# Crop_Production_Analysis

August 23, 2024

## 1 Crop Production Analysis

[1]:
```
#Importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
#Import the dataset
df = pd.read_csv("Crop Production data.csv")
df.head()
```

[2]:

| | 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 |

### 1.1 Exploratory Data Analysis

[3]:
```
df.shape
```

[3]:
```
(246091, 7)
```

[4]:
```
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
```

[160]:
```
df.Crop_Year = pd.to_datetime(df.Crop_Year)
```

[162]:
```
df.dtypes
```

[162]:
```
State_Name               object
District_Name            object
Crop_Year        datetime64[ns]
Season                   object
Crop                     object
Area                    float64
Production              float64
dtype: object
```

[5]:
```
df.describe()
```

[5]:

| | 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 |

[6]:
```
df.columns
```

[6]:
```
Index(['State_Name', 'District_Name', 'Crop_Year', 'Season', 'Crop', 'Area',
       'Production'],
      dtype='object')
```

[7]:
```
#Alternate method
columns = df.columns.values
```

```
columns
```

[7]:
```
array(['State_Name', 'District_Name', 'Crop_Year', 'Season', 'Crop',
       'Area', 'Production'], dtype=object)
```

[8]:
```
df.isna().sum()
```

[8]:
```
State_Name          0
District_Name       0
Crop_Year           0
Season              0
Crop                0
Area                0
Production       3730
dtype: int64
```

[9]:
```
a = (3730/426091)*100
a
```

[9]:
```
0.8753998558993268
```

[10]:
```
df = df.dropna(subset= ['Production'], axis = 0)
```

[11]:
```
df.isna().sum()
```

[11]:
```
State_Name       0
District_Name    0
Crop_Year        0
Season           0
Crop             0
Area             0
Production       0
dtype: int64
```

[12]:
```
print(df['State_Name'].value_counts())
print(df['State_Name'].nunique())
```

```
Uttar Pradesh                33189
Madhya Pradesh               22604
Karnataka                    21079
Bihar                        18874
Assam                        14622
Odisha                       13524
Tamil Nadu                   13266
Maharashtra                  12496
Rajasthan                    12066
Chhattisgarh                 10368
West Bengal                   9597
```

```
Andhra Pradesh                  9561
Gujarat                         8365
Telangana                       5591
Uttarakhand                     4825
Haryana                         4540
Kerala                          4003
Nagaland                        3904
Punjab                          3143
Meghalaya                       2867
Arunachal Pradesh               2545
Himachal Pradesh                2456
Jammu and Kashmir               1632
Tripura                         1412
Manipur                         1266
Jharkhand                       1266
Mizoram                          954
Puducherry                       872
Sikkim                           714
Dadra and Nagar Haveli           263
Goa                              207
Andaman and Nicobar Islands      201
Chandigarh                        89
Name: State_Name, dtype: int64
33
```

[13]:
```
print(df['District_Name'].value_counts())
print(df['District_Name'].nunique())
```

```
TUMKUR       931
BELGAUM      924
BIJAPUR      905
HASSAN       895
BELLARY      887
            ...
HYDERABAD      8
KHUNTI         6
RAMGARH        6
NAMSAI         1
MUMBAI         1
Name: District_Name, Length: 646, dtype: int64
646
```

[28]:
```
year_count = df['Crop_Year'].nunique()
print(f'Total {year_count} years of data is given in the dataset')
print(df['Crop_Year'].value_counts())
```

```
Total 19 years of data is given in the dataset
2003    17139
2002    16536
```

```
2007    14269
2008    14230
2006    13976
2004    13858
2010    13793
2011    13791
2009    13767
2000    13553
2005    13519
2013    13475
2001    13293
2012    13184
1999    12441
1998    11262
2014    10815
1997     8899
2015      561
Name: Crop_Year, dtype: int64
```

[32]:
```
Seasons = df['Season'].nunique()
print(f'Total {Seasons} seasons are mensioned in above dataset, and they are as␣
 ↪follows')
print(df['Season'].value_counts())
```

```
Total 6 seasons are mensioned in above dataset, and they are as follows
Kharif         94283
Rabi           66160
Whole Year     56127
Summer         14811
Winter          6050
Autumn          4930
Name: Season, dtype: int64
```

[16]:
```
print(df['Crop'].nunique())
print(df['Crop'].value_counts())
```

```
124
Rice                 15082
Maize                13787
Moong(Green Gram)    10106
Urad                  9710
Sesamum               8821
                     …
Litchi                   6
Coffee                   6
Apple                    4
Peach                    4
Other Dry Fruit          1
Name: Crop, Length: 124, dtype: int64
```

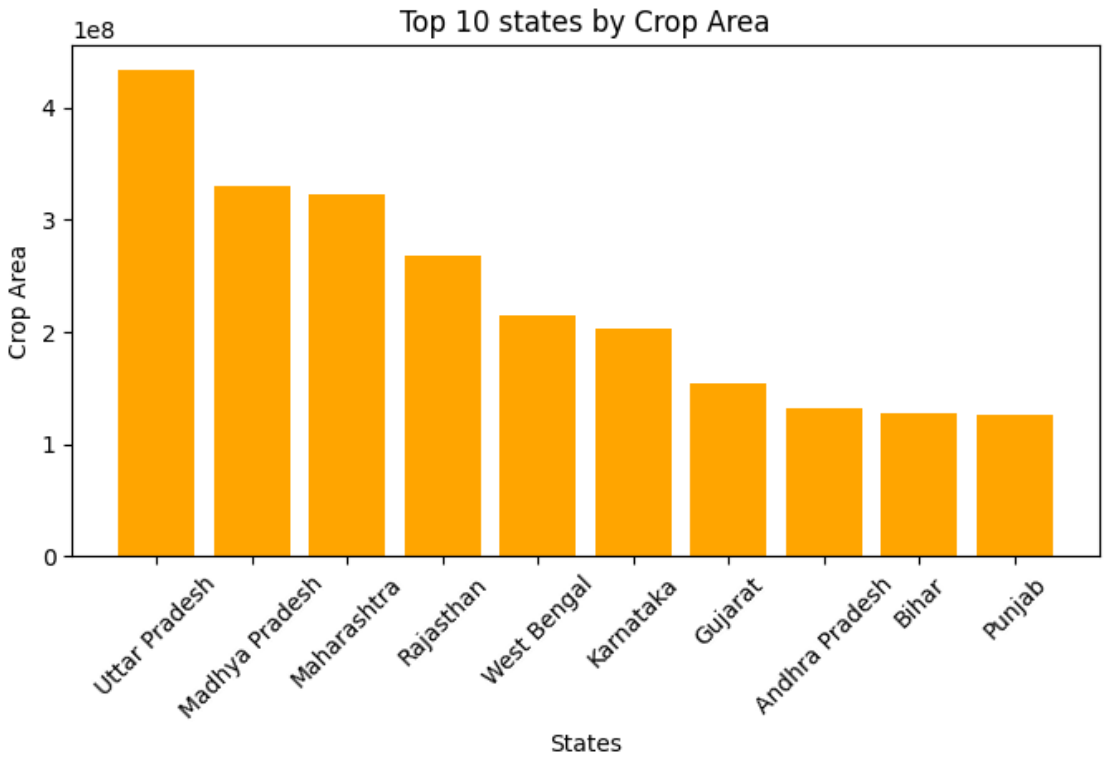
[17]:
```
State_crop_area = df.groupby('State_Name').agg({'Area':'sum'}).reset_index()
```

[18]:
```
Top_10_states_by_Area = State_crop_area.sort_values(by='Area', ascending =␣
↪False).head(10)
Top_10_states_by_Area
```

[18]:

| | State_Name | Area |
|---|---|---|
| 30 | Uttar Pradesh | 4.336223e+08 |
| 16 | Madhya Pradesh | 3.297913e+08 |
| 17 | Maharashtra | 3.221860e+08 |
| 25 | Rajasthan | 2.687882e+08 |
| 32 | West Bengal | 2.154030e+08 |
| 14 | Karnataka | 2.029086e+08 |
| 9 | Gujarat | 1.549261e+08 |
| 1 | Andhra Pradesh | 1.315073e+08 |
| 4 | Bihar | 1.282695e+08 |
| 24 | Punjab | 1.267152e+08 |

[48]:
```
plt.figure(figsize = (8,4))
plt.bar(x = Top_10_states_by_Area['State_Name'],color = 'orange', height =␣
↪Top_10_states_by_Area['Area'])
plt.title('Top 10 states by Crop Area')
plt.xlabel('States')
plt.xticks(rotation = 45)
plt.ylabel('Crop Area')
plt.show()
```




[19]:
```
Lowest_10_states_by_Area = State_crop_area.sort_values(by = 'Area', ascending =␣
↪False).tail(10)
Lowest_10_states_by_Area
```

[19]:

| | State_Name | Area |
|---|---|---|
| 2 | Arunachal Pradesh | 4364340.00 |
| 19 | Meghalaya | 4035028.00 |
| 18 | Manipur | 2007254.00 |
| 26 | Sikkim | 1524479.00 |
| 8 | Goa | 1205678.50 |
| 20 | Mizoram | 993640.17 |
| 23 | Puducherry | 548736.00 |
| 7 | Dadra and Nagar Haveli | 396515.00 |
| 0 | Andaman and Nicobar Islands | 337083.40 |
| 5 | Chandigarh | 12502.00 |

[20]:
```
Production_by_state = df.groupby('State_Name').agg({'Production':'sum'}).
↪reset_index()
Production_by_state
```

[20]:

| | State_Name | Production |
|---|---|---|
| 0 | Andaman and Nicobar Islands | 7.182232e+08 |

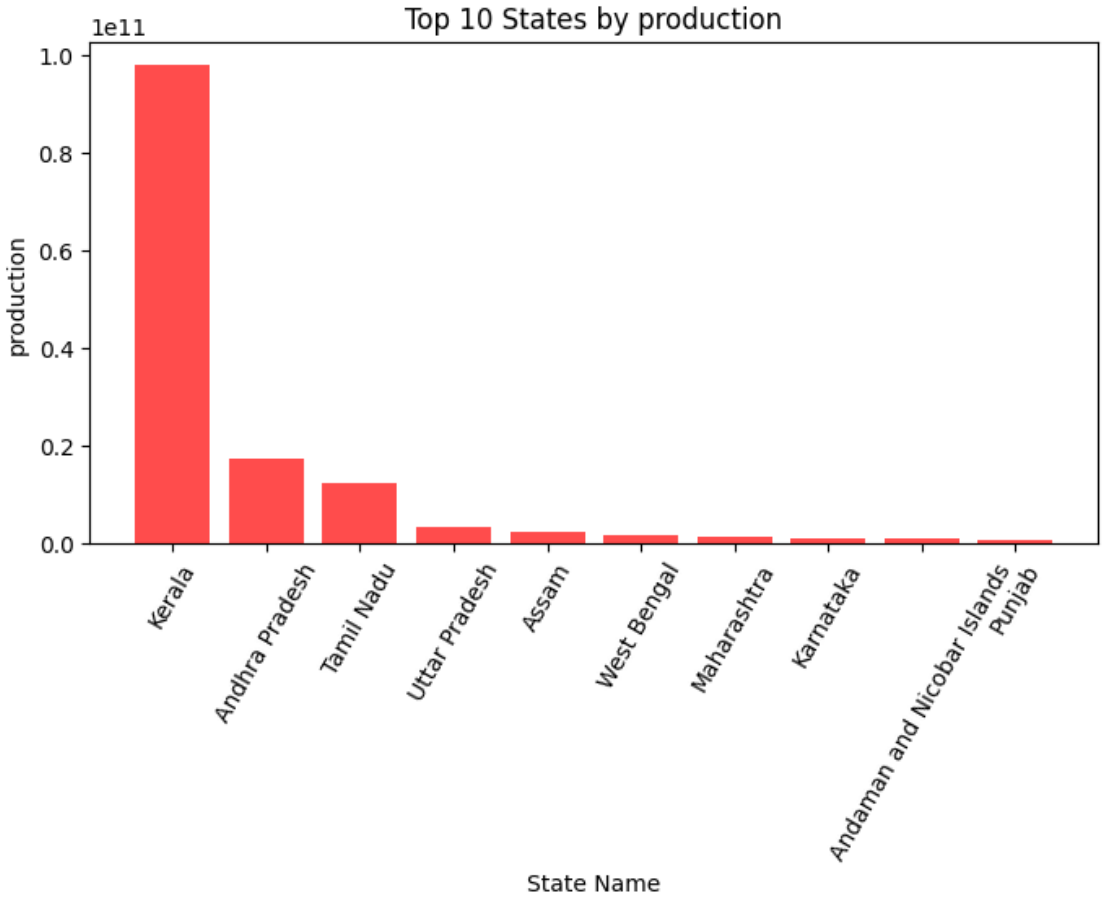
```
1                Andhra Pradesh  1.732459e+10
2             Arunachal Pradesh  6.823913e+06
3                         Assam  2.111752e+09
4                         Bihar  3.664836e+08
5                    Chandigarh  6.395650e+04
6                  Chhattisgarh  1.009519e+08
7         Dadra and Nagar Haveli  1.847871e+06
8                           Goa  5.057558e+08
9                       Gujarat  5.242913e+08
10                      Haryana  3.812739e+08
11             Himachal Pradesh  1.780517e+07
12           Jammu and Kashmir   1.329102e+07
13                    Jharkhand  1.077774e+07
14                    Karnataka  8.634298e+08
15                       Kerala  9.788005e+10
16                Madhya Pradesh  4.488407e+08
17                   Maharashtra  1.263641e+09
18                       Manipur  5.230917e+06
19                     Meghalaya  1.211250e+07
20                       Mizoram  1.661540e+06
21                      Nagaland  1.276595e+07
22                        Odisha  1.609041e+08
23                    Puducherry  3.847245e+08
24                        Punjab  5.863850e+08
25                     Rajasthan  2.813203e+08
26                        Sikkim  2.435735e+06
27                    Tamil Nadu  1.207644e+10
28                    Telangana   3.351479e+08
29                       Tripura  1.252292e+07
30                  Uttar Pradesh  3.234493e+09
31                    Uttarakhand  1.321774e+08
32                    West Bengal  1.397904e+09
```

[21]:
```
Top_10_States_by_Production = Production_by_state.sort_values(by =␣
 ↪'Production', ascending = False).head(10)
Top_10_States_by_Production
```

[21]:
```
                       State_Name    Production
15                         Kerala  9.788005e+10
1                  Andhra Pradesh  1.732459e+10
27                      Tamil Nadu  1.207644e+10
30                  Uttar Pradesh  3.234493e+09
3                           Assam  2.111752e+09
32                    West Bengal  1.397904e+09
17                    Maharashtra  1.263641e+09
14                      Karnataka  8.634298e+08
0   Andaman and Nicobar Islands  7.182232e+08
```

```
24                    Punjab  5.863850e+08
```

[55]:
```
plt.figure(figsize = (8,4))
plt.bar(x = Top_10_States_by_Production['State_Name'], height =␣
 ↪Top_10_States_by_Production['Production'], color = 'red', alpha = 0.7)
plt.xlabel ('State Name')
plt.ylabel('production')
plt.xticks(rotation = 60)
plt.title('Top 10 States by production')
plt.show()
```




[22]:
```
Production_by_Year = df.groupby('Crop_Year').agg({'Production':'sum'}).
 ↪reset_index()
Production_by_Year
```

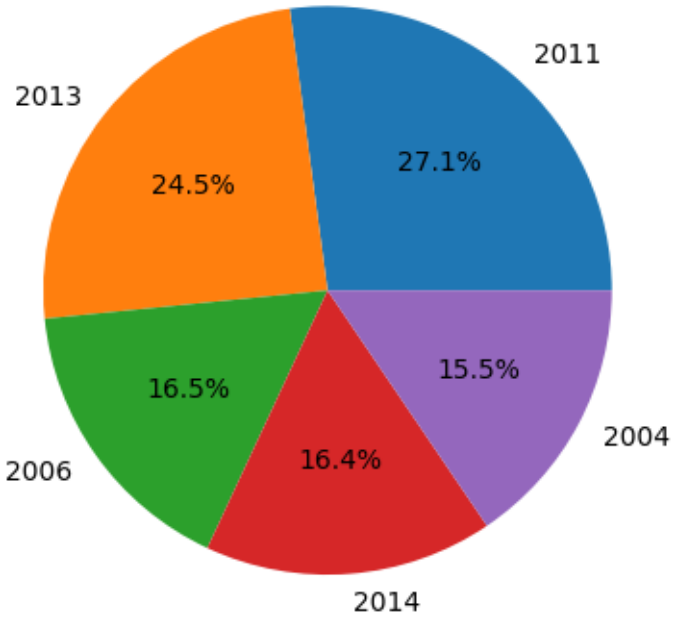
[22]:
```
    Crop_Year    Production
0        1997  8.512329e+08
1        1998  5.825321e+09
```

```
2         1999  6.434666e+09
3         2000  7.449709e+09
4         2001  7.465541e+09
5         2002  7.696955e+09
6         2003  7.917974e+09
7         2004  8.189462e+09
8         2005  8.043757e+09
9         2006  8.681913e+09
10        2007  6.879442e+09
11        2008  7.717018e+09
12        2009  7.660494e+09
13        2010  6.307609e+09
14        2011  1.430890e+10
15        2012  8.171055e+09
16        2013  1.290359e+10
17        2014  8.664541e+09
18        2015  6.935065e+06
```

[23]:
```
Top_5_production_years = Production_by_Year.sort_values(by = 'Production',␣
↪ascending = False).head()
Top_5_production_years
```

[23]:
```
    Crop_Year    Production
14       2011  1.430890e+10
16       2013  1.290359e+10
9        2006  8.681913e+09
17       2014  8.664541e+09
7        2004  8.189462e+09
```

[60]:
```
plt.pie(Top_5_production_years.Production, labels = Top_5_production_years.
↪Crop_Year, autopct = "%1.1f%%")
plt.show()
```




[24]:
```
Crops_by_Production = df.groupby('Crop').agg({'Production':'sum'}).reset_index()
```

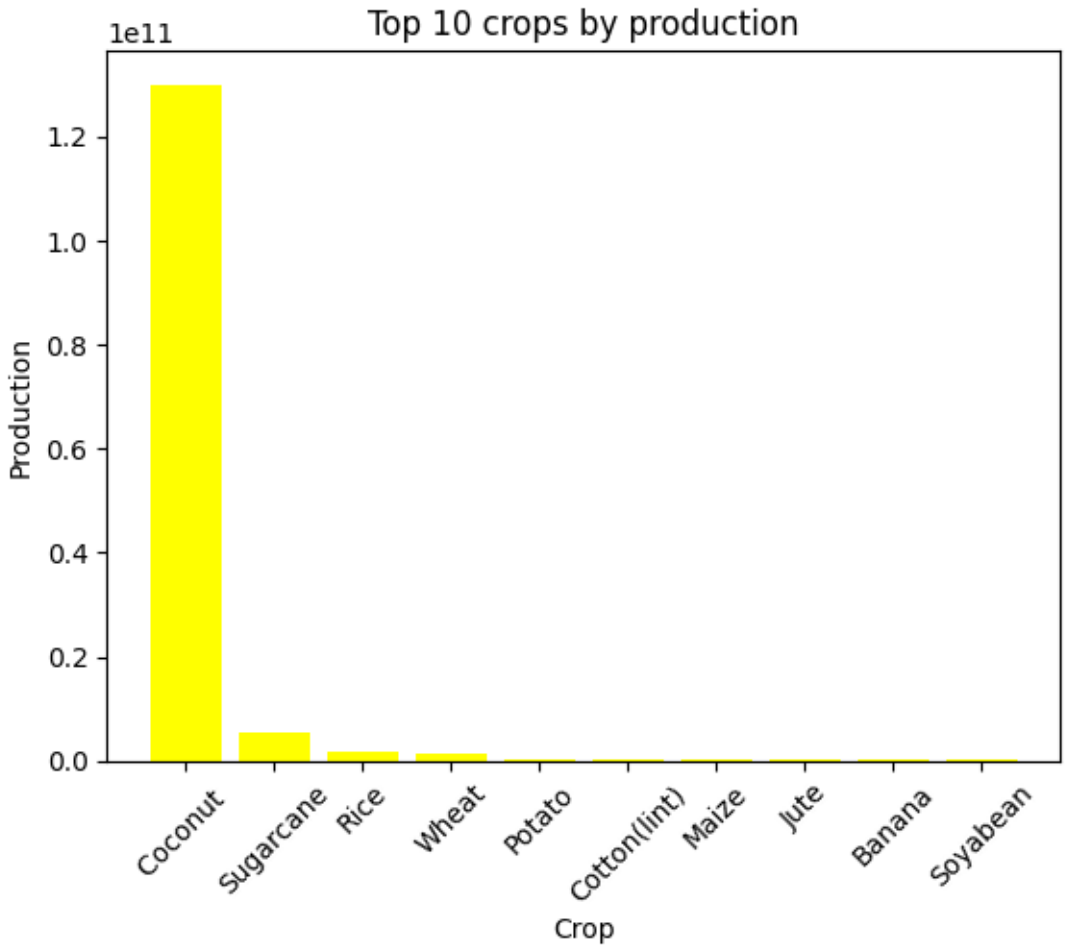
[25]:
```
Top_10_crops_by_production = Crops_by_Production.sort_values(by =␣
↪'Production',ascending = False).head(10)
Top_10_crops_by_production
```

[25]:

| | Crop | Production |
|---|---|---|
| 28 | Coconut | 1.299816e+11 |
| 106 | Sugarcane | 5.535682e+09 |
| 95 | Rice | 1.605470e+09 |
| 119 | Wheat | 1.332826e+09 |
| 87 | Potato | 4.248263e+08 |
| 33 | Cotton(lint) | 2.970000e+08 |
| 59 | Maize | 2.733418e+08 |
| 49 | Jute | 1.815582e+08 |
| 7 | Banana | 1.461327e+08 |
| 105 | Soyabean | 1.418372e+08 |

[68]:
```
plt.bar(Top_10_crops_by_production.Crop, Top_10_crops_by_production.␣
↪Production, color = 'yellow')
plt.title('Top 10 crops by production')
plt.xlabel('Crop')
plt.ylabel('Production')
```

```
plt.xticks(rotation = 45)
plt.show()
```



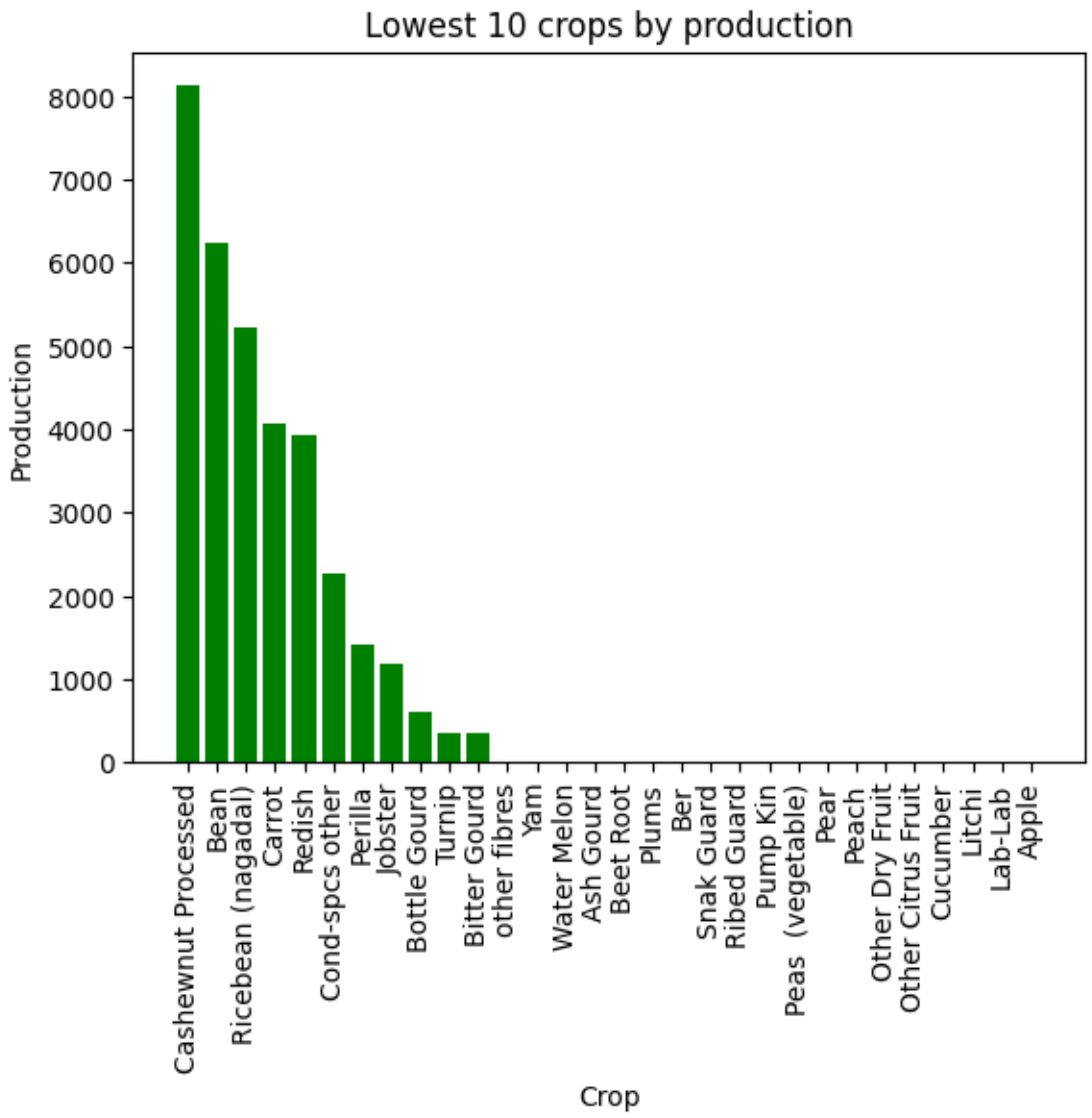

[26]:
```
Lowest_10_Crop_Production = Crops_by_Production.sort_values(by = 'Production',␣
↪ascending = False).tail(30)
Lowest_10_Crop_Production
```

[26]:

| | Crop | Production |
|---|---|---|
| 23 | Cashewnut Processed | 8121.0 |
| 9 | Bean | 6240.0 |
| 96 | Ricebean (nagadal) | 5230.0 |
| 21 | Carrot | 4066.0 |
| 93 | Redish | 3936.0 |
| 31 | Cond-spcs other | 2260.4 |
| 82 | Perilla | 1410.0 |
| 47 | Jobster | 1180.0 |
| 17 | Bottle Gourd | 598.0 |

```
115               Turnip       363.0
14          Bitter Gourd       353.0
121         other fibres         0.0
120                  Yam         0.0
118          Water Melon         0.0
4              Ash Gourd         0.0
11             Beet Root         0.0
84                 Plums         0.0
12                   Ber         0.0
104           Snak Guard         0.0
94           Ribed Guard         0.0
89              Pump Kin         0.0
80     Peas  (vegetable)         0.0
79                  Pear         0.0
78                 Peach         0.0
72       Other Dry Fruit         0.0
71    Other Citrus Fruit         0.0
35              Cucumber         0.0
58                Litchi         0.0
54               Lab-Lab         0.0
0                  Apple         0.0
```

[165]:
```
plt.bar(Lowest_10_Crop_Production.Crop, Lowest_10_Crop_Production. Production,␣
↪color = 'green')
plt.title('Lowest 10 crops by production')
plt.xlabel('Crop')
plt.ylabel('Production')
plt.xticks(rotation = 90)
plt.show()
```




[75]:
```
area =df.Area.sum()
print(f'Total area under Harvest is: {area}')
```

```
Total area under Harvest is: 2948906741.25
```

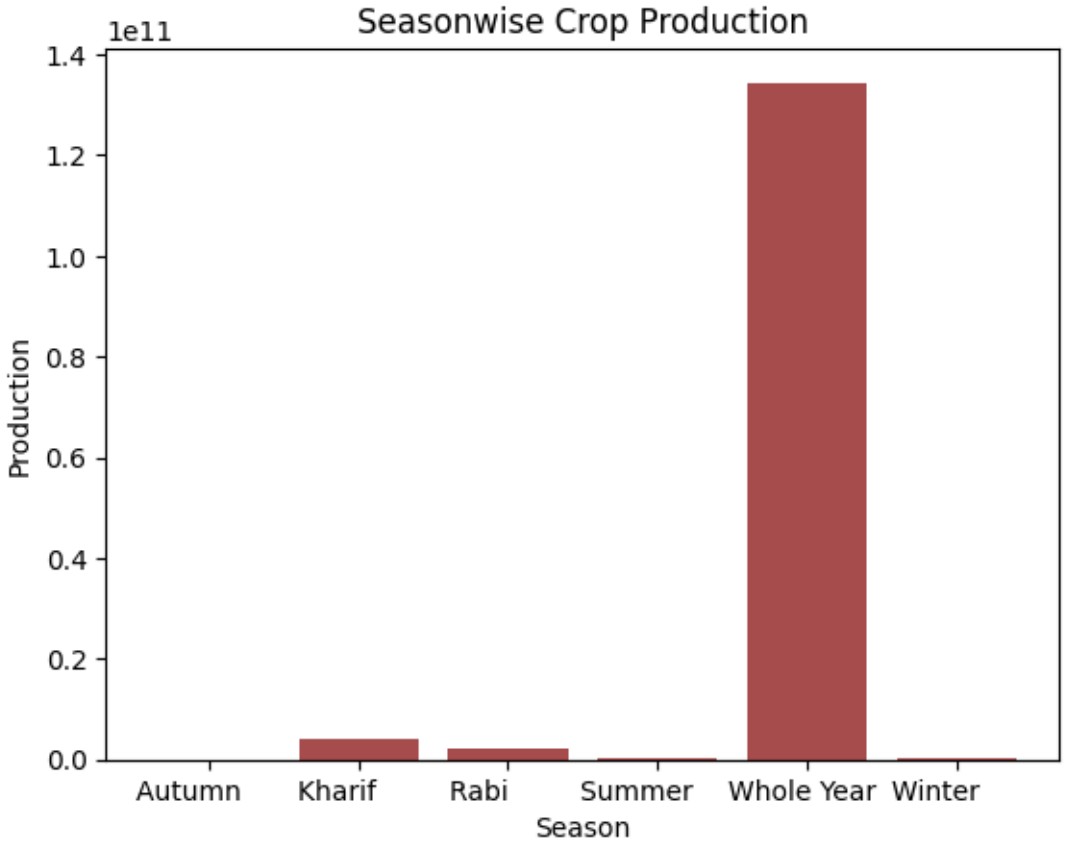
[79]:
```
production = df.Production.sum()
print(f'Total Production in {year_count} years is: {production}')
```

```
Total Production in 19 years is: 141176116767.39
```

[83]:
```
#Visualizing top 5 crops seasonwise
Seasonwise_crop_production = df.groupby('Season')['Production'].sum()
Seasonwise_crop_production
```

[83]:
```
Season
Autumn          6.441377e+07
Kharif          4.029970e+09
Rabi            2.051688e+09
Summer          1.706579e+08
Whole Year      1.344248e+11
Winter          4.345498e+08
Name: Production, dtype: float64
```

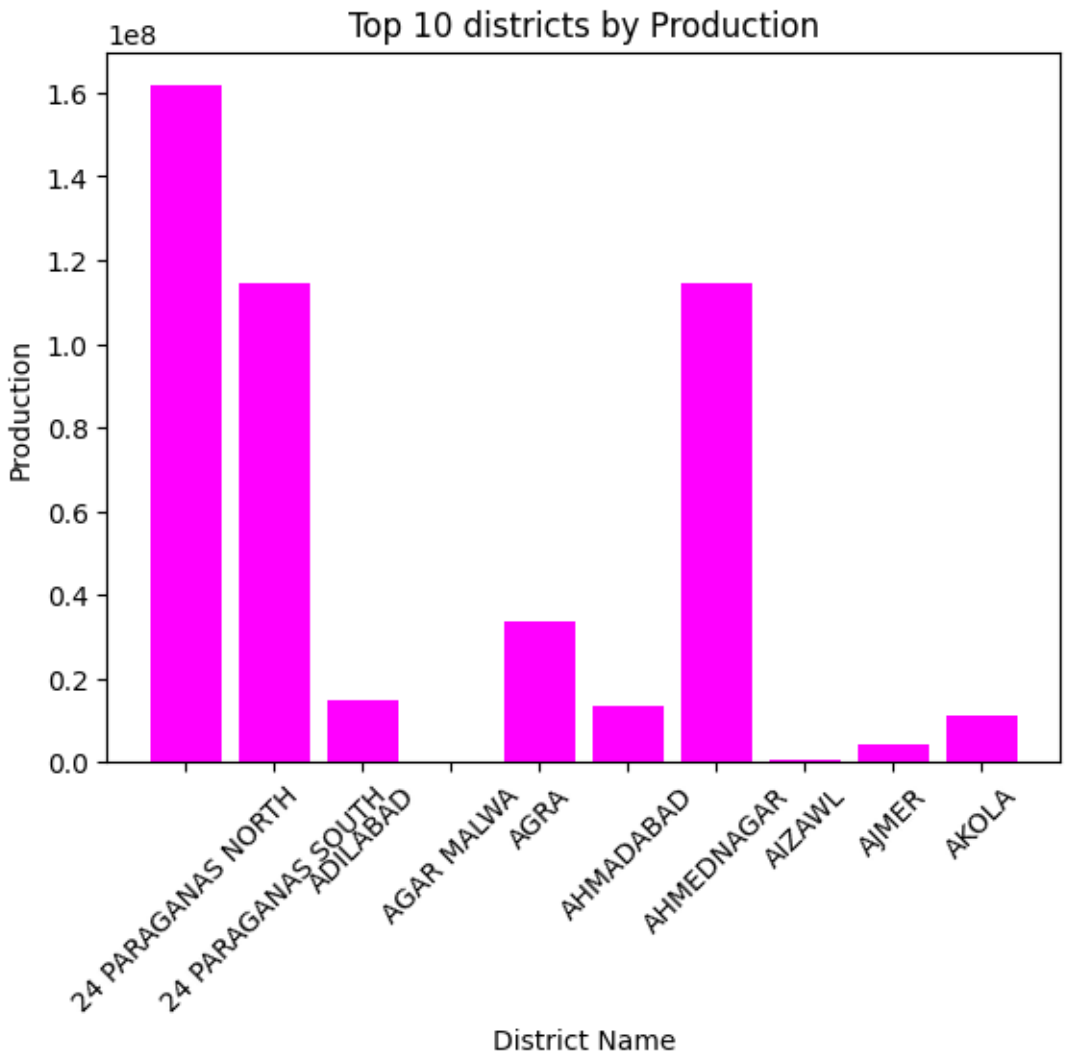
[103]:
```
plt.bar(height =Seasonwise_crop_production.values, x=␣
  ↪Seasonwise_crop_production.index, color = 'maroon', alpha =0.7  )
plt.title('Seasonwise Crop Production')
plt.xlabel('Season')
plt.ylabel('Production')
plt.show()
```




[112]:
```
Top_10_Districts_by_production = df.groupby('District_Name')['Production'].
  ↪sum().head(10)
Top_10_Districts_by_production
```

[112]:
```
District_Name
24 PARAGANAS NORTH    1.616859e+08
24 PARAGANAS SOUTH    1.143209e+08
ADILABAD              1.481831e+07
AGAR MALWA            2.790010e+05
AGRA                  3.366908e+07
AHMADABAD             1.322545e+07
AHMEDNAGAR            1.145497e+08
AIZAWL                5.925400e+05
AJMER                 4.252280e+06
AKOLA                 1.107945e+07
Name: Production, dtype: float64
```

[159]:
```
plt.bar(x = Top_10_Districts_by_production.index, height =␣
 ↪Top_10_Districts_by_production.values, color = 'magenta')
plt.xlabel("District Name")
plt.title('Top 10 districts by Production')
plt.xticks(rotation = 45)
plt.ylabel('Production')
plt.show()
```




## 2 Model Building

Let's, first observe the dataset and we'll decide which model is best for the given dataset. 1. columns: 'State_Name', 'District_Name', 'Crop_Year', 'Season', 'Crop', 'Area', 'Production' 2. Linear Regression: Good for understanding linear relationships between features. 3. Decision Trees: Can model non-linear relationships and interactions between features. 4. Random Forests: An ensemble method that improves prediction accuracy and handles non-linearity. 5. Gradient Boosting Machines (GBMs): Includes algorithms like XGBoost, LightGBM, and CatBoost, which are effective for complex datasets.

## 2.1 1. Linear regression

[118]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

[134]:
```
encoded_df = pd.get_dummies(df, columns = ['State_Name', 'District_Name',␣
 ↪'Season', 'Crop'], drop_first = True)
encoded_df.head()
```

[134]:

| | Crop_Year | Area | Production | State_Name_Andhra Pradesh | \ |
|---|---|---|---|---|---|
| 0 | 2000 | 1254.0 | 2000.0 | 0 | |
| 1 | 2000 | 2.0 | 1.0 | 0 | |
| 2 | 2000 | 102.0 | 321.0 | 0 | |
| 3 | 2000 | 176.0 | 641.0 | 0 | |
| 4 | 2000 | 720.0 | 165.0 | 0 | |

| | State_Name_Arunachal Pradesh | State_Name_Assam | State_Name_Bihar | \ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 0 | |
| 2 | 0 | 0 | 0 | |
| 3 | 0 | 0 | 0 | |
| 4 | 0 | 0 | 0 | |

| | State_Name_Chandigarh | State_Name_Chhattisgarh | \ |
|---|---|---|---|
| 0 | 0 | 0 | |
| 1 | 0 | 0 | |
| 2 | 0 | 0 | |
| 3 | 0 | 0 | |
| 4 | 0 | 0 | |

| | State_Name_Dadra and Nagar Haveli | … | Crop_Turmeric | Crop_Turnip | \ |
|---|---|---|---|---|---|
| 0 | 0 | … | 0 | 0 | |
| 1 | 0 | … | 0 | 0 | |
| 2 | 0 | … | 0 | 0 | |
| 3 | 0 | … | 0 | 0 | |
| 4 | 0 | … | 0 | 0 | |

| | Crop_Urad | Crop_Varagu | Crop_Water Melon | Crop_Wheat | Crop_Yam | \ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 0 | 0 | 0 | |
| 2 | 0 | 0 | 0 | 0 | 0 | |
| 3 | 0 | 0 | 0 | 0 | 0 | |
| 4 | 0 | 0 | 0 | 0 | 0 | |

| | Crop_other fibres | Crop_other misc. pulses | Crop_other oilseeds |
|---|---|---|---|
| 0 | 0 | 0 | 0 |

```
1                    0                          0                          0
2                    0                          0                          0
3                    0                          0                          0
4                    0                          0                          0

[5 rows x 808 columns]
```

[141]:
```
print(encoded_df.shape)
encoded_df.columns
```

```
(242361, 808)
```

[141]:
```
Index(['Crop_Year', 'Area', 'Production', 'State_Name_Andhra Pradesh',
       'State_Name_Arunachal Pradesh', 'State_Name_Assam', 'State_Name_Bihar',
       'State_Name_Chandigarh', 'State_Name_Chhattisgarh',
       'State_Name_Dadra and Nagar Haveli',
       …
       'Crop_Turmeric', 'Crop_Turnip', 'Crop_Urad', 'Crop_Varagu',
       'Crop_Water Melon', 'Crop_Wheat', 'Crop_Yam', 'Crop_other fibres',
       'Crop_other misc. pulses', 'Crop_other oilseeds'],
      dtype='object', length=808)
```

[138]:
```
target = 'Production'
X = encoded_df.drop(columns = ['Production'])
y = encoded_df['Production']
```

[140]:
```
X.head()
```

[140]:
```
   Crop_Year    Area  State_Name_Andhra Pradesh  State_Name_Arunachal Pradesh  \
0       2000  1254.0                          0                             0
1       2000     2.0                          0                             0
2       2000   102.0                          0                             0
3       2000   176.0                          0                             0
4       2000   720.0                          0                             0

   State_Name_Assam  State_Name_Bihar  State_Name_Chandigarh  \
0                 0                 0                      0
1                 0                 0                      0
2                 0                 0                      0
3                 0                 0                      0
4                 0                 0                      0

   State_Name_Chhattisgarh  State_Name_Dadra and Nagar Haveli  State_Name_Goa  \
0                        0                                  0               0
1                        0                                  0               0
2                        0                                  0               0
3                        0                                  0               0
4                        0                                  0               0
```

```
   …  Crop_Turmeric  Crop_Turnip  Crop_Urad  Crop_Varagu  Crop_Water Melon  \
0  …              0            0          0            0                 0
1  …              0            0          0            0                 0
2  …              0            0          0            0                 0
3  …              0            0          0            0                 0
4  …              0            0          0            0                 0

   Crop_Wheat  Crop_Yam  Crop_other fibres  Crop_other misc. pulses  \
0           0         0                  0                        0
1           0         0                  0                        0
2           0         0                  0                        0
3           0         0                  0                        0
4           0         0                  0                        0

   Crop_other oilseeds
0                    0
1                    0
2                    0
3                    0
4                    0

[5 rows x 807 columns]
```

[139]:
```
print(X.shape)
print(y.shape)
```

```
(242361, 807)
(242361,)
```

[142]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 42,␣
↪test_size = 0.3)
```

[143]:
```
Linreg = LinearRegression()
Linreg.fit(X_train, y_train)
```

[143]:
```
LinearRegression()
```

[145]:
```
Linreg_pred = Linreg.predict(X_test)
```

[146]:
```
mean_squared_error(y_test, Linreg_pred)
```

[146]:
```
286351792212021.3
```

[147]:
```
r2_score(y_test, Linreg_pred)
```

[147]:
```
0.16760064507981864
```

### 2.2 2. Decision Tree

[149]:
```
from sklearn.tree import DecisionTreeRegressor
```

[150]:
```
DTR = DecisionTreeRegressor()
```

[151]:
```
DTR.fit(X_train, y_train)
```

[151]:
```
DecisionTreeRegressor()
```

[154]:
```
DTR_pred = DTR.predict(X_test)
```

[155]:
```
mean_squared_error(y_test, DTR_pred)
```

[155]:
```
55556216408303.89
```

[156]:
```
r2_score(y_test, DTR_pred)
```

[156]:
```
0.83850298842957
```