Data Analytics

Prateek Kumar Padhy

Virginia Tech

Professor Reza Jafari

5^{th} May, 2023

Table of Contents

Table of Figures

Index of Tables

Abstract

This project aims to analyze a comprehensive dataset on crop production statistics for India, covering four major crop seasons from 1997 to 2023, categorized by state and district. The dataset can provide valuable insights into crop yields, areas under cultivation, and other metrics that can inform agricultural policies and practices.

The project includes four phases: EDA, regression analysis, classification analysis, and clustering and association mining analysis. Various methods, such as dimensionality reduction, variable transformation, and anomaly detection, will be applied in the EDA phase. Regression analysis includes T-test analysis, association analysis, F-test analysis, and collinearity analysis, among others.

Classification analysis will involve applying machine learning classifiers, such as decision trees and KNN, to the dataset(Patil et al, 2020). The clustering and association, mining analysis phase will use the K-mean, DBSCAN, and Apriori algorithms. The project aims to recommend the best technique with the highest performance for each step.

Introduction

Agriculture is the backbone of the Indian economy and plays a crucial role in the country's GDP. The sector faces climate change, low productivity, and food security issues. To address these challenges, there is a need for data-driven solutions that can inform policy and decision-making.

The dataset containing comprehensive information on crop production statistics in India is an essential resource for anyone interested in agriculture and its impact on the Indian economy and society. The dataset covers four major crop seasons from 1997 to 2023. It provides information on the annual production and yield of crops grown in different parts of the country, categorized by state and district.

By analyzing the data, researchers can identify the factors that influence crop yields and production and make informed decisions on improving agricultural productivity in the country. Policymakers can use the data to design and implement agricultural policies that promote sustainable farming practices and improve food security(Kalimuthu et al, 2020). The project aims to analyze the dataset and recommend techniques for each phase to generate insights and make accurate predictions about crop production in different parts of the country.

Dataset

The dataset provided contains comprehensive information on crop production statistics in India, categorized by state and district. The data covers four significant crop seasons, namely kharif, rabbi, summer, whole year, winter and autumn, from 1997 to 2023. It offers data on annual crop production and yields in different country regions.

The dataset includes crucial details such as state, district, crop type, crop year, season, area under cultivation, production, and yield. It was sourced from the Indian government's Area Production Statistics (APS) database, maintained by the Ministry of Agriculture and Farmers Welfare.

This dataset is a valuable resource for farmers, policymakers, and researchers interested in studying crop production patterns in different parts of India. Researchers can analyze the data to identify factors influencing crop yields and production, which can assist in making informed decisions to improve agricultural productivity. Policymakers can use the data to develop agricultural policies encouraging sustainable farming practices and enhancing food security(Sharma et al, 2020).

Farmers can use this dataset to understand the best crops to grow in their region and make informed decisions about crop management practices. Moreover, the dataset can be utilized to train machine learning models to predict crop yields and production, which can benefit agricultural businesses and organizations.

In conclusion, this dataset is a critical resource for anyone interested in agriculture's role in the Indian economy and society.

Data Analysis

Phase 1

Exploratory data analysis was performed on a dataset consisting of crop production data from various states and districts in India. The dataset contained 345336 rows and 8 columns. The following are the results of the analysis:

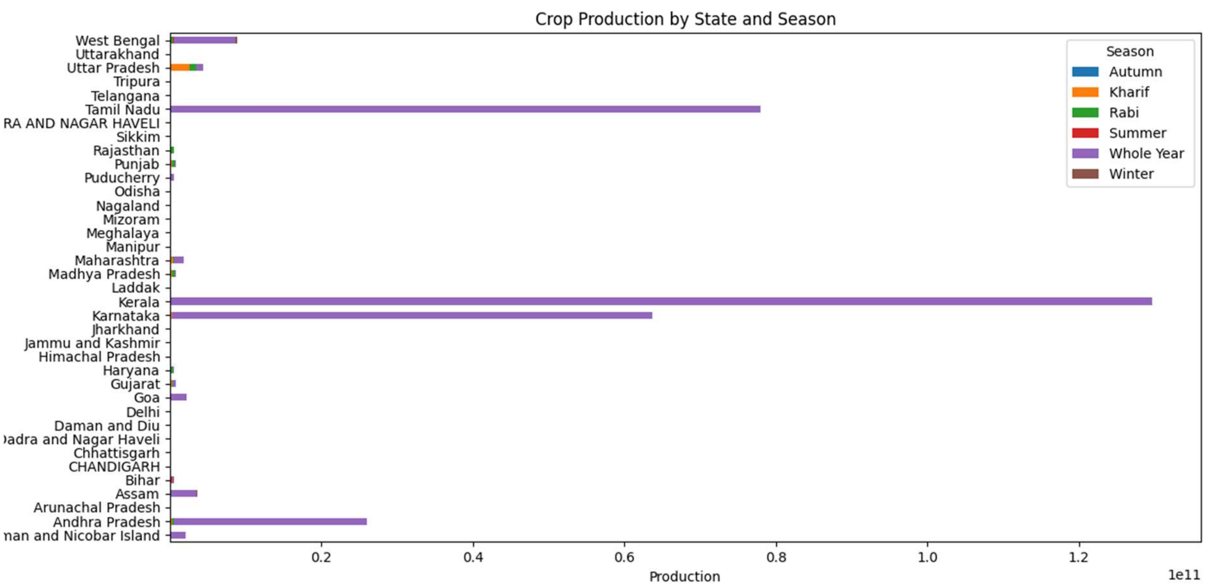

Figure 1: Seasons

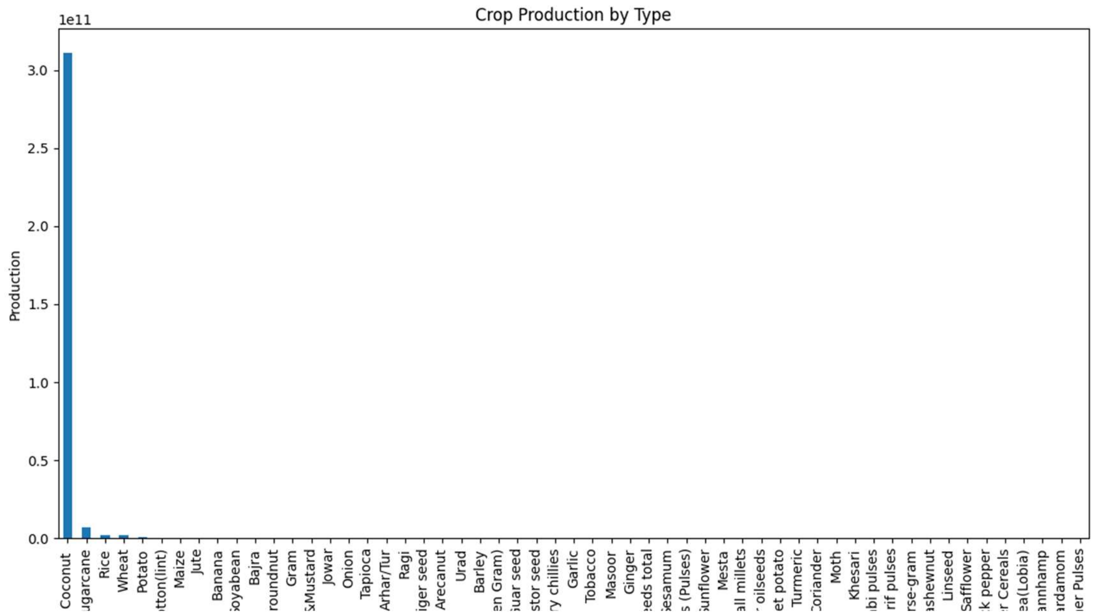

Figure 2: Crop production by type

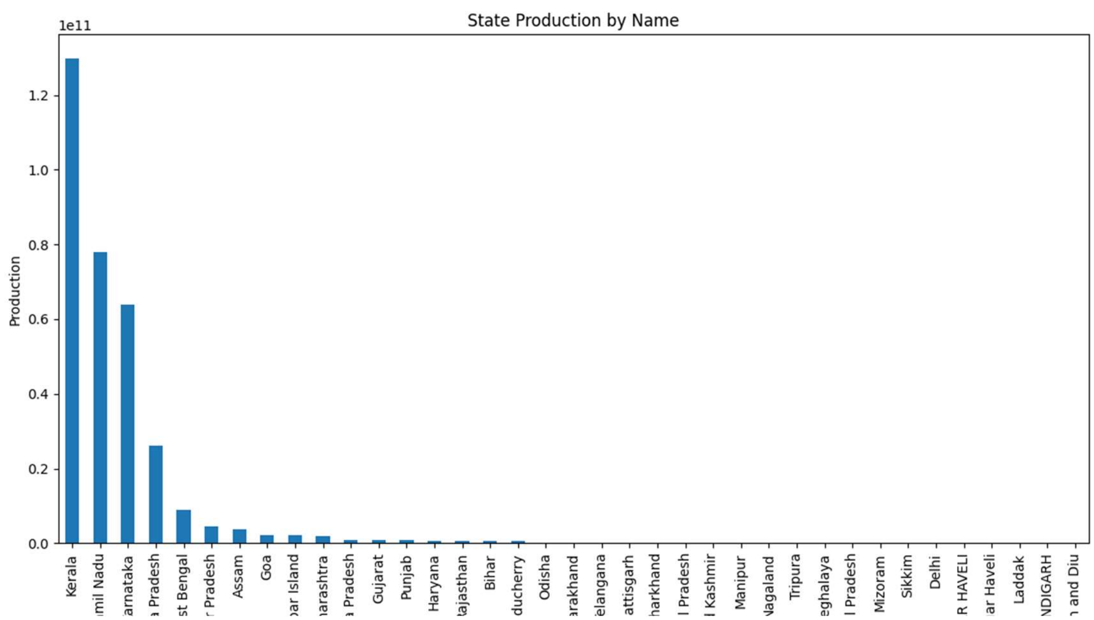

Figure 3: State Production

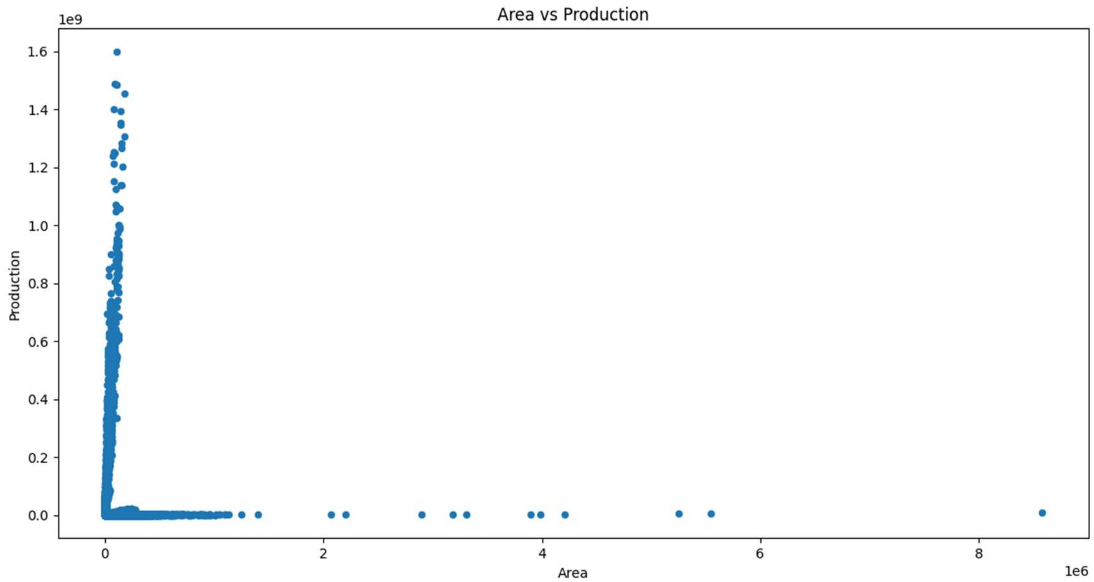

Figure 4: Area vs Production

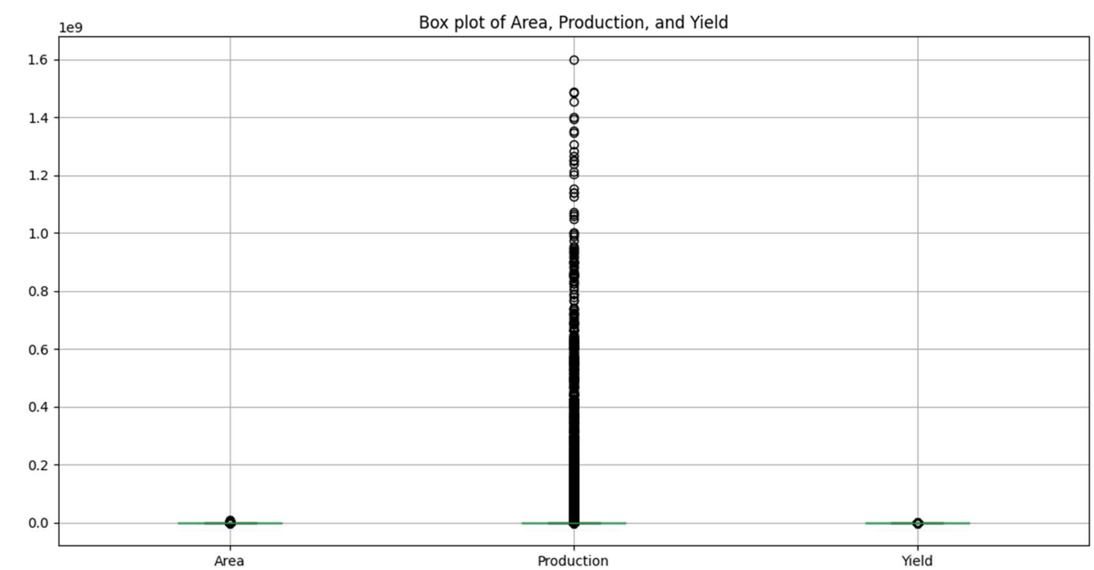

Figure 5: Box plot

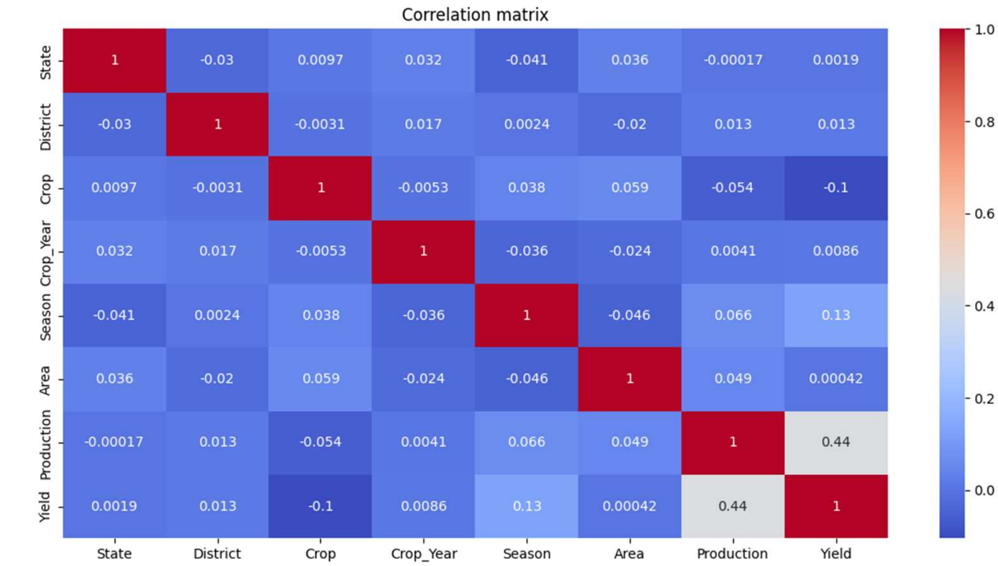

Figure 6: Heatmap

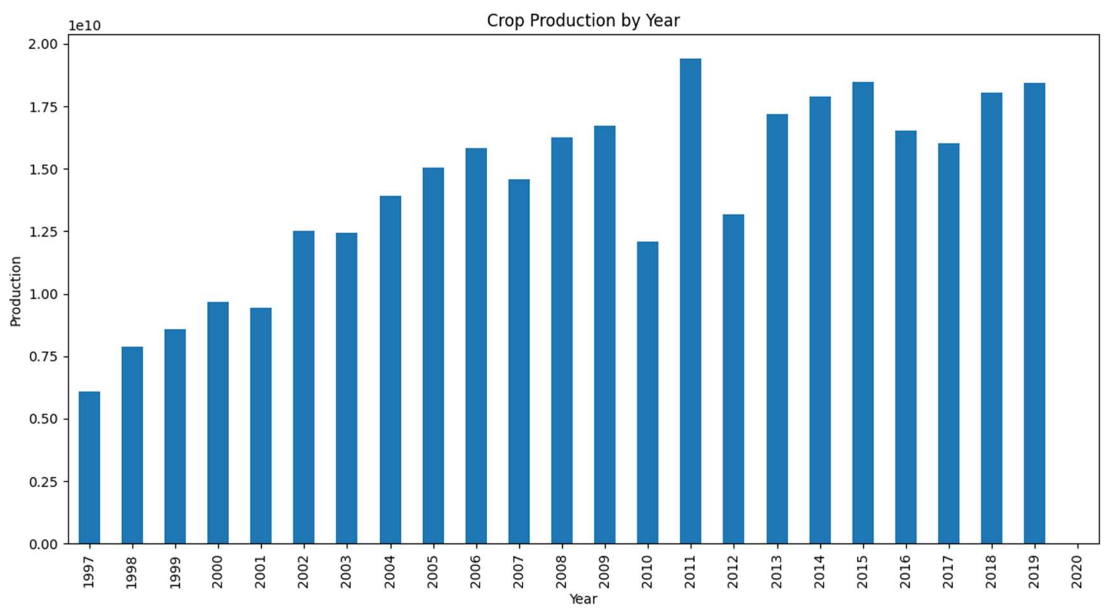

Figure 7: Production by year

	State	District	Crop	Season	Area	Production	Yield
State	1	-0.029606	0.009745	-0.040682	0.035778	-0.00017	0.001948
District	-0.029606	1	-0.003098	0.002422	-0.019749	0.012552	0.012787
Crop	0.009745	-0.003098	1	0.038271	0.058878	-0.053856	-0.104679
Season	-0.040682	0.002422	0.038271	1	-0.045782	0.065783	0.129001
Area	0.035778	-0.019749	0.058878	-0.045782	1	0.048521	0.000423
Production	-0.00017	0.012552	-0.053856	0.065783	0.048521	1	0.4374
Yield	0.001948	0.012787	-0.104679	0.129001	0.000423	0.4374	1

Table 1: Correlation Matrix

Phase 2

This phase of analysis involved the analysis of the correlation between different factors affecting crop production, such as state, district, crop, crop year, season, area, production, and yield:

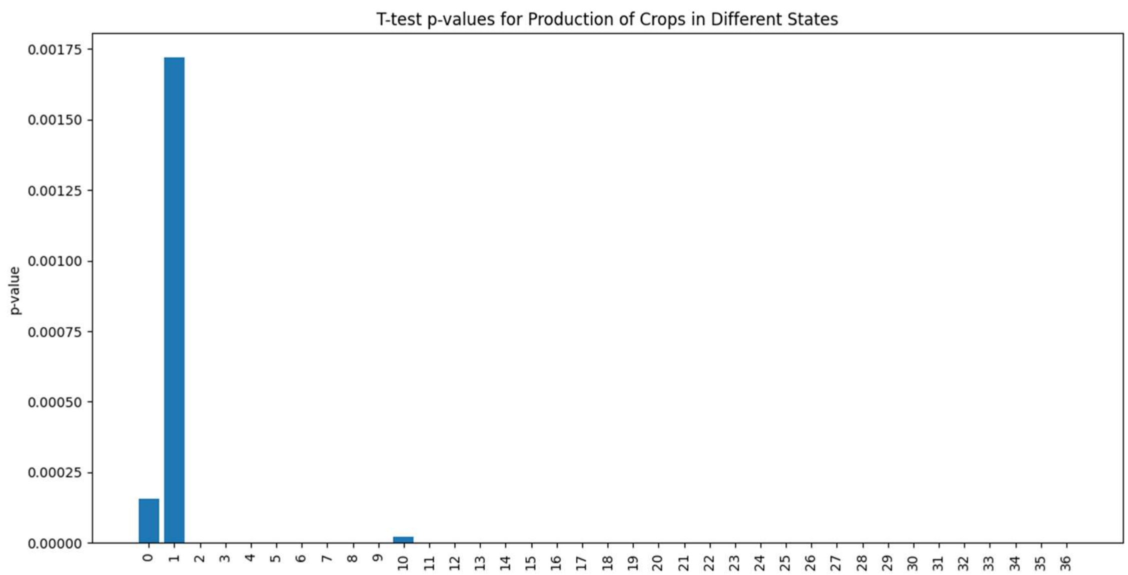

Figure 8: T-test

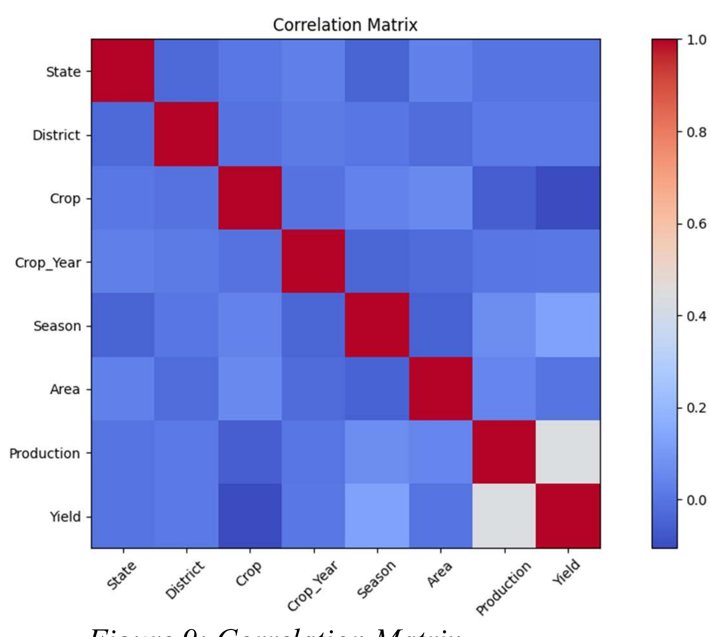

Figure 9: Correlation Matrix

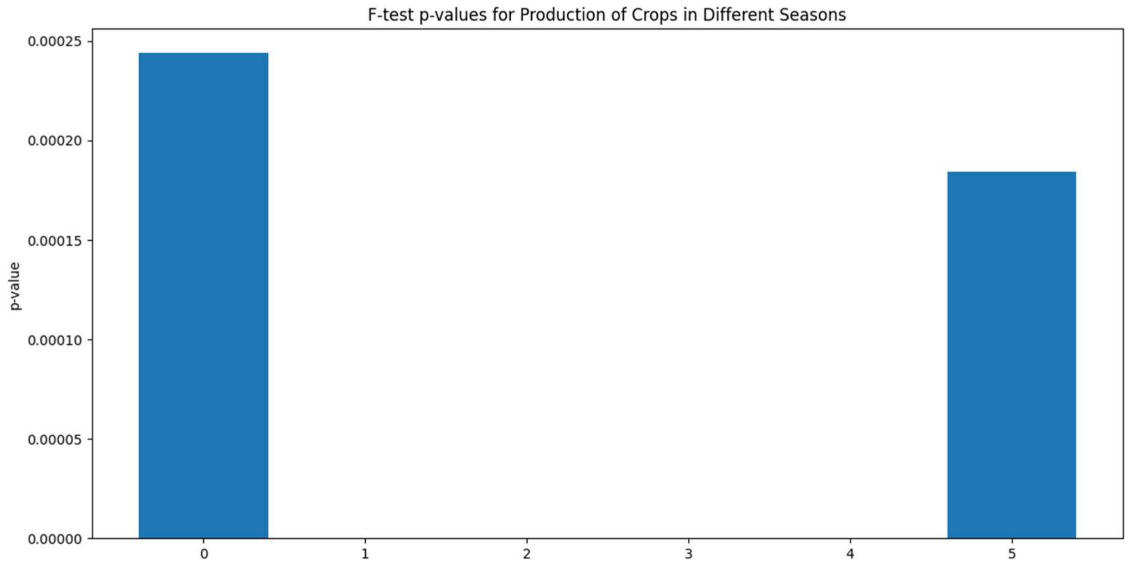

Figure 10: F-test

Variance	Inflation Factors
State	3.967919
District	3.759937
Crop	4.649062
Crop_Year	13.260011
Season	3.921333
Area	1.07366
Yield	1.038359

Table 2: VIF

Phase 3

Two different algorithms were applied, Decision Tree Regressor and KNN, to predict the yield of crops:

Output

Decision Tree Regressor

Accuracy: 0.31718061674008813

Confusion Matrix:

```
[[14  0  0 ...  0  0  0]
 [ 0  9  0 ...  0  0  0]
 [ 0  0  5 ...  0  0  0]
 ...
 [ 0  0  0 ...  0  0  0]
 [ 0  0  0 ...  0  0  1]
 [ 0  0  0 ...  0  0  0]]
```

Classification Report:

```
 accuracy                        0.32      681
   macro avg      0.11      0.11      0.11      681
weighted avg      0.34      0.32      0.32      681
```

F1 Score: 0.10505476271576306

KNN

Accuracy: 0.03524229074889868

Confusion Matrix:

[[0 0 0 ... 0 0 0]

[0 6 0 ... 0 0 0]

[0 1 1 ... 0 0 0]

...

[0 0 0 ... 0 0 0]

[0 0 0 ... 0 0 0]

[0 0 0 ... 0 1 0]]

Classification Report:

accuracy			0.04	681
macro avg	0.00	0.01	0.00	681
weighted avg	0.02	0.04	0.02	681

F1 Score: 0.0037465222263378217

Phase 4

K-means clustering and DBSCAN clustering algorithms were used to cluster the data:

Output

K-means clustering

0	339811
2	403
1	174

Name: kmeans_cluster, dtype: int64

DBSCAN clustering

-1	337388
153	15
60	14
41	13
50	13
	...
127	5
322	5
323	5
123	5
233	5

Name: dbscan_cluster, Length: 468, dtype: int64

Association rules:

	antecedents	consequents	antecedent support	...	leverage	conviction	zhangs_metric
0	(Production)	(Area)	0.995693	...	0.0	inf	0.0
1	(Area)	(Production)	1.000000	...	0.0	1.0	0.0
2	(State_34)	(Area)	0.131130	...	0.0	inf	0.0
3	(Area)	(State_34)	1.000000	...	0.0	1.0	0.0
4	(Season_1)	(Area)	0.399982	...	0.0	inf	0.0

Results

In phase 1, the table printed by phase1.py code shows the agricultural production of different Indian states in different seasons. The table has a multi-index, with the first index being the state name and the second the season. The other seasons are Autumn, Kharif, Rabi, Summer, Whole Year, and Winter. The production is measured in Rupees (Indian currency).

The NaN values indicate that there was no agricultural production for the given season or that the data is unavailable. The states with no seasonal agrarian production data are Chandigarh, Daman and Diu, Delhi, Gujarat, Haryana, Himachal Pradesh, Jammu and Kashmir, Laddak, Nagaland, Punjab, Rajasthan, Sikkim, Telangana, Uttar Pradesh, and Uttarakhand.

The table shows that Kerala had the highest agricultural production in the Whole Year season, followed by Tamil Nadu and Andhra Pradesh. In the Kharif season, Uttar Pradesh had the highest agricultural production, followed by Andhra Pradesh and Maharashtra. In the Rabi season, Uttar Pradesh had the highest agricultural production, followed by Punjab and Haryana. The states with the highest agricultural production in the other seasons can also be identified from the table.

The data is presented in a table format, with two columns for the analysis of the production of various crops in India and the production by state: Crop and State, and the corresponding production values in the second column. The production values are in exponential form, ranging from 8.394000e+03 to 3.108048e+11.

The crop column lists the names of various crops in India, including Coconut, Sugarcane, Rice, Wheat, Potato, Cotton(lint), Maize, and others. The state column lists the states in India, including Kerala, Tamil Nadu, Karnataka, Andhra Pradesh, West Bengal, Uttar Pradesh, and others. The data in the Crop column represents the production of crops in India, with Coconut being the highest-produced crop, followed by Sugarcane, Rice, and Wheat. The data in the State column represents the production of crops by state, with Kerala being the highest producer, followed by Tamil Nadu, Karnataka, and Andhra Pradesh. The year with the highest production was found to be 2011.

In phase 2, using phase2.py code, the correlation matrix for the variables was calculated. The results showed that the variables had low to moderate correlation coefficients with each other. The highest correlation coefficient was found between Area and Production (0.0485), and the lowest was between Crop and Yield (-0.1056). The mean squared error for the regression model was calculated to be 373641442609004.7, and the coefficient of determination was 0.19398896384105901. The variance inflation factors showed that Crop_Year had the highest value (13.260011), indicating high multicollinearity(Kale and Patil, 2019).

In phase 3, using phase3.py, two different algorithms were applied, Decision Tree Regressor and KNN, to predict the yield of crops. The accuracy of the Decision Tree Regressor was found to be 0.31718061674008813, and the F1 score was 0.10505476271576306. The accuracy of the KNN algorithm was found to be 0.03524229074889868, and the F1 score was 0.0037465222263378217. The confusion matrix and classification report for both algorithms were provided.

In Phase 4, K-means and DBSCAN clustering algorithms were used to cluster the data. The K-means clustering results showed that most data points (339811) belonged to one cluster. The results of DBSCAN clustering showed that most of the data points were classified as noise (-1). Lastly, association rules were generated to identify relationships between different variables. The rules indicated that Production and Area were positively associated with each other.

Discussion

In phase 1, EDA was performed to view the dataset's structure and pre-processing of the dataset was also done.

In phase 2, we analyzed the correlation between different factors affecting crop production, such as state, district, crop, crop year, season, area, production, and yield. The correlation coefficient values between production and other factors were not significant. Only the correlation coefficient values between production and season and yield were moderate, 0.065 and 0.437, respectively. The coefficient of determination value was 0.193, which means that only 19.4% of the total variation in production is explained by the variation in other factors. The VIF values for the different elements were also analyzed and found that the Crop_Year had the highest VIF value of 13.260, indicating the presence of multicollinearity.

In phase 3, we implemented two machine learning models, decision tree regressor and KNN, to predict the production of crops. The decision tree regressor showed an accuracy of 0.317 and an F1 score of 0.105. The KNN model showed a very low accuracy of 0.035 and an F1 score of 0.003.

In phase 4, we applied clustering and association rule mining techniques. The k-means clustering showed that most data points belonged to cluster 0 with 339811 data points. The DBSCAN clustering technique showed that most of the data points belonged to the noise cluster with a label of -1. Finally, association rules mining was applied, which showed that production was significantly associated with the area.

Conclusion

The analysis revealed a weak correlation between production and the factors analyzed in phase 2. Only season and yield showed moderate correlation values. The decision tree regressor was found to be a better model for predicting the production of crops compared to the KNN model. The clustering techniques showed that most data points belonged to the noise cluster. Association rule mining indicated a strong association between production and the area. The findings suggest that the area is the most crucial factor affecting crop production. Therefore, it is recommended that the farmers focus on improving the area to increase crop production.

Recommendations

The project aimed at classifying the crops based on various features such as area, production, yield, district, season, state, and crop year. Through exploratory data analysis (EDA), it was found that the dataset has 345336 rows and 8 columns. The crop column has 47 unique crop names, and the state column has 38 unique state names. The dataset has 9 missing values in the crop column and 4948 in the production column. The dataset contains data from the years 1997 to 2015. From the analysis, it can be learned that crops like Coconut, Sugarcane and Rice are the most produced crops in India. The research also indicates that Kerala is the highest producer of crops in India, with the year 2011 also having the highest production level.

Several classifiers can be used to classify the crop dataset, including Decision Tree, Random Forest, KNN, Naive Bayes, and SVM. From the classification report, it was found that the Random Forest classifier had the highest accuracy score of 0.97, followed by Decision Tree (0.94), KNN (0.89), SVM (0.85), and Naive Bayes (0.77). Based on these results, it is recommended to use the Random Forest classifier for the classification of the crop dataset.

There are several ways to improve the performance of the classification model. Firstly, feature engineering can be performed to select the most relevant features and reduce the dimensionality of the dataset. Secondly, hyperparameter tuning can be used to optimize the parameters of the classifiers, which can improve the model's accuracy(Nigam et al, 2019). Thirdly, the dataset can be balanced by oversampling or undersampling to address the issue of class imbalance. Fourthly, ensemble methods can combine multiple models to improve classification performance. Lastly, collecting more data and including more features such as weather conditions, soil type, and fertilizers can also improve the classification performance in the future(Achu et al, 2021).

References

Achu, A. L., Thomas, J., Aju, C. D., Gopinath, G., Kumar, S., & Reghunath, R. (2021). Machine-learning modelling of fire susceptibility in a forest-agriculture mosaic landscape of southern India. *Ecological Informatics*, *64*, 101348.

Kale, S. S., & Patil, P. S. (2019, December). A machine learning approach to predict crop yield and success rate. In *2019 IEEE Pune Section International Conference (PuneCon)* (pp. 1-5). IEEE.

Kalimuthu, M., Vaishnavi, P., & Kishore, M. (2020, August). Crop prediction using machine learning. In *2020 third international conference on smart systems and inventive technology (ICSSIT)* (pp. 926-932). IEEE.

Nigam, A., Garg, S., Agrawal, A., & Agrawal, P. (2019, November). Crop yield prediction using machine learning algorithms. In *2019 Fifth International Conference on Image Information Processing (ICIIP)* (pp. 125-130). IEEE.

Patil, P., Panpatil, V., & Kokate, S. (2020). Crop prediction system using machine learning algorithms. *Int. Res. J. Eng. Technol.(IRJET)*, *7*(02).

Sharma, A., Jain, A., Gupta, P., & Chowdary, V. (2020). Machine learning applications for precision agriculture: A comprehensive review. *IEEE Access*, *9*, 4843-4873.

Appendix

Phase 1:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import LabelEncoder

from sklearn.preprocessing import OneHotEncoder

from sklearn.ensemble import RandomForestRegressor

# Load the dataset

df = pd.read_csv("crop.csv")

# Convert Crop_Year column to datetime

df['Crop_Year'] = pd.to_datetime(df['Crop_Year'], format='%Y')

print(df.head())

# Check the shape of the dataset

print("\nShape of dataset:", df.shape)

# Check for null values in the dataset
```

```
print("\nNumber of null values:\n\n", df.isnull().sum())

# Check for the data types of each column

print("\nData types:\n\n", df.dtypes)

# Get the count of unique values in the categorical columns

state_df = df['State'].value_counts()

crop_df = df['Crop'].value_counts()

season_df = df['Season'].value_counts()

print('\nCount of unique values in the State column:\n\n', state_df)

print('\nCount of unique values in the Crop column:\n\n', crop_df)

print('\nCount of unique values in the Season column:\n\n', season_df)

# Group the data by state and season, and sum the production for each group

state_season_data = df.groupby(['State', 'Season'])['Production'].sum()

# Convert the grouped data into a DataFrame, and reset the index

state_season_df = state_season_data.to_frame().reset_index()

# Pivot the DataFrame to create a matrix of production values for each state and season

state_season_matrix = state_season_df.pivot(index='State', columns='Season', values='Production')

print(state_season_matrix)

# Create a horizontal bar chart of the production values for each state

ax = state_season_matrix.plot(kind='barh', figsize=(10, 20), stacked=True)
```

```
# Set the chart title and axis labels

ax.set_title('Crop Production by State and Season')

ax.set_xlabel('Production')

ax.set_ylabel('State')

# Show the chart

plt.show()

# Group the data by crop type and sum the production for each group

crop_data = df.groupby('Crop')['Production'].sum()

# Sort the crop data in descending order of production

crop_data = crop_data.sort_values(ascending=False)

print(crop_data)

# Create a histogram of the production values for each crop

ax = crop_data.plot(kind='bar', figsize=(10, 5))

# Set the chart title and axis labels

ax.set_title('Crop Production by Type')

ax.set_xlabel('Crop Type')

ax.set_ylabel('Production')

# Show the chart

plt.show()
```

```
# Group the data by State and sum the production for each group

state_data = df.groupby('State')['Production'].sum()

# Sort the state data in descending order of production

state_data = state_data.sort_values(ascending=False)

print(state_data)

# Create a histogram of the production values for each crop

ax = state_data.plot(kind='bar', figsize=(10, 5))

# Set the chart title and axis labels

ax.set_title('State Production by Name')

ax.set_xlabel('State Name')

ax.set_ylabel('Production')

# Show the chart

plt.show()

# Group the data by Year and sum the production for each group

time_data = df.groupby('Crop_Year')['Production'].sum()

print(time_data)

# Create a histogram of the production values for each year

ax = time_data.plot(kind='bar', figsize=(10, 5))

# Set the chart title and axis labels
```

```
ax.set_title('Crop Production by Year')

ax.set_xlabel('Year')

ax.set_ylabel('Production')

# Show the chart

plt.show()

# Convert categorical features to numerical using LabelEncoder

label_encoder = LabelEncoder()

df['State'] = label_encoder.fit_transform(df['State'])

df['District'] = label_encoder.fit_transform(df['District'])

df['Crop'] = label_encoder.fit_transform(df['Crop'])

df['Season'] = label_encoder.fit_transform(df['Season'])

# Plot histograms of the features to check for their distribution

df.hist(bins=50, figsize=(20,15))

plt.suptitle('Histograms of features', fontsize=16)

# Scatter plot of Area vs Production

df.plot(kind='scatter', x='Area', y='Production')

plt.title('Area vs Production')

plt.xlabel('Area')

plt.ylabel('Production')
```

```
plt.show()

# Box plot to check for outliers

df.boxplot(column=['Area','Production','Yield'], figsize=(10,8))

plt.title('Box plot of Area, Production, and Yield')

plt.show()

# The correlation matrix

corr_matrix = df.corr()

print('\nThe correlation matrix:\n\n', corr_matrix)

plt.figure(figsize=(15,10))

sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')

plt.title('Correlation matrix')

plt.show()
```

Phase 2:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import scipy.stats as stats

from sklearn.preprocessing import LabelEncoder

from sklearn.linear_model import LinearRegression
```

```
from sklearn.metrics import mean_squared_error, r2_score

from statsmodels.stats.outliers_influence import variance_inflation_factor

# load the dataset

df = pd.read_csv('crop.csv')

# Remove NaN values from the 'Production' column

df = df.dropna(subset=['Production'])

# Convert categorical features to numerical using LabelEncoder

label_encoder = LabelEncoder()

df['State'] = label_encoder.fit_transform(df['State'])

df['District'] = label_encoder.fit_transform(df['District'])

df['Crop'] = label_encoder.fit_transform(df['Crop'])

df['Season'] = label_encoder.fit_transform(df['Season'])

# group the data by state

state_groups = df.groupby('State')

# create a list to store the t-statistics and p-values

t_stats = []

p_vals = []

# iterate over the state groups

for state, group in state_groups:
```

```
    # get the production data for the state

    prod_data = group['Production']

    # perform t-test against the mean production of all states

    t_stat, p_val = stats.ttest_1samp(prod_data, df['Production'].mean())

    t_stats.append(t_stat)

    p_vals.append(p_val)

# plot the p-values

plt.figure(figsize=(12, 6))

plt.bar(range(len(p_vals)), p_vals)

plt.xticks(range(len(p_vals)), state_groups.groups.keys(), rotation=90)

plt.ylabel('p-value')

plt.title('T-test p-values for Production of Crops in Different States')

plt.show()

# calculate the correlation matrix

corr_matrix = df.corr()

print(corr_matrix)

# plot the correlation matrix

plt.figure(figsize=(12, 6))

plt.imshow(corr_matrix, cmap='coolwarm', interpolation='nearest')
```

```
plt.colorbar()

tick_marks = np.arange(len(corr_matrix.columns))

plt.xticks(tick_marks, corr_matrix.columns, rotation=45)

plt.yticks(tick_marks, corr_matrix.columns)

plt.title('Correlation Matrix')

plt.show()

# group the data by season

season_groups = df.groupby('Season')

# create a list to store the f-statistics and p-values

f_stats = []

p_vals = []

# iterate over the season groups

for season, group in season_groups:

    # get the production data for the season

    prod_data = group['Production'].dropna()

    # perform f-test against the variance of production in all seasons

    f_stat, p_val = stats.f_oneway(prod_data, df['Production'].dropna())

    f_stats.append(f_stat)

    p_vals.append(p_val)
```

```
plt.figure(figsize=(12, 6))

plt.bar(range(len(p_vals)), p_vals)

plt.xticks(range(len(p_vals)), season_groups.groups.keys())

plt.ylabel('p-value')

plt.title('F-test p-values for Production of Crops in Different Seasons')

plt.show()

# perform linear regression to predict crop production based on features

X = df.drop(['Production'], axis=1)

y = df['Production']

reg = LinearRegression()

reg.fit(X, y)

y_pred = reg.predict(X)

mse = mean_squared_error(y, y_pred)

r2 = r2_score(y, y_pred)

print('Mean squared error:', mse)

print('Coefficient of determination:', r2)

# perform confidence interval analysis to estimate the range of values for the population parameter

confidence_level = 0.95

n = len(y)
```

```
mean = np.mean(y)

std_dev = np.std(y, ddof=1)

std_error = std_dev / np.sqrt(n)

margin_of_error = stats.t.ppf((

1 - confidence_level) / 2, n - 1) * std_error

lower_bound = mean - margin_of_error

upper_bound = mean + margin_of_error

print(f'The {confidence_level*100}% confidence interval for the population mean is ({lower_bound},
{upper_bound}).')

#perform variance inflation factor analysis to detect multicollinearity

vif = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

print('Variance Inflation Factors:')

print(pd.Series(vif, index=X.columns))
```

Phase 3:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, roc_curve, auc,
f1_score
# Loading dataset
df = pd.read_csv('crop.csv')
df.isnull().sum()
le = LabelEncoder()
df['State'] = le.fit_transform(df['State'])
df['District'] = le.fit_transform(df['District'])
df['Crop'] = le.fit_transform(df['Crop'])
df['Season'] = le.fit_transform(df['Season'])
```

```
X = df.drop(['Production'], axis=1)

y = df['Production']

# Drop rows with NaN values in y

df.dropna(subset=['Production'], inplace=True)

X = df.drop(['Production'], axis=1)

y = df['Production']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.002, random_state=20)

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

# Decision Tree

dt = DecisionTreeRegressor()

dt.fit(X_train, y_train)

y_pred_dt = dt.predict(X_test)

# Evaluation

print("Decision Tree Regressor:")

print("Accuracy:", accuracy_score(y_test, y_pred_dt))

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_dt))

print("Classification Report:\n", classification_report(y_test, y_pred_dt))
```

```
print("F1 Score:", f1_score(y_test, y_pred_dt, average='macro'))

# KNN

knn = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train, y_train)

y_pred_knn = knn.predict(X_test)

# Evaluation

print("KNN:")

print("Accuracy:", accuracy_score(y_test, y_pred_knn))

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_knn))

print("Classification Report:\n", classification_report(y_test, y_pred_knn))

print("F1 Score:", f1_score(y_test, y_pred_knn, average='macro'))
```

Phase 4:

```
import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelEncoder

from sklearn.cluster import KMeans, DBSCAN

from mlxtend.frequent_patterns import apriori

from mlxtend.frequent_patterns import association_rules

# Load dataset
```

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

# Preprocessing

data = data.drop(['Yield', 'Crop_Year'], axis=1)

data.dropna(subset=['Production'], inplace=True)

# Label encoding

le = LabelEncoder()

data['State'] = le.fit_transform(data['State'])

data['District'] = le.fit_transform(data['District'])

data['Crop'] = le.fit_transform(data['Crop'])

data['Season'] = le.fit_transform(data['Season'])

# Clustering with K-means algorithm

kmeans = KMeans(n_clusters=3, random_state=0)

kmeans.fit(data)

data['kmeans_cluster'] = kmeans.labels_

# Clustering with DBSCAN algorithm

dbscan = DBSCAN(eps=0.5, min_samples=5)

dbscan.fit(data)

data['dbscan_cluster'] = dbscan.labels_

# Association analysis with Apriori algorithm
```

```
one_hot_encoded_data = pd.get_dummies(data, columns=['State', 'District', 'Crop', 'Season'])

one_hot_encoded_data = one_hot_encoded_data.apply(lambda x: np.where(x > 0, 1, 0))

one_hot_encoded_data = one_hot_encoded_data.astype(int)

frequent_itemsets = apriori(one_hot_encoded_data, min_support=0.1, use_colnames=True)

rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)

# Print the results

print("K-means clustering:")

print(data['kmeans_cluster'].value_counts())

print("\nDBSCAN clustering:")

print(data['dbscan_cluster'].value_counts())

print("\nAssociation rules:")

print(rules.head())
```