WEEK-6
MODEL EVALUATION AND TESTING
Session – 6

Perform Data Preprocessing, Exploration and Splitting on Dataset like Crop Production Dataset

The main objective of Crop Production dataset is to predict the crop production data for different years. The Dataset contains different crops and their production of different districts of India from the year 1997– 2014. We can access this dataset from the Data World website.

We followed the general machine learning workflow step-by-step:

1. Import the required libraries

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

2. Import Dataset and Understand It

Load the dataset

```
data = pd.read_csv('crop_production.csv')
```

To display top 5 values of crop_production.csv dataset

```
data.head()
```

Out[35]:

	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

To display Data description of crop_production.csv dataset

```
data.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
```

3. Data Preprocessing

After loading the data, it's a good practice to see if there are any missing values in the data. We count the number of missing values for each feature using isnull()

```
data.isnull().sum()
```

There are some missing values in the dataset as shown below

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

To remove missing values in dataset

```
data = data.dropna(how='any',axis=0)
```

Again we count the number of missing values for each feature using isnull()

```
data.isnull().sum()
```

Now there are no missing values in the dataset as shown below

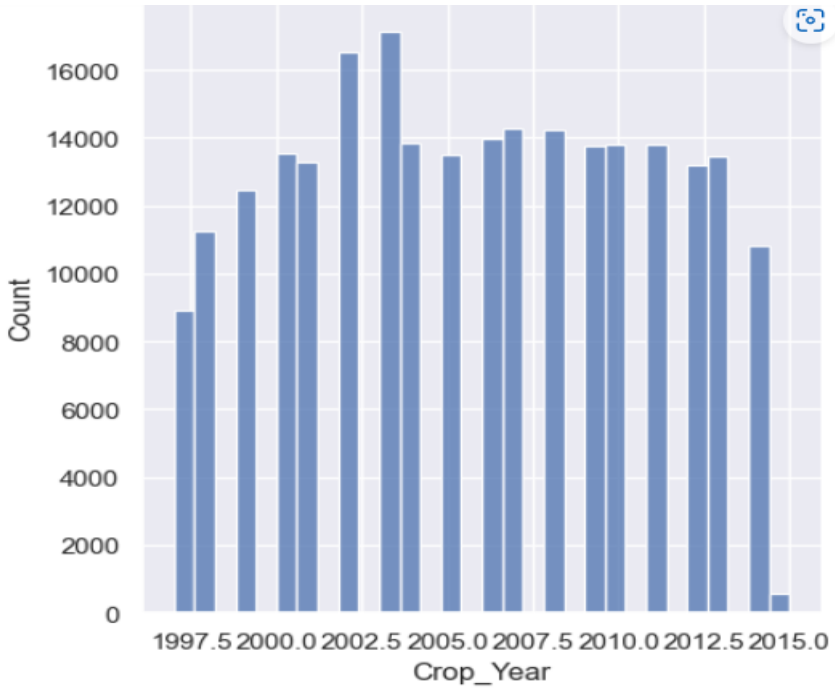
```
State_Name       0
District_Name    0
Crop_Year        0
Season           0
Crop             0
Area             0
Production       0
dtype: int64
```

4. Exploratory Data Analysis

Exploratory Data Analysis is a very important step before training the model. In this section, we will use some visualizations to understand the relationship of the target variable with other features.

Let's first plot the distribution of the target variable Crop_Year. We will use the displot function from the seaborn library.

```
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.displot(data['Crop_Year'], bins=30)
plt.show()
```



Using a scatter plot let's see how Production and Area features vary with Crop_Year.

```
plt.figure(figsize=(20, 5))
features = ['Production', 'Area']
target = data['Crop_Year']
for i, col in enumerate(features):
        plt.subplot(1, len(features) , i+1)
        x = data[col]
         y = target
         plt.scatter(x, y, marker='o')
        plt.title(col)
       plt.xlabel(col)
        plt.ylabel('Crop_Year')
```

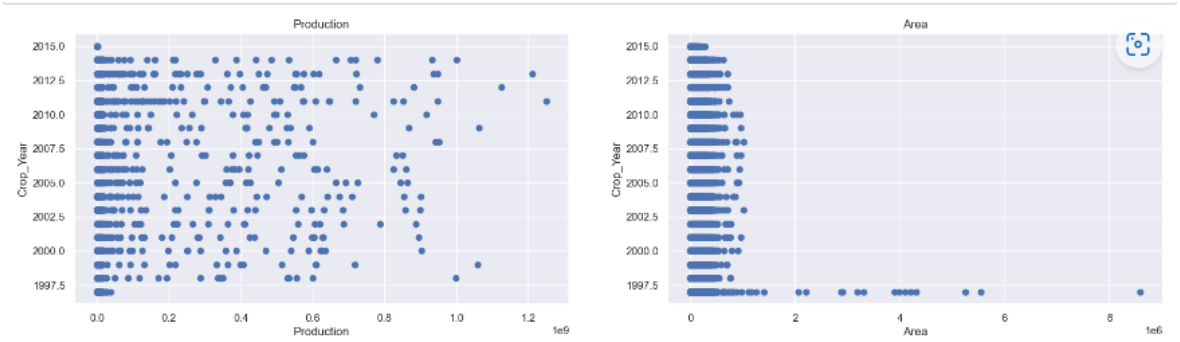


Fig: Comparison of Production and Area

5. Preparing the Data for Training the Model

We concatenate the Production and Area columns using np.c_ provided by the numpy library.

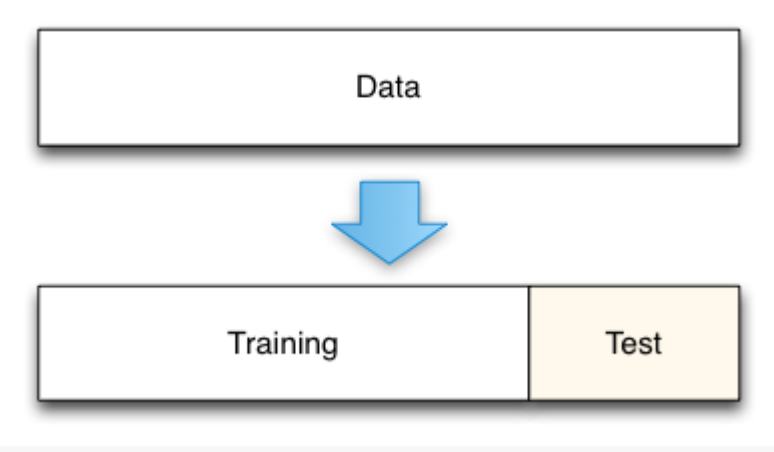
```
X = pd.DataFrame(np.c_[data['Production'], data['Area']], columns = ['Production','Area'])
Y = data['Crop_Year']
```

6. Splitting the Data into Training and Testing Sets

Next, we split the data into training and testing sets. We train the model with 80% of the samples and test with the remaining 20%. We do this to assess the model's performance on unseen data. To split the data, we use train_test_split function provided by scikit-learn library. We finally print the sizes of our training and test set to verify if the splitting has occurred properly.

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
# get the locations
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
# split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.05,
random_state=0)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(230242, 6)
(12119, 6)
(230242,)
(12119,)
```



Test/Train Data

This data is now ready to be fed to a Machine Learning Algorithm