Machine Learning Model Development for Optimal Irrigation Timings

Machine learning aims to answer business questions, identify and analyze trends, and assist in problem solving by identifying patterns in your data and then making predictions based on these frequently complicated results. Our research question was to predict if there is need for irrigation or not. These predictions were to provide instruction to the pump on the embedded system.

Due to the limited time, with in which we had to complete this project, the dataset we used to train the model was provided with the project instructions. The custom dataset we used was called Irrigation.csv and it contained information to help us in our project, having six features that is; Crop Type, Crop Days, Soil Moisture, temperature, Humidity and Irrigation with a total of 501 observations. Crop Type was the only categorical feature in our data with nine unique items. The data was all clean with no missing values. The Crop Days were measured in the number of days, the temperature in degrees, the Humidity in percentage, and the Soil Moisture in resistance of the soil, which is inversely proportional to the amount of water in the soil.

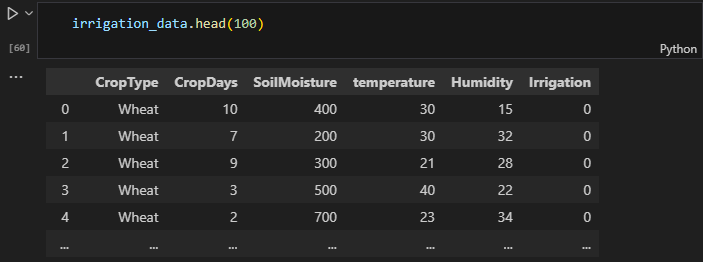


Figure 1: How the first columns looked like before editing anything.

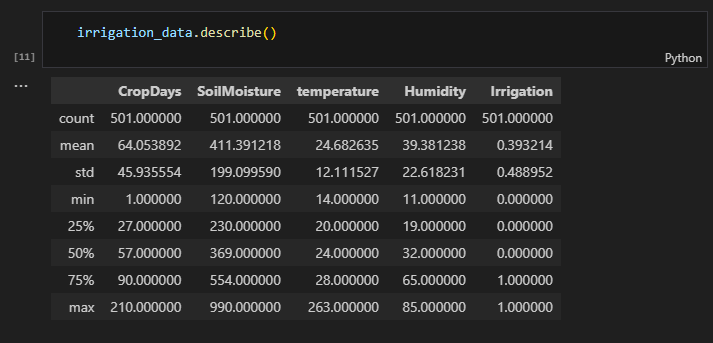


Figure 2: The mathematical description of our dataset.

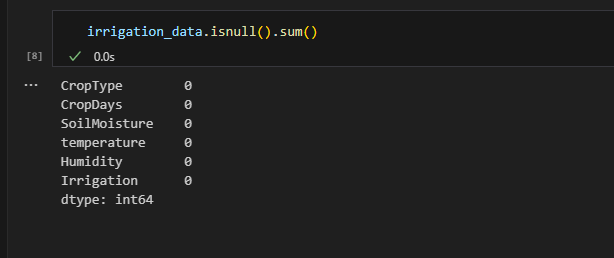


Figure 3: Checking for missing values. There was none.

**Exploratory Data Analysis**

Exploratory Data Analysis was aimed at helping us better understand our data, by relieving the limitations in the data, what features are most important and how to best answer our research question. During our exploratory data analysis, we had to draw count plots for each feature, the scatter plot, cat plots and, for the different features against our target, Irrigation. These visualizations helped us to get meaning full insights and conclusions. We therefore dropped the humidity feature since it did not have a significant effect on our target. The temperature feature also had one outlier that we removed using the drop function. We also identified and dropped the outlier in the temperature column as this would have a negative effect on the overall performance of our model by causing a bias.

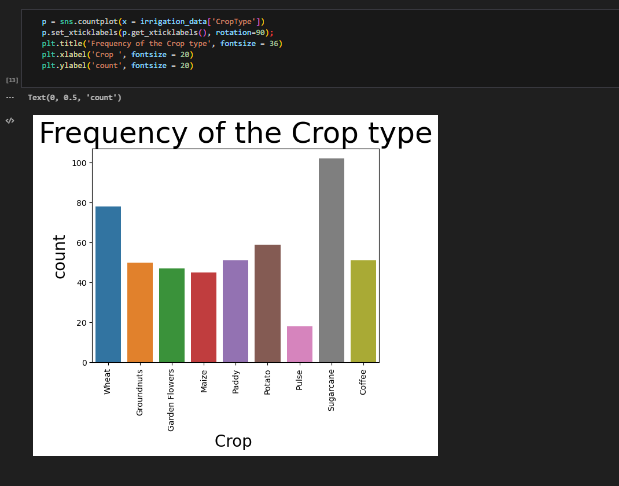


Figure 4: Visualizing the distribution of our CropType feature.

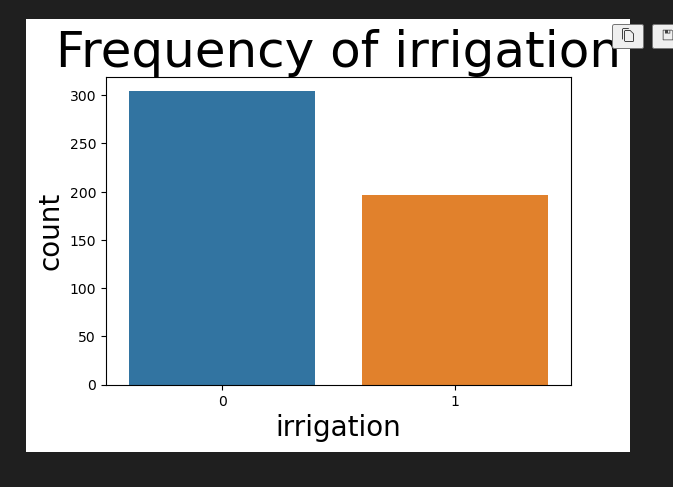
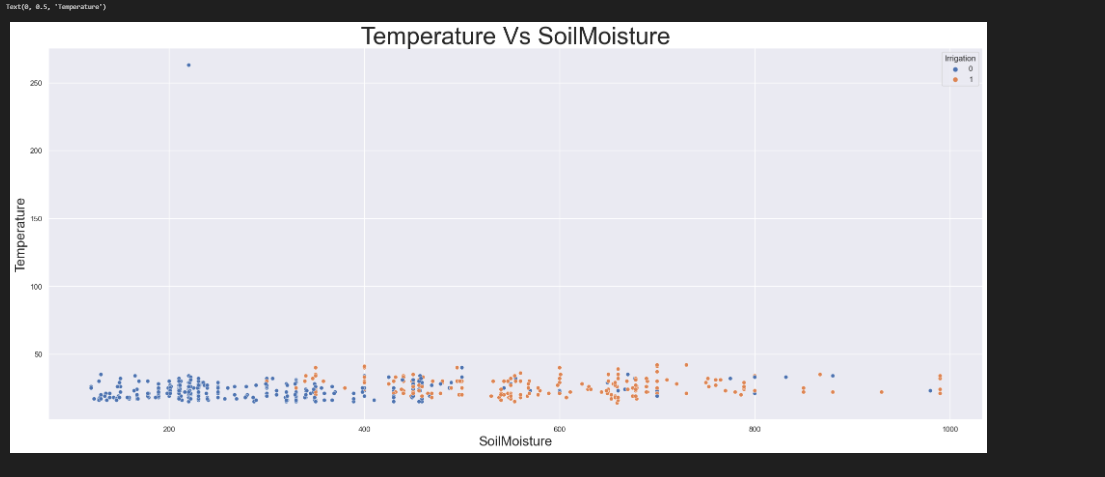


Figure 5: Visualizing the distribution of our Target feature.



**OUTLIER**

Figure 6: Visualization of the temperature column with an outlier, that we later dropped.

* Include details about the types of data collected, such as sensor readings, time stamps, and crop characteristics.
* Quantify the size of the dataset (number of samples) and the number of features.
* Highlight any challenges or anomalies in the data that need to be addressed during preprocessing.

During our preprocessing, we used label encode to convert the Crop Type feature into integer values that the model can work with. We also used the min max scaler to scale our data to a range of 0 to 1 where the smallest value in our data was assigned a 0 and the largest, a 1. This was done to bring our values as close to 0 and 1 as possible as the computer understands these two binary values most.

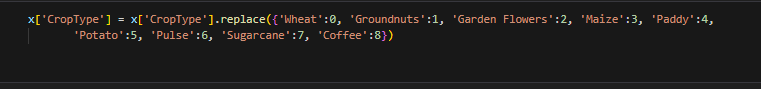


Figure 7: Label encoding to convert the categorical CropType feature into numerical values

From this, we separated our target, irrigation from the other features using the drop function. We went ahead to split our data into the training and validation data using the train\_test\_split function and started training our models and testing for the accuracy by using the confusion matrix, classification report, accurancy\_score function.

We then trained using different machine learning algorithms to determine which one worked best for our case. And since we were handling a Classification problem, we had to only choose classification models. We tried out we the Random Forest Classifier, the XG Boost model, Logistic regression, and the SMV Classifier model.

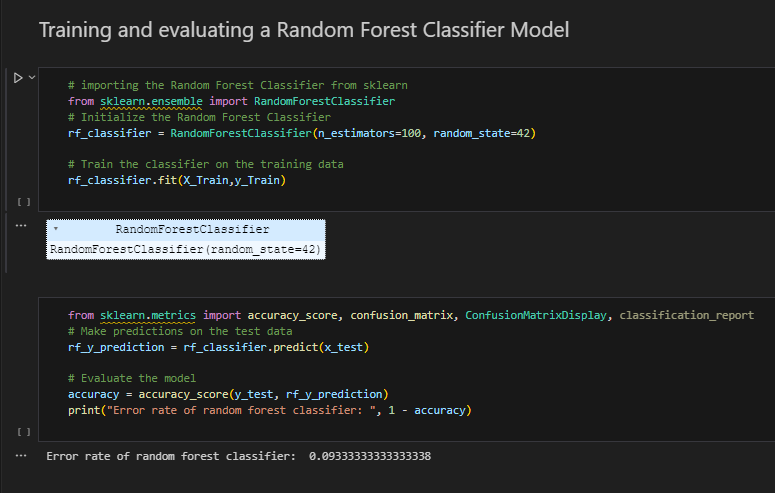


Figure 8:Code snippet for training and testing the random classifier model.

To determine how good each model was, we used a confusion matrix and a classification report. A confusion matrix is a table that presents the performance of a classification algorithm on a dataset. It is very helpful when dealing with multiple classes to classify. The matrix compares the predicted classes against the actual classes and breakdown the results into four categories, that is true positives, true negatives, false positives and false positives. True positives are instances that are correctly predicted as the positive class, true negatives are instances that are correctly predicated as the negative class, false positives are instances that are incorrectly predicted as the positive class, also known as Type I errors, and False negatives are the instances that are incorrectly predicted as the negative class, also known as Type II errors. This helped us understand the errors in our model and determine where its weakness and strengths lie.

We also worked with classification reports. A classification report is a summary of the various performance metrics derived from the confusion matrix and it provided detailed evaluation of how our classification performed across multiple classes. It was made up Precision, recall, F-1 score, suppor, accuracy, macro average and weighted average. Precision is the ratio of correctly predicted positive instances to the total predicted positive instances, and it measures the model’s ability to avoid false positives.

Recall/ Sensitivity/ Positive Rate, is the ratio of true positives to the total actual positive instances. It measures the model’s ability to correctly identify positive instances.

F-1 score is the harmonic mean of precision and recall. It provides a balance between the two metrics.

Support on the other hand, is the number of occurrences of each class in the dataset. Accuracy is the ratio of correctly predicted instances to the total instances. Macro Average is the average of precision, recall, and F-1 score across all the classes and it treats all the classes equally. Finally, weighted average of precision, recall, and F-1 score weighted by the support of each class and it considers class imbalance.

Both the confusion matrix and the classification report are crucial tools we used to assess the performance of each classification model and made our final decision to use the Random Forest Classifier depending on its strength and ability to reduce the number of false negatives as compared to all the other models.

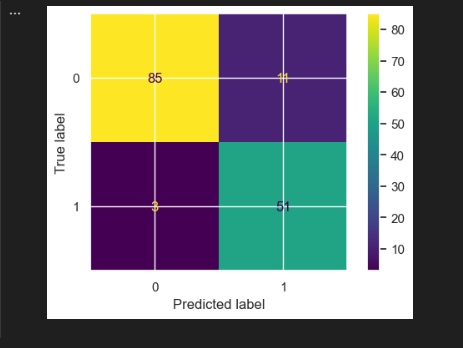


Figure 9:Confusion matrix for the Random Forest Classifier model.

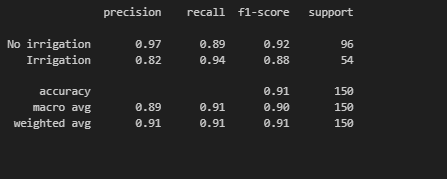


Figure 10:Classification report for the Random Classifier model.

We realized, that the Random forest classifier from Scikit learn library was not compatible with the Arduino IDE, we instead used the same model but from the Everywhereml third party library. This works like a wrapper around the Scikit learn library.

We retrained the model and first save it as a joblib file and deployed it on a website to test if it worked. Upon working as expected, we converted it into a C++ file by running this command:

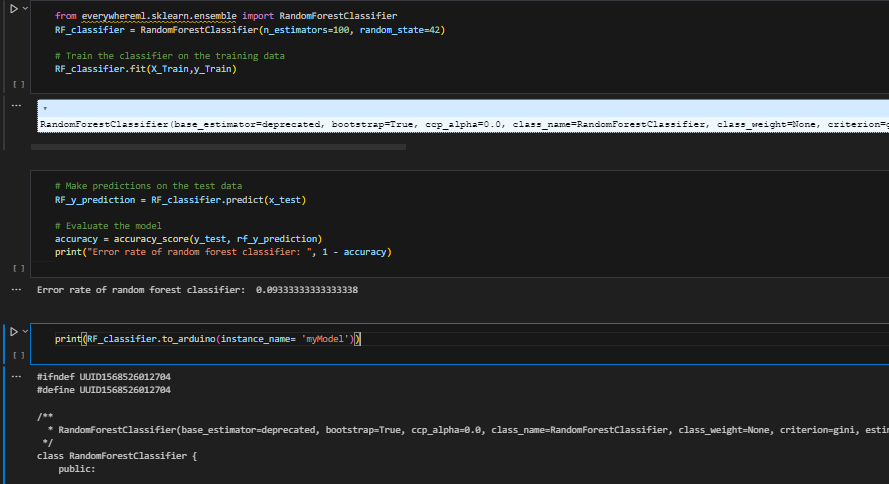


Figure 11: Importing the Random Forest Classifier from everywhereml, training it and converting it into a C++ file.

We copied the output C++ code into a .h file, that we included in the same folder as our Arduino sketch, also added it to the Arduino libraries and included it in our sketch (#include mymodel.h). We created an instance of our model and called it on an array of the features that we recorded as the sensor readings in the same order as that we had used during the training for on device predictions. This was a success as the model predicted as expected.

Since our model was giving numerical output, 0 and 1, we wrote an if statement to convert it into strings that were easy for a farmer to understand. That is Irrigation for a 1 and No Irrigation for a 0.