Gathering & Cleaning Data

Crops Yield Data:

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

```
df_yield = pd.read_csv('/content/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	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

Next steps: Generate code with df_yield | View recommended plots | New interactive sheet

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

	Domain Code	Domain	Area Code	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	Value
56707	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2007	2007	hg/ha	29998
56708	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2008	2008	hg/ha	30097
56709	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2009	2009	hg/ha	30000
56710	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2010	2010	hg/ha	27681
56711	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2011	2011	hg/ha	26274
56712	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2012	2012	hg/ha	24420
56713	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2013	2013	hg/ha	22888
56714	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2014	2014	hg/ha	21357
56715	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2015	2015	hg/ha	19826
56716	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2016	2016	hg/ha	18294

```
# 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	hg/ha_yield
0	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1961	1961	hg/ha	14000
1	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1962	1962	hg/ha	14000
2	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1963	1963	hg/ha	14260
3	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1964	1964	hg/ha	14257
4	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1965	1965	hg/ha	14400

Next steps: Generate code with df_yield | View recommended plots | New interactive sheet

```
# drop unwanted columns.
df_yield = df_yield.drop(['Year Code','Element Code','Element','Year Code','Area Code','Domain Code','Domain','Unit','Item Code'], axis=1)
df_yield.head()
```

	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

Next steps: Generate code with df_yield | View recommended plots | New interactive sheet

```
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: 4.2+ MB
```

Climate Data : Rainfall

```
df_rain = pd.read_csv('/content/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

Next steps: Generate code with df_rain | View recommended plots | New interactive sheet

```
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'})
```

```
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
```

```
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
```

```
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	Maize	1986	16875	327.0
2	Afghanistan	Maize	1987	17020	327.0
3	Afghanistan	Maize	1989	16963	327.0
4	Afghanistan	Maize	1990	17582	327.0

Next steps: Generate code with yield_df | View recommended plots | New interactive sheet

```
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

Pesticides Data:

```
df_pes = pd.read_csv('/content/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
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

Next steps: Generate code with df_pes | View recommended plots | New interactive sheet

```
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

Next steps: Generate code with df_pes | View recommended plots | New interactive sheet

```
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: 264.9+ KB
```

```
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_tonnes
0	Albania	Maize	1990	36613	1485.0	121.0
1	Albania	Maize	1991	29068	1485.0	121.0
2	Albania	Maize	1992	24876	1485.0	121.0
3	Albania	Maize	1993	24185	1485.0	121.0
4	Albania	Maize	1994	25848	1485.0	201.0

Next steps: Generate code with yield_df | View recommended plots | New interactive sheet

Average Temprature:

```
avg_temp=  pd.read_csv('/content/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

Next steps: Generate code with avg_temp | View recommended plots | New interactive sheet

```
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

```
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

Next steps: Generate code with avg_temp | View recommended plots | New interactive sheet

```
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_tonnes	avg_temp
0	Albania	Maize	1990	36613	1485.0	121.0	16.37
1	Albania	Maize	1991	29068	1485.0	121.0	15.36
2	Albania	Maize	1992	24876	1485.0	121.0	16.06
3	Albania	Maize	1993	24185	1485.0	121.0	16.05
4	Albania	Maize	1994	25848	1485.0	201.0	16.96

Next steps: Generate code with yield_df | View recommended plots | New interactive sheet

```
yield_df.shape
```

```
(28242, 7)
```

```
yield_df.describe()
```

	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
count	28242.000000	28242.000000	28242.00000	28242.000000	28242.000000
mean	2001.544296	77053.332094	1149.05598	37076.909344	20.542627
std	7.051905	84956.612897	709.81215	59958.784665	6.312051
min	1990.000000	50.000000	51.00000	0.040000	1.300000
25%	1995.000000	19919.250000	593.00000	1702.000000	16.702500
50%	2001.000000	38295.000000	1083.00000	17529.440000	21.510000
75%	2008.000000	104676.750000	1668.00000	48687.880000	26.000000
max	2013.000000	501412.000000	3240.00000	367778.000000	30.650000

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

	0
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

Data Exploration

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

Item	Area	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
Cassava	2045	2045	2045	2045	2045	2045
Maize	4121	4121	4121	4121	4121	4121
Plantains and others	556	556	556	556	556	556
Potatoes	4276	4276	4276	4276	4276	4276
Rice, paddy	3388	3388	3388	3388	3388	3388
Sorghum	3039	3039	3039	3039	3039	3039
Soybeans	3223	3223	3223	3223	3223	3223
Sweet potatoes	2890	2890	2890	2890	2890	2890
Wheat	3857	3857	3857	3857	3857	3857
Yams	847	847	847	847	847	847

```
yield_df.describe()
```

	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
count	28242.000000	28242.000000	28242.00000	28242.000000	28242.000000
mean	2001.544296	77053.332094	1149.05598	37076.909344	20.542627
std	7.051905	84956.612897	709.81215	59958.784665	6.312051
min	1990.000000	50.000000	51.00000	0.040000	1.300000
25%	1995.000000	19919.250000	593.00000	1702.000000	16.702500
50%	2001.000000	38295.000000	1083.00000	17529.440000	21.510000
75%	2008.000000	104676.750000	1668.00000	48687.880000	26.000000
max	2013.000000	501412.000000	3240.00000	367778.000000	30.650000

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

```
101
```

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

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

dtype: int64

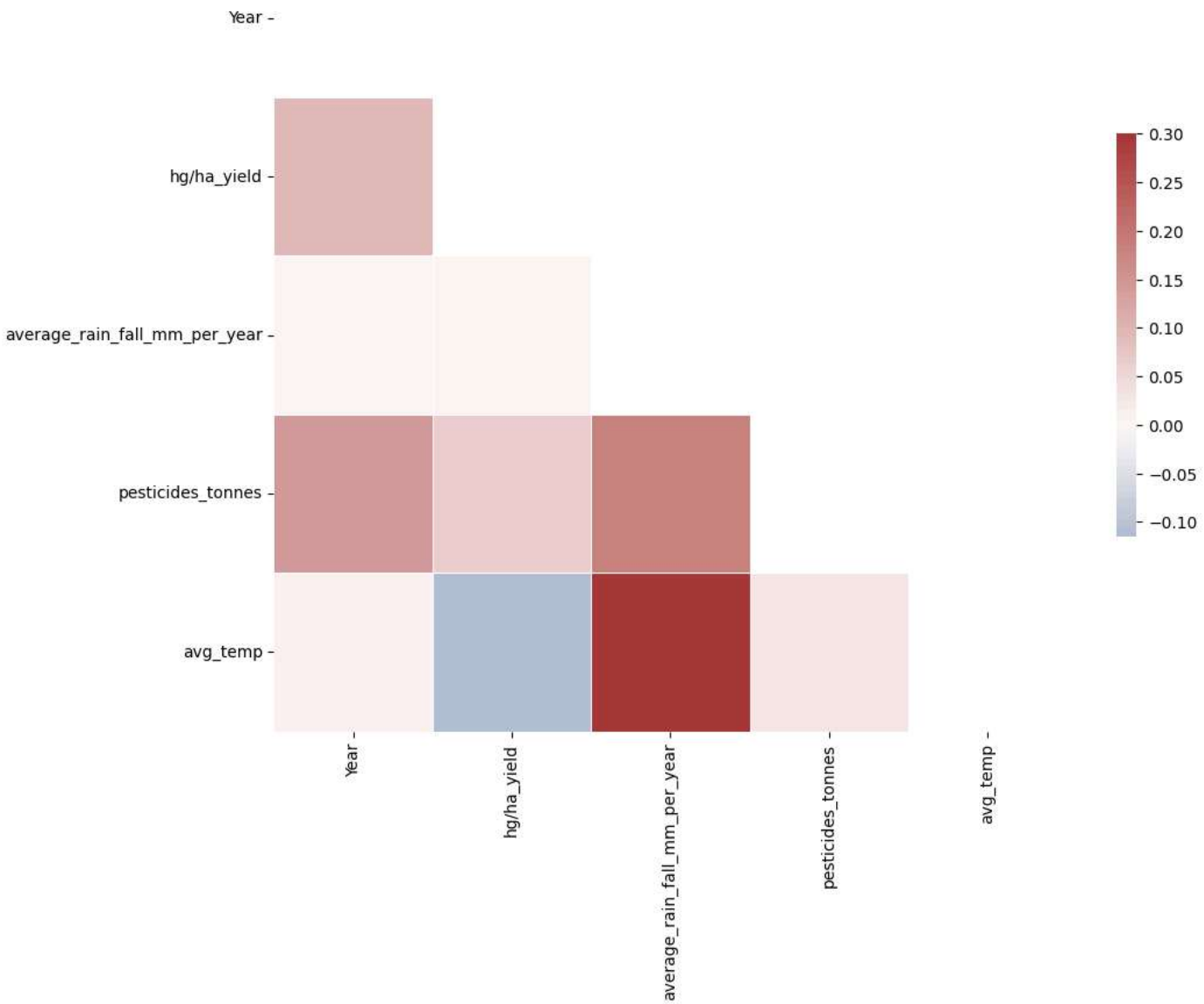
```
yield_df.groupby(['Item','Area'],sort=True)['hg/ha_yield'].sum().nlargest(10)
```

Item	Area	hg/ha_yield
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

dtype: int64

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

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

# Compute the correlation matrix
correlation_data = yield_df.select_dtypes(include=[np.number]).corr()

# Generate a mask for the upper triangle
mask = np.zeros_like(correlation_data, dtype=bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Define the colormap correctly
cmap = "vlag"

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

plt.show()
```



Data Preprocessing

```
yield_df.head()
```

	Area	Item	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
0	Albania	Maize	1990	36613	1485.0	121.0	16.37
1	Albania	Maize	1991	29068	1485.0	121.0	15.36
2	Albania	Maize	1992	24876	1485.0	121.0	16.06
3	Albania	Maize	1993	24185	1485.0	121.0	16.05
4	Albania	Maize	1994	25848	1485.0	201.0	16.96

Next steps: Generate code with yield_df | View recommended plots | New interactive sheet

Encoding Categorical Variables:

```
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	Country_Algeria	Country_Angola	Country_Argentina
0	1990	1485.0	121.0	16.37	True	False	False	False
1	1991	1485.0	121.0	15.36	True	False	False	False
2	1992	1485.0	121.0	16.06	True	False	False	False
3	1993	1485.0	121.0	16.05	True	False	False	False
4	1994	1485.0	201.0	16.96	True	False	False	False

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

```
features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28242 entries, 0 to 28241
Columns: 114 entries, average_rain_fall_mm_per_year to Item_Yams
dtypes: bool(111), float64(3)
memory usage: 3.6 MB
```

```
features.head()
```

	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country_Albania	Country_Algeria	Country_Angola	Country_Argentina	Coun
0	1485.0	121.0	16.37	True	False	False	False	
1	1485.0	121.0	15.36	True	False	False	False	
2	1485.0	121.0	16.06	True	False	False	False	
3	1485.0	121.0	16.05	True	False	False	False	
4	1485.0	201.0	16.96	True	False	False	False	

Scaling Features:

```
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, 4.79045997e-01, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [4.49670743e-01, 3.28894097e-04, 5.02896082e-01, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       ...,
       [1.90028222e-01, 9.08240940e-03, 6.63713799e-01, ...,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00],
       [1.90028222e-01, 9.17806494e-03, 6.54855196e-01, ...,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00],
       [1.90028222e-01, 6.93361288e-03, 6.28960818e-01, ...,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00]])
```

Training Data

```
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)
```

Model Comparison & Selection

```
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")
```

```
['GradientBoostingRegressor', 0.8948534135078133]
['RandomForestRegressor', 0.6937411507173485]
['SVR', -0.1954729063349249 8]
['DecisionTreeRegressor', 0.9623519545171448]
```

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np

# Function to train the model and calculate evaluation metrics
def evaluate_model(model):
    """
    Train the model, generate predictions, and calculate evaluation metrics.
    """
    model_name = model.__class__.__name__

    # Train the model
    model.fit(train_data, train_labels)

    # Make predictions
    y_pred = model.predict(test_data)

    # Calculate metrics
    mse = mean_squared_error(test_labels, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(test_labels, y_pred)
    r2 = r2_score(test_labels, y_pred)

    # Adjusted R² Calculation
    n = len(test_labels)  # Number of test samples
    p = test_data.shape[1]  # Number of features
    adjusted_r2 = 1 - ((1 - r2) * (n - 1) / (n - p - 1))

    # Convert metrics to hectogram per hectare (hg/ha)
    mse_hg = mse * 100  # Convert kg²/ha² to hg²/ha²
    rmse_hg = rmse * 10  # Convert kg/ha to hg/ha
    mae_hg = mae * 10  # Convert kg/ha to hg/ha

    # Print results
    print(f"\n {model_name} - Model Evaluation:")
```

```
    print(f"MSE (hg²/ha²): {mse_hg:.2f}")
    print(f"RMSE (hg/ha): {rmse_hg:.2f}")
    print(f"MAE (hg/ha): {mae_hg:.2f}")
    print(f"R² Score: {r2:.4f}")
    print(f"Adjusted R² Score: {adjusted_r2:.4f}")

# Define models
models = {
    "GradientBoostingRegressor": GradientBoostingRegressor(n_estimators=200, max_depth=3, random_state=0),
    "RandomForestRegressor": RandomForestRegressor(n_estimators=200, max_depth=3, random_state=0),
    "DecisionTreeRegressor": DecisionTreeRegressor(),
    "SVR": svm.SVR()
}

# Evaluate all models
for model_name, model in models.items():
    evaluate_model(model)
```

```
 GradientBoostingRegressor - Model Evaluation:
MSE (hg²/ha²): 75291435220.94
RMSE (hg/ha): 274392.85
MAE (hg/ha): 178202.52
R² Score: 0.8949
Adjusted R² Score: 0.8927

 RandomForestRegressor - Model Evaluation:
MSE (hg²/ha²): 219300208222.34
RMSE (hg/ha): 468295.00
MAE (hg/ha): 317483.63
R² Score: 0.6937
Adjusted R² Score: 0.6874

 DecisionTreeRegressor - Model Evaluation:
MSE (hg²/ha²): 27365626771.44
RMSE (hg/ha): 165425.59
MAE (hg/ha): 57995.08
R² Score: 0.9618
Adjusted R² Score: 0.9610

 SVR - Model Evaluation:
MSE (hg²/ha²): 856032267794.01
RMSE (hg/ha): 925220.12
MAE (hg/ha): 565948.34
R² Score: -0.1955
Adjusted R² Score: -0.2201
```

```
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_Albania	Country_Algeria	Country_Angola	Country_Ar
0	36613	1485.0	121.0	16.37	True	False	False	
1	29068	1485.0	121.0	15.36	True	False	False	
2	24876	1485.0	121.0	16.06	True	False	False	
3	24185	1485.0	121.0	16.05	True	False	False	
4	25848	1485.0	201.0	16.96	True	False	False	

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

test_df=pd.DataFrame(test_data,columns=yield_df_onehot.loc[:, yield_df_onehot.columns != 'hg/ha_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]].st
cntrylist=list(pd.DataFrame(cntry).index.get_level_values(1))
countries=[i.split("_")[1] for i in cntrylist]
itm=test_df[[col for col in test_df.columns if 'Item' in col]].stack()[test_df[[col for col in test_df.columns if 'Item' in col]].stack()>0]
itmlist=list(pd.DataFrame(itm).index.get_level_values(1))
items=[i.split("_")[1] for i in itmlist]
```

```
test_df.head()
```

	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country_Albania	Country_Algeria	Country_Angola	Country_Argentina	Coun
0	0.183443	0.090370	0.535264	0.0	0.0	0.0	0.0	
1	0.458451	0.000135	0.631005	0.0	0.0	0.0	0.0	
2	0.183443	0.132330	0.552300	0.0	0.0	0.0	0.0	
3	1.000000	0.179695	0.867802	0.0	0.0	0.0	0.0	
4	0.458451	0.000305	0.629983	0.0	0.0	0.0	0.0	

```
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	0.090370	0.535264
1	0.458451	0.000135	0.631005
2	0.183443	0.132330	0.552300
3	1.000000	0.179695	0.867802
4	0.458451	0.000305	0.629983

Next steps: Generate code with test_df | View recommended plots | New interactive sheet

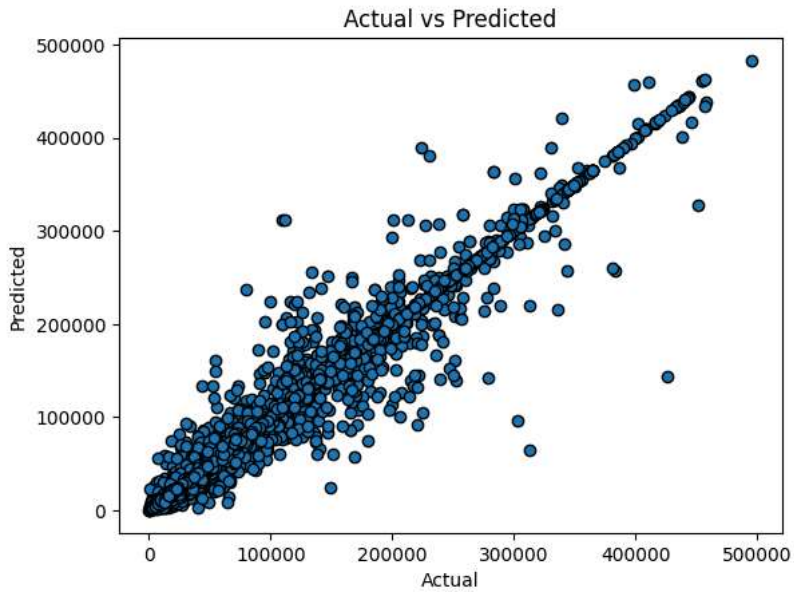
```
test_df['Country']=countries
test_df['Item']=items
test_df.head()
```

	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country	Item
0	0.183443	0.090370	0.535264	Spain	Sweet potatoes
1	0.458451	0.000135	0.631005	Madagascar	Potatoes
2	0.183443	0.132330	0.552300	Spain	Sweet potatoes
3	1.000000	0.179695	0.867802	Colombia	Soybeans
4	0.458451	0.000305	0.629983	Madagascar	Rice, paddy

Next steps: Generate code with test_df | View recommended plots | New interactive sheet

```
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()
```

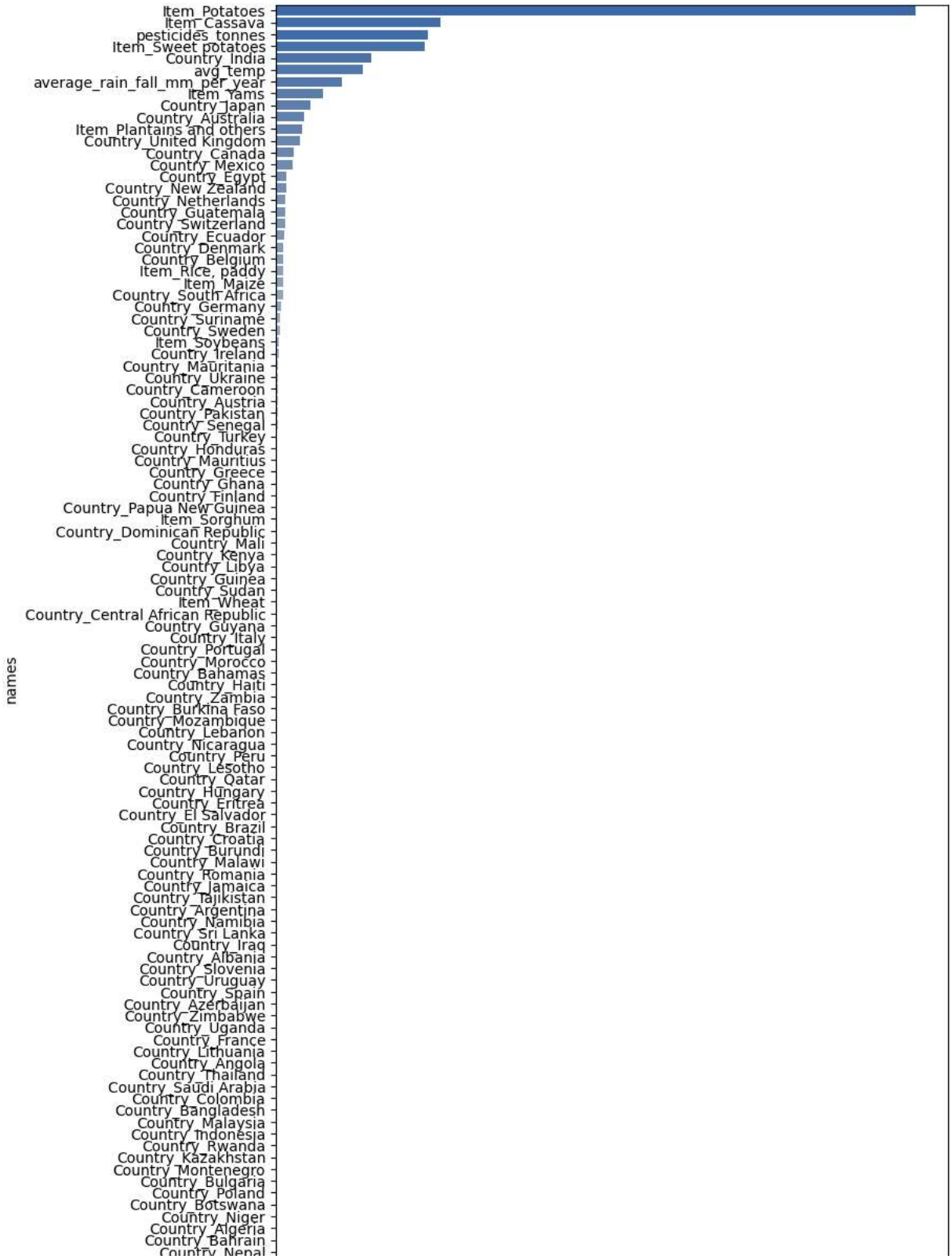


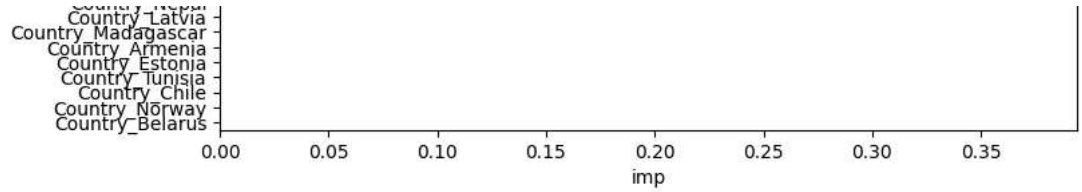
Model Results & Conclusions

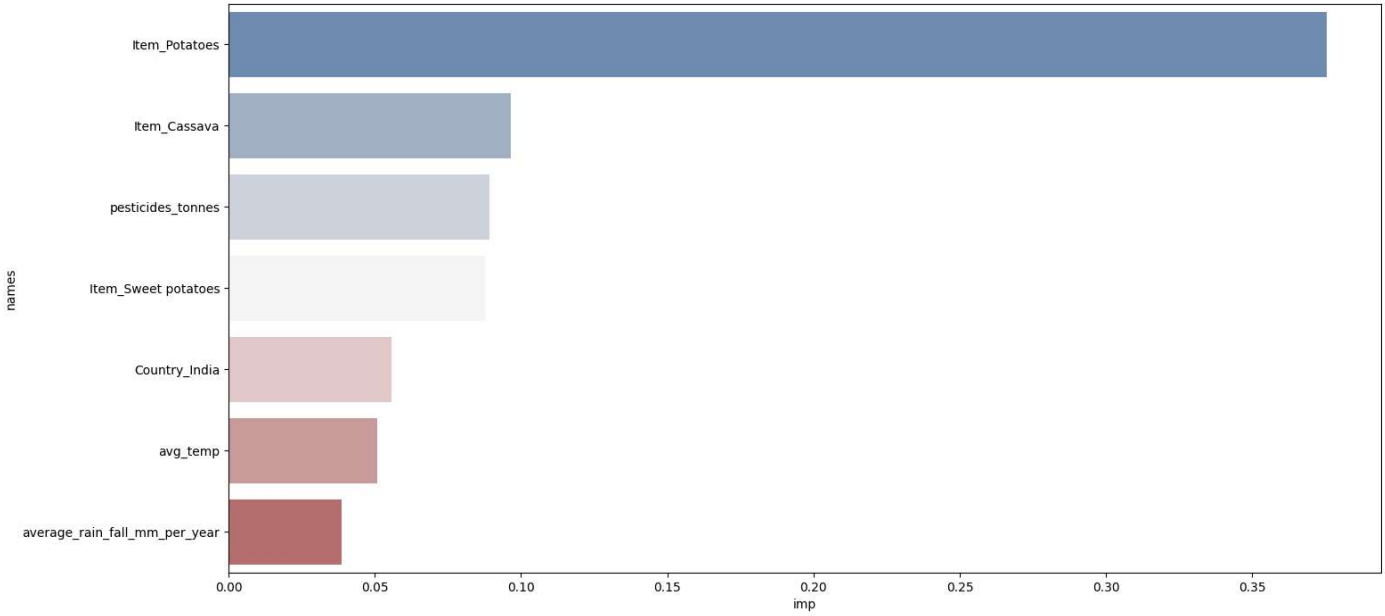
```
varimp= {'imp':model.feature_importances_,'names':yield_df_onehot.columns[yield_df_onehot.columns!="hg/ha_yield"]}
```

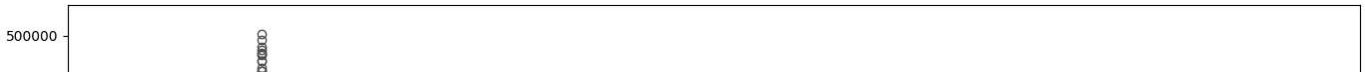
```
a4_dims = (8.27,16.7)
fig, ax = plt.subplots(figsize=a4_dims)
df=pd.DataFrame.from_dict(varimp)
df.sort_values(ascending=False,by=["imp"],inplace=True)
df=df.dropna()
sns.barplot(x="imp",y="names",palette="vlag",data=df,orient="h",ax=ax);

#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);

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

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

```
<ipython-input-68-9382a76d0a38>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend

  sns.barplot(x="imp",y="names",palette="vlag",data=df,orient="h",ax=ax);
<ipython-input-68-9382a76d0a38>:16: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend

  sns.barplot(x="imp",y="names",palette="vlag",data=df,orient="h",ax=ax);
<ipython-input-68-9382a76d0a38>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

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




Country_Latvia
Country_Madagascar
Country_Armenia
Country_Estonia
Country_Tunisia
Country_Chile
Country_Norway
Country_Belarus
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
imp


Item_Potatoes
Item_Cassava
pesticides_tonnes
Item_Sweet potatoes
Country_India
avg_temp
average_rain_fall_mm_per_year
names
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
imp


500000