Data Preparation & Visualisation

Process of acquiring raw data

First of all I searched dataset related to agriculture. As my task is related to Ireland so I searched datasets on Ireland official website I found many datasets but I choose crop production.

I get two detests related to Ireland because both detests are same. But here problem is that these datasets cannot be used for country comparisons as it contains the data of only Ireland so I got another dataset which is crop production international dataset.

So there are three datasets which I got from internet sources

Positive and negative aspects of the research and acquisition

Positive aspects:

- We know about what is going what are trends in countries.
- We can analyse that a country on which position by exploring data.
- We can make good decisions

Negative aspects:

- It is very time consuming process to acquire datasets.
- data may be incomplete or inconsistent so we have to structure it .
- it can be used for wrong purposes.

Importing libraries:

The first step is to import all required libraries.so here I am importing four Major Libraries which are here:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
pd.options.mode.chained_assignment = None  # default='warn'
```

1)**pandas**:
pandas is used for data crunching.
2)**numpy** :
It is used for scientific computing it provides some mathematics and statistics like functions.
3)**Matplotlib:**
It is used for data Visualization For example to make Charts and graphs.
4)**Seaborn**:
It is also used for data Visualization For example to make Charts and graphs.

Importing first dataset

Now here I am loading data. As data is in excel file so I am importing Excel File with .csv file format. ***t***

```python
data1=pd.read_csv('AQA04.Crop Yield and Production.csv')
data1
```

	STATISTIC	Statistic	TLIST(A1)	Year	C02039V02469	Type of Crop	UNIT	VALUE
0	AQA04C1	Area under Crops	2008	2008	1	Total wheat, oats and barley	000 Hectares	320.7
1	AQA04C1	Area under Crops	2008	2008	11	Total wheat	000 Hectares	110.7
2	AQA04C1	Area under Crops	2008	2008	111	Winter wheat	000 Hectares	87.5
3	AQA04C1	Area under Crops	2008	2008	112	Spring wheat	000 Hectares	23.2
4	AQA04C1	Area under Crops	2008	2008	12	Total oats	000 Hectares	22.9
...	...	...	...	...	...	...	...	...
541	AQA04C3	Crop Production	2021	2021	131	Winter barley	000 Tonnes	638.8
542	AQA04C3	Crop Production	2021	2021	132	Spring barley	000 Tonnes	917.6
543	AQA04C3	Crop Production	2021	2021	2	Beans and peas	000 Tonnes	54.4
544	AQA04C3	Crop Production	2021	2021	3	Oilseed rape	000 Tonnes	52.2
545	AQA04C3	Crop Production	2021	2021	4	Potatoes	000 Tonnes	407.5

Importing second dataset

```
data2=pd.read_csv('AQA03_Crop Yield (1985 - 2007).csv')
data2
```

	STATISTIC	Statistic	TLIST(A1)	Year	C02039V02469	Type of Crop	UNIT	VALUE
0	AQA03C1	Area under Crops	1985	1985	1	Total wheat, oats and barley	000 Hectares	386.2
1	AQA03C1	Area under Crops	1985	1985	11	Total wheat	000 Hectares	77.7
2	AQA03C1	Area under Crops	1985	1985	111	Winter wheat	000 Hectares	60.3
3	AQA03C1	Area under Crops	1985	1985	112	Spring wheat	000 Hectares	17.4
4	AQA03C1	Area under Crops	1985	1985	12	Total oats	000 Hectares	24.9
...	...	...	...	...	...	...	...	...
1168	AQA03C3	Crop Yield	2007	2007	4	Potatoes	000 Tonnes	399.0
1169	AQA03C3	Crop Yield	2007	2007	5	Turnips	000 Tonnes	NaN
1170	AQA03C3	Crop Yield	2007	2007	61	Sugar beet	000 Tonnes	NaN
1171	AQA03C3	Crop Yield	2007	2007	62	Fodder beet	000 Tonnes	NaN
1172	AQA03C3	Crop Yield	2007	2007	7	Kale and field cabbage	000 Tonnes	NaN

EDA

Checking for any missing values

```
data1.isnull().sum()
```

```
STATISTIC       0
Statistic       0
TLIST(A1)       0
Year            0
C02039V02469    0
Type of Crop    0
UNIT            0
VALUE           0
dtype: int64
```

As shown above there are no null or missing values in data.

Description of data1

```
data1['VALUE'].describe()
```

```
count     546.000000
mean      214.459341
std       457.556839
min         2.500000
25%         8.400000
50%        23.750000
75%       165.550000
max      2633.600000
Name: VALUE, dtype: float64
```

As by looking on mean and median it is clear that there is huge difference is observed so this show that data is inconsistent.

So it is needed to structure data.

Exploring unit and Statistic feature

```
data1['UNIT'].unique()
```

```
array(['000 Hectares', 'Tonnes', '000 Tonnes'], dtype=object)
```

```
data1['Statistic'].unique()
```

```
array(['Area under Crops', 'Crop Yield per Hectare', 'Crop Production'],
      dtype=object)
```

As analysing above there are three units and three types of measurements. It is must that unit must be same. Here above the huge difference in mean and median is observed due to this reason.

But here problem is that every unit belongs to different quantity not same quantity.

Checking every unit based on every measure

```
data1[data1['Statistic']=='Area under Crops']['UNIT'].unique()
```

```
array(['000 Hectares'], dtype=object)
```

```
data1[data1['Statistic']=='Crop Yield per Hectare']['UNIT'].unique()
```

```
array(['Tonnes'], dtype=object)
```

```
data1[data1['Statistic']=='Crop Production']['UNIT'].unique()
```

```
array(['000 Tonnes'], dtype=object)
```

As looking above following is concluded:

Area under crop has unit of hectares

Crop yield per Hectare has unit of Tonnes

Crop production also has unit of Tonnes but there is difference of some characters

So three measures representing three units

Checking Length of every measure

```
len(data1[data1['Statistic']=='Area under Crops'])
```

```
182
```

```
len(data1[data1['Statistic']=='Crop Yield per Hectare'])
```

```
182
```

```
len(data1[data1['Statistic']=='Crop Production'])
```

```
182
```

As observing above it is cleared that every measure have 182 records

Exploring Year

```
data1['Year'].min()
```

```
2008
```

```
data1['Year'].max()
```

```
2021
```

```
data1['Year'].max()-data1['Year'].min()
```

```
13
```

```
data1['Year'].value_counts()
```

```
2008    39
2009    39
2010    39
2011    39
2012    39
2013    39
2014    39
2015    39
2016    39
2017    39
2018    39
2019    39
2020    39
2021    39
Name: Year, dtype: int64
```

This is agriculture record from 2008 to 2021

So this is record contains 13 years

Every year having same number of records as 39

Exploring type of crop feature

```
len(data1['Type of Crop'].unique())
```

```
13
```

```
data1['Type of Crop'].value_counts()
```

```
Total oats                      42
Oilseed rape                    42
Beans and peas                  42
Winter wheat                    42
Winter barley                   42
Total wheat                     42
Potatoes                        42
Total barley                    42
Total wheat, oats and barley    42
Spring wheat                    42
Winter oats                     42
Spring barley                   42
Spring oats                     42
Name: Type of Crop, dtype: int64
```

There are 13 types of crops are mentioned in dataset

Every crop having same number of records as 42

Exploring 2nd dataset

Checking for null values

```
data2.isnull().sum()
```

```
STATISTIC        0
Statistic        0
TLIST(A1)        0
Year             0
C02039V02469     0
Type of Crop     0
UNIT             0
VALUE           69
dtype: int64
```

There are 69 null values present in 2nd dataset

Description of data2

```
data2['VALUE'].describe()
```

```
count    1104.000000
mean      204.333062
std       411.534975
min         0.000000
25%         7.100000
50%        31.050000
75%       158.425000
max      2500.900000
Name: VALUE, dtype: float64
```

As by looking on mean and median it is clear that there is huge difference is observed so this show that data is inconsistent.

So it is needed to structure data.

Exploring unit and Statistic feature

```
data2['UNIT'].unique()
```

```
array(['000 Hectares', 'Tonnes', '000 Tonnes'], dtype=object)
```

```
data2['Statistic'].unique()
```

```
array(['Area under Crops', 'Crop Yield per Hectare', 'Crop Yield'],
      dtype=object)
```

As analysing above there are three units and three types of measurements. It is must that unit must be same. Here above the huge difference in mean and median is observed due to this reason.

But here problem is that every unit belongs to different quantity not same quantity.

Checking every unit based on every measure

```
data2[data2['Statistic']=='Area under Crops']['UNIT'].unique()
```

```
array(['000 Hectares'], dtype=object)
```

```
data2[data2['Statistic']=='Crop Yield per Hectare']['UNIT'].unique()
```

```
array(['Tonnes'], dtype=object)
```

```
data2[data2['Statistic']=='Crop Yield']['UNIT'].unique()
```

```
array(['000 Tonnes'], dtype=object)
```

As looking above following is concluded:

Area under crop has unit of hectares

Crop yield per Hectare has unit of Tonnes

Crop Yield also has unit of Tonnes but there is difference of some characters

So three measures representing three units

Exploring Year

```
data2['Year'].min()
```

```
1985
```

```
data2['Year'].max()
```

```
2007
```

```
data2['Year'].max()-data2['Year'].min()
```

```
22
```

```
data2['Year'].value_counts()
```

```
1985    51
1997    51
2006    51
2005    51
2004    51
2003    51
2002    51
2001    51
2000    51
1999    51
1998    51
1996    51
1986    51
1995    51
1994    51
1993    51
1992    51
1991    51
1990    51
1989    51
1988    51
1987    51
2007    51
```

This is agriculture record from 1985 to 2007

So this is record contains 22 years

Every year has 51 records.

Exploring type of crop feature

```
len(data2['Type of Crop'].unique())
```

```
17
```

```
data2['Type of Crop'].value_counts()
```

```
Total oats                        69
Spring barley                     69
Winter oats                       69
Sugar beet                        69
Turnips                           69
Winter wheat                      69
Oilseed rape                      69
Spring oats                       69
Spring wheat                      69
Beans and peas                    69
Total wheat, oats and barley      69
Total barley                      69
Potatoes                          69
Total wheat                       69
Winter barley                     69
Fodder beet                       69
Kale and field cabbage            69
```

There are 17 types of crops are mentioned in dataset

Every crop having same number of records as 69

Checking Length of every measure

```
len(data2[data2['Statistic']=='Area under Crops'])
```

```
391
```

```
len(data2[data2['Statistic']=='Crop Yield per Hectare'])
```

```
391
```

```
len(data2[data2['Statistic']=='Crop Yield'])
```

```
391
```

As observing above it is cleared that every measure have 391 records

Comparing 1st five records first dataset of every measure

```
data1[data1['Statistic']=='Area under Crops'].head()
```

	STATISTIC	Statistic	TLIST(A1)	Year	C02039V02469	Type of Crop	UNIT	VALUE
0	AQA04C1	Area under Crops	2008	2008	1	Total wheat, oats and barley	000 Hectares	320.7
1	AQA04C1	Area under Crops	2008	2008	11	Total wheat	000 Hectares	110.7
2	AQA04C1	Area under Crops	2008	2008	111	Winter wheat	000 Hectares	87.5
3	AQA04C1	Area under Crops	2008	2008	112	Spring wheat	000 Hectares	23.2
4	AQA04C1	Area under Crops	2008	2008	12	Total oats	000 Hectares	22.9

```
data1[data1['Statistic']=='Crop Yield per Hectare'].head()
```

	STATISTIC	Statistic	TLIST(A1)	Year	C02039V02469	Type of Crop	UNIT	VALUE
182	AQA04C2	Crop Yield per Hectare	2008	2008	1	Total wheat, oats and barley	Tonnes	7.7
183	AQA04C2	Crop Yield per Hectare	2008	2008	11	Total wheat	Tonnes	9.0
184	AQA04C2	Crop Yield per Hectare	2008	2008	111	Winter wheat	Tonnes	9.6
185	AQA04C2	Crop Yield per Hectare	2008	2008	112	Spring wheat	Tonnes	6.6
186	AQA04C2	Crop Yield per Hectare	2008	2008	12	Total oats	Tonnes	7.6

```
data1[data1['Statistic']=='Crop Production'].head()
```

	STATISTIC	Statistic	TLIST(A1)	Year	C02039V02469	Type of Crop	UNIT	VALUE
364	AQA04C3	Crop Production	2008	2008	1	Total wheat, oats and barley	000 Tonnes	2461.3
365	AQA04C3	Crop Production	2008	2008	11	Total wheat	000 Tonnes	992.8
366	AQA04C3	Crop Production	2008	2008	111	Winter wheat	000 Tonnes	839.9
367	AQA04C3	Crop Production	2008	2008	112	Spring wheat	000 Tonnes	153.0
368	AQA04C3	Crop Production	2008	2008	12	Total oats	000 Tonnes	174.3

As by comparing first five records of three measures it concluded that:

Features of year, Co2039VO2469 and type of crop are same for all three measures.

There is difference of value and unit in every measure

Comparing 1st five records first dataset of every measure

```
data2[data2['Statistic']=='Area under Crops'].head()
```

	STATISTIC	Statistic	TLIST(A1)	Year	C02039V02469	Type of Crop	UNIT	VALUE
0	AQA03C1	Area under Crops	1985	1985	1	Total wheat, oats and barley	000 Hectares	386.2
1	AQA03C1	Area under Crops	1985	1985	11	Total wheat	000 Hectares	77.7
2	AQA03C1	Area under Crops	1985	1985	111	Winter wheat	000 Hectares	60.3
3	AQA03C1	Area under Crops	1985	1985	112	Spring wheat	000 Hectares	17.4
4	AQA03C1	Area under Crops	1985	1985	12	Total oats	000 Hectares	24.9

```
data2[data2['Statistic']=='Crop Yield per Hectare'].head()
```

	STATISTIC	Statistic	TLIST(A1)	Year	C02039V02469	Type of Crop	UNIT	VALUE
391	AQA03C2	Crop Yield per Hectare	1985	1985	1	Total wheat, oats and barley	Tonnes	5.1
392	AQA03C2	Crop Yield per Hectare	1985	1985	11	Total wheat	Tonnes	6.4
393	AQA03C2	Crop Yield per Hectare	1985	1985	111	Winter wheat	Tonnes	6.5
394	AQA03C2	Crop Yield per Hectare	1985	1985	112	Spring wheat	Tonnes	5.8
395	AQA03C2	Crop Yield per Hectare	1985	1985	12	Total oats	Tonnes	5.1

```
data2[data2['Statistic']=='Crop Yield'].head()
```

	STATISTIC	Statistic	TLIST(A1)	Year	C02039V02469	Type of Crop	UNIT	VALUE
782	AQA03C3	Crop Yield	1985	1985	1	Total wheat, oats and barley	000 Tonnes	1986.0
783	AQA03C3	Crop Yield	1985	1985	11	Total wheat	000 Tonnes	495.0
784	AQA03C3	Crop Yield	1985	1985	111	Winter wheat	000 Tonnes	395.0
785	AQA03C3	Crop Yield	1985	1985	112	Spring wheat	000 Tonnes	100.0
786	AQA03C3	Crop Yield	1985	1985	12	Total oats	000 Tonnes	126.0

As by comparing first five records of three measures it concluded that:

Features of year, Co2039VO2469 and type of crop are same for all three measures.

There is difference of value and unit in every measure

Final conclusion of EDA

By exploring 1st and 2nd dataset it is concluded that:

Both datasets having same columns so we can combine both to make dataset efficient

Both datasets has three measures represented by three units respectively so this thing makes dataset inconsistent

Year and type of crop column is same for all measures

Every measure have same number of records

Feature Engineering

Combining two datasets by concatenation

As two datasets has same features so two datasets can be combined in such a way that rows of 2nd dataset are added where rows of 1st dataset terminating

```
data=pd.concat([data1,data2],axis=0).reset_index(drop=True)
data
```

	STATISTIC	Statistic	TLIST(A1)	Year	C02039V02469	Type of Crop	UNIT	VALUE
0	AQA04C1	Area under Crops	2008	2008	1	Total wheat, oats and barley	000 Hectares	320.7
1	AQA04C1	Area under Crops	2008	2008	11	Total wheat	000 Hectares	110.7
2	AQA04C1	Area under Crops	2008	2008	111	Winter wheat	000 Hectares	87.5
3	AQA04C1	Area under Crops	2008	2008	112	Spring wheat	000 Hectares	23.2
4	AQA04C1	Area under Crops	2008	2008	12	Total oats	000 Hectares	22.9
...	...	...	...	...	...	...	...	...
1714	AQA03C3	Crop Yield	2007	2007	4	Potatoes	000 Tonnes	399.0
1715	AQA03C3	Crop Yield	2007	2007	5	Turnips	000 Tonnes	NaN
1716	AQA03C3	Crop Yield	2007	2007	61	Sugar beet	000 Tonnes	NaN
1717	AQA03C3	Crop Yield	2007	2007	62	Fodder beet	000 Tonnes	NaN
1718	AQA03C3	Crop Yield	2007	2007	7	Kale and field cabbage	000 Tonnes	NaN

```
data.shape
```

```
(1719, 8)
```

So after concatenation the total records in concatenated data is 1719

Separating data related to measurement of area

```
Area=data[data['Statistic']=='Area under Crops']
Area
```

	STATISTIC	Statistic	TLIST(A1)	Year	C02039V02469	Type of Crop	UNIT	VALUE
0	AQA04C1	Area under Crops	2008	2008	1	Total wheat, oats and barley	000 Hectares	320.7
1	AQA04C1	Area under Crops	2008	2008	11	Total wheat	000 Hectares	110.7
2	AQA04C1	Area under Crops	2008	2008	111	Winter wheat	000 Hectares	87.5
3	AQA04C1	Area under Crops	2008	2008	112	Spring wheat	000 Hectares	23.2
4	AQA04C1	Area under Crops	2008	2008	12	Total oats	000 Hectares	22.9
...	...	...	...	...	...	...	...	...
932	AQA03C1	Area under Crops	2007	2007	4	Potatoes	000 Hectares	11.7
933	AQA03C1	Area under Crops	2007	2007	5	Turnips	000 Hectares	NaN
934	AQA03C1	Area under Crops	2007	2007	61	Sugar beet	000 Hectares	NaN
935	AQA03C1	Area under Crops	2007	2007	62	Fodder beet	000 Hectares	NaN
936	AQA03C1	Area under Crops	2007	2007	7	Kale and field cabbage	000 Hectares	NaN

573 rows × 8 columns

Separating data related to measurement of crop yield per Hectare

```
Crop_Yield_per_Hectare=data[data['Statistic']=='Crop Yield per Hectare']
Crop_Yield_per_Hectare
```

	STATISTIC	Statistic	TLIST(A1)	Year	C02039V02469	Type of Crop	UNIT	VALUE
182	AQA04C2	Crop Yield per Hectare	2008	2008	1	Total wheat, oats and barley	Tonnes	7.7
183	AQA04C2	Crop Yield per Hectare	2008	2008	11	Total wheat	Tonnes	9.0
184	AQA04C2	Crop Yield per Hectare	2008	2008	111	Winter wheat	Tonnes	9.6
185	AQA04C2	Crop Yield per Hectare	2008	2008	112	Spring wheat	Tonnes	6.6
186	AQA04C2	Crop Yield per Hectare	2008	2008	12	Total oats	Tonnes	7.6
...	...	...	...	...	...	...	...	...
1323	AQA03C2	Crop Yield per Hectare	2007	2007	4	Potatoes	Tonnes	34.0
1324	AQA03C2	Crop Yield per Hectare	2007	2007	5	Turnips	Tonnes	NaN
1325	AQA03C2	Crop Yield per Hectare	2007	2007	61	Sugar beet	Tonnes	NaN
1326	AQA03C2	Crop Yield per Hectare	2007	2007	62	Fodder beet	Tonnes	NaN
1327	AQA03C2	Crop Yield per Hectare	2007	2007	7	Kale and field cabbage	Tonnes	NaN

Replacing crop yield with crop production

```
data['Statistic'].unique()
```

```
array(['Area under Crops', 'Crop Yield per Hectare', 'Crop Production',
       'Crop Yield'], dtype=object)
```

```
data['Statistic']=data['Statistic'].replace(['Crop Yield'],['Crop Production'])
```

As in resultant data there are four measurements but Crop production and Crop Yield is same.

So crop yield is replaced by crop production

Separating data related to measurement of production

```
crop_production=data[data['Statistic']=='Crop Production']
crop_production
```

	STATISTIC	Statistic	TLIST(A1)	Year	C02039V02469	Type of Crop	UNIT	VALUE
364	AQA04C3	Crop Production	2008	2008	1	Total wheat, oats and barley	000 Tonnes	2461.3
365	AQA04C3	Crop Production	2008	2008	11	Total wheat	000 Tonnes	992.8
366	AQA04C3	Crop Production	2008	2008	111	Winter wheat	000 Tonnes	839.9
367	AQA04C3	Crop Production	2008	2008	112	Spring wheat	000 Tonnes	153.0
368	AQA04C3	Crop Production	2008	2008	12	Total oats	000 Tonnes	174.3
...	...	...	...	...	...	...	...	...
1714	AQA03C3	Crop Production	2007	2007	4	Potatoes	000 Tonnes	399.0
1715	AQA03C3	Crop Production	2007	2007	5	Turnips	000 Tonnes	NaN
1716	AQA03C3	Crop Production	2007	2007	61	Sugar beet	000 Tonnes	NaN
1717	AQA03C3	Crop Production	2007	2007	62	Fodder beet	000 Tonnes	NaN
1718	AQA03C3	Crop Production	2007	2007	7	Kale and field cabbage	000 Tonnes	NaN

Preparing data frame with year and type of crop feature

As year and type of crop is same for all measurements and number of records in all measurements are same three columns are derived from area

```
df=Area[['Year','C02039V02469','Type of Crop']]
df
```

	Year	C02039V02469	Type of Crop
0	2008	1	Total wheat, oats and barley
1	2008	11	Total wheat
2	2008	111	Winter wheat
3	2008	112	Spring wheat
4	2008	12	Total oats
...	...	...	...
932	2007	4	Potatoes
933	2007	5	Turnips
934	2007	61	Sugar beet
935	2007	62	Fodder beet
936	2007	7	Kale and field cabbage

Adding three new columns to data frame

```
df['Area_under_Crops_Hectares']=list(Area['VALUE'])
df['Crop_Yield_per_Hectare_in_Tonnes']=list(Crop_Yield_per_Hectare['VALUE'])
df['crop_production_in_Tonnes']=list(crop_production['VALUE'])
```

```
df
```

	Year	C02039V02469	Type of Crop	Area_under_Crops_Hectares	Crop_Yield_per_Hectare_in_Tonnes	crop_production_in_Tonnes
0	2008	1	Total wheat, oats and barley	320.7	7.7	2461.3
1	2008	11	Total wheat	110.7	9.0	992.8
2	2008	111	Winter wheat	87.5	9.6	839.9
3	2008	112	Spring wheat	23.2	6.6	153.0
4	2008	12	Total oats	22.9	7.6	174.3
...	...	...	...	...	...	...
932	2007	4	Potatoes	11.7	34.0	399.0
933	2007	5	Turnips	NaN	NaN	NaN
934	2007	61	Sugar beet	NaN	NaN	NaN
935	2007	62	Fodder beet	NaN	NaN	NaN
936	2007	7	Kale and field cabbage	NaN	NaN	NaN

As three new columns are added which are derived from area under crop, crop yield per hectare and crop production

Checking missing values

```
df.isnull().sum()
```

```
Year                                 0
C02039V02469                         0
Type of Crop                         0
Area_under_Crops_Hectares           23
Crop_Yield_per_Hectare_in_Tonnes    23
crop_production_in_Tonnes           23
dtype: int64
```

In three newly added columns total 69 null values are found

Removing null values

```
df=df.dropna()
df.shape
```

```
(550, 6)
```

As null values are removed so remaining rows are 550

Saving dataset

As now dataset is structured and cleaned it can be used for analysis and machine learning

```
df.to_csv('crop_production_modified.csv', index=None)
```

One hot encoding

One hot encoding is used to convent categorical feature into numerical feature.

```
df=pd.get_dummies(df, columns = ["Type of Crop"],drop_first=True, prefix="crop_")
```

```
df.shape
```

```
(550, 21)
```

```
df.columns
```

```
Index(['Year', 'C02039V02469', 'Area_under_Crops_Hectares',
       'Crop_Yield_per_Hectare_in_Tonnes', 'crop_production_in_Tonnes',
       'crop__Fodder beet', 'crop__Kale and field cabbage',
       'crop__Oilseed rape', 'crop__Potatoes', 'crop__Spring barley',
       'crop__Spring oats', 'crop__Spring wheat', 'crop__Sugar beet',
       'crop__Total barley', 'crop__Total oats', 'crop__Total wheat',
       'crop__Total wheat, oats and barley', 'crop__Turnips',
       'crop__Winter barley', 'crop__Winter oats', 'crop__Winter wheat'],
      dtype='object')
```

As type of crop is a categorical feature so it is converted into numerical to apply machine learning on it.

After applying one hot encoding there are 21 features total.

Logical justification based on the reasoning for the specific choice of machine learning approaches:

Why Supervised Machine Learning is used rather than Unsupervised Machine Learning?

Here Supervised Machine Learning is used because with the help of supervised learning, the model can predict the output on the basis of training data.

Unsupervised Machine Learning is used to view patterns or clusters

As here the task is to prepare a Machine Learning Model which predicts the value of crop production so Supervised Machine Learning Techniques are solution here.

Why Regression is used here rather than classification?

Regression is used where dependant feature or target feature is continues and classification is used Where dependant feature or target feature is categorical

As here target variable crop production contains continues values so regression is used here.

Import dataset for Machine Learning

Here below structured dataset is imported for building machine learning models.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
pd.options.mode.chained_assignment = None  # default='warn'
```

```
df=pd.read_csv('crop_production_predictionML.csv')
df
```

	Year	C02039V02469	Area_under_Crops_Hectares	Crop_Yield_per_Hectare_in_Tonnes	crop_production_in_Tonnes	crop__Fodder beet	cro
0	2008	1	320.7	7.7	2461.3	0	
1	2008	11	110.7	9.0	992.8	0	
2	2008	111	87.5	9.6	839.9	0	
3	2008	112	23.2	6.6	153.0	0	
4	2008	12	22.9	7.6	174.3	0	
...	...	...	...	...	...	...	
545	2007	131	18.7	7.6	142.4	0	

Splitting dataset into dependant and independent

Here data is divided into dependent and Independent features. Dependant features are inputs for training process and Independent features is output for machine learning process.

```
X=df.drop(['crop_production_in_Tonnes'],axis=1)
X.shape
```

```
(550, 20)
```

```
y=df['crop_production_in_Tonnes']
```

As crop production is targeted variable so it is removed from inputs and added on outputs

Feature Scaling

Standardization is scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.

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

```
array([[ 0.58211472, -1.00907364,  3.02939687, ..., -0.26856053,
        -0.26856053, -0.26856053],
       [ 0.58211472, -0.82827142,  0.55963168, ..., -0.26856053,
        -0.26856053, -0.26856053],
       [ 0.58211472,  0.97975081,  0.28678143, ..., -0.26856053,
        -0.26856053,  3.72355541],
       ...,
       [ 0.48822525, -0.99099342, -0.70818111, ..., -0.26856053,
        -0.26856053, -0.26856053],
       [ 0.48822525, -0.9729132 , -0.64584894, ..., -0.26856053,
        -0.26856053, -0.26856053],
       [ 0.48822525, -0.95483297, -0.60468619, ..., -0.26856053,
        -0.26856053, -0.26856053]])
```

Dimensional reduction:

Dimensionality reduction refers to techniques for reducing the number of input variables in training data. When dealing with high dimensional data, it is often useful to reduce the dimensionality by reducing features.

It has two main techniques:

● Principal Component Analysis (PCA)

● Linear Discriminant Analysis (LDA)

Principal Component Analysis (PCA)

Principal Component Analysis, or PCA, is a dimensionality-reduction method to find lower-dimensional space by preserving the variance as measured in the high dimensional input space. It is an unsupervised method for dimensionality reduction.

```
from sklearn.decomposition import PCA

pca = PCA(0.95)
X_pca = pca.fit_transform(X_scaled)
X_pca.shape
```

```
(550, 16)
```

```
pca.explained_variance_ratio_
```

```
array([0.1175678 , 0.11047456, 0.08033868, 0.05692333, 0.05360624,
       0.05360624, 0.05360624, 0.05360624, 0.05360624, 0.05360624,
       0.05360624, 0.05360624, 0.05198806, 0.05192557, 0.05157712,
       0.04107943])
```

Here above number of features are reduced to 16 .which makes 95 % importance to Model.

Train test splitting

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.

```
from sklearn.model_selection import train_test_split
X_train_pca, X_test_pca, y_train, y_test = train_test_split(X_pca, y, test_size=0.2, random_state=30)
```

Here above size of test data is 20% so size of training dataset is 80%.

Model building

This is Machine Learning Model building stage in which threetypes of regression algorithims are used along with parameters.

In this stage training data is given to model so in this stage classification models learns from training data.

```
from sklearn.linear_model import LinearRegression
LR= LinearRegression()
LR.fit(X_train_pca, y_train)
```

```
LinearRegression()
```

```
from sklearn.tree import DecisionTreeRegressor
DTR= DecisionTreeRegressor()
DTR.fit(X_train_pca, y_train)
```

```
DecisionTreeRegressor()
```

```
from sklearn.ensemble import RandomForestRegressor
RFR= RandomForestRegressor()
RFR.fit(X_train_pca, y_train)
```

```
RandomForestRegressor()
```

Comparison between the chosen modelling approaches:

Here below performance of regression models are compared using Accuracy Score.

Accuracy of classification model:

```
LR.score(X_test_pca, y_test)
```

```
0.9501566595154014
```

```
DTR.score(X_test_pca, y_test)
```

```
0.9571615588315115
```

```
RFR.score(X_test_pca, y_test)
```

```
0.974447322614494
```

As observed accuracy of linear regression model is 95%,

Accuracy of decision Tree Regression is 95% and accuracy of Random Forest Regression is 97%

So Random Forest Regression is best performing model here

But it is not finalized that this is best performing model

So other Evaluation techniques are also used.

K Fold Cross validation

Here K Fold Cross validation is performed on Machine Learning Models to to find which model is performing best by giving highest accuracy.

```
from sklearn.model_selection import cross_val_score
model_scoring={}
def all_model_scores(model,X,y):
    scores=cross_val_score(model,X,y,cv=10)
    mean_score=scores.mean()
    model_scoring.update({model:mean_score})
    return model_scoring

Model_list=[LinearRegression(),DecisionTreeRegressor(),RandomForestRegressor()]
for model in Model_list:
    score_dict=all_model_scores(model,X_pca,y)
df_Models_scores=pd.DataFrame(score_dict,index=[0])
df_Models_scores
```

	LinearRegression()	DecisionTreeRegressor()	RandomForestRegressor()
0	0.934236	0.966176	0.976309

Result:

Random Forest Regression is regression model here with highest score of accuracy so Random Forest Regression is best performing model

Evaluation techniques:

R2 score

The R2 score is a very important metric that is used to evaluate the performance of a regression-based machine learning model. It is pronounced as R squared and is also known as the coefficient of determination. It works by measuring the amount of variance in the predictions explained by the dataset

Mean Absolute Error

Mean absolute error refers to **the magnitude of difference between the prediction of an observation and the true value of that observation**. MAE takes the average of absolute errors for a group of predictions and observations as a measurement of the magnitude of errors for the entire group.

Root Mean Square Error:

Root mean squared error (**RMSE**) is the square root of the mean of the square of all of the **error**. The use of **RMSE** is very common

Table of accuracy scores, Mean Absolute Error and Root Mean Square error:

```
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error

Models_names=['Linear Regresser','Decision Tree Regressor','Random Forest Regressor']
Model_list=[LinearRegression(),DecisionTreeRegressor(),RandomForestRegressor()]
Accuracy_Scores=[]
MAE_Scores=[]
RMSE_Scores=[]

for model in Model_list:
    model.fit(X_train_pca,y_train)
    y_pred=model.predict(X_test_pca)
    Accuracy_Scores.append(model.score(X_test_pca, y_test))
    MAE_Scores.append(mean_absolute_error(y_test,y_pred))
    RMSE_Scores.append(np.sqrt(mean_squared_error(y_test,y_pred)))

Table=pd.DataFrame(list(zip(Models_names,Accuracy_Scores,MAE_Scores,RMSE_Scores)),columns =['Machine Learning Models','Accuracy S
```

```
Table
```

	Machine Learning Models	Accuracy Scores	Mean_Absolute_Error	root_mean_squared_error
0	Linear Regresser	0.950157	110.016822	148.049836
1	Decision Tree Regressor	0.962281	74.646364	128.790875
2	Random Forest Regressor	0.974919	64.697618	105.020630

This table shows similarities and difference of Accuracy Scores, Mean Absolute Error and Root Mean squared error.

So random Forest Regression is best performing machine learning model as it gives largest accuracy scores but gives smallest Mean Absolute Error and Root Mean Square Error .

Importing dataset for stats:

Here a dataset is loaded for statistical analysis and comparisons. It is international data because it is used to compare the results of Ireland with rest of countries in world.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

```
df=pd.read_csv('crop_production.csv')
df
```

	LOCATION	INDICATOR	SUBJECT	MEASURE	FREQUENCY	TIME	Value	Flag Codes
0	AUS	CROPYIELD	RICE	TONNE_HA	A	1990	8.314607	NaN
1	AUS	CROPYIELD	RICE	TONNE_HA	A	1991	8.394737	NaN
2	AUS	CROPYIELD	RICE	TONNE_HA	A	1992	8.094340	NaN
3	AUS	CROPYIELD	RICE	TONNE_HA	A	1993	8.336000	NaN
4	AUS	CROPYIELD	RICE	TONNE_HA	A	1994	8.537815	NaN
...	...	...	...	...	...	...	...	...
20561	OECD	CROPYIELD	SOYBEAN	THND_HA	A	2021	37010.208830	NaN
20562	OECD	CROPYIELD	SOYBEAN	THND_HA	A	2022	37069.214850	NaN

Importing Ireland dataset

This is structured dataset of Ireland so these two datasets are imported to do statistical analysis

```
df2=pd.read_csv('crop_production_modified.csv')
df2.head()
```

	Year	C02039V02469	Type of Crop	Area_under_Crops_Hectares	Crop_Yield_per_Hectare_in_Tonnes	crop_production_in_Tonnes
0	2008	1	Total wheat, oats and barley	320.7	7.7	2461.3
1	2008	11	Total wheat	110.7	9.0	992.8
2	2008	111	Winter wheat	87.5	9.6	839.9
3	2008	112	Spring wheat	23.2	6.6	153.0
4	2008	12	Total oats	22.9	7.6	174.3

Overview of features:

This is mean, median and mode value of features

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20566 entries, 0 to 20565
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   LOCATION    20566 non-null  object
 1   INDICATOR   20566 non-null  object
 2   SUBJECT     20566 non-null  object
 3   MEASURE     20566 non-null  object
 4   FREQUENCY   20566 non-null  object
 5   TIME        20566 non-null  int64
 6   Value       20566 non-null  float64
 7   Flag Codes  0 non-null      float64
dtypes: float64(2), int64(1), object(5)
memory usage: 1.3+ MB
```

Removing FLAG Codes feature

Flag Codes feature is removed as it contains all null values

```
df=df.drop(['Flag Codes'],axis=1)
df.shape
```

```
(20566, 7)
```

As there are 20566 records and 7 features

Exploring Measure

```
df['MEASURE'].unique()
```

```
array(['TONNE_HA', 'THND_TONNE', 'THND_HA'], dtype=object)
```

```
df['INDICATOR'].unique()
```

```
array(['CROPYIELD'], dtype=object)
```

There are three types of measurements are here:

- Tonne per hectare :this is unit of crop production per area
- Thousand Tonne: this is unit of Crop production
- Thousand Hectare: This is unit of Land area
  So this data representing three different types of quantities

Due to which this data is inconsistent so we have to make it structured

Exploring Measure Deeply

```
df[df['MEASURE']=='TONNE_HA']
```

	LOCATION	INDICATOR	SUBJECT	MEASURE	FREQUENCY	TIME	Value
0	AUS	CROPYIELD	RICE	TONNE_HA	A	1990	8.314607
1	AUS	CROPYIELD	RICE	TONNE_HA	A	1991	8.394737
2	AUS	CROPYIELD	RICE	TONNE_HA	A	1992	8.094340
3	AUS	CROPYIELD	RICE	TONNE_HA	A	1993	8.336000
4	AUS	CROPYIELD	RICE	TONNE_HA	A	1994	8.537815
...	...	...	...	...	...	...	...
20497	OECD	CROPYIELD	SOYBEAN	TONNE_HA	A	2021	3.254959
20498	OECD	CROPYIELD	SOYBEAN	TONNE_HA	A	2022	3.291244
20499	OECD	CROPYIELD	SOYBEAN	TONNE_HA	A	2023	3.323189
20500	OECD	CROPYIELD	SOYBEAN	TONNE_HA	A	2024	3.350868
20501	OECD	CROPYIELD	SOYBEAN	TONNE_HA	A	2025	3.378216

6826 rows × 7 columns

```
df[df['MEASURE']=='THND_TONNE']
```

	LOCATION	INDICATOR	SUBJECT	MEASURE	FREQUENCY	TIME	Value
3284	WLD	CROPYIELD	RICE	THND_TONNE	A	1994	361820.7121
3285	WLD	CROPYIELD	RICE	THND_TONNE	A	1995	369091.1804
3286	WLD	CROPYIELD	RICE	THND_TONNE	A	1996	384153.4470
3287	WLD	CROPYIELD	RICE	THND_TONNE	A	1997	387716.7782
3288	WLD	CROPYIELD	RICE	THND_TONNE	A	1998	389939.8791
...	...	...	...	...	...	...	...
20434	OECD	CROPYIELD	SOYBEAN	THND_TONNE	A	2021	120466.7257
20435	OECD	CROPYIELD	SOYBEAN	THND_TONNE	A	2022	122003.8408
20436	OECD	CROPYIELD	SOYBEAN	THND_TONNE	A	2023	123434.7370
20437	OECD	CROPYIELD	SOYBEAN	THND_TONNE	A	2024	124027.8756
20438	OECD	CROPYIELD	SOYBEAN	THND_TONNE	A	2025	125133.8397

6877 rows × 7 columns

```
df[df['MEASURE']=='THND_HA']
```

	LOCATION	INDICATOR	SUBJECT	MEASURE	FREQUENCY	TIME	Value
3316	IRN	CROPYIELD	WHEAT	THND_HA	A	1990	6278.000260
3317	IRN	CROPYIELD	WHEAT	THND_HA	A	1991	6557.999758
3318	IRN	CROPYIELD	WHEAT	THND_HA	A	1992	6929.999940
3319	IRN	CROPYIELD	WHEAT	THND_HA	A	1993	7189.999794
3320	IRN	CROPYIELD	WHEAT	THND_HA	A	1994	6782.000195
...	...	...	...	...	...	...	...
20561	OECD	CROPYIELD	SOYBEAN	THND_HA	A	2021	37010.208830
20562	OECD	CROPYIELD	SOYBEAN	THND_HA	A	2022	37069.214850
20563	OECD	CROPYIELD	SOYBEAN	THND_HA	A	2023	37143.459750
20564	OECD	CROPYIELD	SOYBEAN	THND_HA	A	2024	37013.651900
20565	OECD	CROPYIELD	SOYBEAN	THND_HA	A	2025	37041.401580

6863 rows × 7 columns

As by looking above three quantities it is clear that there is huge difference between Values of three quantities

Finding maximum of Tonne per hectare

```
df[df['MEASURE']=='TONNE_HA']['Value'].max()
```

```
15.0
```

Making a new data frame

Here a new dataset is developed by using measurement of Tonne per hectare so it can be compared with crop production per hectare feature of Ireland dataset

```python
df_TONNE_HA=df[df['MEASURE']=='TONNE_HA']
df_TONNE_HA
```

	LOCATION	INDICATOR	SUBJECT	MEASURE	FREQUENCY	TIME	Value
0	AUS	CROPYIELD	RICE	TONNE_HA	A	1990	8.314607
1	AUS	CROPYIELD	RICE	TONNE_HA	A	1991	8.394737
2	AUS	CROPYIELD	RICE	TONNE_HA	A	1992	8.094340
3	AUS	CROPYIELD	RICE	TONNE_HA	A	1993	8.336000
4	AUS	CROPYIELD	RICE	TONNE_HA	A	1994	8.537815
...	...	...	...	...	...	...	...
20497	OECD	CROPYIELD	SOYBEAN	TONNE_HA	A	2021	3.254959
20498	OECD	CROPYIELD	SOYBEAN	TONNE_HA	A	2022	3.291244
20499	OECD	CROPYIELD	SOYBEAN	TONNE_HA	A	2023	3.323189
20500	OECD	CROPYIELD	SOYBEAN	TONNE_HA	A	2024	3.350868
20501	OECD	CROPYIELD	SOYBEAN	TONNE_HA	A	2025	3.378216

Exploring Time feature

```python
df_TONNE_HA['TIME'].unique()
```

```
array([1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000,
       2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011,
       2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022,
       2023, 2024, 2025], dtype=int64)
```

```python
df_TONNE_HA=df_TONNE_HA[df_TONNE_HA['TIME']<=2021]
df_TONNE_HA['TIME'].unique()
```

```
array([1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000,
       2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011,
       2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021],
      dtype=int64)
```

As here a problem is detected that year data shows years of 2025, 2024, 2023. These are feature years so these years are eliminated from original dataset.

Exploring Location

```
df_TONNE_HA['LOCATION'].unique()
```

```
array(['AUS', 'CAN', 'JPN', 'KOR', 'MEX', 'NZL', 'TUR', 'USA', 'DZA',
       'ARG', 'BGD', 'BRA', 'CHL', 'CHN', 'COL', 'EGY', 'ETH', 'GHA',
       'IND', 'IDN', 'IRN', 'KAZ', 'MYS', 'MOZ', 'NGA', 'PAK', 'PRY',
       'PER', 'PHL', 'RUS', 'SAU', 'ZAF', 'SDN', 'TZA', 'THA', 'UKR',
       'URY', 'VNM', 'ZMB', 'WLD', 'SSA', 'OECD', 'BRICS', 'NOR', 'CHE',
       'EU28', 'ISR', 'HTI'], dtype=object)
```

```
len(df_TONNE_HA['LOCATION'].unique())
```

```
48
```

Here location shows names of countries in world so there are 48 countries In data

Showing Mean values of crop production of every country

```
df_TONNE_HA.groupby('LOCATION')['Value'].mean().sort_values(ascending=False)
```

```
LOCATION
EGY      5.666778
EU28     5.284387
USA      4.991947
CHL      4.879347
OECD     4.811134
AUS      4.565987
NZL      4.515404
CHE      4.501183
TUR      3.961190
CHN      3.955179
ARG      3.640061
KOR      3.637315
URY      3.449577
CAN      3.370863
WLD      3.298083
MEX      3.205045
JPN      3.172529
BRICS    3.094354
```

Here above are mean values of crop production of countries

Mean Value on the basis of types of crops:

```
df_TONNE_HA.groupby('SUBJECT')['Value'].mean().sort_values(ascending=False)
```

```
SUBJECT
MAIZE      4.166053
RICE       2.638179
WHEAT      2.534300
SOYBEAN    1.573560
Name: Value, dtype: float64
```

Mean of crop production value in Ireland :

```
df2['Crop_Yield_per_Hectare_in_Tonnes'].mean()
```

```
14.127272727272727
```

Here 14.12 Tonnes per hectare is Average Value of crop production per hectare

Comparison of Ireland with other:

As looking above Ireland average crop production is 14 tonnes per hectare it is highest value as comparing with other countries has less mean value of crop production

Descriptive stats

```
df2['Crop_Yield_per_Hectare_in_Tonnes'].max()
```

```
73.7
```

```
df2['Crop_Yield_per_Hectare_in_Tonnes'].min()
```

```
0.0
```

```
df2['Crop_Yield_per_Hectare_in_Tonnes'].std()
```

```
16.21486033381029
```

```
df2['Crop_Yield_per_Hectare_in_Tonnes'].describe()
```

```
count    550.000000
mean      14.127273
std       16.214860
min        0.000000
25%        6.100000
50%        7.600000
75%        9.100000
max       73.700000
```

Here crop yield of Ireland is explored as this data has huge variance is observed between mean and median

This show that this data is very skewed and not normally distributed

An huge evidence of greater variance is that seventy five percentile is 9 and maximum (100 percentile) is 73 so this shows that 73 is outlier

```
df2['Crop_Yield_per_Hectare_in_Tonnes'].quantile(0.80)
```

```
10.520000000000005
```

```
df2['Crop_Yield_per_Hectare_in_Tonnes'].quantile(0.05)
```

```
3.5450000000000004
```

80 percentile is 10

Inferential statistics

Inferential statistics use measurements from the sample of subjects in the experiment to compare the treatment groups and make generalizations about the larger population of subjects.

Why inferential statistics is used?

While descriptive statistics summarize the characteristics of a data set, inferential statistics help our come to conclusions and make predictions based on your data. When we have collected data from a sample, we can use inferential statistics to understand the larger population from which the sample is taken.

Hypothesis testing:

Hypothesis testing is a form of statistical inference that uses data from a sample to draw conclusions about a population parameter or a population probability distribution.

Chi-square test

A chi-square test is a statistical test used to compare observed results with expected results. The purpose of this test is to determine if a difference between observed data and expected data is due to chance, or if it is due to a relationship between the variables you are studying.

We use the chi-square test to compare categorical variables

Here chi-square is used to show that is there is a relationship between two categorical features

H_0: relation exist between two categorical features Location and Subjuct

H1:there is no relationship between Location and Type of crop

```
import scipy.stats as stats
```

```
dataset_table=pd.crosstab(df_TONNE_HA['LOCATION'],df_TONNE_HA['SUBJECT'])
print(dataset_table)
```

SUBJECT LOCATION	MAIZE	RICE	SOYBEAN	WHEAT
ARG	32	32	32	32
AUS	32	32	32	32
BGD	32	32	32	32
BRA	32	32	32	32
BRICS	32	32	32	32
CAN	32	32	32	32
CHE	32	32	32	32
CHL	32	32	32	32
CHN	32	32	32	32
COL	32	32	32	32
DZA	32	32	32	32
EGY	32	32	32	32
ETH	32	32	32	32
EU28	28	28	0	27
GHA	32	32	32	32
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

```python
dataset_table.values
```

```
array([[32, 32, 32, 32],
       [32, 32, 32, 32],
       [32, 32, 32, 32],
       [32, 32, 32, 32],
       [32, 32, 32, 32],
       [32, 32, 32, 32],
       [32, 32, 32, 32],
       [32, 32, 32, 32],
       [32, 32, 32, 32],
       [32, 32, 32, 32],
       [32, 32, 32, 32],
       [32, 32, 32, 32],
       [32, 32, 32, 32],
       [28, 28,  0, 27],
       [32, 32, 32, 32],
       [32, 32, 32, 32],
       [32, 32, 32, 32],
       [32, 32, 32, 32],
       [32, 32, 32, 32],
       [32, 32, 32, 32],
       [32, 32, 32, 32],
       [32, 32, 32, 32]
```

```python
#Observed Values
Observed_Values = dataset_table.values
print("Observed Values :-\n",Observed_Values)
```

```
Observed Values :-
 [[32 32 32 32]
 [32 32 32 32]
 [32 32 32 32]
 [32 32 32 32]
 [32 32 32 32]
 [32 32 32 32]
 [32 32 32 32]
 [32 32 32 32]
 [32 32 32 32]
 [32 32 32 32]
 [32 32 32 32]
 [32 32 32 32]
 [32 32 32 32]
 [28 28  0 27]
 [32 32 32 32]
 [32 32 32 32]
 [32 32 32 32]
```

```
val=stats.chi2_contingency(dataset_table)
```

```
val
```

```
(27.646639164611248,
 1.0,
 141,
 array([[32.17947872, 32.15836358, 31.54602441, 32.11613329],
        [32.17947872, 32.15836358, 31.54602441, 32.11613329],
        [32.17947872, 32.15836358, 31.54602441, 32.11613329],
        [32.17947872, 32.15836358, 31.54602441, 32.11613329],
        [32.17947872, 32.15836358, 31.54602441, 32.11613329],
        [32.17947872, 32.15836358, 31.54602441, 32.11613329],
        [32.17947872, 32.15836358, 31.54602441, 32.11613329],
        [32.17947872, 32.15836358, 31.54602441, 32.11613329],
        [32.17947872, 32.15836358, 31.54602441, 32.11613329],
        [32.17947872, 32.15836358, 31.54602441, 32.11613329],
        [32.17947872, 32.15836358, 31.54602441, 32.11613329],
        [32.17947872, 32.15836358, 31.54602441, 32.11613329],
        [32.17947872, 32.15836358, 31.54602441, 32.11613329]
```

```
Expected_Values=val[3]
```

```
no_of_rows=len(dataset_table.iloc[0:2,0])
no_of_columns=len(dataset_table.iloc[0,0:2])
ddof=(no_of_rows-1)*(no_of_columns-1)
print("Degree of Freedom:-",ddof)
alpha = 0.05
```

```
Degree of Freedom:- 1
```

```
from scipy.stats import chi2
chi_square=sum([(o-e)**2./e for o,e in zip(Observed_Values,Expected_Values)])
chi_square_statistic=chi_square[0]+chi_square[1]
```

```
print("chi-square statistic:-",chi_square_statistic)
```

```
chi-square statistic:- 5.011079995341223
```

```
from scipy.stats import chi2
chi_square=sum([(o-e)**2./e for o,e in zip(Observed_Values,Expected_Values)])
chi_square_statistic=chi_square[0]+chi_square[1]
```

```
print("chi-square statistic:-",chi_square_statistic)
```

```
chi-square statistic:- 5.011079995341223
```

```
critical_value=chi2.ppf(q=1-alpha,df=ddof)
print('critical_value:',critical_value)
```

```
critical_value: 3.841458820694124
```

```
#p-value
p_value=1-chi2.cdf(x=chi_square_statistic,df=ddof)
print('p-value:',p_value)
print('Significance level: ',alpha)
print('Degree of Freedom: ',ddof)
print('p-value:',p_value)
```


```
#p-value
p_value=1-chi2.cdf(x=chi_square_statistic,df=ddof)
print('p-value:',p_value)
print('Significance level: ',alpha)
print('Degree of Freedom: ',ddof)
print('p-value:',p_value)
```


```
p-value: 0.025185590452726503
Significance level:  0.05
Degree of Freedom:  1
p-value: 0.025185590452726503
```

```
if chi_square_statistic>=critical_value:
    print("Reject H0,There is a relationship between 2 categorical variables")
else:
    print("Retain H0,There is no relationship between 2 categorical variables")

if p_value<=alpha:
    print("Reject H0,There is a relationship between 2 categorical variables")
else:
    print("Retain H0,There is no relationship between 2 categorical variables")
```

```
Reject H0,There is a relationship between 2 categorical variables
Reject H0,There is a relationship between 2 categorical variables
```

As there is relationship between two categorical features so null hypothesis is rejected

Because value of p value is less than 0.025

Taking sample of 10 randomly chosen data points

```
sample_size=10
sample=np.random.choice(df_TONNE_HA['Value'],sample_size)
sample
```

```
array([ 2.1779964 ,  0.67599384,  2.08930183,  2.61651151, 10.49513331,
        0.75442271,  5.58465277,  2.        ,  1.43936384,  2.93761253])
```

T_test:

A t-test is a statistical test that is used to **compare the means of two groups**. It is often used in hypothesis testing to determine whether a process or treatment actually has an effect on the population of interest, or whether two groups are different from one another.

The **t test** tells us how significant the differences between group means are. It lets our know if those differences in means could have happened by chance.

H0

```
from scipy.stats import ttest_1samp
```

```
ttest,p_value=ttest_1samp(sample,30)
```

```
print(p_value)
```

```
7.000029584407508e-15
```

```
if p_value < 0.05:    # alpha value is 0.05 or 5%
    print(" we are rejecting null hypothesis")
else:
    print("we are accepting null hypothesis")
```

```
 we are rejecting null hypothesis
```

As p_value is less than 0.05 so null hypothesis is rejected so this show that mean of sample and mean of overall population is not same

T Test for two samples:

H0: Mean is same as sample as population

H1: mean is different in sample and their parent population

Taking two samples

```
sample_size=10
sample1=np.random.choice(df_TONNE_HA['Value'],sample_size)
sample1
```

```
array([1.62385964e+00, 8.84948600e+00, 8.84948600e+00, 5.96302891e+00,
       2.16666675e+00, 9.00000000e-04, 1.12500000e+00, 1.99086058e+00,
       2.33251238e+00, 9.00000000e-04])
```

```
sample_size=10
sample2=np.random.choice(df2['Crop_Yield_per_Hectare_in_Tonnes'],sample_size)
sample2
```

```
array([ 7.4,  7.1, 44.5, 25.8,  6.4,  4.8,  6.3,  8.3,  5. , 52.3])
```

Here two samples are randomly chosen for T test to make comparisons

First sample from value of Tonnes from first dataset and 2nd sample is taken from 2nd dataset so this is used to make comparison

```
t_value,p_value=stats.ttest_ind(sample1,sample2)

print('Test statistic is %f'%float("{:.6f}".format(t_value)))

print('p-value for two tailed test is %f'%p_value)


if p_value <=0.05:    # alpha value is 0.05 or 5%
    print(" we are rejecting null hypothesis")
else:
    print("we are accepting null hypothesis")
```

```
Test statistic is -2.351262
p-value for two tailed test is 0.030302
 we are rejecting null hypothesis
```

Here null hypothesis is rejected because p value is less than 0.05 so mean of two quantities are very different

Correlation

```
df2.corr()
```

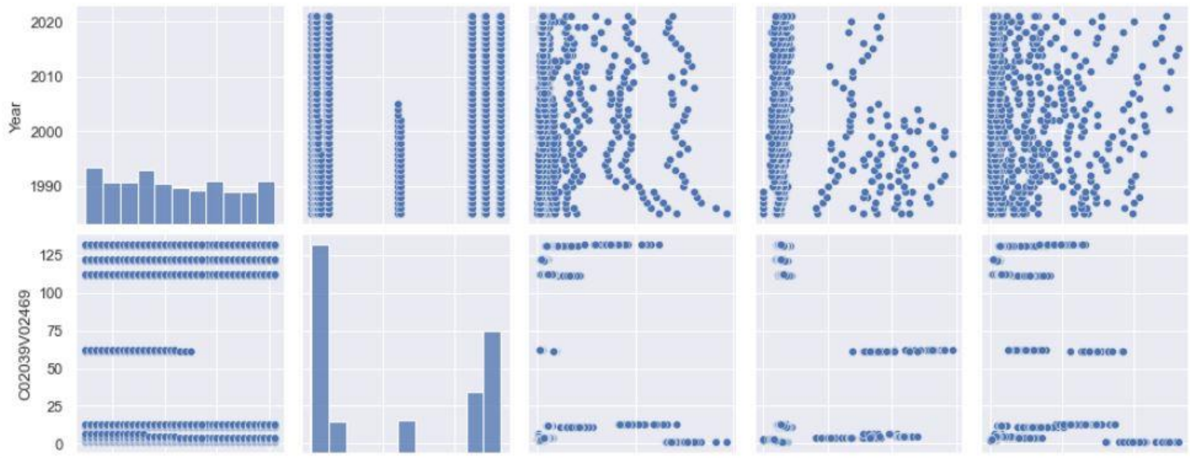
	Year	C02039V02469	Area_under_Crops_Hectares	Crop_Yield_per_Hectare_in_Tonnes	crop_production_in_Tonnes
Year	1.000000	0.045495	0.030966	-0.192102	0.029505
C02039V02469	0.045495	1.000000	-0.154555	-0.181995	-0.202705
Area_under_Crops_Hectares	0.030966	-0.154555	1.000000	-0.251131	0.881368
Crop_Yield_per_Hectare_in_Tonnes	-0.192102	-0.181995	-0.251131	1.000000	0.073780
crop_production_in_Tonnes	0.029505	-0.202705	0.881368	0.073780	1.000000

The correlation shows positive or negative relationship

Correlation is **an indication about the changes between two variables**.

```
sns.pairplot(df2)
```

```
<seaborn.axisgrid.PairGrid at 0x1c0f825f8e0>
```




```
df2.corr()
```

	Year	C02039V02469	Area_under_Crops_Hectares	Crop_Yield_per_Hectare_in_Tonnes	crop_production_in_Tonnes
Year	1.000000	0.045495	0.030966	-0.192102	0.029505
C02039V02469	0.045495	1.000000	-0.154555	-0.181995	-0.202705
Area_under_Crops_Hectares	0.030966	-0.154555	1.000000	-0.251131	0.881368
Crop_Yield_per_Hectare_in_Tonnes	-0.192102	-0.181995	-0.251131	1.000000	0.073780
crop_production_in_Tonnes	0.029505	-0.202705	0.881368	0.073780	1.000000

```
df_TONNE_HA.corr()
```

	TIME	Value
TIME	1.000000	0.164199
Value	0.164199	1.000000

It shown that area, year and crop production has direct relation as as area increased crop production also increases

Z test

A z-test is a statistical test to determine whether two population means are different when the variances are known and the sample size is large. A z-test is a hypothesis test in which the z-statistic follows a normal distribution. A z-statistic, or z-score, is a number representing the result from the z-test.

Here H1 is null hypothesis and H1 is alternative hypothesis.

H0: NO difference of mean in two samples

H1: Difference of Mean in two samples

```
from statsmodels.stats.weightstats import ztest as ztest
```

```
z_test,p_value=ztest(sample1,sample2, value=0)
z_test,p_value
```

```
(-2.3512620746732367, 0.018709850835706 61)
```

```
if p_value < 0.05:    # alpha value is 0.05 or 5%
    print(" we are rejecting null hypothesis")
else:
    print("we are accepting null hypothesis")
```

```
we are rejecting null hypothesis
```

Are null hypothesis is rejected as there are difference between means .

Outcome of my analysis

Here two detests are used Ireland Crop production data and international crop production data of 48 countries as Ireland is not mentioned here so it is needed to import data of Ireland.

By comparing mean value with other countries it is cleared that Ireland crop production capability is more than other countries as Ireland has highest average production per hectare

As z test is used for comparison of two samples which shows mean value is different.

Here chie square test shows that there is relationship between categorical variables of subject and location.

Challenges:

There are many problems which I faced which are following:

- to find a good dataset.
- To arrange dataset.
- To find pattern of dataset
- Measurement problems
  So these are problems which I faced

Datasets Finding:

- ## Ireland Crop production 1985-2007
  This data is can be found on official website of Ireland datasets IRELAND'S OPEN DATA PORTAL. This is data from 1985 to 2007.
  Link:
  https://data.gov.ie/dataset/aqa03-crop-yield-1985-2007?package_type=dataset

- ## Ireland crop production 2008-2021

  This data is found on official website of Ireland datasets IRELAND'S OPEN DATA PORTAL .this data contains record from 2008 to 2021
  Link:

  https://data.gov.ie/dataset/aqa04-crop-yield-and-production?package_type=dataset

- ## International crop production data:

  This data is found on data. World. It contains data of 48 countries.
  Link:
  https://data.world/oecd/crop-production

Reference:

Hypothesis testing in Machine learning using Python posted by Yogesh Agrawal on Towards data science portal

Link:

https://towardsdatascience.com/hypothesis-testing-in-machine-learning-using-python-a0dc89e169ce

Statistical Inference Using Python **posted by ELLURU PAVAN KUMAR REDDY on analyticsvidhya.com**

Link: https://www.analyticsvidhya.com/blog/2022/02/statistical-inference-using-python/