SOIL MOISTURE PREDICTION USING MACHINE LEARNING ALGORITHMS

ABSTRACT

The amount of water in the soil is one of the most important factors affecting crop growth. The amount of water that is available to plants is determined by the soil moisture. The ability to predict soil moisture is a critical component of many agricultural and environmental processes, such as crop and forest growth, soil erosion, and flooding. Soil moisture is one of the most important soil quality factors for plant growth and development. This article describes how a process known as soil moisture prediction is used to understand how soil moisture will be distributed in the future. This article also describes how soil moisture is measured, and how that information is used to predict soil moisture in the future. Three datasets gives the best idea about the individual data attributed and Metrics which primary keys to the dataset execution and to explore the use of machine learning algorithms to improve the prediction accuracy of image analytics on the use of machine learning algorithms. For few Machine learning even with train and train of the dataset there won't accuracy values. In this step, I decided to model the five different algorithms which are XGBOOST has prediction values of 95%, LINEAR REGRESSION has prediction values of 50%, RANDOM FOREST has prediction values of 98%, K-NEIGHBOUR has prediction values of 90 %, DECISION TREE has prediction values of 81 % and After the prediction of each algorithm, each algorithms gives the accuracy values and prediction values. Futhermore into prediciton is the hyper parameter tuning for the best two algorithms will take place and that will gives the best accuracy for the dataset. Hyperparameter Tuning of XGBoost Classifier using GridSearchCV and Hyperparameter Tuning of Random Forest Regressor using RandomisedSearchCV are highest predicted values of 95% and 94 % accuracy values . so Random forest and Extreme gradient boosting has best accuracy value in this dataset.

CONTENTS

LIST OF FIGURES

LIST OF ABBREVATION:

1. XG BOOST- EXTREME GRADIENT BOOSTING
2. KNN- K- NEAREAST NEIGHBOR

LIST OF APPENDICES

Chapter-1

Introduction

1.1 INTRODUCTION

This Introduction area aims to provide the complete overview of this project, the modules in it, the project's goals, challenges and objectives. Furthermore, the entire details of the report's structure are provided at the end.

1.2 PROJECT OVERVIEW

1.2.1 SOIL MOISTURE PREDICTION

The project aims to build and to understand the soil moisture prediction where the attributes containing the soil moisture data in different layers .These different layers indirectly determines the soil layers present in the soil where we see different moisture data contains which helps to understand about soil types and different types of crops and moisture changing in terms of season and how the crop yields.

1.2.2 TWO SECONDARY DATASETS

In This project, we uses two secondary datasets which helps to give clear idea about the datasets and attributes with components in the dataset shown . Moreover the secondary dataset talks about the soil and flowers in the dataset continuing with Sensors have 6 +/- 1 cm vertical distance from each other in four different depths.

Two secondary dataset is one of the important components for the project. In the soil moisture dataset, it often requires the initial evaluation of specific regions and atmosphere and collection of the dataset is difficult

1.2.3 PRIMARY DATASET

In This project, we use one primary datasets which presents the combined form of two secondary dataset . After understanding the secondary datasets, we can understand the intent regarding the soil and flowers in the dataset continuing with Sensors have 6 +/- 1 cm vertical distance from each other in four different depths. The primary dataset gives the clear cut understanding of the data preprocessing though the process may be longer and running timing will be more extend .Primary dataset will the give clear understanding of the attributes and what the project about.

1.3 PROJECT GOALS

The project goals should be able to achieve the soil moisture prediction which further helps to crop production and crop price estimation based on the yield development. The agriculture is always one growing fields which the dataset grows larger and larger. The problem in the larger dataset is training the dataset always take enormous time and labor. In the dataset is very unique because of the larger algorithm applied to the dataset which gave the best accuracy.

- Machine learning algorithms: A immense task to initiate this project was rather lot of self learning. To begin with , the understanding how the dataset works and what is algorithms? how to apply these algorithms ? are basic question but when comes to implement it rather a big task. The Understanding of the machine learning is very important and necessary to achieve the project .

- Soil moisture Prediction : The workflow ought to identify the attributes present in the dataset is tedious job.The attributes helps the important in any dataset. Understanding the attributes is primarly to explore the dataset and get a grasp to execute the commands.

- Display Results : The workflow is to give the values as conclusion about the soil

moisture prediction and results known.

1.4 PROJECT CHALLENGES

This section aims to provide a summary on the various challenges faced or encountered during the completion of this project which are expressed in detail in each subsequent chapter's

1.4.1 DATA MANAGEMENT AND PREPROCESSING

The project's most difficult is to find the dataset that has to have big dataset and as well as the dataset should be apply the novelty to make sure the project succeed. Two secondary dataset is already large dataset where one as the plant and other about the soil. Soil moisture dataset taken from Kaggle. Using cloud frameworks like google colab will have computational workload. Accessing the important attributes is the major part and complex task. Before any data preprocessing the dataset make sure ,only to discard the repeated and unwanted data.

1.4.2 MACHINE LEANING CONCEPTS

Machine Learning concepts have a way to solve the problem with the right approach. This type of methodology is frequently used in machine learning concepts. A big difference between humans and computers has long been that humans tend to automatically improve the way they approach a problem. People learn from past mistakes and try to solve them by correcting them or finding new approaches to tackle the problem. Conventional computer programs do not analyze the result of their tasks and therefore cannot improve their behavior. The field of machine learning addresses exactly this problem and involves creating computer programs that can learn and thus improve their performance by gathering more data and experience.

1.4.3 MODEL TRAINING

Model training in machine learning requires a lot of training time before it is used for testing. It takes much longer to run tests and get results compared to normal computerized calculations. This issue has the potential to halt the

progress of the task. Carefully checking the analysis in each stage may require but the process won't be twice if anything goes in the initial stages

1.5 PROBLEM STATEMENT

Soil moisture dataset is very unique and understanding the dataset is very difficult. Many soil moisture prediction has been done on specific region or country to understand there soil moisture . In this project, the aim is to predict and optimize any dataset that any sensor values that have collected ,later convert into dataset those dataset can be used to predict the soil moisture using the approach. I have done to make the prediction any dataset using these approach can be quite useful.

1.6 REPORT STRUCTURE

The research chapter dicusses the Study on the Application Monitoring Moisture Content and Estimation on Soil moisture using machine learning algorithms. Future of the agriculture will depends on the sensors values to predict .These predict can in those prospect.

The Machine Learning chapter deals with the concepts and theoretical understanding of model training and learning. This chapter includes research papers related to the project which helped to understand the deep aspects of the machine learning algorithms. If we understand the issues face by other research it will helps to overcome those problems quicker .

The Project's work flow is discussed on the procedure and techniques chapter. It narrate the processes that were done, and the techniques that were operated to perform the tasks. The architecture of each algorithm and how they were used in the project.

The Design section shows how the techniques were used , it quicker way to make understand about the project what has been done and how it is has been done with modules shows in the design way to make to understand the outline of the project.

The Implementation chapter outlines the step-by-step workings with code snippets, highlighting how the entire project system is assembled together.

The Results chapter explains the obtained results of the projects with output snapshots and details on the inferences, accuracy and challenges encountered.

The conclusion chapter outlines and brings the entire summary of the project, its benefits, challenges and the future scope of the project.

1.7 CONCLUSION

In this chapter, the complete overview of the project, a jest of the modules in it, the goals and the challenges were outlined in detail.

The Project combines many statistical solutions from dataset to machine leaning algorithms and Hyper parameter optimization and this combination to provide the understanding of the soil moisture prediction.

The next chapter deals with the research work and background study on the recent and existing works on the Soil moisture dataset.

CHAPTER-2

Literature Survey

2.1 INTRODUCTION

This Introduction aims to provide the information undertaken related to the project. The chapter includes research from various sources about various topics required to understand the project. The chapter provides knowledge on the current working on Soil moisture and also details out the technologies used and the requirements required for the project

2.2 RESEARCH WORK

The section provides the various existing techniques and studies employed and are being used in the development of Soil moisture prediction and also highlights the different subdivisions of the soil based characteristics like yield production, region research ,disease detection etc..,

2.2.1 DESCRIPTION

Prediction of soil moisture content is a critical component of most hydrology and soil science models and is used for a variety of purposes, such as water supply management, flood forecasting, and drought monitoring. The amount of soil moisture storage will, in turn, determine how much water is available for plants and how much water will be available for precipitation and runoff.

Soil moisture is one of the most critical factors for the plant growth and development. It directly impacts plant health and crop yields. Unfortunately, current soil moisture prediction models are limited in their ability to accurately predict soil moisture.

This happens because current models generally focus on only one factor when determining soil moisture: the total amount of precipitation.

The amount of soil moisture in the pore space is important for plant health and crop yield. Field soils can be pore waterlogged or water deficient. Pore waterlogged soils are saturated with water, so much that water is available to plant roots through the soil texture.

2.2.2 SOIL MOISTURE DATASETS

One of the most important resources on our planet is soil moisture. Soil moisture is the amount of water in soil, and is an important factor in plant growth, crop yields, and the health of ecosystems. There are many ways to measure soil moisture, but two of the most common are air temperature and soil moisture content. The amount of heat given off by the Earth's surface is the primary factor in determining soil moisture content.

Plants need soil to grow, but soil is mostly air and water. When the soil is dry, the roots can't get the water they need. This can lead to a number of issues, including but not limited to: stunted, unhealthy growth, yellowing leaves and reduced yields.

Soil moisture plays a critical role in plant health and can even impact the weather. Soil moisture datasets are an important component of soil health monitoring because they provide an indication of when plants are most in need of moisture. There are two primary soil moisture datasets used to monitor plant health and performance: Precipitation and soil moisture. The precipitation dataset measures the amount of precipitation that has fallen on a given location over a given period of time.

The amount of moisture in the soil influences the amount of water that plants can absorb. The amount of water that is held in the soil is determined by the amount of precipitation and the structure of the soil. These are deciding factors.

The quality of the soil determines the quality of the plants that grow in it, and the quality of the plants that grow in it determines the quality of the food that is grown in it. The nutrient-rich soil that is necessary to grow healthy plants also holds water, which is necessary for plant growth. The amount of water in the

soil will vary based on the amount of moisture in the air, the amount of water that has been applied, and other factors. Plants take up water through their leaves, roots, and other parts.

2.2.3 EXISTING DATASETS

There isn't a single clear-cut dataset to tackle the whole soil moisture. The soil moisture dataset which consists of various attributes and rather large characteristics for the soil . The soil moisture is rapidly changing to climate variability.USA working on the datasets that remote sensing for the groundwater they took hydrologic water and river passage into consideration for soil moisture.For The SuSi framework is worked three major machine learning techniques like supervised and unsupervised and semi-supervised learning. Hyperspectral data on Semi supervised learning and(SUSI frame work) on the soi Their findings:

They demonstrate regression and classification results of the Supervised Self-organizing Maps in the SuSi framework .The soil sample consists of bare soil with no vegetation and was collected near Waldbronn, Germany.

The various parameter of soil like temperature, date with minutes and moisture are directly displayed to the user and depending on those parameters the crop is predicted by the system. It uses temperature, moisture for prediction of crops it uses various machine learning algorithms.

2.3 EXISTING TECHNOLOGICAL SOLUTIONS

The Soil moisture is a work under research and is widely being worked upon by various computer scientists worldwide, the following existing solutions aids in the building of the project's prediction model:

S. Prakash et al, 2018 [1] predicted Soil Moisture Using Machine Learning with combined three datasets with the comparison result demonstrates that multiple linear regression is bigger in terms of MSE and R2 of 0.14 and 0.975 for 1 day ahead, 0.353 and 0.939 for 2 days ahead, 1.59, and 0.786 for 7 days ahead.

D Ramesh et al, 2012 [2] demonstrated on the crop yield analysis , When compared to the actual average production, the MLR Technique estimates average production at 98 percent and the K-Means algorithm estimates average production at 96 percent with respect to four parameters.

Jeonghwan Hwnag et al, 2010[3] proposed the number of technological solutions have been developed to help farmers monitor and apply moisture to the soil at the right times. The Harvest Sapling sensor is a wireless sensor that is placed in the soil and measures the moisture in the soil. The sensor sends the data to the Harvest app on a smart phone, which farmers can use to apply moisture to the soil at the right times. The app also provides farmers with tips on when to apply fertilizer and other nutrients to the soil.

Umesh Acharya et al, 2021 [4] demonstrated that soil moisture at nearby weather stations had the greatest relative influence on moisture prediction, followed by 4-day cumulative rainfall and PET, and finally bulk density and Ksat with with mean absolute errors (MAE) of RFR, SVR, and BRT showed high correlations (r2 of 0.72,0.65 and 0.67) respectively.

Rupanjali D. Baruah et al., 2016 [5], Planters/farmers could use the technique to predict future crop productivity and take alternative adaptation measures as a result to maximise yield if the predictions fall short of expectations and results shown of multiple regressions that the developed model can be used to predict tea production in a particular region with precision.

NemanjaFilipovi et al., 2022 [6] provide an example. The system was tested with a Yr weather forecast, and the predictions were consistent with the ERA5 reanalysis data, demonstrating the system's potential to be used by farmers as an irrigation scheduling service, particularly during drought conditions.

M. Kashif et al 2006 [7] gives an application of a new learning tool, called Support Vector Machines (SVMs), for predicting soil moisture with the number of support vectors is 89 out of 180 data points.

Fan Chen et al. 2010 [8] show the potential for improving the Soil and Water Assessment Tool (SWAT) hydrological predictions of soil moisture, evapotranspiration and stream flow in the root zone of Southwest Oklahoma, and the attempt to improve the stream flow prediction is also attributed to the Inability attributed to the EnKF to correct existing biases in the SWAT-predicted flow components.

Sagarika Paul et al., 2020 [9] showed that the efforts is to improve the soil moisture prediction predicament through a machine learning methods . The data collected from the fields in 13 districts of West Bengal were used in this work for machine learning models for learning.

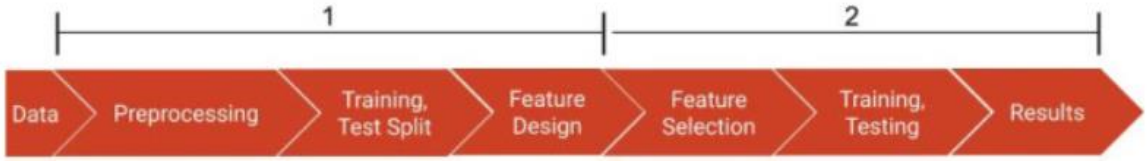

Figure 1: BASIC FRAMEWORK FOR MACHINE LEARNING ALGORITHM

Majorly most of the dataset follows these workflow the understand the dataset and using that to make application that useful for the environment.

2.4 TECHINCAL TOOLS

The following section summarizes all the technological software's or Application's and the packages needed for the implementation of the system.

2.4.1 PYTHON

Python is an interpreter language that is quite good at solving issues in general. It is one of the most widely used languages for scientific computations. It is capable of a wide range of tasks, from data analysis and mining to software

development. This project's major programming language is Python.

2.4.2 ANACONDA

Anaconda is a popular platform that gives users access to nearly all of the tools and software needed to complete a Data Science or Machine Learning project. It provides with various Development Environments such as Jupyter, Spyder, R Studio and many more. It also includes various Python libraries such as NumPy, Pandas, TensorFlow and many more to work with.

2.4.3 JYPTER NOTEBOOK IDE

Anaconda Navigator includes various variety of software such as Spyder, R Studio, Visual Basic and many more. One of them is Jupyter Notebook Development Environment. Jupyter mainly deals with scientific computations of data science as well as provides as easy platform to share live code documents. It is an open-source software.

2.4.4 NUMPY

NumPy is a Python library that specializes in numerical computations and is extremely fast at completing mathematical operations. NumPy is mostly utilized in this project for data preprocessing. It is used to efficiently conduct operations on picture arrays and prepare array features. In comparison to Python's math library, NumPy can also execute more efficient numerical operations.

2.4.5 MATPLOTLIB

Matplotlib is a Python package for plotting data. It allows programmers and developers to create a wide range of graphs and make data visualization simple. One of the prime features of Matplotlib is that it works brilliantly.

In the Jupyter environment and gives out easy to understand visualizations. It integrates well with NumPy as well as Pandas.

2.4.6 OPENCV

OpenCV, or Open-Source Computer Vision, is a well-known C/C++ library in the field of computer vision. Though written in C/C++, it has been used efficiently in other languages like Python and Java. It is one of the strong libraries available to work with images and image datasets. Tasks like manipulation and feature extraction become a lot easier with it

2.4.7 KAGGLE

Kaggle, a cloud service related to Google, is a large group of data science learners andmachine learning coders. It is a service which gives permission to a user to create dataset and build models on a given dataset or problem. Various tasks of deep learningand model training can be performed with it. It provides the user the access to both CPUand GPU to work with. Apart from being highly dominant, it is easy to use and create documents.

2.4.8 SEABORN

A Python visualization library for data analytics. Seaborn provides an expressive set of elementary statistical graphics that can be used to visualize your data in a variety of ways. It has extensive built-in support for common types of data visualization, such as bar plots, histograms, and scatter plots, but also makes extensive use of the Python standard library to provide a wider range of complementary visuals. Seaborn has been designed to be extensible, so it is easy to write new visualizations that can be added to the library**.**

2.4.9 SCI-KIT LEARN

The scikit-learn package provides powerful and flexible machine learning tools for learning and modeling in Python.

It has many high-quality implementations of common machine learning algorithms, as well as tools for data pre-processing, feature engineering, and model debugging. The scikit-learn library is a great starting point for building large-scale machine learning applications and versatile data processing pipelines.

2.4.10 PLOTLY

This package is designed to make it easy to get started with Plotly analytics. It includes a Plotly account, a Python client, a JavaScript client, and a dataset with pre-generated data.

2.5 CONCLUSION

Soil erosion is a big problem to protect soil and reduce financial losses. As a result of advancements in computer technology, the better understanding of soil problem but though using soil moisture prediction were are able to mointor the humidity of the soil and moisture of the soil . This is another application innovation that incorporates different specialized methods like artificial intelligence, sensor network, computer vision, etc.

In the next segment, various deep learning concepts, that have been used in the project, will be discussed. The main motive of the next section will be to develop and showcase various topics in deep learning and software development.

Chapter-3

SUPERVISED LEARNING

3.1 INTRODUCTION

When air temperature increases, the soil releases moisture into the atmosphere and when air temperature decreases, the soil holds onto moisture in the soil. This is how air

temperature is a predictor of soil moisture content. However, air temperature changes alone are not enough to measure soil moisture changes throughout the world. This is where the soil moisture sensor comes in. Supervised learning is a machine learning Method that uses a bundle of labeled data points (samples) to help the machine learn the relationship between input and output.

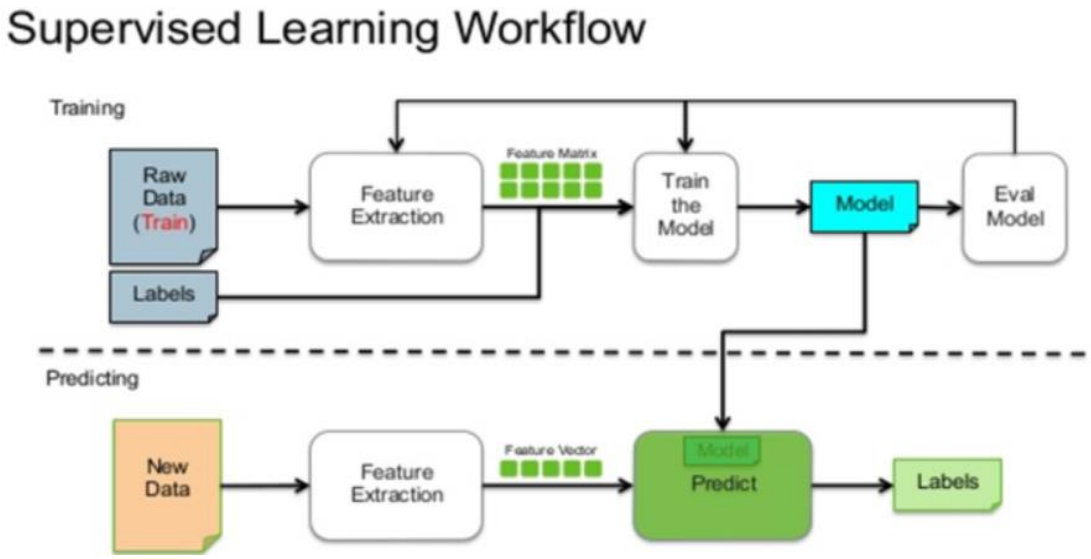

Figure 2: SUPERVISED LEARNING WORKFLOW

3.2 LINEAR REGRESSION MODEL

The linear regression model is used to predict the value of two a dependent variable as a function of one the independent variable. In other words, this model describes the relationship between one variable and another based on a straight line. LINEAR REGRESSION MODELS are used to model the relationship between a response and one or more explanatory variables. The goal is to find the best combination of variables that can explain the response as much as possible.

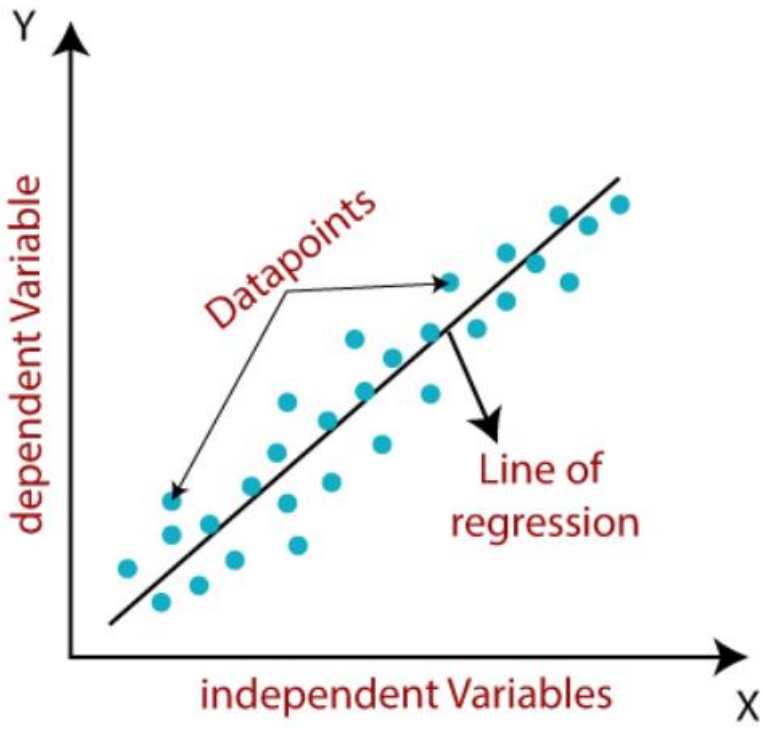

Figure 3: SCATTER PLOT SKECTH FOR LINEAR REGRESSION

3.2.1 ARCHITECTURE

The process of linear regression involves finding the best-fitting line through a bundle of data points. The slope and y-intercept of the best-fitting line are used to attain the value of the y-axis variable (represented by the value of the x-axis variable) for a new set of data. This process can be repeated many times, each time refining the line used to predict the future values of the y-axis variable. The result of the process is a line that represents the relationship between the x-axis and y-axis variables.

In statistics, regression is the process of determining the mathematical equations that relate one or more variables to the value of another variable or variables. Regression is used to predict the value of one or more variables based on the values of one or more other variables. The process of regression is critical to data scientists, as it is used to build models that can be used to make predictions and build machine learning algorithms.

One of the most basic forms of prediction is linear regression, which uses a set of data points to tell a story of how one variable changed over time with the help of another.

This is the simplest and most intuitive prediction method. When used well, it can be a powerful tool for understanding the world.

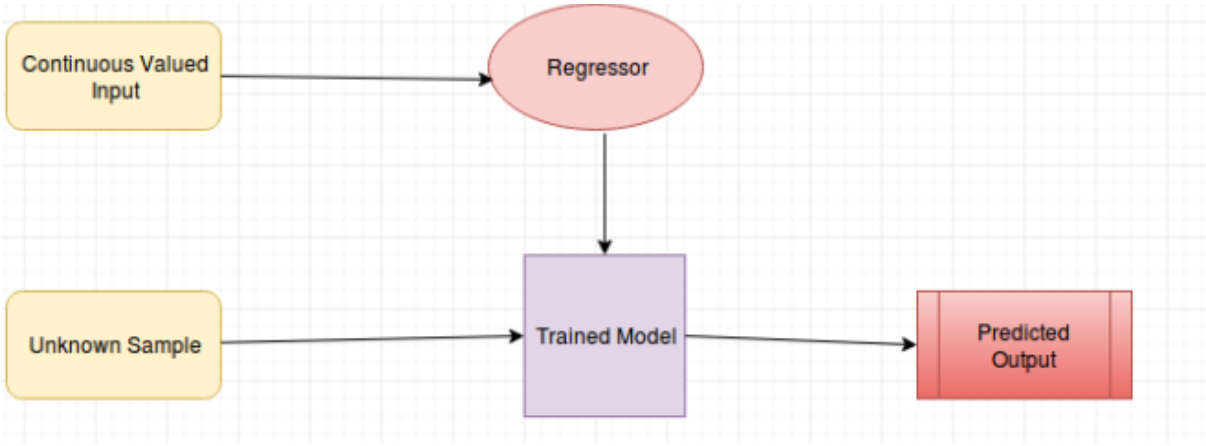

Figure 4: ARCHITECTURE DIAGRAM FOR LINEAR REGRESSION

3.3 ENSEMBLE METHODLOGY

One of the most common ways to improve the execution of a machine learning algorithm is to use a combination of different algorithms. This is often referred to as an ensemble method. There are many different ensemble methods, but they all have the same goal: to improve the performance of a single, specific algorithm. By combining the predictions of different algorithms, you can often increase the generalization of your predictions.

Some of the most popular machine learning algorithms are ensemble methods. The output of an ensemble method is the weighted average of the predictions of each model.

Another common type of machine learning algorithm is ensemble learning. Ensemble methods are groups of simple algorithms that are combined to improve performance on a given task. This type of machine learning is used

when it is difficult to hand-code an exact algorithm for a given task. In image classification, for example, it is difficult to hand-code the best hyperparameters for each algorithm in an ensemble.

Another common approach to machine learning is ensemble learning, which involves training several different machine learning models on the same dataset to achieve better performance than any one model could on its own. This is in contrast to hyper-parameter tuning, which involves adjusting the parameters of a single model to achieve the best performance possible. Ensemble methods combine the predictions of several models to produce a single prediction. Some of the most well-known ensemble methods are bagging and boosting.

Ensemble methods are a group of ml algorithms that work together to achieve better results than any individual algorithm. The most common types of ensemble methods are boosting and bagging. They both work by taking a number of individual models and training them together to increase the overall accuracy of the final model. This is in contrast to bagging and boosting algorithms, which are used to train individual models and then put them together to create a final model.

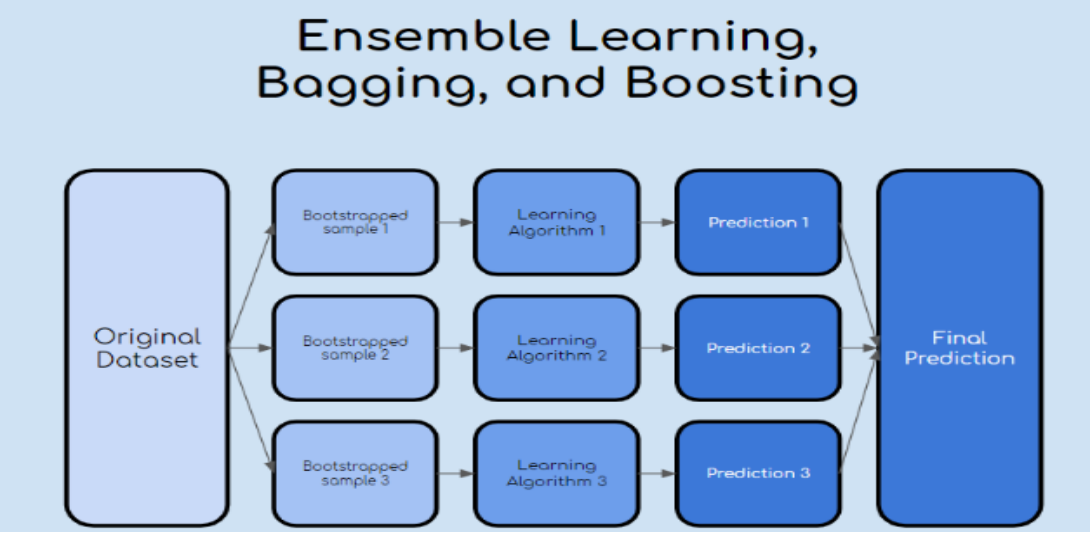

Figure 5:FRAME WORK OF ENSEMBLE ,BAGGING AND BOOSTING TECHNIQUES

3.3.1 BAGGING

One of the most common types of bagging in a random forest is bagging on random subclasses. In random forests, each decision tree is a separate model. That is, each tree in the forest is trained to make a different prediction. When you bag on random subclasses, you pick a random subset of the training data to train each tree on.

In random forests, bagging addressess to the process of splitting the training information into subsets and training multiple trees on each subset. This is in contrast to training a single tree on the entire training data. The benefit of bagging is that it boosts model performance by exploiting the special characteristics of different portions of the training data.

Random Forest is a machine learning algorithm that performs prediction by combining the predictions of individual trees. The Random Forest algorithm is a bagging algorithm, which means it groups data points into clusters and then trains separate models on each cluster. This is in contrast to tree-based algorithms, which are based on the premise that the output of a single model can be used to predict the value of a single variable. Each model in a bagging algorithm is trained on a subset of the original data.

3.3.2 BOOSTING

Boosting is a way utilized in device mastering algorithms to enhance overall performance. The concept of boosting is to introduce the desired data and "boost" it with additional attributes to improve the overall performance of the given ruleset.

In XGBoost, boosting is the process of iterating on a model to improve its performance on a given set of data. The main way to do this is to change the hyper parameters of the model, such that it performs better on the dataset. However, other methods exist to achieve the same result.

In the world of machine learning, boosting is the process of iteratively

improving the performance of a model on a dataset by using the previous or forward pass of the model as a basis for future predictions. The goal is to take the best performing model on the training dataset and use it as a starting point for the next generation model. This process is repeated multiple times to continuously improve the model. This type of machine learning is used in a variety of applications, including image and speech recognition, spam filtering, and Click Fraud detection.

3.4 RANDOM FOREST REGRESSOR

Random forest regressor is one of the powerful machine learning algorithms. It is a collection of decision trees, and each tree is grown from a random starting point. This means that each decision tree will have a unique, non-deterministic algorithm for choosing the next feature to split on. This produces a powerful classifier that can generalize well, but also makes it very hard Random forests are a type of machine learning methodology that is often avail for classification, and are especially useful for classification problems where the training data is limited or missing entirely. In a nutshell, random forests are ensemble methods that consist of many decision trees. Each individual tree in the ensemble is trained to make the best possible decision on its own, and the trees are combined at the end to make the final prediction. The randomness comes from the fact that the trees are grown independently and then combined at the end, rather than being grown together, to begin with to optimize**.**

3.4.1 ARCHITECTURE

Many machine learning algorithms are built on top of a base model, also called the architecture. The architecture is the set of parameters that determine the behavior of the model. In a random forest, for example, the architecture is the set of parameters that determine the behavior of the model. The random forest algorithm is a machine learning algorithm that uses many simple decision trees to make a single prediction.

One of the most common machine learning algorithms is random forest. Random forests are ensemble methods that combine the predictions of many individual machine learning models to produce a final prediction. Each tree in a random forest is an individual machine learning model that is trained on a subset of the data. The output of each tree is weighted by the probability that the tree is correct.

Ensemble methods combine the predictions of several machine learning models to produce a single, more accurate prediction. One of the most common ensemble methods is random forest, which is a collection of decision trees that are combined to make a single prediction. Random forest models are a great way to get started with ensemble methods. They are relatively easy to understand, implement, and tune.

One of the most common ways to construct an ensemble machine learning model is through the use of random forests. Random forests are machine learning algorithms that create composite decisions by combining the predictions of many decision trees. Random forests are a popular tool for large-scale machine learning because of their ability to make complex decisions about complex datasets with a large number of variables. In simple terms, a random forest is a collection of decision trees that are combined to produce a single, more powerful decision tree.

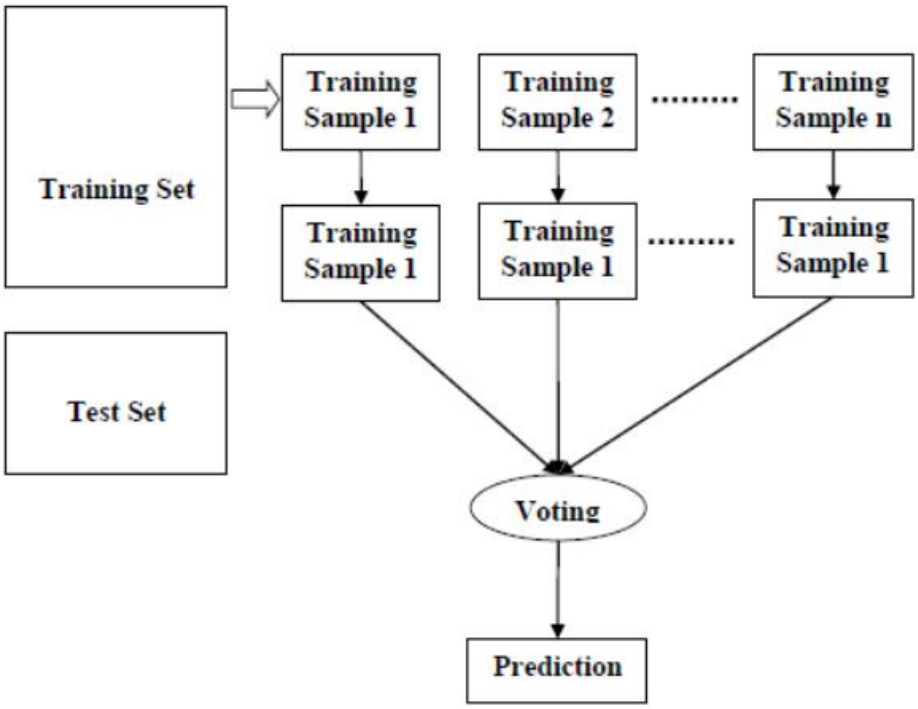

Figure 6: ARCHITECTURE DIAGRAM FOR RANDOM FOREST REGRESSOR

3.5 EXTREME GRADIENT BOOSTING

The most common approach to gradient boosting in deep learning is Xgboost. xgboost is a variant of Gradient Boosting, which is a machine learning technique for learning multiple features from a dataset. The xgboosting algorithm is a series of steps that are repeated until the optimization goals are met.. The process of building and testing the models is repeated with different algorithms called hyper-parameters.

The first step in the algorithm is called feature extraction. In this step, the data is separated into a training set and a test set. The training set is used to learn the model and the test set is used to evaluate the model.

The model achieves the needed outcomes. The algorithm starts by finding the best boosting parameters for the model on a training set. The next step is to train the model on a new dataset. Each step consists of a collection of boosting models, which are a set of algorithms that are combined together to form the Xgboost. The next step in the

Xgboost algorithm is to find the best-performing model in each of the previous steps. The best-performing model is then used to make predictions on the next step in the boosting algorithm.Till a set of criteria is met. The next step in Xgboost is to fit a series of boosting trees to the data, which are binary trees that split data into two subsets. Next, the algorithm finds the best subset for each training example. The final step of Xgboost is to find the set of boosting parameters that maximize the accuracy of the entire tree.

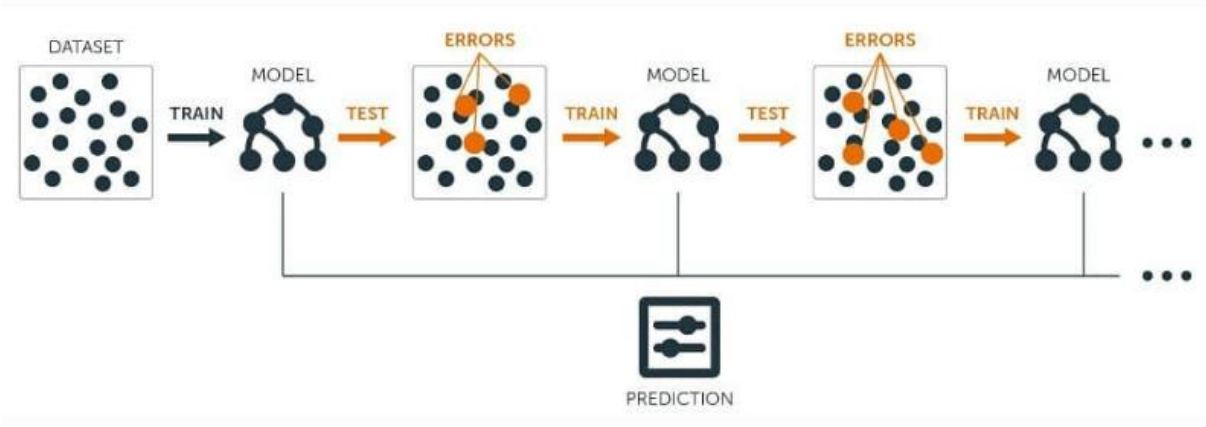

FIGURE 7: WORKFLOW DIAGRAM OF XG BOOST

3.5.1 ARCHITECTURE

The XGboost algorithm is a machine learning process that uses a boosting algorithm to improve accuracy. The algorithm is used in a variety of machine learning applications, and has been implemented in many different languages. XG Boost is a machine learning algorithm that can be used to increase the efficiency of a machine learning model in a variety of ways. The algorithm can be used to increase the performance of a machine learning model on a specific task, or in a machine learning context in general.

The approach is best suited to models that have been trained on large data sets, as it will not work for models that have been trained on small data sets. XG Boost is one of several algorithms that we have developed to help increase the performance of models in a machine learning context.

The goal of this project is to implement a Boost implementation of the XGboost algorithm. The algorithm is a variation of gradient descent, which tries to find

the parameters of a function that minimize the mean squared error. The XGboost algorithm is a variation of gradient descent that is able to solve very large problems. It is the algorithm of choice in machine learning for tasks that require large amounts of training data, such as image recognition and translation.

The XGboost algorithm is a data-driven boosting algorithm designed to give a near-optimal result in the most efficient way possible. It is an extension of the XGBOOST scoring function, which is itself an extension of the PageRank algorithm. The XGboost scoring function is used to give a near-optimal result in the most efficient way possible. It is an extension of the XGBOOST scoring function, which is itself an extension of the PageRank algorithm.

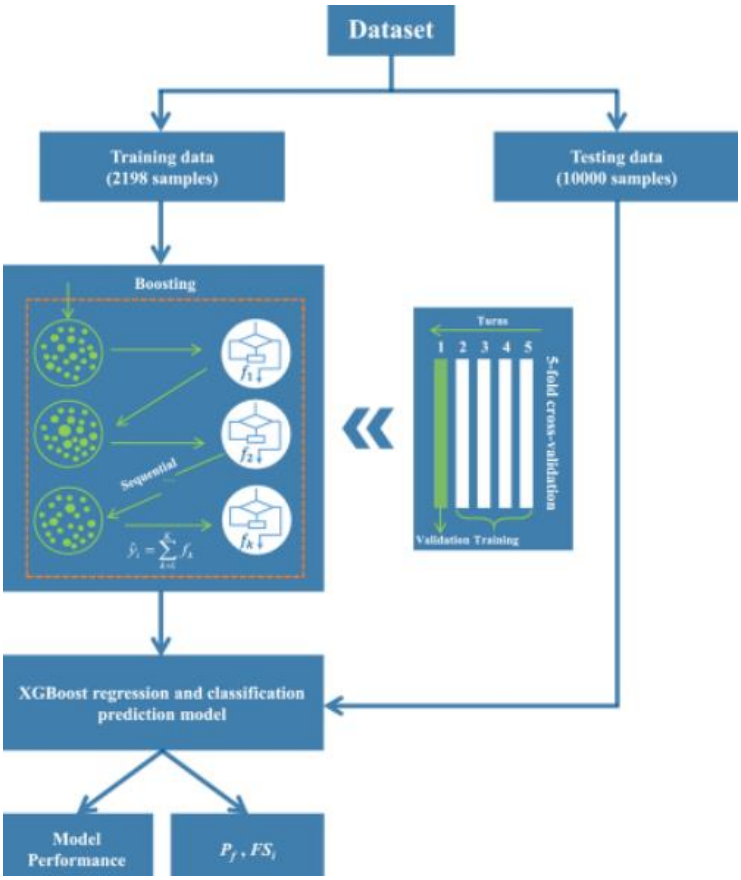

FIGURE 8: ARCHITECTURE DIAGRAM FOR XG BOOST

3.6 K- NEIGHBOR REGRESSOR

KNN, originally known as the K-Nearest Neighbors method, is a machine learning algorithm that employs a set of rules to categorize or identify data. KNN works by finding the closest match or neighbors for a given set of data. It is a good choice for smaller amounts of data where precision is more important than recall. KNN is a supervised learning algorithm, which means that the input variables (e.g. date of birth, address, etc.)

K-Nearest Neighbors is an algorithm that maps input data to output data by finding the point in the input space that is closest to a given point in the output space. It is one of the most well-known algorithms used in machine learning.

It is often used for text classification problems. KNN finds the k-nearest neighbors of the input text and uses their class labels to make the classification.

K-Nearest Neighbors (KNN) is a method for finding the category (or labels) that are most similar to a given object in a dataset by comparing it to the K nearest neighbors. This means that KNN is a super general approach that can be applied to learn a wide variety of models. KNN is often used to find categories in images or text, but it can also be used to predict the labels on a new dataset. For example, if you have a new dataset of animals, KNN could be used to predict the category that a new animal belongs to, such as bird, mammal, fish, etc.

K-Nearest Neighbor (KNN) is a unsupervised machine learning technique that uses a set of input data points to create a set of output predictions. The algorithm finds the points that are closest to an input point, in both space and time, and uses this information to create a set of output predictions. The KNN algorithm is a powerful tool for many applications, including classification, regression, clustering and dimensionality reduction.

3.6.1 ARCHITECTURE

The knn algorithm takes in a set of points as input and predicts a likely location for an unknown point.

The first step in training a deep learning model is to create a training dataset by collecting data from the source domain. In image classification, this might mean collecting images of a specific object or area. For example, if we want to train a model to identify road signs, we might take a large collection of images of signs and train our model on the sign images, hoping that our model will learn to identify signs in images. This process is called training the model.

The goal of the KNN algorithm is to predict which category a new observation belongs to in a dataset by using the observations that came before it. The KNN algorithm tells the model which categories are most likely to contain new observations. This reduces the need for expensive manual curation.

One of the most common machine learning algorithms is the k-Nearest Neighbor (kNN) algorithm. k-NN works by calculating the k-nearest values in a dataset to a given query value.

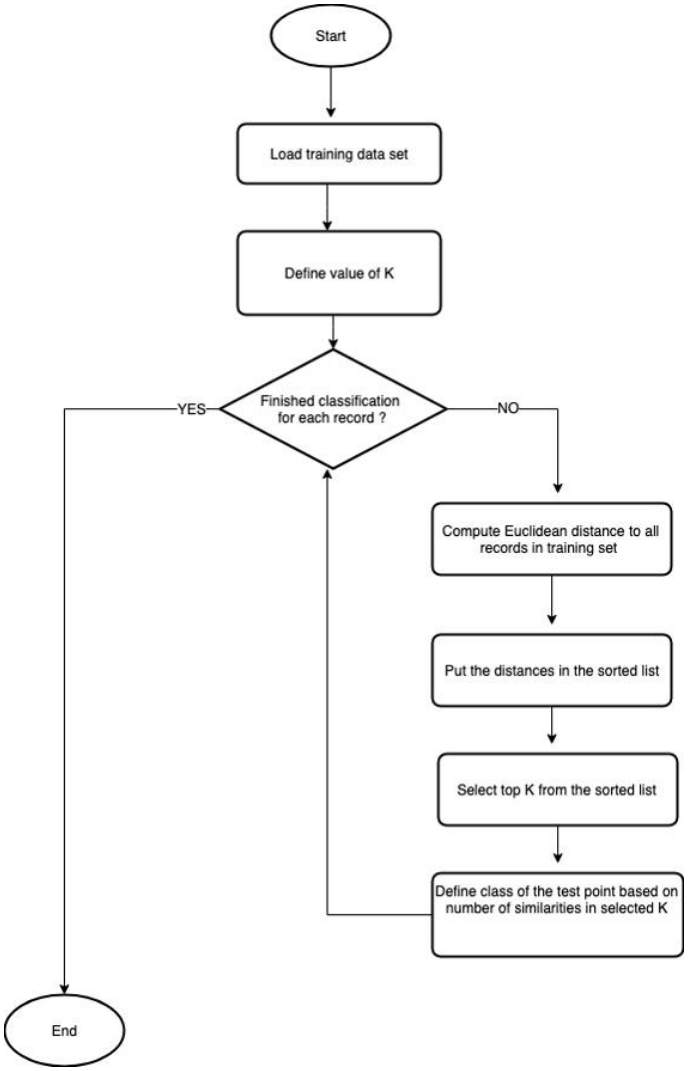

FIGURE 9: ARCHITECTURE DIAGRAM FOR K-NEIGHBOR

3.7 DECISION TREE

One of the most common algorithms used in machine learning is the decision tree. A decision tree is an algorithm that uses the outcome of a set of observations to determine which observations to examine next. The decision tree algorithm is one of the most effective algorithms for small to medium-sized data sets. It is often used to predict the outcome of a categorical variable.

The most common way to find a needle in a haystack is to scan through the haystack a few times, looking for a needle in each piece of hay.

It is an example of a brute-force approach to solving a problem. The decision tree algorithm is a very different approach that is better suited to finding needles in

haystacks. The decision tree algorithm works by creating a tree-like structure to organize the search space.

A decision tree is a model that is made up of multiple decision nodes. Each decision node represents a choice that needs to be made by a user,

Decision trees are a way of finding similarities and differences among sets of data. They are a powerful tool for finding patterns and trends in large sets of data. Decision trees are tree-based algorithms that can be used to efficiently find patterns and trends in data. They are especially useful for finding patterns and trends that are related to a specific question or hypothesis.

One of the most common ways to sort through large data sets is to use a decision tree algorithm. A decision tree is a data structure that can be used to represent the choices that a person might make when faced with a decision, such as what job to pursue or which college to attend. Each internal node in a decision tree is a decision that a person might make, and each leaf is a possible outcome of that decision.

3.7.1 ARCHITECTURE

A decision tree is a way of examining a set of training data and finding patterns within it. Using these patterns, a decision tree can make predictions about new data. To build a decision tree, the first step is to split the data into two sets: a training set and a test set.

When making a complex decision, it’s often the case that different factors are involved. When making these decisions, it’s helpful to be able to analyze the different factors involved and weigh them up and consider which are the most important.

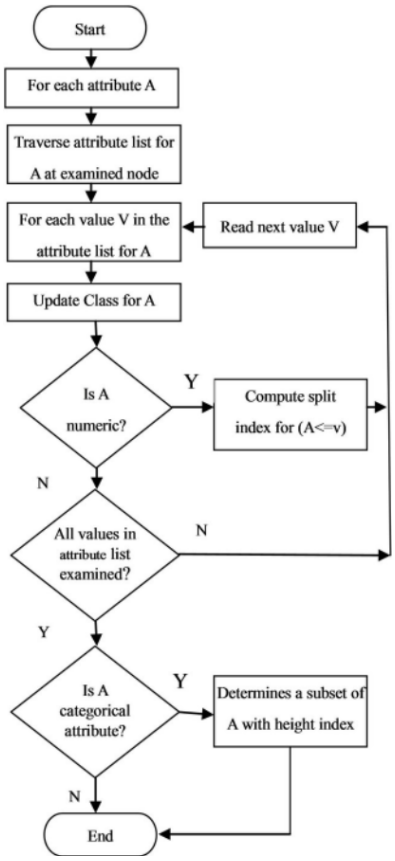

FIGURE 10: ARCHITECTURE DIAGRAM FOR DECISION TREE

3.8 CONCLUSION

With these views in mind, one can have a clearer understanding of what is going on inside a machine learning model during its training period. When training deep learning models, there is some black-box uncertainty, but these ideas provide a better understanding of what might be going on when a network doesn't operate as expected. In short, knowing these theories aids in the correction of the code.

Chapter -4

FEATURE SCALING AND MODEL SELECTION

4. INTRODUCTION

Hyperparameter tuning is an often underutilized technique for improving the performance of machine learning models. In most cases, hyperparameter tuning involves selecting a subset of hyperparameters from a machine learning model and measuring the effects of those hyperparameters on the model. This process is then repeated for each of the possible hyperparameter values in the space until the model is no longer improved on the training dataset.

4.1 OVERVIEW

The hyperparameters are the settings for a machine learning algorithm. A hyperparameter is a parameter that can be modified during data analysis. You can use hyperparameters to optimize an algorithm's performance on a given dataset. A hyperparameter is usually a number that can take on many different values. In a machine learning algorithm, each hyper- parameter is a value that the algorithm tries to find. The algorithm iterates through a number of possible values for each hyper-parameter and chooses the best value for that parameter. We can think of the hyper-parameter as a tuning knob, one that we can turn to find the best value. However, we can also think of the hyper- parameter as a random variable, which will usually have a range of possible values.

4.2 HYPER-PARAMETER OPTIMIZATION

Hyperparameter tuning is an important step in machine learning because it provides a way to fine-tune the parameters of a model so that it captures the structure of the dataset while reducing the risk of overfitting. Hyperparameter tuning is often performed by selecting a subset of the hyperparameters in a model, measuring the effects of those hyperparameters on the model, and then changing the values of only those hyperparameters that were effective.

This process is then repeated for each of the possible hyperparameters in the model

until the model no longer performs better on the training dataset. This process is often referred to as hyperparameter tuning because it is used to tune the parameters of a model.

Hyperparameter tuning is a technique for adjusting the parameters of a machine learning model so that it performs as well as possible on the data of interest. Hyperparameters are the parameters of a model that are tunable using techniques such as tuning curves and grid search. Hyperparameter tuning is often used to increase the performance of a machine learning model. Perhaps the most common use for hyperparameter tuning is to increase the accuracy of a model.

Hyperparameter tuning is an essential part of deep learning training. Hyperparameters are the settings used to define the structure of the neural networks that perform the task of prediction. Hyperparameters are often tuned to optimize the performance of the model on the training dataset. However, without hyperparameter tuning, models will not generalize well to unseen data.

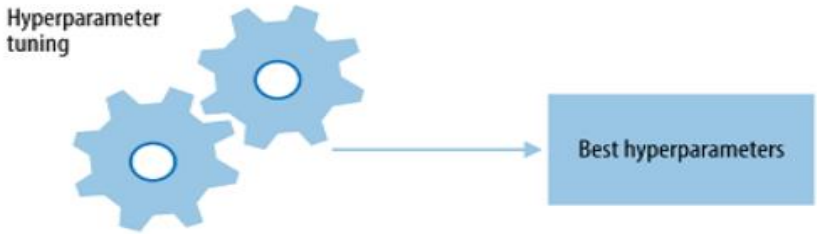

FIGURE 11: BASIC INTEPRETATION OF HYPERPARMETER TUNING

4.3. GridSearchCV

Hyperparameter tuning is often performed using a technique called grid search. Grid search is a search algorithm that iterates over a range of hyperparameters in order to find the best hyperparameter settings for a given machine learning model. Grid search is useful for hyperparameter tuning because it can be used to search over a large range of values for a single hyperparameter. This makes it possible to tune over a large set of possible hyperparameter settings in a short amount of time.

In a grid search, a set of hyperparameters are randomly initialized and then the model is trained on the data using each of the possible hyperparameters. After the model has been trained on the data, the performance of the model is measured on the test dataset.

Grid search is a technique for optimizing a machine learning model by searching over the space of possible hyperparameters. The general idea of grid search is to design a model, create a grid of hyperparameter values, fit the model to the training data using the first hyperparameter value in the grid, evaluate the model on the training data using the second hyperparameter value in the grid, and so on until the model is found to perform best on the training data. This process is then repeated for each hyperparameter value in the grid until the model is no longer found to perform better on the training data.

When performing a grid search, a set of hyperparameters are randomly selected from the dataset and their effects on the model are measured. This process is then repeated with another set of hyperparameters and the model's performance is measured again. The most common way to tune hyperparameters is through the use of grid search. Grid search is a search algorithm used to optimize hyperparameters of a machine learning model. In grid search, the hyperparameters of a machine learning model are divided into small, independent units called cells. In each cell, the hyperparameter of the model is iteratively optimized using a search algorithm.

4.3.1 ARCHITECTURE

The architect of modern machine learning is the grid search, an optimization technique that harnesses the power of computers to quickly find the best model parameters. The grid search is a framework for finding the best hyper parameters for a machine learning model. It is a form of stochastic optimization that uses a set of candidate solutions to the problem at hand to produce a model. The grid search is a refinement of the random search, a form of stochastic optimization that uses random sampling to find the best candidate solutions to the problem at hand.

Machine learning remains a critical path to autonomous vehicles, medical

diagnosis, and many other breakthroughs. But it is also an extremely difficult field to master.

This is particularly true for computer vision, the discipline of processing images and video with the help of artificial intelligence. To get the best results from computer vision algorithms, researchers need to understand how they work at a very fundamental level.

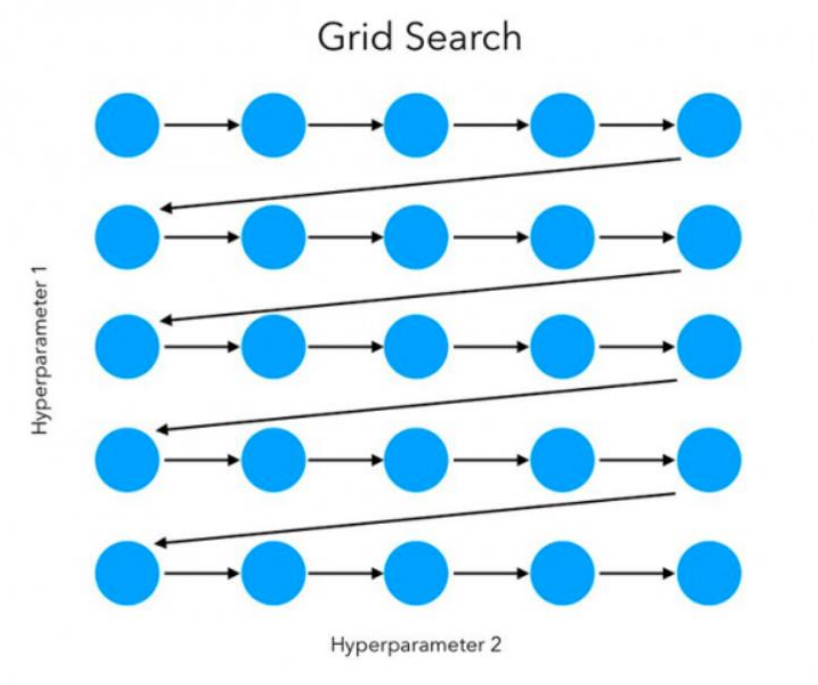

FIGURE 12: SYSTEMATICAL REPRESNTATION OF GRID SEARCHCV

4.3.2 MODEL IMPLEMENTATION

Hyperparameter tuning is an essential part of machine learning that involves adjusting the values of hyperparameters, such as the number of neurons in a neural network, during training to achieve optimal performance. Hyperparameters are the key to making deep learning models work well on complex tasks, but finding the best values for hyperparameters can sometimes be difficult. Hyperparameter tuning, or grid search, is a common method for finding the best values for hyperparameters.

This algorithm is one of the most efficient ways to tune hyperparameters for a

machine learning model. It allows you to iteratively find the best set of hyperparameters for your model using a grid search. We will use this algorithm to tune the hyperparameter of our SVM model.

Hyperparameter tuning is a critical step in deep learning training, particularly for high-dimensional problems. Hyperparameter tuning is the process of finding parameters for a deep neural network that best fits a given dataset. Grid search is an optimization technique that allows for fast hyperparameter optimization by systematically searching for the best parameters.

Hyperparameter tuning is a technique used to find the "best" or "optimal" values for parameters that control the behavior of a model. Hyperparameter tuning is often used to tune the parameters of machine learning models to optimize model performance on a given dataset. One of the most common ways to tune hyperparameters is to use grid search.

Hyperparameter tuning is a process of repeatedly creating, testing, and evolving a set of hyperparameter values in an attempt to best fit the training data. The process of hyperparameter tuning can be broken down into three steps: defining the search space, performing a grid search, and making a final choice.

4.4 RANDIOMISED SEARCHCV

Randomized SearchCV takes the idea of grid search one step further by creating random sets of hyperparameter values that are used to search over the space of possible hyperparameters. Rather than selecting a single hyperparameter value to use as a starting point for the search, randomized sets are used to create a grid of hyperparameter values to use as a starting point for the search. This process is then repeated for each of the possible hyperparameter values in the grid until the model is no longer found to perform better on the training data.

Hyperparameters are randomly selected from the dataset and their effects on the model are measured. This process is then repeated with a different set of hyperparameters and

the model's performance is measured again.

.Randomized SearchCV is a technique for hyperparameter tuning that uses random selection to perform hyperparameter searches.
Instead of designing a grid of hyperparameter settings and performing hyperparameter searches using the first hyperparameter value in the grid, Randomised SearchCV randomly selects a hyperparameter value from the dataset and then searches over the space of possible hyperparameter values using the selected hyperparameter value.

A slightly more advanced technique for performing a grid search is to randomize the hyperparameters that are used. In this scenario, rather than repeatedly selecting the same set of hyperparameters, a different set of hyperparameters is chosen each time the grid search is performed. Randomized searchCV is useful for hyperparameter tuning because it can be used to search over a large range of possible hyperparameters in a short amount of time.

Hyperparameter tuning can be performed using a technique called Randomised SearchCV. In Randomised SearchCV, instead of designing a grid of hyperparameters and performing a single iteration of the grid search, a random subset of the hyperparameters is chosen at each iteration and the model is trained on the dataset using the hyperparameters in the current random subset. This process is then repeated for each random subset until the model is no longer found to perform better on the training data. This process is then repeated for each random subset of the hyperparameters, providing a comprehensive way to search over a large range of hyperparameter values in a short amount of time.

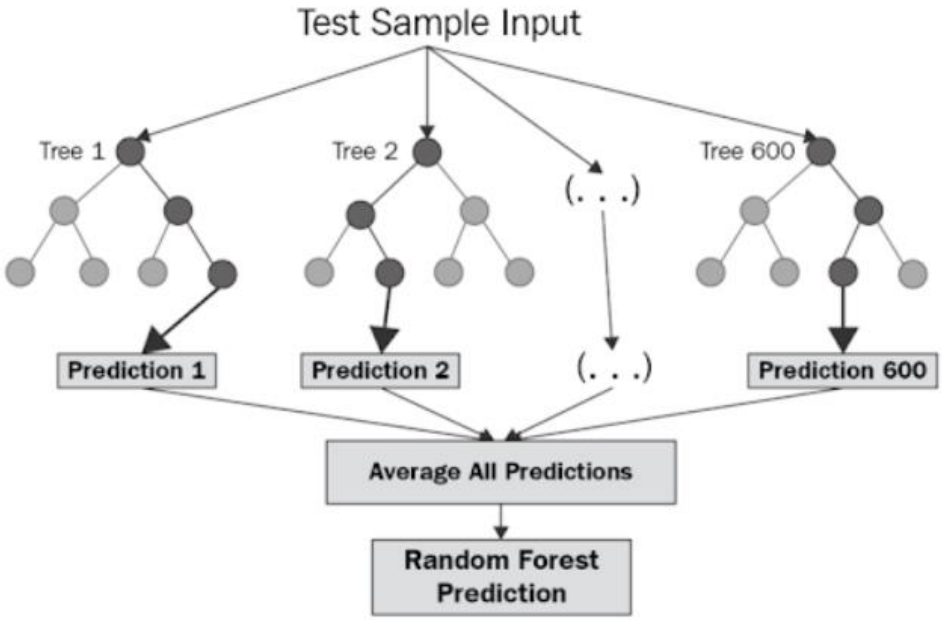

FIGURE 13: SYSTEMATICAL REPRESNTATION OF RANDOM FOREST ON RANDOMISED SEARCHCV

4.4.1 IMPLEMENTATION OF MODEL

Hyperparameter tuning is a powerful tool for improving the performance of machine learning models. However, it is often difficult to find the optimal hyperparameters for a given problem. This can be a real challenge when the number of hyperparameters to tune is large, or when the desired performance is difficult to measure.

Hyperparameter tuning is a technique used to fine-tune the parameters of a machine learning model. When designing a machine learning model, the first step is to identify the characteristics of the data that the model is expected to process. Once the data has been identified, the next step is to design a model to process that data. This process is generally iterated until the model is found to perform best on that data.

Hyperparameter tuning is the process of searching over hyperparameters of a machine learning model in order to find the best hyperparameters for a specific problem. Hyperparameters are the parameters of a machine learning model that affect the model's ability to perform a specific task. Hyperparameters include

everything from the learning rate used to train a model to the size of a model's input and output data sets. One of the most common ways to tune hyperparameters is through the use of grid search.

Hyperparameter tuning is the process of adjusting the values of hyperparameters of a machine learning model to achieve optimal performance on a given dataset. Hyperparameter tuning can be difficult and time-consuming, which is where RandomisedSearchCV comes in. Using RandomisedSearchCV, you can set aside a fixed amount of time to tune your hyperparameters, and it will work iteratively to get you the best possible solution in that time.

4.4.2 ARCHITECTURE

Randomized searchCV is probably one of the most important machine learning algorithms. Randomized SearchCV is very useful when we need to test many parameters and the training time is very long. In Randomized searchCV, the first step is to write the parameters that we want to consider and from these parameters select the best ones. It leverages modern data-intensive computing to tackle the most difficult, time-consuming, and resource-consuming parts of the search. It can be used to identify patterns in data, particularly in large datasets with many columns and rows. It is used to find the best hyperparameters in other machine learning algorithms.

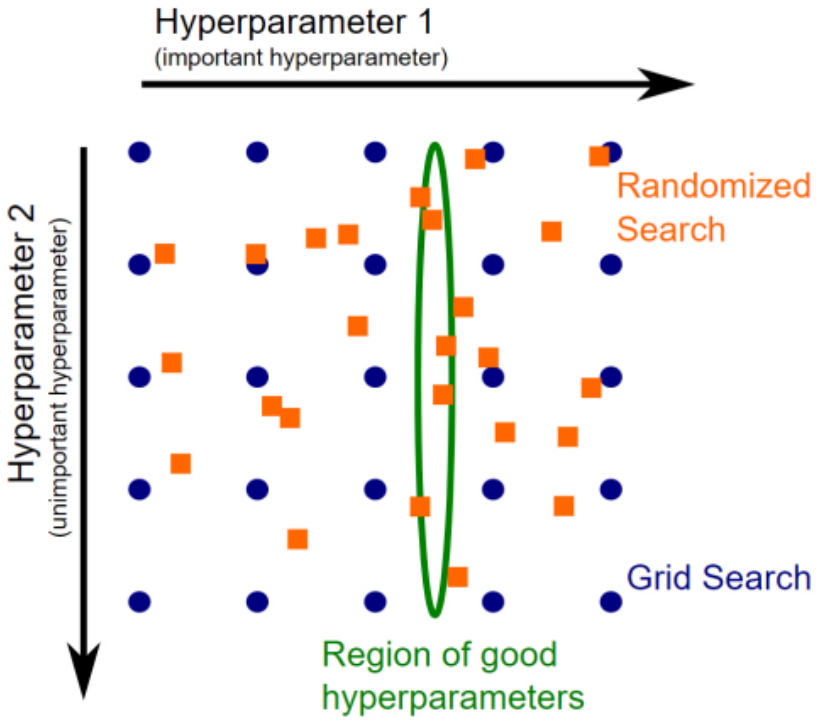

FIGURE 14: ARCHITECTURE MODEL FOR GRID SEARCHCV AND RANDOMIZED SEARCHCV

4.5 CONCLUSION

Hyperparameter tuning is all about fine-tuning your algorithm's parameters to achieve optimal performance and finding the right balance between the trade-offs between the parameters, such as choosing between using millions of examples to learn a model, or using fewer examples but with more data to improve the model's performance.

Hyperparameter tuning is one of the most powerful ways to tune a machine learning model. It helps you avoid overfitting the training data, which can be dangerous. It can also help you avoid overfitting to the data you have available, which is often the case when tuning a model on just the training data.

It's the process of fine-tuning the parameters of your model so that it performs optimally on your data.

Hyperparameter tuning is the process of fine-tuning your deep learning model so that it performs well on your training data but not on other data. Hyperparameter tuning is a highly iterative process and requires experimentation to optimize your model.

Chapter-5

SYSTEM DESIGN

5.1 INTRODUCTION

The aim of this chapter is to delve into the system's architecture and its proposed workings. The modules and the end-to-end flow is highlighted out in this section.

5.2 GANTT CHART PLAN

Soil moisture data which we are collected from two dataset has been alternate into one dataset to make the process effective because using machine learning in dataset more space and more running and execution time to optimize the process by not comprising the data. The two dataset has been combined the one dataset.

GANTT CHART	weeks														
ACTIVITY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
STUDY MACHINE LEARING	x	x	x												
STUDY JYPTER NOTEBOOK AND ANACONDA				x	x	x	x	x	x						
SUPERVISED LEARNING				x	x	x	x	x	x	x	x				
MACHINE LEARNING ALGORITHMS				x											
DATA ENVIRONMENT SETUP							x	x	x	x	x	x	x		
DATA UNDERSTANDING				x	x										
DATA PREPARATION				x	x					x					
DATA MODELLING					x	x	x	x	x	x					
DATA EVALUATION					x	x	x	x	x	x					
FIRST REVIEW						x	x				x	x	x	x	
SECOND REVIEW					x										
DATASET MODELING										x					
MODEL TRAINING				x	x										
TESTING								x	x	x	x	x	x	x	
ANALYSIS										x	x	x	x	x	
FINAL REVIEW						x	x	x	x	x	x	x	x	x	
												x	x	x	
															x

FIGURE 15: GANTT CHART REPRESENTATION

5.3 DESIGN FLOW

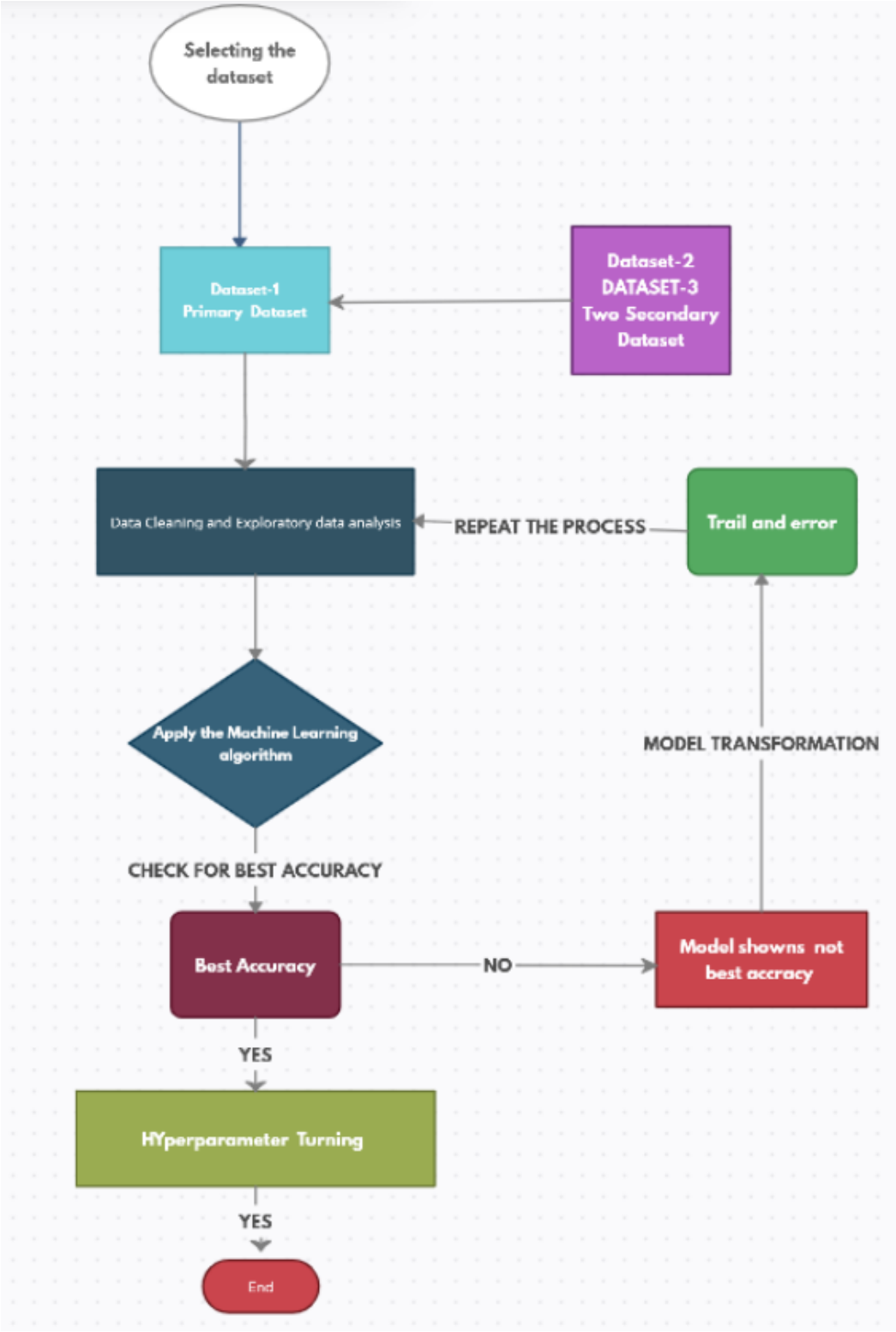

FIGURE 16: DESIGN FLOW FOR PROJECT

The figure, shows the system flow of the project. The end to end flow has five stages. The first stage is the identification of the dataset. It is not easy to find the dataset that has required size with novelty to predict the accuracy for the dataset.
Data collecting is the primary step in this project, two secondary dataset has been combined into one primary dataset but here lies the problem of duplicated values and attributes. To solve that problem, we are moving into

The next stage is where the data cleaning for two secondary dataset into one primary dataset. Combining the two dataset into one bigger step is easier to alternating the two secondary dataset into one primary dataset. After Alternating the primary dataset. we can careful for the losing the important metrics. To solve this problem , we are moving into next step

The third stage , Exploratory analysis- Exploratory analysis gives the best idea about the individual data attributed and Metrics which primary keys to the dataset execution and to explore the use of machine learning methods to improve the accuracy of image analytics on the use of machine learning algorithms.

The fourth step is the machine learning methods is, In this step the dataset is to ready to train or test the dataset but choosing the particular machine learning is tedious job .For few Machine learning even with train and train of the dataset there won't accuracy values. In the step ,I decided to model the five different algorithms which are

XGBOOST,
LINEAR REGRESSION,
RANDOM FOREST
K-NEIGHBOUR,
DECISION TREE

After the prediction of each algorithm, each algorithms gives the accuracy values and prediction values.

Final step is the hyper parameter tuning for the best two algorithms will take place and that will gives the best accuracy for the dataset

5.4 CONCLUSION

The entire workflow of the system as been detailed out and the models used in each stage have been explained. Understanding of the models used and the system flow helps in the implementation of the system.

The next chapter is Implementation, and it highlights the step-by-step procedure with code snippets

Chapter-6

IMPLEMENTATION

6.1 INTRODUCTION

This section of the project describes how it will be implemented. The section includes the step-by-step implementation of all the modules involved and the flow in which they take place.

6.2 PROJECT FLOW

The section aims to display the flow of the project with respect to the code structure
In this project we have two secondary dataset and primary dataset so I am going to process with respective code.

Anaconda Navigator is a graphic user Interface were we have the Jypter notebook Idle we use the python as main programming language to understand the dataset and to make the prediction with accuracy.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Figure 17: CODE: IMPORTING THE PACKAGES

To begin with, Importing the necessary packages for the three datasets ,After the import of the packages

6.3 DATA UNDERSTANDING

Data understanding is important play in the project without understanding the data and **what the attributes are plays an important role in the predictive model,** so we going to see the data understanding for the two secondary datasets.

Dataset -1

```
run.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10289 entries, 0 to 10288
Data columns (total 12 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   year       10289 non-null  int64
 1   month      10289 non-null  int64
 2   day        10289 non-null  int64
 3   hour       10289 non-null  int64
 4   minute     10289 non-null  int64
 5   second     10289 non-null  int64
 6   moisture0  10289 non-null  float64
 7   moisture1  10289 non-null  float64
 8   moisture2  10289 non-null  float64
 9   moisture3  10289 non-null  float64
 10  moisture4  10289 non-null  float64
 11  irrgation  10289 non-null  bool
dtypes: bool(1), float64(5), int64(6)
```

Figure 17.1 :RESULT ON DATA TYPES FOR DATASET -1

Dataset-2

```
stop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4117 entries, 0 to 4116
Data columns (total 12 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   year       4117 non-null   int64
 1   month      4117 non-null   int64
 2   day        4117 non-null   int64
 3   hour       4117 non-null   int64
 4   minute     4117 non-null   int64
 5   second     4117 non-null   int64
 6   moisture0  4117 non-null   float64
 7   moisture1  4117 non-null   float64
 8   moisture2  4117 non-null   float64
 9   moisture3  4117 non-null   float64
 10  moisture4  4117 non-null   float64
 11  irrgation  4117 non-null   bool
dtypes: bool(1), float64(5), int64(6)
```

Figure 18 :RESULT ON DATA TYPES FOR DATASET -2

Dataset-3–primarydataset

```
df.info()
```

```
Data columns (total 12 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   year       4409 non-null   int64
 1   month      4409 non-null   int64
 2   day        4409 non-null   int64
 3   hour       4409 non-null   int64
 4   minute     4409 non-null   int64
 5   second     4409 non-null   int64
 6   moisture0  4409 non-null   float64
 7   moisture1  4409 non-null   float64
 8   moisture2  4409 non-null   float64
 9   moisture3  4409 non-null   float64
 10  moisture4  4409 non-null   float64
 11  irrgation  4409 non-null   bool
dtypes: bool(1), float64(5), int64(6)
```

Figure 19 :RESULT ON DATA TYPES FOR DATASET -3

So After understanding the data types for each attributes and important characeristics of the dataset.

6.4 DATA PREPARATION

It is an iterative process , going one by one through all the features and applying operations to improve the peformance of the model.so that we can combine some of the strong features and try to improve the model. It will take a lot of trial and error.

DATASET-1

```
print(list(stop))
```

```
['year', 'month', 'day', 'hour', 'minute', 'second', 'moisture0', 'moisture1', 'moisture2', 'moisture3', 'moisture4', 'irrgatio
n']
```

Figure 20 :CODE : DATA PREPROCESSING OF THE DATASET-1

DATASET-2

```
print(list(run))
```

```
['year', 'month', 'day', 'hour', 'minute', 'second', 'moisture0', 'moisture1', 'moisture2', 'moisture3', 'moisture4', 'irrgatio
n']
```

Figure 21 :CODE : DATA PREPROCESSING OF THE DATASET-2

DATASET-3 – PRIMARY DATASET

```
print(list(df))
```

```
['year', 'month', 'day', 'hour', 'minute', 'second', 'moisture0', 'moisture1', 'moisture2', 'moisture3', 'moisture4', 'irrgatio
n']
```

Figure 22 :CODE : DATA PREPROCESSING OF THE DATASET-3

Though we get the same output for three dataset we have to make sure the rows and columns are concide with three dataset , That means the dataset is the same level which make our process easy to easy for the data preprocessing. Here we found out the columns and rows of the dataset . The next step we have to take out the duplicated without discarding the important attributes or characteristics.

DATASET -3 – PRIMARY DATASET

```
df.head()
```

	day	hour	minute	second	moisture0	moisture1	moisture2	moisture3	moisture4
0	6	22	16	11	0.33	0.40	0.36	0.23	0.02
1	6	22	17	11	0.32	0.39	0.35	0.23	0.02
2	6	22	18	11	0.31	0.39	0.34	0.22	0.02
3	6	22	19	11	0.30	0.38	0.33	0.21	0.02
4	6	22	20	11	0.29	0.38	0.33	0.21	0.02

Figure 23 :CODE : DATA PREPARATION OF THE DATASET-3

DATASET -1 – FIRST SECONDARY DATASET

```
stop
```

	year	month	day	hour	minute	second	moisture0	moisture1	moisture2	moisture3	moisture4	irrgation
0	2020	3	6	22	16	11	0.70	0.64	0.73	0.40	0.02	False
1	2020	3	6	22	17	11	0.70	0.64	0.71	0.39	0.02	False
2	2020	3	6	22	18	11	0.69	0.63	0.70	0.39	0.02	False
3	2020	3	6	22	19	11	0.69	0.63	0.70	0.39	0.02	False
4	2020	3	6	22	20	12	0.69	0.62	0.69	0.39	0.02	False
...	...	...	...	...	...	...	...	...	...	...	...	...
4112	2020	3	9	19	12	49	0.18	0.62	0.60	0.10	0.03	False
4113	2020	3	9	19	13	49	0.18	0.62	0.59	0.10	0.02	False
4114	2020	3	9	19	14	49	0.18	0.62	0.59	0.10	0.02	False
4115	2020	3	9	19	15	49	0.18	0.62	0.60	0.10	0.02	False
4116	2020	3	9	19	16	49	0.18	0.63	0.60	0.09	0.02	False

4117 rows × 12 columns

Figure 24 :CODE : DATA PREPRAATION OF THE DATASET-1

```
stop.head()
```

	day	hour	minute	second	moisture0	moisture1	moisture2	moisture3	moisture4
0	6	22	16	11	0.70	0.64	0.73	0.40	0.02
1	6	22	17	11	0.70	0.64	0.71	0.39	0.02
2	6	22	18	11	0.69	0.63	0.70	0.39	0.02
3	6	22	19	11	0.69	0.63	0.70	0.39	0.02
4	6	22	20	12	0.69	0.62	0.69	0.39	0.02

Figure 25 :CODE : DATA PREPRAATION OF THE DATASET-1

DATASET -2 – SECOND SECONDARY DATASET

	year	month	day	hour	minute	second	moisture0	moisture1	moisture2	moisture3	moisture4	irrgation
0	2020	3	11	14	44	39	0.59	0.63	0.51	0.45	0.01	False
1	2020	3	11	14	47	9	0.56	0.60	0.52	0.48	0.01	False
2	2020	3	11	14	49	39	0.56	0.54	0.54	0.51	0.01	False
3	2020	3	11	14	52	9	0.56	0.50	0.57	0.51	0.01	False
4	2020	3	11	14	54	39	0.57	0.53	0.58	0.51	0.01	False
...	...	...	...	...	...	...	...	...	...	...	...	...
10284	2020	3	29	11	53	33	0.03	0.96	0.93	0.99	0.01	False
10285	2020	3	29	11	56	3	0.03	0.96	0.90	0.99	0.01	False
10286	2020	3	29	11	58	33	0.03	0.96	0.91	0.99	0.01	False
10287	2020	3	29	12	1	3	0.03	0.96	0.89	0.99	0.01	False
10288	2020	3	29	12	3	33	0.03	0.96	0.89	0.99	0.01	False

```
run.head()
```

	year	month	day	hour	minute	second	moisture0	moisture1	moisture2	moisture3	moisture4
0	2020	3	11	14	44	39	0.59	0.63	0.51	0.45	0.01
1	2020	3	11	14	47	9	0.56	0.60	0.52	0.48	0.01
2	2020	3	11	14	49	39	0.56	0.54	0.54	0.51	0.01
3	2020	3	11	14	52	9	0.56	0.50	0.57	0.51	0.01
4	2020	3	11	14	54	39	0.57	0.53	0.58	0.51	0.01

Figure 26 :CODE : DATA PREPRAATION OF THE DATASET-2

In these dataset, two secondary dataset having different attributes and different values, In this year 2020 is taken into consideration , we can also compare the period of time to find out the soil moisture for specific period. This will give clear idea about the soil moisture content.

Here in these commands we can understand the sensors has been recording the values for for different months and dates and moreover added with hours, minutes and seconds, if we recorded with sunset we can find out the atmospheric moisture and soil moisture to get better understanding of the soil moisture. Soil Moisture data which are collected where sensors gives the values for the moisture contained on irrigation system for Moisture 0, moisture1, Moisture 2, moisture 3, moisture 4 are containing soil moisture data in different layers.

DIFFERENT PLOTS FOR THE DATASET.

To get better grasp of the moisture values like Moisture 0, moisture 1,moisture 2,

moisture 3, moisture 4, moisture 5.

Axes Subplot

DATASET- 1

```
stop[['moisture0', 'moisture1','moisture2', 'moisture3','moisture4']].plot()
```

```
<AxesSubplot:>
```

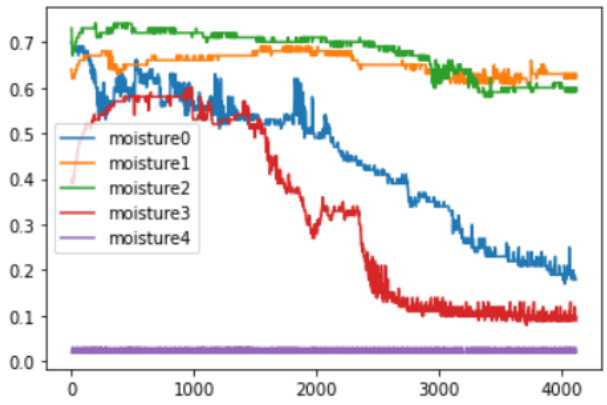

Figure 27 : AXES SUBPLOT FOR MOISTURE ATTRIBUTES on Dataset -1

DATASET-2

```
run[['moisture0', 'moisture1','moisture2', 'moisture3','moisture4']].plot()
```

```
<AxesSubplot:>
```

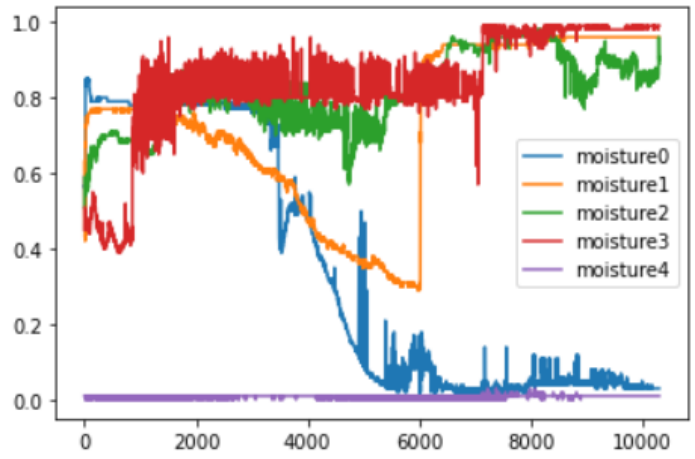

Figure 28 : AXES SUBPLOT FOR MOISTURE ATTRIBUTES on dataset-2

DATASET-3

```
df[['moisture0', 'moisture1','moisture2', 'moisture3']].plot()
```

```
<AxesSubplot:>
```

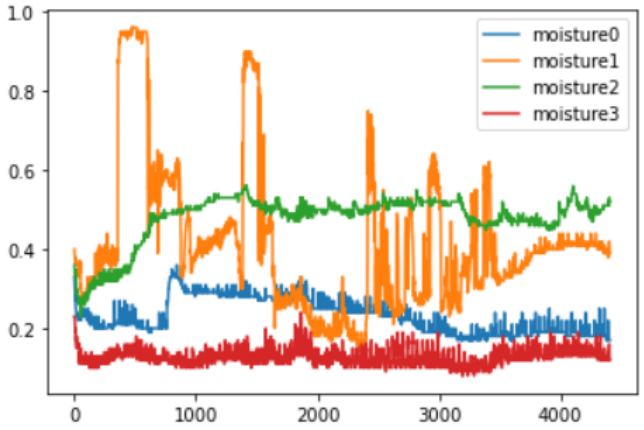

Figure 29 : AXES SUBPLOT FOR MOISTURE ATTRIBUTES on dataset-3

These Axes subplot shows how the data has been distributed that we can find out which attributes we can make constant to make the understand the dataset and to make the prediction. The plot shows that its comparison of all moistures and the Moisture 1 attributes has the high moisture metrics followed by Moisture 2 then Moisture 0 and finally Moisture 3.

CORRELATION GRADIENT MAP.

DATASET- 1

```
corr = stop.corr()
corr.style.background_gradient(cmap='coolwarm')
```

	day	hour	minute	second	moisture0	moisture1	moisture2	moisture3	moisture4
day	1.000000	-0.216850	-0.016147	-0.072125	-0.924280	-0.611622	-0.878232	-0.891575	-0.025887
hour	-0.216850	1.000000	-0.001180	0.070989	-0.081659	-0.034199	-0.033951	-0.087577	0.018892
minute	-0.016147	-0.001180	1.000000	-0.002409	-0.002609	-0.013826	-0.005886	-0.001073	0.001120
second	-0.072125	0.070989	-0.002409	1.000000	0.093240	0.093260	0.091372	0.120826	-0.003285
moisture0	-0.924280	-0.081659	-0.002609	0.093240	1.000000	0.735426	0.943547	0.913997	0.024469
moisture1	-0.611622	-0.034199	-0.013826	0.093260	0.735426	1.000000	0.783724	0.676025	-0.004386
moisture2	-0.878232	-0.033951	-0.005886	0.091372	0.943547	0.783724	1.000000	0.862147	0.002836
moisture3	-0.891575	-0.087577	-0.001073	0.120826	0.913997	0.676025	0.862147	1.000000	0.035159
moisture4	-0.025887	0.018892	0.001120	-0.003285	0.024469	-0.004386	0.002836	0.035159	1.000000

Figure 30 : CORRELATION GRADIENT MAP FOR DATA ATTRIBUTES OF DATASET-1

DATASET- 2

	year	month	day	hour	minute	second	moisture0	moisture1	moisture2	moisture3	moisture4	irrgation
year	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
month	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
day	nan	nan	1.000000	-0.081286	-0.002565	-0.003688	-0.901576	0.446852	0.688953	0.759488	0.209223	nan
hour	nan	nan	-0.081286	1.000000	0.002132	-0.033278	0.008766	-0.032911	-0.061425	-0.038817	0.038641	nan
minute	nan	nan	-0.002565	0.002132	1.000000	-0.001658	0.000744	0.000057	-0.000239	-0.000235	-0.015091	nan
second	nan	nan	-0.003688	-0.033278	-0.001658	1.000000	0.006799	-0.004114	-0.006818	-0.005488	-0.006834	nan
moisture0	nan	nan	-0.901576	0.008766	0.000744	0.006799	1.000000	-0.223934	-0.546708	-0.600088	-0.060366	nan
moisture1	nan	nan	0.446852	-0.032911	0.000057	-0.004114	-0.223934	1.000000	0.644527	0.274376	0.195880	nan
moisture2	nan	nan	0.688953	-0.061425	-0.000239	-0.006818	-0.546708	0.644527	1.000000	0.648850	0.144897	nan
moisture3	nan	nan	0.759488	-0.038817	-0.000235	-0.005488	-0.600088	0.274376	0.648850	1.000000	0.313640	nan
moisture4	nan	nan	0.209223	0.038641	-0.015091	-0.006834	-0.060366	0.195880	0.144897	0.313640	1.000000	nan
irrgation	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan

Figure 31 : CORRELATION GRADIENT MAP FOR DATA ATTRIBUTES OF DATASET-2

DATASET- 3 – PRIMARY DATASET

```
corr = df.corr()
corr.style.background_gradient(cmap='coolwarm')
```

	day	hour	minute	second	moisture0	moisture1	moisture2	moisture3	moisture4
day	1.000000	-0.089261	-0.015033	0.141041	-0.619495	-0.351188	0.382697	-0.050850	0.069838
hour	-0.089261	1.000000	0.000517	-0.200921	0.076410	0.071533	0.323242	0.129913	-0.146300
minute	-0.015033	0.000517	1.000000	0.002732	-0.004522	-0.001688	-0.005108	-0.025485	-0.014328
second	0.141041	-0.200921	0.002732	1.000000	-0.088478	0.075844	0.119998	0.021293	0.227909
moisture0	-0.619495	0.076410	-0.004522	-0.088478	1.000000	0.015492	0.221128	0.177459	0.071691
moisture1	-0.351188	0.071533	-0.001688	0.075844	0.015492	1.000000	-0.188187	0.068022	0.240432
moisture2	0.382697	0.323242	-0.005108	0.119998	0.221128	-0.188187	1.000000	0.023048	-0.069030
moisture3	-0.050850	0.129913	-0.025485	0.021293	0.177459	0.068022	0.023048	1.000000	0.568807
moisture4	0.069838	-0.146300	-0.014328	0.227909	0.071691	0.240432	-0.069030	0.568807	1.000000

Figure 32 : CORRELATION GRADIENT MAP FOR DATA ATTRIBUTES OF DATASET-3

Correlation gradient map is see to how to gradient color mapping on spefici columns of a dataframe in the dataset.

SEABORN PLOT:

DATASET- 3 – PRIMARY DATASET

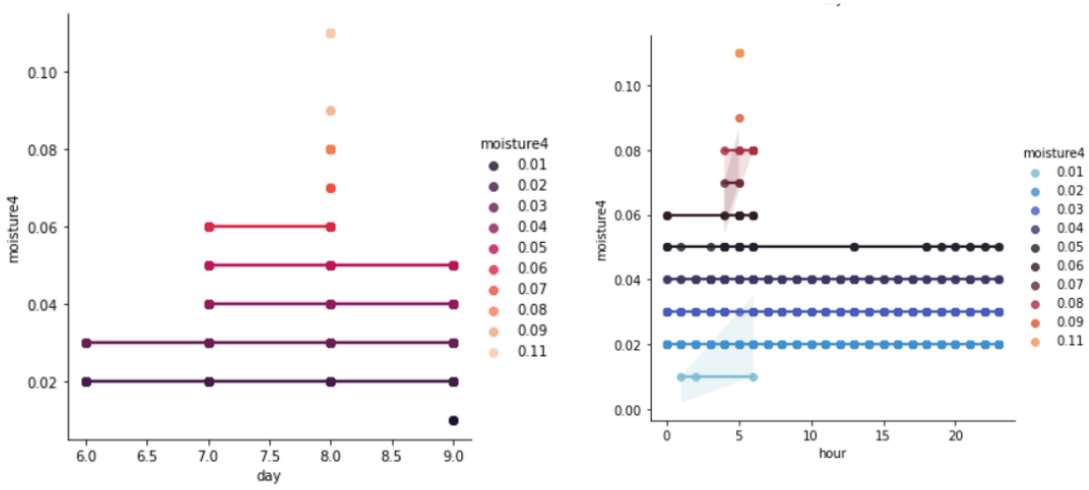

Figure 33 : SEABORN PLOT OF DAY ATTRIBUTES FOR DATASET-3

Figure 34 : SEABORN PLOT OF HOUR ATTRIBUTES FOR DATASET-3

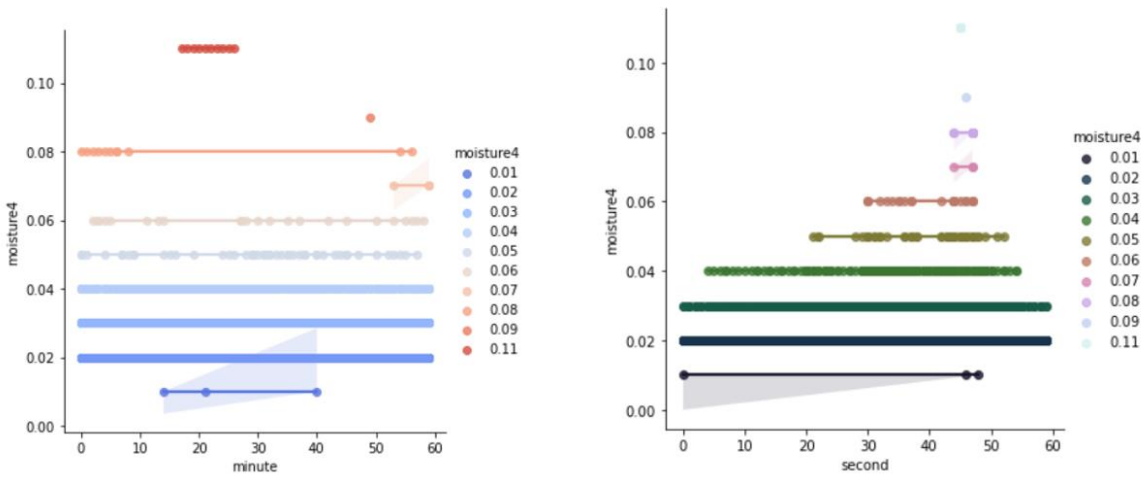

Figure 35 : SEABORN PLOT OF MINUTE ATTRIBUTES FOR DATASET-3

Figure 36 : SEABORN PLOT OF SECOND ATTRIBUTES FOR DATASET-3

Seaborn lmplot for the analysis, By keeping the moisture 4 as constant to compare on the plot for the various y-axis like date, minute, hours,second etc..,

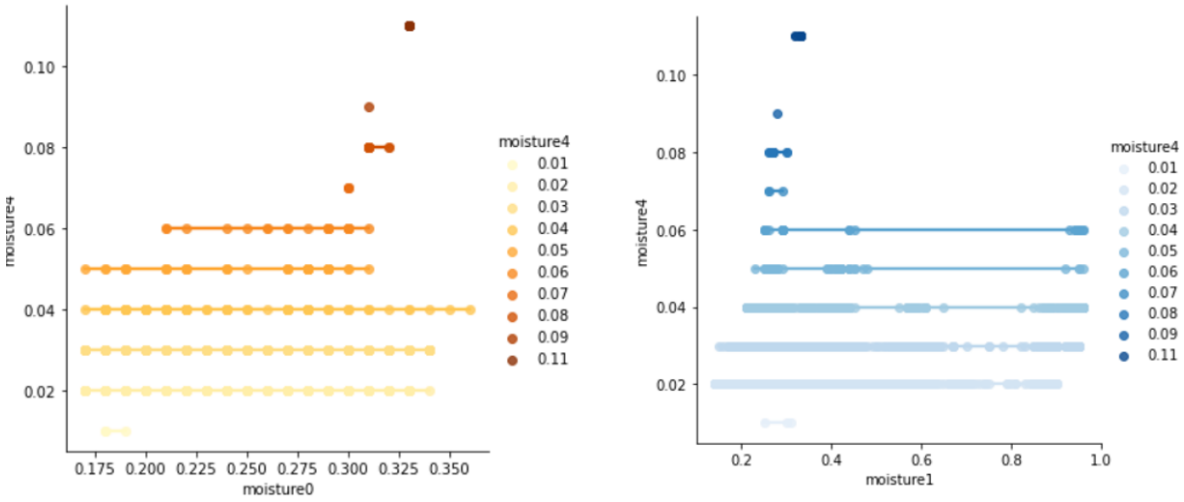

Figure 37 : SEABORN PLOT OF MOISTURE 0 AS ATTRIBUTES FOR DATASET-3

Figure 38 : SEABORN PLOT OF MOISTURE 1 AS ATTRIBUTES FOR DATASET-3

By keeping the moisture 4 as constant to compare on with other moisture values which are collected like Moisture 0, Moisture 1, Moisture 2, Moisture 3 etc..,

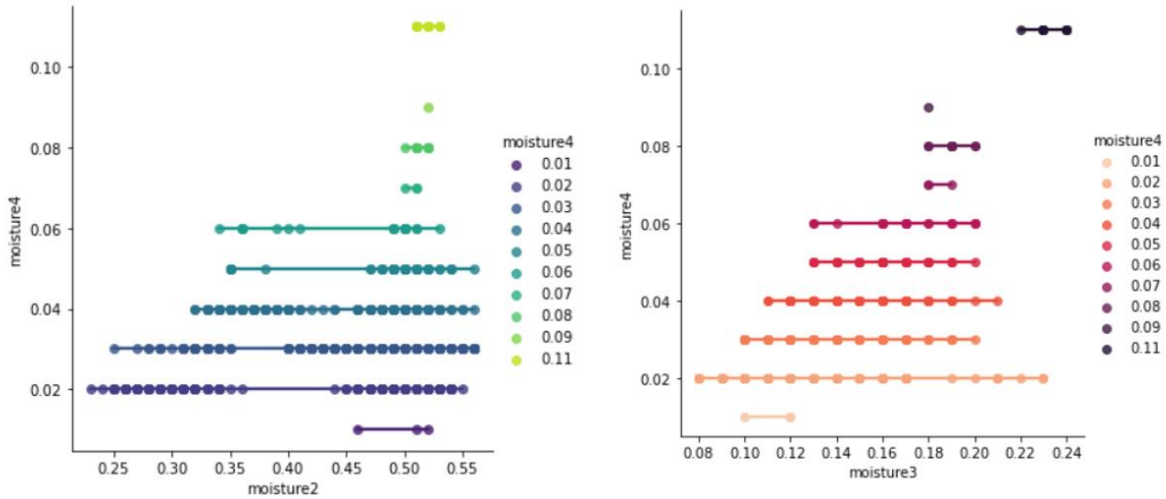

Figure 38 : SEABORN PLOT OF MOISTURE 2 AS ATTRIBUTES FOR DATASET-3

Figure 39 : SEABORN PLOT OF MOISTURE 3 AS ATTRIBUTES FOR DATASET-3

By keeping the moisture 4 as constant to compare on with other moisture values which are collected like Moisture 0, Moisture 1,Moisture2,moisture 3 etc..,

6.5 MODELLING

First algorithm is Linear Regression:

```
: def predict(algorithm):
    model = algorithm.fit(X_train,y_train)
    print('Training Score: {}'.format(model.score(X_train,y_train)))
    print('Test Accuracy: {}'.format(model.score(X_test, y_test)))

    preds = model.predict(X_test)
    print('Predictions are: {}'.format(preds))
    print('\n')

    r2_score = metrics.r2_score(y_test,preds)
    print('r2_score is:{}'.format(r2_score))

    print('MAE:',metrics.mean_absolute_error(y_test,preds))
    print('MSE:',metrics.mean_squared_error(y_test,preds))
    print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,preds)))
    sns.distplot(y_test-preds,color='red')
```

Figure 40 : CODE: PREDICTION ON LINEAR REGRESSION MODEL

```
from sklearn.metrics import accuracy_score as score
```

```
from sklearn.linear_model import LinearRegression
```

Figure 41 : CODE: IMPORTING SCKILIT LEARN PACKAGES FOR MACHINE LEARNING MODELS

DATASET-1

```
predict(LinearRegression())
```

```
Training Score: 0.011281276597404633
Test Accuracy: 0.002788402154185632
Predictions are: [0.02048459 0.02100679 0.02061148 ... 0.02074936 0.02131462 0.02079831]


r2_score is:0.002788402154185632
MAE: 0.001477051083049631
MSE: 7.211835925786223e-06
RMSE: 0.002685486161905554
```

Figure 42 : CODE: PREDICTION VALUE FOR LINEAR REGRESSION OF DATASET-1

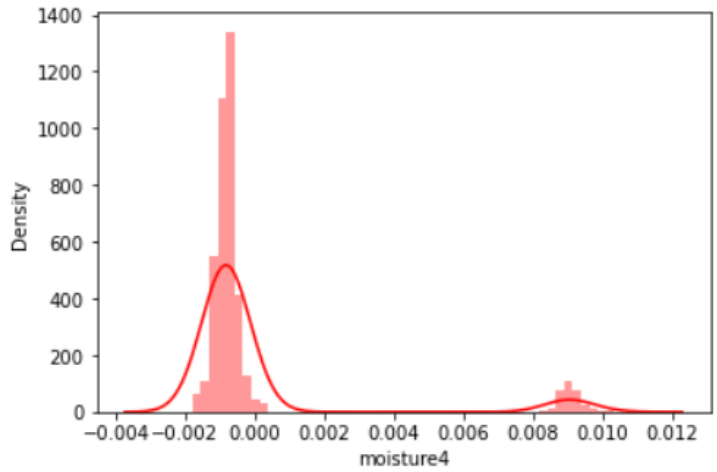

Figure 43 : CODE: DIAGRAMETICAL RESPRENTATION ON PREDICTION VALUE FOR LINEAR REGRESSION OF DATASET-1

DATASET -2

```
predict(LinearRegression())
```

```
Training Score: 0.1799268532108098
Test Accuracy: 0.1914286131772025
Predictions are: [0.0081312  0.00483921 0.00675845 ... 0.0061497  0.00912347 0.01056536]


r2_score is:0.1914286131772025
MAE: 0.0029209458009128264
MSE: 1.4439294462881086e-05
RMSE: 0.003799907165034573
```

Figure 44 : CODE: PREDICTION VALUE FOR LINEAR REGRESSION OF DATASET-2

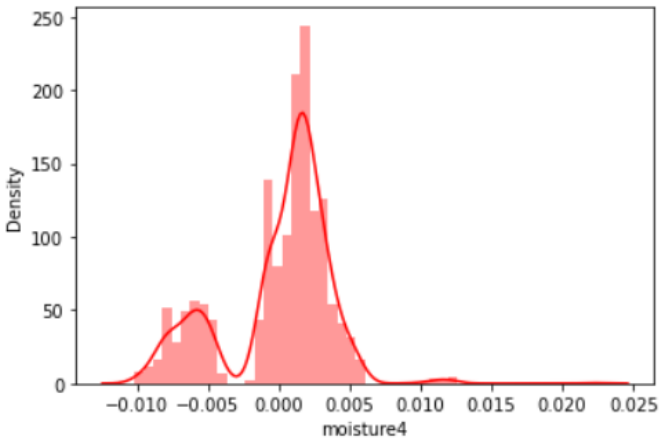

Figure 45 : CODE: DIAGRAMETICAL RESPRENTATION ON PREDICTION VALUE FOR LINEAR REGRESSION OF DATASET-2

DATASET-3

```
from sklearn.metrics import accuracy_score as score
```

```
from sklearn.linear_model import LinearRegression

predict(LinearRegression())
```

```
Training Score: 0.5041848832473712
Test Accuracy: 0.5326493426777246
Predictions are: [0.03098928 0.02595137 0.02978913 ... 0.05211637 0.0283853  0.03645733]


r2_score is:0.5326493426777246
MAE: 0.0043283793066388205
MSE: 3.804635379579165e-05
RMSE: 0.0061681726463995515
```

Figure 46 : CODE: PREDICTION VALUE FOR LINEAR REGRESSION OF DATASET-3

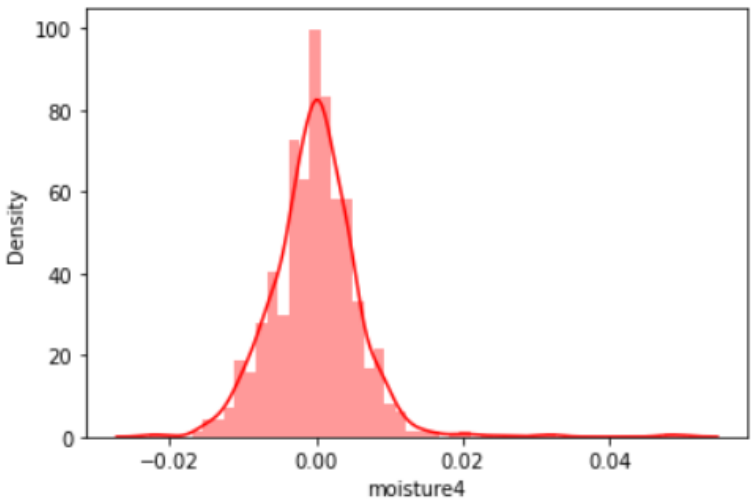

Figure 47 : DIAGRAMETICAL RESPRENTATION ON PREDICTION VALUE FOR LINEAR REGRESSION OF DATASET-3

NOTE: linear regression has train score and test accuracy of 0.5041848832473712 % and 0.5326493426777246 %

Second algorithm is RandomForestRegressor:

.

DATASET-1

```
from sklearn.ensemble import RandomForestRegressor
predict(RandomForestRegressor())
```

```
Training Score: 0.865947116264778
Test Accuracy: 0.08552915109111858
Predictions are: [0.0204 0.0205 0.0206 ... 0.0218 0.0216 0.024 ]


r2_score is:0.08552915109111858
MAE: 0.0011678802588996884
MSE: 6.613454692556629e-06
RMSE: 0.0025716637985080068
```

Figure 48 : CODE: PREDICTION VALUE FOR RANDOM FOREST OF DATASET-1

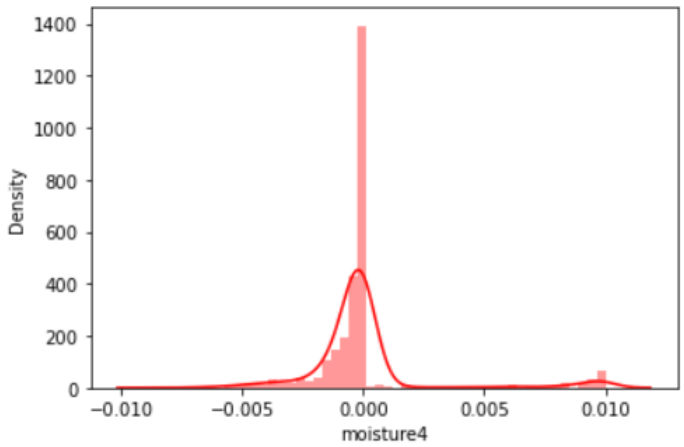

Figure 49 : DIAGRAMETICAL RESPRENTATION ON PREDICTION VALUE FOR RANDOM FOREST OF DATASET-1

DATASET -2

```
from sklearn.ensemble import RandomForestRegressor
predict(RandomForestRegressor())
```

```
Training Score: 0.9049078388860572
Test Accuracy: 0.340331045205874
Predictions are: [0.006  0.0098 0.0084 ... 0.0095 0.0098 0.01  ]


r2_score is:0.340331045205874
MAE: 0.0020980887593132496
MSE: 1.1780226757369616e-05
RMSE: 0.0034322334940049774
```

Figure 50 : CODE: PREDICTION VALUE FOR RANDOM FOREST OF DATASET-2

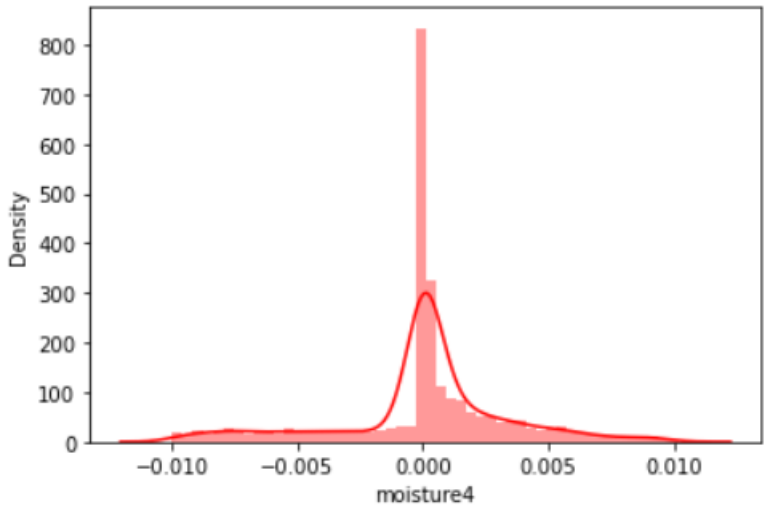

Figure 51 : DIAGRAMETICAL RESPRENTATION ON PREDICTION VALUE FOR RANDOM FOREST OF DATASET-2

DATASET-3

```
from sklearn.ensemble import RandomForestRegressor

predict(RandomForestRegressor())
```

```
Training Score: 0.9819756983340945
Test Accuracy: 0.8903660118687531
Predictions are: [0.0249 0.0202 0.03   ... 0.0528 0.0264 0.04  ]


r2_score is:0.8903660118687531
MAE: 0.0013640967498110515
MSE: 8.92514739229025e-06
RMSE: 0.0029874985175377495
```

Figure 52 : CODE: PREDICTION VALUE FOR RANDOM FOREST OF DATASET-3

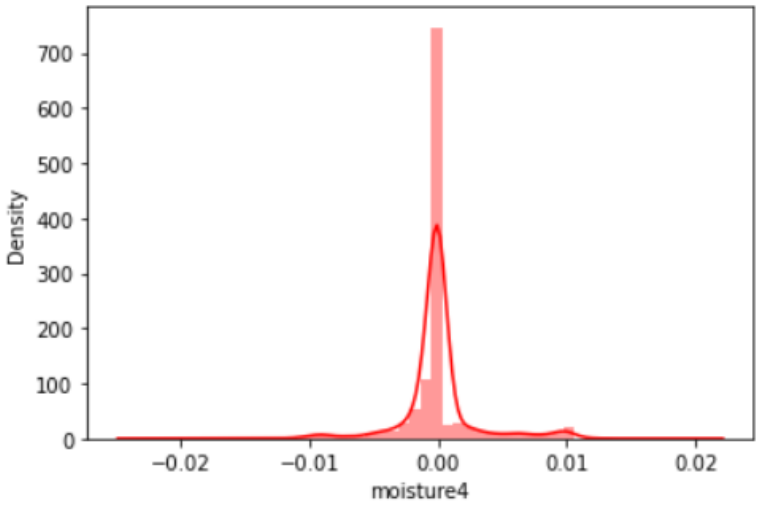

Figure 53: CODE: DIAGRAMETICAL RESPRENTATION ON PREDICTION VALUE FOR RANDOM FOREST OF DATASET-3

NOTE: Random forest has train score and test accuracy of 0.9819756983340945 % and 0.8903660118687531 %

Third algorithm is XG Boost

DATASET-1

```
: from xgboost.sklearn import XGBRegressor

  predict( XGBRegressor())
```

```
Training Score: 0.5397815666750336
Test Accuracy: 0.14051909973409826
Predictions are: [0.0207539  0.02072221 0.02062525 ... 0.02051513 0.02170599 0.02459675]


r2_score is:0.14051909973409826
MAE: 0.0012292668977481062
MSE: 6.215767292974362e-06
RMSE: 0.0024931440578061996
```

Figure 54 : CODE: PREDICTION VALUE FOR XG BOOST REGRESSOR OF DATASET-1

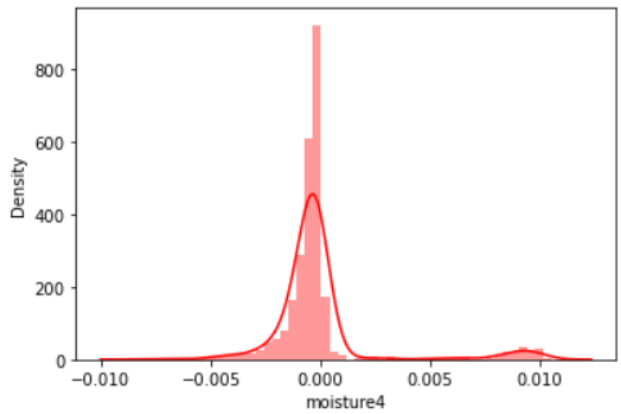

Figure 55: CODE: DIAGRAMETICAL RESPRENTATION ON PREDICTION VALUE FOR XG BOOST OF DATASET-1

DATASET -2

```
from xgboost.sklearn import XGBRegressor

predict( XGBRegressor())
```

```
Training Score: 0.714357284952877
Test Accuracy: 0.3272909589754486
Predictions are: [0.00922653 0.01143963 0.00779632 ... 0.00851323 0.00929431 0.00961161]


r2_score is:0.3272909589754486
MAE: 0.0021950082869981423
MSE: 1.201309382139267e-05
RMSE: 0.003465991030195068
```

Figure 56 : CODE: PREDICTION VALUE FOR XG BOOST REGRESSOR OF DATASET-2

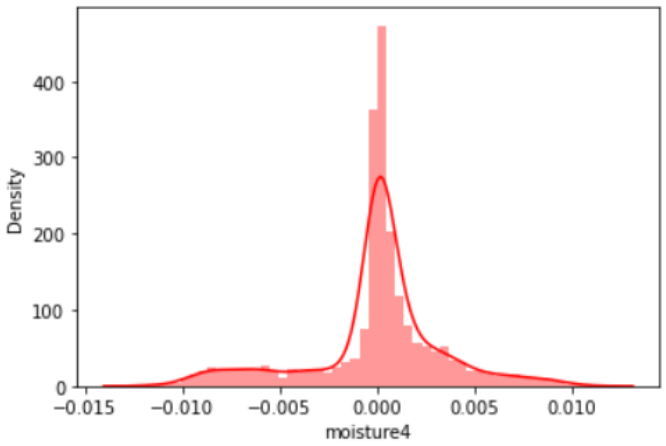

Figure 57: DIAGRAMETICAL RESPRENTATION ON PREDICTION VALUE FOR XG BOOST OF DATASET-2

DATASET-3

```
from xgboost.sklearn import XGBRegressor

predict( XGBRegressor())
```

```
Training Score: 0.958998725106501
Test Accuracy: 0.8894250637732644
Predictions are: [0.02488854 0.0192304  0.03029314 ... 0.06017064 0.02752487 0.04212941]


r2_score is:0.8894250637732644
MAE: 0.0015555962965241544
MSE: 9.00174864144556e-06
RMSE: 0.003000291426086066
```

Figure 58 : CODE: PREDICTION VALUE FOR XG BOOST REGRESSOR OF DATASET-3

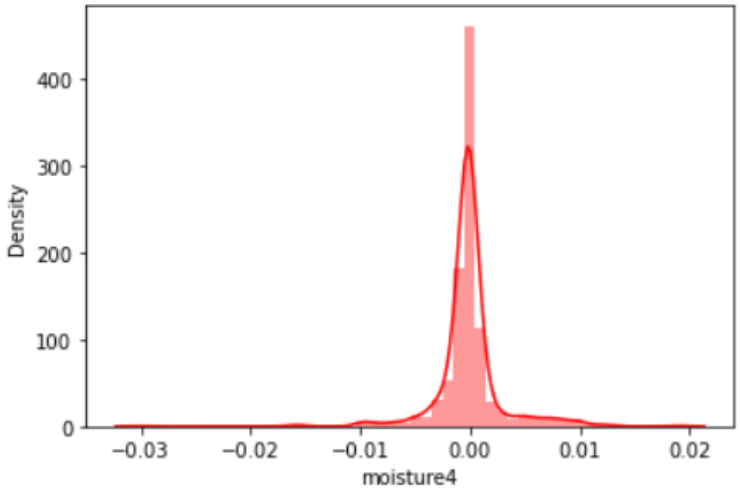

Figure 59: DIAGRAMETICAL RESPRENTATION ON PREDICTION VALUE FOR XG BOOST OF DATASET-3

NOTE: XG BOOST has training score and test accuracy of 0.958998725106501% 0.8894250637732644%

Fourth algorithm is Decision Tree

DATASET-1

```
from sklearn.tree import DecisionTreeRegressor

predict(DecisionTreeRegressor())
```

```
Training Score: 1.0
Test Accuracy: -0.57740104812505
Predictions are: [0.02 0.02 0.02 ... 0.02 0.02 0.02]


r2_score is:-0.57740104812505
MAE: 0.0011407766990291313
MSE: 1.1407766990291253e-05
RMSE: 0.003377538599378437
```

Figure 60 : CODE: PREDICTION VALUE FOR DECISION TREE OF DATASET-1

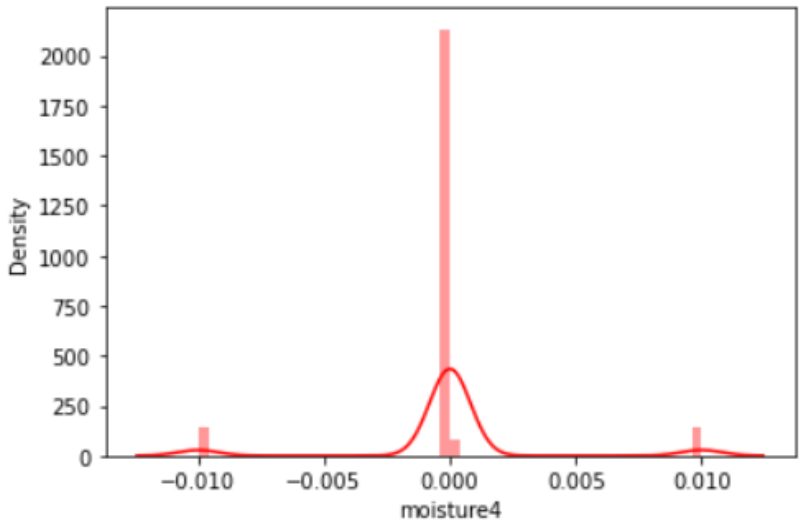

Figure 61: DIAGRAMETICAL RESPRENTATION ON PREDICTION VALUE FOR DECISION TREE OF DATASET-1

DATASET -2

```
from sklearn.tree import DecisionTreeRegressor

predict(DecisionTreeRegressor())
```

```
Training Score: 1.0
Test Accuracy: -0.12649006622516534
Predictions are: [0.   0.01 0.01 ... 0.01 0.01 0.01]


r2_score is:-0.12649006622516534
MAE: 0.0020116618075801777
MSE: 2.011661807580175e-05
RMSE: 0.004485155301191002
```

Figure 62 : CODE: PREDICTION VALUE FOR DECISION TREE OF DATASET-2

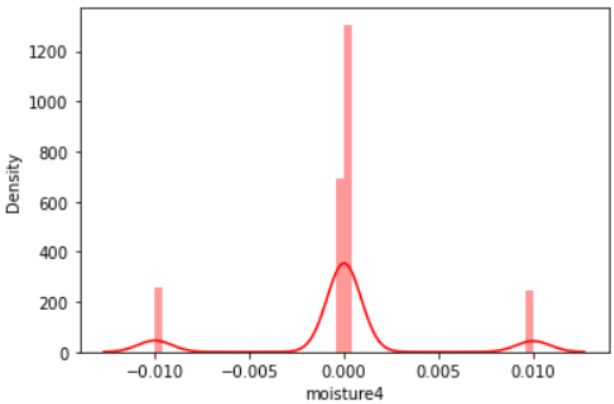

Figure 63 : DIAGRAMETICAL RESPRENTATION ON PREDICTION VALUE FOR DECISION

TREE OF DATASET-2

DATASET-3

```
from sklearn.tree import DecisionTreeRegressor

predict(DecisionTreeRegressor())
```

```
Training Score: 1.0
Test Accuracy: 0.8170905273145543
Predictions are: [0.02 0.02 0.03 ... 0.06 0.03 0.04]


r2_score is:0.8170905273145543
MAE: 0.0012925170068027302
MSE: 1.4890400604686315e-05
RMSE: 0.00385880818448991
```

FIGURE 64: CODE: PREDICTION VALUE FOR DECISION TREE OF DATASET-3

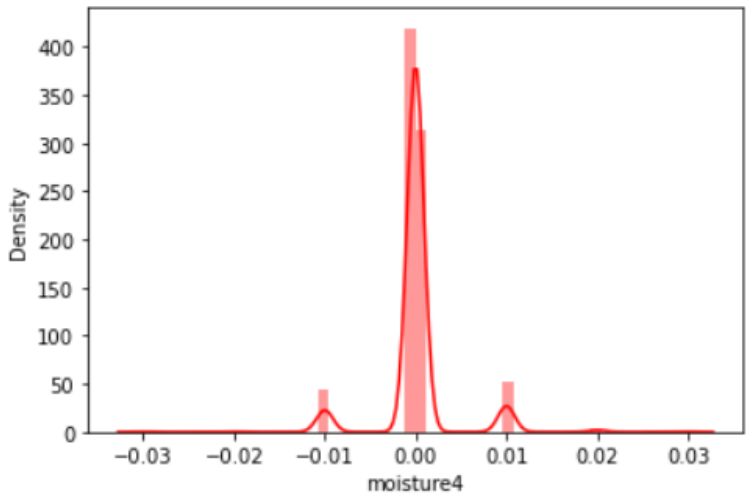

Figure 65 : DIAGRAMETICAL RESPRENTATION ON PREDICTION VALUE FOR DECISION TREE OF DATASET-3

NOTE: DECISiON Tree has training score and test accuracy of 1.0 and 0.8170905273145543 %

Fifth algorithm is K-neighbor

DATASET-1

```
from sklearn.neighbors import KNeighborsRegressor

predict(KNeighborsRegressor())
```

```
Training Score: 0.18146928122792771
Test Accuracy: -0.19792981725695347
Predictions are: [0.02  0.02  0.022 ... 0.022 0.02  0.022]


r2_score is:-0.19792981725695347
MAE: 0.0013576051779935275
MSE: 8.663430420711972e-06
RMSE: 0.002943370588409141
```

Figure 66 : CODE: PREDICTION VALUE FOR K-NEIGHBOR OF DATASET-1

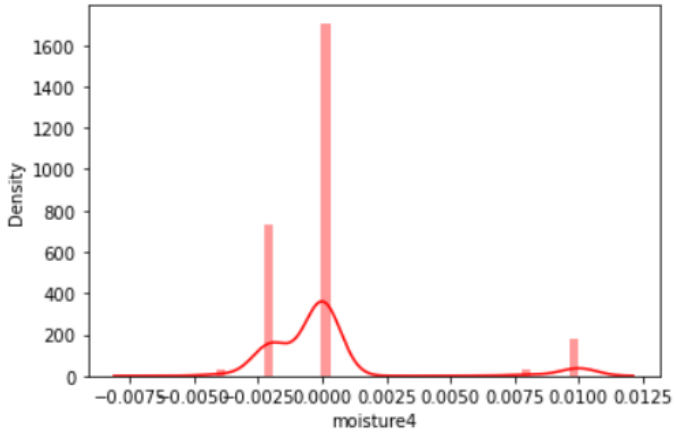

Figure 67: DIAGRAMETICAL RESPRENTATION ON PREDICTION VALUE FOR K-NEIGHBOR OF DATASET-1

DATASET -2

```
from sklearn.neighbors import KNeighborsRegressor

predict(KNeighborsRegressor())
```

```
Training Score: 0.48203501320363396
Test Accuracy: 0.2580765908436512
Predictions are: [0.006 0.01  0.008 ... 0.01  0.01  0.01 ]


r2_score is:0.2580765908436512
MAE: 0.0022073210236475543
MSE: 1.3249109167476514e-05
RMSE: 0.0036399325773256454
```

Figure 68: CODE: PREDICTION VALUE FOR K-NEIGHBOR OF DATASET-2

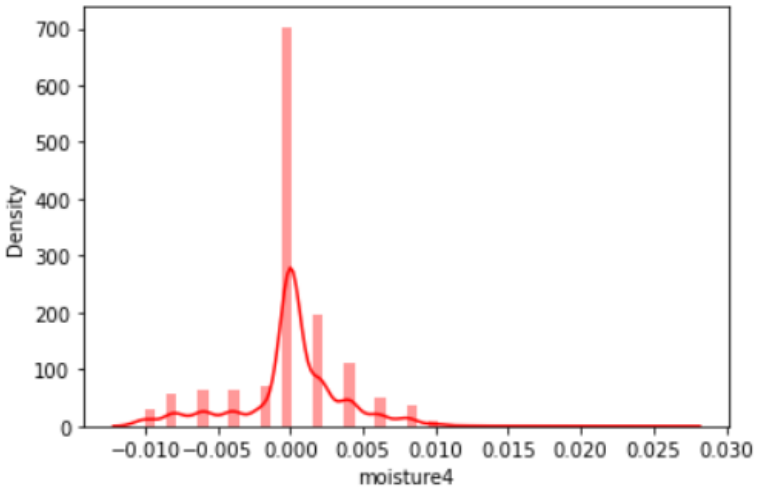

Figure 69: DIAGRAMETICAL RESPRENTATION ON PREDICTION VALUE FOR K-NEIGHBOR OF DATASET-2

DATASET-3

```
from sklearn.neighbors import KNeighborsRegressor

predict(KNeighborsRegressor())
```

```
Training Score: 0.902437059019282
Test Accuracy: 0.8537095608308689
Predictions are: [0.024 0.024 0.03  ... 0.036 0.028 0.04 ]


r2_score is:0.8537095608308689
MAE: 0.0016190476190476193
MSE: 1.1909297052154195e-05
RMSE: 0.0034509849394273218
```

Figure 70 : CODE: PREDICTION VALUE FOR K-NEIGHBOR OF DATASET-3

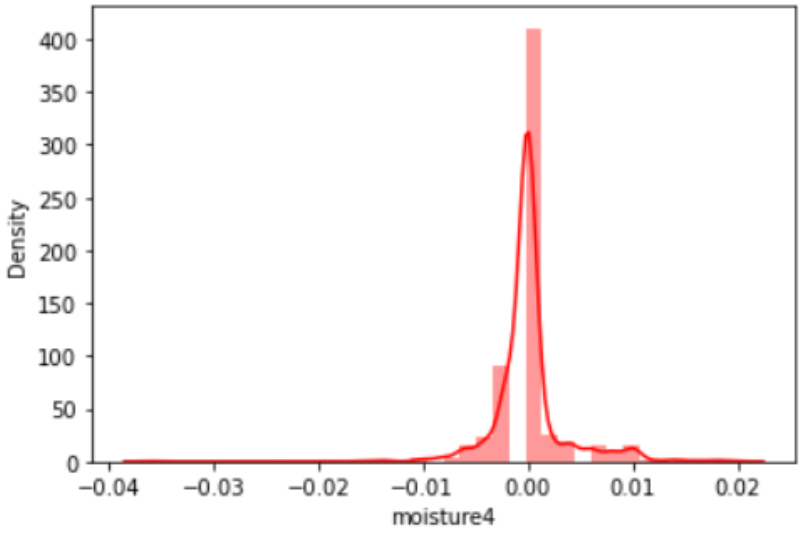

Figure 71: DIAGRAMETICAL RESPRENTATION ON PREDICTION VALUE FOR K-NEIGHBOR OF DATASET-3

NOTE: k-NEIGHBOR Has traing score an test accuracy of 0.902437059019282 and 0.8537095608308689

6.6 DEPLOYMENT

Once you have built and evaluated your model, it is finally time to deploy in the real world. Here I am shown The primary dataset deployment with plotly to understand five algorithm

- Linear Regression

```
ln_model = LinearRegression()
ln_model.fit(X_train, y_train)
preds1 = ln_model.predict(X_test)
preds1
```

```
array([0.03098928, 0.02595137, 0.02978913, ..., 0.05211637, 0.0283853 ,
       0.03645733])
```

```
import plotly.express as px

fig = px.scatter(x=y_test, y=preds1, labels={'x': 'True Moisture', 'y': 'Predicted Moisture'})
fig.add_shape(
    type="line", line=dict(dash='dash'),
    x0=y.min(), y0=y.min(),
    x1=y.max(), y1=y.max()
)
fig.show()
```

Figure 72 : CODE PREDICTED VALUE OFOR LINEAR REGRESSION OF DATASET-3

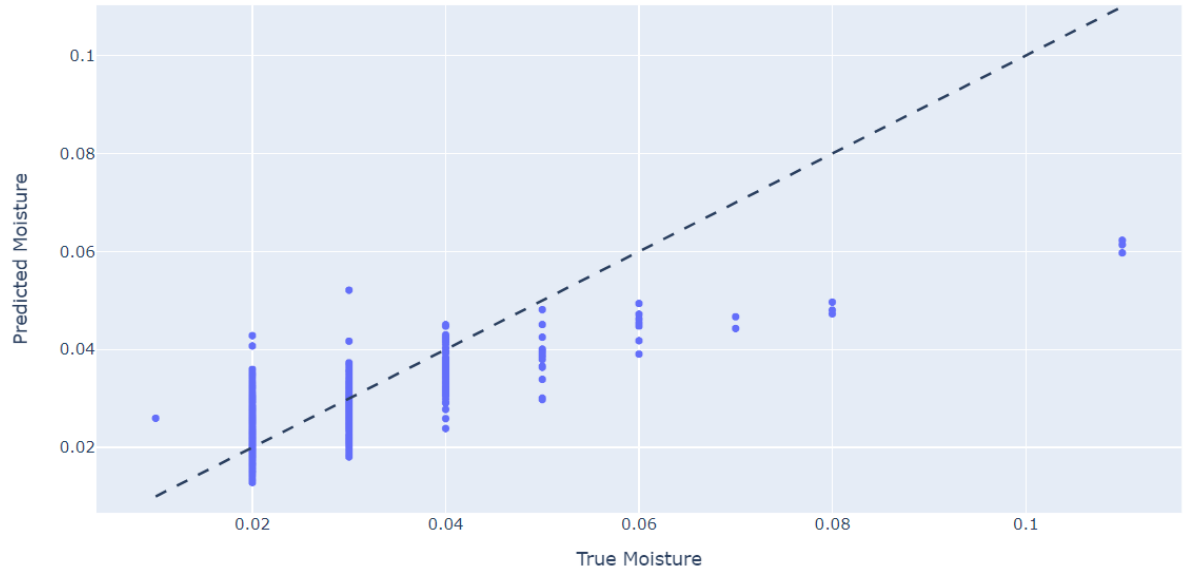

Figure 73: PLOTLY RESPRESENTATION OF LINEAR REGRESSION FOR DATASET-3 ON PREDICTED VALUE

- Random forest

```
rf = RandomForestRegressor()
rf.fit(X_train, y_train)
preds2 = rf.predict(X_test)
preds2
```

```
array([0.0245, 0.02  , 0.03  , ..., 0.0523, 0.0274, 0.04  ])
```

```
import plotly.express as px

fig = px.scatter(x=y_test, y=preds2, labels={'x': 'True Moisture', 'y': 'Predicted Moisture'})
fig.add_shape(
    type="line", line=dict(dash='dash'),
    x0=y.min(), y0=y.min(),
    x1=y.max(), y1=y.max()
)
fig.show()
```

Figure 74: CODE PREDICTED VALUE OF RANDOM FOREST REGRESSION OF DATASET-3

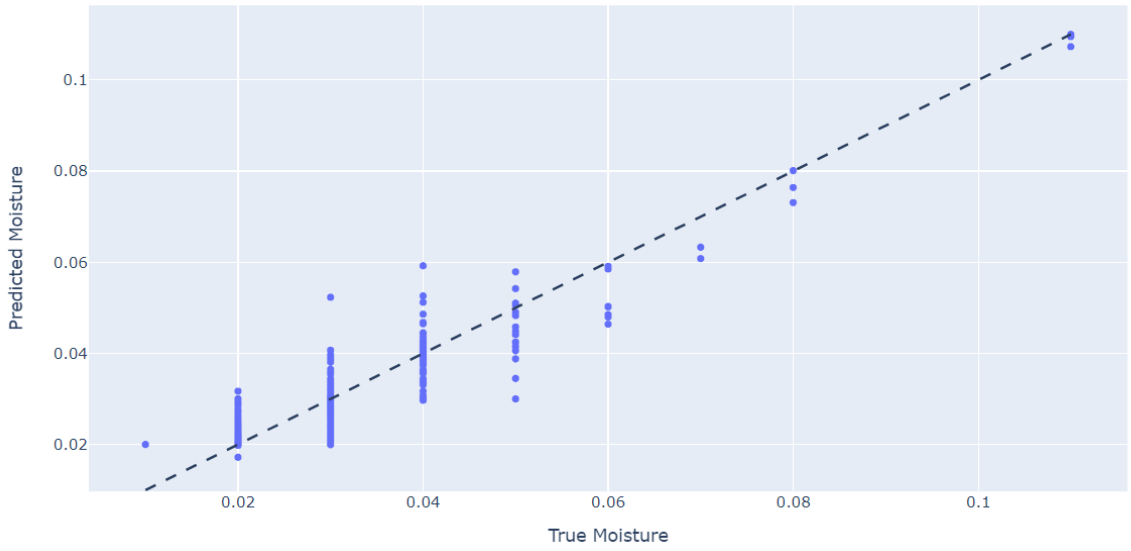

Figure 75 : PLOTLY RESPRESENTATION OF RANDOM FOREST FOR DATASET-3 ON PREDICTED VALUE

- K- Neighbor

```
knn = KNeighborsRegressor()
knn.fit(X_train, y_train)
preds3 = knn.predict(X_test)
preds3
```

```
array([0.024, 0.024, 0.03 , ..., 0.036, 0.028, 0.04 ])
```

```
import plotly.express as px

fig = px.scatter(x=y_test, y=preds3, labels={'x': 'True Moisture', 'y': 'Predicted Moisture'})
fig.add_shape(
    type="line", line=dict(dash='dash'),
    x0=y.min(), y0=y.min(),
    x1=y.max(), y1=y.max()
)
fig.show()
```

Figure 76: CODE PREDICTED VALUE OF K-NEIGHBOR OF DATASET-3

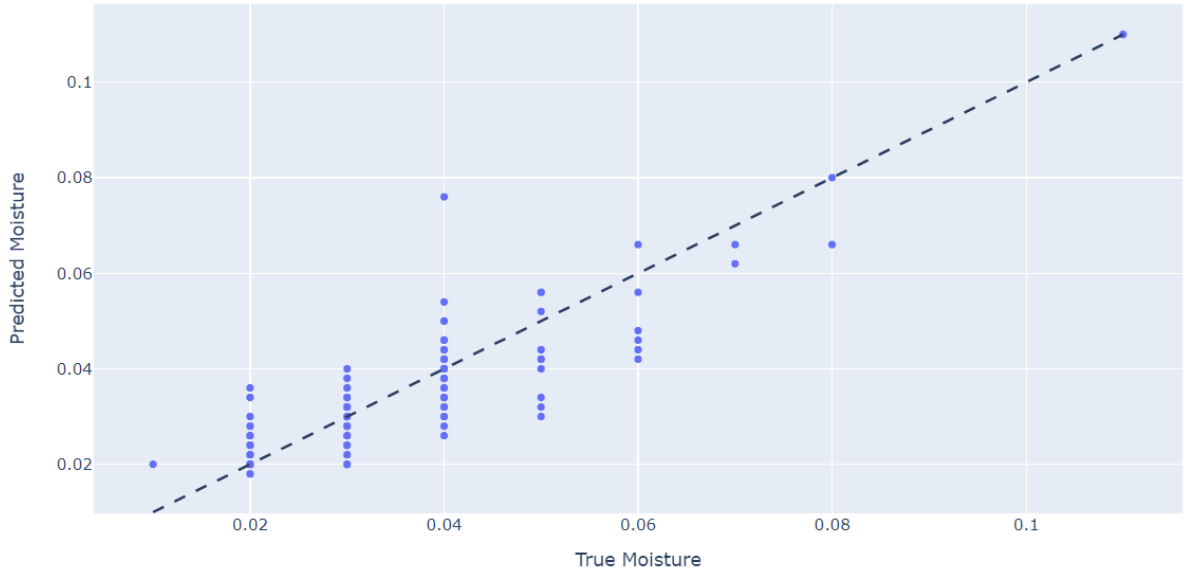

Figure 77 : PLOTLY RESPRESENTATION OF K-NEIGHBOR FOR DATASET-3 ON PREDICTED VALUE

- Extreme Gradient Boost

```
xgb = XGBRegressor()
xgb.fit(X_train, y_train)
preds5 = xgb.predict(X_test)
preds5
```

```
array([0.02488854, 0.0192304 , 0.03029314, ..., 0.06017064, 0.02752487,
       0.04212941], dtype=float32)
```

```
import plotly.express as px

fig = px.scatter(x=y_test, y=preds5, labels={'x': 'True Moisture', 'y': 'Predicted Moisture'})
fig.add_shape(
    type="line", line=dict(dash='dash'),
    x0=y.min(), y0=y.min(),
    x1=y.max(), y1=y.max()
)
fig.show()
```

Figure 78: CODE PREDICTED VALUE OF XG BOOST OF DATASET-3

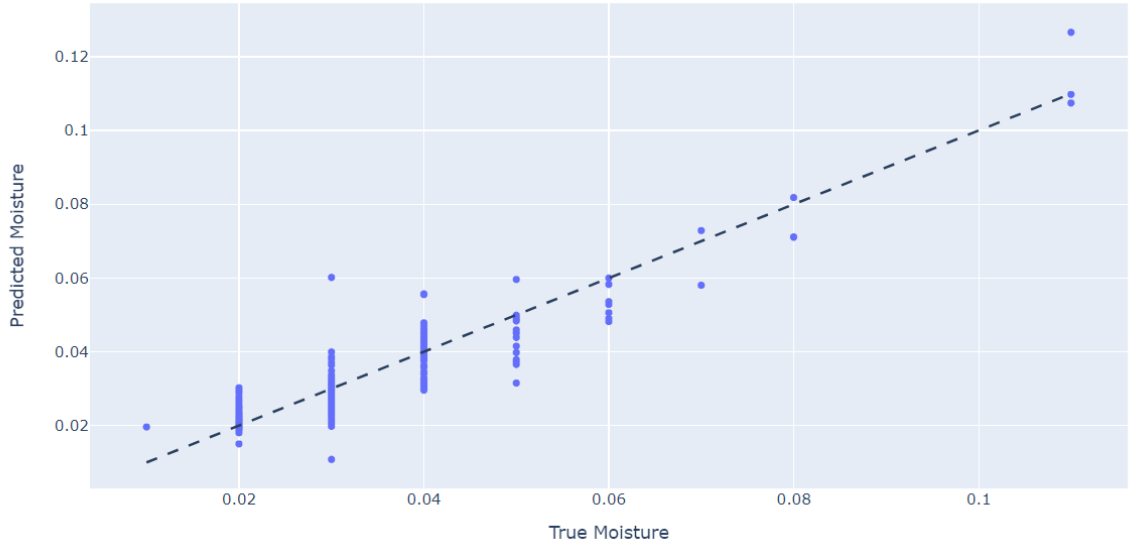

Figure 79 : PLOTLY RESPRESENTATION OF XG BOOST FOR DATASET-3 ON PREDICTED VALUE

```
dt = KNeighborsRegressor()
dt.fit(X_train, y_train)
preds4 = dt.predict(X_test)
preds4
```

```
import plotly.express as px

fig = px.scatter(x=y_test, y=preds4, labels={'x': 'True Moisture', 'y': 'Predicted Moisture'})
fig.add_shape(
    type="line", line=dict(dash='dash'),
    x0=y.min(), y0=y.min(),
    x1=y.max(), y1=y.max()
)
fig.show()
```

FIGURE 80: CODE PREDICTED VALUE OF XG BOOST OF DATASET-3

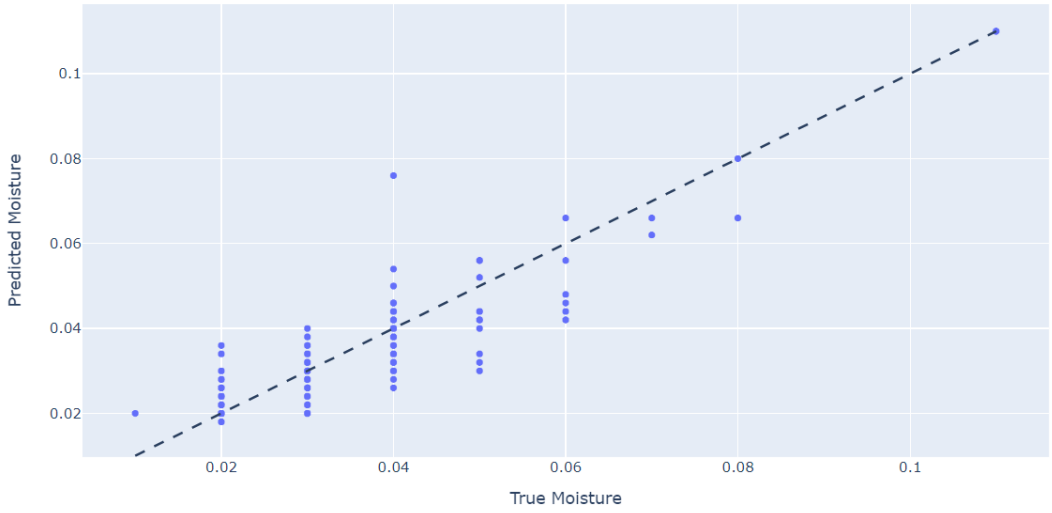

FIGURE 81: PLOTLY RESPRESENTATION OF dECISION TREE FOR DATASET-3 ON PREDICTED VALUE

6.7 CONCLUSION

The chapter detailed out the implementation part of the entire system and its usage using the knowledge that was introduced in the previous chapter for the building of this project. The next chapter focuses on the results of the system.

These prediction values shows that Xg boost and Random forest has best accuracy value.

Chapter-7

RESULTS

7.1 INTRODCTION

The section aims to highlight out the results obtained from the system after its completion explaining the accuracy of the model, inferences and the challenges in it.

7.2 INFERENCE

Hyperparameter Tuning of Random Forest Regressor using RandomisedSearchCV

```
from sklearn.model_selection import RandomizedSearchCV
```

```
param_random = {
'bootstrap': [True],
'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, None],
'max_features': ['auto', 'sqrt'],
'min_samples_leaf': [1, 2, 4],
'min_samples_split': [2, 5, 10],
'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]
}
```

Figure 82: IMPORTING THE RANDOMISED SEARCH CV PACKAGES FOR HYPERPARAMATER

```
param_random
```

```
{'bootstrap': [True],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]}
```

```
rf = RandomForestRegressor()
```

```
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = param_random, n_iter = 100, cv = 5, verbose=2, random_state=
```

```
rf_random.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits

RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_iter=100,
                   n_jobs=-1,
                   param_distributions={'bootstrap': [True],
                                        'max_depth': [10, 20, 30, 40, 50, 60,
                                                      70, 80, 90, 100, None],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 4],
                                        'min_samples_split': [2, 5, 10],
                                        'n_estimators': [200, 400, 600, 800,
                                                         1000, 1200, 1400, 1600,
                                                         1800, 2000]},
                   random_state=42, verbose=2)
```

Figure 83: ASSIGNING THE HYPERPARAMATER FOR PREDCITON VALUES

Executing the randomized search for dataset with random forest as algorithms because it has best prediction value

```
rf_random.best_params_
```

```
{'n_estimators': 400,
 'min_samples_split': 5,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': 70,
 'bootstrap': True}
```

```
preds6 = rf_random.predict(X_test)
```

```
preds6
```

```
array([0.02602431, 0.02058173, 0.0300175 , ..., 0.04496166, 0.02791085,
       0.04027153])
```

Figure 84 : PREDICTION VALUES – INPUT TO CONFUSION MATRIX

Hyperparameter Tuning of XGBoost Classifier using GridSearchCV

```
]: from xgboost import XGBRegressor
```

```
]: def hyperParameterTuning(X_train, y_train):
    param_tuning = {
        'learning_rate': [0.01, 0.1],
        'max_depth': [3, 5, 7, 10],
        'min_child_weight': [1, 3, 5],
        'subsample': [0.5, 0.7],
        'colsample_bytree': [0.5, 0.7],
        'n_estimators' : [100, 200, 500],
        'objective': ['reg:squarederror']
    }
    XGB = XGBRegressor()

    gridsearch = GridSearchCV(estimator = XGB,
                              param_grid = param_tuning,
                              cv = 5,
                              n_jobs = -1,
                              verbose = 1)

    gridsearch.fit(X_train,y_train)

    return gridsearch.best_params
```

Figure 85: IMPORTING THE GRID SEARCH CV PACKAGES FOR HYPERPARAMATER

```
XGB_grid = XGBRegressor(
        objective = 'reg:squarederror',
        colsample_bytree = 0.5,
        learning_rate = 0.05,
        max_depth = 6,
        min_child_weight = 1,
        n_estimators = 1000,
        subsample = 0.7)

XGB_grid.fit(X_train, y_train, early_stopping_rounds=5, eval_set=[(X_test, y_test)], verbose=False)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=0.5, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.05, max_delta_step=0,
             max_depth=6, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=1000, n_jobs=8,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=0.7,
             tree_method='exact', validate_parameters=1, verbosity=None)
```

```
preds8 = XGB_grid.predict(X_test)
```

```
preds8
```

```
array([0.02459604, 0.0198236 , 0.02997729, ..., 0.04995943, 0.02784717,
       0.0398635 ], dtype=float32)
```

Figure 86: ASSIGNING THE HYPERPARAMATER FOR PREDCITON VALUES

7.3 ACCURACY

```
print("Train Accuracy:", rf_random.score(X_train, y_train))
print("Test Accuracy:" , rf_random.score(X_test, y_test))
```

```
Train Accuracy: 0.9562221201267929
Test Accuracy: 0.9031679239436798
```

```
import plotly.express as px

fig = px.scatter(x=y_test, y=preds6, labels={'x': 'True Moisture', 'y': 'Predicted Moisture'})
fig.add_shape(
    type="line", line=dict(dash='dash'),
    x0=y.min(), y0=y.min(),
    x1=y.max(), y1=y.max()
)
fig.show()
```

Figure 87: RESULTS- RANDOMIZED SEACH CV ACCURACY

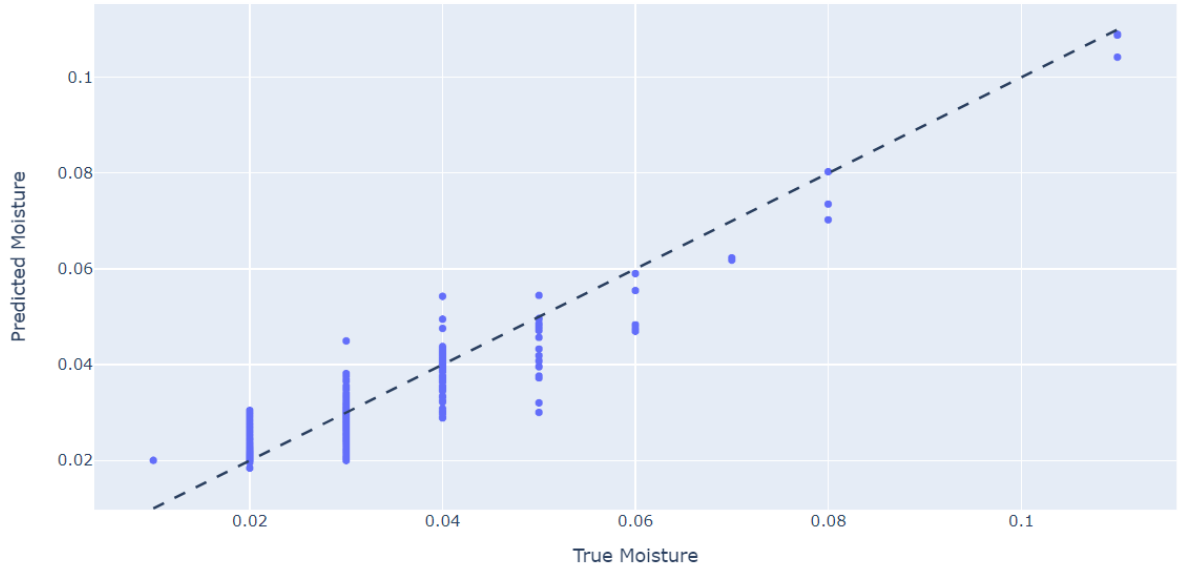

Figure 88: PLOTLY RESPRESENTATION OF XG BOOST FOREST ON RANDOMIZED SEARCH CV

```
print("Train Accuracy:", XGB_grid.score(X_train, y_train))
```

```
Train Accuracy: 0.9450647698352641
```

```
print("Test Accuracy:" , XGB_grid.score(X_test, y_test))
```

```
Test Accuracy: 0.9073673913666416
```

```
import plotly.express as px

fig = px.scatter(x=y_test, y=preds8, labels={'x': 'True Moisture', 'y': 'Predicted Moisture'})
fig.add_shape(
    type="line", line=dict(dash='dash'),
    x0=y.min(), y0=y.min(),
    x1=y.max(), y1=y.max()
)
fig.show()
```

Figure 89: RESULTS- RANDOMIZED SEACH CV ACCURACY

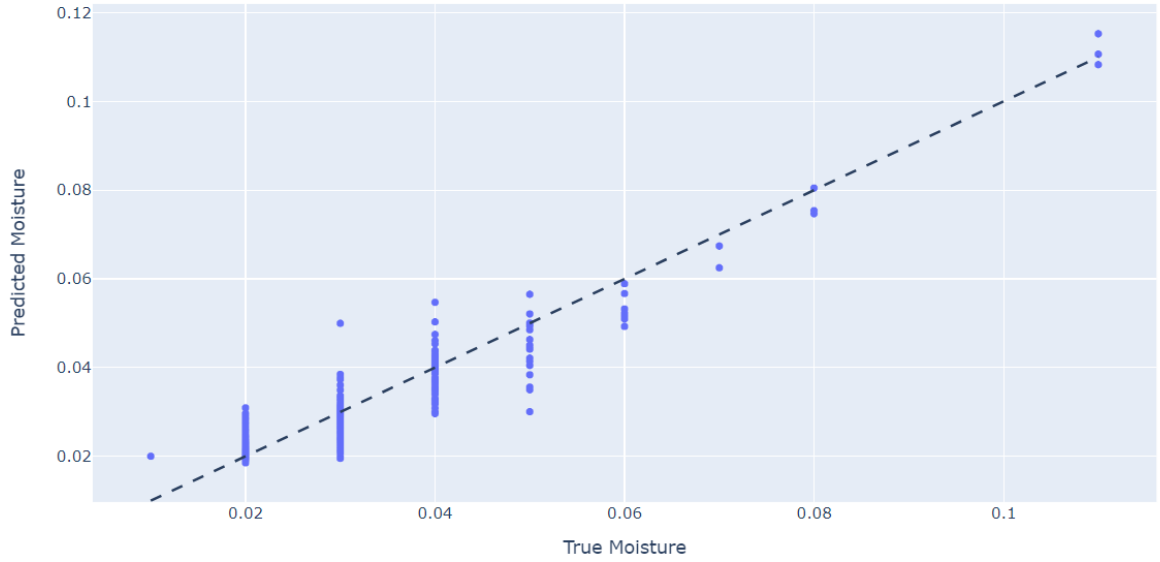

Figure 90: PLOTLY RESPRESENTATION OF XG BOOST ON GRID SEARCH CV

7.4 CHALLENGES

These are some challenges

- To get right threshold
- Soil moisture dataset is very to find when the novelty is in the consideration
- Discarding outliers with the help of class structure
- To check tuning sanity so that pipeline does not become null between frames
- Overfitting, Too Few Hyperparameters,Using the Wrong Metric these are few metrics that make hyperparmater tuning days to train.

7.5 CONCLUSION

Hyperparameter Tuning of XGBoost Classifier using GridSearchCV and Hyperparameter Tuning of Random Forest Regressor using RandomisedSearchCV are highest predicted values of 98% and 94 % accuracy values .

Chapter-8

CONCLUSION

8.1 INTRODUCTION

This chapter will discuss the model's results and outcomes, as well as an appraisal of the proof of theory, future work to develop and expand lane departure warning systems, and data science applications

8.2 ALGORITHMS USED

8.2.1 MACHINE LEARNING ALGORITHMS

Machine learning architecture is applied to informal organization separating, misrepresentation location, picture and discourse acknowledgment, sound acknowledgment, PC vision, clinical picture handling, bioinformatics, client relationship the executives, and a lot more fields. Machine learning models are all over, and the groups equipped for preparing neural systems to convey amazing outcomes areamong the most looked for after experts today.
The algorithms of machine learning are:

- Linear Regression
- Random forest
- K- Neighbor
- Decision Tree
- Extreme Gradient Boost

8.2.2 HYPERPARAMTER TUNING ALGORITHMS

After the prediction on the five algorithms for three datasets (one primary and two secondary datasets) For primary dataset produced the best result with prediction of 98 and 95 accuracy values .To make the prediction further on the algorithms ,the hyperparameter tuning used on two algorithms to get better prediction value with produced results on random forest of 95 % accuracy and

xg boost of 94 %accuracy

8.3 MODEL RESULTS

The main part of project consists soil moisture prediction using machine learning algorithms. We have successfully implemented five best algorithms for the dataset . Using the accuracy score and confusion matrix on the training set, the training model resulted in a 98 percent accuracy in Random forest . On a dataset including both positive and negativesamples of the values, the confusion matrix produces a high true and false positive rate.This demonstrates how well the model distinguishes between soil moisture prediction. This helps for the future soil moisture predictions.

8.4 CHALLENGES

- To get right threshold
- Soil moisture dataset is very to find when the novelty is in the consideration
- Discarding outliers with the help of class structure
- To check tuning sanity so that pipeline does not become null between frames
- Overfitting, Too Few Hyperparameters,Using the Wrong Metric these are few metrics that make hyperparmater tuning days to train.

8.5 FUTURE WORK

I am working on a project where we use machine learning to predict soil moisture. This is complicated because soil moisture can vary significantly over time. The rate of change of moisture varies from year to year, and soil moisture can also change temporarily. So I've been thinking about what would happen if I took the soil moisture prediction model I've built for an annual riverbed, and used it in a riverbed that only goes for a year.

I am using hyperparameter optimization to better understand climate change. I see clouds a little over the Amazon rainforest area, which is part of the Amazon biome. They're cloudy in the dry season, but not in the rainy season. Then I see that rain has a big impact on the rainfall, so I see that the amount of rain during those months is more important than the amount of rain during the rainy season.

The prediction of soil moisture is a good example of future works of soil moisture prediction. This is because the prediction of soil moisture from soil moisture sensors has historical limitations that go back to the 1960s with the first observations of soil moisture from sensors at the measurement sites. The sensor observations were not a good indication of soil moisture because of the absence of a strong relationship between soil moisture and precipitation. Take a look at the legacy of the first sensors and what they can tell you about soil moisture.

I believe the time is now ripe for doing environmental forecasting in a way that will improve our understanding of how climate change is affecting the Earth. I have been inspired by the work of several groups including the National Institute of Standards and Technology, the National Oceanic and Atmospheric Administration, the US Department of Energy, and others, and have written a few papers. I think this is a great time to be working on this problem.

I want to help out with a project that's going to be focused on predicting soil moisture. It’s a relatively simple problem. We’re going to have a dataset of soil moisture measurements and we want to be able to predict which soil moisture level the soil has reached given a set of measurements.

8.6 PERSONAL STATEMENT

During the project I **learned** several new skills in data science, deep learning, machine learning, **OpenCV** and image **processing. My** research **skills have also** improved. Combining multiple computer skills to develop a proof of concept to tackle a **real-world** problem is **inspiring** and **challenging,** but also **leads to** the best **learning outcomes.** I am **sure** that **everyone** who **is into computer science** has a **great** talent for solving puzzles and **difficulties and** making the world a better place.

REFERENCES

[1] Zhang, X., Zhao, W., Liu, Y., Fang, X., & Feng, Q. (2016). The relationships between grasslands and soil moisture on the Loess Plateau of China: A review. *Catena*, *145*, 56-67.

[2] Daly, E., & Porporato, A. (2005). A review of soil moisture dynamics: from rainfall infiltration to ecosystem response. *Environmental engineering science*, *22*(1), 9-24.

[3] Podlipnov, V. V., Shchedrin, V. N., Babichev, A. N., Vasilyev, S. M., & Blank, V. A. (2018). Experimental determi-nation of soil moisture on hyperspectral images. *Computer optics*, *42*(5), 877-884.

[4] Ochsner, E., Cosh, M. H., Cuenca, R., Hagimoto, Y., Kerr, Y. H., Njoku, E. G., & Zreda, M. (2013). State of the art in large-scale soil moisture monitoring. *Soil Science Society of America Journal*, 1-32.

[5] Njoku, E. G., Jackson, T. J., Lakshmi, V., Chan, T. K., & Nghiem, S. V. (2003). Soil moisture retrieval from AMSR-E. *IEEE transactions on Geoscience and remote sensing*, *41*(2), 215-229.

[6] Wagner, W., Hahn, S., Kidd, R., Melzer, T., Bartalis, Z., Hasenauer, S., ... & Rubel, F. (2013). The ASCAT Soil Moisture Product: A Review of its. *Meteorologische Zeitschrift*, *22*(1), 1-29.

[7] Brocca, L., Hasenauer, S., Lacava, T., Melone, F., Moramarco, T., Wagner, W., ... & Bittelli, M. (2011). Soil moisture estimation through ASCAT and AMSR-E sensors: An intercomparison and validation study across Europe. *Remote Sensing of Environment*, *115*(12), 3390-3408.

[8] Cardell-Oliver, R., Kranz, M., Smettem, K., & Mayer, K. (2005). A reactive soil moisture sensor network: Design and field evaluation. *International journal of distributed sensor networks*, *1*(2), 149-162.

[9] Ge, X., Ding, J., Jin, X., Wang, J., Chen, X., Li, X., ... & Xie, B. (2021). Estimating Agricultural Soil Moisture Content through UAV-Based Hyperspectral Images in the Arid Region. *Remote Sensing*, *13*(8), 1562.

[10] Haubrock, S., Chabrillat, S., & Kaufmann, H. (2005, April). Application of hyperspectral imaging for the quantification of surface soil moisture in erosion monitoring and modeling.

In *Proceedings 4th Workshop on Imaging Spectroscopy. New quality in Environmental Studies, Warsaw, Poland* (pp. 27-29).

[11] Heathman, G. C., Starks, P. J., Ahuja, L. R., & Jackson, T. J. (2003). Assimilation of surface soil moisture to estimate profile soil water content. *Journal of Hydrology*, *279*(1-4), 1-17.

[12] Riese, F. M., & Keller, S. (2018, July). Introducing a framework of self-organizing maps for regression of soil moisture with hyperspectral data. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium* (pp. 6151-6154). IEEE.

[13] Kaggle URL https://www.kaggle.com/

Appendice-1

Dataset-2

```
sns.lmplot(x="day", y="moisture4", hue="moisture4",palette="rocket", data=run)
plt.show()
sns.lmplot(x="hour", y="moisture4", hue="moisture4",palette="icefire", data=run)
plt.show()
sns.lmplot(x="minute", y="moisture4", hue="moisture4",palette="coolwarm",data=run)
plt.show()
sns.lmplot(x="hour", y="moisture4", hue="moisture4",palette="cubehelix",data=run)
plt.show()

plt.show()
```

Figure 91 : CODE- SEABORN PLOT ON MOISTURE VS OTHER ATTRIBUTES OF DATASET-2

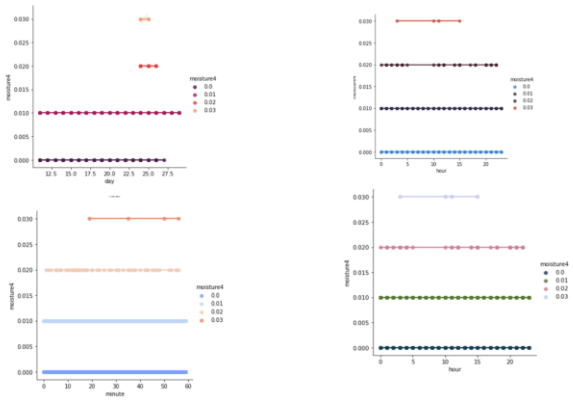

Figure 92 : SEABORN PLOT ON MOISTURE VS OTHER ATTRIBUTES OF DATASET-2

```
sns.lmplot(x="moisture0", y="moisture4", hue="moisture4",palette="YlOrBr", data=run)
plt.show()
sns.lmplot(x="moisture1", y="moisture4", hue="moisture4",palette="Blues", data=run)
plt.show()
sns.lmplot(x="moisture2", y="moisture4", hue="moisture4",palette="viridis", data=run)
plt.show()
sns.lmplot(x="moisture3", y="moisture4", hue="moisture4",palette="rocket_r", data=run)
plt.show()

plt.show()
```

Figure 93: CODE -SEABORN PLOT ON MOISTURE VS OTHER MOISTURE ATTRIBUTES OF DATASET-2

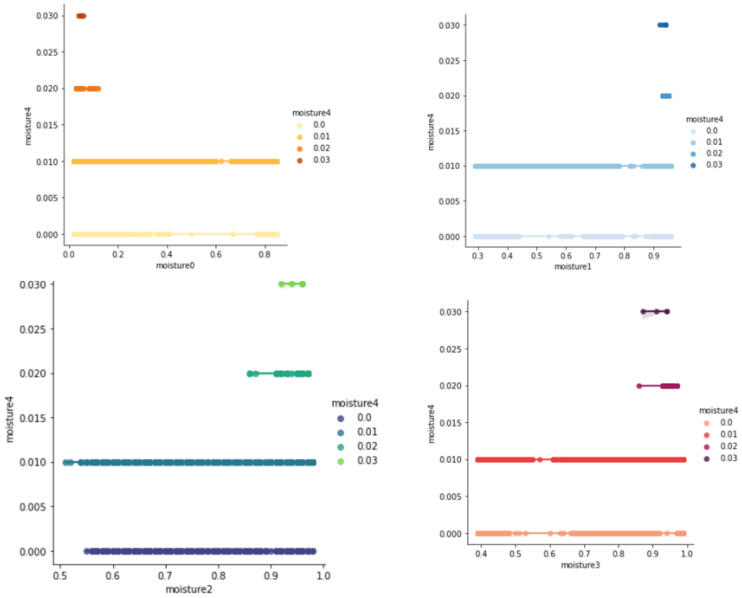

Figure 94: SEABORN PLOT ON MOISTURE VS OTHER MOISTURES VALUES OF DATASET-2

Appendice-2

DATASET-1

```
sns.lmplot(x="day", y="moisture4", hue="moisture4",palette="rocket", data=stop)
plt.show()
sns.lmplot(x="hour", y="moisture4", hue="moisture4",palette="icefire", data=stop)
plt.show()
sns.lmplot(x="minute", y="moisture4", hue="moisture4",palette="coolwarm",data=stop)
plt.show()
sns.lmplot(x="hour", y="moisture4", hue="moisture4",palette="cubehelix",data=stop)
plt.show()

plt.show()
```

FIGURE 95: CODE SEABORN PLOT ON MOISTURE VS OTHER MOISTURES VALUES OF DATASET-1

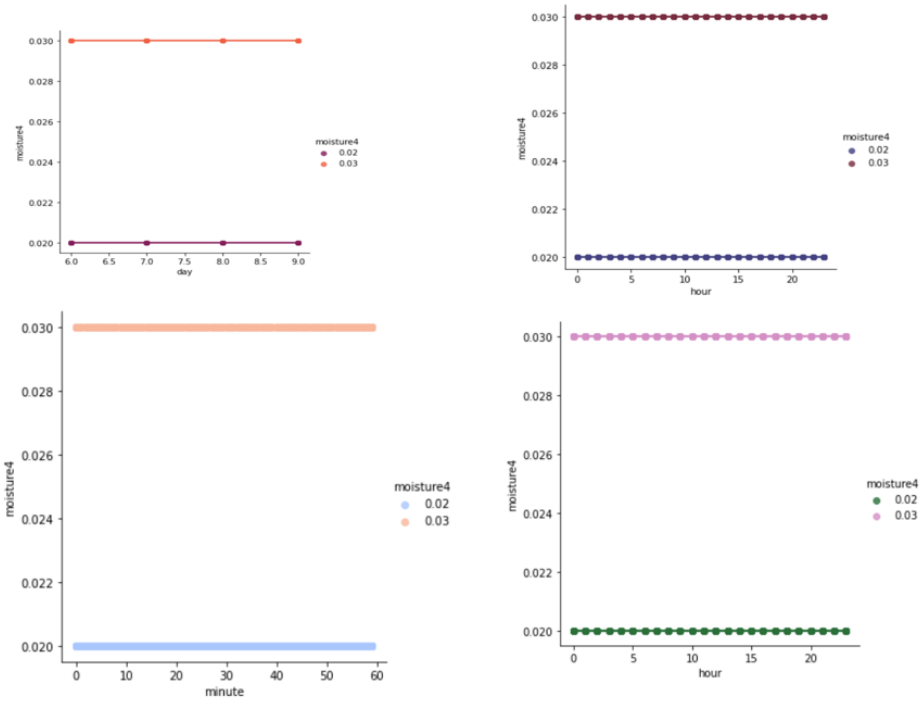

Figure 96: SEABORN PLOT ON MOISTURE VS OTHER ATTRIBUTES OF DATASET-1

```
sns.lmplot(x="moisture0", y="moisture4", hue="moisture4",palette="YlOrBr", data=stop)
plt.show()
sns.lmplot(x="moisture1", y="moisture4", hue="moisture4",palette="Blues", data=stop)
plt.show()
sns.lmplot(x="moisture2", y="moisture4", hue="moisture4",palette="viridis", data=stop)
plt.show()
sns.lmplot(x="moisture3", y="moisture4", hue="moisture4",palette="rocket_r", data=stop)
plt.show()

plt.show()
```

Figure 97 : CODE- SEABORN PLOT ON MOISTURE VS OTHER MOISTURE ATTRIBUTES OF DATASET-1

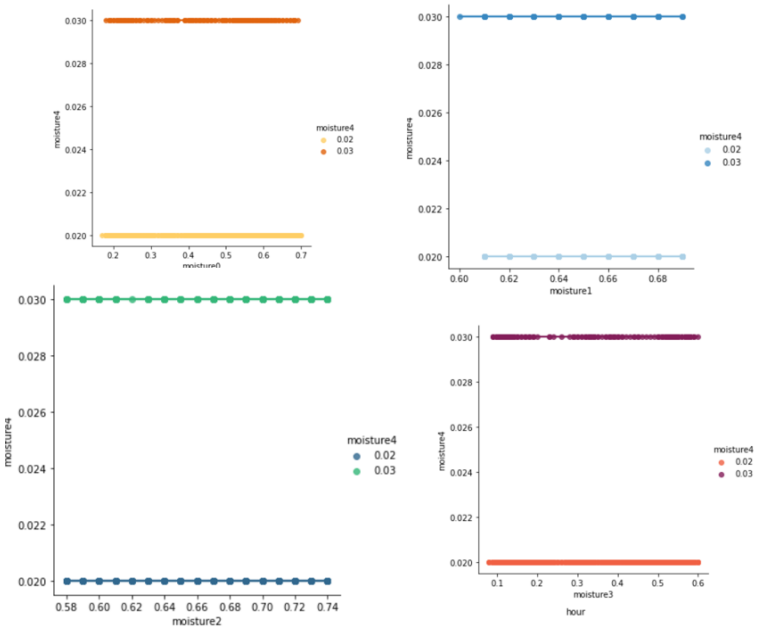

Figure 98 : SEABORN PLOT ON MOISTURE VS OTHER MOISTURE OF DATASET-1