```
import gdown
file_id = '1rmJoBX-Hrm6MtPtjnVQ4lLUXqytyARPI'
url = f'https://drive.google.com/uc?id={file_id}'
output = 'Major_5_District.csv'
gdown.download(url, output, quiet=False)
import pandas as pd
df = pd.read_csv('Major_5_District.csv')
```

```
Downloading...
From: https://drive.google.com/uc?id=1rmJoBX-Hrm6MtPtjnVQ4lLUXqytyARPI
To: /content/Major_5_District.csv
100%|██████████| 57.8k/57.8k [00:00<00:00, 53.0MB/s]
```

```
df = df.drop(columns=['TOTAL AREA (1000 ha)'])
```

```
df.columns
```

```
Index(['Year', 'Dist Name', 'Area', 'Production', 'Yield', 'Irrigated Area',
       'Annual Rainfall', 'NITROGEN CONSUMPTION (tons)',
       'NITROGEN SHARE IN NPK (Percent)', 'NITROGEN PER HA OF NCA (Kg per ha)',
       'NITROGEN PER HA OF GCA (Kg per ha)', 'PHOSPHATE CONSUMPTION (tons)',
       'PHOSPHATE SHARE IN NPK (Percent)',
       'PHOSPHATE PER HA OF NCA (Kg per ha)',
       'PHOSPHATE PER HA OF GCA (Kg per ha)', 'POTASH CONSUMPTION (tons)',
       'POTASH SHARE IN NPK (Percent)', 'POTASH PER HA OF NCA (Kg per ha)',
       'POTASH PER HA OF GCA (Kg per ha)', 'TOTAL CONSUMPTION (tons)',
       'TOTAL PER HA OF NCA (Kg per ha)', 'FOREST AREA (1000 ha)',
       'BARREN AND UNCULTIVABLE LAND AREA (1000 ha)',
       'LAND PUT TO NONAGRICULTURAL USE AREA (1000 ha)',
       'CULTIVABLE WASTE LAND AREA (1000 ha)',
       'PERMANENT PASTURES AREA (1000 ha)', 'OTHER FALLOW AREA (1000 ha)',
       'CURRENT FALLOW AREA (1000 ha)', 'NET CROPPED AREA (1000 ha)',
       'GROSS CROPPED AREA (1000 ha)', 'CROPING INTENSITY (Percent)',
       'Min Temp (Centigrate)', 'Max Temp (Centigrate)', 'Precipitation (mm)',
       'Evapotranspiration (mm)'],
      dtype='object')
```

```
import pandas as pd
import numpy as np
from sklearn.svm import SVR
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import matplotlib.pyplot as plt

# Assuming 'df' is the dataframe containing the dataset
# List of districts
districts = df['Dist Name'].unique()

# Initialize results storage
district_results = {}

# Define the evaluation metrics
def evaluate_model(y_true, y_pred):
    r2 = r2_score(y_true, y_pred)
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_true, y_pred)
    mape = np.mean(np.abs((y_true - y_pred) / y_true)) * 100
    return r2, mse, rmse, mae, mape

# Loop through each district
for district in districts:
```

```
district_data = df[df['Dist Name'] == district]

# Prepare the feature columns and target column
features = [col for col in district_data.columns if col not in ['Year', 'Dist Name']]
target_columns = features  # We will forecast all columns

# Initialize a list to store results for this district
district_metric_results = []

# Split the data for training and testing
X = district_data[features].values
y = district_data[target_columns].values

# Scale the features and target values
scaler_X = StandardScaler()
scaler_y = StandardScaler()

X_scaled = scaler_X.fit_transform(X)

# Split into training and testing
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=

# Apply SVR with hyperparameter tuning for each feature
svr = SVR()

# Define parameter grid for tuning
param_grid = {
    'C': [1, 10, 100],
    'epsilon': [0.1, 0.2, 0.5],
    'kernel': ['linear', 'rbf']
}

# Hyperparameter tuning using GridSearchCV
grid_search = GridSearchCV(svr, param_grid, cv=5, scoring='neg_mean_squared_error')

# Evaluate for each target column (feature)
for target_col in target_columns:
    # Reshape the target column to be 1D (required by SVR)
    y_target = y[:, features.index(target_col)].reshape(-1, 1).ravel()  # Flatten the target

    # Split the data again for training and testing for this feature
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_target, test_size=0.2, ra

    # Perform hyperparameter tuning on the current feature
    grid_search.fit(X_train, y_train)

    best_model = grid_search.best_estimator_

    # Fit the best model to the current feature
    best_model.fit(X_train, y_train)

    # Predict on the test set
    y_pred = best_model.predict(X_test)

    # Calculate evaluation metrics
    r2, mse, rmse, mae, mape = evaluate_model(y_test, y_pred)

    # Store the results for this district and feature
```

```
        district_metric_results.append((target_col, r2, mse, rmse, mae, mape))

    # Store average results for this district
    district_results[district] = {
        'average_r2': np.mean([result[1] for result in district_metric_results]),
        'average_mse': np.mean([result[2] for result in district_metric_results]),
        'average_rmse': np.mean([result[3] for result in district_metric_results]),
        'average_mae': np.mean([result[4] for result in district_metric_results]),
        'average_mape': np.mean([result[5] for result in district_metric_results])
    }

# Convert results to a DataFrame for easier viewing
results_df = pd.DataFrame(district_results).T
results_df = results_df.sort_values(by='average_r2', ascending=False)

# Print the results
print(results_df)
```

```
           average_r2   average_mse  average_rmse  average_mae  average_mape
Akola        0.989386  1.098333e+06    361.084036   269.952245      2.582050
Wardha       0.989072  3.556850e+05    217.880787   159.691850      2.198246
Kolhapur     0.987192  1.984799e+06    445.143641   347.889885      1.667296
Ratnagiri    0.986486  1.610079e+05     95.758992    74.385694      1.713528
Pune         0.985461  1.946210e+07   1205.923178   996.726564      5.879397
```

```
results_df.to_csv('results.csv')
```

```
import pandas as pd
import numpy as np
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# List of future years for forecasting
future_years = [2018, 2019, 2020, 2021, 2022, 2023, 2025, 2030, 2035, 2040, 2045, 2050]

# Define evaluation metrics
def evaluate_model(y_true, y_pred):
    r2 = r2_score(y_true, y_pred)
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_true, y_pred)
    mape = np.mean(np.abs((y_true - y_pred) / y_true)) * 100
    return r2, mse, rmse, mae, mape

# Initialize results storage
district_results = {}

# Loop through each district
districts = df['Dist Name'].unique()

for district in districts:
    district_data = df[df['Dist Name'] == district].copy()

    # Prepare the feature columns
    features = [col for col in district_data.columns if col not in ['Year', 'Dist Name']]
```

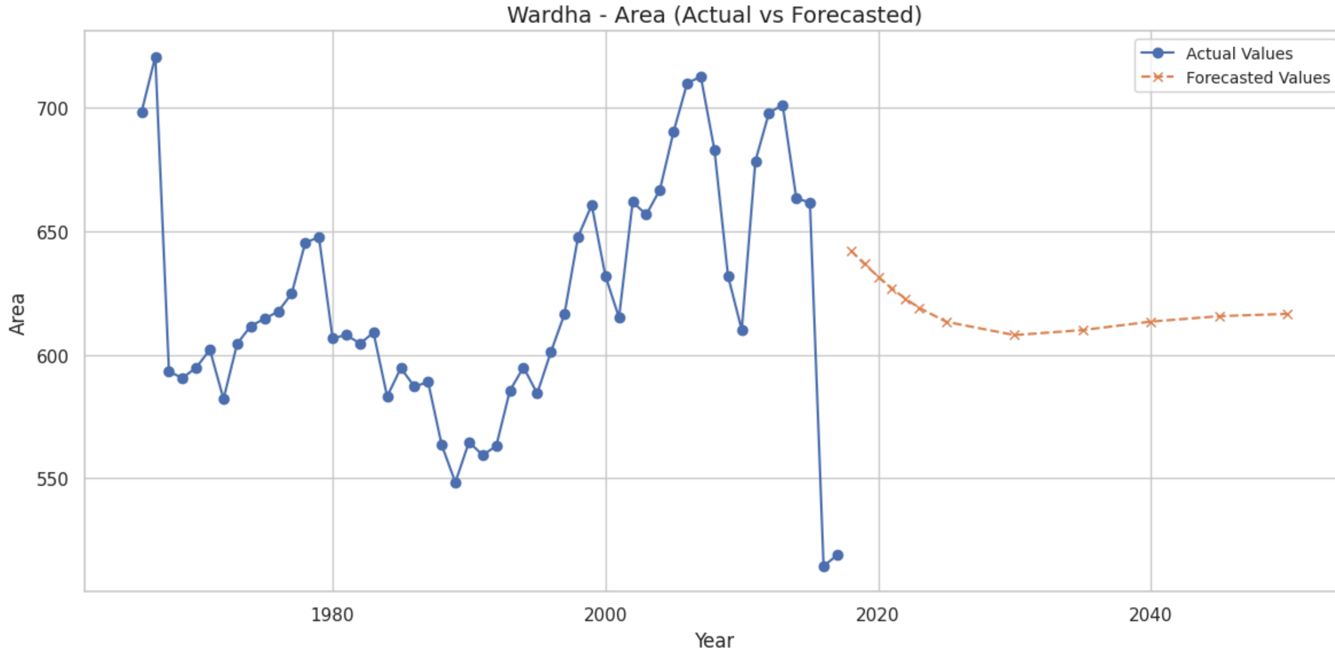
```
    # Store results for the district
    district_forecast = {}

    # Scale the features
    scaler = StandardScaler()
    scaled_data = scaler.fit_transform(district_data[features])

    # Replace original feature data with scaled values
    district_data[features] = scaled_data

    for feature in features:
        # Historical data for the feature
        X = district_data['Year'].values.reshape(-1, 1)
        y = district_data[feature].values

        # Train-test split (up to 2017 for training)
        X_train = X[X.flatten() <= 2017]
        y_train = y[X.flatten() <= 2017]

        # Define SVR and hyperparameter grid
        svr = SVR()
        param_grid = {
            'C': [1, 10, 100],
            'epsilon': [0.05, 0.1, 0.2],
            'kernel': ['rbf', 'linear'],
            'gamma': ['scale', 'auto']
        }

        # GridSearchCV for hyperparameter tuning
        grid_search = GridSearchCV(svr, param_grid, cv=5, scoring='neg_mean_squared_error', n_job
        grid_search.fit(X_train, y_train)

        # Best model for the feature
        best_model = grid_search.best_estimator_

        # Forecast future values for the feature
        X_future = np.array(future_years).reshape(-1, 1)
        forecast_scaled = best_model.predict(X_future)

        # Inverse scale the forecasted values
        forecast_original = scaler.inverse_transform(
            np.column_stack([np.zeros(len(forecast_scaled)) for _ in range(len(features))])
        )
        forecast_original[:, features.index(feature)] = forecast_scaled
        forecast_original = scaler.inverse_transform(forecast_original)[:, features.index(feature

        # Store forecasts for this feature
        district_forecast[feature] = forecast_original

    # Save results for the district
    district_results[district] = pd.DataFrame({
        'Year': future_years,
        **district_forecast
    })

# Combine results from all districts
final_forecast_df = pd.concat(district_results, names=['District', 'Index']).reset_index(level=0)
```

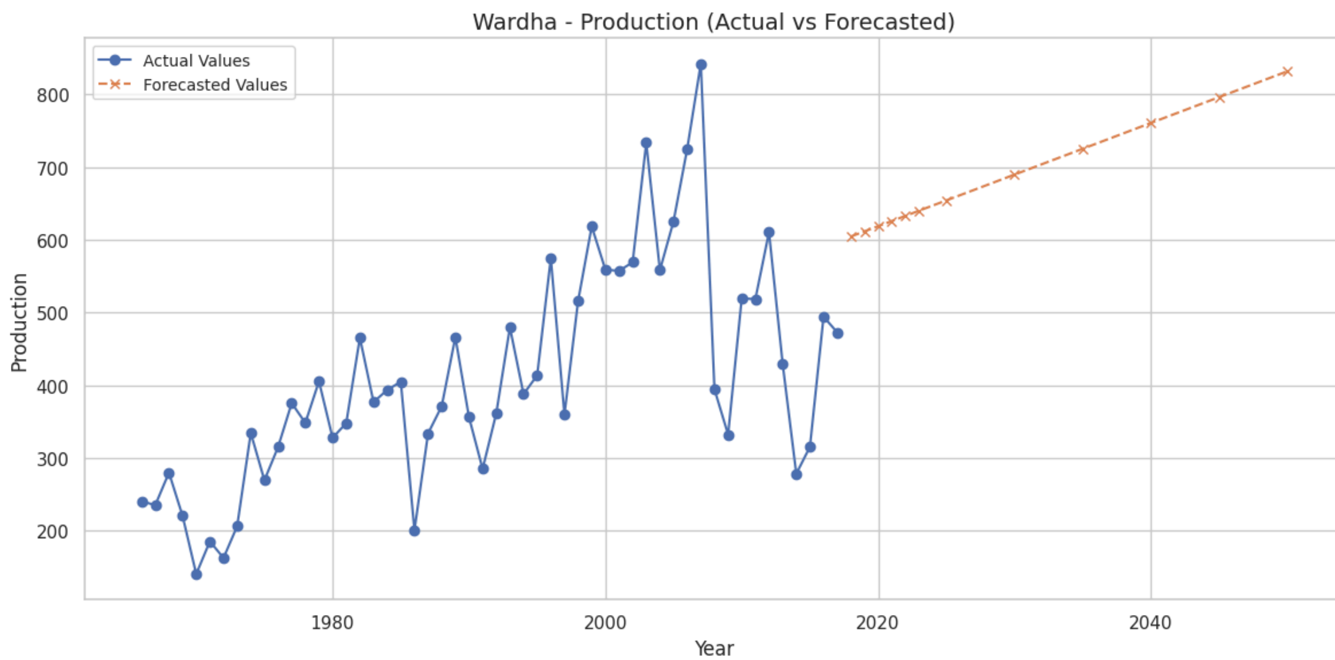
```
# Save the forecast to a CSV
final_forecast_df.to_csv('district_forecasts_svr.csv', index=False)

# Print sample output
print(final_forecast_df.head())
```

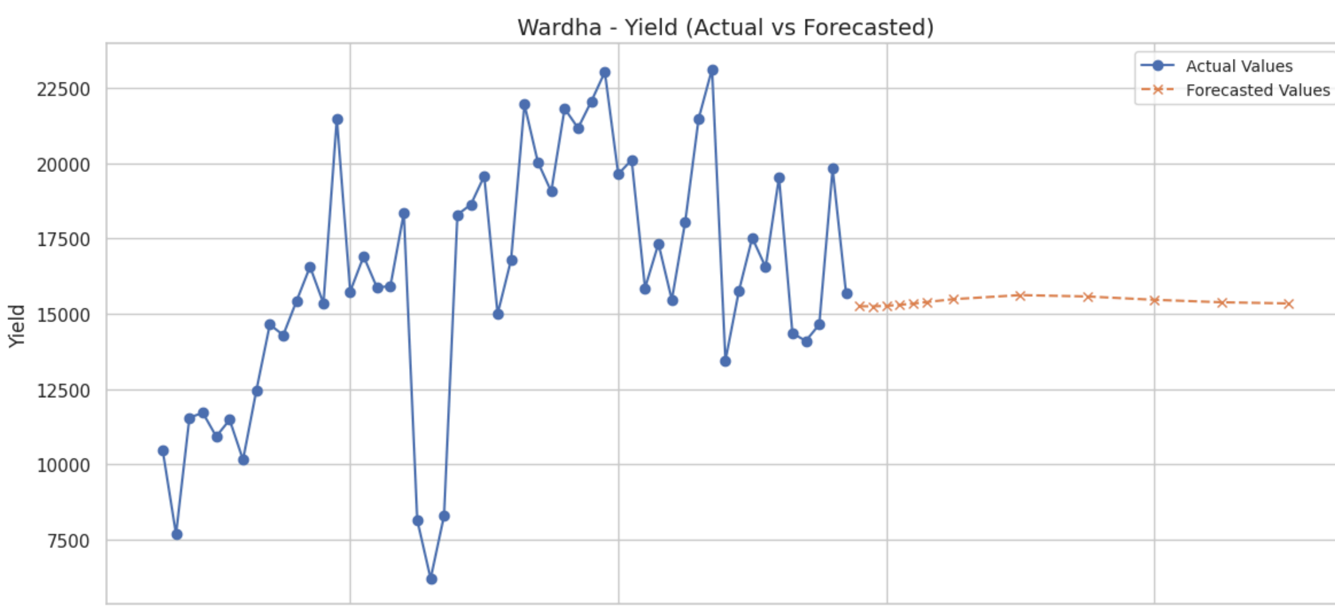
```
      District  Year        Area  Production         Yield  Irrigated Area  \
Index
0       Wardha  2018  642.084236  604.487833  15251.821736      -21.591030
1       Wardha  2019  636.647525  611.587838  15247.302714      -19.196632
2       Wardha  2020  631.530931  618.687834  15264.158317      -16.217645
3       Wardha  2021  626.832141  625.787840  15296.742722      -12.815237
4       Wardha  2022  622.623528  632.887836  15339.678840       -9.152007

       Annual Rainfall  NITROGEN CONSUMPTION (tons)  \
Index
0          1009.977443                 29602.009039
1          1009.976909                 25929.432277
2          1009.976908                 21603.447209
3          1009.976908                 16807.193184
4          1009.976908                 11745.035204

       NITROGEN SHARE IN NPK (Percent)  NITROGEN PER HA OF NCA (Kg per ha)  \
Index
0                            49.064245                           87.383565
1                            48.720809                           78.298618
2                            48.377370                           67.351426
3                            48.033939                           55.014328
4                            47.690498                           41.824164

       ...  PERMANENT PASTURES AREA (1000 ha)  OTHER FALLOW AREA (1000 ha)  \
Index  ...
0      ...                          59.531708                    19.897169
1      ...                          65.669116                    19.789080
2      ...                          72.087364                    19.669494
3      ...                          78.580821                    19.537032
4      ...                          84.933987                    19.391482

       CURRENT FALLOW AREA (1000 ha)  NET CROPPED AREA (1000 ha)  \
Index
0                          51.792338                  346.223316
1                          48.472881                  342.759318
2                          44.894186                  339.474823
3                          41.152400                  336.512369
4                          37.352606                  333.999979

       GROSS CROPPED AREA (1000 ha)  CROPING INTENSITY (Percent)  \
Index
0                        427.208127                   126.832306
1                        410.212889                   124.292494
2                        391.556681                   121.249246
3                        372.059491                   117.856460
4                        352.570815                   114.280383

       Min Temp (Centigrate)  Max Temp (Centigrate)  Precipitation (mm)  \
Index
0                  22.154487              33.547943         1155.055233
1                  22.301808              33.554813         1158.246226
2                  22.442838              33.561683         1161.437213
3                  22.572476              33.568553         1164.628206
4                  22.686213              33.575422         1167.819205

       Evapotranspiration (mm)
Index
```

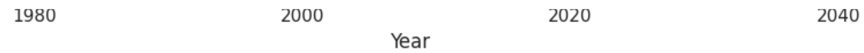
```
import matplotlib.pyplot as plt
import seaborn as sns

# Set seaborn style for plots
sns.set(style="whitegrid")

# Ensure 'target_columns' is defined (for example, it could be the list of feature names like ['f
target_columns = [col for col in df.columns if col not in ['Year', 'Dist Name']]

# Iterate over districts and target columns to visualize actual and forecasted values
```

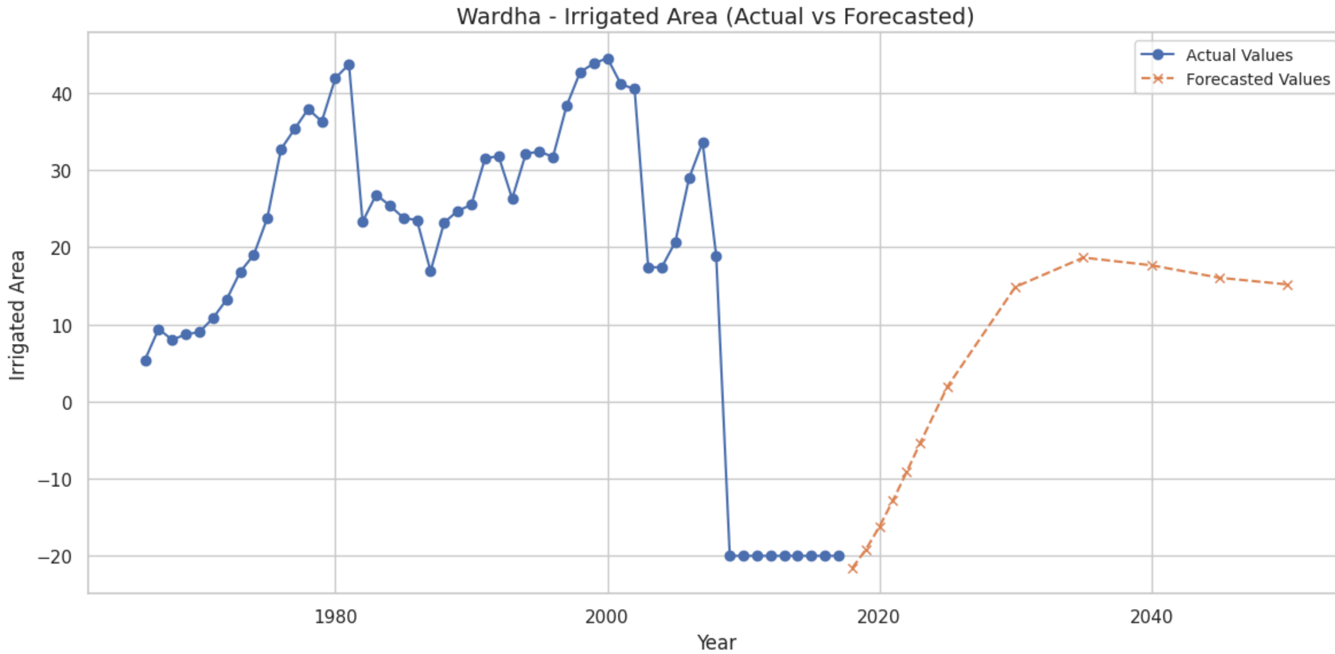
```
for district in districts:
    # Get the actual data for the district
    district_data = df[df['Dist Name'] == district]

    # Get the forecasted data for the district from the final forecast dataframe
    forecasted_data = final_forecast_df[final_forecast_df['District'] == district]

    for target_column in target_columns:
        # Prepare the actual data
        if target_column not in district_data.columns:
            continue
        actual_data = district_data[['Year', target_column]].dropna()

        # Prepare the forecasted data for the target column
        # Forecasted data is stored under the column named after the target column
        forecasted_target_data = forecasted_data[['Year', target_column]].dropna()

        # Plot the data
        plt.figure(figsize=(12, 6))
        plt.plot(
            actual_data['Year'], actual_data[target_column], label="Actual Values", marker='o'
        )
        plt.plot(
            forecasted_target_data['Year'], forecasted_target_data[target_column],
            label="Forecasted Values", linestyle='--', marker='x'
        )

        # Add title, labels, and legend
        plt.title(f"{district} - {target_column} (Actual vs Forecasted)", fontsize=14)
        plt.xlabel("Year", fontsize=12)
        plt.ylabel(target_column, fontsize=12)
        plt.legend(loc="best", fontsize=10)
        plt.grid(True)
        plt.tight_layout()
        plt.show()
```

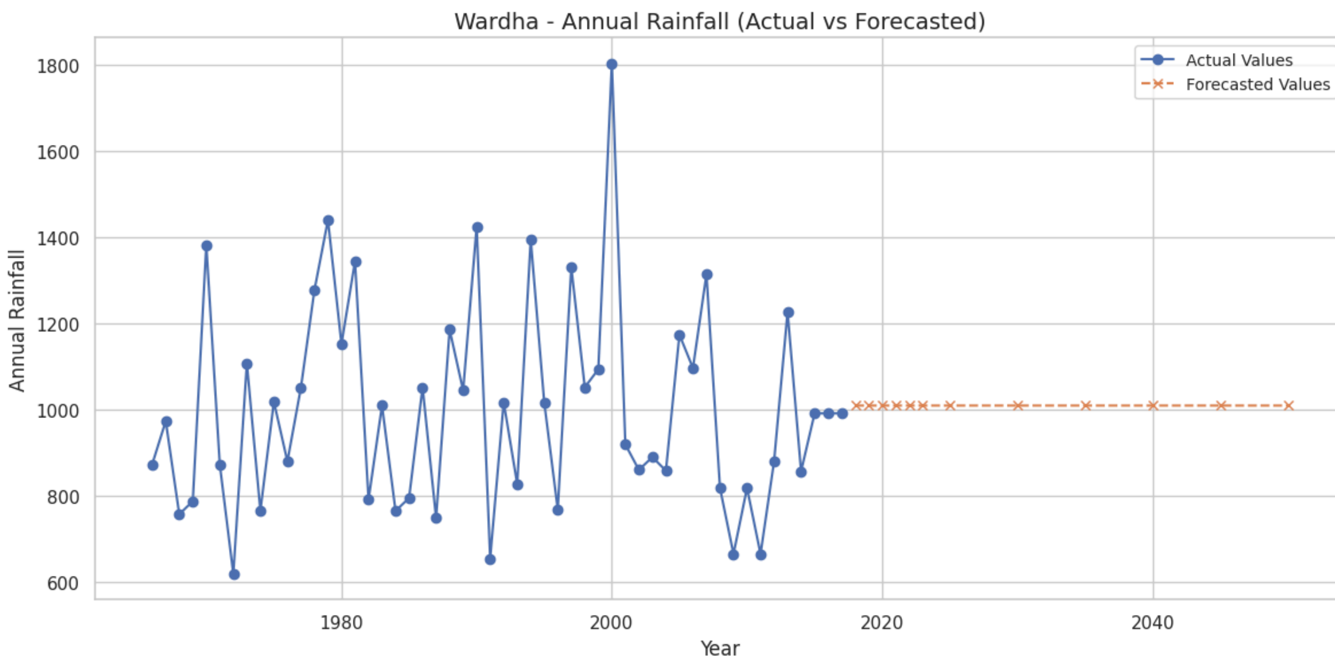

Wardha - Area (Actual vs Forecasted)
Actual Values
Forecasted Values
Area
700
650
600
550
1980
2000
2020
2040
Year

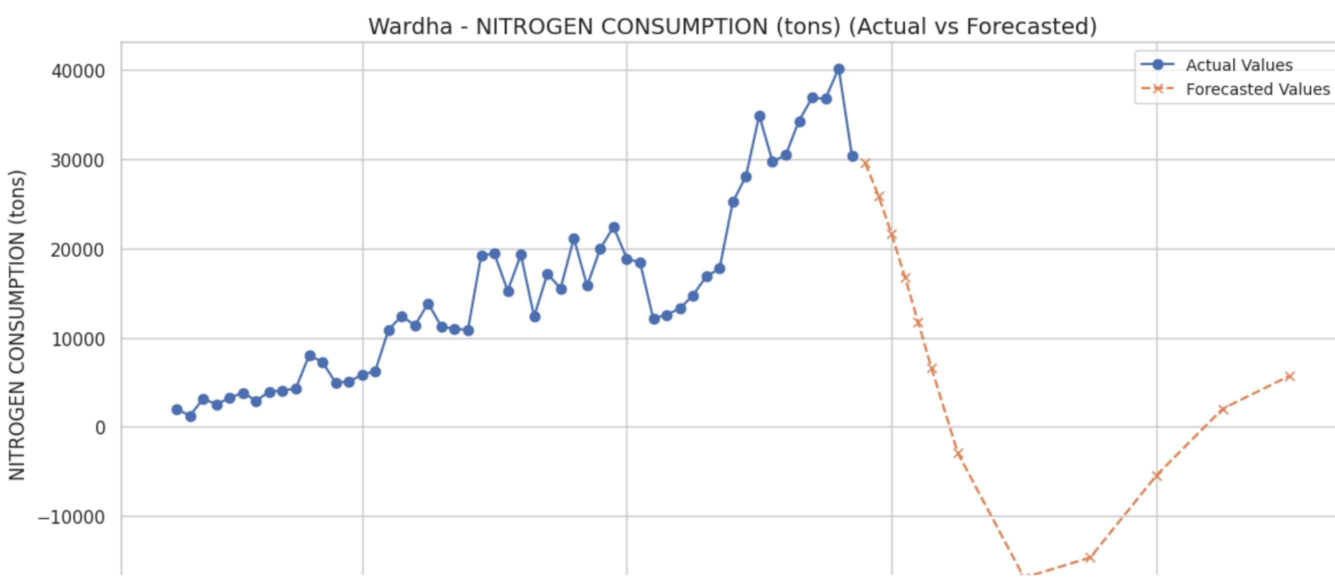

Wardha - Production (Actual vs Forecasted)
Actual Values
Forecasted Values
Production
800
700
600
500
400
300
200
1980
2000
2020
2040
Year

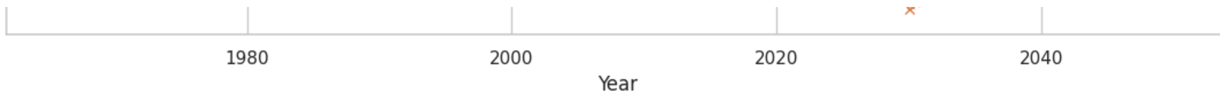

Wardha - Yield (Actual vs Forecasted)
Actual Values
Forecasted Values
Yield
22500
20000
17500
15000
12500
10000
7500

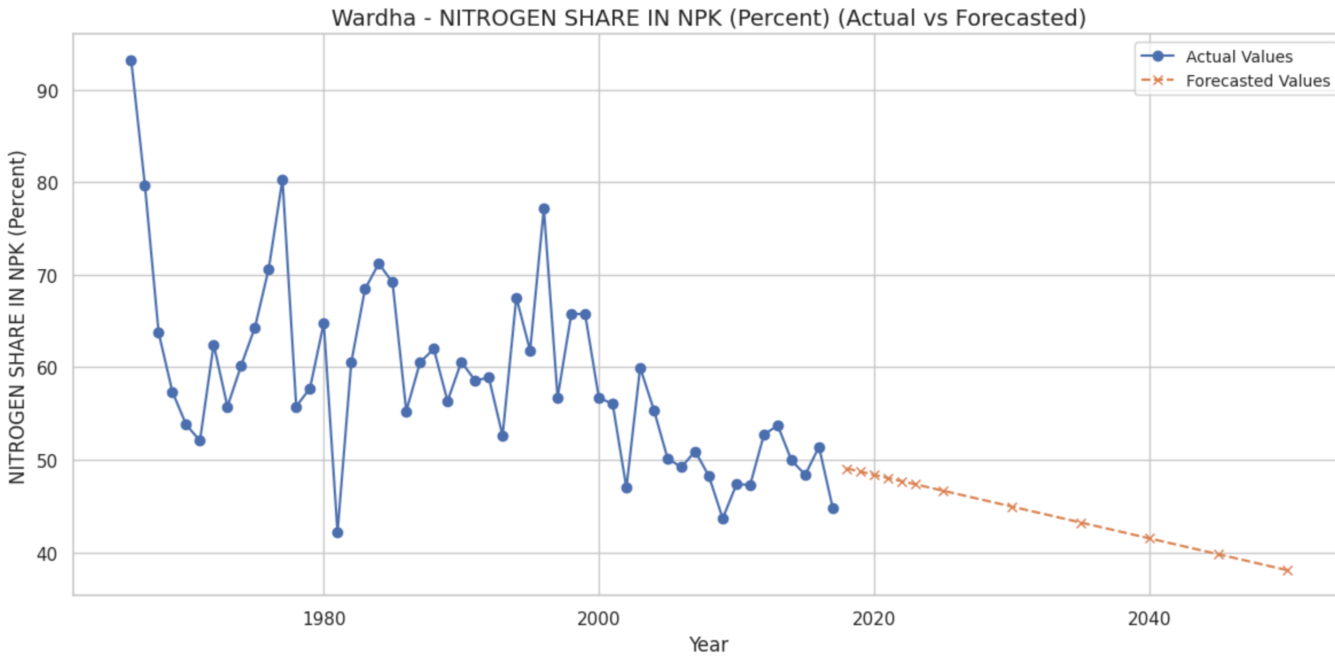

1980
2000
2020
2040
Year

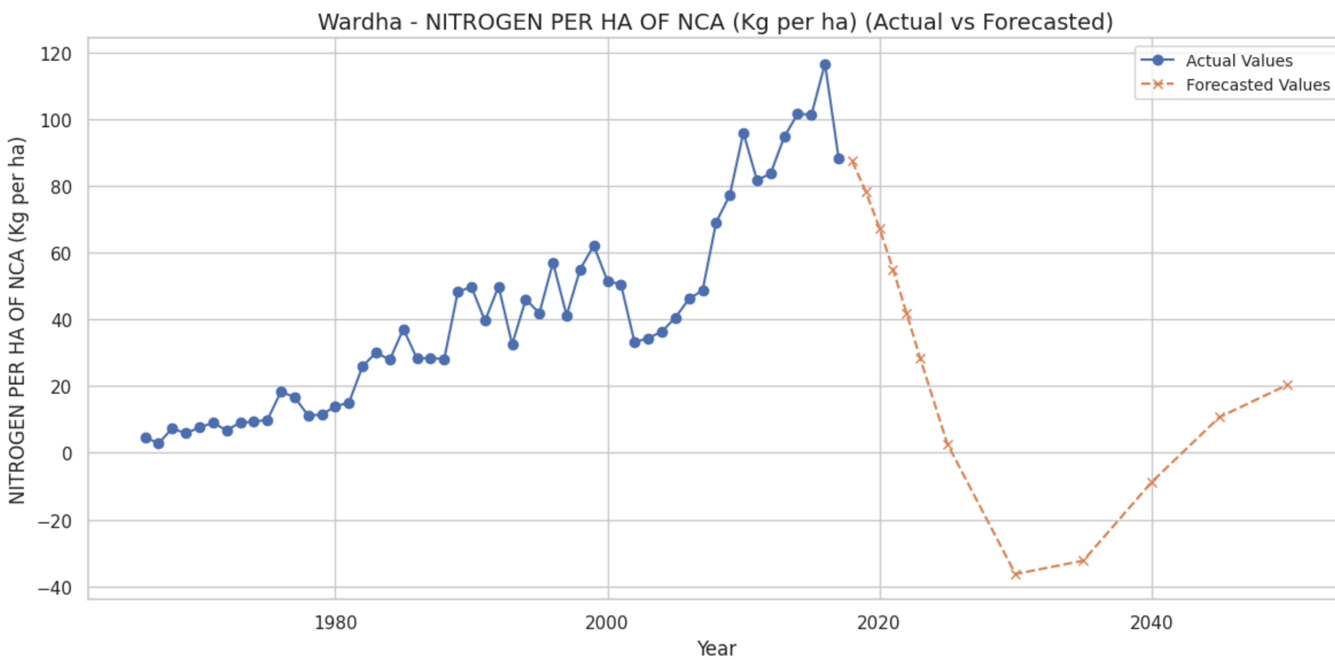

Wardha - Irrigated Area (Actual vs Forecasted)
Actual Values
Forecasted Values
Irrigated Area
40
30
20
10
0
−10
−20
1980
2000
2020
2040
Year

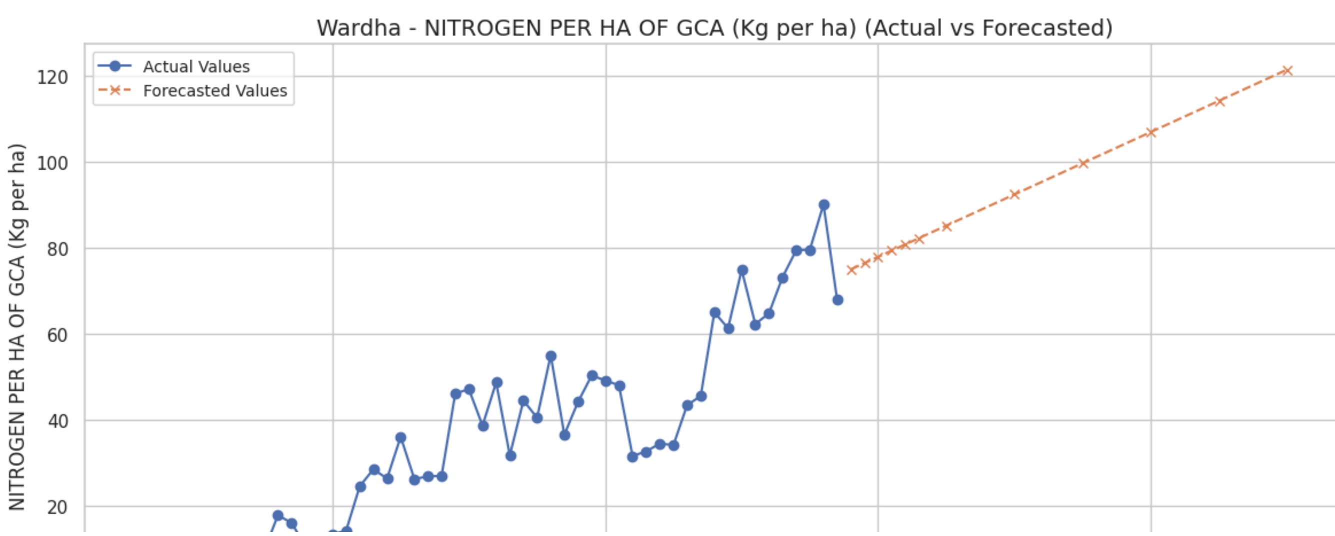

Wardha - Annual Rainfall (Actual vs Forecasted)
Actual Values
Forecasted Values
Annual Rainfall
1800
1600
1400
1200
1000
800
600
1980
2000
2020
2040
Year


Wardha - NITROGEN CONSUMPTION (tons) (Actual vs Forecasted)
Actual Values
Forecasted Values
NITROGEN CONSUMPTION (tons)
40000
30000
20000
10000
0
−10000

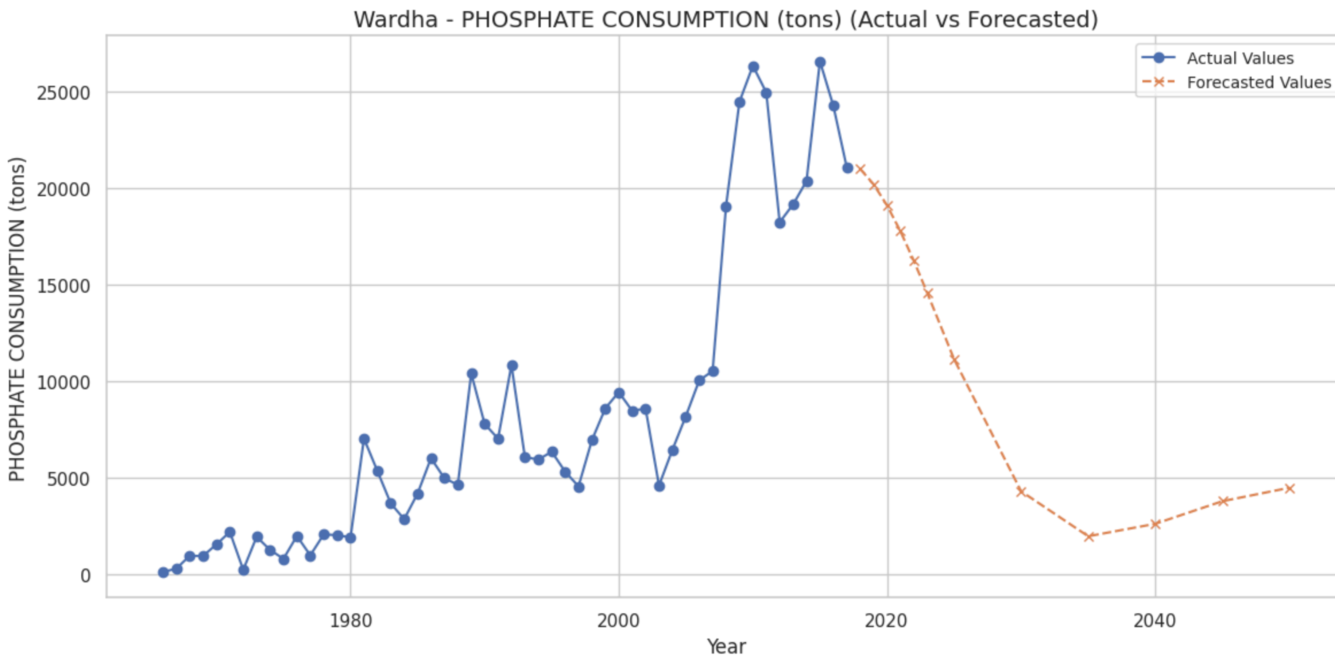

1980
2000
2020
2040
Year

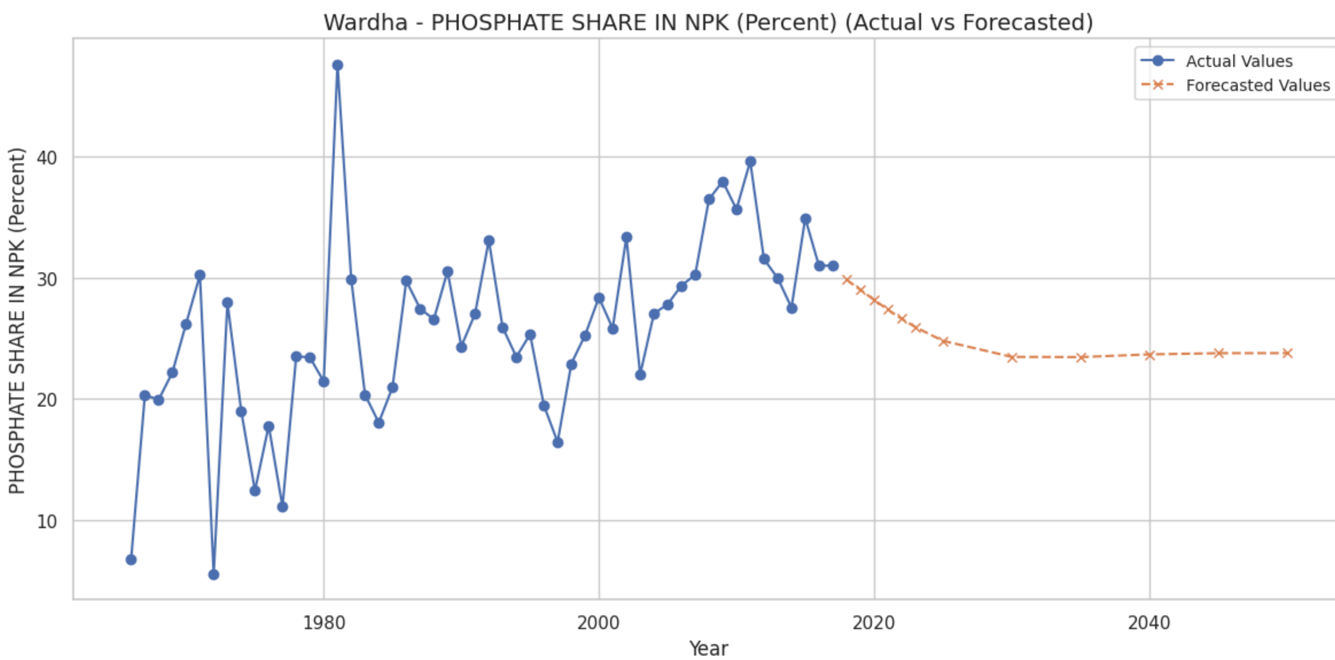

Wardha - NITROGEN SHARE IN NPK (Percent) (Actual vs Forecasted)
Actual Values
Forecasted Values
NITROGEN SHARE IN NPK (Percent)
90
80
70
60
50
40
1980
2000
2020
2040
Year

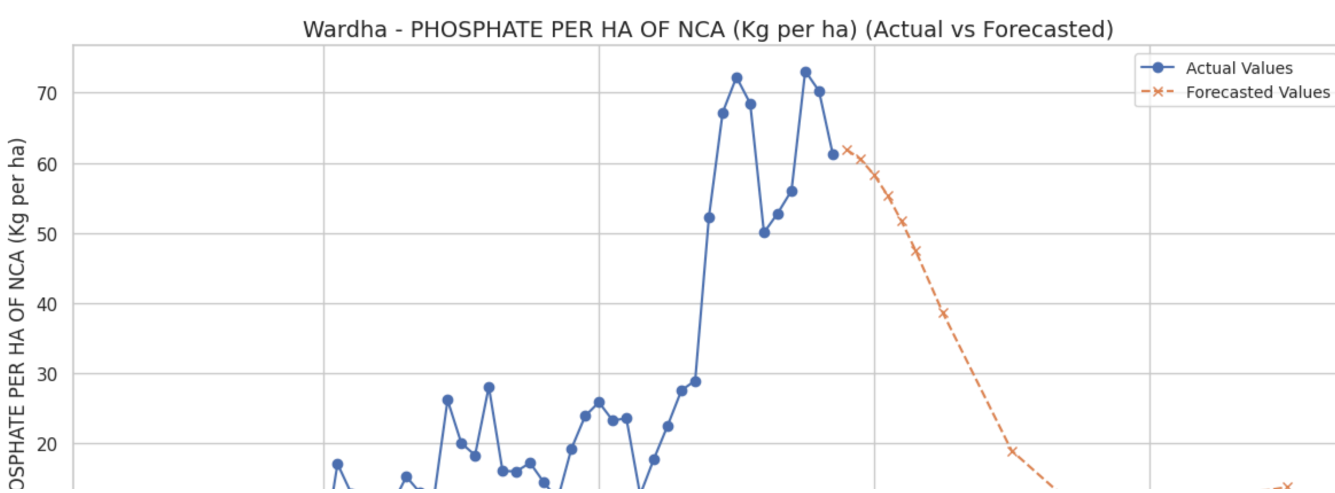

Wardha - NITROGEN PER HA OF NCA (Kg per ha) (Actual vs Forecasted)
Actual Values
Forecasted Values
NITROGEN PER HA OF NCA (Kg per ha)
120
100
80
60
40
20
0
−20
−40
1980
2000
2020
2040
Year

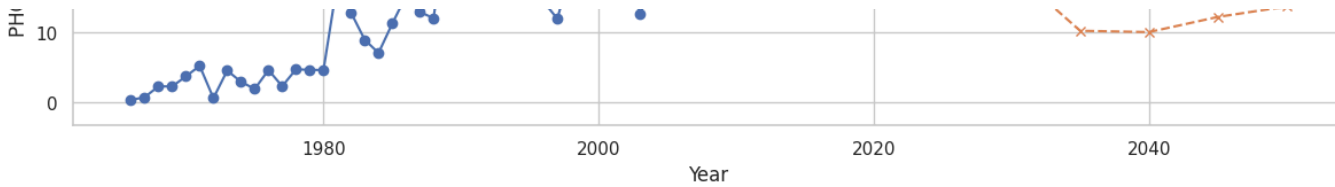

Wardha - NITROGEN PER HA OF GCA (Kg per ha) (Actual vs Forecasted)
Actual Values
Forecasted Values
NITROGEN PER HA OF GCA (Kg per ha)
120
100
80
60
40
20

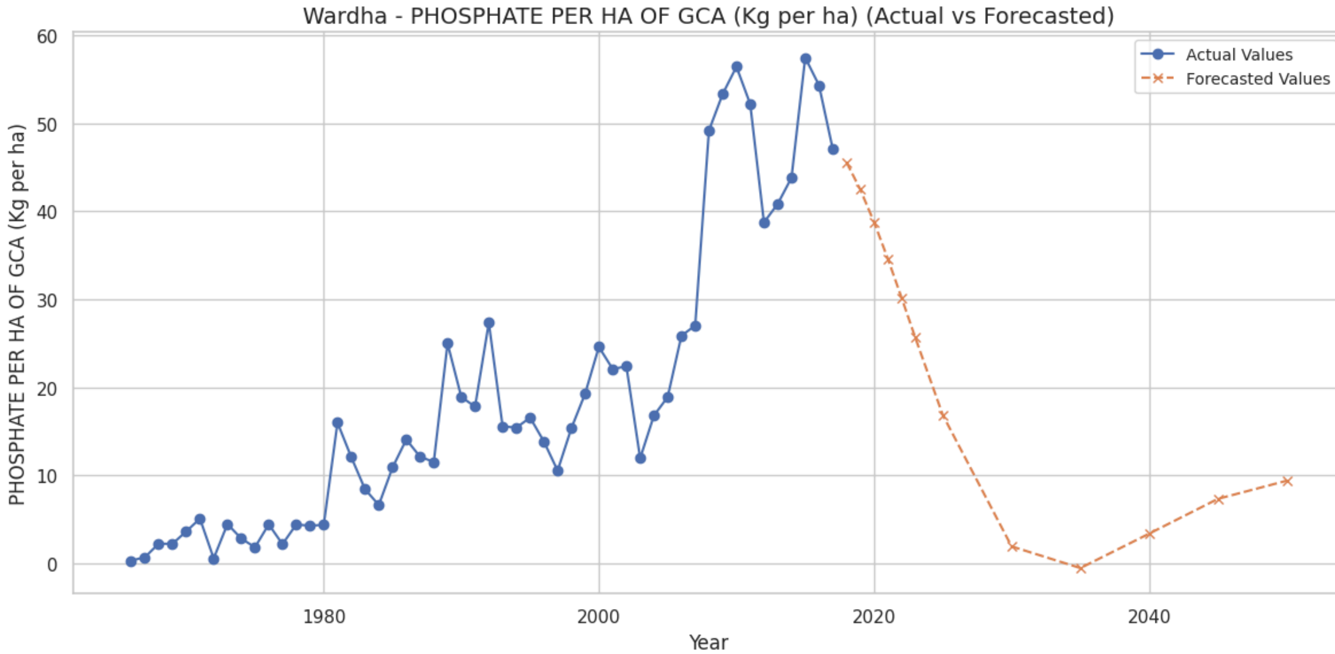

0
1980
2000
2020
2040
Year

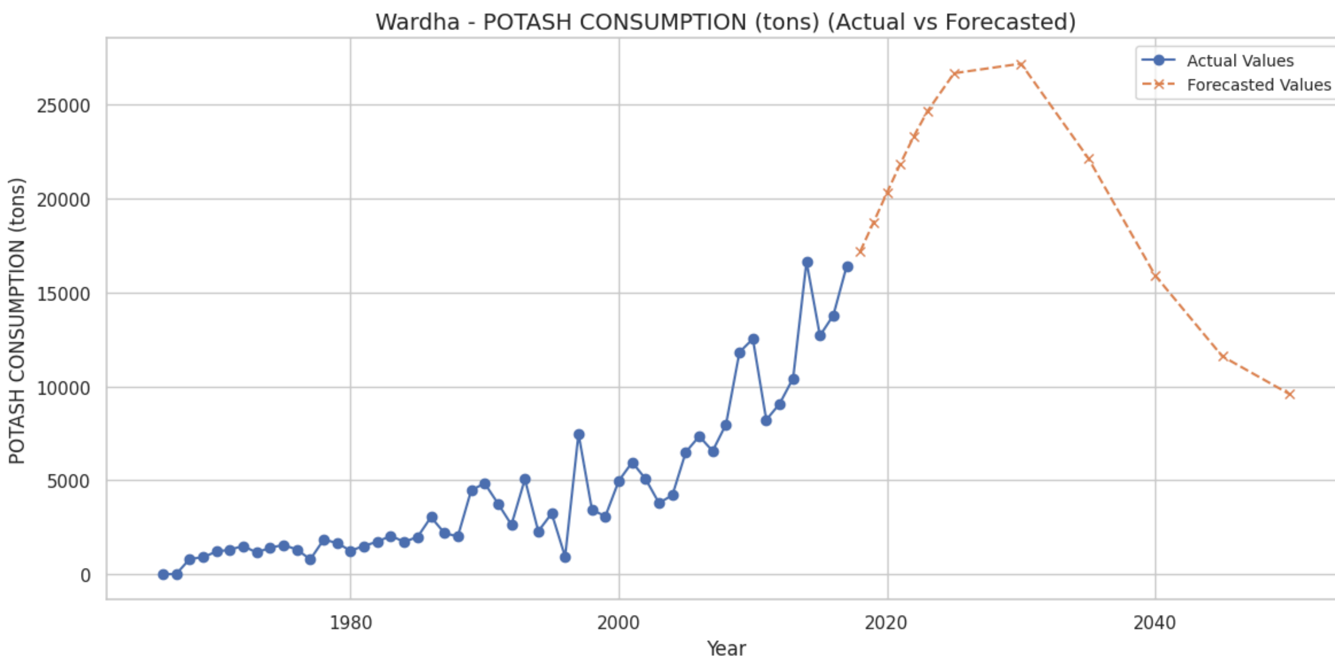

Wardha - PHOSPHATE CONSUMPTION (tons) (Actual vs Forecasted)
Actual Values
Forecasted Values
PHOSPHATE CONSUMPTION (tons)
25000
20000
15000
10000
5000
0
1980
2000
2020
2040
Year

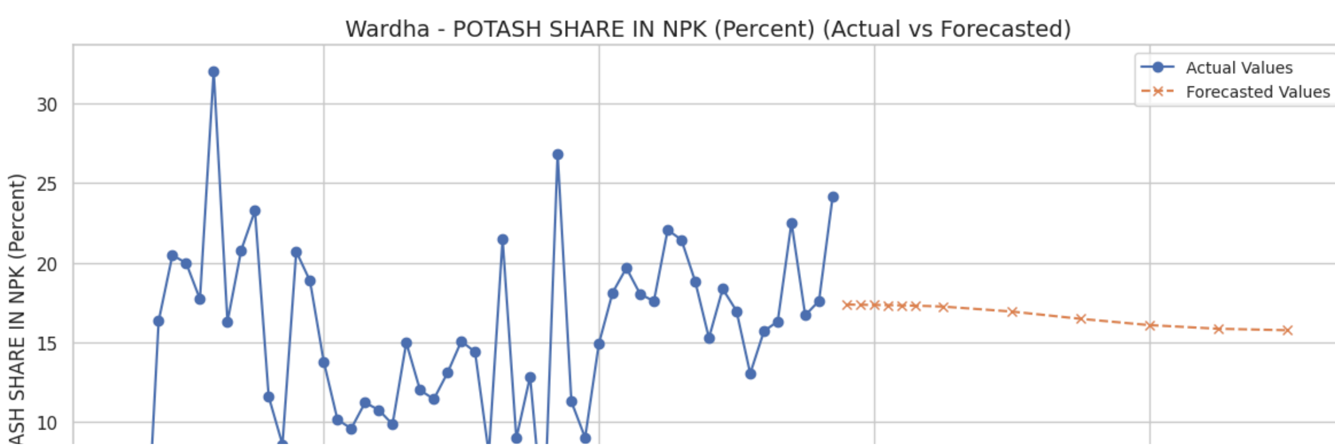

Wardha - PHOSPHATE SHARE IN NPK (Percent) (Actual vs Forecasted)
Actual Values
Forecasted Values
PHOSPHATE SHARE IN NPK (Percent)
40
30
20
10
1980
2000
2020
2040
Year

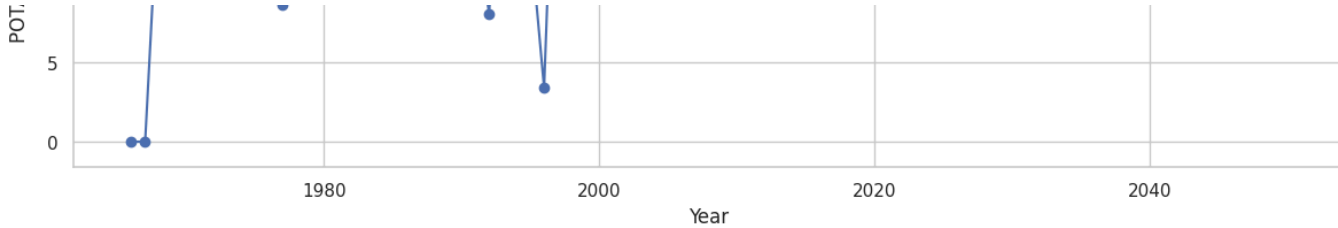

Wardha - PHOSPHATE PER HA OF NCA (Kg per ha) (Actual vs Forecasted)
Actual Values
Forecasted Values
70
60
50
40
30
20


PH
Year

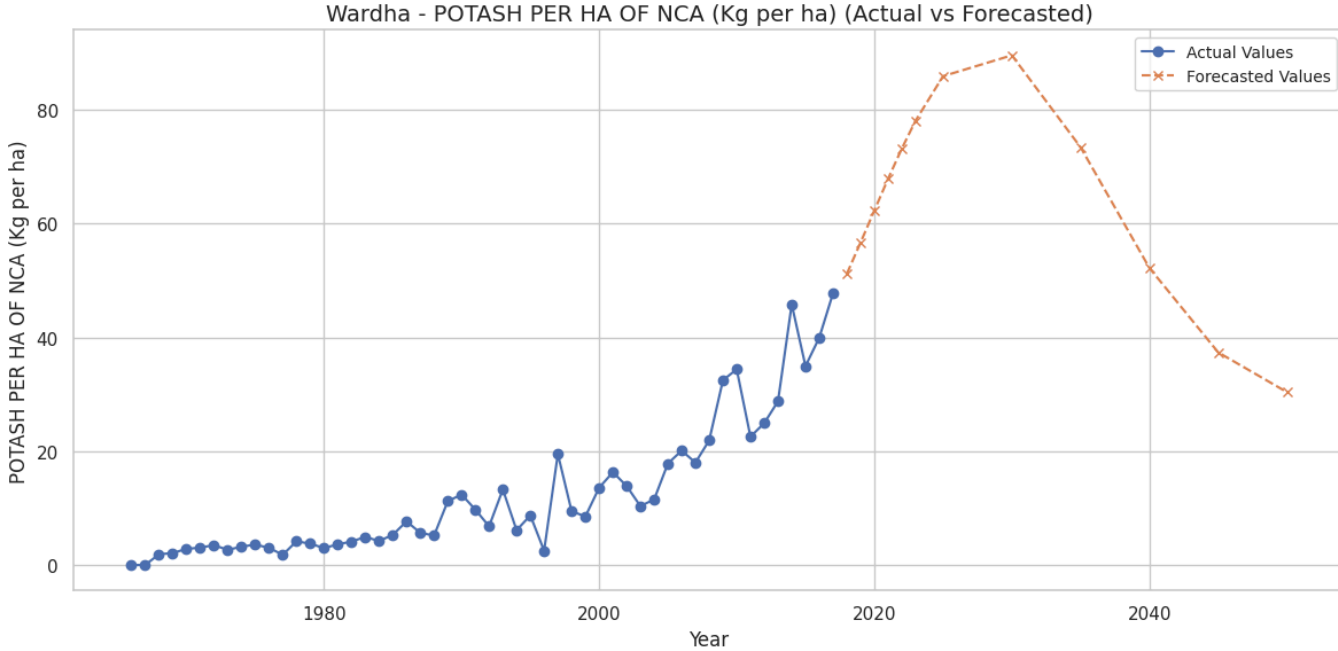

Wardha - PHOSPHATE PER HA OF GCA (Kg per ha) (Actual vs Forecasted)
Actual Values
Forecasted Values
PHOSPHATE PER HA OF GCA (Kg per ha)
Year


Wardha - POTASH CONSUMPTION (tons) (Actual vs Forecasted)
Actual Values
Forecasted Values
POTASH CONSUMPTION (tons)
Year

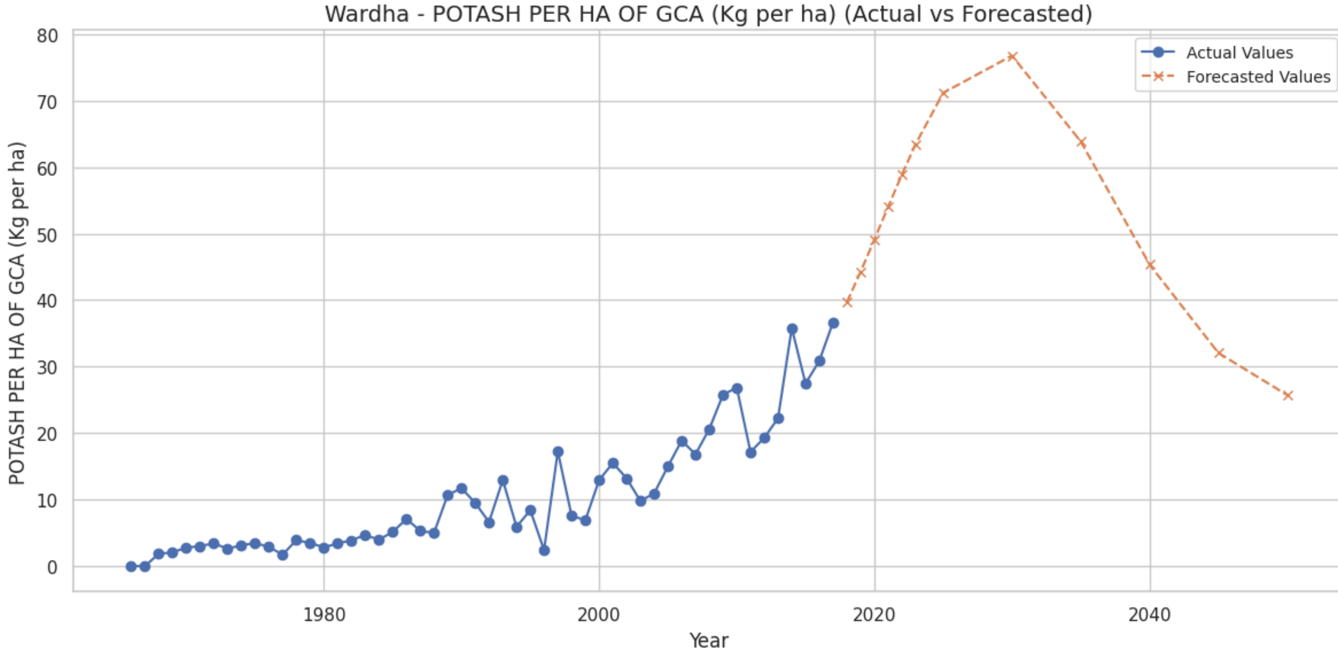

Wardha - POTASH SHARE IN NPK (Percent) (Actual vs Forecasted)
Actual Values
Forecasted Values
ASH SHARE IN NPK (Percent)



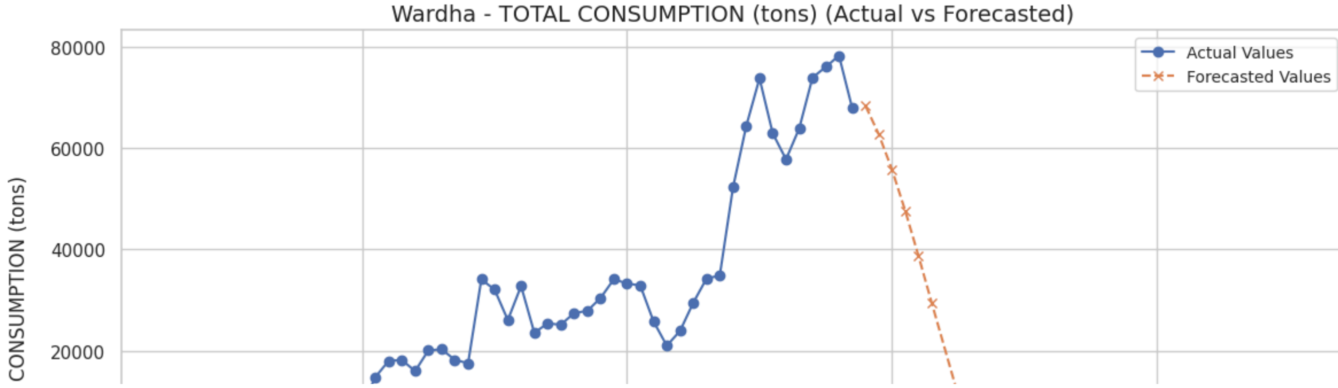
Wardha - POTASH PER HA OF NCA (Kg per ha) (Actual vs Forecasted)

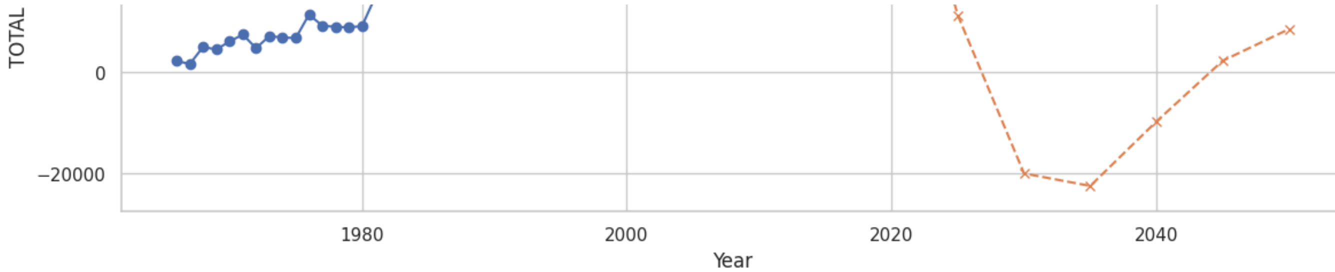


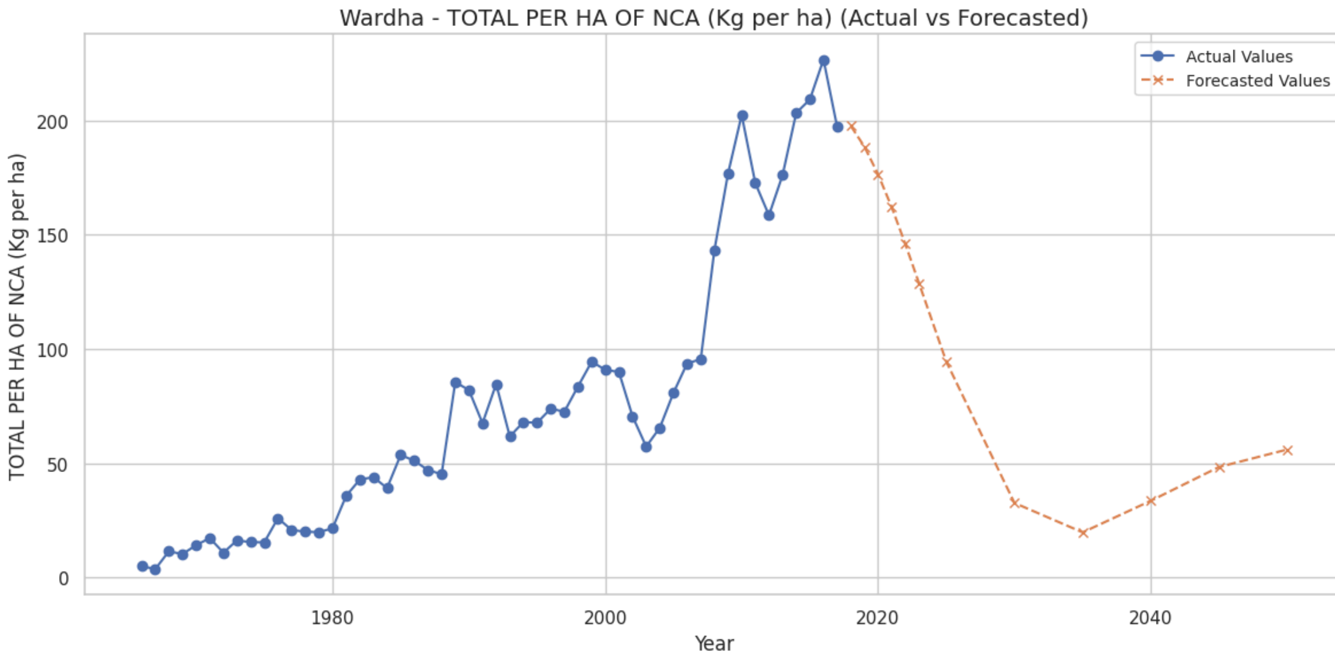
Wardha - POTASH PER HA OF GCA (Kg per ha) (Actual vs Forecasted)

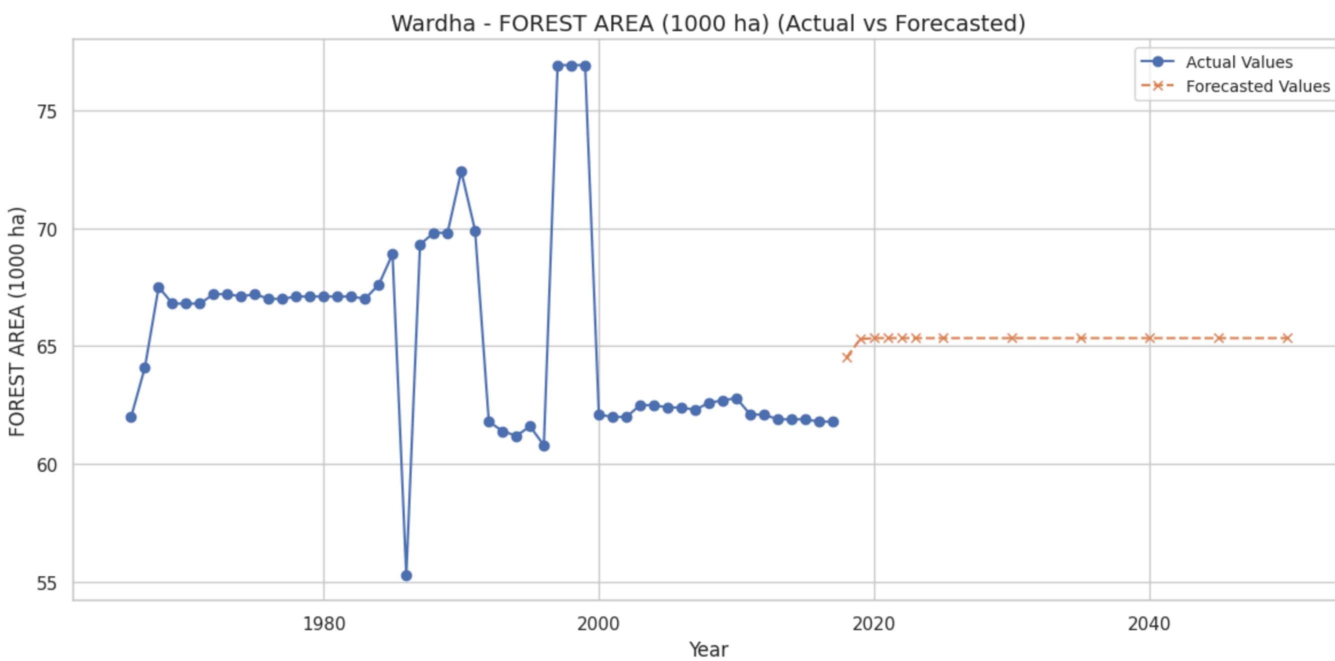


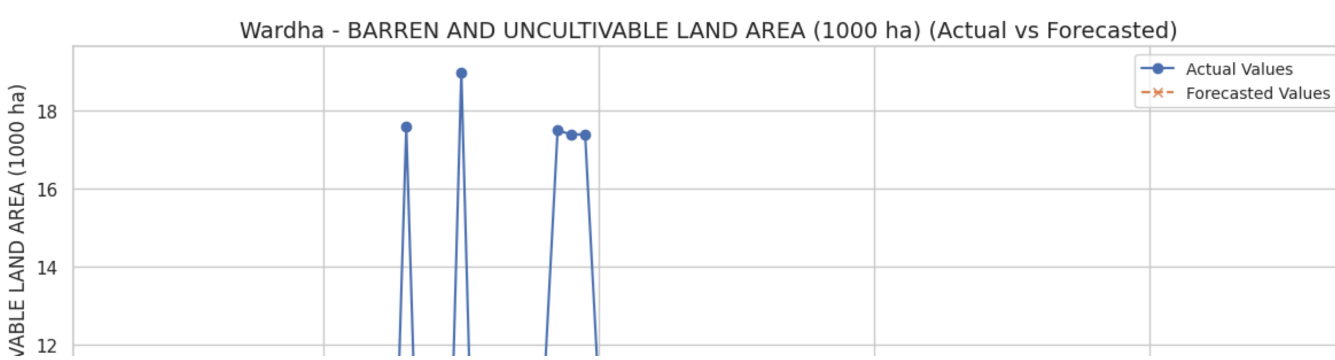
Wardha - TOTAL CONSUMPTION (tons) (Actual vs Forecasted)

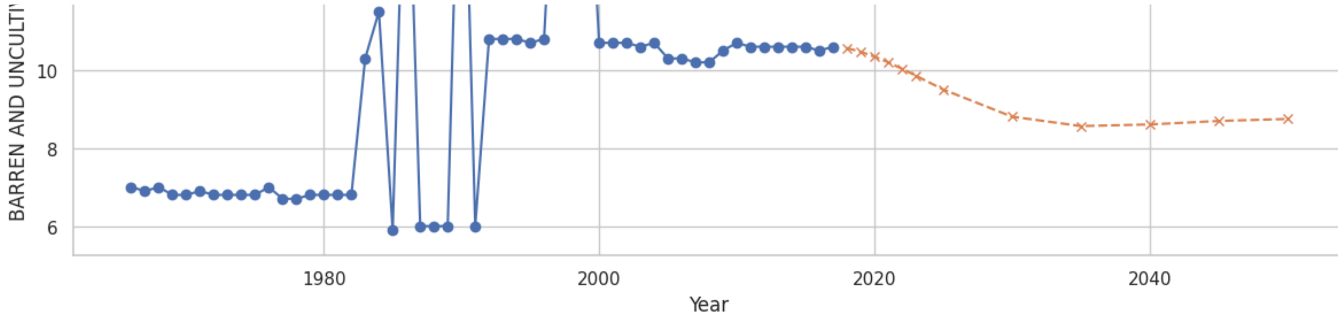



TOTAL
0
−20000
1980
2000
2020
2040
Year

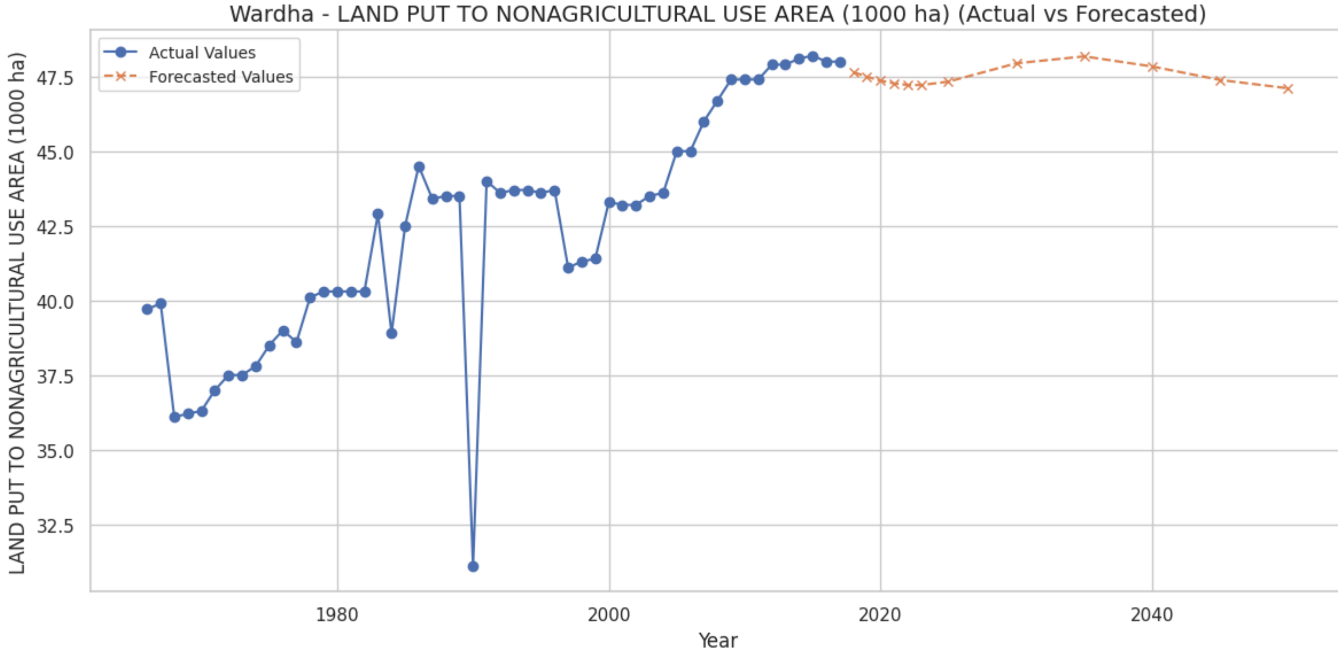

Wardha - TOTAL PER HA OF NCA (Kg per ha) (Actual vs Forecasted)
Actual Values
Forecasted Values
TOTAL PER HA OF NCA (Kg per ha)
0
50
100
150
200
1980
2000
2020
2040
Year


Wardha - FOREST AREA (1000 ha) (Actual vs Forecasted)
Actual Values
Forecasted Values
FOREST AREA (1000 ha)
55
60
65
70
75
1980
2000
2020
2040
Year

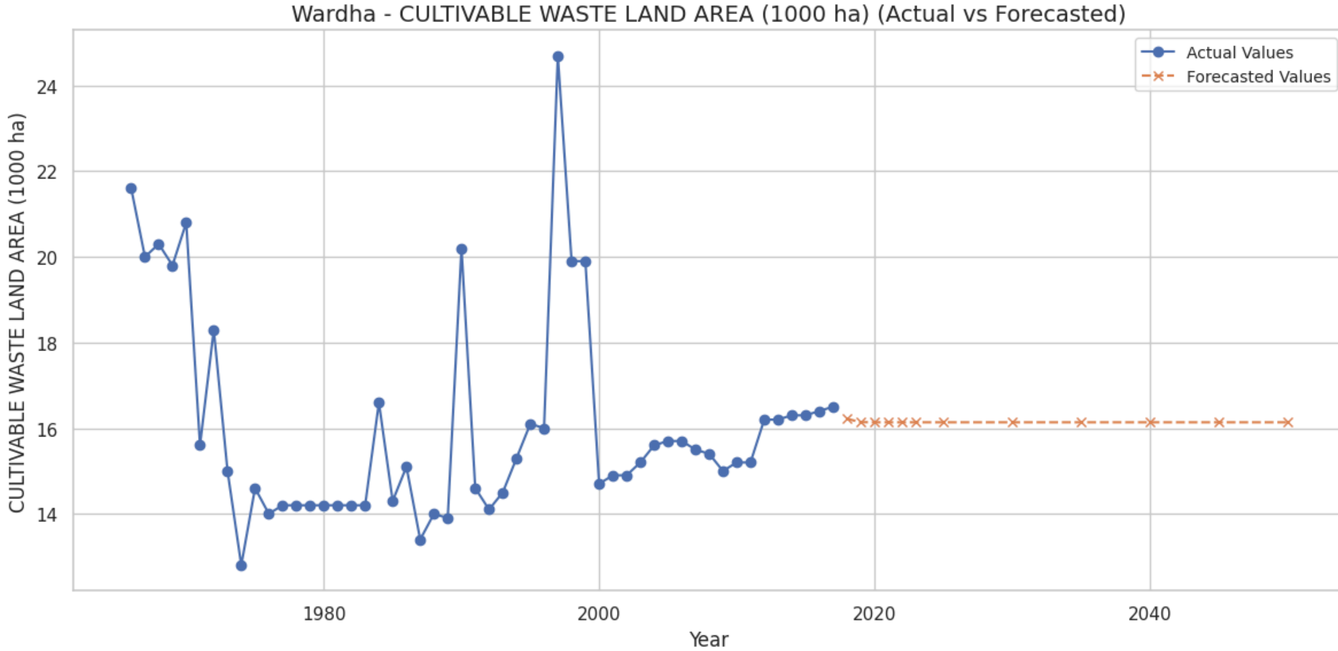

Wardha - BARREN AND UNCULTIVABLE LAND AREA (1000 ha) (Actual vs Forecasted)
Actual Values
Forecasted Values
12
14
16
18



Wardha - LAND PUT TO NONAGRICULTURAL USE AREA (1000 ha) (Actual vs Forecasted)



Wardha - CULTIVABLE WASTE LAND AREA (1000 ha) (Actual vs Forecasted)



Wardha - PERMANENT PASTURES AREA (1000 ha) (Actual vs Forecasted)