```
import pandas as pd
import numpy as np


from google.colab import files
uploaded = files.upload()
```

```
Choose Files No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving crop_production.csv to crop_production.csv
```

```
df = pd.read_csv("crop_production.csv")
print(df)
```

```
                        State_Name District_Name  Crop_Year       Season  \
0       Andaman and Nicobar Islands      NICOBARS       2000  Kharif
1       Andaman and Nicobar Islands      NICOBARS       2000  Kharif
2       Andaman and Nicobar Islands      NICOBARS       2000  Kharif
3       Andaman and Nicobar Islands      NICOBARS       2000  Whole Year
4       Andaman and Nicobar Islands      NICOBARS       2000  Whole Year
...                             ...           ...        ...          ...
246086                  West Bengal       PURULIA       2014  Summer
246087                  West Bengal       PURULIA       2014  Summer
246088                  West Bengal       PURULIA       2014  Whole Year
246089                  West Bengal       PURULIA       2014  Winter
246090                  West Bengal       PURULIA       2014  Winter

                       Crop      Area  Production
0                  Arecanut    1254.0      2000.0
1       Other Kharif pulses       2.0         1.0
2                      Rice     102.0       321.0
3                    Banana     176.0       641.0
4                 Cashewnut     720.0       165.0
...                     ...       ...         ...
246086                 Rice     306.0       801.0
246087              Sesamum     627.0       463.0
246088            Sugarcane     324.0     16250.0
246089                 Rice  279151.0    597899.0
246090              Sesamum     175.0        88.0

[246091 rows x 7 columns]
```

Exploratory data analysis

```
df.shape
```

```
(246091, 7)
```

```
df.columns
```

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

```
df.describe()
```

	Crop_Year	Area	Production
count	246091.000000	2.460910e+05	2.423610e+05
mean	2005.643018	1.200282e+04	5.825034e+05
std	4.952164	5.052340e+04	1.706581e+07
min	1997.000000	4.000000e-02	0.000000e+00
25%	2002.000000	8.000000e+01	8.800000e+01
50%	2006.000000	5.820000e+02	7.290000e+02
75%	2010.000000	4.392000e+03	7.023000e+03
max	2015.000000	8.580100e+06	1.250800e+09

```
print(df['Crop'].unique())      #viewing all the unique crops available in the dataset
```

```
['Arecanut' 'Other Kharif pulses' 'Rice' 'Banana' 'Cashewnut' 'Coconut '
 'Dry ginger' 'Sugarcane' 'Sweet potato' 'Tapioca' 'Black pepper'
 'Dry chillies' 'other oilseeds' 'Turmeric' 'Maize' 'Moong(Green Gram)'
 'Urad' 'Arhar/Tur' 'Groundnut' 'Sunflower' 'Bajra' 'Castor seed'
 'Cotton(lint)' 'Horse-gram' 'Jowar' 'Korra' 'Ragi' 'Tobacco' 'Gram'
```

```
 'Wheat' 'Masoor' 'Sesamum' 'Linseed' 'Safflower' 'Onion'
 'other misc. pulses' 'Samai' 'Small millets' 'Coriander' 'Potato'
 'Other  Rabi pulses' 'Soyabean' 'Beans & Mutter(Vegetable)' 'Bhindi'
 'Brinjal' 'Citrus Fruit' 'Cucumber' 'Grapes' 'Mango' 'Orange'
 'other fibres' 'Other Fresh Fruits' 'Other Vegetables' 'Papaya'
 'Pome Fruit' 'Tomato' 'Rapeseed &Mustard' 'Mesta' 'Cowpea(Lobia)' 'Lemon'
 'Pome Granet' 'Sapota' 'Cabbage' 'Peas  (vegetable)' 'Niger seed'
 'Bottle Gourd' 'Sannhamp' 'Varagu' 'Garlic' 'Ginger' 'Oilseeds total'
 'Pulses total' 'Jute' 'Peas & beans (Pulses)' 'Blackgram' 'Paddy'
 'Pineapple' 'Barley' 'Khesari' 'Guar seed' 'Moth'
 'Other Cereals & Millets' 'Cond-spcs other' 'Turnip' 'Carrot' 'Redish'
 'Arcanut (Processed)' 'Atcanut (Raw)' 'Cashewnut Processed'
 'Cashewnut Raw' 'Cardamom' 'Rubber' 'Bitter Gourd' 'Drum Stick'
 'Jack Fruit' 'Snak Guard' 'Pump Kin' 'Tea' 'Coffee' 'Cauliflower'
 'Other Citrus Fruit' 'Water Melon' 'Total foodgrain' 'Kapas' 'Colocosia'
 'Lentil' 'Bean' 'Jobster' 'Perilla' 'Rajmash Kholar' 'Ricebean (nagadal)'
 'Ash Gourd' 'Beet Root' 'Lab-Lab' 'Ribed Guard' 'Yam' 'Apple' 'Peach'
 'Pear' 'Plums' 'Litchi' 'Ber' 'Other Dry Fruit' 'Jute & mesta']
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 246091 entries, 0 to 246090
Data columns (total 7 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   State_Name     246091 non-null  object
 1   District_Name  246091 non-null  object
 2   Crop_Year      246091 non-null  int64
 3   Season         246091 non-null  object
 4   Crop           246091 non-null  object
 5   Area           246091 non-null  float64
 6   Production     242361 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 13.1+ MB
```

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

	0
State_Name	0
District_Name	0
Crop_Year	0
Season	0
Crop	0
Area	0
Production	3730

dtype: int64

```
df = df.dropna()
print(df)
```

```
                        State_Name District_Name  Crop_Year       Season  \
0       Andaman and Nicobar Islands      NICOBARS       2000  Kharif
1       Andaman and Nicobar Islands      NICOBARS       2000  Kharif
2       Andaman and Nicobar Islands      NICOBARS       2000  Kharif
3       Andaman and Nicobar Islands      NICOBARS       2000  Whole Year
4       Andaman and Nicobar Islands      NICOBARS       2000  Whole Year
...                             ...           ...        ...          ...
246086                West Bengal       PURULIA       2014  Summer
246087                West Bengal       PURULIA       2014  Summer
246088                West Bengal       PURULIA       2014  Whole Year
246089                West Bengal       PURULIA       2014  Winter
246090                West Bengal       PURULIA       2014  Winter

                       Crop      Area  Production
0                  Arecanut    1254.0      2000.0
1       Other Kharif pulses       2.0         1.0
2                      Rice     102.0       321.0
3                    Banana     176.0       641.0
4                 Cashewnut     720.0       165.0
...                     ...       ...         ...
246086                 Rice     306.0       801.0
246087              Sesamum     627.0       463.0
246088            Sugarcane     324.0     16250.0
246089                 Rice  279151.0    597899.0
246090              Sesamum     175.0        88.0

[242361 rows x 7 columns]
```

```
df.isnull().values.any()
```

```
False
```

```
df.State_Name.unique()      #displaying all the unique state in the dataset
```

```
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', 'Maharashtra', 'Manipur',
       'Meghalaya', 'Mizoram', 'Nagaland', 'Odisha', 'Puducherry',
       'Punjab', 'Rajasthan', 'Sikkim', 'Tamil Nadu', 'Telangana ',
       'Tripura', 'Uttar Pradesh', 'Uttarakhand', 'West Bengal'],
      dtype=object)
```

```
#adding a new column yield which gives production per unit area
df['Yield'] = (df['Production'] / df['Area'])
df.head()
```

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	Yield
0	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Arecanut	1254.0	2000.0	1.594896
1	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Other Kharif pulses	2.0	1.0	0.500000
2	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Rice	102.0	321.0	3.147059
3	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Banana	176.0	641.0	3.642045
4	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Cashewnut	720.0	165.0	0.229167

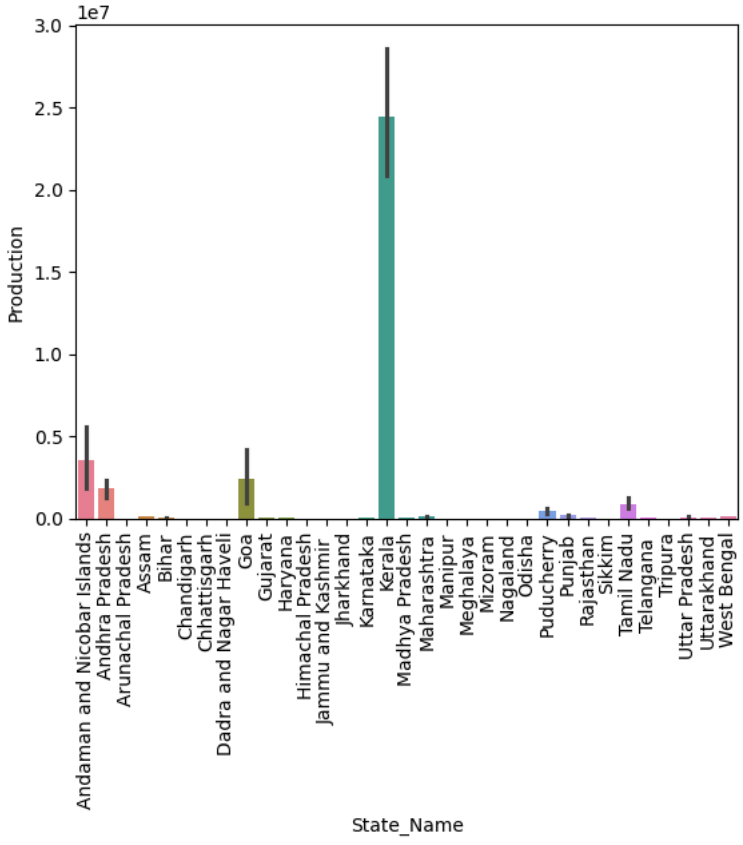
Visualization

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

```
sns.barplot(x=df["State_Name"], y=df["Production"],palette = 'husl')
plt.xticks(rotation=90)
```

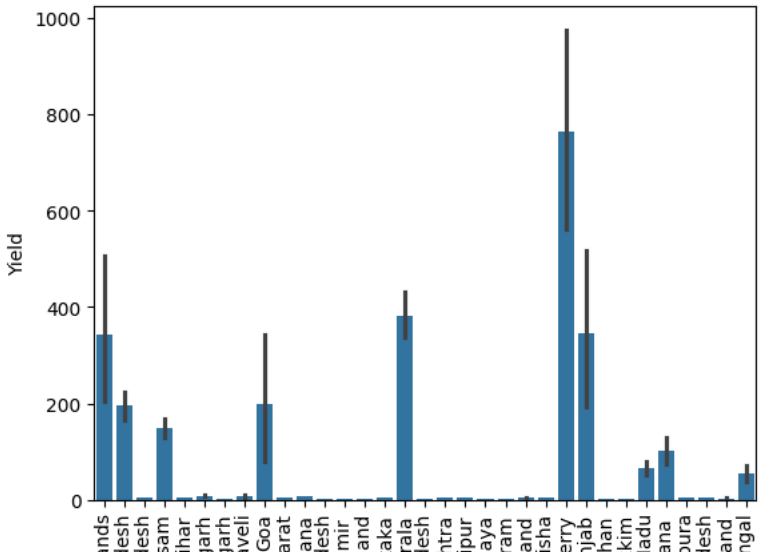
```
<ipython-input-15-15bf409c3d3a>:1: FutureWarning:

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

  sns.barplot(x=df["State_Name"], y=df["Production"],palette = 'husl')
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
```

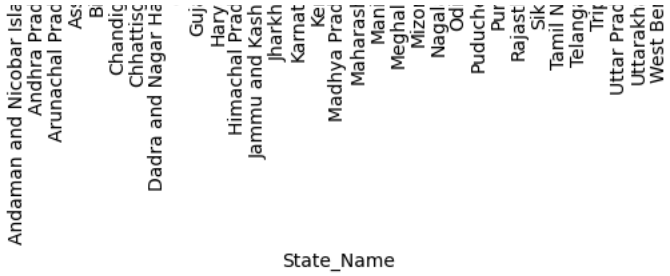
```
 19,
 20,
 21,
 22,
 23,
 24,
 25,
 26,
 27,
 28,
 29,
 30,
 31,
 32],
[Text(0, 0, 'Andaman and Nicobar Islands'),
 Text(1, 0, 'Andhra Pradesh'),
 Text(2, 0, 'Arunachal Pradesh'),
 Text(3, 0, 'Assam'),
 Text(4, 0, 'Bihar'),
 Text(5, 0, 'Chandigarh'),
 Text(6, 0, 'Chhattisgarh'),
 Text(7, 0, 'Dadra and Nagar Haveli'),
 Text(8, 0, 'Goa'),
 Text(9, 0, 'Gujarat'),
 Text(10, 0, 'Haryana'),
 Text(11, 0, 'Himachal Pradesh'),
 Text(12, 0, 'Jammu and Kashmir '),
 Text(13, 0, 'Jharkhand'),
 Text(14, 0, 'Karnataka'),
 Text(15, 0, 'Kerala'),
 Text(16, 0, 'Madhya Pradesh'),
 Text(17, 0, 'Maharashtra'),
 Text(18, 0, 'Manipur'),
 Text(19, 0, 'Meghalaya'),
 Text(20, 0, 'Mizoram'),
 Text(21, 0, 'Nagaland'),
 Text(22, 0, 'Odisha'),
 Text(23, 0, 'Puducherry'),
 Text(24, 0, 'Punjab'),
 Text(25, 0, 'Rajasthan'),
 Text(26, 0, 'Sikkim'),
 Text(27, 0, 'Tamil Nadu'),
 Text(28, 0, 'Telangana '),
 Text(29, 0, 'Tripura'),
 Text(30, 0, 'Uttar Pradesh'),
 Text(31, 0, 'Uttarakhand'),
 Text(32, 0, 'West Bengal')])
```



```
sns.barplot(x = df['State_Name'], y = df['Yield'])
plt.xticks(rotation = 90)
```

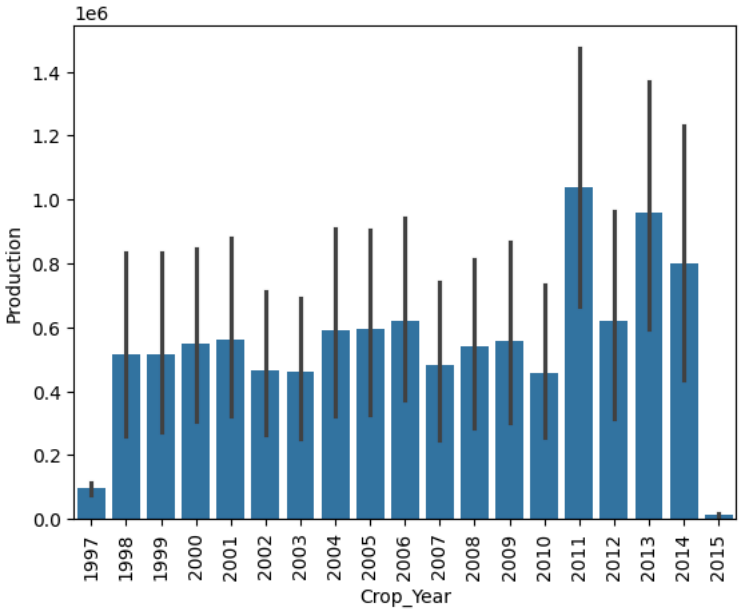
```
([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21,
  22,
  23,
  24,
  25,
  26,
  27,
  28,
  29,
  30,
  31,
  32],
 [Text(0, 0, 'Andaman and Nicobar Islands'),
  Text(1, 0, 'Andhra Pradesh'),
  Text(2, 0, 'Arunachal Pradesh'),
  Text(3, 0, 'Assam'),
  Text(4, 0, 'Bihar'),
  Text(5, 0, 'Chandigarh'),
  Text(6, 0, 'Chhattisgarh'),
  Text(7, 0, 'Dadra and Nagar Haveli'),
  Text(8, 0, 'Goa'),
  Text(9, 0, 'Gujarat'),
  Text(10, 0, 'Haryana'),
  Text(11, 0, 'Himachal Pradesh'),
  Text(12, 0, 'Jammu and Kashmir '),
  Text(13, 0, 'Jharkhand'),
  Text(14, 0, 'Karnataka'),
  Text(15, 0, 'Kerala'),
  Text(16, 0, 'Madhya Pradesh'),
  Text(17, 0, 'Maharashtra'),
  Text(18, 0, 'Manipur'),
  Text(19, 0, 'Meghalaya'),
  Text(20, 0, 'Mizoram'),
  Text(21, 0, 'Nagaland'),
  Text(22, 0, 'Odisha'),
  Text(23, 0, 'Puducherry'),
  Text(24, 0, 'Punjab'),
  Text(25, 0, 'Rajasthan'),
  Text(26, 0, 'Sikkim'),
  Text(27, 0, 'Tamil Nadu'),
  Text(28, 0, 'Telangana '),
  Text(29, 0, 'Tripura'),
  Text(30, 0, 'Uttar Pradesh'),
  Text(31, 0, 'Uttarakhand'),
  Text(32, 0, 'West Bengal')])
```





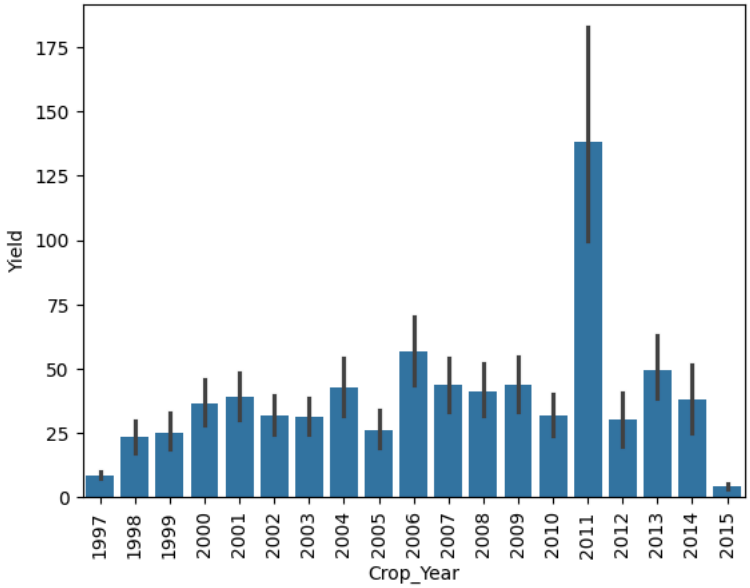

```
sns.barplot(x=df['Crop_Year'], y=df['Production'])
plt.xticks(rotation=90)
```

```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18],
 [Text(0, 0, '1997'),
  Text(1, 0, '1998'),
  Text(2, 0, '1999'),
  Text(3, 0, '2000'),
  Text(4, 0, '2001'),
  Text(5, 0, '2002'),
  Text(6, 0, '2003'),
  Text(7, 0, '2004'),
  Text(8, 0, '2005'),
  Text(9, 0, '2006'),
  Text(10, 0, '2007'),
  Text(11, 0, '2008'),
  Text(12, 0, '2009'),
  Text(13, 0, '2010'),
  Text(14, 0, '2011'),
  Text(15, 0, '2012'),
  Text(16, 0, '2013'),
  Text(17, 0, '2014'),
  Text(18, 0, '2015')])
```



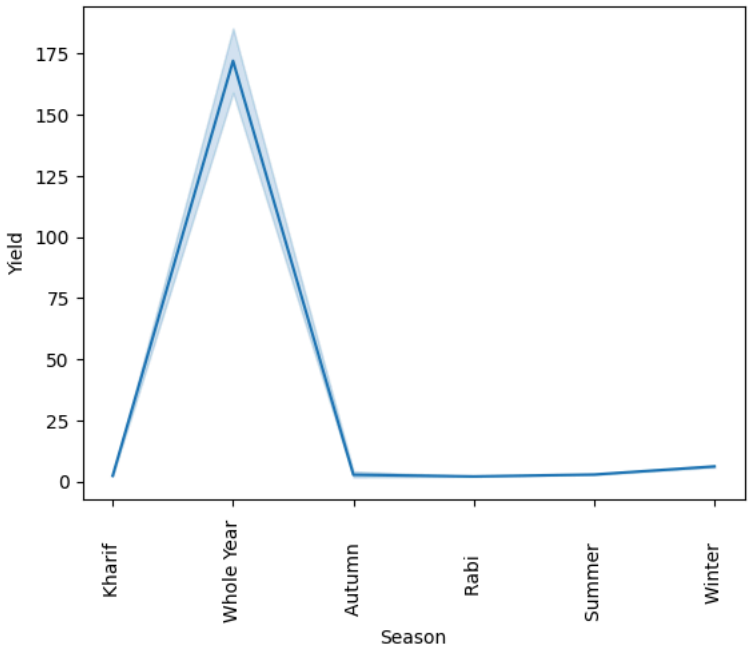

```
sns.barplot(x=df['Crop_Year'], y=df['Yield'])
plt.xticks(rotation=90)
```

```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18],
 [Text(0, 0, '1997'),
  Text(1, 0, '1998'),
  Text(2, 0, '1999'),
  Text(3, 0, '2000'),
  Text(4, 0, '2001'),
  Text(5, 0, '2002'),
  Text(6, 0, '2003'),
  Text(7, 0, '2004'),
  Text(8, 0, '2005'),
  Text(9, 0, '2006'),
  Text(10, 0, '2007'),
  Text(11, 0, '2008'),
  Text(12, 0, '2009'),
  Text(13, 0, '2010'),
  Text(14, 0, '2011'),
  Text(15, 0, '2012'),
  Text(16, 0, '2013'),
  Text(17, 0, '2014'),
  Text(18, 0, '2015')])
```



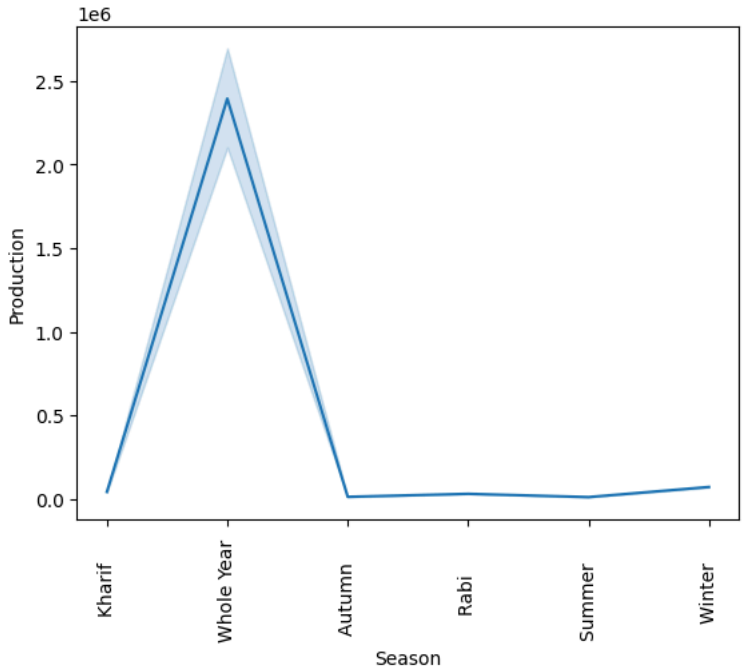

```
sns.lineplot(x=df['Season'],y=df['Yield'])
plt.xticks(rotation=90)
```

```
([0, 1, 2, 3, 4, 5],
 [Text(0, 0, 'Kharif     '),
  Text(1, 0, 'Whole Year '),
  Text(2, 0, 'Autumn     '),
  Text(3, 0, 'Rabi       '),
  Text(4, 0, 'Summer     '),
  Text(5, 0, 'Winter     ')])
```

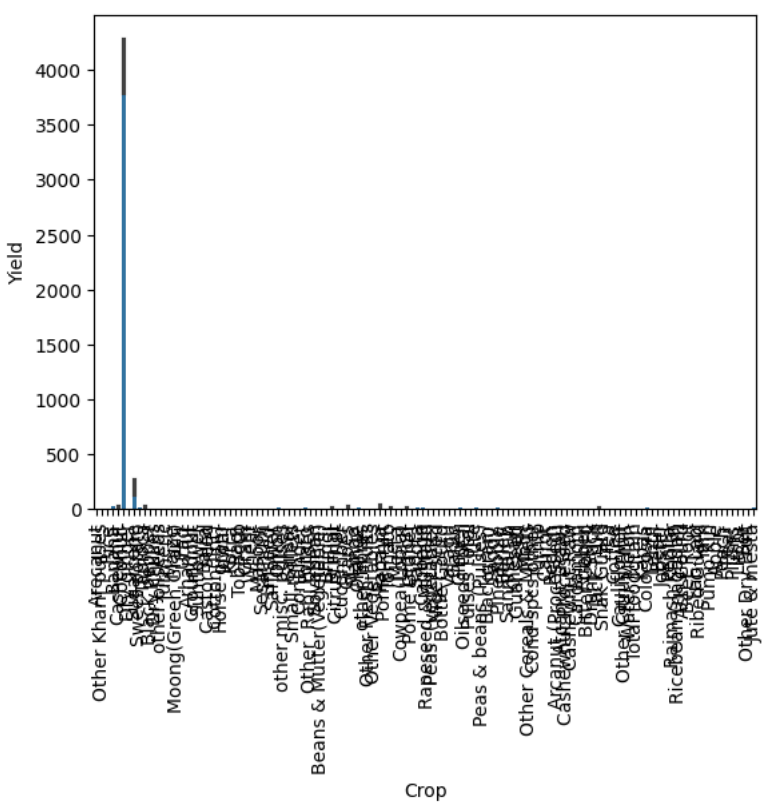


```
sns.lineplot(x=df['Season'],y=df['Production'])
plt.xticks(rotation=90)
```

```
([0, 1, 2, 3, 4, 5],
 [Text(0, 0, 'Kharif     '),
  Text(1, 0, 'Whole Year '),
  Text(2, 0, 'Autumn     '),
  Text(3, 0, 'Rabi       '),
  Text(4, 0, 'Summer     '),
  Text(5, 0, 'Winter     ')])
```




```
sns.barplot(x=df['Crop'],y=df['Yield'])
plt.xticks(rotation=90)
plt.show()
```



CONCLUSIONS

Kerala is the largest producer of crop in INDIA

Production per unit area of Puducherry is maximum

In 2011 , crop yield was good

Analyzing each crop

RICE

```
rice_data = df[df["Crop"]=="Rice"]
rice_data
```

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	Yield
2	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Rice	102.00	321.00	3.147059
12	Andaman and Nicobar Islands	NICOBARS	2001	Kharif	Rice	83.00	300.00	3.614458
18	Andaman and Nicobar Islands	NICOBARS	2002	Kharif	Rice	189.20	510.84	2.700000
27	Andaman and Nicobar Islands	NICOBARS	2003	Kharif	Rice	52.00	90.17	1.734038
36	Andaman and Nicobar Islands	NICOBARS	2004	Kharif	Rice	52.94	72.57	1.370797
...	...	...	...	...	...	...	...	...
246049	West Bengal	PURULIA	2013	Summer	Rice	516.00	1274.00	2.468992
246052	West Bengal	PURULIA	2013	Winter	Rice	302274.00	730136.00	2.415477
246058	West Bengal	PURULIA	2014	Autumn	Rice	264.00	721.00	2.731061
246086	West Bengal	PURULIA	2014	Summer	Rice	306.00	801.00	2.617647
246089	West Bengal	PURULIA	2014	Winter	Rice	279151.00	597899.00	2.141848

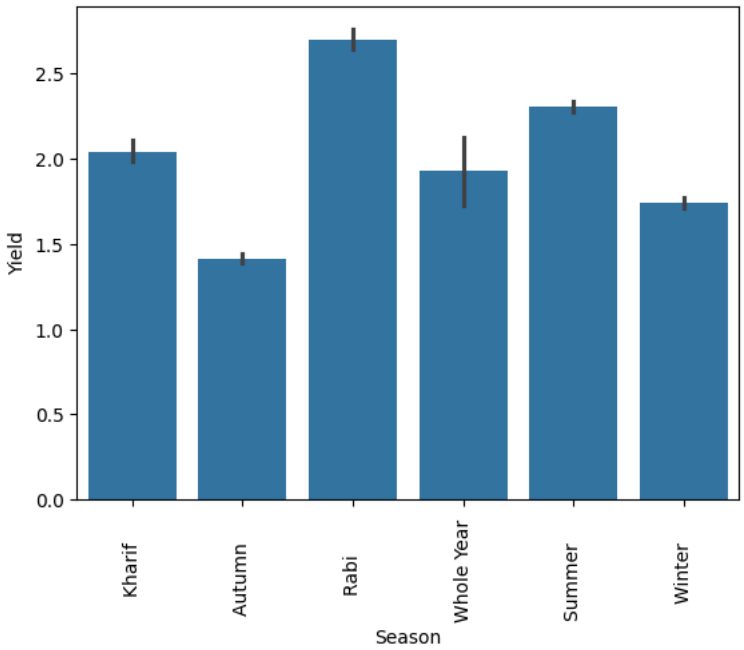
15082 rows × 8 columns

```
df.shape
```

```
(242361, 8)
```

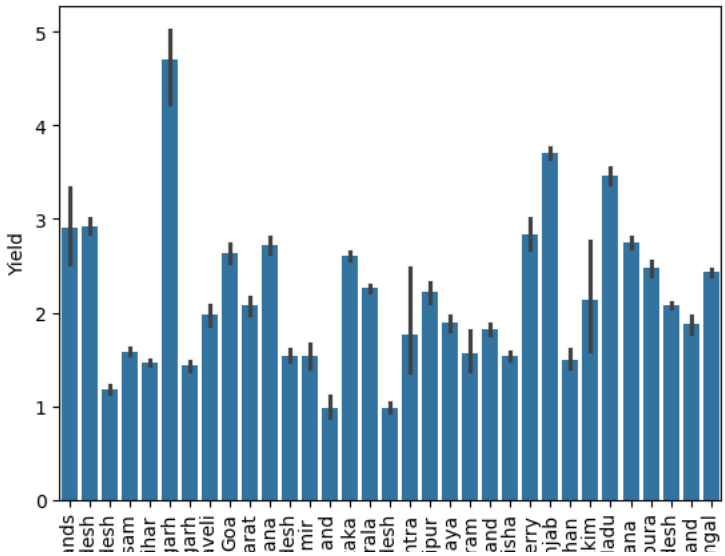
```
sns.barplot(x = "Season",y = "Yield",data = rice_data)
plt.xticks(rotation = 90)
```

```
([0, 1, 2, 3, 4, 5],
 [Text(0, 0, 'Kharif     '),
  Text(1, 0, 'Autumn     '),
  Text(2, 0, 'Rabi       '),
  Text(3, 0, 'Whole Year '),
  Text(4, 0, 'Summer     '),
  Text(5, 0, 'Winter     ')])
```



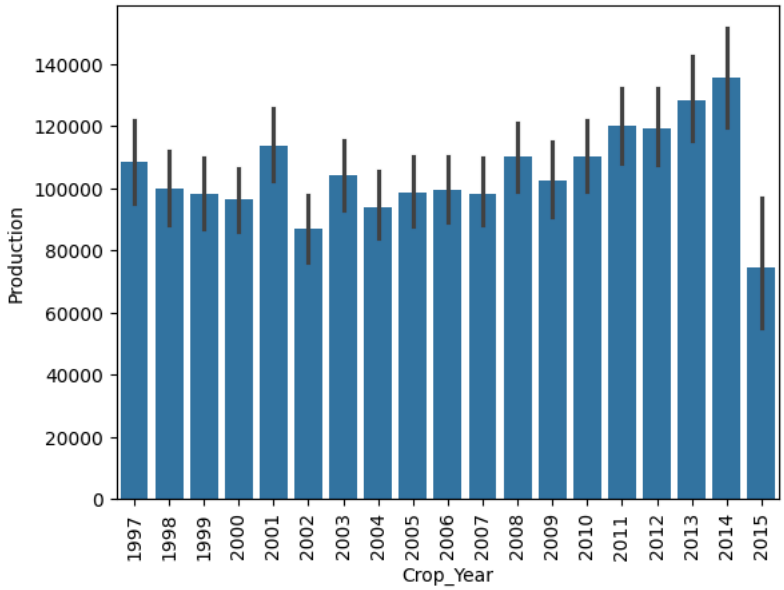

```
sns.barplot(x = "State_Name",y = "Yield",data = rice_data)
plt.xticks(rotation = 90)
```

```
([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21,
  22,
  23,
  24,
  25,
  26,
  27,
  28,
  29,
  30,
  31,
  32],
 [Text(0, 0, 'Andaman and Nicobar Islands'),
  Text(1, 0, 'Andhra Pradesh'),
  Text(2, 0, 'Arunachal Pradesh'),
  Text(3, 0, 'Assam'),
  Text(4, 0, 'Bihar'),
  Text(5, 0, 'Chandigarh'),
  Text(6, 0, 'Chhattisgarh'),
  Text(7, 0, 'Dadra and Nagar Haveli'),
  Text(8, 0, 'Goa'),
  Text(9, 0, 'Gujarat'),
  Text(10, 0, 'Haryana'),
  Text(11, 0, 'Himachal Pradesh'),
  Text(12, 0, 'Jammu and Kashmir '),
  Text(13, 0, 'Jharkhand'),
  Text(14, 0, 'Karnataka'),
  Text(15, 0, 'Kerala'),
  Text(16, 0, 'Madhya Pradesh'),
  Text(17, 0, 'Maharashtra'),
  Text(18, 0, 'Manipur'),
  Text(19, 0, 'Meghalaya'),
  Text(20, 0, 'Mizoram'),
  Text(21, 0, 'Nagaland'),
  Text(22, 0, 'Odisha'),
  Text(23, 0, 'Puducherry'),
  Text(24, 0, 'Punjab'),
  Text(25, 0, 'Rajasthan'),
  Text(26, 0, 'Sikkim'),
  Text(27, 0, 'Tamil Nadu'),
  Text(28, 0, 'Telangana '),
  Text(29, 0, 'Tripura'),
  Text(30, 0, 'Uttar Pradesh'),
  Text(31, 0, 'Uttarakhand'),
  Text(32, 0, 'West Bengal')])
```



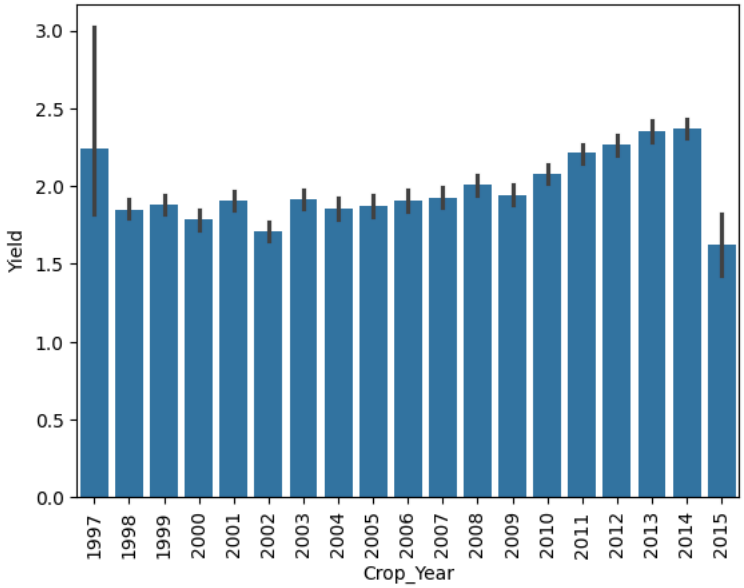

```
sns.barplot(x = "Crop_Year",y = "Production",data=rice_data)
plt.xticks(rotation=90)
```

```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18],
 [Text(0, 0, '1997'),
  Text(1, 0, '1998'),
  Text(2, 0, '1999'),
  Text(3, 0, '2000'),
  Text(4, 0, '2001'),
  Text(5, 0, '2002'),
  Text(6, 0, '2003'),
  Text(7, 0, '2004'),
  Text(8, 0, '2005'),
  Text(9, 0, '2006'),
  Text(10, 0, '2007'),
  Text(11, 0, '2008'),
  Text(12, 0, '2009'),
  Text(13, 0, '2010'),
  Text(14, 0, '2011'),
  Text(15, 0, '2012'),
  Text(16, 0, '2013'),
  Text(17, 0, '2014'),
  Text(18, 0, '2015')])
```



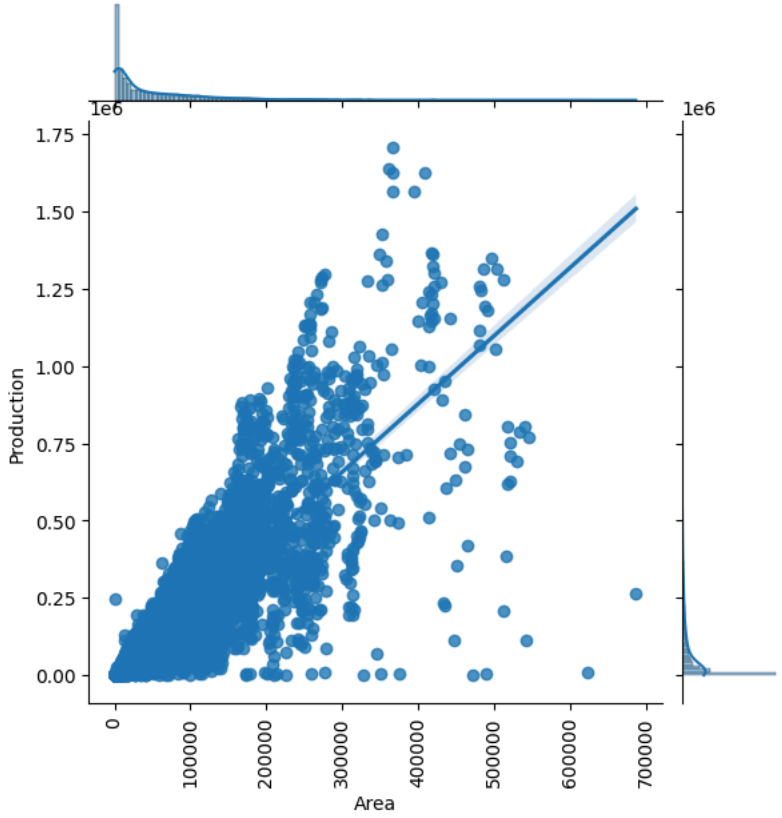

```
sns.barplot(x = "Crop_Year",y = "Yield",data=rice_data)
plt.xticks(rotation=90)
```

```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18],
 [Text(0, 0, '1997'),
  Text(1, 0, '1998'),
  Text(2, 0, '1999'),
  Text(3, 0, '2000'),
  Text(4, 0, '2001'),
  Text(5, 0, '2002'),
  Text(6, 0, '2003'),
  Text(7, 0, '2004'),
  Text(8, 0, '2005'),
  Text(9, 0, '2006'),
  Text(10, 0, '2007'),
  Text(11, 0, '2008'),
  Text(12, 0, '2009'),
  Text(13, 0, '2010'),
  Text(14, 0, '2011'),
  Text(15, 0, '2012'),
  Text(16, 0, '2013'),
  Text(17, 0, '2014'),
  Text(18, 0, '2015')])
```




```
sns.jointplot(x = "Area",y = "Production",data=rice_data,kind="reg")
plt.xticks(rotation=90)
```

```
(array([-100000.,       0.,  100000.,  200000.,  300000.,  400000.,
         500000.,  600000.,  700000.,  800000.]),
 [Text(-100000.0, 0, '−100000'),
  Text(0.0, 0, '0'),
  Text(100000.0, 0, '100000'),
  Text(200000.0, 0, '200000'),
  Text(300000.0, 0, '300000'),
  Text(400000.0, 0, '400000'),
  Text(500000.0, 0, '500000'),
  Text(600000.0, 0, '600000'),
  Text(700000.0, 0, '700000'),
  Text(800000.0, 0, '800000')])
```



Conclusions for rice:- maximum yield in Rabi season and in Chandigarh

WHEAT

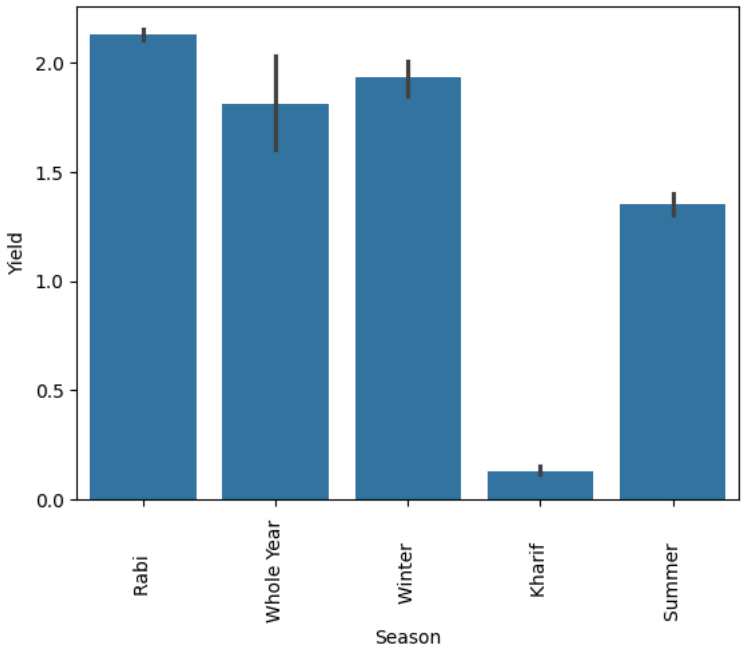
```
wheat_data = df[df["Crop"]=="Wheat"]
wheat_data
```

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	Yield
230	Andhra Pradesh	ANANTAPUR	1997	Rabi	Wheat	300.0	200.0	0.666667
255	Andhra Pradesh	ANANTAPUR	1998	Rabi	Wheat	400.0	200.0	0.500000
284	Andhra Pradesh	ANANTAPUR	1999	Rabi	Wheat	439.0	294.0	0.669704
326	Andhra Pradesh	ANANTAPUR	2000	Rabi	Wheat	520.0	297.0	0.571154
372	Andhra Pradesh	ANANTAPUR	2001	Rabi	Wheat	307.0	213.0	0.693811
...	...	...	...	...	...	...	...	...
245949	West Bengal	PURULIA	2010	Rabi	Wheat	2013.0	5152.0	2.559364
245980	West Bengal	PURULIA	2011	Rabi	Wheat	1880.0	4206.0	2.237234
246012	West Bengal	PURULIA	2012	Rabi	Wheat	1648.0	3310.0	2.008495
246047	West Bengal	PURULIA	2013	Rabi	Wheat	1187.0	2675.0	2.253580
246084	West Bengal	PURULIA	2014	Rabi	Wheat	1622.0	3663.0	2.258323

7878 rows × 8 columns

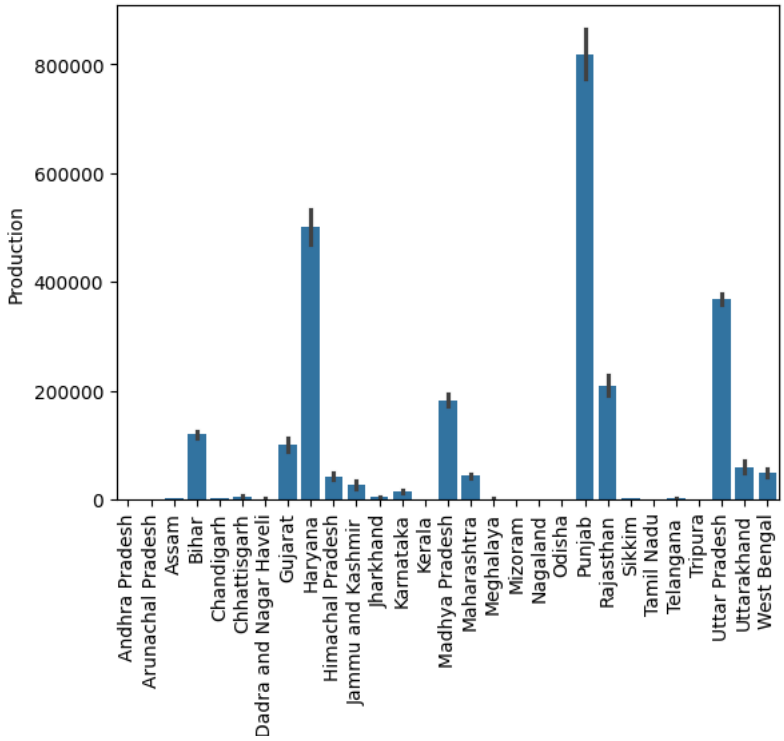
```
sns.barplot(x = "Season",y = "Yield",data = wheat_data)
plt.xticks(rotation = 90)
```

```
([0, 1, 2, 3, 4],
 [Text(0, 0, 'Rabi       '),
  Text(1, 0, 'Whole Year '),
  Text(2, 0, 'Winter     '),
  Text(3, 0, 'Kharif     '),
  Text(4, 0, 'Summer     ')])
```




```
sns.barplot(x="State_Name",y="Production",data = wheat_data)
plt.xticks(rotation = 90)
```

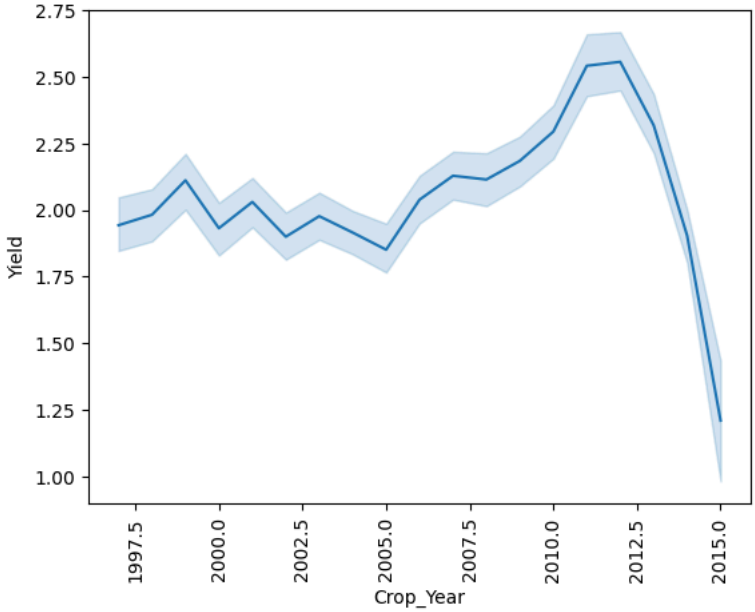
```
([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21,
  22,
  23,
  24,
  25,
  26,
  27,
  28],
 [Text(0, 0, 'Andhra Pradesh'),
  Text(1, 0, 'Arunachal Pradesh'),
  Text(2, 0, 'Assam'),
  Text(3, 0, 'Bihar'),
  Text(4, 0, 'Chandigarh'),
  Text(5, 0, 'Chhattisgarh'),
  Text(6, 0, 'Dadra and Nagar Haveli'),
  Text(7, 0, 'Gujarat'),
  Text(8, 0, 'Haryana'),
  Text(9, 0, 'Himachal Pradesh'),
  Text(10, 0, 'Jammu and Kashmir '),
  Text(11, 0, 'Jharkhand'),
  Text(12, 0, 'Karnataka'),
  Text(13, 0, 'Kerala'),
  Text(14, 0, 'Madhya Pradesh'),
  Text(15, 0, 'Maharashtra'),
  Text(16, 0, 'Meghalaya'),
  Text(17, 0, 'Mizoram'),
  Text(18, 0, 'Nagaland'),
  Text(19, 0, 'Odisha'),
  Text(20, 0, 'Punjab'),
  Text(21, 0, 'Rajasthan'),
  Text(22, 0, 'Sikkim'),
  Text(23, 0, 'Tamil Nadu'),
  Text(24, 0, 'Telangana '),
  Text(25, 0, 'Tripura'),
  Text(26, 0, 'Uttar Pradesh'),
  Text(27, 0, 'Uttarakhand'),
  Text(28, 0, 'West Bengal')])
```



State_Name

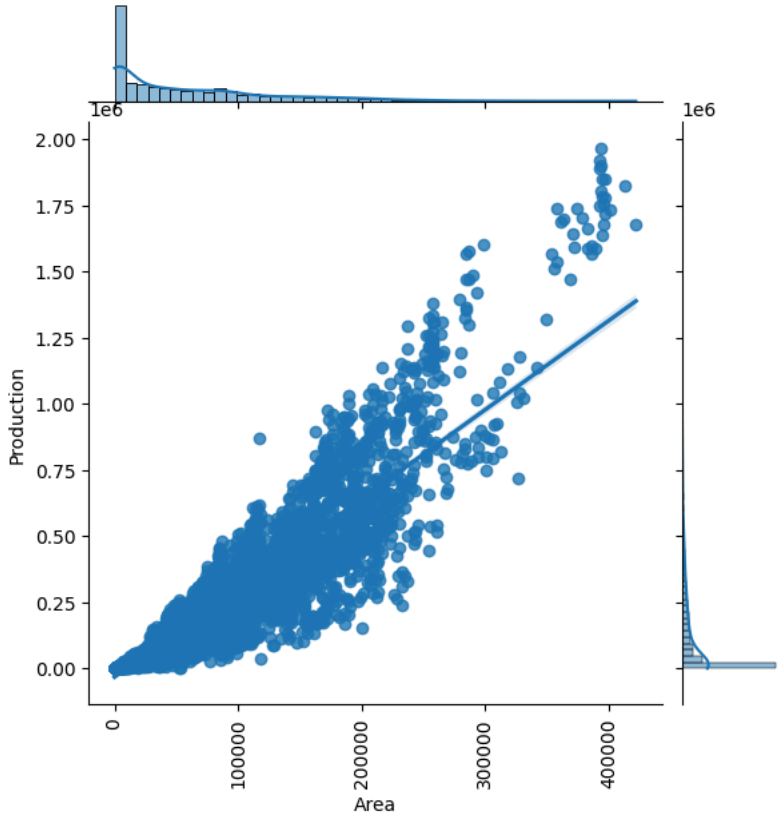
```
sns.lineplot(x = "Crop_Year",y = "Yield",data=wheat_data)
plt.xticks(rotation=90)
```

```
(array([1995. , 1997.5, 2000. , 2002.5, 2005. , 2007.5, 2010. , 2012.5,
        2015. , 2017.5]),
 [Text(1995.0, 0, '1995.0'),
  Text(1997.5, 0, '1997.5'),
  Text(2000.0, 0, '2000.0'),
  Text(2002.5, 0, '2002.5'),
  Text(2005.0, 0, '2005.0'),
  Text(2007.5, 0, '2007.5'),
  Text(2010.0, 0, '2010.0'),
  Text(2012.5, 0, '2012.5'),
  Text(2015.0, 0, '2015.0'),
  Text(2017.5, 0, '2017.5')])
```




```
sns.jointplot(x = "Area",y= "Production",data=wheat_data,kind="reg")
plt.xticks(rotation=90)
```

```
(array([-100000.,       0.,  100000.,  200000.,  300000.,  400000.,
         500000.]),
 [Text(-100000.0, 0, '−100000'),
  Text(0.0, 0, '0'),
  Text(100000.0, 0, '100000'),
  Text(200000.0, 0, '200000'),
  Text(300000.0, 0, '300000'),
  Text(400000.0, 0, '400000'),
  Text(500000.0, 0, '500000')])
```



Conclusions:-wheat is produced maximum in punjab and in rabi season

COCONUT

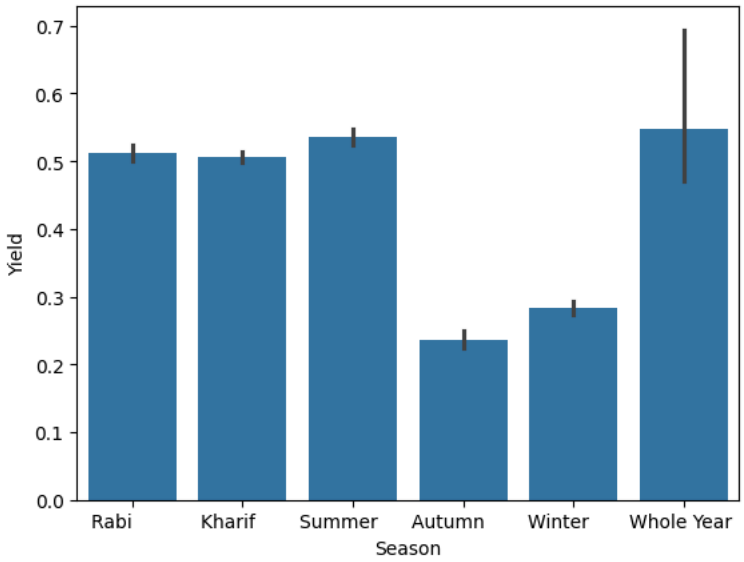
```
coconut_data = df[df["Crop"]=="Urad"]
coconut_data
```

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	Yield
75	Andaman and Nicobar Islands	NICOBARS	2010	Rabi	Urad	1.5	1.16	0.773333
125	Andaman and Nicobar Islands	NORTH AND MIDDLE ANDAMAN	2010	Rabi	Urad	1059.5	458.79	0.433025
199	Andaman and Nicobar Islands	SOUTH ANDAMANS	2010	Rabi	Urad	34.0	15.05	0.442647
270	Andhra Pradesh	ANANTAPUR	1999	Kharif	Urad	4.0	2.00	0.500000
309	Andhra Pradesh	ANANTAPUR	2000	Kharif	Urad	135.0	8.00	0.059259
...	...	...	...	...	...	...	...	...
246000	West Bengal	PURULIA	2012	Kharif	Urad	9258.0	2910.00	0.314323
246011	West Bengal	PURULIA	2012	Rabi	Urad	259.0	133.00	0.513514
246033	West Bengal	PURULIA	2013	Kharif	Urad	12986.0	2877.00	0.221546
246070	West Bengal	PURULIA	2014	Kharif	Urad	11493.0	3287.00	0.286000
246083	West Bengal	PURULIA	2014	Rabi	Urad	220.0	113.00	0.513636

9710 rows × 8 columns

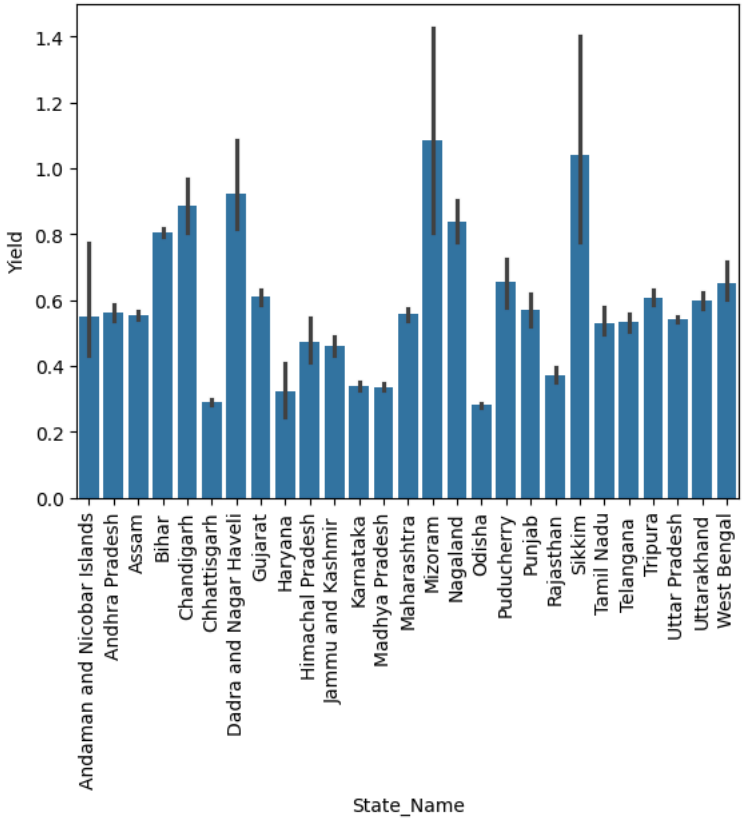
```
sns.barplot(x = "Season",y = "Yield",data = coconut_data)
```

```
<Axes: xlabel='Season', ylabel='Yield'>
```



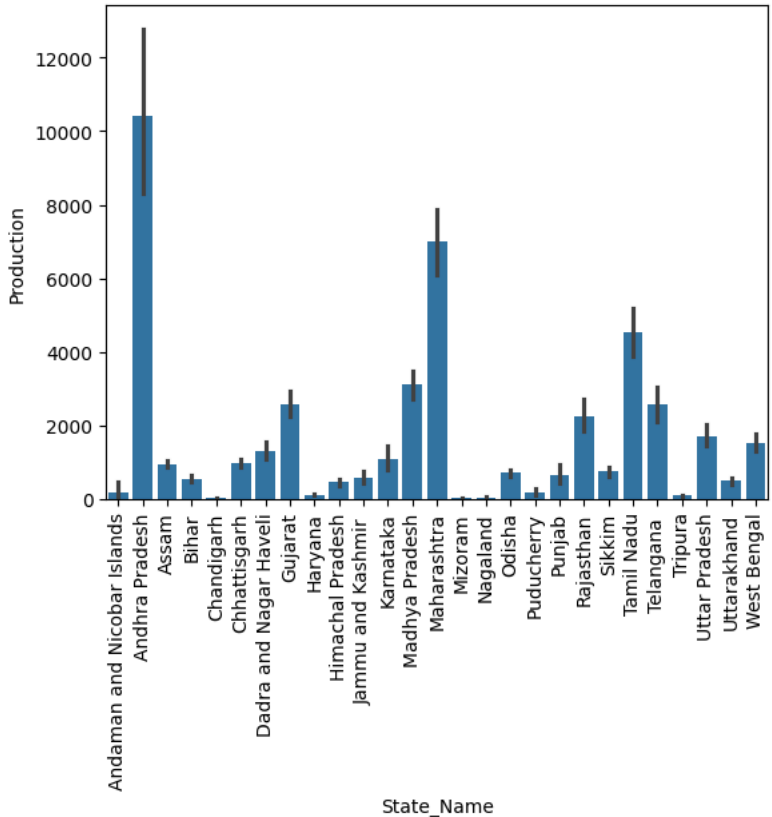

```
sns.barplot(x = "State_Name",y = "Yield",data = coconut_data)
plt.xticks(rotation = 90)
```

```
([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21,
  22,
  23,
  24,
  25,
  26],
 [Text(0, 0, 'Andaman and Nicobar Islands'),
  Text(1, 0, 'Andhra Pradesh'),
  Text(2, 0, 'Assam'),
  Text(3, 0, 'Bihar'),
  Text(4, 0, 'Chandigarh'),
  Text(5, 0, 'Chhattisgarh'),
  Text(6, 0, 'Dadra and Nagar Haveli'),
  Text(7, 0, 'Gujarat'),
  Text(8, 0, 'Haryana'),
  Text(9, 0, 'Himachal Pradesh'),
  Text(10, 0, 'Jammu and Kashmir '),
  Text(11, 0, 'Karnataka'),
  Text(12, 0, 'Madhya Pradesh'),
  Text(13, 0, 'Maharashtra'),
  Text(14, 0, 'Mizoram'),
  Text(15, 0, 'Nagaland'),
  Text(16, 0, 'Odisha'),
  Text(17, 0, 'Puducherry'),
  Text(18, 0, 'Punjab'),
  Text(19, 0, 'Rajasthan'),
  Text(20, 0, 'Sikkim'),
  Text(21, 0, 'Tamil Nadu'),
  Text(22, 0, 'Telangana '),
  Text(23, 0, 'Tripura'),
  Text(24, 0, 'Uttar Pradesh'),
  Text(25, 0, 'Uttarakhand'),
  Text(26, 0, 'West Bengal')])
```



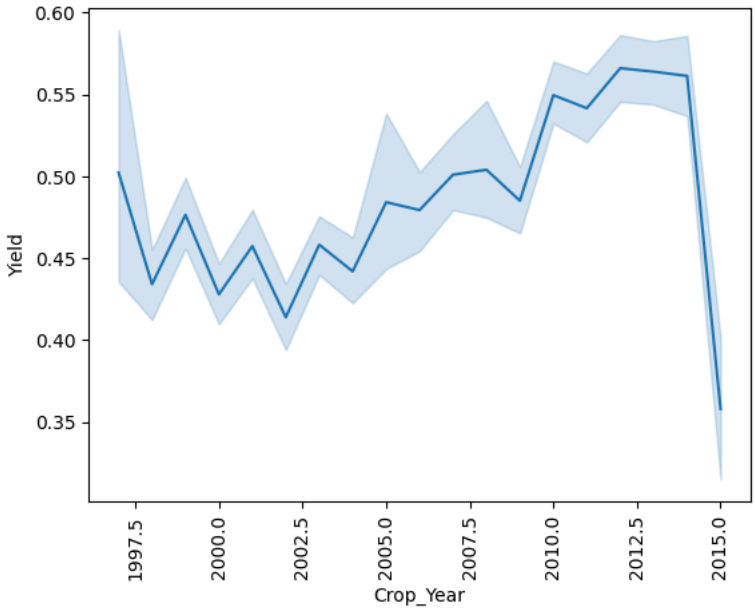
```
sns.barplot(x = "State_Name",y = "Production",data = coconut_data)
plt.xticks(rotation = 90)
```

```
([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21,
  22,
  23,
  24,
  25,
  26],
 [Text(0, 0, 'Andaman and Nicobar Islands'),
  Text(1, 0, 'Andhra Pradesh'),
  Text(2, 0, 'Assam'),
  Text(3, 0, 'Bihar'),
  Text(4, 0, 'Chandigarh'),
  Text(5, 0, 'Chhattisgarh'),
  Text(6, 0, 'Dadra and Nagar Haveli'),
  Text(7, 0, 'Gujarat'),
  Text(8, 0, 'Haryana'),
  Text(9, 0, 'Himachal Pradesh'),
  Text(10, 0, 'Jammu and Kashmir '),
  Text(11, 0, 'Karnataka'),
  Text(12, 0, 'Madhya Pradesh'),
  Text(13, 0, 'Maharashtra'),
  Text(14, 0, 'Mizoram'),
  Text(15, 0, 'Nagaland'),
  Text(16, 0, 'Odisha'),
  Text(17, 0, 'Puducherry'),
  Text(18, 0, 'Punjab'),
  Text(19, 0, 'Rajasthan'),
  Text(20, 0, 'Sikkim'),
  Text(21, 0, 'Tamil Nadu'),
  Text(22, 0, 'Telangana '),
  Text(23, 0, 'Tripura'),
  Text(24, 0, 'Uttar Pradesh'),
  Text(25, 0, 'Uttarakhand'),
  Text(26, 0, 'West Bengal')])
```



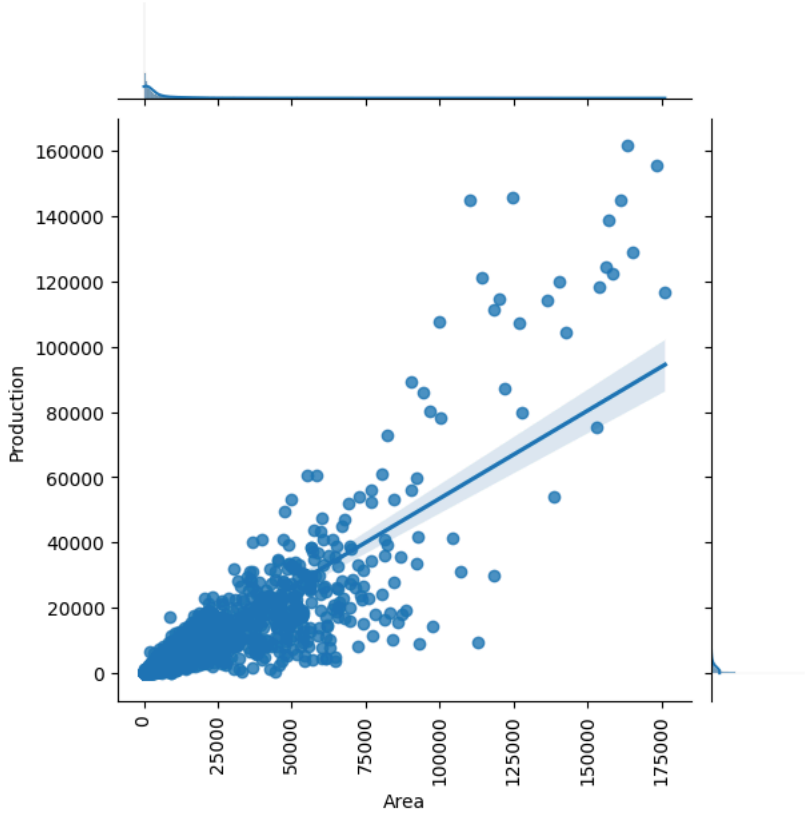
```
sns.lineplot(x = "Crop_Year",y = "Yield",data=coconut_data)
plt.xticks(rotation=90)
```

```
(array([1995. , 1997.5, 2000. , 2002.5, 2005. , 2007.5, 2010. , 2012.5,
        2015. , 2017.5]),
 [Text(1995.0, 0, '1995.0'),
  Text(1997.5, 0, '1997.5'),
  Text(2000.0, 0, '2000.0'),
  Text(2002.5, 0, '2002.5'),
  Text(2005.0, 0, '2005.0'),
  Text(2007.5, 0, '2007.5'),
  Text(2010.0, 0, '2010.0'),
  Text(2012.5, 0, '2012.5'),
  Text(2015.0, 0, '2015.0'),
  Text(2017.5, 0, '2017.5')])
```




```
sns.jointplot(x = "Area",y = "Production",data=coconut_data,kind="reg")
plt.xticks(rotation=90)
```

```
(array([-25000.,      0.,  25000.,  50000.,  75000., 100000., 125000.,
        150000., 175000., 200000.]),
 [Text(-25000.0, 0, '−25000'),
  Text(0.0, 0, '0'),
  Text(25000.0, 0, '25000'),
  Text(50000.0, 0, '50000'),
  Text(75000.0, 0, '75000'),
  Text(100000.0, 0, '100000'),
  Text(125000.0, 0, '125000'),
  Text(150000.0, 0, '150000'),
  Text(175000.0, 0, '175000'),
  Text(200000.0, 0, '200000')])
```



Observations:- Andhra Pradesh is the largest producing coconut state , Production per unit area is higher in Mizoram

POTATO

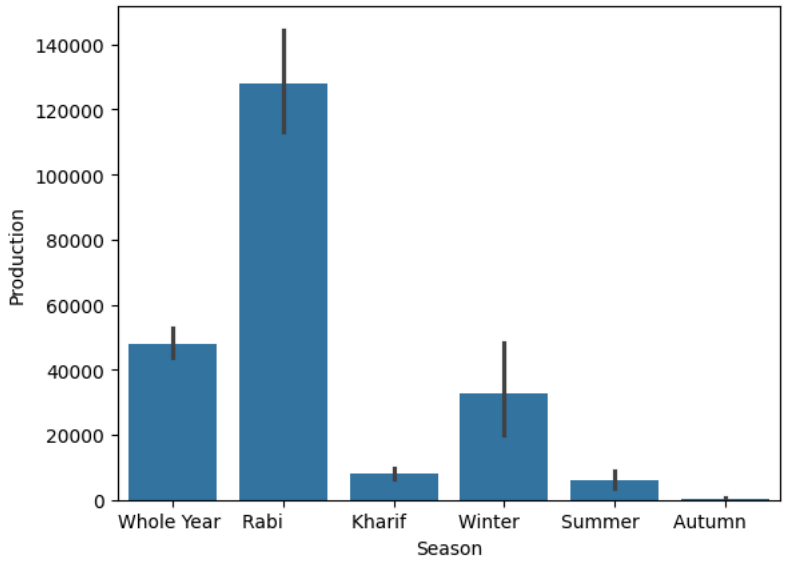
```
potato_data = df[df["Crop"]=="Potato"]
potato_data
```

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	Yield
331	Andhra Pradesh	ANANTAPUR	2000	Whole Year	Potato	4.0	34.0	8.500000
433	Andhra Pradesh	ANANTAPUR	2002	Whole Year	Potato	2.0	17.0	8.500000
530	Andhra Pradesh	ANANTAPUR	2004	Whole Year	Potato	2.0	20.0	10.000000
745	Andhra Pradesh	ANANTAPUR	2010	Whole Year	Potato	21.0	236.0	11.238095
792	Andhra Pradesh	ANANTAPUR	2011	Whole Year	Potato	18.0	181.0	10.055556
...	...	...	...	...	...	...	...	...
245945	West Bengal	PURULIA	2010	Rabi	Potato	977.0	15920.0	16.294780
245976	West Bengal	PURULIA	2011	Rabi	Potato	1077.0	17412.0	16.167131
246008	West Bengal	PURULIA	2012	Rabi	Potato	913.0	17203.0	18.842278
246043	West Bengal	PURULIA	2013	Rabi	Potato	1726.0	43703.0	25.320394
246080	West Bengal	PURULIA	2014	Rabi	Potato	477.0	9995.0	20.953878

6914 rows × 8 columns

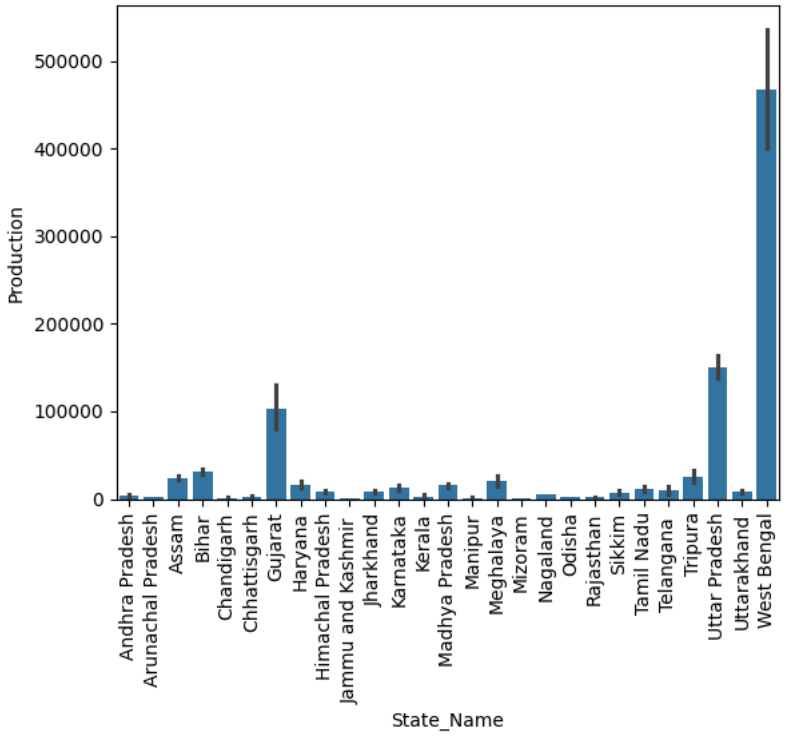
```
sns.barplot(x = "Season",y = "Production",data = potato_data)
```

```
<Axes: xlabel='Season', ylabel='Production'>
```



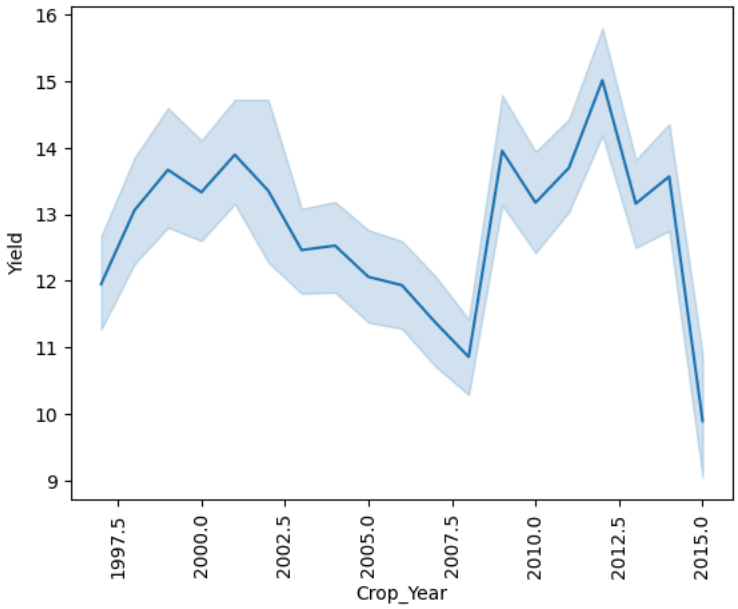

```
sns.barplot(x = "State_Name",y = "Production",data = potato_data)
plt.xticks(rotation = 90)
```

```
([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21,
  22,
  23,
  24,
  25,
  26],
 [Text(0, 0, 'Andhra Pradesh'),
  Text(1, 0, 'Arunachal Pradesh'),
  Text(2, 0, 'Assam'),
  Text(3, 0, 'Bihar'),
  Text(4, 0, 'Chandigarh'),
  Text(5, 0, 'Chhattisgarh'),
  Text(6, 0, 'Gujarat'),
  Text(7, 0, 'Haryana'),
  Text(8, 0, 'Himachal Pradesh'),
  Text(9, 0, 'Jammu and Kashmir '),
  Text(10, 0, 'Jharkhand'),
  Text(11, 0, 'Karnataka'),
  Text(12, 0, 'Kerala'),
  Text(13, 0, 'Madhya Pradesh'),
  Text(14, 0, 'Manipur'),
  Text(15, 0, 'Meghalaya'),
  Text(16, 0, 'Mizoram'),
  Text(17, 0, 'Nagaland'),
  Text(18, 0, 'Odisha'),
  Text(19, 0, 'Rajasthan'),
  Text(20, 0, 'Sikkim'),
  Text(21, 0, 'Tamil Nadu'),
  Text(22, 0, 'Telangana '),
  Text(23, 0, 'Tripura'),
  Text(24, 0, 'Uttar Pradesh'),
  Text(25, 0, 'Uttarakhand'),
  Text(26, 0, 'West Bengal')])
```



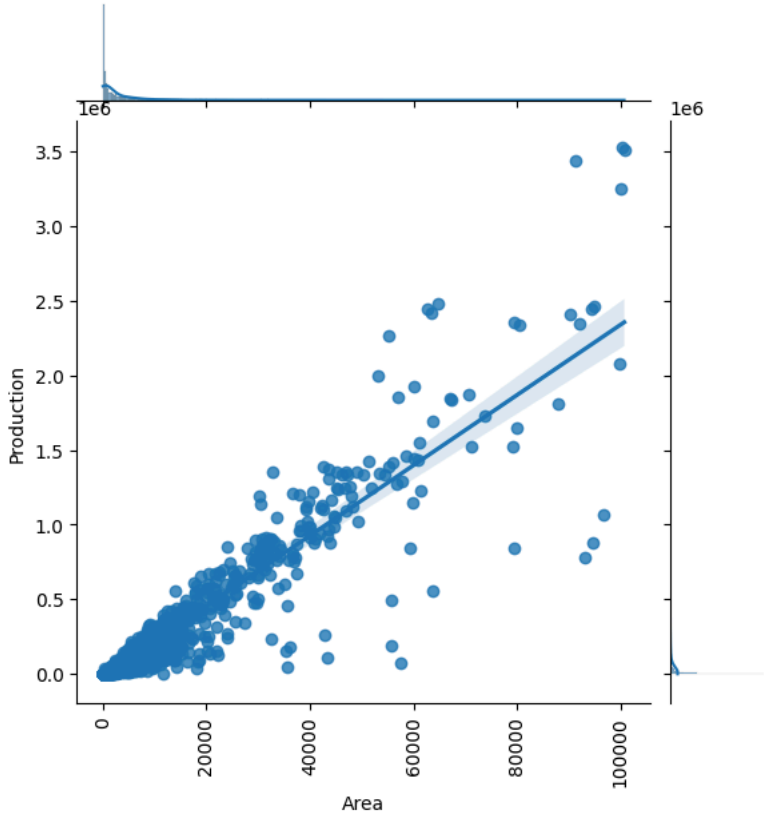
```
sns.lineplot(x = "Crop_Year",y = "Yield",data=potato_data)
plt.xticks(rotation=90)
```

```
(array([1995. , 1997.5, 2000. , 2002.5, 2005. , 2007.5, 2010. , 2012.5,
        2015. , 2017.5]),
 [Text(1995.0, 0, '1995.0'),
  Text(1997.5, 0, '1997.5'),
  Text(2000.0, 0, '2000.0'),
  Text(2002.5, 0, '2002.5'),
  Text(2005.0, 0, '2005.0'),
  Text(2007.5, 0, '2007.5'),
  Text(2010.0, 0, '2010.0'),
  Text(2012.5, 0, '2012.5'),
  Text(2015.0, 0, '2015.0'),
  Text(2017.5, 0, '2017.5')])
```




```
sns.jointplot(x ="Area",y="Production",data=potato_data,kind="reg")
plt.xticks(rotation=90)
```

```
(array([-20000.,      0.,  20000.,  40000.,  60000.,  80000., 100000.,
        120000.]),
 [Text(-20000.0, 0, '-20000'),
  Text(0.0, 0, '0'),
  Text(20000.0, 0, '20000'),
  Text(40000.0, 0, '40000'),
  Text(60000.0, 0, '60000'),
  Text(80000.0, 0, '80000'),
  Text(100000.0, 0, '100000'),
  Text(120000.0, 0, '120000')])
```



Conclusions:-Potato is a rabi crop and west bengal is the largest producer of potatoes

ONION

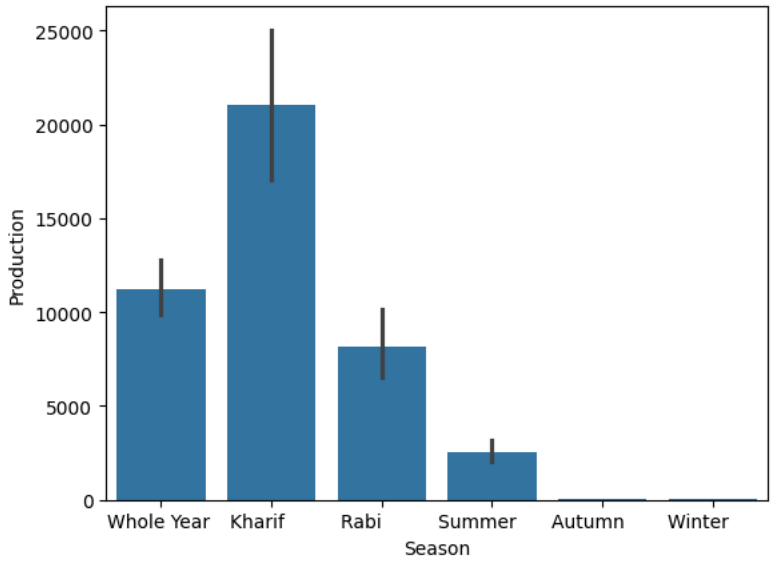
```
onion_data = df[df["Crop"]=="Onion"]
onion_data
```

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	Yield
286	Andhra Pradesh	ANANTAPUR	1999	Whole Year	Onion	1770.0	32364.0	18.284746
300	Andhra Pradesh	ANANTAPUR	2000	Kharif	Onion	1103.0	15470.0	14.025385
320	Andhra Pradesh	ANANTAPUR	2000	Rabi	Onion	482.0	11514.0	23.887967
348	Andhra Pradesh	ANANTAPUR	2001	Kharif	Onion	1165.0	19232.0	16.508155
365	Andhra Pradesh	ANANTAPUR	2001	Rabi	Onion	380.0	8465.0	22.276316
...	...	...	...	...	...	...	...	...
236375	Uttarakhand	UTTAR KASHI	2010	Whole Year	Onion	47.0	306.0	6.510638
236398	Uttarakhand	UTTAR KASHI	2011	Whole Year	Onion	8.0	52.0	6.500000
236420	Uttarakhand	UTTAR KASHI	2012	Whole Year	Onion	23.0	150.0	6.521739
236443	Uttarakhand	UTTAR KASHI	2013	Whole Year	Onion	32.0	212.0	6.625000
236473	Uttarakhand	UTTAR KASHI	2014	Whole Year	Onion	26.0	172.0	6.615385

6984 rows × 8 columns

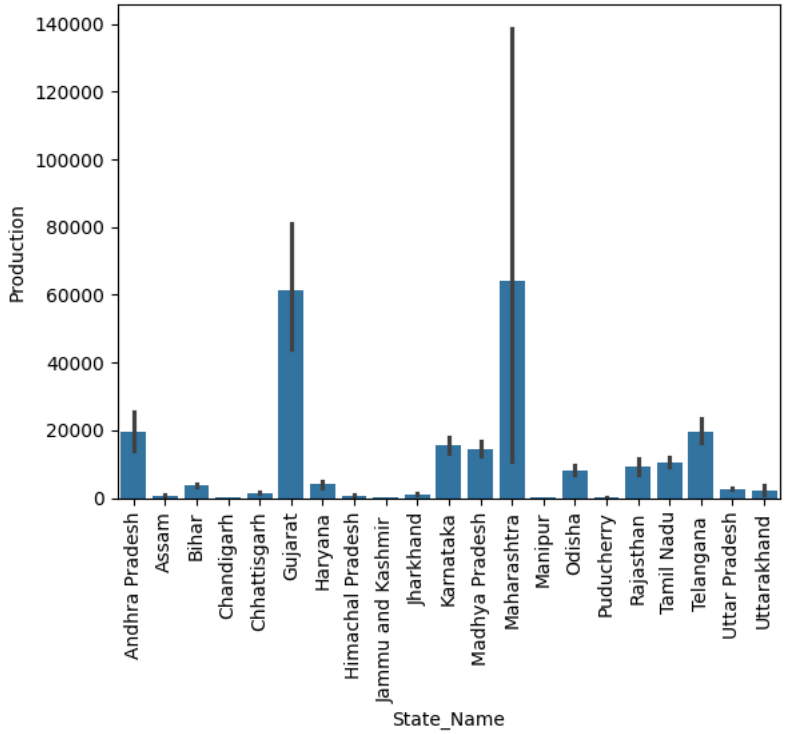
```
sns.barplot(x = "Season",y = "Production",data = onion_data)
```

```
<Axes: xlabel='Season', ylabel='Production'>
```



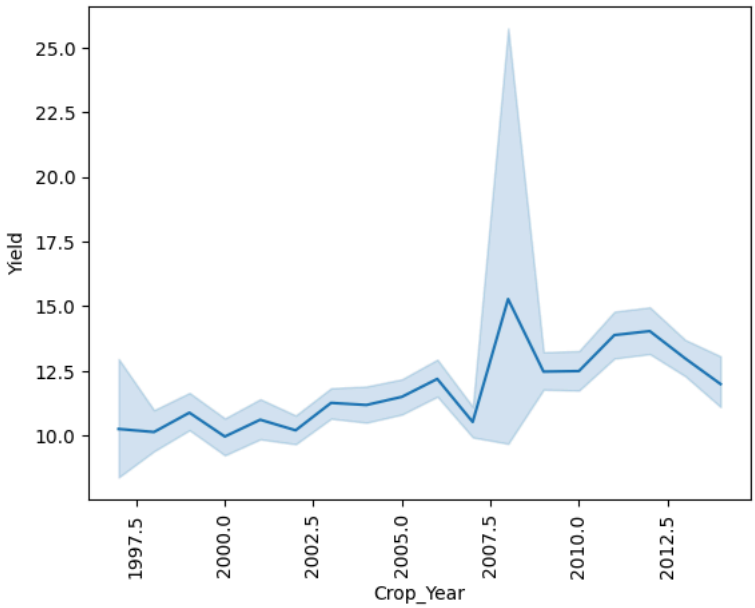

```
sns.barplot(x = "State_Name",y = "Production",data = onion_data)
plt.xticks(rotation = 90)
```

```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20],
 [Text(0, 0, 'Andhra Pradesh'),
  Text(1, 0, 'Assam'),
  Text(2, 0, 'Bihar'),
  Text(3, 0, 'Chandigarh'),
  Text(4, 0, 'Chhattisgarh'),
  Text(5, 0, 'Gujarat'),
  Text(6, 0, 'Haryana'),
  Text(7, 0, 'Himachal Pradesh'),
  Text(8, 0, 'Jammu and Kashmir '),
  Text(9, 0, 'Jharkhand'),
  Text(10, 0, 'Karnataka'),
  Text(11, 0, 'Madhya Pradesh'),
  Text(12, 0, 'Maharashtra'),
  Text(13, 0, 'Manipur'),
  Text(14, 0, 'Odisha'),
  Text(15, 0, 'Puducherry'),
  Text(16, 0, 'Rajasthan'),
  Text(17, 0, 'Tamil Nadu'),
  Text(18, 0, 'Telangana '),
  Text(19, 0, 'Uttar Pradesh'),
  Text(20, 0, 'Uttarakhand')])
```



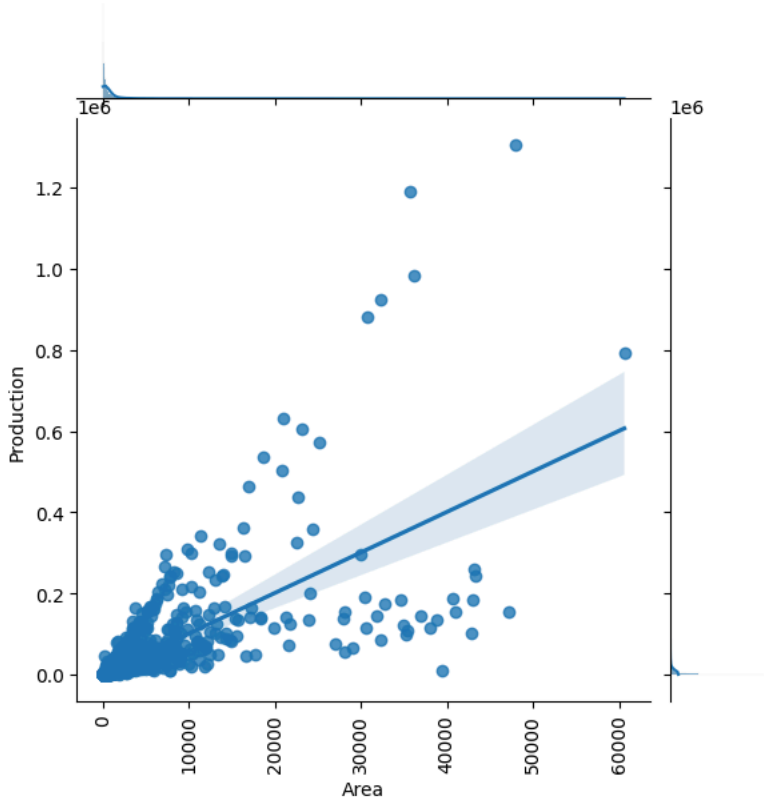
```
sns.lineplot(x = "Crop_Year",y = "Yield",data=onion_data)
plt.xticks(rotation=90)
```

```
(array([1995. , 1997.5, 2000. , 2002.5, 2005. , 2007.5, 2010. , 2012.5,
        2015. ]),
 [Text(1995.0, 0, '1995.0'),
  Text(1997.5, 0, '1997.5'),
  Text(2000.0, 0, '2000.0'),
  Text(2002.5, 0, '2002.5'),
  Text(2005.0, 0, '2005.0'),
  Text(2007.5, 0, '2007.5'),
  Text(2010.0, 0, '2010.0'),
  Text(2012.5, 0, '2012.5'),
  Text(2015.0, 0, '2015.0')])
```




```
sns.jointplot(x = "Area",y = "Production",data=onion_data,kind="reg")
plt.xticks(rotation=90)
```

```
(array([-10000.,      0.,  10000.,  20000.,  30000.,  40000.,  50000.,
         60000.,  70000.]),
 [Text(-10000.0, 0, '-10000'),
  Text(0.0, 0, '0'),
  Text(10000.0, 0, '10000'),
  Text(20000.0, 0, '20000'),
  Text(30000.0, 0, '30000'),
  Text(40000.0, 0, '40000'),
  Text(50000.0, 0, '50000'),
  Text(60000.0, 0, '60000'),
  Text(70000.0, 0, '70000')])
```



Observations:- Onion is a Rabi crop and Gujarat and Maharashtra are the major producer

MAIZE

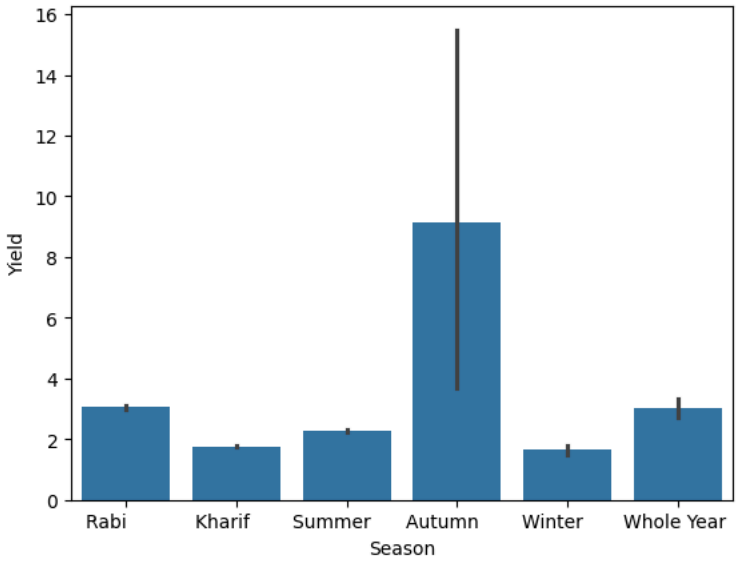
```
maize_data = df[df["Crop"]=="Maize"]
maize_data
```

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	Yield
71	Andaman and Nicobar Islands	NICOBARS	2010	Rabi	Maize	3.84	18.22	4.744792
120	Andaman and Nicobar Islands	NORTH AND MIDDLE ANDAMAN	2010	Rabi	Maize	86.70	96.40	1.111880
194	Andaman and Nicobar Islands	SOUTH ANDAMANS	2010	Rabi	Maize	73.00	253.00	3.465753
212	Andhra Pradesh	ANANTAPUR	1997	Kharif	Maize	2800.00	4900.00	1.750000
226	Andhra Pradesh	ANANTAPUR	1997	Rabi	Maize	600.00	2400.00	4.000000
...	...	...	...	...	...	...	...	...
246013	West Bengal	PURULIA	2012	Summer	Maize	290.00	476.00	1.641379
246020	West Bengal	PURULIA	2013	Autumn	Maize	6189.00	10665.00	1.723219
246048	West Bengal	PURULIA	2013	Summer	Maize	325.00	522.00	1.606154
246056	West Bengal	PURULIA	2014	Autumn	Maize	6317.00	13337.00	2.111287
246085	West Bengal	PURULIA	2014	Summer	Maize	325.00	2039.00	6.273846

13787 rows × 8 columns

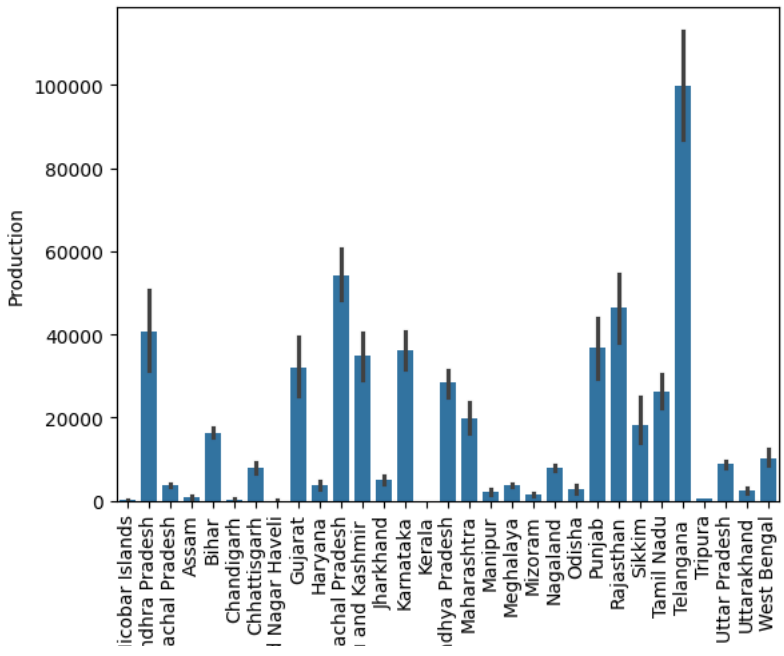
```
sns.barplot(x = "Season",y = "Yield",data = maize_data)
```

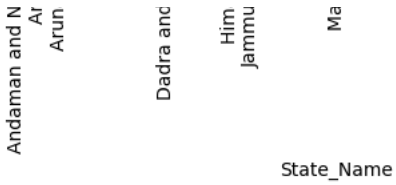
```
<Axes: xlabel='Season', ylabel='Yield'>
```




```
sns.barplot(x = "State_Name",y = "Production",data = maize_data)
plt.xticks(rotation = 90)
```

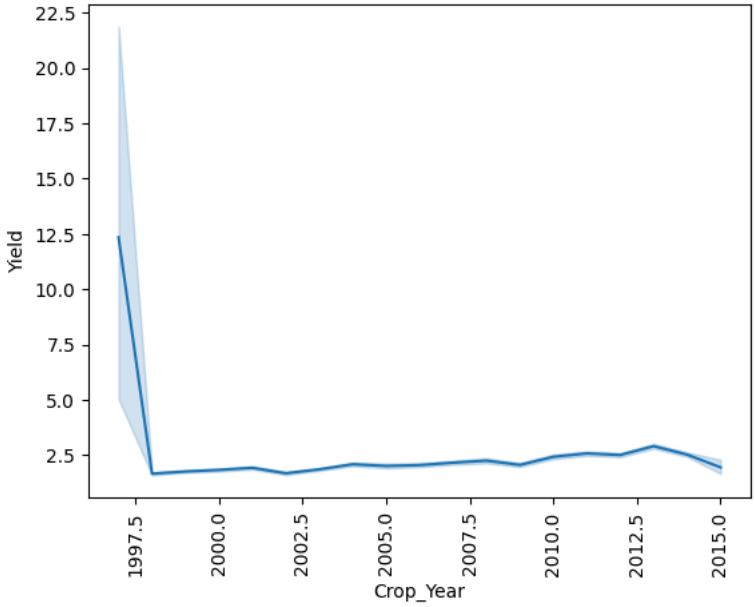
```
([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21,
  22,
  23,
  24,
  25,
  26,
  27,
  28,
  29,
  30],
 [Text(0, 0, 'Andaman and Nicobar Islands'),
  Text(1, 0, 'Andhra Pradesh'),
  Text(2, 0, 'Arunachal Pradesh'),
  Text(3, 0, 'Assam'),
  Text(4, 0, 'Bihar'),
  Text(5, 0, 'Chandigarh'),
  Text(6, 0, 'Chhattisgarh'),
  Text(7, 0, 'Dadra and Nagar Haveli'),
  Text(8, 0, 'Gujarat'),
  Text(9, 0, 'Haryana'),
  Text(10, 0, 'Himachal Pradesh'),
  Text(11, 0, 'Jammu and Kashmir '),
  Text(12, 0, 'Jharkhand'),
  Text(13, 0, 'Karnataka'),
  Text(14, 0, 'Kerala'),
  Text(15, 0, 'Madhya Pradesh'),
  Text(16, 0, 'Maharashtra'),
  Text(17, 0, 'Manipur'),
  Text(18, 0, 'Meghalaya'),
  Text(19, 0, 'Mizoram'),
  Text(20, 0, 'Nagaland'),
  Text(21, 0, 'Odisha'),
  Text(22, 0, 'Punjab'),
  Text(23, 0, 'Rajasthan'),
  Text(24, 0, 'Sikkim'),
  Text(25, 0, 'Tamil Nadu'),
  Text(26, 0, 'Telangana '),
  Text(27, 0, 'Tripura'),
  Text(28, 0, 'Uttar Pradesh'),
  Text(29, 0, 'Uttarakhand'),
  Text(30, 0, 'West Bengal')])
```





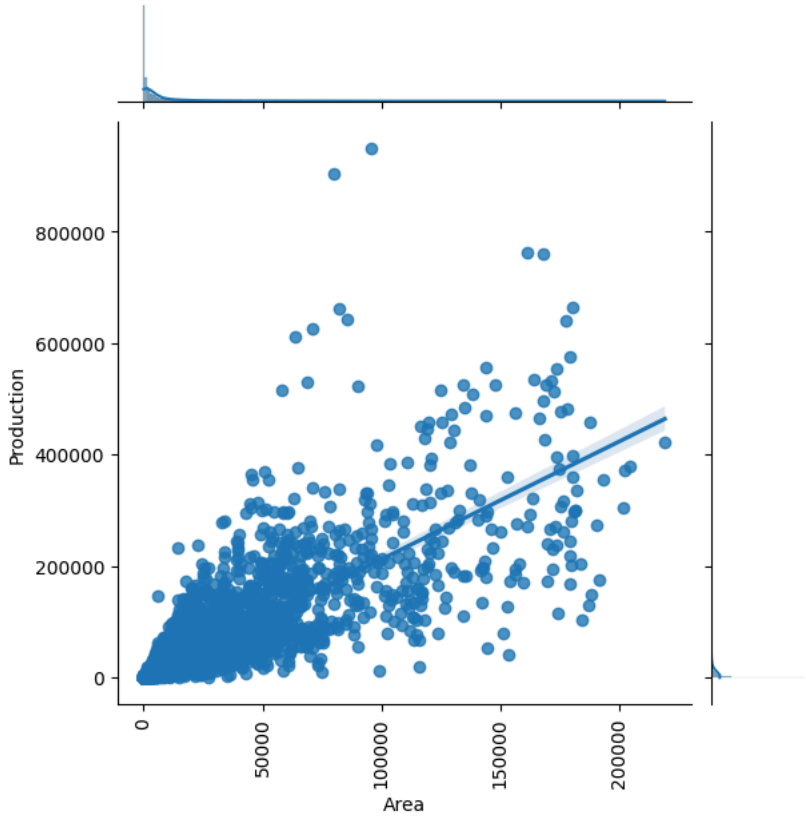

```
sns.lineplot(x = "Crop_Year",y = "Yield",data=maize_data)
plt.xticks(rotation=90)
```

```
(array([1995. , 1997.5, 2000. , 2002.5, 2005. , 2007.5, 2010. , 2012.5,
        2015. , 2017.5]),
 [Text(1995.0, 0, '1995.0'),
  Text(1997.5, 0, '1997.5'),
  Text(2000.0, 0, '2000.0'),
  Text(2002.5, 0, '2002.5'),
  Text(2005.0, 0, '2005.0'),
  Text(2007.5, 0, '2007.5'),
  Text(2010.0, 0, '2010.0'),
  Text(2012.5, 0, '2012.5'),
  Text(2015.0, 0, '2015.0'),
  Text(2017.5, 0, '2017.5')])
```




```
sns.jointplot(x = "Area",y = "Production",data=maize_data,kind="reg")
plt.xticks(rotation=90)
```

```
(array([-50000.,      0.,  50000., 100000., 150000., 200000., 250000.]),
 [Text(-50000.0, 0, '-50000'),
  Text(0.0, 0, '0'),
  Text(50000.0, 0, '50000'),
  Text(100000.0, 0, '100000'),
  Text(150000.0, 0, '150000'),
  Text(200000.0, 0, '200000'),
  Text(250000.0, 0, '250000')])
```



Observations:- Maize is produced in autumn season and Telangana is the major maize-producing state

RECOMMENDATION SYSTEM

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

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Crop_recommendation.csv to Crop_recommendation (1).csv

```
df = pd.read_csv("Crop_recommendation.csv")
print(df)
```

```
         N   P   K  temperature   humidity        ph    rainfall   label
0       90  42  43    20.879744  82.002744  6.502985  202.935536    rice
1       85  58  41    21.770462  80.319644  7.038096  226.655537    rice
2       60  55  44    23.004459  82.320763  7.840207  263.964248    rice
3       74  35  40    26.491096  80.158363  6.980401  242.864034    rice
4       78  42  42    20.130175  81.604873  7.628473  262.717340    rice
...    ...  ..  ..          ...        ...       ...         ...     ...
2195   107  34  32    26.774637  66.413269  6.780064  177.774507  coffee
2196    99  15  27    27.417112  56.636362  6.086922  127.924610  coffee
2197   118  33  30    24.131797  67.225123  6.362608  173.322839  coffee
2198   117  32  34    26.272418  52.127394  6.758793  127.175293  coffee
2199   104  18  30    23.603016  60.396475  6.779833  140.937041  coffee

[2200 rows x 8 columns]
```

```
df.shape
```

```
(2200, 8)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   N            2200 non-null   int64  
 1   P            2200 non-null   int64  
 2   K            2200 non-null   int64  
 3   temperature  2200 non-null   float64
 4   humidity     2200 non-null   float64
 5   ph           2200 non-null   float64
 6   rainfall     2200 non-null   float64
 7   label        2200 non-null   object 
dtypes: float64(4), int64(3), object(1)
memory usage: 137.6+ KB
```

```
df.columns
```

```
Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label'], dtype='object')
```

```
df.rename(columns = {'label':"Crop"},inplace = True)
df
```

	N	P	K	temperature	humidity	ph	rainfall	Crop
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice
...	...	...	...	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	coffee

2200 rows × 8 columns

```
df.describe()
```

	N	P	K	temperature	humidity	ph	rainfall
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000
mean	50.551818	53.362727	48.149091	25.616244	71.481779	6.469480	103.463655
std	36.917334	32.985883	50.647931	5.063749	22.263812	0.773938	54.958389
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	20.211267
25%	21.000000	28.000000	20.000000	22.769375	60.261953	5.971693	64.551686
50%	37.000000	51.000000	32.000000	25.598693	80.473146	6.425045	94.867624
75%	84.250000	68.000000	49.000000	28.561654	89.948771	6.923643	124.267508
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298.560117

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

	0
N	0
P	0
K	0
temperature	0
humidity	0
ph	0
rainfall	0
Crop	0

dtype: int64

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

	N	P	K	temperature	humidity	ph	rainfall	Crop
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice
...	...	...	...	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	coffee

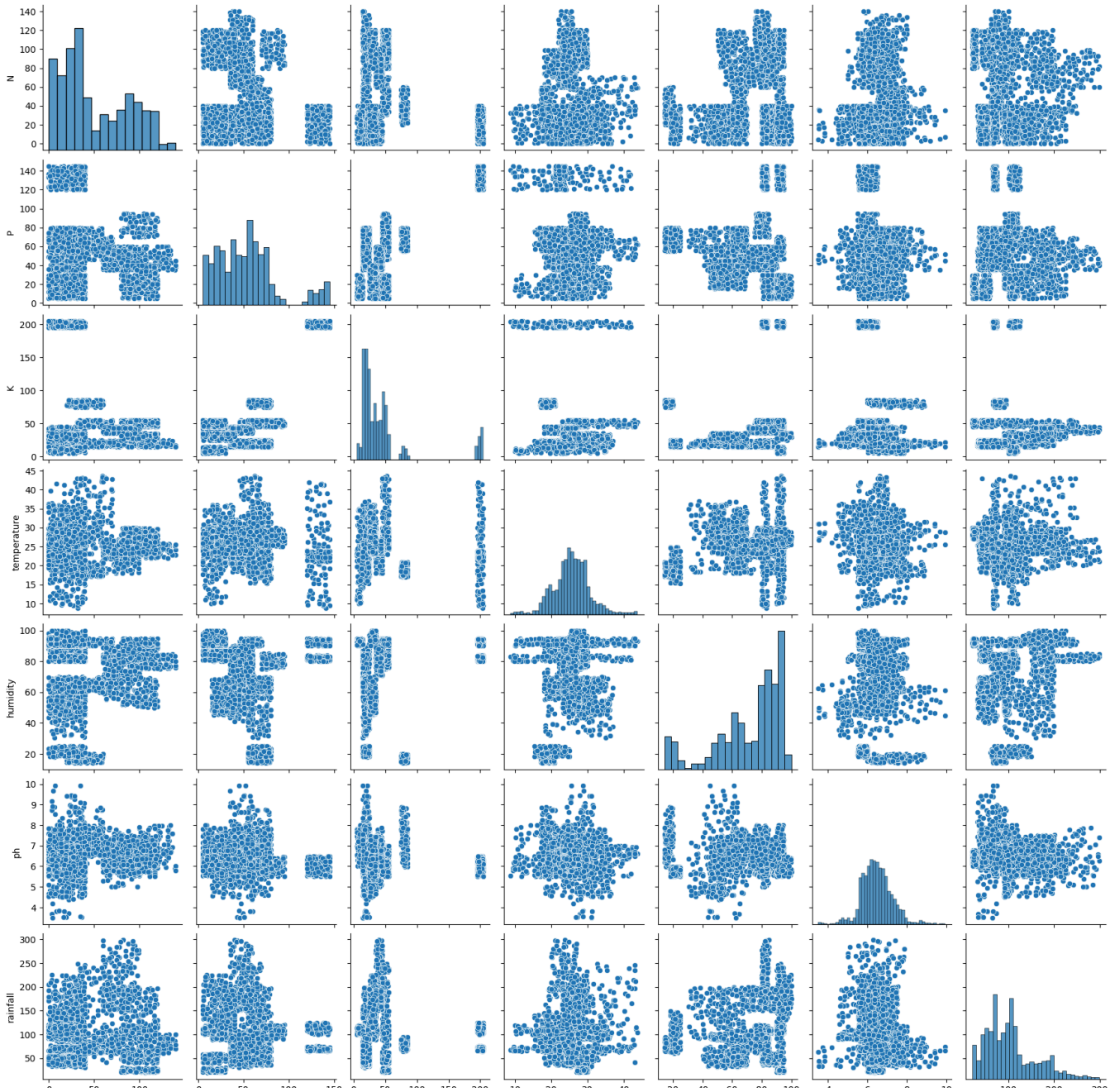
2200 rows × 8 columns

```
df.isnull().values.any()
```

```
False
```

```
ax = sns.pairplot(df)
ax
```

```
<seaborn.axisgrid.PairGrid at 0x7cbf240b7160>
```




```
df.Crop.unique()
```

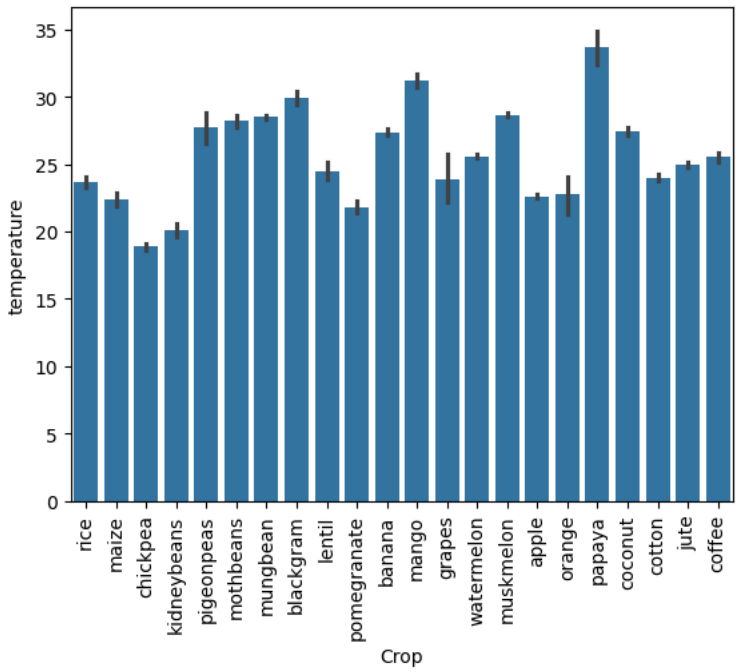
```
array(['rice', 'maize', 'chickpea', 'kidneybeans', 'pigeonpeas',
       'mothbeans', 'mungbean', 'blackgram', 'lentil', 'pomegranate',
       'banana', 'mango', 'grapes', 'watermelon', 'muskmelon', 'apple',
       'orange', 'papaya', 'coconut', 'cotton', 'jute', 'coffee'],
      dtype=object)
```

```
n = 5
df['Crop'].value_counts()[:5].index.tolist()
```

```
['rice', 'maize', 'jute', 'cotton', 'coconut']
```

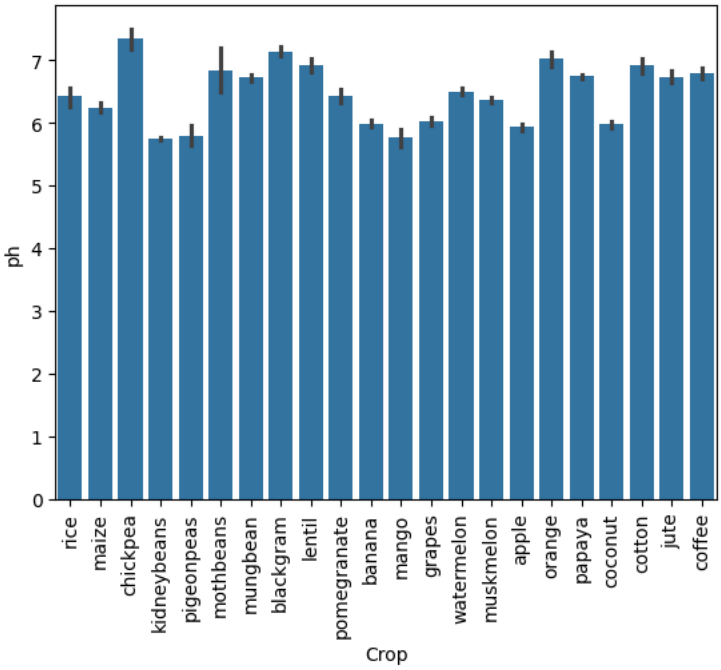
```
sns.barplot(x = df["Crop"],y = df["temperature"])
plt.xticks(rotation = 90)
```

```
([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21],
 [Text(0, 0, 'rice'),
  Text(1, 0, 'maize'),
  Text(2, 0, 'chickpea'),
  Text(3, 0, 'kidneybeans'),
  Text(4, 0, 'pigeonpeas'),
  Text(5, 0, 'mothbeans'),
  Text(6, 0, 'mungbean'),
  Text(7, 0, 'blackgram'),
  Text(8, 0, 'lentil'),
  Text(9, 0, 'pomegranate'),
  Text(10, 0, 'banana'),
  Text(11, 0, 'mango'),
  Text(12, 0, 'grapes'),
  Text(13, 0, 'watermelon'),
  Text(14, 0, 'muskmelon'),
  Text(15, 0, 'apple'),
  Text(16, 0, 'orange'),
  Text(17, 0, 'papaya'),
  Text(18, 0, 'coconut'),
  Text(19, 0, 'cotton'),
  Text(20, 0, 'jute'),
  Text(21, 0, 'coffee')])
```



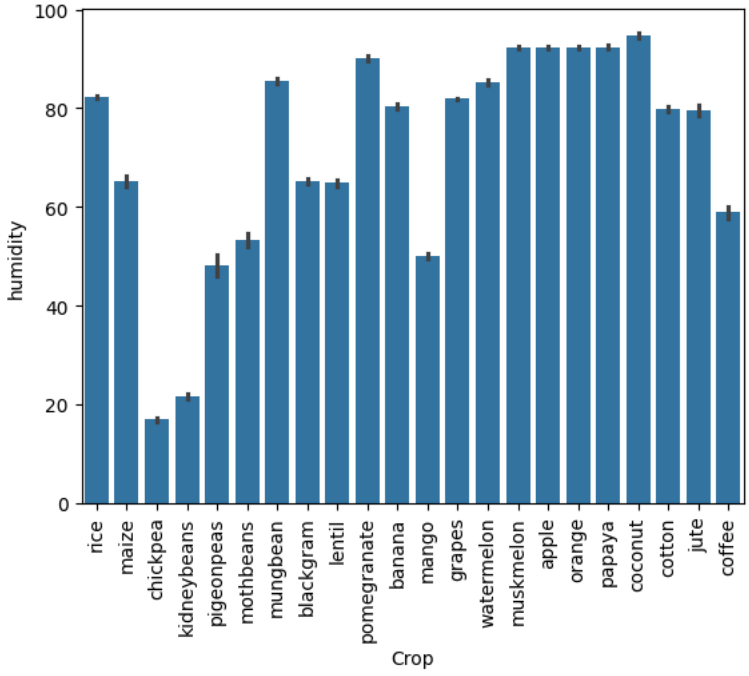

```
sns.barplot(x = df["Crop"],y = df["ph"])
plt.xticks(rotation = 90)
```

```
([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21],
 [Text(0, 0, 'rice'),
  Text(1, 0, 'maize'),
  Text(2, 0, 'chickpea'),
  Text(3, 0, 'kidneybeans'),
  Text(4, 0, 'pigeonpeas'),
  Text(5, 0, 'mothbeans'),
  Text(6, 0, 'mungbean'),
  Text(7, 0, 'blackgram'),
  Text(8, 0, 'lentil'),
  Text(9, 0, 'pomegranate'),
  Text(10, 0, 'banana'),
  Text(11, 0, 'mango'),
  Text(12, 0, 'grapes'),
  Text(13, 0, 'watermelon'),
  Text(14, 0, 'muskmelon'),
  Text(15, 0, 'apple'),
  Text(16, 0, 'orange'),
  Text(17, 0, 'papaya'),
  Text(18, 0, 'coconut'),
  Text(19, 0, 'cotton'),
  Text(20, 0, 'jute'),
  Text(21, 0, 'coffee')])
```



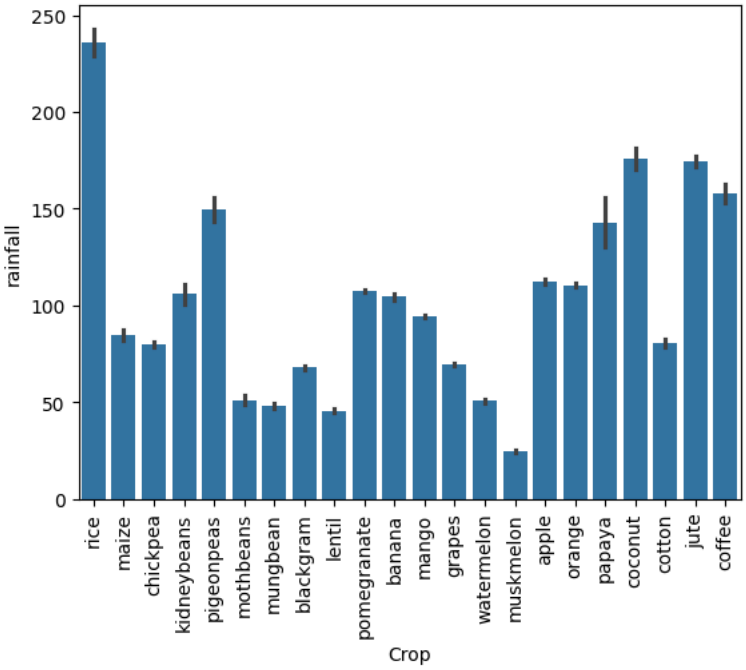

```
sns.barplot(x = df["Crop"],y = df["humidity"])
plt.xticks(rotation = 90)
```

```
([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21],
 [Text(0, 0, 'rice'),
  Text(1, 0, 'maize'),
  Text(2, 0, 'chickpea'),
  Text(3, 0, 'kidneybeans'),
  Text(4, 0, 'pigeonpeas'),
  Text(5, 0, 'mothbeans'),
  Text(6, 0, 'mungbean'),
  Text(7, 0, 'blackgram'),
  Text(8, 0, 'lentil'),
  Text(9, 0, 'pomegranate'),
  Text(10, 0, 'banana'),
  Text(11, 0, 'mango'),
  Text(12, 0, 'grapes'),
  Text(13, 0, 'watermelon'),
  Text(14, 0, 'muskmelon'),
  Text(15, 0, 'apple'),
  Text(16, 0, 'orange'),
  Text(17, 0, 'papaya'),
  Text(18, 0, 'coconut'),
  Text(19, 0, 'cotton'),
  Text(20, 0, 'jute'),
  Text(21, 0, 'coffee')])
```




```
sns.barplot(x = df["Crop"],y = df["rainfall"])
plt.xticks(rotation = 90)
```

```
([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21],
 [Text(0, 0, 'rice'),
  Text(1, 0, 'maize'),
  Text(2, 0, 'chickpea'),
  Text(3, 0, 'kidneybeans'),
  Text(4, 0, 'pigeonpeas'),
  Text(5, 0, 'mothbeans'),
  Text(6, 0, 'mungbean'),
  Text(7, 0, 'blackgram'),
  Text(8, 0, 'lentil'),
  Text(9, 0, 'pomegranate'),
  Text(10, 0, 'banana'),
  Text(11, 0, 'mango'),
  Text(12, 0, 'grapes'),
  Text(13, 0, 'watermelon'),
  Text(14, 0, 'muskmelon'),
  Text(15, 0, 'apple'),
  Text(16, 0, 'orange'),
  Text(17, 0, 'papaya'),
  Text(18, 0, 'coconut'),
  Text(19, 0, 'cotton'),
  Text(20, 0, 'jute'),
  Text(21, 0, 'coffee')])
```

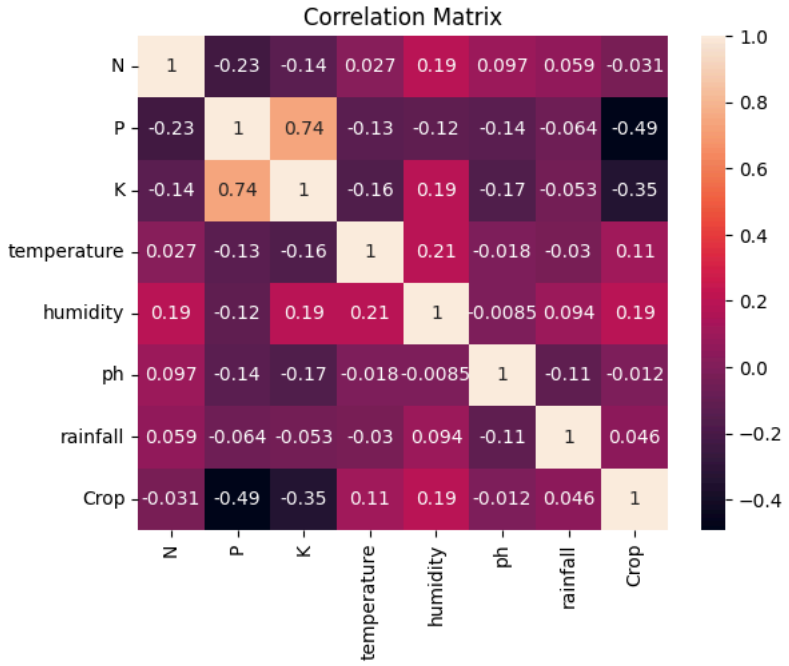



```
df["Crop"] = df["Crop"].astype('category')
df.corr(numeric_only=True)
```

	N	P	K	temperature	humidity	ph	rainfall
N	1.000000	-0.231460	-0.140512	0.026504	0.190688	0.096683	0.059020
P	-0.231460	1.000000	0.736232	-0.127541	-0.118734	-0.138019	-0.063839
K	-0.140512	0.736232	1.000000	-0.160387	0.190859	-0.169503	-0.053461
temperature	0.026504	-0.127541	-0.160387	1.000000	0.205320	-0.017795	-0.030084
humidity	0.190688	-0.118734	0.190859	0.205320	1.000000	-0.008483	0.094423
ph	0.096683	-0.138019	-0.169503	-0.017795	-0.008483	1.000000	-0.109069
rainfall	0.059020	-0.063839	-0.053461	-0.030084	0.094423	-0.109069	1.000000

```
df["Crop"] = df["Crop"].cat.codes # convert categorical data to numerical representation
sns.heatmap(df.corr(), annot =True)
plt.title('Correlation Matrix')
```

```
Text(0.5, 1.0, 'Correlation Matrix')
```




```
from sklearn.utils import shuffle

df  = shuffle(df,random_state=5)
df.head()
```

	N	P	K	temperature	humidity	ph	rainfall	Crop
1270	6	140	205	17.665584	82.929034	6.313086	69.867126	7
1481	98	22	47	29.072653	91.915332	6.341401	28.835684	15
1832	38	14	30	26.924495	91.201060	5.570745	194.902214	4
293	35	63	76	17.815645	17.607566	7.714153	90.820976	3
1307	85	22	53	25.965342	89.770767	6.849472	59.463386	21

```
# Selection of Feature and Target variables.

x = df[['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']]
target = df['Crop']


# Encoding target variable

y = pd.get_dummies(target)
y
```

	0	1	2	3	4	5	6	7	8	9	...	12	13	14	15	16	17	18	19	20	
1270	False	False	False	False	False	False	False	True	False	False	...	False	False	False	False	False	False	False	False	False	F
1481	False	False	False	False	False	False	False	False	False	False	...	False	False	False	True	False	False	False	False	False	F
1832	False	False	False	False	True	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	F
293	False	False	False	True	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	F
1307	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
740	False	False	True	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	F
1032	False	True	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	F
2121	False	False	False	False	False	True	False	False	False	False	...	False	False	False	False	False	False	False	False	False	F
1424	False	False	False	False	False	False	False	False	False	False	...	False	False	False	True	False	False	False	False	False	F
1725	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	True	False	False	False	F

2200 rows × 22 columns

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


from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25, random_state= 0)

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 : (1650, 7)
x_test : (550, 7)
y_train : (1650, 22)
y_test : (550, 22)
```

```
# Importing necessary libraries for multi-output classification

from sklearn.datasets import make_classification
from sklearn.multioutput import MultiOutputClassifier
from sklearn.ensemble import RandomForestClassifier


# Training

forest = RandomForestClassifier(random_state=1)
multi_target_forest = MultiOutputClassifier(forest, n_jobs=-1)
multi_target_forest.fit(x_train, y_train)
```

```
MultiOutputClassifier
estimator: RandomForestClassifier
RandomForestClassifier
```

```
# Predicting test results

forest_pred = multi_target_forest.predict(x_test)
forest_pred
```

```
array([[False, False, False, ..., False, False, False],
       [False, False, False, ..., False,  True, False],
       [False, False, False, ..., False, False, False],
       ...,
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False,  True, False],
       [False, False, False, ..., False, False, False]])
```

```
# Calculating Accuracy

from sklearn.metrics import accuracy_score
```

```
a1 = accuracy_score(y_test, forest_pred)
print('Accuracy score:', accuracy_score(y_test, forest_pred))
```

```
Accuracy score: 0.98
```

Cross Validation

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(multi_target_forest,X = x_train, y = y_train,cv=5)
score
```

```
array([0.97575758, 0.96666667, 0.95454545, 0.96666667, 0.96969697])
```

```
b1 = "{:.2f}".format(score.mean()*100)
b1 = float(b1)
b1
```

```
96.67
```

```
c1 = (score.std()*100)
c1
```

```
0.6910154091509904
```

```
print("Accuracy : {:.2f}%".format (score.mean()*100))
print("Standard Deviation : {:.2f}%".format(score.std()*100))
```

```
Accuracy : 96.67%
Standard Deviation : 0.69%
```

```
# Training
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(random_state=6)
multi_target_decision = MultiOutputClassifier(clf, n_jobs=-1)
multi_target_decision.fit(x_train, y_train)
```

```
MultiOutputClassifier
estimator: DecisionTreeClassifier
DecisionTreeClassifier
```

```
# Predicting test results

decision_pred = multi_target_decision.predict(x_test)
decision_pred
```

```
array([[False, False, False, ..., False, False, False],
       [False, False, False, ..., False,  True, False],
       [False, False, False, ..., False, False, False],
       ...,
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False,  True, False],
       [False, False, False, ..., False, False, False]])
```

```
# Calculating Accuracy

from sklearn.metrics import accuracy_score
a2 = accuracy_score(y_test,decision_pred)
print('Accuracy score:', accuracy_score(y_test,decision_pred))
a2
```

```
Accuracy score: 0.9436363636363636
0.9436363636363636
```

Cross - validation

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(multi_target_decision,X = x_train, y = y_train,cv=7)
score
```

```
array([0.88135593, 0.91525424, 0.90677966, 0.93220339, 0.92372881,
       0.96595745, 0.94468085])
```

```
b2 = "{:.2f}".format(score.mean()*100)
b2 = float(b2)
b2
```

```
92.43
```

```
c2 = (score.std()*100)
c2
```

```
2.5203429649690308
```

```
from sklearn.neighbors import KNeighborsClassifier

knn_clf=KNeighborsClassifier()
model = MultiOutputClassifier(knn_clf, n_jobs=-1)
model.fit(x_train, y_train)
```

```
MultiOutputClassifier
estimator: KNeighborsClassifier
KNeighborsClassifier
```

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

```
array([[False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False],
       ...,
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False,  True, False],
       [False, False, False, ..., False, False, False]])
```

```
# Calculating Accuracy

from sklearn.metrics import accuracy_score
a3 = accuracy_score(y_test,knn_pred)
print('Accuracy score:', accuracy_score(y_test,knn_pred))
a3
```

```
Accuracy score: 0.9745454545454545
0.9745454545454545
```

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(model,X = x_train, y = y_train,cv=7)
score
```

```
array([0.99152542, 0.97457627, 0.97881356, 0.97457627, 0.98305085,
       0.9787234 , 0.99148936])
```

```
b3 = "{:.2f}".format(score.mean()*100)
b3 = float(b3)
b3
```

```
98.18
```

```
c3 = (score.std()*100)
c3
```

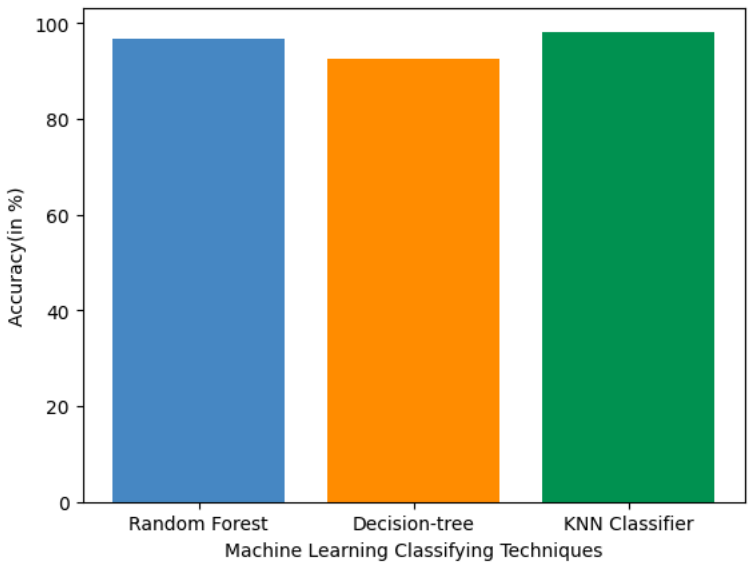
```
0.668449766475269
```

```
import pandas as pd

# initialise data of lists.
data = {'Algorithms':['Random Forest', 'Decision-tree', 'KNN Classifier'],
        'Accuracy':[b1, b2, b3],
        'Standard Deviation':[c1,c2,c3]}
```

```
# Creates pandas DataFrame.
df = pd.DataFrame(data)

# print the data
df
```

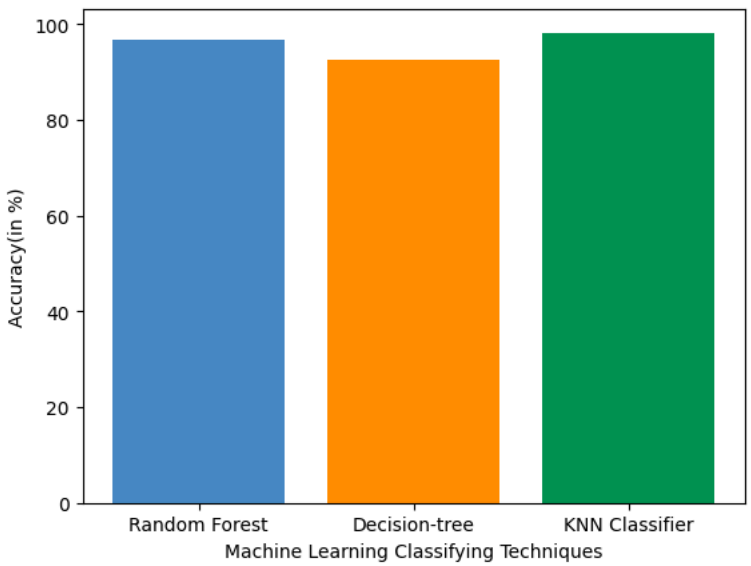
	Algorithms	Accuracy	Standard Deviation
0	Random Forest	96.67	0.691015
1	Decision-tree	92.43	2.520343
2	KNN Classifier	98.18	0.668450

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

# create a dataset
Algorithms = ['Random Forest', 'Decision-tree','KNN Classifier']
Accuracy = [b1, b2, b3]

x_pos = np.arange(len(Accuracy))

# Create bars with different colors
plt.bar(x_pos, Accuracy, color=['#488AC7','#ff8c00','#009150'])

# Create names on the x-axis
plt.xticks(x_pos, Algorithms)
plt.ylabel('Accuracy(in %)')
plt.xlabel('Machine Learning Classifying Techniques')

# Show graph
plt.show()
```

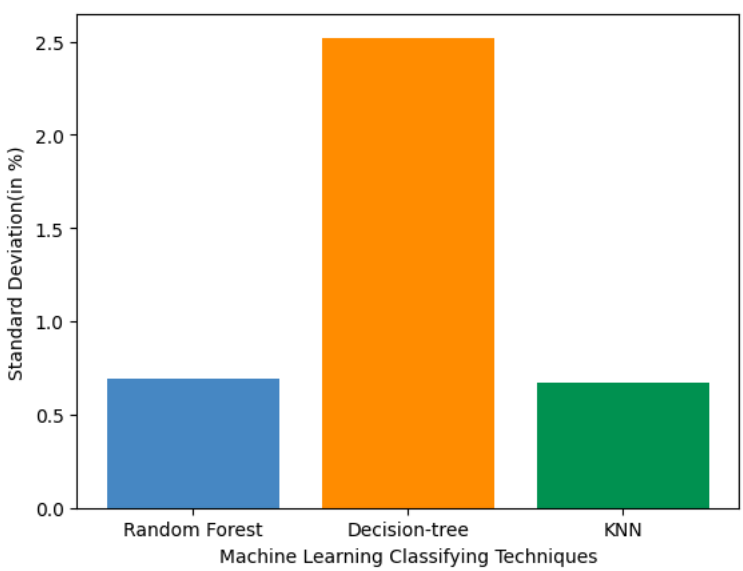



```
import numpy as np
import matplotlib.pyplot as plt

# create a dataset
Algorithms = ['Random Forest', 'Decision-tree','KNN Classifier']
Accuracy = [b1, b2, b3]

x_pos = np.arange(len(Accuracy))

# Create bars with different colors
plt.bar(x_pos, Accuracy, color=['#488AC7','#ff8c00','#009150'])

# Create names on the x-axis
plt.xticks(x_pos, Algorithms)
plt.ylabel('Accuracy(in %)')
plt.xlabel('Machine Learning Classifying Techniques')
```

```
# Show graph
plt.show()
```

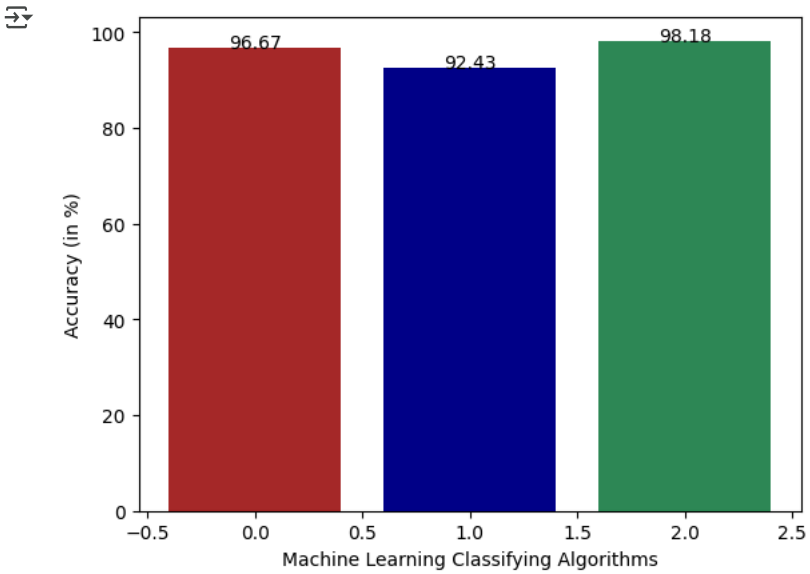



```
import numpy as np
import matplotlib.pyplot as plt

# create a dataset
Algorithms = ['Random Forest', 'Decision-tree','KNN']
Accuracy = [c1, c2, c3]

x_pos = np.arange(len(Accuracy))

# Create bars with different colors
plt.bar(x_pos, Accuracy, color= ['#488AC7','#ff8c00','#009150'])

# Create names on the x-axis
plt.xticks(x_pos, Algorithms)
plt.ylabel('Standard Deviation(in %)')
plt.xlabel('Machine Learning Classifying Techniques')

# Show graph
plt.show()
```

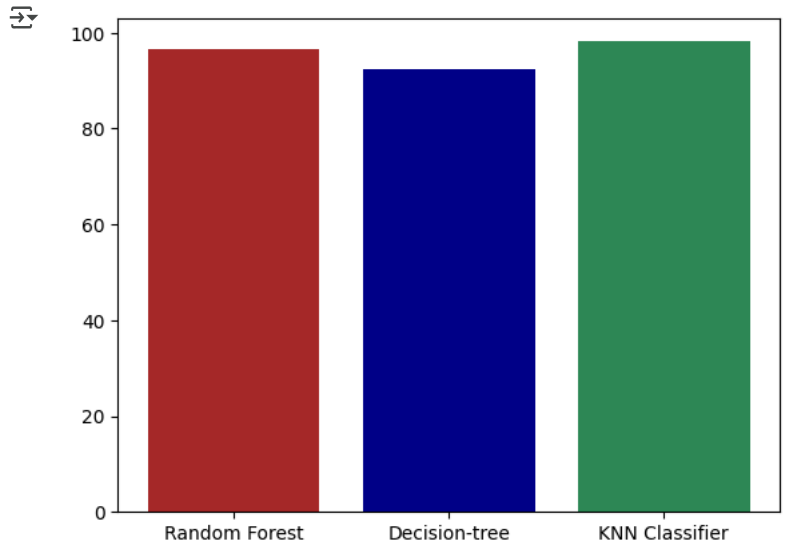


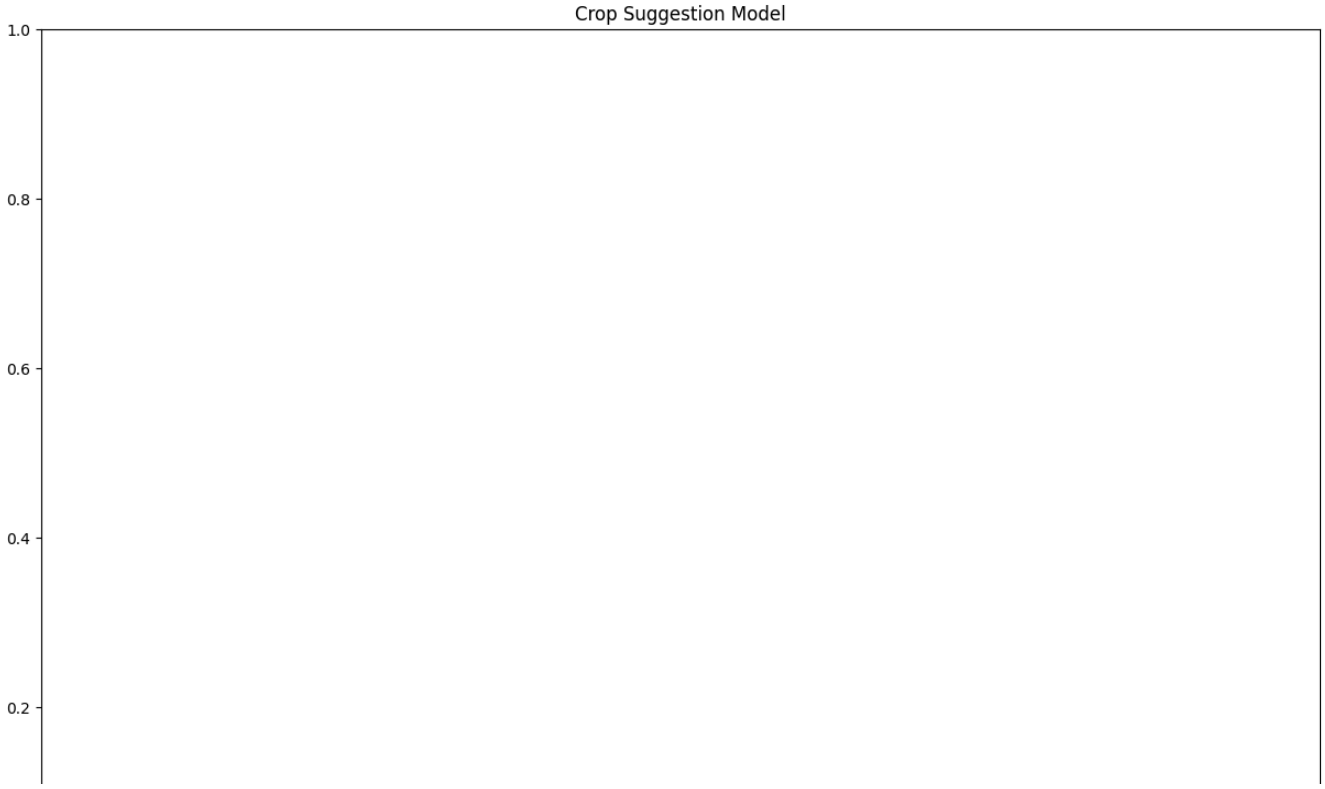

```
def addlabels(x,y):
    for i in range(len(x)):
        plt.text(i,y[i],y[i],ha = 'center')

if __name__ == '__main__':
    # creating data on which bar chart will be plot
```

```
x = ["Random Forest", "Decision tree", "KNN"]
y = [b1,b2,b3]


x_pos = np.arange(len(y))

# Create bars with different colors
plt.bar(x_pos, y, color= ['#A52A2A','#00008B','#2E8B57'])

# calling the function to add value labels
addlabels(x, y)

# giving X and Y labels
plt.xlabel("Machine Learning Classifying Algorithms")
plt.ylabel("Accuracy (in %)")

# visualizing the plot
plt.show()
```



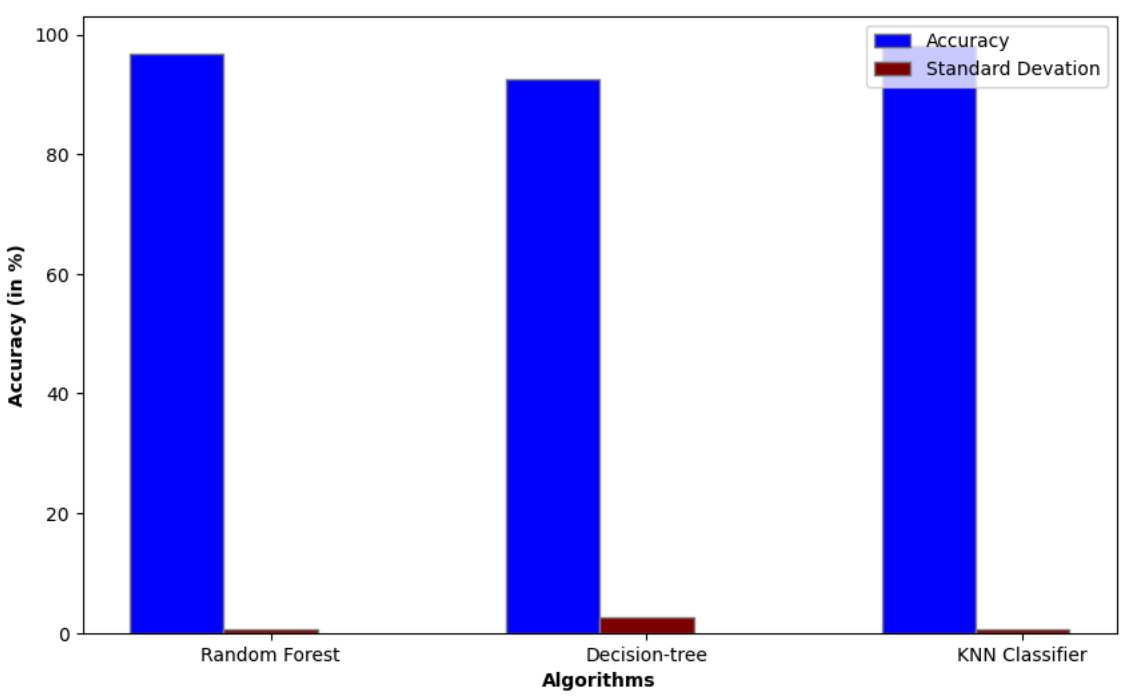

```
plt.bar(df['Algorithms'], df['Accuracy'], color = ['#A52A2A','#00008B','#2E8B57'])
fig = plt.figure(figsize =(15, 10))
plt.title('Crop Suggestion Model')

# Show Plot
plt.show()
```






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

# set width of bar
barWidth = 0.25
fig = plt.subplots(figsize =(10, 6))

# set height of bar
Algorithms = ['Random Forest', 'Decision-tree', 'KNN Classifier']
Accuracy = [b1, b2, b3]
Standard_Deviation = [c1,c2,c3]

# Set position of bar on X axis
br1 = np.arange(len(Accuracy))
br2 = [x + barWidth for x in br1]
br3 = [x + barWidth for x in br2]

# Make the plot
plt.bar(br1, Accuracy, color ='blue', width = barWidth,
        edgecolor ='grey', label ='Accuracy')
plt.bar(br2, Standard_Deviation, color ='maroon', width = barWidth,
        edgecolor ='grey', label ='Standard Devation')

# Adding Xticks
plt.xlabel('Algorithms', fontweight ='bold', fontsize = 10)
```

```
plt.ylabel('Accuracy (in %)', fontweight ='bold', fontsize = 10)
plt.xticks([r + barWidth for r in range(len(Accuracy))],
        Algorithms)

plt.legend()
plt.show()
```




```
# Saving the trained Random Forest model
import pickle
# Dump the trained Naive Bayes classifier with Pickle
RF_pkl_filename = 'RandomForest.pkl'
# Open the file to save as pkl file
RF_Model_pkl = open(RF_pkl_filename, 'wb')
pickle.dump(multi_target_forest, RF_Model_pkl)
# Close the pickle instances
RF_Model_pkl.close()
```

```
df.columns
```

```
Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'Crop'], dtype='object')
```

```
df.describe()
```

	N	P	K	temperature	humidity	ph	rainfall	Crop
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000
mean	50.551818	53.362727	48.149091	25.616244	71.481779	6.469480	103.463655	10.500000
std	36.917334	32.985883	50.647931	5.063749	22.263812	0.773938	54.958389	6.345731
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	20.211267	0.000000
25%	21.000000	28.000000	20.000000	22.769375	60.261953	5.971693	64.551686	5.000000
50%	37.000000	51.000000	32.000000	25.598693	80.473146	6.425045	94.867624	10.500000
75%	84.250000	68.000000	49.000000	28.561654	89.948771	6.923643	124.267508	16.000000
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298.560117	21.000000

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

	0
N	0
P	0
K	0
temperature	0
humidity	0
ph	0
rainfall	0
Crop	0

dtype: int64

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

	N	P	K	temperature	humidity	ph	rainfall	Crop
1270	6	140	205	17.665584	82.929034	6.313086	69.867126	7
1481	98	22	47	29.072653	91.915332	6.341401	28.835684	15
1832	38	14	30	26.924495	91.201060	5.570745	194.902214	4
293	35	63	76	17.815645	17.607566	7.714153	90.820976	3
1307	85	22	53	25.965342	89.770767	6.849472	59.463386	21
...	...	...	...	...	...	...	...	...
740	24	80	19	29.678925	69.085455	6.808042	65.664366	2
1032	105	74	45	25.145176	81.382041	6.098369	119.218154	1
2121	83	21	28	25.567483	60.492446	7.466901	190.225784	5
1424	102	25	50	28.204808	92.914404	6.099662	20.360011	15
1725	47	46	52	23.194511	91.403016	6.502289	206.399921	17

2200 rows × 8 columns

Linear Regression

```
# Training the Simple Linear Regression model .

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

```
LinearRegression()
```

```
# Predicting the test Results

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

```
array([[ 0.0432267 ,  0.22353167,  0.02355126, ..., -0.00071938,
         0.16445985, -0.11408559],
       [ 0.04460984,  0.14685568,  0.06060497, ..., -0.02407102,
         0.2817105 , -0.04511437],
       [-0.0587435 ,  0.25118269,  0.06171324, ..., -0.04947957,
         0.09731872,  0.15173739],
       ...,
       [-0.00821384,  0.02159181, -0.02420635, ...,  0.12172085,
        -0.09879945,  0.29045424],
       [ 0.06555954,  0.06328019, -0.08704225, ...,  0.08727566,
         0.43491123, -0.05032421],
       [ 0.07752188,  0.16482299,  0.13629275, ..., -0.07687474,
         0.25844957, -0.08197713]])
```

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

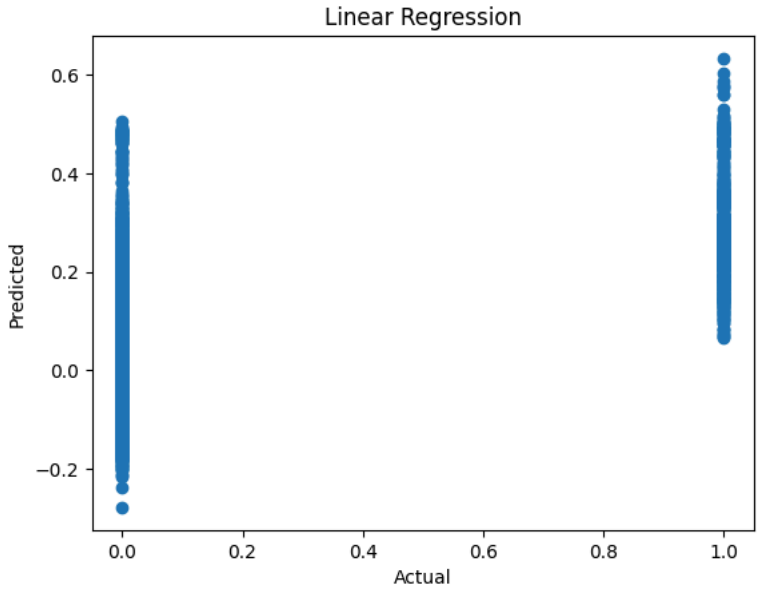
```
0.2428937761428853
```

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

```
R2 score :  0.2428937761428853
```

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

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



Assumptions taken:- Linearity , Homoscedasticity , Multivariate mormality , Lack of multicollinearity

R2 score:- tells how well the unknown samples will be predicted by our model

Random Forest Algorithm

```
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([[0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.90909091,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 1.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.09090909,
        0.        ]])
```

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

```
0.9830678773733783
```

```
# Calculating R2 score

from sklearn.metrics import r2_score
r1 = r2_score(y_test,rf_predict)
print("R2 score : ",r1)
```

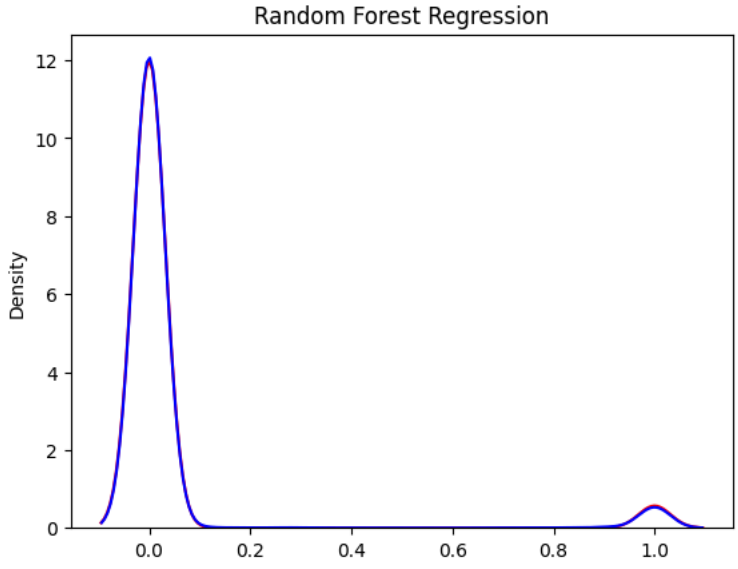
```
R2 score :  0.9830678773733783
```

```
# Calculating Adj. R2 score:

Adjr2_1 = 1 - (1-r)*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1)
print("Adj. R-Squared : {}".format(Adjr2_1))
```

```
Adj. R-Squared : 0.23311565148052404
```

```
ax = sns.distplot(y_test, hist = False, color = "r", label = "Actual value ")
sns.distplot(rf_predict, hist = False, color = "b", label = "Predicted Values", ax = ax)
plt.title('Random Forest Regression')
```

```
<ipython-input-148-536534997cd4>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  ax = sns.distplot(y_test, hist = False, color = "r", label = "Actual value ")
<ipython-input-148-536534997cd4>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(rf_predict, hist = False, color = "b", label = "Predicted Values", ax = ax)
Text(0.5, 1.0, 'Random Forest Regression')
```



Comparison between Linear Regression and Random Forest Algorithm

accuracy for linear regression is very low as compared to accuracy from Random Forest Algorithm (85% to 90%) but it us slow

Support Vector Machine

```
 # Feature Scaling

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

```
print(x_train)
print(x_test)
```

```
[[-0.7296725   2.33743662  2.91453858 ...  0.55160889 -0.96704062
  -0.69269661]
 [-0.97513364  2.39776794  2.97366646 ...  0.90922661 -0.84587838
   0.26098342]
 [-0.53875828 -0.16631335 -0.61342487 ...  0.69701203  0.16457571
```

```
         -0.90339279]
        ...
        [ 1.34311048 -1.19194587  0.01727251 ...  0.88493738 -0.13566618
         -1.41025392]
        [-1.22059479 -0.16631335 -0.47545982 ...  0.59118738  0.42179126
         -1.00035817]
        [-0.64785212  0.07501195 -0.55429699 ... -2.32296897 -1.2160758
         -0.15243101]]
       [[-0.29092628  0.74227485 -0.54091829 ... -0.47142409 -2.30974703
          1.0070614 ]
        [ 0.87857802  0.21844137 -0.24289666 ...  0.39302979  0.53718773
          1.75022386]
        [ 2.26071946 -0.30539212 -0.64025883 ...  0.46188929  0.31004878
         -0.31012102]
        ...
        [ 1.06463552 -1.16817669  0.07499308 ...  0.88109608 -0.02705073
         -1.44991761]
        [ 0.66594088 -0.45946079 -0.18329233 ...  0.59492767 -0.31534501
          3.25113965]
        [ 0.53304266  0.46495124 -0.06408368 ...  0.94505321  0.64564859
          1.79663372]]
```

```
# Predicting Result

svr_predict = regressor.predict(x_test)
svr_predict
```

```
ax = sns.distplot(y_test, hist = False, color = "r", label = "Actual value ")
sns.distplot(svr_predict, hist = False, color = "b", label = "Predicted Values", ax = ax)
plt.title('Support Vector Regression')
```

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([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

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

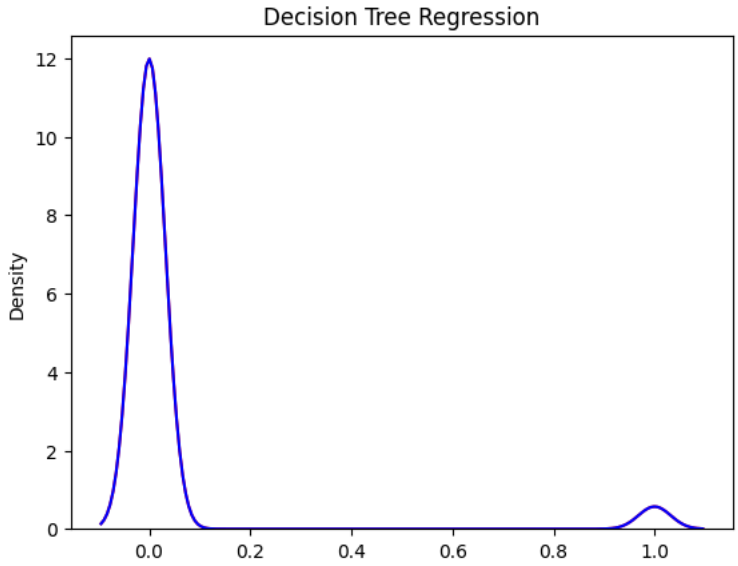
```
0.9477272663246094
```

```
# Calculating R2 score :

from sklearn.metrics import r2_score
r2 = r2_score(y_test,decisiontree_predict)
print("R2 score : ",r2)
```

```
R2 score :  0.9477272663246094
```

```
ax = sns.distplot(y_test, hist = False, color = "r", label = "Actual value ")
sns.distplot(decisiontree_predict, hist = False, color = "b", label = "Predicted Values", ax = ax)
plt.title('Decision Tree Regression')
```

```
<ipython-input-157-514925a0f08b>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  ax = sns.distplot(y_test, hist = False, color = "r", label = "Actual value ")
<ipython-input-157-514925a0f08b>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(decisiontree_predict, hist = False, color = "b", label = "Predicted Values", ax = ax)
Text(0.5, 1.0, 'Decision Tree Regression')
```



Cross Validation

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = model, X = x_train, y=y_train, cv = 10)


a1 = (accuracies.mean()*100)
b1 = (accuracies.std()*100)


# Mean Accuracy and SD of 10 fold results

print("Accuracy : {:.2f}%".format (accuracies.mean()*100))
print("Standard Deviation : {:.2f}%".format(accuracies.std()*100))
```

```
Accuracy : 97.56%
Standard Deviation : 1.21%
```

Cross validation

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = regressor, X = x_train, y=y_train)


a2 = (accuracies.mean()*100)
b2 = (accuracies.std()*100)


print("Accuracy : {:.2f}%".format (accuracies.mean()*100))
print("Standard Deviation : {:.2f}%".format(accuracies.std()*100))
```

```
Accuracy : 96.28%
Standard Deviation : 1.85%
```