```
import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'crop-yield-prediction-dataset:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F1760177%2F2874008%2Fbundle

KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
  os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'), target_is_directory=True)
except FileExistsError:
  pass
try:
  os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'), target_is_directory=True)
except FileExistsError:
  pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
            dl = 0
            data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
                sys.stdout.flush()
                data = fileres.read(CHUNK_SIZE)
            if filename.endswith('.zip'):
              with ZipFile(tfile) as zfile:
                zfile.extractall(destination_path)
            else:
              with tarfile.open(tfile.name) as tarfile:
                tarfile.extractall(destination_path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destination_path}')
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')
```

```
Downloading crop-yield-prediction-dataset, 981807 bytes compressed
[==================================================] 981807 bytes downloaded
Downloaded and uncompressed: crop-yield-prediction-dataset
Data source import complete.
```

```
import numpy as np
import pandas as pd
```

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

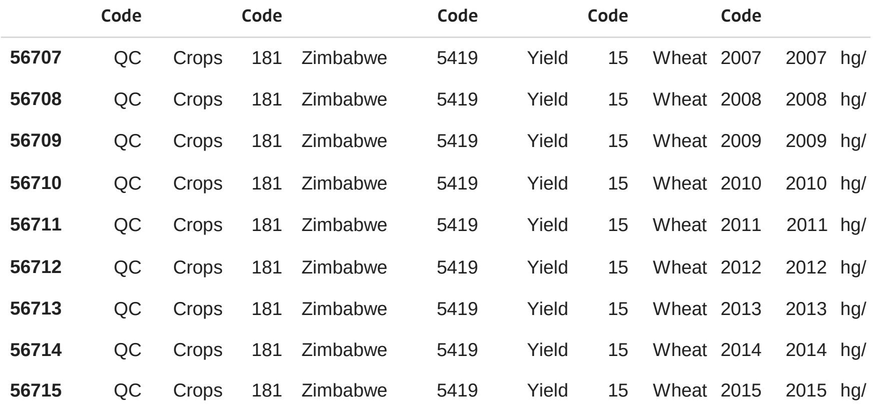
```
    (56717, 12)
```

```
df_yield.head()
```

	Domain Code	Domain	Area Code	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit
0	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1961	1961	hg/ha
1	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1962	1962	hg/ha
2	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1963	1963	hg/ha
3	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1964	1964	hg/ha

```
df_yield.tail(10)
```

OUTPUT:



	Code		Code		Code		Code		Code		
56707	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2007	2007	hg/
56708	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2008	2008	hg/
56709	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2009	2009	hg/
56710	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2010	2010	hg/
56711	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2011	2011	hg/
56712	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2012	2012	hg/
56713	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2013	2013	hg/
56714	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2014	2014	hg/
56715	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2015	2015	hg/

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

	Domain Code	Domain	Area Code	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit
0	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1961	1961	hg/ha
1	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1962	1962	hg/ha
2	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1963	1963	hg/ha
3	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1964	1964	hg/ha

```
# drop unwanted columns.
df_yield = df_yield.drop(['Year Code','Element Code','Element','Year Code','Area Code','Domain Code','Domain','Unit','Item Code'], axis
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.describe()
```

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

```
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_rain = pd.read_csv('../input/crop-yield-prediction-dataset/rainfall.csv')
df_rain.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_rain.tail()
```

	Area	Year	average_rain_fall_mm_per_year
6722	Zimbabwe	2013	657
6723	Zimbabwe	2014	657
6724	Zimbabwe	2015	657
6725	Zimbabwe	2016	657
6726	Zimbabwe	2017	657

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

Checking for the datatypes

```
df_rain.info()
```

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

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
```

Next, droping any empty rows from dataset and merge yield dataframe with rain dataframe by year and area columns

```
df_rain = df_rain.dropna()
```

```
df_rain.describe()
```

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

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

```
yield_df.head()
```

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

We can see that now the years start from the first yield dataframe the starting year was 1961, now it's 1985 because that's when the rainfalldata begins.

```
yield_df.describe()
```

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

```
df_pes = pd.read_csv('../input/crop-yield-prediction-dataset/pesticides.csv')
df_pes.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
	Pesticides			Pesticides		tonnes of active	

```
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()
```

	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_pes.describe()
```

	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

```
df_pes.info()
```

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

Merge Pesticides dataframe with yield dataframe

```
yield_df = pd.merge(yield_df, df_pes, on=['Year','Area'])
yield_df.shape
```

```
(18949, 6)
```

```
yield_df.head()
```

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

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

```
avg_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

```
avg_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

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

```
avg_temp = avg_temp.rename(index=str, columns={"year": "Year", "country":'Area'})
avg_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
3	1852	Côte D'Ivoire	NaN
4	1853	Côte D'Ivoire	NaN

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

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

```
yield_df.shape
```

```
(28242, 7)
```

```
yield_df.describe()
```

	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes
count	28242.000000	28242.000000	28242.00000	28242.000000
mean	2001.544296	77053.332094	1149.05598	37076.909344
std	7.051905	84956.612897	709.81215	59958.784665
min	1990.000000	50.000000	51.00000	0.040000
25%	1995.000000	19919.250000	593.00000	1702.000000
50%	2001.000000	38295.000000	1083.00000	17529.440000
75%	2008.000000	104676.750000	1668.00000	48687.880000
max	2013.000000	501412.000000	3240.00000	367778.000000

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

```
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
```

yield_df is the final obtained dataframe;

```
yield_df.groupby('Item').count()
```

	Area	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes
Item					
Cassava	2045	2045	2045	2045	2045
Maize	4121	4121	4121	4121	4121
F antains and others	556	556	556	556	556
Potatoes	4276	4276	4276	4276	4276
Rice, paddy	3388	3388	3388	3388	3388
S rghum	3039	3039	3039	3039	3039
S ybeans	3223	3223	3223	3223	3223
Sweet	2890	2890	2890	2890	2890

```
yield_df.describe()
```

	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes
count	28242.000000	28242.000000	28242.00000	28242.000000
mean	2001.544296	77053.332094	1149.05598	37076.909344
std	7.051905	84956.612897	709.81215	59958.784665
min	1990.000000	50.000000	51.00000	0.040000
25%	1995.000000	19919.250000	593.00000	1702.000000
50%	2001.000000	38295.000000	1083.00000	17529.440000
75%	2008.000000	104676.750000	1668.00000	48687.880000
max	2013.000000	501412.000000	3240.00000	367778.000000

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

```
101
```

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

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

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

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

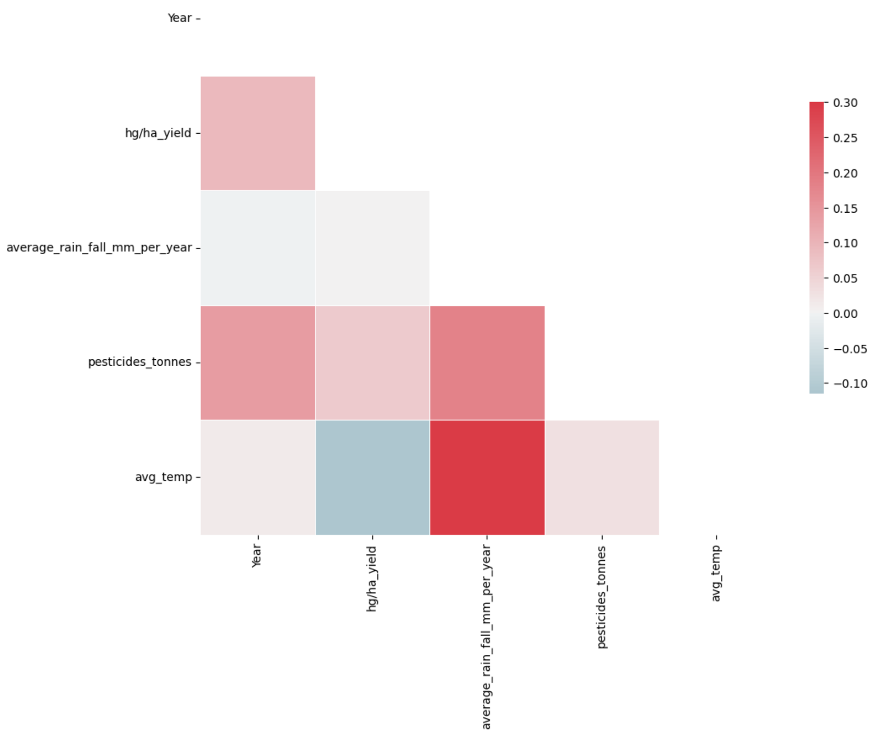
```
yield_df.groupby(['Item','Area'],sort=True)['hg/ha_yield'].sum().nlargest(10)
```

```
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
```

```
import sklearn
import seaborn as sns
import matplotlib.pyplot as plt
```

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

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

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

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

cmap = sns.diverging_palette(220, 10, as_cmap=True)


sns.heatmap(correlation_data, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5});
```



```
yield_df.head()
```

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

```
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	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country_Albania
0	1990	1485.0	121.0	16.37	1
1	1990	1485.0	121.0	16.37	1
2	1990	1485.0	121.0	16.37	1
3	1990	1485.0	121.0	16.37	1
4	1990	1485.0	121.0	16.37	1

5 rows × 115 columns

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

```
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
```

```
features.head()
```

	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country_Albania	Count
0	1485.0	121.0	16.37	1	
1	1485.0	121.0	16.37	1	
2	1485.0	121.0	16.37	1	
3	1485.0	121.0	16.37	1	
4	1485.0	121.0	16.37	1	

5 rows × 114 columns

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
features=scaler.fit_transform(features)
```

```
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]])
```

```
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)
```

```
#write final df to csv file
#yield_df.to_csv('../input/crop-yield-prediction-dataset/yield_df.csv')
```

```
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)
```

```
# 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])
```

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

# models = [
#     GradientBoostingRegressor(n_estimators=200, max_depth=3, random_state=0),
#      RandomForestRegressor(n_estimators=200, max_depth=3, random_state=0),
#     svm.SVR(),
#    DecisionTreeRegressor()
# ]
```

```
# model_train=list(map(compare_models,models))
```

```
# print(*model_train, sep = "\n")
```

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

```
yield_df_onehot.head()
```

	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country_A
0	36613	1485.0	121.0	16.37	
1	66667	1485.0	121.0	16.37	
2	23333	1485.0	121.0	16.37	
3	12500	1485.0	121.0	16.37	
4	7000	1485.0	121.0	16.37	

5 rows × 115 columns

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

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

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
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
itmlist=list(pd.DataFrame(itm).index.get_level_values(1))
items=[i.split("_")[1] for i in itmlist]
```

```
test_df.head()
```

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

5 rows × 114 columns

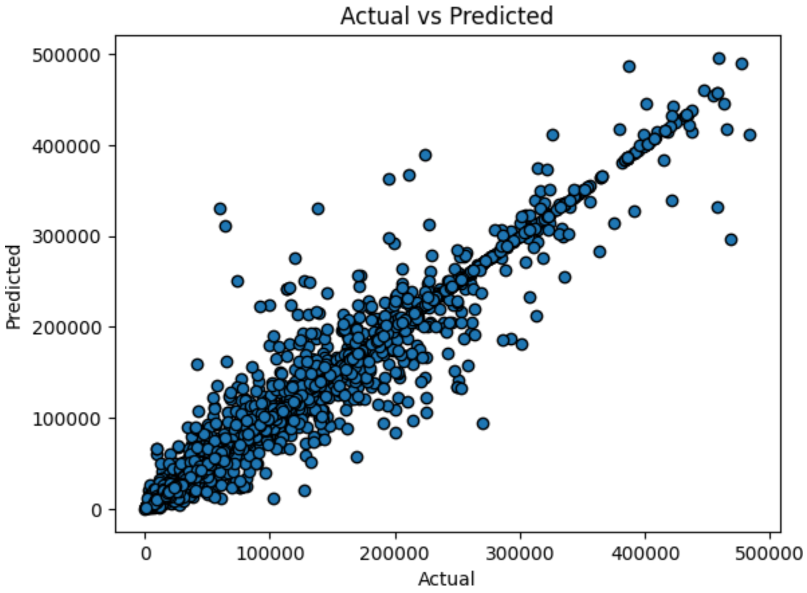
```
test_df.drop([col for col in test_df.columns if 'Item' in col],axis=1,inplace=True)
test_df.drop([col for col in test_df.columns if 'Country' in col],axis=1,inplace=True)
test_df.head()
```

	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

```
test_df['Country']=countries
test_df['Item']=items
test_df.head()
```

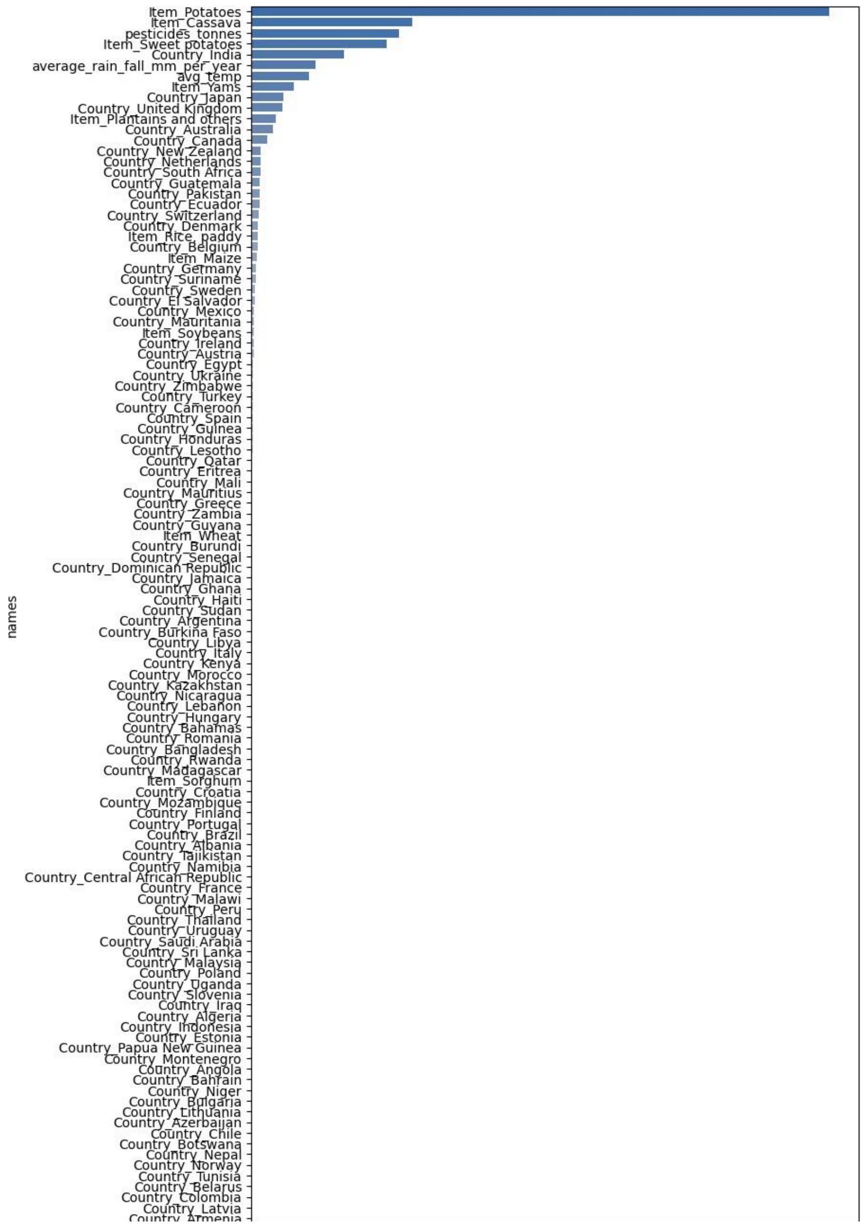
	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country	Item
0	0.183443	0.110716	0.542078	Spain	Rice, paddy
1	0.458451	0.000413	0.627257	Madagascar	Wheat
2	0.183443	0.106159	0.518228	Spain	Sorghum
3	1.000000	0.224154	0.890971	Colombia	Potatoes
4	0 458451	0 000355	0 625213	Madagascar	Sweet

```
from sklearn.tree import DecisionTreeRegressor
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))


# So let's run the model actual values against the predicted ones

fig, ax = plt.subplots()

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()
```



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


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);
```

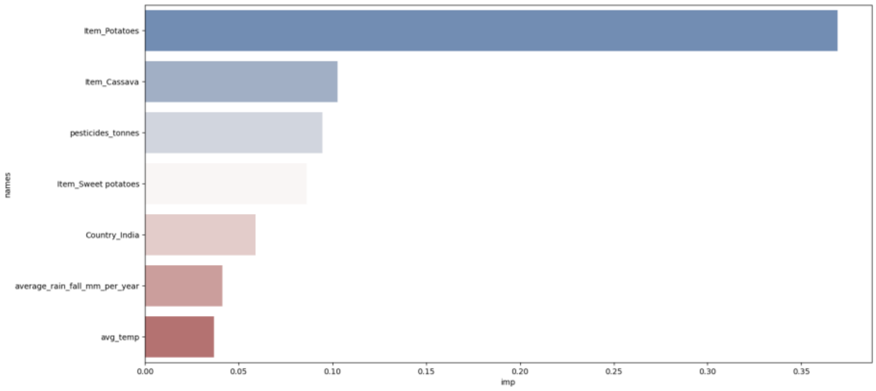
```
<ipython-input-67-fc39f57c7b3a>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.barplot(x="imp",y="names",palette="vlag",data=df,orient="h",ax=ax);
```



Getting only top 7 of features importance in the model:

```
#7 most important factors that affect crops
a4_dims = (16.7, 8.27)

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()
df=df.nlargest(7, 'imp')
sns.barplot(x="imp",y="names",palette="vlag",data=df,orient="h",ax=ax);
```

```
<ipython-input-68-3ecae61a09e3>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.barplot(x="imp",y="names",palette="vlag",data=df,orient="h",ax=ax);
```



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