CROPS PRODUCTION ANALYSIS

INTRODUCTION

In the provided data, we have state and district-wise crops production of India from 2000 to 2015. With the data we first did the cleaning of dataset and remove NULL values. Now, with the cleaned data, we analyzed the data and made various insights with the data. Also, we have applied Machine Learning models to predict the future production of corps in each state.

TECHNIQUES USED

1. Python (Pandas) for data cleaning
2. Power BI for KPIs and making insights of data
3. Python (Scikit-learn) for prediction the crops production


DATA CLEANING WITH
PYTHON

CHECK NULL VALUES

```python
import pandas as pd

df = pd.read_csv("dataset/Crop Production data.csv")
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
```

We can notice that, “Production” column has NULL values. In our dataset, total rows are 246091, out of them 242361 are Non-NULL. So, the remaining are NULL.

Now, we have to remove those NULL values from dataset for better model prediction for Machine Learning.

REMOVE NULL VALUES

```
df.dropna(subset=["Production"],axis=0,inplace=True)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 242361 entries, 0 to 246090
Data columns (total 7 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   State_Name     242361 non-null  object
 1   District_Name  242361 non-null  object
 2   Crop_Year      242361 non-null  int64
 3   Season         242361 non-null  object
 4   Crop           242361 non-null  object
 5   Area           242361 non-null  float64
 6   Production     242361 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 14.8+ MB
```

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

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

Here, with the help of Pandas, we removed/dropped NULL values from dataset.

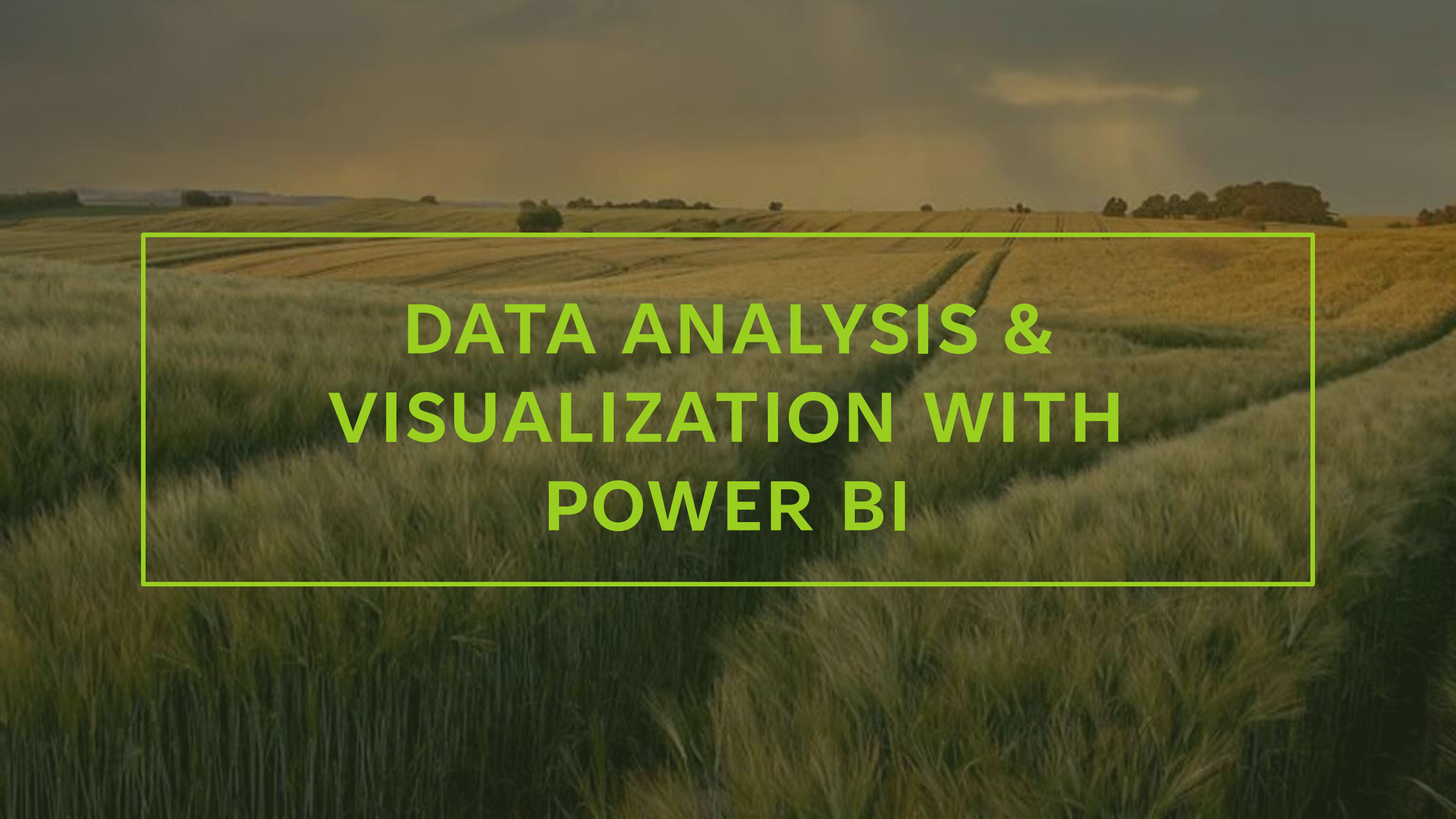
Now, we can see there is total 242361 entries and no NULL values present.

Also, we used Pandas isnull() and sum() function to check the number of NULL values in each columns and there is 0 NULL values in count.

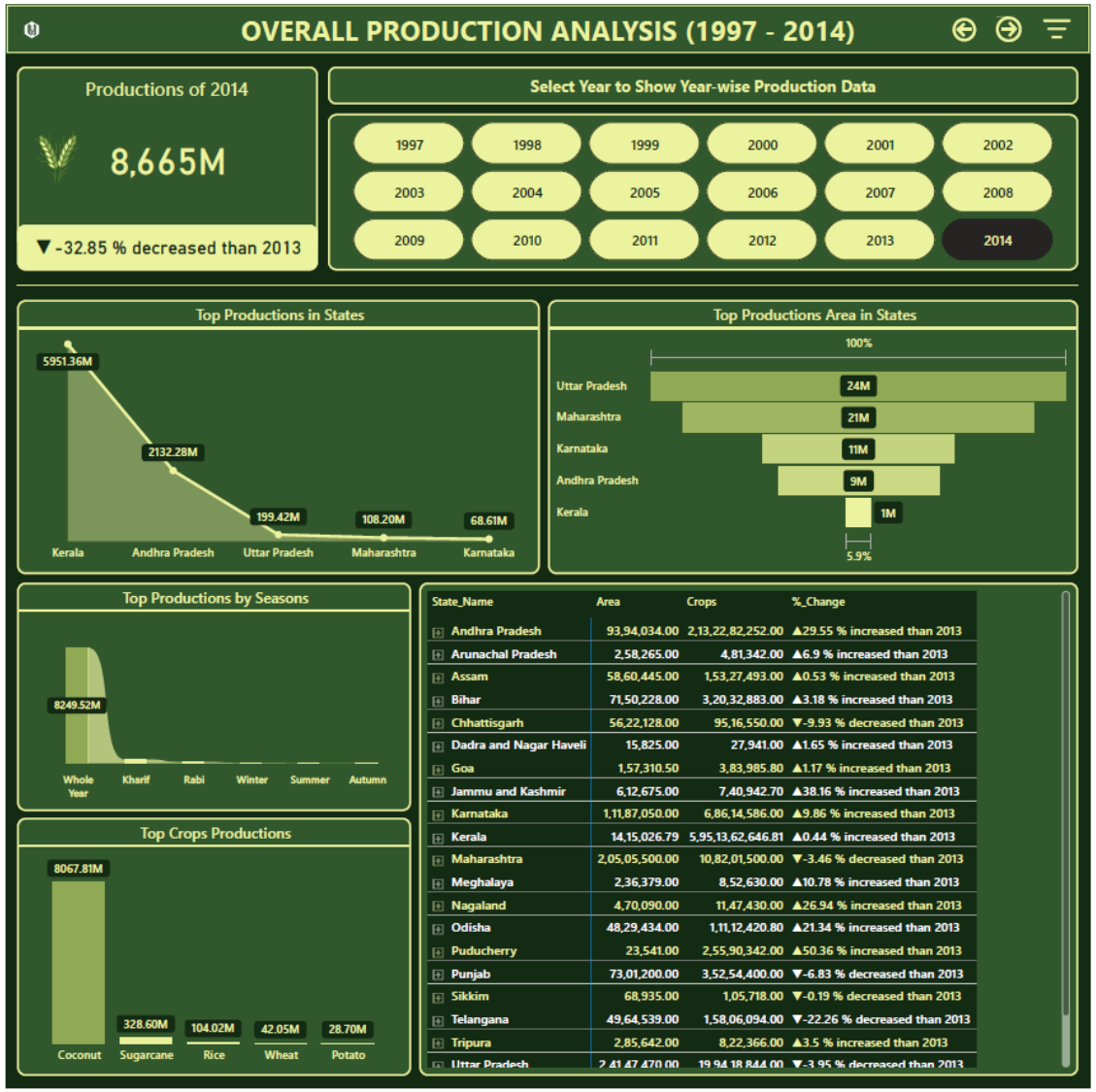
EXPORT CLEANED DATASET

```
df.to_csv("Crop_Production_Data.csv", index=False)
```

Used Pandas to_csv() function to export the cleaned dataset for Data Analysis using Power BI and also for Machine Learning prediction using Python

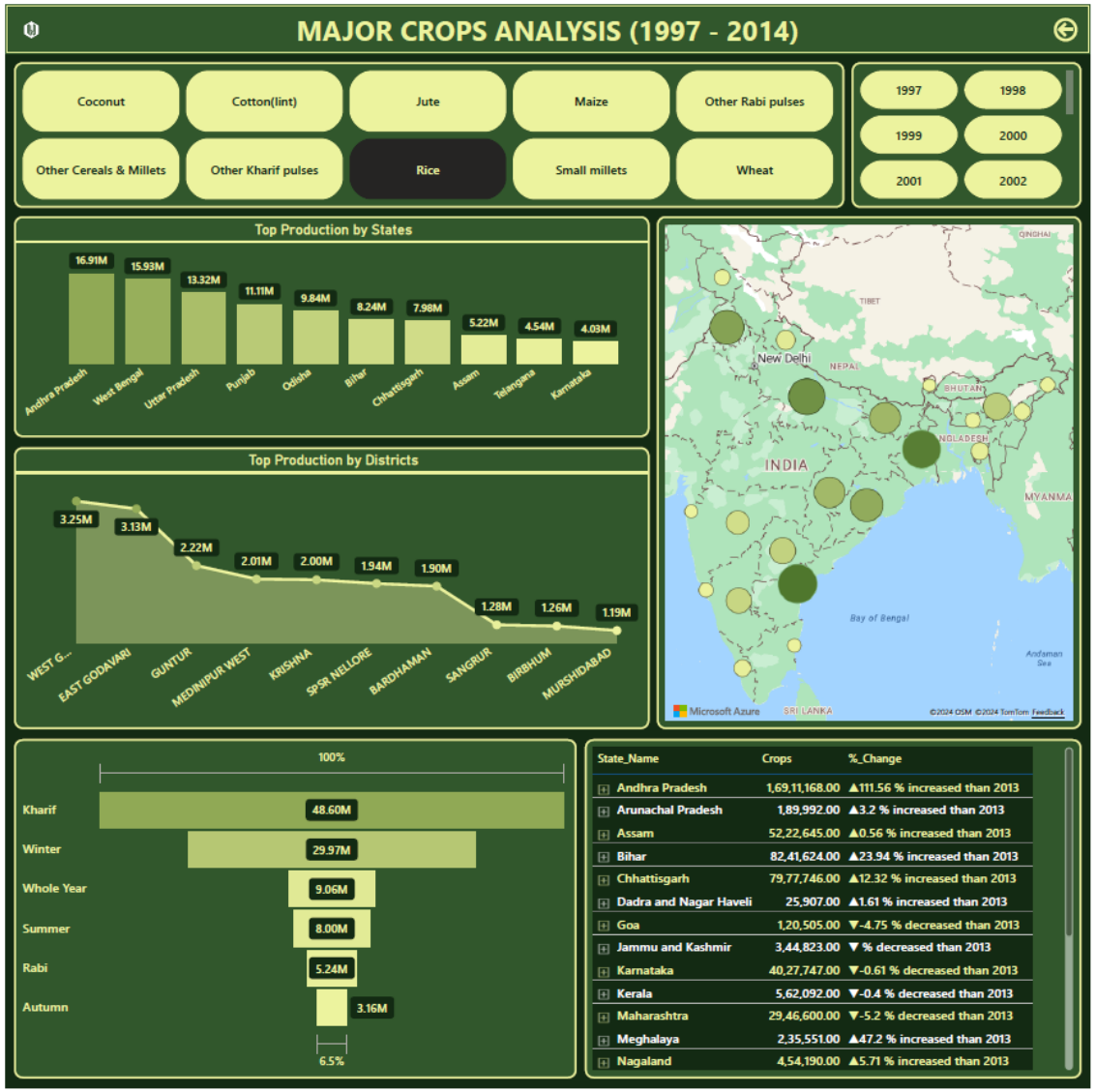

DATA ANALYSIS &
VISUALIZATION WITH
POWER BI

PRODUCTION ANALYSIS WITH POWER BI

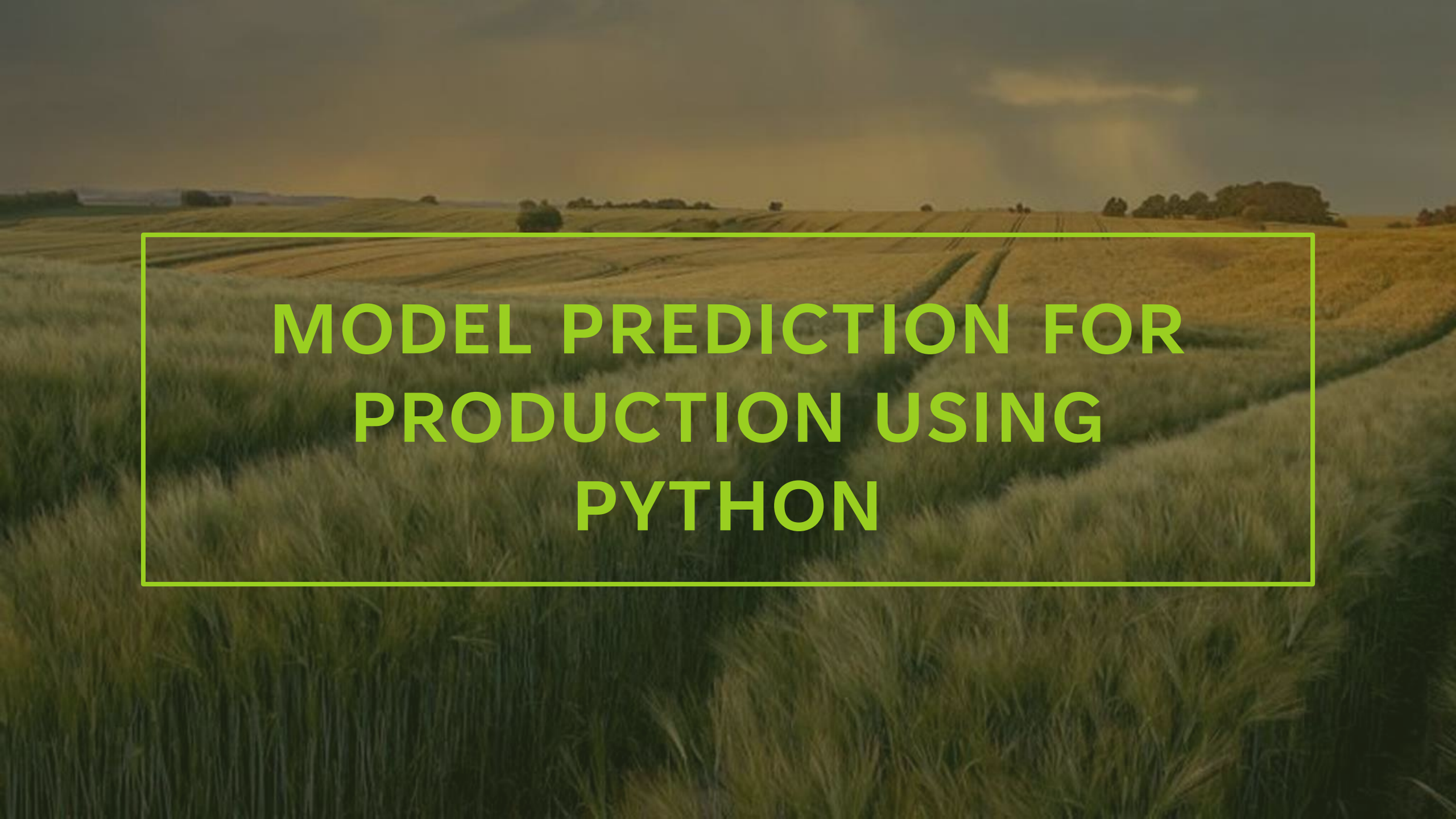


1. By default 2014 as the “Year” will be selected in slicer. We can also select other years by selecting options.
2. According to selected year, it’ll show total production of the selected year, e.g. in 2014, 8665M corps produced. Also, how much production percentage increase/decrease than last year will show.
3. Top states will show by total productions according to selected year using area chart, e.g. in 2014, Kerala was the top.
4. Top seasons will show by total productions according to selected year using ribbon chart, e.g. in 2014, Whole Year was the top.
5. Top crops will show by total productions according to selected year using clustered column chart, e.g. in 2014, Coconut was the top.
6. Also depicted the tabular form of state-wise data to show Area, Crops production and % change than previous year.

MAJOR CROPS ANALYSIS WITH POWERBI



1. By default 2014 as the “Year” will be selected in slicer. We can also select other years by selecting options.
2. By default Rice as the “Crop” will be selected in slicer. We can also select other crops by selecting options.
3. According to selected year and selected crop, it’ll show total production by states of the selected year and crop, e.g. in 2014 and for Rice, top state was Andhra Pradesh. Used clustered column chart to depict the data.
4. According to selected year and selected crop, it’ll show total production by districts of the selected year and crop, e.g. in 2014 and for Rice, top district was West Godavari. Used area chart to depict the data.
5. Top seasons will show by total productions according to selected year and crop using funnel chart, e.g. in 2014 and for Rice, Kharif was the top.
6. Also depicted the tabular form of state-wise data to show Crops production and % change than previous year according to selected year and crop.


MODEL PREDICTION FOR
PRODUCTION USING
PYTHON

PERCENTAGE OF PRODUCTION

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
import xgboost as xgb
from sklearn.metrics import mean_squared_error, r2_score

data = pd.read_csv("dataset/Cleaned_Crop_Production_Data.csv")

sum_maxp = data["Production"].sum()
data["percent_of_production"] = data["Production"].map(lambda x:(x/sum_maxp)*100)

data[:5]
```

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	percent_of_production
0	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Arecanut	1254.0	2000.0	1.416670e-06
1	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Other Kharif pulses	2.0	1.0	7.083351e-10
2	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Rice	102.0	321.0	2.273756e-07
3	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Banana	176.0	641.0	4.540428e-07
4	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Cashewnut	720.0	165.0	1.168753e-07

We imported essential libraries and load the cleaned dataset which we cleaned at first step using python. Then we sum the values of column – “Production” and used lambda function to calculate percentage of production of each row data and made a new column named, “percent_of_production”.

DROP UNNECESSARY COLUMN & OHE

```
# Drop unnecessary columns
data1 = data.drop(["District_Name"],axis=1)

# Applied OHE - One Hot Encoding
data_dum = pd.get_dummies(data1, dtype=int)
data_dum[:5]
```

Here we dropped unnecessary column – “District_Name” and then applied OHE using Pandas get_dummies() method and depicted the 5 rows of dataset.

One hot encoding is a technique that we use to represent categorical variables as numerical values in a machine learning model. It allows the use of categorical variables in models that require numerical input. It can improve model performance by providing more information to the model about the categorical variable.

	Crop_Year	Area	Production	percent_of_production	State_Name_Andaman and Nicobar Islands	State_Name_Andhra Pradesh	State_Name_Arunachal Pradesh	State_Name_Assam	State_Name_Bihar	State_Name_Chandigarh	...
0	2000	1254.0	2000.0	1.416670e-06	1	0	0	0	0	0	...
1	2000	2.0	1.0	7.083351e-10	1	0	0	0	0	0	...
2	2000	102.0	321.0	2.273756e-07	1	0	0	0	0	0	...
3	2000	176.0	641.0	4.540428e-07	1	0	0	0	0	0	...
4	2000	720.0	165.0	1.168753e-07	1	0	0	0	0	0	...

5 rows × 167 columns

SPLIT DATASET FOR ML MODELS

```
x = data_dum.drop("Production",axis=1)
y = data_dum[["Production"]]

x_train,x_test,y_train,y_test = train_test_split(x,y,
                                test_size=0.33,
                                random_state=42)

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 : (162381, 166)
x_test : (79980, 166)
y_train : (162381, 1)
y_test : (79980, 1)
```

For training the Machine Learning models, we assign whole dataframe except “Production” column to variable “x” and for testing we assign only “production” column as dataframe to variable “y”.

Then we used “train_test_split” method of scikit-learn library to split the data into train and test format.

Here is the shape of train & test dataset.

ML MODELS - XGB PREDICTION ACCURACY

```
xgbr = xgb.XGBRegressor(verbosity=0)
xgbr.fit(x_train,y_train)

preds = xgbr.predict(x_test)

mean_squared_error(y_test,preds)
r2_score(y_test,preds)
```

Here, we used XGB Regressor model to predict the crops productions.

It gives us :

MSE -> np.float64(142408674129 39.752)

R2 Score –> 0.95

ML MODELS – DECISION TREE PREDICTION ACCURACY

```
regressor = DecisionTreeRegressor(random_state=42)
regressor.fit(x_train,y_train)

preds = regressor.predict(x_test)

mean_squared_error(y_test,preds)
r2_score(y_test,preds)
```

Here, we used Decision Tree Regressor model to predict the crops productions.

It gives us :

MSE -> np.float64(378537510326.2657)

R2 Score –> 0.99

CONCLUSION

We have noticed that, Decision Tree providing more accuracy than XGB. So, we will use Decision Tree model for Machine Learning prediction for production of crops.

THANK YOU

Akashdeep Dam

akashdeepdam@gmail.com

UNID : òâ Ŗ