Crop Yield Analysis

The science of training machines to learn and produce models 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 impacts of climate change.

Crop yield prediction is an important agricultural problem. The Agricultural yield primarily depends on weather conditions (rain, temperature, etc), pesticides and accurate information about history of crop yield is an important thing for making decisions related to agricultural risk management and future predictions. 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 consist of 10 most consumed crops. It is a regression problem

These corps include :

Cassava

Maize

Plantains and others Potatoes

Rice, paddy

Sorghum

Soybeans

Sweet potatoes

Wheat

Yams

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

```
df=pd.read_csv("/content/yield_df.csv")
df
```

	Unnamed: 0	Area	Item	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonn
0	0	Albania	Maize	1990	36613	1485.0	121.
1	1	Albania	Potatoes	1990	66667	1485.0	121.
2	2	Albania	Rice, paddy	1990	23333	1485.0	121.
3	3	Albania	Sorghum	1990	12500	1485.0	121.
4	4	Albania	Soybeans	1990	7000	1485.0	121.
...	...	...	...	...	...	...	
28237	28237	Zimbabwe	Rice, paddy	2013	22581	657.0	2550.
28238	28238	Zimbabwe	Sorghum	2013	3066	657.0	2550.
28239	28239	Zimbabwe	Soybeans	2013	13142	657.0	2550.
28240	28240	Zimbabwe	Sweet potatoes	2013	22222	657.0	2550.

```
df_yield = pd.read_csv('yield.csv')
df_yield
```

	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
...	...	...	...	...	...	...	...	...	...	...	...	...
56712	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2012	2012	hg/ha	24420

```
df_pesticides = pd.read_csv('pesticides.csv')
df_rainfall = pd.read_csv('rainfall.csv')
df_temp = pd.read_csv('temp.csv')
```

56715	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2015	2015	hg/ha	19826

```
df_pesticides.head()
```

	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

```
df_rainfall.head()
```

	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

```
df_temp.head()
```

	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

```
df_yield.head()
```

	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

```
df_yield['Domain Code'].nunique()
```

```
1
```

```
df_yield['Domain'].nunique()
```

```
1
```

```
df_yield.drop({'Domain','Domain Code'},axis=1,inplace=True)
df_yield.head()
```

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

```
df_yield['Area'].value_counts()
```

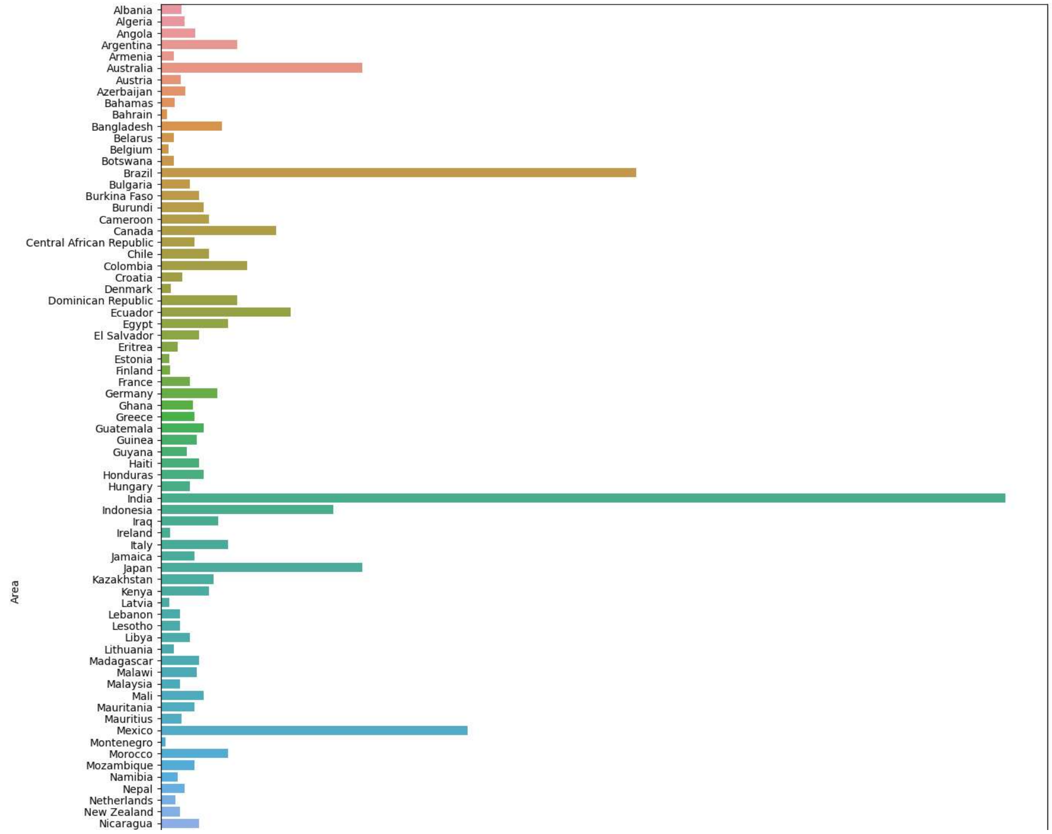
```
United Republic of Tanzania          560
Democratic Republic of the Congo     560
Nigeria                              560
Venezuela (Bolivarian Republic of)   532
Cameroon                             528
                                    ...
Estonia                               50
Djibouti                              36
Sudan                                 35
Montenegro                            33
South Sudan                           20
Name: Area, Length: 212, dtype: int64
```

```
df_yield['Area'].nunique()
```

```
212
```

```
plt.figure(figsize=(15, 20))
sns.countplot(y=df['Area'])
```

```
<Axes: xlabel='count', ylabel='Area'>
```




```
df_yield['Element'].nunique()
```

```
1
```


```
df_yield.drop({'Area Code','Element Code','Item Code','Year Code','Element','Unit'},axis=1,inplace = True)
df_yield.head()
```

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

```
df['Item'].nunique()
```

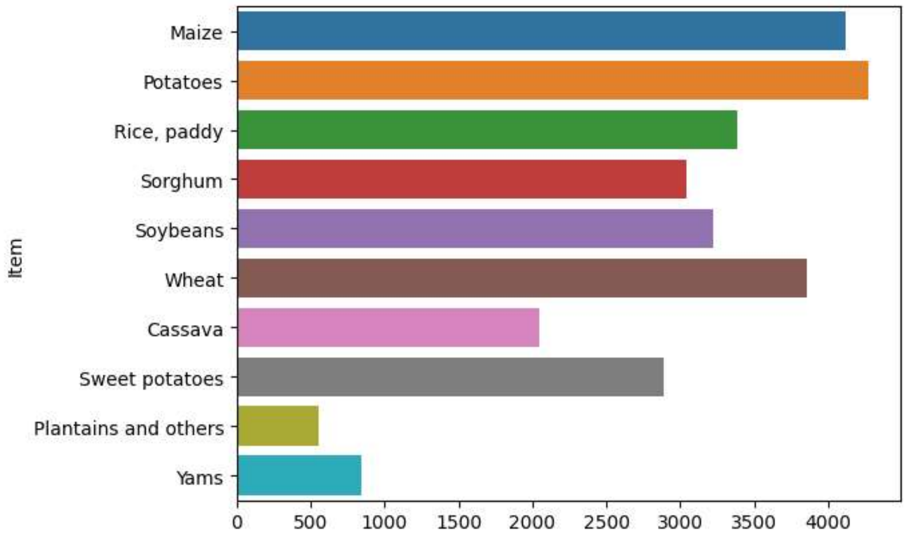
```
10
```

```
df_yield['Item'].value_counts()
```

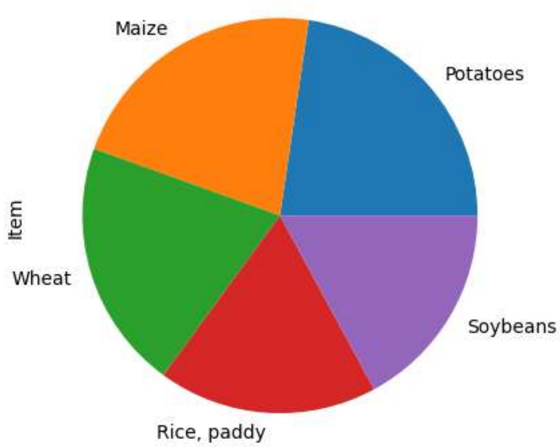
```
Maize                   8631
Potatoes                7876
Rice, paddy             6469
Sweet potatoes          6356
Wheat                   6160
Cassava                 5718
Sorghum                 5511
Soybeans                4192
Yams                    3150
Plantains and others    2654
Name: Item, dtype: int64
```

```
sns.countplot(y=df['Item'])
```

```
<Axes: xlabel='count', ylabel='Item'>
```




```
df['Item'].value_counts()[:5].plot(kind='pie')
plt.show()
```




```
df_yield = df_yield.rename(index=str, columns={"Value": "hg/ha_yield"})
df_yield.head()
```

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

```
df_yield.isnull().sum()
```

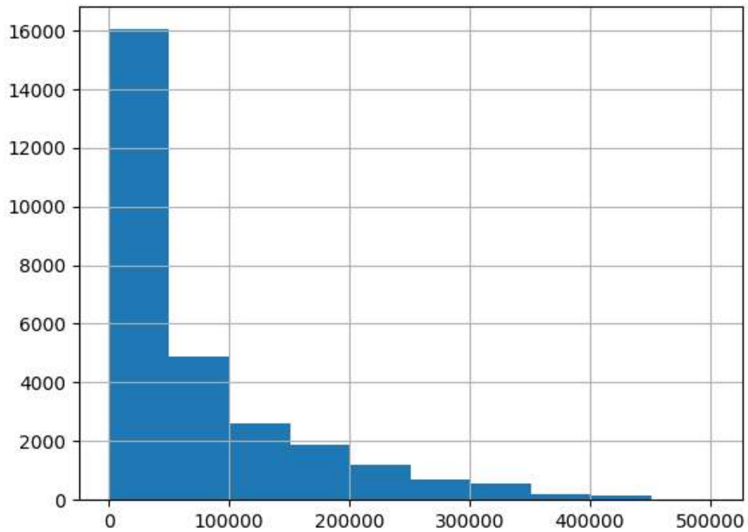
```
Area           0
Item           0
Year           0
hg/ha_yield    0
dtype: int64
```

```
df_yield.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 56717 entries, 0 to 56716
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
```

```
 0   Area         56717 non-null  object
 1   Item         56717 non-null  object
 2   Year         56717 non-null  int64
 3   hg/ha_yield  56717 non-null  int64
dtypes: int64(2), object(2)
memory usage: 2.2+ MB
```

```
df['hg/ha_yield'].hist()
plt.show()
```



CLIMATE CONDITIONS

```
df_rainfall.head()
```

	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

```
df_rainfall['average_rain_fall_mm_per_year'].nunique()
df_rainfall.rename(index=str,columns={'average_rain_fall_mm_per_year':'Avg. RainFall'},inplace=True)
df_rainfall.head()
```

	Area	Year	Avg. RainFall
0	Afghanistan	1985	327
1	Afghanistan	1986	327
2	Afghanistan	1987	327
3	Afghanistan	1989	327
4	Afghanistan	1990	327

```
df_rainfall.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   Avg. RainFall  5953 non-null   object
```

```
dtypes: int64(1), object(2)
memory usage: 210.2+ KB
```

```
df_rainfall.rename(index=str,columns={' Area':'Area'},inplace=True)
df_rainfall.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   Avg. RainFall  5953 non-null   object
dtypes: int64(1), object(2)
memory usage: 210.2+ KB
```

```
df_rainfall.isnull().sum()
```

```
Area               0
Year               0
Avg. RainFall    774
dtype: int64
```

```
df_rainfall.isna().any()
```

```
Area             False
Year             False
Avg. RainFall     True
dtype: bool
```

```
df_rainfall['Avg. RainFall'] = pd.to_numeric(df_rainfall['Avg. RainFall'],errors = 'coerce')
df_rainfall.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   Avg. RainFall  5947 non-null   float64
dtypes: float64(1), int64(1), object(1)
memory usage: 210.2+ KB
```

```
df_rainfall.describe()
```

	Year	Avg. RainFall
count	6727.000000	5947.000000
mean	2001.354839	1124.743232
std	9.530114	786.257365
min	1985.000000	51.000000
25%	1993.000000	534.000000
50%	2001.000000	1010.000000
75%	2010.000000	1651.000000
max	2017.000000	3240.000000

```
df_rainfall.dropna(inplace=True)
df_rainfall.isnull().sum()
```

```
Area             0
Year             0
Avg. RainFall    0
dtype: int64
```

```
df_yield.head()
```

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

```
df_rainfall.head()
```

	Area	Year	Avg. RainFall
0	Afghanistan	1985	327.0
1	Afghanistan	1986	327.0
2	Afghanistan	1987	327.0
3	Afghanistan	1989	327.0
4	Afghanistan	1990	327.0

```
yield_df = pd.merge(df_yield, df_rainfall, on=['Year','Area'])
yield_df.head()
```

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

PESTICIDES DATA

```
df_pesticides.head()
```

	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

```
df_pesticides['Domain'].unique()
```

```
array(['Pesticides Use'], dtype=object)
```

```
df_pesticides['Element'].unique()
```

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

```
df_pesticides.drop({'Domain','Element'},axis=1,inplace=True)
df_pesticides.head()
```

	Area	Item	Year	Unit	Value

```
df_pesticides['Item'].unique()
```

```
array(['Pesticides (total)'], dtype=object)
```

```
df_pesticides.drop('Item',axis=1,inplace=True)
df_pesticides.head()
```

	Area	Year	Unit	Value
0	Albania	1990	tonnes of active ingredients	121.0
1	Albania	1991	tonnes of active ingredients	121.0
2	Albania	1992	tonnes of active ingredients	121.0
3	Albania	1993	tonnes of active ingredients	121.0
4	Albania	1994	tonnes of active ingredients	201.0

```
df_pesticides['Unit'].nunique()
```

```
1
```

```
df_pesticides.drop('Unit',axis=1,inplace=True)
df_pesticides.head()
```

	Area	Year	Value
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

```
df_pesticides.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4349 entries, 0 to 4348
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Area    4349 non-null   object
 1   Year    4349 non-null   int64
 2   Value   4349 non-null   float64
dtypes: float64(1), int64(1), object(1)
memory usage: 102.1+ KB
```

```
df_pesticides.rename(index=str,columns={'Value':'pesticides_tonnes'},inplace=True)
df_pesticides.head()
```

	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

```
df_pesticides.isnull().sum()
```

```
Area                 0
Year                 0
pesticides_tonnes    0
dtype: int64
```

```
yield_df = pd.merge(yield_df,df_pesticides, on=['Year','Area'])
yield_df.head()
```

	Area	Item	Year	hg/ha_yield	Avg. RainFall	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 TEMPERATURE

```
df_temp.head()
```

	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

```
df_temp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 71311 entries, 0 to 71310
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   year      71311 non-null  int64  
 1   country   71311 non-null  object 
 2   avg_temp  68764 non-null  float64
dtypes: float64(1), int64(1), object(1)
memory usage: 1.6+ MB
```

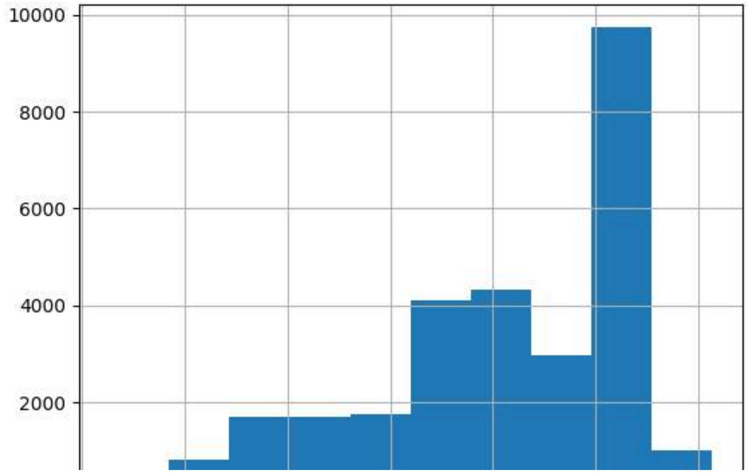
```
df_temp.describe()
```

	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

```
df_temp.isnull().sum()
```

```
year           0
country        0
avg_temp    2547
dtype: int64
```

```
df['avg_temp'].hist()
plt.show()
```




```
df_temp.dropna(inplace=True)
df_temp.isnull().sum()
```

```
year        0
country     0
avg_temp    0
dtype: int64
```

```
df_temp = df_temp.rename(index=str, columns={"year": "Year", "country":'Area'})
df_temp.head()
```

	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
7	1856	Côte D'Ivoire	26.28
8	1857	Côte D'Ivoire	25.17

```
yield_df = pd.merge(yield_df,df_temp, on=['Area','Year'])
yield_df.head()
```

	Area	Item	Year	hg/ha_yield	Avg. RainFall	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

```
yield_df.shape
```

```
(28242, 7)
```

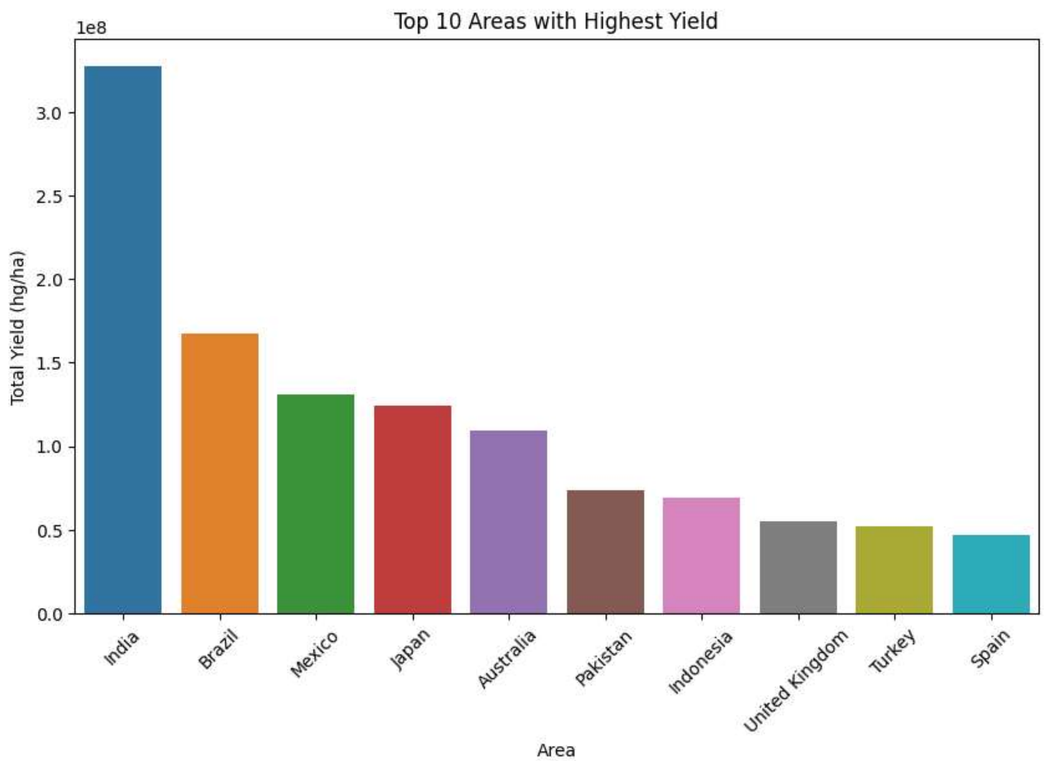
```
yield_df.isnull().sum()
```

```
Area                 0
Item                 0
Year                 0
hg/ha_yield          0
Avg. RainFall        0
pesticides_tonnes    0
avg_temp             0
dtype: int64
```

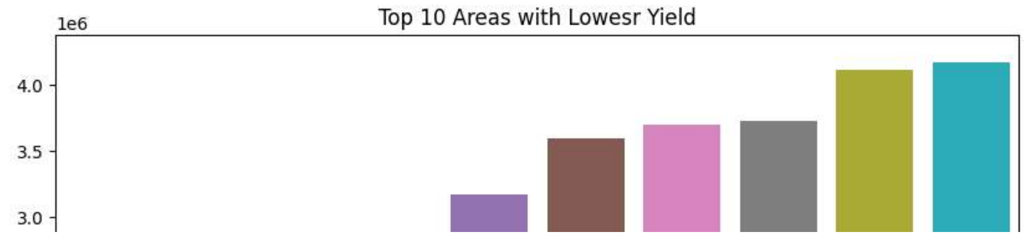
```
highest_yield = yield_df.groupby(['Area'],sort=True)['hg/ha_yield'].sum().nlargest(10)
lowest_yield = yield_df.groupby(['Area'],sort=True)['hg/ha_yield'].sum().nsmallest(10)
```

India has the highest yield production where as Botswana has the lowest

```
plt.figure(figsize=(10, 6))
sns.barplot(x=highest_yield.index, y=highest_yield.values)
plt.xlabel('Area')
plt.ylabel('Total Yield (hg/ha)')
plt.title('Top 10 Areas with Highest Yield')
plt.xticks(rotation=45)
plt.show()
```

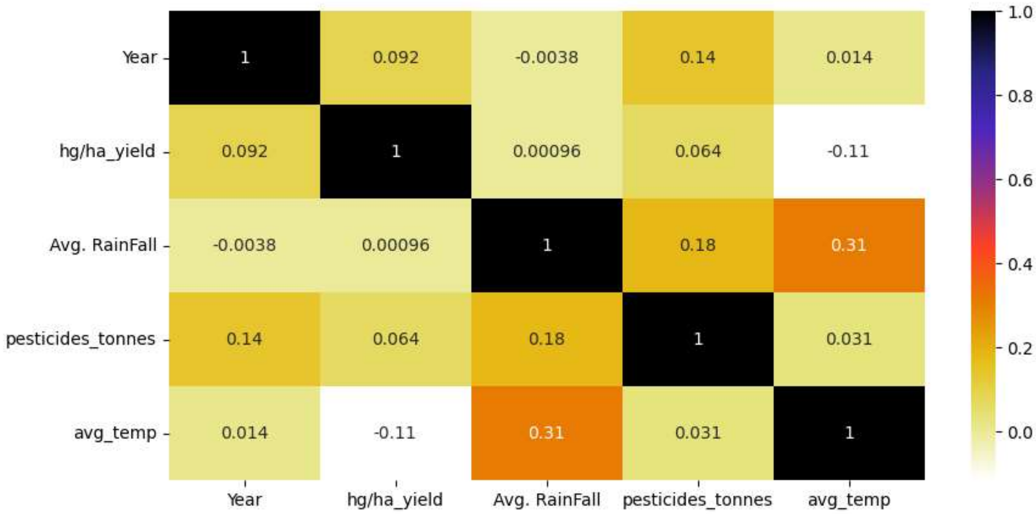



```
plt.figure(figsize=(10, 6))
sns.barplot(x=lowest_yield.index, y=lowest_yield.values)
plt.xlabel('Area')
plt.ylabel('Total Yield (hg/ha)')
plt.title('Top 10 Areas with Lowesr Yield')
plt.xticks(rotation=45)
plt.show()
```




```
plt.figure(figsize=(10,5))
cor = yield_df.corr()
sns.heatmap(cor,annot=True,cmap=plt.cm.CMRmap_r)
plt.show()
```

```
<ipython-input-197-a8f2a077f070>:2: FutureWarning: The default value of numeric_only in DataFrame.corr
  cor = yield_df.corr()
```




```
yield_df.head()
```

	Area	Item	Year	hg/ha_yield	Avg. RainFall	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

```
yield_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 28242 entries, 0 to 28241
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Area               28242 non-null  object
 1   Item               28242 non-null  object
 2   Year               28242 non-null  int64
 3   hg/ha_yield        28242 non-null  int64
 4   Avg. RainFall      28242 non-null  float64
 5   pesticides_tonnes  28242 non-null  float64
 6   avg_temp           28242 non-null  float64
dtypes: float64(3), int64(2), object(2)
memory usage: 1.7+ MB
```

Encoding Categorical Variables:

There are two categorical columns in the dataframe, categorical data are variables that contain label values rather than numeric 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 represents the numerical value of the entry in the dataset. This encoding will create a binary column for each category and returns a matrix with the results.

```
categorical_values=[cn for cn in df.columns if df[cn].dtype == object]
categorical_values
```

```
['Area', 'Item']
```

```
from sklearn.preprocessing import OneHotEncoder
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()
```

	Year	Avg. RainFall	pesticides_tonnes	avg_temp	Country_Albania	Country_Algeria	Country_Angola	Count
0	1990	1485.0	121.0	16.37	1	0	0	
1	1990	1485.0	121.0	16.37	1	0	0	
2	1990	1485.0	121.0	16.37	1	0	0	
3	1990	1485.0	121.0	16.37	1	0	0	
4	1990	1485.0	121.0	16.37	1	0	0	

5 rows × 115 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 calculations than features with low magnitudes.

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

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

```
features
```

```
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],
       [4.49670743e-01, 3.28894097e-04, 5.13458262e-01, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       ...,
       [1.90028222e-01, 6.93361288e-03, 6.28960818e-01, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [1.90028222e-01, 6.93361288e-03, 6.28960818e-01, ...,
        1.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [1.90028222e-01, 6.93361288e-03, 6.28960818e-01, ...,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00]])
```

Training Data:

The dataset will be split to two datasets, the training dataset and test dataset. 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 80/20 for train/test.

The training dataset is the intial dataset used to train ML algorithm 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)

```
from sklearn.model_selection import train_test_split
train_data, test_data, train_labels, test_labels = train_test_split(features, label, test_size=0.2, random_state=42)
```

MODEL CREATION

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
```

```
models = [
    RandomForestRegressor(),
    GradientBoostingRegressor(),
    SVR(),
    DecisionTreeRegressor()
]
from sklearn.metrics import mean_squared_error, r2_score

for model in models:
    # Fit the model on the training data
    model.fit(train_data, train_labels)

    # Make predictions on the test data
    predictions = model.predict(test_data)

    # Evaluate the model
    mse = mean_squared_error(test_labels, predictions)
    r2 = r2_score(test_labels, predictions)

    # Print the evaluation metrics
    print(model.__class__.__name__)
    print("Mean Squared Error:", mse)
    print("R2 Score:", r2)
    print()
```

```
RandomForestRegressor
Mean Squared Error: 189896780.05786735
R2 Score: 0.973820599991205

GradientBoostingRegressor
Mean Squared Error: 989654013.3782226
R2 Score: 0.8635650995312147

SVR
Mean Squared Error: 8665721562.269018
R2 Score: -0.19466686625412555

DecisionTreeRegressor
Mean Squared Error: 285799942.85446787
R2 Score: 0.9605992738571034
```

Based on the evaluation metrics provided, it appears that the RandomForestRegressor model has the best performance among the models you evaluated.

The RandomForestRegressor model achieved a relatively low Mean Squared Error (MSE) of 191,858,309.45, indicating that the average squared difference between the predicted and actual values is comparatively small. Additionally, the R2 score of 0.9735 suggests that the model explains approximately 97.35% of the variability in the target variable, indicating a strong fit to the data.

On the other hand, the GradientBoostingRegressor, SVR, and DecisionTreeRegressor models had higher MSE values and lower R2 scores. This indicates that they may not perform as well as the RandomForestRegressor model in capturing the patterns and variability in the data.

0s completed at 10:06 PM