```
# importing necessary libraries

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

```
# loading the dataset

crop_data=pd.read_csv("/content/crop_production.csv")
crop_data
```

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production
0	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Arecanut	1254.0	2000.0
1	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Other Kharif pulses	2.0	1.0
2	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Rice	102.0	321.0
3	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Banana	176.0	641.0
4	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Cashewnut	720.0	165.0
...	...	...	...	...	...	...	...
103163	Madhya Pradesh	BALAGHAT	2000	Rabi	Safflower	6.0	1.0
103164	Madhya Pradesh	BALAGHAT	2000	Rabi	Wheat	14004.0	9796.0
103165	Madhya Pradesh	BALAGHAT	2000	Whole Year	Coriander	291.0	65.0
103166	Madhya Pradesh	BALAGHAT	2000	Whole Year	Dry chillies	405.0	72.0
103167	Madhya Pradesh	BALAGHAT	2000	Whole Year	Garlic	131.0	449.0

103168 rows × 7 columns

```
crop_data.shape

#rows X columns
```

```
(103168, 7)
```

```
# dataset columns
crop_data.columns
```

```
Index(['State_Name', 'District_Name', 'Crop_Year', 'Season', 'Crop', 'Area',
       'Production'],
      dtype='object')
```

```
# statistical inference of the dataset

crop_data.describe()
```

	Crop_Year	Area	Production
count	103168.000000	103168.000000	1.009830e+05
mean	2005.893455	9081.339826	1.196607e+06
std	4.931049	30605.983819	2.528363e+07
min	1997.000000	0.040000	0.000000e+00
25%	2002.000000	87.000000	1.000000e+02
50%	2006.000000	566.000000	7.760000e+02
75%	2010.000000	3652.000000	6.771500e+03
max	2014.000000	877029.000000	1.125000e+09

```
# Checking missing values of the dataset in each column
crop_data.isnull().sum()
```

```
State_Name          0
District_Name       0
Crop_Year           0
Season              0
Crop                0
Area                0
Production       2185
dtype: int64
```

```
# Dropping missing values
crop_data = crop_data.dropna()
crop_data
```

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production
0	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Arecanut	1254.0	2000.0
1	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Other Kharif pulses	2.0	1.0
2	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Rice	102.0	321.0
3	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Banana	176.0	641.0
4	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Cashewnut	720.0	165.0
...	...	...	...	...	...	...	...

```
#checking
crop_data.isnull().values.any()
```

```
False
```

```
# Displaying State Names present in the dataset
crop_data.State_Name.unique()
```

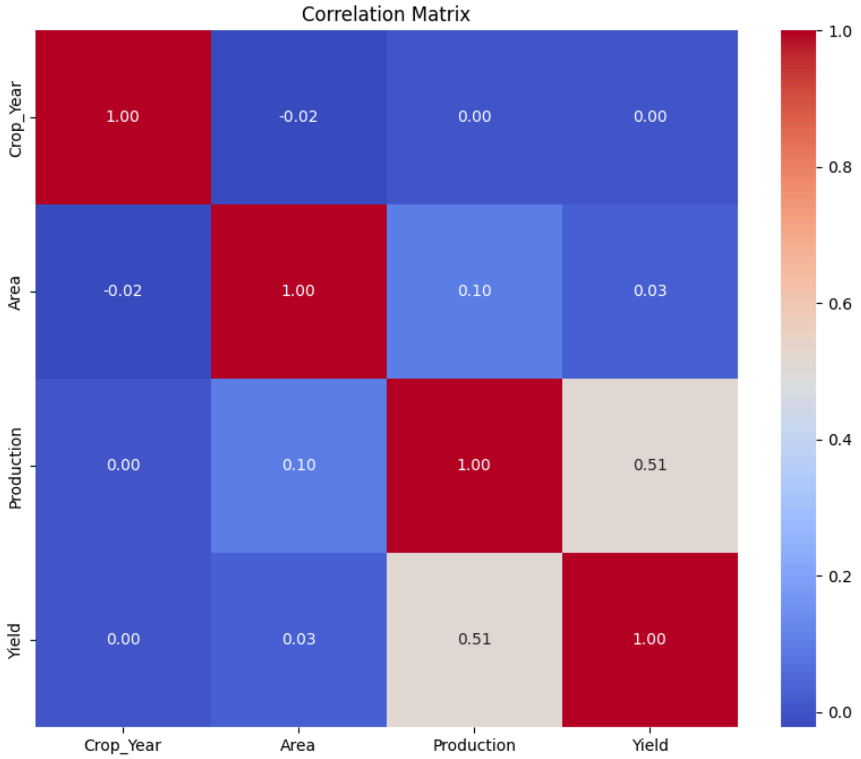
```
array(['Andaman and Nicobar Islands', 'Andhra Pradesh',
       'Arunachal Pradesh', 'Assam', 'Bihar', 'Chandigarh',
       'Chhattisgarh', 'Dadra and Nagar Haveli', 'Goa', 'Gujarat',
       'Haryana', 'Himachal Pradesh', 'Jammu and Kashmir ', 'Jharkhand',
       'Karnataka', 'Kerala', 'Madhya Pradesh'], dtype=object)
```

```
# Adding a new column Yield which indicates Production per unit Area.

crop_data['Yield'] = (crop_data['Production'] / crop_data['Area'])
crop_data.head(10)
```

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	
0	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Arecanut	1254.0	2000.0	1
1	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Other Kharif pulses	2.0	1.0	0
2	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Rice	102.0	321.0	3
3	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Banana	176.0	641.0	3
4	Andaman and Nicobar	NICOBARS	2000	Whole	Cashewnut	720.0	165.0	0

```
# Dropping unnecessary columns

data = crop_data.drop(['State_Name'], axis = 1)
```

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

# Select only numeric columns from your DataFrame
numeric_data = data.select_dtypes(include='number')

# Compute the correlation matrix for numeric data
corr_matrix = numeric_data.corr()

# Plotting the heatmap with annotations and a title
plt.figure(figsize=(10, 8))  # Adjust the figure size if needed
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```



```
dummy = pd.get_dummies(data)
dummy
```

	Crop_Year	Area	Production	Yield	District_Name_AGAR MALWA	District_Name_A
0	2000	1254.0	2000.0	1.594896	False	
1	2000	2.0	1.0	0.500000	False	
2	2000	102.0	321.0	3.147059	False	
3	2000	176.0	641.0	3.642045	False	
4	2000	720.0	165.0	0.229167	False	
...	...	...	...	...	...	
103163	2000	6.0	1.0	0.166667	False	
103164	2000	14004.0	9796.0	0.699514	False	
103165	2000	291.0	65.0	0.223368	False	
103166	2000	405.0	72.0	0.177778	False	
103167	2000	131.0	449.0	3.427481	False	

100983 rows × 392 columns

```
from sklearn.model_selection import train_test_split

x = dummy.drop(["Production","Yield"], axis=1)
y = dummy["Production"]

# Splitting data set - 25% test dataset and 75%

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25, random_state=5)

print("x_train :",x_train.shape)
print("x_test :",x_test.shape)
print("y_train :",y_train.shape)
print("y_test :",y_test.shape)
```

```
x_train : (75737, 390)
x_test : (25246, 390)
y_train : (75737,)
y_test : (25246,)
```

```
print(x_train)
print(y_train)
```

```
59924                 False                False                    False
5560                  False                False                    False
20536                 False                False                    False
18709                 False                False                    False
35767                 False                False                    False

       ...  Crop_Tobacco  Crop_Tomato  Crop_Turmeric  Crop_Turnip  Crop_Urad  \
98859  ...         False        False          False        False      False
40637  ...         False        False          False        False      False
85066  ...         False        False          False        False      False
87863  ...         False        False          False        False      False
82926  ...         False        False          False        False      False
...    ...           ...          ...            ...          ...        ...
59924  ...         False        False          False        False      False
5560   ...         False        False          False        False      False
20536  ...         False        False          False        False      False
18709  ...         False        False          False        False      False
35767  ...         False        False          False        False      False

       Crop_Varagu  Crop_Wheat  Crop_other fibres  Crop_other misc. pulses  \
98859        False       False              False                    False
40637        False       False              False                    False
85066        False       False              False                    False
87863        False       False              False                    False
82926        False       False              False                    False
...            ...         ...                ...                      ...
59924        False       False              False                    False
5560         False       False              False                    False
20536        False       False              False                    False
18709        False       False              False                    False
35767        False       False              False                    False

       Crop_other oilseeds
98859                False
40637                False
85066                False
87863                False
82926                False
...                    ...
59924                False
5560                 False
20536                False
18709                False
35767                False

[75737 rows x 390 columns]
98859      108.0
40637        6.0
85066    12745.0
87863     2284.0
82926      161.0
          ...
59924     1500.0
5560       600.0
20536       27.0
18709     4779.0
35767       81.0
Name: Production, Length: 75737, dtype: float64
```

```
# Training the Simple Linear Regression model .

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
# Predicting the test Results

lr_predict = model.predict(x_test)
lr_predict
```

```
array([ -179651.33984375,  -989905.37890625, -3914674.02148438, ...,
       -5991530.2265625 ,  1107923.00585938, -1027953.0390625 ])
```

```
model.score(x_test,y_test)
```

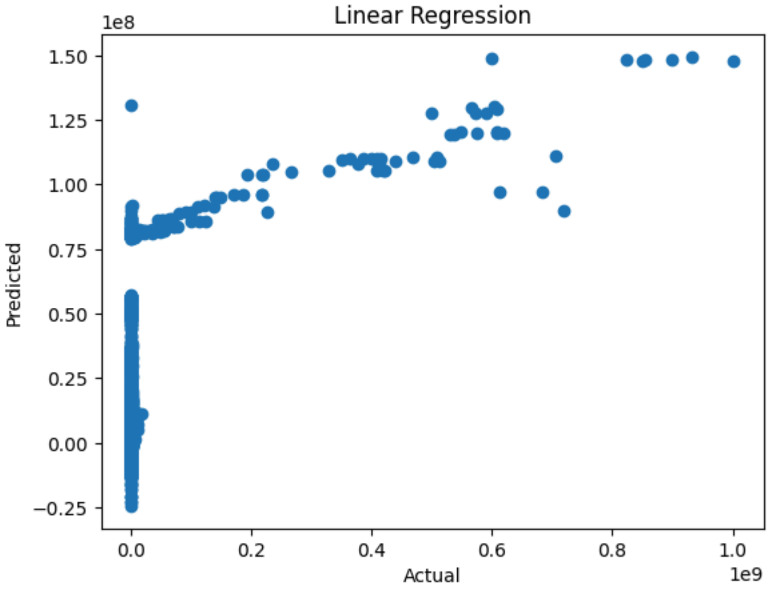
```
0.21903980552344915
```

```
from sklearn.metrics import r2_score
r = r2_score(y_test,lr_predict)
print("R2 score : ",r)
```

```
R2 score :  0.21903980552344915
```

```
plt.scatter(y_test,lr_predict)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Linear Regression')
```

```
Text(0.5, 1.0, 'Linear Regression')
```



Clearly, the dataset is not good for linear regression.

Using the random Forest regressor

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators = 11)
model.fit(x_train,y_train)
rf_predict = model.predict(x_test)
rf_predict
```

```
array([ 277.18181818, 2317.18181818,   15.54545455, ..., 1469.72727273,
        679.72727273, 2196.        ])
```

```
model.score(x_test,y_test)
```

```
0.9811208481033443
```

Using Decision tree

```
# Training model
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 5)
regressor.fit(x_train,y_train)

# Predicting results
decisiontree_predict = regressor.predict(x_test)
decisiontree_predict
```

```
array([ 250., 3000.,    9., ..., 1442.,  473., 2080.])
```

```
regressor.score(x_test,y_test)
```

```
0.9753368569952419
```