**TEAM DELIVERABLE 5 DOCX- GROUP11 SE PROJECT**

**FUNCTIONAL REQUIRMENTS:**

|  |  |  |  |
| --- | --- | --- | --- |
| # | FR name | FR description | phase |
| FR1 | Data Selection | The **user** can **choose** the current or **present data** | Phase 1 |
| FR2 | Prediction history | The system shall render the 3 recent predictions made by the user. | Phase 1 |
| FR3 | Dashboard | The system shall display rainfall over months vs. Annual and plot of graphs, scatterplot, and bar graph are driven to visualize the data. | Phase 2 |
| FR4 | Year Slider | A slider was provided for user to select the year to visualize the rainfall occurred in the Respective year. | Phase 2 |
| FR5 | Save Dashboard results | User can save the dashboard images. | Phase 2 |
| FR6 | Data Entry | The user can enter the recorded data of rainfall. | Phase 3 |
| FR7 | User clicks | The user can predict by entering values and clicking the predict interface | Phase 3 |

We divided our project modules into 3 phases

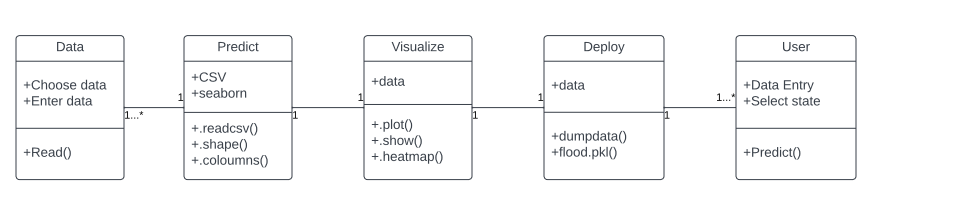
Phase- 1: Phase 1 involves pre-processing the data using procedures for data validation. Python's Panda package will be used for a number of data cleansing tasks. By using these data cleaning techniques and viewing the gathered data, the missing values or outliers will be located.

Phase-2: We have included some machine learning techniques like Decision tree, Random forest algorithm and logistic Regression algorithms

Phase-3: In phase 3, we use the Python-based Flask web framework to put our code into practice.

The source code is run, and flood prediction is performed, using the local host address.

**Class Diagram:**



**Use Case diagram: Diagram

Description automatically generated**

**Sequence Diagram:**

**A picture containing diagram

Description automatically generated**

**Test Cases:**

**UNIT TESTING:**

**Test case1:**

def test\_data\_columns():

data = pd.read\_csv('demo.csv')

assert all(data.columns == ['Unnamed: 0', 'STATE\_UT\_NAME', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN',

'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC', 'ANNUAL', 'Jan-Feb',

'Mar-May', 'Jun-Sep', 'Oct-Dec', 'flood', 'Avg\_june10days',

'maytojune'])

Result: Test Pass

**Test Case2:**

def test\_data\_columns():

data = pd.read\_csv('demo.csv')

assert all(data.columns == ['Unnamed: 0', 'STATE\_UT\_NAME', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN',

'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC', 'ANNUAL', 'Jan-Feb',

'Mar-May', 'Jun-Sep', 'Oct-Dec', 'flood', 'Avg\_june10days',

'maytojune'])

assert isinstance (data.index, pd.Datetimeindex)

Result: Assertion Error: Fail

**Test Cased3:**

def test\_data\_missingcolumns():

data = pd.read\_csv('demo.csv')

assert all(data.columns == ['Unnamed: 0', 'STATE\_UT\_NAME', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN',

'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC', 'ANNUAL', 'Jan-Feb',

'Mar-May', 'Jun-Sep', 'Oct-Dec', 'flood', 'Avg\_june10days',

])

Result : Expected Values(22),(23)

Our model mainly focuses on whether the flood will happen or not happens. Here, the user will select the state and give the climate conditions and upon clicking on predict button the result will be displayed.

We had collected the dataset and tested 30% of the data by four different algorithms and deployed it using flask web framework. Among those algorithms, Random Forest Algorithm gave the accuracy of 96.8% and hence this algorithm will be used for prediction of floods.

The probability relation between each other is as below after testing the data:

Chart, treemap chart

Description automatically generated

We have used the random forest algorithm and below is the classification report for the same.Table

Description automatically generated

The graph of confusion matrix of the tested data is below:

Chart

Description automatically generated

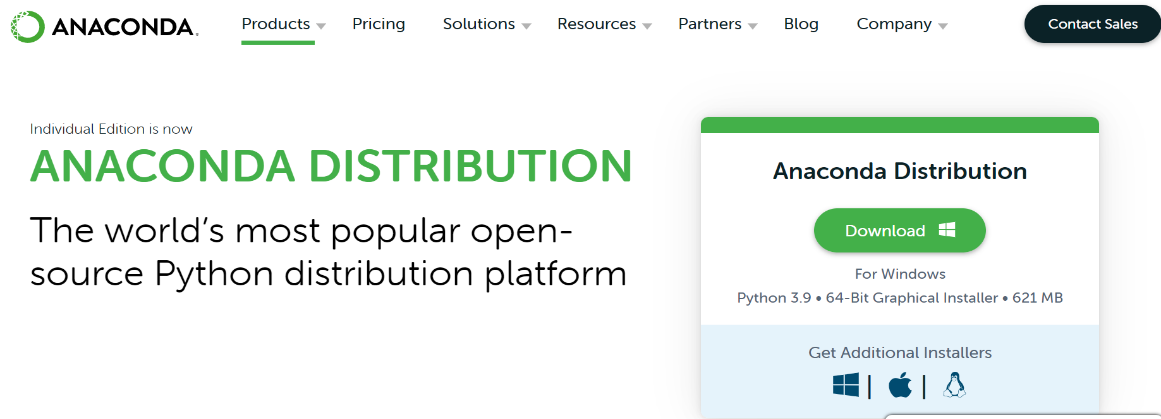
**User manual:**

This is the thing where we are going to explain how we are using this program, it is an end-end user for the software system.

**1**: **The end user need to install anaconda navigator**

This enables users to run applications and manage Conda packages, environments, and channels without using command-line tools

* Installing Anaconda



* Open the downloaded file when done.
* Then click next and agree the licenseGraphical user interface, text, application, chat or text message, email

  Description automatically generated
* If you don’t want to install for all users, select Just for me.
* Select the destination folder for anaconda
* Select the path environment variable. Adding anaconda to your path environment variable is not recommended, because it may interfere with other software.
* Choose anaconda as default browser.
* After this go to windows and search for anaconda prompt
* Then, type jupyter notebook, jupyter will be launched

Graphical user interface

Description automatically generated

* Then open jupyter notebook

Graphical user interface, text, application, email

Description automatically generated

* Anaconda installation is complete.

**2: Running the jupyter notebook:**

* After opening the Jupyter notebook
* To create a new notebook select on new and click on the type of file you need
* A new jupyter notebook will be opened in another tab.
* When launched, the Jupyter Notebook App can only access files in its start-up folder (including any sub-folder). There is no need to configure anything if you save your notebooks in your home folder or any subfolders. Or else, choose a start-up folder for the Jupyter Notebook App that will contain all of the notebooks.

**3: Save notebooks**

* To save click on the disk symbol present on the top left side of the page
* To make a copy of the notebook document (menu file -> make a copy...)

**4: Executing the notebook**

* Open the Notebook dashboard and navigate to the notebook. When you click it, the tab will open.
* To view a description of the notebook, go to Help -> User Interface Tour. It will show you about the Jupyter Notebook.
* Holding shift and entering allows you to run one cell at a time.
* You can run the entire notebook at once by selecting Cell -> Execute All from the menu.
* Select Kernel -> Restart from the menu to restart the kernel, also known as the computational engine. This is useful if you want to rerun a calculation from the beginning (by deleting variables or closing open files, for example).

**Compilation Instructions:**

We have used the anaconda to compile and run the code that we had written, first of all we need anaconda navigator

**Step1**: Go to start menu and search or type anaconda navigator, after you find it click to open

**Step2**: To launch jupyter notebook, when the Anaconda Prompt app opens, navigate to the required folder and type jupyter notebook in the cd command.

**Step3**: Open a new Python file in Jupyter notebook

**Step4**: Write each individual developed code into appropriate module.

**Step5**:Upload all the data sets into jupyter notebook

* **Module 1**: Data preprocessing / Cleaning

Set up all of the data sets into a Jupyter notebook. Pandas, a Python library, is used to perform data cleaning tasks, with a focus on cleaning tasks and outliers. It prefers to spend less time cleaning data and more time experimenting and modeling.

Finally, this transforms raw data into clear and understandable information.

* **Module 2:** Data Visualization

For easy comprehension of the flow, the entire data set that we cleaned and filtered is represented graphically.

* **Module 3**: Logistic regression

During this process, the data that we collected will be processed and displayed in the form of estimated models. In general, this displays binary outcomes, such as yes or no, based on previously collected data sets.

* **Module 4:** Decision-tree

This reads the data and partitions it; based on the results, it makes decisions and implements the algorithm. The data is split until the termination condition is met.

* **Module 5:** Random forests classifier

It will perform classification and regression as it constructs trees from various data samples.

* **Module 6:** support Vector Machine

The line will be plotted using data divided into parts, and the prediction will be done based on the results.

**Step6:** Deployment: Flask Framework

When we run the deployment file after successfully running the modules, we will get output presented with a link to the file's location, i.e., a local host link.

**Step7:** We will be shown the state name, months, and annual blocks after clicking the link. The state, month, and other details must then be chosen. Then, based on the input data, we should click predict, and the user will be notified.

**Step8:** Finally, a machine learning algorithm will be used to forecast the flood through the flask deployment.

**Dashboard View:**

Step1: Select the Interactive ipynb file from the git and upload in the Jupyter notebook.

Step2: Run all the cells in the notebook to get the output.

Step3: Run the command panel ‘serve <Module name>. ipynb’

Step4: The dashboard can be viewed the rainfall data will be visualized.

**Implemented Features:**

* Implemented code through flask web framework.
* Random Forest algorithm is used to predict the flood.
* Flood is predicted using past data analysis with present data inputs.

**Limitations:**

* It is not recommended or suitable for big applications
* No admin sites are provided.
* No login or authentication is provided
* Difficulties in migrations

**Unimplemented Features:**

* We can add login for use in the flask
* It should be more effective if it is developed with android studio.

**Future Plans:**

* The process can be automated by showing the prediction result in desktop application.
* The work can be optimized in artificial Environment.

**Report Code Inspection:**

**Feedback meeting**: 25 minutes

**Team meeting summary:**

* We talked about the results of both projects and showed the other team how the code runs and updates.
* Showed them implementation in each module and how all these values get updated.
* We suggested minor changes to the final project to each other.

**Feedback meeting summary:**

* As this is the final project discussion, it went smoothly, and there aren't many suggestions since everything has been rectified in previous modules.
* The algorithm suggested at the previous meeting has now been added, and the working with the algorithm in this meeting has been demonstrated.
* They had previously advised us on the flask framework, which had been completed and implemented. We discussed the updated specification during the meeting.
* They showed how their final project works and how they intend to apply the newly developed idea.

**A brief reflection on what has been accomplished, what went well and could be improved.**

During this phase of the project, we completed the recommendations such as adding an extra module and the code we created. As this is the final phase, everything is implemented based on the suggestions of the other team members and the TA.

**Description on what we accomplished, what we could improve:**

Everything has been verified and corrected in accordance with the previous suggestion. We had previously encountered several issues with the Flask framework, code, accuracy, and several other issues. However, all the issues have been resolved, and the project has gone as planned.

**Contribution Table:**

|  |  |  |
| --- | --- | --- |
| Member | Contribution | Description |
| Srikanth Pavuluri | 14 | Report, code, canvas, git, video |
| Venkat Subba reddy Kattamedi | 14 | Report, code, git |
| Chandana shivannagari | 12 | Report, code |
| Priya Kuppireddy | 12 | Report, code, git |
| Madhuri sri yarramreddy | 12 | Report, code, git |
| Kalyan Kumar Goparaju | 12 | Report, code |
| Mamatha Amireddy | 12 | Report, code |
| Mahesh Reddy | 12 | Report, code |