A PROJECT REPORT ON

"CROP PREDICTION FROM SOIL ANALYSIS USING MACHINE LEARNING"

BY

CHETAN SANJAY DHUMNE Roll No: 1313
PANKAJ KAMBIRE Roll No: 1333

Under The Guidance of

Mr. AKSHAY TILEKAR

POST GRADUATE DIPLOMA IN BIG DATA ANALYTICS FEBUARY-2020

INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT, AKURDI, PUNE

CERTIFICATE

This is to certify that the Project Entitled

"CROP PREDICTION FROM SOIL ANALYSIS USING MACHINE LEARNING"

Submitted by

CHETAN SANJAY DHUMNE	Roll No: 1313
PANKAJ KAMBIRE	Roll No: 1333

is a bonafide student of this institute and the work has been carried out by him/her under the supervision of Mr.Akshay Tilekar and it is approved for the partial fulfillment of the requirement of Post Graduate Diploma in Big Data Analytics.

Mr. AKSHAY TILEKAR Project Guide PG-DBDA	Mr. PRASHANT KARHALE Centre Coordinator IACSD, AKURDI

ACKNOWLEDGEMENT

It gives us great pleasure in presenting the preliminary project report on

"**CROP PREDICTION FROM SOIL ANALYSIS USING MACHINE LEARNING"**

I would like to take this opportunity to thank my internal guide Mr. AKSHAY TILEKAR for giving me all the help and guidance I needed. I am really grateful to them for their kind support. Their valuable suggestions were very helpful.

I am also grateful to Mr. PRASHANT KARHALE, Centre Coordinator, Akurdi, Pune for his indispensable support, suggestions.

In the end our special thanks to IACSD for providing various resources such as laboratory with all needed software platforms, continuous Internet connection, for our Project.

Chetan Dhumne

Pankaj Kambire

ABSTRACT

Soil's physical and chemical properties plays a dominant role to assess the soil quality and soil fertility. The fertilizers like Nitrogen, Phosphorous, Potassium and micro-nutrients of the soil are incumbent for the development and productiveness of the soil and plants.

Soil is most important item of farming. The soil quality is most important feature of crop productions. If the soil quality is not good we losses some quality in crops. In this project, the use of distinguish features and characteristics of soil to improve the accuracy of crop prediction. The soil dataset is of 23 different districts and talukas of MAHARASHTRA state from which soil nutrients values like N, P, K, and other micro-nutrients are used to form a column of Crop on which the predictions are done. Other than Crop column there are two columns with indicates soil quality (Fertile or non-fertile) and soil type (Neutral, Acidic or Alkaline).This columns are formed using pH values and macro-micro nutritional values of soil. In this project, stable climate with adequate humidity and average rainfall conditions are taken into consideration.

Machine learning algorithms are useful to achieve the better accuracy in crop prediction and assessment. In this project, KNN Classifier, Decision Tree Classifier, Random Forest, Support Vector Machine, XGBoost, and Voting Classifier are used to predict crops. On the basis of accuracy, confusion matrix and f1-score the best training model is deicide which is Random Forest Classifier, who gives 92% of accuracy on test data.

Table of Contents

TABLE OF FIGURE

LIST OF TABLES

CHAPTER 1
SYNOPSIS

1.1 Project Title
"CROP PREDICTION FROM SOIL ANALYSIS USING MACHINE LEARNING"

1.2 Project Option
Internal Project

1.3 Internal Guide
Mr. AKSHAY TILEKAR

1.4 Sponsorship and External Guide
NA

1.5 Problem Statement
In this project we study and analyse the soil nutrition values using the ML classification technique and build the model for giving the best crop for productions.

1.6 Abstract
Soil's physical and chemical properties plays a dominant role to assess the soil quality and soil fertility. The fertilizers like Nitrogen, Phosphorous, Potassium and micro-nutrients of the soil are incumbent for the development and productiveness of the soil and plants.

Soil is most important item of farming. The soil quality is most important feature of crop productions. If the soil quality is not good we losses some quality in crops. In this project, the use of distinguish features and characteristics of soil to improve the accuracy of crop prediction. The soil dataset is of 23 different districts and talukas of MAHARASHTRA state from which soil nutrients values like N, P, K, and other micro-nutrients are used to form a column of Crop on which the predictions are done. Other than Crop column there are two columns with indicates soil quality (Fertile or non-fertile) and soil type (Neutral, Acidic or Alkaline).This columns are formed using pH values and macro-micro nutritional values of soil. In this project, stable climate with adequate humidity and average rainfall conditions are taken into consideration.

Machine learning algorithms are useful to achieve the better accuracy in crop prediction and assessment. In this project, KNN Classifier, Decision Tree Classifier, Random Forest, Support Vector Machine, XGBoost, and Voting Classifier are used to predict crops. On the basis of accuracy, confusion matrix and f1-score the best training model is deicide which is Random Forest Classifier, who gives 92% of accuracy on test data.

1.7 Goals and Objectives

1. To predict the best fitted Crop.
2. Soil quality analysis.

CHAPTER 2
INTRODUCTION

India is the rich country having large number of natural and human resources and almost 70% economy of India is based upon agriculture sector. There is numerous ingredient of agriculture sector. Soil is one of the important factors among them. Fit and healthy Soils provide healthy yields. Naturally, soils contain many prime nutrients.

Nitrogen, Phosphorous, potassium (NPK) and Calcium are of key important fertilizers.
All these nutrients are crucial for growing and developing the plants. When any soil nutrients and parameters are missing from the soil or in under or over supply with respect to required ratio of the nutrients; plants and crops suffer from nutrient deficiency and superfluous respectively and hence they can stop growing. Then, necessity of fertilizers to soil is essential to provide stable nutrients to the plants grown on it.

In this project, using the dataset of soil test results at the "**Soil Health Card**" program started by **Department of Agriculture, Cooperation & Farmers Welfare Ministry of Agriculture and Farmers Welfare Government of India,** the data set has contain the soil testing parameters like the pH level of soil, nitrogen, potassium, phosphorus, calcium, magnesium, zinc, boron, sulphur, iron, copper and electric conductivity. The analysis of this content of soils and aim to give the best crop for suitable for the soil.

Soil quality is the capability to define precise type of soil to perform functions within its managed ecosystem, capacity and natural boundaries. Soil quality is useful to sustain living organism productivity, maintaining or enhancing the quality of water and air and it also support human fitness and habitation. Plants growth are based on soil organic matter, the physicochemical properties of soil such as texture of the soil, pH values, water holding capacity and availability of the nutrients in the soil.

2.1 Purpose

Motivation behind this project is to create a ML model for predicting crop using soil nutrition values. This model will help farmers to utilize the overall soil nutrients to get better crop production and minimize the use of fertilizers.

2.2 Scope of the project

2.2.1 Initial functional requirement will be:-

- Selecting the algorithm meeting requirement.
- Choosing the optimum algorithm form set of algorithm.
- Testing it on the datasets.
- After getting the result if the result is low change the hyper parameters.
- Out of all result get best of all

Terms	Definitions
Dataset	Data for training and testing for the model.
Variance	Difference between the training and testing accuracy.
Bias	Both learning and training accuracy is low.
Overfitting	Model is very complex
Under fitting	Model is bias
Developer	Who is developing the model
Review	A written recommendation about the appropriateness of a Product for selling and buying may include suggestions for improvement.
Reviewer	A person that examines a Product and hasthe ability to recommend approval Product for buying or to request that changes be made in the Product.

Software Requirement Specification	A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document
User	Reviewer

Table 2.1: Non-Functional Requirements

2.1.1 Initial non-functional requirement will be:-

- Getting the large datasets which can provide developer enough imageto train the model.
- Maintain the minimum variance bias so the model is successfully work.
- Avoid the under fitting and overfitting.

2.3 Software Development Life Cycle Model

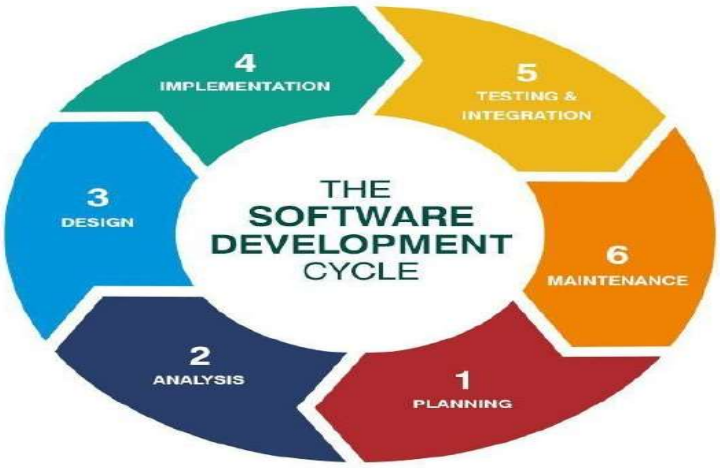

Figure 2.1: Software Development Life Cycle Model

In order to make this Project we are going to use Classic LIFE CYCLE MODEL .Classic life cycle model is also known as WATERFALL MODEL. The life cycle model demands a Systematic sequential approach to software development that begins at the system level and progress through analysis design coding, testing and maintenance.

2.3.1 The Classic Life Cycle Model

The waterfall model is sequential software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of conception initiation, Analysis, Design (validation), construction.Testing and maintained.

1. **System Engineering and Analysis:**

   Because software is always a part of larger system work. Begins byestablishing requirement for all system elements and Then allocating some subset of these requirement to the software system Engineeringand analysis encompasses the requirement gathering at the system levelwith a small amount of top level design and analysis.

2. **Software requirement Analysis:**

   The requirement gathering process is intensified and focused specificallyon the software Requirement for the both system and software are discussed and reviewed with the customer. The customer specifies the entire requirement or the software and the function to be performed by the software.

3. **Design:**

   Software design is actually a multi-step process that focuses on distinct attributes of the program data structure, software architecture, procedural detail and interface of the software that can be assessed or qualitybefore coding begins .Like requirement the design is documented andbecomes part of the software.

4. **Coding:**

   The design must be translated into a machine readable form. The coding step performs this task. If design is programmed in a detailed manner, coding can be accomplished mechanically.

5. **Testing:**

   Once code has been generated programmed testing begins. The testing process focuses on the internals of the software ensuring that all statement have been tested and on the functional externals hat is con- ducting tests to uncover the errors and ensure that defined input will produce the results that agree with the required results.

5.1 Unit testing:

In computer programming, Unit testing is software Verification and validation method where the programmer gains confidence that individual units of source code are fit to use a unit is the smallest testable part of an application. In procedural programming a unit may bean individual programmed, function, procedure, etc. while in object- oriented programming, the smallest Unit is a class, which may belongto a base/super class abstract class or derived/child class.

5.2 Benefits:

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict writ- ten contract that the piece of code must satisfy.

5.3 Documentation:

Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit and how to use it can look at the tests to gain a basic understanding of theunit API

5.4 Limitation of unit testing: Testing cannot be expected to catch error in the program –It is impossible to evaluate all execution paths for all but the most trivial programs. The same is true for unit testing. Additionally, by unit testing only types the functionality if the units themselves.

6. Maintenance:

Software will undoubtedly undergo change after it is Delivered to the customer .Change will occur because errors have been encountered be- cause the software must be able adopted to accommodate changes in its external environment because the customer requires functional or performance enhancement enhancements. The classic life cycle is the oldest and most widely used paradigm or software engineering.

2.4 CRISP-DM Method Life Cycle

CRISP-DM methodology that stands for Cross Industry Standard Process for Data Mining, is a cycle that describes commonly used approaches thatdata mining experts use to tackle problems in traditional BI data mining. Itis still being used in traditional BI data mining teams. Take a look at the following illustration. It shows the major stages of the cycle as described by the CRISP-DM methodology and how they are interrelated.

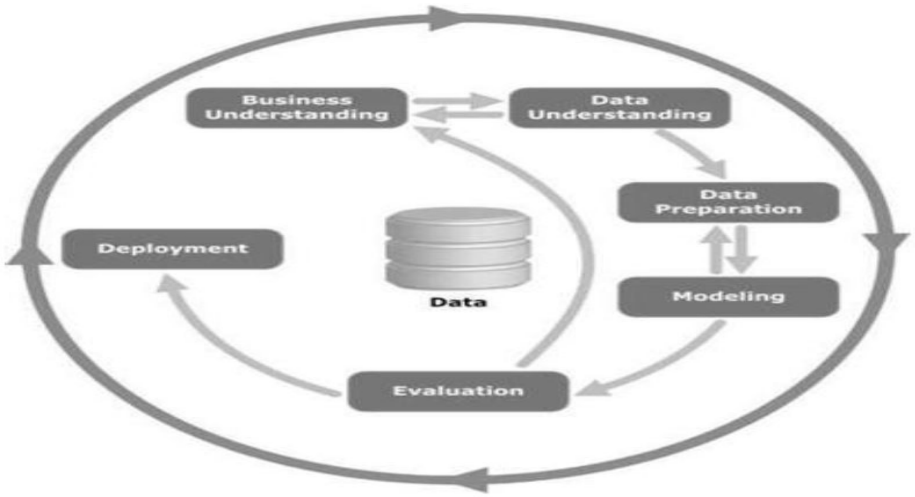

Figure 2.2: Life cycle of Data Mining

Let us now learn a little more on each of the stages involved in the CRISP-DM life cycle-

- **Business Understanding-** This initial phase focuses on understanding the project objectives and requirements from a business perspective, and then converting this knowledge into a data mining problem definition. A preliminary plan is designed to achieve the objectives.A decision model, especially one built using the Decision Model and Notation standard can be used.

- **Data Understanding-** The data understanding phase starts with an initial data collection and proceeds with activities in order to get familiar with the data, to identify data quality problems, to discover first insights into the data, or to detect interesting subsets to form hypotheses for hidden information.

- **Data Preparation-** The data preparation phase covers all activities to construct the final dataset (data that will be fed into the modeling tool(s)) from the initial raw data. Data preparation tasks are likely to be performed multiple times, and not in any prescribed order. Tasks include table, record, and attribute selection as well as transformation and cleaning of data for modeling tools.

- **Modeling-** In this phase, various modeling techniques are selected and applied and their parameters are calibrated to optimal values. Typically, there are several techniques for the same data mining problemtype. Some techniques have specific requirements on the form of data. Therefore, it is often required to step back to the data preparation phase.

- **Evaluation-** At this stage in the project, you have built a model (or models) that appears to have high quality, from a data analysis perspective. Before proceeding to final deployment of the model, it is important to evaluate the model thoroughly and review the steps executed to construct the model, to be certain it properly achieves the business objectives. A key objective is to determine if there is some important business issue that has not been sufficiently considered. At the end of this phase, a decision on the use of the data mining results should be reached.

- **Deployment** Creation of the model is generally not the end of the project. Even if the purpose of the model is to increase knowledge of the data, the knowledge gained will need to be organized and presented in away that is useful to the customer. Depending on the requirements, the deployment phase can be as simple as generating a report or as complexas implementing a repeatable data scoring (e.g. segment allocation) or data mining process.

2.5 Overview of Document

Increasing the number of users is an important for any industry or business. Payment methods and the subscription fee effects on the subscriptions renewal.

- **Extracting Data**: Dataset is gathered from Indian Government Data Repository. Dataset contains the Data of 8 EAG States.
- **Objective**: The main goal and challenge of the system is to find out what lacks in these areas and to find how much percentage of deaths can be preventable
- **Imports**: This section describes way to connect external analysis requirements to python.
- **Cleaning and prep**: Section describes procedure to cleaning the data to make it analysis ready.
- **Analysis**: This is most important section of project describing all analysis done and its visualization.
- **Conclusion**: Conclusion concludes the project.

CHAPTER 3

OVERALL DESCRIPTION

3.1 Data:

The data set obtain from the "Soil Health Card" program started by Department o f Agriculture, Cooperation & Farmers Welfare Ministry of Agriculture and Farme rs Welfare Government of India. For this project we downloaded the sets of 21 Di stricts' datasets in CSV file format from https://soilhealth.dac.gov.in . The file co ntains following columns- sub_district_name_english1, sub_district_name_english, sampl e_no, farmername, LandArea, DagNo, SurveyNo, longitude, Latitude, pH, EC, OC, N, P, K, S , Zn, Fe, Cu, Mn and B.

Field	Description
sub_district_name_english1	District Names
sub_district_name_english	Village Names
sample_no	Sample Number
farmername	Farmer Name
LandArea	Land Area in Hector
SurveyNo	Survey Number
Longitude	Longitude
Latitude	Latitude
pH	pH value of soil
EC	Electrical conductivity (Total Dissolve salts dSm/m)
OC	Organic Carbon (%)
N	Nitrogen (kg/hector)
P	Phosphorous (kg/hector)
K	Potassium (kg/hector)
S	Sulphur(PPM)
Zn	Zinc (PPM)
Fe	Iron (PPM)
Cu	Copper (PPM)
Mn	Manganese (PPM)
B	Boron (PPM)

Table 3.1: Attribute Description

3.2 Imports

- **Matplotlib**: Matplotlib is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makessome change to a figure. It is used to show visualizations of analysis.
- **Pandas**: Pandas is an open-source, BSD-licensed Python library pro-viding high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Pandas is used to perform operation on data.
- **Numpy**: Numpy is used to for mathematical operations. This packageprovides easy use of mathematical function.
- **Seaborn**: Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractiveand informative statistical graphics.
- **Sklearn**: Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines.
- **Glob**: The glob module finds all the pathnames matching a specified pattern. It is use to import datasets.
- **Warnings**: The warnings filter controls whether warnings are ignored, displayed, or turned into errors (raising an exception).
- **Plotly**: The plotly Python library is an interactive, open-source plotting library that supports over 40 unique chart types covering a wide range of statistical, financial, geographic, scientific, and 3-dimensional use-cases.

CHAPTER 4

REQUIREMENT SPECIFICATION

4.1 External Requirement Specification

4.1.1 Hardware Interfaces

- Processor : C o r e i3 or above
- RAM : 6 GB (min)
- Hard Disk : 1 TB
- GPU : 2GB or above

4.1.2 Software Interfaces

Functional

First of all model should be successfully trained by developer. Installing Anaconda on Windows.

1. Download and install Anaconda (windows version).
2. Select the default options when prompted during the installation ofAnaconda.

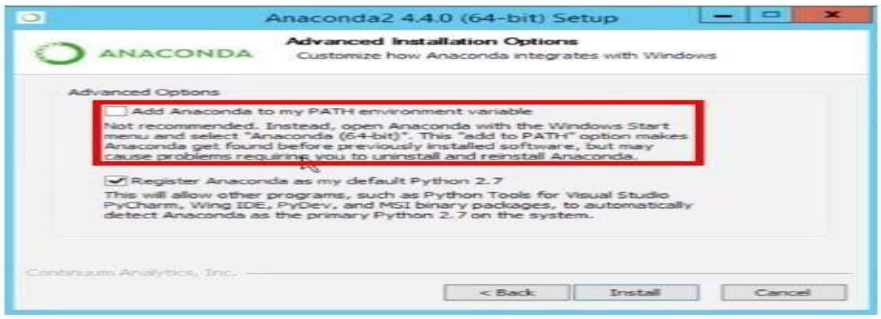

Figure 4.1: Anaconda Path Setting

3. After you finished installing, open Anaconda Prompt. Type the command below to see that you can use a Jupyter (IPython) Notebook.
4. If you didn't check the add Anaconda to path argument during the installation process, you will have to add python and conda to your environment variables.
5. This step gives two options for adding python and conda to your path.

Anaconda Navigator:

Anaconda Navigator is a desktop graphical user interface (GUI) included in

Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command line commands. Navigator can search for packages on Anaconda Cloud orin a local Anaconda Repository. It is available for Windows, macOS and Linux. To get Navigator, get the Navigator cheat sheet and install Anaconda. The Navigator Getting started with Navigator section shows how to start Navigator from the shortcuts or from a terminal window. Preferable in case data is small or can be performable in Anaconda.

Google Colab:

In case of very large data where Anaconda takes time use Google Colab for better performance. Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machinelearning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing freeaccess to computing resources including GPUs. Colab is free to use.

CHAPTER 5
DATA PRE-PROCESSING

5.1 Data Pre-processing and Data Cleaning:

Data pre-processing includes merging and manipulation Data from 21 dataset files. Using pandas lib, we load all csv files into the one single data frame. Our Dataset have contained many unnecessary data columns using pd.drop pandas function we drop all those columns from our data set.

```
In [3]: final_df=frame.drop(columns=['Textbox48', 'textbox2', 'textbox3', 'Textbox4', 'textbox5', 'textbox6',
              'textbox10', 'textbox11', 'textbox16', 'Textbox12', 'Textbox17',
              'Textbox19', 'Textbox21', 'Textbox23', 'Textbox25', 'Textbox27',
              'Textbox29', 'Textbox31', 'Textbox33', 'Textbox35','Textbox49',
                                  'Textbox50','DagNo','longitude','Latitude','sample_no','farmername','SurveyNo','LandArea'])
```

This dataset has many columns contains value like miss place pointer as we know that pH standard range is 1 to 14 but in pH columns have some value like 27.5,52,78 etc. Same with other columns. For correcting this error in data frame, we use the Data frame manipulation.

The main Aim of data pre-processing is that dataset have not any null values also the dataset not conations any outlier in dataset because this will rise the problem for ML model.

```
In [9]: for col in list:
            final_df[col] = final_df[col].fillna(final_df[col].mean())
```

Unrealistic and outlier data will be removed to develop a representative and more general model.

```
In [11]: x=final_df
         final_df=x[(x.N<600)&(x.OC<1.8)&(x.EC<1)&(x.K<600)&(x.S<25)&(x.Zn<1.75)&(x.Fe<10)&(x.Cu<3.5)&(x.Mn<20)&(x.B<1.2)]
```

In our data set there is no columns call Crop. We apply some rules on data frame to create the Crop column the rules like a specific crop will required the some predefine Range of NPK macro nutrition's from soil. Also, specific crop only grows specific location. Consider all the rules we apply those rules on data frame.

```
In [13]: soil=final_df
         soil['Crop']='No Crop'

In [14]: #Cotton
         soil.Crop[((soil.N>=100) & (soil.K>=120)  & (soil.P>=20)) & ((soil.District =='Akola')|(soil.District =='Jamner')|(soil.District
                    |(soil.District =='Nashik')|(soil.District =='Junnar')|(soil.District =='Bhusawal')|(soil.District =='Jalgaon')|(so
         soil[(soil.Crop == 'Cotton')]

         #Soybean
         soil.Crop[((soil.N>=120)  & (soil.K>=100) & (soil.P>=15)) &((soil.District =='Buldana')|(soil.District =='Washim')|(soil.District
                    |(soil.District =='Yavatmal'))]='Soybean'
         soil[(soil.Crop == 'Soybean')]


         #Sugarcane
         soil.Crop[((soil.N>=100) & (soil.K>=150) & (soil.P>=30))&
                 ((soil.District =='Koregaon')|(soil.District =='Karad')|(soil.District =='Karvir')
                    |(soil.District =='Kagal'))]='Sugarcane'
         soil[(soil.Crop == 'Sugarcane')]
```

Figure 5.1: Target variable creation

5.2 Dataset after all the Cleaning:

Out[14]:

	District	Village	pH	EC	OC	N	P	K	S	Zn	Fe	Cu	Mn	B	Crop
40186	Jamner	Ambadi	8.20	0.32	0.135	191.296	121.572	272.5672	9.40	0.770	8.80	0.38	9.50	0.50	Cotton
40190	Jamner	Ambadi	7.90	0.30	0.500	297.920	32.419	271.6200	10.80	0.720	7.30	0.12	7.60	1.10	Cotton
40193	Jamner	Ambadi	7.50	0.73	0.200	269.696	121.570	271.0800	12.70	0.530	5.20	0.14	7.50	1.10	Cotton
40208	Jamner	Ambadi	7.50	0.50	0.300	269.696	24.314	353.6100	12.00	0.850	8.20	0.31	8.20	1.10	Cotton
40209	Jamner	Ambadi	7.50	0.50	0.300	269.696	24.314	353.6100	12.00	0.850	8.20	0.31	8.20	1.10	Cotton
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1001293	Akola	Waranghushi	6.75	0.45	0.850	119.200	136.000	425.6300	2.74	0.845	4.30	1.28	10.50	0.36	Cotton
1001294	Akola	Waranghushi	6.75	0.45	0.850	119.200	136.000	425.6300	2.74	0.845	4.30	1.28	10.50	0.36	Cotton
1001295	Akola	Waranghushi	6.75	0.45	0.850	119.200	136.000	425.6300	2.74	0.845	4.30	1.28	10.50	0.36	Cotton
1001296	Akola	Waranghushi	6.75	0.45	0.850	119.200	136.000	425.6300	2.74	0.845	4.30	1.28	10.50	0.36	Cotton
1001317	Akola	Waranghushi	6.96	0.38	0.850	360.600	136.000	581.9400	3.01	0.610	0.74	0.49	0.19	0.46	Cotton

88539 rows × 15 columns

Figure 5.2: Dataset after per-processing

5.3 Finding Correlation:

Relationship can be an important tool for feature engineering in building machine learning models. (describe a possible future event)or which are uncorrelated with the goal (number or thing that changes) are probably good candidates to trim from the model (shoe size is not a useful (describe a possible future event)or for (money paid for working)). Also, if two (describe a possible future event) or are strongly or related to each other, then we only need to use one of them (in (describing a possible future event) (money paid for working), there is no need to use both age in years, and age in months). Taking these steps means that the resulting model will be simpler, and simpler models are easier to understand/explain.

There are many measures for relationship, but by far the most widely used one is Pearson's Product-Moment coefficient, or Pearson's r. Given a collection of paired (x, y) values, Pearson's coefficient produces a value between -1 and +1 to put into numbers the strength of dependence between the (numbers that change/things that change) x and y. A value of +1 means that all the (x, y) points lie exactly on a line with positive slope, and inversely, a value of -1 means that all of the points lie exactly on a line with negative slope. A Pearson's coefficient of 0 means that there is no relationship between the two (numbers that change/things that change). To see this visually, we can look at plots of our possible data, and the Pearson's coefficient figured out/calculated from them.

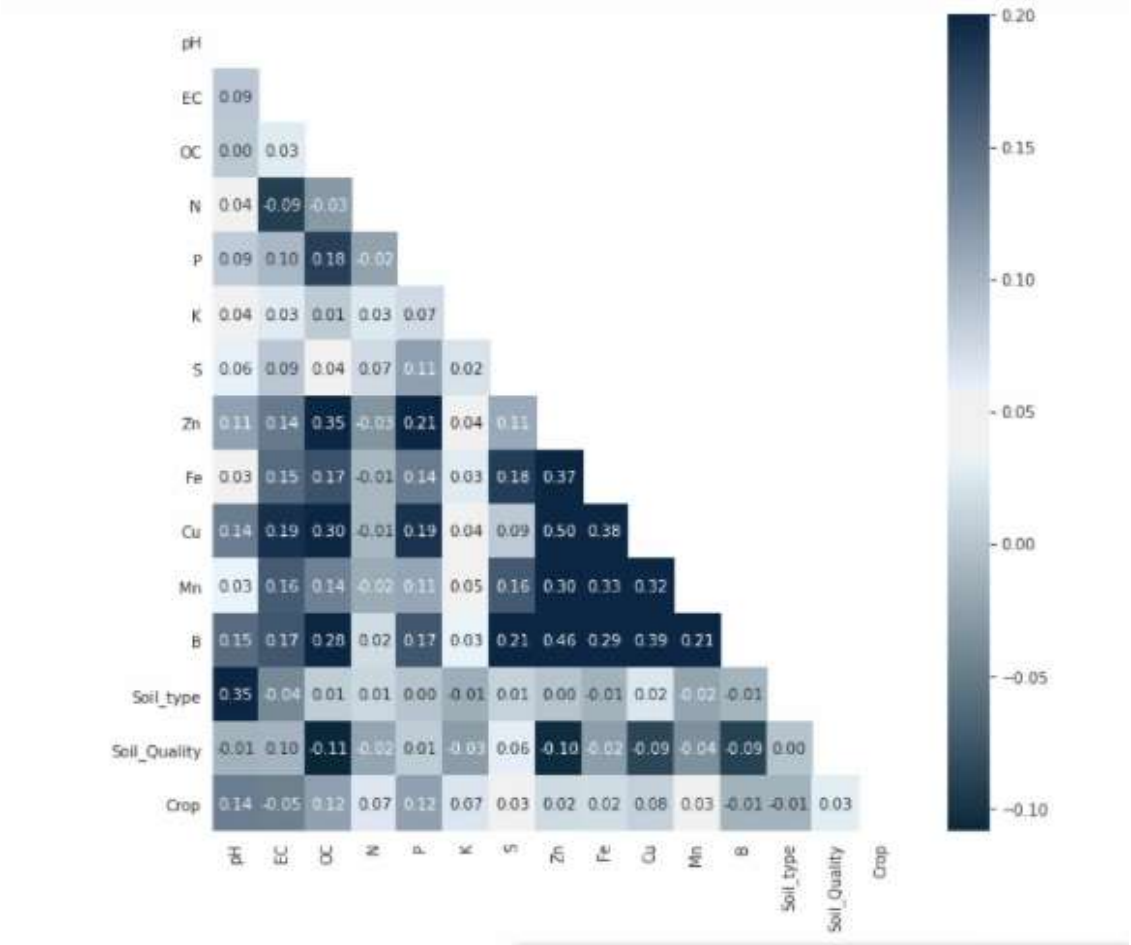

Figure 5.3: Correlation Heat map

From above plot, it is clear that the target variable Crop has positive correlation with the features considers in this project.

5.4 Over-Fitting Data:

Overfitting happens when a (related to studying numbers) model or machine learning set of computer instructions (records on a camera or computer) the noise of the data. (in an obvious, gut-feeling way), overfitting happens when the model or the set of computer instructions fits the data too well. Specifically, overfitting happens if the model or set of computer instructions shows low bias but high variance.

```
In [42]: soil.Crop.value_counts()
Out[42]: No Crop        238596
         Cotton          86306
         Soybean         34398
         Sugarcane       14722
         GroundNut        5416
         Wheat            2606
         Name: Crop, dtype: int64
```

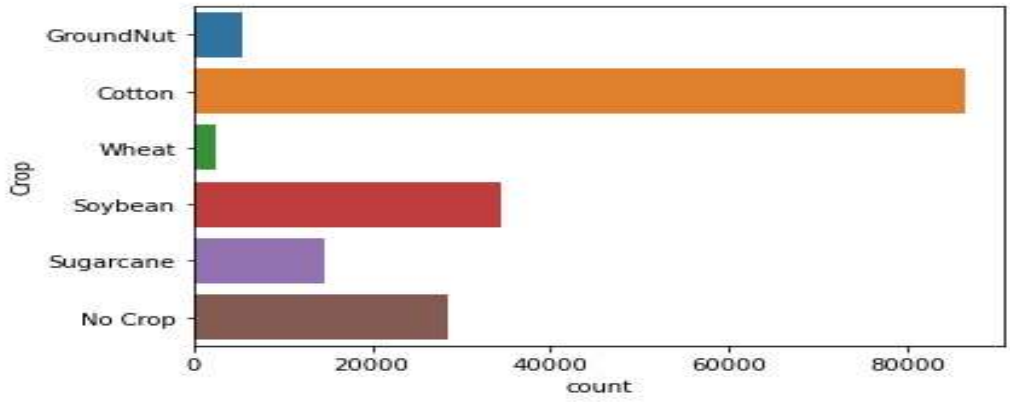

Figure 5.4: Imbalance target variable plot

Overfitting is often a result of an extremely (too much) complicated model, and it can be prevented by fitting many models and using validation or cross validation to compare their (describe a possible future event) vie accuracies on test data.

5.5 Balancing the Dataset:

We use the Random sampling for balancing the other majority variables over the minority variables. We calculate the percentage of minority variable and apply to the majority variable for balancing the Data.

Balancing of data to avoid over-fitting of model

```
In [40]: df1=df.copy()
         s=df1[df1.Crop == 'Sugarcane' ]
         s=s.sample(frac=0.18,random_state=2021)
         c=df1[df1.Crop == 'Cotton' ]
         c=c.sample(frac=0.033,random_state=2021)
         g=df1[df1.Crop == 'GroundNut' ]
         g=g.sample(frac=0.49,random_state=2021)
         so=df1[df1.Crop == 'Soybean' ]
         so=so.sample(frac=0.077,random_state=2021)
         n=df1[df1.Crop == 'No Crop' ]
         n=n.sample(frac=0.095,random_state=2021)
         w=df1[df1.Crop == 'Wheat' ]
         c.shape,s.shape,so.shape,g.shape,n.shape,w.shape

Out[40]: ((2848, 17), (2650, 17), (2649, 17), (2654, 17), (2720, 17), (2606, 17))
```

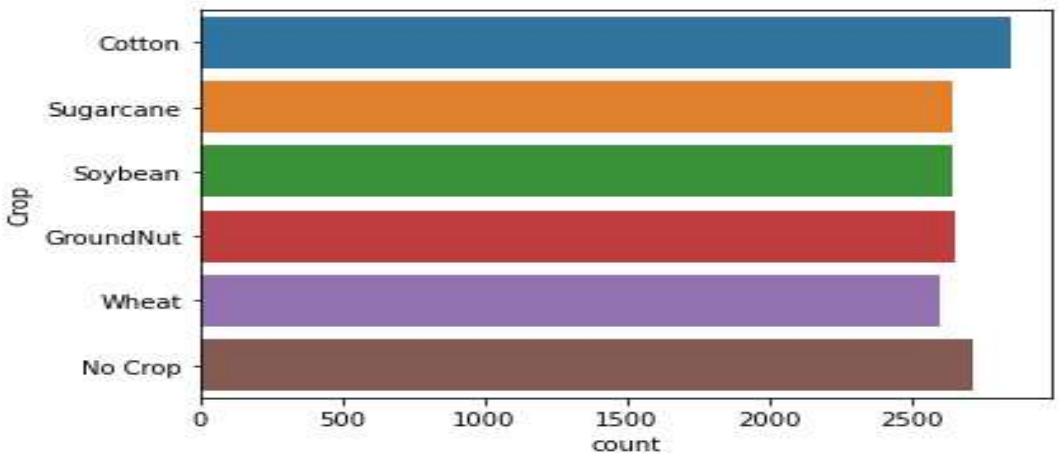

Figure 5.5 Balanced Data to avoid overfitting variables

CHAPTER 6
MODEL EVALUATION

6.1 Training and testing Dataset:

In training and testing Dataset are separated by 70:30 proportion. Train data frame have 70% of data and test Data frame have 30% data.

```
In [11]: X_train
```

Out[11]:

	pH	EC	OC	N	P	K	S	Zn	Fe	Cu	Mn	B
161456	7.29	0.36	0.190	156.600	30.00000	390.00	23.18000	0.890	4.00	0.46	3.56	0.1900
59957	7.20	0.23	0.947	269.700	42.28000	252.00	0.80200	0.724	5.31	2.46	5.62	0.7400
59946	7.80	0.25	0.427	291.650	62.89000	325.92	1.42000	0.639	6.27	2.90	13.00	0.7010
146920	7.36	0.25	0.360	213.248	16.80672	468.16	20.38437	0.400	3.39	2.81	12.82	0.4818
5200	7.37	0.22	0.340	194.600	33.50000	310.50	16.50000	0.810	3.54	1.10	9.70	0.5400
...	...	...	...	...	...	...	...	...	...	...	...	...
59955	7.90	0.32	0.501	238.340	52.79000	192.64	2.03700	0.616	4.89	2.55	10.10	1.0900
95092	8.16	0.47	0.770	337.000	28.00000	459.00	7.00000	0.940	5.15	1.22	7.21	0.2900
126571	7.70	0.36	0.530	291.000	30.31000	302.40	5.73000	0.670	6.11	3.21	16.39	1.0400
122084	6.17	0.11	0.840	150.600	30.65000	212.80	12.16000	0.090	1.33	0.52	2.17	0.5160
132593	7.43	0.65	0.390	103.480	105.20000	280.94	2.25000	0.410	1.40	0.28	1.04	0.1500

11764 rows × 12 columns

```
In [13]: y_train
```

```
Out[13]: 161456      No Crop
         59957        Cotton
         59946         Wheat
         146920      No Crop
         5200       GroundNut
                     ...
         59955         Wheat
         95092       Soybean
         126571        Wheat
         122084    Sugarcane
         132593       Cotton
         Name: Crop, Length: 11764, dtype: object
```

```
In [14]: X_test
```

Out[14]:

	pH	EC	OC	N	P	K	S	Zn	Fe	Cu	Mn	B
174393	7.48	0.199	0.58	234.080	16.7100	164.50	6.30000	1.29	4.27	2.400	13.76	1.0400
92199	7.37	0.520	0.62	240.840	143.0000	295.72	3.56000	1.14	0.84	1.780	10.56	0.7700
73391	7.92	0.267	0.45	315.000	35.6000	267.80	6.48000	0.44	5.98	0.470	4.78	0.2900
90246	7.90	0.510	0.26	179.000	15.4800	369.60	3.48000	0.86	1.64	2.100	5.56	0.6000
129538	7.90	0.690	0.53	331.000	75.0000	264.80	5.69000	0.35	1.22	1.100	6.11	0.5000
...	...	...	...	...	...	...	...	...	...	...	...	...
76679	7.57	0.134	0.39	273.000	26.2000	227.80	18.20000	0.67	2.45	0.470	7.28	0.5200
72046	7.30	0.210	0.73	223.500	40.7100	156.10	6.00000	1.09	5.30	0.711	0.58	0.8900
1232	6.56	0.060	0.74	175.600	36.4000	442.40	5.18000	0.29	3.85	1.750	6.60	0.0640
89269	7.94	0.630	0.66	229.550	121.0000	332.23	2.04000	0.98	0.63	1.250	3.37	0.7600
76110	7.62	0.170	0.54	338.688	23.4192	476.00	20.93856	0.79	3.26	0.530	14.58	0.7665

5043 rows × 12 columns

```
In [15]: y_test
Out[15]: 174393      No Crop
         92199       Soybean
         73391       Soybean
         90246       Soybean
         129538       Cotton
                      ...
         76679       Soybean
         72046        Cotton
         1232      GroundNut
         89269       Soybean
         76110       Soybean
         Name: Crop, Length: 5043, dtype: object
```

Figure 6.1 Train and Test Dataset

6.2 Model Building:

6.2.1 K-Nearest Neighbours Classifier:

Neighbours-based classification is a type of instance-based learning or non-generalizing learning: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbours of each point: a query point is assigned the data class which has the most representatives within the nearest neighbours of the point.

```
In [60]: from sklearn.neighbors import KNeighborsClassifier

         knn_clf=KNeighborsClassifier(weights='uniform')

         knn_clf.fit(X_train,y_train)
         ypred=knn_clf.predict(X_test)
```

```
confusion matrix
 [[441 116  72  99  89  38]
 [ 30 670   5  23  44  24]
 [139  36 498 105  16  22]
 [169  94 118 335  43  36]
 [129 132  18  70 390  56]
 [ 27  21  23  28  23 660]]
Classification report
              precision    recall  f1-score   support

      Cotton       0.47      0.52      0.49       855
   GroundNut       0.63      0.84      0.72       796
     No Crop       0.68      0.61      0.64       816
     Soybean       0.51      0.42      0.46       795
   Sugarcane       0.64      0.49      0.56       795
       Wheat       0.79      0.84      0.82       782

    accuracy                           0.62      4839
   macro avg       0.62      0.62      0.61      4839
weighted avg       0.62      0.62      0.61      4839

Accuracy_score
 0.6187228766274023
```

Figure 6.2 KNN classifiers without tuning

```
In [63]: from sklearn.model_selection import RandomizedSearchCV

         parameters = {'n_neighbors': np.arange(1,10)}
         print(parameters)

         from sklearn.model_selection import StratifiedKFold
         kfold = StratifiedKFold(n_splits=5, random_state=2021,shuffle=True)

         knn = KNeighborsClassifier()

         # Tuned according to accuracy score
         cv = RandomizedSearchCV(knn, param_distributions=parameters,cv=kfold,scoring='accuracy')
         cv.fit( x , y )

         #print(cv.cv_results_)

         print(cv.best_params_)

         print(cv.best_score_)
```

```
{'n_neighbors': array([1, 2, 3, 4, 5, 6, 7, 8, 9])}
C:\Users\chetan dhumne\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:278: UserWarning: The total space of parameters 9 is smaller than n_iter=10. Running 9 iterations. For exhaustive searches, use GridSearchCV.
  warnings.warn(
{'n_neighbors': 1}
0.7334905635894404
```

Figure 6.3 KNN classifiers with tuning parameter

Above KNN-model without tuning predict 61% accuracy on test data whereas the model with K-nearest neighbour {'neighbours': 1} as tuning parameter predict 73% of accuracy.

6.2.2 Decision Tree:

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

```
In [16]: from sklearn.tree import DecisionTreeClassifier
         depth_range = [3,4,5,8,10,15,20]
         minsplit_range = [5,10,20,25,30,40]
         minleaf_range = [5,10]

         parameters = dict(max_depth=depth_range,
                           min_samples_split=minsplit_range,
                           min_samples_leaf=minleaf_range)
```

```
In [17]: from sklearn.model_selection import StratifiedKFold
         kfold = StratifiedKFold(n_splits=5, random_state=2021,shuffle=True)

         from sklearn.model_selection import RandomizedSearchCV
         dtclf = DecisionTreeClassifier(random_state=2021)
         cv = RandomizedSearchCV(dtclf,cv=kfold,param_distributions=parameters,scoring='accuracy')

         cv.fit(x,y)
         # Best Parameters
         print(cv.best_params_)
         print(cv.best_score_)

         {'min_samples_split': 10, 'min_samples_leaf': 10, 'max_depth': 15}
         0.8032483359525561
```

```
In [18]: dtclf.fit(X_train,y_train)
         preds=dtclf.predict(X_test)
         #preds=cv.predict(X_test)
```

```
In [19]: from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
         result=confusion_matrix(y_test,preds)
         print("Confusion matrix\n",result)
         result1=classification_report(y_test,preds)
         print("Classification report\n",result1)
         result2=accuracy_score(y_test,preds)
         print("Accuracy_score\n",result2)

         Confusion matrix
          [[671  19  43  62  48  12]
          [  3 781   1   7   4   0]
          [ 52   6 630  90  20  18]
          [ 59  20  67 611  20  18]
          [ 25  27  13  12 715   3]
          [  2   0   3  11   2 764]]
         Classification report
                       precision    recall  f1-score   support

               Cotton       0.83      0.78      0.81       855
           GroundNut       0.92      0.98      0.95       796
             No Crop       0.83      0.77      0.80       816
             Soybean       0.77      0.77      0.77       795
           Sugarcane       0.88      0.90      0.89       795
               Wheat       0.94      0.98      0.96       782

            accuracy                           0.86      4839
           macro avg       0.86      0.86      0.86      4839
        weighted avg       0.86      0.86      0.86      4839

         Accuracy_score
          0.8621616036371151
```

Figure 6.4 Decision Tree

Above Decision Trees model without tuning predict 86% accuracy on test data whereas the model with {'min_samples_split': 10, 'min_samples_leaf': 10, 'max_depth': 15} as tuning parameter predict 80% of accuracy.

6.2.3 Random Forest:

A random forest is a Meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

```
In [69]: from sklearn.ensemble import RandomForestClassifier
         rm=RandomForestClassifier(random_state=2021)
         rm.fit(X_train,y_train)
         predx=rm.predict(X_test)

In [70]: result=confusion_matrix(y_test,predx)
         print("confusion matrix\n",result)
         result1=classification_report(y_test,predx)
         print("Classification report\n",result1)
         result2=accuracy_score(y_test,predx)
         print("Accuracy_score\n",result2)

confusion matrix
 [[754   5  15  46  27   8]
 [  3 790   0   1   2   0]
 [ 37   8 683  62  24   2]
 [ 32  11  43 696   7   6]
 [ 19  14   0   8 753   1]
 [  1   0   4   5   2 770]]
Classification report
              precision    recall  f1-score   support

      Cotton       0.89      0.88      0.89       855
   GroundNut       0.95      0.99      0.97       796
     No Crop       0.92      0.84      0.88       816
     Soybean       0.85      0.88      0.86       795
   Sugarcane       0.92      0.95      0.94       795
       Wheat       0.98      0.98      0.98       782

    accuracy                           0.92      4839
   macro avg       0.92      0.92      0.92      4839
weighted avg       0.92      0.92      0.92      4839

Accuracy_score
 0.9187848729076256
```

```
In [72]: import numpy as np
         parameters = {'max_features': np.arange(1,11)}
         print(parameters)

         from sklearn.model_selection import StratifiedKFold
         kfold = StratifiedKFold(n_splits=5, random_state=2021,shuffle=True)

         model_rf = RandomForestClassifier(random_state=2021)
         from sklearn.model_selection import RandomizedSearchCV
         cv = RandomizedSearchCV(model_rf,cv=kfold,param_distributions=parameters,scoring='accuracy')

         cv.fit( x , y )

         print(cv.best_params_)

         print(cv.best_score_)

         print(cv.best_estimator_)

{'max_features': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])}
{'max_features': 2}
0.9226137631742096
RandomForestClassifier(max_features=2, random_state=2021)
```

Figure 6.5 Random forest

Above Random Forest model without tuning perdict 91.8% accuracy on test data whereas the model with (max_features=2, random_state=2021) as tuning parameter perdict 92% of accuracy.

6.2.4 Support Vector Machine (SVM):

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outlier's detection's implement the "one-versus-one" approach for multi-class classification. In total, classes * (classes - 1) / 2 classifiers are constructed and each one trains data from two classes. To provide a consistent interface with other classifiers, the classes. To option allows to monotonically transform the results of the "one-versus-one" classifiers to a "one-vs-rest" decision function of shape (n_samples, n_classes).

SVC-OVO

```
from sklearn.preprocessing import StandardScaler scaler=StandardScaler() x=scaler.fit_transform(x)
```

```
In [79]: from sklearn.svm import SVC
         from sklearn.model_selection import RandomizedSearchCV

         C_range = np.linspace(1,10,5)
         gamma_range = np.linspace(0.1,10,5)

         param_grid = dict(gamma=gamma_range,C=C_range)

In [80]: param_grid
Out[80]: {'gamma': array([ 0.1  ,  2.575,  5.05 ,  7.525, 10.   ]),
          'C': array([ 1.  ,  3.25,  5.5 ,  7.75, 10.  ])}

In [81]: from sklearn.model_selection import StratifiedKFold
         kfold = StratifiedKFold(n_splits=5, random_state=2021,shuffle=True)
         svc=SVC(probability=True,kernel='rbf',decision_function_shape='ovo')
         svmGrid =RandomizedSearchCV(svc,param_distributions=param_grid, cv=kfold,verbose=3,n_iter=2)

         svmGrid.fit(X_train,y_train)
         print(svmGrid.best_params_)
         print(svmGrid.best_score_)

         Fitting 5 folds for each of 2 candidates, totalling 10 fits
         [CV] gamma=7.525, C=1.0 ..............................................

In [82]: svc.fit(X_train,y_train)
         pred=svc.predict(X_test)

In [83]: from sklearn.metrics import classification_report,confusion_matrix,accuracy_score

         result=confusion_matrix(y_test,pred)
         print("confusion matrix\n",result)
         result1=classification_report(y_test,pred)
         print("Classification report\n",result1)
         result2=accuracy_score(y_test,pred)
         print("Accuracy_score\n",result2)

         confusion matrix
          [[216 183 208 119  80  49]
          [ 62 410  37  82  45 160]
          [ 63  73 572  76  19  13]
          [ 92 142 170 272  79  40]
          [111 176  70 110 218 110]
          [ 57  37 126  46  48 468]]
         Classification report
                       precision    recall  f1-score   support

              Cotton       0.36      0.25      0.30       855
           GroundNut       0.40      0.52      0.45       796
             No Crop       0.48      0.70      0.57       816
             Soybean       0.39      0.34      0.36       795
           Sugarcane       0.45      0.27      0.34       795
               Wheat       0.56      0.60      0.58       782

            accuracy                           0.45      4839
           macro avg       0.44      0.45      0.43      4839
        weighted avg       0.44      0.45      0.43      4839

         Accuracy_score
          0.4455466005373011
```

Figure 6.6 SVM-OVO

SVM OVR

```
[32]: from sklearn.model_selection import StratifiedKFold
      kfold = StratifiedKFold(n_splits=2, random_state=2020,shuffle=True)
      svc=SVC(probability=True,kernel='rbf',decision_function_shape='ovr')
      svmGrid =RandomizedSearchCV(svc,param_distributions=param_grid, cv=kfold,verbose=True,n_iter=2)
      svmGrid.fit(x, y)
      # Best Parameters
      print(svmGrid.best_params_)
      print(svmGrid.best_score_)
```

```
Fitting 2 folds for each of 2 candidates, totalling 4 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:  3.7min finished

{'gamma': 0.1, 'C': 7.75}
0.5692935140357975
```

```
In [33]: svc.fit(X_train,y_train)
         prede=svc.predict(X_test)
```

```
In [34]: result=confusion_matrix(y_test,prede)
         print("confusion matrix\n",result)
         result1=classification_report(y_test,prede)
         print("Classification report\n",result1)
         result2=accuracy_score(y_test,prede)
         print("Accuracy_score\n",result2)
```

```
confusion matrix
 [[216 183 208 119  80  49]
 [ 62 410  37  82  45 160]
 [ 63  73 572  76  19  13]
 [ 92 142 170 272  79  40]
 [111 176  70 110 218 110]
 [ 57  37 126  46  48 468]]
Classification report
              precision    recall  f1-score   support

           0       0.36      0.25      0.30       855
           1       0.40      0.52      0.45       796
           2       0.48      0.70      0.57       816
           3       0.39      0.34      0.36       795
           4       0.45      0.27      0.34       795
           5       0.56      0.60      0.58       782

    accuracy                           0.45      4839
   macro avg       0.44      0.45      0.43      4839
weighted avg       0.44      0.45      0.43      4839

Accuracy_score
 0.4455466005373011
```

Figure 6.7 SVM-OVR

Support Vector Machine is Worst performer among the all the ML classification models. Without Tuning the Accuracy of SVM is 44% and with tuning Accuracy is 45% on test Data.

6.2.5 Voting Classifier:

The idea behind the Voting Classifier is to combine conceptually different machine learning classifiers and use a majority vote or the average predicted probabilities (soft vote) to predict the class labels. Such a classifier can be useful for a set of equally well performing model in order to balance out their individual weaknesses.

```
In [74]: from sklearn.metrics import confusion_matrix, classification_report
         from sklearn.ensemble import VotingClassifier
         from sklearn.metrics import roc_auc_score
         from sklearn.metrics import accuracy_score
         from sklearn.neighbors import KNeighborsClassifier
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.svm import SVC

         rm=RandomForestClassifier(random_state=2021)
         knn=KNeighborsClassifier()
         clf = DecisionTreeClassifier(random_state=2021)
         svc_ovo=SVC(probability=True,kernel='rbf',decision_function_shape='ovo')
         svc_ovr=SVC(probability=True,kernel='rbf',decision_function_shape='ovr')

         Voting = VotingClassifier(estimators=[('KNN',knn),('RFC',rm),('DTC',clf),('SVC_OVO',svc_ovo),('SVC_OVR',svc_ovr)],voting='soft')
```

```
In [75]: Voting.fit(X_train,y_train)

         y_pred = Voting.predict(X_test)

         print('Confusion Matrix',confusion_matrix(y_test, y_pred))
         print('Classification Report',classification_report(y_test, y_pred))
         print("Accuray=", accuracy_score(y_test,y_pred))
```

```
Confusion Matrix [[691  14  36  57  47  10]
 [  1 787   0   6   2   0]
 [ 48   8 676  65  17   2]
 [ 56  17  68 621  24   9]
 [ 24  27   7  11 723   3]
 [  4   0   5   8   2 763]]
Classification Report              precision    recall  f1-score   support

      Cotton       0.84      0.81      0.82       855
   GroundNut       0.92      0.99      0.95       796
     No Crop       0.85      0.83      0.84       816
     Soybean       0.81      0.78      0.79       795
   Sugarcane       0.89      0.91      0.90       795
       Wheat       0.97      0.98      0.97       782

    accuracy                           0.88      4839
   macro avg       0.88      0.88      0.88      4839
weighted avg       0.88      0.88      0.88      4839

Accuray= 0.8805538334366605
```

Figure 6.8 Voting Classifier

In voting Classifier algorithm KNN, Decision tree, random forest, SVM, XGboost all the algorithm are consider to find the best Score of Voting Classifier. According to the voting classifier results over all Accuracy is 88%.

6.2.6 Extreme Gradient Boosting Classifiers:

Boosting**,** takes a more iterative approach. It's still technically an ensemble technique in that many models are combined together to perform the final one, but takes a cleverer approach. Rather than training all of the models in isolation of one another, boosting trains models in succession, with each new model being trained to correct the errors made by the previous ones. Models are added sequentially until no further improvements can be made.

In [36]:
```
# Import the necessary modules
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import accuracy_score
from xgboost import XGBClassifier

xgclf = XGBClassifier(random_state=2021)
xgclf.fit(X_train,y_train)

y_pred = xgclf.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print('Accuracy:',accuracy_score(y_test,y_pred))
```

```
[00:45:12] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly s
et eval_metric if you'd like to restore the old behavior.
[[777   2  10  37  25   4]
 [  1 794   0   0   1   0]
 [ 15   8 709  71   9   4]
 [ 24   4  32 723   7   5]
 [ 12   8   1   8 764   2]
 [  0   0   2   3   1 776]]
              precision    recall  f1-score   support

           0       0.94      0.91      0.92       855
           1       0.97      1.00      0.99       796
           2       0.94      0.87      0.90       816
           3       0.86      0.91      0.88       795
           4       0.95      0.96      0.95       795
           5       0.98      0.99      0.99       782

    accuracy                           0.94      4839
   macro avg       0.94      0.94      0.94      4839
weighted avg       0.94      0.94      0.94      4839

Accuracy: 0.9388303368464559
```

In [38]:
```
##################Tunning using Randomized Search CV ############

lr_range = [0.1,0.2,0.5,0.6,1]
n_est_range = [70,100,150,180]
depth_range = [3,4,5,6,7,8,9]

parameters = dict(learning_rate=lr_range,
                  n_estimators=n_est_range,
                  max_depth=depth_range)


from sklearn.model_selection import RandomizedSearchCV
clf = XGBClassifier(random_state=2021)
rcv = RandomizedSearchCV(clf, param_distributions=parameters,
                  cv=kfold,scoring='accuracy',n_iter=5,random_state=2021)

rcv.fit(x,y)
#df_rcv = pd.DataFrame(rcv.cv_results_)
print(rcv.best_params_)

print(rcv.best_score_)
```

```
[00:46:07] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly s
et eval_metric if you'd like to restore the old behavior.
[00:46:11] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly s
et eval_metric if you'd like to restore the old behavior.
[00:46:15] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly s
et eval_metric if you'd like to restore the old behavior.
[00:46:19] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly s
et eval_metric if you'd like to restore the old behavior.
{'n_estimators': 150, 'max_depth': 6, 'learning_rate': 0.6}
0.9193277311829069
```

Figure 6.9 XGBoost

6.2.7 Model Accuracies and Result:

Seven different classification models to predict the best fitted crop for given samples of soil nutrients. The accuracies found are as follows,

Model Name	Without Tuning Accuracy	With Tuning Accuracy
KNN Classifier	62%	73%
Decision Tree	86%	78%
Random Forest	91%	92.2%
SVM-OVO	44%	65%
SVM-OVR	44%	56%
XGBoost Classifier	93.8%	91.9%
Voting Classifier	88%	88%

Figure 6.10 All Model Accuracy Table

The library like scikit-learn is used to implement KNN Classifier, Decision Tree Classifier, Random Forest Classifier, SVM-OVO, SVM-OVR, XGBoost Classifier and Voting Classifier. The accuracies of the seven different methods were compared and Random Forest Classifier had the best prediction performance based on accuracy and f1-score with hyper-tuning parameters whereas without tuning XGBoost had the best prediction performance based on accuracy and f1-score.

6.3 Model Deployment

For Deployment of ML model Stream-lit python library is used which is open source App framework to Design and Deploy the ML or Data Science Applications.

Steps to build the web interface:

1. Create the Pickle file of appropriate ML model.
2. Create the app.py file which get the values from User.
3. Create the prediction function to predict the value.
4. Write the web Interface using Stream-lit library.
5. Run the code on machine and Check the Out on Webpage.

Figure 6.11 Model Deployment Frontend

CHAPTER 7
FUTURE SCOPE

- For future work add parameters like moisture contain of soil, climate, locations, etc for crop prediction.
- Increase the verities of crops and vegetables .
- The various Crop Cultivation Seasons like Kharif/Rabbi and Rainfall for recommendation of Crop.
- In Addition intercrop recommendation with primary Crop.
- Recommendation Possible Plant disease if Soil Micro-Nutrient deficiency.

CHAPTER 8
CONCLUSION

The dataset is collected from "Soil Health Card" government website for project. The soil dataset is of 23 different districts and talukas of MAHARASHTRA state from which soil nutrients values like N, P, K, and other micro-nutrients are used to form a column of Crop on which the predictions are done. Other than Crop column there are two columns with indicates soil quality (Fertile or non-fertile) and soil type (Neutral, Acidic or Alkaline).This columns are formed using pH values and macro-micro nutritional values of soil. The data was initially unbalance and uncleaned so the result of ML model was over fitted but after exploratory data analysis, feature engineering and data cleaning techniques application the data was balanced and clean by which overcame the problem of overfitting. In project it is consider that most of the crops production are done in Kharif season in the month of May-June with adequate amount of humidity and temperature and average rainfall. Python 3.8.0 is used as programming language for this project.

The Jupyter notebook file is used to write and execute the python code. Seven different classif ication models were used to recommended crops. The supervised machine learning classificat ion algorithms such as implement KNN Classifier, Decision Tree Classifier, Random Forest C lassifier, SVM-OVO, SVM-OVR, XGBoost Classifier and Voting Classifier to build a highly accurate classification model for the purpose of predicting crop names. The accuracies got fro m these models are shown below:

Model Name	**Without Tuning Accuracy**	**With Tuning Accuracy**
KNN Classifier	62%	73%
Decision Tree	86%	78%
Random Forest	91%	92.2%
SVM-OVO	44%	65%
SVM-OVR	44%	56%
XGBoost Classifier	93.8%	91.9%
Voting Classifier	88%	88%

Based on the models tested, the accuracies of the seven different methods were compared and Random Forest Classifier had the best prediction performance based on accuracy (92%) and f1-score with hyper-tuning parameters (RandomForestClassifier(max_features=2, random_state=2021)) whereas without tuning XGBoost had the best prediction performance based on accuracy (93.8%) and f1-score.

REFERENCES

[1] https://www.soilhealth.dac.gov.in/NewHomePage/NutriReport
[2] https://docs.streamlit.io/en/stable/
[3] Introduction To Machine Learning with Python (3rd Edition)
[4] Learn Python The Hard Way (3rd Edition)
[5] Python Data Science Handbook