RAMNIRANJAN JHUNJHUNWALA COLLEGE

Department Of Computer Science
Ghatkopar (West), Mumbai – 86

Project Report On

"CLIMATE CHANGE IMPACT ON FOOD SUPPLY"

By

Mr. SHAIKH MOHAMMED AMIR
(Seat No- 510)
&
Mr. SAYED MOHAMMED OWAIS
(Seat No- 532)

M.Sc.-II (Computer Science)

University of Mumbai

2022-2023

Project Guide

Mrs. ANITA GAIKWAD

RAMNIRANJAN JHUNJHUNWALA COLLEGE

Department Of Computer Science

Ghatkopar (West), Mumbai – 86

2022-2023

A project report on

<u>"CLIMATE CHANGE IMPACT ON FOOD SUPPLY"</u>

Towards partial fulfilment of the degree of M.Sc. (Computer Science) as prescribed by the University of Mumbai.

By

Mr. Shaikh Mohammed Amir
(Seat No-510)
&
Mr. Sayed Mohammed Owais
(Seat No-532)

Project Guide
Mrs. Anita Gaikwad

RAMNIRANJAN JHUNJHUNWALA COLLEGE

Department Of Computer Science

Ghatkopar (West), Mumbai – 86

2020-2021

CERTIFICATE

This is to certify that Mr. Shaikh Mohammed Amir (Seat No-510) & Mr. Sayed Mohammed Owais (Seat No-532) has successfully completed the project titled,

"CLIMATE CHANGE IMPACT ON FOOD SUPPLY "

Under the guidance towards the partial fulfilment of degree of Masters of Science (Computer Science) submitted to Ramniranjan Jhunjhunwala College, Ghatkopar(W) during the academic year 2022-2023 as specified by the UNIVERSITY OF MUMBAI.

Guide	Course Coordinator
Mrs. Anita Gaikwad	Department Of Computer Science
	Prof. Mrs. Vaidehi Deshpande

Acknowledgement

The development of this project is the result of the cumulative efforts along with the productive input of many of our teachers and friends at Ramniranjan Jhunjhunwala College, Ghatkopar.

First & foremost, I wish to thank to our project guide Prof. Mrs. Anita Gaikwad for her valuable suggestions regarding the selection of project and for frequently collaborating with this project. His guidance not only help me in making an effort fruitful but also transform the whole process of learning and implementing into an enjoyable experience.

It is my pleasure to express sincere thanks to Hon. Principal Dr. Mrs. Usha Mukundan, who have always been the source of inspiration.

I would like to convey special thanks to Prof. Mrs. Vaidehi Deshpande, Head of Department (Computer Science) for his kind and helpful support regarding the project and the course.

I am grateful and thankful to all teaching and non-teaching staff of computer science department who shared their years of experience, excellent support and blossoms of suggestion with me for developing the project.

I would like to thank to all teaching and non-teaching staff of physics department, who helped me a lot during the project working.

Finally heartfelt appreciation to those countless friends who have contributed greatly to the success of my project.

Mr. Shaikh Mohammed Amir
&
Mr. Sayed Mohammed Owais

INDEX

SR.NO		TOPIC	PAGE NO
1		PROJECT INTRODUCTION	
	1.1	PROJECT OUTLINE.	2
	1.2	PROJECT AIMS.	3
2		PROJECT BACKGROUND STUDY	
	2.1	BRIEF HISTORY OF CLIMATE CHANGE IMPACT ON FOOD SUPPLY	5
	2.2	Data Exploration & Feature extraction	8
	2.3	BRIEF HISTORY OF LIBRARY.	11
	2.4	ABOUT ANACONDA JUPYTER	13
4		FLOW CHART	
	4.1	FLOW CHART DIAGRAM.	15
	4.2	FLOW CHART DESCRIPTION.	15
5		SYSTEM REQUIREMENTS	
	5.1	HARDWARE REQUIREMENTS.	16
	5.2	SOFTWARE REQUIREMENTS.	17
6		SCREEN SHOTS	34
7		CODE DEVELOPMENT	55
8		PROBLEM FACED	57
9		SCOPE & LIMITATIONS	59
10		FUTURE ENHANCEMENT	61
11		BIBLIOGRAPHY AND REFERENCES	63

CLIMATE CHANGE IMPACT ON FOOD SUPPLY

1: PROJECT INTRODUCTION

1.1 INTRODUCTION

Droughts, fires, storms, and floods have become vigorous and more periodical, these repercussions of climate variability have become progressively visible. The global ecosystem is changing, including the environmental resources and cultivation on which we are heavily dependent. Hence climate change has become one of the most prominent challenges faced by humanity, and we, as machine learning enthusiasts, aim to play our part in it. Under our project, we will be exploring the role of machine learning in mitigating the adverse effects of climate change and helping society adapt to these changes. Our primary goal is to understand the changing climate and predict its long-term effect on the global food supply/production. In this study, we also saw that linear regression models are less accurate at capturing the variability of crop yield than non-linear models.

We started collecting data on the temperature of the earth from the 1850s and the alarming fact is that the rate at which temperature is increasing is the fastest ever. It is expected that by the end of the 21st century the temperature rise would be somewhere between 1.5 to 5.7°C. The emission of greenhouse gases such as methane, water vapor, ozone, carbon dioxide, and chemicals like chlorofluorocarbons (CFCs) is known as Global Warming. Most of these gases are released in the atmosphere from airplane exhaust, fossil fuel extraction, car exhaust, and factory farming, these gases warm the earth by trapping the heat in the atmosphere as they have an insulating effect.

Global warming at its pinnacle can be the cause that leads to an increase in the frequency of extreme meteorological events such as floods, droughts that affects food production around the world. Over a long period, the small differences in average temperature when compounded result in disproportionately huge changes in the recurrence of these extreme events.

Despite the prodigious advancements in science and crop yield potential, food production remains greatly dependent on climate, because temperature, solar radiation, and precipitation are the main drivers in the cultivation of crops. Pest infestations and plant diseases, as well as the demand for and supply of irrigation water, are governed by climate.

Hence, it is clear later or sooner the build-up of greenhouse gases in the atmosphere remains ignored and unchecked it will warm the surface of the earth. The biophysical processes of respiration and photosynthesis, insects, the infestations of weeds and, the whole hydrological and thermal dominion governing our agricultural systems are expected to be affected by such a warming trend.

Under our project, we use statistical methods to understand the effect of climate variability on global food production/supply in greater depth and propose solutions to tackle the adverse effects caused by climate change and extreme meteorological events.

1.2 Objectives

The objective of this project is to examine the impact of climate change on food supply, with a focus on understanding how changes in weather patterns, temperature, and precipitation are affecting crop production, food security, and access to nutritious food. The project aims to provide a comprehensive analysis of the current state of food supply in the face of climate change and to identify potential solutions to mitigate the negative impacts.

In recent years, the world has experienced significant changes in climate, resulting in extreme weather events such as droughts, floods, and heatwaves. These changes are having a profound impact on food production, as crops are affected by changes in temperature, water availability, and the frequency and intensity of extreme weather events. In addition, changes in climate are also affecting food distribution and access, as transportation systems and supply chains are disrupted by extreme weather events.

The objective of this project is to examine these issues in greater detail, focusing on how climate change is affecting food security, food access, and nutrition. The project will utilize a variety of research methods, including literature reviews, data analysis, and case studies, to provide a comprehensive assessment of the current state of food supply in the face of climate change. The project will also explore potential solutions to mitigate the negative impacts of climate change on food supply, such as the development of new crop varieties, the implementation of sustainable agricultural practices, and the creation of more resilient food systems.

Ultimately, the goal of this project is to raise awareness about the impact of climate change on food supply and to provide stakeholders with the information they need to make informed decisions about how to address this critical issue. By identifying the key challenges and potential solutions, this project aims to contribute to a more sustainable and resilient food supply system in the face of a changing climate.

2: PROJECT BACKGROUND STUDY

2.1 Background

Not any particular kind of technology can make the world a better place. Machine learning has given many contributions across different domain areas. ML has played a major role in weather forecasting to satellite imagery. Through remote sensing, ML can enable automatic monitoring (for example, detecting deforestation, gathering data on structures). It has the potential to speed up the process of scientific discovery and also be used to increase the efficiency of systems.

Using ML to combat changes in the climate has the potential to assist everyone while also furthering the discipline of ML. Problems mentioned in this paper like global warming, carbon dioxide emission, food wastage can be controlled or by prior analysis. Furthermore, taking real action on climate issues necessitates collaboration with fields both inside and outside of computer science, which can lead to multidisciplinary methodological advancements like enhanced physics constrained machine learning approaches.

Climate change is a challenge and a threat to the food security of the whole world. As we all know the emission of greenhouse gases is increasing in the atmosphere every year and therefore due to this greenhouse effect, the temperature is also rising. The average global temperature is predicted to rise by 2 degrees Celsius by the end of the 21st century.

2.1.2 Solution Overview

One way to address the impact of climate change on food supply is to use predictive models that can estimate changes in agricultural production based on climate variables. These models can incorporate various factors such as temperature, precipitation, soil moisture, and crop growth stages, to predict the yield of different crops in different regions.

To develop such models, historical climate and agricultural data can be used to train machine learning algorithms. These algorithms can then be used to forecast the effects of future climate scenarios on agricultural production, allowing policymakers and farmers to make more informed decisions.

In addition to predictive models, another approach is to develop climate-resilient crops through plant breeding and genetic engineering. These crops are designed to withstand the stressors of climate change, such as drought, heat, and floods, while maintaining their productivity and nutritional value.

Furthermore, improving food distribution systems can also help mitigate the impact of climate change on food supply. By improving transportation and storage systems, reducing food waste, and increasing access to food in food-insecure regions, we can ensure that even in times of agricultural shocks, people have access to the food they need.

Overall, a multi-faceted approach is required to address the impact of climate change on food supply. By using predictive models, developing climate-resilient crops, and improving food distribution systems, we can help ensure food security in the face of a changing climate.

2.1.3 Data Sources

Information for crop yield was assembled from FAOSTAT, which offers worldwide insights on food and agribusiness, and gives admittance to throughout 3 million time-series and cross-sectional information identifying with crop yield.

As rainfall dramatically affects food production, for this task precipitation, each year data was assembled from the World Data Bank notwithstanding the normal temperature for every country. The last information outline for normal precipitation incorporates; nation, year, and normal precipitation each year. The information outline begins from 1985 to 2017, on other hand, the normal temperature information outline incorporates nation, year, and normal recorded temperature. The temperature information outline begins at 1743 and closes in 2013. The variety in years will think twice about gathered information a piece was joining a year reach to exclude any invalid qualities. Information for pesticides was gathered from FAO, it is prominent that it begins in 1990 and closes in 2016. Consolidating these information outlines, it's normal that the year reach will begin from 1990 and close in 2013, which is 23 years of information.

For examining the conduct of the world's surface temperature, we have utilized The Berkeley Earth Surface Temperature Study which joins 1.6 billion temperature reports from 16 prior documents. It is well bundled and considers cutting into fascinating subsets (for instance by country). They distribute the source information and the code for the changes they applied. They likewise use strategies that permit climate perceptions from more limited-time series to be incorporated, which means fewer perceptions should be discarded.

2.1.4 Data Pre-processing

Data Pre-processing is a strategy that is utilized to convert the crude information into a perfect informational collection. The information is accumulated from various sources, it is gathered in a crude arrangement which isn't doable for the investigation. By applying unique methods like supplanting missing qualities and invalid qualities, we can change information into a reasonable organization. The last step on information pre-processing is the partitioning of training and testing information. The information normally will more often than not be parted inconsistent since preparing the model typically needs however many data points as could be expected. The preparation dataset is the underlying dataset used to prepare ML calculations to learn and create the right expectations.

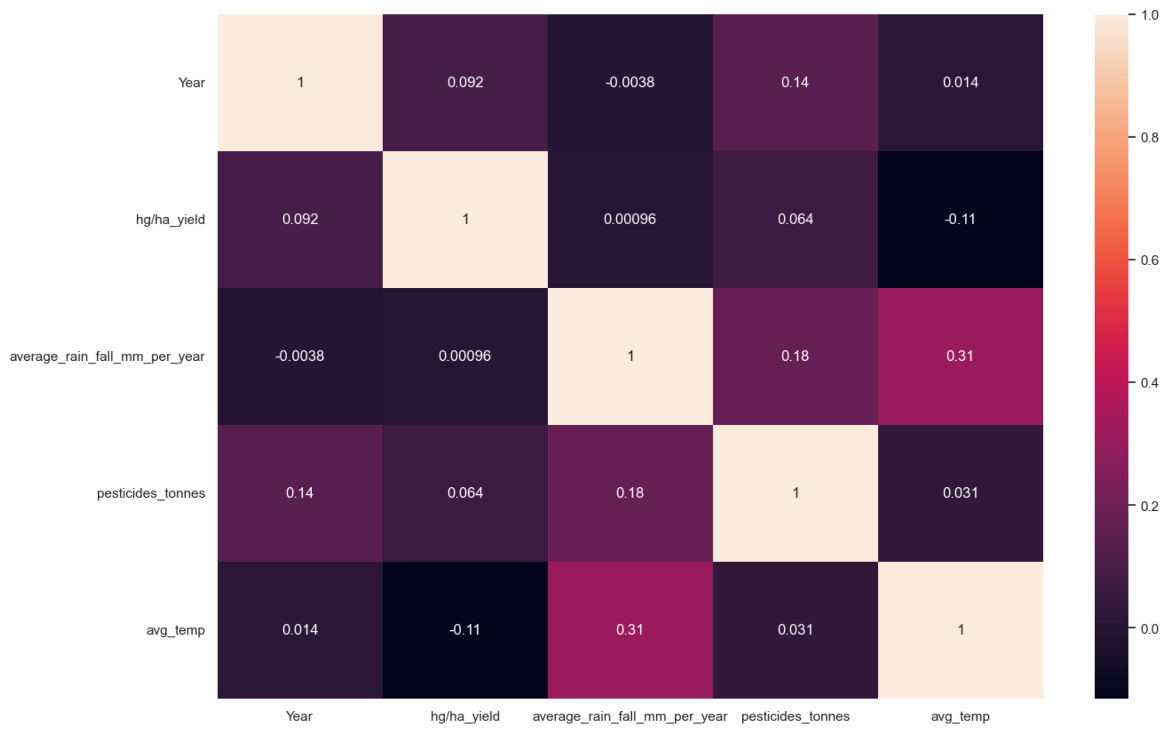
For setting up the harvest yield dataset for displaying, we dropped a couple of sections like Region Code, Area, Thing Code, and so on, which were of no utilization to the examination. Likewise, we renamed the Value attribute to hg/ha_yield to make it clear to perceive that this quality addresses crops yields creation esteem.

In the merged dataset, there are two categorical columns Many ML algorithms cannot operate on categorical data directly, therefore the labelled data must be converted to numeric format. For this, we use One hot encoding. Also, this dataset contains highlights with profoundly differing extents, units, and reaches. The highlights with high extents will make an appearance much more in the estimations than highlights with low sizes. To handle this impact, we want to carry all highlights to similar degree of extents. In our analysis, we have used MinMaxScaler to tackle this problem.

For the earth surface temperature dataset, we handled the missing data by performing listwise deletion because data was missing in chunks and the fact that it was time-series data. We also converted all the date columns to the proper format and created some new features to perform exploratory analysis.

2.2 Data Exploration & Feature extraction

To explore the relationship between the attributes of the dataset we have visualized the correlation matrix as a heatmap.



It is clear from the heatmap above that the attributes have no correlations with each other. Therefore, we consider other elements that influence the yield of any crop and its creation. These are essentially the highlights that help in foreseeing the creation of any yield throughout the year. In our research, we incorporate elements like rainfall, temperature, surface area, moistness, and wind speed.

To understand the behaviour of the earth's surface temperature, we grouped the data by continents and plotted the average land temperature from the year 1750 to 2010.

2.2.1 Model comparison and selection

The best technique to get answers for the problem of crop yield is ML. First evaluate and compare the model which will best complement our data set post which we will decide the best model to use for our dataset. There are a lot of machine learning algorithms available for predicting the yield of crop. In this paper we include the following machine learning algorithms for selection and accuracy comparison:

2.2.2 Gradient Boosting

It is an AI strategy utilized in relapse and order errands, among others. It gives a forecast model as an outfit of frail expectation models, which are normally choice trees. When a choice tree is a feeble student, the subsequent calculation is called gradient boosting tree. 89% accuracy is achieved when this algorithm is applied to our dataset.

Gradient Boosting is a boosting technique that sequentially builds a set of weak learners in order to improve the performance of the overall model. The basic idea of Gradient Boosting is to minimize the loss function of the model by adding weak learners to the model iteratively. Each new weak learner is trained on the residuals (difference between the actual and predicted values) of the previous model, and the final prediction is the sum of the predictions of all weak learners.

2.2.3 Random Forest

It can break down crop development identified with the current climatic conditions and biophysical change. Random forest the calculation makes choice trees on various information tests and afterward foresee the information from every subset and afterward by casting a ballot gives the better answer for the Framework. Forest uses the stowing strategy to train the information which expands the precision of the Result. 68% accuracy is achieved when this algorithm is applied to our dataset.

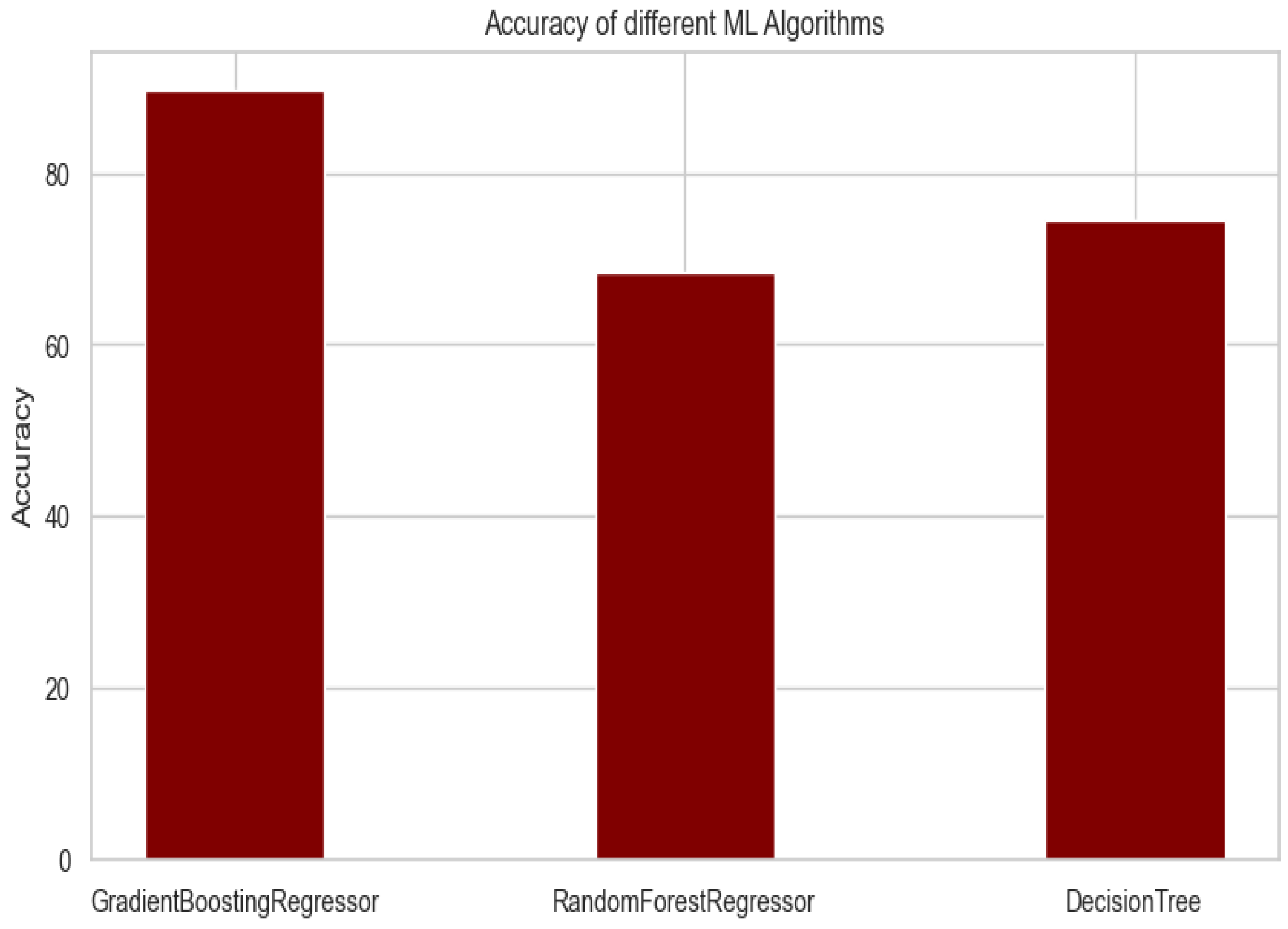
2.2.4 Decision Tree

It is a unique gadget that uses a flowchart like tree structure or is a model of decisions and all of their utility, expected results, input costs including results.74% accuracy is achieved when this algorithm is applied to our dataset

Decision Tree Algorithm is used in both Regression and classification related problems because of which it is used in many competitions and hence has become one of the most used algorithms in Machine Learning.

A decision tree normally begins with a solitary hub, which branches into potential results. Every one of those results prompts extra hubs, which branch off into different conceivable outcomes. A decision hub, addressed by a square, demonstrates a decision to be made, and an end hub shows the result of a decision way. A hub addresses a solitary information variable (X) and a split point on that factor, accepting the variable is numeric. The leaf hubs (additionally called terminal hubs) of the tree contain a yield variable (y) which is utilized to make an expectation.

The decision tree is showing up at a gauge by posing a progression of inquiries to the information, each question limiting our potential qualities until the model gets sufficiently sure to make a solitary forecast. The request for the inquiry just as their substance is being controlled by the model. Likewise, the inquiries posed are all in a Valid/Bogus structure. Decision trees relapse utilizes mean squared mistake (MSE) to choose to part a hub into at least two sub-hubs.



2.3 Libraries

2.3.1 Scikit-learn (sklearn)

Scikit-learn, commonly abbreviated as sklearn, is a free and open-source machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, and DBSCAN, and is designed to interoperate with the numerical and scientific libraries NumPy and SciPy. It also provides tools for data pre-processing, model selection, and evaluation, as well as a range of utilities for model visualization, feature selection, and dimensionality reduction.

Installation

```
pip install -U scikit-learn
```

Scikit-learn provides a wide range of tools for implementing and evaluating machine learning models. It has a user-friendly interface and is extensively used in academia and industry for machine learning tasks. It is structured as a collection of submodules, each containing specific functionalities for data processing, model training, and evaluation. Some of the key modules of scikit-learn are:

1. Pre-processing: This module contains functions for data cleaning, normalization, and feature scaling.

2. Model selection: This module provides tools for selecting the best model based on the given dataset, using techniques such as cross-validation and grid search.

3. Supervised learning: This module contains various algorithms for classification, regression, and time-series forecasting, such as linear regression, decision trees, and support vector machines.

4. Unsupervised learning: This module contains algorithms for clustering, dimensionality reduction, and anomaly detection, such as k-means, principal component analysis (PCA), and isolation forest.

Scikit-learn is an important library for anyone working with machine learning in Python and can greatly simplify the process of implementing and evaluating machine learning models

2.3.2 Flask

Flask is a lightweight web application framework for Python that allows developers to build web applications quickly and easily. It is based on the WSGI toolkit and Jinja2 template engine, and is designed to be easy to use and flexible.

Installation: Flask can be installed using pip command as follows: pip install Flask

Features:

1. Flask is a micro-framework and does not require particular tools or libraries. It is easy to set up and requires minimal coding effort.
2. Flask is a versatile framework and can be used for building web applications ranging from small, single-page applications to large, complex applications.
3. Flask supports a variety of extensions and libraries, including SQL databases, caching, and authentication.
4. Flask is based on the Python programming language, which makes it easy to learn and understand.

Overall, Flask is a powerful and flexible web framework that allows developers to build web applications quickly and easily.

2.3.3 pickle

Pickle is a Python module used for serializing and de-serializing Python objects. It can convert complex Python objects, such as lists, dictionaries, and classes, into a byte stream, allowing them to be stored in a file or transferred across a network. The pickled object can later be de-serialized and turned back into its original form. This makes it useful for tasks such as caching and distributed processing, where the same object needs to be used across multiple processes or computers.

Pickling is the process of converting a Python object into a byte stream, and unpickling is the process of converting the byte stream back into an object. Pickle provides a simple and efficient way to store and retrieve data, without the need for complex database systems or file formats.

To use Pickle, you simply need to import the module and use the dump () and load () functions to serialize and de-serialize objects. Pickle is built into the Python standard library, so it does not need to be installed separately.

Some common use cases for Pickle include caching the results of expensive computations, transferring data between processes or computers, and storing complex objects in a database or file. However, it is important to note that Pickle is not a secure or robust way to store data, as the byte stream can be manipulated or maliciously modified.

2.4 Jupyter

Jupyter is a web-based interactive computing platform that allows users to create and share documents called notebooks, which contain live code, equations, visualizations, and narrative text. It supports multiple programming languages, including Python, R, and Julia, and can be used for data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and more.

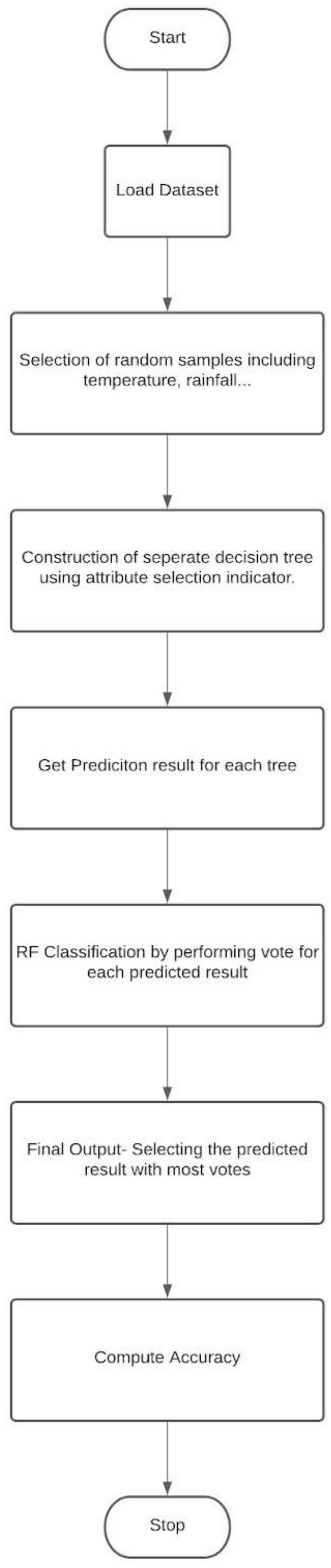
Jupyter notebooks consist of a series of cells, each of which can contain code, Markdown, or raw text. The code cells can be executed independently and the output is displayed directly below the cell. This allows for an iterative and interactive approach to programming, where users can modify the code and see the results immediately.

Jupyter also supports the use of various libraries and tools for data analysis, such as NumPy, Pandas, Matplotlib, and Scikit-learn. Users can install and import these libraries within their notebooks to perform complex data analysis tasks.

Jupyter can be installed locally on a computer, or it can be run in the cloud using services like Anaconda or Microsoft Azure. It is widely used by data scientists, researchers, and educators, and has a large and active community of developers who contribute to its development and maintenance.

4: FLOW CHART

4.1 FLOW CHART DIAGRAM.


Start
Load Dataset
Selection of random samples including temperature, rainfall...
Construction of seperate decision tree using attribute selection indicator.
Get Prediciton result for each tree
RF Classification by performing vote for each predicted result
Final Output- Selecting the predicted result with most votes
Compute Accuracy
Stop

4.2 FLOW CHART DESCRIPRTION

- Data Collection: The first step is to collect data on food production, climate variables, and socio-economic factors. This data is usually obtained from sources such as the World Bank, FAOSTAT, and other relevant agencies.
- Data Cleaning: The collected data may contain errors, missing values, and outliers. Therefore, it is necessary to clean the data to remove such anomalies.
- Data Pre-processing: The data is pre-processed to extract meaningful information that can be used for analysis. This includes scaling, normalization, and feature selection.
- Data Analysis: The pre-processed data is then analysed to determine the impact of climate change on food production. This involves various statistical and machine learning techniques such as correlation analysis, regression analysis, and machine learning algorithms.
- Visualization: The analysis results are then visualized using charts, graphs, and other visualization tools. This helps to better understand the impact of climate change on food production.
- Model Building: Machine learning algorithms such as Decision Trees, Random Forests, and Gradient Boosting are used to build models that can predict the impact of climate change on food production.
- Model Training: The built models are then trained using the pre-processed data. This involves splitting the data into training and testing sets to evaluate the model's performance.
- Model Validation: The trained models are validated using various performance metrics such as accuracy, precision, recall, and F1 score. This helps to determine the model's effectiveness in predicting the impact of climate change on food production.
- Deployment: The validated models are then deployed for use in real-world scenarios. This involves integrating the models with existing systems and software platforms.
- Monitoring: The deployed models are continuously monitored to ensure their accuracy and effectiveness in predicting the impact of climate change on food production.
- Feedback Loop: Feedback from users and stakeholders is collected and used to improve the models and data analysis techniques used in the project. This creates a continuous feedback loop that helps to improve the accuracy and effectiveness of the project over time.
-

5: SYSTEM REQUIREMENTS

5.1 Hardware Requirements:

The following are the minimum hardware requirements needed:

• Windows/Linux/Mac OS Machine

• GPU • RAM 4GB or more.

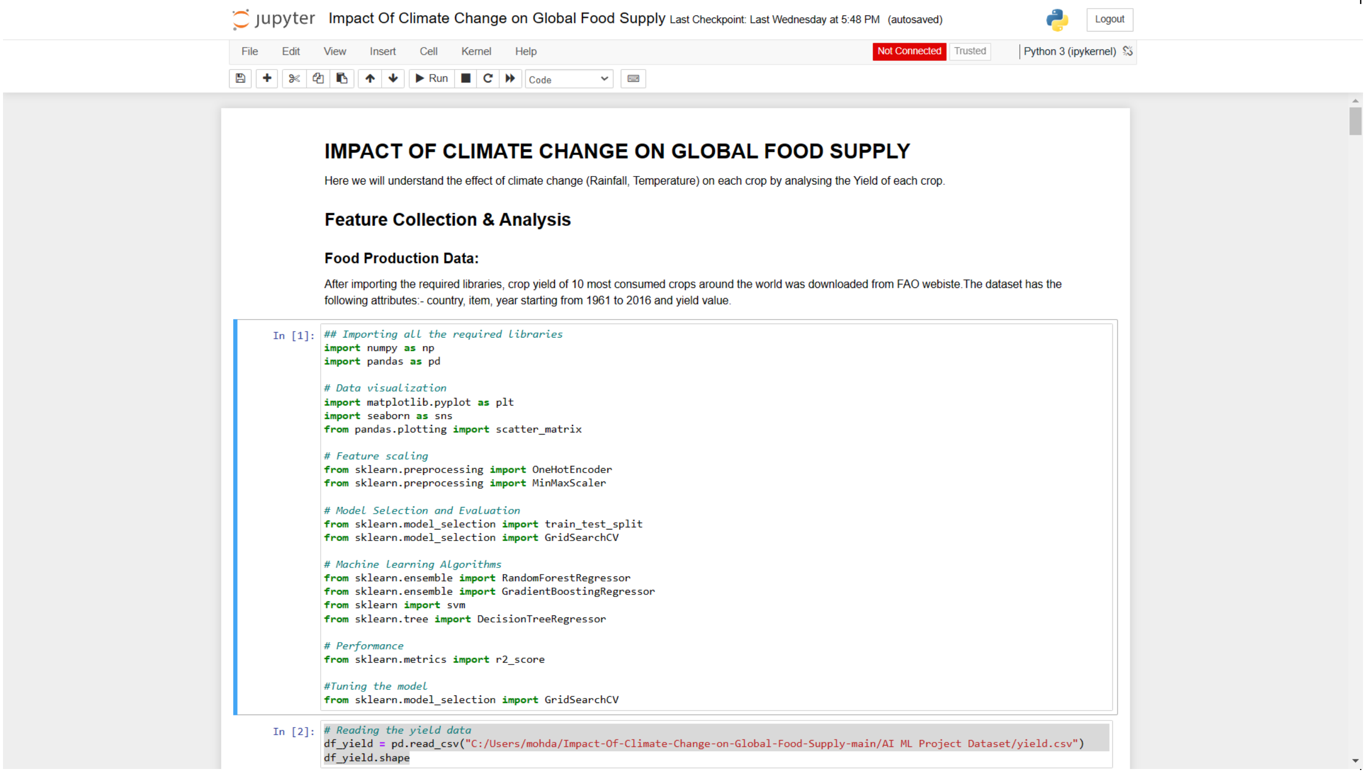
• System type: 64-bit OS, x64-based processor

5.2 Software Requirements:

The following are the minimum software requirements needed:

• Python 3
• Sklearn, Flask & Pickle
• Packages NumPy, Pandas, Matplotlib
• Jupyter Notebook

6: SCREEN SHOTS


jupyter
Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

IMPACT OF CLIMATE CHANGE ON GLOBAL FOOD SUPPLY

Here we will understand the effect of climate change (Rainfall, Temperature) on each crop by analysing the Yield of each crop.

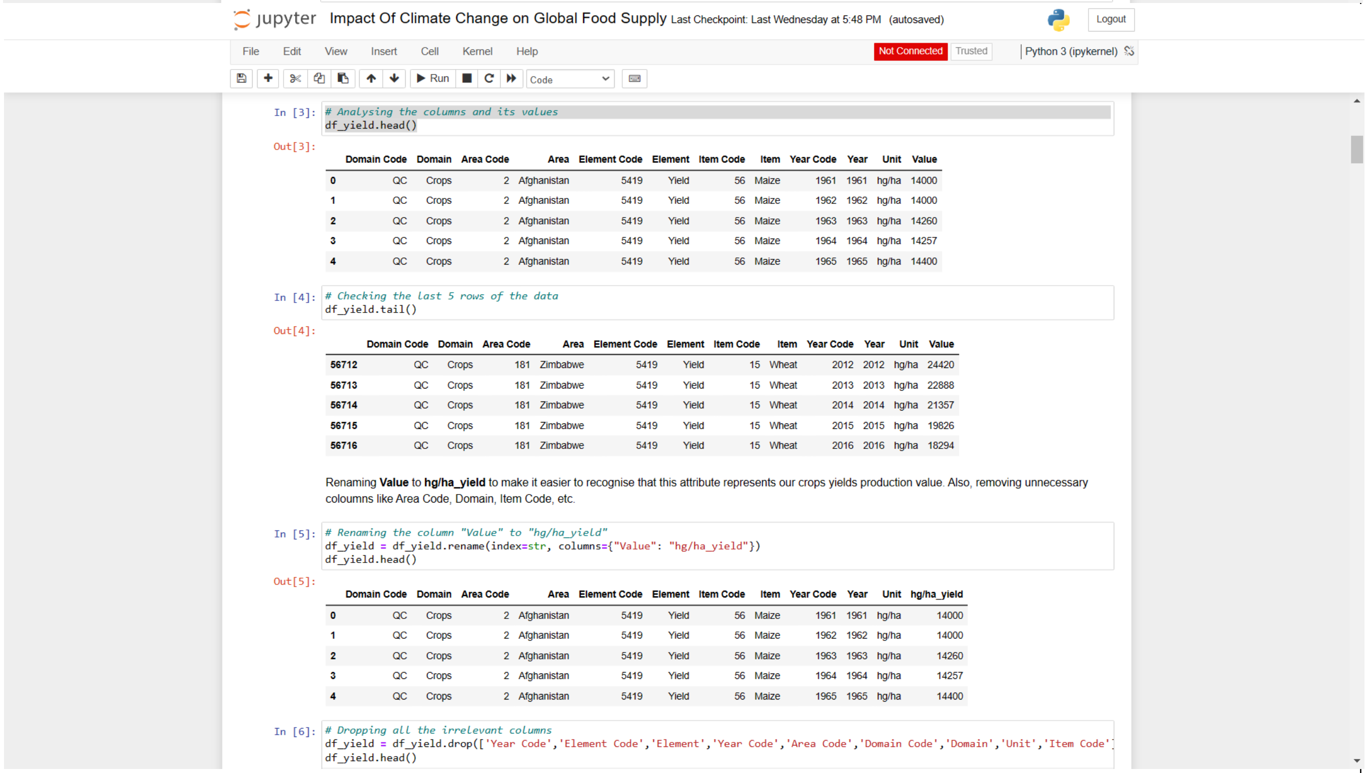
Feature Collection & Analysis

Food Production Data:

After importing the required libraries, crop yield of 10 most consumed crops around the world was downloaded from FAO webiste.The dataset has the following attributes:- country, item, year starting from 1961 to 2016 and yield value.

```
In [1]: ## Importing all the required libraries
        import numpy as np
        import pandas as pd

        # Data visualization
        import matplotlib.pyplot as plt
        import seaborn as sns
        from pandas.plotting import scatter_matrix

        # Feature scaling
        from sklearn.preprocessing import OneHotEncoder
        from sklearn.preprocessing import MinMaxScaler

        # Model Selection and Evaluation
        from sklearn.model_selection import train_test_split
        from sklearn.model_selection import GridSearchCV

        # Machine Learning Algorithms
        from sklearn.ensemble import RandomForestRegressor
        from sklearn.ensemble import GradientBoostingRegressor
        from sklearn import svm
        from sklearn.tree import DecisionTreeRegressor

        # Performance
        from sklearn.metrics import r2_score

        #Tuning the model
        from sklearn.model_selection import GridSearchCV
```

```
In [2]: # Reading the yield data
        df_yield = pd.read_csv("C:/Users/mohda/Impact-Of-Climate-Change-on-Global-Food-Supply-main/AI ML Project Dataset/yield.csv")
        df_yield.shape
```


jupyter
Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

```
In [3]: # Analysing the columns and its values
        df_yield.head()
```

Out[3]:

	Domain Code	Domain	Area Code	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	Value
0	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1961	1961	hg/ha	14000
1	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1962	1962	hg/ha	14000
2	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1963	1963	hg/ha	14260
3	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1964	1964	hg/ha	14257
4	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1965	1965	hg/ha	14400

```
In [4]: # Checking the last 5 rows of the data
        df_yield.tail()
```

Out[4]:

	Domain Code	Domain	Area Code	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	Value
56712	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2012	2012	hg/ha	24420
56713	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2013	2013	hg/ha	22888
56714	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2014	2014	hg/ha	21357
56715	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2015	2015	hg/ha	19826
56716	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2016	2016	hg/ha	18294

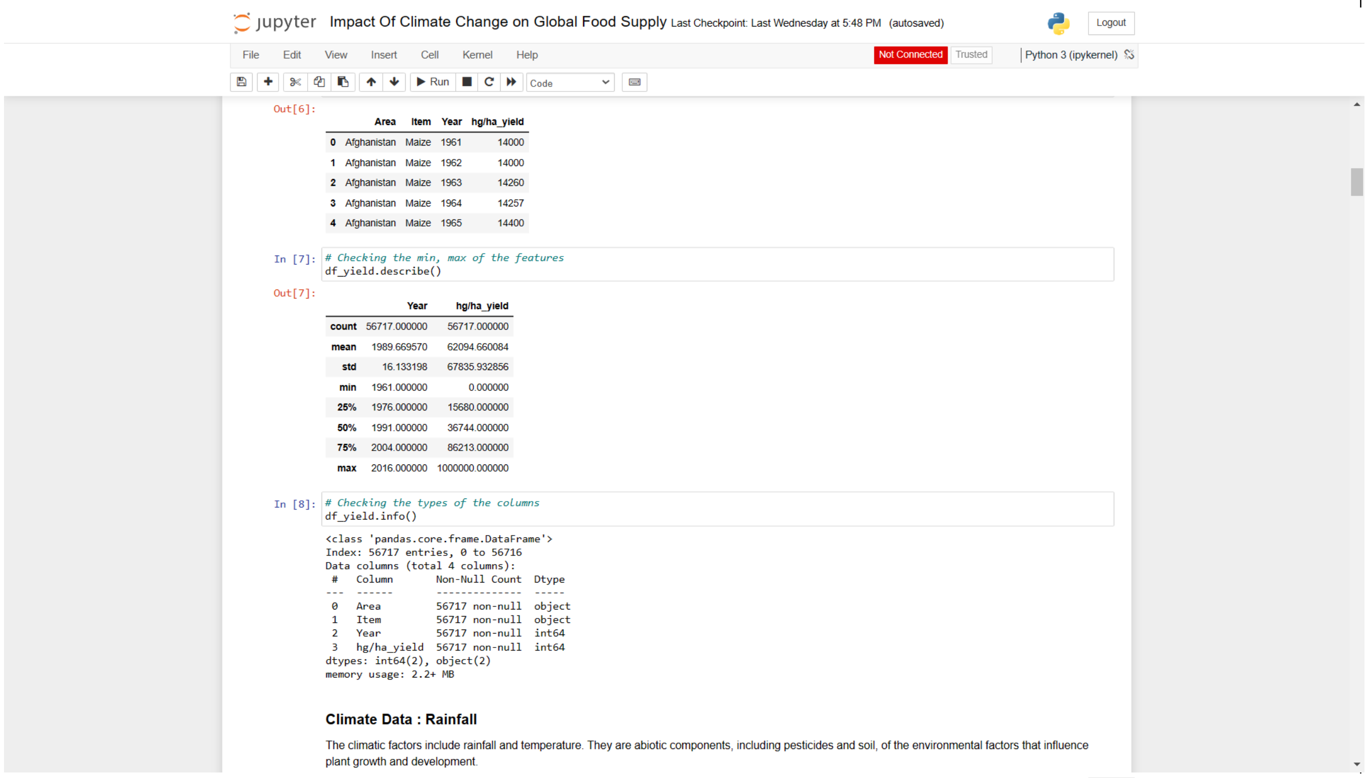
Renaming **Value** to **hg/ha_yield** to make it easier to recognise that this attribute represents our crops yields production value. Also, removing unnecessary coloumns like Area Code, Domain, Item Code, etc.

```
In [5]: # Renaming the column "Value" to "hg/ha_yield"
        df_yield = df_yield.rename(index=str, columns={"Value": "hg/ha_yield"})
        df_yield.head()
```

Out[5]:

	Domain Code	Domain	Area Code	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	hg/ha_yield
0	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1961	1961	hg/ha	14000
1	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1962	1962	hg/ha	14000
2	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1963	1963	hg/ha	14260
3	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1964	1964	hg/ha	14257
4	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1965	1965	hg/ha	14400

```
In [6]: # Dropping all the irrelevant columns
        df_yield = df_yield.drop(['Year Code','Element Code','Element','Year Code','Area Code','Domain Code','Domain','Unit','Item Code']
        df_yield.head()
```


jupyter
Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

Out[6]:

	Area	Item	Year	hg/ha_yield
0	Afghanistan	Maize	1961	14000
1	Afghanistan	Maize	1962	14000
2	Afghanistan	Maize	1963	14260
3	Afghanistan	Maize	1964	14257
4	Afghanistan	Maize	1965	14400

In [7]:
```
# Checking the min, max of the features
df_yield.describe()
```

Out[7]:

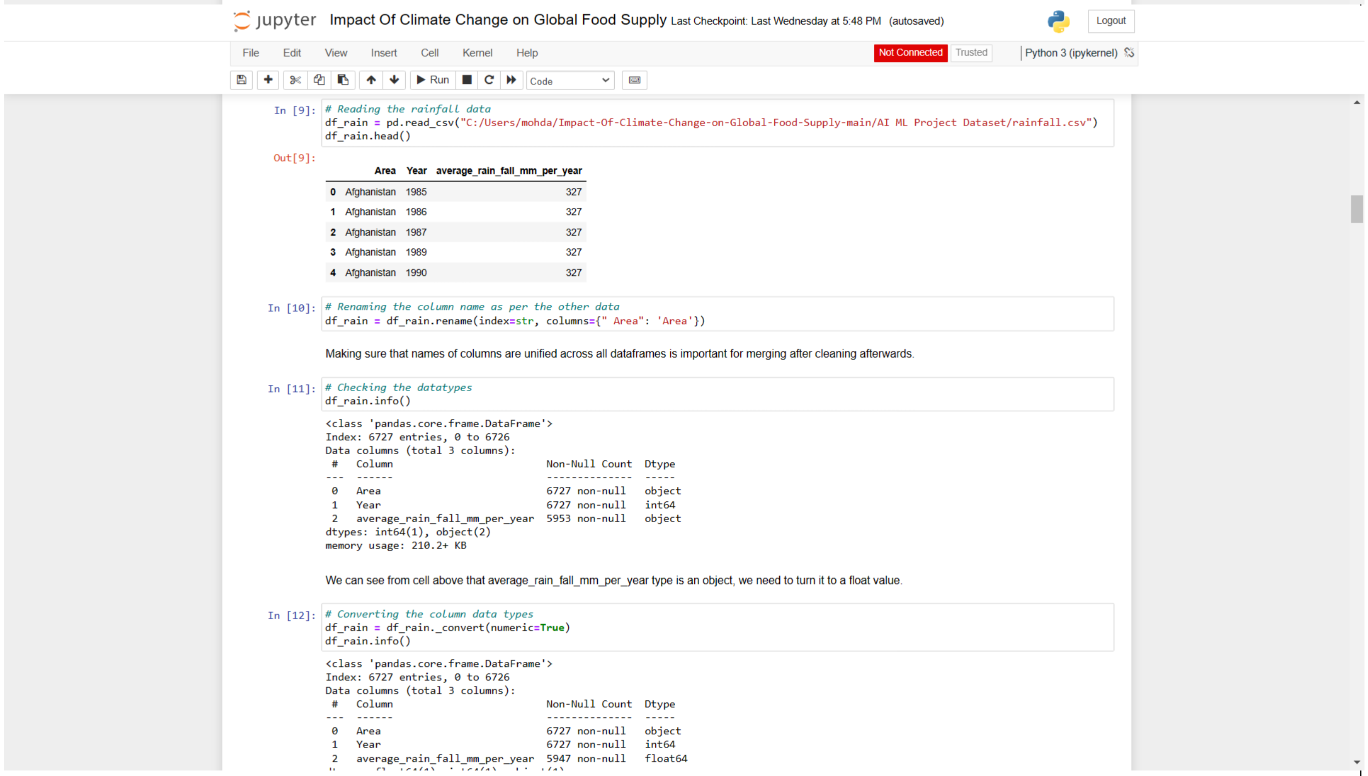
	Year	hg/ha_yield
count	56717.000000	56717.000000
mean	1989.669570	62094.660084
std	16.133198	67835.932856
min	1961.000000	0.000000
25%	1976.000000	15680.000000
50%	1991.000000	36744.000000
75%	2004.000000	86213.000000
max	2016.000000	1000000.000000

In [8]:
```
# Checking the types of the columns
df_yield.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 56717 entries, 0 to 56716
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Area         56717 non-null  object
 1   Item         56717 non-null  object
 2   Year         56717 non-null  int64 
 3   hg/ha_yield  56717 non-null  int64 
dtypes: int64(2), object(2)
memory usage: 2.2+ MB
```

Climate Data : Rainfall

The climatic factors include rainfall and temperature. They are abiotic components, including pesticides and soil, of the environmental factors that influence plant growth and development.


jupyter
Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

In [9]:
```
# Reading the rainfall data
df_rain = pd.read_csv("C:/Users/mohda/Impact-Of-Climate-Change-on-Global-Food-Supply-main/AI ML Project Dataset/rainfall.csv")
df_rain.head()
```

Out[9]:

	Area	Year	average_rain_fall_mm_per_year
0	Afghanistan	1985	327
1	Afghanistan	1986	327
2	Afghanistan	1987	327
3	Afghanistan	1989	327
4	Afghanistan	1990	327

In [10]:
```
# Renaming the column name as per the other data
df_rain = df_rain.rename(index=str, columns={" Area": 'Area'})
```

Making sure that names of columns are unified across all dataframes is important for merging after cleaning afterwards.

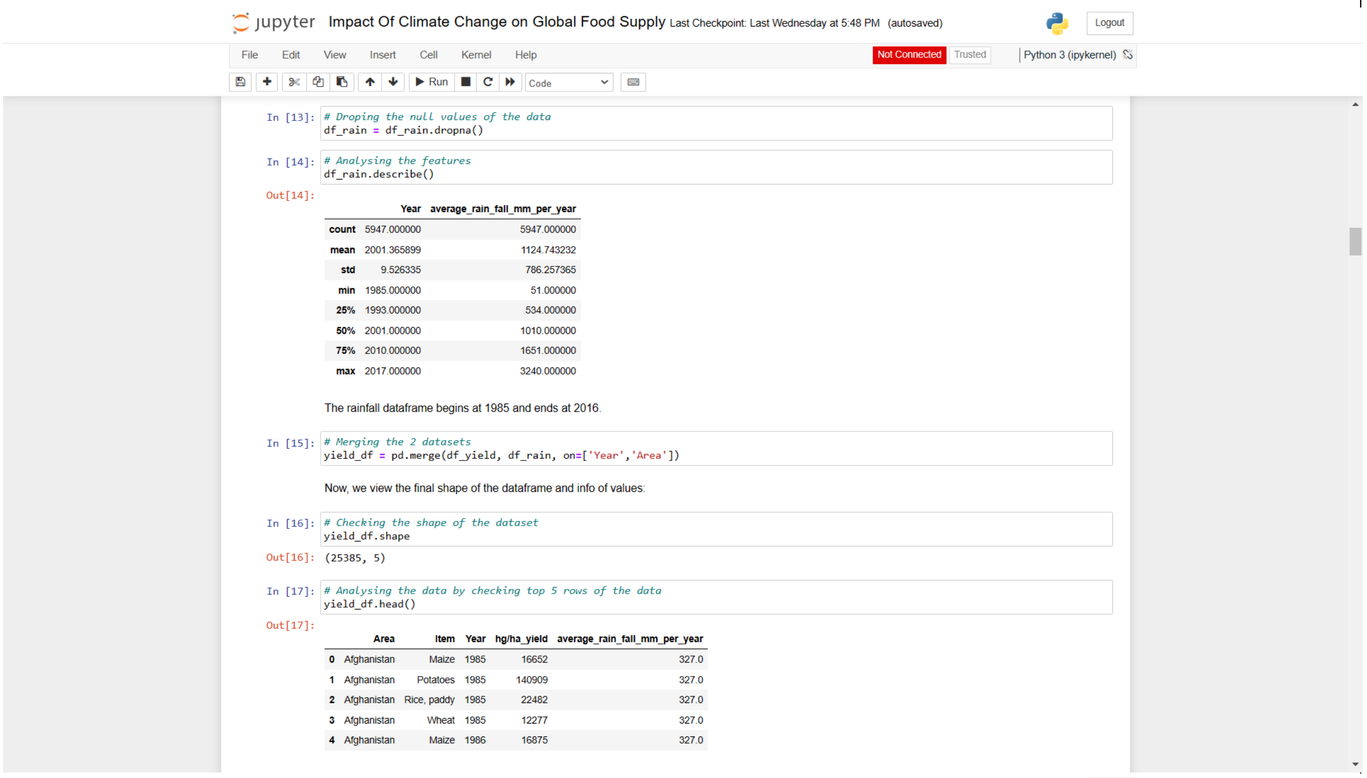
In [11]:
```
# Checking the datatypes
df_rain.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6727 entries, 0 to 6726
Data columns (total 3 columns):
 #   Column                         Non-Null Count  Dtype 
---  ------                         --------------  ----- 
 0   Area                           6727 non-null   object
 1   Year                           6727 non-null   int64 
 2   average_rain_fall_mm_per_year  5953 non-null   object
dtypes: int64(1), object(2)
memory usage: 210.2+ KB
```

We can see from cell above that average_rain_fall_mm_per_year type is an object, we need to turn it to a float value.

In [12]:
```
# Converting the column data types
df_rain = df_rain._convert(numeric=True)
df_rain.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6727 entries, 0 to 6726
Data columns (total 3 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   Area                           6727 non-null   object 
 1   Year                           6727 non-null   int64  
 2   average_rain_fall_mm_per_year  5947 non-null   float64
```


jupyter Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

```
In [13]: # Droping the null values of the data
         df_rain = df_rain.dropna()
```

```
In [14]: # Analysing the features
         df_rain.describe()
```

Out[14]:

	Year	average_rain_fall_mm_per_year
count	5947.000000	5947.000000
mean	2001.365899	1124.743232
std	9.526335	786.257365
min	1985.000000	51.000000
25%	1993.000000	534.000000
50%	2001.000000	1010.000000
75%	2010.000000	1651.000000
max	2017.000000	3240.000000

The rainfall dataframe begins at 1985 and ends at 2016.

```
In [15]: # Merging the 2 datasets
         yield_df = pd.merge(df_yield, df_rain, on=['Year','Area'])
```

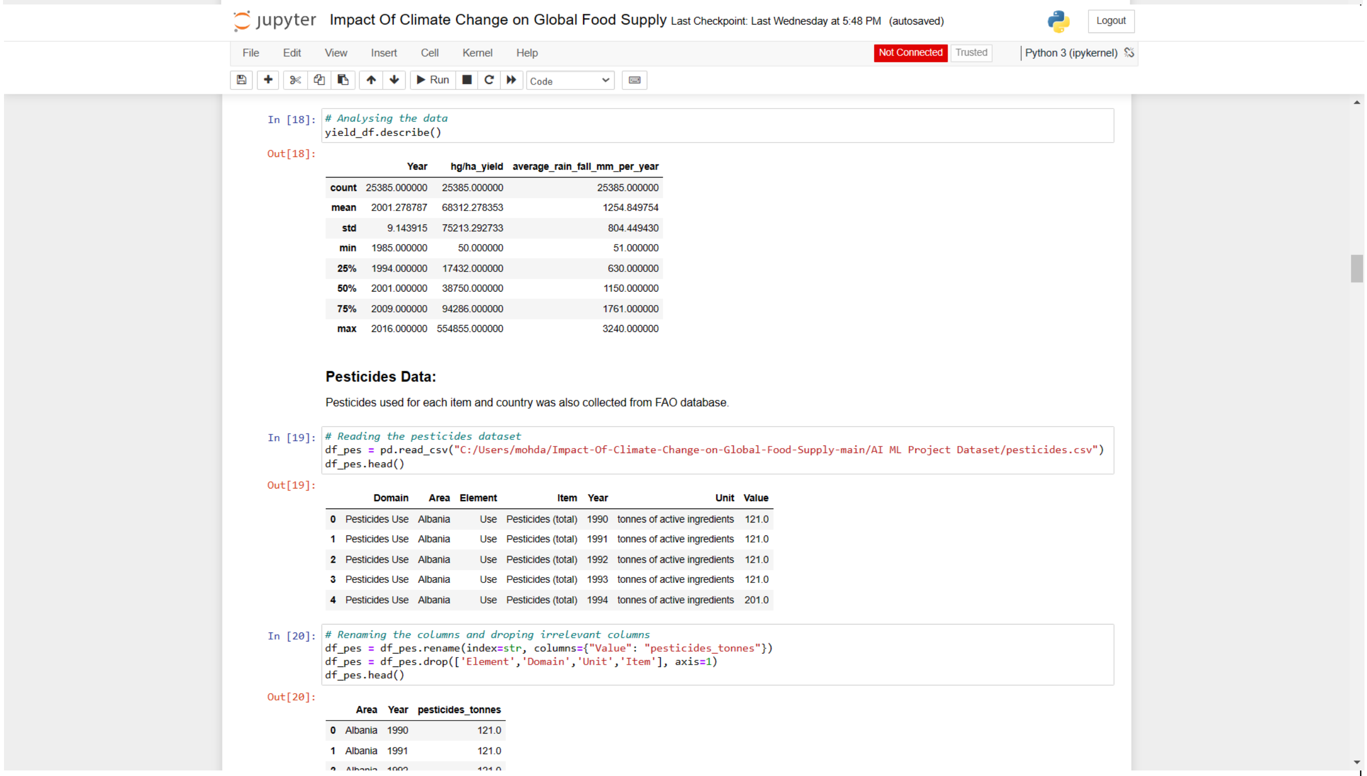
Now, we view the final shape of the dataframe and info of values:

```
In [16]: # Checking the shape of the dataset
         yield_df.shape
Out[16]: (25385, 5)
```

```
In [17]: # Analysing the data by checking top 5 rows of the data
         yield_df.head()
```

Out[17]:

	Area	Item	Year	hg/ha_yield	average_rain_fall_mm_per_year
0	Afghanistan	Maize	1985	16652	327.0
1	Afghanistan	Potatoes	1985	140909	327.0
2	Afghanistan	Rice, paddy	1985	22482	327.0
3	Afghanistan	Wheat	1985	12277	327.0
4	Afghanistan	Maize	1986	16875	327.0


jupyter Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

```
In [18]: # Analysing the data
         yield_df.describe()
```

Out[18]:

	Year	hg/ha_yield	average_rain_fall_mm_per_year
count	25385.000000	25385.000000	25385.000000
mean	2001.278787	68312.278353	1254.849754
std	9.143915	75213.292733	804.449430
min	1985.000000	50.000000	51.000000
25%	1994.000000	17432.000000	630.000000
50%	2001.000000	38750.000000	1150.000000
75%	2009.000000	94286.000000	1761.000000
max	2016.000000	554855.000000	3240.000000

Pesticides Data:

Pesticides used for each item and country was also collected from FAO database.

```
In [19]: # Reading the pesticides dataset
         df_pes = pd.read_csv("C:/Users/mohda/Impact-Of-Climate-Change-on-Global-Food-Supply-main/AI ML Project Dataset/pesticides.csv")
         df_pes.head()
```

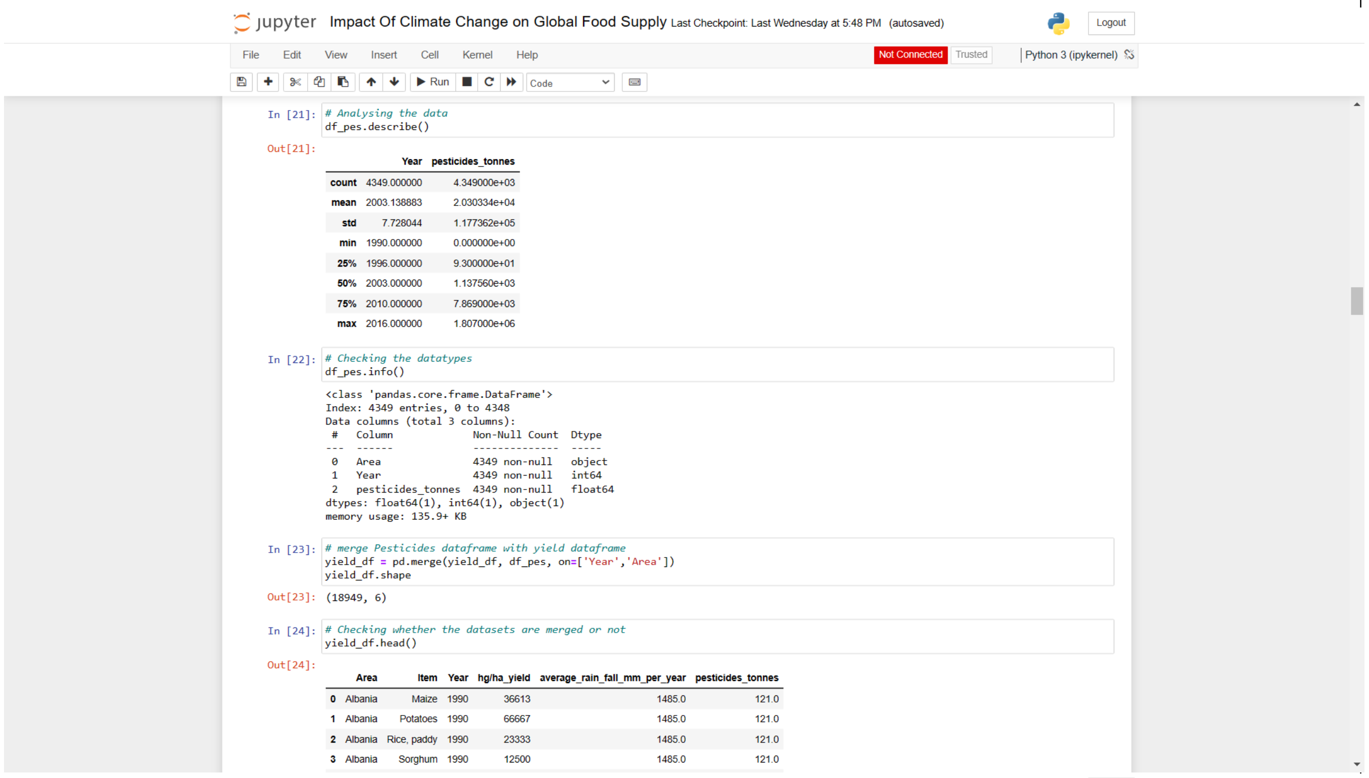
Out[19]:

	Domain	Area	Element	Item	Year	Unit	Value
0	Pesticides Use	Albania	Use	Pesticides (total)	1990	tonnes of active ingredients	121.0
1	Pesticides Use	Albania	Use	Pesticides (total)	1991	tonnes of active ingredients	121.0
2	Pesticides Use	Albania	Use	Pesticides (total)	1992	tonnes of active ingredients	121.0
3	Pesticides Use	Albania	Use	Pesticides (total)	1993	tonnes of active ingredients	121.0
4	Pesticides Use	Albania	Use	Pesticides (total)	1994	tonnes of active ingredients	201.0

```
In [20]: # Renaming the columns and droping irrelevant columns
         df_pes = df_pes.rename(index=str, columns={"Value": "pesticides_tonnes"})
         df_pes = df_pes.drop(['Element','Domain','Unit','Item'], axis=1)
         df_pes.head()
```

Out[20]:

	Area	Year	pesticides_tonnes
0	Albania	1990	121.0
1	Albania	1991	121.0
2	Albania	1992	121.0


jupyter Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

In [21]:
```
# Analysing the data
df_pes.describe()
```

Out[21]:

	Year	pesticides_tonnes
count	4349.000000	4.349000e+03
mean	2003.138883	2.030334e+04
std	7.728044	1.177362e+05
min	1990.000000	0.000000e+00
25%	1996.000000	9.300000e+01
50%	2003.000000	1.137560e+03
75%	2010.000000	7.869000e+03
max	2016.000000	1.807000e+06

In [22]:
```
# Checking the datatypes
df_pes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 4349 entries, 0 to 4348
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Area               4349 non-null   object
 1   Year               4349 non-null   int64
 2   pesticides_tonnes  4349 non-null   float64
dtypes: float64(1), int64(1), object(1)
memory usage: 135.9+ KB
```

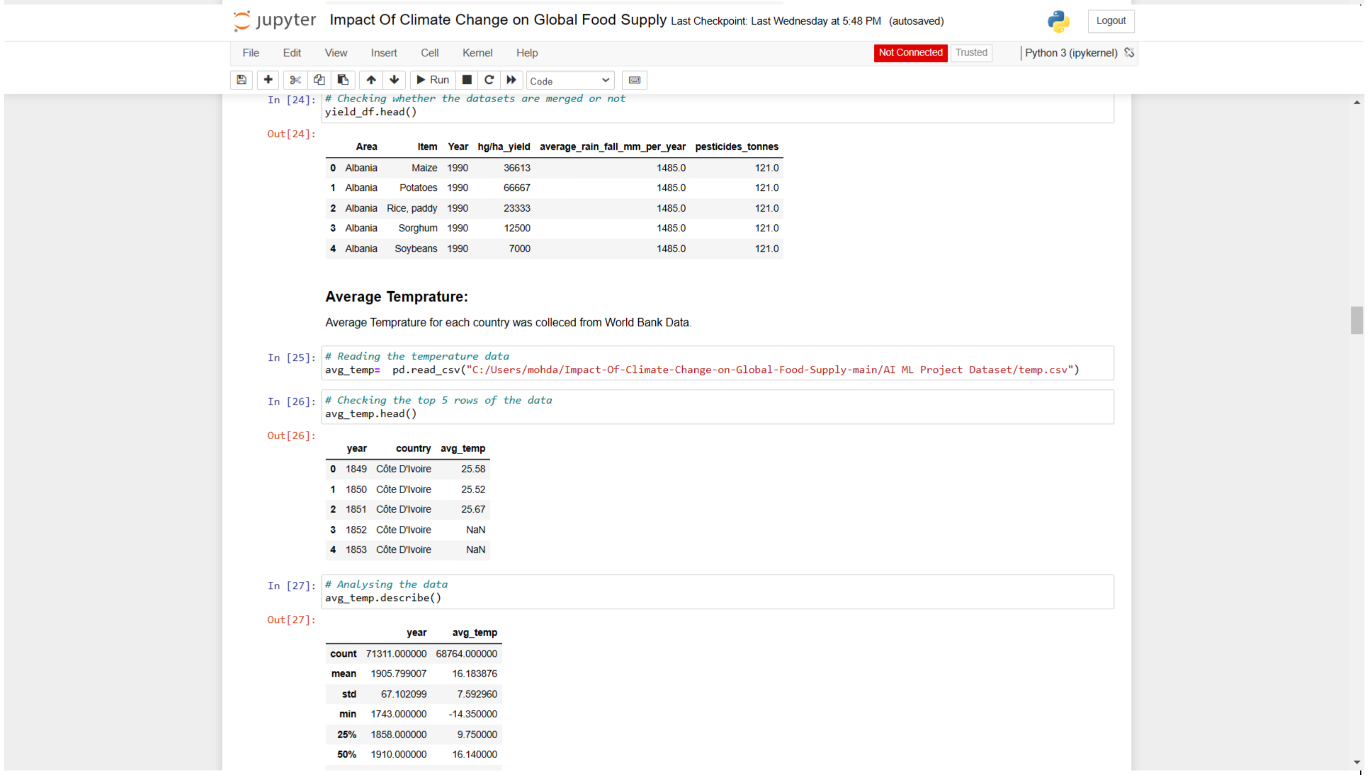
In [23]:
```
# merge Pesticides dataframe with yield dataframe
yield_df = pd.merge(yield_df, df_pes, on=['Year','Area'])
yield_df.shape
```

Out[23]:
```
(18949, 6)
```

In [24]:
```
# Checking whether the datasets are merged or not
yield_df.head()
```

Out[24]:

	Area	Item	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes
0	Albania	Maize	1990	36613	1485.0	121.0
1	Albania	Potatoes	1990	66667	1485.0	121.0
2	Albania	Rice, paddy	1990	23333	1485.0	121.0
3	Albania	Sorghum	1990	12500	1485.0	121.0


jupyter Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

In [24]:
```
# Checking whether the datasets are merged or not
yield_df.head()
```

Out[24]:

	Area	Item	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes
0	Albania	Maize	1990	36613	1485.0	121.0
1	Albania	Potatoes	1990	66667	1485.0	121.0
2	Albania	Rice, paddy	1990	23333	1485.0	121.0
3	Albania	Sorghum	1990	12500	1485.0	121.0
4	Albania	Soybeans	1990	7000	1485.0	121.0

Average Temprature:

Average Temprature for each country was collecred from World Bank Data.

In [25]:
```
# Reading the temperature data
avg_temp=  pd.read_csv("C:/Users/mohda/Impact-Of-Climate-Change-on-Global-Food-Supply-main/AI ML Project Dataset/temp.csv")
```

In [26]:
```
# Checking the top 5 rows of the data
avg_temp.head()
```

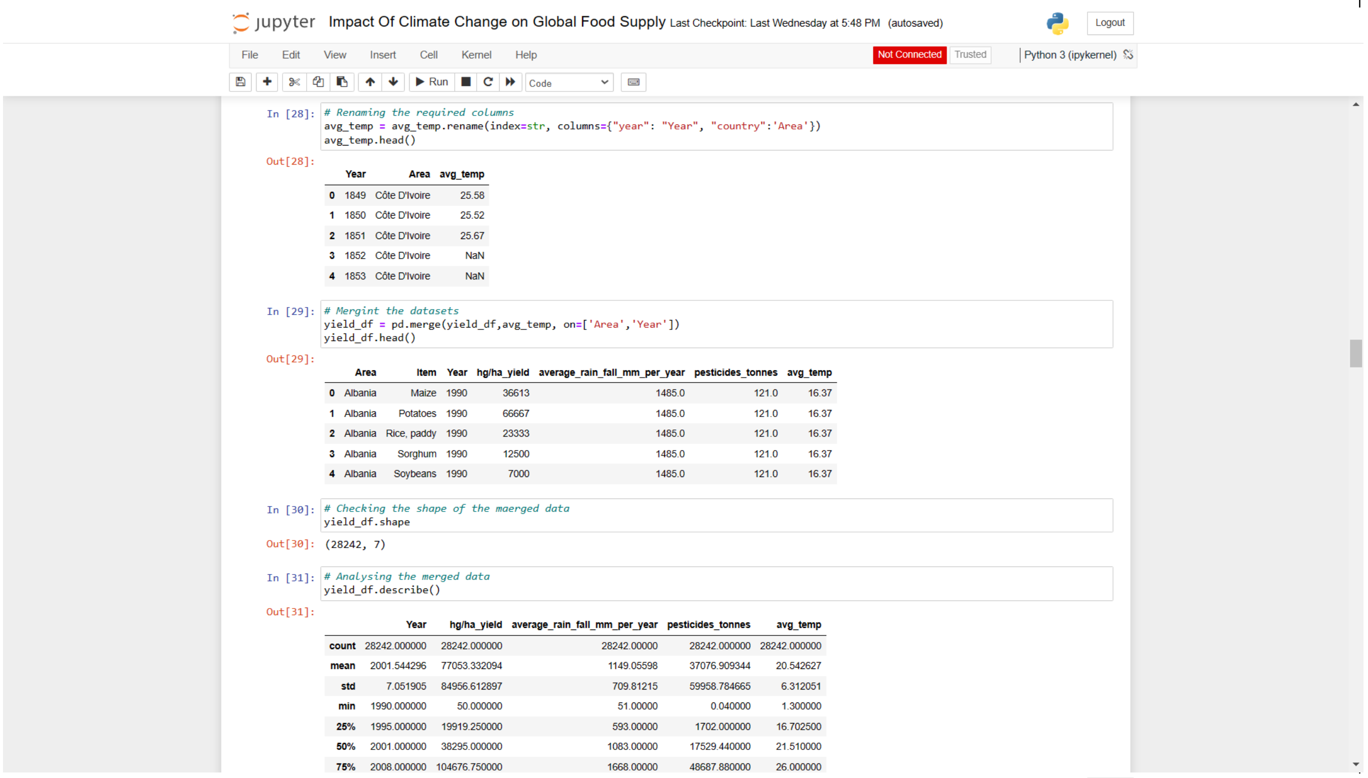
Out[26]:

	year	country	avg_temp
0	1849	Côte D'Ivoire	25.58
1	1850	Côte D'Ivoire	25.52
2	1851	Côte D'Ivoire	25.67
3	1852	Côte D'Ivoire	NaN
4	1853	Côte D'Ivoire	NaN

In [27]:
```
# Analysing the data
avg_temp.describe()
```

Out[27]:

	year	avg_temp
count	71311.000000	68764.000000
mean	1905.799007	16.183876
std	67.102099	7.592960
min	1743.000000	-14.350000
25%	1858.000000	9.750000
50%	1910.000000	16.140000



Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)

```
In [28]: # Renaming the required columns
         avg_temp = avg_temp.rename(index=str, columns={"year": "Year", "country":'Area'})
         avg_temp.head()
```

Out[28]:

	Year	Area	avg_temp
0	1849	Côte D'Ivoire	25.58
1	1850	Côte D'Ivoire	25.52
2	1851	Côte D'Ivoire	25.67
3	1852	Côte D'Ivoire	NaN
4	1853	Côte D'Ivoire	NaN

```
In [29]: # Mergint the datasets
         yield_df = pd.merge(yield_df,avg_temp, on=['Area','Year'])
         yield_df.head()
```

Out[29]:

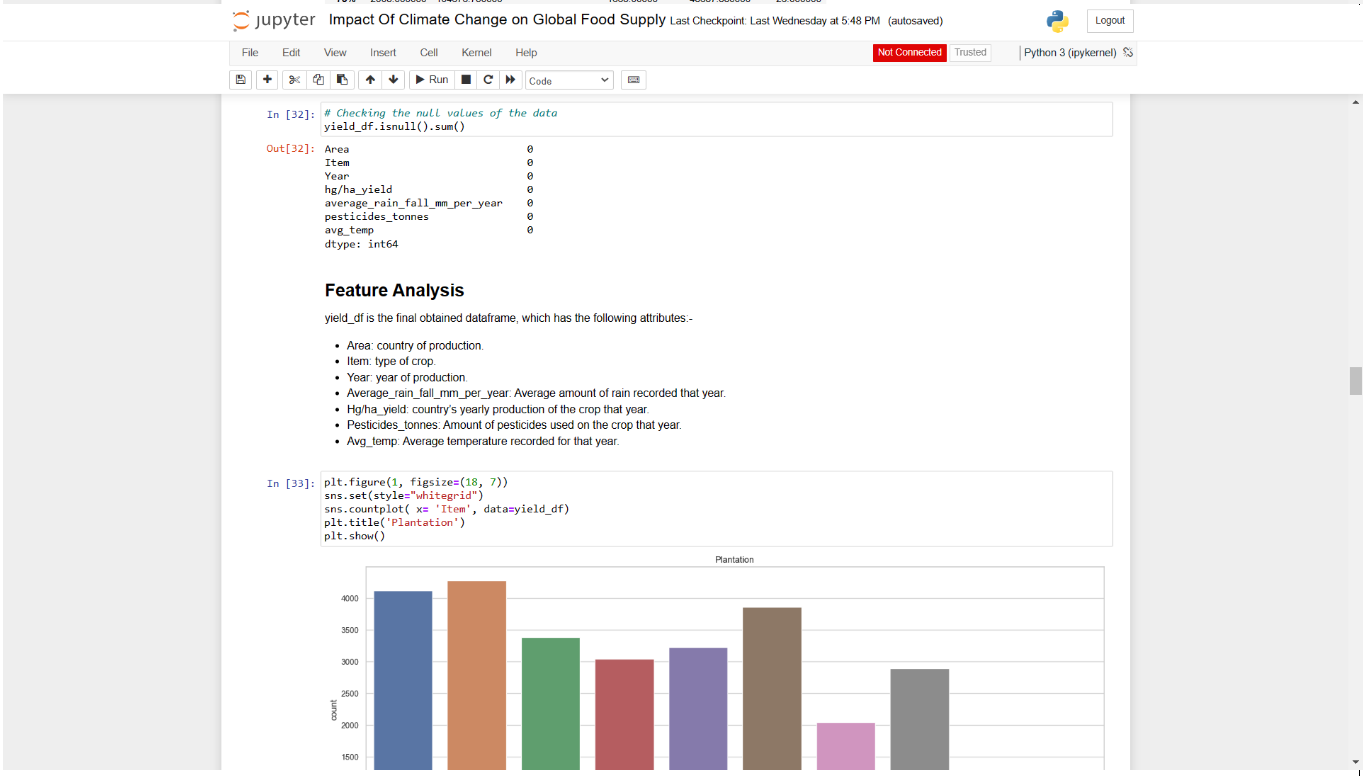
	Area	Item	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
0	Albania	Maize	1990	36613	1485.0	121.0	16.37
1	Albania	Potatoes	1990	66667	1485.0	121.0	16.37
2	Albania	Rice, paddy	1990	23333	1485.0	121.0	16.37
3	Albania	Sorghum	1990	12500	1485.0	121.0	16.37
4	Albania	Soybeans	1990	7000	1485.0	121.0	16.37

```
In [30]: # Checking the shape of the maerged data
         yield_df.shape

Out[30]: (28242, 7)
```

```
In [31]: # Analysing the merged data
         yield_df.describe()
```

Out[31]:

	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
count	28242.000000	28242.000000	28242.00000	28242.000000	28242.000000
mean	2001.544296	77053.332094	1149.05598	37076.909344	20.542627
std	7.051905	84956.612897	709.81215	59958.784665	6.312051
min	1990.000000	50.000000	51.00000	0.040000	1.300000
25%	1995.000000	19919.250000	593.00000	1702.000000	16.702500
50%	2001.000000	38295.000000	1083.00000	17529.440000	21.510000
75%	2008.000000	104676.750000	1668.00000	48687.880000	26.000000



Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)

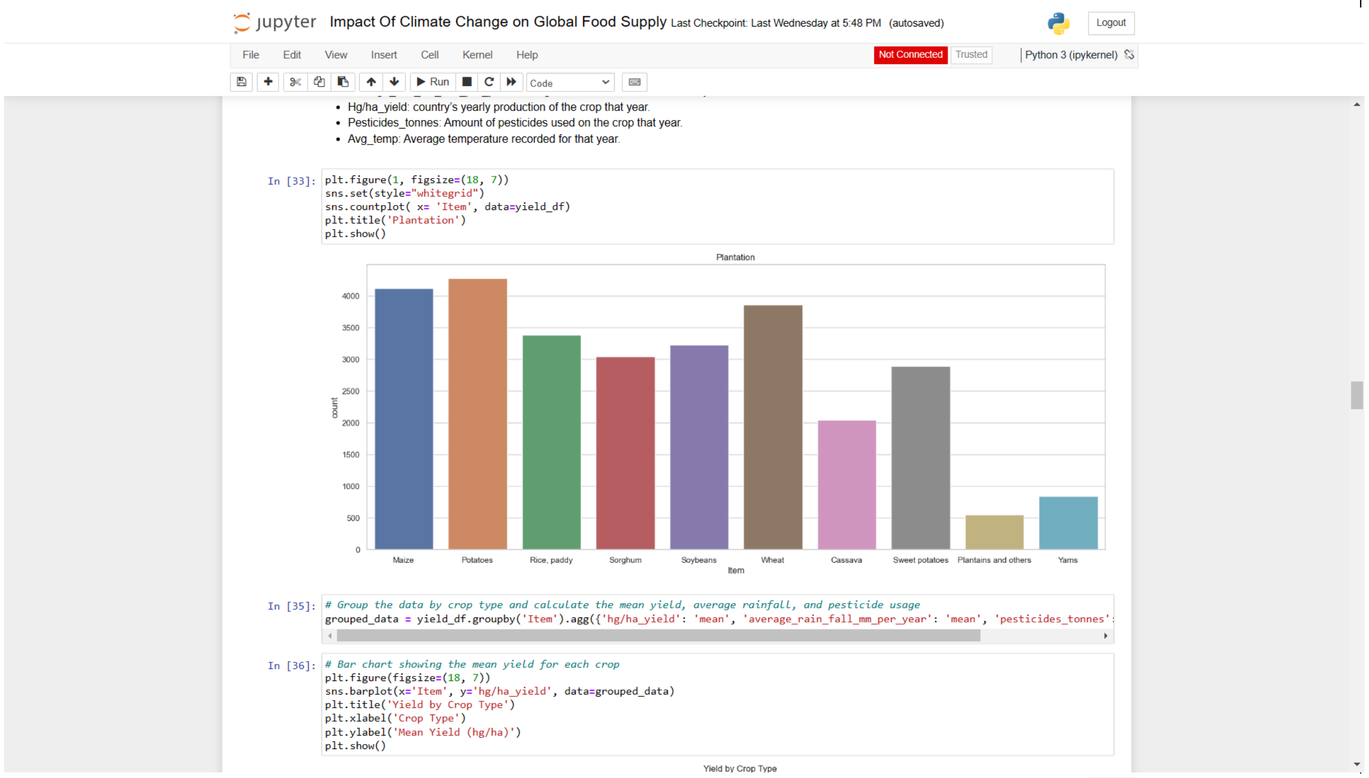
```
In [32]: # Checking the null values of the data
         yield_df.isnull().sum()

Out[32]: Area                             0
         Item                             0
         Year                             0
         hg/ha_yield                      0
         average_rain_fall_mm_per_year    0
         pesticides_tonnes                0
         avg_temp                         0
         dtype: int64
```

Feature Analysis

yield_df is the final obtained dataframe, which has the following attributes:-

- Area: country of production.
- Item: type of crop.
- Year: year of production.
- Average_rain_fall_mm_per_year: Average amount of rain recorded that year.
- Hg/ha_yield: country's yearly production of the crop that year.
- Pesticides_tonnes: Amount of pesticides used on the crop that year.
- Avg_temp: Average temperature recorded for that year.

```
In [33]: plt.figure(1, figsize=(18, 7))
         sns.set(style="whitegrid")
         sns.countplot( x= 'Item', data=yield_df)
         plt.title('Plantation')
         plt.show()
```

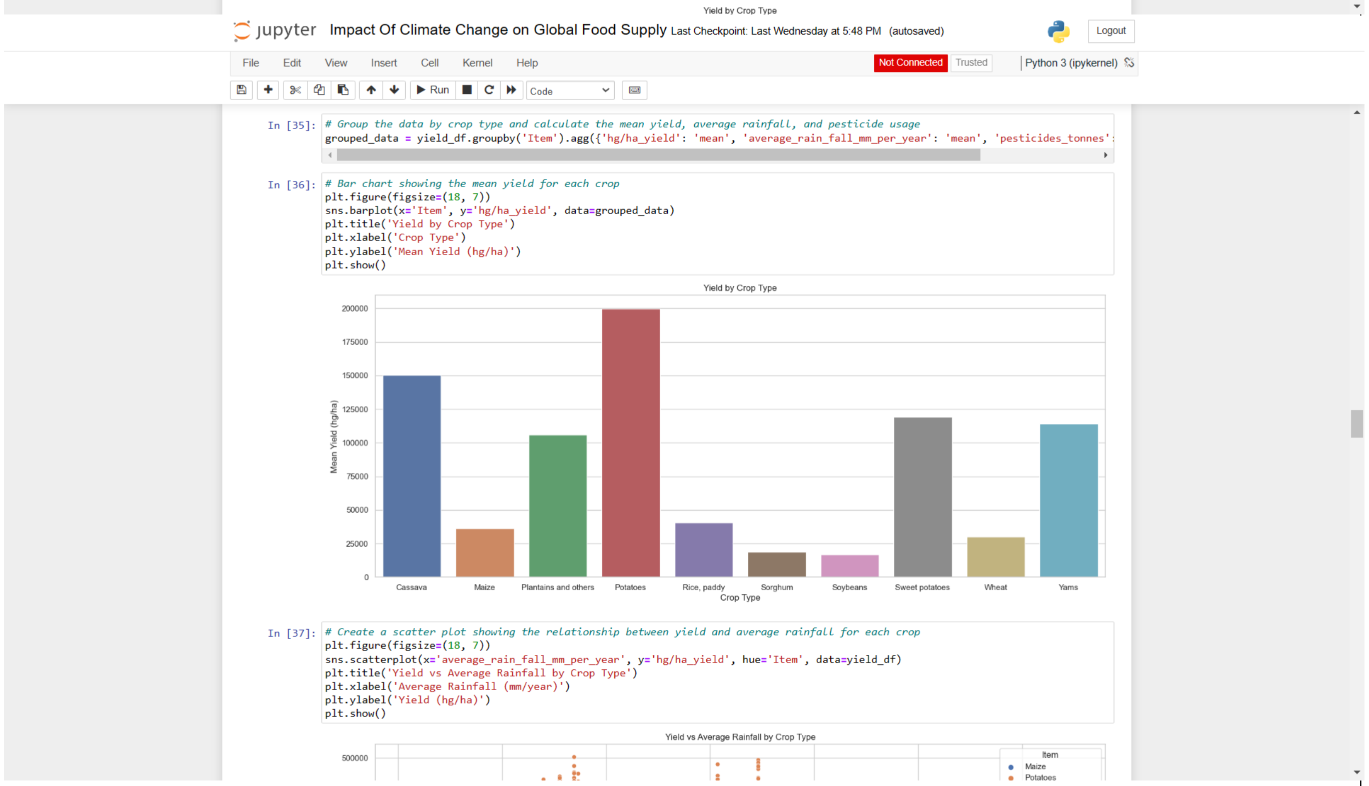

jupyter Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

- Hg/ha_yield: country's yearly production of the crop that year.
- Pesticides_tonnes: Amount of pesticides used on the crop that year.
- Avg_temp: Average temperature recorded for that year.

```
In [33]: plt.figure(1, figsize=(18, 7))
         sns.set(style="whitegrid")
         sns.countplot( x= 'Item', data=yield_df)
         plt.title('Plantation')
         plt.show()
```

```
In [35]: # Group the data by crop type and calculate the mean yield, average rainfall, and pesticide usage
         grouped_data = yield_df.groupby('Item').agg({'hg/ha_yield': 'mean', 'average_rain_fall_mm_per_year': 'mean', 'pesticides_tonnes':
```

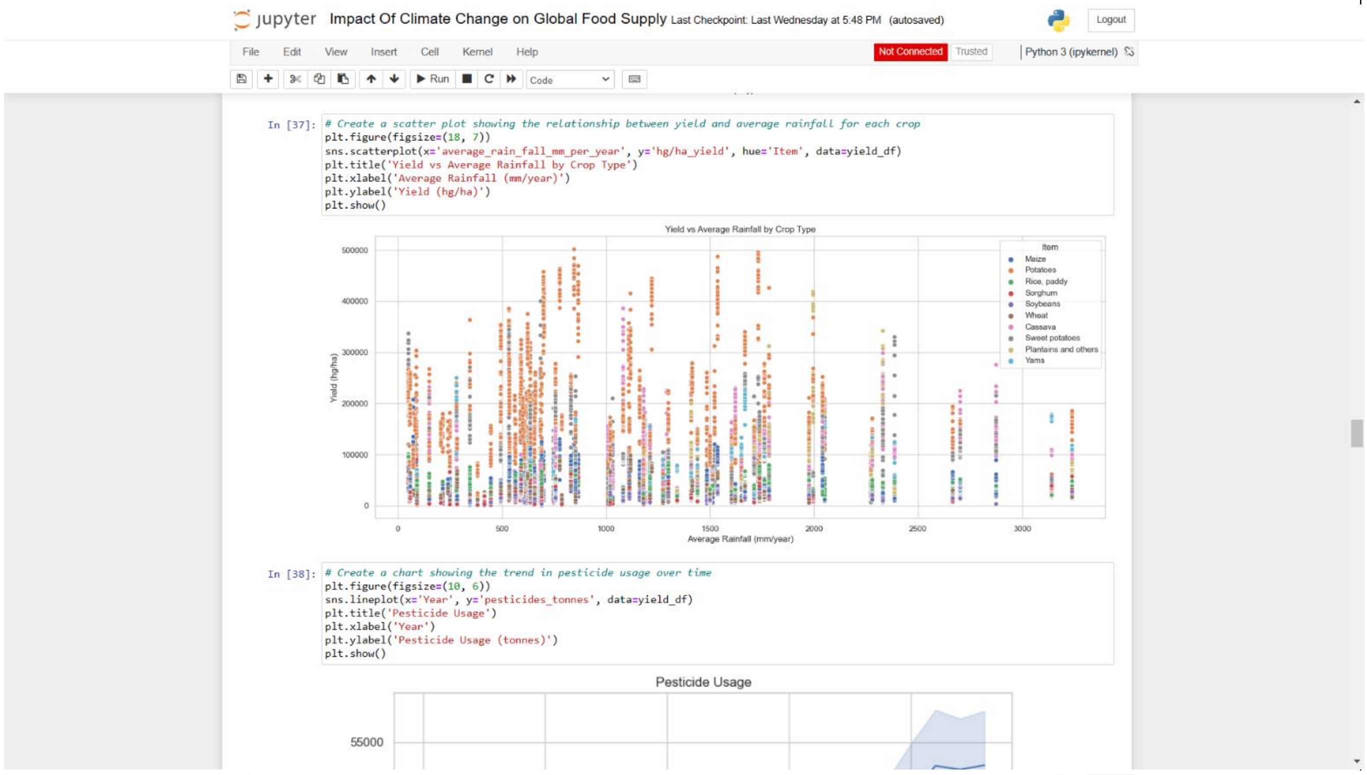
```
In [36]: # Bar chart showing the mean yield for each crop
         plt.figure(figsize=(18, 7))
         sns.barplot(x='Item', y='hg/ha_yield', data=grouped_data)
         plt.title('Yield by Crop Type')
         plt.xlabel('Crop Type')
         plt.ylabel('Mean Yield (hg/ha)')
         plt.show()
```


jupyter Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

```
In [35]: # Group the data by crop type and calculate the mean yield, average rainfall, and pesticide usage
         grouped_data = yield_df.groupby('Item').agg({'hg/ha_yield': 'mean', 'average_rain_fall_mm_per_year': 'mean', 'pesticides_tonnes':
```

```
In [36]: # Bar chart showing the mean yield for each crop
         plt.figure(figsize=(18, 7))
         sns.barplot(x='Item', y='hg/ha_yield', data=grouped_data)
         plt.title('Yield by Crop Type')
         plt.xlabel('Crop Type')
         plt.ylabel('Mean Yield (hg/ha)')
         plt.show()
```

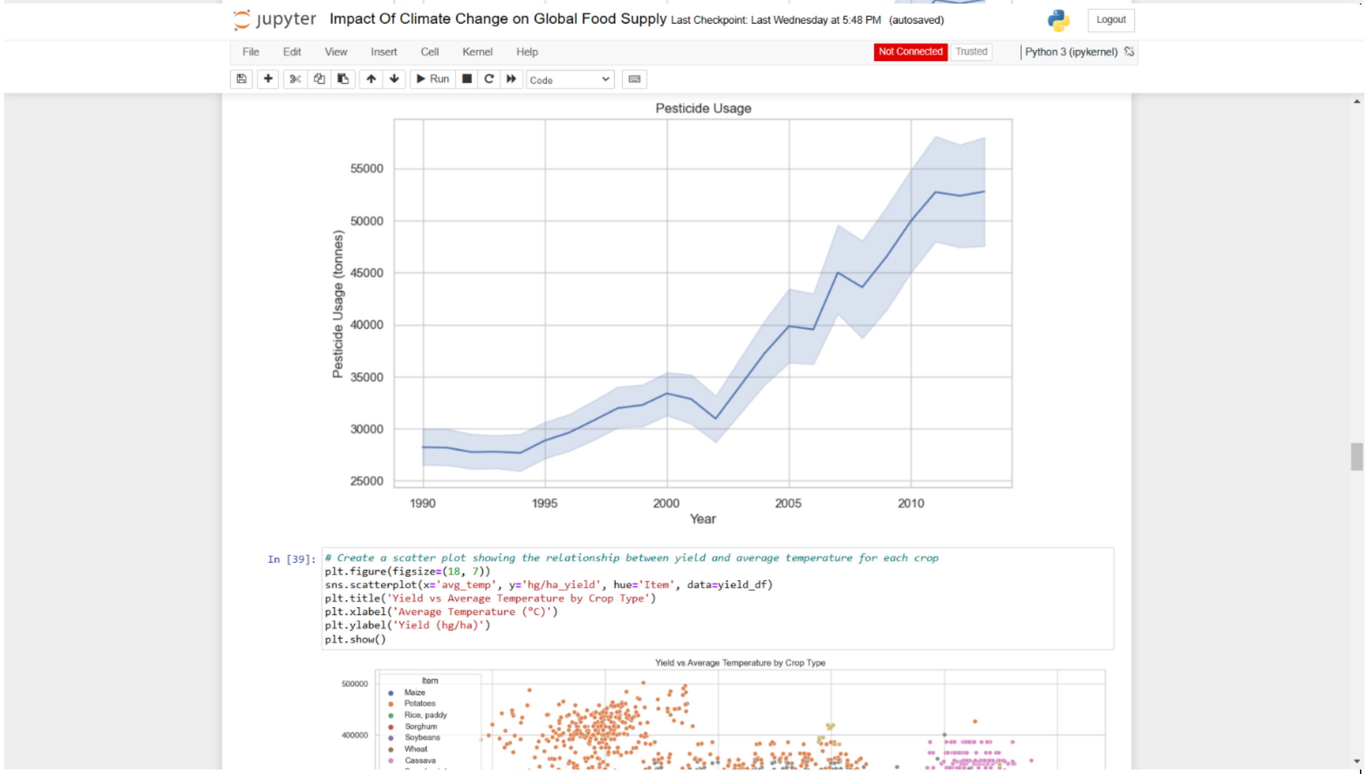
```
In [37]: # Create a scatter plot showing the relationship between yield and average rainfall for each crop
         plt.figure(figsize=(18, 7))
         sns.scatterplot(x='average_rain_fall_mm_per_year', y='hg/ha_yield', hue='Item', data=yield_df)
         plt.title('Yield vs Average Rainfall by Crop Type')
         plt.xlabel('Average Rainfall (mm/year)')
         plt.ylabel('Yield (hg/ha)')
         plt.show()
```


Jupyter Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

In [37]:
```python
# Create a scatter plot showing the relationship between yield and average rainfall for each crop
plt.figure(figsize=(18, 7))
sns.scatterplot(x='average_rain_fall_mm_per_year', y='hg/ha_yield', hue='Item', data=yield_df)
plt.title('Yield vs Average Rainfall by Crop Type')
plt.xlabel('Average Rainfall (mm/year)')
plt.ylabel('Yield (hg/ha)')
plt.show()
```

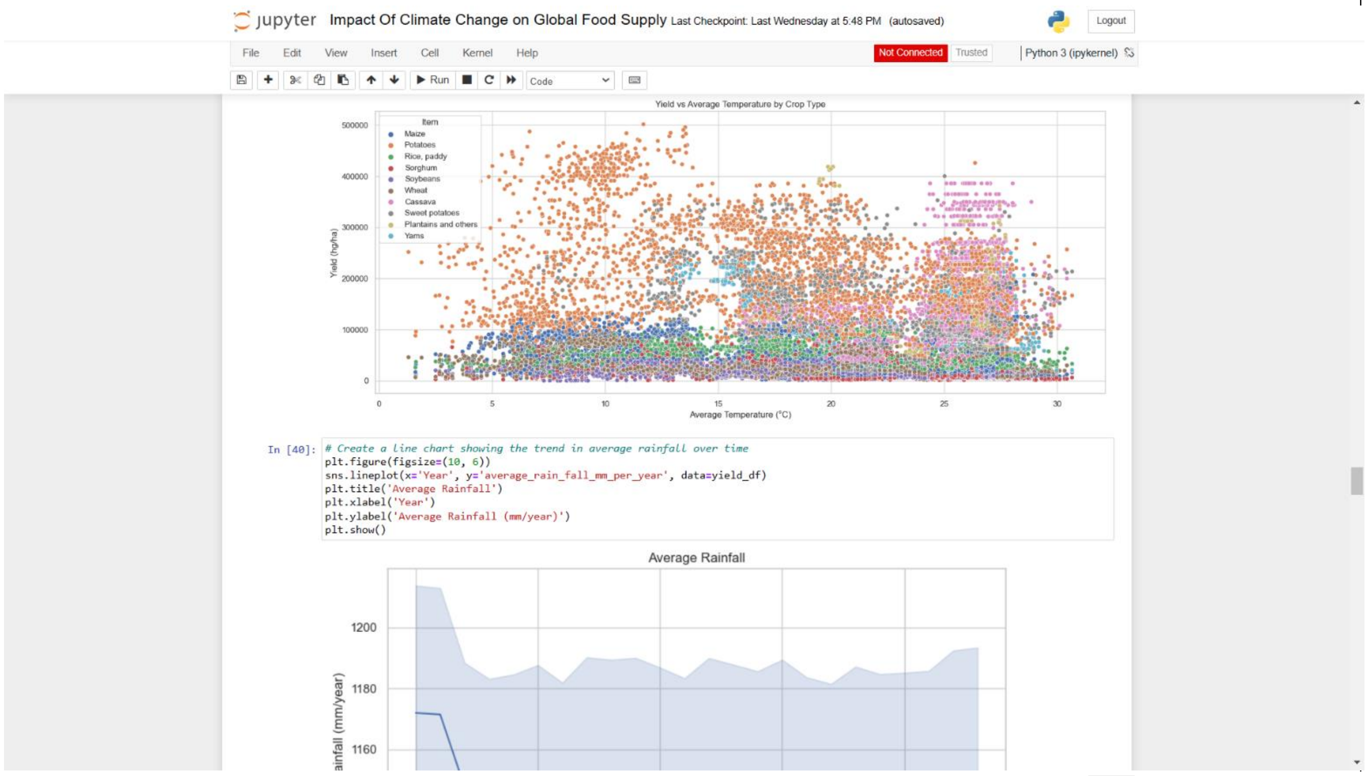
Yield vs Average Rainfall by Crop Type
Item
Maize
Potatoes
Rice, paddy
Sorghum
Soybeans
Wheat
Cassava
Sweet potatoes
Plantains and others
Yams
Yield (hg/ha)
Average Rainfall (mm/year)

In [38]:
```python
# Create a chart showing the trend in pesticide usage over time
plt.figure(figsize=(10, 6))
sns.lineplot(x='Year', y='pesticides_tonnes', data=yield_df)
plt.title('Pesticide Usage')
plt.xlabel('Year')
plt.ylabel('Pesticide Usage (tonnes)')
plt.show()
```

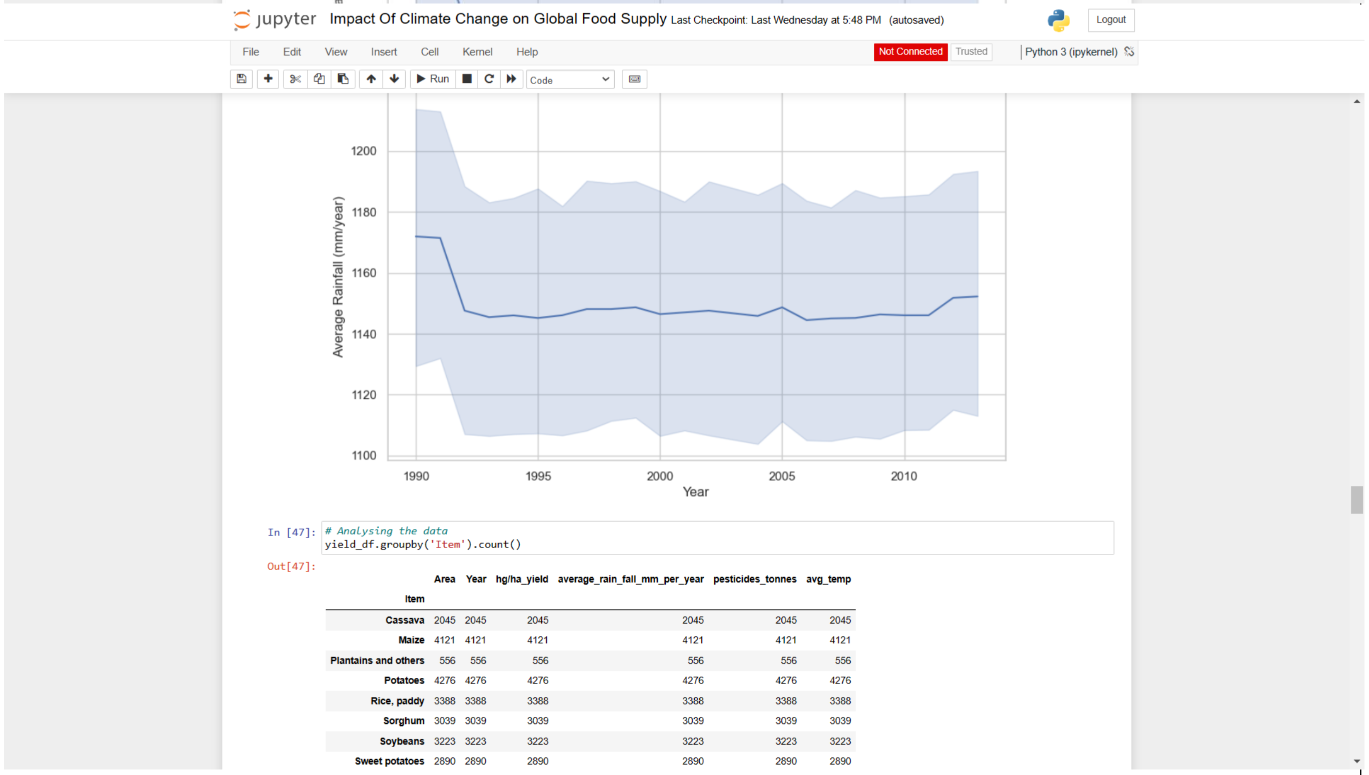

Jupyter Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code
Pesticide Usage
Pesticide Usage (tonnes)
Year

In [39]:
```python
# Create a scatter plot showing the relationship between yield and average temperature for each crop
plt.figure(figsize=(18, 7))
sns.scatterplot(x='avg_temp', y='hg/ha_yield', hue='Item', data=yield_df)
plt.title('Yield vs Average Temperature by Crop Type')
plt.xlabel('Average Temperature (°C)')
plt.ylabel('Yield (hg/ha)')
plt.show()
```

Yield vs Average Temperature by Crop Type
Item
Maize
Potatoes
Rice, paddy
Sorghum
Soybeans
Wheat
Cassava

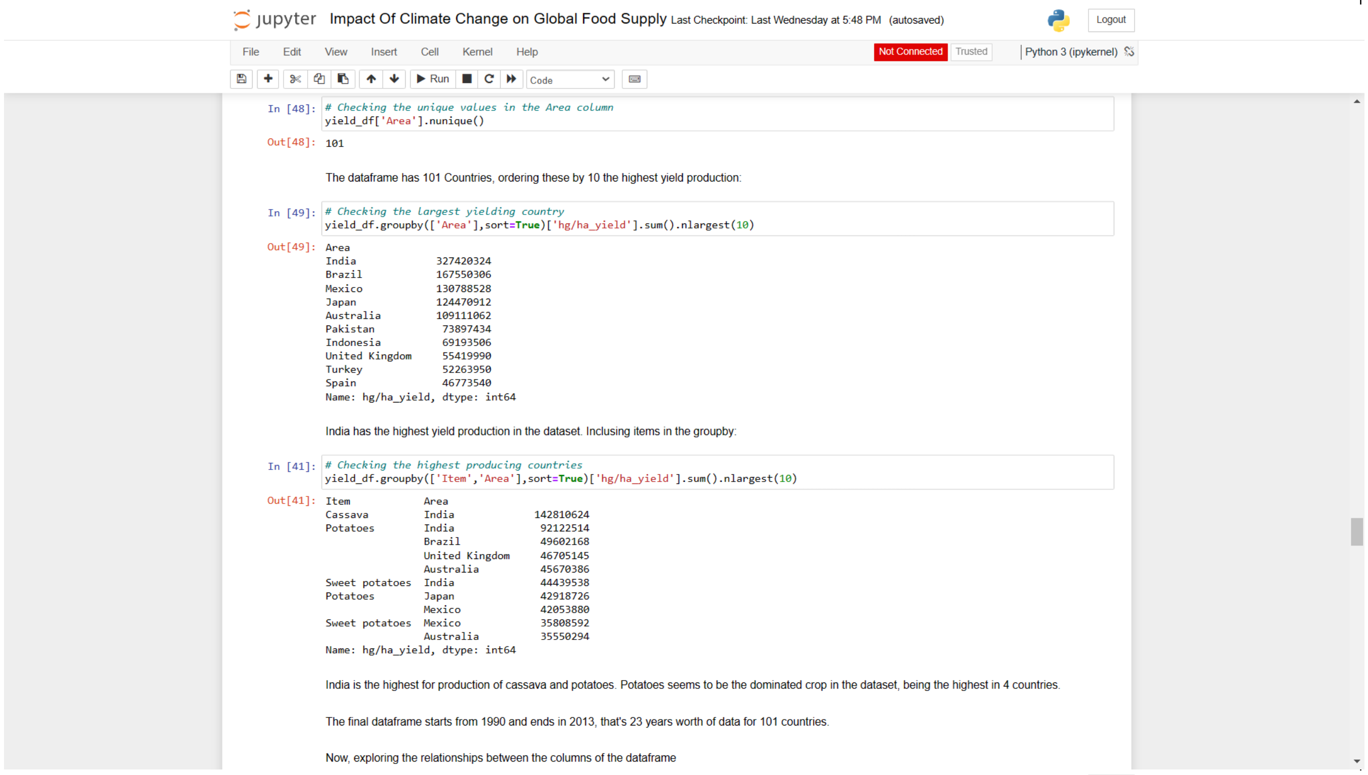

jupyter
Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code
Yield vs Average Temperature by Crop Type
Item
Maize
Potatoes
Rice, paddy
Sorghum
Soybeans
Wheat
Cassava
Sweet potatoes
Plantains and others
Yams
Yield (hg/ha)
Average Temperature (°C)
Average Rainfall
Average Rainfall (mm/year)

```
In [40]: # Create a line chart showing the trend in average rainfall over time
         plt.figure(figsize=(10, 6))
         sns.lineplot(x='Year', y='average_rain_fall_mm_per_year', data=yield_df)
         plt.title('Average Rainfall')
         plt.xlabel('Year')
         plt.ylabel('Average Rainfall (mm/year)')
         plt.show()
```


jupyter
Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code
Average Rainfall (mm/year)
Year

```
In [47]: # Analysing the data
         yield_df.groupby('Item').count()
Out[47]:
```

	Area	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
Item						
Cassava	2045	2045	2045	2045	2045	2045
Maize	4121	4121	4121	4121	4121	4121
Plantains and others	556	556	556	556	556	556
Potatoes	4276	4276	4276	4276	4276	4276
Rice, paddy	3388	3388	3388	3388	3388	3388
Sorghum	3039	3039	3039	3039	3039	3039
Soybeans	3223	3223	3223	3223	3223	3223
Sweet potatoes	2890	2890	2890	2890	2890	2890


jupyter
Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

```
In [48]: # Checking the unique values in the Area column
         yield_df['Area'].nunique()

Out[48]: 101
```

The dataframe has 101 Countries, ordering these by 10 the highest yield production:

```
In [49]: # Checking the largest yielding country
         yield_df.groupby(['Area'],sort=True)['hg/ha_yield'].sum().nlargest(10)

Out[49]: Area
         India             327420324
         Brazil            167550306
         Mexico            130788528
         Japan             124470912
         Australia         109111062
         Pakistan           73897434
         Indonesia          69193506
         United Kingdom     55419990
         Turkey             52263950
         Spain              46773540
         Name: hg/ha_yield, dtype: int64
```

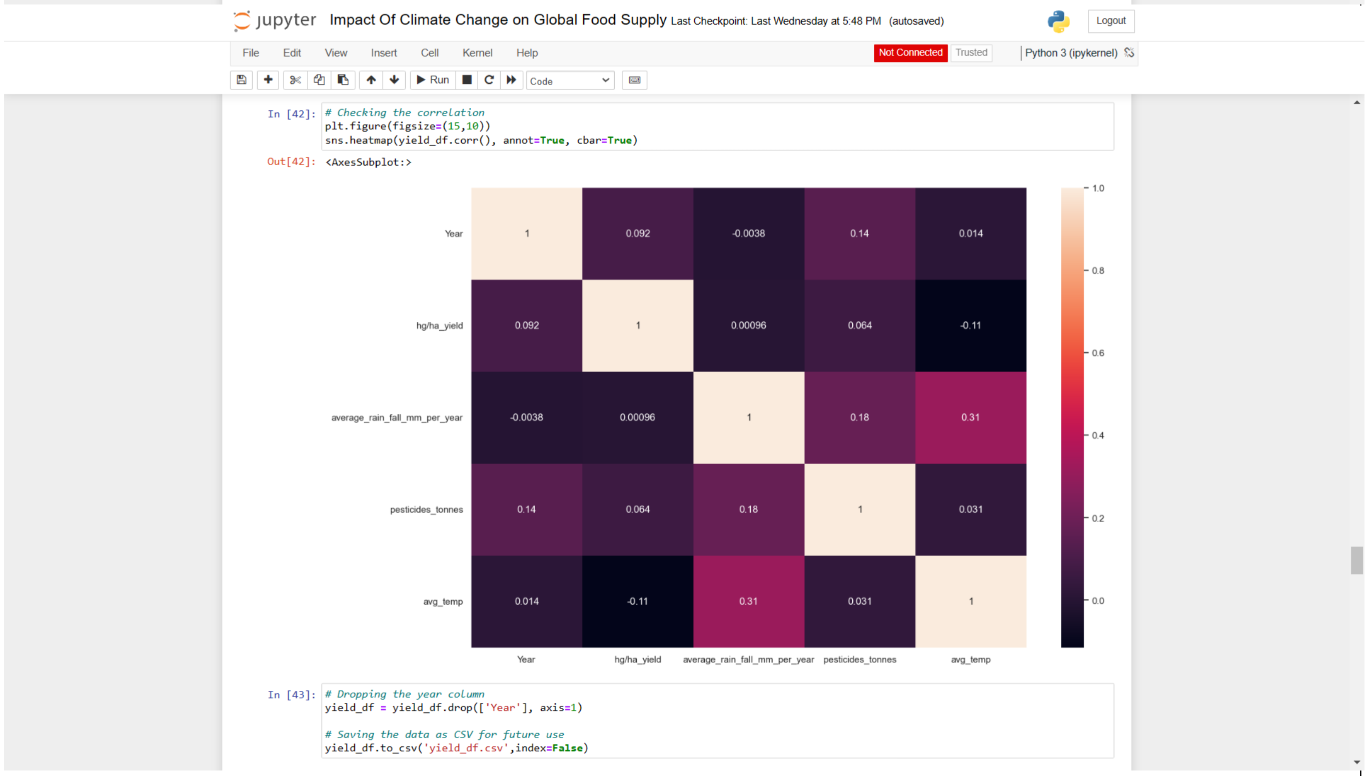
India has the highest yield production in the dataset. Including items in the groupby:

```
In [41]: # Checking the highest producing countries
         yield_df.groupby(['Item','Area'],sort=True)['hg/ha_yield'].sum().nlargest(10)

Out[41]: Item            Area
         Cassava         India             142810624
         Potatoes        India              92122514
                         Brazil             49602168
                         United Kingdom     46705145
                         Australia          45670386
         Sweet potatoes  India              44439538
         Potatoes        Japan              42918726
                         Mexico             42053880
         Sweet potatoes  Mexico             35808592
                         Australia          35550294
         Name: hg/ha_yield, dtype: int64
```

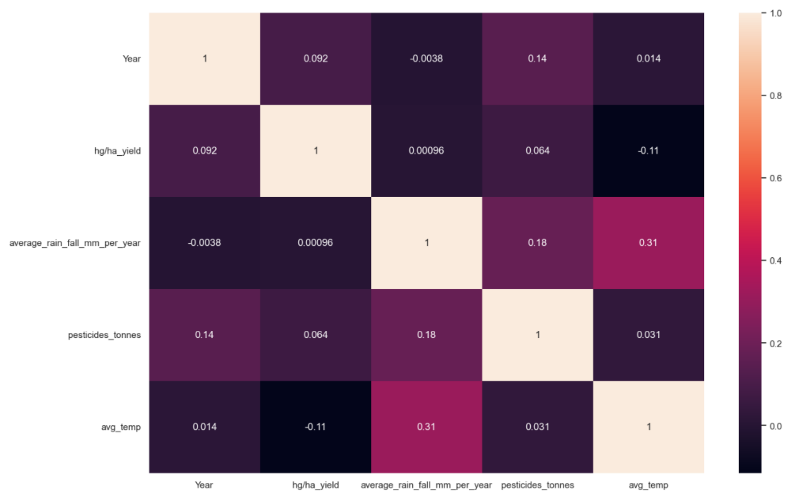
India is the highest for production of cassava and potatoes. Potatoes seems to be the dominated crop in the dataset, being the highest in 4 countries.

The final dataframe starts from 1990 and ends in 2013, that's 23 years worth of data for 101 countries.

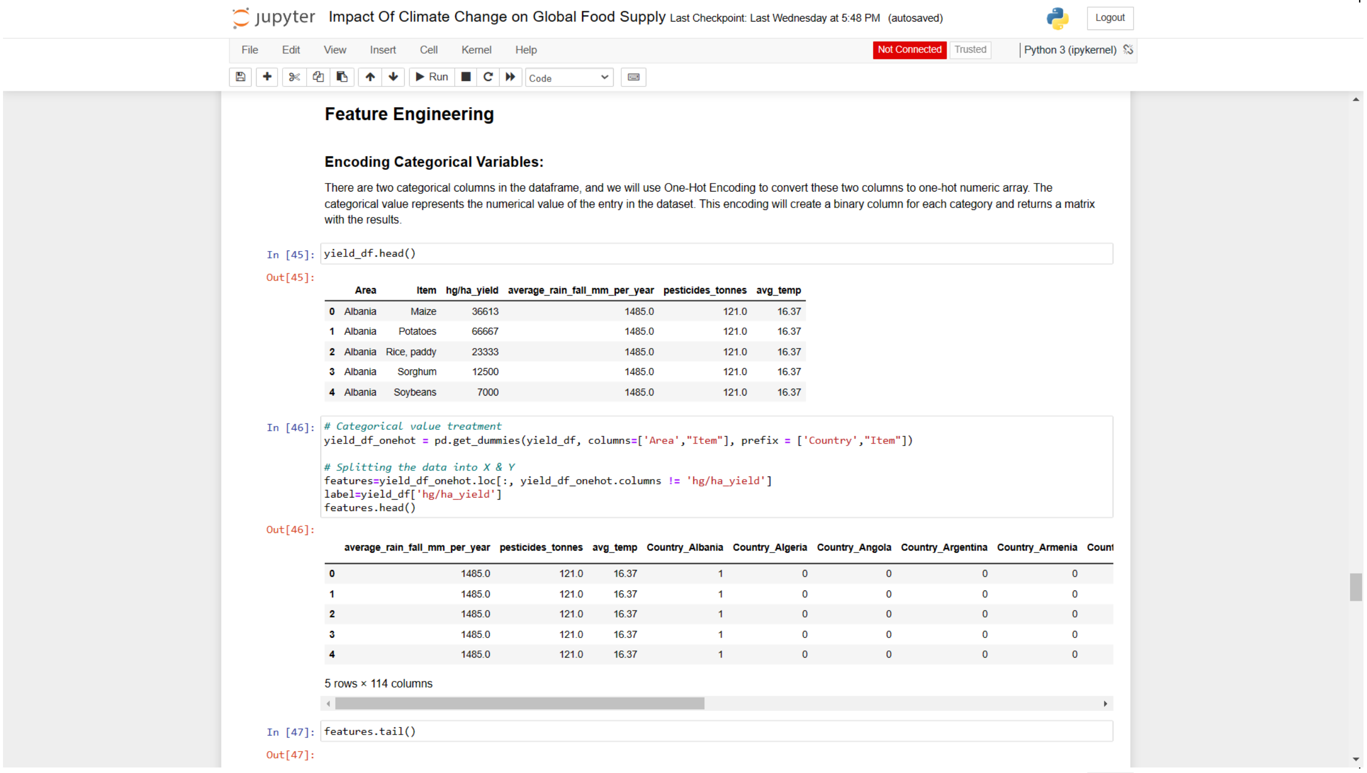
Now, exploring the relationships between the columns of the dataframe


jupyter
Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

```
In [42]: # Checking the correlation
         plt.figure(figsize=(15,10))
         sns.heatmap(yield_df.corr(), annot=True, cbar=True)

Out[42]: <AxesSubplot:>
```



```
In [43]: # Dropping the year column
         yield_df = yield_df.drop(['Year'], axis=1)

         # Saving the data as CSV for future use
         yield_df.to_csv('yield_df.csv',index=False)
```



jupyter
Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

Feature Engineering

Encoding Categorical Variables:

There are two categorical columns in the dataframe, and we will use One-Hot Encoding to convert these two columns to one-hot numeric array. The categorical value represents the numerical value of the entry in the dataset. This encoding will create a binary column for each category and returns a matrix with the results.

In [45]:
```
yield_df.head()
```

Out[45]:

	Area	Item	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
0	Albania	Maize	36613	1485.0	121.0	16.37
1	Albania	Potatoes	66667	1485.0	121.0	16.37
2	Albania	Rice, paddy	23333	1485.0	121.0	16.37
3	Albania	Sorghum	12500	1485.0	121.0	16.37
4	Albania	Soybeans	7000	1485.0	121.0	16.37

In [46]:
```
# Categorical value treatment
yield_df_onehot = pd.get_dummies(yield_df, columns=['Area',"Item"], prefix = ['Country',"Item"])

# Splitting the data into X & Y
features=yield_df_onehot.loc[:, yield_df_onehot.columns != 'hg/ha_yield']
label=yield_df['hg/ha_yield']
features.head()
```

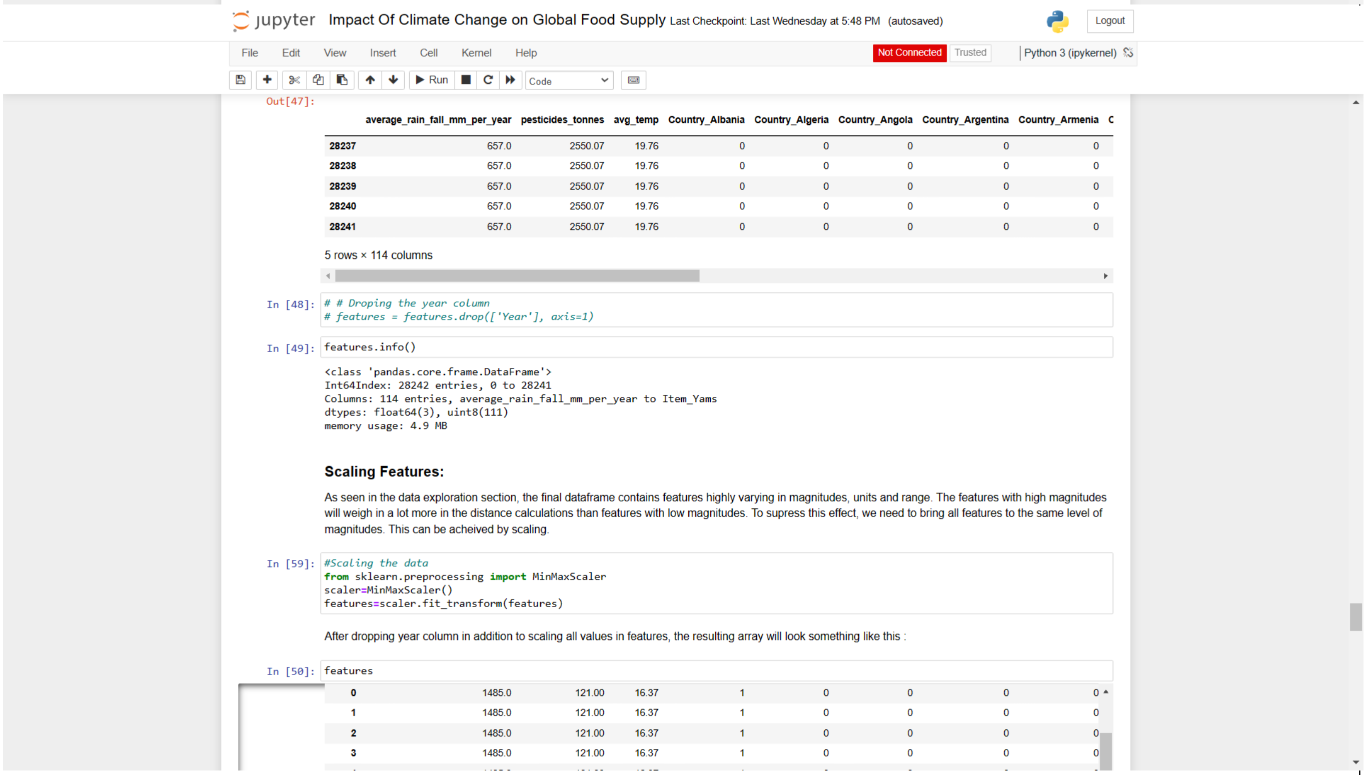
Out[46]:

	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country_Albania	Country_Algeria	Country_Angola	Country_Argentina	Country_Armenia	Count
0	1485.0	121.0	16.37	1	0	0	0	0	
1	1485.0	121.0	16.37	1	0	0	0	0	
2	1485.0	121.0	16.37	1	0	0	0	0	
3	1485.0	121.0	16.37	1	0	0	0	0	
4	1485.0	121.0	16.37	1	0	0	0	0	

5 rows × 114 columns

In [47]:
```
features.tail()
```

Out[47]:



jupyter
Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

Out[47]:

	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country_Albania	Country_Algeria	Country_Angola	Country_Argentina	Country_Armenia	C
28237	657.0	2550.07	19.76	0	0	0	0	0	
28238	657.0	2550.07	19.76	0	0	0	0	0	
28239	657.0	2550.07	19.76	0	0	0	0	0	
28240	657.0	2550.07	19.76	0	0	0	0	0	
28241	657.0	2550.07	19.76	0	0	0	0	0	

5 rows × 114 columns

In [48]:
```
# # Droping the year column
# features = features.drop(['Year'], axis=1)
```

In [49]:
```
features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 28242 entries, 0 to 28241
Columns: 114 entries, average_rain_fall_mm_per_year to Item_Yams
dtypes: float64(3), uint8(111)
memory usage: 4.9 MB
```

Scaling Features:

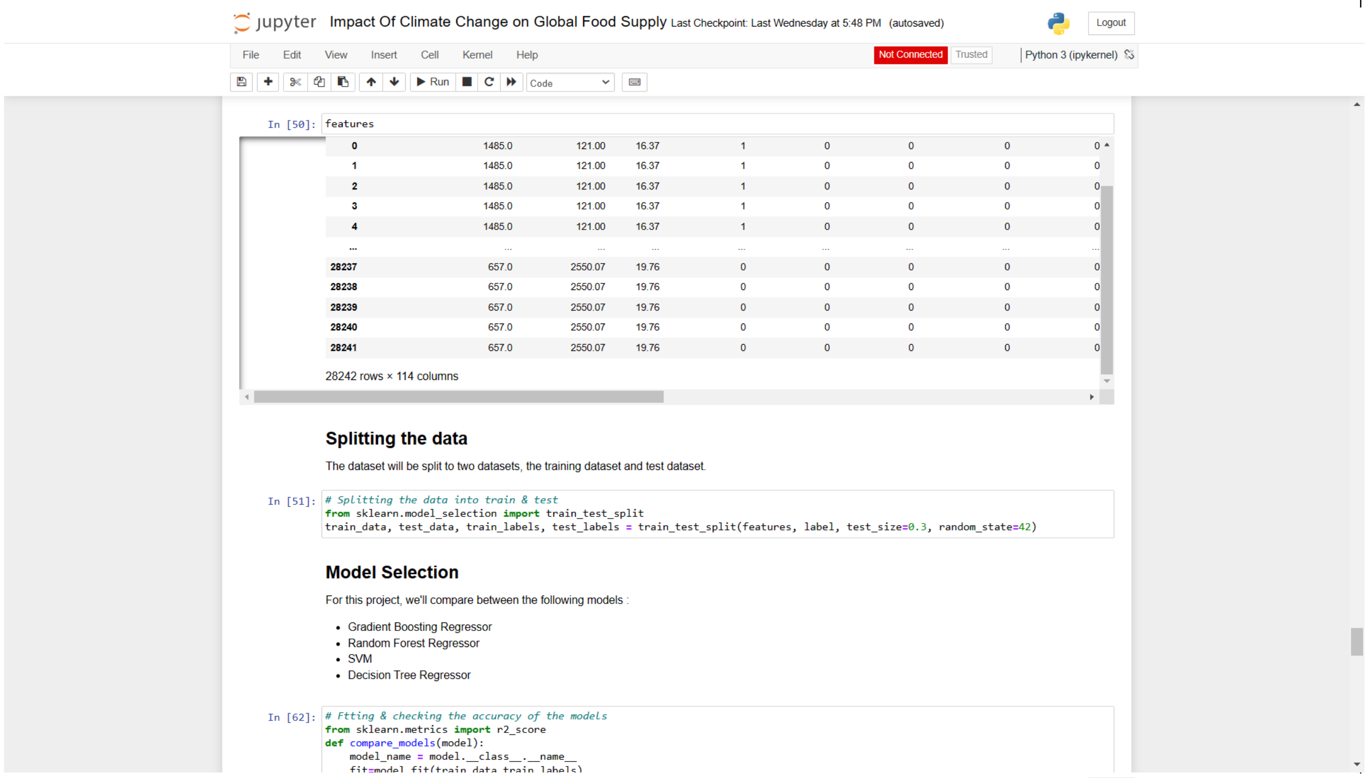
As seen in the data exploration section, the final dataframe contains features highly varying in magnitudes, units and range. The features with high magnitudes will weigh in a lot more in the distance calculations than features with low magnitudes. To supress this effect, we need to bring all features to the same level of magnitudes. This can be acheived by scaling.

In [59]:
```
#Scaling the data
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
features=scaler.fit_transform(features)
```

After dropping year column in addition to scaling all values in features, the resulting array will look something like this :

In [50]:
```
features
```

0	1485.0	121.00	16.37	1	0	0	0	0
1	1485.0	121.00	16.37	1	0	0	0	0
2	1485.0	121.00	16.37	1	0	0	0	0
3	1485.0	121.00	16.37	1	0	0	0	0


jupyter Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)

```
In [50]: features
```

0	1485.0	121.00	16.37	1	0	0	0	0	0
1	1485.0	121.00	16.37	1	0	0	0	0	0
2	1485.0	121.00	16.37	1	0	0	0	0	0
3	1485.0	121.00	16.37	1	0	0	0	0	0
4	1485.0	121.00	16.37	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
28237	657.0	2550.07	19.76	0	0	0	0	0	0
28238	657.0	2550.07	19.76	0	0	0	0	0	0
28239	657.0	2550.07	19.76	0	0	0	0	0	0
28240	657.0	2550.07	19.76	0	0	0	0	0	0
28241	657.0	2550.07	19.76	0	0	0	0	0	0

28242 rows × 114 columns

Splitting the data

The dataset will be split to two datasets, the training dataset and test dataset.

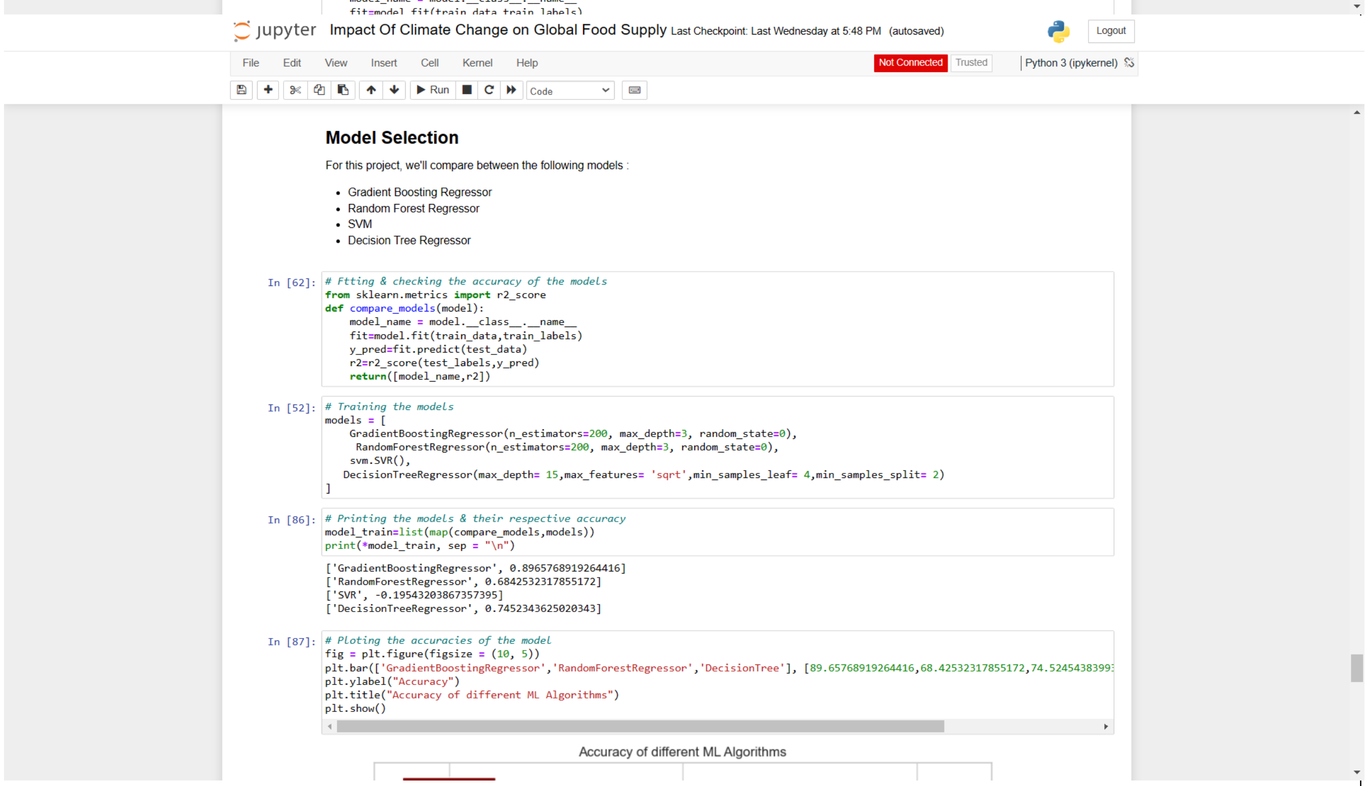
```
In [51]: # Splitting the data into train & test
         from sklearn.model_selection import train_test_split
         train_data, test_data, train_labels, test_labels = train_test_split(features, label, test_size=0.3, random_state=42)
```

Model Selection

For this project, we'll compare between the following models :

- Gradient Boosting Regressor
- Random Forest Regressor
- SVM
- Decision Tree Regressor

```
In [62]: # Ftting & checking the accuracy of the models
         from sklearn.metrics import r2_score
         def compare_models(model):
             model_name = model.__class__.__name__
             fit=model.fit(train_data,train_labels)
```


jupyter Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)

Model Selection

For this project, we'll compare between the following models :

- Gradient Boosting Regressor
- Random Forest Regressor
- SVM
- Decision Tree Regressor

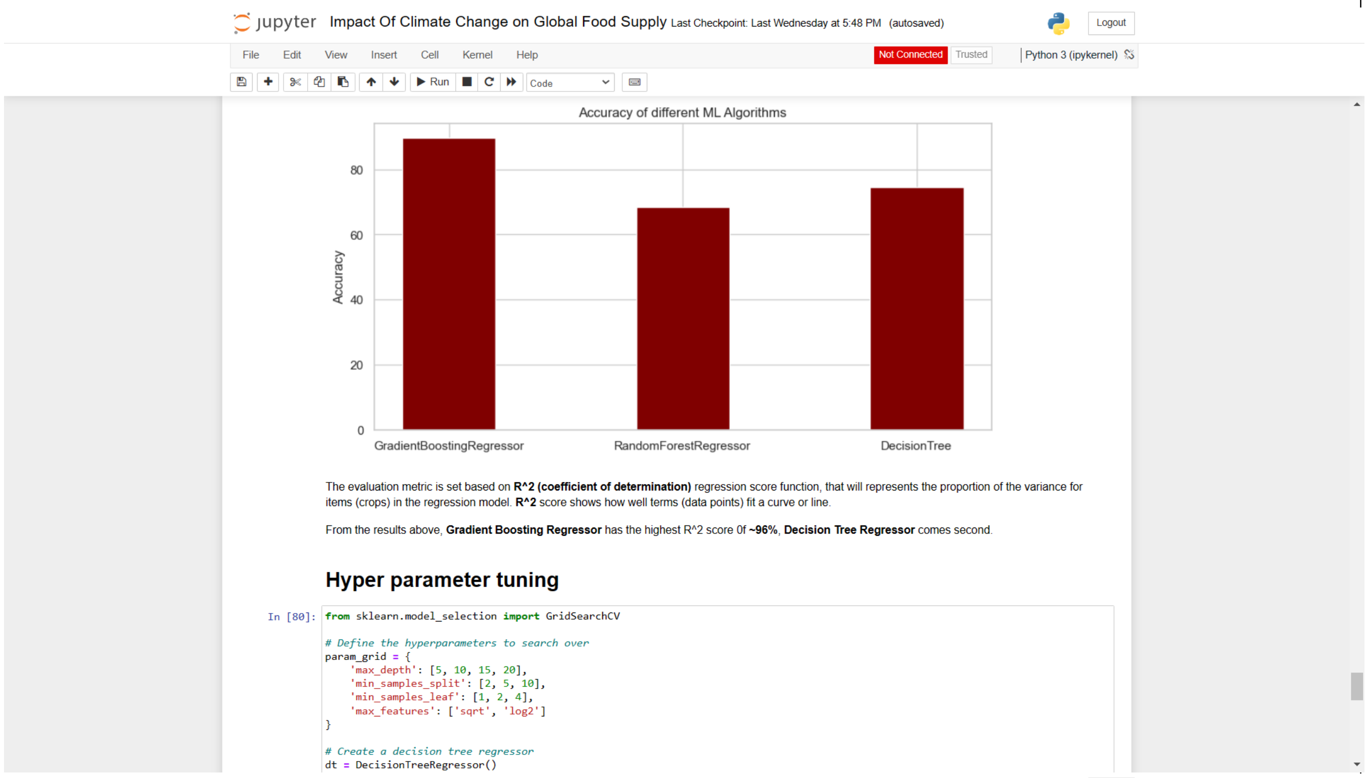
```
In [62]: # Ftting & checking the accuracy of the models
         from sklearn.metrics import r2_score
         def compare_models(model):
             model_name = model.__class__.__name__
             fit=model.fit(train_data,train_labels)
             y_pred=fit.predict(test_data)
             r2=r2_score(test_labels,y_pred)
             return([model_name,r2])
```

```
In [52]: # Training the models
         models = [
             GradientBoostingRegressor(n_estimators=200, max_depth=3, random_state=0),
              RandomForestRegressor(n_estimators=200, max_depth=3, random_state=0),
             svm.SVR(),
            DecisionTreeRegressor(max_depth= 15,max_features= 'sqrt',min_samples_leaf= 4,min_samples_split= 2)
         ]
```

```
In [86]: # Printing the models & their respective accuracy
         model_train=list(map(compare_models,models))
         print(*model_train, sep = "\n")

         ['GradientBoostingRegressor', 0.8965768919264416]
         ['RandomForestRegressor', 0.6842532317855172]
         ['SVR', -0.19543203867357395]
         ['DecisionTreeRegressor', 0.7452343625020343]
```

```
In [87]: # Ploting the accuracies of the model
         fig = plt.figure(figsize = (10, 5))
         plt.bar(['GradientBoostingRegressor','RandomForestRegressor','DecisionTree'], [89.65768919264416,68.42532317855172,74.5245438399
         plt.ylabel("Accuracy")
         plt.title("Accuracy of different ML Algorithms")
         plt.show()
```

Accuracy of different ML Algorithms

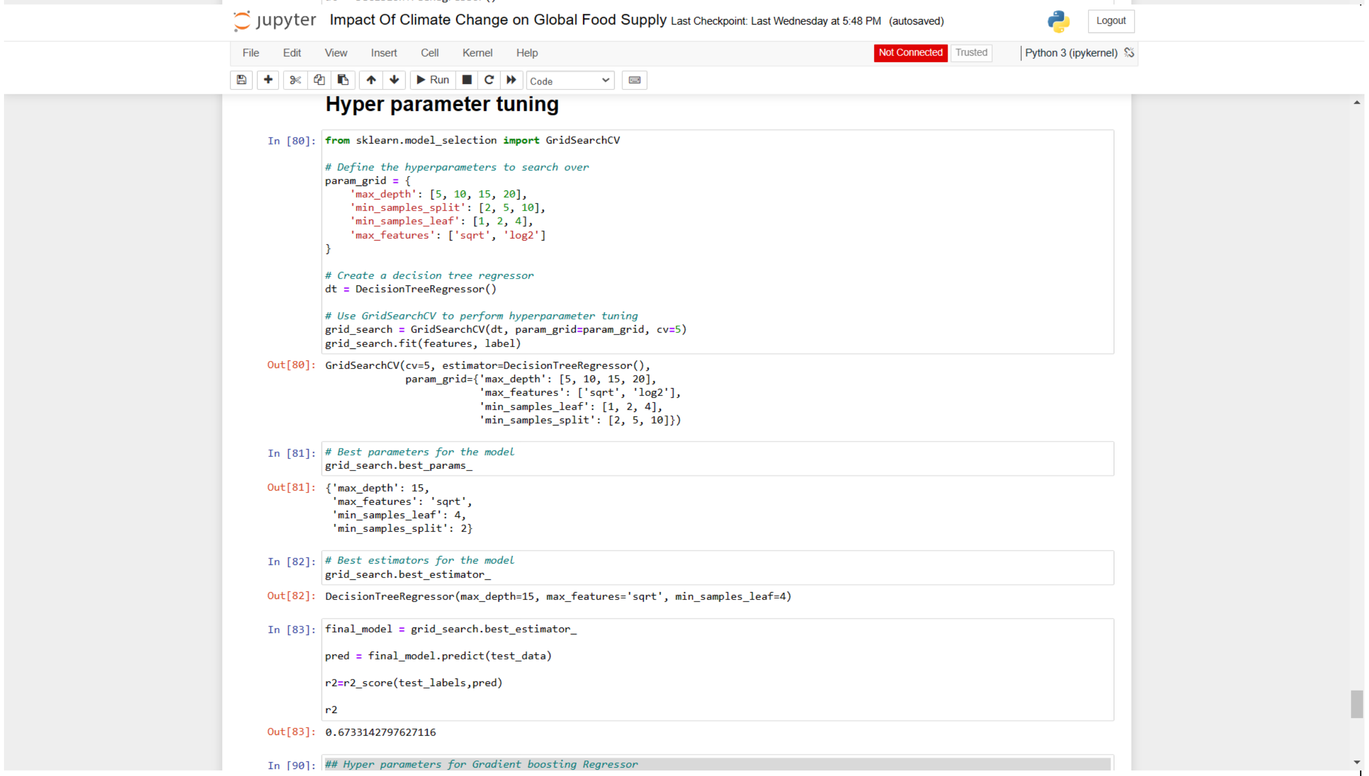

jupyter
Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code
Accuracy of different ML Algorithms
Accuracy
0
20
40
60
80
GradientBoostingRegressor
RandomForestRegressor
DecisionTree

The evaluation metric is set based on **R^2 (coefficient of determination)** regression score function, that will represents the proportion of the variance for items (crops) in the regression model. **R^2** score shows how well terms (data points) fit a curve or line.

From the results above, **Gradient Boosting Regressor** has the highest R^2 score 0f **~96%**, **Decision Tree Regressor** comes second.

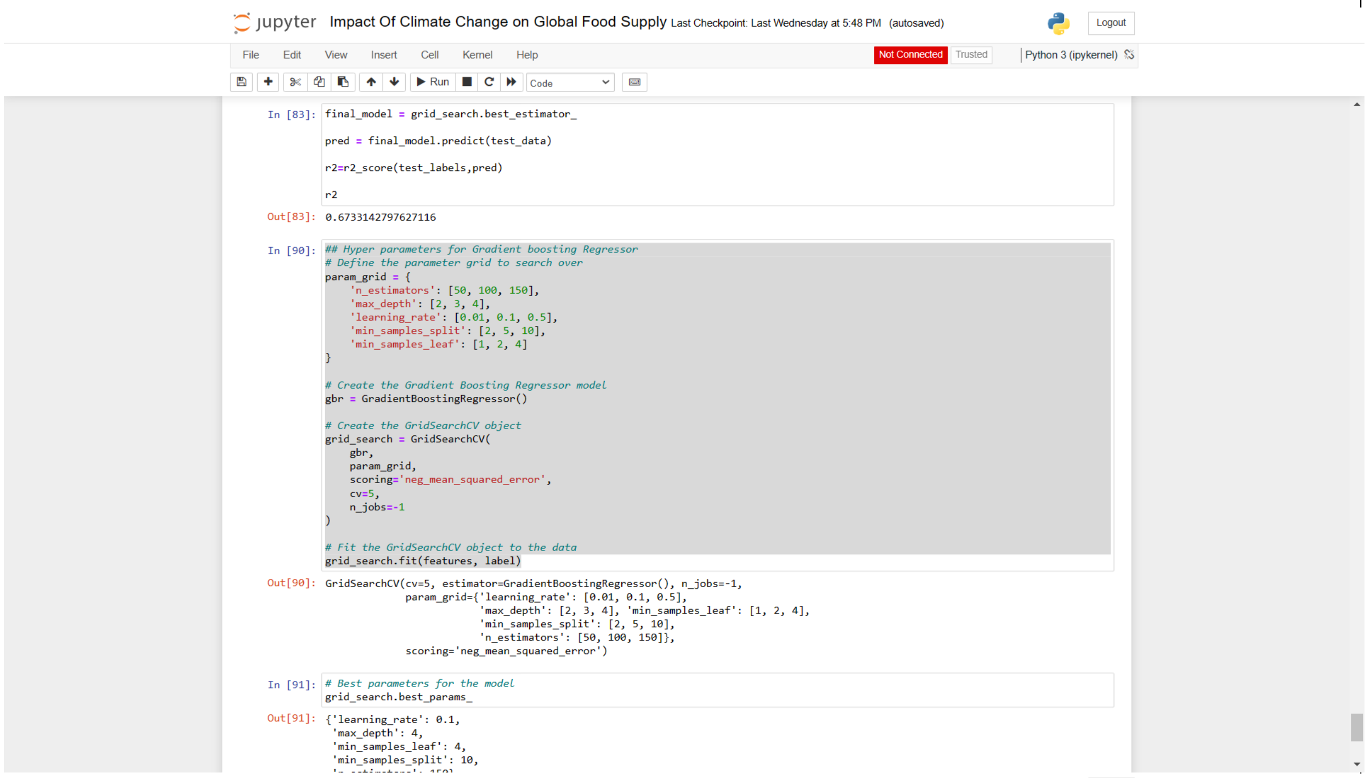
Hyper parameter tuning

```
In [80]: from sklearn.model_selection import GridSearchCV

         # Define the hyperparameters to search over
         param_grid = {
             'max_depth': [5, 10, 15, 20],
             'min_samples_split': [2, 5, 10],
             'min_samples_leaf': [1, 2, 4],
             'max_features': ['sqrt', 'log2']
         }

         # Create a decision tree regressor
         dt = DecisionTreeRegressor()
```


jupyter
Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

Hyper parameter tuning

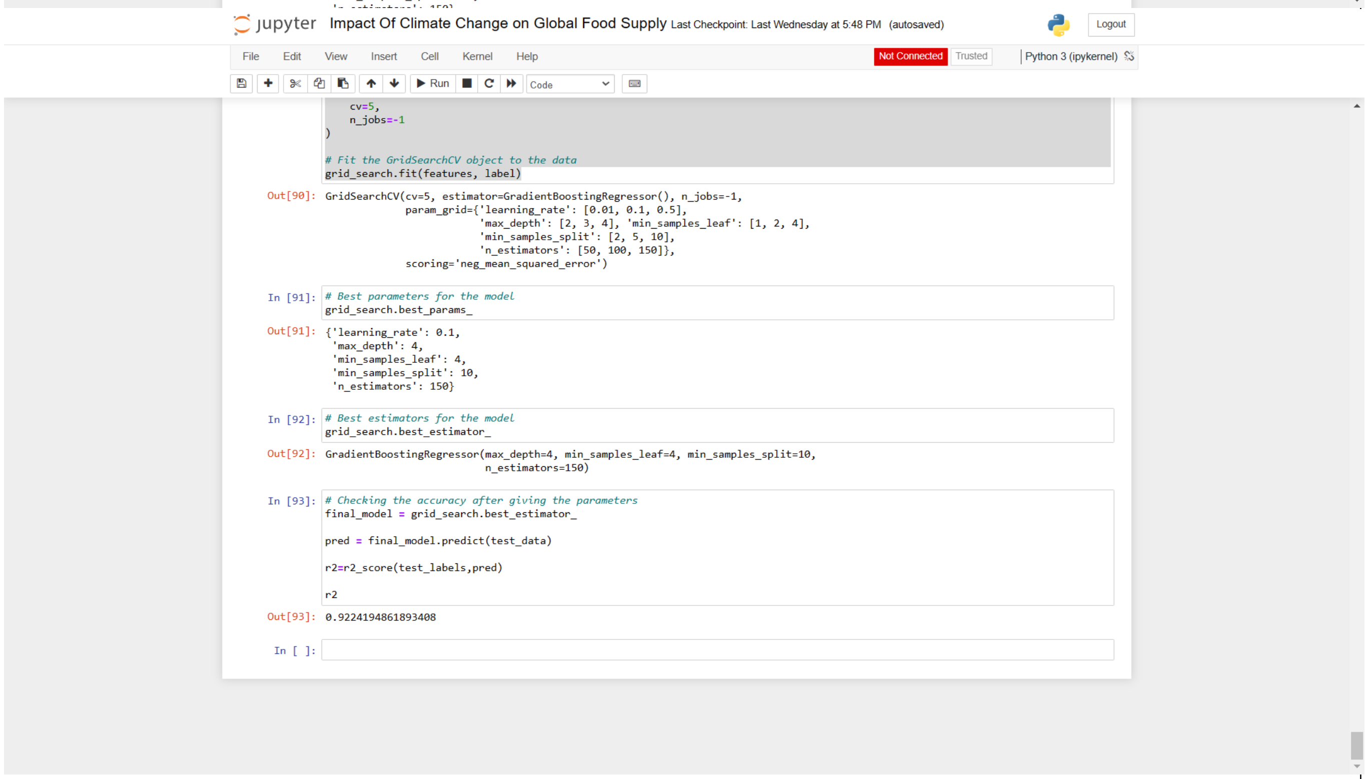
```
In [80]: from sklearn.model_selection import GridSearchCV

         # Define the hyperparameters to search over
         param_grid = {
             'max_depth': [5, 10, 15, 20],
             'min_samples_split': [2, 5, 10],
             'min_samples_leaf': [1, 2, 4],
             'max_features': ['sqrt', 'log2']
         }

         # Create a decision tree regressor
         dt = DecisionTreeRegressor()

         # Use GridSearchCV to perform hyperparameter tuning
         grid_search = GridSearchCV(dt, param_grid=param_grid, cv=5)
         grid_search.fit(features, label)

Out[80]: GridSearchCV(cv=5, estimator=DecisionTreeRegressor(),
                      param_grid={'max_depth': [5, 10, 15, 20],
                                  'max_features': ['sqrt', 'log2'],
                                  'min_samples_leaf': [1, 2, 4],
                                  'min_samples_split': [2, 5, 10]})

In [81]: # Best parameters for the model
         grid_search.best_params_

Out[81]: {'max_depth': 15,
          'max_features': 'sqrt',
          'min_samples_leaf': 4,
          'min_samples_split': 2}

In [82]: # Best estimators for the model
         grid_search.best_estimator_

Out[82]: DecisionTreeRegressor(max_depth=15, max_features='sqrt', min_samples_leaf=4)

In [83]: final_model = grid_search.best_estimator_

         pred = final_model.predict(test_data)

         r2=r2_score(test_labels,pred)

         r2

Out[83]: 0.6733142797627116

In [90]: ## Hyper parameters for Gradient boosting Regressor
```


jupyter Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

```
In [83]: final_model = grid_search.best_estimator_

         pred = final_model.predict(test_data)

         r2=r2_score(test_labels,pred)

         r2

Out[83]: 0.6733142797627116
```

```
In [90]: ## Hyper parameters for Gradient boosting Regressor
         # Define the parameter grid to search over
         param_grid = {
             'n_estimators': [50, 100, 150],
             'max_depth': [2, 3, 4],
             'learning_rate': [0.01, 0.1, 0.5],
             'min_samples_split': [2, 5, 10],
             'min_samples_leaf': [1, 2, 4]
         }

         # Create the Gradient Boosting Regressor model
         gbr = GradientBoostingRegressor()

         # Create the GridSearchCV object
         grid_search = GridSearchCV(
             gbr,
             param_grid,
             scoring='neg_mean_squared_error',
             cv=5,
             n_jobs=-1
         )

         # Fit the GridSearchCV object to the data
         grid_search.fit(features, label)

Out[90]: GridSearchCV(cv=5, estimator=GradientBoostingRegressor(), n_jobs=-1,
                      param_grid={'learning_rate': [0.01, 0.1, 0.5],
                                  'max_depth': [2, 3, 4], 'min_samples_leaf': [1, 2, 4],
                                  'min_samples_split': [2, 5, 10],
                                  'n_estimators': [50, 100, 150]},
                      scoring='neg_mean_squared_error')
```

```
In [91]: # Best parameters for the model
         grid_search.best_params_

Out[91]: {'learning_rate': 0.1,
          'max_depth': 4,
          'min_samples_leaf': 4,
          'min_samples_split': 10,
```

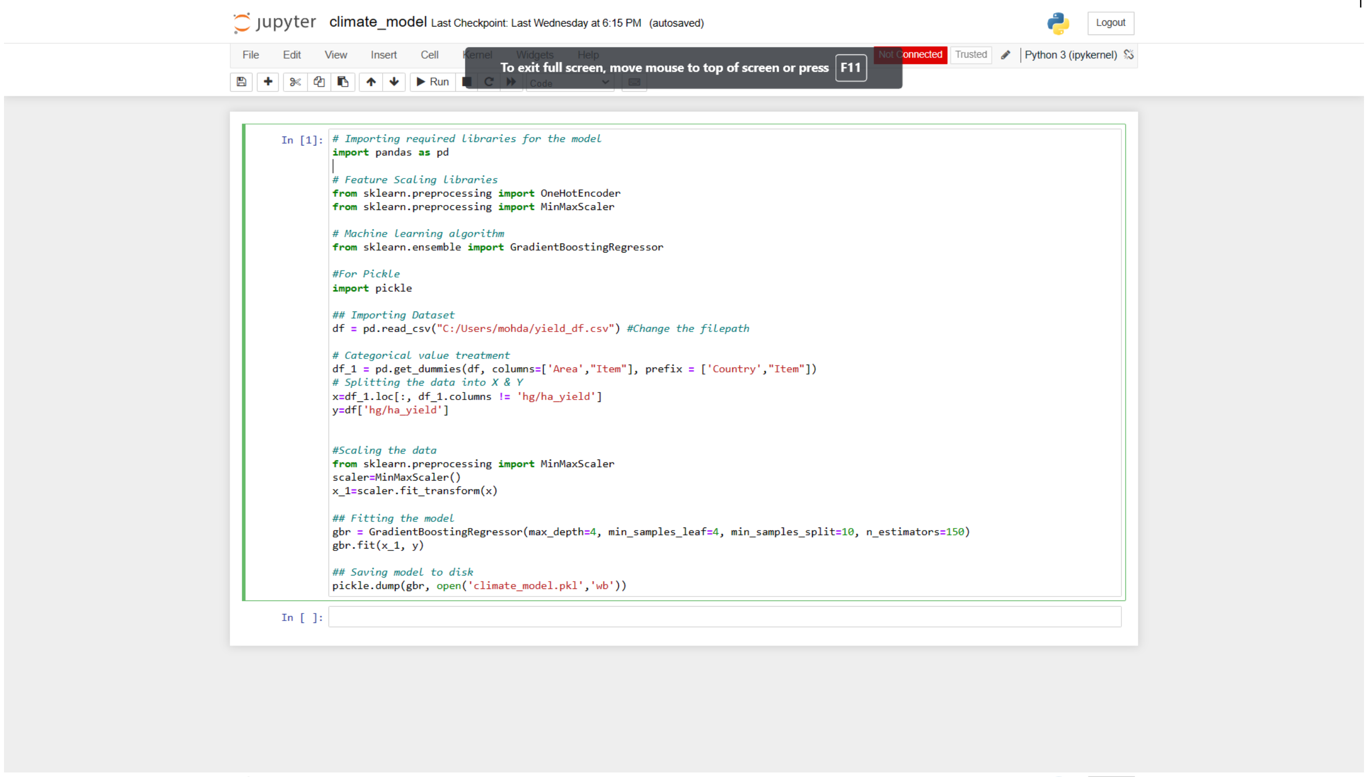

jupyter Impact Of Climate Change on Global Food Supply Last Checkpoint: Last Wednesday at 5:48 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

```
             cv=5,
             n_jobs=-1
         )

         # Fit the GridSearchCV object to the data
         grid_search.fit(features, label)

Out[90]: GridSearchCV(cv=5, estimator=GradientBoostingRegressor(), n_jobs=-1,
                      param_grid={'learning_rate': [0.01, 0.1, 0.5],
                                  'max_depth': [2, 3, 4], 'min_samples_leaf': [1, 2, 4],
                                  'min_samples_split': [2, 5, 10],
                                  'n_estimators': [50, 100, 150]},
                      scoring='neg_mean_squared_error')

In [91]: # Best parameters for the model
         grid_search.best_params_

Out[91]: {'learning_rate': 0.1,
          'max_depth': 4,
          'min_samples_leaf': 4,
          'min_samples_split': 10,
```

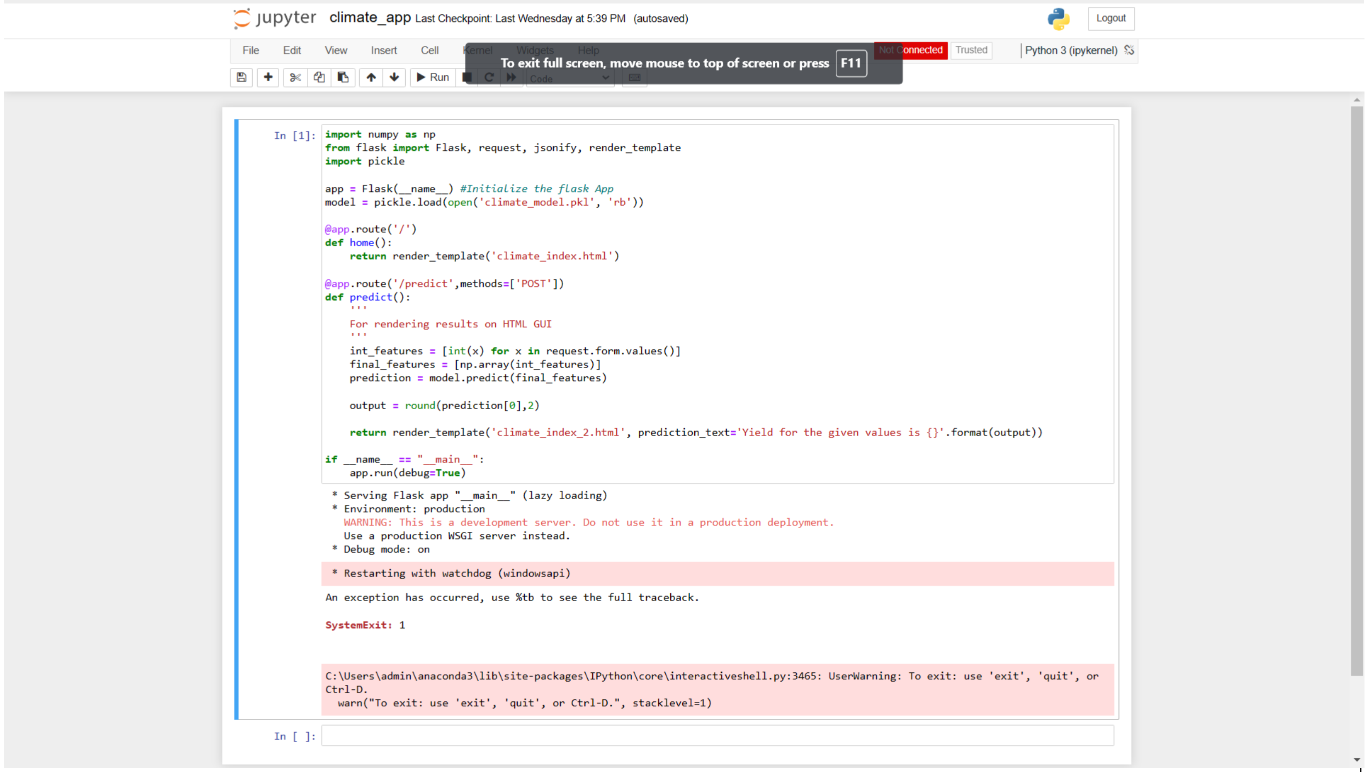
```
          'n_estimators': 150}
```

```
In [92]: # Best estimators for the model
         grid_search.best_estimator_

Out[92]: GradientBoostingRegressor(max_depth=4, min_samples_leaf=4, min_samples_split=10,
                                   n_estimators=150)
```

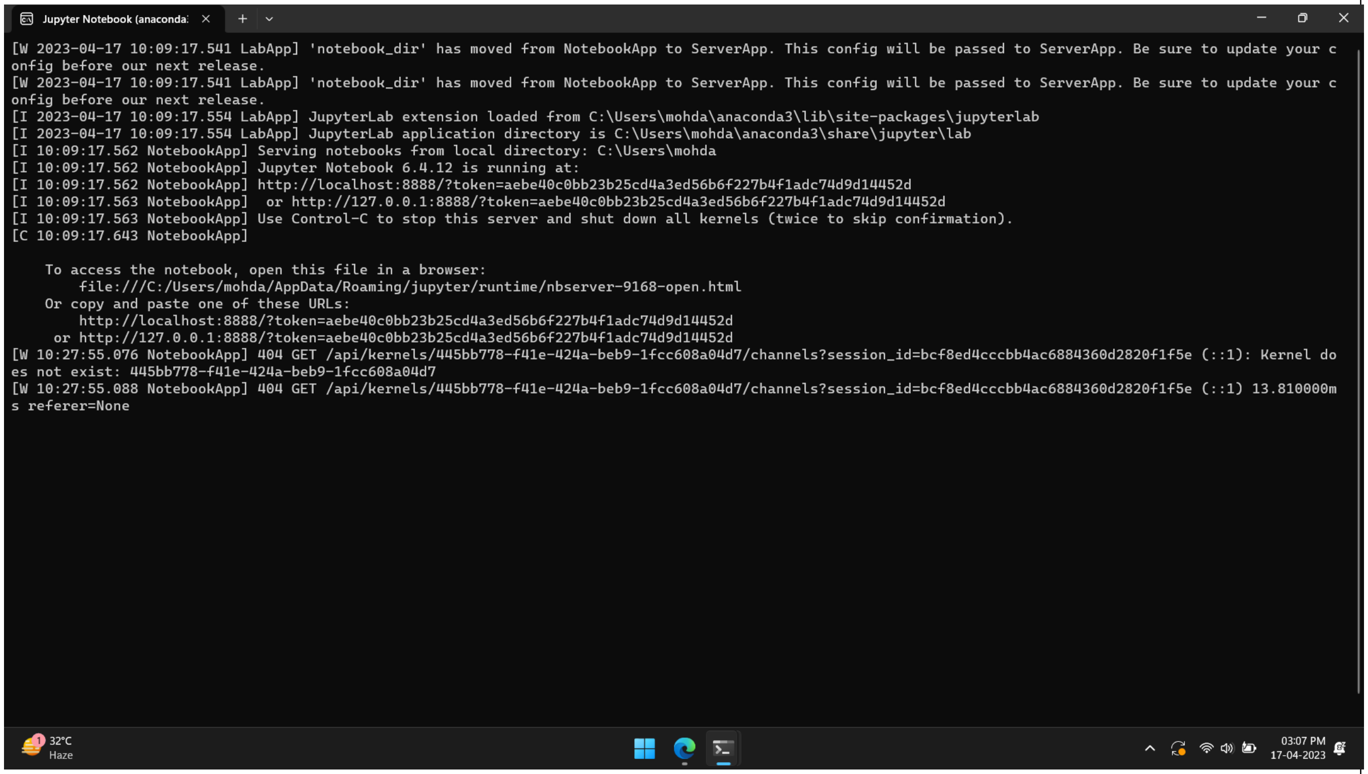
```
In [93]: # Checking the accuracy after giving the parameters
         final_model = grid_search.best_estimator_

         pred = final_model.predict(test_data)

         r2=r2_score(test_labels,pred)

         r2

Out[93]: 0.9224194861893408

In [ ]:
```


jupyter
climate_model Last Checkpoint: Last Wednesday at 6:15 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
To exit full screen, move mouse to top of screen or press F11
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

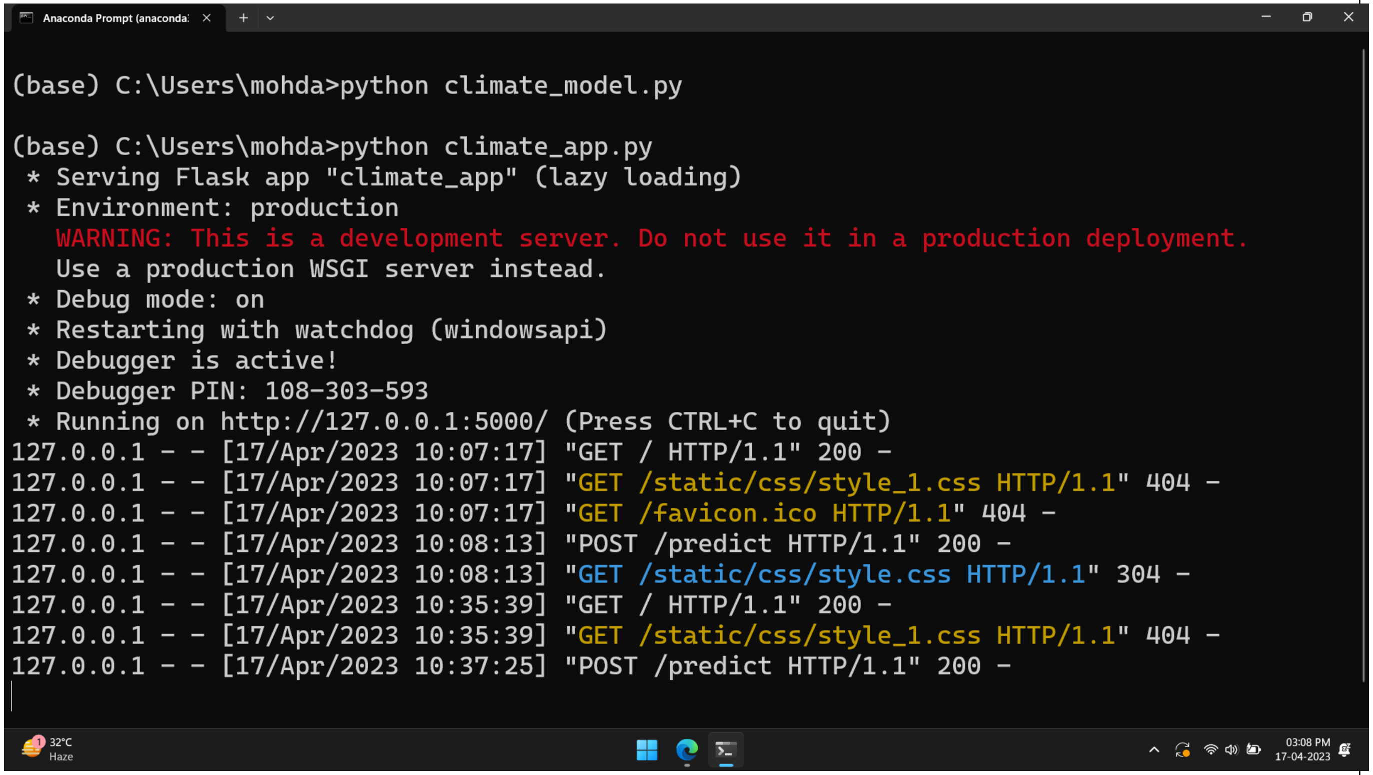
```
In [1]: # Importing required libraries for the model
        import pandas as pd

        # Feature Scaling libraries
        from sklearn.preprocessing import OneHotEncoder
        from sklearn.preprocessing import MinMaxScaler

        # Machine learning algorithm
        from sklearn.ensemble import GradientBoostingRegressor

        #For Pickle
        import pickle

        ## Importing Dataset
        df = pd.read_csv("C:/Users/mohda/yield_df.csv") #Change the filepath

        # Categorical value treatment
        df_1 = pd.get_dummies(df, columns=['Area',"Item"], prefix = ['Country',"Item"])
        # Splitting the data into X & Y
        x=df_1.loc[:, df_1.columns != 'hg/ha_yield']
        y=df['hg/ha_yield']


        #Scaling the data
        from sklearn.preprocessing import MinMaxScaler
        scaler=MinMaxScaler()
        x_1=scaler.fit_transform(x)

        ## Fitting the model
        gbr = GradientBoostingRegressor(max_depth=4, min_samples_leaf=4, min_samples_split=10, n_estimators=150)
        gbr.fit(x_1, y)

        ## Saving model to disk
        pickle.dump(gbr, open('climate_model.pkl','wb'))

In [ ]:
```


jupyter
climate_app Last Checkpoint: Last Wednesday at 5:39 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
To exit full screen, move mouse to top of screen or press F11
Not Connected
Trusted
Python 3 (ipykernel)
Run
Code

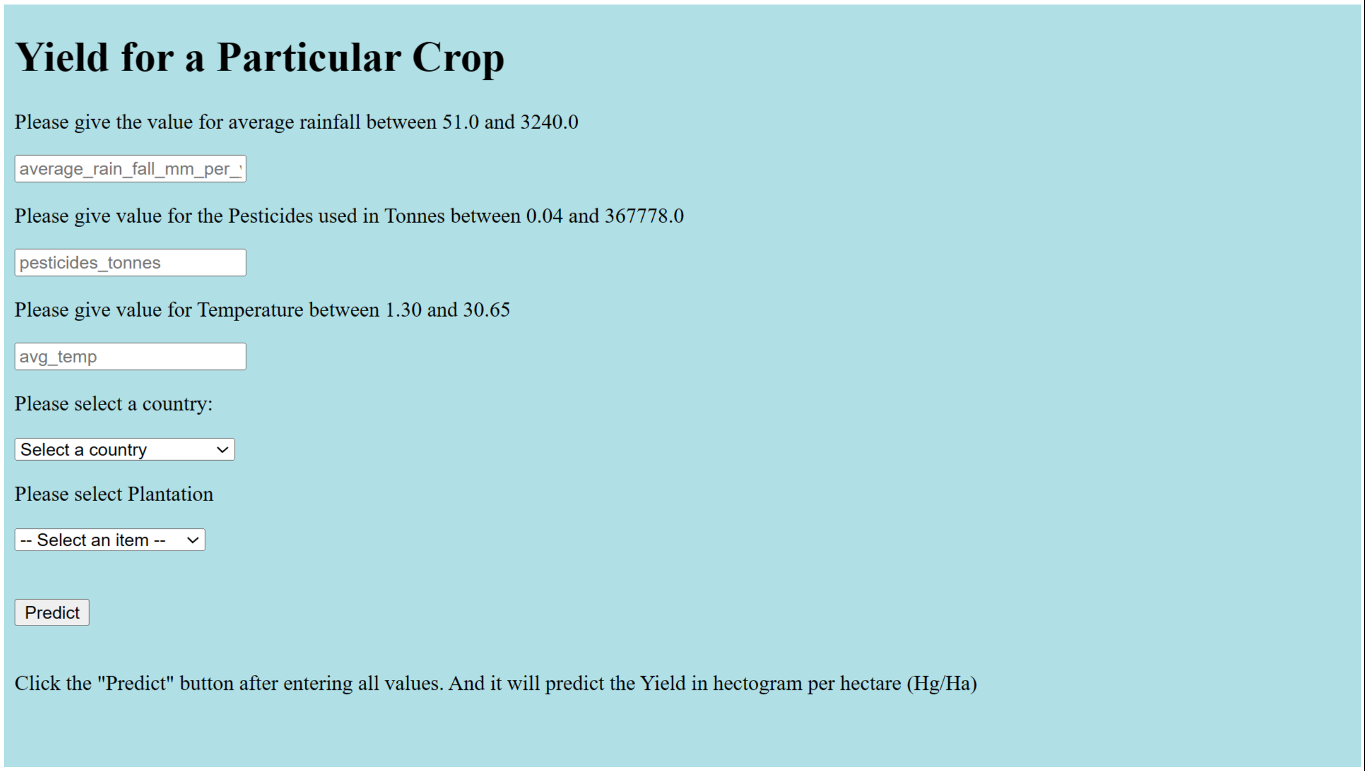
```
In [1]: import numpy as np
        from flask import Flask, request, jsonify, render_template
        import pickle

        app = Flask(__name__) #Initialize the flask App
        model = pickle.load(open('climate_model.pkl', 'rb'))

        @app.route('/')
        def home():
            return render_template('climate_index.html')

        @app.route('/predict',methods=['POST'])
        def predict():
            '''
            For rendering results on HTML GUI
            '''
            int_features = [int(x) for x in request.form.values()]
            final_features = [np.array(int_features)]
            prediction = model.predict(final_features)

            output = round(prediction[0],2)

            return render_template('climate_index_2.html', prediction_text='Yield for the given values is {}'.format(output))

        if __name__ == "__main__":
            app.run(debug=True)

         * Serving Flask app "__main__" (lazy loading)
         * Environment: production
           WARNING: This is a development server. Do not use it in a production deployment.
           Use a production WSGI server instead.
         * Debug mode: on

         * Restarting with watchdog (windowsapi)

        An exception has occurred, use %tb to see the full traceback.

        SystemExit: 1


        C:\Users\admin\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3465: UserWarning: To exit: use 'exit', 'quit', or
        Ctrl-D.
          warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)

In [ ]:
```



```
[W 2023-04-17 10:09:17.541 LabApp] 'notebook_dir' has moved from NotebookApp to ServerApp. This config will be passed to ServerApp. Be sure to update your c
onfig before our next release.
[W 2023-04-17 10:09:17.541 LabApp] 'notebook_dir' has moved from NotebookApp to ServerApp. This config will be passed to ServerApp. Be sure to update your c
onfig before our next release.
[I 2023-04-17 10:09:17.554 LabApp] JupyterLab extension loaded from C:\Users\mohda\anaconda3\lib\site-packages\jupyterlab
[I 2023-04-17 10:09:17.554 LabApp] JupyterLab application directory is C:\Users\mohda\anaconda3\share\jupyter\lab
[I 10:09:17.562 NotebookApp] Serving notebooks from local directory: C:\Users\mohda
[I 10:09:17.562 NotebookApp] Jupyter Notebook 6.4.12 is running at:
[I 10:09:17.562 NotebookApp] http://localhost:8888/?token=aebe40c0bb23b25cd4a3ed56b6f227b4f1adc74d9d14452d
[I 10:09:17.563 NotebookApp]  or http://127.0.0.1:8888/?token=aebe40c0bb23b25cd4a3ed56b6f227b4f1adc74d9d14452d
[I 10:09:17.563 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 10:09:17.643 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/mohda/AppData/Roaming/jupyter/runtime/nbserver-9168-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=aebe40c0bb23b25cd4a3ed56b6f227b4f1adc74d9d14452d
     or http://127.0.0.1:8888/?token=aebe40c0bb23b25cd4a3ed56b6f227b4f1adc74d9d14452d
[W 10:27:55.076 NotebookApp] 404 GET /api/kernels/445bb778-f41e-424a-beb9-1fcc608a04d7/channels?session_id=bcf8ed4cccbb4ac6884360d2820f1f5e (::1): Kernel do
es not exist: 445bb778-f41e-424a-beb9-1fcc608a04d7
[W 10:27:55.088 NotebookApp] 404 GET /api/kernels/445bb778-f41e-424a-beb9-1fcc608a04d7/channels?session_id=bcf8ed4cccbb4ac6884360d2820f1f5e (::1) 13.810000m
s referer=None
```



```
(base) C:\Users\mohda>python climate_model.py

(base) C:\Users\mohda>python climate_app.py
 * Serving Flask app "climate_app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 108-303-593
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [17/Apr/2023 10:07:17] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [17/Apr/2023 10:07:17] "GET /static/css/style_1.css HTTP/1.1" 404 -
127.0.0.1 - - [17/Apr/2023 10:07:17] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [17/Apr/2023 10:08:13] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [17/Apr/2023 10:08:13] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [17/Apr/2023 10:35:39] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [17/Apr/2023 10:35:39] "GET /static/css/style_1.css HTTP/1.1" 404 -
127.0.0.1 - - [17/Apr/2023 10:37:25] "POST /predict HTTP/1.1" 200 -
```



Yield for a Particular Crop

Please give the value for average rainfall between 51.0 and 3240.0

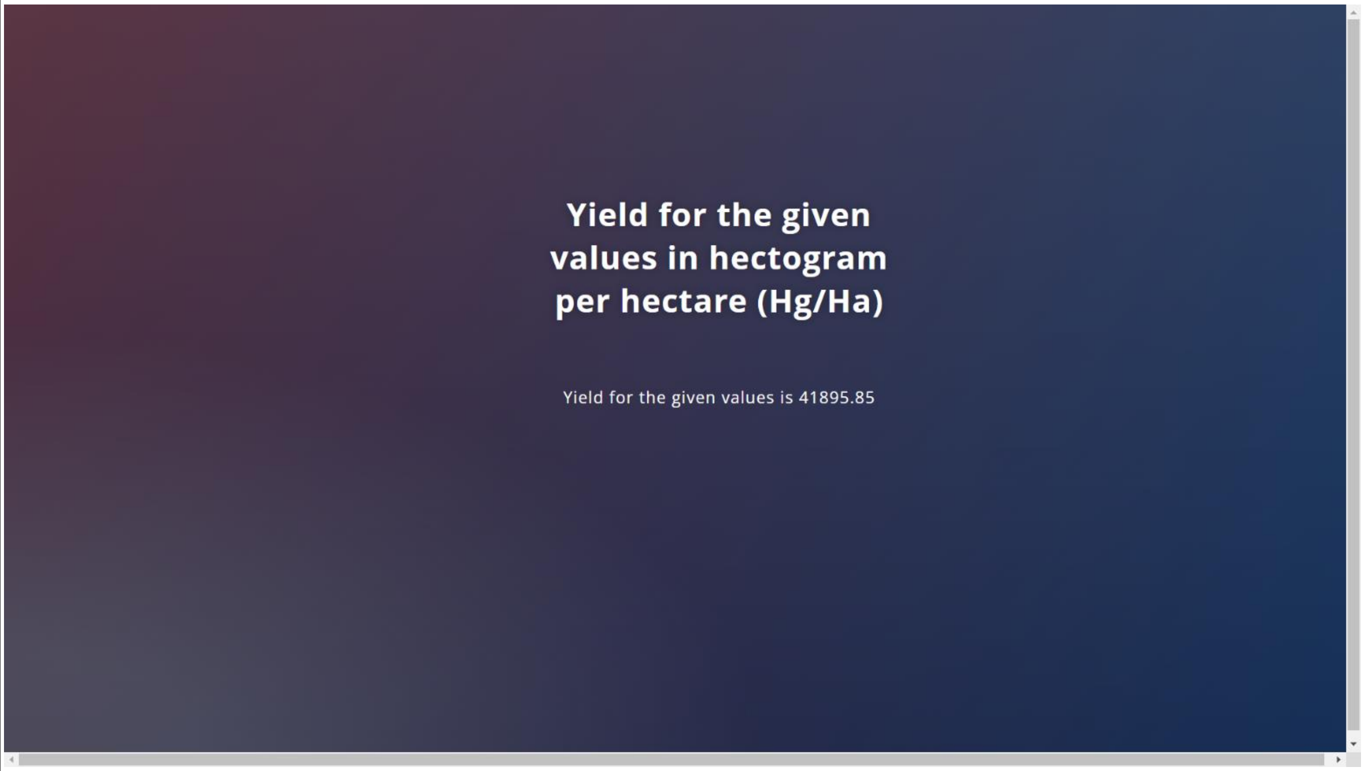
Please give value for the Pesticides used in Tonnes between 0.04 and 367778.0

Please give value for Temperature between 1.30 and 30.65

Please select a country:

Please select Plantation

Click the "Predict" button after entering all values. And it will predict the Yield in hectogram per hectare (Hg/Ha)



Yield for the given values in hectogram per hectare (Hg/Ha)

Yield for the given values is 41895.85

7: CODE DEVELPOMENT

```
## Importing all the required libraries
import numpy as np
import pandas as pd

# Data visualization
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix

# Feature scaling
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import MinMaxScaler

# Model Selection and Evaluation
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV

# Machine learning Algorithms
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn import svm
from sklearn.tree import DecisionTreeRegressor

# Performance
from sklearn.metrics import r2_score

#Tuning the model
from sklearn.model_selection import GridSearchCV

# Reading the yield data
df_yield = pd.read_csv("C:/Users/mohda/Impact-Of-Climate-Change-on-Global-Food-Supply-main/AI ML Project Dataset/yield.csv")
df_yield.shape

# Analysing the columns and its values
df_yield.head()
```

```
# Checking the last 5 rows of the data
df_yield.tail()

# Renaming the column "Value" to "hg/ha_yield"
df_yield = df_yield.rename(index=str, columns={"Value": "hg/ha_yield"})
df_yield.head()

# Dropping all the irrelevant columns

df_yield = df_yield.drop(['Year Code','Element Code','Element','Year Code','Area
Code','Domain Code','Domain','Unit','Item Code'], axis=1)

df_yield.head()


# Checking the min, max of the features

df_yield.describe()


# Checking the types of the columns

df_yield.info()


# Reading the rainfall data

df_rain = pd.read_csv("C:/Users/mohda/Impact-Of-Climate-Change-on-Global-Food-
Supply-main/AI ML Project Dataset/rainfall.csv")

df_rain.head()


# Renaming the column name as per the other data

df_rain = df_rain.rename(index=str, columns={" Area": 'Area'})


# Checking the datatypes

df_rain.info()
```

```
# Converting the column data types
df_rain = df_rain._convert(numeric=True)
df_rain.info()

# Droping the null values of the data
df_rain = df_rain.dropna()

# Analysing the features
df_rain.describe()

# Merging the 2 datasets
yield_df = pd.merge(df_yield, df_rain, on=['Year','Area'])

# Checking the shape of the dataset
yield_df.shape

# Analysing the data by checking top 5 rows of the data
yield_df.head()

# Analysing the data
yield_df.describe()

# Reading the pesticides dataset
df_pes = pd.read_csv("C:/Users/mohda/Impact-Of-Climate-Change-on-Global-Food-
Supply-main/AI ML Project Dataset/pesticides.csv")
df_pes.head()
```

```
# Renaming the columns and droping irrelevant columns
df_pes = df_pes.rename(index=str, columns={"Value": "pesticides_tonnes"})
df_pes = df_pes.drop(['Element','Domain','Unit','Item'], axis=1)
df_pes.head()

# Analysing the data
df_pes.describe()

# Checking the datatypes
df_pes.info()

# merge Pesticides dataframe with yield dataframe
yield_df = pd.merge(yield_df, df_pes, on=['Year','Area'])
yield_df.shape

# Checking whether the datasets are merged or not
yield_df.head()

# Reading the temperature data
avg_temp=  pd.read_csv("C:/Users/mohda/Impact-Of-Climate-Change-on-Global-Food-Supply-main/AI ML Project Dataset/temp.csv")

# Checking the top 5 rows of the data
avg_temp.head()

# Analysing the data
avg_temp.describe()
```

```
# Renaming the required columns
avg_temp = avg_temp.rename(index=str, columns={"year": "Year", "country":'Area'})
avg_temp.head()

# Mergint the datasets
yield_df = pd.merge(yield_df,avg_temp, on=['Area','Year'])
yield_df.head()

# Checking the shape of the maerged data
yield_df.shape

# Analysing the merged data
yield_df.describe()

# Checking the null values of the data
yield_df.isnull().sum()

plt.figure(1, figsize=(18, 7))
sns.set(style="whitegrid")
sns.countplot( x= 'Item', data=yield_df)
plt.title('Plantation')
plt.show()

# Group the data by crop type and calculate the mean yield, average rainfall, and pesticide usage
grouped_data = yield_df.groupby('Item').agg({'hg/ha_yield': 'mean',
'average_rain_fall_mm_per_year': 'mean', 'pesticides_tonnes': 'mean'}).reset_index()
```

```
# Bar chart showing the mean yield for each crop
plt.figure(figsize=(18, 7))
sns.barplot(x='Item', y='hg/ha_yield', data=grouped_data)
plt.title('Yield by Crop Type')
plt.xlabel('Crop Type')
plt.ylabel('Mean Yield (hg/ha)')
plt.show()

# Create a scatter plot showing the relationship between yield and average rainfall for each
crop
plt.figure(figsize=(18, 7))
sns.scatterplot(x='average_rain_fall_mm_per_year', y='hg/ha_yield', hue='Item',
data=yield_df)
plt.title('Yield vs Average Rainfall by Crop Type')
plt.xlabel('Average Rainfall (mm/year)')
plt.ylabel('Yield (hg/ha)')
plt.show()

# Create a chart showing the trend in pesticide usage over time
plt.figure(figsize=(10, 6))
sns.lineplot(x='Year', y='pesticides_tonnes', data=yield_df)
plt.title('Pesticide Usage')
plt.xlabel('Year')
plt.ylabel('Pesticide Usage (tonnes)')
plt.show()
```

```
# Create a scatter plot showing the relationship between yield and average temperature for each crop
plt.figure(figsize=(18, 7))
sns.scatterplot(x='avg_temp', y='hg/ha_yield', hue='Item', data=yield_df)
plt.title('Yield vs Average Temperature by Crop Type')
plt.xlabel('Average Temperature (°C)')
plt.ylabel('Yield (hg/ha)')
plt.show()

# Create a line chart showing the trend in average rainfall over time
plt.figure(figsize=(10, 6))
sns.lineplot(x='Year', y='average_rain_fall_mm_per_year', data=yield_df)
plt.title('Average Rainfall')
plt.xlabel('Year')
plt.ylabel('Average Rainfall (mm/year)')
plt.show()

# Analysing the data
yield_df.groupby('Item').count()

# Checking the unique values in the Area column
yield_df['Area'].nunique()

# Checking the largest yielding country
yield_df.groupby(['Area'],sort=True)['hg/ha_yield'].sum().nlargest(10)
```

```
# Checking the highest producing countries
yield_df.groupby(['Item','Area'],sort=True)['hg/ha_yield'].sum().nlargest(10)


# Checking the correlation
plt.figure(figsize=(15,10))
sns.heatmap(yield_df.corr(), annot=True, cbar=True)


# Dropping the year column
yield_df = yield_df.drop(['Year'], axis=1)


# Saving the data as CSV for future use
yield_df.to_csv('yield_df.csv',index=False)


# Categorical value treatment
yield_df_onehot = pd.get_dummies(yield_df, columns=['Area',"Item"], prefix =
['Country',"Item"])


# Splitting the data into X & Y
features=yield_df_onehot.loc[:, yield_df_onehot.columns != 'hg/ha_yield']
label=yield_df['hg/ha_yield']
features.head()
features.tail()


# # Droping the year column
# features = features.drop(['Year'], axis=1)


features.info()
```

```
#Scaling the data
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
features=scaler.fit_transform(features)

# Splitting the data into train & test
from sklearn.model_selection import train_test_split
train_data, test_data, train_labels, test_labels = train_test_split(features, label,
test_size=0.3, random_state=42)

# Ftting & checking the accuracy of the models
from sklearn.metrics import r2_score
def compare_models(model):
    model_name = model.__class__.__name__
    fit=model.fit(train_data,train_labels)
    y_pred=fit.predict(test_data)
    r2=r2_score(test_labels,y_pred)
    return([model_name,r2])

# Training the models
models = [
    GradientBoostingRegressor(n_estimators=200, max_depth=3, random_state=0),
     RandomForestRegressor(n_estimators=200, max_depth=3, random_state=0),
    svm.SVR(),
   DecisionTreeRegressor(max_depth= 15,max_features= 'sqrt',min_samples_leaf=
4,min_samples_split= 2)
]
```

```
# Ploting the accuracies of the model
fig = plt.figure(figsize = (10, 5))
plt.bar(['GradientBoostingRegressor','RandomForestRegressor','DecisionTree'],
[89.65768919264416,68.42532317855172,74.52454383993655], color
='maroon',width=0.4)
plt.ylabel("Accuracy")
plt.title("Accuracy of different ML Algorithms")
plt.show()

from sklearn.model_selection import GridSearchCV

# Define the hyperparameters to search over
param_grid = {
    'max_depth': [5, 10, 15, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2']
}

# Create a decision tree regressor
dt = DecisionTreeRegressor()

# Use GridSearchCV to perform hyperparameter tuning
grid_search = GridSearchCV(dt, param_grid=param_grid, cv=5)
grid_search.fit(features, label)
# Best parameters for the model
grid_search.best_params_
```

```
final_model = grid_search.best_estimator_
pred = final_model.predict(test_data)
r2=r2_score(test_labels,pred)
r2

## Hyper parameters for Gradient boosting Regressor
# Define the parameter grid to search over
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [2, 3, 4],
    'learning_rate': [0.01, 0.1, 0.5],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Create the Gradient Boosting Regressor model
gbr = GradientBoostingRegressor()

# Create the GridSearchCV object
grid_search = GridSearchCV(
    gbr,
    param_grid,
    scoring='neg_mean_squared_error',
    cv=5,
    n_jobs=-1
)
```

```
# Fit the GridSearchCV object to the data
grid_search.fit(features, label)

# Best parameters for the model
grid_search.best_params_

# Best estimators for the model
grid_search.best_estimator_

# Checking the accuracy after giving the parameters
final_model = grid_search.best_estimator_
pred = final_model.predict(test_data)
r2=r2_score(test_labels,pred)
r2

# Importing required libraries for the model
import pandas as pd

# Feature Scaling libraries
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import MinMaxScaler

# Machine learning algorithm
from sklearn.ensemble import GradientBoostingRegressor
```

```
#For Pickle
import pickle

## Importing Dataset
df = pd.read_csv("C:/Users/mohda/yield_df.csv") #Change the filepath

# Categorical value treatment
df_1 = pd.get_dummies(df, columns=['Area',"Item"], prefix = ['Country',"Item"])
# Splitting the data into X & Y
x=df_1.loc[:, df_1.columns != 'hg/ha_yield']
y=df['hg/ha_yield']



#Scaling the data
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
x_1=scaler.fit_transform(x)

## Fitting the model
gbr = GradientBoostingRegressor(max_depth=4, min_samples_leaf=4,
min_samples_split=10, n_estimators=150)
gbr.fit(x_1, y)

## Saving model to disk
pickle.dump(gbr, open('climate_model.pkl','wb'))
```

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle

app = Flask(__name__) #Initialize the flask App
model = pickle.load(open('climate_model.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('climate_index.html')

@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    int_features = [int(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)

    output = round(prediction[0],2)

    return  render_template('climate_index_2.html',  prediction_text='Yield  for
the given values is {}'.format(output))

if __name__ == "__main__":
    app.run(debug=True)
```

```
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>ML API</title>
   <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet'
type='text/css'>
<link   href='https://fonts.googleapis.com/css?family=Arimo'   rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet'
type='text/css'>
<link
href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style_1.css') }}">

</head>

<body style="background-color:powderblue;">
 <div class="login">
  <h1>Yield for a Particular Crop</h1>


     <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}" method="post">
      <p>Please give the value for average rainfall between 51.0 and 3240.0</p>
                 <input    type="text"    name="average_rain_fall_mm_per_year"
placeholder="average_rain_fall_mm_per_year" required="required" />
       <p>Please give value for the Pesticides used in Tonnes between 0.04 and
367778.0</p>
                              <input      type="text"      name="pesticides_tonnes"
placeholder="pesticides_tonnes" required="required" />
          <p>Please give value for Temperature between 1.30 and 30.65</p>
                 <input  type="text"  name="avg_temp"  placeholder="avg_temp"
required="required" />


          <!-- Country dropdown list -->
          <p>Please select a country:</p>
```

```
        <select name="country" required="required">
          <option value="" disabled selected>Select a country</option>
          <option value="Albania">Albania</option>
          <option value="Algeria">Algeria</option>
          <option value="Angola">Angola</option>
          <option value="Argentina">Argentina</option>
          <option value="Armenia">Armenia</option>
          <option value="Australia">Australia</option>
          <option value="Austria">Austria</option>
          <option value="Azerbaijan">Azerbaijan</option>
          <option value="Bahamas">Bahamas</option>
          <option value="Bahrain">Bahrain</option>
          <option value="Bangladesh">Bangladesh</option>
          <option value="Belarus">Belarus</option>
          <option value="Belgium">Belgium</option>
          <option value="Botswana">Botswana</option>
          <option value="Brazil">Brazil</option>
          <option value="Bulgaria">Bulgaria</option>
          <option value="Burkina Faso">Burkina Faso</option>
          <option value="Burundi">Burundi</option>
          <option value="Cameroon">Cameroon</option>
          <option value="Canada">Canada</option>
                <option value="Central African Republic">Central African
Republic</option>
          <option value="Chile">Chile</option>
          <option value="Colombia">Colombia</option>
          <option value="Croatia">Croatia</option>
          <option value="">--Select a country--</option>
          <option value="Denmark">Denmark</option>
          <option value="Dominican Republic">Dominican Republic</option>
          <option value="Ecuador">Ecuador</option>
          <option value="Egypt">Egypt</option>
          <option value="El Salvador">El Salvador</option>
          <option value="Eritrea">Eritrea</option>
          <option value="Estonia">Estonia</option>
          <option value="Finland">Finland</option>
          <option value="France">France</option>
          <option value="Germany">Germany</option>
```

```
<option value="Ghana">Ghana</option>
<option value="Greece">Greece</option>
<option value="Guatemala">Guatemala</option>
<option value="Guinea">Guinea</option>
<option value="Guyana">Guyana</option>
<option value="Haiti">Haiti</option>
<option value="Honduras">Honduras</option>
<option value="Hungary">Hungary</option>
<option value="India">India</option>
<option value="Indonesia">Indonesia</option>
<option value="Iraq">Iraq</option>
<option value="Ireland">Ireland</option>
<option value="Italy">Italy</option>
<option value="Jamaica">Jamaica</option>
<option value="Japan">Japan</option>
<option value="Kazakhstan">Kazakhstan</option>
<option value="Kenya">Kenya</option>
<option value="Latvia">Latvia</option>
<option value="Lebanon">Lebanon</option>
<option value="Lesotho">Lesotho</option>
<option value="Libya">Libya</option>
<option value="Lithuania">Lithuania</option>
<option value="Madagascar">Madagascar</option>
<option value="Malawi">Malawi</option>
<option value="Malaysia">Malaysia</option>
<option value="Mali">Mali</option>
<option value="Mauritania">Mauritania</option>
<option value="Mauritius">Mauritius</option>
<option value="Mexico">Mexico</option>
<option value="Montenegro">Montenegro</option>
<option value="Morocco">Morocco</option>
<option value="Mozambique">Mozambique</option>
<option value="Namibia">Namibia</option>
<option value="Nepal">Nepal</option>
<option value="Netherlands">Netherlands</option>
<option value="New Zealand">New Zealand</option>
<option value="Nicaragua">Nicaragua</option>
<option value="Niger">Niger</option>
```

```
  <option value="Norway">Norway</option>
  <option value="Pakistan">Pakistan</option>
  <option value="Papua New Guinea">Papua New Guinea</option>
  <option value="Peru">Peru</option>
  <option value="Poland">Poland</option>
  <option value="Portugal">Portugal</option>
  <option value="Qatar">Qatar</option>
  <option value="Romania">Romania</option>
  <option value="Rwanda">Rwanda</option>
  <option value="Saudi Arabia">Saudi Arabia</option>
  <option value="Senegal">Senegal</option>
  <option value="Slovenia">Slovenia</option>
  <option value="South Africa">South Africa</option>
  <option value="Spain">Spain</option>
  <option value="Sri Lanka">Sri Lanka</option>
  <option value="Sudan">Sudan</option>
  <option value="Suriname">Suriname</option>
  <option value="Sweden">Sweden</option>
  <option value="Switzerland">Switzerland</option>
  <option value="Tajikistan">Tajikistan</option>
  <option value="Thailand">Thailand</option>
  <option value="Tunisia">Tunisia</option>
  <option value="Turkey">Turkey</option>
  <option value="Uganda">Uganda</option>
  <option value="Ukraine">Ukraine</option>
  <option value="United Kingdom">United Kingdom</option>
  <option value="Uruguay">Uruguay</option>
  <option value="Zambia">Zambia</option>
  <option value="Zimbabwe">Zimbabwe</option>
</select>

<p>Please select Plantation </p>
<select name="item">
  <option value="">-- Select an item --</option>
  <option value="Cassava">Cassava</option>
  <option value="Maize">Maize</option>
  <option value="Plantains and others">Plantains and others</option>
  <option value="Potatoes">Potatoes</option>
```

```
            <option value="Rice, paddy">Rice, paddy</option>
            <option value="Sorghum">Sorghum</option>
            <option value="Soybeans">Soybeans</option>
            <option value="Sweet potatoes">Sweet potatoes</option>
            <option value="Wheat">Wheat</option>
            <option value="Yams">Yams</option>

          </select>

        </br>
        </br>
        </br>

                <button type="submit" class="btn btn-primary btn-block btn-
large">Predict</button>
    </form>

   <br>
   <p>Click the "Predict" button after entering all values. And it will predict the
Yield in hectogram per hectare (Hg/Ha)</p>

 </div>

</body>
</html>
```

```
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>ML API</title>
   <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet'
type='text/css'>
<link    href='https://fonts.googleapis.com/css?family=Arimo'    rel='stylesheet'
type='text/css'>
<link  href='https://fonts.googleapis.com/css?family=Hind:300'  rel='stylesheet'
type='text/css'>
<link
href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>

<body style="background-color:powderblue;">
 <div class="login">
  <h1>Yield for the given values in hectogram per hectare (Hg/Ha)</h1>

     <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">

    </form>

   <br>
   <br>
   {{ prediction_text }}


 </div>

</body>
</html>
```

8:PROBLEM FACED

PROBLEM FACED

1. While building this model the problem faced was that the Machine Learning Algorithms requires a lot of training data.

2. Data Availability: One of the biggest challenges in any machine learning project is to have enough data for training and testing the model. The availability of data on climate change impacts on food may be limited, making it difficult to build a robust model.

3. Data Quality: Even if there is enough data, the quality of the data may not be sufficient for the model to learn effectively. There may be missing or inaccurate data, outliers, or other data quality issues that can affect the performance of the model.

4. Choosing the Right Algorithm: There are many different machine learning algorithms available, and choosing the right one for your project can be a challenge. You will need to consider the type of data you have, the specific problem you are trying to solve, and the level of accuracy you need.

5. The other problem is that since the features are extracted it takes too much time to extract the features because the model is learning form it and also requires memory for calculating the features.

6. The requirement of GPU can be a problem to process the dataset as it required an efficient GPU for system training.

9:SCOPE AND LIMITATIONS

9.1 Scope

The scope of this project on the impact of climate change is extensive and wide-ranging. It aims to investigate the various effects of climate change on different aspects of life, such as agriculture, water resources, human health, and the economy. The project will analyse the past and present climate patterns and their changes, as well as predict the future trends of climate change. The data and information will be gathered from various sources, including research studies, reports, and datasets from scientific organizations, government agencies, and non-governmental organizations

This project will also explore the potential solutions and strategies that can be implemented to mitigate and adapt to the impact of climate change. The scope of this project extends beyond a mere academic exercise as it addresses real-world issues that affect not only the current generation but also future generations.

It is expected that the findings and recommendations of this project will contribute to the development of policies, practices, and measures that promote sustainability and resilience in the face of climate change.

9.2 Limitations

1. Data p: The accuracy of climate models is heavily reliant on the quality and quantity of data that is available. Limited data on some regions can lead to incomplete or inaccurate projections.
2. Uncertainties: Climate models involve a degree of uncertainty, and it is not always possible to accurately predict the impacts of climate change. The models are based on various assumptions, and unexpected events such as natural disasters can occur, which can greatly affect the outcome.
3. Complexity: Climate models are incredibly complex and involve a wide range of factors and variables. While the models can provide insights into the future impacts of climate change, they cannot take into account all possible scenarios.
4. Political and Social Factors: The impact of climate change is not just a scientific issue, but also a political and social one. The implementation of climate policies and measures depends on the political will of governments and the behavior of society.
5. Technical Limitations: The development and use of climate models require advanced technical knowledge and resources, including specialized software, high-performance computing resources, and skilled personnel. These limitations can hinder the development and use of climate models in some regions.

10: FUTURE ENHANCEMENT

Further Enhancement:

1. Include more data sources: While the project currently uses a range of data sources to model climate change impacts, there are likely other datasets that could be incorporated to improve the accuracy of the predictions. For example, additional climate models or satellite data could be used to provide more accurate information about temperature, precipitation, and other climate variables.

2. Use machine learning algorithms: The current project uses statistical models to predict climate change impacts, but machine learning algorithms could be employed to provide even more accurate predictions. These algorithms could learn from historical data to identify patterns and relationships between different climate variables and impacts, and use this information to make more accurate predictions.

3. Develop a web application: The project could be transformed into a web application that allows users to explore the potential impacts of climate change in different regions around the world. This would require developing an interactive user interface that allows users to input different scenarios (e.g., different levels of greenhouse gas emissions), and see how these scenarios would impact climate variables and different types of impacts (e.g., sea level rise, crop yields).

4. Incorporate social and economic data: While the current project focuses on the physical impacts of climate change, incorporating social and economic data could provide a more complete picture of the potential consequences. For example, including data on population growth and migration patterns could help to predict how climate change will impact human societies, while economic data could be used to estimate the costs of different types of impacts.

5. Use higher-resolution data: The current project uses data that is averaged over large regions, which can mask important local variations. Using higher-resolution data (e.g., data that is collected at the scale of individual cities or even smaller areas) could provide a more detailed picture of how climate change will impact specific regions. However, this would require access to more detailed data and computing resources.

11: REFRENCES AND BIBLOGRAPHY

REFRENCES AND BIBLOGRAPHY

1. David Rolnick1, Priya L. Donti2 and Lynn H. Kaack (2019). Tackling Climate Change with Machine Learning.

2. Sadrach Pierre, Ph.D. (2019). Climate Change and Food Scarcity in Developing Regions.

3. Cynthia Rosenzweig, Ana Iglesius, X.B.Yang, Paul R. Epstein and Eric Chivian (2001). Climate change and extreme weather events.

4. Aman Kharwal (2020). Predict Weather with Machine Learning.

5. Jayalakshmi and Savitha Devi (2021). Predictive Model construction for prediction of soil fertility using ML.

6. https://www.fao.org/faostat/en/#data

7. World Bank Open Data | Data

8. Current weather and forecast - OpenWeatherMap