Crop Growth Prediction

By:

Adarsh Kumar

Versha Prajapati

Aman Rao

Introduction:

Crop growth prediction is an important agricultural problem. The Agricultural growth primarily depends on weather conditions (rain, temperature...etc.) pesticides. Accurate information about history of crop growth is important for making decisions related to agricultural risk management and future predictions.

Skills:

From this prediction we can predict the growth of crop in India. By analyzing the previous year data. This prediction can also help us that how we can increase the growth of crop for the next coming years. By this prediction we can help our country farmers by which they can easily predict the crop growth and if the crop growth is less they can start working for more crop growth production

Packages required:

To predict crop growth, we need some helpful analytic libraries, like:

1) **NumPy** — NumPy(np) is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices.

2) **Pandas** — Pandas(pd) is a python library. It helps us in organizing the data in very simple manner.

3) **Sklearn**— sklearn is used to build machine learning models including classification, regression, clustering and dimensionality reduction. It should not be used for reading the data, manipulating and summarizing it.

4) **Seaborn**—Seaborn is a library in Python predominantly used for making statistical graphics. Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python.

5) **Matplotlib** — Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

CODE:

1. Importing libraries

```
In [1]: import numpy as np # linear algebra
        import pandas as pd # data processing
        from sklearn import tree
        import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        from matplotlib import rcParams
```

2. Data Importing

```
In [5]: df=pd.read_csv("Production.csv.csv" ,encoding = "ISO-8859-1")
        df.dtypes
```

```
Out[5]: State_Name        object
        District_Name     object
        Crop_Year          int64
        Season            object
        Crop              object
        Area             float64
        Production        object
        dtype: object
```

3. Data preparation and encoding

```
In [6]: #indian agricultural production dataset
        df.head()
```

Out[6]:

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production
0	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Arecanut	1254.0	2000
1	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Other Kharif pulses	2.0	1
2	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Rice	102.0	321
3	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Banana	176.0	641
4	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Cashewnut	720.0	165

```
In [7]: #converting production to numeric type
        df['Production']=pd.to_numeric(df['Production'],errors='coerce')
```

```
In [8]: #grouping area and production for each year by mean
        data=df.groupby(['Crop_Year'])['Area','Production'].mean()
        data=data.reset_index(level=0, inplace=False)
        data
```

```
<ipython-input-8-ae108dc10b37>:2: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be
deprecated, use a list instead.
  data=df.groupby(['Crop_Year'])['Area','Production'].mean()
```

Out[8]:

	Crop_Year	Area	Production
0	1997	26038.324081	9.565489e+04
1	1998	14479.153906	5.172545e+05
2	1999	12678.074790	5.172145e+05
3	2000	12102.612169	5.496723e+05
4	2001	12371.499489	5.616144e+05
5	2002	9463.680476	4.654666e+05
6	2003	9954.769395	4.619857e+05
7	2004	11891.933465	5.909555e+05
8	2005	11822.333236	5.949085e+05
[illegible]	[illegible]	[illegible]	[illegible]

➢ Calculating CPI

```
In [9]: #calulation cpi(  )

        data['CPI']=data['Production']/data['Area']
        data.head()
```

Out[9]:

	Crop_Year	Area	Production	CPI
0	1997	26038.324081	95654.894483	3.673619
1	1998	14479.153906	517254.540970	35.724086
2	1999	12678.074790	517214.531396	40.795984
3	2000	12102.612169	549672.332849	45.417661
4	2001	12371.499489	561614.446722	45.395827

4. Discriptive analysis


```
In [9]: #calulation cpi(  )

        data['CPI']=data['Production']/data['Area']
        data.head()
```

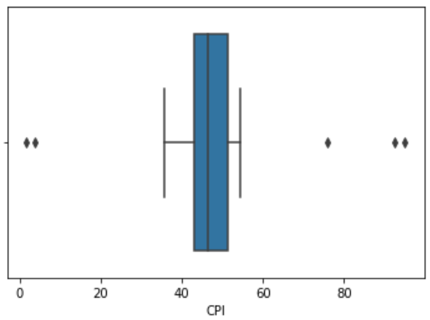
Out[9]:

	Crop_Year	Area	Production	CPI
0	1997	26038.324081	95654.894483	3.673619
1	1998	14479.153906	517254.540970	35.724086
2	1999	12678.074790	517214.531396	40.795984
3	2000	12102.612169	549672.332849	45.417661
4	2001	12371.499489	561614.446722	45.395827

5. Box plots

```
In [11]: #boxplot plotting
         import seaborn as sns
         sns.boxplot(x=data['CPI'])
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x20676a2e5e0>



In [12]:
```
data = data[np.isfinite(data['CPI'])]
data=data[data.CPI >43]
data=data[data.CPI <51]
data.set_index('Crop_Year')
data
```

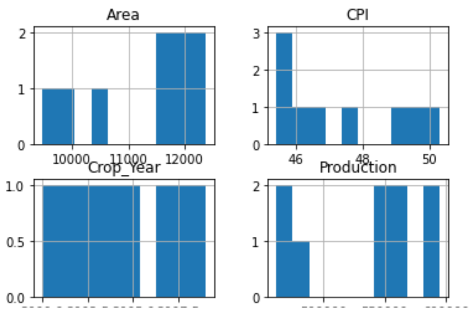
Out[12]:

	Crop_Year	Area	Production	CPI
3	2000	12102.612169	549672.332849	45.417661
4	2001	12371.499489	561614.446722	45.395827
5	2002	9463.680476	465466.567649	49.184519
6	2003	9954.769395	461985.734566	46.408482
7	2004	11891.933465	590955.527122	49.693814
8	2005	11822.333236	594908.463112	50.320732
10	2007	10513.848637	482125.050009	45.856191
11	2008	11768.527148	542306.282654	46.081067
12	2009	11738.077997	556438.877374	47.404599

6. Plotting histogram

In [13]:
```
#plotting histogram
data.hist()
```

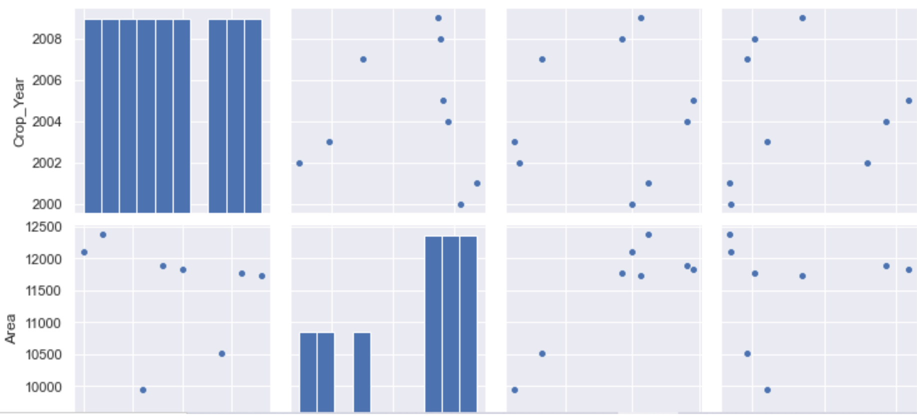
Out[13]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000002067710DBE0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000020677396130>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000206773CD520>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000206773F8970>]],
      dtype=object)
```



7. Scatter plots

In [17]:
```
#scatterplot
sns.set()
cols = ['Crop_Year', 'Area', 'Production', 'CPI']
sns.pairplot(data[cols], size = 2.5)
plt.show();
```

```
C:\Users\Versh\anaconda3\lib\site-packages\seaborn\axisgrid.py:2071: UserWarning: The `size` parameter has been renamed to `hei
ght`; please update your code.
  warnings.warn(msg, UserWarning)
```



8. Comparison Of Production

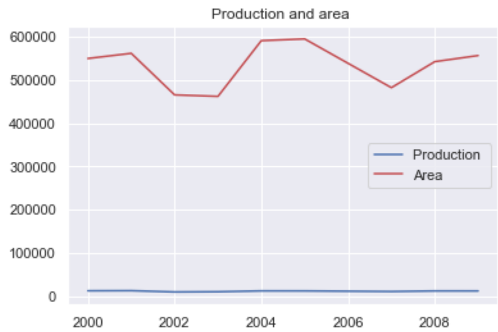
- ## Graph 1….

```
In [18]: #comparison of production and area for each year
x_axis=data.Crop_Year
y_axis=data.Area

y1_axis=data.Production

plt.plot(x_axis,y_axis)
plt.plot(x_axis,y1_axis,color='r')

plt.title("Production and area ")
plt.legend(["Production ","Area"])
plt.show()
```

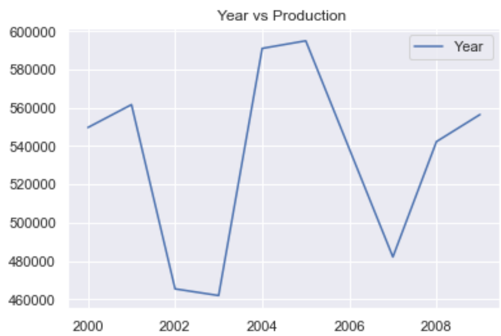


- ## Graph 2….

```
In [19]: #plotting of production
x_axis=data.Crop_Year
y1_axis=data.Production


plt.plot(x_axis,y1_axis)

plt.title("Year vs Production ")
plt.legend(["Year ","Production"])
plt.show()
```



9. Applying random forest

In [20]:
```
#importing random forest regressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
# from sklearn.cross_validation import train_test_split
```

In [21]:
```
#splitting and fitting of the model
x=data.iloc[:,0:1].values
y=data.iloc[:,3].values
regressor=RandomForestRegressor(n_estimators=12,random_state=0,n_jobs=1,verbose=13)

regressor.fit(x,y)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  11 out of  11 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  12 out of  12 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  12 out of  12 | elapsed:    0.0s finished
```

```
building tree 1 of 12
building tree 2 of 12
```

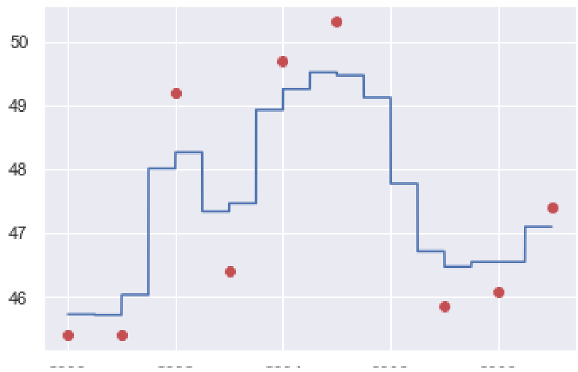
In [22]:
```
#predicting for the test values
y_pred=regressor.predict(x)
y_pred
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  11 out of  11 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  12 out of  12 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  12 out of  12 | elapsed:    0.0s finished
```

Out[22]:
```
array([45.726107  , 45.71519001, 48.00600917, 47.33382739, 48.92472398,
       49.51196079, 46.71271891, 46.54466636, 47.0961381 ])
```

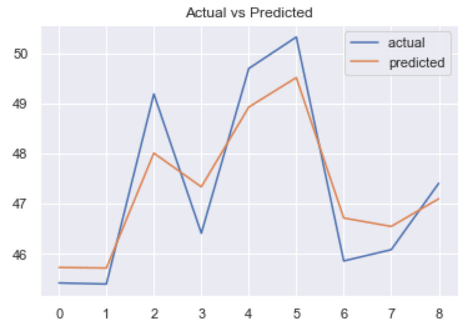
In [23]:
```
#random forest steps plotting
x_grid=np.arange(min(x),max(x),0.001)
x_grid=x_grid.reshape(len(x_grid),1)
plt.scatter(x,y,color='r')
plt.plot(x_grid,regressor.predict(x_grid),color='b')
a=plt.show()
a
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  11 out of  11 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  12 out of  12 | elapsed:    0.0s remaining:    0.0s
```



10. Regression model

```
In [28]: #regression model
         #actual and predicted values
         dm = pd.DataFrame({'Actual': y, 'Predicted': y_pred}).reset_index()
         x_axis=dm.index
         y_axis=dm.Actual
         y1_axis=dm.Predicted
         plt.plot(x_axis,y_axis)
         plt.plot(x_axis,y1_axis)
         plt.title("Actual vs Predicted")
         plt.legend(["actual ","predicted"])
         b=plt.show()
         b
```



Conclusion

Crop growth prediction is still remaining as a challenging issue for farmers. The aim of this research is to propose and implement a rule-based system to predict the crop growth prediction from the collection of past data.

The feature selection approach successfully found important features, and revealed that environmental factors had a greater effect on the crop growth than genotype.

Our future research is to overcome this limitation by looking for more advanced models that are not only more accurate but also more explainable.