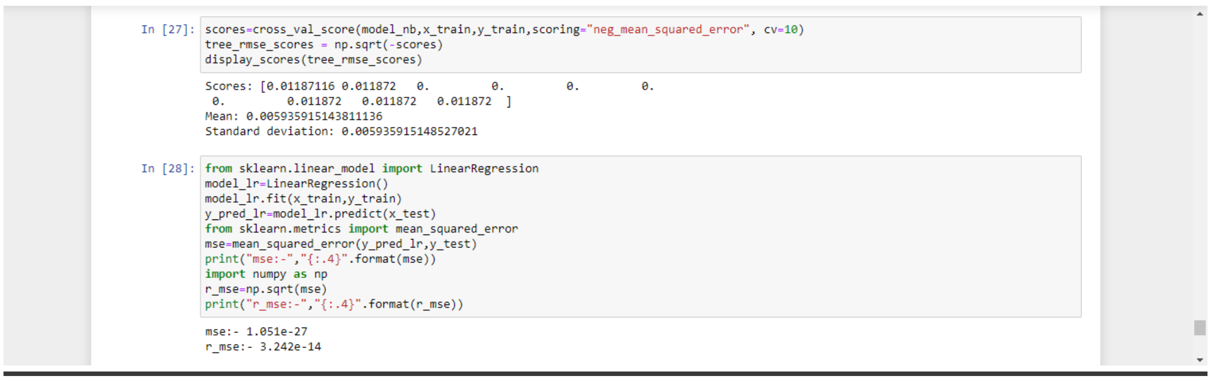
```
In [24]: from sklearn.ensemble import RandomForestClassifier
         model_rf=RandomForestClassifier(n_estimators=20,criterion="entropy",random_state=0)
         model_rf.fit(x_train,y_train)
         y_pred_rf=model_rf.predict(x_test)
         from sklearn.metrics import accuracy_score
         print("random forest classifier:-",accuracy_score(y_test,y_pred_rf))
         mse=mean_squared_error(y_pred_rf,y_test)
         print("mse:-","{:.4}".format(mse))
         import numpy as np
         r_mse=np.sqrt(mse)
         print("r_mse:-","{:.4}".format(r_mse))

         random forest classifier:- 0.9993657773455668
         mse:- 0.005327
         r_mse:- 0.07299
```



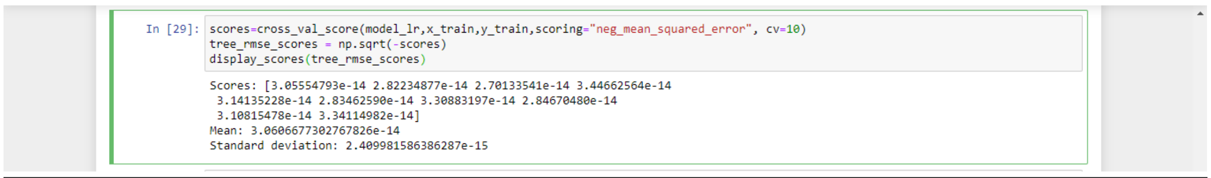
```
In [25]: scores=cross_val_score(model_rf,x_train,y_train,scoring="neg_mean_squared_error", cv=10)
         tree_rmse_scores = np.sqrt(-scores)
         display_scores(tree_rmse_scores)

         Scores: [0.05174519 0.04598005 0.0489495  0.0741406  0.0205629  0.1482812
          0.04748799 0.07318391 0.05174884 0.06819943]
         Mean: 0.06302796175347243
         Standard deviation: 0.03210859552550312 4
```



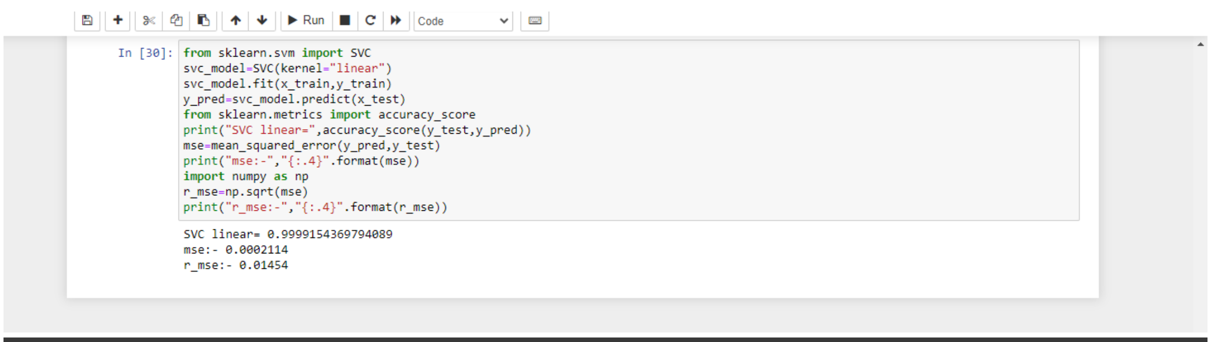
```
In [26]: from sklearn.naive_bayes import GaussianNB
         model_nb=GaussianNB()
         model_nb.fit(x_train,y_train)
         y_pred_2=model_nb.predict(x_test)
         from sklearn.metrics import accuracy_score
         print("GAUSSIAN_CLASS:-",accuracy_score(y_test,y_pred_2))
         mse=mean_squared_error(y_pred_2,y_test)
         print("mse:-","{:.4}".format(mse))
         import numpy as np
         r_mse=np.sqrt(mse)
         print("r_mse:-","{:.4}".format(r_mse))

         GAUSSIAN_CLASS:- 0.9999577184897045
         mse:- 4.228e-05
         r_mse:- 0.006502
```



```
In [27]: scores=cross_val_score(model_nb,x_train,y_train,scoring="neg_mean_squared_error", cv=10)
         tree_rmse_scores = np.sqrt(-scores)
         display_scores(tree_rmse_scores)

         Scores: [0.01187116 0.011872   0.         0.         0.         0.
          0.         0.011872   0.011872   0.011872  ]
         Mean: 0.005935915143811136
         Standard deviation: 0.005935915148527021

In [28]: from sklearn.linear_model import LinearRegression
         model_lr=LinearRegression()
         model_lr.fit(x_train,y_train)
         y_pred_lr=model_lr.predict(x_test)
         from sklearn.metrics import mean_squared_error
         mse=mean_squared_error(y_pred_lr,y_test)
         print("mse:-","{:.4}".format(mse))
         import numpy as np
         r_mse=np.sqrt(mse)
         print("r_mse:-","{:.4}".format(r_mse))

         mse:- 1.051e-27
         r_mse:- 3.242e-14
```



```
In [29]: scores=cross_val_score(model_lr,x_train,y_train,scoring="neg_mean_squared_error", cv=10)
         tree_rmse_scores = np.sqrt(-scores)
         display_scores(tree_rmse_scores)

         Scores: [3.05554793e-14 2.82234877e-14 2.70133541e-14 3.44662564e-14
          3.14135228e-14 2.83462590e-14 3.30883197e-14 2.84670480e-14
          3.10815478e-14 3.34114982e-14]
         Mean: 3.0606677302767826e-14
         Standard deviation: 2.409981586386287e-15
```



```
In [30]: from sklearn.svm import SVC
         svc_model=SVC(kernel="linear")
         svc_model.fit(x_train,y_train)
         y_pred=svc_model.predict(x_test)
         from sklearn.metrics import accuracy_score
         print("SVC linear=",accuracy_score(y_test,y_pred))
         mse=mean_squared_error(y_pred,y_test)
         print("mse:-","{:.4}".format(mse))
         import numpy as np
         r_mse=np.sqrt(mse)
         print("r_mse:-","{:.4}".format(r_mse))

         SVC linear= 0.9999154369794089
         mse:- 0.0002114
         r_mse:- 0.01454
```

## FUTURE SCOPE

The future of weather applications is promising, with the increasing demand for real-time and accurate weather information. One potential development is the improvement in accuracy through the use of advanced data collection and analysis techniques, as well as sophisticated algorithms . The advancement of our ability to predict the weather and climate has been the core aspiration of a global community of scientists and practitioners, in the almost 150 years of international cooperation in meteorology and related Earth system sciences. The demand for weather and climate forecast information in support of critical decision-making has grown rapidly during the last decade and will grow even faster in the coming years. Great advances have been made in the utilization of predictions in many areas of human activities. Nevertheless, further improvements in accuracy and precision, higher spatial and temporal resolution, and a better description of uncertainty are needed for realizing the full potential of forecasts as enablers of a new level of weather and climate-informed decision-making.

# CONCLUSION

Machine learning methods will be a key feature in future weather forecasting and climate analysis, according to the authors' analysis of the literature .The authors identified the most common topics of interest in the field, including photovoltaic and wind energy, atmospheric physics and processes, parameterizations , extreme events, and climate change .The most commonly examined meteorological fields were wind, precipitation, temperature, pressure, wind speed ,wind bearing , humidity and visibility. In this project we have collected the raw data from online sources. Then we take this raw data and format it.

Now we have selected few models for accuracy and error detection .We have used six models namely Random Forest Classifier, Decision Tree, Linear Regression, Gaussian NB, KNN Classifier, Support Vector Machine.

Gaussian NB , Random Forest Classifier and Decision Tree are giving good accuracy score near about  99% which is greater than the other model. In three of this algorithms, Gaussian NB gives 99.9957%  that is greater of all other models.

# **BIBLIOGRAPHY**

For successfully completing our project report on "**WEATHER FORECASTING"** by using machine learning . we have taken help from the following website links:-

- www.google.com
- https://www.kaggle.com/datasets/muthuj7/weather-dataset
- https://cloud.google.com/blog/topics/sustainability/weather-prediction-with-ai#:~:text=Historically%2C%20weather%20forecasting%20is%20done,speed%20of%20making%20these%20predictions
- https://iopscience.iop.org/article/10.1088/1742-6596/2089/1/012059/pdf
- https://www.irjmets.com/uploadedfiles/paper//issue_3_march_2023/34599/final/fin_irjmets1679798881.pdf
- https://www.quora.com/What-are-the-advantages-and-disadvantages-of-using-mathematical-models-to-predict-weather

*Under the guidance of*

**MAHENDRA DATTA**

***THANK YOU***