EXPLORATORY ANALYSIS OF RAINFALL DATA IN INDIA FOR AGRICULTURE

BY : **PNT2022TMID12561** (TEAM ID)
RAKESH M (19Z235)
SOUMEN SAHA (19Z245)
SUDHARSAN V (19Z249)
VIVEKANANDHAN S (19Z261)

19Z039 - PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTREPRENEURSHIP

Dissertation submitted in partial fulfilment of the requirements for the degree of

BACHELOR OF ENGINEERING
BRANCH: COMPUTER SCIENCE AND ENGINEERING
of Anna University

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PSG COLLEGE OF TECHNOLOGY
(Autonomous Institution)

COIMBATORE - 641004

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

Agriculture is the backbone of the Indian economy. For agriculture, the most important thing is water source, i.e. rainfall. The prediction of the amount of rainfall gives alertness to farmers by knowing early they can protect their crops from rain. So, it is important to predict the rainfall accurately as much as possible. Exploration and analysis of data on rainfall over various regions of India and especially the regions where agricultural works have been done persistently in a wide range. With the help of analysis and the resultant data, future rainfall prediction for those regions using various machine learning techniques such as XGBoost classifier, SVM classifier, Decision tree, Naive Bayes classifier & Logistic regression.

1.2 PURPOSE

The purpose of the project is to build a forecasting machine learning-based model that will be critical to the development of an early warning system that can minimise hazards to people and property while also improving the management of agricultural chores.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING PROBLEM

Crops and farming fields may be adversely affected by frequent and heavy rain, which can result in poor crop growth and overall plant health. Limited food access and unsustainable agricultural practises are also a result of this.

2.2 REFERENCES

- Machine Learning based Rainfall Prediction:
  https://ieeexplore.ieee.org/document/9074233
  This paper explains the proposed method MLR [Multiple Linear Regression] based Rain Fall Prediction. The proposed method predicts the rainfall for the Indian dataset using multiple linear regression and provides improved results in terms of accuracy, MSE and correlation. The data for the prediction is collected from the publicly available sources and the 70 percentage of the data is for training and the 30 percentage of the data is for testing.

- Machine Learning Techniques For Rainfall Prediction:
  https://www.researchgate.net/publication/319503839_Machine_Learning_Techniques_For_Rainfall_Prediction_A_Review

  Review work and comparison of different approaches and algorithms used by researchers for rainfall prediction is shown in a tabular form. Intention of this paper is to give non- experts easy access to the techniques and approaches used in the field of rainfall prediction.

- A detailed literature survey for this project can be found in this link - *https://github.com/IBM-EPBL/IBM-Project-13205-1659514193/blob/main/Project%20Design%20and%20Planning/Ideation%20Phase/Literature%20Survey.pdf*

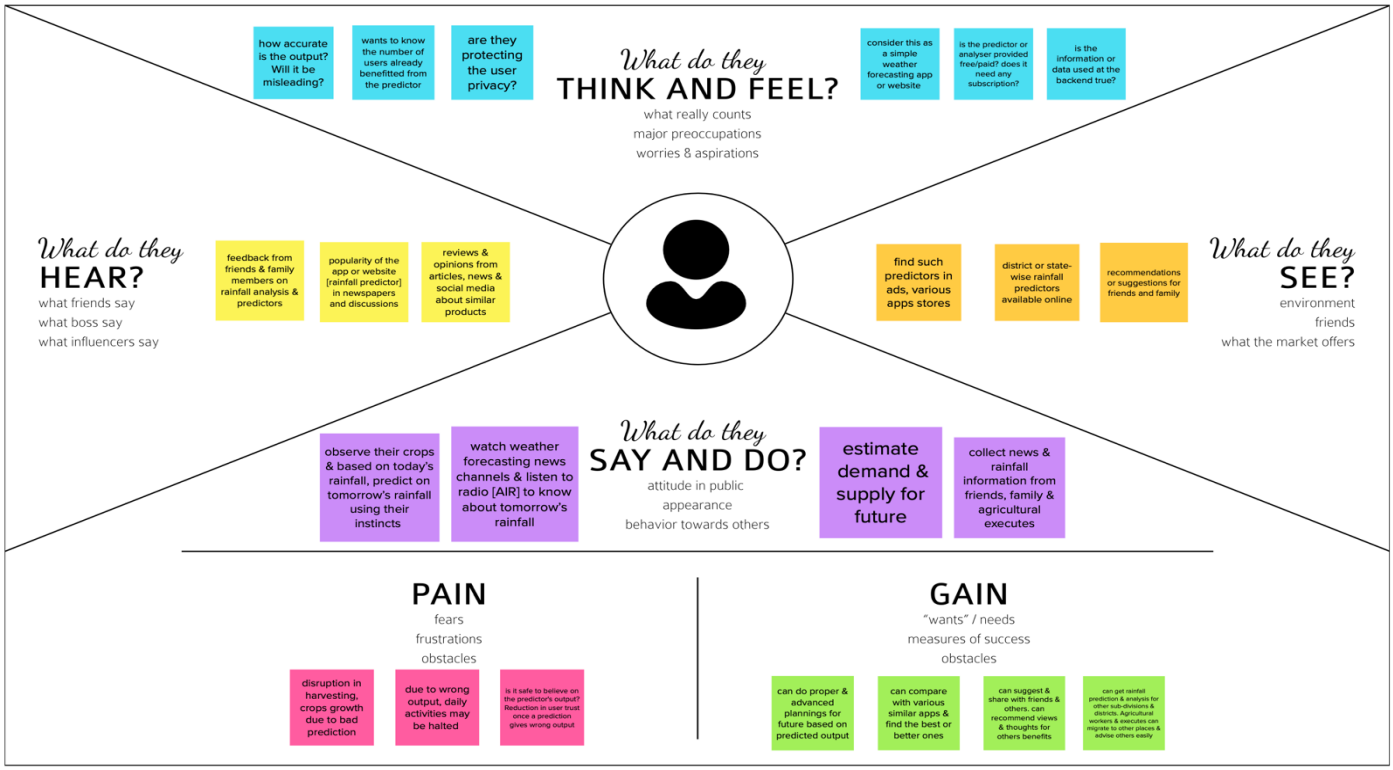
2.3 PROBLEM STATEMENT DEFINITION

Farmers facing the daunting task of gathering their harvest and taking the produce to market after excessive rainfall harmed the winter crops. Accurate and timely rainfall prediction is expected to inject a new intervention phase to the affected sectors accosted with the negative propensities of rainfall extremes. Heavy rainfall can have impacts like damage or destruction of crops so a tool is required which can predict the rainfall more accurate so that it helps farmers and also for utilizing the water resources efficiently.

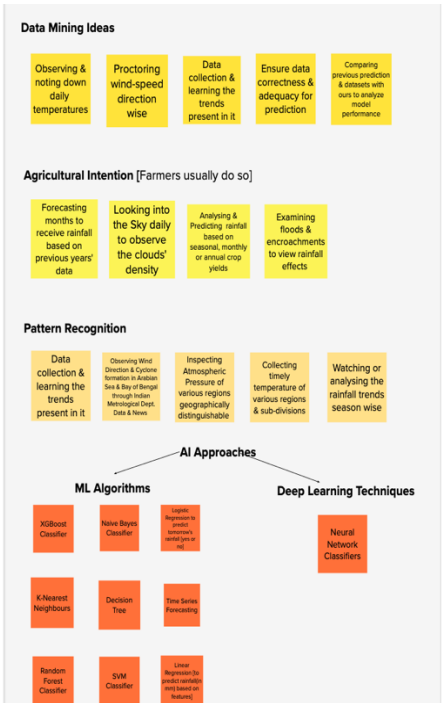
CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP



3.2 IDEATION & BRAINSTORMING



3.3 PROPOSED SOLUTION

S.NO	Parameter	Description
1	Problem Statement	Farmers face the daunting task of gathering their harvest and taking the produce to market after excessive rainfall harmed the crops. Accurate and timely rainfall prediction is expected to inject a new intervention phase to the affected sectors afflicted by the negative propensities of rainfall extremes. Heavy rainfall can have impacts like damage or destruction of crops, so a tool is required that can predict the rainfall more accurately so that it helps farmers efficiently utilize crop production and water resources.
2.	Ideation/ Solution Description	Analysing previous years' rainfall data from all over India to get the seasonal patterns with respect to the production of different sorts of crops. Building an ML-based model to predict the rainfall of places in India with a high concentration of agricultural activities while taking care of the trends and analysis done already**.**
3	Novelty/ Uniqueness	Regional or zonal based prediction of rainfall, which would be helpful to farmer communities of different places having varied crop cultivation. Various ML models [in-built, hybrid or ensemble methods] would be applied to the datasets and chosen to make predictions based on their accuracy, reliability, and sustainability.
4.	Social Impact/ Customer Satisfaction	This application would help the users to maintain an overall balance between demand and supply of agricultural stocks while the farmers can take decisions for cropping, harvesting, and efficient use of the water resources. It would reduce the losses and prevent the farmers from attempting suicide, providing an improved quality of life.
5.	Business / Revenue Model	Correct and accurate predictions from the built model would fetch adequate profits for the respective users and user sectors. As the economy of India is largely dependent on the primary sector especially agriculture and its allied activities, the model is useful to other departments like tea plantations, tourism, metrological dept. etc. Govt. aid and open-sourcing of datasets would allow the farmers and other users to avail the product in low or no charges.
6.	Scalability of the Solution	Effective analysis and prediction will assist not only farmers and other people associated with agriculture in tracking the effect of rainfall on their crops and harvests, but also people in all sectors [government ministry, news agencies, vegetable or crop sellers, common citizens] in using our product or tool for their daily needs. Any feature or module could be easily included into the application to expand the user functionalities.

3.4 PROBLEM SOLUTION FIT

- CUSTOMER SEGMENTS

The customer segments in this case are primarily the farmers, the employees or the workers associated with agricultural activities, and the departments of the government or news organisations seeking agricultural rainfall forecasts.

- JOBS-TO-BE-DONE / PROBLEMS

We have to get the proper analysis from previous data and achieve correct and accurate predictions. The problems that would affect the jobs could be any sudden change in weather, immediate rainfall or showers, and crop damage due to heavy rainfall.

- TRIGGERS

The main triggers include current losses and debts and the yearly crop damage due to heavy rainfall and the evolving market competition and change in demand supply.

- EMOTIONS: BEFORE / AFTER

The emotions that have been noted before are paying debts, incurring losses, and low crop production, and those that have been found afterwards are an increase in crop production, making effective decisions, experiencing growth, and making profits.

- AVAILABLE SOLUTIONS

The available solutions that we found are the news on weather forecasting from various communication media like radio, news channels, etc., the announcements from the concerned authorities, and notifications from connections (friends and family) on upcoming rainfalls affecting agriculture.

- CUSTOMER CONSTRAINTS

The customer challenges include estimating the duration and volume of rainfall beforehand and taking decisions accordingly. They also include getting a prediction with 100% accuracy and determining the cost factors for applications with high prediction accuracy and value. They could have limited time to make use of digital devices to get the prediction information and might face problems with stable network connections.

- BEHAVIOUR

The reaction or behaviour of the customers includes taking suggestions from concerned authorities, agricultural scientists, and other influencers to make decisions. They can also take decisions based on previous experiences and self-analysis.

- CHANNELS OF BEHAVIOUR

    - ONLINE

      Receiving early notifications on their digital devices, especially mobiles or smartphones, through SMS or app alerts corresponds to the online behaviour.

    - OFFLINE

      Community forums, meetings where farmers and other people can share ideas, discuss and decide on crop activities correspond to the offline behaviour.

- PROBLEM ROOT CAUSE(S)

    - Irregular rainfall in various regions of India
    - Drastic variability in climate change
    - Biodiversity loss.

- OUR SOLUTION

The proposed solution includes the region (district or state) based analysis of previous year's rainfall data to get the seasonal patterns with respect to the production of different sorts of crops and the building of a low-cost or free ML-based application (consuming low bandwidth) to predict the rainfall of places with a high concentration of agricultural activities while taking care of the trends and analysis already done.

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

FR No.	Functional Requirement [Epic]	Sub-Requirement [Story / Sub-Task]
FR-1	User Registration	Registration via form or email ID and password creation
FR-2	User Confirmation	Confirmation via Email or OTP
FR-3	User Login	Using the registered email ID and password as login credentials
FR-4	Profile Dashboard	Viewing the profile, changing the password and pages navigation
FR-4	Searching	Searching for results and information by place and region
FR-5	Visualization Dashboard	Visualizing the user-specific data in different forms
FR-6	Prediction	Giving inputs to get the prediction on rainfall using an ML-based model
FR-7	User tracking	Maintaining the history of the user's search operations
FR-8	Feedback & Support	Collecting feedback against the accuracy of the prediction for further improvement and feature inclusion in other modules or functionalities

4.2 NON-FUNCTIONAL REQUIREMENTS

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	• The system should administer a quality attribute that assesses how easy *user interfaces* are to use • The system doesn't expect any technical pre-requisites from the user's side
NFR-2	Security	• User details and login credentials should be safe and secure • The confirmation of a valid user is required for authentication
NFR-3	Reliability	• Portable and cross-platform independent • The application should be subjected to an experiment, test, or measuring procedure that yields the same results on repeated trials • Easy to use and flexible
NFR-4	Performance	• The system should handle the traffic efficiently and service requests while consuming less bandwidth • The accuracy of the result of a measurement, calculation, or specification should be dependent the datasets • The page should not take a lot of time to load the contents and display them
NFR-5	Availability	• The version of the application should be available even at the time of maintenance and updating • The system should run 24 hours a day, 7 days a week *[24/7 available]*
NFR-6	Scalability	• The application should be in the way of adding new functionalities or modules without affecting the existing functionalities • The system should be able to manage numerous users at a time and be less prone to errors

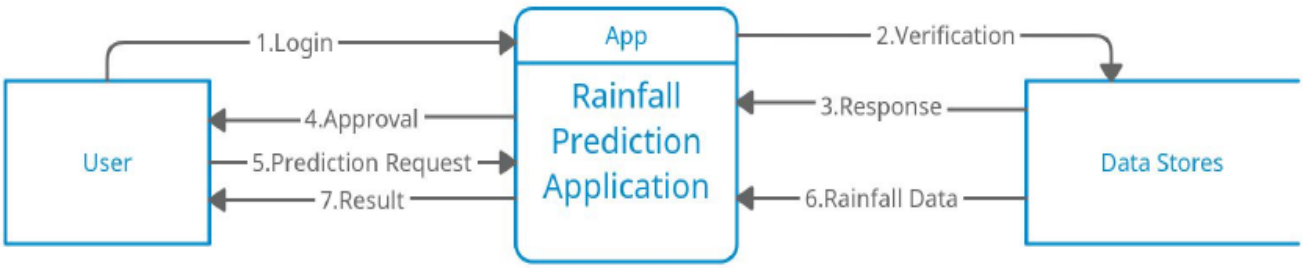
CHAPTER 5

PROJECT DESIGN

5.1 DATA FLOW DIAGRAM

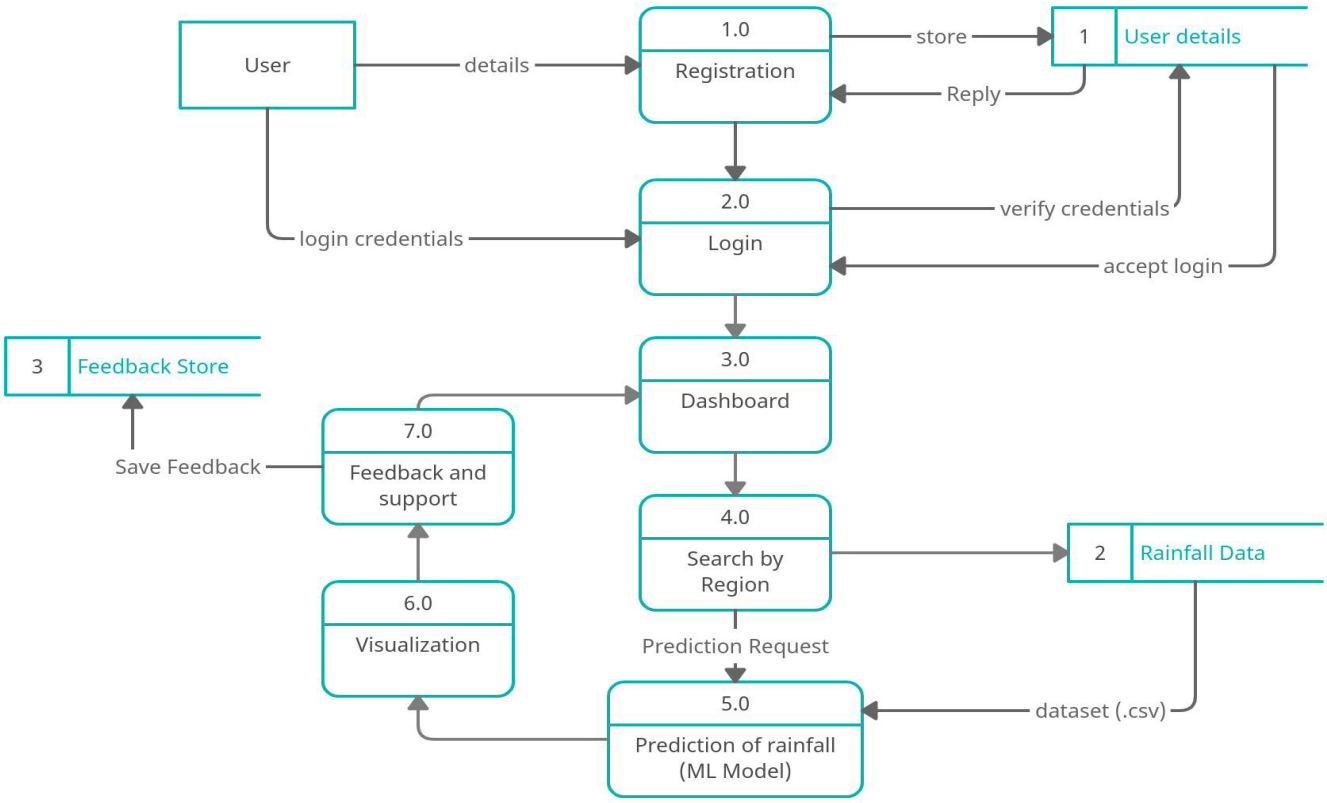
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

0 - LEVEL DATA FLOW DIAGRAM:

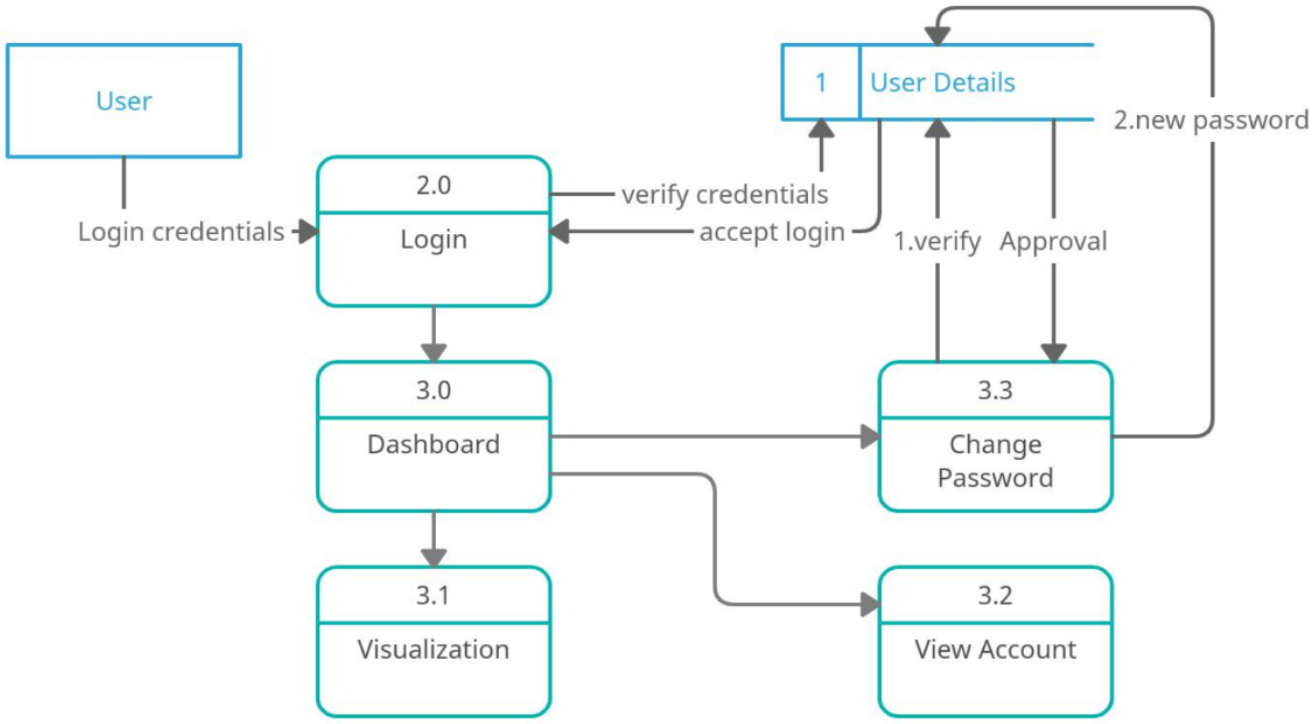


- User logins to the application using his or her credentials.
- Verification of credentials is done using the data stored in the database.
- Application getting the response from the database.
- Approval of login or else an error message for incorrect credentials.
- Prediction requests for the particular area or region are sent by the user.
- Application getting the dataset of previous year/month/day rainfall data from the database/cloud.
- The result has been sent to the user as an output after the prediction has been made using the machine learning model in the application.

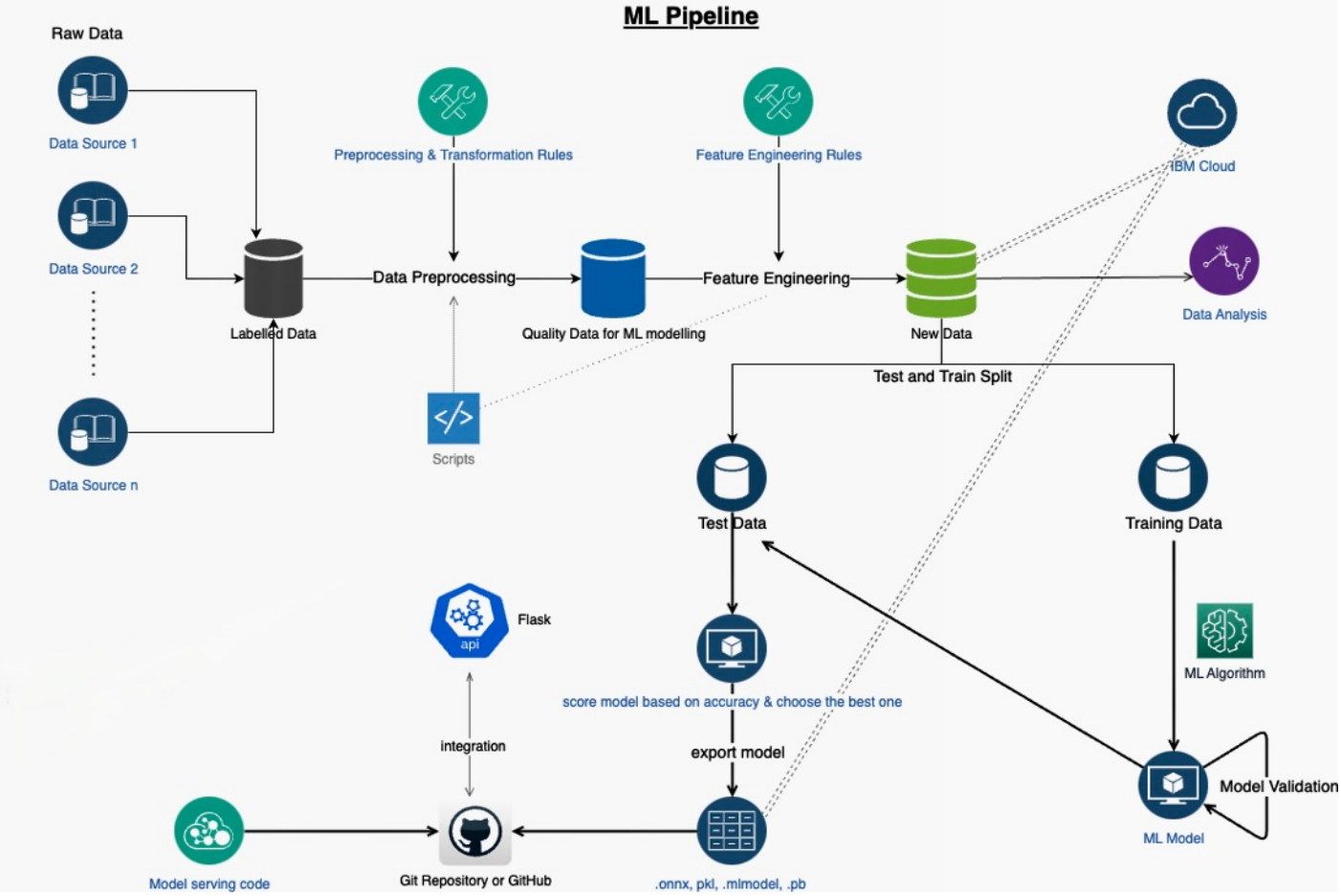
1 - LEVEL DATA FLOW DIAGRAM:



2 - LEVEL DATA FLOW DIAGRAM:



5.2 SOLUTION & TECHNICAL ARCHITECTURE



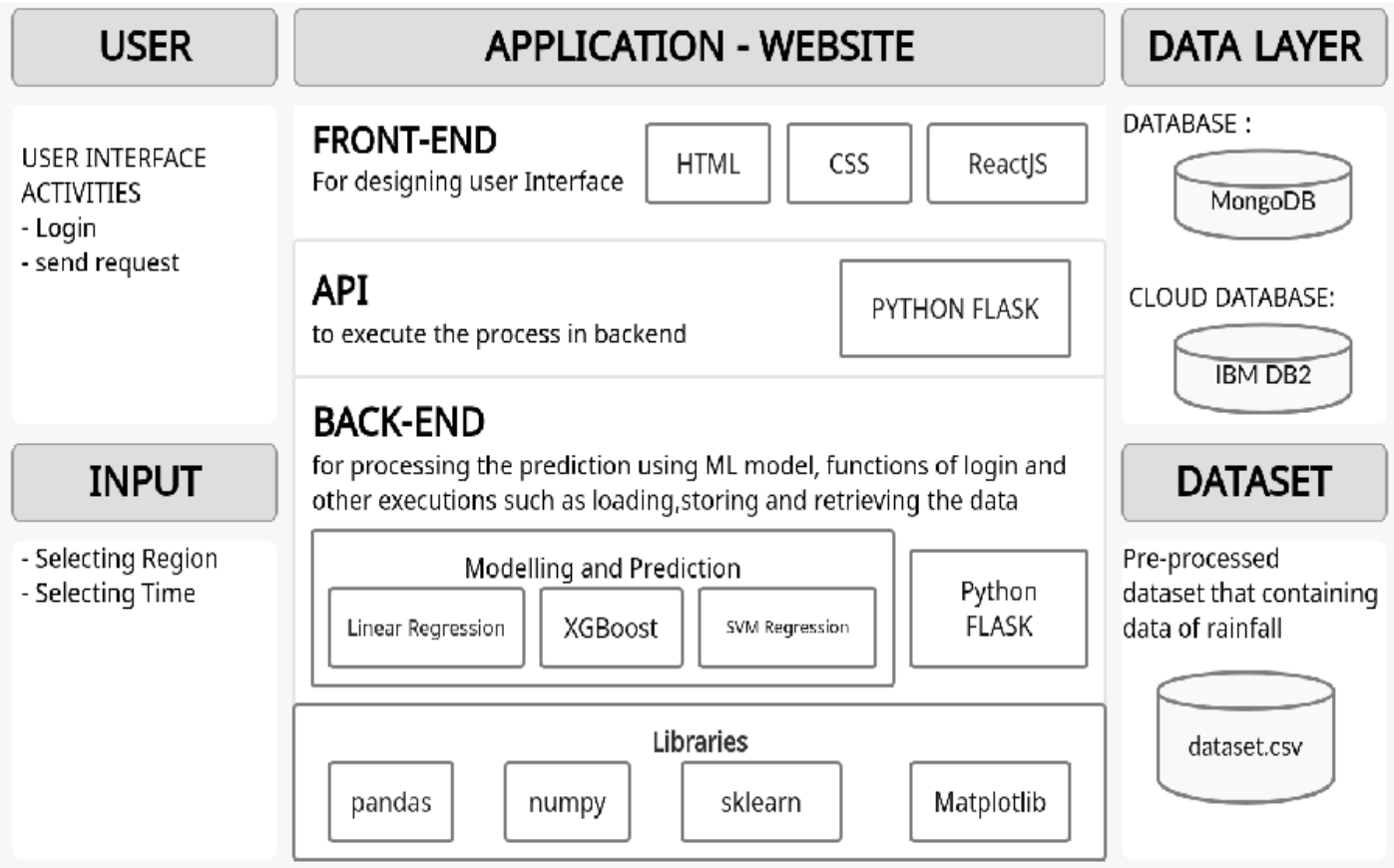


Table-1: Components & Technologies

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g., Web UI, Mobile App.	HTML, CSS, JavaScript, Bootstrap, React JS
2.	Database	The place where data can be stored and retrieved during the execution of the application	CSV Store, NoSQL
3.	Cloud database	Used for integrating components while using python flask	MongoDB Cloud
4.	API	Used to call the functions in order to access the execution in another framework	Python Flask
5.	Visualization Dashboard	Showing rainfall analysis of various regions of India monthly or annually	IBM Cognos Analytics
6.	Application Logics	Logic for each and every process in the application	Python, JavaScript
7.	Machine Learning Model	The model is developed to predict the rainfall using ML algorithms	Scikit learn Classifiers, ML Algorithms, XGBoost
8.	Data Pre-processing and Analysis	The available data is formatted or converted into the format which will be suitable for the ML model	NumPy, Matplotlib, Pandas, Seaborn

Table-2: Application Characteristics

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Backend Framework, Non-structured Database, CSS Framework styling	Python Flask, MongoDB, CSS-3
2.	Availability	The website will be made available by hosting it in cloud hosting platforms	Heroku cloud hosting (for testing) , IBM cloud hosting

5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
User in Website	Registration	USN-1	User can register for the application by entering his or her email, password, and confirming the password.	Account specific tasks and actions can be performed	High	Sprint-1
		USN-2	User will receive confirmation email or message once registered for the application	Verify the registered account	High	Sprint-1
		USN-3	Validation of the user can be done directly using email or OTP	Account validated and got access to profile dashboard	Medium	Sprint-1
	Login	USN-4	Enter the username and password to login to the application	Right account credentials should be entered	High	Sprint-1
		USN-5	The existing credentials should be used for login on multiple systems		Medium	Sprint-1
	Dashboard	USN-6	User can search for the region where he/she wants to know the prediction of rainfall	Searching for the region in India will be accepted only	High	Sprint-2
		USN-7	User can view the visualization of the rainfall data for a specific region in India or for a specific time period		Medium	Sprint-2
		USN-8	User can change his/her password and can view the account details and search history	Verification will be required and new password should be entered	High	Sprint-2
		USN-9	The prediction or analysis request can be asked for the desired region		High	Sprint-2

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
			for future or past events respectively			
		USN-10	User can give the feedback on the accuracy of the prediction and on the user interface		High	Sprint-3
Support Team	Support	USN-11	Responds to user queries via telephone, email etc.	Queries can be raised in situation of doubts	Medium	Sprint-3
		USN-12	The team must analyse all the queries and try to debug and make plans so that such queries wouldn't be raised again		Low	Sprint-3
		USN-13	Organize for a FAQ session where commonly asked doubts can be redressed by the team	The user will get all their doubt clarified	Low	Sprint-3
		USN-14	The team must respond immediately to the queries based on the priority	Queries should get resolved	High	Sprint-3
Core Development Team	Core Function	USN-13	Design, develop the application in such a way that the best user interface and maintenance should be taken care of.	Easy and self-understandable user interface	High	Sprint-4
		USN-14	The website is responsive on all the devices and the screen sizes	User experience should be good irrespective of the devices or platforms	Medium	Sprint-4
		USN-15	The updates should be on time with the solutions of the raised queries	The existing functionalities should not affected by the update	High	Sprint-4

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement [Epic]	User Story Number	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	5	High	Vivekanandhan, Soumen
Sprint-1		USN-2	3	High	Rakesh, Sudharsan
Sprint-1		USN-3	2	Medium	Rakesh, Soumen
Sprint-1	Login	USN-4	2	High	Rakesh, Sudharsan
Sprint-1		USN-5	1	Medium	Sudharsan, Vivekanandhan
Sprint-1	Dashboard	USN-6	5	High	Vivekanandhan, Soumen
Sprint-2		USN-7	3	Medium	Vivekanandhan, Soumen
Sprint-2		USN-8	5	High	Vivekanandhan, Rakesh
Sprint-2		USN-9	8	High	Sudharsan, Rakesh, Soumen
Sprint-3		USN-10	5	High	Vivekanandhan, Rakesh
Sprint-3	Support	USN-11	2	Medium	Sudharsan, Rakesh
Sprint-3		USN-12	1	Low	Sudharsan, Soumen
Sprint-3		USN-13	1	Low	Rakesh, Vivekanandhan
Sprint-3		USN-14	5	High	Rakesh, Sudharsan
Sprint-4	Core Function	USN-13	8	High	Vivekanandhan, Soumen
Sprint-4		USN-14	2	Medium	Vivekanandhan, Rakesh, Sudharsan
Sprint-4		USN-15	5	High	Rakesh, Sudharsan

6.2 Sprint Delivery Schedule

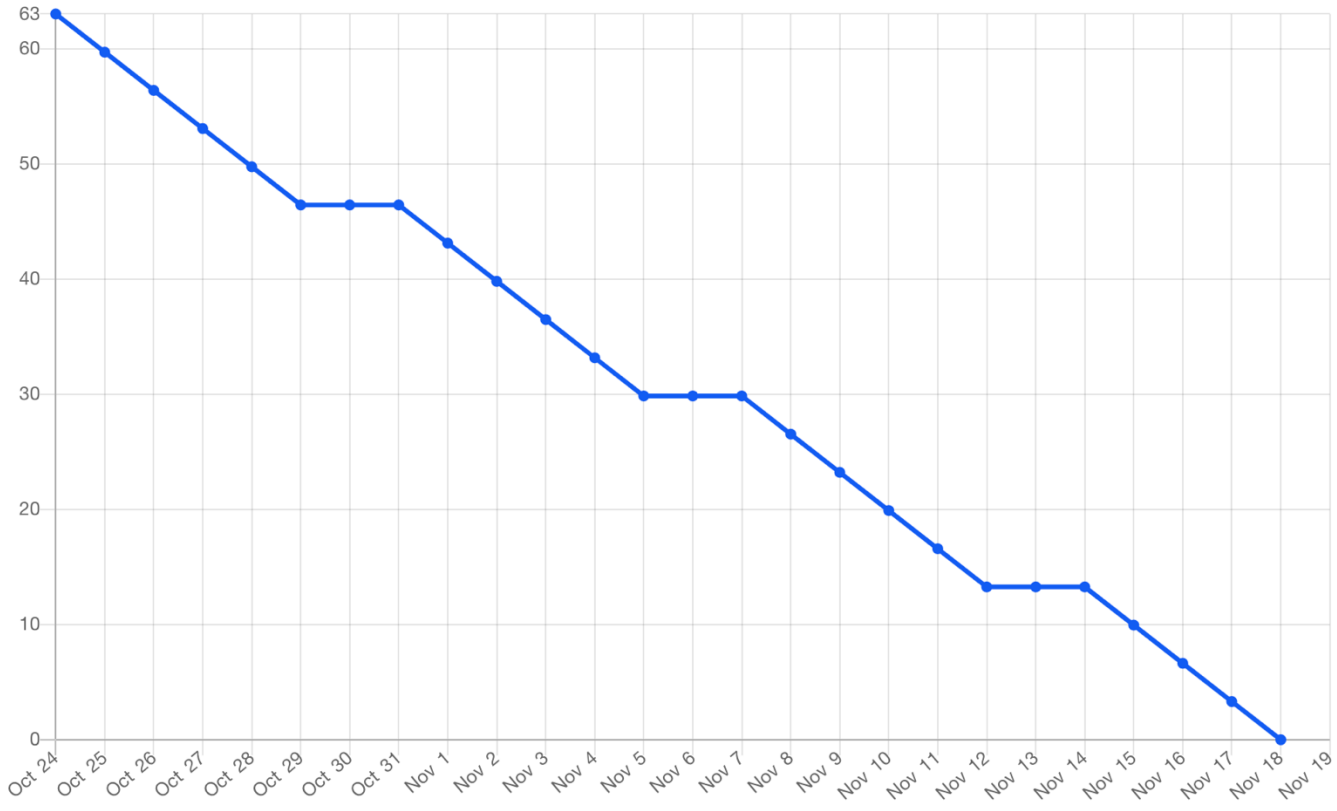
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	18	6 Days	24 Oct 2022	29 Oct 2022	18	29 Oct 2022
Sprint-2	16	6 Days	31 Oct 2022	05 Nov 2022	16	05 Nov 2022
Sprint-3	14	6 Days	07 Nov 2022	12 Nov 2022	14	12 Nov 2022
Sprint-4	15	6 Days	14 Nov 2022	19 Nov 2022	15	19 Nov 2022

Velocity:

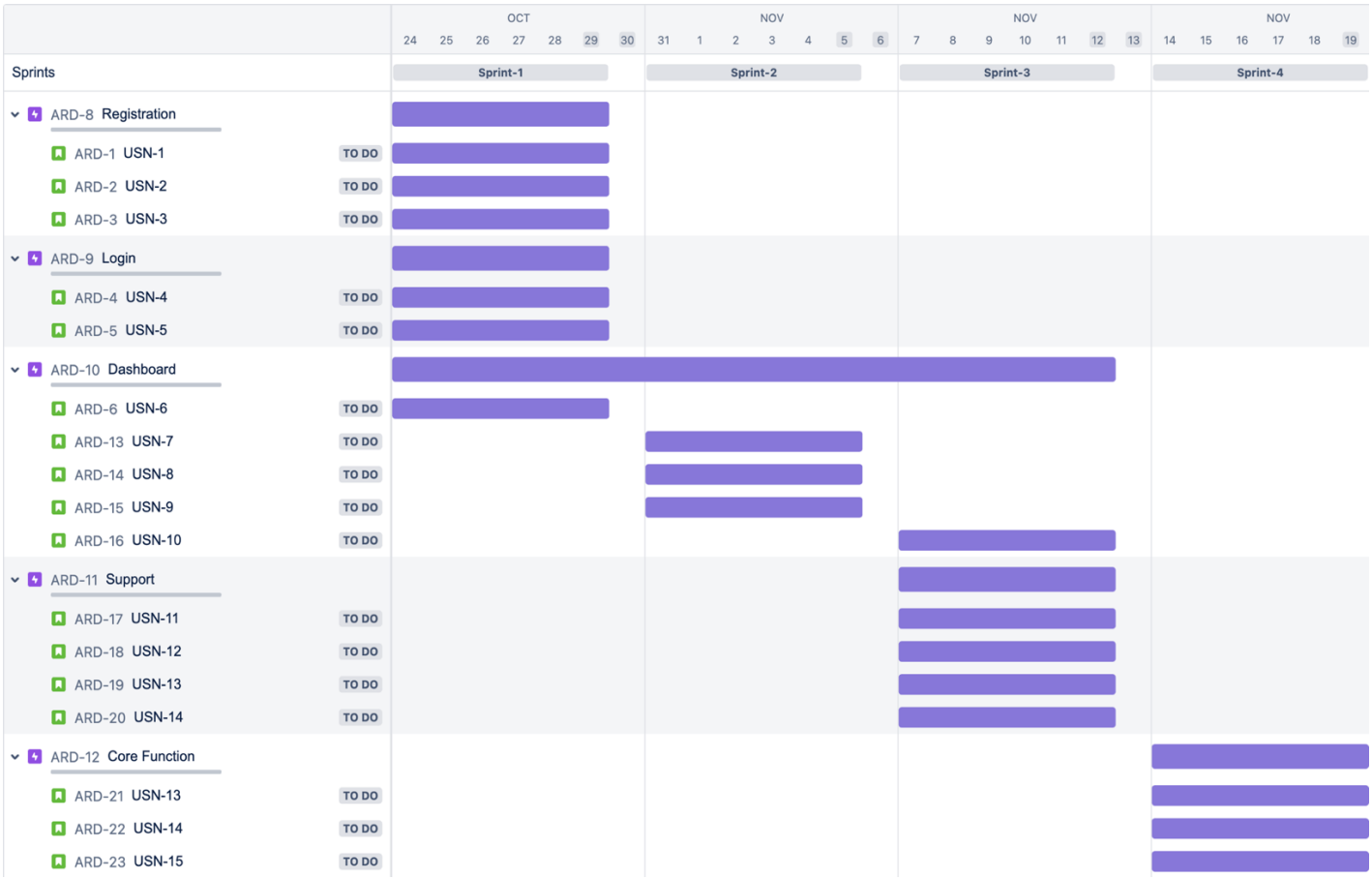
Average Sprint Velocity *[estimated to be ideal]* $= \frac{\text{Story points to be completed out of all user stories}}{\text{Total number of Sprints}} = \frac{63}{4} = 15.75$

Therefore, the amount of work to be done on each sprint is an average of 15.75 story points.

Burndown Chart:



6.3 Reports from JIRA



CHAPTER 7

CODING & SOLUTIONING

7.1 User Registration & Login

```
import React,{Component} from "react";
import { withRouter } from "react-router-dom";
import Modal from 'react-modal';
import './Home.css'

const CustomStyles = {
    content : {
        top: '50%',
        left: '50%',
        right: 'auto',
        bottom: 'auto',
        marginRight: '-50%',
        transform: 'translate(-50%, -50%)',
        border: '2px solid tomato',
        width: '350px',
        fontFamily: 'Montserrat'
    }
}

class Header extends Component{
    constructor(){
        super();
        this.state = {
            clickedNavIcon : false,
            isLoginModalOpen : false,
            isSignupModalOpen : false
        }
    }

    handleClick = () =>{
        const {clickedNavIcon} = this.state;
        this.setState({
            clickedNavIcon : !clickedNavIcon
        });
    }
    loginOpen = () =>{
        this.setState({
            isLoginModalOpen : true
        })
    }
    loginClose = () =>{
        this.setState({
```

```
            isLoginModalOpen : false
        })
    }
    SignUpOpen = () =>{
        this.setState({
            isSignupModalOpen : true
        })
    }
    SignUpClose = () => {
        this.setState({
            isSignupModalOpen : false
        })
    }
    LoginOpenfromSignup = () => {
        this.setState({
            isSignupModalOpen : false,
            isLoginModalOpen : true
        })
    }
    signOpenFromLogin = () =>{
        this.setState({
            isLoginModalOpen:false,
            isSignupModalOpen : true
        })
    }
    render(){
        const {clickedNavIcon,isLoginModalOpen,isSignupModalOpen} = this.state;
        return(
            <div id="NavBar" style={{'fontFamily':'Montserrat'}}>
                <span className="logoWrite">IBM</span>
                <div className="NavIcon" onClick={this.handleClick}>
                    <i className={clickedNavIcon ? 'bi bi-x' : 'bi bi-list'}></i>
                </div>
                <div className="logn">
                <button id="login"className="login" onClick={this.loginOpen}>Login</button>
                <button id="signup" className="signup" onClick={this.SignUpOpen}>Create an
account</button>
                </div>
                <div className={'NavMenu' + (clickedNavIcon ? ' yes' : '')}>
                    <a href="/" className="NavLinks">Home</a>
                    <a href="/" className="NavLinks">Lorem</a>
                    <a href="/" className="NavLinks">ipsum</a>
                    <a href="/" className="NavLinks">other</a>
                </div>

                <Modal isOpen={isLoginModalOpen} style={CustomStyles}>
                    <div className="head" style={{'fontSize':'22px'}}>Login</div>
                    <i className="bi bi-x closebtn" onClick={this.loginClose}></i>
                    <label className="label">Username</label>
                    <input type="text" className="input" />
                    <label className="label">Password</label>
                    <input type="text" className="input"  style={{'marginBottom':'5px'}}/>
                    <a className="forgetPassword">forgot password?</a>
```

```
                <button className="btn-primary btnLogin">LOGIN</button>
                <p className="para">If you don't have an account,<a className="paraLink"
onClick={this.signOpenFromLogin}> create an account</a></p>
            </Modal>
            <Modal isOpen={isSignupModalOpen} style={CustomStyles}>
            <div className="head">Create Your Account</div>
                <i className="bi bi-x closebtn" onClick={this.SignUpClose}></i><br/>
                <label className="label">Username</label>
                <input className="input" type="text"/>
                <label className="label">Email</label>
                <input  className="input" type="email"/>
                <label className="label">Create Password</label>
                <input type="text"  className="input" />
                <button className="btn-primary btnLogin">SIGN UP</button>
                <p className="para">Already having an account ?<br/><a className="paraLink"
onClick={this.LoginOpenfromSignup}>Login</a></p>
            </Modal>
        </div>
      )
   }
}

export default withRouter(Header);
```

7.2 Rainfall Predictor

```
import React,{ Component } from "react";
import { withRouter} from "react-router-dom";
import "./Test.css";
import {locations} from "./locationArray";
import Header from "../Home/Header";

class Test extends Component{


   prediction = () =>{
      const location = document.getElementById("location").value;
      const date = document.getElementById("date").value;
      const req ={
         'location':location
      }
      console.log(location)
      fetch("/locate",{
         'method':"POST",
         headers:{"Content-Type":"application/json"},
         body:JSON.stringify(req)
      }).then(res => res.json())
      .then(res=>{
         const url = `/result?output=${res.result}&date=${date}`;
         this.props.history.push(url);
         window.location.reload();
```

```
    })


    }
render(){
return(
    <>
    <Header/>
        <div className="outbox">
            <div className="container">
                <div className="dummy"></div>
                <div className="row whitebox">
                <h4 id="heading">Predictor</h4>
                    <div className="col-lg-6 col-md-6 col-12">
                        <label className="label">Date</label><br/>
                        <input type="date" className="input" id="date"/>
                        <label className="label">Maximum Temperature</label><br/>
                        <input type="number" className="input" id="mt"/>
                        <label className="label">Evaporation</label><br/>
                        <input type="number" className="input" id="ev"/>
                        <label className="label">Wind Gust Speed</label><br/>
                        <input type="number" className="input" id="wgs"/>
                        <label className="lable">Wind Gust Direction</label>
                        <select className="selection" name="cars" id="wgd">
                            <option value="north">North</option>
                            <option value="south">South</option>
                            <option value="east">East</option>
                            <option value="west">West</option>
                            <option value="north-west">North-West</option>
                            <option value="south-west">South-West</option>
                            <option value="north-east">North-East</option>
                            <option value="south-west">South-East</option>
                        </select><br/>
                        <label className="label">Wind Speed 3pm</label><br/>
                        <input type="number" className="input" id="wp3"/>
                        <label className="label">Humidity 3pm</label><br/>
                        <input type="number" className="input" id="h3"/>
                        <label className="label">Pressure 3pm</label><br/>
                        <input type="number" className="input" id="p3"/>
                        <label className="label">Temperature 3pm</label><br/>
                        <input type="number" className="input" id="t3"/>
                        <label className="label">Cloud 3pm</label><br/>
                        <input type="number" className="input" id="c3"/>
                        <label className="lable">Wind Direction at 3pm</label>
                        <select className="selection" name="cars" id="wd3">
                            <option value="north">North</option>
                            <option value="south">South</option>
                            <option value="east">East</option>
                            <option value="west">West</option>
                            <option value="north-west">North-West</option>
                            <option value="south-west">South-West</option>
                            <option value="north-east">North-East</option>
                            <option value="south-west">South-East</option>
```

```
            </select>
          </div>
          <div className="col-lg-6 col-md-6 col-12">
            <label className="label">Minimum Temperature</label><br/>
            <input type="number" className="input" id="mit"/>
            <label className="label">Rainfall</label><br/>
            <input type="number" className="input" id="rf"/>
            <label className="label">Sunshine</label><br/>
            <input type="number" className="input" id="ss"/>

            <label className="label">Wind Speed 9am</label><br/>
            <input type="number" className="input" id="wp9"/>
            <label className="lable">Rain Today</label>
            <select className="selection" name="cars" id="rt">
              <option value="no">No</option>
              <option value="yes">Yes</option>
            </select><br/>
            <label className="label">Humidity 9am</label><br/>
            <input type="number" className="input" id="h9"/>
            <label className="label">Pressure 9am</label><br/>
            <input type="number" className="input" id="p9"/>
            <label className="label">Temperature 9am</label><br/>
            <input type="number" className="input" id="t9"/>
            <label className="label">Cloud 9am</label><br/>
            <input type="number" className="input" id="c9"/>
            <label className="lable">Wind Direction at 9am</label>
            <select className="selection" name="cars" id="wd9">
              <option value="north">North</option>
              <option value="south">South</option>
              <option value="east">East</option>
              <option value="west">West</option>
              <option value="north-west">North-West</option>
              <option value="south-west">South-West</option>
              <option value="north-east">North-East</option>
              <option value="south-west">South-East</option>
            </select><br/>
            <label className="lable">Location</label>
            <select className="selection" name="cars" id="location">
              {
                locations.map((item)=>{
                  return(
                    <option value={item.name}>{item.name}</option>
                  )
                })
              }
            </select>
          </div>
          <a className="prediction" onClick={this.prediction}>Predict</a>

        </div>
        <div className="dummy"></div>
      </div>
```

```
        </div>
        </>
    )
}
}

export default withRouter(Test);
```

7.3 User Feedback

```
import React,{Component} from "react";
import querString from "query-string";
import {withRouter} from "react-router-dom";
import "./Test.css"

class Result extends Component{
    constructor(){
        super();
        this.state = {
            output:"",
            date:""
        }
    }
    componentDidMount(){
        const qs = querString.parse(this.props.location.search);
        const {output,date} = qs;
        console.log(output === "Sunny day")
        this.setState({
            output:output,
            date:date
        })
    }
    render(){
        const {output,date} =  this.state;
    return(
        <>
        <div className="output">The day {date} is <span className="result">{output}</span> !!

        </div>
        {
            output === "Sunny day" ?
            <div style={{"textAlign":"center"}}>
            <img className="resultmage" src={require('../../Images/download.jpeg')} />
            </div>
            :
            <div style={{"textAlign":"center"}}>
            <img className="resultmage" src={require('../../Images/rainy.jpeg')}/>
            </div>
        }
```

```
        <div style={{"textAlign":"center"}}>
        <a href="/test" className="button">RETURN TO PREDICTION</a>
        </div>
        <div style={{"textAlign":"center"}} className="feedBack">
            Feedback<br/>
            <textarea className="feedback"></textarea><br/>
            <button className="btn-primary submit-b">Submit</button>
        </div>
        </>
    )
}
}

export default withRouter(Result);
```

7.4 Database & Server Operations

```
from crypt import methods
from flask import Flask,request,jsonify
from pymongo import MongoClient
from mongopass import mongopass
from bson.objectid import ObjectId
import pickle
import numpy as np
import random
import warnings


app = Flask(__name__)

warnings.filterwarnings('ignore')

scaler_path = "./scale.pkl"
model_path = "./rainfall.pkl"
encoder_path = "./encoder.pkl"
data_path = "./data.pkl"

client = MongoClient(mongopass)
db = client.curd
myCollection = db.myColl

@app.route("/signup",methods=["POST"])
def signup():
    name = request.json['user']
    email = request.json['email']
    password = request.json['password']
    myVal = {"name":name,"email":email,"password":password}
    x = myCollection.insert_one(myVal)
    return jsonify({"message":"User Created Successfully"})

@app.route("/login",methods=["POST"])
```

```
def login():
    x = list(myCollection.find({"email":request.json['email'],"password":request.json['password']}))

    for i in x:
        i["_id"]=str(i["_id"])
    d = x[0]
    return jsonify(d)

@app.route("/get",methods=["GET"])
def get():
    return jsonify({"he":"wo"})

def make_prediction(test_data):
  scaler_custom_loaded = pickle.load(open(scaler_path,'rb'))
  model_custom_loaded = pickle.load(open(model_path,'rb'))
  x_test_data = np.array(test_data).reshape(1,-1)
  x_test_data = scaler_custom_loaded.transform(x_test_data)
  prediction = model_custom_loaded.predict(x_test_data.reshape(1,-1))[0]
  if prediction == 1:
    st = "Rainy day"
  else:
    st = "Sunny day"
  return st

def data_preprocessing(test_data_location):
  lencoders = pickle.load(open(encoder_path,'rb'))
  features = pickle.load(open(data_path,'rb'))
  location = lencoders['Location'].transform([test_data_location])[0]
  Data = features[features['Location']==location]
  x_test_data_series = Data.iloc[random.randint(0,len(Data))]
  return x_test_data_series

def prediction(location):
  preprocessed_data = data_preprocessing(location)
  p = make_prediction(preprocessed_data)
  return p

@app.route("/locate",methods=["POST"])
def locate():
    location = request.json['location']
    x = prediction(location)
    return jsonify({"result":x})


if __name__ == "__main__":
    app.run(debug=True)
```

CHAPTER 8

TESTING

8.1 Test Cases

				Date	28-Oct-22								
				Team ID	PNT2022TMID12561								
				Project Name	Exploratory Analysis of RainFall Data in India for Agriculture.								
				Maximum Marks	4 marks								
Test suite ID	**Feature Type**	**Component**	**Test Scenario**	**Pre-Requisite**	**Steps To Execute**	**Test Data [for a random Test Case]**	**Expected Result**	**Actual Result**	**Status**	**Commnets**	**TC for Automation(Y/N)**	**BUG ID**	**Executed By**
CreateAccount_TC_OO1	UI	Create Account Page	Verify user is able to see the 'Login if already an user' message when user clicked on Create account button	NIL	1. Click on Create Account button from the nav bar 3. Verify login link displayed or not.	localhost:3000	Login if already an user otherwise proceed with creation of account	Working as expected	Pass	NIL	N	NIL	Soumen, Vivekanadhan
LoginPage_TC_OO1	Functional	Login page	Verify user is able to log into application with Valid credentials	User should already have created and registerted with an account	1. Click on login button from the nav bar 2. Enter Valid username/email in Email text box 3. Enter valid password in password text box 4. Click on login button	Username: dummy@gmail.com password: vivek!@#123A	User should get message of logged in	Working as expected	Pass	NIL	N	NIL	Vivekanandhan, Sudharsan
HomePage_TC_OO1	UI	Home Page	Verify the UI elements in Navigation Bar and check the same for other devices like Mobiles for size and orientation	NIL	1. Check if any component from the navigation bar present at the top is not directed to 2. Click each of the component and check if it is directed to the respective module	NIL	Application should show below UI elements in the nav bar: a. Home b. Dashboard c. Predict d. Login e. Create Account	Working as expected	Pass	NIL	N	NIL	Rakesh
PredictPage_TC_OO1	Functional	Prediction Page	Verify user is able to give the inputs and get pop up for invalid inputs	User should be aware of the weather parameters or put the fields as empty	1. Enter valid inputs first for each field 2. Try with invalid ones randomly for each field 2. Click on the Predict button	Username: chalam@gmail password: Testing123	Application should show either of two outpus: Rainy or Sunny	Working as expected	Pass	NIL	N	NIL	Vivekanandhan, Soumen
DashboardPage_TC_OO1	Functional	Dashboard	Verify user is able to see the rainfall analysis through an interactive dashboard	Basic knowledge of Indian Political Map	1. Click on the Dashboard button present at the top bar 2. Filter data based on the month range 3. Resize the visualiation and graphs by finger gentures (if present at user hardware device) 4. Zoom in or out the map to view the location details 5.Use scroll bar to see below contents	Lower Range: 419.26 for JUN-SEP Upper Range: 490.15 for JUN-SEP	Application should show ranifall info for map, bar graph, line graphs, differnetiated with color once filter is given	Working as expected	Pass	Could be embedded in Home page itself	N	BUG001 - iframe not worked	Soumen, Vivekanadhan
FeedbackPage_TC_OO1	Functional	Feedback page	Verify user is able to give feedback based on the their views and the prediction shown as result	NIL	1. Once the result is shown, give a feedback or put forward your opinion in the Text Area field 2. Click Submit	Prediction is correct and UI is awesome....	Application should show message once clicked on Submit button.	Working as expected	Pass	NIL	N	NIL	Rakesh

8.2 User Acceptance Testing

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	6	3	1	2	12
Duplicate	0	0	0	0	0
External	1	2	0	0	3
Fixed	7	2	4	14	27
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	1	3	0	4
Totals	14	8	8	16	46

3. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	3	0	0	3
Client Application	36	0	0	36
Security	2	0	0	2
Exception Reporting	11	0	0	11
Final Output	15	0	0	15
Version Control	3	0	0	3

CHAPTER 9

RESULTS

9.1 Performance Metrics

<table>
<tr>
<td>Metrics</td>
<td>Cat Boost Classifier

Confusion Matrix –

[[21416 1301]
[2891 3484]]

Accuracy Score- 0.855905403547367

& Classification Report –
precision recall f1-score support

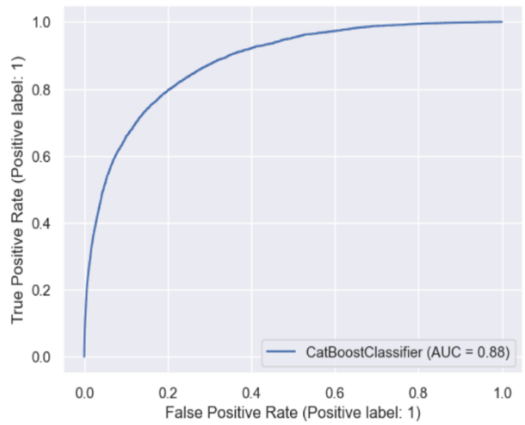
0 0.88 0.94 0.91 22717
1 0.73 0.55 0.62 6375

accuracy 0.86 29092
macro avg 0.80 0.74 0.77 29092
weighted avg 0.85 0.86 0.85 29092</td>
<td>
<pre>y_pred1 = catBoostC.predict(X_test1)
print(confusion_matrix(y_test1, y_pred1))
print()
print(accuracy_score(y_test1, y_pred1))
print()
print(classification_report(y_test1, y_pred1))</pre>
<pre>[[21416 1301]
[2891 3484]]

0.855905403547367

precision recall f1-score support

0 0.88 0.94 0.91 22717
1 0.73 0.55 0.62 6375

accuracy 0.86 29092
macro avg 0.80 0.74 0.77 29092
weighted avg 0.85 0.86 0.85 29092</pre>
<pre>metrics.plot_roc_curve(catBoostC, X_test1, y_test1)
metrics.roc_auc_score(y_test1, y_pred1, average = None)</pre>
<pre>0.744619958966551</pre>


</td>
</tr>
<tr>
<td>Hyperparameters Tuning</td>
<td>iterations = {1000, 2000, 3000,}
eval_metric = 'AUC</td>
<td>
<pre>catBoostC = CatBoostClassifier(iterations = 3000, eval_metric = 'AUC')</pre>
</td>
</tr>
</table>

CHAPTER 10

ADVANTAGES & DISADVANTAGES

One of the most noticeable benefits of weather forecasting is the ability to make proper plans. In the agricultural sector, it can help with a farmer's business decisions. Forecasts can help them plan for the many day-to-day decisions. These decisions include crop irrigation, when to fertilize, and what days are suitable for working in the field. The decisions that farmers make will result in a profitable crop or failure.

A key drawback of data-driven forecasting methods like this is the lack of physical meaning in the dataset and the lack of high-quality training data. Some physical forecast models need to use years of exact climate data pertaining to the required location in their calculations. Data-driven models with climate data from another tectonic region would not forecast weather as accurately as physical models.

CHAPTER 11

CONCLUSION

Data science and machine learning are used in the application area of rainfall prediction to foretell atmospheric conditions. In order to produce crops efficiently and decrease flood-related and other monsoon mortality, it is crucial to forecast rainfall intensity. This article examined different machine learning algorithms for predicting rainfall. Using the dataset, three machine learning algorithms were presented and evaluated. Using sensor and meteorological datasets with extra diverse environmental factors helps increase the accuracy of rainfall forecasts. Therefore, if sensors and meteorological datasets are employed for the daily rainfall amount prediction study in future work, big data analysis can be used for rainfall prediction.

CHAPTER 12

FUTURE SCOPE

Decision-making is greatly affected by the crucial data mining task of predicting future rainfall trends. The model's insights could be used to schedule the times when crops are sown and determine when to harvest them. Additionally, it can assist in foreseeing drought trends in various parts of India and build response plans to deal with them. Along with rainfall prediction, the system may also include a flood prediction system. It is possible to integrate evacuation zones with flood prediction systems so that, in the event of a flood, the system will advise both the user and the community to evacuate. It would be beneficial for society to have a recommendation system coupled with a prediction system.

CHAPTER 13

APPENDIX

Source Code:

Source code related to model building

```
# Importing the modules

# In[1]:

import numpy as np
import random
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.utils import resample
import warnings
import pickle
import time
from sklearn import preprocessing
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier as rf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.metrics import accuracy_score, roc_auc_score, cohen_kappa_score,
plot_confusion_matrix, roc_curve, classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder

import xgboost as xgb
import seaborn as sns


# Loading the data

# In[2]:
```

```
df = pd.read_csv('weatherDataset.csv')
df.head()


# In[3]:


df.columns


# Replacing the categorical columns

# In[4]:


df['RainToday'].replace({'No': 0, 'Yes': 1},inplace = True)
df['RainTomorrow'].replace({'No': 0, 'Yes': 1},inplace = True)


# Checking if data is balanced or not

# In[5]:


fig = plt.figure(figsize = (8,5))
df.RainTomorrow.value_counts(normalize = True).plot(kind='bar', color= ['skyblue','navy'], alpha =
0.9, rot=0)
plt.title('RainTomorrow Indicator No(0) and Yes(1) in the Imbalanced Dataset')
plt.show()


# Oversampling

# In[6]:


no = df[df['RainTomorrow'] == 0]
yes = df[df['RainTomorrow'] == 1]
yes_oversampled = resample(yes, replace=True, n_samples=len(no), random_state=123)
oversampled = pd.concat([no, yes_oversampled])

fig = plt.figure(figsize = (8,5))
oversampled.RainTomorrow.value_counts(normalize = True).plot(kind='bar', color=
['skyblue','navy'], alpha = 0.9, rot=0)
plt.title('RainTomorrow Indicator No(0) and Yes(1) after Oversampling (Balanced Dataset)')
```

```
plt.show()


# Handling Null Values

# In[7]:


oversampled.isna().sum()


# In[8]:


oversampled.select_dtypes(include=['object']).columns


# Imputing categorical features with mode

# In[9]:


oversampled['Date'] = oversampled['Date'].fillna(oversampled['Date'].mode()[0])
oversampled['Location'] = oversampled['Location'].fillna(oversampled['Location'].mode()[0])
oversampled['WindGustDir'] =
oversampled['WindGustDir'].fillna(oversampled['WindGustDir'].mode()[0])
oversampled['WindDir9am'] =
oversampled['WindDir9am'].fillna(oversampled['WindDir9am'].mode()[0])
oversampled['WindDir3pm'] =
oversampled['WindDir3pm'].fillna(oversampled['WindDir3pm'].mode()[0])


# Converting categorical features to continuous features with Label Encoding

# In[10]:


lencoders = {}
for col in oversampled.select_dtypes(include=['object']).columns:
    lencoders[col] = LabelEncoder()
    oversampled[col] = lencoders[col].fit_transform(oversampled[col])


# In[11]:
```

```
lencoders
pickle.dump(lencoders,open('encoder.pkl','wb'))


# Multiple Imputation by Chained Equation

# In[12]:


MiceImputed = oversampled.copy(deep=True)
mice_imputer = IterativeImputer()
MiceImputed.iloc[:, :] = mice_imputer.fit_transform(oversampled)


# Detecting outliers with IQR

# In[13]:


Q1 = MiceImputed.quantile(0.25)
Q3 = MiceImputed.quantile(0.75)
IQR = Q3 - Q1
print(IQR)


# Removing outliers from the dataset

# In[14]:


MiceImputed = MiceImputed[~((MiceImputed < (Q1 - 1.5 * IQR)) |(MiceImputed > (Q3 + 1.5 *
IQR))).any(axis=1)]
MiceImputed.shape


# Data Standardization

# In[15]:


r_scaler = preprocessing.MinMaxScaler()
r_scaler.fit(MiceImputed)
modified_data = pd.DataFrame(r_scaler.transform(MiceImputed), index=MiceImputed.index,
columns=MiceImputed.columns)
```

```
# Feature Importance using Filter Method

# In[16]:


X = modified_data.loc[:,modified_data.columns!='RainTomorrow']
y = modified_data[['RainTomorrow']]
selector = SelectKBest(chi2, k=10)
selector.fit(X, y)
X_new = selector.transform(X)
print(X.columns[selector.get_support(indices=True)])


# In[17]:


X = MiceImputed.drop('RainTomorrow', axis=1)
y = MiceImputed['RainTomorrow']
selector = SelectFromModel(rf(n_estimators=100, random_state=0))
selector.fit(X, y)
support = selector.get_support()
features = X.loc[:,support].columns.tolist()
print(features)
print(rf(n_estimators=100, random_state=0).fit(X,y).feature_importances_)


# Splitting into test and train

# In[18]:


features = MiceImputed[['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
'WindGustDir',
                        'WindGustSpeed', 'WindDir9am', 'WindDir3pm', 'WindSpeed9am', 'WindSpeed3pm',
'Humidity9am',
                        'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
'Temp3pm',
                        'RainToday']]
target = MiceImputed['RainTomorrow']

X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.25,
random_state=12345)


# In[19]:
```

```
# Normalize Features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)


# In[20]:


# Function to plot the roc_curve
def plot_roc_cur(fper, tper):
    plt.plot(fper, tper, color='orange', label='ROC')
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend()
    plt.show()


# In[21]:


def run_model(model, X_train, y_train, X_test, y_test, verbose=True):
    t0=time.time()
    if verbose == False:
        model.fit(X_train,y_train, verbose=0)
    else:
        model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    roc_auc = roc_auc_score(y_test, y_pred)
    coh_kap = cohen_kappa_score(y_test, y_pred)
    time_taken = time.time()-t0
    print("Accuracy = {}".format(accuracy))
    print("ROC Area under Curve = {}".format(roc_auc))
    print("Cohen's Kappa = {}".format(coh_kap))
    print("Time taken = {}".format(time_taken))
    print(classification_report(y_test,y_pred,digits=5))

    probs = model.predict_proba(X_test)
    probs = probs[:, 1]
    fper, tper, thresholds = roc_curve(y_test, probs)
    plot_roc_cur(fper, tper)
```

```
    plot_confusion_matrix(model, X_test, y_test,cmap=plt.cm.Blues, normalize = 'all')

    return model, accuracy, roc_auc, coh_kap, time_taken


# Using Logistic Regression

# In[22]:


params_lr = {'penalty': 'l1', 'solver':'liblinear'}

model_lr = LogisticRegression(**params_lr)
model_lr, accuracy_lr, roc_auc_lr, coh_kap_lr, tt_lr = run_model(model_lr, X_train, y_train,
X_test, y_test)


# Using Decision tree

# In[23]:


params_dt = {'max_depth': 16,
            'max_features': "sqrt"}

model_dt = DecisionTreeClassifier(**params_dt)
model_dt, accuracy_dt, roc_auc_dt, coh_kap_dt, tt_dt = run_model(model_dt, X_train, y_train,
X_test, y_test)


# Using Random Forest

# In[24]:


params_rf = {'max_depth': 16,
            'min_samples_leaf': 1,
            'min_samples_split': 2,
            'n_estimators': 100,
            'random_state': 12345}

model_rf = RandomForestClassifier(**params_rf)
model_rf, accuracy_rf, roc_auc_rf, coh_kap_rf, tt_rf = run_model(model_rf, X_train, y_train,
X_test, y_test)
```

```
# In[25]:


accuracy_scores = [accuracy_lr, accuracy_dt, accuracy_rf]
roc_auc_scores = [roc_auc_lr, roc_auc_dt, roc_auc_rf ]
coh_kap_scores = [coh_kap_lr, coh_kap_dt, coh_kap_rf]
tt = [tt_lr, tt_dt, tt_rf]

model_data = {'Model': ['Logistic Regression','Decision Tree','Random Forest'],
              'Accuracy': accuracy_scores,
              'ROC_AUC': roc_auc_scores,
              'Cohen_Kappa': coh_kap_scores,
              'Time taken': tt}
data = pd.DataFrame(model_data)

fig, ax1 = plt.subplots(figsize=(12,10))
ax1.set_title('Model Comparison: Accuracy and Time taken for execution', fontsize=13)
color = 'tab:green'
ax1.set_xlabel('Model', fontsize=13)
ax1.set_ylabel('Time taken', fontsize=13, color=color)
ax2 = sns.barplot(x='Model', y='Time taken', data = data, palette='summer')
ax1.tick_params(axis='y')
ax2 = ax1.twinx()
color = 'tab:red'
ax2.set_ylabel('Accuracy', fontsize=13, color=color)
ax2 = sns.lineplot(x='Model', y='Accuracy', data = data, sort=False, color=color)
ax2.tick_params(axis='y', color=color)


# Model Comparision

# In[26]:


fig, ax3 = plt.subplots(figsize=(12,10))
ax3.set_title('Model Comparison: Area under ROC and Cohens Kappa', fontsize=13)
color = 'tab:blue'
ax3.set_xlabel('Model', fontsize=13)
ax3.set_ylabel('ROC_AUC', fontsize=13, color=color)
ax4 = sns.barplot(x='Model', y='ROC_AUC', data = data, palette='winter')
ax3.tick_params(axis='y')
ax4 = ax3.twinx()
color = 'tab:red'
ax4.set_ylabel('Cohen_Kappa', fontsize=13, color=color)
ax4 = sns.lineplot(x='Model', y='Cohen_Kappa', data = data, sort=False, color=color)
ax4.tick_params(axis='y', color=color)
```

```
plt.show()

# In[27]:

features.iloc[100]

# In[28]:

lencoders = pickle.load(open('encoder.pkl','rb'))
lencoders

# In[29]:

features.columns

# In[30]:

lencoders.keys()

# In[31]:


def make_prediction(scaler_path,model_path,test_data):
    scaler_custom_loaded = pickle.load(open(scaler_path,'rb'))
    model_custom_loaded = pickle.load(open(model_path,'rb'))
    x_test_data = np.array(test_data).reshape(1,-1)
    x_test_data = scaler_custom_loaded.transform(x_test_data)
    prediction = model_custom_loaded.predict(x_test_data.reshape(1,-1))[0]
    if prediction == 1:
        print('It will Rain')
    else:
        print('It wont Rain tomorrow')

# In[32]:

def data_preprocessing(data):
    lst = ['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
       'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
       'Temp3pm', 'RainToday']
    for col in ['Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm']:
        data[col] = lencoders[col].transform([data[col]])[0]
    return data

df = df[['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
```

```
    'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
    'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
    'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
    'Temp3pm', 'RainToday']]


# In[33]:

# Saving Model Binary File

pickle.dump(features,open('data.pkl','wb'))
```

Cognos Dashboard Link

https://dataplatform.cloud.ibm.com/dashboards/0ab13048-a068-4321-8eb9-cd173fd6e8c4/view/7116e02d319402d341cbd0e407cc7a577561735bb3bb800182837b490d367597f06c1192c82b4c5cdf125735a6e41b0a9d

GitHub & Project Demo Link

GitHub Repository Link: https://github.com/IBM-EPBL/IBM-Project-13205-1659514193

Demo Video Link: https://drive.google.com/file/d/1fJgO9BPq9AWUuiNGbdiuCW55qM6CqZ7d/view?usp=share_link