Unveiling Insights: A report of the Analysis of Key Problem Statements

TEAM_INFINITY

Problem Statement 1: Analyze Rice Production Data In India and Predict Rice Production in Indian States/Union Territories

The dataset provided spans from the agricultural sessions of 2004-2005 to 2022-2023, detailing the quantity of rice produced annually. The task is to predict the production of rice on a state-wise or union territory-wise basis.

Analyzing Goals:

- How the production vary from state to state (through bar charts, pie charts, etc.).
- What the state-wise rate of production is.

- For which provinces there is a need to improve their production of rice.

Exploratory Data Analysis:

Steps followed in performing EDA analysis:

1. Load the dataset.
2. Explore the dataset to understand its structure and contents.
3. Visualize the production trends across states over the years.
4. Calculate the state-wise rate of production.
5. Identify states where there is a need to improve rice production.

1.Load the dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
data=pd.read_excel("/content/State_wise_rice_production_in_In
df=pd.DataFrame(data)
```

	State/Union Territory	2004-05	2005-06	2006-07	2007-08	2008-09	2009-10	2010-11	2011-12	2012-13	2013-14	2014-15
0	Andhra Pradesh	9601	11704	11872	13324	14241	10538	7882.4	7746.2	6862.4	6969.7	7233.9
1	Arunachal Pradesh	135	146.2	146.2	158.1	163.9	215.8	234.0	255.0	263.0	276.2	285.0
2	Assam	3470.7	3552.5	2916	3319	4008.5	4335.9	4736.6	4516.3	5128.5	4927.1	5222.7
3	Bihar	2472.2	3495.5	4989.3	4418.1	5590.3	3599.3	3102.1	7162.6	7529.3	5505.8	6356.7
4	Chhattisgarh	4383.3	5011.6	5041.4	5426.6	4391.8	4110.4	6159.0	6028.4	6608.8	6716.4	6322.1
5	NCT of Delhi	14.3	24	31.1	31.4	31.4	19.3	19.6	19.8	19.7	29.6	25.9
6	Goa	145.2	147.3	130.3	121.6	123.3	100.6	115.0	121.8	122.8	126.5	120.5
7	Gujarat	1238.2	1298	1390	1474	1303	1292	1496.6	1790.0	1541.0	1636.0	1830.9
8	Haryana	3023	3210	3371	3613	3298	3625	3472.0	3759.0	3976.0	3998.0	4006.0
9	Himachal Pradesh	122	112.1	123.5	121.5	118.3	105.9	128.9	131.6	125.3	120.8	125.2

2.Explore the dataset to understand its structure and contents.

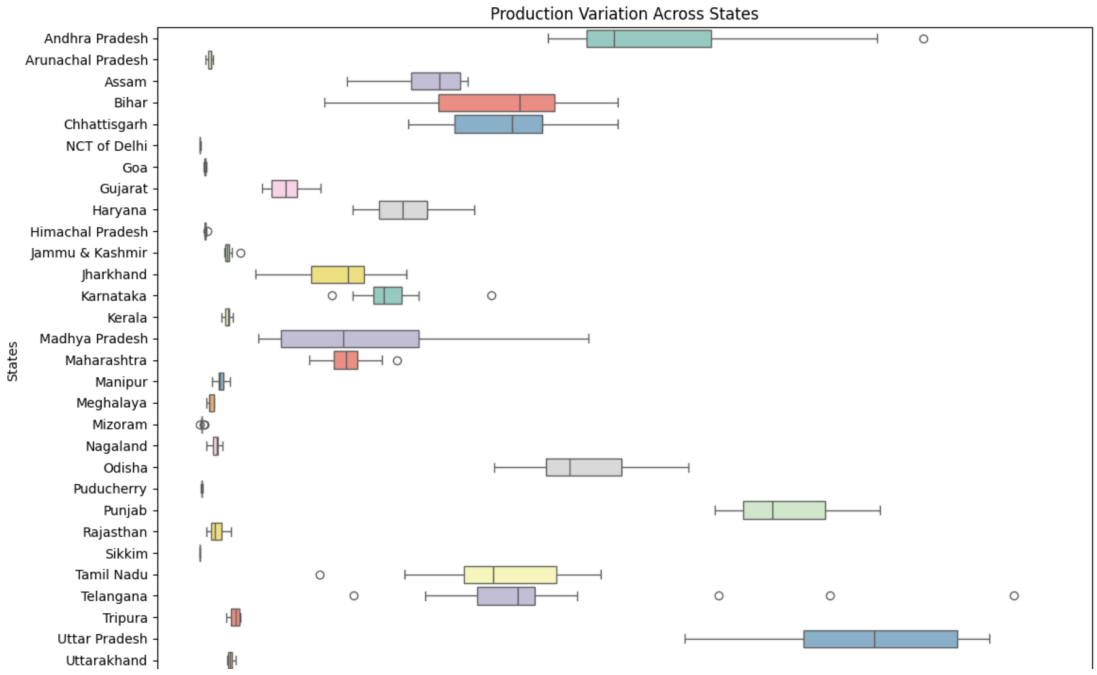
The dataset contains many missing values which will be handled in ‘Data Pre Processing phase’.

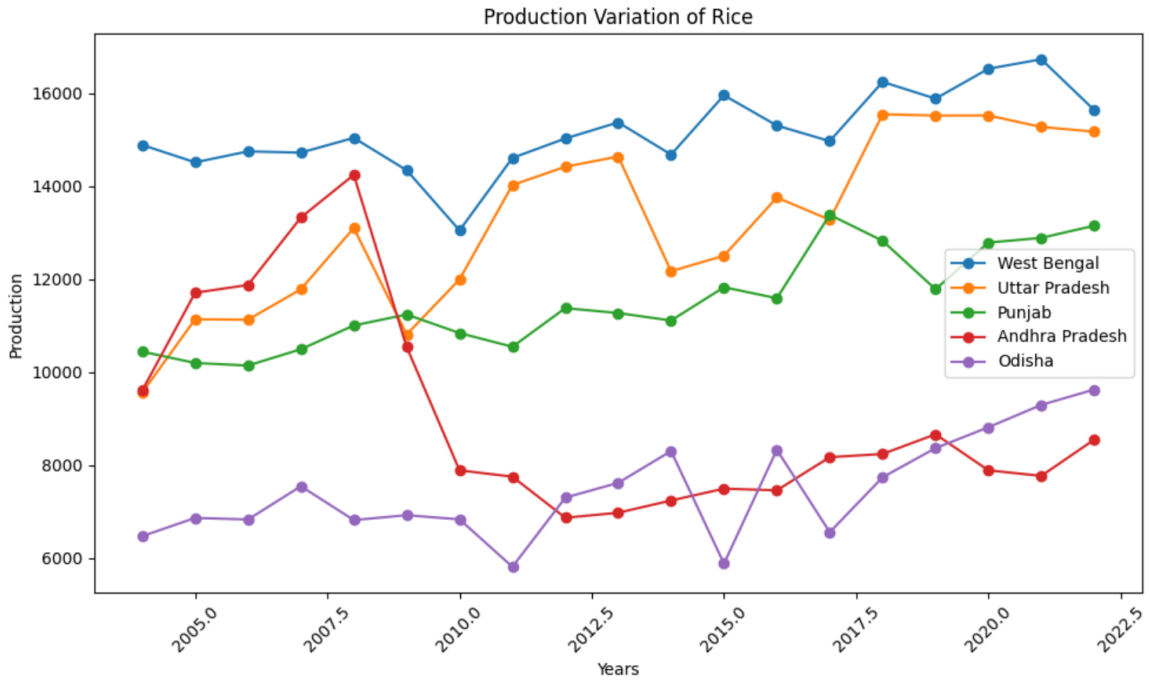
We replace all the null values with the medians of respective state’s production.

3.Visualize the production trends across states over the years.

```
df_states = df_final.drop(columns=['ALL INDIA'])

plt.figure(figsize=(12, 8))
sns.boxplot(data=df_states, orient="h", palette="Set3")
plt.title('Production Variation Across States')
plt.xlabel('Production')
plt.ylabel('States')
plt.tight_layout()
plt.show()


average_production = df_final.drop(columns=["ALL INDIA"]).mea

sorted_states = average_production.sort_values(ascending=Fals

top_states = sorted_states[:5]

plt.figure(figsize=(10, 6))

for state in top_states:
    plt.plot(years, df_final[state], label=state, marker='o',

plt.title('Production Variation of Rice')
plt.xlabel('Years')
plt.ylabel('Production')
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```

Output:





Insights

We can see a drastic fall in the production of rice by 14% in India especially in the states Andhra Pradesh , West Bengal, Uttar Pradesh due to droughts and floods.

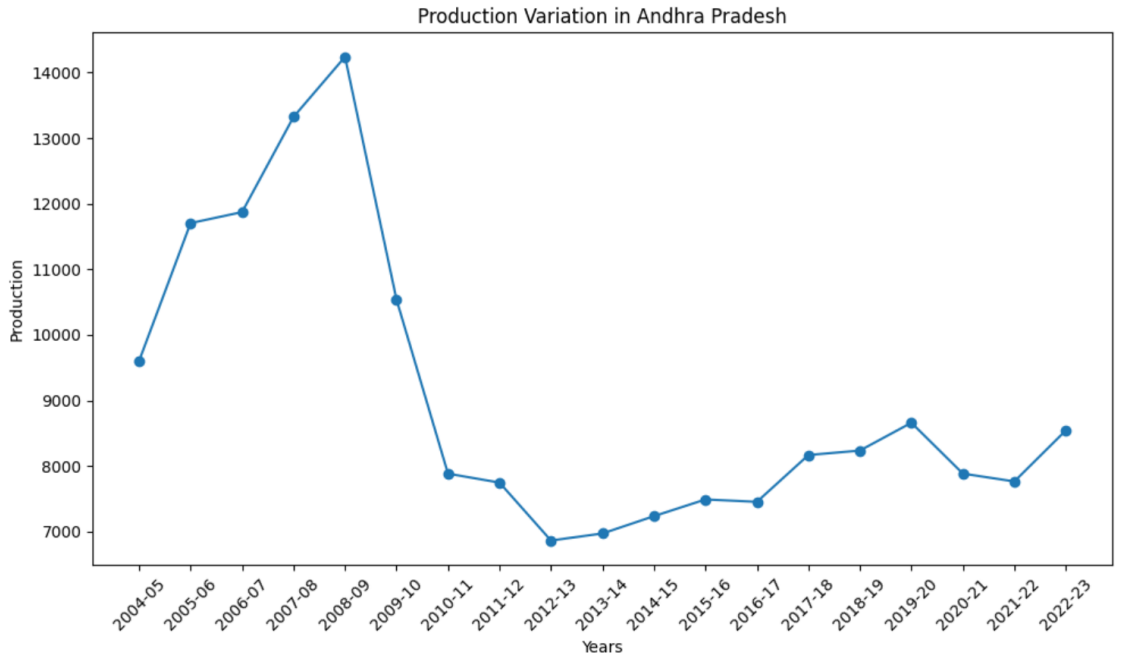
Reference: https://www.thehindu.com/sci-tech/agriculture/IGC-2009-10-Rice-output-may-fall-14-per-cent-in-India/article16891650.ece

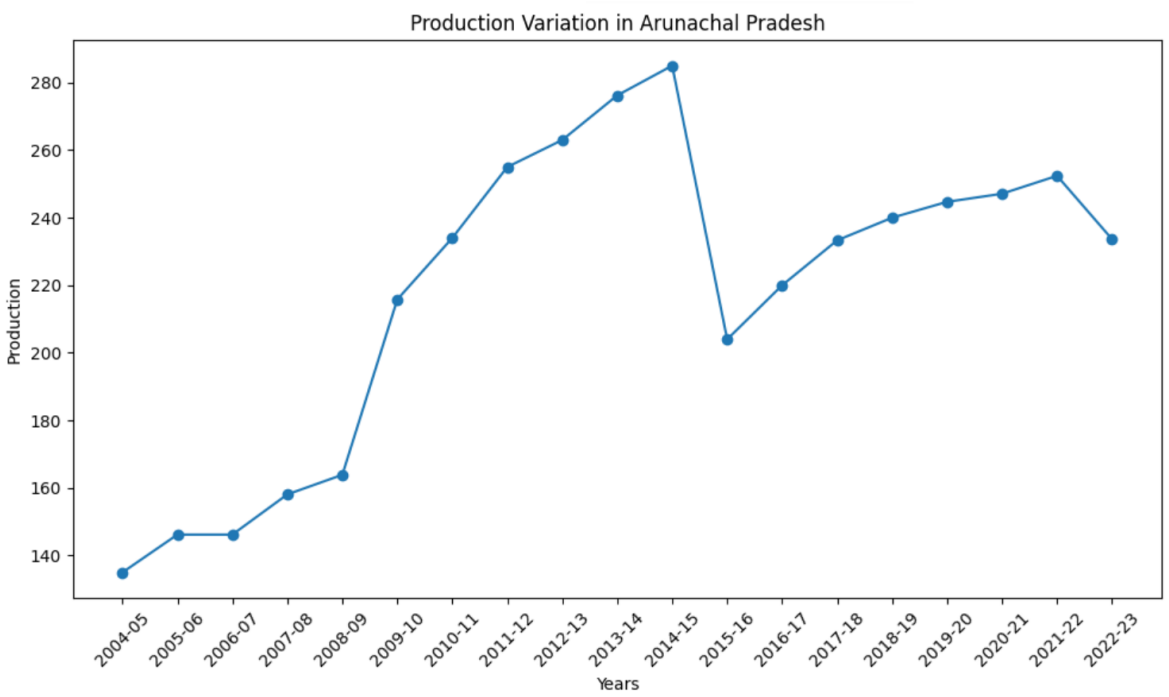
4. State-wise rate of production.

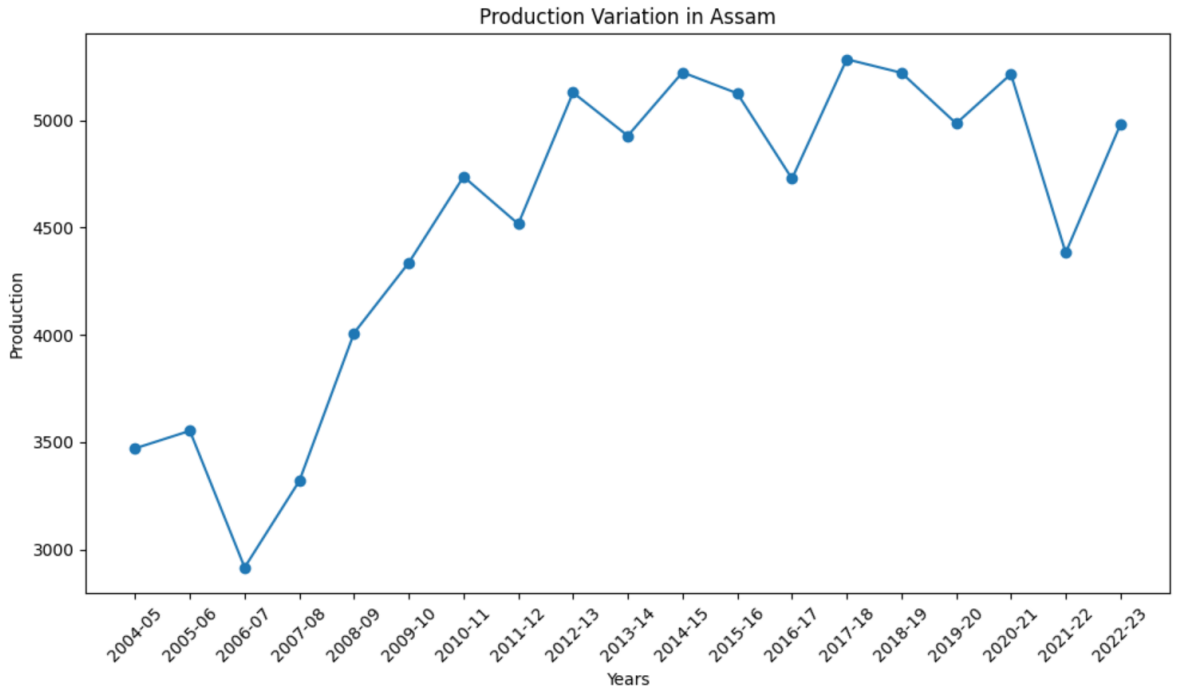
```
states = df_final.columns[:-1]
years = df_final.index

for state in states:
    plt.figure(figsize=(10, 6))
    plt.plot(years, df_final[state], marker='o', linestyle='-
    plt.title(f'Production Variation in {state}')
    plt.xlabel('Years')
    plt.ylabel('Production')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

Results:

Given below the visualizations for the first three states and for all the given states please refer the code file in .ipynb format.







Analysis:

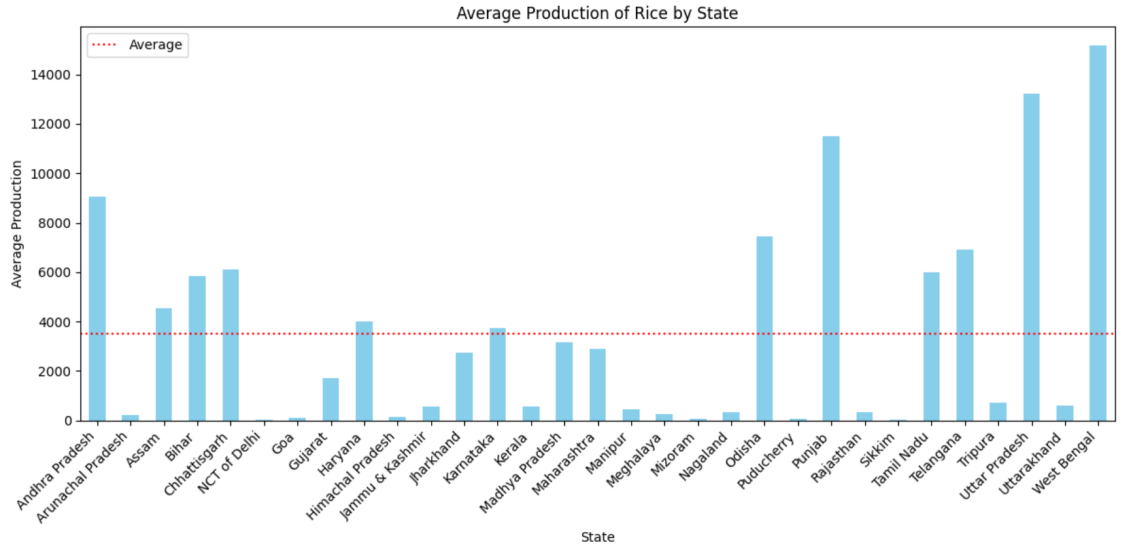
1. From the line graphs, we can observe the absolute production levels of rice across different states for the specified year. It provides a clear comparison of production levels, showcasing which states contribute more or less to the overall production.(West Bengal, Uttar Pradesh, Punjab, Andhra Pradesh contributes more and Sikkim, Delhi contributes less)

2. Based on these visualizations, we can draw conclusions about the relative importance of different states in rice production. It can help policymakers, agricultural experts, and stakeholders in identifying states that play significant roles in rice production and where potential improvements or interventions might be needed to enhance production levels.

5. For which provinces there is a need to improve their production of rice.

```
df_final_without_all_india = df_final.drop('ALL INDIA', axis=
average_production = df_final_without_all_india.mean(axis=0)
plt.figure(figsize=(12, 6))
average_production.plot(kind='bar', color='skyblue')
plt.axhline(y=average_production.mean(), color='red', linesty
plt.title('Average Production of Rice by State')
plt.xlabel('State')
plt.ylabel('Average Production')
plt.xticks(rotation=45, ha='right')
plt.legend()
plt.tight_layout()
plt.show()

threshold = 5000
low_production_states = average_production[average_production
print("Provinces needing improvement in rice production:")
print(low_production_states)
```

Output




```
Provinces needing improvement in rice production:
['Arunachal Pradesh', 'Assam', 'NCT of Delhi', 'Goa', 'Gujarat', 'Haryana', 'Himachal Pradesh', 'Jammu & Kashmir', 'Jharkhand', 'Karnataka', 'Kerala', 'Madhya Pradesh
```

Analysis

From the Bar plot of the average production of rice across all the states, the states with the average production less than the overall average production need to improve. Those states are mentioned in the output result.

Data Pre-Processing

Dealing with the missing data

The dataset contains missing data, which were filled with the median after thorough analysis, following common practice for time-series problems.

1. Finding the null values:

```
df.replace(".", np.nan, inplace=True)
df.isnull().sum()
```

```
2004-05     1
2005-06     1
2006-07     1
2007-08     1
2008-09     1
2009-10     1
2010-11     0
2011-12     0
2012-13     0
2013-14     0
2014-15     0
2015-16     0
2016-17     0
2017-18     0
2018-19     0
2019-20     0
2020-21     0
2021-22     0
2022-23    11
dtype: int64
```

2.Replacing the null values with the median of respective state's production

```
df16 = pd.DataFrame()
for col_name in df_f.columns:
    df16[col_name] =df_f[col_name].fillna(value=df_f[col_n
df_final=df16
df_final
```

	Andhra Pradesh	Arunachal Pradesh	Assam	Bihar	Chhattisgarh	NCT of Delhi	Goa	Gujarat	Haryana	Himachal Pradesh	...	Punjab	Rajasthan	Sikkim	Tamil Nadu	Telangana	Tripura	Uttar Pradesh
2004-05	9601.0	135.00	3470.7	2472.2	4383.3	14.30	145.2	1238.2	3023.00	122.00	...	10437.00	150.40	21.60	5062.2	6262.2	545.10	9555.6
2005-06	11704.0	146.20	3552.5	3495.5	5011.6	24.00	147.3	1298.0	3210.00	112.10	...	10193.00	153.00	21.50	5220.0	6262.2	552.90	11133.7
2006-07	11872.0	146.20	2916.0	4989.3	5041.4	31.10	130.3	1390.0	3371.00	123.50	...	10138.00	169.80	21.50	6610.6	6262.2	620.50	11124.0
2007-08	13324.0	158.10	3319.0	4418.1	5426.6	31.40	121.6	1474.0	3613.00	121.50	...	10489.00	259.60	22.90	5040.2	6262.2	624.60	11780.0
2008-09	14241.0	163.90	4008.5	5590.3	4391.8	31.40	123.3	1303.0	3298.00	118.30	...	11000.00	241.10	21.70	5182.7	6262.2	627.10	13097.0
2009-10	10538.0	215.80	4335.9	3599.3	4110.4	19.30	100.6	1292.0	3625.00	105.90	...	11236.00	228.30	24.30	5665.2	6262.2	640.00	10807.1
2010-11	7882.4	234.00	4736.6	3102.1	6159.0	19.60	115.0	1496.6	3472.00	128.90	...	10837.00	265.50	21.00	5792.4	6535.6	702.50	11992.0
2011-12	7746.2	255.00	4516.3	7162.6	6028.4	19.80	121.8	1790.0	3759.00	131.60	...	10542.00	253.40	20.90	7458.7	5148.8	718.30	14022.0
2012-13	6862.4	263.00	5128.5	7529.3	6608.8	19.70	122.8	1541.0	3976.00	125.30	...	11374.00	222.50	21.30	4049.9	4647.6	713.20	14416.0
2013-14	6969.7	276.20	4927.1	5505.8	6716.4	29.60	126.5	1636.0	3998.00	120.80	...	11267.00	312.60	20.30	5349.8	5755.0	711.80	14636.0
2014-15	7233.9	285.00	5222.7	6356.7	6322.1	25.90	120.5	1830.9	4006.00	125.20	...	11107.00	366.70	20.10	5727.8	4440.8	747.00	12167.9
2015-16	7488.7	204.00	5125.1	6802.2	5789.4	17.30	115.1	1702.0	4145.00	129.90	...	11823.00	369.80	13.10	7517.1	3047.0	794.80	12501.0
2016-17	7452.4	220.00	4727.4	8239.3	8048.4	17.30	113.2	1930.0	4453.00	146.60	...	11586.20	452.70	19.70	2369.4	5173.4	814.60	13754.0

Model Training, Evaluation and Prediction

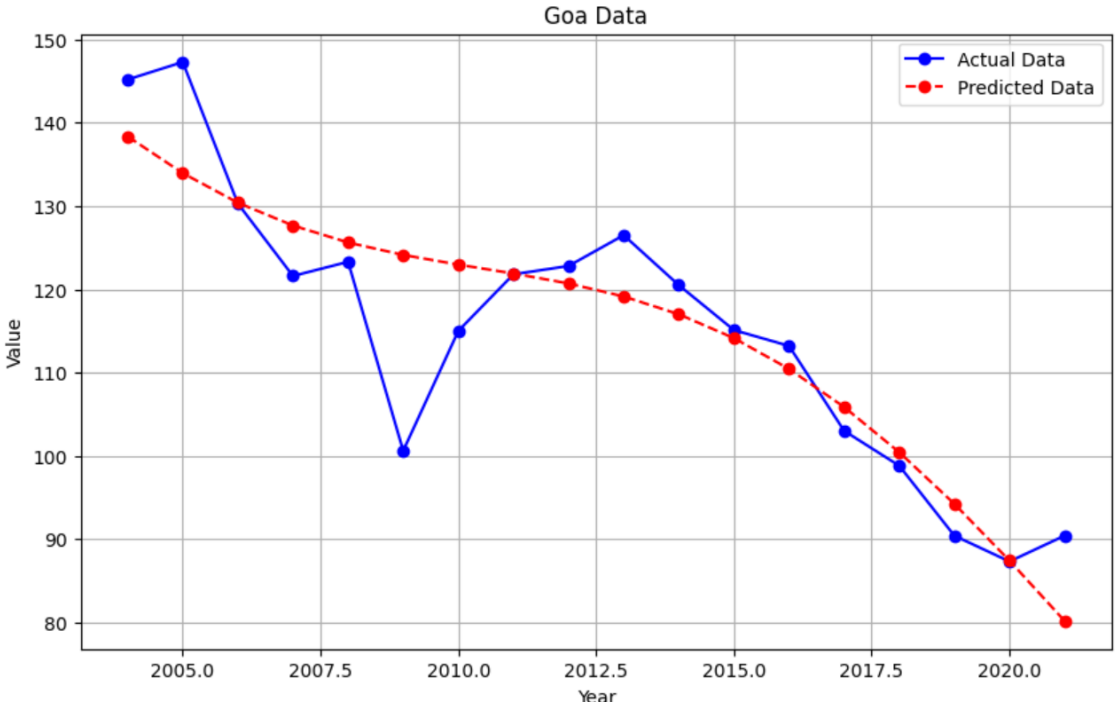
Model training

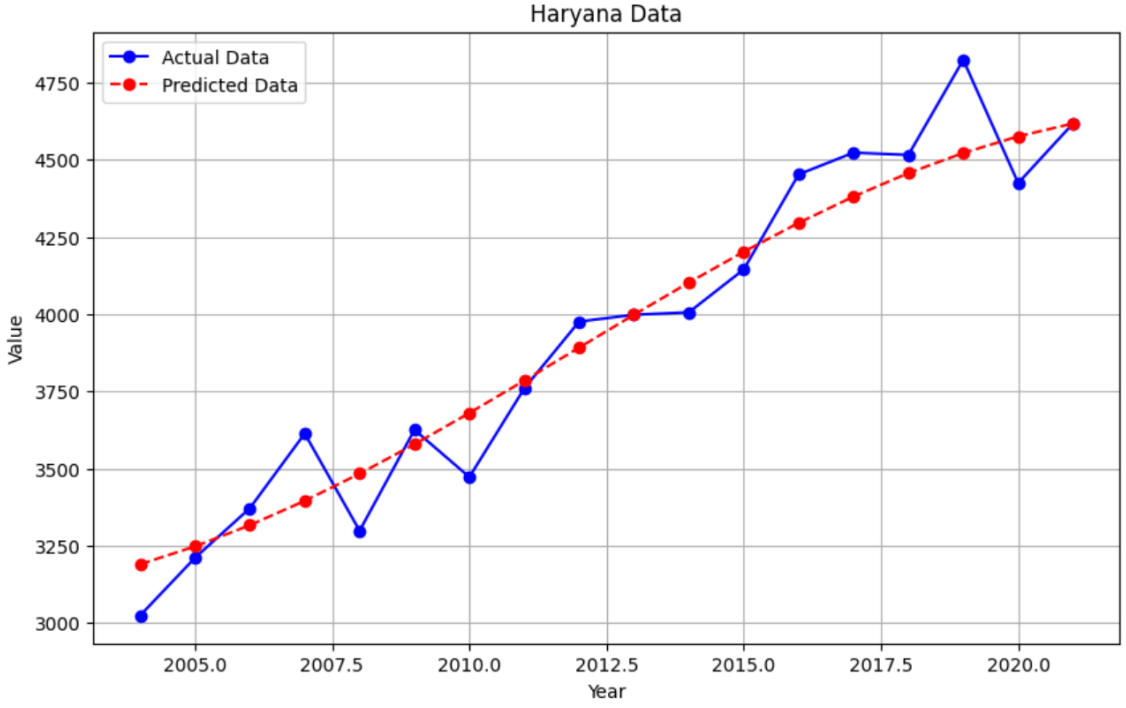
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
import numpy as np


# Define function to train SVR model and make predictions
def train_and_predict(data):
    # Initialize empty dictionary to store predicted value
    predictions = {}

    # Get unique states
    states = data['State/Union Territory'].unique()

    # Iterate over each state
    for state in states:
        # Filter data for the current state
        state_data = data[data['State/Union Territory'] ==

        # Extract years and corresponding values
        years = state_data.columns[1:].str.split('-').str[
        values = state_data.values[0][1:]

        # Handle missing values and non-numeric characters
        values = pd.to_numeric(values, errors='coerce')

        # Remove NaN values
        years = years[~np.isnan(values)]
        values = values[~np.isnan(values)]

        # Prepare training data
```

```
        X_train = years.values.reshape(-1, 1)
        y_train = values.astype(float)

        # Scale features
        scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train)

        # Train SVR model
        svr = SVR(kernel='rbf', C=1e3, gamma=0.1)
        svr.fit(X_train_scaled, y_train)

        # Predict values for existing years
        predictions[state] = (years, values, svr.predict(X

    return predictions

# Call the function to train models and make predictions
predicted_values = train_and_predict(data)

# Plot graphs for each state
for state, (years, actual_values, predicted_values) in pre
    plt.figure(figsize=(10, 6))
    plt.plot(years, actual_values, marker='o', linestyle='
    plt.plot(years, predicted_values, marker='o', linestyl
    plt.title(f"{state} Data")
    plt.xlabel('Year')
    plt.ylabel('Value')
    plt.legend()
    plt.grid(True)
    plt.show()
```

Output

Provided two of all the results, to view for all the states please refer the code file





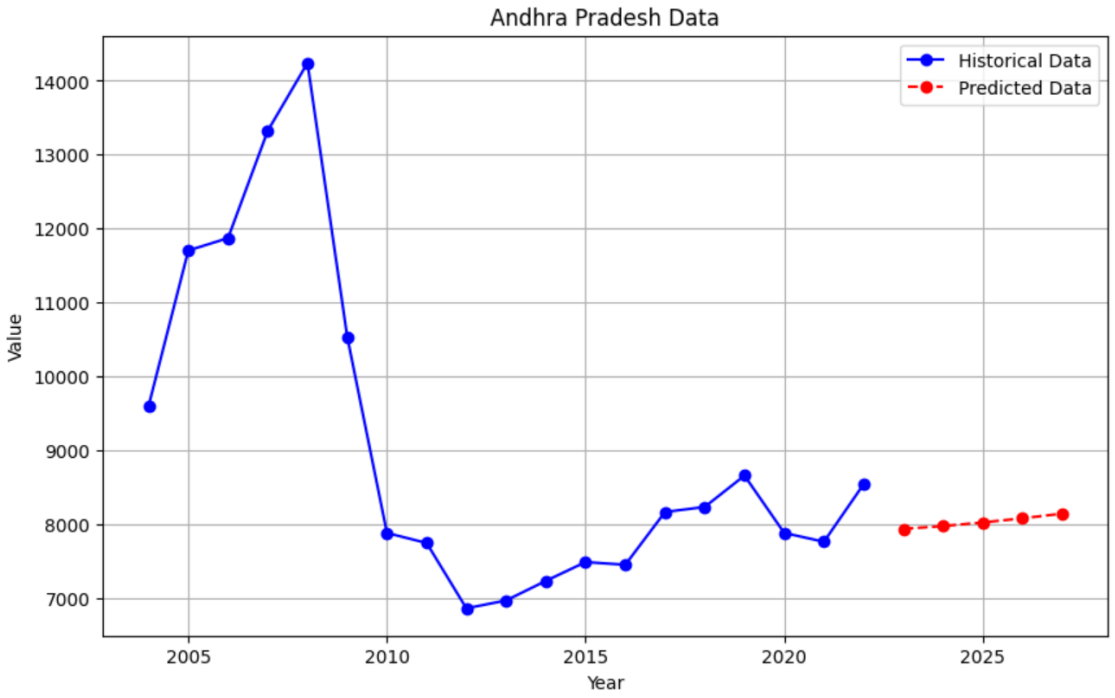
Predicting the state-wise/union territory-wise production of rice for the next 5 years using Support Vector Regression model

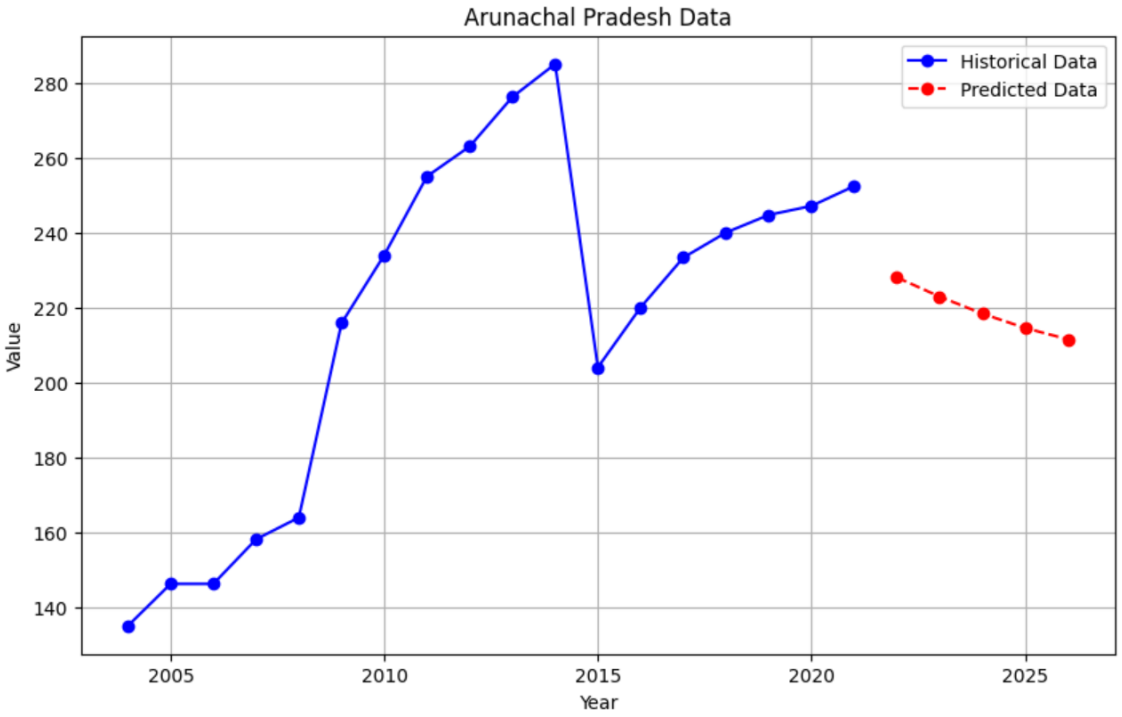
```
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
```

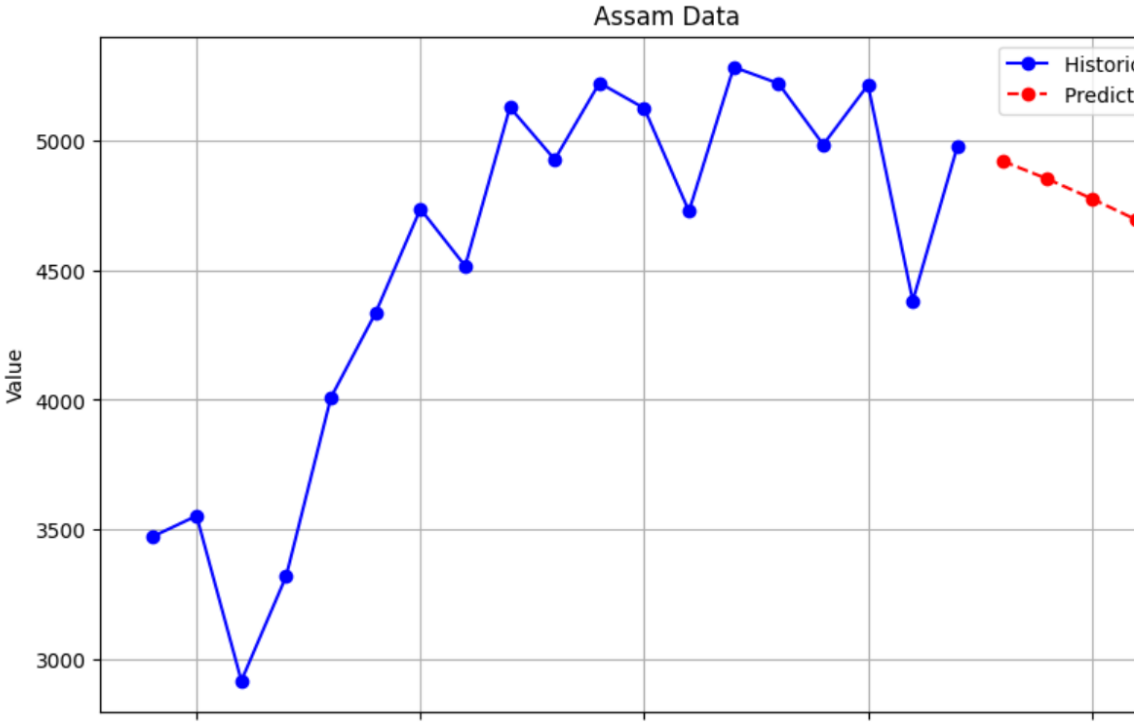
```
import matplotlib.pyplot as plt
def train_and_predict(df_final):
    predictions = {}

    states = df_final['State/Union Territory'].unique()

    for state in states:
        state_data = df_final[df_final['State/Union Territ

        years = state_data.columns[1:].str.split('-').str[
        values = state_data.values[0][1:]

        values = pd.to_numeric(values, errors='coerce')

        years = years[~np.isnan(values)]
        values = values[~np.isnan(values)]

        X_train = years.values.reshape(-1, 1)
        y_train = values.astype(float)

        scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train)

        svr = SVR(kernel='rbf', C=1e3, gamma=0.1)
        svr.fit(X_train_scaled, y_train)

        future_years = np.arange(years.max() + 1, years.ma
        future_years_scaled = scaler.transform(future_year
        future_predictions = svr.predict(future_years_scal

        predictions[state] = (years, values, future_years,
    return predictions

predicted_values = train_and_predict(data)


for state, (years, values, future_years, future_prediction
    plt.figure(figsize=(10, 6))
```

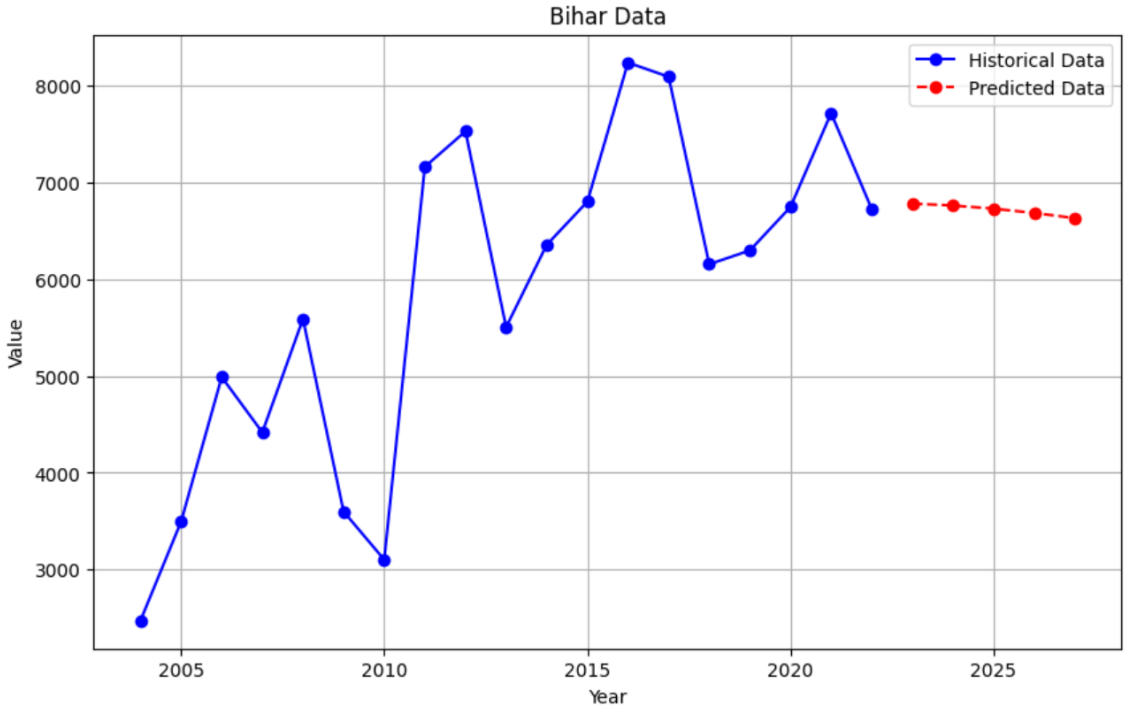
```
    plt.plot(years, values, marker='o', linestyle='-', col
    plt.plot(future_years, future_predictions, marker='o',
    plt.title(f"{state} Data")
    plt.xlabel('Year')
    plt.ylabel('Value')
    plt.legend()
    plt.grid(True)
    plt.show()
```

Output:




Arunachal Pradesh Data
Historical Data
Predicted Data
Value
Year
280
260
240
220
200
180
160
140
2005
2010
2015
2020
2025


Assam Data
Historic
Predict
Value
5000
4500
4000
3500
3000



Results:

	State	Year	Predicted Value
0	Andhra Pradesh	2023	7937.045809
1	Andhra Pradesh	2024	7975.946976
2	Andhra Pradesh	2025	8024.631679
3	Andhra Pradesh	2026	8081.172448
4	Andhra Pradesh	2027	8143.628911

Similarly it would print all the predicted values in tabular format of all the state's production for the next 5 years. Please refer the code file for more results.

Link to the source code file:

https://colab.research.google.com/drive/1ZnnlKE5pROef0oenGRv2Be-8CR3IwjBl?usp=sharing

References :

1.https://www.mdpi.com/2624-7402/3/2/12

2.https://www.thehindu.com/sci-tech/agriculture/IGC-2009-10-Rice-output-may-fall-14-per-cent-in-India/article16891650.ece