**Predictive Water Pump Activation for Smart Irrigation System**

**Project Description**

Data related to IoT-based environmental monitoring and control. The data encompasses various sensor readings and actuator states, aimed at automating and optimizing environmental conditions. The goal is to explore patterns, trends, and anomalies within the dataset, providing insights that can enhance the efficiency and effectiveness of environmental control systems. By leveraging this data, the project aims to develop predictive models and decision-making tools that can improve the management and sustainability of environmental resources.

**Scenario 1: Automated Greenhouse Management**

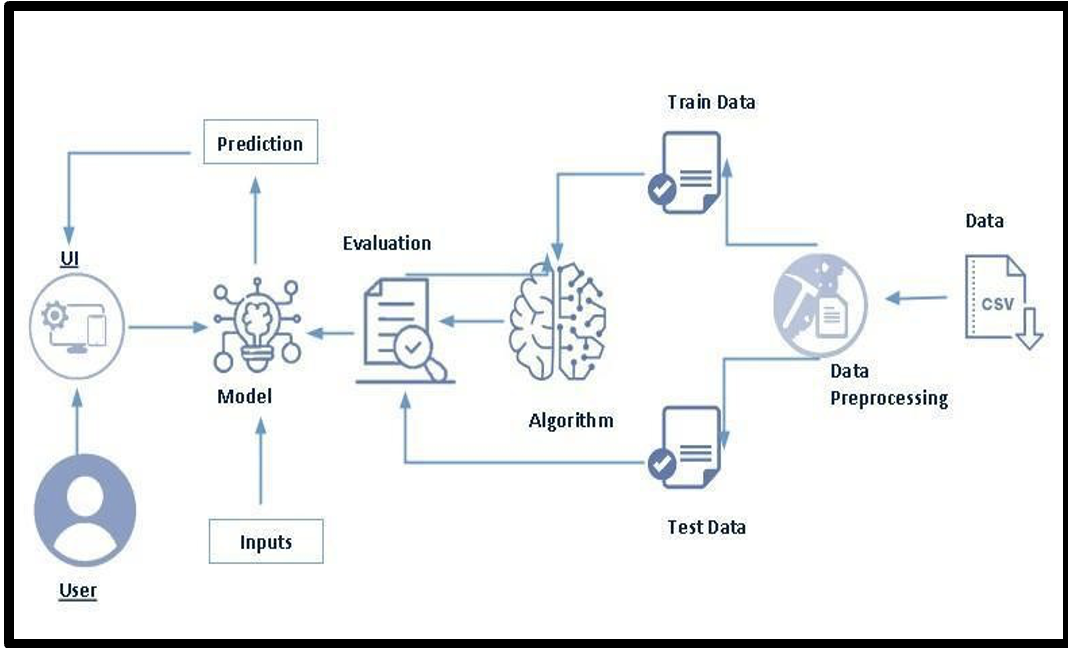
In a controlled greenhouse environment, the IoT sensors continuously monitor various environmental factors. By analysing this data, the system can predict the best times to activate fans, water pumps, and other actuators to maintain optimal growing conditions. This scenario focuses on enhancing crop yield and reducing resource waste by automating responses to real-time environmental changes.

**Scenario 2: Smart Irrigation System**

An agricultural field is equipped with IoT sensors to monitor soil moisture, temperature, and humidity levels. The data collected is used to create a predictive model that determines the precise amount of water needed for different sections of the field. The smart irrigation system can then automatically activate the watering pumps, ensuring efficient water usage and promoting sustainable farming practices.

**Scenario 3: Environmental Anomaly Detection**

In an urban area, IoT devices are deployed to monitor environmental parameters such as air quality, temperature, and humidity. By analysing historical data and real-time sensor readings, the system can detect anomalies or unusual patterns that may indicate potential environmental hazards. This scenario aims to provide early warnings and actionable insights to city officials and residents, helping to mitigate the impact of environmental issues and improve public health and safety.

**Technical Architecture:**

**Project Flow:**

* User interacts with the UI to enter the input.
* Entered input is analysed by the model which is integrated.
* Once model analyses the input the prediction is showcased on the UI

**To accomplish this, we have to complete all the activities listed below:**

* **Data Collection & Preparation** 
  + Collect the dataset
  + Data Preparation
* **Exploratory Data Analysis** 
  + Descriptive statistical
  + Visual Analysis
  + Balancing
* **Model Building** 
  + Training the model in multiple algorithms
  + Testing the model
* **Performance Testing** 
  + Testing model with multiple evaluation metrics
  + Comparing model accuracy before & after applying hyperparameter tuning
* **Model Deployment** 
  + Save the best model
  + Integrate with Web Framework

**Prior Knowledge:**

You must have the prior knowledge of the following topics to complete this project.

* ML Concepts:
* Supervised learning: [https://www.javatpoint.com/supervised-machine-learning](http://www.javatpoint.com/supervised-machine-learning)
* Logistic Regression: https://www.geeksforgeeks.org/understanding-logistic-regression/
* Gradient boosting regressor: https://www.geeksforgeeks.org/ml-gradient-boosting/
* Decision Tree Classifier: https://www.geeksforgeeks.org/decision-tree/
* Flask Basics: [https://www.youtube.com/watch?v=lj4I\_CvBnt0](http://www.youtube.com/watch?v=lj4I_CvBnt0)

**Project Structure**

|  |  |
| --- | --- |
|  | * We are building a flask application which needs HTML pages stored in the Template folder and python script app.py for scripting * RF.joblib is our saved model. Further we will use this model for flask integration. * Training folder contains a model training file. |

**Milestone 1: Data Collection & Preparation**

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

**Activity 1: Collect the Dataset**

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc. In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

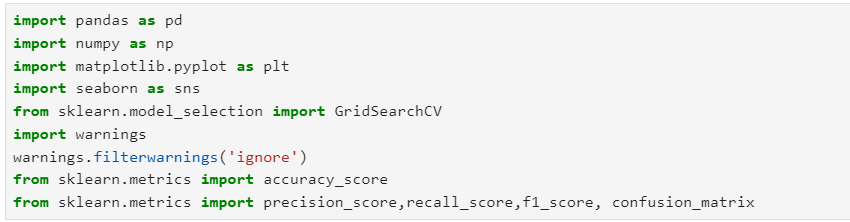
Dataset: [LINK](https://www.kaggle.com/datasets/wisam1985/iot-agriculture-2024/data)

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

**Note:** There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

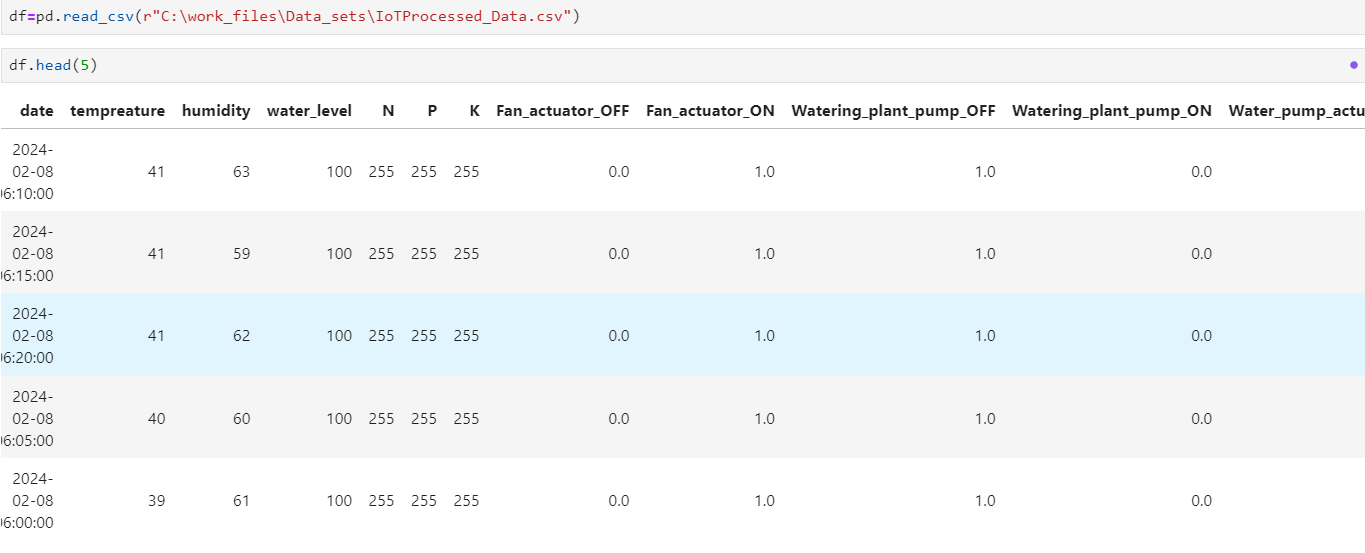
**Activity 1.1: Importing the libraries**

Import the necessary libraries as shown in the image.



## Activity 1.2: Read the Data set

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas. In pandas we have a function called read\_csv() to read the dataset. As a parameter we have to give the directory of the csv file.



**Activity 2: Data Preparation**

As we have understood how the data is, let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much randomness and noise .so we need to clean the dataset properly in order to fetch good results.

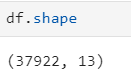
This activity includes the following steps.

* Handling missing values
* Handling categorical data
* Handling Outliers

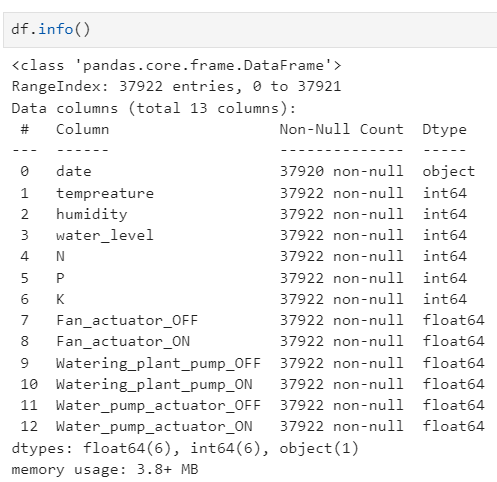
Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

**Activity 2.1: Handling Missing Values**

Let’s find the shape of our dataset first. To find the shape of our data, the df. shape method is used. To find the data type, df.info () function is used.



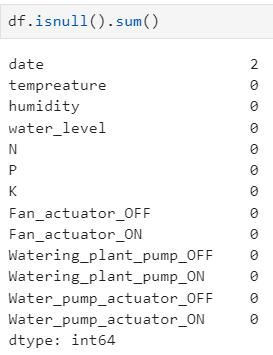
Above Figure Describes the Shape of the Dataset i.e, there are 32537 rows and 15 columns including the Target column as well.



df.info() provides the information about the column’s datatype and provides the count of non-null values in the column is concerned.

Dataset do not have any missing values.

For checking the null values, df.isnull() function is used. To sum those null values, we use .sum() function. From the below image we found that there no null values present in our dataset:



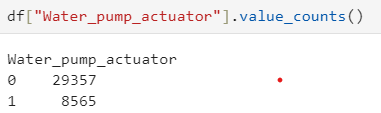
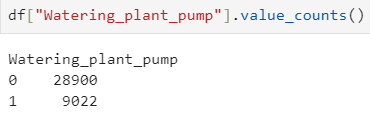
From the above figure we can observe that there are no null values present in the dataset except Date Column.

**Activity 2.2: Handling Categorical Values:**

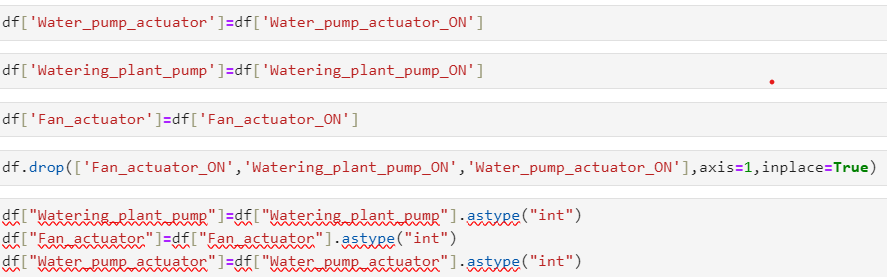
There are multiple categorical columns present in the dataset, they are Work class, Sex, Race, Education, Occupation, Relationship, Martial Status and native country.

As we know they are no missing values/ null values present in the dataset. We need to know their number of categories present in each column with their counts.

There are several operations to find different insights using categorical values some of the functions are **value\_counts**, **replacement of values.**

Considering only one column for Watering\_plant\_pump and its actions as ON and OFF as a single column. In the same way we need to consider single column for Fan\_actuator and Water\_pump\_actuator, simultaniously drop the rest of the columns.related to the changed columns and change the type of modified column into Int from float.

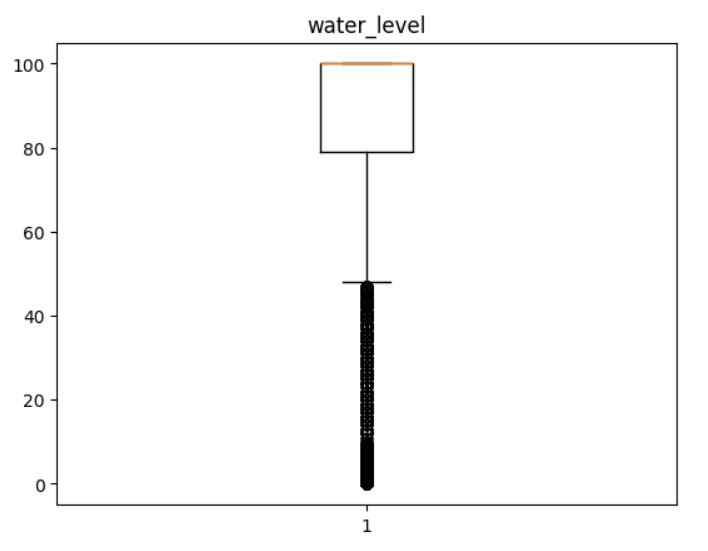
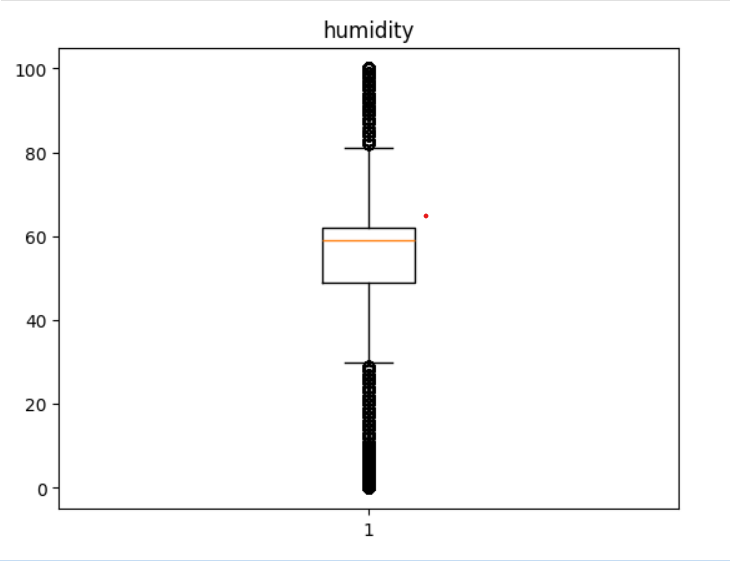


There are no Categorical columns present in the data so there no columns to encode .

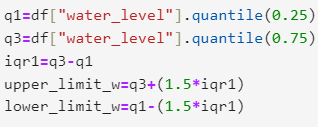
**Activity 2.3: Treating Outliers:**

Outliers are the abnormal data which are away from the range of the distribution of the data of each column in the data. Here we have the box plot to find whether the Outliers present or not

but we cannot know how many Outliers present in the Data



There are outliers present in the data, we need to Handel the outliers for the various columns present in the data. For handling the outliers, we need to find the upper and lower limits. The data beyond these limits are treated outliers. These outliers are treated using Winsorizer and Auxiliary capper method.



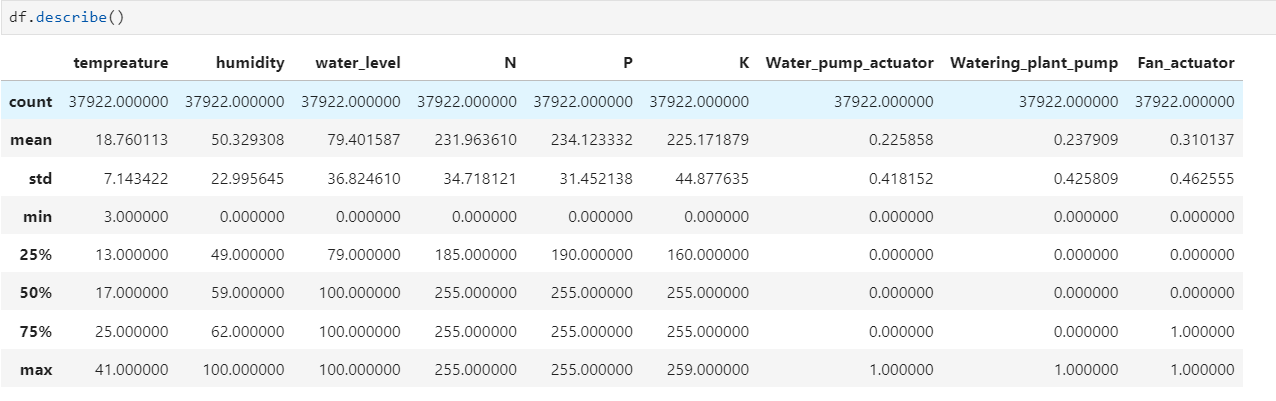
Below figure shows the process of handling outliers using Winsorizer Technique.



**Milestone 2: Exploratory Data Analysis**

**Activity 1: Descriptive Analysis**

Descriptive analysis involves examining fundamental characteristics of data using statistical methods. It provides insights into the mean, standard deviation, minimum, maximum, and percentile values of continuous features.

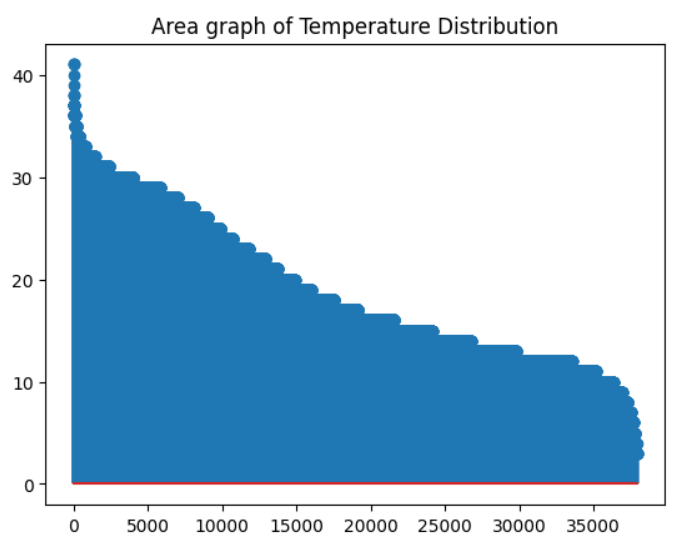
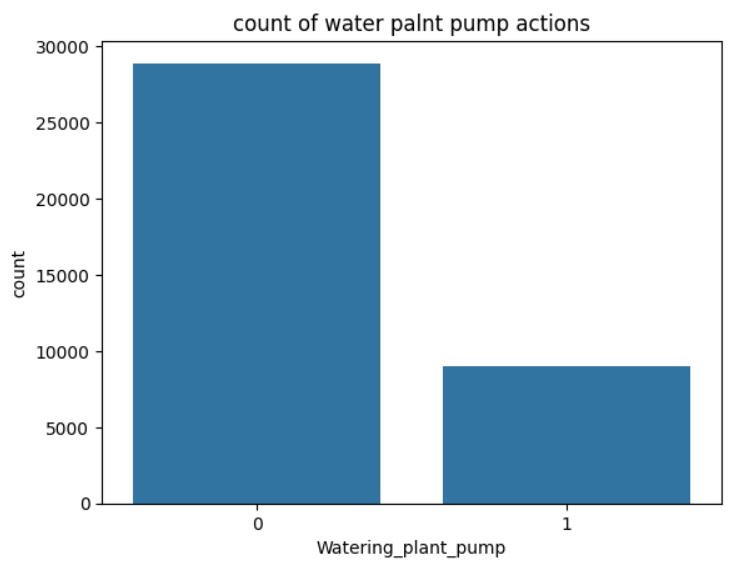
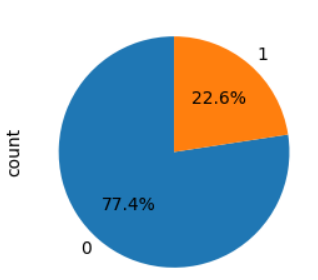


**Activity 2: Visual analysis**

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

**Activity 2.1: univariate analysis**

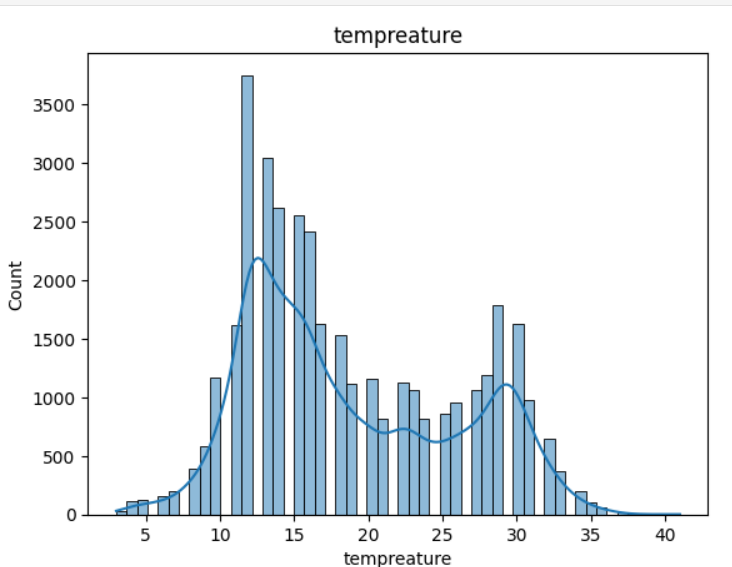
In simple words, univariate analysis is understanding the data with single feature. Here we have displayed a histogram, Pie plot and Horizontal bar plot.

Temperature gradually decreases Percentage of output categories. Water plant pump

From 35 to 15 degrees. Actions (ON & OFF).

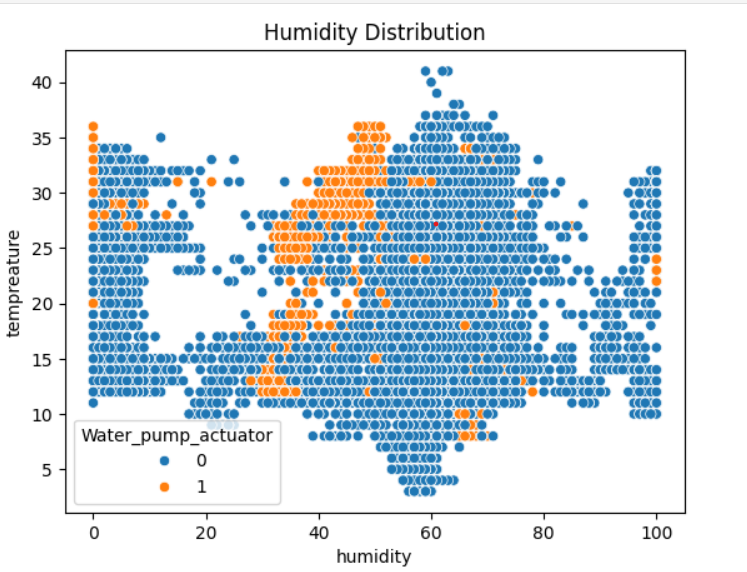
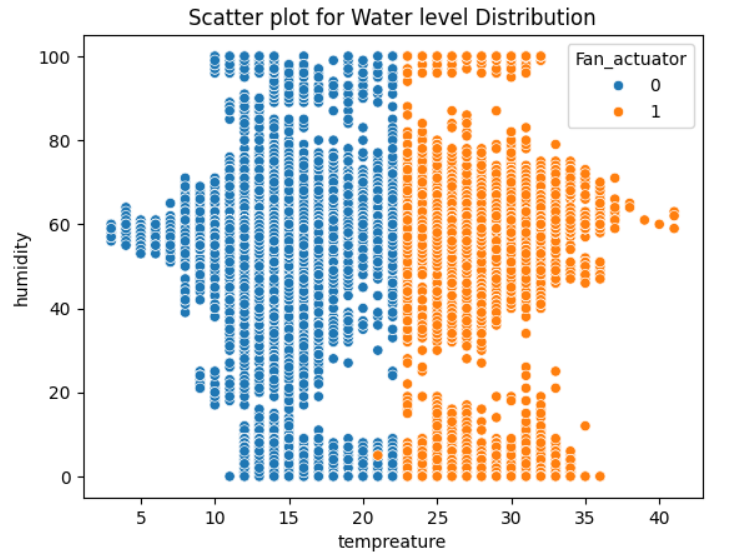
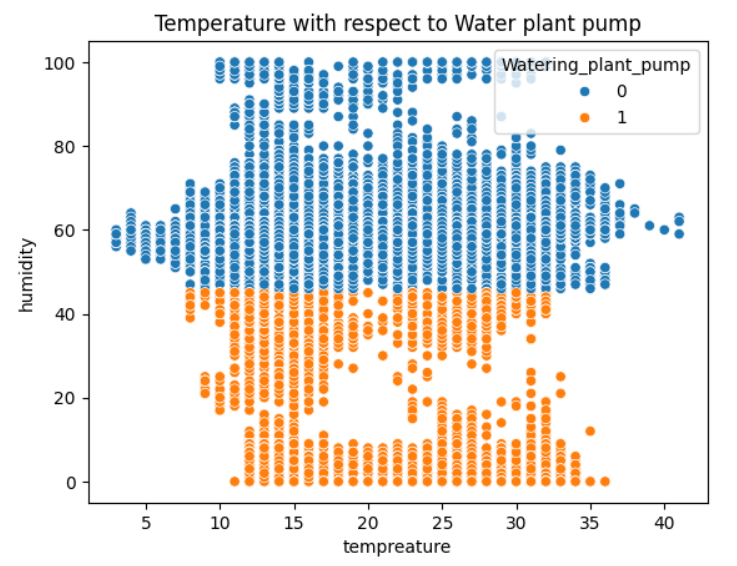
The area plot describes the distribution of Temperature in the dataset and a pie plot describes the percentage of actions by water pump actuator. A Count plot represents the water Plant pump action of on and OFF positions in count.



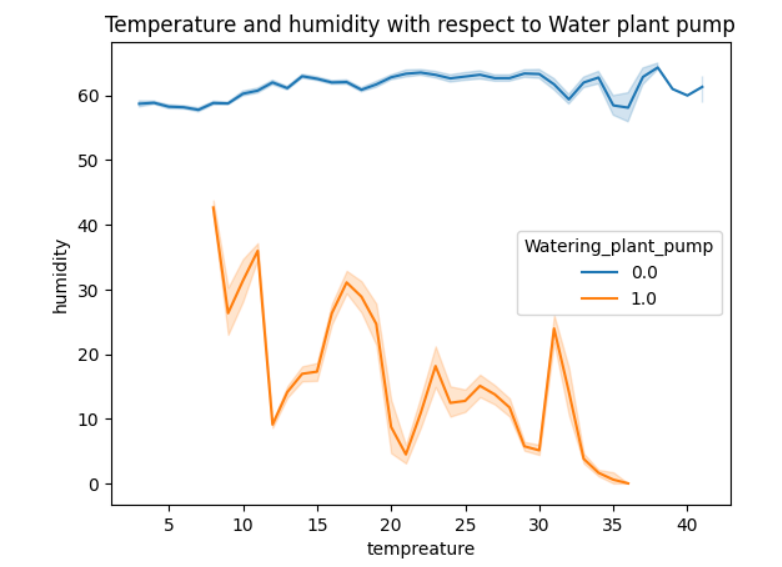
Above Horizontal histogram represents the educational Background of adults.

**Activity 2.2: Bivariate analysis**

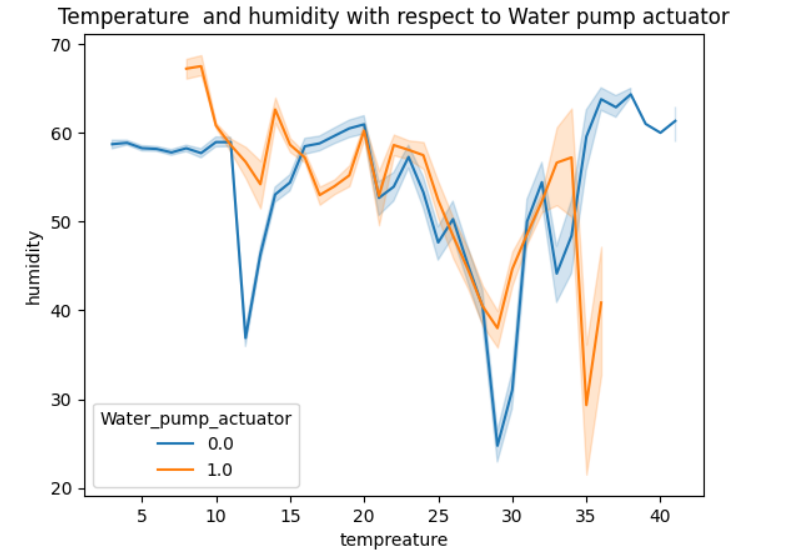
Bivariate analysis is a statistical method that involves the analysis of two variables to determine the empirical relationship between them. Here we have used Scatter plots for finding the insights in each scatter plot.

**** 

Above scatter plots represents the actions of Water pump actuator, Fan actuator and Watering plant pump within different ranges of temperature and Humidity.



Humidity remains constant when Temperature is constant. Humidity constantly decreasing by Increasing in temperature when water Plant pump in ON position.

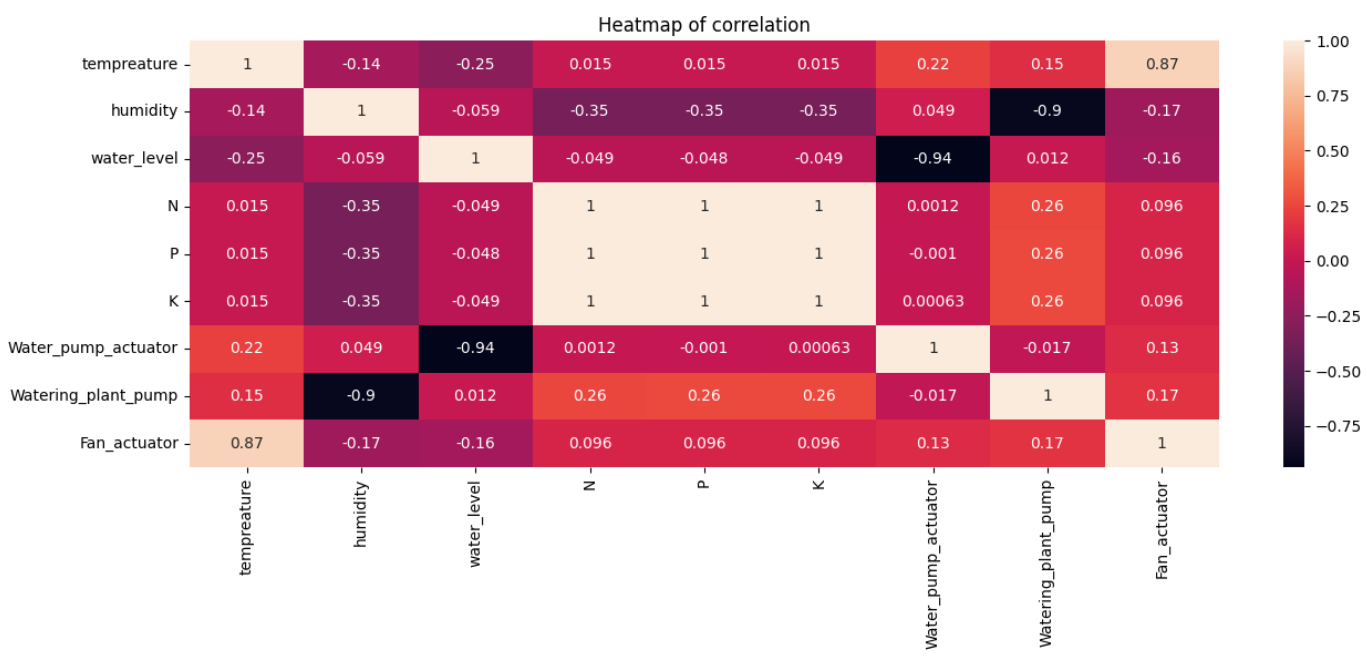


Small Change is observed in humidity along the change in temperature when the water pump actuator is in off position.

Gradual drop of humidity with the change in temperature when the water pump actuator is in on position.

**Activity 2.3: Multi-variate analysis**

Multi-variate analysis is a statistical method that involves the analysis more than 2 variables to determine the empirical relationship among them. Here we have a heatmap representing the correlation among the variables in the Data.

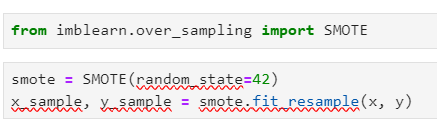
****

Humidity is non correlated to Water\_plamt\_pump

Water\_level in non correlated to Water\_pump\_actuator

**Activity 3: Balancing**

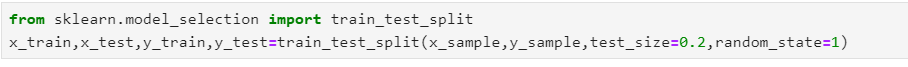
Data is imbalanced since the category for the target category column counts having a major different between the categories so we need to make the data balanced so we use SMOTE technique to balance the data.



**Activity 4: Splitting data into train and test**

Now let’s split the Dataset into train and test sets. First, split the dataset into x and y and then split the data set. “x” represents the whole data columns other than the target column,“y” represents the Target column in the dataset. We need to build the model by giving the training to the model and make the predictions on the test data.so we need to divide the whole dataset into training and testing data.

For splitting training and testing data we are using train\_test\_split () function from sklearn. As parameters, we are passing x, y, test\_size, random\_state.



**Milestone 3: Model Building**

**Activity 1: Training and testing the models using multiple algorithms**

Now our data is cleaned and it’s time to build the model. We can train our data on different algorithms. For this project we are applying classification algorithms. The best model is saved based on its performance.

**Activity 2.1 Logistic Regression:**

Logistic Regression is the base line model where we import the Logistic Regression from the sklearn library and importing the linear\_model and assigning the model to the variable model1 and making the data to fit and moulding the data with respect to the training data.

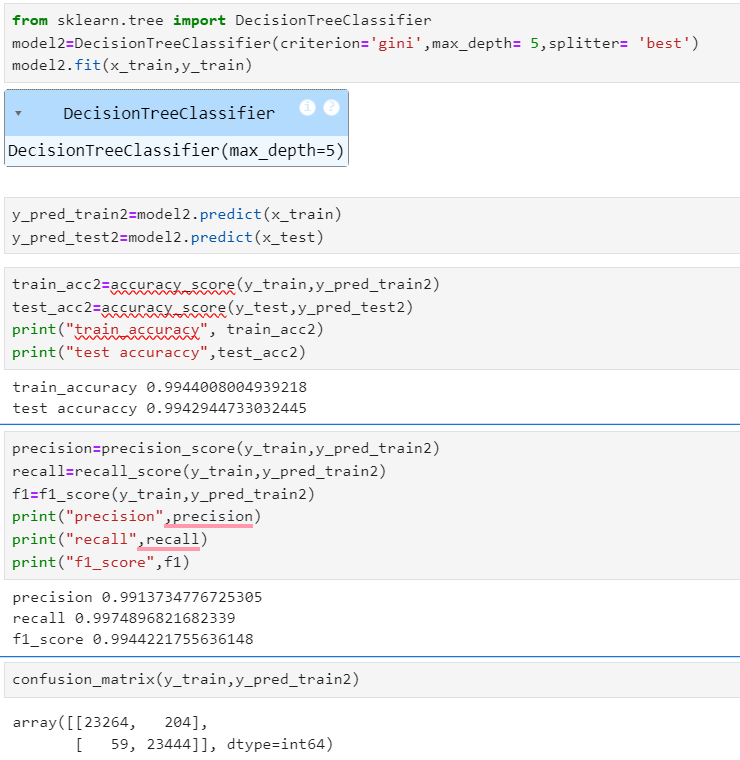


We make the predictions on the train and test data using predict function and assigning the predicted values to the y\_pred\_train and y\_pred\_test

We are calculating the accuracy scores on how the model is working on the data and we use cross-validations to reduce the Bias and trade condition to the dataset. Actually we are able to divide the dataset based on the chosen test size. If CV=4. then we are training the model with 75% of data and we are testing with 25% of data. If CV=5 then we are training the model with 80% of data and testing the model with 20% of data.

**Activity 2.2: Decision Tree Classifier:**

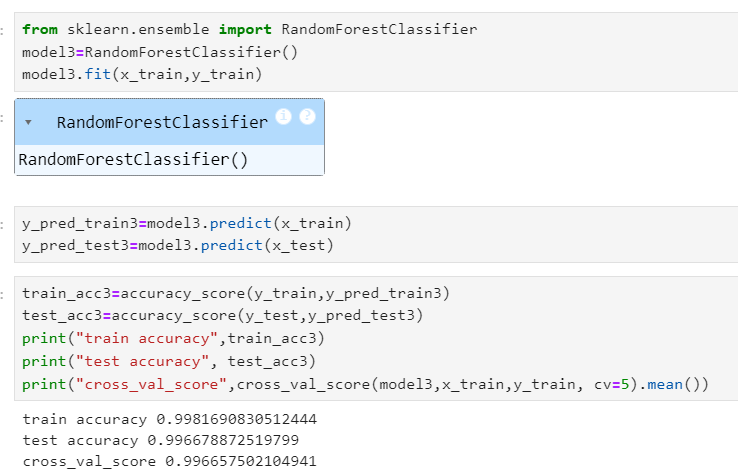
Decision Tree Classifier and regression algorithm is initialized and the training data is passed to the model and assigning the variable as model2 with the .fit() function. Test data is predicted with model2.predict() function and saved in a new variable. For evaluating the model, accuracy is calculated

****

At the end we need to observe the train accuracy and test accuracy with cross-validation score. There is a constrain that the difference of train and test accuracy should not be more than 5%and the difference of test accuracy and cross-validation should not be more than 5% the best model is fixed by this scenario.

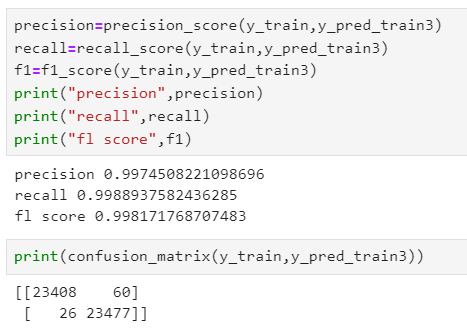
## Activity 2.2 : Random-Forest Classifier

Random Forest algorithm is the classification and regressions algorithm initialized and training data is passed to the model and assigned to the variable as model3 with .fit() function. Test data is predicted with model3.predict() function and saved in a new variable. For evaluating the model accuracy is calculated. For the best obtaining of accuracy. we use hyper parameter tuning to tune the model with the best hyper parameters using the Grid Search CV by choosing the best params we can able to get the best accuracy this method is known as Hyper parameter Tuning.



**Milestone 4: Performance Testing**

Under performance Testing we need to test the model’s accuracy with different testing Metrics like Precision, Recall and F1\_score. Below are the performance Metrics of final fixed model



# **Milestone 5: Model Deployment**

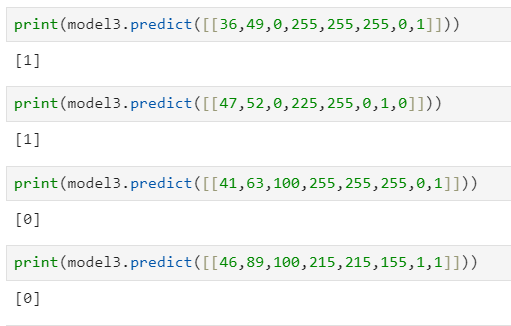
# **Activity 1: Save and load the best model**

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

****

## Activity 2 : Test the model

Let’s test the model first in python notebook itself. As we have 7 features in this model, let’s check the output by giving all the inputs.



The predicted values result same as actual values.

**Activity 3: Integrate with Web Framework**

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

* Building HTML Pages
* Building server-side script
* Run the web application

**Activity 3.1: Building Html Pages:**

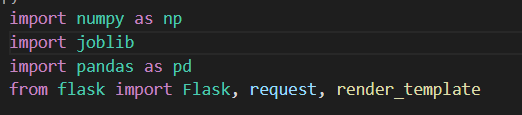
For this project create two HTML files namely

* index.html
* predict.html
* result.html

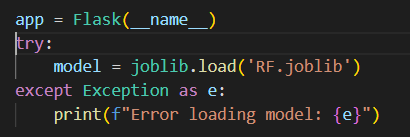
and save them in the templates folder.

**Activity 3.2: Build Python code:**

Import the libraries

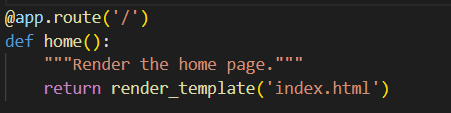


Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (\_\_name\_\_) as argument.



We render index.html for the displaying the web application , similarly we render the predict.html for the user input values of the forms to predict the income. Simultaneously we render result .html to display the result of the prediction value.

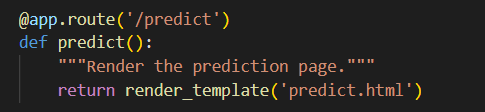
Render Index.html:



Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, ‘/’ URL is bound with the index.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered.

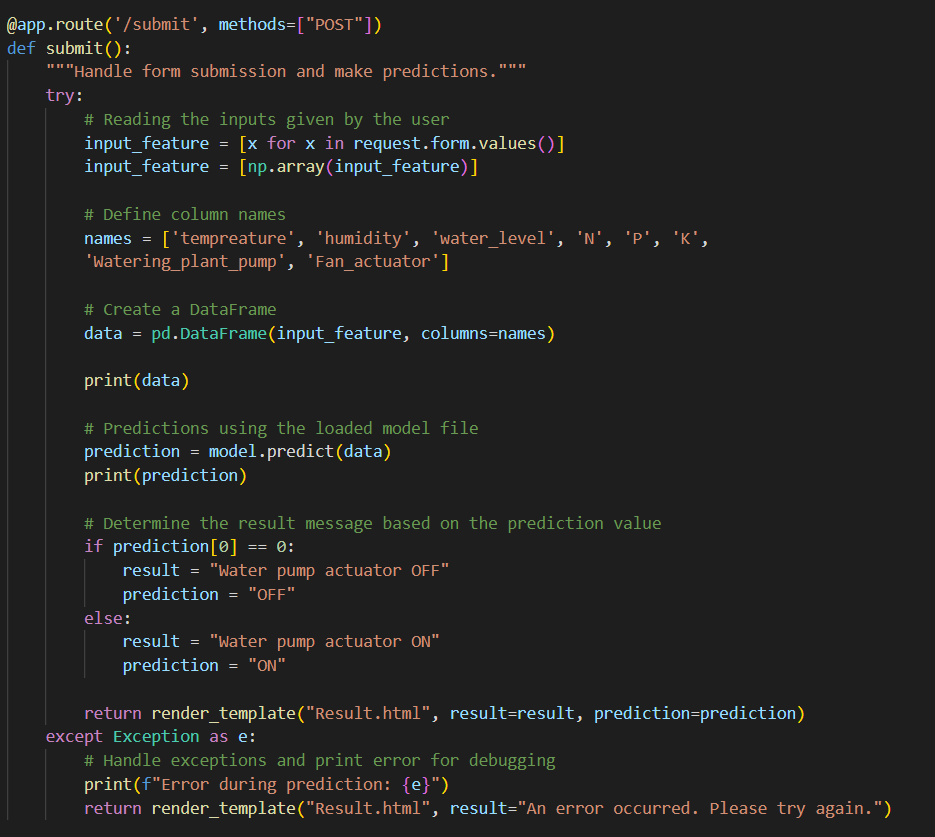
Render Predict.html:



In the predict.html where we provide the user inputs in the form for the prediction of income

Whenever you enter the values from the html page the values can be retrieved using POST and GET Methods.

Retrieving the value from UI:



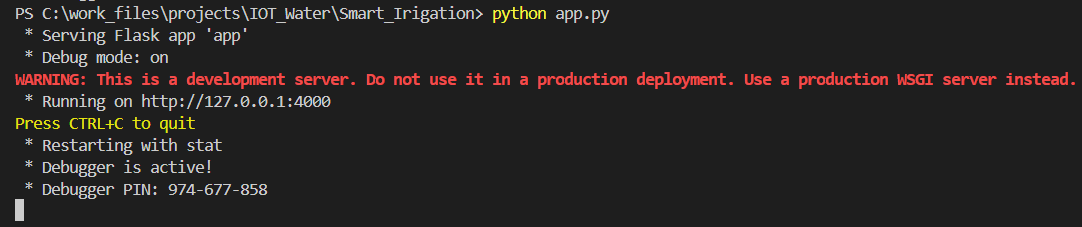
Here we are routing our app to conditional statement. This will retrieve all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

Main Function:



**Activity 3.3: Run the web application**

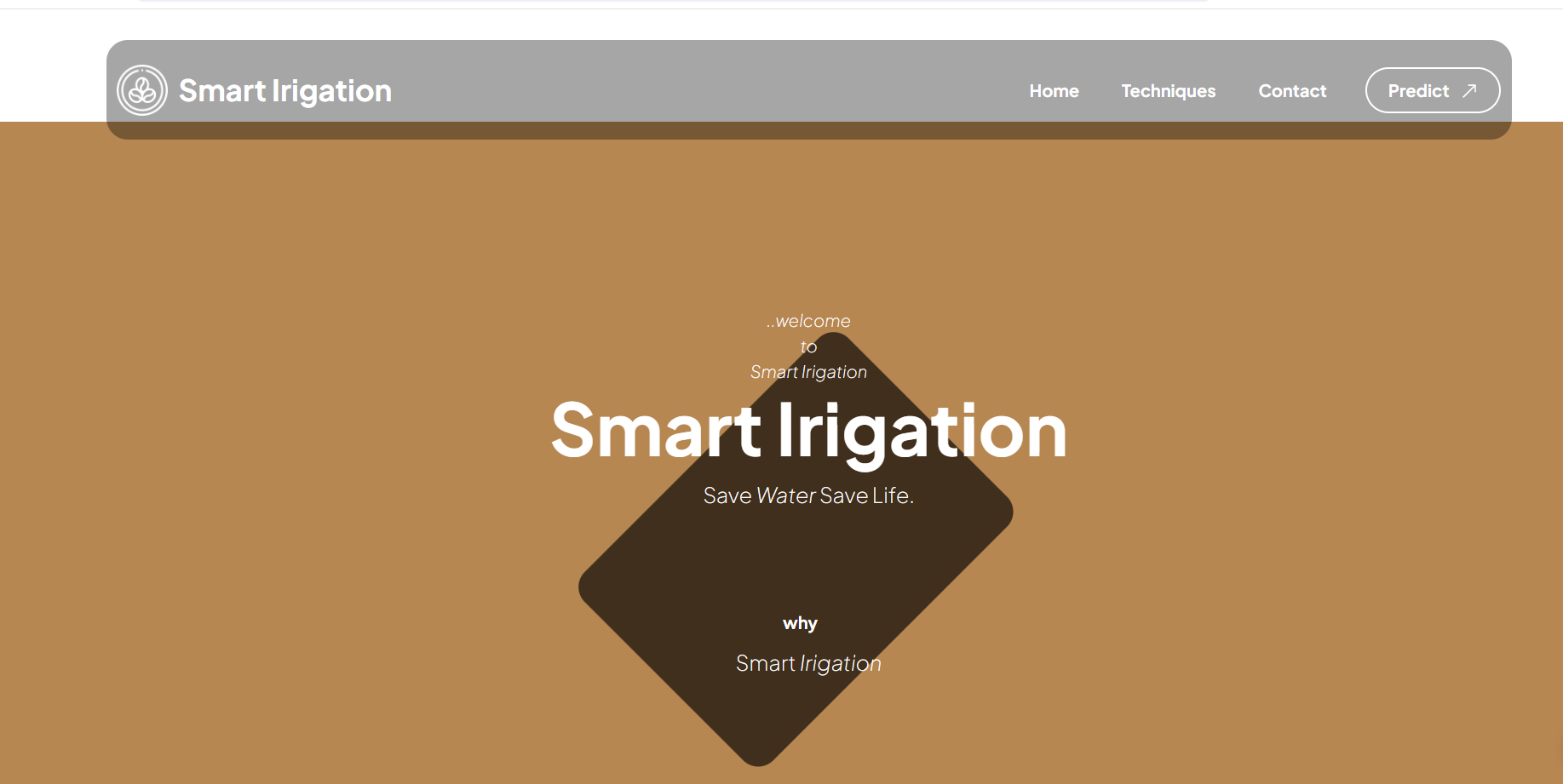
* Open vs code application in the search menu.
* Navigate to the folder where your flask folder of your files exist.
* Click on the view button in the vs code nav bar and click on the terminal option in the dropdown menu.
* Now type “app.py” command
* You will have a link displayed in the terminal as “<http://127.0.0.1:4000> “.
* Double click on the link then you will be navigated to the web application.
* Click on the predict button in the nav bar, enter the inputs, click on the predict button, and see the result/prediction in the result.html.



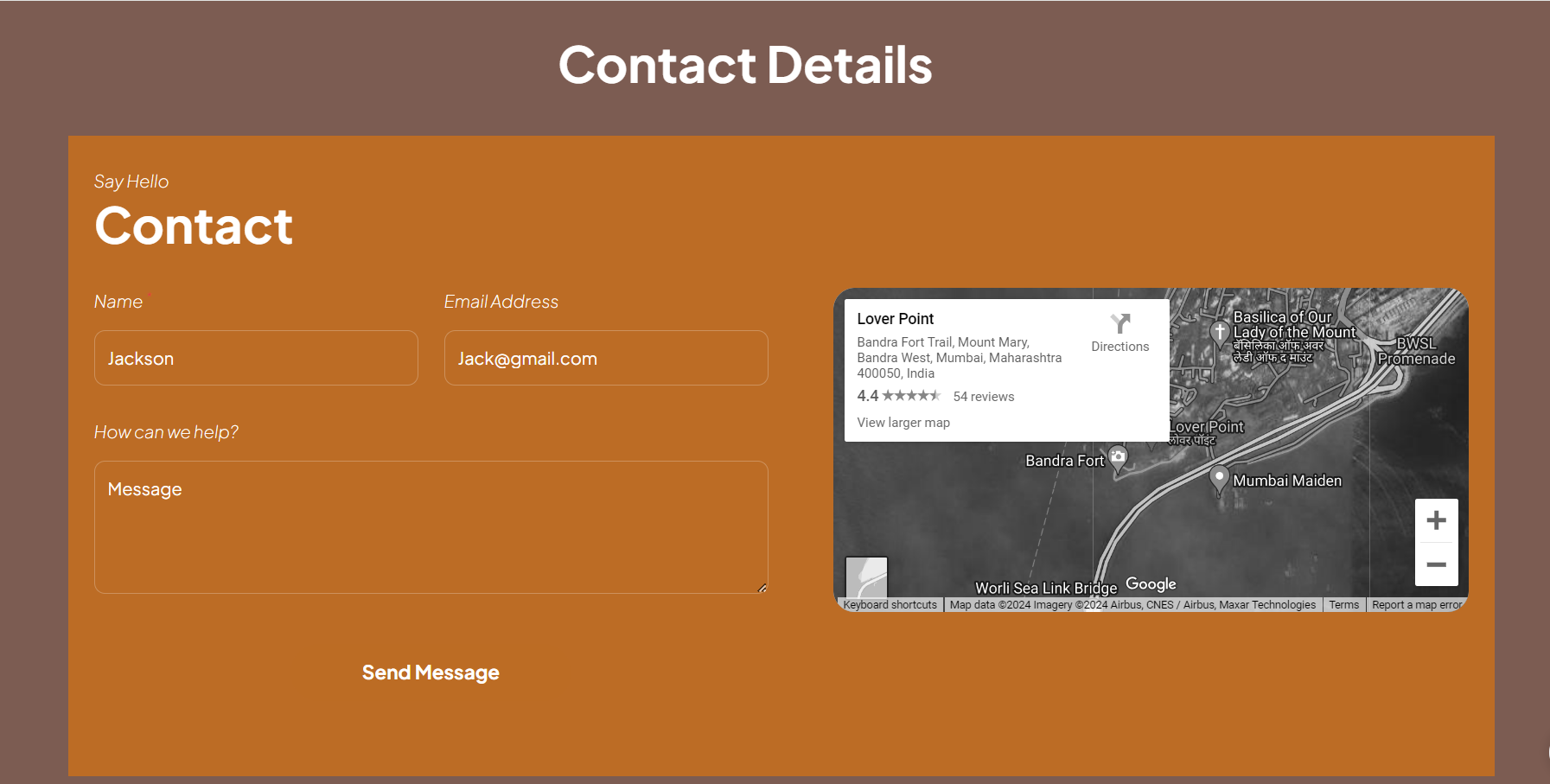
Now, Go the web browser and write the localhost URL (http://127.0.0.1:4000) to get the below results

Results:

1. Index page (Index.html)

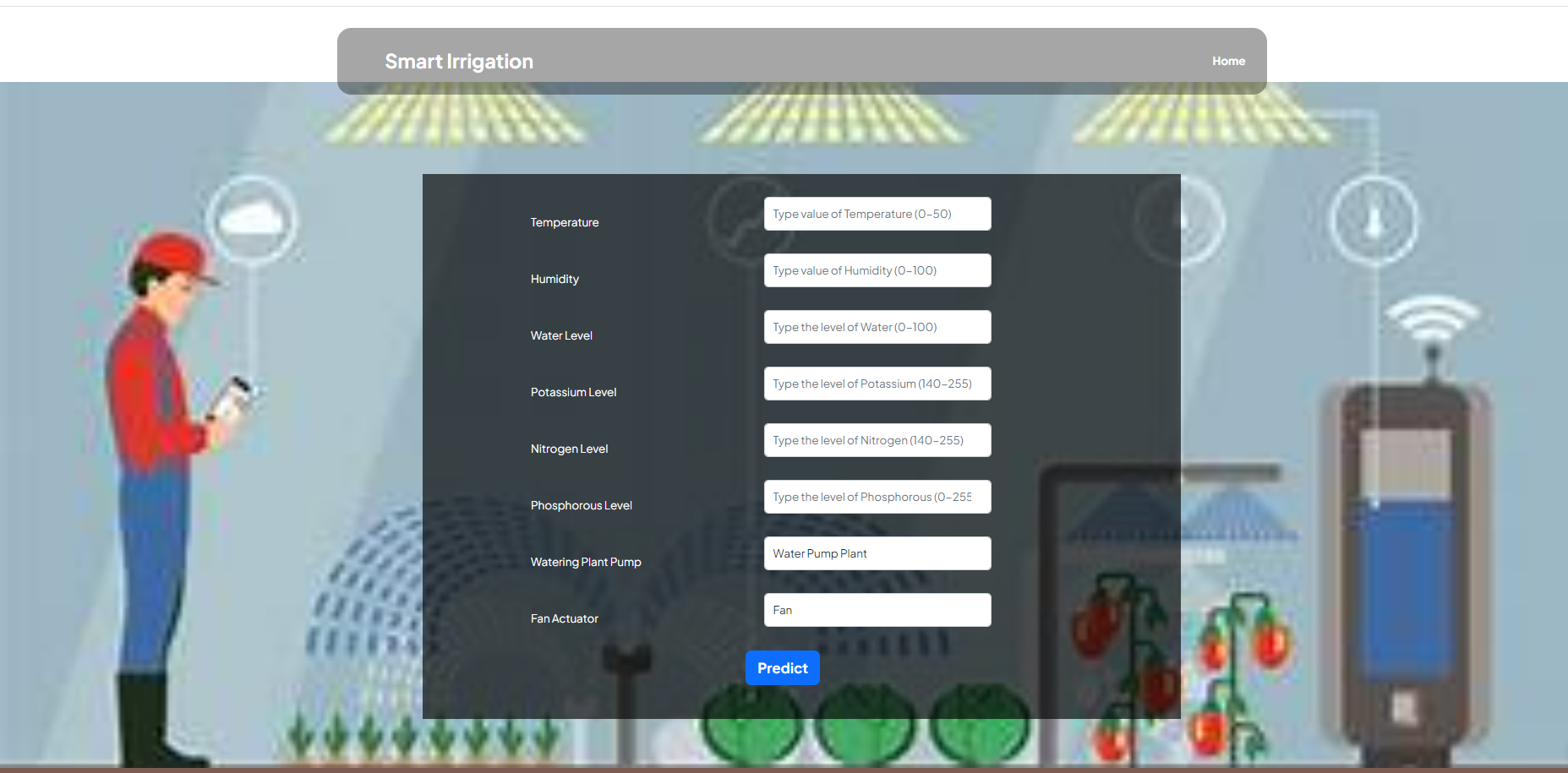






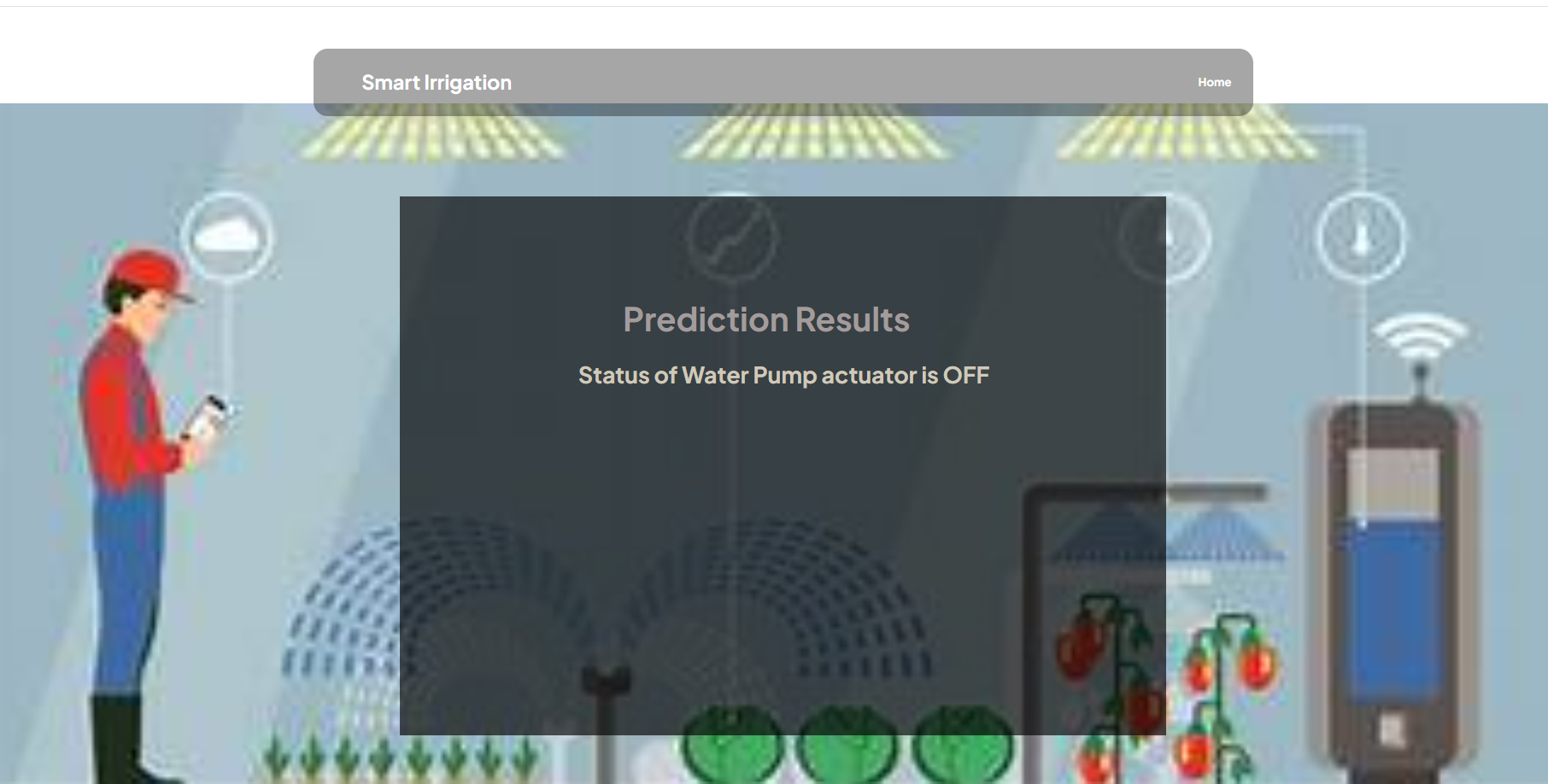
If you click on the predict on the nav bar we can able to navigate to predict.html below figure represents the UI of predict.html page

b) Prediction page (Predict.html)



By providing the inputs by the user and click on the predict button you will be navigating to the result.html it displays the result. Based on the result you can navigate to the home and click on the investments in the nav bar of index.html then you can have an info about different mode of investments.

Result.html:

****