Predicting the impact of climate changes on Crop Yield

A Project Report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF SCIENCE

in

MATHEMATICAL SCIENCES

Submitted by :

Md. Abid

Supervised by :

Dr. Veena Jain

Associate Professor, Department of Operational Research

Deen Dayal Upadhyay College

Mr. Grijesh Sharma

Mentor, School of learning, DataIQ

Department of Operational Research

Deen Dayal Upadhyay College

University of Delhi, New Delhi - 110078

May 2022

Department of Operational Research

Deen Dayal Upadhyay College

University of Delhi, New Delhi - 110078

Candidate's Declaration

I **MD Abid**, a student of **B.Sc. Mathematical Sciences**, hereby declare that, I am submitting the project entitled **"Predicting the impact of climate changes on Crop Yield"** to the Department of Operational Research, for the partial fulfillment of the requirement. for the award of the degree of Bachelor of Science in Mathematical Sciences , which is original and not copied from any source, without proper citation. This work has not previously formed the basis for awarding any Degree, Diploma, Associateship, Fellowship or other similar title or recognition.

MD Abid

Enrollment No. – 19015587010

B.Sc. Mathematical Sciences

Department of Operational Research

Deen Dayal Upadhyay College

University of Delhi, New Delhi - 110078

Department of Operational Research

Deen Dayal Upadhyay College

University of Delhi, New Delhi - 110078

Certificate

This is to certify that the project entitled "**Predicting the impact of climate changes on Crop Yield**" has been carried out by Md. Abid (19015587010), under my guidance for the partial fulfillment of the degree of Bachelor of Science in Mathematical Sciences (6^{th} Semester) of University of Delhi, during the academic year 2019-2022.To the best of my knowledge. this work has not previously formed the basis for awarding any Degree, Diploma, Associateship, Fellowship or other similar title or recognition.

Date : June 2, 2022

Supervisors :

Dr. Veena Jain

Associate Professor

Department of Operational Research

Deen Dayal Upadhyay College

Acknowledgement

It has been a matter of pride for us, the final year students of Delhi University to carry out User defined project under at the Department of Operational Research at our institute, Deen Dayal Upadhyay College. We take a moment to thank DU Innovation council for giving us the opportunity to learn from the industry, by bridging the gap between industry and academics.

We extend our gratitude to respected Dr. Hem Chand Jain Sir, the Principal of our college for providing all the facilities to perform our project work, thereby giving it the present shape.

We express our sincerest gratitude to Dr. Veena Jain, the Head of the Operational Research Department Mr. Grijesh Sharma for their continuous guidance throughout the eventful learning odyssey of this project.

Md Abid

Abstract

Crop agriculture is the backbone of our economy. Due to global warming and climate change, traditional farming in the regular months have been distorted and crops' yield has been ruined. This not only gives economic losses but also attributes as the main reason for farmer sucide. Now in tis advanced era of Sciences, time has come for technology to take over changes.

For a crop to grow, ambient rainfall and temperature is necessary. So as now due to climate change, temperature and rainfall cannot be well defined, for example, rains in December and January or irregular temperatures have made it difficult for farmers and common man to yield efficient plantations and get a beneficial load of yield of crops.

In this project, we are analyzing the impact of Precipitation, Climate Changes, Temperature Distribution on Annual Crop Yield. This will exemplify how Data Science can yield efficient and meaningful data ,which will play a vital role in prediction and IOT based applications. Use of Data Science in Agriculture is a growing field and has a wide scope in future.

.

Content

Chapter No.	Topic	Page No.
Chapter 1.	Introduction to Operational Research	7
	1.1 Needs of Operational Research.	7
	1.2 Some OR Methods & Techniques.	7
	1.3 Advantages of Operational Research.	7
	1.4 Limitations of Operational Research.	8
Chapter 2	An Introduction to Project	9
	2.1 Project Definition.	9
	2.2 Scope of Project.	9
	2.3 Objectives of Project.	9
	2.4 Tool Required.	10
Chapter 3	Dataset & Data Analysis.	11
	3.1 Acknowledgement of Data	11
	3.2 Data Cleaning.	11
	3.3 Data Exploration.	16
	3.4 Data Preprocessing.	19
	3.5 Data Training.	20
Chapter 4	Result Analysis	22
	4.1 Model Comparison.	22
Chapter 5	Conclusion.	32
Chapter 6	Reference.	35

Chapter 1: Introduction to Operational Research.

Operations Research (OR) is an analytical method of problem solving and decision making, that is useful in the management of organization. In operations research, problem is broken down into basic components and then solved in defined steps by mathematical analysis.

The process operations research can be broadly broken down into the following steps;

1. Identifying a problem that needs to be solved.
2. Constructing a model around the problem that resembles the real world and variables.
3. Using the model to derive solutions to the problem.
4. Testing each solution on the model and analyzing its success.
5. Implementing the solution to the actual problem.

1.1 Need of Operational research

Operations research is important because it creates implementable solutions to complex business challenges. It uses data to create information, which can then be used as insights to improve results and make better decisions about the future of the business.

1.2 Some OR Methods and Techniques.

There are some methods for Data analysis;

- Cluster analysis.
- Cohort analysis.
- Regression analysis.
- Factor analysis.
- Neural network.
- Data mining.
- Text analysis.
- Time series analysis.

1.3 Advantages of Operational research.

- Enhanced productivity. Operations research helps in improving the productivity of the organization.
- Linear programming. Management is responsible for making important decisions about the organization.

- Improved coordination.
- Lower risks of failure.
- Control on the system.

1.4 Limitations of Operational research.

- Magnitude of Computation.
- Non-Quantifiable Factors.
- Distance between User and Analyst.
- Time and Money Costs.
- Implementation.

Chapter 2: An Introduction to Project

Climate change weather impact on the crop yield to analyzing a prediction for future reference like weather plays an important role in crop production. Thus there is no aspect of crop culture that is immune to impact of weather. Weather factors contribute to optimal crop growth, development and yield.

2.1 Project Definition

Crop yield prediction is an important agricultural problem. The agricultural yield primarily depends on weather condition (rain, temperature, etc), pesticides and accurate information about history of crop yield is an important thing for making decisions related to agriculture risk management and future predictions.

2.2 Scope of Project

The science of training machines to learn and produce model for future predictions is widely used, and not for nothing. Agriculture plays a critical role in the global economy. With the continuing expansion of the human population understanding worldwide crop yield is central to addressing food security challenges and reducing the impact of climate change.

2.3 Objectives of Project

The aim of the project is seems like that the basic ingredients that sustain humans are similar. We eat a lot of corn, wheat, rice and other simple crops. In this project the prediction of top 10 most consumed yields all over the world is established by applying machine learning techniques. It consists of 10 most consumed crops. It is a regression problem.

These crops Include:

1) Cassava
2) Maize
3) Plantains & others
4) Potatoes
5) Rice, paddy
6) Sorghum
7) Soyabeans
8) Sweet potatoes
9) Wheat
10) Yams

2.4 Tools Required

There are many tools for data analysis like; R-Programming, Tableau, Python, SAS, Excel, Rapidminer and so on…

But we are using python because of Python is easy to learn as it is very similar to JavaScript, Ruby, and PHP. Alao, Python has very good machine learning libraries viz. Scikitlearn, Theano, and it can also handle text data so we are using Python notebook.

Chapter 3 Dataset & data analysis.

3.1 Acknowledgement of Data

All dataset (publicly available dataset) here are taken from FAO (Food & Agriculture Organization) and World Data Bank.

http://www.fao.org/home/en/

http://data.worldbank.org/

3.2 Data Cleaning

Gathering & Cleaning Data:

After importing required libraries, crops yield of 10 most consumed crops around the world was downloaded from FAO websites. The collected data include country, item, year starting from 1961 to 2016 and yield value.

```
In [1]:
import numpy as np
import pandas as pd

In [2]:
df_yield = pd.read_csv('../input/crop-yield-prediction-dataset/yield.csv')
df_yield.shape

Out[2]:
(56717, 12)

In [3]:
df_yield.head()
```

```
In [3]:
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]:
df_yield.tail()

Out[4]:
```

In [4]:

```
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

Looking at the columns in the csv, we can rename **Value** to **hg/ha_yield** to make it easier to recognise that this is our crops yields production value. In addition to removal of unnecessary coloumns like Area Code, Domain, Item Code, etc.

Climate Data: Rainfall

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

Rainfall has a dramatic effect on agriculture. For this project rainfall per year information was gathered from World Data Bank.

In [9]:

```
df_rain = pd.read_csv('../input/crop-yield-prediction-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]:

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

```
In [11]:
# check data types
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

In [12]:
# convert average_rain_fall_mm_per_year from object to float
```

```
# convert average_rain_fall_mm_per_year from object to float
df_rain['average_rain_fall_mm_per_year'] = pd.to_numeric(df_rain['average_rain_fall_mm_per_year'],errors =
'coerce')
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
dtypes: float64(1), int64(1), object(1)
memory usage: 210.2+ KB
```

```
In [4]:
df_yield.tail()

Out[4]:
```

Pesticide Data:

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


```
In [11]:
# check data types
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

In [12]:
# convert average_rain_fall_mm_per_year from object to float
```

Climate Change Impact On Yield [EDA + Viz] 20349

Notebook Data Logs Comments (0) Settings

In [19]:

```
df_pes = pd.read_csv('../input/crop-yield-prediction-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

Climate Change Impact On Yield [EDA + Viz] 20349

Notebook Data Logs Comments (0) Settings

In [20]:

```
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
3	Albania	1993	121.0
4	Albania	1994	201.0

In [21]:

```
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

```
yield_df.shape
```

Out[23]:

```
(18949, 6)
```

In [24]:

```
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 temperature to to each country was collected from World Data Bank.

In [25]:

```
avg_temp=  pd.read_csv('../input/crop-yield-prediction-dataset/temp.csv')
```

In [26]:

```
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]:

```
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
75%	1962.000000	23.762500
max	2013.000000	30.730000

So average temprature starts from 1743 and ends at 2013, with some empty rows that we have to drop.

In [28]:

```
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]:

```
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]:

```
yield_df.shape
```

In [32]:

```
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
```

Great, no empty values!

3.3 Data Exploration.

Yield_df is the final obtained dataframe;

In [33]:
```
yield_df.groupby('Item').count()
```
Out[33]:

	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
Wheat	3857	3857	3857	3857	3857	3857
Yams	847	847	847	847	847	847

It can be noticed to high varience in the value for which columns, later on I'll account for the scalling. The dataframe has 101 countries, ordering these by 10 the highest yield production.

In [36]:
```
yield_df.groupby(['Area'],sort=True)['hg/ha_yield'].sum().nlargest(10)
```
Out[36]:
```
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 datasets. Including items in the group by.

In [37]:
```
yield_df.groupby(['Item','Area'],sort=True)['hg/ha_yield'].sum().nlargest(10)
```
Out[37]:
```
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
```

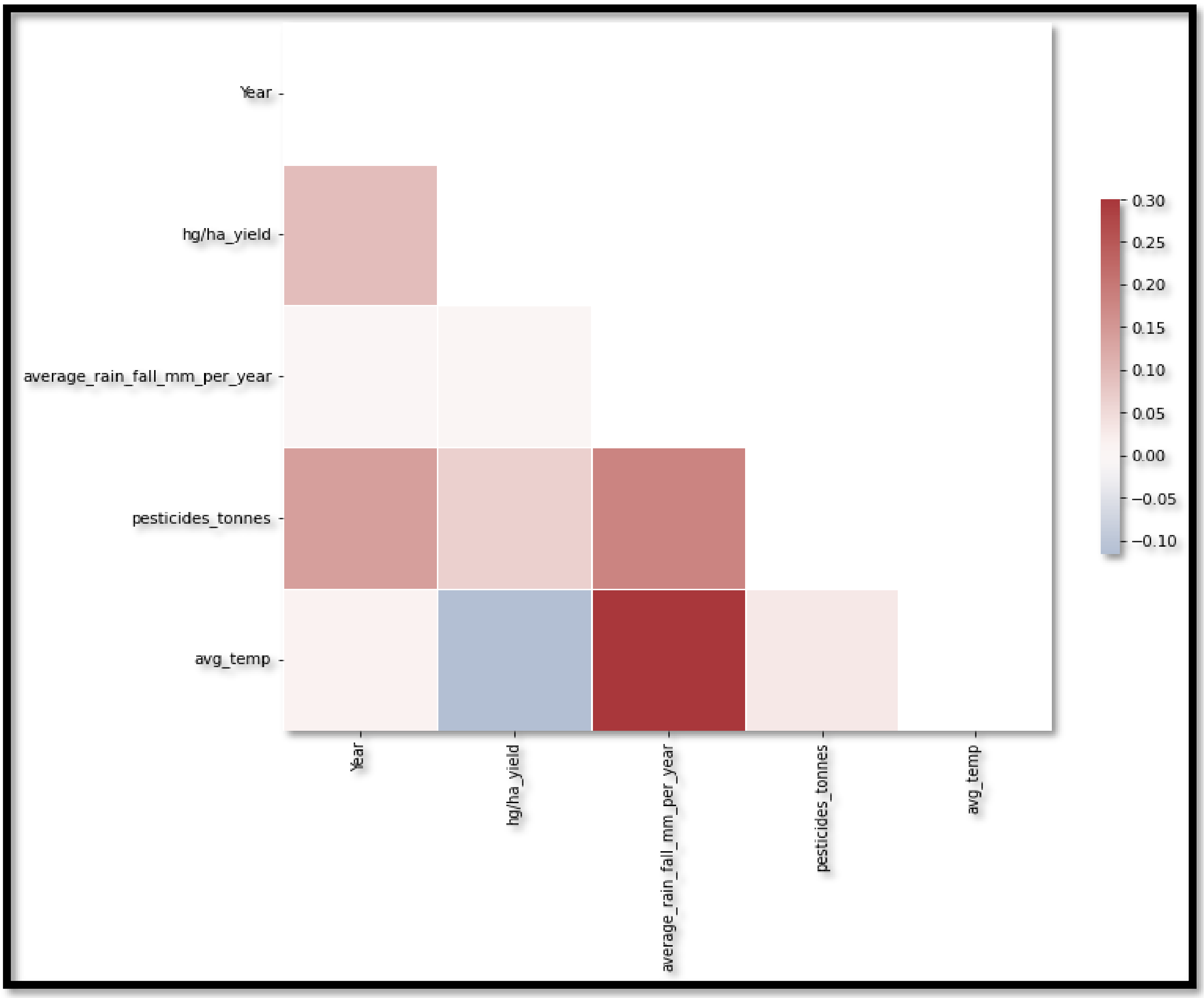
India is the highest for production of cassava and potatoes. Potatoes seems to be the dominated crop in the dataset, being in the highest in 4 countries

The final dataframe, starts from 1990 and end in 2013, that's 23 years worth 0f data for 101 countries.

Now, exploring the relationship between the column of the dataframe, a good way to quickly check correlation among column is by visualizing the correlation matrix as a heatmap.

```
In [39]:
correlation_data=yield_df.select_dtypes(include=[np.number]).corr()

mask = np.zeros_like(correlation_data, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.palette="vlag"

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(correlation_data, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5});
```

It can be seen from the correlation map that there is no correlation between any of the columns in the dataframe.



3.4 Data Preprocessing.

Data preprocessing is a technique that is used to convert the row data into a clean data set. In other word whenever the data is gathered from different sources. It is collected in row format which is not feasible for the analysis.

Encoding Categorical variables:

There are two categorical columns in the dataframe, categorical data are variables that contain label values rather than values. The number of possible values is often limited to a fixed set, like in this case, items and countries values. Many machine learning algorithms cannot operate on label data directly. They require all input variables and output variables to be numeric.

This means that categorical data must be converted to a numerical form. One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. For that purpose, One-Hot Encoding will be used to convert these two columns to one-hot numeric array.

The categorical value represent the numerical value of the entry in the dataset. This according will create a binary column for each category and returns a matrix with the results.

In [41]:
```
from sklearn.preprocessing import OneHotEncoder
```

In [42]:
```
yield_df_onehot = pd.get_dummies(yield_df, columns=['Area',"Item"], prefix = ['Country',"Item"])
features=yield_df_onehot.loc[:, yield_df_onehot.columns != 'hg/ha_yield']
label=yield_df['hg/ha_yield']
features.head()
```

Out[42]:

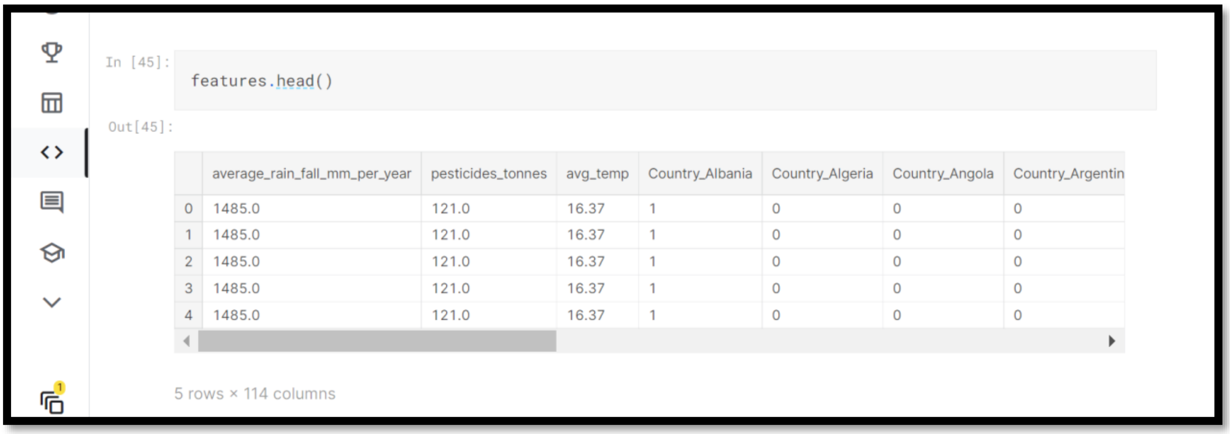
	Year	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country_Albania	Country_Algeria	Country_Angola	Country_
0	1990	1485.0	121.0	16.37	1	0	0	0
1	1990	1485.0	121.0	16.37	1	0	0	0
2	1990	1485.0	121.0	16.37	1	0	0	0
3	1990	1485.0	121.0	16.37	1	0	0	0

In [43]:
```
features = features.drop(['Year'], axis=1)
```

In [44]:
```
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: 3.9 MB
```

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



In [45]:
```
features.head()
```
Out[45]:

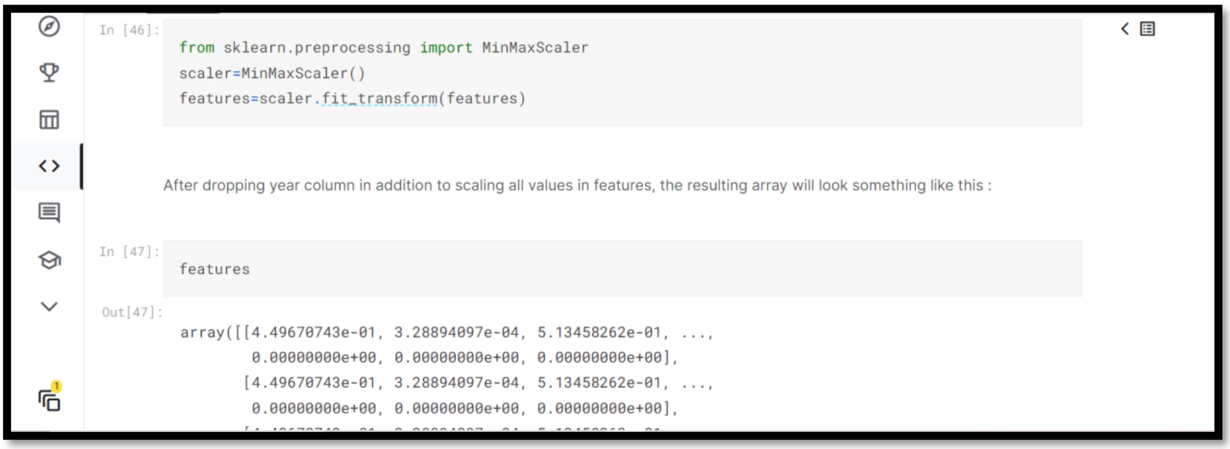
	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country_Albania	Country_Algeria	Country_Angola	Country_Argentin
0	1485.0	121.0	16.37	1	0	0	0
1	1485.0	121.0	16.37	1	0	0	0
2	1485.0	121.0	16.37	1	0	0	0
3	1485.0	121.0	16.37	1	0	0	0
4	1485.0	121.0	16.37	1	0	0	0

5 rows × 114 columns

Scaling Features:

Taking a look at the dataset above, it contains features highly varying in magnitudes, units and range. The features with high magnitudes will weigh in a lot more in the distance calculation than features with low magnitudes.

To suppress this effect, we need to bring all features to the same level of magnitudes. This can be achieved by scaling.



In [46]:
```
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 [47]:
```
features
```
Out[47]:
```
array([[4.49670743e-01, 3.28894097e-04, 5.13458262e-01, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [4.49670743e-01, 3.28894097e-04, 5.13458262e-01, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
```

3.5 Training Data:

The dataset will be split to two datasets, the training datasets and test datasets. The data is usually tend to be split inequality because training the model usually requires as much data-points as possible. The common splits are 70/30 or 80/20for train/test.

The training dataset is the initial dataset used to train ML algorithms to learn and produce right predictions (70% of dataset is training dataset)

The test dataset, however, is used to assess how well ML algorithm is trained with the training dataset. You can't simply reuse the training dataset in the testing stage because ML algorithm will already "know" the expected output, which defeats the purpose of testing the algorithm. (30% of dataset is testing dataset).

```
In [48]:
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)

In [49]:
#write final df to csv file
#yield_df.to_csv('../input/crop-yield-prediction-dataset/yield_df.csv')

In [50]:
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)
```

Chapter 4: Result analysis.

4.1 Model Comparison

Regression Analysis:-

A regression analysis is a statical technique for determining the relationship between a single dependent variable (actual) and one more independent (predict) variable. The analysis yields a predicted value for the criterion resulting from a linear combination of the predictors.

Regression is a way of mathematically sorting out which of those variable does indeed have an impact. It answer the question : Which factors matter most? How do those factor interact with each other?

Formula **Y = a + b(x) + u**

Type of Regression model

1. **Linear Regression:-** This is used when the outcome variable is linearly dependent on the independent variables. It is normally used when we don't have a huge data set. It is also sensitive to outliers, so if the data set contains outliers, then it's better to treat them before applying linear regression. There are single and multi-variable regression techniques. Simple Linear Regression is the analysis when the outcome variable is linearly dependent on a single independent variable. Simple Linear Regression follows the equation of a straight line which is given below:

Y=mx+c

Where,

- Y= Target, Dependent, or Criterion Variable
- x= Independent or predictor variable
- m= Slope or Regression Coefficient

- c= constant

Multi-Variable Linear regression defines the relationship between the outcome variable and more than one independent variable. It follows the below equation of a straight line where dependent variables are the linear combination of all the independent variables:

Y= m1x1+m2x2+m3x3+…mnan+c

Where,

- Y= Target, Dependent, or Criterion Variable
- x1, x2, x3…xn= Independent or predictor variables
- m1, m2, m3…mn= Slope or Regression Coefficients of respective variables
- c= constant

Linear Regression follows the principle of the Least Square method. This method states that a line of best fit is chosen by minimizing the sum of square error. The line of best fit is chosen where the sum of square error between the observed data and the line is minimum

2. **Logistic Regression :-** This regression technique is used when the target or outcome variable is categorical or binary in nature. The main difference between linear and logistic regression lies in the target variable, in linear regression, it should be continuous whereas in logistic it should be categorical. The outcome variable should only have two classes, not more than that. Some of the examples are spam filters in emails (Spam or not), fraud detection (Fraud/ Not Fraud), etc. It works on the principle of probability. It can be classified into two categories by setting the threshold

Non-Linear Analysis:- Nonlinear regression is a mathematical model that fits an equation to certain data using a generated line. As is the case with a linear regression that uses a straight-line equation (such as Ỵ= c + m x), nonlinear regression shows association using a curve, making it nonlinear in the parameter.

A simple nonlinear regression model is expressed as follows:

$$Y = f(X,\beta) + \epsilon$$

Where:

- **X** is a vector of P predictors
- **β** is a vector of k parameters
- **F** (-) is the known regression function
- **ϵ** is the error term

Alternatively, the model can also be written as follows:

$$Y_i = h\,[x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(m)}\,;\, \Theta_1, \Theta_2, \dots, \Theta_p] + E_i$$

Where:

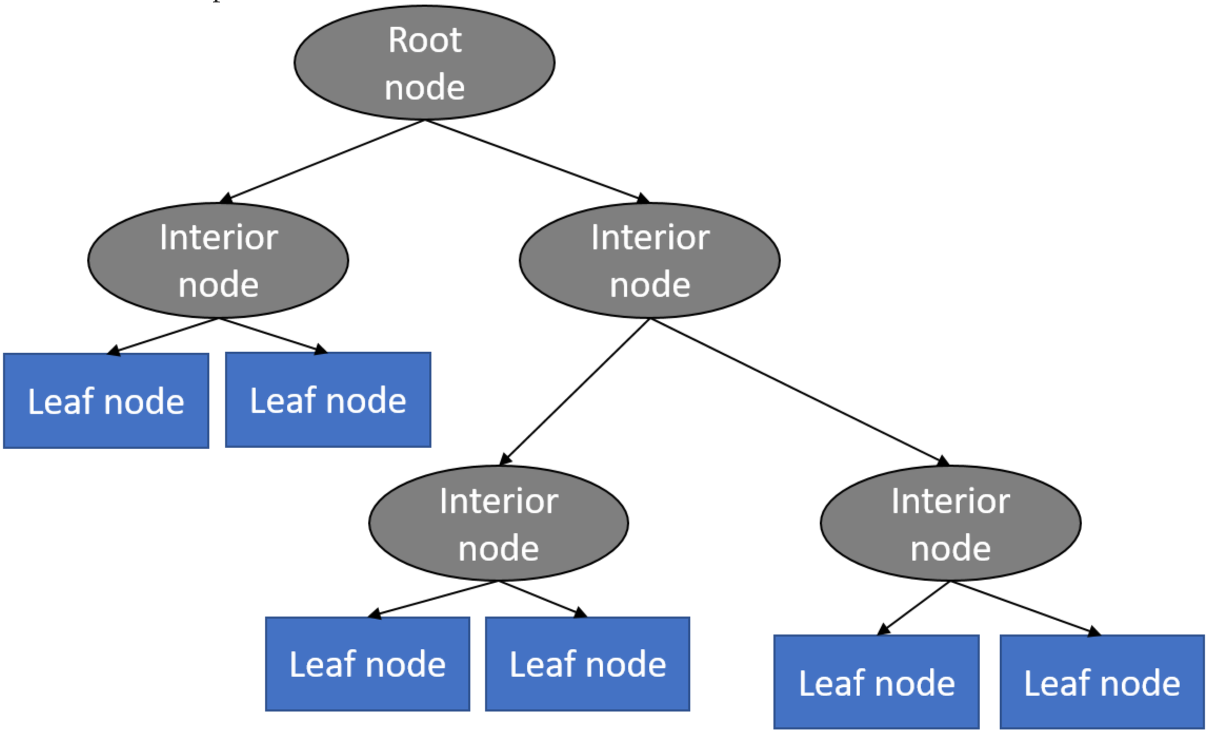
- **Y_i** is the responsive variable
- **h** is the function
- **x** is the input
- **Θ** is the parameter to be estimated

Now we are using DecisionTreeRegressor model for analysis:

DecisionTreeRegressor :=

Decision Tree is one of the most commonly used, practical approaches for supervised learning. It can be used to solve both Regression and Classification tasks with the latter being put more into practical application.

It is a tree-structured classifier with three types of nodes. The ***Root Node*** is the initial node which represents the entire sample and may get split further into further nodes. The ***Interior Nodes*** represent the features of a data set and the branches represent the decision rules. Finally, the ***Leaf Nodes*** represent the outcome. This algorithm is very useful for solving decision-related problems.



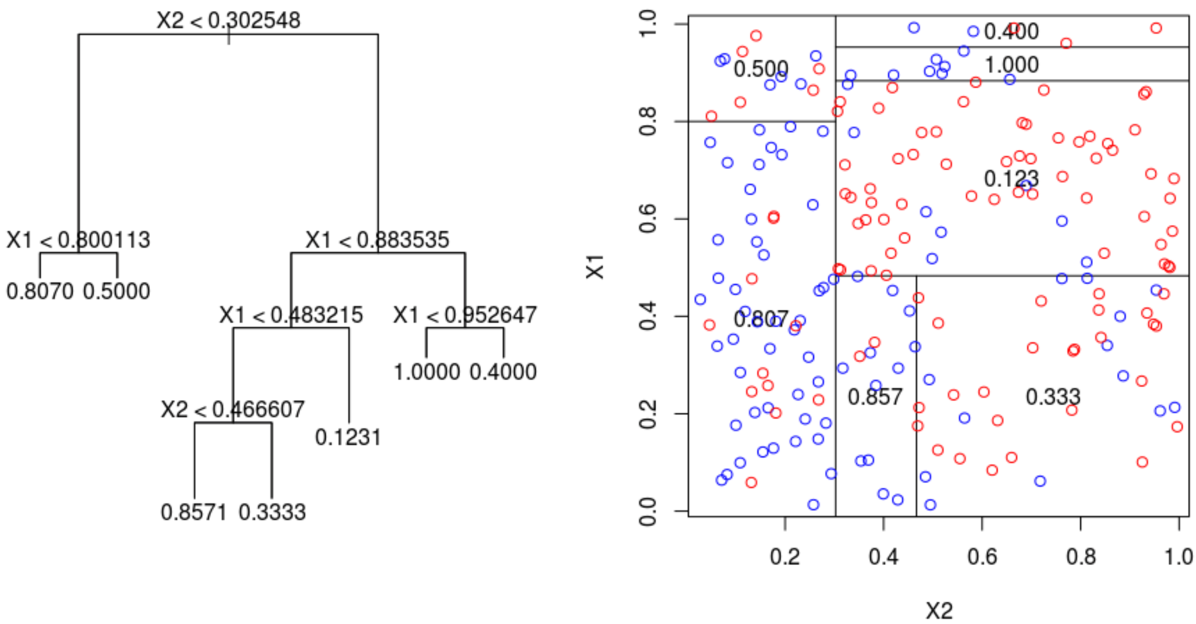
With a particular data point, it is run completely through the entirely tree by answering *True/False* questions till it reaches the leaf node. The final prediction is the average of the value of the dependent variable in that particular leaf node. Through multiple iterations, the Tree is able to predict a proper value for the data point.

The above diagram is a representation for the implementation of a Decision Tree algorithm. Decision trees have an advantage that it is easy to understand, lesser data cleaning is required, non-linearity does not affect the model's performance and the number of hyper-parameters to be tuned is almost null. However, it may have an over-fitting problem, which can be resolved using the ***Random Forest*** algorithm

How does it work?

A decision tree is arriving at an estimate by asking a series of questions to the data, each question narrowing our possible values until the model get confident enough to make a single prediction. The order of the question as well as their content are being determined by the model. In addition, the questions asked are all in a True/False form.

This is a little tough to grasp because it is not how humans naturally think, and perhaps the best way to show this difference is to create a real decision tree from. In the above problem x1, x2 are two features which allow us to make predictions for the target variable y by asking True/False questions.



For each True and False answer there are separate branches. No matter the answers to the questions, we eventually reach a prediction (leaf node). Start at the root node at the top and progress through the tree answering the questions along the way. So given any pair of X1, X2.

One aspect of the decision tree I should mention is how it actually learns (how the 'questions' are formed and how the thresholds are set). As a supervised machine learning model, a decision tree learns to map data to outputs in what is called the training phase of model building.

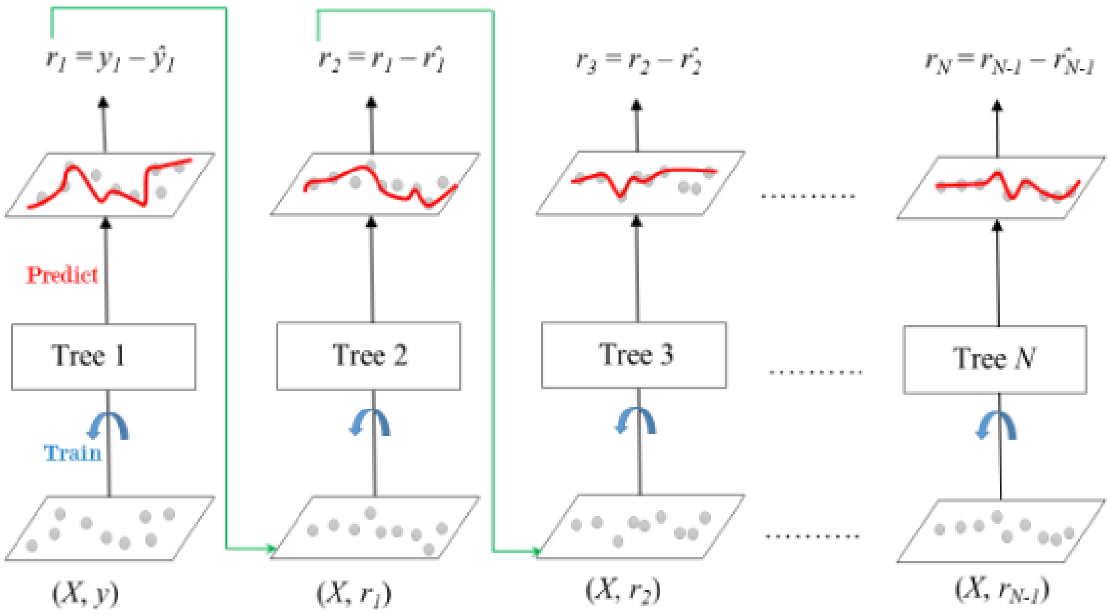
During training, the model is fitted with any historical data that is relevant to the problem domain and the true value we want the model to learn to predict. The model learns any relationships between the data and the target variable.

After the training phase, the decision tree produces a tree similar to the one shown above, calculating the best questions as well as their order to ask in order to make the most accurate estimates possible. When we want to make a prediction the same data format should be provided to the model in order to make a prediction. **The prediction will be an estimate based on the train data that it has been trained on**.

GradiantBoostingRegressor:-

Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor's error. In contrast to Adaboost, the weights of the training

instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels.
There is a technique called the **Gradient Boosted Trees** whose base learner is CART (Classification and Regression Trees).
The below diagram explains how gradient boosted trees are trained for regression problems.



Gradient Boosted Trees for Regression

The ensemble consists of *N* trees. Tree1 is trained using the feature matrix *X* and the labels *y*. The predictions labelled *y1(hat)* are used to determine the training set residual errors *r1*. Tree2 is then trained using the feature matrix *X* and the residual errors *r1* of Tree1 as labels. The predicted results *r1(hat)* are then used to determine the residual *r2*. The process is repeated until all the *N* trees forming the ensemble are trained.
There is an important parameter used in this technique known as **Shrinkage**.
Shrinkage refers to the fact that the prediction of each tree in the ensemble is shrunk after it is multiplied by the learning rate (eta) which ranges between 0 to 1. There is a trade-off between eta and number of estimators, decreasing learning rate needs to be compensated with increasing estimators in order to reach certain model performance. Since all trees are trained now, predictions can be made.
Each tree predicts a label and final prediction is given by the formula,

y(pred) = y1 + (eta * r1) + (eta * r2) + ... + (eta * rN)

The class of the gradient boosting regression in scikit-learn is **GradientBoostingRegressor**. A similar algorithm is used for classification known as **GradientBoostingClassifier**.

```
In [51]:
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]:
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn import svm
from sklearn.tree import DecisionTreeRegressor

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()
]

In [53]:
model_train=list(map(compare_models,models))

In [54]:
print(*model_train, sep = "\n")

['GradientBoostingRegressor', 0.8965768919264416]
```

```
['GradientBoostingRegressor', 0.8965768919264416]
['RandomForestRegressor', 0.6842532317855172]
['SVR', -0.19543203867357395]
['DecisionTreeRegressor', 0.9596499629269858]
```

The evaluation matric 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.

R^2 is a statical measure between 0 and 1 which calculates how similar a regression line is to the data it's fitted to. If it's a 1, the model 100% predicts the data variance; if it's a 0, the model predicts none of the variance.

From results viewed above**, Decision True Regressor** has the highest R^2 score of 96%,

Gradient Boosting Regressor comes second

I'll also calculated **Adjusted R^2** also indicates how well terms fit a curve or line, but adjust for the number of terms in a model. If you add more and more useless variables to a model, adjusted r-squared will decrease. If you add more useful variables, adjusted r-squared will always be less than or equal to R2.

```
yield_df_onehot = yield_df_onehot.drop(['Year'], axis=1)
```

In [56]:

```
yield_df_onehot.head()
```

Out[56]:

	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country_Albania	Country_Algeria	Country_Angola	Co
0	36613	1485.0	121.0	16.37	1	0	0	0
1	66667	1485.0	121.0	16.37	1	0	0	0
2	23333	1485.0	121.0	16.37	1	0	0	0
3	12500	1485.0	121.0	16.37	1	0	0	0
4	7000	1485.0	121.0	16.37	1	0	0	0

5 rows × 115 columns

```
#setting test data to columns from dataframe and excluding 'hg/ha_yield' values where ML model should
be predicting

test_df=pd.DataFrame(test_data,columns=yield_df_onehot.loc[:, yield_df_onehot.columns != 'hg/ha_yie
ld'].columns)

# using stack function to return a reshaped DataFrame by pivoting the columns of the current datafram
e

cntry=test_df[[col for col in test_df.columns if 'Country' in col]].stack()[test_df[[col for col in
test_df.columns if 'Country' in col]].stack()>0]
cntrylist=list(pd.DataFrame(cntry).index.get_level_values(1))
countries=[i.split("_")[1] for i in cntrylist]
itm=test_df[[col for col in test_df.columns if 'Item' in col]].stack()[test_df[[col for col in test
_df.columns if 'Item' in col]].stack()>0]
itmlist=list(pd.DataFrame(itm).index.get_level_values(1))
```

In [58]:

```
test_df.head()
```

Out[58]:

	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country_Albania	Country_Algeria	Country_Angola	Country_Argentir
0	0.183443	0.110716	0.542078	0.0	0.0	0.0	0.0
1	0.458451	0.000413	0.627257	0.0	0.0	0.0	0.0
2	0.183443	0.106159	0.518228	0.0	0.0	0.0	0.0
3	1.000000	0.224154	0.890971	0.0	0.0	0.0	0.0
4	0.458451	0.000355	0.625213	0.0	0.0	0.0	0.0

5 rows × 114 columns

In [59]:

```
test_df.drop([col for col in test_df.columns if 'Item' in col],axis=1,inplace=True)
test_df.drop([col for col in test_df.columns if 'Country' in col],axis=1,inplace=True)
test_df.head()
```

Out[59]:

	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
0	0.183443	0.110716	0.542078
1	0.458451	0.000413	0.627257
2	0.183443	0.106159	0.518228
3	1.000000	0.224154	0.890971
4	0.458451	0.000355	0.625213

In [60]:
```python
test_df['Country']=countries
test_df['Item']=items
test_df.head()
```

Out[60]:

	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country	Item
0	0.183443	0.110716	0.542078	Spain	Rice, paddy
1	0.458451	0.000413	0.627257	Madagascar	Wheat
2	0.183443	0.106159	0.518228	Spain	Sorghum
3	1.000000	0.224154	0.890971	Colombia	Potatoes
4	0.458451	0.000355	0.625213	Madagascar	Sweet potatoes

In [61]:
```python
clf=DecisionTreeRegressor()
model=clf.fit(train_data,train_labels)

test_df["yield_predicted"]= model.predict(test_data)
test_df["yield_actual"]=pd.DataFrame(test_labels)["hg/ha_yield"].tolist()
test_group=test_df.groupby("Item")
test_group.apply(lambda x: r2_score(x.yield_actual,x.yield_predicted))
```

Out[61]:
```
Item
Cassava                 0.927303
Maize                   0.888311
Plantains and others    0.786304
Potatoes                0.911489
Rice, paddy             0.896767
Sorghum                 0.800556
```

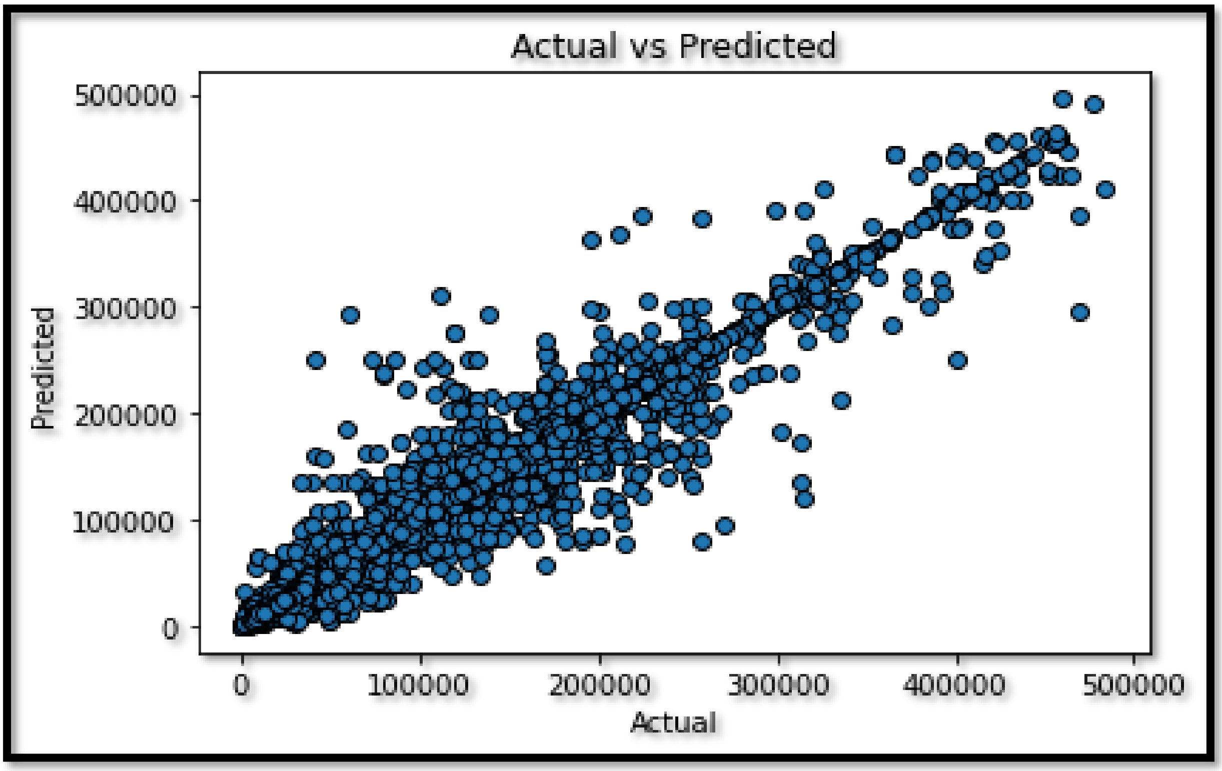

```
Cassava                 0.927303
Maize                   0.888311
Plantains and others    0.786304
Potatoes                0.911489
Rice, paddy             0.896767
Sorghum                 0.800556
```


```
Soybeans                0.855499
Sweet potatoes          0.847695
Wheat                   0.924474
Yams                    0.928255
dtype: float64
```

In [62]:
```python
# So let's run the model actual values against the predicted ones

fig, ax = plt.subplots()
```


In [62]:
```python
# So let's run the model actual values against the predicted ones

fig, ax = plt.subplots()
```


```python
ax.scatter(test_df["yield_actual"], test_df["yield_predicted"],edgecolors=(0, 0, 0))

ax.set_xlabel('Actual')
ax.set_ylabel('Predicted')
ax.set_title("Actual vs Predicted")
plt.show()
```




```
def adjusted_r_squared(y,yhat,x):
    score=1- (((1-(r2_score(y,yhat)))*(len(y)-1))/(len(y)-x.shape[1]-2))
    return score

test_group.apply(lambda x: adjusted_r_squared(x.yield_actual,x.yield_predicted,x))
```

```
Out[63]:
Item
Cassava               0.926359
Maize                 0.887589
Plantains and others  0.775275
Potatoes              0.910944
Rice, paddy           0.895909
Sorghum               0.798770
Soybeans              0.854327
Sweet potatoes        0.846276
Wheat                 0.923947
```

Chapter 5: Conclusion.

The crop yield prediction on the basic of the above models using decision and gradiant boosting regressor we have to get the analyzing conclusion of these dataset the crop being potatoes has the highest importance in the decision making for the model, where it's the highest crops in the dataset. Cassava too, then as expected we see the effect of pesticides, where it's the third most important feature, and then if the crop is sweet potatoes , we see some of the highest crops in features importance in dataset

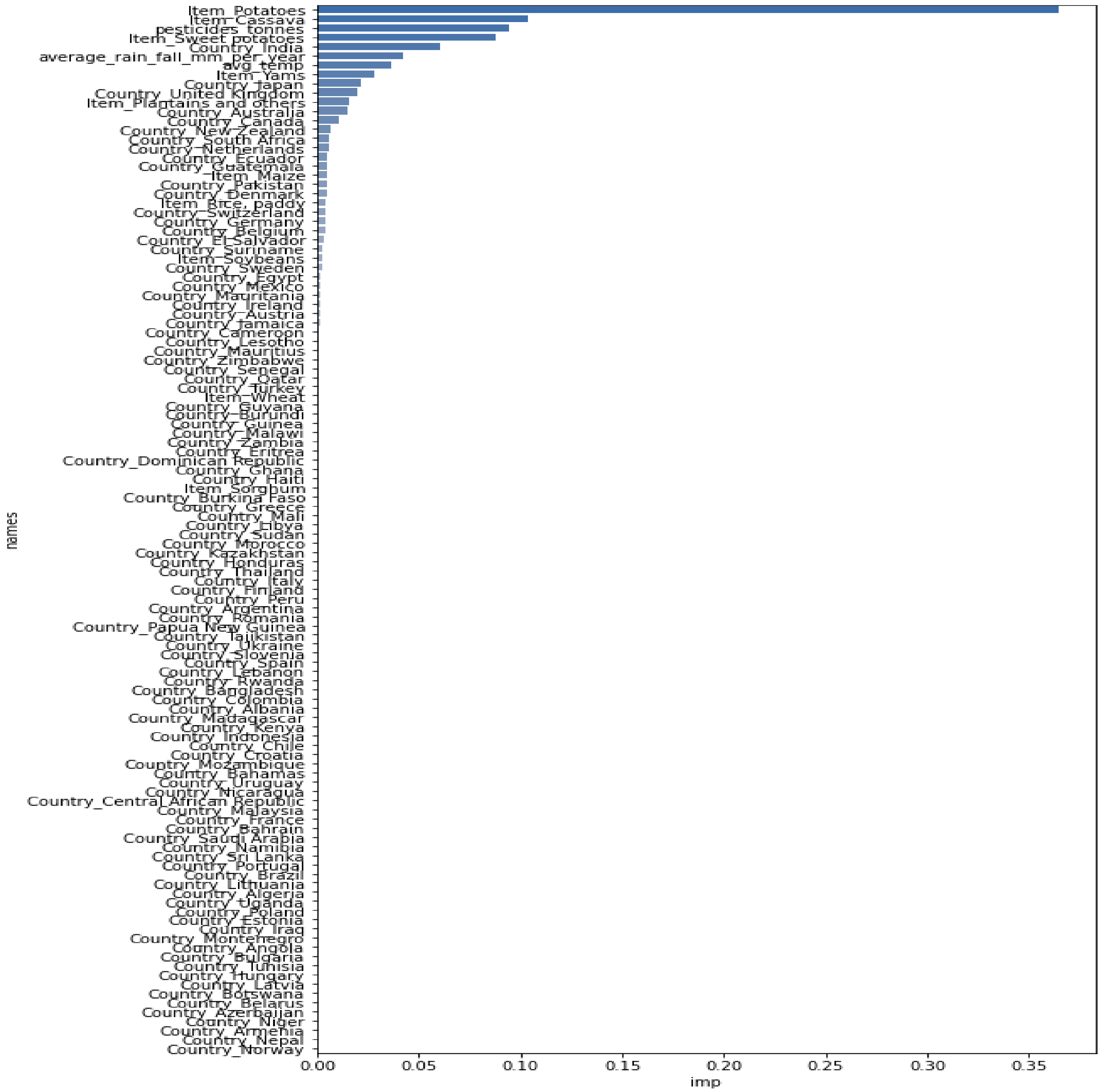
If the crops is grown in India, makes sense since indis has the largest crops sum in the dataset. Then comes rainfall and temperature. The first assumption about these features were correct, where they all significantly impact the expected crops yield in the model

In [64]:

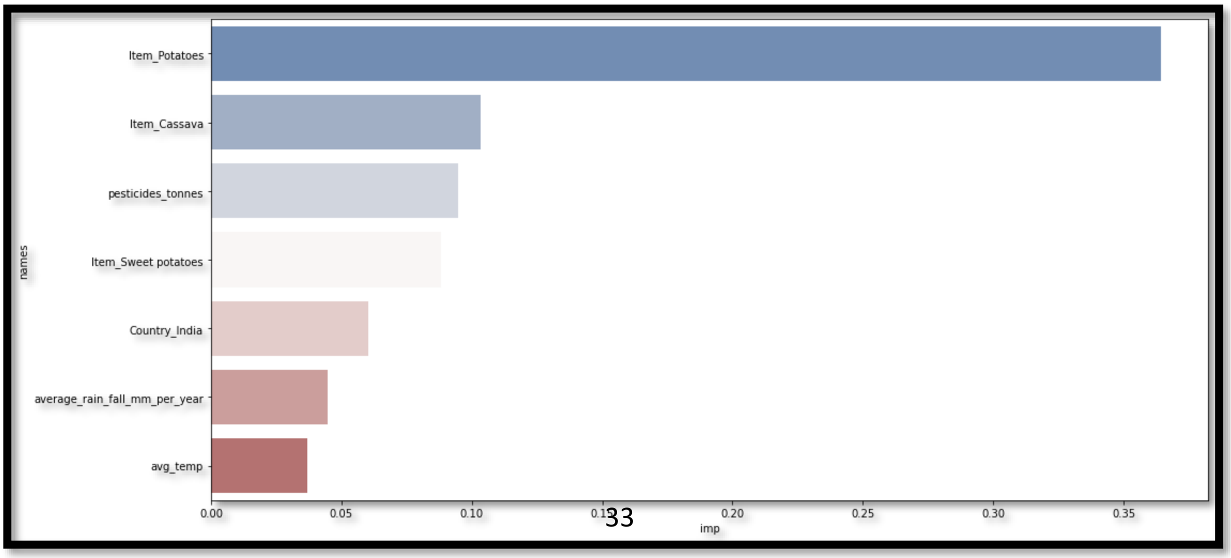
```
varimp= {'imp':model.feature_importances_,'names':yield_df_onehot.columns[yield_df_onehot.columns!
="hg/ha_yield"]}
```

In [65]:

```
a4_dims = (8.27,16.7)

fig, ax = plt.subplots(figsize=a4_dims)
df=pd.DataFrame.from_dict(varimp)
df.sort_values(ascending=False,by=["imp"],inplace=True)
df=df.dropna()
sns.barplot(x="imp",y="names",palette="vlag",data=df,orient="h",ax=ax);
```

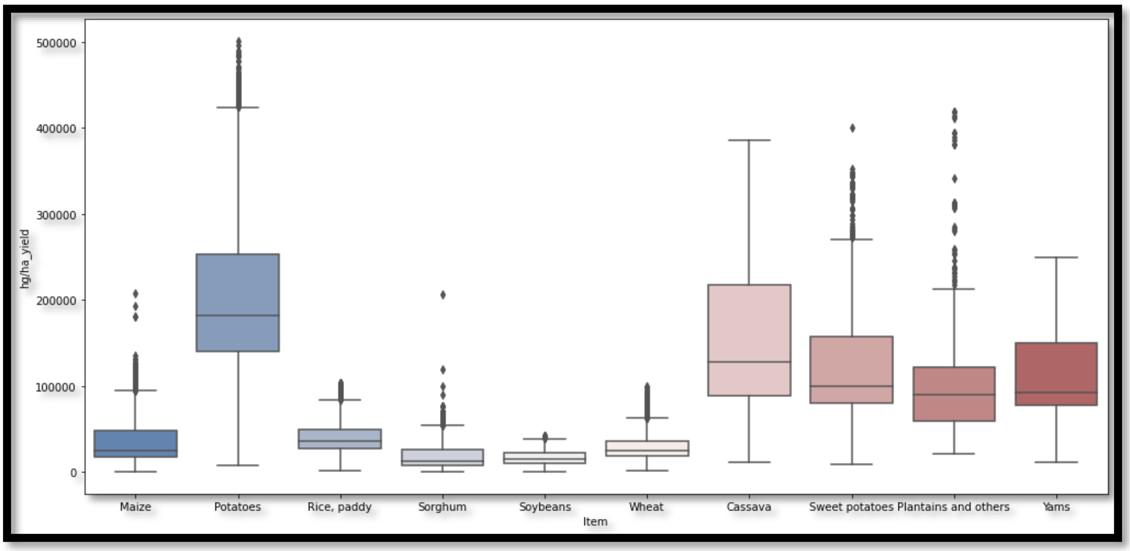


Getting only top 7 of feature importance in the model



In [67]:

```
#Boxplot that shows yield for each item
a4_dims = (16.7, 8.27)

fig, ax = plt.subplots(figsize=a4_dims)
sns.boxplot(x="Item",y="hg/ha_yield",palette="vlag",data=yield_df,ax=ax);
```



Chapter 6: Reference

1. Horie, T., Yajima, M., Nakagawa, H.: Yield forecasting. Agric. Syst. (1992). https://doi.org/10.1016/0308-521X(92)90022-G
2. Government of India: "Crop Forecasts". Ministry of Statistics & Programme Implementation (2020). http://mospi.nic.in/44-crop-forecasts
3. Ratkal, A.G., Akalwadi, G., Patil, V.N., Mahesh, K.: Farmer's analytical assistant. In: Proc. 2016 IEEE Int. Conf. Cloud Comput. Emerg. Mark, CCEM 2016, pp. 84–89 (2017). https://doi.org/10.1109/CCEM.2016.023
4. Kantanantha, N., Serban, N., Griffin, P.: Yield and price forecasting for stochastic crop decision planning. J. Agric. Biol. Environ. Stat. **15**(3), 362–380 (2010). https://doi.org/10.1007/s13253-010-0025-7
5. Gayatri, M.K., Jayasakthi, J., Mala, G.S.A.: Providing smart agricultural solutions to farmers for better yielding using IoT. In: Proc. 2015 IEEE Int. Conf. Technol. Innov. ICT Agric. Rural Dev. TIAR 2015, Tiar, pp. 40–43 (2015). https://doi.org/10.1109/TIAR.2015.7358528
6. Manjula, E., Djodiltachoumy, S.: A Model for prediction of crop yield. Int. J. Comput. Intell. Inform. **6**(4), 298–305 (2017)
7. Gandge, Y., Sandhya: A study on various data mining techniques for crop yield prediction. In: Int. Conf. Electr. Electron. Commun. Comput. Technol. Optim. Tech. ICEECCOT 2017, vol. 2018, pp. 420–423 (2018). https://doi.org/10.1109/ICEECCOT.2017.8284541
8. Ramesh, B.V., Vardhan, D.: Analysis of crop yield prediction using data mining techniques. Int. J. Res. Eng. Technol. **4**(1), 470–473 (2015). https://doi.org/10.23956/ijarcsse.v7i11.468
9. Kumar, R., Singh, M.P., Kumar, P., Singh, J.P.: Crop selection method to maximize crop yield rate using machine learning technique. In: 2015 Int. Conf. Smart Technol. Manag. Comput. Commun. Control. Energy Mater. ICSTM 2015 – Proc., May, pp. 138–145 (2015). https://doi.org/10.1109/ICSTM.2015.7225403
10. Medar, R., Rajpurohit, V.S., Shweta, S.: Crop yield prediction using machine learning techniques. In: 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), March, pp. 1–5 (2019). https://doi.org/10.1109/I2CT45611.2019.9033611