A PROJECT REPORT ON

CROP YIELD PREDICTION SYSTEM

A Project report submitted in partial fulfillment of the requirement for the award of the Degree of

MASTER OF COMPUTER APPLICATION

Submitted By

Ashish Kumar Sahoo

(2224100029)

Under the esteemed guidance of

Mrs. Rojalin Mallick

School Of Computer Science

ODISHA UNIVERSITY OF TECHNOLOGY AND RESEARCH

(Techno Campus, Ghatikia, Mahalaxmi Vihar, Bhubaneswar-751003)

Academic Year 2023-2024

ODISHA UNIVERSITY OF TECHNOLOGY AND RESEARCH
(Techno Campus, Ghatikia, Mahalaxmi Vihar, Bhubaneswar-751003)

School Of Computer Science

CERTIFICATE

This is to certify that the project report entitled **“CROP YIELD PREDICTION SYSTEM”** being submitted by **Mr. Ashish Kumar Sahoo** bearing registration no: **2224100029** in partial fulfillment for the award of the Degree of Master in Computer Application to the Odisha University of Technology and Research is a record of bonafide work carried out by him under my guidance and supervision.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

Internal Guide | **Head of School of Computer Science**

Mrs. Rojalin Mallick | **Dr Jibitesh Mishra**

ODISHA UNIVERSITY OF TECHNOLOGY AND RESEARCH

(Techno Campus, Ghatikia, Mahalaxmi Vihar, Bhubaneswar-751003)

School Of Computer Science

DECLARATION

I ***Ashish Kumar Sahoo*** bearing Registration No: **2224100029,** a bonafide student of **Odisha University of Technology and Research,** would like to declare that the project titled **"Crop Yield Prediction System"** in partial fulfillment of the MCA Degree course of Odisha University of Technology and Research is my original work in the year 2023 under the guidance of **Mrs. Rojalin Mallick,** School Of Computer Science and it has not previously formed the basis for any degree or diploma or other any similar title submitted to any university.

Date: 05 January 2024 **Ashish Kumar Sahoo**

Place: OUTR, BBSR **(2224100029)**

ODISHA UNIVERSITY OF TECHNOLOGY AND RESEARCH

(Techno Campus, Ghatikia, Mahalaxmi Vihar, Bhubaneswar-751003)

School Of Computer Science

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my advisor, **Mrs. Rojalin Mallick,** Professor, School of Computer Science whose knowledge and guidance have motivated me to achieve my goals. He has consistently been a source of motivation, encouragement, and inspiration. The time I have spent working under his supervision has truly been a pleasure.

I take it as a great privilege to express our heartfelt gratitude to **Dr. Jibitesh Mishra**, Head of School of Computer Science for his valuable support and to all senior faculty members of the School of Computer Science for their help during my course. Thanks to the programmers and non-teaching staff of School of Computer Science.

Finally, special thanks to my parents for their support and encouragement throughout my life and this course. Thanks to all my friends and well-wishers for their constant support.

Date: 05 January 2024 **Ashish Kumar Sahoo**

Place: OUTR, BBSR **(2224100029)**

ABSTRACT

Using machine learning techniques has revolutionized agricultural practices by enabling accurate prediction of crop yields. This system employs various algorithms such as linear regression, decision tree, K-Neighbours, and Random Forest algorithm including regression and neural networks, to analyze historical agricultural data encompassing factors like area, weather patterns like annual rainfall and seasons, use of fertilizers and pesticide characteristics, and crop management practices. Through comprehensive data processing and model training, the system forecasts crop yields with a high degree of accuracy, aiding farmers in making informed decisions regarding planting strategies, resource allocation, and harvest planning. The implementation of this predictive system stands as a promising advancement in enhancing agricultural productivity and sustainability.

CONTENTS

1. INTRODUCTION

INTRODUCTION

1.1. Background

- The project titled 'Crop Yield Prediction System' is entirely dedicated to forecasting crop production based on provided features.
- This project optimizes agricultural decisions, empowering farmers with precise yield predictions to plan resources efficiently, maximize productivity, and enhance overall crop management, thereby contributing to food security and sustainability.
- Traditional machine learning algorithms like random forest is implemented in predicting the yield quantity of a crop.

1.2. Project Scope

1. In this proposed system every user is provided with a form with fields like the area in heacter, production in tonnes, annual rainfall, fertilizer, crop, state

2. Utilizing user-provided data, the model forecasts the expected crop yield for a specific agricultural harvest.

3. In practical application, using the predicted crop yield from the model, agricultural stakeholders and farmers can make informed decisions and take necessary actions to optimize cultivation practices and maximize crop productivity

1.3. Existing Work

- Several projects have addressed crop yield prediction.
- Some initiatives treated this as a classification problem, forecasting whether the crop yield would surpass a certain threshold based on input variables.
- Other endeavors utilized extensive datasets encompassing numerous crop instances, employing Convolutional Neural Networks (CNNs) to predict yield values.
- However , there is a scarcity of projects focusing on grassroots- level analysis or smaller-scale agricultural contexts in crop yield prediction endeavors.

1.4. Proposed system

On the internet, similar projects with larger datasets can be found which uses complex algorithms and uses world wide dataset according to country. But I was unable to find a project that deals with the grassroot issues like for example different states in india.

In my proposed system:

1. I have used a dataset of a different crop along with different state of india

2. This project aims to enhance crop yield prediction methods..

3. It will explore potential correlations between crop yield and weather factors such as different seasons along with ,quantity of fertilizer use,quantity of pesticide , different crop name etc.

4. Based on the insights derived from the correlations between crop yield and various factors, the objective is to construct a robust machine learning model capable of predicting crop yield

1.5. Project Workflow

This project involves the following steps for building the project:

1. Data Collection: In the step we collect the data from different sources.

2. Exploratory Data Analysis:

   2.1. Data Processing: In this step we clean the data, we perform operations like Dealing with missing values, removing outliers, etc.

   2.2. Feature Extraction and Engineering: In this step, with the help of different plotting methods and statistics we find out which features are relevant to the project and add more relevant columns which will improve the efficiency of the project. This will prepare the dataset for the next steps.

   2.3. Feature Scaling and selection: In this step, we identify the numerical features and categorical features. In the categorical features we perform the One Hot Encoding to convert the categories into numbers and in the numerical features, we perform the standard scaling to make the numeric values compactible with the rest of the data.

   2.4. Splitting the data: In this step, the data is split into the training and testing set (8 : 2). The training set is used to train the machine learning model. And the testing dataset is used to test the prediction of out model.

3. Choosing an Machine Learning Algorithm: In this step, we test different algorithms on our training dataset, and based on the results from the evaluation metrics, we pick the algorithm with the highest accuracy.

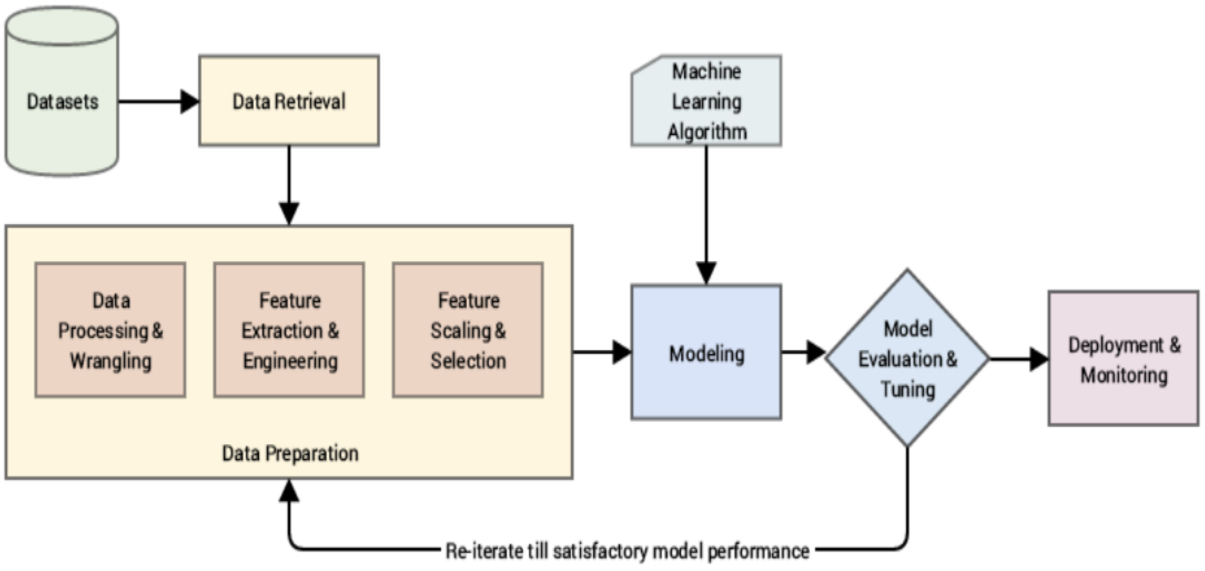
4. Modelling: In this step, the model with the highest accuracy is used to build the final

model, which will build the final ".pkl" file.

5. Model Evaluation and Tuning: In this step, we use a greater number of evaluation metrics to test the accuracy of the model using our testing dataset. The accuracy of our model can be improved using a method known as the Hyperparameter Tuning. In this step of our model's accuracy score doesn't get to a specific point (0., in our case), we again go back to step 2 and redo the whole thing.

6. Deployment and Monitoring: In this step, we our machine learning user friendly by building a pipeline and connecting it to a user interface. After building the UI we put our model in the public domain by deploying it using a cloud service like amazon azure.

Diagrammatic representation of the "Project Work Flow"


Datasets
Data Retrieval
Machine Learning Algorithm
Data Processing & Wrangling
Feature Extraction & Engineering
Feature Scaling & Selection
Data Preparation
Modeling
Model Evaluation & Tuning
Deployment & Monitoring
Re-iterate till satisfactory model performance

2. THEORITICAL BACKGROUND

THEORY OF MACHINE LEARNING

2.1.1. Steps-by-step process of Machine Learning

1. The most basic workflow for data mining, and therefore machine learning, can be divided into six steps. In the first step data acquisition has to be mentioned, as insufficient or biased data can lead to wrong results. In machine learning or data mining this data usually has to be quite big, as patterns might only emerge with thousands or millions of individual data points.

2. The second and maybe most important step is the cleaning of given data. Problems with data can include unsuitable features, noisy and so on. Features can be nominal (such as yes/no or male/female), ordinal (ranking such as school grades) or numerical (temperatures, cost and other parameters) and sometimes features have to be converted. An example would be to label all days with average temperatures over 10°C as "warm" and all below as "cold". Outliers might either be interesting, as in anomaly detection, or can change the outcome of the learning process negatively, e.g. when using experimental data where outliers have to be removed, therefore this has to be considered in the cleaning process. After all, a scientific or an analytical process can only be as good as the data provided is, as what often jokingly is said: "garbage in causes garbage out".

3. In the third and fourth step, with the cleaned data and a decision about the chosen modeling technique, is to build and train a model. This is possibly the most abstract step in data mining, especially when pre-built programs such as Azure Machine Learning Studio or scikit-learn (in Python) are used. Simply put, the training process is finding structures in the given training data. What kind of data or features these are is heavily dependent on the goal (clustering or prediction etc.). Taking a simple example of model training: a very famous dataset is the iris-dataset, that includes features of three different iris-species and uses clustering to determine the correct subspecies. Features included in this set are different parts of the flower, such as petal or sepal length. When using clustering, the training algorithm iteratively calculates the most common features and allows therefore a grouping process of all the data points. The result is heavily dependent on the complexity of these clusters, as is in regression the result dependent on the complexity of a curve. As can be seen in Figure 2.1 it can be quite hard to determine if the bottom-most red data point belongs to the species-A-cluster or to the species-B-cluster, depending on the method of measurement.

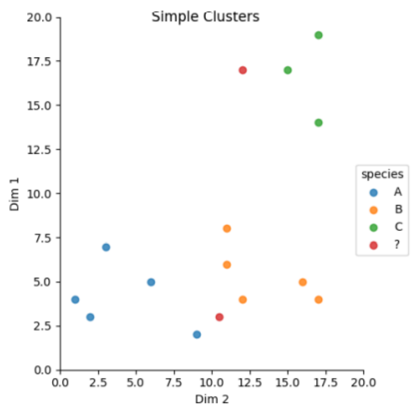


Figure 2.1: A simple, made-up, data set of three different species.

4. Testing and evaluating the model is mostly done by statistical methods and will seldom give a result of 100 % match between the training and validation data. Considering one of the most intuitive and simple data mining models, linear regression, this uncertainty is mostly covered by introducing a measurement of uncertainty.

2.1.2. Supervised and Unsupervised Learning

1. Supervised learning: "Supervised learning is a type of machine learning algorithm that uses a known dataset with labelled columns (called the training dataset) to make predictions. The training dataset includes input data and response values."

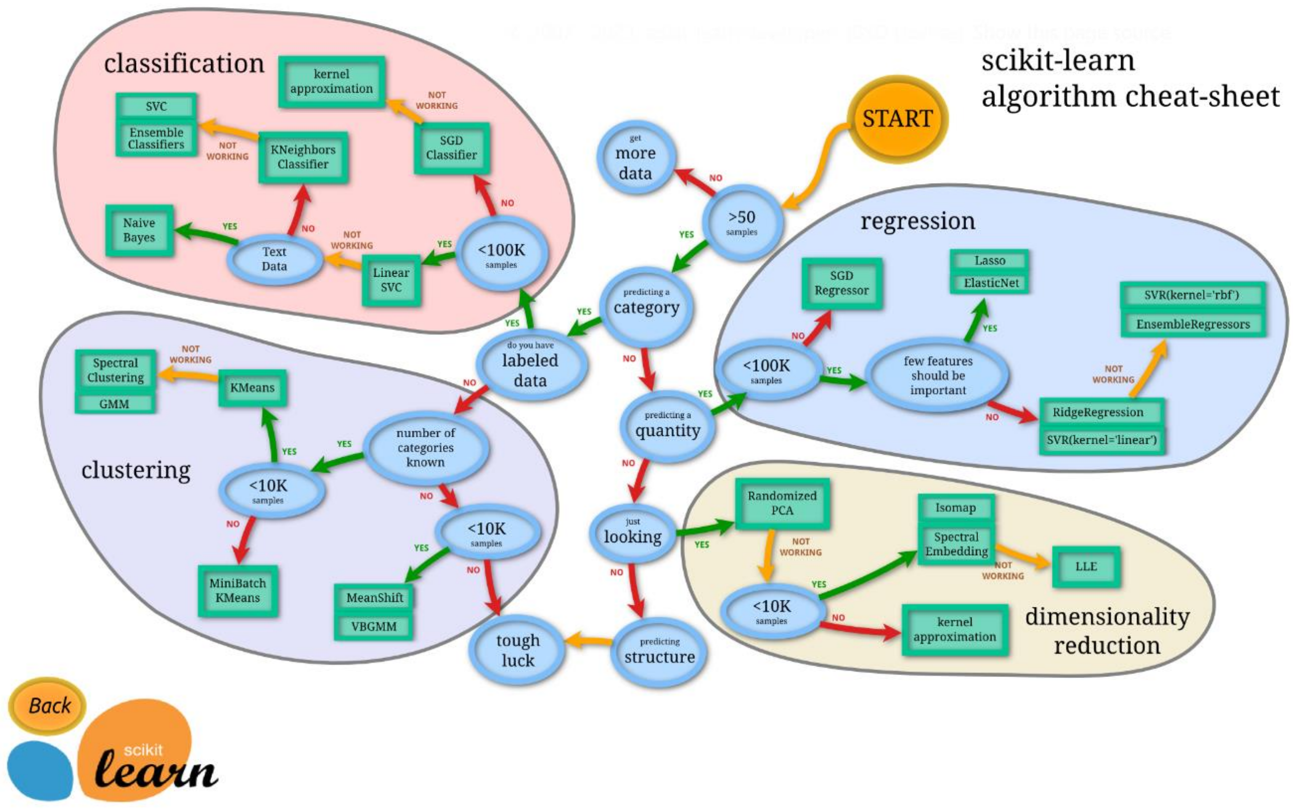
2. Unsupervised learning: "Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses."

3. In Machine learning an abundance of models and algorithms can be found, but most fundamentally these are divided into supervised and unsupervised learning.

    a. One fundamental example has been mentioned in the foregoing section, the clustering of iris-species. Former is a supervised process where data points are labeled ("species A", "species B" or "species C") and labels are calculated for new data points. Comparing calculated labels according to the trained model with the original label gives the model’s accuracy, hence supervised.

    b. Unsupervised learning on the other hand does not require any labeling, since the algorithm is searching for a pattern in the data. This might be useful when categorizing customers into different groups without a priori knowledge of which groups they belong to.

2.1.3. Machine Learning Types



In our project only Regression is used, so here are some of the commonly used regression algorithms:

2.1.3.1. Multiple Linear Regression

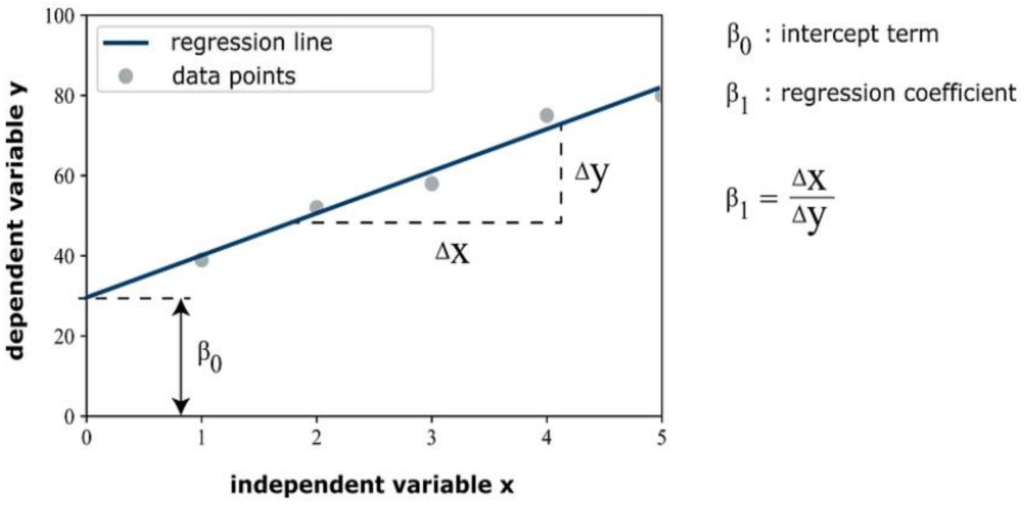
Linear regression models assume that the relationships between input and output variables are linear. These models are quite simplistic, but in many cases provide adequate and tractable representations of the relationships. The model aims a prediction of real output data Y by the given input data X = (x_1, x_2, …, x_p) and has the following form:

$$f(x) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

β describes initially unknown coefficients. Linear models with more than one input variable p > 1 are called multiple linear regression models. The best known estimation method of linear regression is the least squares method. In this method, the coefficients β = β_0, β_1…, β_p are determined in such a way that the Residual Sum of Squares (RSS) becomes minimal.

$$RSS = \sum_{i=1}^{n}(y_i - f(x_i))^2 = \sum_{i=1}^{n}(y_i - \beta_0 + \sum_{j=1}^{p} X_j \beta_j)^2$$

Here, y_i-f(x_i) describes the residuals, β_0 the estimate of the intercept term, and β_j the estimate of the slope parameter



Linear Regression: interception term and regression coefficients — Image by the author

2.1.3.2. Decision Trees

A Decision Tree grows by iteratively splitting tree nodes until the ‘leaves’ contain no more impurities or a termination condition is reached. The creation of the Decision Tree starts at the root of the tree and splits the data in a way that results in the largest Information Gain IG.

In general, the Information Gain IG of a feature a is defined as follows:

$$IG(D_p, a) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$$

with: N_p = Total number of instances of the parent node
N_j = Total number of instances of the j-th child node
D_p = Data amount of the parent node
D_j = Data amount of the j-th child node
I = Measure of impurity

In binary Decision Trees, the division of the total dataset D_p by attribute a into D_left and D_right is done. Accordingly, the information gain is defined as:

$$IG(D_p, a_i) = I(D_p) - \frac{N_{links}}{N_p} I(D_{links}) - \frac{N_{rechts}}{N_p} I(D_{rechts})$$

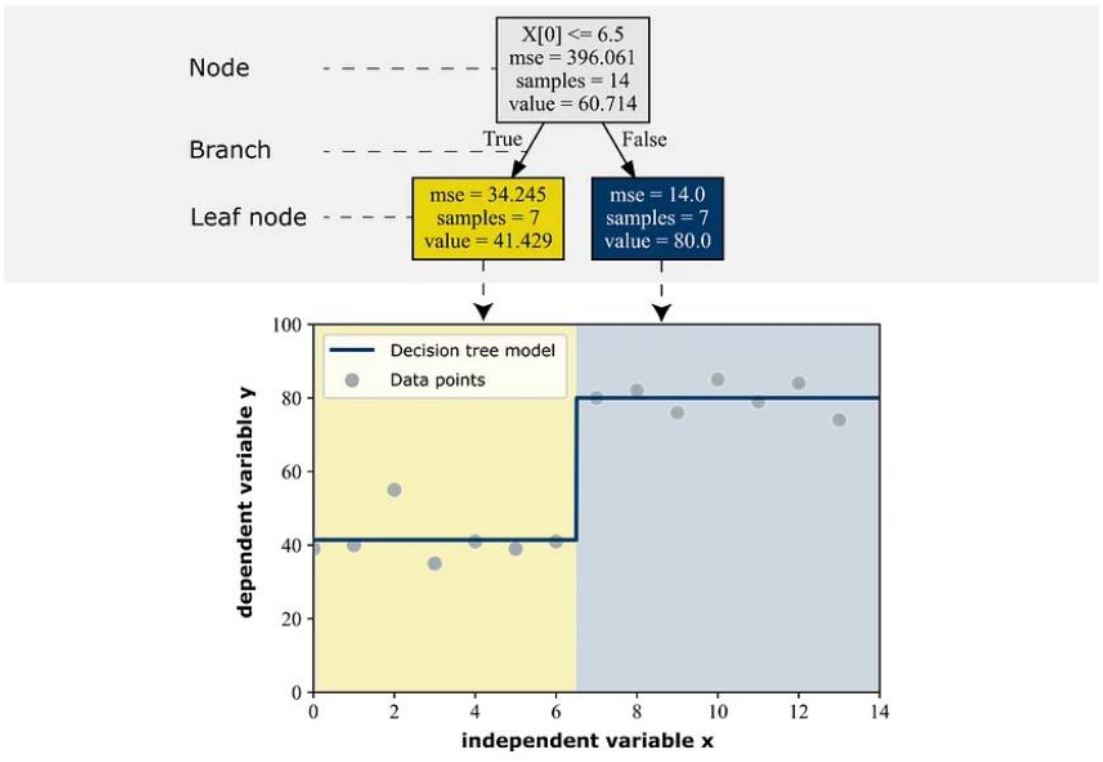
The algorithm aims at maximizing the information gain, i.e. the method wants to split the total dataset in such a way that the impurity in the child nodes is reduced the most.

While classification uses entropy or the Gini coefficient as a measure of impurity, regression uses, for example, the Mean Squared Error (MSE) as a measure of the impurity of a node.

$$I(t) = MSE(t) = \frac{1}{N_t} \sum_{i \in D_t} (i^{(i)} - \hat{y}_t)^2$$

with: N_t = Number of training objects of the node
D_t = Training subset of the node
$y^{(i)}$ = Actual target value
$\hat{y}_t$ = Predicted target value

Splitting methods that use the Mean Squared Error to determine impurity are also called variance reduction methods. Usually, the tree size is controlled by the maximum number of nodes max_depth, at which the division of the dataset stops.



Decision tree for a simple two-dimensional case with a depth of one — Image by the author

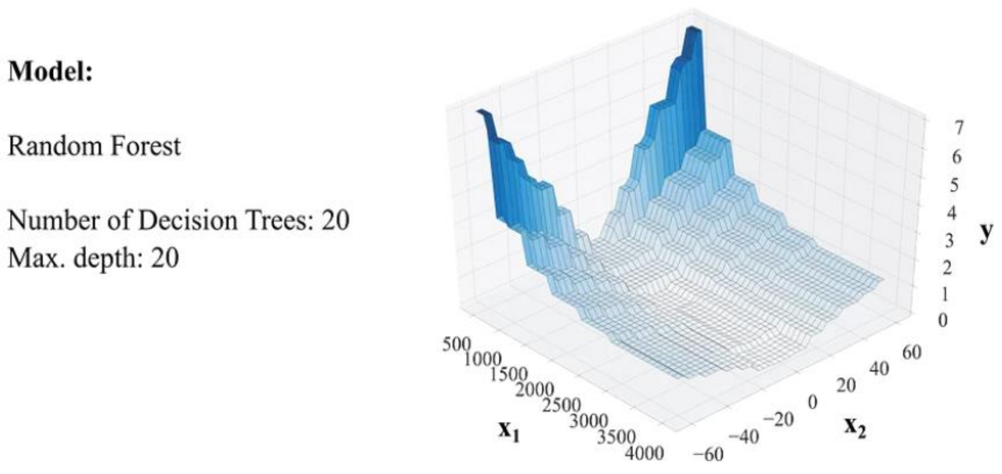
2.1.3.3. Random Forest

By merging several uncorrelated Decision Trees, often a significant improvement of the model accuracy can be achieved. This method is called Random Forest. The trees are influenced by certain random processes (randomization) as they grow. The final model reflects an averaging of the trees.

Different methods of randomization exist. According to Breiman, who coined the term ‘Random Forest’ in 1999, random forests are established according to the following procedure. First, a random sample is chosen from the total dataset for each tree. As the tree grows, a selection of a subset of the features takes place at each node. These serve as criteria for splitting the dataset. The target value is then determined for each Decision Tree individually. The averaging of these predictions represents the prediction of the Random Forest.

The Random Forest has a number of hyperparameters. The most crucial one, besides the maximum depth of the trees max_depth, is the number of decision trees n estimators. By default, the Mean Square Error (MSE) is used as criterion for splitting the dataset as the trees grow.

The following figure shows an example model for the Random Forest. The way it works results in the characteristic “step” form :



Random Forest: Sample Model — Image by the author

2.1.3.4. K Nearest Neighbors Regressor

K nearest neighbors is a simple algorithm that stores all available cases and predict the numerical target based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970’s as a non-parametric technique. KNN regression uses the following distance functions:

Distance functions

Euclidean $$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan $$\sum_{i=1}^{k}|x_i - y_i|$$

Minkowski $$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$

The above three distance measures are only valid for continuous variabl

2.1.4. Evaluation Metrics

2.1.4.1. Root Mean Squared Error

RMSE is the most popular evaluation metric used in regression problems. It follows an assumption that errors are unbiased and follow a normal distribution. Here are the key points to consider on RMSE:

1. The power of 'square root' empowers this metric to show large number deviations.
2. The 'squared' nature of this metric helps to deliver more robust results, which prevent canceling the positive and negative error values. In other words, this metric aptly displays the plausible magnitude of the error term.
3. It avoids the use of absolute error values, which is highly undesirable in mathematical calculations.
4. When we have more samples, reconstructing the error distribution using RMSE is considered to be more reliable.
5. RMSE is highly affected by outlier values. Hence, make sure you've removed outliers from your data set prior to using this metric.
6. As compared to mean absolute error, RMSE gives higher weightage and punishes large errors.

$$RMSE=\sqrt{\frac{\sum_{i=1}^{N}(Predicted_i-Actual_i)^2}{N}}$$

where N is the Total Number of Observations.

2.1.4.2. R-Squared

We learned that when the RMSE decreases, the model's performance will improve. But these values alone are not intuitive.
In the case of a classification problem, if the model has an accuracy of 0.8, we could gauge how good our model is against a random model, which has an accuracy of 0.5. So the random model can be treated as a benchmark. But when we talk about the RMSE metrics, we do not have a benchmark to compare.
This is where we can use the R-Squared metric. The formula for R-Squared is as follows:

$$R^2 = 1 - \frac{\text{MSE(model)}}{\text{MSE(baseline)}}$$

$$\frac{\text{MSE(model)}}{\text{MSE(baseline)}} \qquad \frac{\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{N}(\overline{y}_i-\hat{y}_i)^2}$$

MSE (model): Mean Squared Error of the predictions against the actual values

MSE (baseline): Mean Squared Error of mean prediction against the actual values

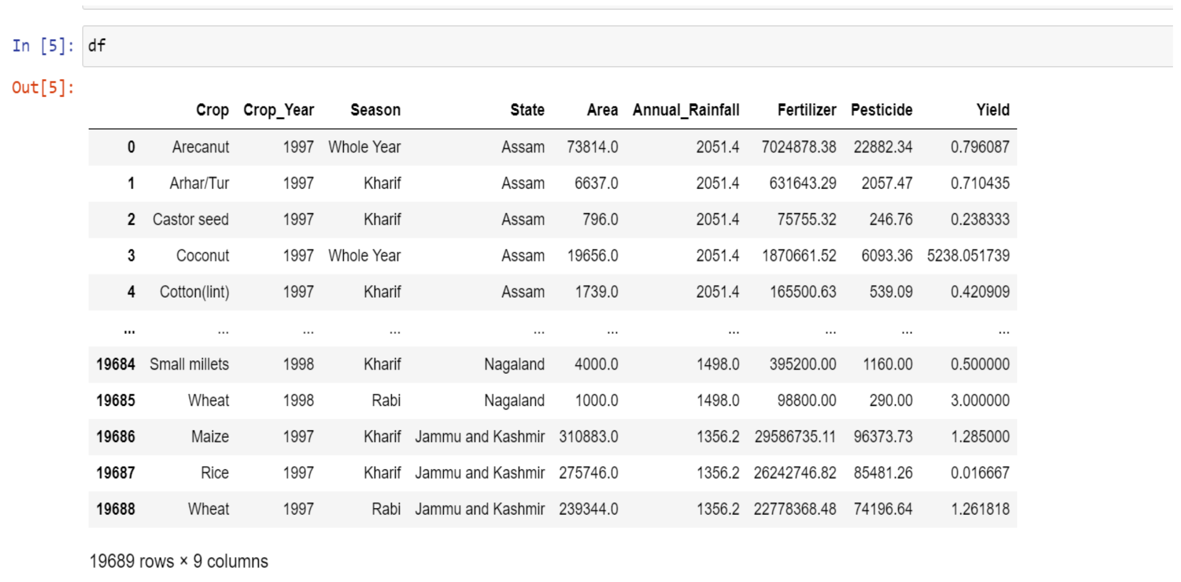
3.Methodology

3.1. Data Review

1. Dataset Source - https://www.kaggle.com/datasets/akshatgupta7/crop-yield-in-indian-states-dataset/data

2. This data contains 10 columns and 19689 rows.

3. **Logical View**



```
In [5]: df
Out[5]:
```

	Crop	Crop_Year	Season	State	Area	Annual_Rainfall	Fertilizer	Pesticide	Yield
0	Arecanut	1997	Whole Year	Assam	73814.0	2051.4	7024878.38	22882.34	0.796087
1	Arhar/Tur	1997	Kharif	Assam	6637.0	2051.4	631643.29	2057.47	0.710435
2	Castor seed	1997	Kharif	Assam	796.0	2051.4	75755.32	246.76	0.238333
3	Coconut	1997	Whole Year	Assam	19656.0	2051.4	1870661.52	6093.36	5238.051739
4	Cotton(lint)	1997	Kharif	Assam	1739.0	2051.4	165500.63	539.09	0.420909
...	...	...	...	...	...	...	...	...	...
19684	Small millets	1998	Kharif	Nagaland	4000.0	1498.0	395200.00	1160.00	0.500000
19685	Wheat	1998	Rabi	Nagaland	1000.0	1498.0	98800.00	290.00	3.000000
19686	Maize	1997	Kharif	Jammu and Kashmir	310883.0	1356.2	29586735.11	96373.73	1.285000
19687	Rice	1997	Kharif	Jammu and Kashmir	275746.0	1356.2	26242746.82	85481.26	0.016667
19688	Wheat	1997	Rabi	Jammu and Kashmir	239344.0	1356.2	22778368.48	74196.64	1.261818

19689 rows × 9 columns

4. **About the Dataset:**
   This dataset file contains agricultural data for various crops cultivated in multiple states in India during the years 1997-2020. The dataset is focused on predicting crop yields based on several agronomic factors, such as weather conditions, fertilizer and pesticide usage, and other relevant variables. The dataset is presented in tabular form, with each row representing data for a specific crop and its corresponding features. It has 19698 rows and 10 columns (9 features and 1 label).The aim of collection of data for this dataset was to find any correlations between the various attributes enlisted in the dataset. The attributes are

   a. **Crop:** Specifies the type of crop being cultivated.

   b. **Crop_Year:** Indicates the year in which the crop was cultivated.

   c. **Season:** Represents the specific growing season (Kharif, Rabi, Whole Year)

   d. **State:** Denotes the state where the crop was cultivated.

   e. **Area:** Signifies the land area used for cultivating the particular crop.

   f. **Annual_Rainfall:** Represents the amount of rainfall in that particular area during the year of cultivation.

   g. **Fertilizer:** Indicates the amount of fertilizer used for cultivation.

   h. **Pesticide:** Denotes the quantity of pesticides used for crop protection.

   i. **Yield:** Reflects the yield or output obtained from cultivating the crop.

3.2. Software Introduction

The following software have been used in the project:

1. **Anaconda**: It is a Python distribution for data science and machine learning. The main purpose of Python virtual environments is to create isolated environments where each project is independent from the other, using its own dependencies.

2. **Jupyter Notebook:** Jupyter Notebook is an open-source web-based interactive computing environment used for data analysis, visualization, and the creation of computational notebooks. It allows users to create and share documents that contain live code, equations, visualizations, and narrative text.

3. **VS Code:** Short for Visual Studio Code, is a free and open-source source code editor developed by Microsoft. It is widely used by programmers and developers for various programming languages and platforms.
   VS Code is designed to be lightweight, customizable, and highly extensible, making it suitable for a wide range of programming tasks. It provides a user-friendly interface and a powerful set of features that enhance productivity and streamline the development process.

4. **Python**: Python is a high-level, general-purpose programming language that was created by Guido van Rossum and first released in 1991. It emphasizes code readability and has a simple and clean syntax, making it easy to learn and use. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

5. **Python modules used:**
   a. **Pandas**: Pandas is an open-source Python library that provides high-performance, easy-to-use data manipulation and analysis tools. It is built on top of the NumPy library and is widely used in data science, machine learning, and data analysis workflows.

   b. **NumPy:** NumPy (Numerical Python) is a powerful open-source Python library that provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. It serves as a fundamental building block for scientific computing and data analysis in Python.

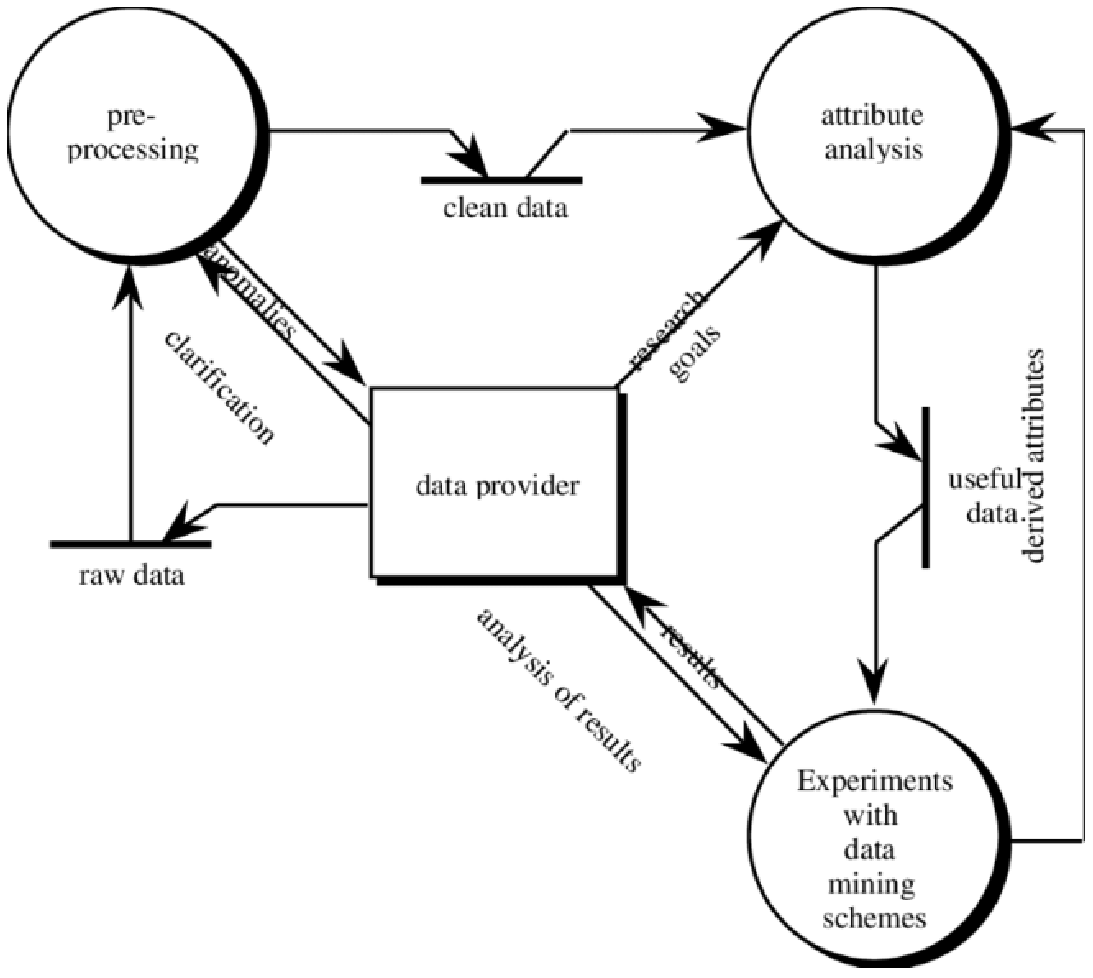
   c. **Matplotlib.pyplot:** Itis a module within the Matplotlib library, which is a widely used plotting library in Python. The pyplot module provides a high-level interface for creating various types of plots, charts, and visualizations.

   d. **Seaborn:** Seaborn is a Python data visualization library built on top of Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn is particularly useful for visualizing complex datasets and exploring relationships between variables.

e. **Scikit-Learn:** Scikit-learn, also known as sklearn, is a widely used open-source machine learning library for Python. It is built on top of NumPy, SciPy, and Matplotlib, and provides a comprehensive set of tools for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, model selection, and preprocessing.

f. **Flask:** Flask is a lightweight web framework for Python that allows developers to build web applications quickly and with minimal boilerplate code. It is known for its simplicity, flexibility, and ease of use. Flask is often referred to as a "micro" framework because it provides only the essential features needed to create web applications, without imposing any strict architectural patterns or dependencies.

3.3. Data Flow Diagram:

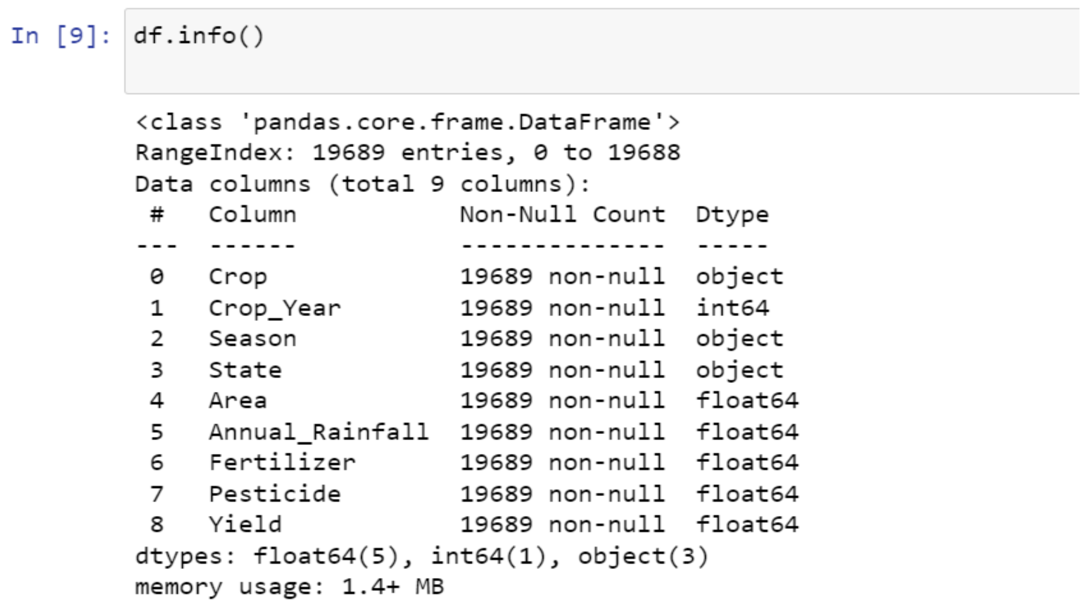


4 – Analysis

4.1 – Exploratory Data Analysis

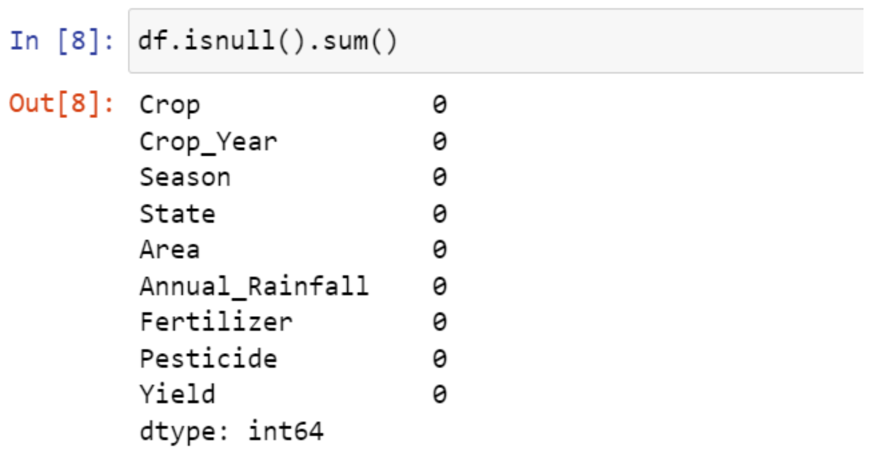
4.1.1 – Data Identification

4.1.1.1 – Check for datatypes:



```
In [9]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19689 entries, 0 to 19688
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Crop             19689 non-null  object 
 1   Crop_Year        19689 non-null  int64  
 2   Season           19689 non-null  object 
 3   State            19689 non-null  object 
 4   Area             19689 non-null  float64
 5   Annual_Rainfall  19689 non-null  float64
 6   Fertilizer       19689 non-null  float64
 7   Pesticide        19689 non-null  float64
 8   Yield            19689 non-null  float64
dtypes: float64(5), int64(1), object(3)
memory usage: 1.4+ MB
```

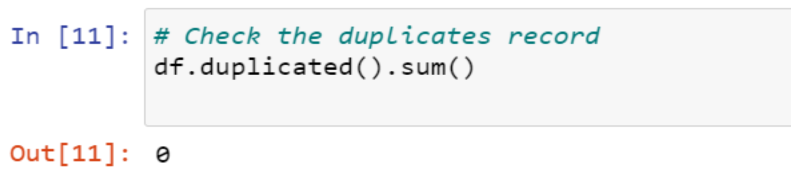
4.1.1.2. Check for missing values



```
In [8]: df.isnull().sum()

Out[8]: Crop               0
        Crop_Year          0
        Season             0
        State              0
        Area               0
        Annual_Rainfall    0
        Fertilizer         0
        Pesticide          0
        Yield              0
        dtype: int64
```

1. There are no missing values in the dataset.

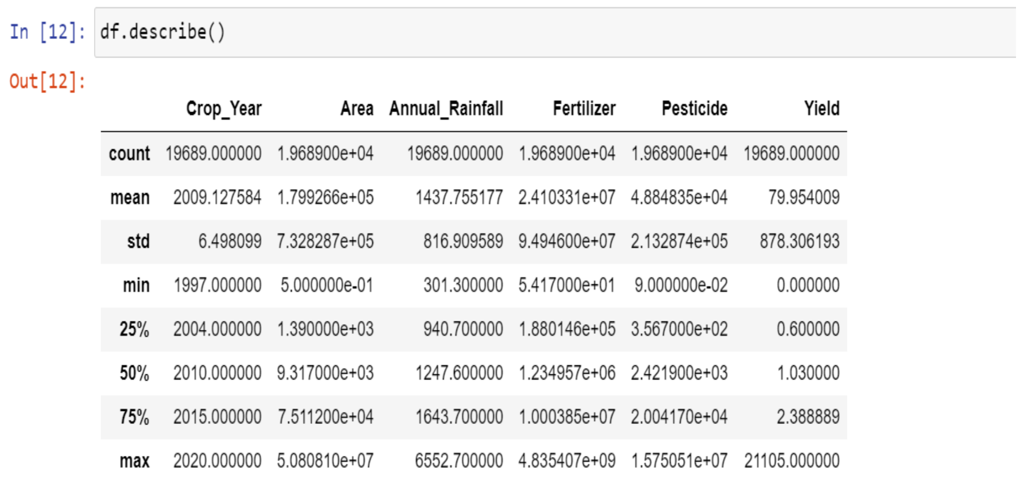
4.1.1.3. Check for duplicates



```
In [11]: # Check the duplicates record
         df.duplicated().sum()

Out[11]: 0
```

1. There are no duplicate elements in the dataset.

4.1.1.4. Describe the Mathematical Calculation

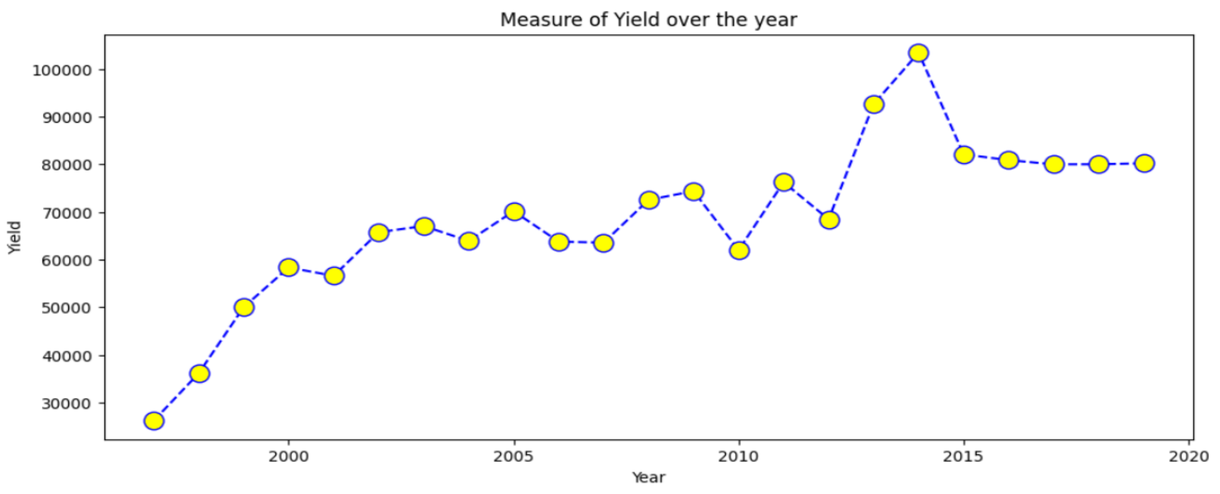


```
In [12]: df.describe()
```

```
Out[12]:
          Crop_Year          Area  Annual_Rainfall    Fertilizer     Pesticide         Yield
count  19689.000000  1.968900e+04     19689.000000  1.968900e+04  1.968900e+04  19689.000000
mean    2009.127584  1.799266e+05      1437.755177  2.410331e+07  4.884835e+04     79.954009
std        6.498099  7.328287e+05       816.909589  9.494600e+07  2.132874e+05    878.306193
min     1997.000000  5.000000e-01       301.300000  5.417000e+01  9.000000e-02      0.000000
25%     2004.000000  1.390000e+03       940.700000  1.880146e+05  3.567000e+02      0.600000
50%     2010.000000  9.317000e+03      1247.600000  1.234957e+06  2.421900e+03      1.030000
75%     2015.000000  7.511200e+04      1643.700000  1.000385e+07  2.004170e+04      2.388889
max     2020.000000  5.080810e+07      6552.700000  4.835407e+09  1.575051e+07  21105.000000
```

4.1.2. Visualizing the data

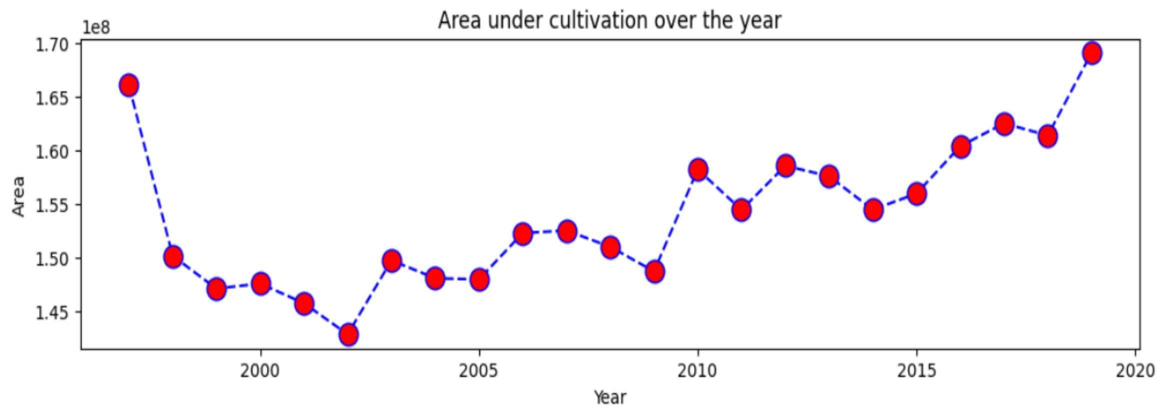
4.1.2.2. Measure Of Yield Over The Year



Insights:

1. Measure of yield increase every year.

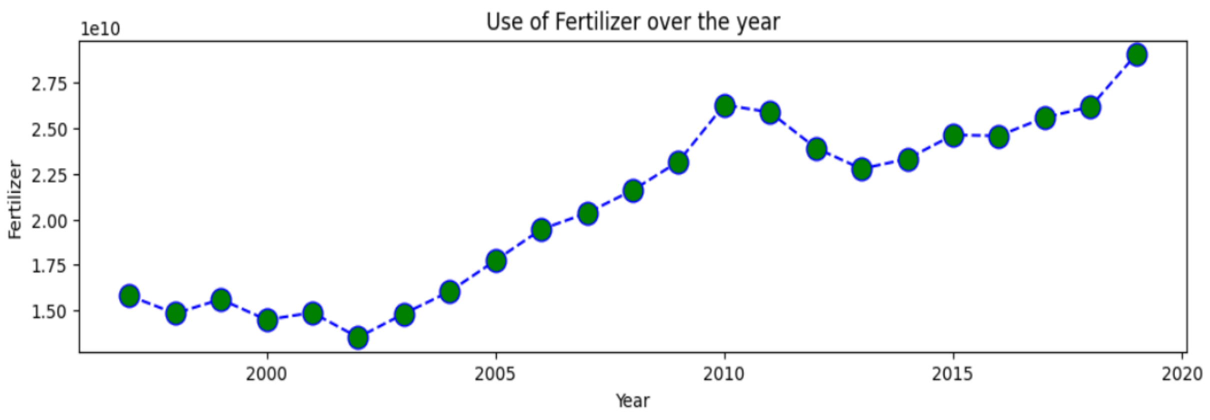
4.1.2.3. Area under cultivation over the year



Insights:

1. Every year area of cultivating land also increases.

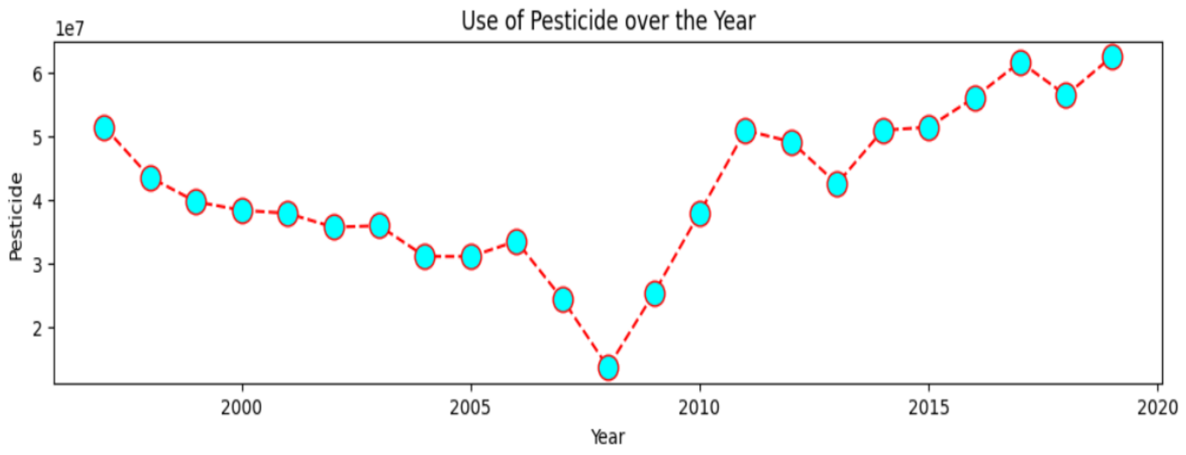
4.1.2.4. Uses Of Fertilizer Over the Year



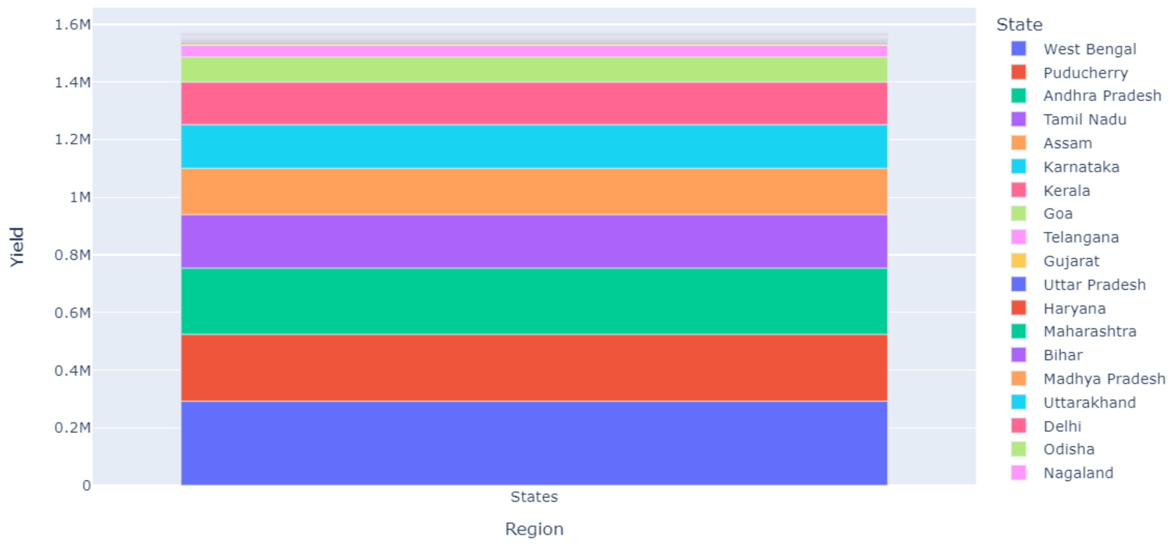
Insights:

1. Uses of Fertilizer increases every year

4.1.2.5. Uses Of Pesticide Over The Year



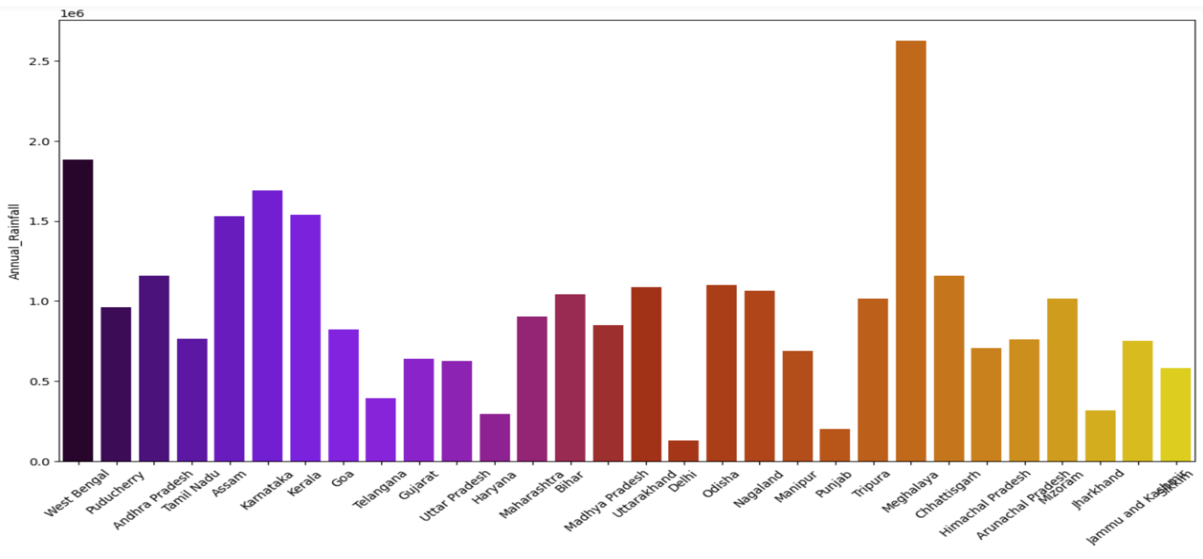
4.1.2.6. Yield range of Different States



Insights:

1. From the above graph it can be observed that the yield of West Bengal is highest. Reason can be more annual rainfall, use of fertilizers

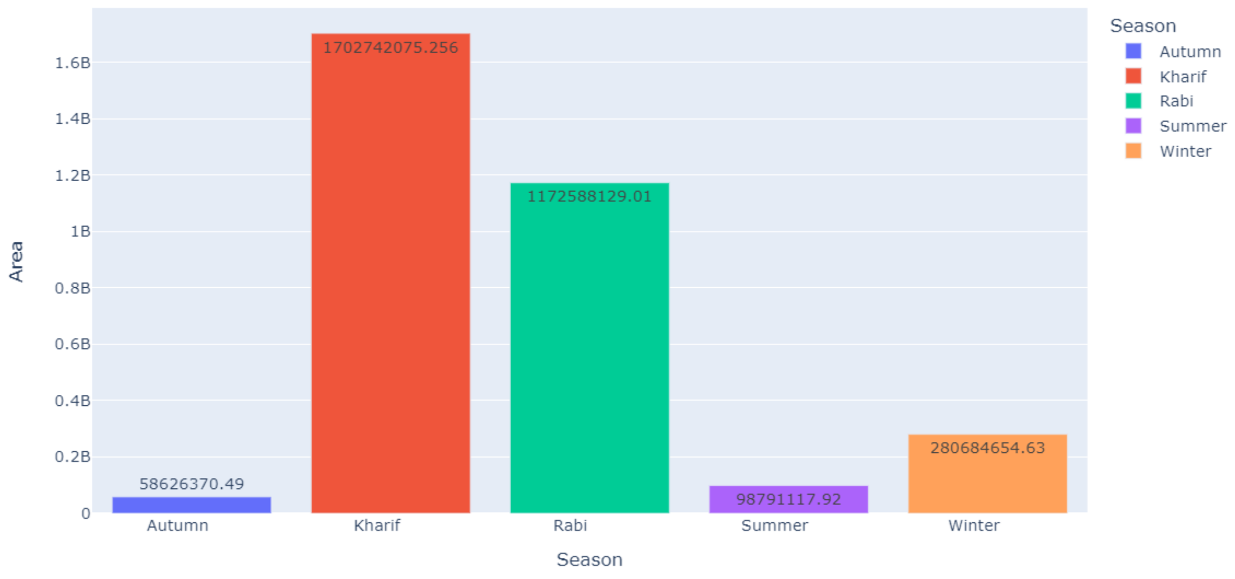
4.1.2.7. Annual Rainfall across different states



Insights:

1. From the plot it is clearly visible that, In some of state quantity of rainfall is high and in Some of state quantity of rainfall is low but in maximum state rainfall range lie between 1.0-1.5.

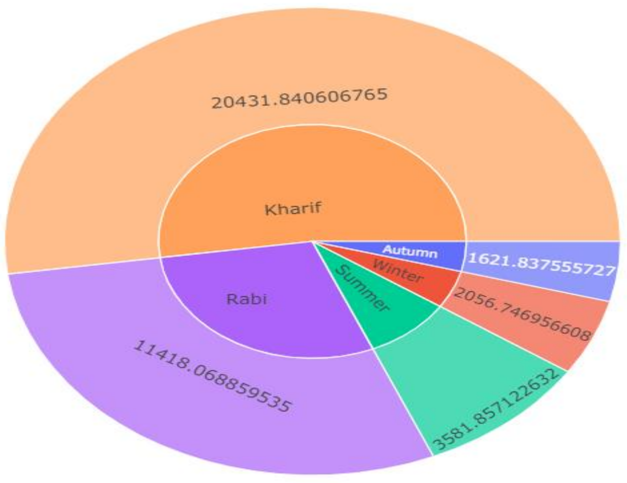
4.1.2.8. Season wise cultivate in area of field



Insights:

1. It is observed that in kharif and rabi seasons maximum area uses for cultivate.

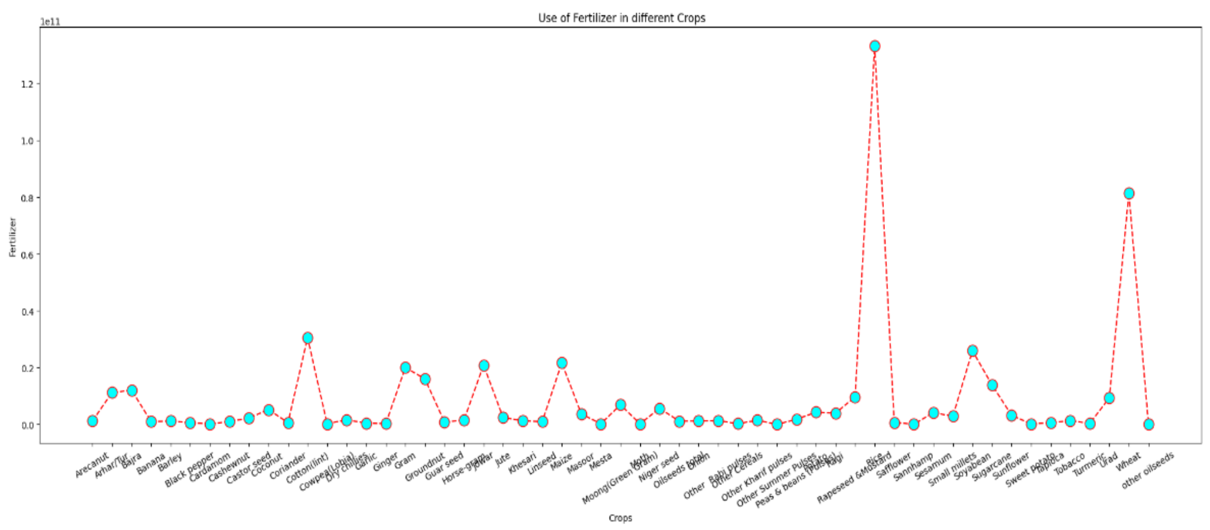
4.1.2.9. Production of crop in different seasons



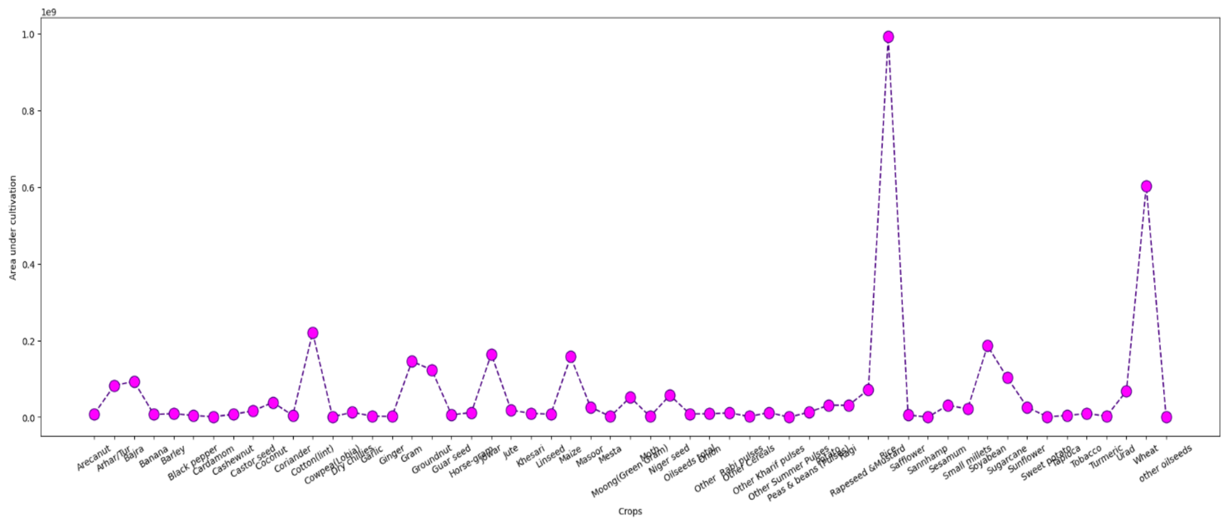
Insights:

1. It is observed that maximum crop yield in kharif and rabi seasons.

4.1.2.10. Use of fertilizer in different crops



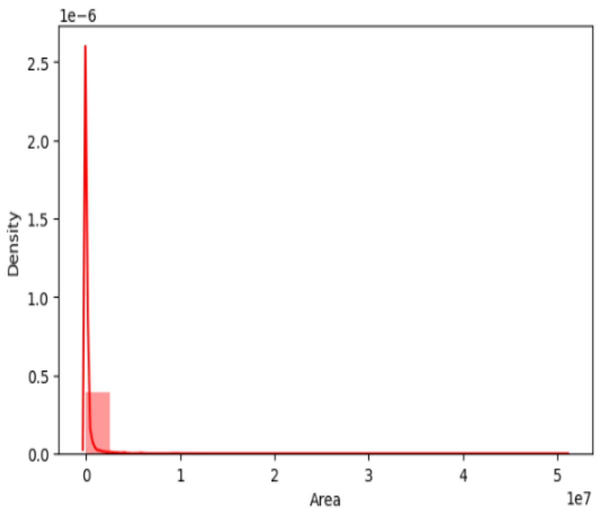
4.1.2.11. Measure of area uses for Different Crop

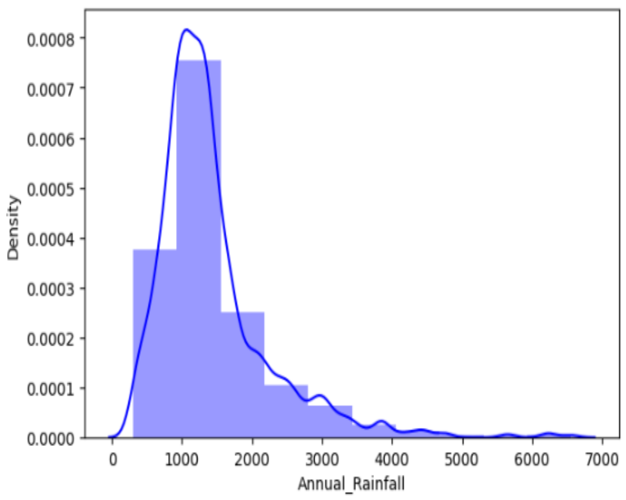


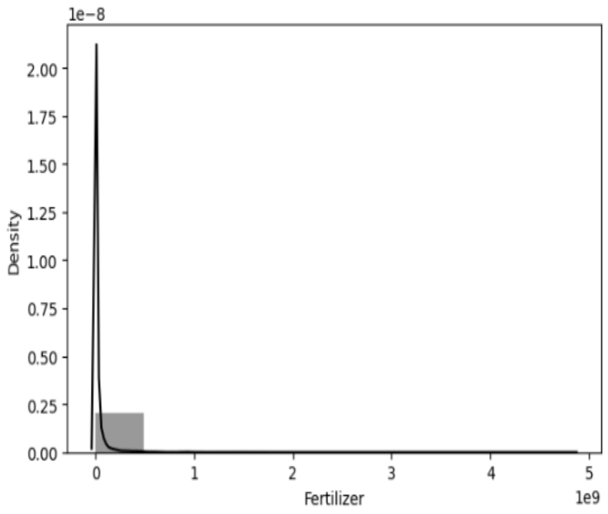
Insights:

1. It is observed that except for rice and wheat cultivation maximum area use.
2. And except rice and wheat crop other crop lie within the range 0.0-0.2

4.1.2.12. To check distribution of dataset


1e−6
Density
Area
1e7


Density
Annual_Rainfall


1e−8
Density
Fertilizer
1e9

1.

CONCLUSIONS FROM EXPLORATORY DATA ANALYSIS

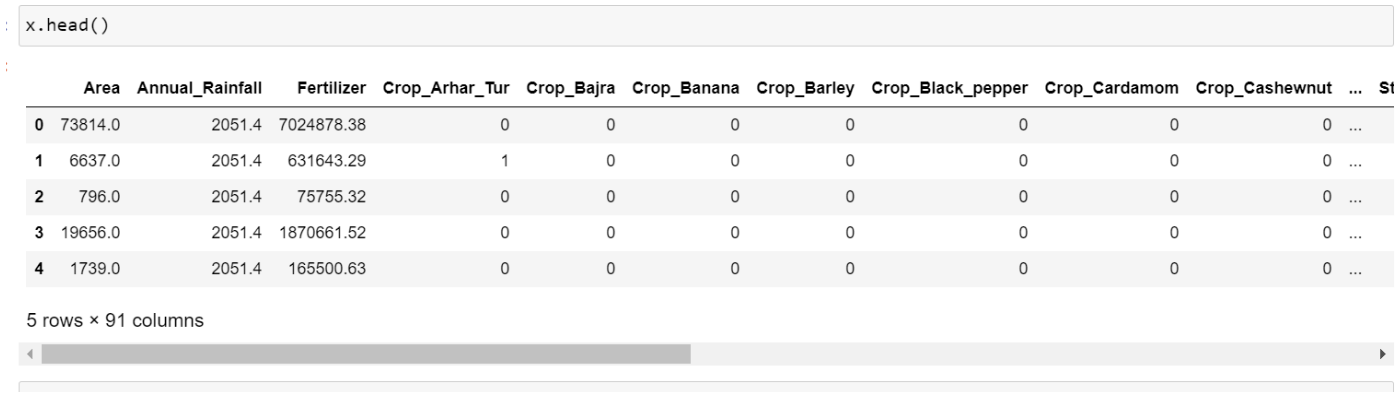
1. Historical data analysis of past crop yields within a region can contribute to predictive models, enabling better forecasting for future harvests.
2. Crop yield is significantly influenced by factors like Seasons, rainfall and fertilizer.
3. Certain crop varieties exhibit higher yields based on specific environmental conditions, such as rainfall and seasons.
4. Also uses of fertilizer can positively impacts on overall crop productivity.

5 – Model Training

5.1. Preparing the data for modelling

5.1.1 – Separating dependent and independent variables

1. Independent Variables

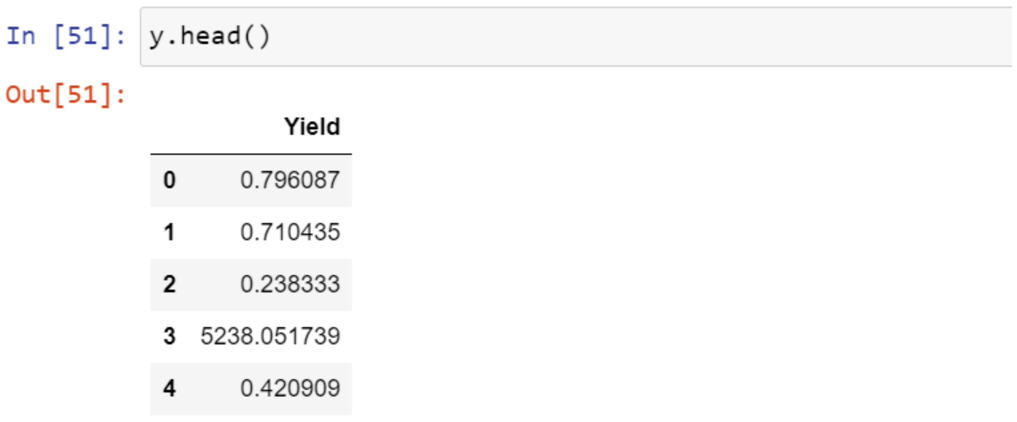


```
x.head()
```

	Area	Annual_Rainfall	Fertilizer	Crop_Arhar_Tur	Crop_Bajra	Crop_Banana	Crop_Barley	Crop_Black_pepper	Crop_Cardamom	Crop_Cashewnut	...	St
0	73814.0	2051.4	7024878.38	0	0	0	0	0	0	0	...	
1	6637.0	2051.4	631643.29	1	0	0	0	0	0	0	...	
2	796.0	2051.4	75755.32	0	0	0	0	0	0	0	...	
3	19656.0	2051.4	1870661.52	0	0	0	0	0	0	0	...	
4	1739.0	2051.4	165500.63	0	0	0	0	0	0	0	...	

5 rows × 91 columns

2. Dependent variables (Target Variable)



```
In [51]: y.head()
Out[51]:
```

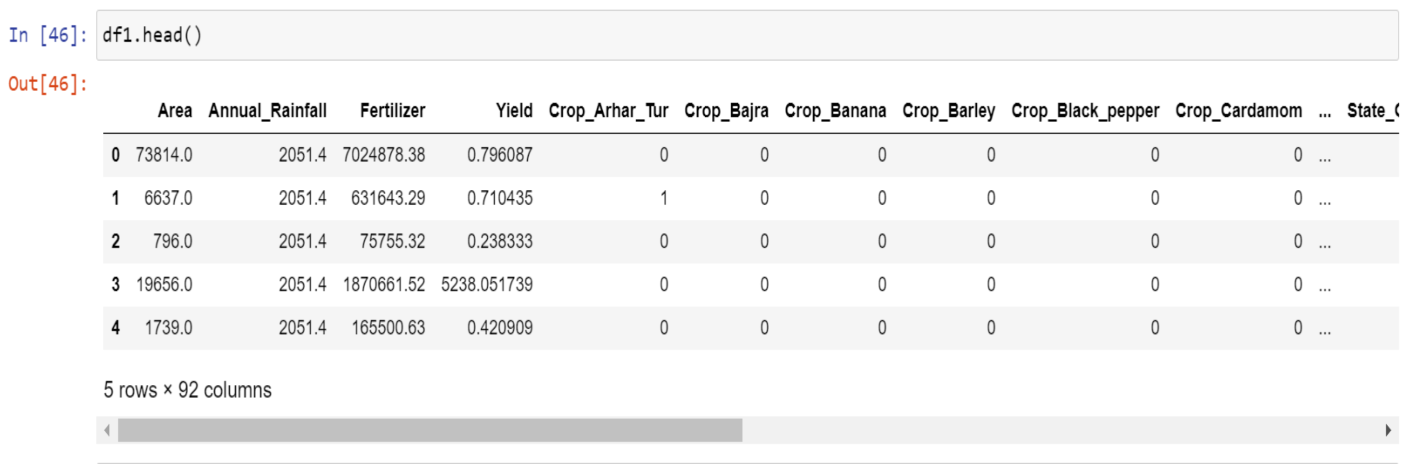
	Yield
0	0.796087
1	0.710435
2	0.238333
3	5238.051739
4	0.420909

5.1.3. Feature Engineering

1. As our categorical variables have string values, which our model does not understand. So by using OneHotEncoder we convert our categorical variables into numerical variables.

2. As the values of our numerical features are in range 50-100 and this does not match with the numerical values of the other variables. So to fix this problem, we scale-up or scale-down our data by using StandardScaler. It helps in improving the accuracy of our model.

3. After Conversion the data look like this:



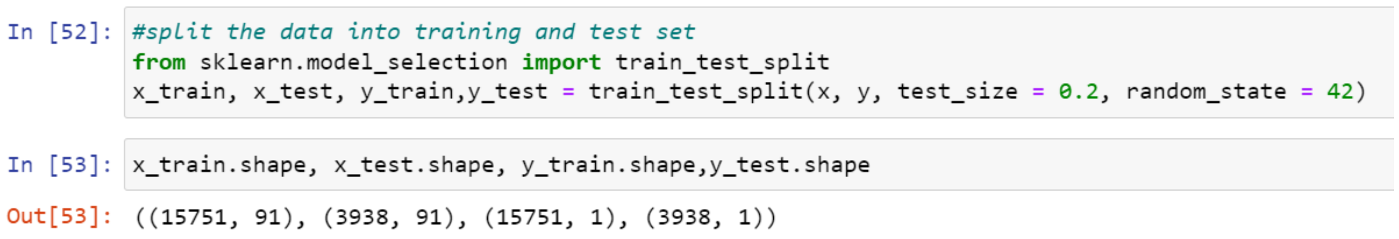
In [46]: `df1.head()`

Out[46]:

	Area	Annual_Rainfall	Fertilizer	Yield	Crop_Arhar_Tur	Crop_Bajra	Crop_Banana	Crop_Barley	Crop_Black_pepper	Crop_Cardamom	...	State_
0	73814.0	2051.4	7024878.38	0.796087	0	0	0	0	0	0	...	
1	6637.0	2051.4	631643.29	0.710435	1	0	0	0	0	0	...	
2	796.0	2051.4	75755.32	0.238333	0	0	0	0	0	0	...	
3	19656.0	2051.4	1870661.52	5238.051739	0	0	0	0	0	0	...	
4	1739.0	2051.4	165500.63	0.420909	0	0	0	0	0	0	...	

5 rows × 92 columns

5.1.4 – Splitting training and testing data



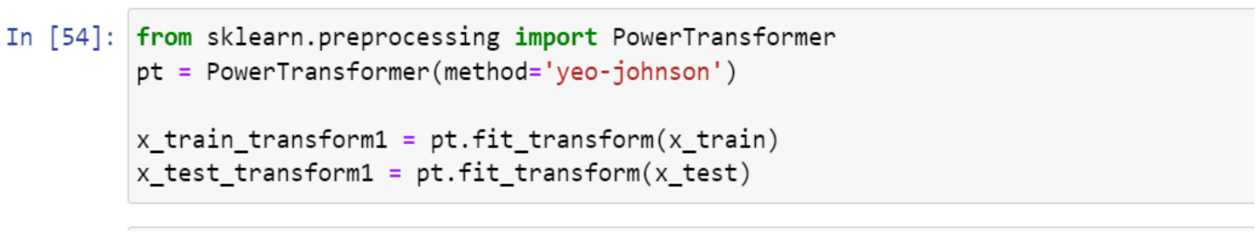
In [52]:
```python
#split the data into training and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train,y_test = train_test_split(x, y, test_size = 0.2, random_state = 42)
```

In [53]:
```python
x_train.shape, x_test.shape, y_train.shape,y_test.shape
```

Out[53]:
```
((15751, 91), (3938, 91), (15751, 1), (3938, 1))
```

1. The dataset of 19689 rows is split into a training set of 15751 rows and a testing set of 3937 rows.
2. The training will be used for training the model, and the testing set will be used for testing the accuracy of the model.

5.2. Model testing and Evaluation Function



In [54]:
```python
from sklearn.preprocessing import PowerTransformer
pt = PowerTransformer(method='yeo-johnson')

x_train_transform1 = pt.fit_transform(x_train)
x_test_transform1 = pt.fit_transform(x_test)
```

5.2.1. Description:

In model testing code, first we define a list where we initialize all of our machine learning algorithms on which we will be testing out the accuracy of the model using the evaluate method. In the evaluate method we have used r2_score method :

The evaluate function takes in the actual values and the predicted values as the parameters and returns the r2_score. These values will tell the accuracy of the model.

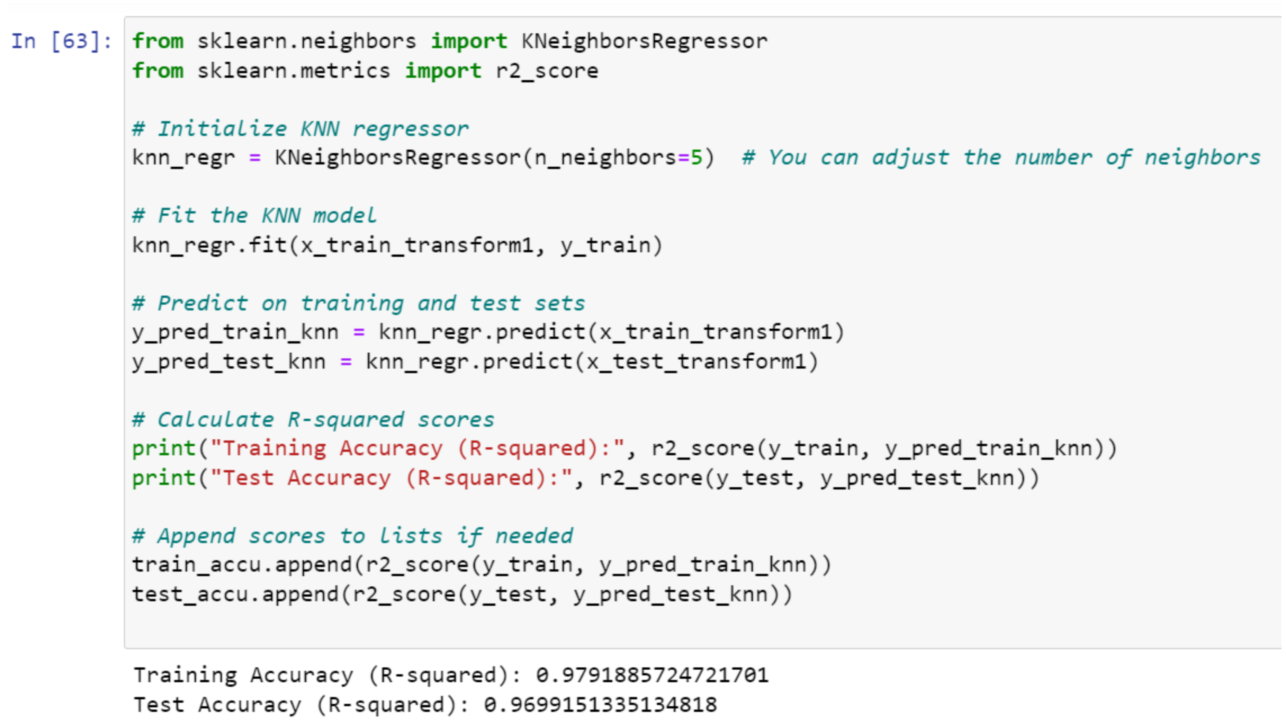
In the model testing code, we loop through the list of all algorithms, use the predict function on each of the algorithm and obtain a set of predicted values. These predicted values will be given to the evaluate function along with the actual values to get the accuracy of the model. It will give the following output.

In [59]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

lr = LinearRegression()
lr.fit(x_train,y_train)

y_pred_train = lr.predict(x_train)
print("Training Accuracy : ",r2_score(y_train,y_pred_train))

y_pred_test = lr.predict(x_test)
print("Test Accuracy : ",r2_score(y_test,y_pred_test))
```

```
Training Accuracy :  0.8513920002599723
Test Accuracy :  0.8104377403968901
```

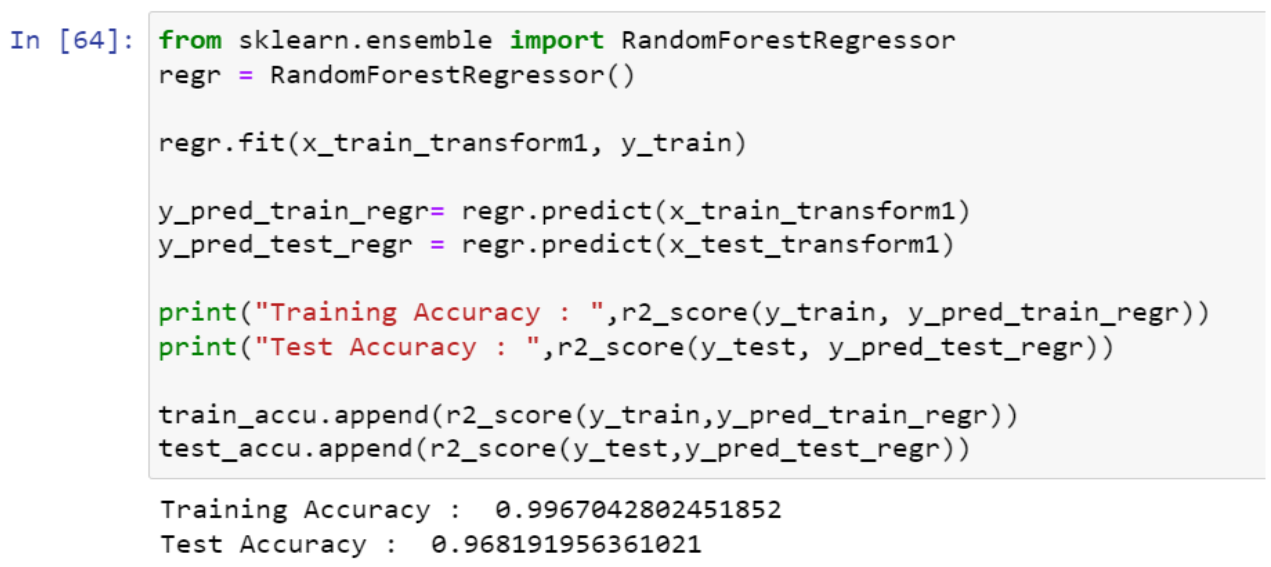
In [62]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score

# Create a Decision Tree Regressor
tree_regr = DecisionTreeRegressor()

# Fit the model
tree_regr.fit(x_train_transform1, y_train)

# Make predictions
y_pred_train_tree = tree_regr.predict(x_train_transform1)
y_pred_test_tree = tree_regr.predict(x_test_transform1)

# Calculate R2 scores for training and test sets
train_accuracy_tree = r2_score(y_train, y_pred_train_tree)
test_accuracy_tree = r2_score(y_test, y_pred_test_tree)

print("Training Accuracy (Decision Tree): ", train_accuracy_tree)
print("Test Accuracy (Decision Tree): ", test_accuracy_tree)
train_accu.append(r2_score(y_train, y_pred_train_tree))
test_accu.append(r2_score(y_test, y_pred_test_tree))
```

```
Training Accuracy (Decision Tree):  1.0
Test Accuracy (Decision Tree):  0.9426946773159396
```



```
In [63]: from sklearn.neighbors import KNeighborsRegressor
         from sklearn.metrics import r2_score

         # Initialize KNN regressor
         knn_regr = KNeighborsRegressor(n_neighbors=5)  # You can adjust the number of neighbors

         # Fit the KNN model
         knn_regr.fit(x_train_transform1, y_train)

         # Predict on training and test sets
         y_pred_train_knn = knn_regr.predict(x_train_transform1)
         y_pred_test_knn = knn_regr.predict(x_test_transform1)

         # Calculate R-squared scores
         print("Training Accuracy (R-squared):", r2_score(y_train, y_pred_train_knn))
         print("Test Accuracy (R-squared):", r2_score(y_test, y_pred_test_knn))

         # Append scores to lists if needed
         train_accu.append(r2_score(y_train, y_pred_train_knn))
         test_accu.append(r2_score(y_test, y_pred_test_knn))
```

```
Training Accuracy (R-squared): 0.9791885724721701
Test Accuracy (R-squared): 0.9699151335134818
```



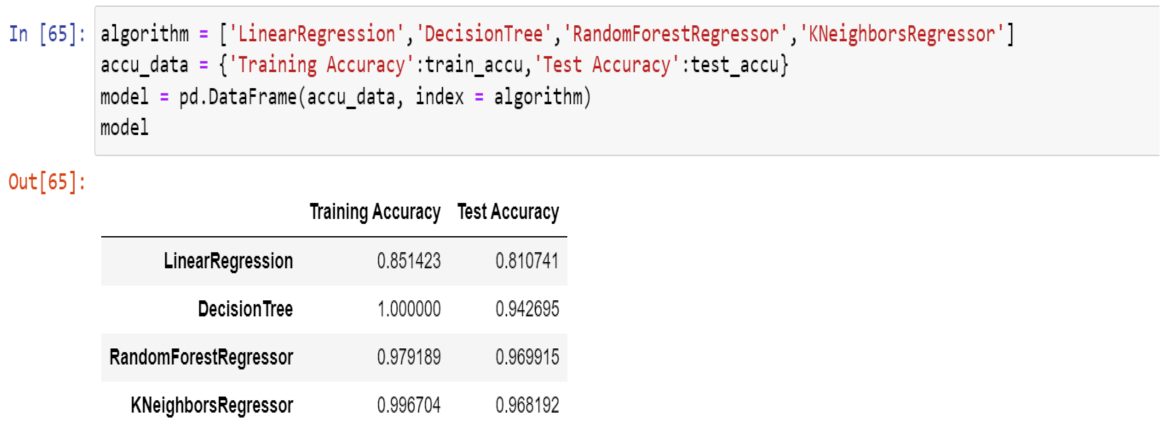
```
In [64]: from sklearn.ensemble import RandomForestRegressor
         regr = RandomForestRegressor()

         regr.fit(x_train_transform1, y_train)

         y_pred_train_regr= regr.predict(x_train_transform1)
         y_pred_test_regr = regr.predict(x_test_transform1)

         print("Training Accuracy : ",r2_score(y_train, y_pred_train_regr))
         print("Test Accuracy : ",r2_score(y_test, y_pred_test_regr))

         train_accu.append(r2_score(y_train,y_pred_train_regr))
         test_accu.append(r2_score(y_test,y_pred_test_regr))
```

```
Training Accuracy :  0.9967042802451852
Test Accuracy :  0.968191956361021
```

As we can see Random Forest gives the most efficient results:

1. r2 score - 0.96

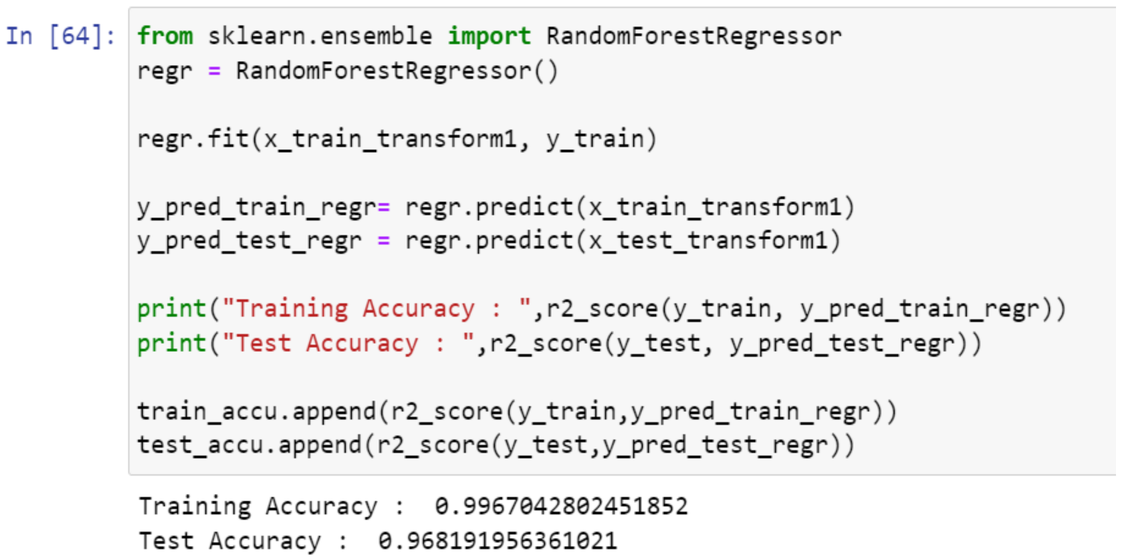
5.2.2. Results



```
In [65]: algorithm = ['LinearRegression','DecisionTree','RandomForestRegressor','KNeighborsRegressor']
         accu_data = {'Training Accuracy':train_accu,'Test Accuracy':test_accu}
         model = pd.DataFrame(accu_data, index = algorithm)
         model
Out[65]:
```

	Training Accuracy	Test Accuracy
LinearRegression	0.851423	0.810741
DecisionTree	1.000000	0.942695
RandomForestRegressor	0.979189	0.969915
KNeighborsRegressor	0.996704	0.968192

So, from all the above modelling and testing, we have found that random forest is the best suited algorithm for the dataset, which gives the most accurate prediction.



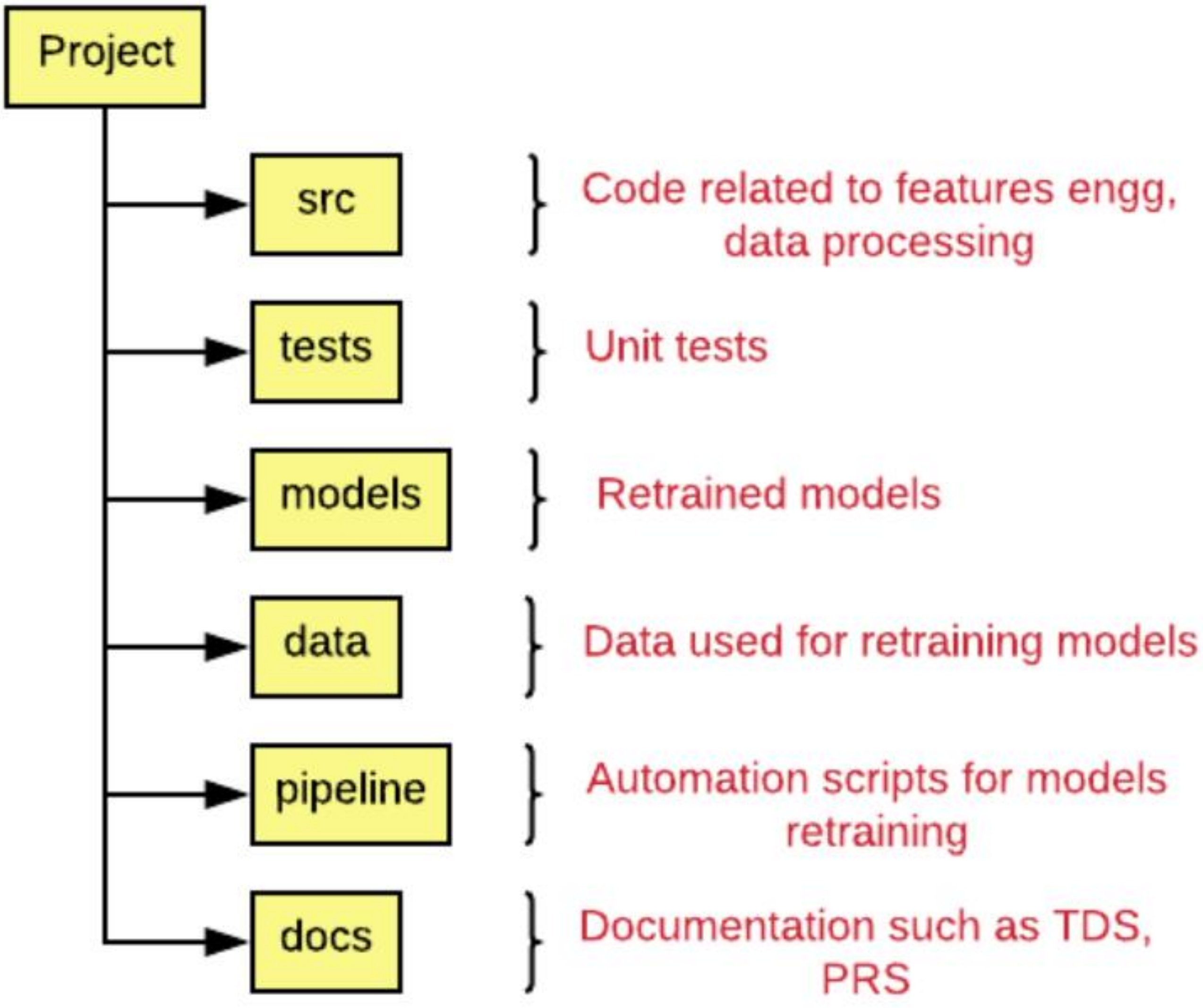
```
In [64]: from sklearn.ensemble import RandomForestRegressor
         regr = RandomForestRegressor()

         regr.fit(x_train_transform1, y_train)

         y_pred_train_regr= regr.predict(x_train_transform1)
         y_pred_test_regr = regr.predict(x_test_transform1)

         print("Training Accuracy : ",r2_score(y_train, y_pred_train_regr))
         print("Test Accuracy : ",r2_score(y_test, y_pred_test_regr))

         train_accu.append(r2_score(y_train,y_pred_train_regr))
         test_accu.append(r2_score(y_test,y_pred_test_regr))

         Training Accuracy :  0.9967042802451852
         Test Accuracy :  0.968191956361021
```

6 – Modular Project Structure

6.1. Modular Project Structure

In order to modularize the project and give it proper structure, we follow the following rules

6.1.1. Folder Structure


Project
src
Code related to features engg, data processing
tests
Unit tests
models
Retrained models
data
Data used for retraining models
pipeline
Automation scripts for models retraining
docs
Documentation such as TDS, PRS

The following are the details of the above-mentioned folder structure:

1. **project_name:** Name of the project.

2. **src**: The folder that consists of the source code related to data gathering, data preparation, feature extraction, etc.

3. **tests:** The folder that consists of the code representing unit tests for code maintained with the src folder.

4. **models:** The folder that consists of files representing trained/retrained models as part of build jobs, etc. The model names can be appropriately set as project name date time or project build id (in case the model is created as part of build jobs). Another approach is to store the model files in a separate storage such as AWS S3, Google Cloud Storage, or any other

form of storage.

5. **data**: The folder consists of data used for model training/retraining. The data could also be stored in a separate storage system.

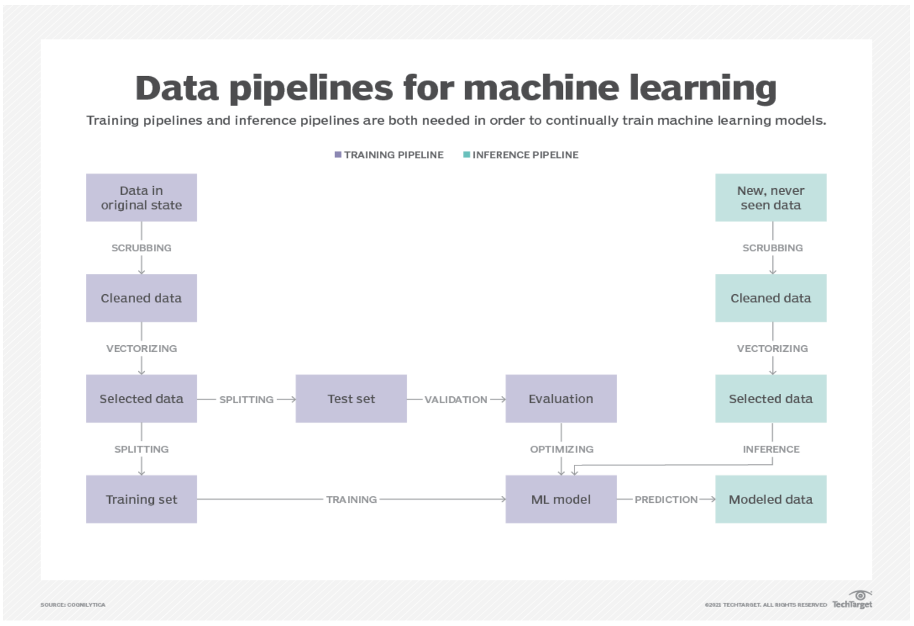
6. **pipeline:** The folder consists of code that's used for retraining and testing the model in an automated manner. These could be docker containers related code, scripts, workflow related code, etc.

7. **docs:** The folder that consists of code related to the product requirement specifications (PRS), technical design specifications (TDS), etc.

6.2. Data Pipelining

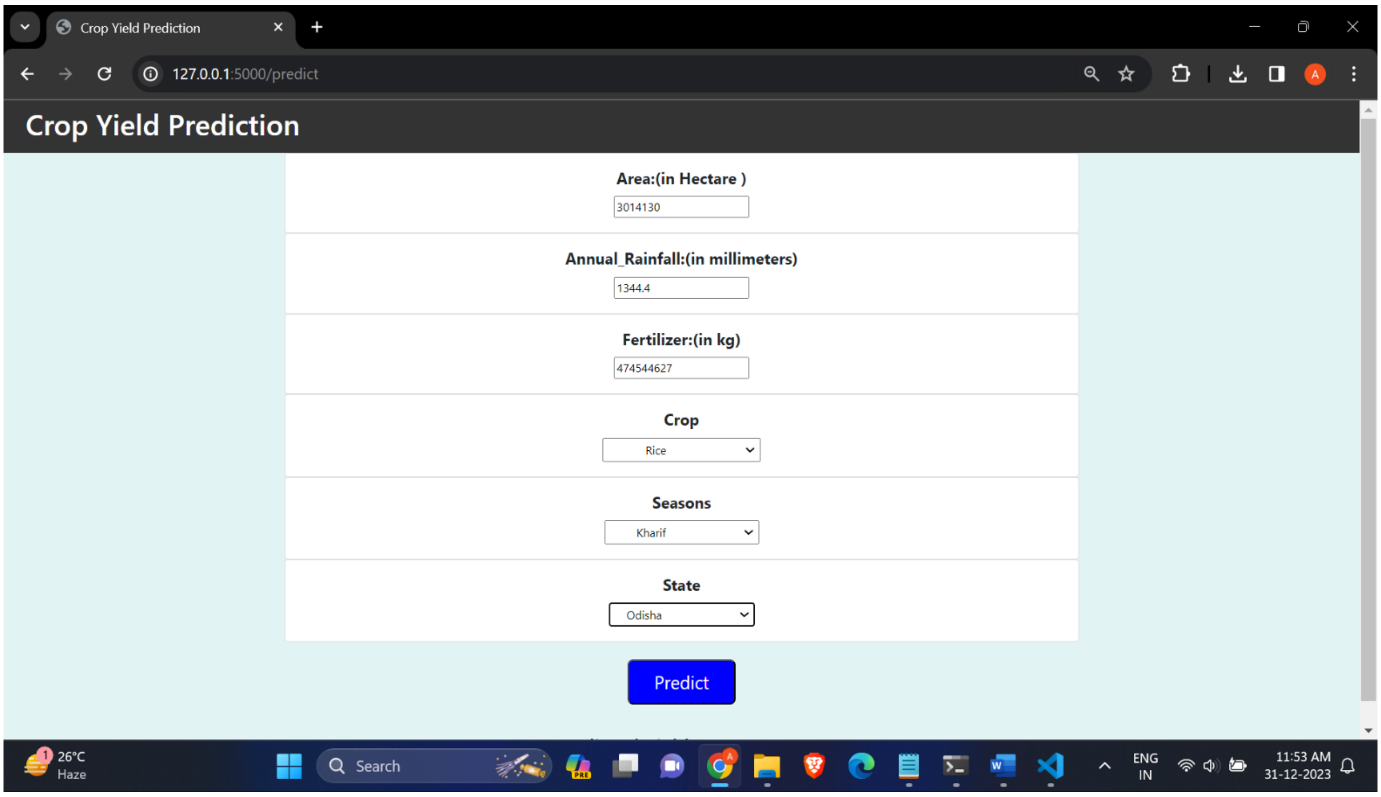
Data pipelines operate on the same principle; only they deal with information rather than liquids or gasses. Data pipelines are a sequence of data processing steps, many of them accomplished with special software. The pipeline defines how, what, and where the data is collected. Data pipelining automates data extraction, transformation, validation, and combination, then loads it for further analysis and visualization. The entire pipeline provides speed from one end to the other by eliminating errors and neutralizing bottlenecks or latency.

In our modular project, we have used the concept of pipelining to take the user input from the UI and sort of tunneled the data through a pipeline to our model. And in this way predict the data.



7 – User View

The UI is built using HTML and CSS which is Bundled up in a local server using Flask.



Crop Yield Prediction

127.0.0.1:5000/predict

Crop Yield Prediction

Area:(in Hectare) 3014130

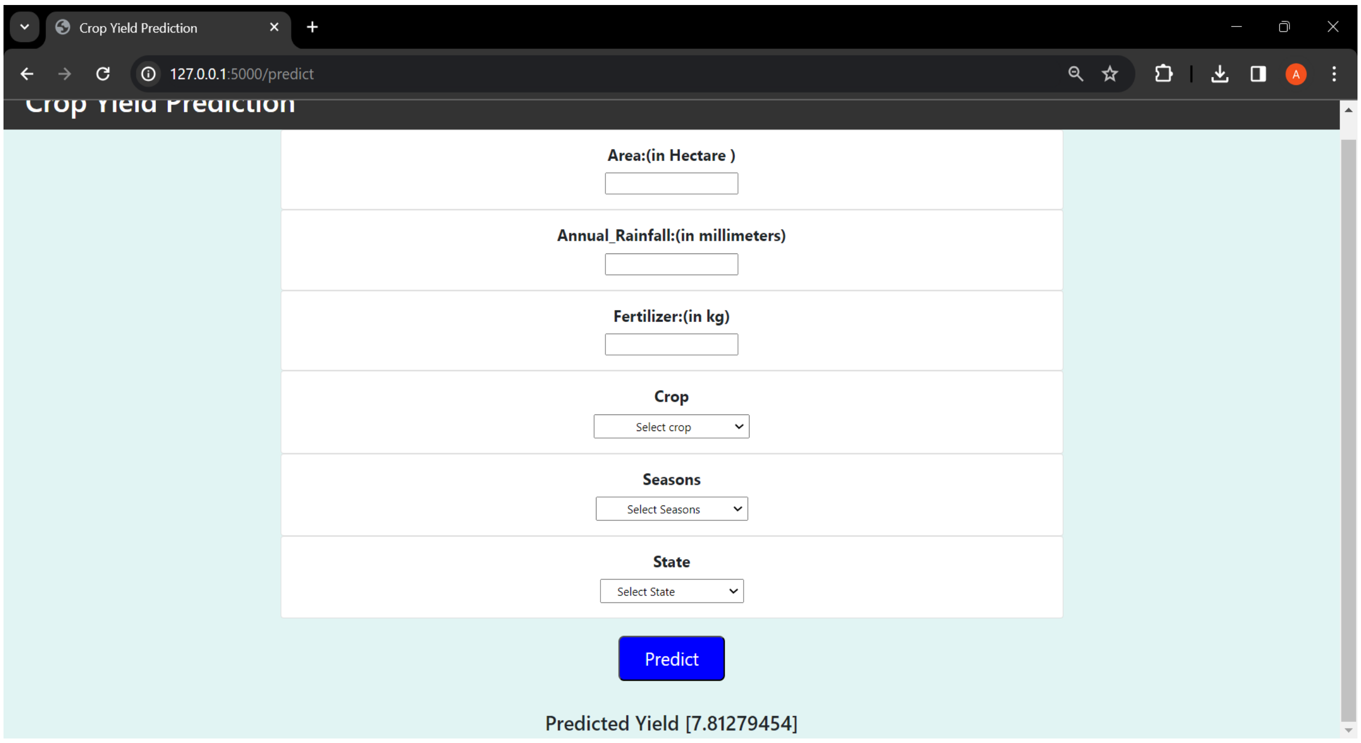
Annual_Rainfall:(in millimeters) 1344.4

Fertilizer:(in kg) 474544627

Crop: Rice

Seasons: Kharif

State: Odisha



Crop Yield Prediction

127.0.0.1:5000/predict

Crop Yield Prediction

Area:(in Hectare)

Annual_Rainfall:(in millimeters)

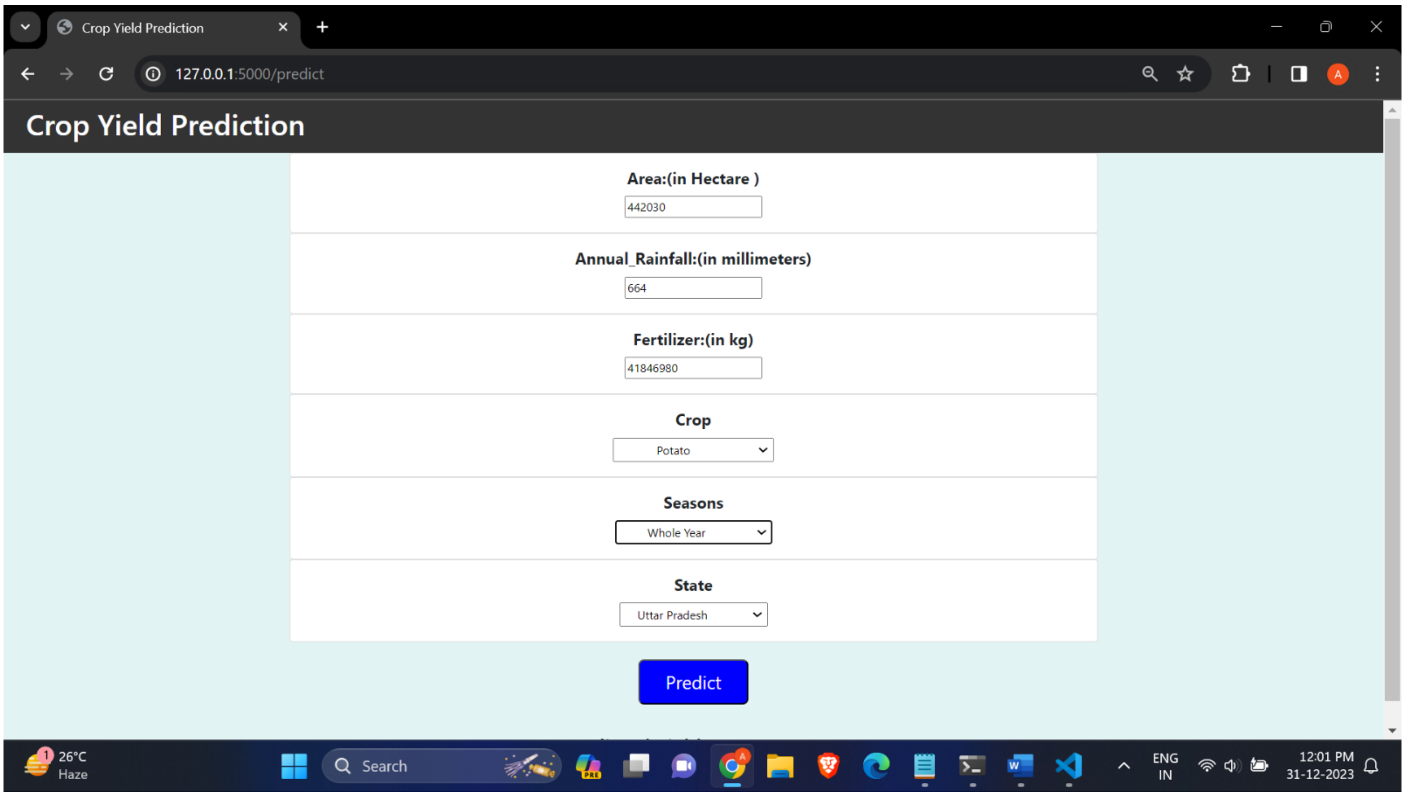
Fertilizer:(in kg)

Crop: Select crop

Seasons: Select Seasons

State: Select State

Predicted Yield [7.81279454]


Crop Yield Prediction

127.0.0.1:5000/predict

Crop Yield Prediction

Area:(in Hectare) 442030

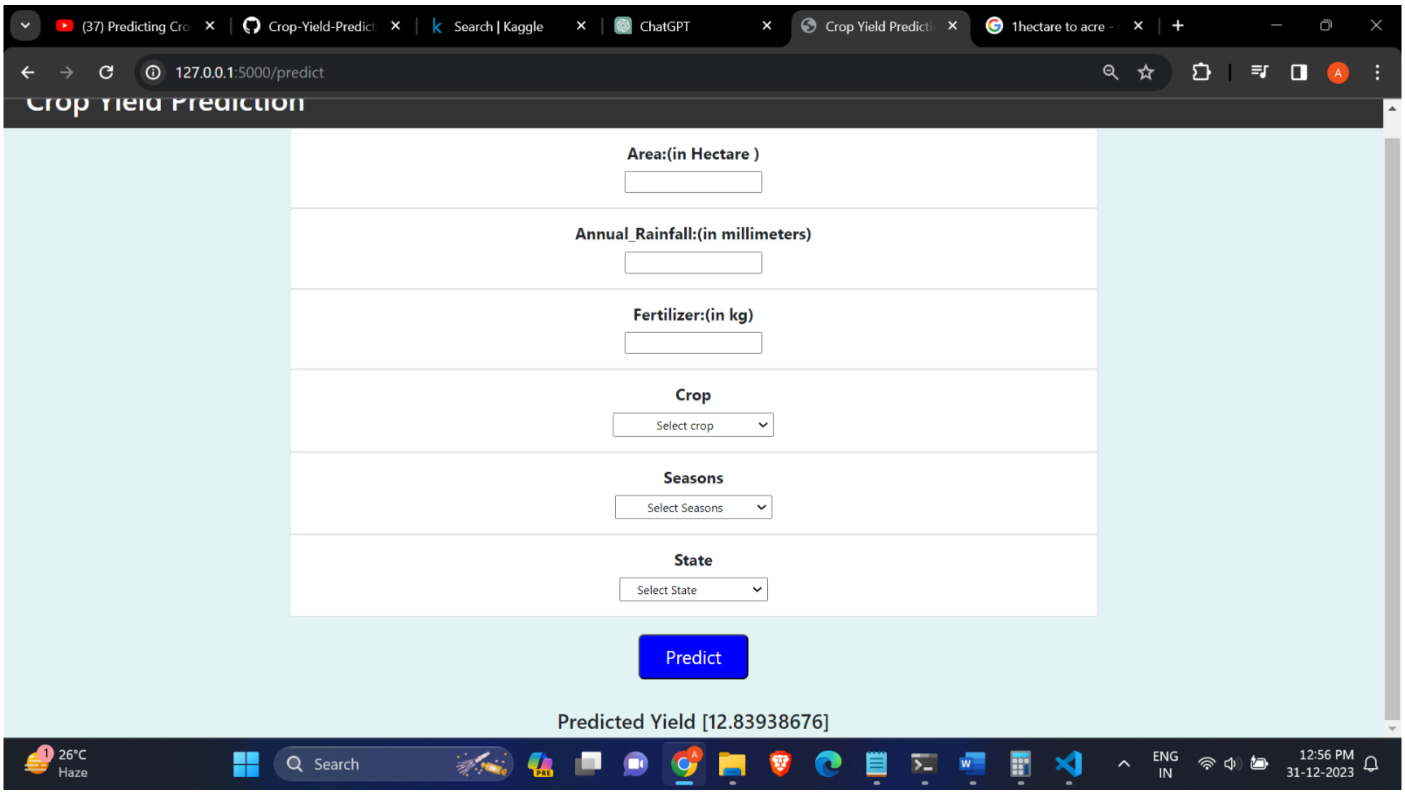
Annual_Rainfall:(in millimeters) 664

Fertilizer:(in kg) 41846980

Crop Potato

Seasons Whole Year

State Uttar Pradesh


127.0.0.1:5000/predict

Crop Yield Prediction

Area:(in Hectare)

Annual_Rainfall:(in millimeters)

Fertilizer:(in kg)

Crop Select crop

Seasons Select Seasons

State Select State

Predicted Yield [12.83938676]

7. Conclusion

This project endeavors to bridge the gap in empirical crop yield prediction by surpassing existing regression techniques. Leveraging grassroots-level data, it provides an accurate portrayal of on-ground realities, crucial for improving our understanding of agricultural productivity. By addressing these fundamental needs, we aim to elevate agricultural standards, thereby enhancing the lively-hoods of farmers.

Understanding crop performance beforehand is vital to support at-risk crops by preemptively addressing challenges they face, facilitating their growth and optimizing agricultural processes. Predicting crop yields, especially for newer or smaller datasets, presents a challenge, yet our outcomes showcase promising accuracy levels, indicating the potential for reliable predictions even with limited records.

Our model's success is attributed to the data's linear nature, allowing effective utilization of linear regression. However, with more complex agricultural dynamics, future endeavors might necessitate employing algorithms like random forest regressors or decision tree regressors. Adapting to such complexities will be crucial for advancing crop yield prediction methodologies.

8. References

1. Fox, J. , Applied Regression Analysis, Linear Models, and Related Methods, ISBN: 080394540X, Sage Pubns.

2. Machine Learning, Tom Mitchell, McGraw Hill, 1997

3. Sujatha, R., & Isakki, P. (2022, January). A study on crop yield forecasting using classification techniques. In 2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16) (pp. 1-4). IEEE.

4. Veenadhari, S., Misra, B., & Singh, C. D. (2020, January). Machine learning approach for forecasting crop yield based on climatic parameters. In 2014 International Conference on Computer Communication and Informatics (pp. 1-5). IEEE

5. Manjula, A., & Narsimha, G. (2019, January). XCYPF: A flexible and extensible framework for agricultural Crop Yield Prediction. In 2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO) (pp. 1-5). IEEE.

6. Bang, S., Bishnoi, R., Chauhan, A. S., Dixit, A. K., & Chawla, I. (2019, August). Fuzzy Logic based Crop Yield Prediction using Temperature and Rainfall parameters predicted through ARMA, SARIMA, and ARMAX models. In 2019 Twelfth International Conference on Contemporary Computing (IC3) (pp. 1-6). IEEE.

7. Mariappan, A. K., & Das, J. A. B. (2017, April). A paradigm for rice yield prediction in Tamilnadu. In 2017 IEEE Technological Innovations in ICT for Agriculture and Rural Development (TIAR) (pp. 18-21). IEEE.

8. Javadinejad, S.; Eslamian, S.; Askari, K.O.A. The analysis of the most important climatic parameters affecting performance of crop variability in a changing climate. Int. J. Hydrol. Sci. Technol. 2021, 11, 1–25. [Google Scholar] [CrossRef]