Industrial Internship Report on

”Prediction of Agriculture Crop Production in India”

Prepared by

Name: Pratyush Kishan

Branch: Computer Science

Semester: 6th

Mobile Number: 9708198009

Email id: pkrockzz2002@gmail.com

College Name: Birla Institute Of Technology, Mesra, Patna Campus

Executive Summary
This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT). This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks’ time. My project was based on **Data Science and Machine Learning** any my assigned project was **“Prediction of Agriculture Crop Production in India”** This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship.

TABLE OF CONTENTS

1 Preface

I was thrilled to begin this internship and continued my energy and determination to complete this internship and I completed all the tasks given to me on weekly basis.

In the first week I was assigned the task to explore about the company, related to UCT means , about their domain, the technologies they use, major achievements, etc., and I was also given a task related to project selection according to my own interest from the various project lists that they provided me, after that I explored a lot about that and selected agriculture related project.

In the second week, I learned about lots of machine learning algorithms and selected the appropriate ones for my project work.

In the third week, I started the implementation part according to strategy that I had made and also made some use case, system architecture.

In the fourth week, I continued my work on implementation part and made a machine learning model using the dataset provided by the company.

In the fifth week, I wanted to improve the accuracy, so I tried with different dataset and ultimately, I improved the model's performance and its accuracy. I also built second model for another prediction.

In the sixth week, I compiled all the work and made the final report of my project that was all about the summary of my works of 6 weeks.

Relevant Internship play a significant role in career development for many reasons like:

1. Gain Practical Experience 2. Skill Development 3. Help me to connect with various people of my professional background 4. Resume Enhancement 5. Help me in personal growth.

My project/problem statement was **Prediction of Agriculture Crop Production in India**

Context

Agriculture Production in India from 2001-2014

Content

First Dataset describes the weather conditions and soil conditions for agriculture in India. This is from

https://drive.google.com/drive/folders/1jLGvYRj1KxvyjktBRwD3z2Dq14FP2eU?usp=drive_link

Second Dataset Describes the Agriculture Crops Cultivation/Production in India. This is

from https://data.gov.in/ fully Licensed Acknowledgements

These Dataset can solve the problems of various crops Cultivation/production in India.

Columns

Crop name: string, Crop: string, State name: string, Nitrogen: Integer, Phosphorus: Integer, Potassium: Integer, rainfall: Integer, Temperature: Integer, pH: Integer, Area: Integer, Production: Integer, Yield: Integer

state: string, Crops Cultivation/production Place Quantity: Integer, no of Quintals/Hectares production: Integer

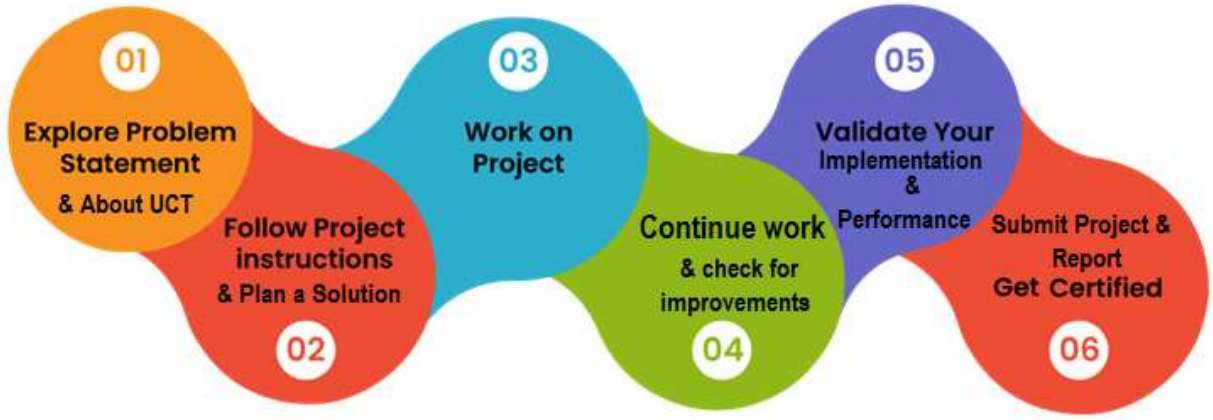
Inspiration

Across the Globe, India Is the Second Largest Country having People more than 1.3 Billion. Many People are dependent on the Agriculture and it is the Main Resource.
In Agriculture, Cultivation and Production is the main problem.
I want to solve the Big problem in India which will be useful to many more people .

Data set Link:

https://drive.google.com/drive/folders/1jLGvYRj1KxvyjktBRwD3z2Dq14FP-2eU

https://drive.google.com/file/d/1zfqvs8-mAO6E0JpgvhBdueNx8Th03pUp/view? usp=sharing



Thanks for the Opportunity given by USC/UCT. I learned lots of things from upskill campus, UCT and mentors through this internship program. I gained practical experience from this internship and got lot of exposure from the members of UCT. My overall experience is very good with upskill campus and UCT.

Thanks to all (upskill campus, UCT, Kaushlendra Singh Sisodia sir, coordinators, mentors) for helping me directly or indirectly in this internship journey of 42 days. I was lucky to get the support from you all.

Once again Thank You All of you for bringing new changes in my life/career.

Dear juniors and peers,

I wanted to share my thoughts and experiences as an intern, hoping to provide you with some valuable insights and encouragement for your own career journeys. First and foremost, embrace the opportunities that internships offer. They are stepping stones towards your professional growth and development. Treat your internships as valuable learning experiences where you can apply your academic knowledge in real -world settings. Networking is key during your internship. Take advantage of the connections you make with your supervisors, colleagues and industry professionals. These connections can lead to mentorship opportunities, potential job offers, and a broader professional network.

Wishing you all the best in your internships and beyond. Embrace the learning, grab the opportunities, and let your internships be stepping stones towards a fulfilling and successful career.

Best regards,

Pratyush Kishan

2 Introduction

2.1 About UniConverge Technologies Pvt Ltd

A company established in 2013 and working in Digital Transformation domain and providing Industrial solutions with prime focus on sustainability and RoI.

For developing its products and solutions it is leveraging various **Cutting Edge Technologies e.g. Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end** etc.

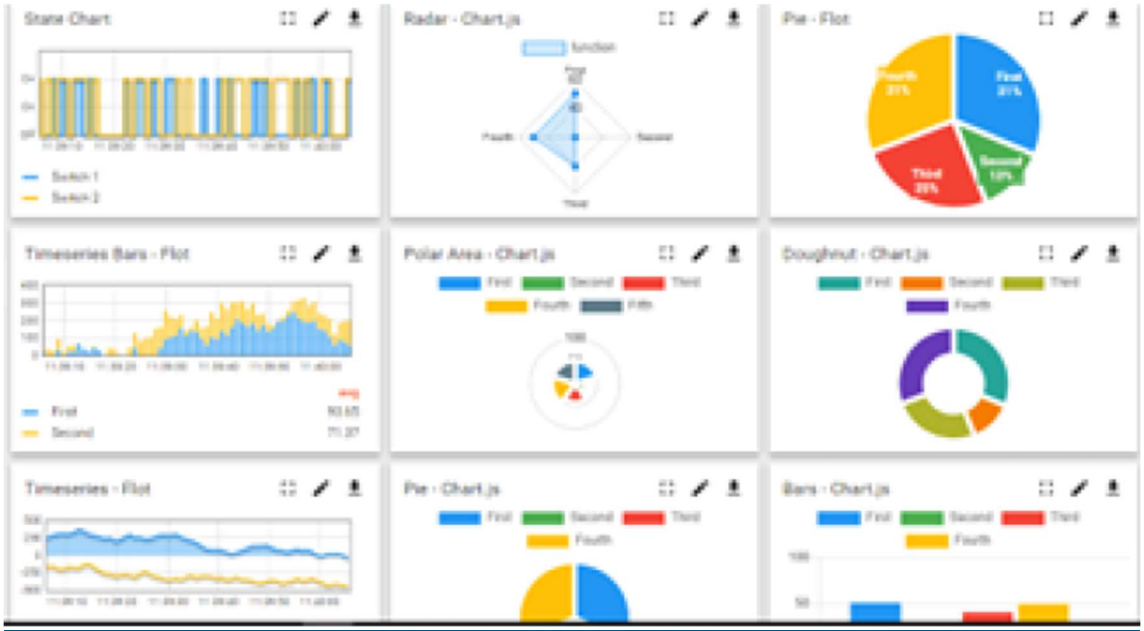
i. UCT IoT Platform (UCT Insight)

UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable "insight" for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA
- It supports both cloud and on-premises deployments.

It has features to

- Build Your own dashboard
- Analytics and Reporting
- Alert and Notification
- Integration with third party application(Power BI, SAP, ERP)
- Rule Engine



ii. Smart Factory Platform (FACTORY WATCH)

Factory watch is a platform for smart factory needs.

It provides Users/ Factory

- with a scalable solution for their Production and asset monitoring
- OEE and predictive maintenance solution scaling up to digital twin for your assets.
- to unleased the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
- A modular architecture that allows users to choose the service that they what to start and then can scale to more complex solutions as per their demands.

Its unique SaaS model helps users to save time, cost and money.

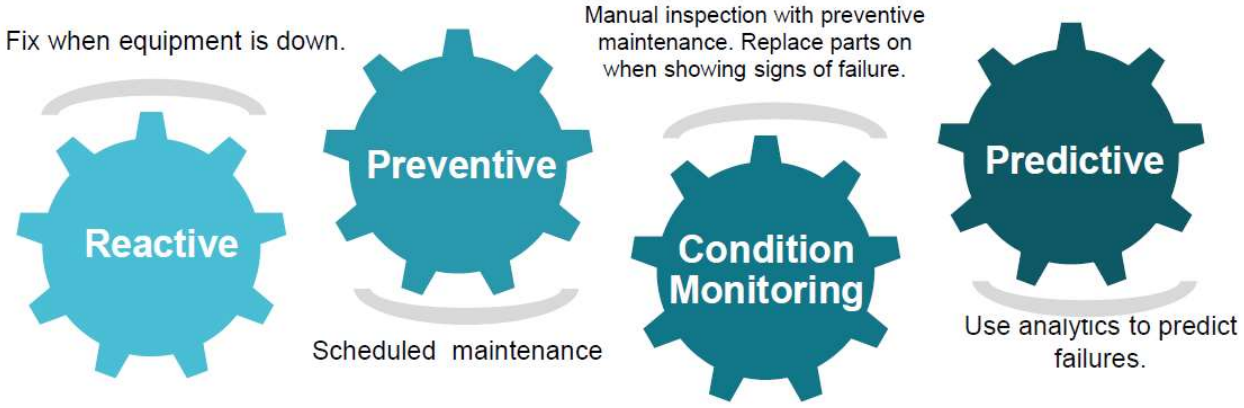
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i

iii. LoRaWAN based Solution

UCT is one of the early adopters of LoRAWAN teschnology and providing solution in Agritech, Smart cities, Industrial Monitoring, Smart Street Light, Smart Water/ Gas/ Electricity metering solutions etc.

iv. Predictive Maintenance

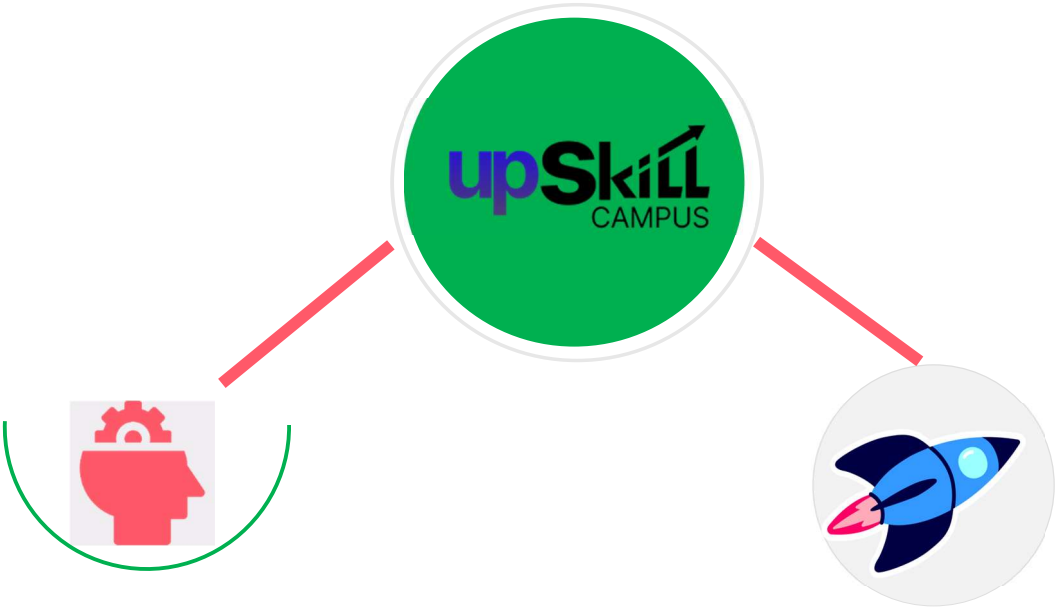
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



2.2 About upskill Campus (USC)

upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.

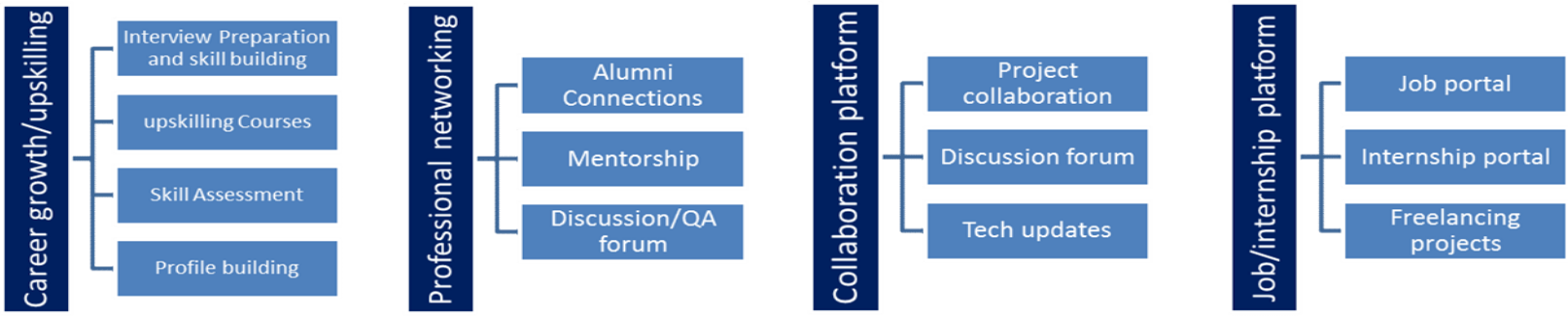
USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



Seeing need of upskilling in self paced manner along-with additional support services e.g. Internship, projects, interaction with Industry experts, Career growth Services

upSkill Campus aiming to upskill 1 million learners in next 5 year

https://www.upskillcampus.com/



2.3 The IoT Academy

The IoT academy is EdTech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

2.4 Objectives of this Internship program

The objective for this internship program was to

- ☛ get practical experience of working in the industry.
- ☛ to solve real world problems.
- ☛ to have improved job prospects.
- ☛ to have Improved understanding of our field and its applications.
- ☛ to have Personal growth like better communication and problem solving.

2.5 Reference

[1] Google

[2] Youtube

[3] JavaTPoint

[4] GeeksForGeeks

2.6 Glossary

Terms	Acronym
accuracy	The number of correct classification **predictions** divided by the total number of predictions
bagging	A method to **train** an **ensemble** where each constituent **model** trains on a random subset of training examples **sampled** *with replacement*
boosting	A machine learning technique that iteratively combines a set of simple and not very accurate classifiers (referred to as "weak" classifiers) into a classifier with high accuracy (a "strong" classifier) by **upweighting** the examples that the model is currently misclassifying
categorical data	Categorical features are sometimes called **discrete features**
confusion matrix	An NxN table that summarizes the number of correct and incorrect predictions that a **classification model** made.
data analysis	Obtaining an understanding of data by considering samples, measurement, and visualization. Data analysis can be particularly useful when a dataset is first received, before one builds the first **model**
data set or dataset	A collection of raw data, commonly (but not exclusively) organized in one of the following formats: •a spreadsheet •a file in CSV (comma-separated values) format
decision tree	A supervised learning model composed of a set of **conditions** and **leaves** organized hierarchically
gradient boosting	A training algorithm where weak models are trained to iteratively improve the quality (reduce the loss) of a strong model. For example, a weak model could be a linear or small decision tree model. The strong model becomes the sum of all the previously trained weak models.
matplotlib	An open-source Python 2D plotting library. ***matplotlib*** helps you visualize different aspects of machine learning

3 Problem Statement

My project/problem statement was **Prediction of Agriculture Crop Production in India**

Context

Agriculture Production in India from 2001-2014

Content

First Dataset describes the weather conditions and soil conditions for agriculture in India. This is from

https://drive.google.com/drive/folders/1jLGvYRj1KxvyjktBRwD3z2Dq14FP-2eU

Second Dataset Describes the Agriculture Crops Cultivation/Production in India. This is

from https://data.gov.in/ fully Licensed Acknowledgements

These Dataset can solve the problems of various crops Cultivation/production in India.

Columns

Crop name: string, Crop: string, State name: string, Nitrogen: Integer, Phosphorus: Integer, Potassium: Integer, rainfall: Integer, Temperature: Integer, pH: Integer, Area: Integer, Production: Integer, Yield: Integer

state: string, Crops Cultivation/production Place Quantity: Integer, no of Quintals/Hectares production: Integer

Inspiration

Across the Globe, India Is the Second Largest Country having People more than 1.3 Billion. Many People are dependent on the Agriculture and it is the Main Resource.
In Agriculture, Cultivation and Production is the main problem.
I want to solve the Big problem in India which will be useful to many more people .

Data set Link:

https://drive.google.com/drive/folders/1jLGvYRj1KxvyjktBRwD3z2Dq14FP-2eU

https://drive.google.com/file/d/1zfqvs8-mAO6E0JpgvhBdueNx8Th03pUp/view? usp=sharing

4 Existing and Proposed solution

For most developing countries, agriculture is their primary source of revenue. Modern agriculture is a constantly growing approach for agricultural advances and farming techniques. It becomes challenging for the farmers to satisfy our planet's evolving requirements and the expectations of merchants, customers, etc. Some of the challenges the farmers face are;

(i) Dealing with climatic changes because of soil erosion and industry emissions

(ii) Nutrient deficiency in the soil, caused by a shortage of crucial minerals such as potassium, nitrogen, and phosphorus can result in reduced crop growth.

(iii) Farmers make a mistake by cultivating the same crops year after year without experimenting with different varieties. They add fertilizers randomly without understanding the inferior quality or quantity.

Agriculture gave birth to civilization. India is an agrarian country and its economy largely based upon crop productivity. Thus agriculture is the backbone of all business in India. Now India stands in second rank in worldwide in farm production, Agriculture and allied sectors like forestry and fisheries considered for 14.5% of the GDP in 2015 and about 50% of the total manpower. The economy improvement of agriculture towards India's GDP is strongly declining growth still agriculture is statistically the broadest economic background and plays a significant role in the various socio economic frame work of India. Indian agriculture is affected by various factors such as climate, due to topography, historical, geographical, biological, political, and institutional and socio-economic factors.

As time passed there are variations in natural factors and nature of technology so policies also changed. So, agriculture production performance also changes in drastic path and large gap in different geographic locations of country. The factors that affect agriculture are independent of one another. So this arise risk and the consistent output of food also affected. Agricultural production is mostly affected by environmental factors. Weather influences crop growth and development, causing large intra-seasonal yield variability. In addition, spatial variability of soil properties, interacting with the weather, cause spatial yield variability. Crop agronomic management that is planting, fertilizer application, irrigation, tillage etc., can be used to offset the loss in yield due to effects of weather. As a result, yield forecasting represents an important tool for optimizing crop yield and to evaluate the crop-area insurance contracts. As the climate changes time to time due to the pollution, population, solid waste management, surface and ground water hydrology etc. and the impact of climate change in the developing world described by G Yamuna.

I get the Motivation from the farmers and I talked to few of them about the Agriculture related things, there problems, how can they managed to Cultivate particular crop in a region, how can they decide to cultivate a particular crops that give maximum yields means about parameters that they used to decide that. So after Knowing lots of things from farmer i have decided to design a machine learning model that

make his task easier to decide the best crops for their land, what is yield of that crop, what is cost of cultivation ,what is cost of production ,etc. so, after getting Motivation from their lifestyle i have thinking about that model.

My Proposed Solution:

In this project, I made two different models to predict (Suitable crop by the help of soil's NPK level, pH, rainfall, temperature, area, production and yield), (Crop Yield by the help of state name, crops name, cultivation cost, production cost).

Following are the two machine learning models that I have designed :-

MACHINE LEARNING MODEL NO 1: Design machine Learning model for prediction of suitable crop on the basis of soil's Nitrogen, Phosphorus, Potassium levels, pH, rainfall , temperature, area, production and yield.

Dataset used: Crop_production.csv

MACHINE LEARNING MODEL NO 2: Design machine Learning model for prediction of crop yield on the basis of the crop name, State name, Cost of Cultivation (/Hectare) A2+FL, Cost of Cultivation (/Hectare) C2, Cost of Production (`/Quintal) C2.

Dataset used: datafile (1).csv

4.1 Code submission (Github link)

click the link to view the code

4.2 Report submission (Github link) : Click the link to view to report

Github link for both code and report : Click the link to view both code and report

5 Proposed Design/ Model

First, I have done the EDA (Exploratory Data Analysis) in which I did Data Cleaning and Data preprocessing part, after that I have done the Data Visualization part in which I have shown the different-different scope of agriculture things like rainfall in different states, temperature in different states, most crop grown, leading producer state, crop yield, state wise crop production, cultivation cost, recommended zone for various crops and its variety, etc. and then I did machine learning model designing, training and testing parts (implementation parts) of different-different model and checked how efficiently they work and also checked accuracy, precision, recall, f1 score, R2 score, etc., to evaluate the performance of different-different machine learning algorithms. In this project, I have used various supervised learning Algorithms (classification and regression both) for designing the different-different machine learning sub project.

Various Machine Learning Algorithms that I have used in this project are as follows:

1. Linear Regression: Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc. Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

2. Logistic regression: Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1. Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems. In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

3. Support Vector Machine: Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

4. Naïve Bayes algorithm: Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

5. Decision Tree: Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

6. Random Forest: Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

7. Ensemble learning: Ensemble learning gives us credence to the idea of the "wisdom of crowds," it suggests that the choice-making for a more extensive organization of humans is usually higher than that of an individual professional. Another side, ensemble learning refers to a collection (or ensemble) of base newbies or fashions, which are paintings collectively to attain a better very last of the prediction. A single model, also called a base or susceptible learner, may not perform well due to high variance or bias. But, while vulnerable learners are aggregated, they could shape a sturdy learner, as their combination reduces bias or variance, yielding higher model performance. Ensemble learning is a widely used and desired tool learning technique in which more than one person models, often referred to as base models, are blended to produce a powerful ideal of the prediction version. An example of ensemble learning is the Random Forest algorithm. Ensemble learning is frequently illustrated using selection timber as this

algorithm may be liable to overfitting (excessive variance and low bias) when it has not been pruned. It could additionally lend itself to underfitting (low variance and extreme bias) when it is very small, like a decision stump, a decision tree with one stage. While an algorithm overfits or fits its education set, it cannot generalize nicely to new datasets, so ensemble strategies are used to counteract this conduct to allow for the generalization of the model to new datasets. While selection timber can showcase excessive variance or high bias, it is worth noting that it is not the best modelling approach that leverages ensemble learning to find the "sweet spot" in the bias-variance trade-off.

5.1 High Level Diagram

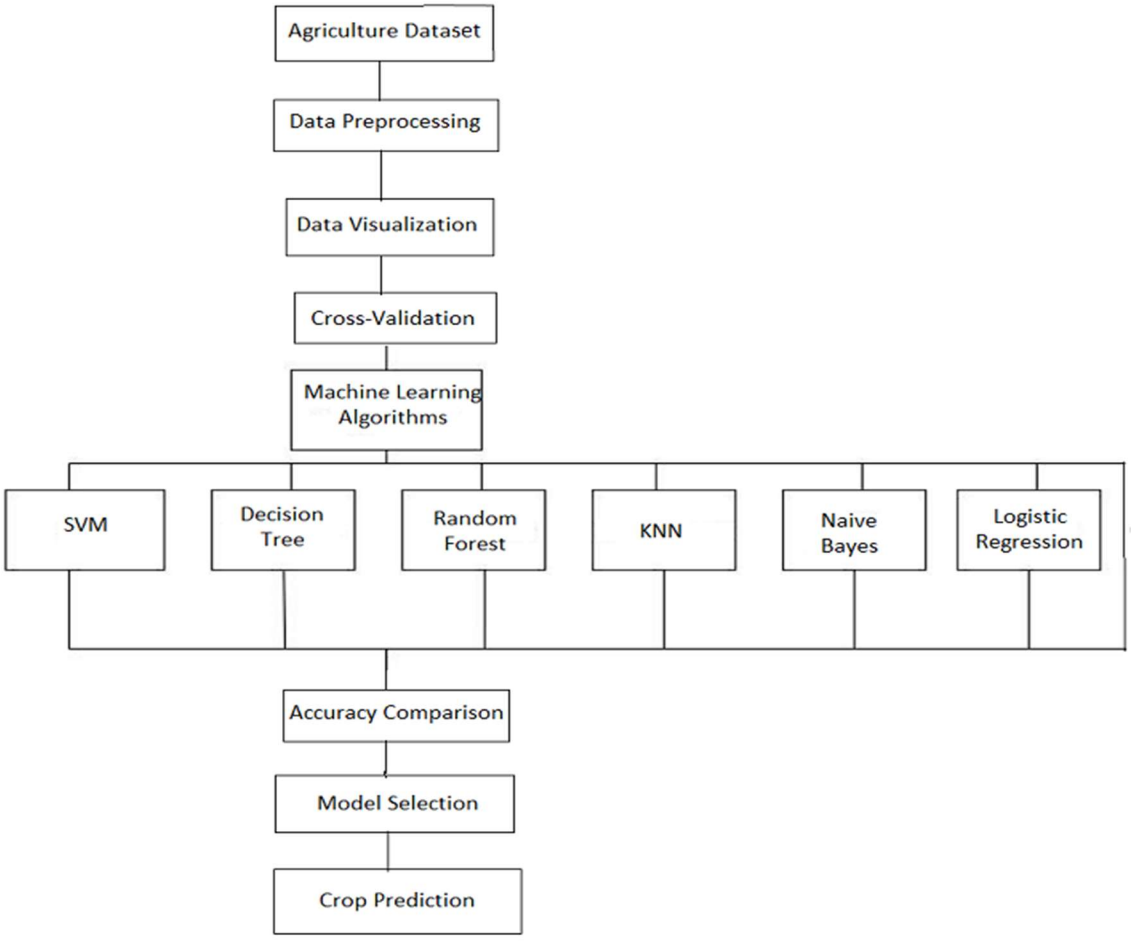


Figure 1: HIGH LEVEL DIAGRAM OF CROP PREDICTION

5.2 Use Case Diagram

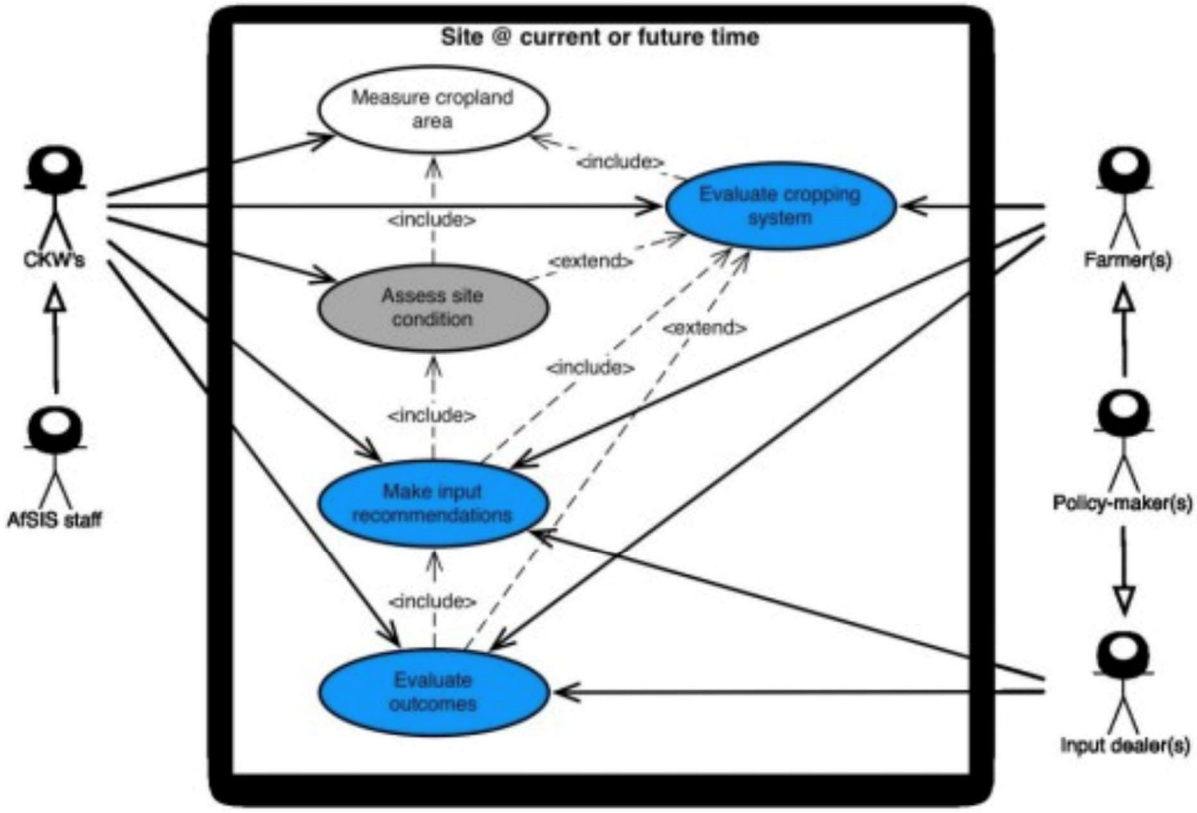


Figure 2: Use Case Diagram of Yield Prediction

5.3 Flow Diagram of Crop Yield predictions

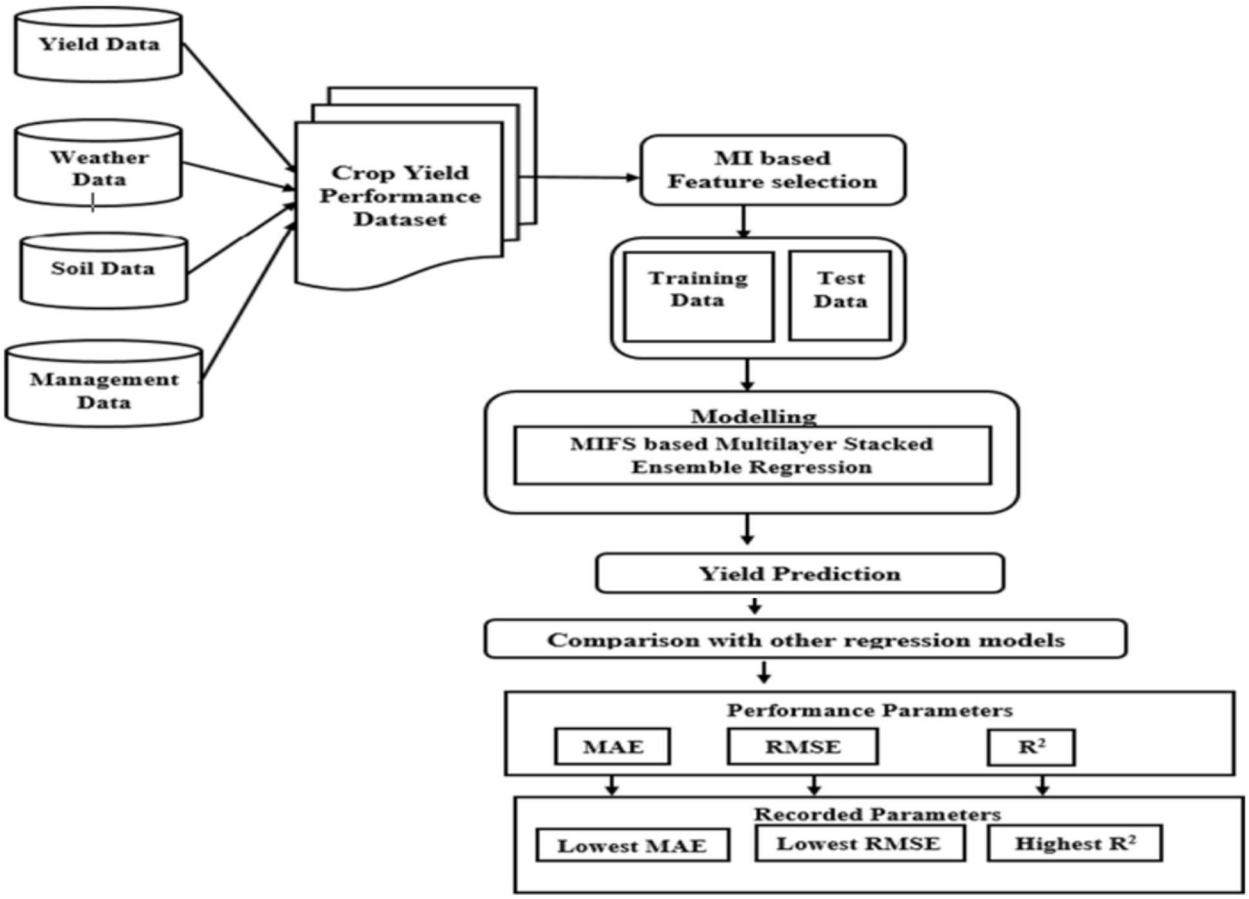
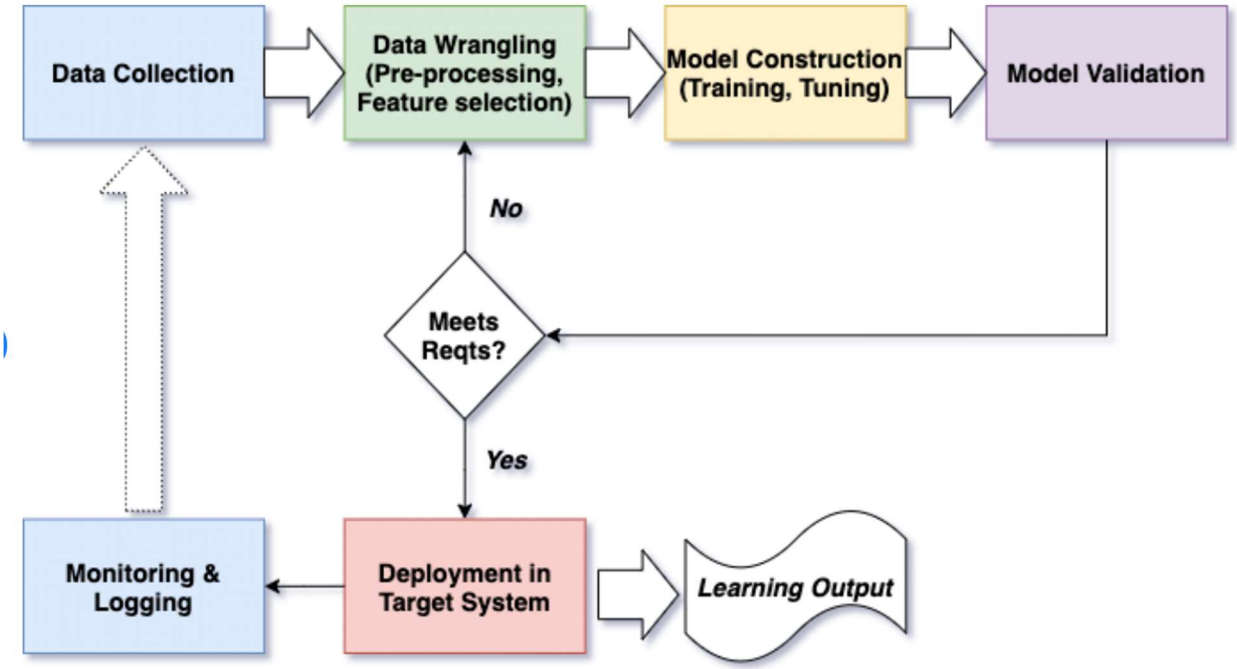
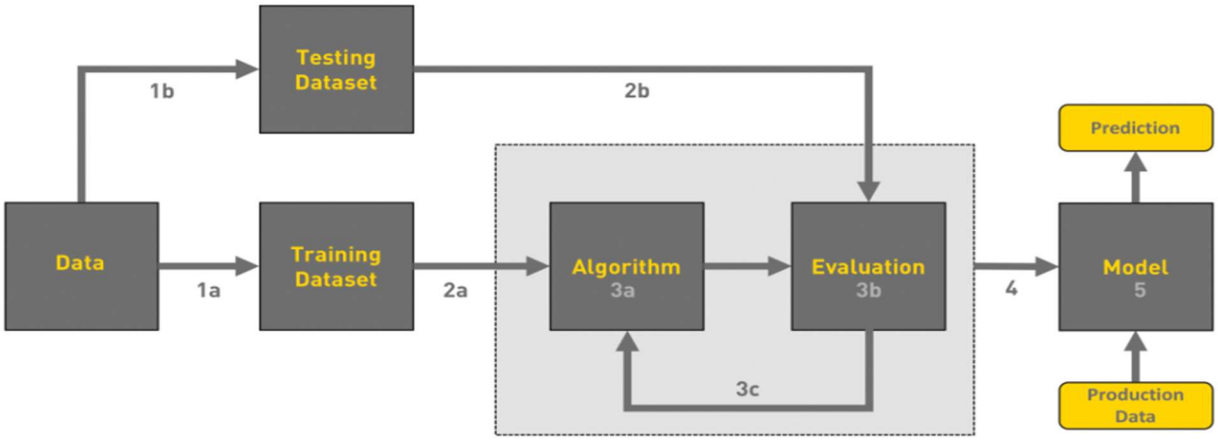


Fig.3: Flow Diagram of Crop Yield predictions

5.4 Interfaces (if applicable)



Block diagram of a generic ML-based system and important activities.



Basic block diagram of machine learning model.

1.Gathering data

2.Data pre-processing

3.Researching the model that will be best for the type of data

4.Training and testing the model

5.Evaluation

6.Prediction

Note: For implementation of model, I used Jupyter Notebook

Data Preprocessing:

Data preprocessing is a crucial step in machine learning that involves preparing and cleaning the raw data before feeding it into a machine learning model. It aims to enhance the quality of the data, making it suitable for analysis and modeling. Data preprocessing can significantly impact the performance and accuracy of the machine learning model, as the quality of the data directly affects the model's ability to learn and generalize from the data.

Here are the screenshots from my code, where I performed data preprocessing :

For first model :

Importing the libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
```

Declaring the size of every figure which will be plotted

In [2]:
```
plt.rcParams['figure.figsize'] = 8, 5
plt.style.use("fivethirtyeight")
pd.options.plotting.backend = "plotly"
```

Importing and reading the dataset

In [3]:
```
data = pd.read_csv("Crop_production.csv")
```

In [4]:
```
data.head(20)
```

Out[4]:

	Num	State_Name	Crop_Type	Crop	N	P	K	pH	rainfall	temperature	Area_in_hectares	Production_in_tons	Yield_ton_per_hec
0	37680	tamil nadu	whole year	apple	20	125	200	6.48	884.50	27.654545	3.0	0.0	0.000000
1	38056	tamil nadu	whole year	apple	20	125	200	6.44	884.50	27.654545	1.0	0.0	0.000000
2	46425	tamil nadu	whole year	apple	20	125	200	6.46	884.50	27.654545	4.0	0.0	0.000000
3	262	assam	whole year	arecanut	100	40	140	6.18	2169.32	23.736364	7160.0	5360.0	0.748603
4	277	assam	whole year	arecanut	100	40	140	5.98	2169.32	23.736364	2950.0	3521.0	1.193559
5	292	assam	whole year	arecanut	100	40	140	6.00	2169.32	23.736364	4505.0	1058.0	0.234850
6	307	assam	whole year	arecanut	100	40	140	6.06	2169.32	23.736364	4325.0	1710.0	0.395376
7	322	assam	whole year	arecanut	100	40	140	5.88	2169.32	23.736364	835.0	756.0	0.905389
8	337	assam	whole year	arecanut	100	40	140	6.14	2169.32	23.736364	2210.0	979.0	0.442986
9	352	assam	whole year	arecanut	100	40	140	6.06	2169.32	23.736364	3378.0	2615.0	0.774127
10	367	assam	whole year	arecanut	100	40	140	5.90	2169.32	23.736364	360.0	264.0	0.733333
11	382	assam	whole year	arecanut	100	40	140	6.08	2169.32	23.736364	2296.0	1666.0	0.725610
12	397	assam	whole year	arecanut	100	40	140	6.02	2169.32	23.736364	4731.0	2787.0	0.589093
13	411	assam	whole year	arecanut	100	40	140	5.90	2169.32	23.736364	2400.0	2164.0	0.901667
14	426	assam	whole year	arecanut	100	40	140	6.02	2169.32	23.736364	2700.0	3135.0	1.161111
15	441	assam	whole year	arecanut	100	40	140	6.14	2169.32	23.736364	5611.0	3395.0	0.605061
16	456	assam	whole year	arecanut	100	40	140	6.18	2169.32	23.736364	1213.0	988.0	0.814509
17	470	assam	whole year	arecanut	100	40	140	5.86	2169.32	23.736364	2035.0	1330.0	0.653563
18	484	assam	whole year	arecanut	100	40	140	5.86	2169.32	23.736364	1345.0	2344.0	1.742751
19	499	assam	whole year	arecanut	100	40	140	6.02	2169.32	23.736364	2235.0	1337.0	0.598210

Removing unnecessary columns

In [5]:
```
data=data.drop(columns='Num')
```

In [6]:
```
data.head()
```

Out[6]:

	State_Name	Crop_Type	Crop	N	P	K	pH	rainfall	temperature	Area_in_hectares	Production_in_tons	Yield_ton_per_hec
0	tamil nadu	whole year	apple	20	125	200	6.48	884.50	27.654545	3.0	0.0	0.000000
1	tamil nadu	whole year	apple	20	125	200	6.44	884.50	27.654545	1.0	0.0	0.000000
2	tamil nadu	whole year	apple	20	125	200	6.46	884.50	27.654545	4.0	0.0	0.000000
3	assam	whole year	arecanut	100	40	140	6.18	2169.32	23.736364	7160.0	5360.0	0.748603
4	assam	whole year	arecanut	100	40	140	5.98	2169.32	23.736364	2950.0	3521.0	1.193559

In [7]:
```
data=data.drop(columns='Crop_Type')
```

In [8]:
```
data.head()
```

Out[8]:

	State_Name	Crop	N	P	K	pH	rainfall	temperature	Area_in_hectares	Production_in_tons	Yield_ton_per_hec
0	tamil nadu	apple	20	125	200	6.48	884.50	27.654545	3.0	0.0	0.000000
1	tamil nadu	apple	20	125	200	6.44	884.50	27.654545	1.0	0.0	0.000000
2	tamil nadu	apple	20	125	200	6.46	884.50	27.654545	4.0	0.0	0.000000
3	assam	arecanut	100	40	140	6.18	2169.32	23.736364	7160.0	5360.0	0.748603
4	assam	arecanut	100	40	140	5.98	2169.32	23.736364	2950.0	3521.0	1.193559

Number of rows and columns of the dataset

In [9]:
```
data.shape
```

Out[9]:
```
(99849, 11)
```

In [10]:
```
data.describe()
```

Out[10]:

	N	P	K	pH	rainfall	temperature	Area_in_hectares	Production_in_tons	Yield_ton_per_hec
count	99849.000000	99849.000000	99849.000000	99849.000000	99849.000000	99849.000000	99849.000000	9.984900e+04	99849.000000
mean	69.816823	41.593656	42.037827	5.643624	701.151085	26.684154	16476.585668	3.776291e+04	3.931149
std	39.571469	15.056508	28.430263	0.505283	604.701552	4.851214	43604.268231	1.222447e+05	33.872242
min	10.000000	10.000000	10.000000	3.820000	3.274569	1.180000	0.580000	0.000000e+00	0.000000
25%	50.000000	40.000000	20.000000	5.360000	157.310000	23.106000	130.000000	1.620000e+02	0.586207
50%	75.000000	40.000000	30.000000	5.540000	579.750000	27.333333	1010.000000	1.506000e+03	1.329268
75%	80.000000	60.000000	50.000000	5.960000	1110.780000	29.266667	8099.000000	1.439500e+04	2.997288
max	180.000000	125.000000	200.000000	7.000000	3322.060000	35.346667	726300.000000	3.530571e+06	9801.000000

Information about the dataset

In [11]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99849 entries, 0 to 99848
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   State_Name          99849 non-null  object
 1   Crop                99849 non-null  object
 2   N                   99849 non-null  int64
 3   P                   99849 non-null  int64
 4   K                   99849 non-null  int64
 5   pH                  99849 non-null  float64
 6   rainfall            99849 non-null  float64
 7   temperature         99849 non-null  float64
 8   Area_in_hectares    99849 non-null  float64
 9   Production_in_tons  99849 non-null  float64
 10  Yield_ton_per_hec   99849 non-null  float64
dtypes: float64(6), int64(3), object(2)
memory usage: 8.4+ MB
```

Check if there are any null values present in the dataset

In [12]:
```
data.isnull()
```

Out[12]:

	State_Name	Crop	N	P	K	pH	rainfall	temperature	Area_in_hectares	Production_in_tons	Yield_ton_per_hec
0	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...
99844	False	False	False	False	False	False	False	False	False	False	False
99845	False	False	False	False	False	False	False	False	False	False	False
99846	False	False	False	False	False	False	False	False	False	False	False
99847	False	False	False	False	False	False	False	False	False	False	False
99848	False	False	False	False	False	False	False	False	False	False	False

99849 rows × 11 columns

There are not any values in the dataset

```
In [13]: data.isnull().sum()
```

```
Out[13]: State_Name            0
         Crop                  0
         N                     0
         P                     0
         K                     0
         pH                    0
         rainfall              0
         temperature           0
         Area_in_hectares      0
         Production_in_tons    0
         Yield_ton_per_hec     0
         dtype: int64
```

```
In [14]: data.duplicated().sum()
```

```
Out[14]: 0
```

Check if there are any space between words in the column names and convert it into underscore

```
In [15]: data.columns = data.columns.str.replace(' ', '_')
```

For Second model :

Importing and reading the dataset

```
In [36]: data2 = pd.read_csv("datafile (1).csv")
```

```
In [37]: data2.head(15)
```

Out[37]:

	Crop	State	Cost of Cultivation (`/Hectare) A2+FL	Cost of Cultivation (`/Hectare) C2	Cost of Production (`/Quintal) C2	Yield (Quintal/ Hectare)
0	ARHAR	Uttar Pradesh	9794.05	23076.74	1941.55	9.83
1	ARHAR	Karnataka	10593.15	16528.68	2172.46	7.47
2	ARHAR	Gujarat	13468.82	19551.90	1898.30	9.59
3	ARHAR	Andhra Pradesh	17051.66	24171.65	3670.54	6.42
4	ARHAR	Maharashtra	17130.55	25270.26	2775.80	8.72
5	COTTON	Maharashtra	23711.44	33116.82	2539.47	12.69
6	COTTON	Punjab	29047.10	50828.83	2003.76	24.39
7	COTTON	Andhra Pradesh	29140.77	44756.72	2509.99	17.83
8	COTTON	Gujarat	29616.09	42070.44	2179.26	19.05
9	COTTON	Haryana	29918.97	44018.18	2127.35	19.90
10	GRAM	Rajasthan	8552.69	12610.85	1691.66	6.83
11	GRAM	Madhya Pradesh	9803.89	16873.17	1551.94	10.29
12	GRAM	Uttar Pradesh	12833.04	21618.43	1882.68	10.93
13	GRAM	Maharashtra	12985.95	18679.33	2277.68	8.05
14	GRAM	Andhra Pradesh	14421.98	26762.09	1559.04	16.69

Number of rows and columns in the dataset

```
In [38]: data2.shape
```

```
Out[38]: (49, 6)
```

In [39]:
```
data2.describe()
```

Out[39]:

	Cost of Cultivation (`/Hectare) A2+FL	Cost of Cultivation (`/Hectare) C2	Cost of Production (`/Quintal) C2	Yield (Quintal/ Hectare)
count	49.000000	49.000000	49.000000	49.000000
mean	20363.537347	31364.666735	1620.537755	98.086735
std	13561.435306	20095.783569	1104.990472	245.293123
min	5483.540000	7868.640000	85.790000	1.320000
25%	12774.410000	19259.840000	732.620000	9.590000
50%	17022.000000	25909.050000	1595.560000	13.700000
75%	24731.060000	35423.480000	2228.970000	36.610000
max	66335.060000	91442.630000	5777.480000	1015.450000

Information about the dataset

In [40]:
```
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49 entries, 0 to 48
Data columns (total 6 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   Crop                                  49 non-null     object 
 1   State                                 49 non-null     object 
 2   Cost of Cultivation (`/Hectare) A2+FL  49 non-null     float64
 3   Cost of Cultivation (`/Hectare) C2     49 non-null     float64
 4   Cost of Production (`/Quintal) C2      49 non-null     float64
 5   Yield (Quintal/ Hectare)              49 non-null     float64
dtypes: float64(4), object(2)
memory usage: 2.4+ KB
```

In [41]:
```
data2['Yield (Quintal/ Hectare) '].value_counts()
```

Out[41]:
```
9.83       1
3.01       1
1.32       1
5.90       1
6.70       1
36.61      1
32.42      1
39.04      1
67.41      1
56.00      1
12.94      1
13.54      1
13.57      1
11.61      1
19.94      1
448.89     1
986.21     1
757.92     1
744.01     1
1015.45    1
23.59      1
39.83      1
34.99      1
4.05       1
42.68      1
7.47       1
13.70      1
9.59       1
6.42       1
8.72       1
12.69      1
24.39      1
```

Removing the spaces from the column name

In [42]:
```
data2.rename(columns={'Yield (Quintal/ Hectare) ':'Yield(Quintal/Hectare)'},inplace=True)
```

In [43]:
```
data2.head()
```

Out[43]:

	Crop	State	Cost of Cultivation (`/Hectare) A2+FL	Cost of Cultivation (`/Hectare) C2	Cost of Production (`/Quintal) C2	Yield(Quintal/Hectare)
0	ARHAR	Uttar Pradesh	9794.05	23076.74	1941.55	9.83
1	ARHAR	Karnataka	10593.15	16528.68	2172.46	7.47
2	ARHAR	Gujarat	13468.82	19551.90	1898.30	9.59
3	ARHAR	Andhra Pradesh	17051.66	24171.65	3670.54	6.42
4	ARHAR	Maharashtra	17130.55	25270.26	2775.80	8.72

In [44]:
```
data2['Yield(Quintal/Hectare)'].value_counts()
```

Out[44]:
```
9.83       1
3.01       1
1.32       1
5.90       1
6.70       1
36.61      1
32.42      1
39.04      1
67.41      1
56.00      1
12.94      1
13.54      1
13.57      1
11.61      1
19.94      1
448.89     1
986.21     1
757.92     1
744.01     1
1015.45    1
23.59      1
39.83      1
34.99      1
4.05       1
42.68      1
7.47       1
13.70      1
9.59       1

13.45      1
9.33       1
42.95      1
31.10      1
23.56      1
37.19      1
Name: Yield(Quintal/Hectare), dtype: int64
```

Checking if there is any null values present in the dataset

In [45]:
```
data2.isnull().sum()
```

Out[45]:
```
Crop                                0
State                               0
Cost of Cultivation (`/Hectare) A2+FL    0
Cost of Cultivation (`/Hectare) C2       0
Cost of Production (`/Quintal) C2        0
Yield(Quintal/Hectare)              0
dtype: int64
```

In [46]:
```
data2.duplicated().sum()
```

Out[46]:
```
0
```

There are not any null values in the dataset

Data Visualization :

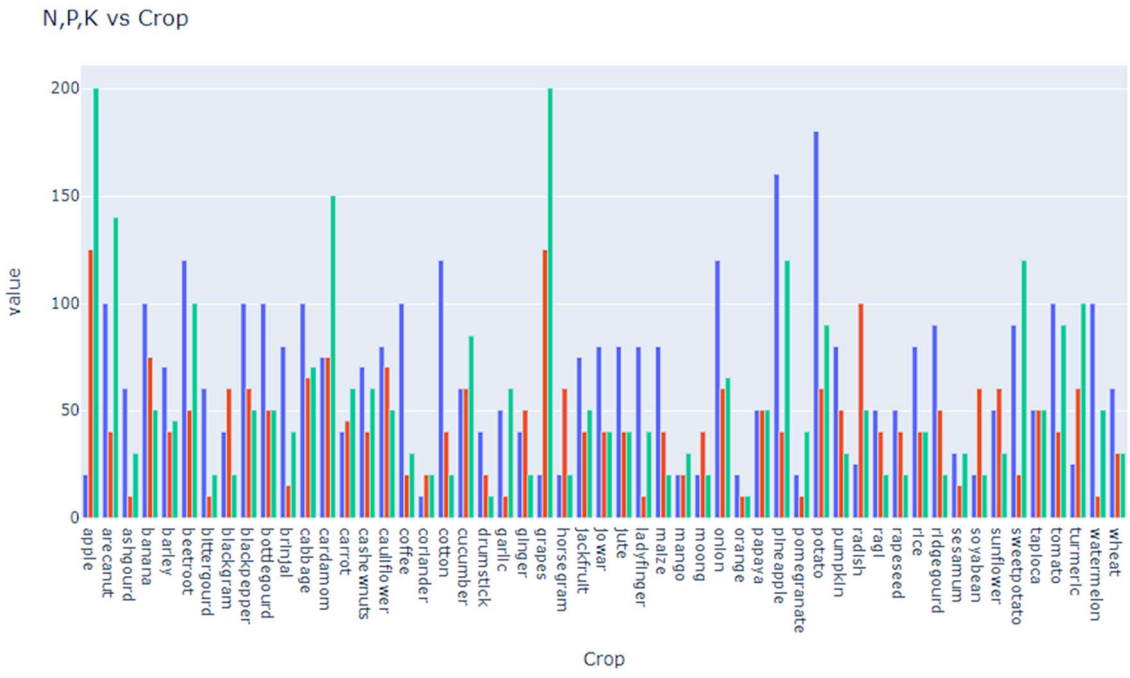
Data visualization plays a crucial role in machine learning as it helps in gaining insights, understanding patterns, and communicating results effectively. Data visualization techniques allow ML practitioners to explore the data, identify trends, detect anomalies, and make informed decisions about data preprocessing, feature engineering, and model selection.

Here are the screenshots from my code where I performed data visualization :

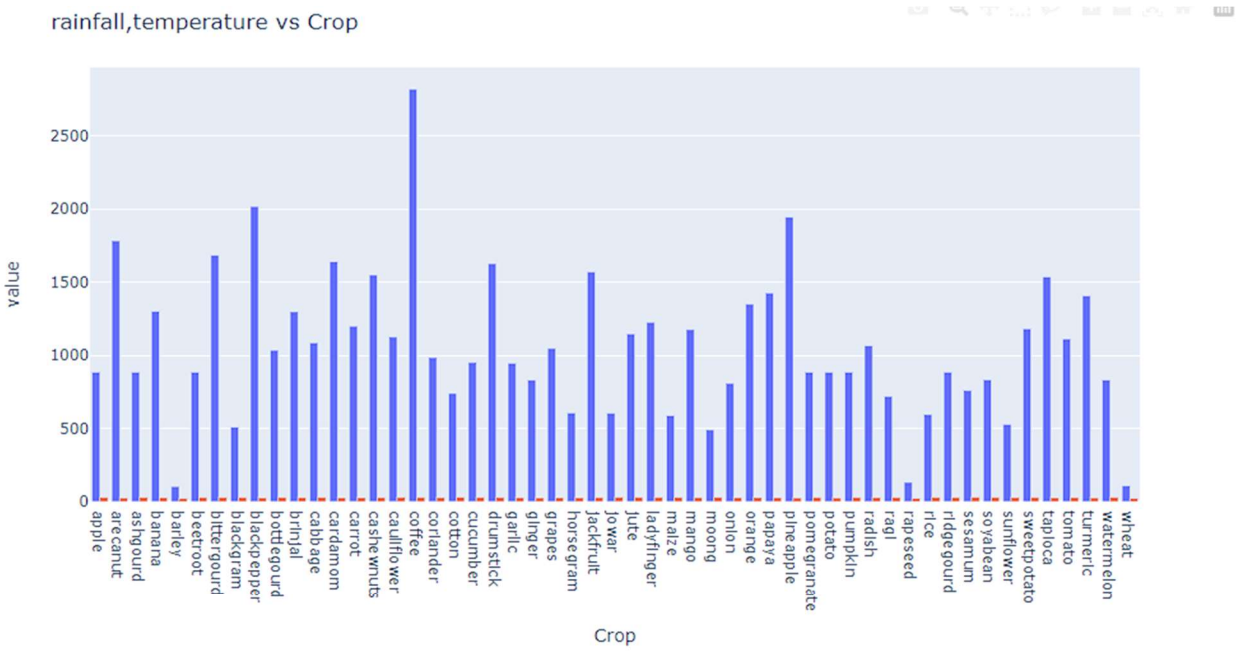
For First Model:

Data Visualization

In [16]:
```
df = data[["Crop","N","P","K"]].copy()
df = df.groupby('Crop')[["N","P","K"]].mean().reset_index()
fig = df.plot(kind='bar',x='Crop',y=["N","P","K"],barmode='group')
fig.update_layout(showlegend=False, title='N,P,K vs Crop')
fig.show()
```

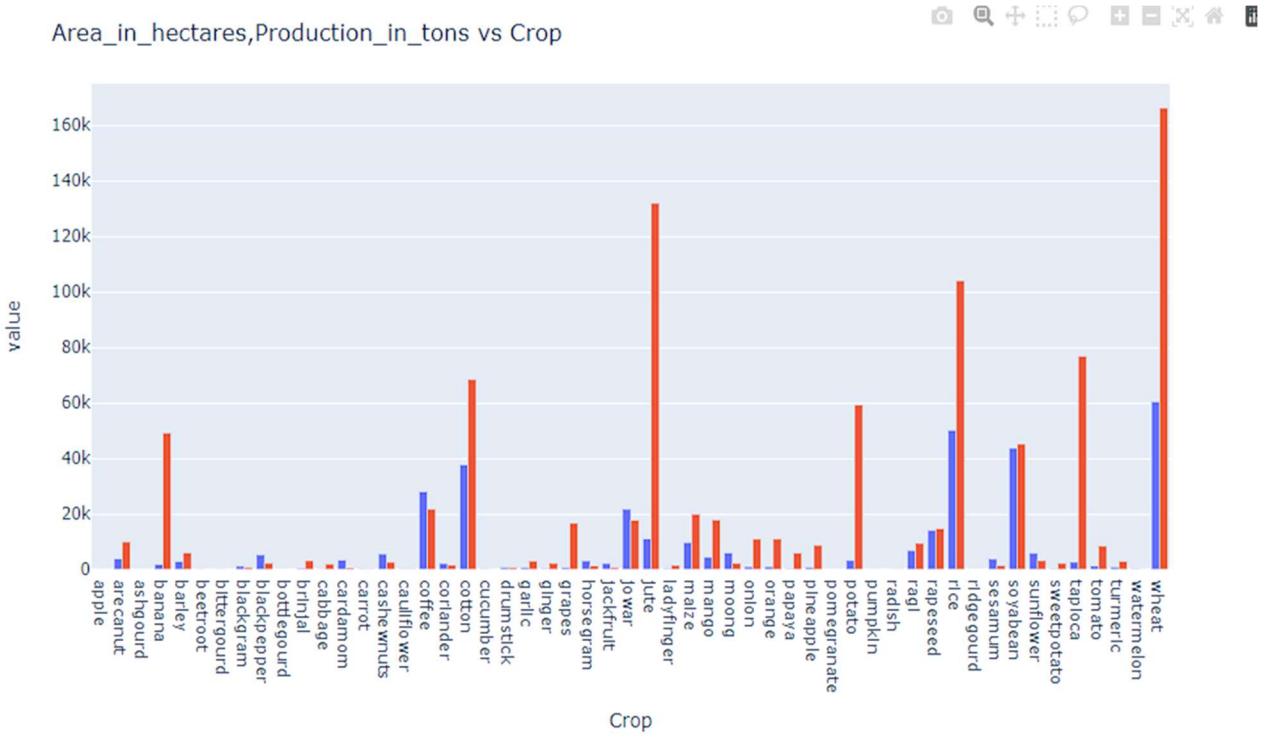


In [17]:
```
df = data[["Crop","rainfall","temperature"]].copy()
df = df.groupby('Crop')[["rainfall","temperature"]].mean().reset_index()
fig = df.plot(kind='bar',x='Crop',y=["rainfall","temperature"],barmode='group')
fig.update_layout(showlegend=False, title='rainfall,temperature vs Crop')
fig.show()
```

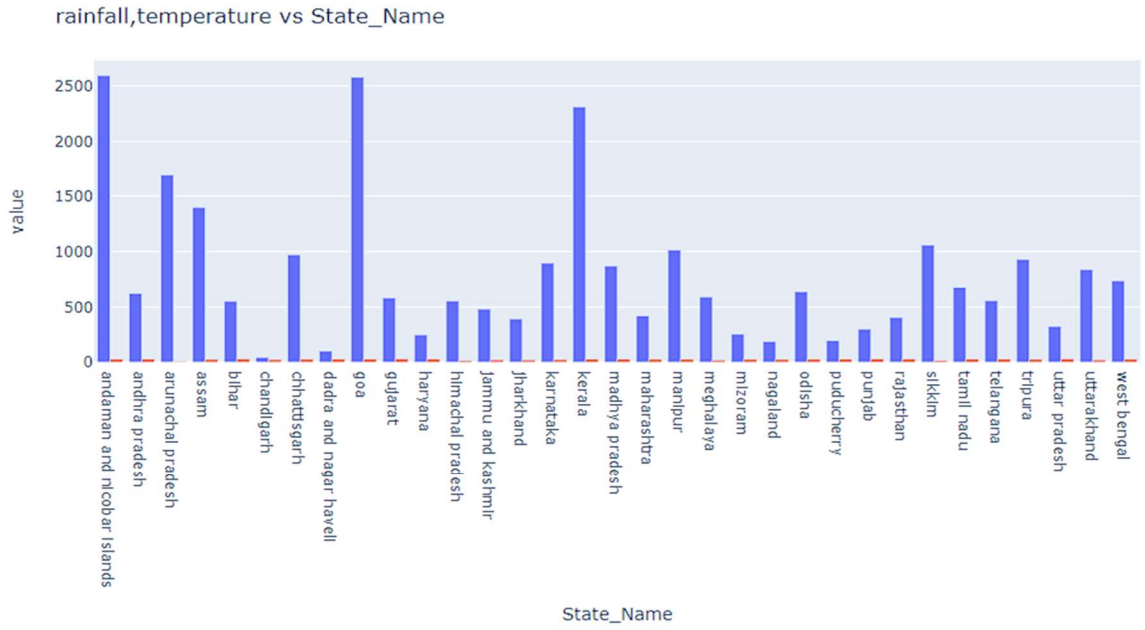


Coffee requires maximum amount of rainfall and temperature

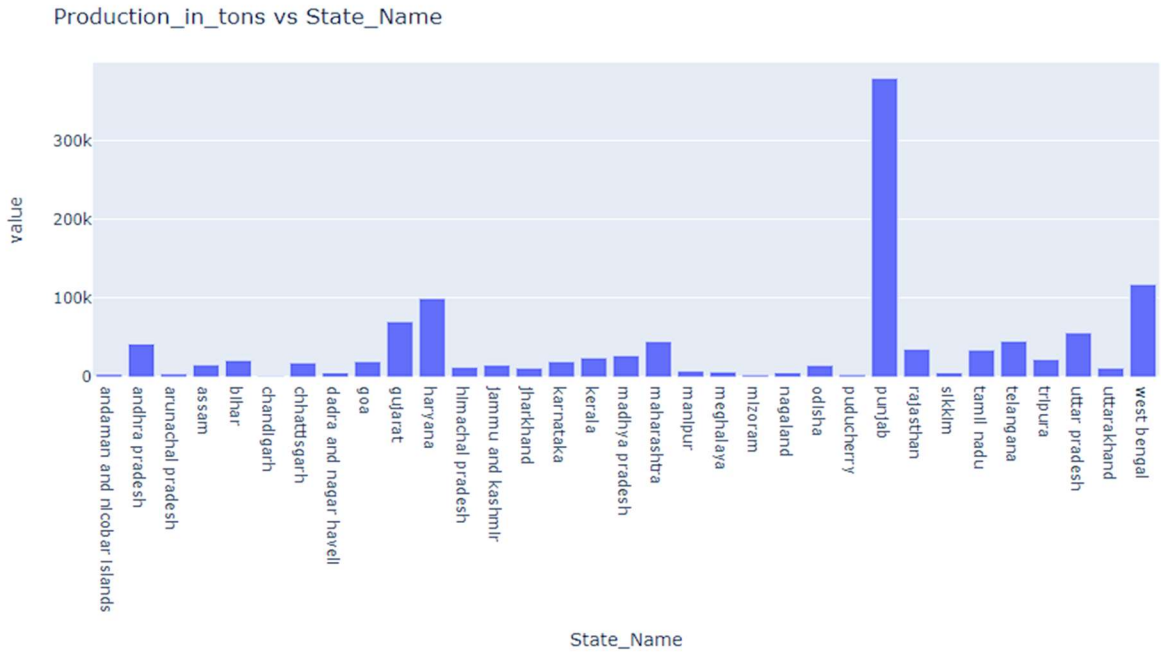
In [18]:
```
df = data[["Crop","Area_in_hectares","Production_in_tons"]].copy()
df = df.groupby('Crop')[["Area_in_hectares","Production_in_tons"]].mean().reset_index()
fig = df.plot(kind='bar',x='Crop',y=["Area_in_hectares","Production_in_tons"],barmode='group')
fig.update_layout(showlegend=False, title='Area_in_hectares,Production_in_tons vs Crop')
fig.show()
```



In [19]:
```
df = data[["State_Name","rainfall","temperature"]].copy()
df = df.groupby('State_Name')[["rainfall","temperature"]].mean().reset_index()
fig = df.plot(kind='bar',x='State_Name',y=["rainfall","temperature"],barmode='group')
fig.update_layout(showlegend=False, title='rainfall,temperature vs State_Name')
fig.show()
```

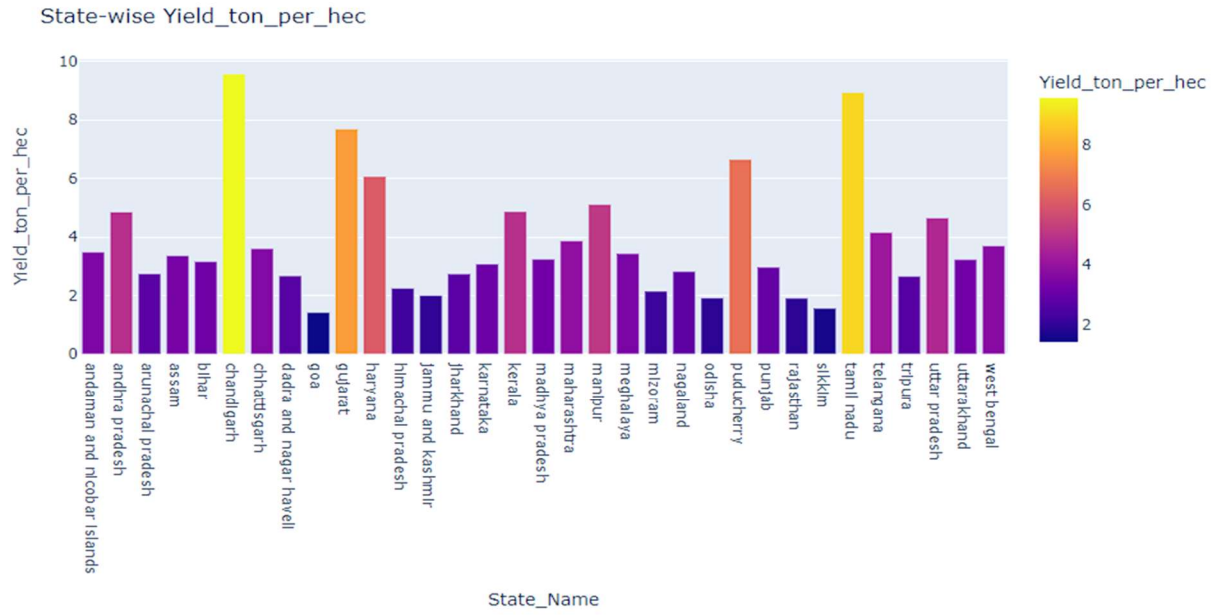


In [20]:
```
df = data[["State_Name","Production_in_tons"]].copy()
df = df.groupby('State_Name')[["Production_in_tons"]].mean().reset_index()
fig = df.plot(kind='bar',x='State_Name',y=["Production_in_tons"],barmode='group')
fig.update_layout(showlegend=False, title='Production_in_tons vs State_Name')
fig.show()
```



Punjab has the most production of crops

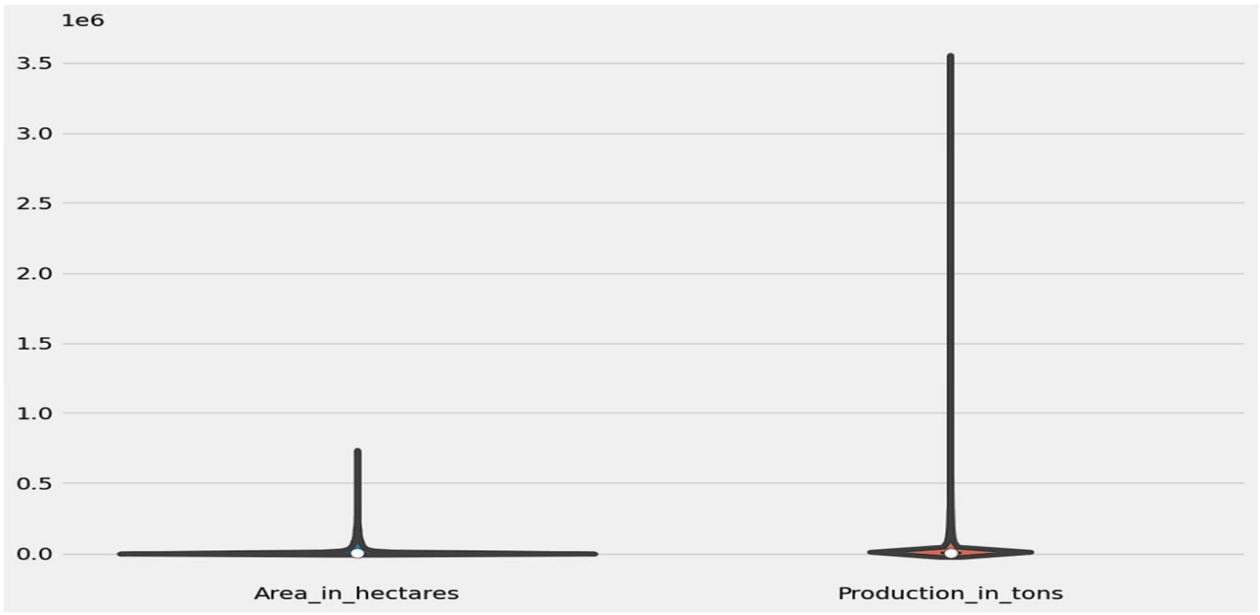
In [21]:
```
Yield = data[["State_Name","Yield_ton_per_hec"]]
df = Yield.groupby('State_Name')[["Yield_ton_per_hec"]].mean().reset_index()
fig = df.plot(kind='bar',x='State_Name',y="Yield_ton_per_hec", color="Yield_ton_per_hec")
fig.update_layout(title="State-wise Yield_ton_per_hec")
fig.show()
```



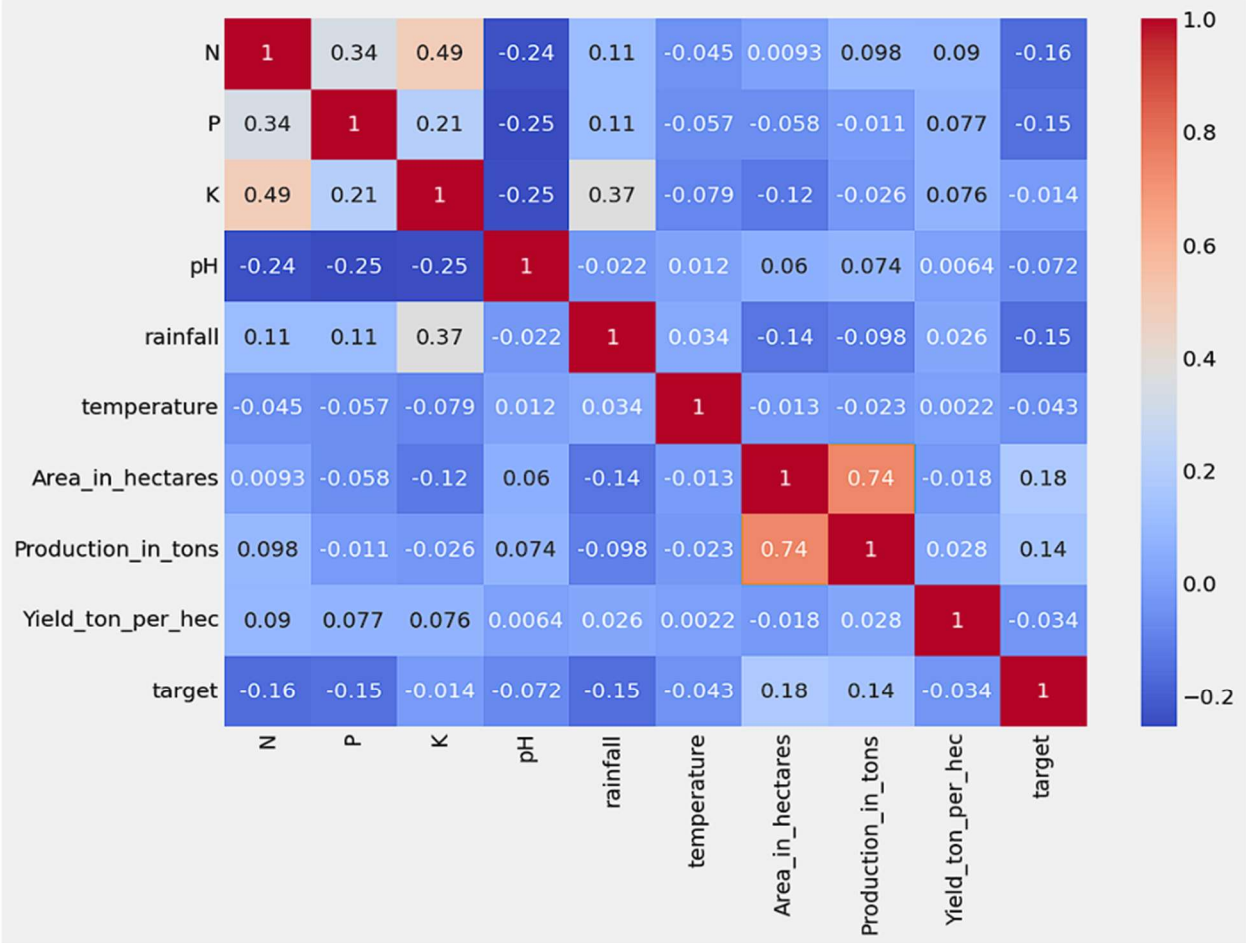
Chandigarh has the highest yield(ton/ hectare)

In [22]:
```
print('Area and Productions')
sns.violinplot(data=data[["Area_in_hectares", "Production_in_tons"]] , size=20)
plt.show()
```

Area and Productions



```
In [37]: sns.heatmap(corr,annot=True,cbar=True,cmap='coolwarm',fmt='.2g')
Out[37]: <AxesSubplot: >
```



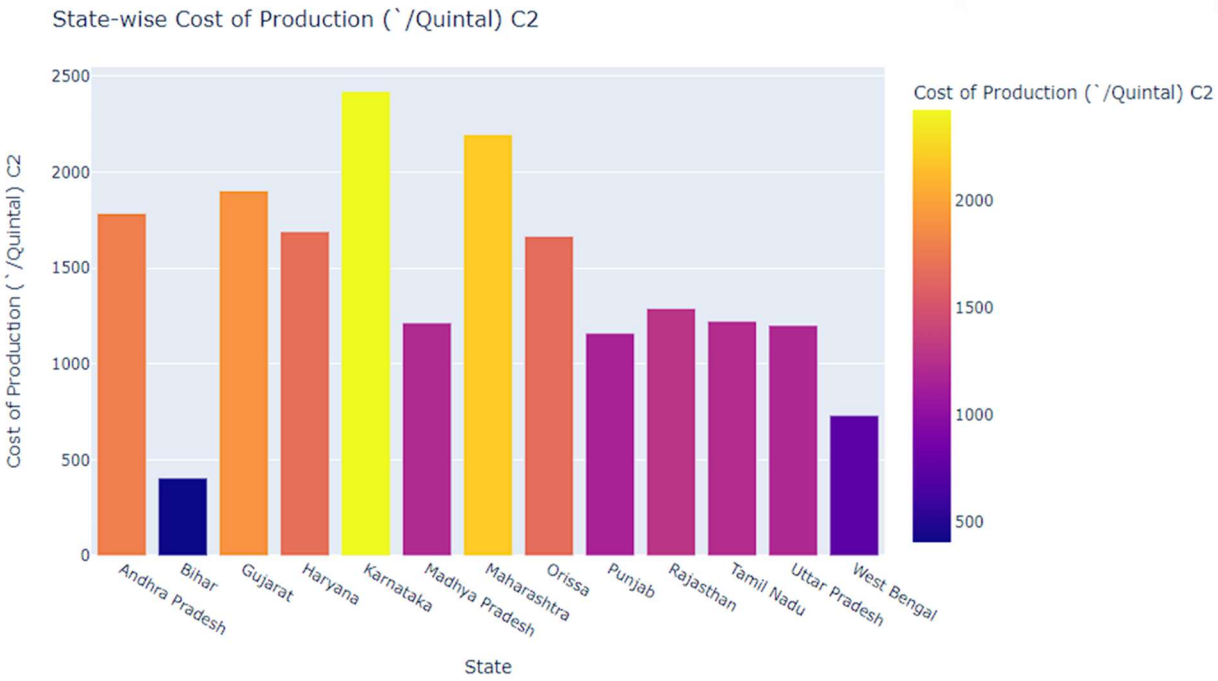
For Second model :

Data Visualization

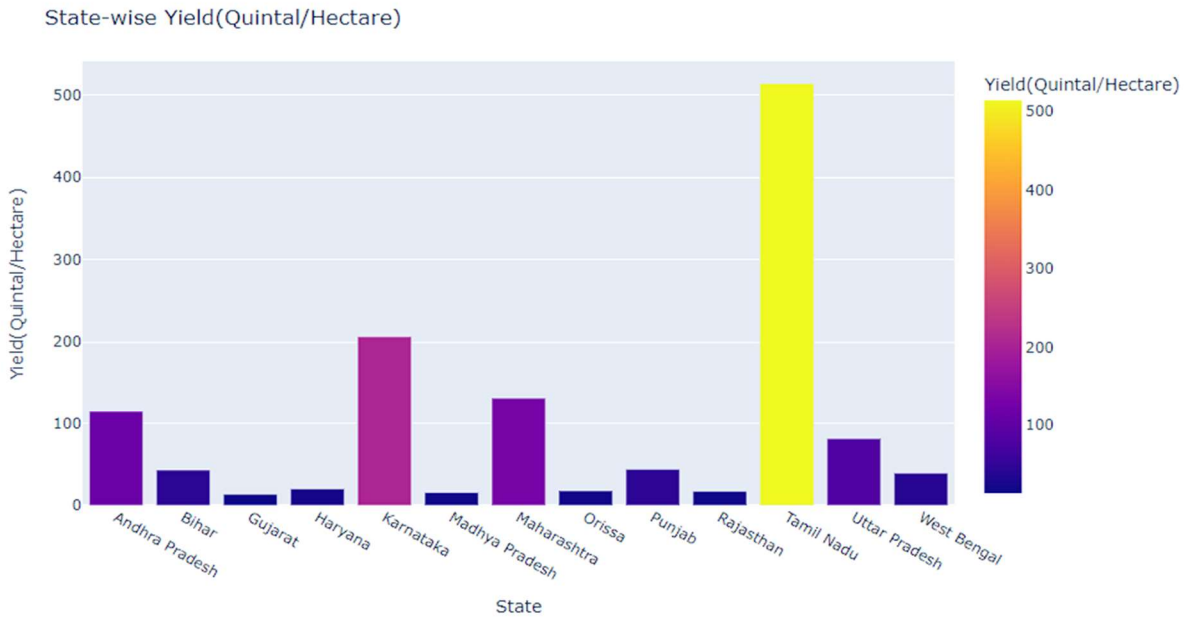
```
In [37]: df = data2[["Crop","Cost of Cultivation (`/Hectare) A2+FL","Cost of Cultivation (`/Hectare) C2"]].copy()
df = df.groupby('Crop')[["Cost of Cultivation (`/Hectare) A2+FL","Cost of Cultivation (`/Hectare) C2"]].mean().reset_index()
fig = df.plot(kind='bar',x='Crop',y=["Cost of Cultivation (`/Hectare) A2+FL","Cost of Cultivation (`/Hectare) C2"],barmode='group
fig.update_layout(showlegend=False, title='Cost of Cultivation A2+FL vs C2')
fig.show()
```

Sugarcane has the highest cost of cultivation

In [38]:
```
df = data2[["State","Cost of Production (`/Quintal) C2"]].copy()
df = df.groupby('State')[["Cost of Production (`/Quintal) C2"]].mean().reset_index()
fig = df.plot(kind='bar',x='State',y="Cost of Production (`/Quintal) C2", color="Cost of Production (`/Quintal) C2")
fig.update_layout(title="State-wise Cost of Production (`/Quintal) C2")
fig.show()
```



In [39]:
```
# Yield vs state plot
Yield = data2[["State","Yield(Quintal/Hectare)"]]
df = Yield.groupby('State')[["Yield(Quintal/Hectare)"]].mean().reset_index()
fig = df.plot(kind='bar',x='State',y="Yield(Quintal/Hectare)", color="Yield(Quintal/Hectare)")
fig.update_layout(title="State-wise Yield(Quintal/Hectare)")
fig.show()
```

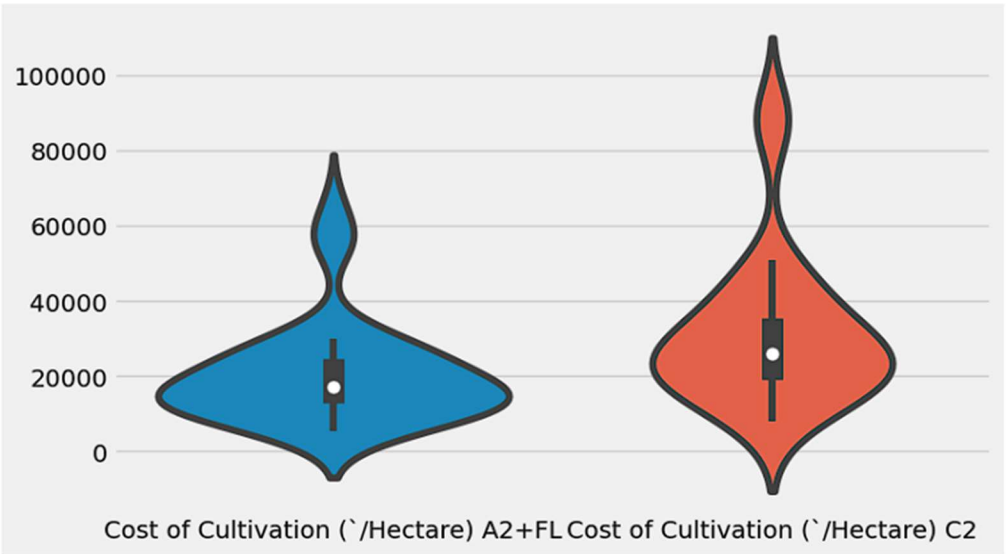


Tamil Nadu has the highest yield

Tamil Nadu has the highest yield

In [40]:
```
print('Cost of Cultivation (`/Hectare) Distributions')
sns.violinplot(data=data2[["Cost of Cultivation (`/Hectare) A2+FL", "Cost of Cultivation (`/Hectare) C2"]] , size=8)
plt.show()
```

```
Cost of Cultivation (`/Hectare) Distributions
```



6 Performance Test

This is very important part and defines why this work is meant of Real industries, instead of being just academic project.

Here we need to first find the constraints.

Constraints used in my models for evaluation are:

1.Accuracy: Accuracy simply measures how often the classifier correctly predicts. We can define accuracy as the ratio of the number of correct predictions and the total number of predictions

2.Recall: Recall explains how many of the actual positive cases we were able to predict correctly with our model. It is a useful metric in cases where False Negative is of higher concern than False Positive

3.Precision: Precision explains how many of the correctly predicted cases actually turned out to be positive. Precision is useful in the cases where False Positive is a higher concern than False Negatives

4.R2 score: R2 score, also known as the coefficient of determination, is a metric used to evaluate the performance of a regression model in machine learning. It assesses how well the model predicts the variation in the dependent variable (target) based on the independent variables (features).

4.Root Mean Squared Error (RMSE): The Root Mean Squared Error (RMSE) has been used as a standard statistical metric to measure model performance in meteorology, air quality, and climate research studies

5.MSE: Mean Squared Error (MSE) calculates the average sum of the squared difference between the actual and predicted value for the entire data points.

6.Confusion Matrix: A confusion matrix is defined as the table that is often used to describe the performance of a classification model on a set of the test data for which the true values are known.

6.1 Test Plan/ Test Cases

I checked the performance of both models using the suitable parameters. For the first model, which uses classification algorithm, I uses various performance measuring parameter such as accuracy, precision, recall, f-1 score, etc.

For the second model, which uses regression algorithm, I used the R2 _score, RMSE, etc. So, in this way I measured the performance of the model.

General Test Plan/Test Cases for Machine Learning Model:
1. Data Preprocessing:

- Test if the data is correctly loaded into the model.
- Test the handling of missing values (if any) by the model.
- Test the correctness of data scaling/normalization.
- Test the handling of categorical variables (if any) by the model.

2. Model Training:

- Test if the model can be trained without any errors.
- Test the convergence of the training process.
- Test if the model is correctly learning from the training data.
- Test the impact of different hyperparameters on model performance.
- Test the ability of the model to handle large datasets (if applicable).
- Test the model's sensitivity to class imbalance (if applicable).

3. Model Evaluation:

- Test the model's performance on a separate validation dataset.
- Test the accuracy of the model's predictions.
- Test the model's precision, recall, and F1-score.
- Test the model's performance on different subsets of the data.
- Test the model's robustness to noise and outliers.
- Test the model's interpretability (if applicable).

These are general categories for test cases and test plan for a machine learning model.

Below I have attached the screenshots of these steps: -

For first model :

Importing the dependencies

In [24]:
```
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn import tree

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder

%matplotlib inline
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier, export_graphviz

from sklearn import preprocessing
import plotly.express as px
import plotly.graph_objects as go
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
```

Data Encoding

Using label encoding for encoding the categorical values to numerical values

In [25]:
```
label_encode=LabelEncoder()
```

In [26]:
```
labels=label_encode.fit_transform(data['Crop'])
```

In [27]:
```
print(labels)
```

```
[ 0  0  0 ... 52 52 52]
```

In [28]:
```
data['target']=labels
```

In [29]:
```
data
```

Out[29]:

	State_Name	Crop	N	P	K	pH	rainfall	temperature	Area_in_hectares	Production_in_tons	Yield_ton_per_hec	target
0	tamil nadu	apple	20	125	200	6.48	884.50	27.654545	3.0	0.0	0.000000	0
1	tamil nadu	apple	20	125	200	6.44	884.50	27.654545	1.0	0.0	0.000000	0
2	tamil nadu	apple	20	125	200	6.46	884.50	27.654545	4.0	0.0	0.000000	0
3	assam	arecanut	100	40	140	6.18	2169.32	23.736364	7160.0	5360.0	0.748603	1
4	assam	arecanut	100	40	140	5.98	2169.32	23.736364	2950.0	3521.0	1.193559	1
...	...	...	...	...	...	...	...	...	...	...	...	...
99844	west bengal	wheat	60	30	30	6.60	152.54	22.280000	565.0	1452.0	2.569912	52
99845	west bengal	wheat	60	30	30	6.80	152.54	22.280000	4740.0	11345.0	2.393460	52
99846	west bengal	wheat	60	30	30	6.30	152.54	22.280000	96390.0	286362.0	2.970868	52
99847	west bengal	wheat	60	30	30	6.60	152.54	22.280000	38740.0	98264.0	2.536500	52
99848	west bengal	wheat	60	30	30	6.70	152.54	22.280000	2013.0	5152.0	2.559364	52

99849 rows × 12 columns

In [30]:
```
data=data.drop(columns='Crop',axis=1)
```

Dropped the column 'crop' and included new column as target

In [31]:
```
data.head(8)
```

Out[31]:

	State_Name	N	P	K	pH	rainfall	temperature	Area_in_hectares	Production_in_tons	Yield_ton_per_hec	target
0	tamil nadu	20	125	200	6.48	884.50	27.654545	3.0	0.0	0.000000	0
1	tamil nadu	20	125	200	6.44	884.50	27.654545	1.0	0.0	0.000000	0
2	tamil nadu	20	125	200	6.46	884.50	27.654545	4.0	0.0	0.000000	0
3	assam	100	40	140	6.18	2169.32	23.736364	7160.0	5360.0	0.748603	1
4	assam	100	40	140	5.98	2169.32	23.736364	2950.0	3521.0	1.193559	1
5	assam	100	40	140	6.00	2169.32	23.736364	4505.0	1058.0	0.234850	1
6	assam	100	40	140	6.06	2169.32	23.736364	4325.0	1710.0	0.395376	1
7	assam	100	40	140	5.88	2169.32	23.736364	835.0	756.0	0.905389	1

In [32]:
```
data['State_Name'].value_counts()
```

Out[32]:

```
uttar pradesh        12598
madhya pradesh        9299
karnataka             9224
bihar                 8608
odisha                6244
tamil nadu            6147
rajasthan             5600
assam                 5525
maharashtra           4243
andhra pradesh        3802
west bengal           3785
```

```
sikkim                          290
andaman and nicobar islands     118
dadra and nagar haveli           82
chandigarh                       63
goa                              54
Name: State_Name, dtype: int64
```

In [33]:
```
data.head()
```

Out[33]:

	State_Name	N	P	K	pH	rainfall	temperature	Area_in_hectares	Production_in_tons	Yield_ton_per_hec	target
0	tamil nadu	20	125	200	6.48	884.50	27.654545	3.0	0.0	0.000000	0
1	tamil nadu	20	125	200	6.44	884.50	27.654545	1.0	0.0	0.000000	0
2	tamil nadu	20	125	200	6.46	884.50	27.654545	4.0	0.0	0.000000	0
3	assam	100	40	140	6.18	2169.32	23.736364	7160.0	5360.0	0.748603	1
4	assam	100	40	140	5.98	2169.32	23.736364	2950.0	3521.0	1.193559	1

In [34]:
```
data=data.drop(columns='State_Name',axis=1)
```

In [35]:
```
data.head()
```

Out[35]:

	N	P	K	pH	rainfall	temperature	Area_in_hectares	Production_in_tons	Yield_ton_per_hec	target
0	20	125	200	6.48	884.50	27.654545	3.0	0.0	0.000000	0
1	20	125	200	6.44	884.50	27.654545	1.0	0.0	0.000000	0
2	20	125	200	6.46	884.50	27.654545	4.0	0.0	0.000000	0
3	100	40	140	6.18	2169.32	23.736364	7160.0	5360.0	0.748603	1
4	100	40	140	5.98	2169.32	23.736364	2950.0	3521.0	1.193559	1

In [36]:
```
corr=data.corr()
```

Data Splitting into attributes and target or we can say data is being split into Feature matrix and Response Vector

X is the attribute and Y is target

In [38]:
```
X=data.drop(columns='target',axis=1)
Y=data['target']
```

Min Max Scaling of data X

In [41]:
```
from sklearn.preprocessing import MinMaxScaler
scalers=MinMaxScaler()
X=scalers.fit_transform(X)
```

In [42]:
```
X
```

Out[42]:
```
array([[5.88235294e-02, 1.00000000e+00, 1.00000000e+00, ...,
        3.33195915e-06, 0.00000000e+00, 0.00000000e+00],
       [5.88235294e-02, 1.00000000e+00, 1.00000000e+00, ...,
        5.78273903e-07, 0.00000000e+00, 0.00000000e+00],
       [5.88235294e-02, 1.00000000e+00, 1.00000000e+00, ...,
        4.70880178e-06, 0.00000000e+00, 0.00000000e+00],
       ...,
       [2.94117647e-01, 1.73913043e-01, 1.05263158e-01, ...,
        1.32713062e-01, 8.11092597e-02, 3.03118901e-04],
       [2.94117647e-01, 1.73913043e-01, 1.05263158e-01, ...,
        5.33380847e-02, 2.78323251e-02, 2.58800096e-04],
       [2.94117647e-01, 1.73913043e-01, 1.05263158e-01, ...,
        2.77078564e-03, 1.45925404e-03, 2.61132959e-04]])
```

Standardization of data X

In [43]:
```
scaler=StandardScaler()
```

In [44]:
```
X=scaler.fit_transform(X)
```

In [45]:
```
print(X)
```

```
[[-1.25891391  5.53958213  5.55615552 ... -0.37779944 -0.30891409
  -0.11605872]
 [-1.25891391  5.53958213  5.55615552 ... -0.37784531 -0.30891409
  -0.11605872]
 [-1.25891391  5.53958213  5.55615552 ... -0.37777651 -0.30891409
  -0.11605872]
 ...
 [-0.24807956 -0.77001352 -0.42341808 ...  1.83270624  2.03362921
  -0.02835023]
 [-0.24807956 -0.77001352 -0.42341808 ...  0.51058134  0.49492048
  -0.04117403]
 [-0.24807956 -0.77001352 -0.42341808 ... -0.33170281 -0.26676889
  -0.04049901]]
```

Splitting the data into training and testing data

22% of data is test data and 78 % if train data

In [46]:
```
#testsize=0.22 means give 22 percent of data as test data and remaining 78 percent as training data
X_train,X_test,Y_train,Y_test =train_test_split(X,Y,test_size=0.22,stratify=Y,random_state=72)
#stratify=Y means data is splitted on the basis of ....and .. in equal manner
#random_state is used to splitted the data in particular order
```

Training and Testing of data starts here

Importing different algorithms from Scikit learn and checking their training and testing accuracy

In [49]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import AdaBoostClassifier
results=[]
names=[]
# create instances of all models
models = {
 'Logistic Regression': LogisticRegression(),
 'Naive Bayes': GaussianNB(),
 'Support Vector Machine': SVC(),
 'K-Nearest Neighbors': KNeighborsClassifier(),
 'Decision Tree': DecisionTreeClassifier(),
 'Random Forest': RandomForestClassifier(),
 'Bagging': BaggingClassifier(),
 'AdaBoost': AdaBoostClassifier(),
}
from sklearn.metrics import accuracy_score
print("TRAINING ACCURACY:")
for name, model in models.items():
 model.fit(X_train, Y_train)
 X_train_prediction = model.predict(X_train)
 trainacc = accuracy_score( X_train_prediction, Y_train)
 print(f'{name}:\ntrainAccuracy: {trainacc:.4f}')
#for testing data
print("TESTING ACCURACY:")
for name, model in models.items():
 model.fit(X_train, Y_train)
 y_pred = model.predict(X_test)
 testacc = accuracy_score(Y_test, y_pred)
 print(f'{name}:\ntestingAccuracy: {testacc:.4f}')
```

```
TRAINING ACCURACY:

Logistic Regression:
trainAccuracy:  0.9500
Naive Bayes:
trainAccuracy: 0.9331
Support Vector Machine:
trainAccuracy:  0.9631
K-Nearest Neighbors:
trainAccuracy:  0.9880
Decision Tree:
trainAccuracy: 1.0000
Random Forest:
trainAccuracy: 1.0000
Bagging:
trainAccuracy: 0.9988
AdaBoost:
trainAccuracy: 0.2906

TESTING ACCURACY:

Logistic Regression:
testingAccuracy:  0.9502
Naive Bayes:
testingAccuracy:  0.9344
Support Vector Machine:
testingAccuracy:  0.9633
K-Nearest Neighbors:
testingAccuracy:  0.9805
Decision Tree:
testingAccuracy:  0.9827
Random Forest:
testingAccuracy:  0.9875
Bagging:
testingAccuracy:  0.9873
AdaBoost:
testingAccuracy:  0.2907
```

For second model :

Data Encoding

Encoding the categorical values to numerical values

Using One Hot Encoding to encode cost and state column

```
In [17]: data2=pd.get_dummies(data2,prefix=['Cropn','statename'],columns=['Crop','State'])

In [18]: data2.head()
```

Out[18]:

	Cost of Cultivation (`/Hectare) A2+FL	Cost of Cultivation (`/Hectare) C2	Cost of Production (`/Quintal) C2	Yield(Quintal/Hectare)	Cropn_ARHAR	Cropn_COTTON	Cropn_GRAM	Cropn_GROUNDNUT	Cropn_MAIZE	Cropn_MOONG
0	9794.05	23076.74	1941.55	9.83	1	0	0	0	0	0
1	10593.15	16528.68	2172.46	7.47	1	0	0	0	0	0
2	13468.82	19551.90	1898.30	9.59	1	0	0	0	0	0
3	17051.66	24171.65	3670.54	6.42	1	0	0	0	0	0
4	17130.55	25270.26	2775.80	8.72	1	0	0	0	0	0

5 rows × 27 columns

Splitting the dataset into X and Y where X is the attributes and Y is the target

In [19]:
```
X=data2.drop(columns=['Yield(Quintal/Hectare)'])
```

In [20]:
```
Y=data2['Yield(Quintal/Hectare)']
```

In [21]:
```
print(X)
```

```
      Cost of Cultivation (`/Hectare) A2+FL  Cost of Cultivation (`/Hectare) C2  \
0                                   9794.05                           23076.74
1                                  10593.15                           16528.68
2                                  13468.82                           19551.90
3                                  17051.66                           24171.65
4                                  17130.55                           25270.26
5                                  23711.44                           33116.82
6                                  29047.10                           50828.83
7                                  29140.77                           44756.72
8                                  29616.09                           42070.44
9                                  29918.97                           44018.18
10                                  8552.69                           12610.85
11                                  9803.89                           16873.17
12                                 12833.04                           21618.43
13                                 12985.95                           18679.33
14                                 14421.98                           26762.09
15                                 13647.10                           17314.20
16                                 21229.01                           30434.61
17                                 22507.86                           30393.66
```

Min Max Scaling of X

In [25]:
```
from sklearn.preprocessing import MinMaxScaler
scalers=MinMaxScaler()
X=scalers.fit_transform(X)
```

In [26]:
```
print(X)
```

```
[[0.07083652 0.18197169 0.32604727 ... 0.         1.         0.        ]
 [0.08396849 0.10362123 0.36661695 ... 0.         0.         0.        ]
 [0.13122565 0.13979541 0.31844847 ... 0.         0.         0.        ]
 ...
 [0.20479423 0.32970593 0.12632628 ... 0.         0.         0.        ]
 [0.22178312 0.28757871 0.12018399 ... 0.         1.         0.        ]
 [0.22407887 0.26333217 0.10502856 ... 0.         0.         0.        ]]
```

Standardization of X

In [27]:
```
scaler=StandardScaler()
```

In [28]:
```
X=scaler.fit_transform(X)
```

Now splitting X and Y into training and testing data so that different data will be used for training and testing

In [30]:
```
X_train,X_test,Y_train,Y_test =train_test_split(X,Y,test_size=0.22,random_state=42)
```

In [31]:
```
X.shape,X_test.shape,X_train.shape
```

Out[31]:
```
((49, 26), (11, 26), (38, 26))
```

In [32]:
```
X_train=X_train.astype('float')
```

In [34]:
```
Y_train=Y_train.astype('float')
```

Now check train and test accuracy of different models and select the best model

1. Linear Regression

In [36]:
```
model = LinearRegression()
model.fit(X_train,Y_train)
```

Out[36]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [37]:
```
X_train_prediction = model.predict(X_train)
X_train_prediction
```

Training score

In [38]:
```
model.score(X_train,Y_train)
```

Out[38]:
```
0.986072936308964
```

In [47]:
```
from sklearn.metrics import r2_score
r = r2_score(Y_train,X_train_prediction)
print("R2 score : ",r)
```

```
R2 score :  0.986072936308964
```

In [48]:
```
X_test_prediction = model.predict(X_test)
X_test_prediction
```

Out[48]:
```
array([ -4.51328226,  -3.32565417,  58.76146506, 868.32985919,
         32.30993726, 163.68184105,  41.31784417,  13.62177946,
        -20.15498277,  40.23193914,  53.24789454])
```

Test score

In [49]:
```
model.score(X_test,Y_test)
```

Out[49]:
```
0.9377100274106578
```

R2 score

R2(R- squared) is the measure of the goodness of fit of a model

In [50]:
```
rv= r2_score(Y_test,X_test_prediction)
print("R2 score : ",rv)
```

```
R2 score :  0.9377100274106578
```

Importing some models from scikit learn

In [52]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LinearRegression
```

2. Random Forest Regressor

In [53]:
```
rfr = RandomForestRegressor(n_estimators = 11)
rfr.fit(X_train,Y_train)
```

Out[53]:
```
RandomForestRegressor(n_estimators=11)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [54]:
```
X_train_prediction = rfr.predict(X_train)
X_train_prediction
```

Training score

In [55]:
```
rfr.score(X_train,Y_train)
```

Out[55]:
```
0.9815422228405967
```

In [56]:
```
from sklearn.metrics import r2_score
r = r2_score(Y_train,X_train_prediction)
print("R2 score : ",r)
```

```
R2 score :  0.9815422228405967
```

In [57]:
```
X_test_prediction = rfr.predict(X_test)
X_test_prediction
```

Out[57]:
```
array([  8.13363636,  38.88363636,  40.00272727, 787.02909091,
        12.75818182,   6.19727273,   6.35727273,   6.18818182,
        40.62454545,  12.13454545,  10.53727273])
```

Test score

In [58]:
```
rfr.score(X_test,Y_test)
```

Out[58]:
```
0.9424492601873122
```

R2 Score

In [59]:
```
rb = r2_score(Y_test,X_test_prediction)
print("R2 score : ",rb)
```

```
R2 score :  0.9424492601873122
```

3. Decision Tree

In [62]:
```
Dt = DecisionTreeRegressor(random_state = 5)
Dt.fit(X_train,Y_train)
```

Out[62]:
```
DecisionTreeRegressor(random state=5)
```

```
In [63]: X_train_prediction = Dt.predict(X_train)
         X_train_prediction
```

Training score

```
In [64]: Dt.score(X_train,Y_train)
Out[64]: 1.0

In [65]: from sklearn.metrics import r2_score
         r = r2_score(Y_train,X_train_prediction)
         print("R2 score : ",r)

         R2 score :  1.0

In [66]: X_test_prediction = Dt.predict(X_test)
         X_test_prediction
Out[66]: array([  4.71,  36.61,  39.83, 744.01,  13.45,   5.9 ,   5.9 ,   5.9 ,
                 36.61,  12.69,   4.71])
```

Test score

```
In [67]: Dt.score(X_test,Y_test)
Out[67]: 0.9189735872455934
```

R2 score

```
In [68]: rs = r2_score(Y_test,X_test_prediction)
         print("R2 score : ",rs)

         R2 score :  0.9189735872455934
```

6.2 Test Procedure

Test Procedure for Machine Learning Model:

1. Define Test Objectives:

- Clearly define the objectives and goals of the testing process.
- Determine what aspects of the machine learning model need to be tested.
- Identify the desired outcomes and performance metrics for the model.

In both machine learning models, my objectives and goal of the testing process was to select the dataset attributes for actual and targeted value in such a way that my both model could predict target value very well with good performance score. It means selection of attributes for training and testing phase should be such that it could predict my desired outcome very well .

2. Test Data Preparation:

- Collect or generate the test dataset, representing a realistic scenario.
- Preprocess the test data to align with the model's input requirements.

- Split the dataset into training, validation, and test sets (if applicable).

In cross validation, I assigned 78 percent data value for training the model and 22 percent for testing phase means prediction of target value.

3. Test Execution:

- Train the machine learning model using the training dataset.
- Monitor the training process for any errors or anomalies.
- Evaluate the model's performance using the validation dataset.
- Adjust the model's hyperparameters or architecture if necessary.
- Conduct any necessary retraining cycles based on the results.

4. Performance Testing:

- Measure the model's accuracy, precision, recall, and F1-score.
- Evaluate the model's performance on different subsets of the data.
- Test the model's ability to handle outliers and noisy data.
- Assess the model's robustness to variations in the input data.

I have attached the screenshot of all these steps for test procedure.

6.3 Performance Outcome

The performance outcome of a machine learning model can be assessed using various evaluation metrics and techniques. The specific metrics used depend on the type of problem being solved (classification, regression, etc.) and the goals of the project. Here are some common performance outcomes for machine learning models:

1. Classification Problems:

- Accuracy: Measures the overall correctness of the model's predictions.
- Precision: Indicates the proportion of correctly predicted positive instances among all predicted positive instances.
- Recall: Measures the proportion of correctly predicted positive instances among all actual positive instances.
- F1-Score: Combines precision and recall into a single metric that balances their trade-off.
- Area Under the Receiver Operating Characteristic Curve (AUC-ROC): Evaluates the model's ability to distinguish between different classes.

2. Regression Problems:

- Mean Absolute Error (MAE): Measures the average absolute difference between the predicted and actual values.

- Root Mean Squared Error (RMSE): Similar to MAE but penalizes larger errors more heavily.
- R-squared (Coefficient of Determination): Represents the proportion of the variance in the dependent variable explained by the model.

3. Clustering Problems:

- Silhouette Score: Quantifies the compactness and separation of clusters.
- Inertia: Measures the within-cluster sum of squared distances, reflecting the compactness of the clusters.

4. Time Series Problems:

- Mean Absolute Percentage Error (MAPE): Calculates the average percentage difference between predicted and actual values.
- Mean Squared Error (MSE): Measures the average of the squared differences between predicted and actual values.

5. Anomaly Detection:

- True Positive Rate (TPR): Measures the proportion of true anomalies correctly identified.
- False Positive Rate (FPR): Indicates the proportion of non-anomalies incorrectly identified as anomalies.

6. Recommendation Systems:

- Precision at K: Measures the proportion of relevant items among the top K recommendations.
- Mean Average Precision (MAP): Calculates the average precision across different values of K.

It's important to note that the choice of performance metrics should align with the specific problem, the nature of the data, and the project's objectives. Furthermore, the performance outcome should be interpreted in conjunction with domain knowledge and the context of the problem to assess the overall effectiveness and utility of the machine learning model.

Here is my performance outcome for the two models that I made in this project.

For first model:

Accuracy on test data is **96%** and the algorithm used is **Support Vector Classifier**.

I am using Support Vector Machine model

In [57]:
```
model = SVC()
```

In [58]:
```
sk=model.fit(X_train,Y_train)
```

In [59]:
```
print(sk)
```

```
SVC()
```

In [60]:
```
X_train_prediction=sk.predict(X_train).round()
training_data_accuracy=accuracy_score(X_train_prediction,Y_train)
```

In [54]:
```
print(X_train_prediction)
```

```
[47 52 37 ... 50 26  4]
```

Training Accuracy

In [55]:
```
print(training_data_accuracy)
```

```
0.9631108600190031
```

Training accuracy is around 96%

In [61]:
```
X_test_prediction=sk.predict(X_test).round()#that is Ypred also
test_data_accuracy=accuracy_score(X_test_prediction,Y_test)
```

In [62]:
```
print('Accuracy on test data:',test_data_accuracy)
```

```
Accuracy on test data: 0.963263076432831
```

Testing accuracy is around 96% , which is very good

Confusion matrix

In [63]:
```
from sklearn import metrics
# Print the confusion matrix
metrics.confusion_matrix(Y_test, X_test_prediction)
```

Out[63]:
```
array([[   1,    0,    0, ...,    0,    0,    0],
       [   0,  254,    0, ...,    0,    0,    0],
       [   0,    0,    6, ...,    0,    0,    0],
       ...,
       [   0,    0,    0, ...,  564,    0,    0],
       [   0,    0,    0, ...,    0,   13,    0],
       [   0,    0,    0, ...,    0,    0, 1369]], dtype=int64)
```

In [64]:
```
print(metrics.classification_report(Y_test, X_test_prediction))
```

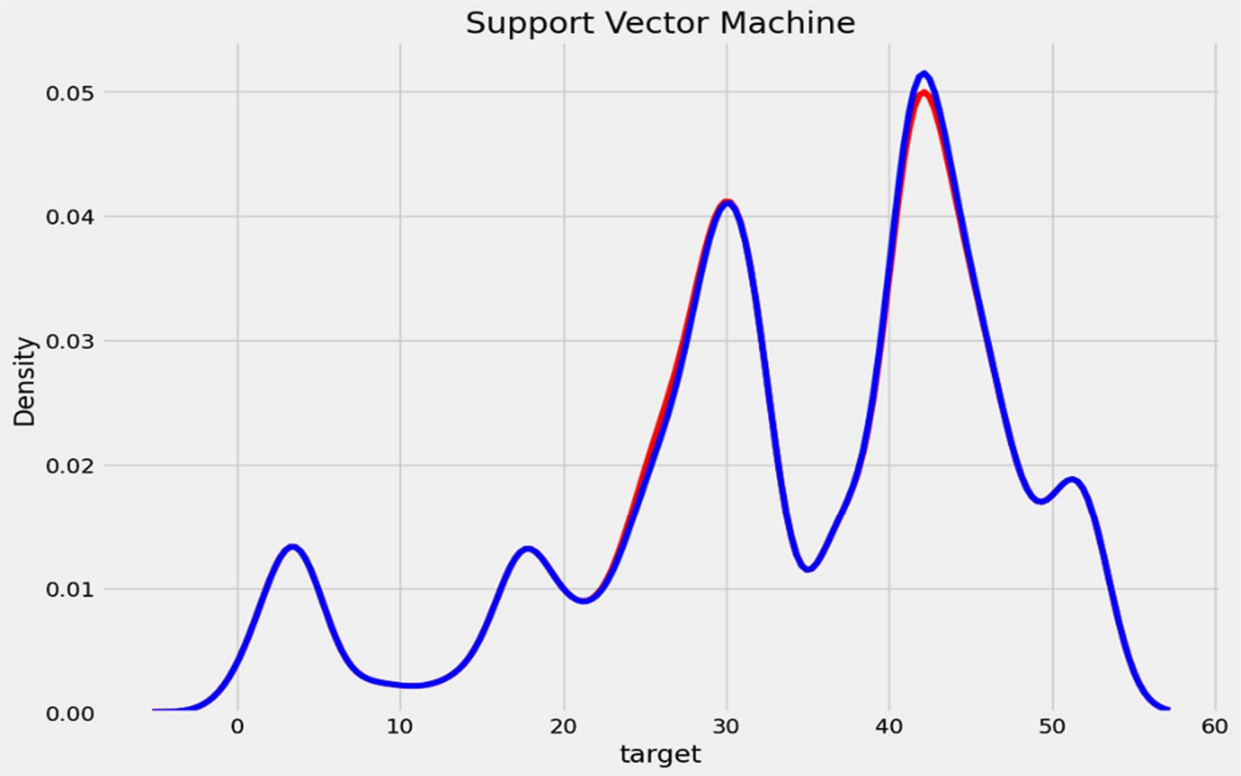
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00       254
           2       1.00      1.00      1.00         6
           3       1.00      1.00      1.00       530
           4       1.00      1.00      1.00       691
           5       1.00      1.00      1.00         3
           6       1.00      1.00      1.00        15
           7       1.00      1.00      1.00        13
           8       1.00      1.00      1.00       126
           9       1.00      1.00      1.00        13
          10       1.00      1.00      1.00        62
          11       1.00      1.00      1.00        27
          12       1.00      1.00      1.00        52
```

47	1.00	1.00	1.00	638
48	1.00	1.00	1.00	249
49	1.00	1.00	1.00	62
50	1.00	1.00	1.00	564
51	1.00	1.00	1.00	13
52	1.00	1.00	1.00	1369
accuracy			0.96	21967
macro avg	0.99	0.99	0.99	21967
weighted avg	0.96	0.96	0.96	21967

Plotting a graph to see the comparison between actual and target value

In [65]:
```
ax=sns.distplot(Y_test,hist=False,color="r",label="Actual value")
sns.distplot(X_test_prediction,hist=False,color="b",label="Predicted value",ax=ax)
plt.title('Support Vector Machine')
```

Out[65]: Text(0.5, 1.0, 'Support Vector Machine')



Finally , a predictive system for prediction of suitable crop

In [66]:
```
input_data=(180,60,90,4.96,689.88,29.04,2738,59475,21.72)
#changing the input data to a numpy arrary
input_data_as_numpy_array=np.asarray(input_data)
#reshape the np array s we are predictive for one instance
input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)
#as i already standarize the data so here for input value we once standarize this
std_data=scalers.transform(input_data_reshaped)
print(std_data)
std_data=scaler.transform(std_data)
print(std_data)
prediction=sk.predict(std_data)#model=Support Vector Machine
print("target:",prediction)
```

```
if(prediction==0):
 print("THE SUITABLE CROP IS: APPLE")
elif(prediction==1):
 print("THE SUITABLE CROP IS: ARECANUT")
elif(prediction==2):
 print("THE SUITABLE CROP IS: ASHGOURD")
elif(prediction==3):
 print("THE SUITABLE CROP IS: BANANA")
elif(prediction==4):
 print("THE SUITABLE CROP IS: BARLEY")
elif(prediction==5):
 print("THE SUITABLE CROP IS: BEETROOT")
elif(prediction==6):
 print("THE SUITABLE CROP IS: BITTERGOURD")
elif(prediction==7):
 print("THE SUITABLE CROP IS: BLACKGRAM")
elif(prediction==8):
 print("THE SUITABLE CROP IS: BLACK PEPPER")
elif(prediction==9):
 print("THE SUITABLE CROP IS: BOTTLEGOURD")
elif(prediction==10):
  print("THE SUITABLE CROP IS: BRINJAL")
elif(prediction==11):
  print("THE SUITABLE CROP IS: CABBAGE")
elif(prediction==12):
  print("THE SUITABLE CROP IS: CARDAMOM")
elif(prediction==13):
  print("THE SUITABLE CROP IS: CARROT")
elif(prediction==14):
  print("THE SUITABLE CROP IS: CASHEWNUTS")
elif(prediction==15):
  print("THE SUITABLE CROP IS: CAULIFLOWER")
elif(prediction==16):
  print("THE SUITABLE CROP IS: COFFEE")
elif(prediction==17):
  print("THE SUITABLE CROP IS: CORIANDER")
elif(prediction==18):
  print("THE SUITABLE CROP IS: COTTON")
elif(prediction==19):
  print("THE SUITABLE CROP IS: CUCUMBER")
elif(prediction==20):
  print("THE SUITABLE CROP IS: DRUMSTICK")
elif(prediction==21):
  print("THE SUITABLE CROP IS: GARLIC")
elif(prediction==22):
  print("THE SUITABLE CROP IS: GINGER")
elif(prediction==23):
  print("THE SUITABLE CROP IS: GRAPES")
elif(prediction==24):
  print("THE SUITABLE CROP IS: HORSEGRAM")
```

```
elif(prediction==25):
  print("THE SUITABLE CROP IS: JACKFRUIT")
elif(prediction==26):
  print("THE SUITABLE CROP IS: JOWAR")
elif(prediction==27):
  print("THE SUITABLE CROP IS: JUTE")
elif(prediction==28):
  print("THE SUITABLE CROP IS: LADYFINGER")
elif(prediction==29):
  print("THE SUITABLE CROP IS: MAIZE")
elif(prediction==30):
  print("THE SUITABLE CROP IS: MANGO")
elif(prediction==31):
  print("THE SUITABLE CROP IS: MOONG")
elif(prediction==32):
  print("THE SUITABLE CROP IS: ONION")
elif(prediction==33):
  print("THE SUITABLE CROP IS: ORANGE")
elif(prediction==34):
  print("THE SUITABLE CROP IS: PAPAYA")
elif(prediction==35):
  print("THE SUITABLE CROP IS: PINEAPPLE")
elif(prediction==36):
  print("THE SUITABLE CROP IS: POMEGRANATE")
elif(prediction==37):
  print("THE SUITABLE CROP IS: POTATO")
```

```
elif(prediction==38):
  print("THE SUITABLE CROP IS: PUMPKIN")
elif(prediction==39):
  print("THE SUITABLE CROP IS: RADISH")
elif(prediction==40):
  print("THE SUITABLE CROP IS: RAGI")
elif(prediction==41):
  print("THE SUITABLE CROP IS: RAPESEED")
elif(prediction==42):
  print("THE SUITABLE CROP IS: RICE")
elif(prediction==43):
  print("THE SUITABLE CROP IS: RIDGEGOURD")
elif(prediction==44):
  print("THE SUITABLE CROP IS: SESAMUM")
elif(prediction==45):
  print("THE SUITABLE CROP IS: SOYABEAN")
elif(prediction==46):
  print("THE SUITABLE CROP IS: SUNFLOWER")
elif(prediction==47):
  print("THE SUITABLE CROP IS: SWEETPOTATO")
elif(prediction==48):
  print("THE SUITABLE CROP IS: TAPIOCA")
elif(prediction==49):
  print("THE SUITABLE CROP IS: TOMATO")
elif(prediction==50):
  print("THE SUITABLE CROP IS: TURMERIC")
elif(prediction==51):
  print("THE SUITABLE CROP IS: WATERMELON")
elif(prediction==52):
  print("THE SUITABLE CROP IS: WHEAT")
else:
 print("THE SUITABLE CROP IS OTHER THAN THIS")
```

```
[[1.         0.43478261 0.42105263 0.35849057 0.20688455 0.81541463
  0.003769   0.01684572 0.0022161 ]]
[[ 2.7844235   1.22249037  1.68701966 -1.35296116 -0.01863918  0.48562233
  -0.31507591  0.17761262  0.52517752]]
target: [37]
THE SUITABLE CROP IS: POTATO
```

For Second Model :

Accuracy on test data is **91%** and the algorithm used is **Decision Tree Regressor**.

Training score

```
In [64]: Dt.score(X_train,Y_train)
Out[64]: 1.0

In [65]: from sklearn.metrics import r2_score
         r = r2_score(Y_train,X_train_prediction)
         print("R2 score : ",r)

         R2 score :  1.0

In [66]: X_test_prediction = Dt.predict(X_test)
         X_test_prediction
Out[66]: array([  4.71,  36.61,  39.83, 744.01,  13.45,   5.9 ,   5.9 ,   5.9 ,
                 36.61,  12.69,   4.71])
```

Test score

```
In [67]: Dt.score(X_test,Y_test)
Out[67]: 0.9189735872455934
```

R2 score

In [68]:
```
rs = r2_score(Y_test,X_test_prediction)
print("R2 score : ",rs)
```

```
R2 score :  0.9189735872455934
```

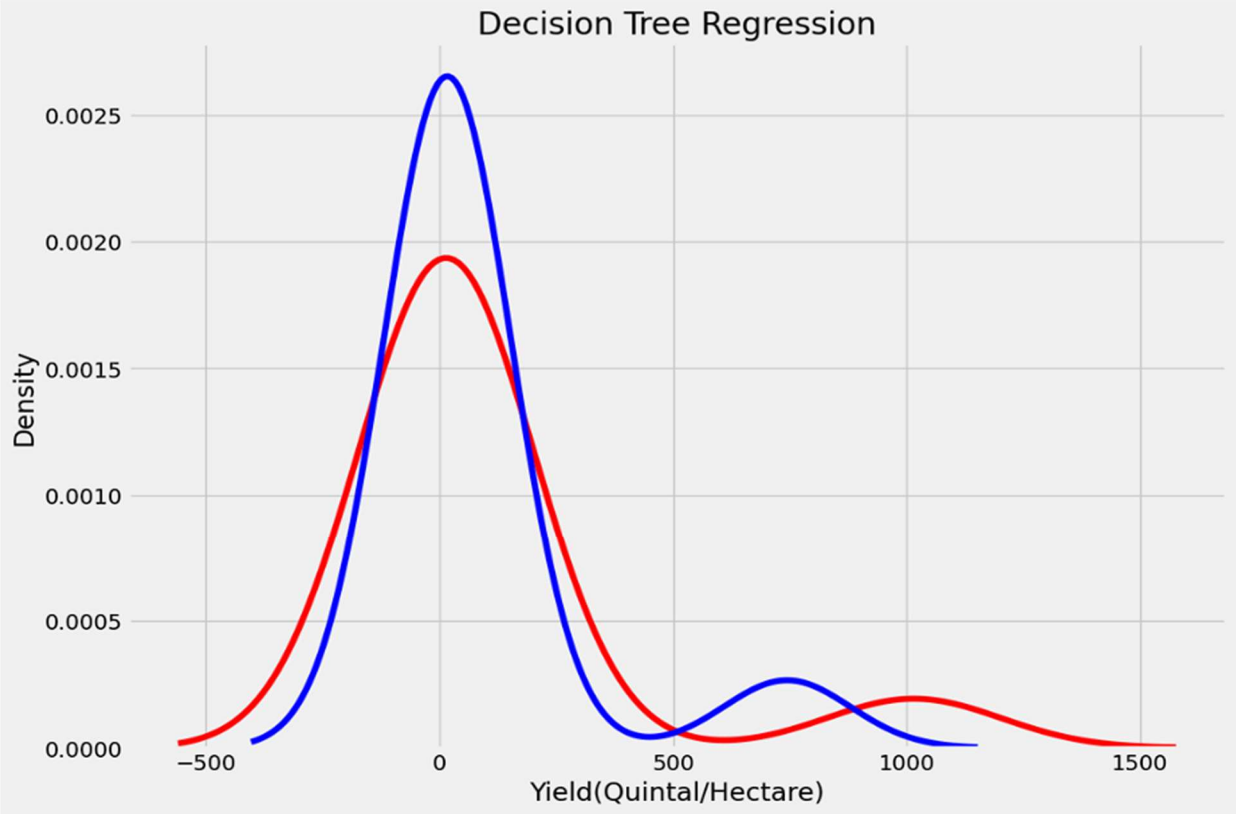
In [69]:
```
Adjr2_1 = 1 - (1-r)*(len(Y_test)-1)/(len(Y_test)-X_test.shape[1]-1)
print("Adj. R-Squared : {}".format(Adjr2_1))
```

```
Adj. R-Squared : 1.0
```

Graph for Decision Tree Regressor

In [70]:
```
ax = sns.distplot(Y_test, hist = False, color = "r", label = "Actual value ")
sns.distplot(X_test_prediction, hist = False, color = "b", label = "Predicted Values", ax = ax)
plt.title('Decision Tree Regression')
```

Out[70]: Text(0.5, 1.0, 'Decision Tree Regression')



Making a predictive System and chosing Decision Tree for prediction

In [76]:
```
# making a predictive System
input_data=(10593.15,16528.68,2172.46,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0)
#changing the input data to a numpy arrary
input_data_as_numpy_array=np.asarray(input_data)
#reshape the np array s we are predictive for one instance
input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)
#as i already standarize the data so here for input value we once standarize this
std_data=scalers.transform(input_data_reshaped)
print(std_data)
std_data=scaler.transform(std_data)
print(std_data)
prediction=Dt.predict(std_data)#model=Decision Tree
print("Yield prediction:",prediction)
```

```
[[0.08396849 0.10362123 0.36661695 1.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  1.         0.        ]]
[[-0.72791982 -0.74591427  0.50465756  2.96647939 -0.33709993 -0.33709993
  -0.33709993 -0.33709993 -0.33709993 -0.33709993 -0.33709993 -0.33709993
  -0.2981424  -0.4417261  -0.14433757 -0.2981424  -0.20628425 -0.33709993
  -0.25537696 -0.37354368 -0.20628425 -0.25537696 -0.33709993 -0.20628425
   2.44948974 -0.14433757]]
Yield prediction: [7.47]
```

So we can see that the predicted yield is also 7.47 which matches from the index 1 of this dataset and the model accuracy is 91%

So the model is giving correct predictions.

7 My learnings

Throughout this internship, I got to learn a lot about the machine learning and its aspects. While making the project on Prediction of agriculture crop production in India, I came across the process of building a machine learning model and I gained several key insights about it.

I learned about different libraries that are used with python for building a machine learning model. I got to learn about handling datasets and the importance of cleaning dataset for the prediction. I came across various algorithms and their uses, evaluation of model so as to give correct predictions.

The project aimed to leverage machine learning techniques to predict crop yields and identify influential factors for crop production in different regions of India.

Here are the key learning points summarized that I learned during the whole project:

1. **Domain Knowledge and Expertise:**
   - **Acquiring domain knowledge:** Through this project, i have deepened my understanding of agricultural practices, crop production, and the various factors that influence yields in India. This knowledge strengthens my expertise in the intersection of data science and agriculture, positioning you as a valuable resource in this domain.

- **Collaborating with domain experts:** Engaging with agricultural experts during the project has expanded my network and exposed me to real-world challenges and insights. Collaborative efforts contribute to my professional growth and enable me to bridge the gap between data science and domain-specific knowledge.

2. **Data Science Skills:**
   - **Data preprocessing and feature engineering:** Cleaning and preprocessing agricultural datasets, handling missing values, and deriving meaningful features from the data have enhanced my data manipulation skills. These skills are fundamental in data science projects and can be applied to diverse domains beyond agriculture.
   - **Model selection and evaluation:** Exploring and evaluating various machine learning algorithms for yield prediction has improved my understanding of algorithm selection, model training, and performance evaluation. This knowledge can be transferred to other predictive modeling projects across industries.
   - **Interpretability and explainability:** Gaining insights into the inner workings of the model and interpreting feature importance have enhanced my ability to communicate complex machine learning concepts to stakeholders. Explainability is a sought-after skill in the industry, ensuring transparency and trust in machine learning models.

3. **Project Management and Problem-Solving:**
   - **End-to-end project execution:** Successfully completing a machine learning project from data acquisition to model deployment demonstrates my ability to manage and execute projects independently. This experience showcases my project management skills and my capability to navigate challenges throughout the project lifecycle.
   - **Problem-solving and critical thinking:** Tackling the complexities and uncertainties inherent in agriculture yield prediction has honed my problem-solving and critical thinking abilities. These skills are invaluable in data-driven decision-making and can be applied to various business problems.

4. **Impact and Future Opportunities:**
   - **Real-world impact:** Demonstrating the practical application of machine learning in agriculture, specifically in crop production, highlights my potential to drive positive change in the industry. The insights and recommendations from my project can contribute to informed decision-making, resource optimization, and sustainable agricultural practices.
   - **Career opportunities:** The expertise gained through this project opens up diverse career opportunities in fields such as agricultural technology, precision farming, agronomy, or data science roles within agricultural organizations. my demonstrated ability to apply data science in a domain-specific context makes you a valuable asset in these industries.

Conclusion of my project:

This project aimed to predict agricultural crop and yields in India using machine learning techniques. The significance of this project lies in its potential to contribute to sustainable agriculture practices, enhance food security, and assist policymakers and farmers in making informed decisions.

The dataset for crop prediction consisted of several features, out of which I selected the required feature for the model, then performed Exploratory Data Analysis (Data Preprocessing and Data Visualization) and did the feature scaling on the training data. Further, I applied several algorithms for training and testing and based on their accuracy, I selected SVM for the crop prediction which gave 96% accuracy and Decision Tree for yield prediction which gave 91% accuracy.

My project will help farmers make informed decisions, optimize resource allocation, and enhance overall agricultural productivity. It can serve as valuable tools for farmers to assess potential risks associated with crop failures, weather fluctuations, or market conditions. Armed with such information, farmers can take proactive measures to mitigate risks and make better-informed choices.

The project contributes to the body of knowledge in the field of agriculture and data science. Research outcomes and methodologies can be shared with the academic community and other stakeholders, fostering further research and development in the domain.

8 Future work scope

The future work scope of this project is vast and holds the potential for significant advancements in agricultural practices and food security. Here are some potential areas of focus and opportunities for further development:

- **Enhancing Prediction Accuracy:** Continuous efforts can be made to improve the accuracy of crop production predictions. This involves exploring advanced machine learning algorithms, incorporating ensemble methods, and leveraging deep learning techniques to capture complex relationships within the data.
- **Incorporating Satellite Imagery:** Integrating satellite imagery data can provide valuable insights into crop health, growth patterns, and potential yield estimation. The use of remote sensing techniques and high-resolution imagery can enhance the accuracy and granularity of crop production predictions.
- **Big Data and IoT Integration:** Leveraging data from Internet of Things (IoT) devices and agricultural sensors can offer real-time data on soil moisture, temperature, and other environmental factors. Combining this data with historical records can create more dynamic and responsive prediction models.
- **Crop Disease and Pest Prediction:** Expanding the scope to predict crop diseases and pest infestations can help farmers take timely preventive measures and minimize yield losses. Machine learning models can analyze environmental factors and historical disease patterns to forecast disease outbreaks.
- **User-Friendly Interfaces:** Developing user-friendly interfaces and mobile applications can enable easy access to crop production predictions for farmers and stakeholders. Providing actionable insights and visualizations can empower users to make informed decisions.
- **Multi-Crop Prediction:** Expanding the prediction models to encompass multiple crop types can facilitate a holistic approach to agricultural planning and diversification.

Overall, the future work scope of this project is an exciting and evolving field with the potential to revolutionize farming practices, increase agricultural productivity, and contribute to food security efforts in the country. Continuous research, innovation, and collaboration will be critical to realizing the full potential of artificial intelligence and machine learning.