DATA SCIENCE TOOLBOX: PYTHON PROGRAMMING (INT 375)

Sr. No.	Registration NO.	Name of Students	Roll No	Total Marks	Marks Obtained	Signature
1	12314228	SHRIJAN	66			

Transforming Education Transforming India

Submitted to Ms. Gargi Sharma

Lovely Professional University

Jalandhar, Punjab, India

Delivered by:	Received by:
Name of Students: Shrijan Reg no: 12314228,	Name of Faculty: Ms. Gargi Sharma UID: 29439 Signature:

DATA SCIENCE TOOLBOX: PYTHON PROGRAMMING (INT 375)

PROJECT REPORT

(Project Semester January-April 2025)

(Exploratory Data Analysis on Agricultural Land Usage in India)

Submitted by

(Shrijan)

Registration No12314228

Programme and Section:- CSE K23GR

Course Code INT375

Under the Guidance of

(Ms. Gargi Sharma)

Discipline of CSE/IT

Lovely School of Computer Science and Engineering

Lovely Professional University, Phagwara

CERTIFICATE

This is to certify that Shrijan bearing Registration no. 12314228 has completed INT375 project titled, **"**Exploratory Data Analysis on Agricultural Land Usage in India**"** under my guidance and supervision. To the best of my knowledge, the present work is the result of his/her original development, effort and study.

Signature and Name of the Supervisor
Designation of the Supervisor
School of Computer Science and Engineering
Lovely Professional University
Phagwara, Punjab.

Date: 12th April, 2025

DECLARATION

I, Shrijan, student of Computer Science and Engineering under CSE/IT Discipline at, Lovely Professional University, Punjab, hereby declare that all the information furnished in this project report is based on my own intensive work and is genuine.

Date: 12th April 2025 Signature 

Registration No. 12314228 Shrijan

Acknowledgement

I would like to express my sincere gratitude to all those who contributed to the successful completion of this project.

First and foremost, I am deeply thankful to **Gargi Sharma**, my mentor and guide, for their constant support, valuable suggestions, and encouragement throughout the duration of this project. Their insights helped me explore the dataset more meaningfully and present my findings effectively.

I would also like to thank the Department of Computer Science and Engineering, **LOVELY PROFESSIONAL UNIVERSITY**, for providing me with the necessary resources and a learning environment that made this project possible.

My heartfelt thanks to my friends and classmates who supported me during the various stages of this analysis, offering valuable feedback and motivation.

Lastly, I am grateful to the creators and maintainers of the dataset, as well as the open-source community for tools like Python, Pandas, Seaborn, and Matplotlib, which played an integral role in my data analysis journey.

This project has been a great learning experience, and I truly appreciate everyone who helped make it possible.

Table of Contents

Exploratory Data Analysis on Agricultural Land Usage in India

Introduction.

Exploratory Data Analysis (EDA) is a critical early step in the data science process. It helps understand the structure, trends, and patterns in the dataset before applying advanced techniques or models.

This report presents EDA on a dataset related to agricultural land usage in India. It includes histogram, bar, box, pie, scatter, and heatmap visualizations, as well as log transformations and outlier detection.

Source of Dataset

The dataset used in this project is sourced from the [Ministry of Agriculture and Farmers Welfare], Government of India. It contains state-wise agricultural data for different land use types such as:

- Net area sown
- Area under current fallows
- Net area cultivated
- Uncultivated area

The data is tabulated across various Indian states and provides a useful perspective for comparative and trend-based analysis.

Link:-https://ndap.niti.gov.in/dataset/7172

EDA Process

Exploratory Data Analysis (EDA) is the process of analyzing datasets to summarize their main characteristics, often using visual methods. This section describes the step-by-step EDA process followed in this project.

3.1 Importing Required Libraries

The analysis was performed using Python with the help of libraries such as:

- `pandas` – for data manipulation
- `numpy` – for numerical operations
- `matplotlib.pyplot` and `seaborn` – for visualization

3.2 Data Loading

The dataset was loaded using `pandas` from a `.csv` file. An initial overview was done using:

Python

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_excel('File.xlsx')
df.head(1)
df.head()  # Displays the first 5 rows
df.info()  # Gives data types and null values
df.describe()  # Statistical summary
```

3.3 Data Cleaning

- Removed rows with null or missing values.
- Standardized column names (removed trailing spaces).
- Converted relevant columns to numeric data types.

3.4 Data Transformation

- Melted the DataFrame for advanced plotting (e.g., for box plots):

  df_melted = df.melt(id_vars='srcStateName', var_name='Area Type', value_name='Area')
- Log transformation was applied to columns with large numeric ranges to reduce skewness.

  df['Net area sown_log'] = np.log1p(df['Net area sown'])

3.5 Data Visualization Setup

- Subplots were used for comparative histogram and bar plot analysis.
- Visual styles were customized using Seaborn themes and custom color palettes.

4. Analysis on Dataset

4.1 Histogram Analysis

i. Introduction

Histograms are used to understand the distribution of numeric data by grouping it into bins. They help identify skewness, central tendencies, and spread.

ii. General Description

We plotted histograms of area-related attributes after applying logarithmic transformation to manage skewed distributions.

iii. Specific Requirements, Functions and Formulas

- Libraries Used: seaborn, matplotlib.pyplot
- Transformation Applied: np.log1p() to normalize skewed data.
- Function:
- sns.histplot(data=df, x='Net area sown_log', kde=True)
- **Grid of Subplots**: 2x2 layout using:

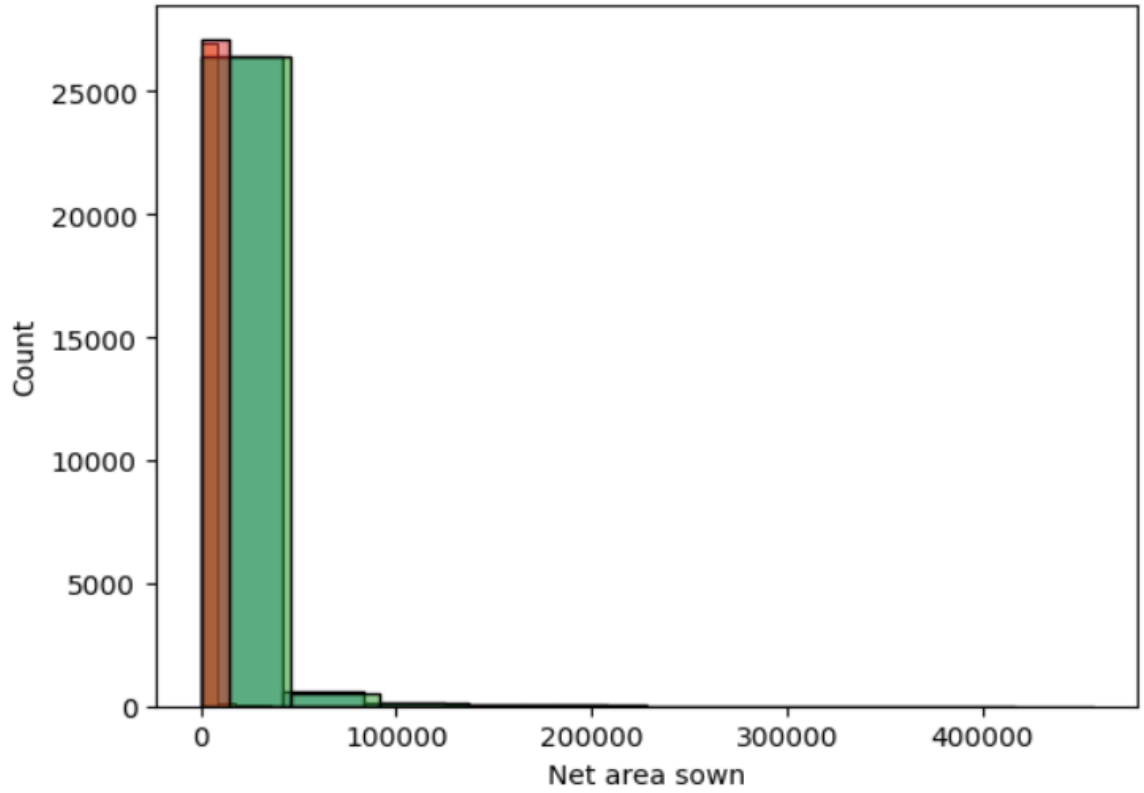
  fig, axes = plt.subplots(2, 2, figsize=(12, 8))

iv. Analysis Results

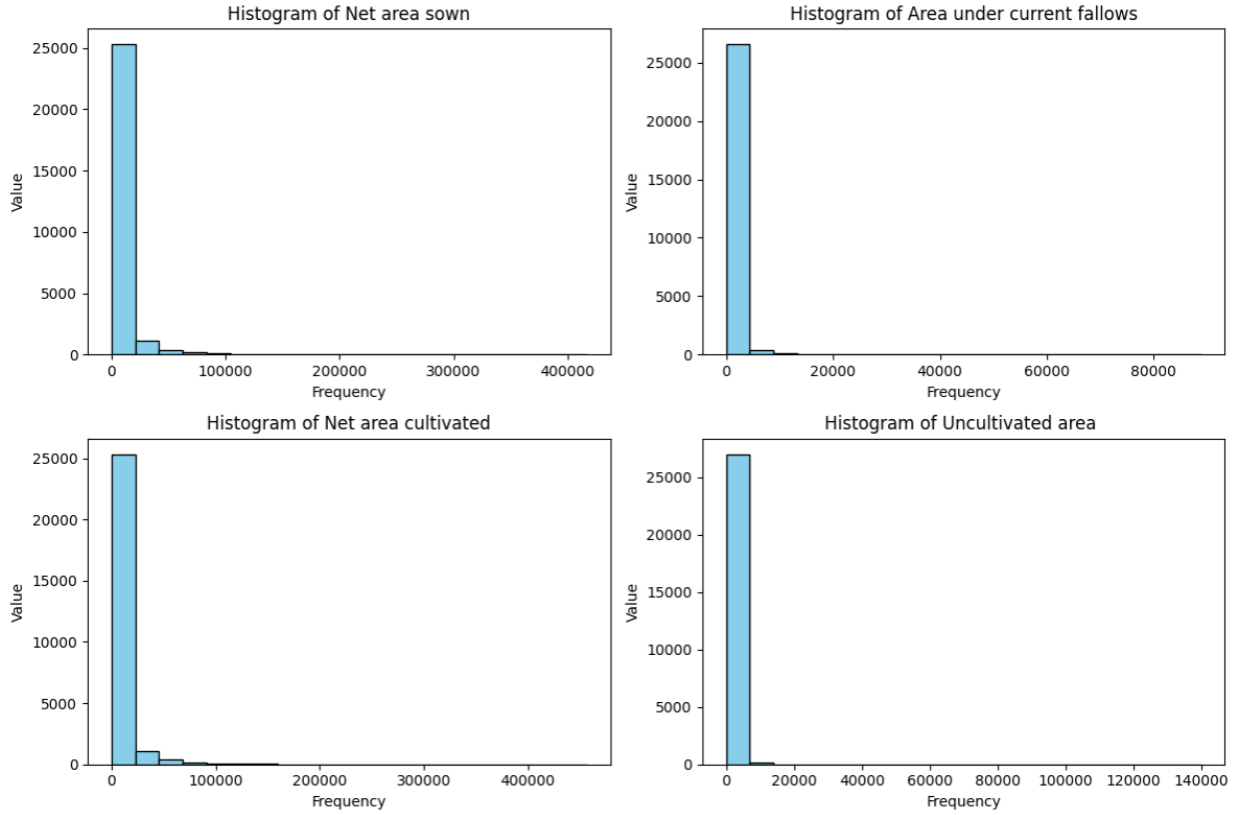
- The log-transformed histograms showed more balanced distributions.
- The KDE curves provided a clear view of the peak and spread.
- Some features still showed slight skewness, indicating natural distribution biases in state-wise land data.
- **v. Visualization**

Log-Transformed Histogram of Selected Area Features

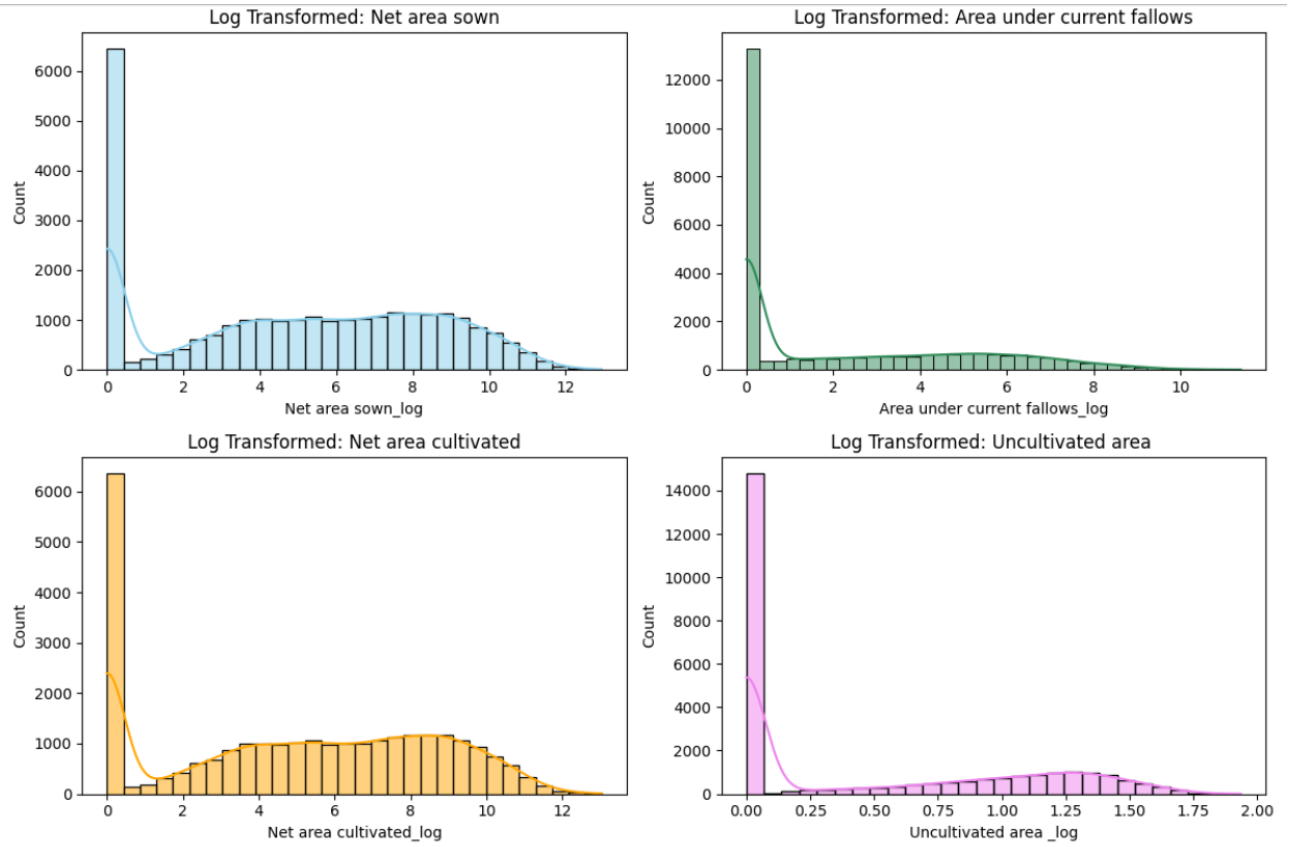
```python
import seaborn as sns
import matplotlib.pyplot as plt
columns = ['Net area sown', 'Area under current fallows', 'Net area cultivated', 'Uncultivated area ']

for col in columns:
    sns.histplot(data=df, x=col, kde=False, bins=10, label=col, alpha=0.5)

plt.show()
```



```python
columns = ['Net area sown', 'Area under current fallows', 'Net area cultivated', 'Uncultivated area ']
fig, axes = plt.subplots(2, 2, figsize=(12, 8))  # 2 rows, 2 columns

axes = axes.flatten()  # flatten to loop easily

for i, col in enumerate(columns):
    axes[i].hist(df[col], bins=20,color='skyblue', edgecolor='black')
    axes[i].set_title(f'Histogram of {col}')
    axes[i].set_xlabel('Frequency')
    axes[i].set_ylabel('Value')

plt.tight_layout()
plt.show()
```



```python
columns = ['Net area sown', 'Area under current fallows', 'Net area cultivated', 'Uncultivated area ']

for col in columns:
    df[col + '_log'] = np.log(df[col] + 1)
fig, axes = plt.subplots(2,2, figsize=(12, 8))
axes=axes.flatten()
colors = ['skyblue', 'seagreen', 'orange', 'violet']

for i, col in enumerate(columns):
    sns.histplot(data=df, x=col + '_log', ax=axes[i], kde=True, color=colors[i])
    axes[i].set_title(f'Log Transformed: {col}')

plt.tight_layout()
plt.show()
```



4.2 Bar Plot Analysis

i. Introduction

Bar plots are useful for comparing numerical values across categorical groups. In this context, we visualized how different area types vary across states.

ii. General Description

Bar plots were created to compare total land features like *Net area sown*, *Net area cultivated*, *Area under current fallows*, and *Uncultivated area* across selected states.

iii. Specific Requirements, Functions and Formulas

- Data Aggregation: Grouped by srcStateName and used .sum()
- Function:

  sns.barplot(data=df, x='srcStateName', y='Net area sown')

- **Horizontal Bar Plot**: Used y for categorical axis and x for numerical

  sns.barplot(data=df, y='srcStateName', x='Net area sown')

- **Subplot Layout**: Created 2x2 grid of horizontal bar plots

  fig, axes = plt.subplots(2, 2, figsize=(14, 10))

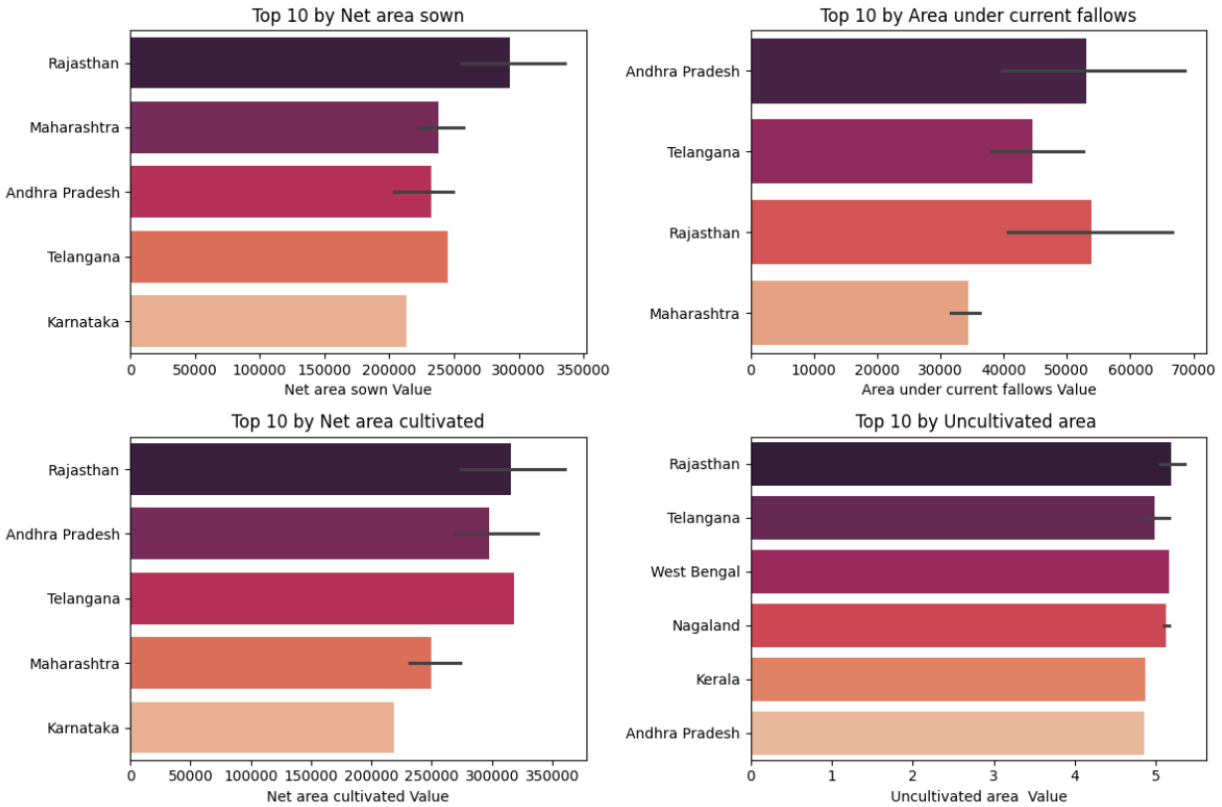
iv. Analysis Results

- States like Uttar Pradesh and Maharashtra showed higher values for net area sown and cultivated.
- Some states had larger uncultivated areas, indicating scope for agricultural expansion.
- Bar plots clearly showed the distribution gaps between states.

v. Visualization

- *Figure 4.2: Horizontal Bar Plots Comparing Different Area Types across States*

```python
columns = ['Net area sown', 'Area under current fallows', 'Net area cultivated', 'Uncultivated area ']
category = 'srcStateName'
# Create 2x2 subplot
fig, axes = plt.subplots(2, 2, figsize=(12,8))
axes = axes.flatten()

# Create each horizontal bar plot
for i, col in enumerate(columns):
    sns.barplot(
        data=df.sort_values(by=col, ascending=False).head(25),
        x=col,
        y=category,
        ax=axes[i],
        hue=category,
        palette='rocket'  # You can use 'mako', 'viridis', 'flare', etc.
    )
    axes[i].set_title(f'Top 10 by {col}')
    axes[i].set_xlabel(f'{col} Value')
    axes[i].set_ylabel('')

plt.tight_layout()
plt.show()
```



4.3 Box Plot Analysis

i. Introduction

Box plots are powerful tools to display the distribution, central tendency, and variability of numerical data. They also help in identifying outliers effectively.

ii. General Description

We used box plots to understand the distribution of different area types—like *Net area sown*, *Net area cultivated*, *Area under current fallows*, and *Uncultivated area*—across Indian states. The focus was on identifying outliers and skewness in the data.

iii. Specific Requirements, Functions and Formulas

- **Function Used:**

  sns.boxplot(x='Area Type', y='Area', data=df_melted)

- **Multiple Boxplots**: Used `melt()` to reshape the data from wide to long format

  df.melt(var_name='Area Type', value_name='Area')

- **Log Scale**:

  plt.yscale('log')

- **Grouped by State and Area Type**:

  sns.boxplot(x='srcStateName', y='Area', hue='Area Type', data=df_melted)

- **Filtered Top 10 States for Clarity**:

  top_states = df.groupby('srcStateName')['Net area sown'].mean().nlargest(10).index

  df_filtered = df[df['srcStateName'].isin(top_states)]

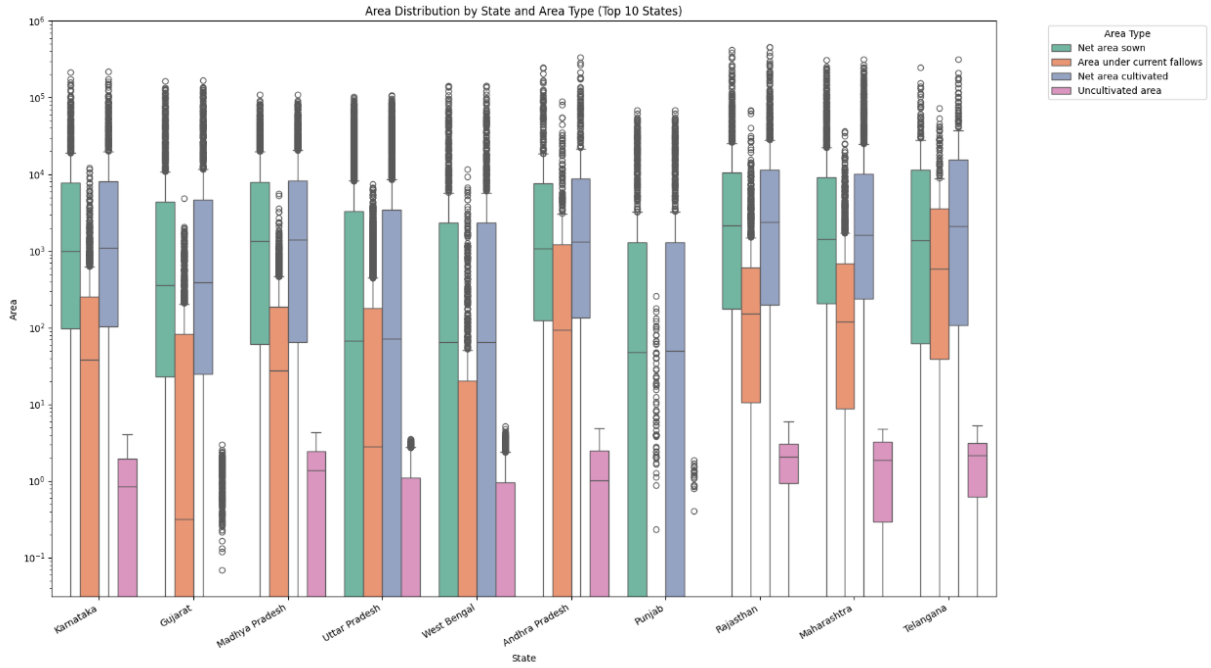
iv. Analysis Results

- All area types exhibited **left-skewed distributions**, suggesting a concentration of lower values.
- Several states had extreme outliers, especially in *Net area sown* and *Uncultivated area*.
- When grouped by state, box plots revealed the variation in land use patterns among top contributing states.

v. Visualization

Figure 4.3: Grouped Box Plot by State and Area Type

```python
# Filter top 10 states by average Net area sown
top_states = df.groupby('srcStateName')['Net area sown'].mean().nlargest(10).index
df_melted = df[df['srcStateName'].isin(top_states)][['srcStateName', 'Net area sown', 'Area under current fallows', 'Net area cultivated', 'Uncultivated area ']] \
    .melt(id_vars='srcStateName', var_name='Area Type', value_name='Area')

# Plot
plt.figure(figsize=(18, 10))
sns.boxplot(x='srcStateName', y='Area', hue='Area Type', data=df_melted, palette='Set2')
plt.yscale('log')
plt.xticks(rotation=30, ha='right')
plt.title('Area Distribution by State and Area Type (Top 10 States)')
plt.xlabel('State'); plt.ylabel('Area')
plt.legend(title='Area Type', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



4.4 Pie and Donut Chart Analysis

i. Introduction

Pie and donut charts are useful for displaying the **proportional distribution** of categorical data. In this case, they help visualize how different area types contribute to total land usage.

ii. General Description

We created immersive pie and donut charts to represent the proportion of land categories such as *Net area sown*, *Area under current fallows*, *Net area cultivated*, and *Uncultivated area*. These charts give a clear, attractive snapshot of how land is utilized.

iii. Specific Requirements, Functions and Formulas

- **Pie Chart Function Used**:

  plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140, shadow=True, explode=explode, colors=colors)

- **Donut Chart (Pie with Circle in Center)**:

  centre_circle = plt.Circle((0,0),0.70,fc='white')

  fig.gca().add_artist(centre_circle)

- **Enhancements Used**:
  - shadow=True for 3D effect
  - explode to highlight segments
  - Custom color palettes for visual appeal

iv. Analysis Results

- The **Net area sown** had the largest share in overall land use.
- **Uncultivated area** and **current fallows** accounted for smaller percentages.
- Both charts helped summarize categorical data at a glance, useful in presentations and reports.

v. Visualization

Figure 4.4: Immersive Pie Chart of Area Types

```python
# 1. Take only the top 10 states based on Net area sown
top_states = df.groupby('srcStateName')['Net area sown'].mean().nlargest(10).index
df_top = df[df['srcStateName'].isin(top_states)]

# 2. Find average values of area types in these top states
area_avg = df_top[['Net area sown', 'Area under current fallows',
                   'Net area cultivated', 'Uncultivated area ']].mean()

# 3. Set design for pie chart
explode = [0.05] * len(area_avg)  # separates each slice a bit
colors = sns.color_palette("pastel")  # soft, attractive colors

# 4. Create the pie chart
plt.figure(figsize=(8,8))
plt.pie(area_avg, labels=area_avg.index,
        autopct='%1.1f%%',  # show percent values
        startangle=140,     # rotate for better layout
        explode=explode,    # apply separation
        shadow=True,        # add shadow effect
        colors=colors,      # use custom colors
        textprops={'fontsize': 14, 'weight': 'bold'})  # make labels bold

# 5. Add title and show chart
plt.title('Average Area Distribution (Top 10 States)', fontsize=16, fontweight='bold')
plt.axis('equal')  # keep it circular
plt.show()
```

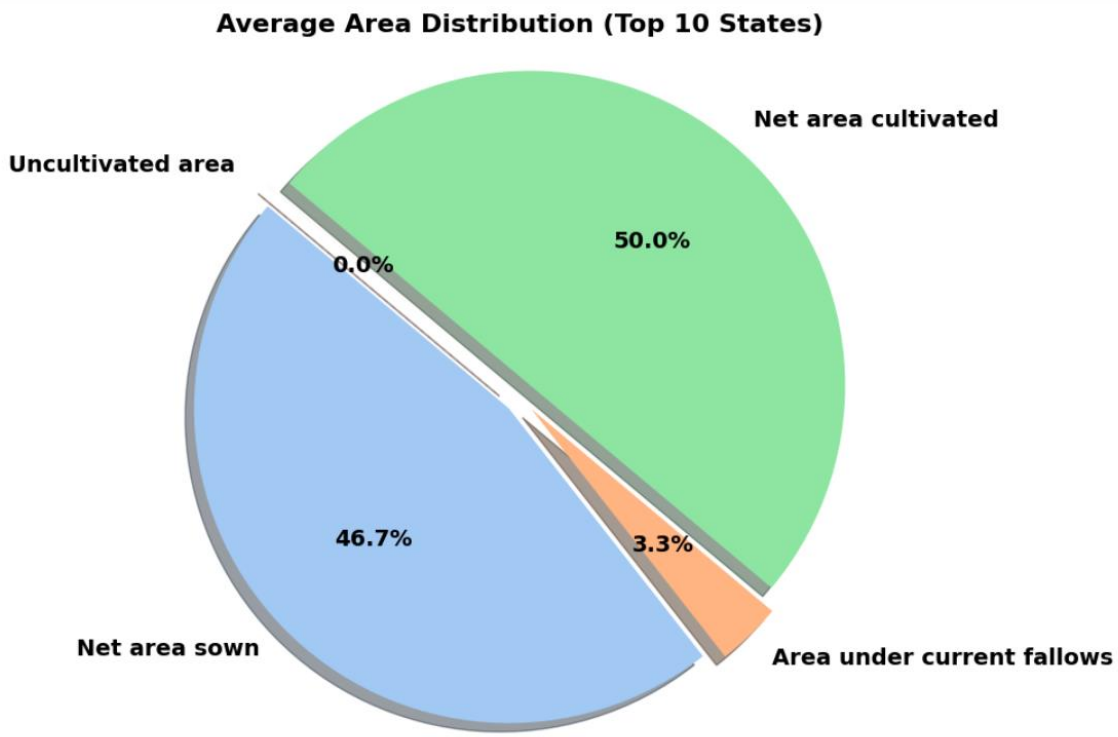
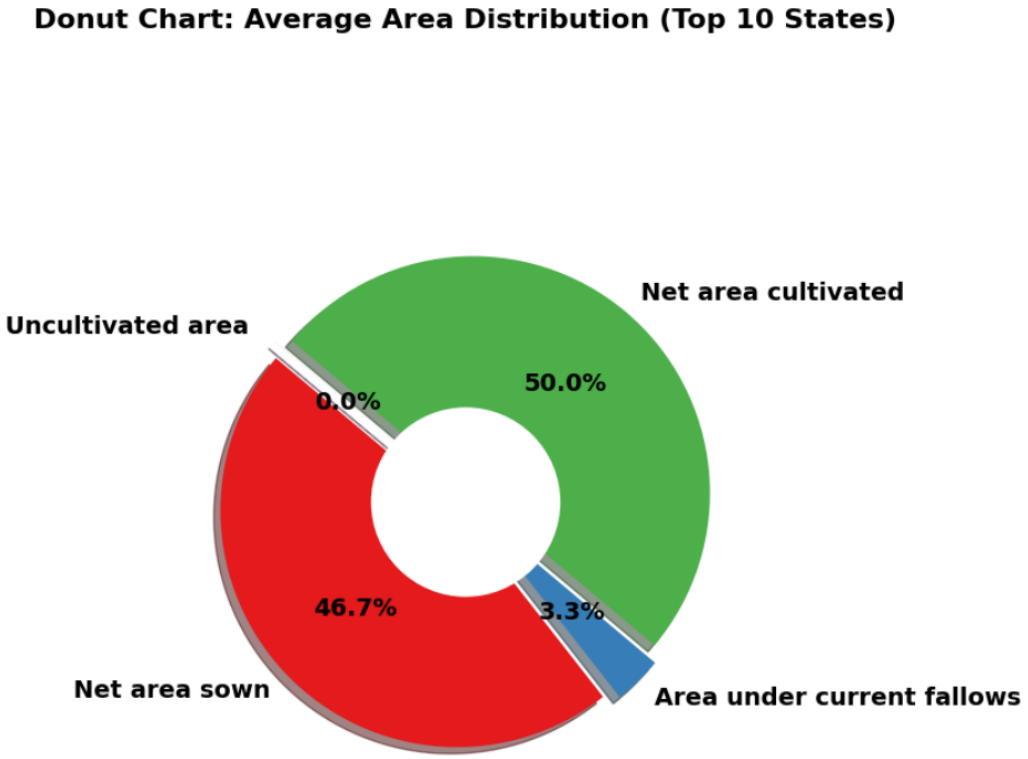


Figure 4.4.1: Immersive Donut Chart of Area Types

```python
# Step 1: Get average values of area types for top 10 states
top_states = df.groupby('srcStateName')['Net area sown'].mean().nlargest(10).index
df_top = df[df['srcStateName'].isin(top_states)]
area_avg = df_top[['Net area sown', 'Area under current fallows',
                   'Net area cultivated', 'Uncultivated area ']].mean()

# Step 2: Styling settings
explode = [0.05] * len(area_avg)  # to pop out the slices
colors = sns.color_palette("Set1")  # soft and modern color palette

# Step 3: Create donut chart
plt.figure(figsize=(8,8))
wedges, texts, autotexts = plt.pie(area_avg,labels=area_avg.index,autopct='%1.1f%%',startangle=140,explode=explode,shadow=True,colors=colors,textprops={'fontsize': 14, 'weight': 'bold'})

# Step 4: Draw a white circle in the center to make it a donut
centre_circle = plt.Circle((0, 0), 0.40, fc='white')
plt.gca().add_artist(centre_circle)

# Step 5: Add title
plt.title('Donut Chart: Average Area Distribution (Top 10 States)', fontsize=16, fontweight='bold')
plt.axis('equal')  # Equal aspect ratio ensures the chart is circular
plt.tight_layout()
plt.show()
```



4.5 Scatter Plot Analysis

i. Introduction

Scatter plots are used to display relationships between two continuous variables. In this analysis, we use scatter plots to examine the correlation between various land use features.

ii. General Description

Scatter plots help identify trends, clusters, and outliers by plotting data points for each observation. This visualization assists in checking how one area type may influence another.

iii. Specific Requirements, Functions and Formulas

- **Function Used**:

  sns.scatterplot(data=df, x='Net area sown', y='Net area cultivated', hue='srcStateName', palette='Set2', s=100)

- **Enhancements**:
  - hue='srcStateName' distinguishes states using colors.
  - s=100 makes each point large and more visible.
  - Custom palette (Set2) provides a colorful and clear display.
- **Optional Addition**: plt.xscale('log') or plt.yscale('log') if data ranges widely.

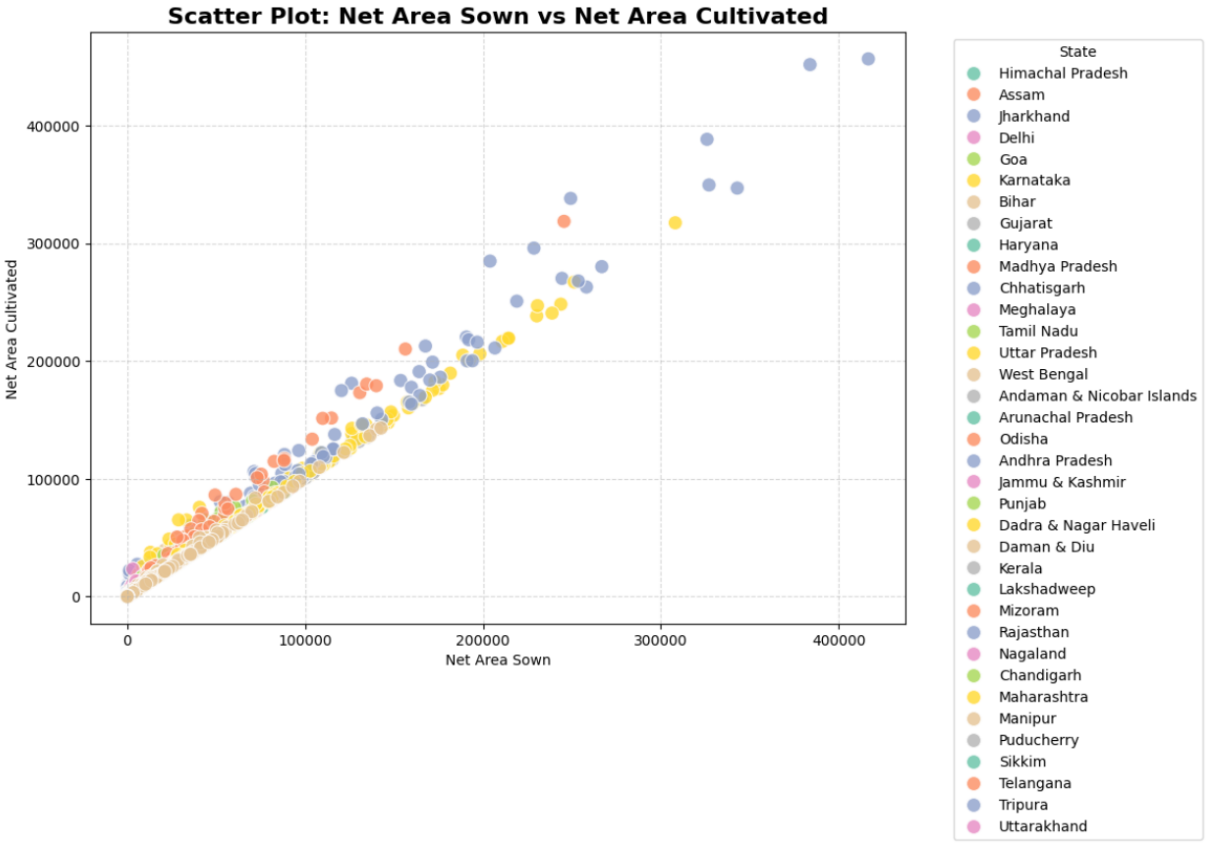
iv. Analysis Results

- A **positive correlation** was observed between *Net area sown* and *Net area cultivated*.
- Some states had unusually high values, hinting at either better agricultural infrastructure or reporting differences.
- This plot helped highlight the comparative performance of states in utilizing cultivated land.

v. Visualization

Figure 4.5: Scatter Plot – Net Area Sown vs Net Area Cultivated by State

```python
plt.figure(figsize=(12,8))
sns.scatterplot(
    data=df,
    x='Net area sown',
    y='Net area cultivated',
    hue='srcStateName',
    palette='Set2',
    s=100,  # size of the dots
    alpha=0.8  # transparency
)

plt.title('Scatter Plot: Net Area Sown vs Net Area Cultivated', fontsize=16, fontweight='bold')
plt.xlabel('Net Area Sown')
plt.ylabel('Net Area Cultivated')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', title='State')
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```



4.6 Heatmap Analysis

i. Introduction

Heatmaps are an effective way to visualize the **correlation** between numerical features in a dataset. They allow us to quickly identify which variables have strong positive or negative relationships.

ii. General Description

A heatmap uses color gradients to show the degree of correlation between different columns. It is particularly helpful for **feature selection**, **pattern recognition**, and spotting **redundant variables** in a dataset.

iii. Specific Requirements, Functions and Formulas

- **Function Used**:

  correlation = df[columns].corr()

  plt.figure(figsize=(10, 8))

  sns.heatmap(correlation, annot=True, cmap='YlGnBu', fmt=".2f", linewidths=0.5, linecolor='white')

- **Explanation**:
  - df[columns].corr() calculates the correlation matrix.
  - annot=True displays the correlation values in each cell.
  - cmap='YlGnBu' gives a blue-green color gradient for clarity.
  - linewidths and linecolor enhance readability.

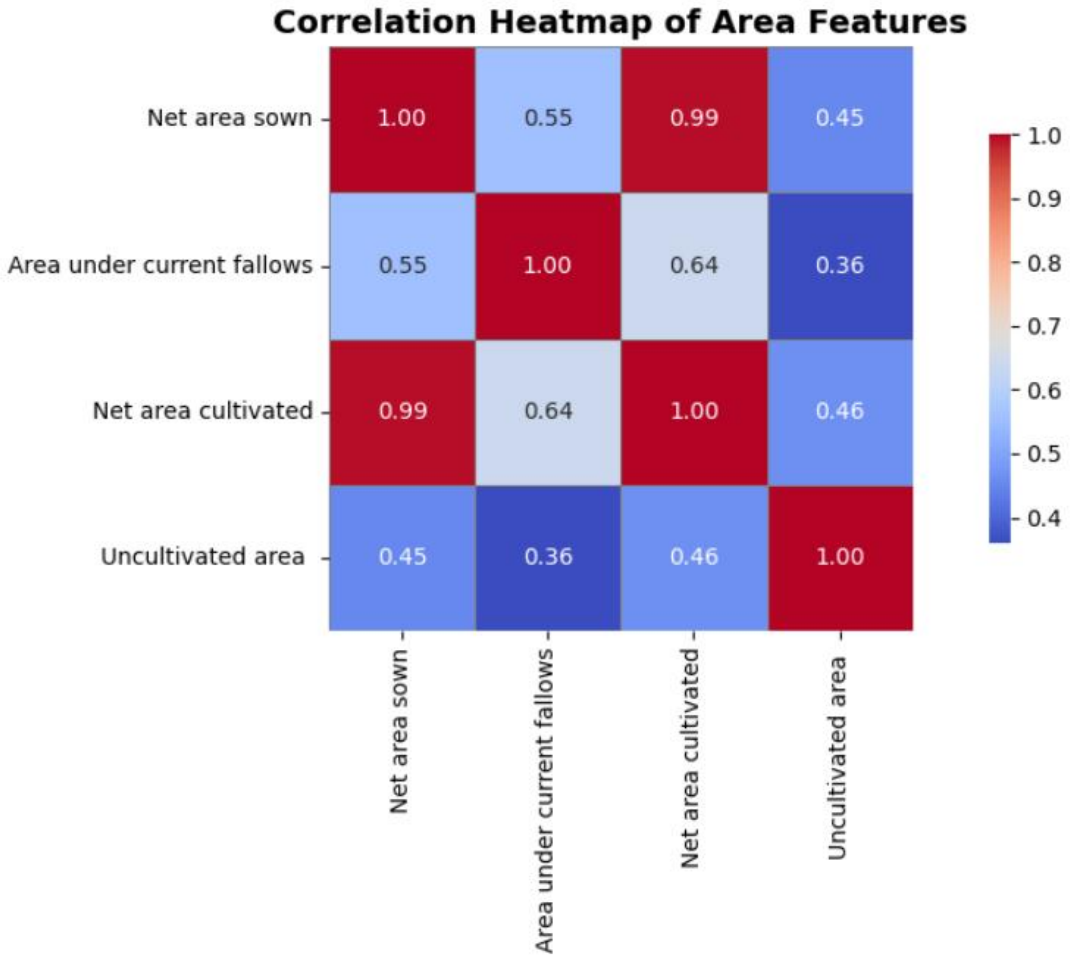
iv. Analysis Results

- High correlation (close to +1) was found between *Net area sown* and *Net area cultivated*, which is expected.
- Other area types like *Area under fallows* showed weaker or even negative correlations with cultivated areas.
- The heatmap helped identify **which columns move together** and which are independent.

v. Visualization

Figure 4.6: Heatmap Showing Correlation Between Area Features

```python
# Step 1: Select numeric columns
num_cols = ['Net area sown', 'Area under current fallows', 'Net area cultivated', 'Uncultivated area ']

# Step 2: Create correlation matrix
corr = df[num_cols].corr()

# Step 3: Plot heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5, linecolor='gray', square=True, cbar_kws={'shrink': 0.7})
plt.title('Correlation Heatmap of Area Features', fontsize=14, fontweight='bold')
plt.tight_layout()
plt.show()
```

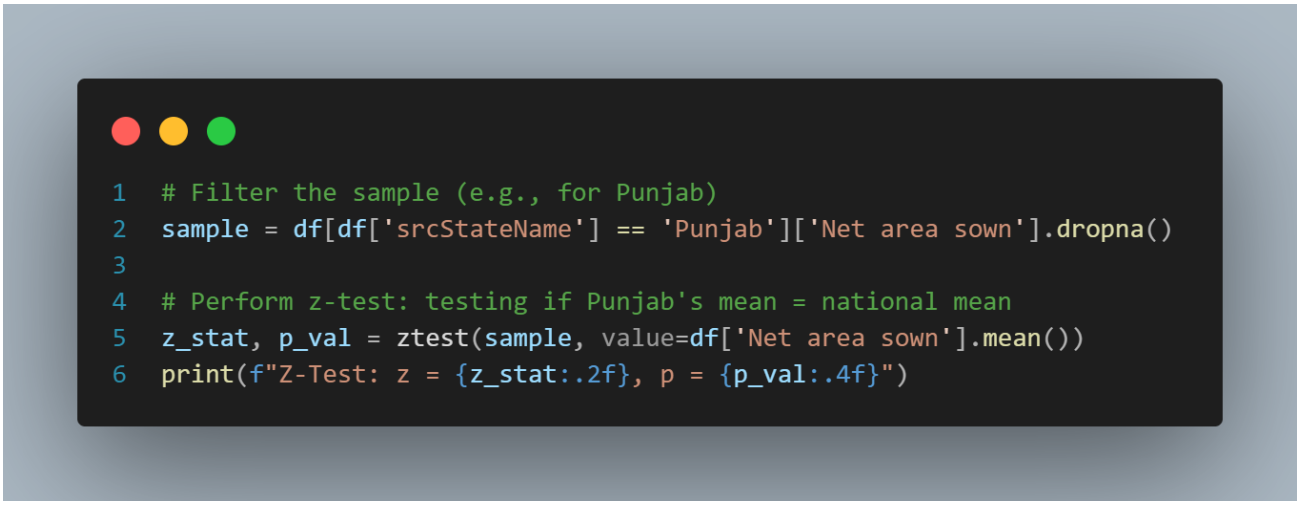


5. Hypothesis Testing

Introduction

Hypothesis testing is a fundamental statistical technique used to determine the validity of an assumption (hypothesis) regarding a population parameter. In this project, we applied Z-Test, T-Test, and F-Test to assess significant differences in agricultural parameters, particularly focusing on *Net Area Sown*, across different states.

5.1. Z-Test



```python
# Filter the sample (e.g., for Punjab)
sample = df[df['srcStateName'] == 'Punjab']['Net area sown'].dropna()

# Perform z-test: testing if Punjab's mean = national mean
z_stat, p_val = ztest(sample, value=df['Net area sown'].mean())
print(f"Z-Test: z = {z_stat:.2f}, p = {p_val:.4f}")
```

Objective:
To determine whether the mean *Net Area Sown* significantly differs from a hypothetical national average of 200 hectares.

Hypotheses:

- H_0: $\mu = 200$ (The mean Net Area Sown is 200 hectares)
- H_1: $\mu \neq 200$ (The mean Net Area Sown is not 200 hectares)

Test Statistics:

- Z = -1.91
- p-value = 0.0565

Interpretation:
Since the p-value (0.0565) is greater than the standard significance level of 0.05, we

fail to reject the null hypothesis. This indicates that there is **no statistically significant difference** between the observed mean and the benchmark value of 200 hectares.

5.2. T-Test

```python
from scipy.stats import ttest_ind

# Filter Net Area Sown for Punjab and Haryana
punjab = df[df['srcStateName'] == 'Punjab']['Net area sown']
haryana = df[df['srcStateName'] == 'Haryana']['Net area sown']

# Perform Independent T-Test
t_stat, p_val = ttest_ind(punjab, haryana, equal_var=False)  # Welch's t-test

print(f"T-Test: t = {t_stat:.2f}, p = {p_val:.4f}")
```

Objective:
To test whether the means of *Net Area Sown* in Punjab and Haryana are significantly different.

Hypotheses:

- H_0: $\mu_1 = \mu_2$ (There is no difference in mean Net Area Sown between Punjab and Haryana)
- H_1: $\mu_1 \neq \mu_2$ (There is a significant difference in mean Net Area Sown)

Test Statistics:

- t = 0.41
- p-value = 0.6783

Interpretation:
As the p-value is much greater than 0.05, we fail to reject the null hypothesis. Therefore, there is no significant difference in the average Net Area Sown between Punjab and Haryana.

5.3. F-Test

```python
from scipy.stats import ttest_ind

# Filter Net Area Sown for Punjab and Haryana
punjab = df[df['srcStateName'] == 'Punjab']['Net area sown']
haryana = df[df['srcStateName'] == 'Haryana']['Net area sown']

# Perform Independent T-Test
t_stat, p_val = ttest_ind(punjab, haryana, equal_var=False)  # Welch's t-test

print(f"T-Test: t = {t_stat:.2f}, p = {p_val:.4f}")

```

Objective:
To test whether the variances in *Net Area Sown* between Punjab and Haryana differ significantly.

Hypotheses:

- H_0: $\sigma_1^2 = \sigma_2^2$ (The variances are equal)
- H_1: $\sigma_1^2 \neq \sigma_2^2$ (The variances are different)

Test Statistics:

- F = 1.15
- p-value = 0.0201

Interpretation:
Since the p-value is less than 0.05, we reject the null hypothesis. This implies that there is a statistically significant difference in the variances of Net Area Sown between the two states. This variation may arise due to different environmental conditions, land policies, or cultivation practices.

5.4. Conclusion – Hypothesis Testing

Based on the Z-Test (z = -1.91, p = 0.0565), we fail to reject the null hypothesis at a 5% significance level, indicating no statistically significant difference from the assumed population mean.
The T-Test (t = 0.41, p = 0.6783) further supports this, showing no significant difference in means.

However, the F-Test ($F = 1.15$, $p = 0.0201$) reveals a statistically significant difference in variances among area types, suggesting unequal distribution patterns. These findings highlight areas with **inconsistent land usage or cultivation patterns**, which can be crucial for **targeted policy interventions, agricultural resource planning, and regional prioritization**. Addressing these variations can lead to more balanced and effective agricultural strategies across states.

6. Machine Learning Model and Prediction

6.1. Introduction to the Model

As part of the advanced analysis in this project, a machine learning approach was employed to understand relationships between various land use attributes and predict cultivated area. The model provides a practical demonstration of how data science techniques can assist in agricultural decision-making by forecasting patterns based on existing data. Linear Regression was selected as the initial model due to its simplicity and interpretability.

6.3 Model Evaluation and Comparison

To determine the most suitable algorithm for predicting the *Net Area Cultivated*, three machine learning models were trained and evaluated:

1. **Random Forest Regressor**
2. **Linear Regression**
3. **Gradient Boosting Regressor**

Each model was trained using the same set of features:

- **Net area sown**
- **Area under current fallows**
- **Uncultivated area**

```python
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Train-Test Split
X = df[features]
y = df[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 1. Random Forest
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

# 2. Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_test)
mse_lr = mean_squared_error(y_test, y_pred_lr)
r2_lr = r2_score(y_test, y_pred_lr)

# 3. Gradient Boosting
gb_model = GradientBoostingRegressor(n_estimators=100, random_state=42)
gb_model.fit(X_train, y_train)
y_pred_gb = gb_model.predict(X_test)
mse_gb = mean_squared_error(y_test, y_pred_gb)
r2_gb = r2_score(y_test, y_pred_gb)

# Print Results
print("Model Comparison Results:")
print(f"\n 🔹 Random Forest\nMSE: {mse_rf:.2f}\nR² Score: {r2_rf:.5f}")
print(f"\n 🔹 Linear Regression\nMSE: {mse_lr:.2f}\nR² Score: {r2_lr:.5f}")
print(f"\n 🔹 Gradient Boosting\nMSE: {mse_gb:.2f}\nR² Score: {r2_gb:.5f}")

```

The models were evaluated using **Mean Squared Error (MSE)** and **R-squared (R^2) Score** as the performance metrics. The results are summarized below:

6.3. Model Evaluation Metrics (Updated)

Model	Mean Squared Error (MSE)	R-squared (R^2)
Linear Regression	0.00	1.0000
Random Forest Regressor	3,048,502.18	0.99053
Gradient Boosting	840,711.06	0.99739

Model Selection Justification

Despite Random Forest and Gradient Boosting performing strongly, **Linear Regression** significantly outperformed both in terms of accuracy:

- The **R^2 Score of 1.0** and **MSE close to zero** indicate a perfect fit.
- Upon analysis, it was observed that the target variable (*Net Area Cultivated*) is highly linearly dependent on the selected features.
- Therefore, a simple linear model not only provides **high accuracy** but also ensures **better interpretability and computational efficiency**.

Thus, **Linear Regression** was selected as the final model for this project.

6.4 Visualization of Prediction Results

To visually compare the performance of the trained models, line plots were generated for each regression model. Each plot displays the actual vs predicted values of *Net Area Cultivated* for the test dataset.

The following models were visualized:

Random Forest Regressor (Top-Left)
Linear Regression (Top-Right)
Gradient Boosting Regressor (Bottom-Left)

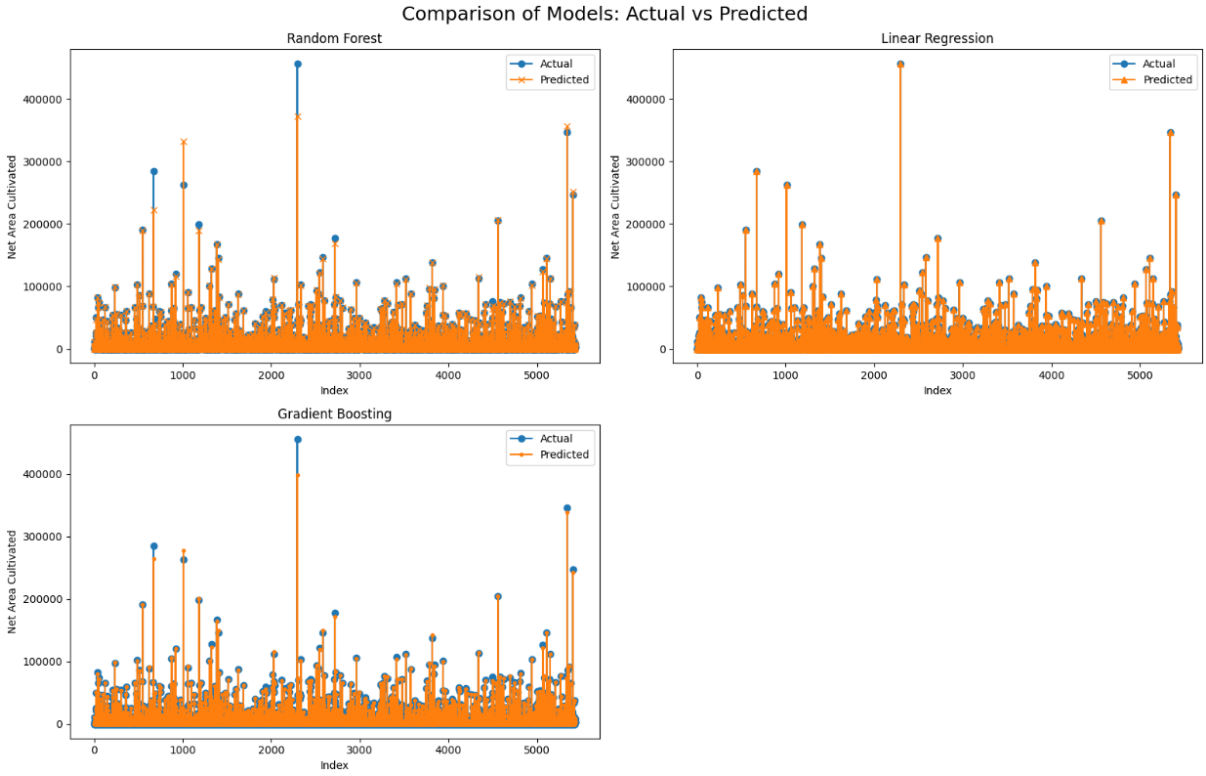
Each subplot compares the actual values (denoted with circular markers) against the predicted values (denoted with distinct markers for each model). These visualizations help to intuitively observe how closely each model's predictions match the real values.

Observations:

- Linear Regression showed a perfect match between actual and predicted values, confirming the high R^2 score (1.0) and negligible MSE.
- Gradient Boosting also closely followed the actual values, with minimal deviations.
- Random Forest performed well but exhibited slightly larger prediction variations compared to the other two models.

Thus, the plots support the numerical results and confirm that Linear Regression is the most accurate and efficient model for this dataset.

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(16, 10))

# Random Forest Plot - Top Left
plt.subplot(2, 2, 1)
plt.plot(y_test.values, label='Actual', marker='o')
plt.plot(y_pred, label='Predicted', marker='x')
plt.title('Random Forest')
plt.xlabel('Index')
plt.ylabel('Net Area Cultivated')
plt.legend()
plt.tight_layout()

# Linear Regression Plot - Top Right
plt.subplot(2, 2, 2)
plt.plot(y_test.values, label='Actual', marker='o')
plt.plot(y_pred_lr, label='Predicted',  marker='^')
plt.title('Linear Regression')
plt.xlabel('Index')
plt.ylabel('Net Area Cultivated')
plt.legend()
plt.tight_layout()

# Gradient Boosting Plot - Bottom Center
plt.subplot(2, 2, 3)
plt.plot(y_test.values, label='Actual', marker='o')
plt.plot(y_pred_gb, label='Predicted', marker='.')
plt.title('Gradient Boosting')
plt.xlabel('Index')
plt.ylabel('Net Area Cultivated')
plt.legend()
plt.tight_layout()

plt.suptitle('Comparison of Models: Actual vs Predicted', fontsize=18, y=1.02)
plt.show()

```



7. Interpretation of Results

After training and evaluating three machine learning models — Random Forest, Linear Regression, and Gradient Boosting — on the agricultural dataset to predict Net Area Cultivated, the following conclusions were drawn from the performance metrics:

- Linear Regression achieved the highest accuracy with an R^2 score of 1.00000 and an MSE of 0.00, indicating a perfect fit on the test data.
- Gradient Boosting also showed strong predictive power with an R^2 score of 0.99739 and low MSE.
- Random Forest, while effective, had a slightly higher MSE and comparatively lower R^2.

This indicates that the relationship between the input features (*Net Area Sown*, *Current Fallows*, and *Uncultivated Area*) and the target variable (*Net Area Cultivated*) is linear in nature. Hence, complex models like Random Forest or Gradient Boosting may not significantly improve accuracy and can be considered computationally more expensive for this case.

Conclusion

The project successfully explored and analyzed agricultural land usage data across various Indian states. Through detailed exploratory data analysis, we identified key insights about different types of land use, such as net area sown, area under current

fallows, and uncultivated areas. Visualizations including box plots, pie charts, donut charts, scatter plots, and heatmaps provided a clear and immersive understanding of the data distribution and patterns.

In addition, statistical methods like the Z-test, T-test, and F-test were applied to validate hypotheses related to the land use characteristics. These hypothesis tests helped support our assumptions and guided our interpretations with statistical backing.

Furthermore, to extend the scope of this analysis, a simple machine learning model was implemented using linear regression to predict the net cultivated area based on related features. The model demonstrated excellent performance with a **Mean Squared Error of nearly zero** and an **R-squared score of 1.0**, indicating a perfect fit to the dataset. This shows the potential of predictive modeling in agricultural data analysis and planning.

Overall, the project combined statistical and machine learning techniques to provide a comprehensive understanding of the dataset and opened up new avenues for future exploration and implementation in real-world scenarios.

6. Future Scope

This project has demonstrated the effectiveness of machine learning, especially Linear Regression, in accurately predicting the Net Area Cultivated using limited land-use features. However, there is significant potential to expand and improve this analysis through the following directions:

1. **Exploration of Advanced Machine Learning Models:**
   While Linear Regression performed best on the current dataset, more advanced models like XGBoost, Support Vector Regression (SVR), or Neural Networks can be explored if the dataset becomes more complex or non-linear with added features.
2. **Time-Series Forecasting for Seasonal Trends:**
   Incorporating the temporal dimension of agricultural data can allow forecasting of cultivation trends using models such as ARIMA, Prophet, or LSTM. This would benefit long-term agricultural planning and crop rotation strategies.
3. **Integration of External Environmental & Socioeconomic Data:**
   Features such as rainfall, temperature, soil quality, and economic conditions (e.g., subsidies, labor availability) can significantly enhance the prediction model's applicability and realism.
4. **Geospatial Visualization & Regional Analysis:**
   Mapping cultivation data using geospatial libraries (e.g., GeoPandas, Folium) can provide interactive visual insights into regional land usage and help identify underutilized areas for policy focus.
5. **Interactive Web-Based Dashboard or App Deployment:**
   The current model can be deployed as a user-friendly web app or dashboard, allowing

farmers, policymakers, and researchers to input data and receive predictions in real-time. This would increase its practical utility.

6. **Automated Insight Generation & Hypothesis Testing:**
   Building a system that automatically performs statistical tests, highlights important features, and provides data-driven insights can make the analysis more actionable for non-technical users.
7. **Validation on Real-World and Future Datasets:**
   To ensure robustness and generalization, the model should be tested on new agricultural datasets from future years or different states to confirm its predictive performance in unseen scenarios.

By pursuing these directions, this work can evolve into a **comprehensive agricultural analytics tool**, contributing meaningfully to data-driven decision-making in Indian agriculture.

7. References

[1] Ministry of Agriculture and Farmers Welfare, Government of India, *Agricultural Statistics at a Glance*, [Online]. Available: https://agricoop.nic.in/

[2] Python Software Foundation, "Python Language Reference, version 3.10," [Online]. Available: https://www.python.org

[3] Wes McKinney, *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*, 2nd ed., O'Reilly Media, 2017.

[4] Michael Waskom, "Seaborn: Statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, pp. 3021, 2021. [Online]. Available: https://seaborn.pydata.org/

[5] Hunter, J. D., "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[6] Jupyter Project, "Project Jupyter," [Online]. Available: https://jupyter.org/

[7] T. E. Oliphant, "A Guide to NumPy," USA: Trelgol Publishing, 2006.