Rice Crop Yield Prediction

The goal of this project is to find insights on predicting crop yields of rice in India. Smallholder farmers, who make up most of the workforce there, face challenges like poverty and malnutrition. Predictive insights can help them make better decisions about resource use, and help in supporting food security.

The **dataset** can be found on Kaggle

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, MinMaxScaler, OneHotEncoder,
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, roc_curve, auc, classification_r
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn import tree
from sklearn.model_selection import cross_val_score, cross_validate
import imblearn
from imblearn.over_sampling import SMOTE, SMOTENC
from imblearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.compose import ColumnTransformer, make_column_selector
from sklearn.impute import SimpleImputer
from sklearn.model_selection import GridSearchCV
from sklearn.tree import export_text, plot_tree
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.preprocessing import FunctionTransformer
from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.compose import TransformedTargetRegressor
from sklearn.feature_selection import VarianceThreshold
from lime.lime_tabular import LimeTabularExplainer
```

Let's load and examine the data

In [2]:
```
# to show all columns
pd.set_option('display.max_columns', None)
```

```python
# to revert: pd.reset_option('display.max_columns')
```

In [3]:
```python
df = pd.read_csv("data/Train.csv", index_col=0)
df.head(12)
```

Out[3]:

	District	Block	CultLand	CropCultLand	LandPreparationMet
ID					
ID_GTFAC7PEVWQ9	Nalanda	Noorsarai	45	40	TractorPlo FourWheelTracRotav
ID_TK40ARLSPOKS	Nalanda	Rajgir	26	26	WetTillagePudc TractorPlo FourWheelTra
ID_1FJY2CRIMLZZ	Gaya	Gurua	10	10	TractorPlo FourWheelTracRotav
ID_I3IPXS4DB7NE	Gaya	Gurua	15	15	TractorPlo FourWheelTracRotav
ID_4T8YQWXWHB4A	Nalanda	Noorsarai	60	60	TractorPlo WetTillagePudc
ID_W5MM9H353RL9	Vaishali	Garoul	10	5	TractorPlo
ID_6O44Z25H1JAV	Jamui	Khaira	12	12	TractorPlo
ID_VRI9LEL2W3DR	Nalanda	Rajgir	80	80	FourWheelTracRotav
ID_6YA9Y09O55LE	Jamui	Khaira	25	25	TractorPlo
ID_EDA8RK1CP60K	Nalanda	Noorsarai	20	10	WetTillagePudc

ID_92OQSAHCN51N	Jamui	Khaira	25	14	TractorPlo
ID_LPERFIRDG4R1	Nalanda	Noorsarai	30	30	TractorPlo

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3870 entries, ID_GTFAC7PEVWQ9 to ID_KEPOQDTCZC6S
Data columns (total 43 columns):
 #   Column                              Non-Null Count  Dtype  
---  ------                              --------------  -----  
 0   District                            3870 non-null   object 
 1   Block                               3870 non-null   object 
 2   CultLand                            3870 non-null   int64  
 3   CropCultLand                        3870 non-null   int64  
 4   LandPreparationMethod               3870 non-null   object 
 5   CropTillageDate                     3870 non-null   object 
 6   CropTillageDepth                    3870 non-null   int64  
 7   CropEstMethod                       3870 non-null   object 
 8   RcNursEstDate                       3787 non-null   object 
 9   SeedingSowingTransplanting          3870 non-null   object 
 10  SeedlingsPerPit                     3581 non-null   float64
 11  NursDetFactor                       3581 non-null   object 
 12  TransDetFactor                      3581 non-null   object 
 13  TransplantingIrrigationHours        3677 non-null   float64
 14  TransplantingIrrigationSource       3755 non-null   object 
 15  TransplantingIrrigationPowerSource  3367 non-null   object 
 16  TransIrriCost                       2988 non-null   float64
 17  StandingWater                       3632 non-null   float64
 18  OrgFertilizers                      2535 non-null   object 
 19  Ganaura                             1453 non-null   float64
 20  CropOrgFYM                          1196 non-null   float64
 21  PCropSolidOrgFertAppMethod          2533 non-null   object 
 22  NoFertilizerAppln                   3870 non-null   int64  
 23  CropbasalFerts                      3682 non-null   object 
 24  BasalDAP                            3327 non-null   float64
 25  BasalUrea                           2166 non-null   float64
 26  MineralFertAppMethod                3870 non-null   object 
 27  FirstTopDressFert                   3385 non-null   object 
 28  1tdUrea                             3314 non-null   float64
 29  1appDaysUrea                        3314 non-null   float64
 30  2tdUrea                             1176 non-null   float64
 31  2appDaysUrea                        1170 non-null   float64
 32  MineralFertAppMethod.1              3389 non-null   object 
 33  Harv_method                         3870 non-null   object 
 34  Harv_date                           3870 non-null   object 
 35  Harv_hand_rent                      3618 non-null   float64
 36  Threshing_date                      3870 non-null   object 
 37  Threshing_method                    3870 non-null   object 
 38  Residue_length                      3870 non-null   int64  
 39  Residue_perc                        3870 non-null   int64  
 40  Stubble_use                         3870 non-null   object 
 41  Acre                                3870 non-null   float64
 42  Yield                               3870 non-null   int64  
dtypes: float64(14), int64(7), object(22)
memory usage: 1.3+ MB
```

In [5]:
```
df.describe()
```

Out[5]:

	CultLand	CropCultLand	CropTillageDepth	SeedlingsPerPit	TransplantingIr
count	3870.000000	3870.000000	3870.000000	3581.000000	
mean	28.527907	24.727132	4.488372	2.706507	
std	30.454218	27.994802	1.133044	7.624397	
min	1.000000	1.000000	1.000000	1.000000	
25%	12.000000	10.000000	4.000000	2.000000	
50%	20.000000	20.000000	4.000000	2.000000	
75%	35.000000	30.000000	5.000000	3.000000	
max	800.000000	800.000000	8.000000	442.000000	

Get the descriptive statistics

In [6]:
```
desc = df.describe()
# Calculate the 95th percentile for numeric columns only
percentile_95 = df.select_dtypes(include=[float, int]).quantile(0.99)
# Append the 95th percentile to the descriptive statistics
desc.loc['99%'] = percentile_95
desc
```

Out[6]:

	CultLand	CropCultLand	CropTillageDepth	SeedlingsPerPit	TransplantingIr
count	3870.000000	3870.000000	3870.000000	3581.000000	
mean	28.527907	24.727132	4.488372	2.706507	
std	30.454218	27.994802	1.133044	7.624397	
min	1.000000	1.000000	1.000000	1.000000	
25%	12.000000	10.000000	4.000000	2.000000	
50%	20.000000	20.000000	4.000000	2.000000	
75%	35.000000	30.000000	5.000000	3.000000	
max	800.000000	800.000000	8.000000	442.000000	
99%	120.000000	100.000000	8.000000	12.000000	

We can see there are outliers for most numeric features.

In [7]:
```
# Check for duplicate rows
df.duplicated().sum()
```

Out[7]:
```
0
```

In [8]:
```
# Count missing values in each column
df.isnull().sum()
```

Out[8]:
```
District                                0
Block                                   0
CultLand                                0
CropCultLand                            0
LandPreparationMethod                   0
CropTillageDate                         0
CropTillageDepth                        0
CropEstMethod                           0
RcNursEstDate                          83
SeedingSowingTransplanting              0
SeedlingsPerPit                       289
NursDetFactor                         289
TransDetFactor                        289
TransplantingIrrigationHours          193
TransplantingIrrigationSource         115
TransplantingIrrigationPowerSource    503
TransIrriCost                         882
StandingWater                         238
OrgFertilizers                       1335
Ganaura                              2417
CropOrgFYM                           2674
PCropSolidOrgFertAppMethod           1337
NoFertilizerAppln                       0
CropbasalFerts                        188
BasalDAP                              543
BasalUrea                            1704
MineralFertAppMethod                    0
FirstTopDressFert                     485
1tdUrea                               556
1appDaysUrea                          556
2tdUrea                              2694
2appDaysUrea                         2700
MineralFertAppMethod.1                481
Harv_method                             0
Harv_date                               0
Harv_hand_rent                        252
Threshing_date                          0
Threshing_method                        0
Residue_length                          0
Residue_perc                            0
Stubble_use                             0
Acre                                    0
Yield                                   0
dtype: int64
```

In [9]:
```
# Calculate the percentage of missing values in each column
(df.isnull().sum() * 100 / len(df)).round(2)
```

Out[9]:
```
District                             0.00
Block                                0.00
CultLand                             0.00
CropCultLand                         0.00
LandPreparationMethod                0.00
CropTillageDate                      0.00
CropTillageDepth                     0.00
CropEstMethod                        0.00
RcNursEstDate                        2.14
SeedingSowingTransplanting           0.00
SeedlingsPerPit                      7.47
NursDetFactor                        7.47
TransDetFactor                       7.47
TransplantingIrrigationHours         4.99
TransplantingIrrigationSource        2.97
TransplantingIrrigationPowerSource  13.00
TransIrriCost                       22.79
StandingWater                        6.15
OrgFertilizers                      34.50
Ganaura                             62.45
CropOrgFYM                          69.10
PCropSolidOrgFertAppMethod          34.55
NoFertilizerAppln                    0.00
CropbasalFerts                       4.86
BasalDAP                            14.03
BasalUrea                           44.03
MineralFertAppMethod                 0.00
FirstTopDressFert                   12.53
1tdUrea                             14.37
1appDaysUrea                        14.37
2tdUrea                             69.61
2appDaysUrea                        69.77
MineralFertAppMethod.1              12.43
Harv_method                          0.00
Harv_date                            0.00
Harv_hand_rent                       6.51
Threshing_date                       0.00
Threshing_method                     0.00
Residue_length                       0.00
Residue_perc                         0.00
Stubble_use                          0.00
Acre                                 0.00
Yield                                0.00
dtype: float64
```

In [10]:
```
df['NoFertilizerAppln'].value_counts()
```

Out[10]:
```
NoFertilizerAppln
2    2201
3    1181
1     481
4       7
Name: count, dtype: int64
```

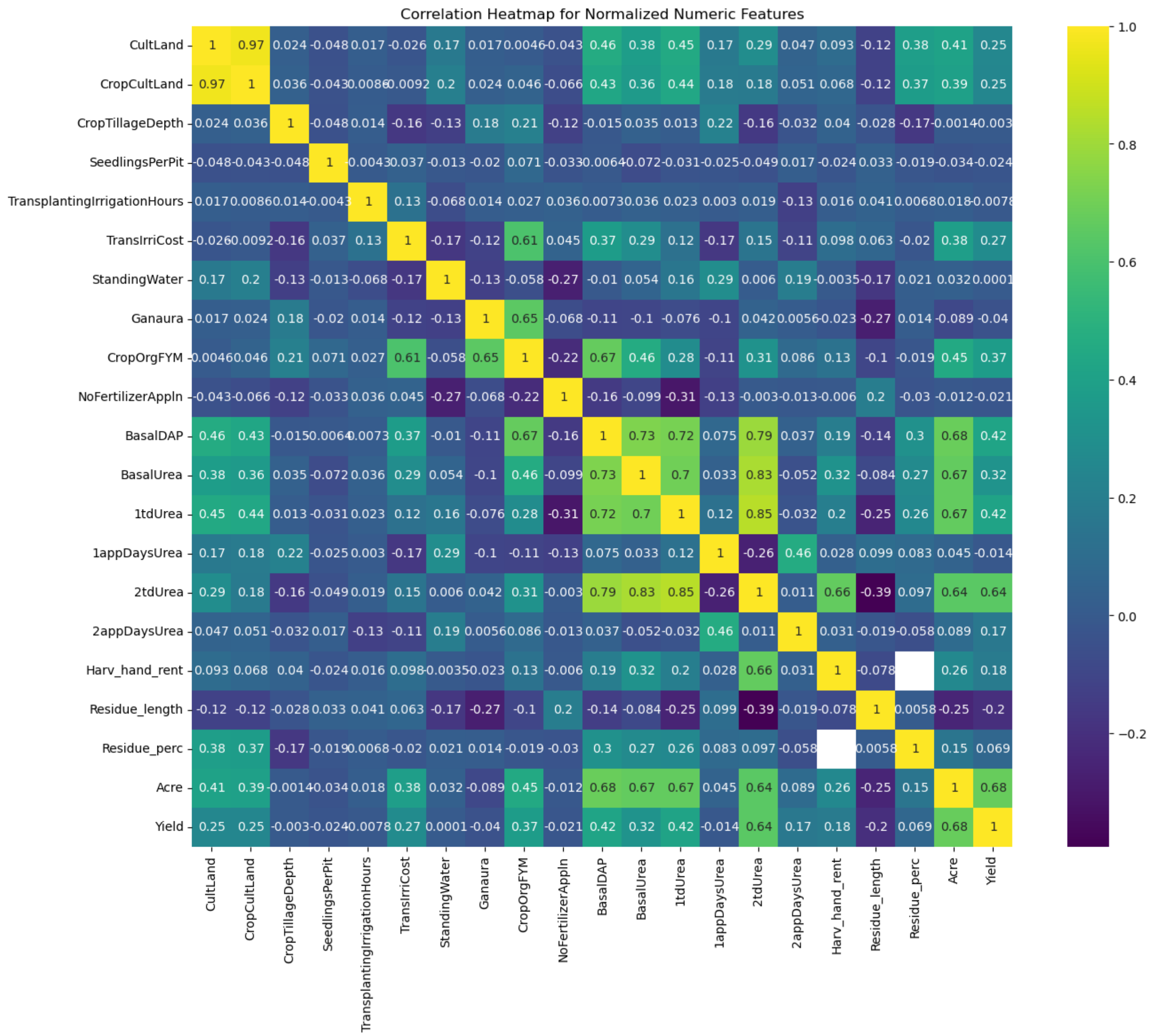
In [11]:
```
df['Harv_method'].value_counts()
```

Out[11]:
```
Harv_method
hand       3642
machine     228
Name: count, dtype: int64
```

Visualize Numerical Variables

Correlation Heatmap for Numeric Columns

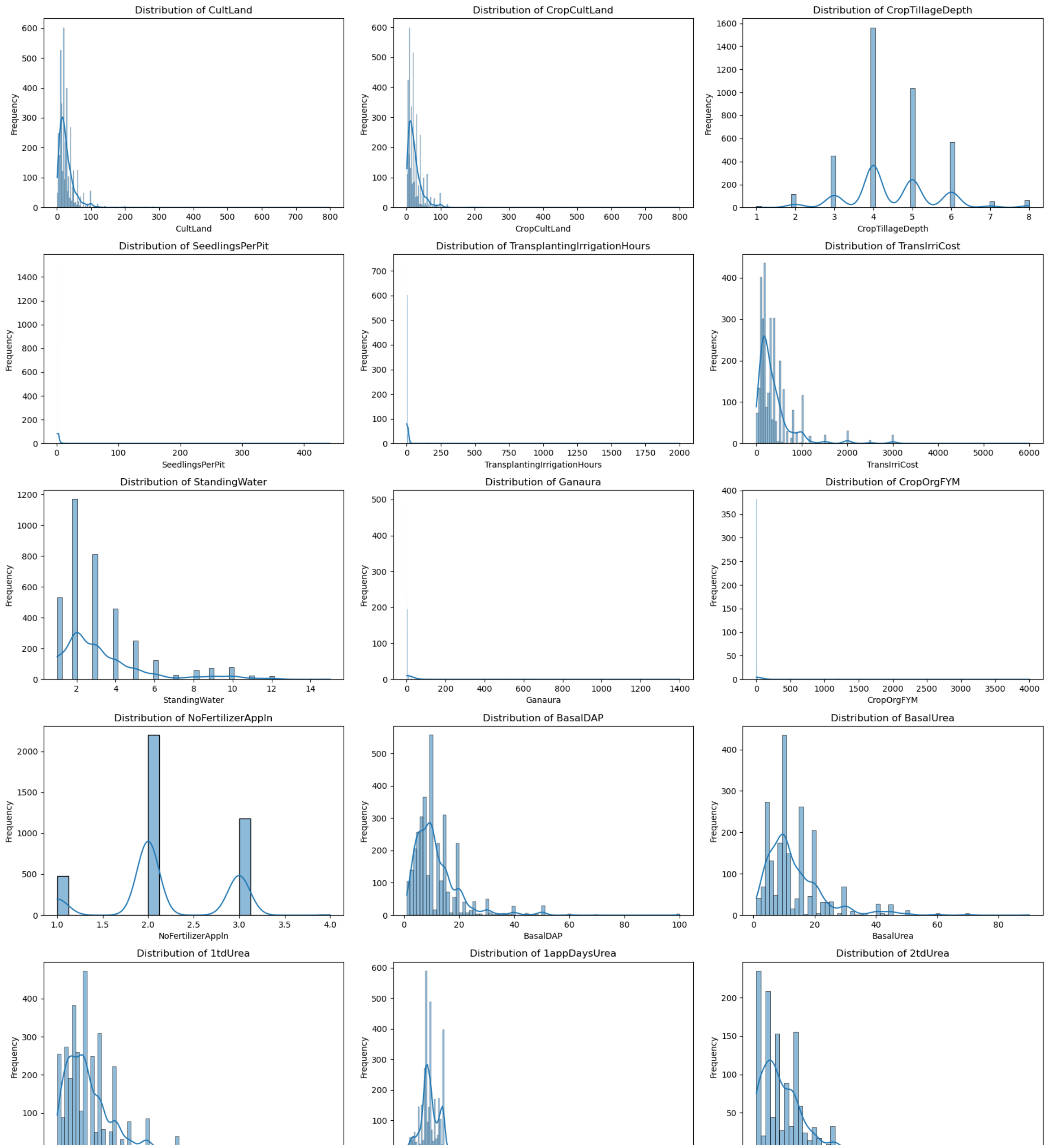
In [12]:
```
# Select and normalize only the numeric columns
numeric_data = df.select_dtypes(include=['float64', 'int64'])
scaler = MinMaxScaler()
normalized_data = scaler.fit_transform(numeric_data)
normalized_df = pd.DataFrame(normalized_data, columns=numeric_data.columns)

# Compute the correlation matrix on normalized data
correlation_matrix = normalized_df.corr()

# Plot the correlation heatmap
plt.figure(figsize=(16, 12))
sns.heatmap(correlation_matrix, annot=True, cmap='viridis', square=True)
plt.title("Correlation Heatmap for Normalized Numeric Features")
plt.show()
```

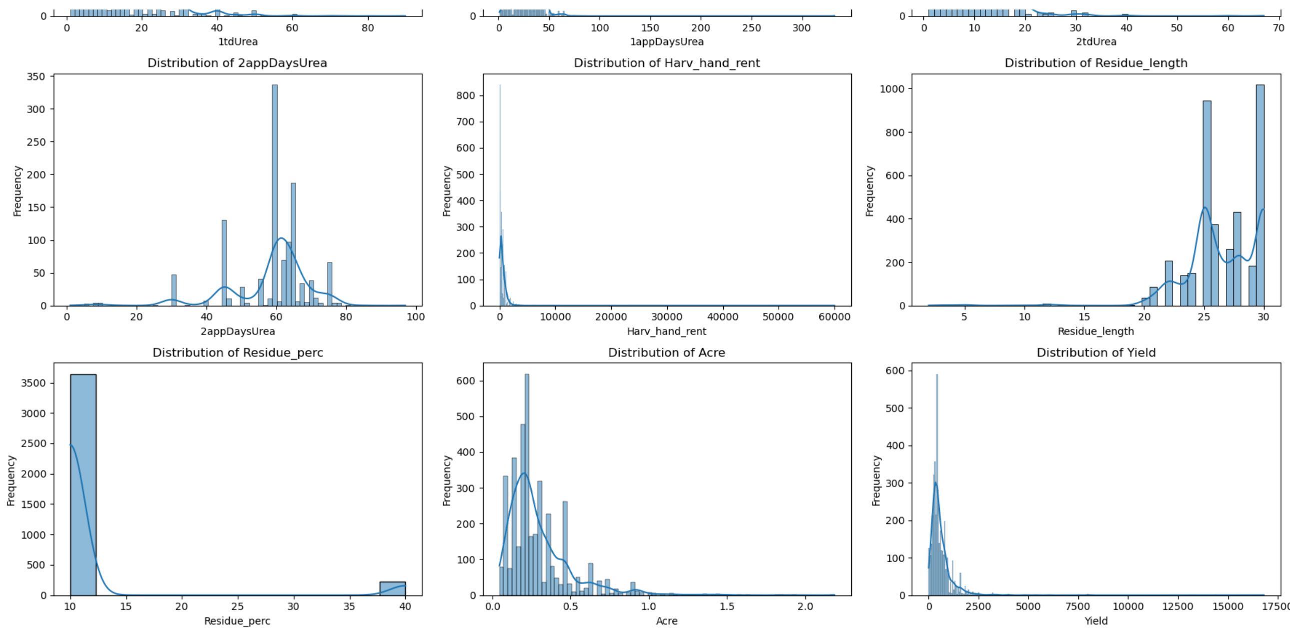


There are no significant correlations between features except 'CultLand' and 'CropCultLand'.

In [13]:
```
# Select numerical columns
numerical_columns = df.select_dtypes(include=['float64', 'int64']).columns

# Set up rows of 3 plots each
num_plots = len(numerical_columns)
num_rows = (num_plots // 3) + (num_plots % 3 > 0)

fig, axes = plt.subplots(num_rows, 3, figsize=(18, num_rows * 4))
axes = axes.flatten()

# Create histograms for each numerical column
for i, col in enumerate(numerical_columns):
```

```
    sns.histplot(df[col].dropna(), kde=True, ax=axes[i])
    axes[i].set_title(f'Distribution of {col}')
    axes[i].set_xlabel(col)
    axes[i].set_ylabel('Frequency')

# Hide any extra subplots
for j in range(i + 1, num_rows * 3):
    axes[j].set_visible(False)

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





In [14]:
```
numerical_columns
```

Out[14]:
```
Index(['CultLand', 'CropCultLand', 'CropTillageDepth', 'SeedlingsPerPit',
       'TransplantingIrrigationHours', 'TransIrriCost', 'StandingWater',
       'Ganaura', 'CropOrgFYM', 'NoFertilizerAppln', 'BasalDAP', 'BasalUre
a',
       '1tdUrea', '1appDaysUrea', '2tdUrea', '2appDaysUrea', 'Harv_hand_ren
t',
       'Residue_length', 'Residue_perc', 'Acre', 'Yield'],
      dtype='object')
```

Let's look closer at some of the features

In [15]:
```
# Calculate the 95th percentile for the 'Ganaura' column
q90 = df['Ganaura'].quantile(0.95)
print("95th Percentile (Upper Limit):", q90)
```

```
95th Percentile (Upper Limit): 80.0
```

In [16]:
```
# Ganaura, CropOrgFYM
df['CropOrgFYM'].quantile(0.95)
```

Out[16]:
```
22.75
```

In [17]:
```
# Calculate the 95th percentile
q95 = df['1appDaysUrea'].quantile(0.95)
print("95th Percentile (Upper Limit):", q95)
```

```
95th Percentile (Upper Limit): 45.0
```

In [18]:
```
q95 = df['TransplantingIrrigationHours'].quantile(0.95)
```

```
print("95th Percentile (Upper Limit):", q95)
```

```
95th Percentile (Upper Limit): 15.0
```

In [19]:
```
# Checking correlation between the two columns
correlation = df[['CropCultLand', 'Acre']].corr()
print(correlation)
```

```
              CropCultLand     Acre
CropCultLand       1.00000  0.39407
Acre               0.39407  1.00000
```

In [20]:
```
# Checking correlation between the two columns
correlation = df[['CultLand', 'Acre']].corr()
print(correlation)
```

```
          CultLand      Acre
CultLand  1.000000  0.409604
Acre      0.409604  1.000000
```

In [21]:
```
df['Yield'].quantile(0.90)
```

Out[21]:
```
1200.0
```

In [22]:
```
df['Harv_hand_rent'].value_counts().sort_index()
```

Out[22]:
```
Harv_hand_rent
1.0         1
2.0         2
3.0        11
4.0         4
5.0         7
           ..
6137.0      1
7221.0      1
7931.0      1
9300.0      2
60000.0     1
Name: count, Length: 131, dtype: int64
```

Let's inspect the number of fertilizer applications and their kind

In [23]:
```
df['NoFertilizerAppln'].value_counts()
```

Out[23]:
```
NoFertilizerAppln
2    2201
3    1181
1     481
4       7
Name: count, dtype: int64
```

In [24]:
```
df[df['NoFertilizerAppln'] != 1][['FirstTopDressFert', '1tdUrea', '1appDaysU
```

Out[24]:

ID	FirstTopDressFert	1tdUrea	1appDaysUrea
ID_GTFAC7PEVWQ9	Urea	15.0	18.0
ID_TK40ARLSPOKS	Urea	20.0	39.0
ID_1FJY2CRIMLZZ	Urea	5.0	65.0
ID_I3IPXS4DB7NE	Urea	5.0	5.0
ID_4T8YQWXWHB4A	Urea	30.0	26.0
...	...	...	...
ID_DU6AHQ06QMXV	Urea	9.0	23.0
ID_PW2LN7ACB8MM	Urea DAP	2.0	30.0
ID_7ZZQ6R4XB4FK	Urea	12.0	45.0
ID_PVVDF6LK6FO8	Urea	6.0	45.0
ID_KEPOQDTCZC6S	Urea	10.0	28.0

3389 rows × 3 columns

In [25]:
```
# Filter rows where 'FirstTopDressFert' does not contain "Urea" and 'NoFerti
filtered_df = df[(~df['FirstTopDressFert'].str.contains("Urea", na=False)) &

# Display the filtered result
print(filtered_df)
```

```
                  FirstTopDressFert  1tdUrea  1appDaysUrea
ID
ID_GRREKUJLG8N5                 DAP      NaN           NaN
ID_RQ2X90R8F30U                 DAP      NaN           NaN
ID_6VW9ED51TTXA                 DAP      NaN           NaN
ID_6SNCEIE9GIKJ                 DAP      NaN           NaN
ID_W53ULSZ3UZJK                 DAP      NaN           NaN
...                             ...      ...           ...
ID_3KR27BO0615L            DAP NPKS      NaN           NaN
ID_USJIQ3L9ZQLD                 DAP      NaN           NaN
ID_9II4YBSXYK4Q                 DAP      NaN           NaN
ID_2KJP9DE2ZBIY                 DAP      NaN           NaN
ID_8I7QB5U74AUJ                 DAP      NaN           NaN

[75 rows x 3 columns]
```

In [26]:
```
df[df['NoFertilizerAppln']==3][['2tdUrea', '2appDaysUrea']]
```

Out[26]:

	2tdUrea	2appDaysUrea
ID		
ID_6O44Z25H1JAV	6.0	67.0
ID_6YA9Y09O55LE	7.0	58.0
ID_92OQSAHCN51N	12.0	65.0
ID_SJYVZSXJCX8S	1.0	60.0
ID_O99ZE30OJQ0E	1.0	60.0
...	...	...
ID_OQG31JUGU5JL	7.0	65.0
ID_RSC7O6MY665W	10.0	45.0
ID_DU6AHQ06QMXV	9.0	55.0
ID_PW2LN7ACB8MM	1.0	60.0
ID_PVVDF6LK6FO8	6.0	6.0

1181 rows × 2 columns

In [27]:
```
df[(df['NoFertilizerAppln'].isin([3, 4])) & (df['2tdUrea'].isna() | df['2app
```

Out[27]:

ID	2tdUrea	2appDaysUrea
ID_X4024AVMS14D	6.0	NaN
ID_N1ELLJ1F32VU	NaN	NaN
ID_JI5NFWHR5G3H	4.0	NaN
ID_90IBO92HGZYD	5.0	NaN
ID_T5LYSK1D2TZX	NaN	NaN
ID_K9MMVQBHNSWY	NaN	NaN
ID_AXUCB1MQPA7Q	NaN	NaN
ID_Q5NRUPWTA5NX	3.0	NaN
ID_V7KHHGC42620	NaN	NaN
ID_4BUF8KA1KQN5	NaN	NaN
ID_S6XXFVWMWIEC	NaN	NaN
ID_FTCLUK815GGN	6.0	NaN
ID_LVFLM1AU48RZ	NaN	NaN
ID_6E1ZCZXS0M7H	6.0	NaN
ID_7W8Z7KFB21B8	NaN	NaN
ID_I35G50DPKNCY	NaN	NaN
ID_05LJ8YMX1TWL	NaN	NaN
ID_HZ9W937YJ3RI	NaN	NaN

In [28]:
```
df[(df['2tdUrea'].isna() | df['2appDaysUrea'].isna())][['2tdUrea', '2appDays
```

Out[28]:

	2tdUrea	2appDaysUrea
ID		
ID_GTFAC7PEVWQ9	NaN	NaN
ID_TK40ARLSPOKS	NaN	NaN
ID_1FJY2CRIMLZZ	NaN	NaN
ID_I3IPXS4DB7NE	NaN	NaN
ID_4T8YQWXWHB4A	NaN	NaN
...	...	...
ID_NOASMX3TXXY9	NaN	NaN
ID_7ZZQ6R4XB4FK	NaN	NaN
ID_RBYVUPRATVMW	NaN	NaN
ID_ARE9QWENJNJ2	NaN	NaN
ID_KEPOQDTCZC6S	NaN	NaN

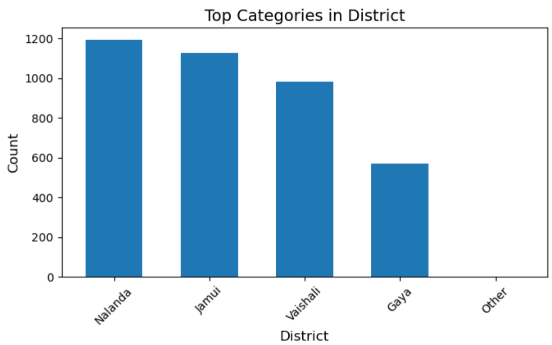
2700 rows × 2 columns

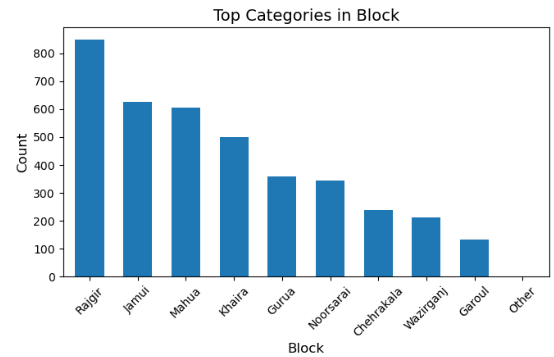
Visualize Categorical Variables

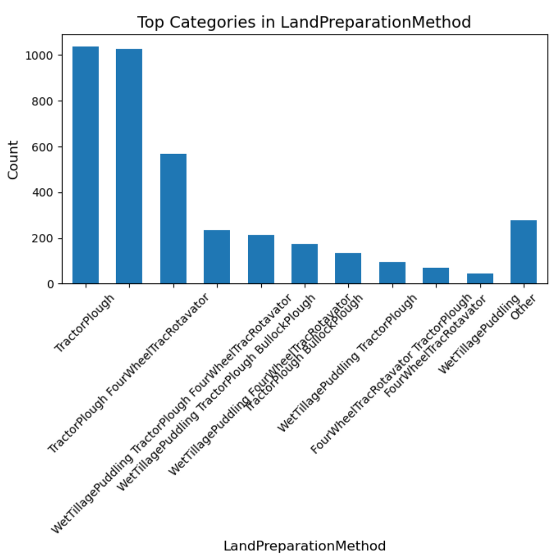
In [29]:

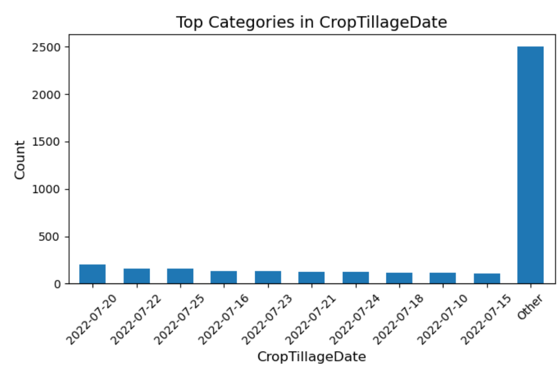
```
# Define the number of top categories to display (including "Other")
top_n = 10
categorical_columns = df.select_dtypes(include=['object']).columns

# Calculate rows and columns for the subplots grid
num_plots = len(categorical_columns)
num_rows = (num_plots // 2) + (num_plots % 2 > 0)  # Two plots per row
fig, axes = plt.subplots(num_rows, 2, figsize=(16, num_rows * 8))  # Adjust
axes = axes.flatten()

# Plot top categories with an "Other" category as the 11th bar
for i, col in enumerate(categorical_columns):
    # Get the top N categories
    top_categories = df[col].value_counts().nlargest(top_n)

    # Calculate "Other" as the sum of all categories outside the top N
    other_count = df[col].value_counts()[top_n:].sum()
    top_categories['Other'] = other_count  # Add "Other" category as the 11t

    # Plot
    top_categories.plot(kind='bar', ax=axes[i], width=0.6)  # Adjust width f
```

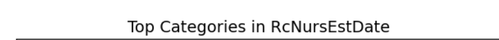
```python
    axes[i].set_title(f'Top Categories in {col}', fontsize=14)
    axes[i].set_xlabel(col, fontsize=12)
    axes[i].set_ylabel('Count', fontsize=12)
    axes[i].tick_params(axis='x', rotation=45)

# Hide any extra subplots if fewer than 2 plots in the last row
for j in range(i + 1, len(axes)):
    axes[j].set_visible(False)

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

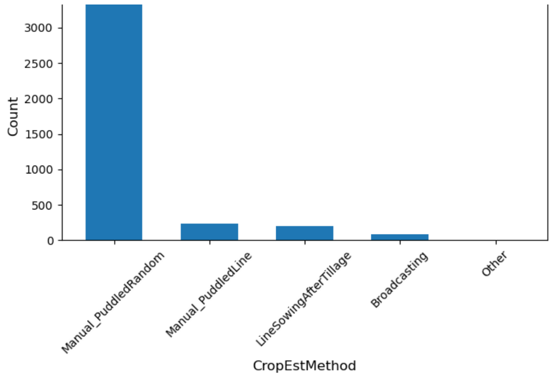


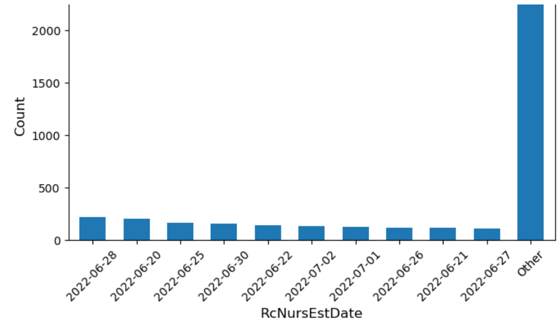




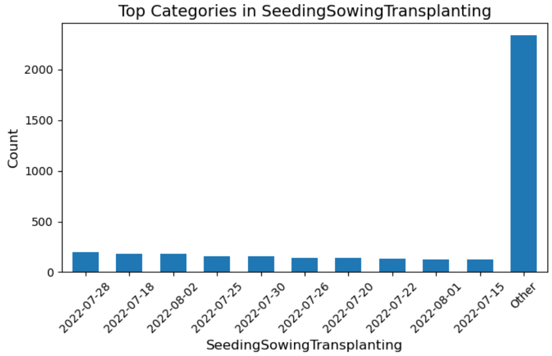




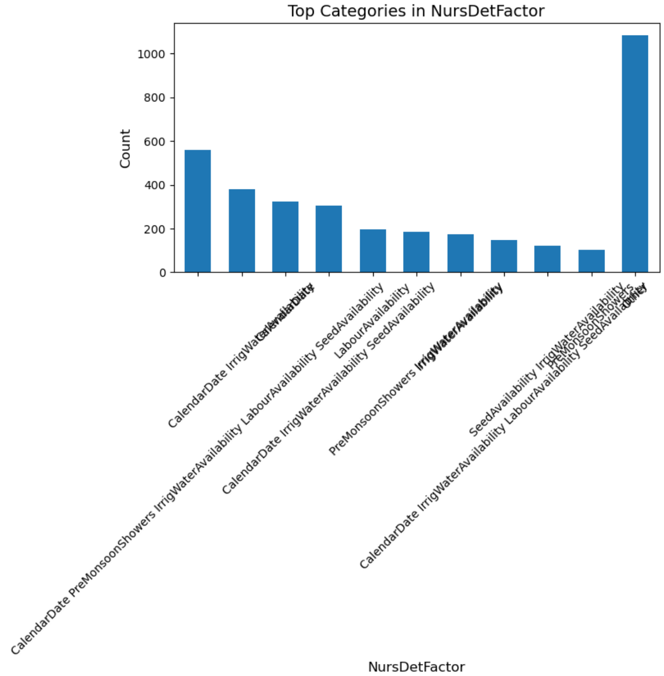




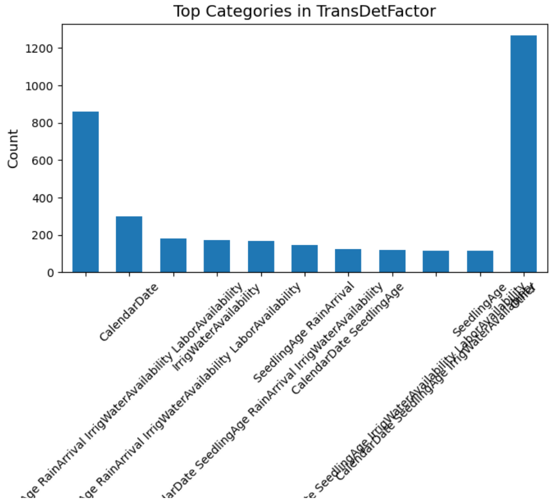
Top Categories in SeedingSowingTransplanting



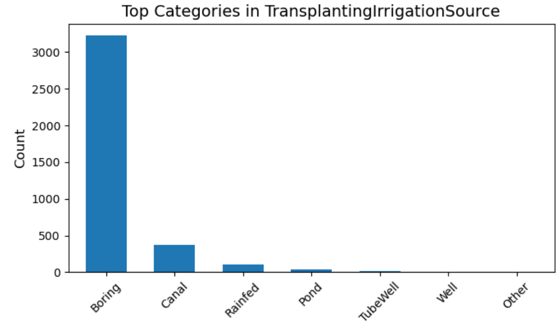
Top Categories in NursDetFactor

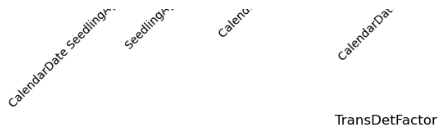


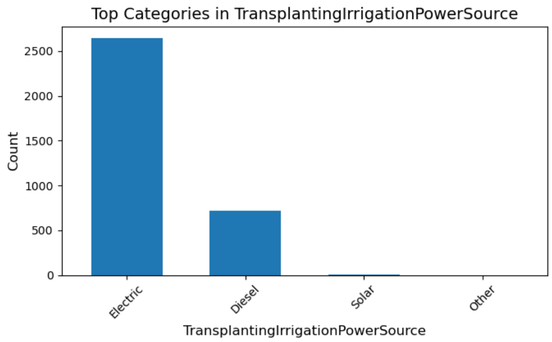
Top Categories in TransDetFactor

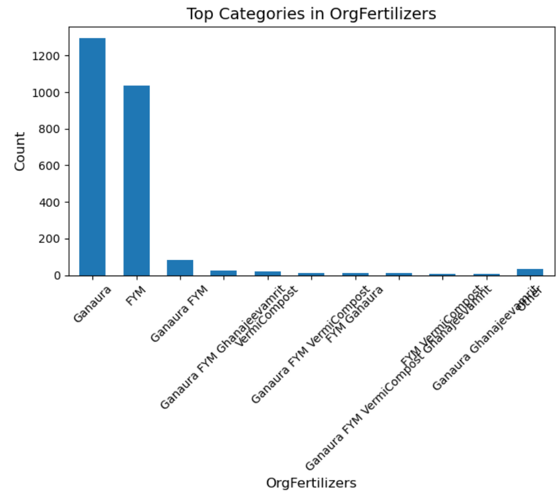


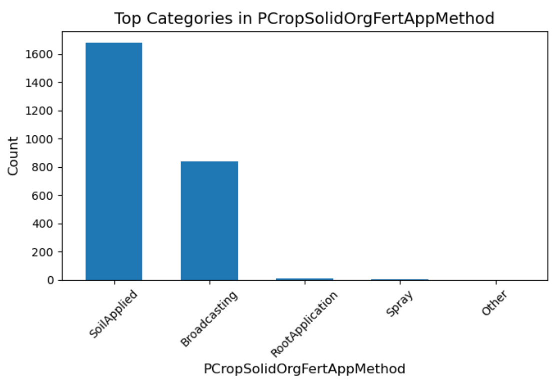
Top Categories in TransplantingIrrigationSource

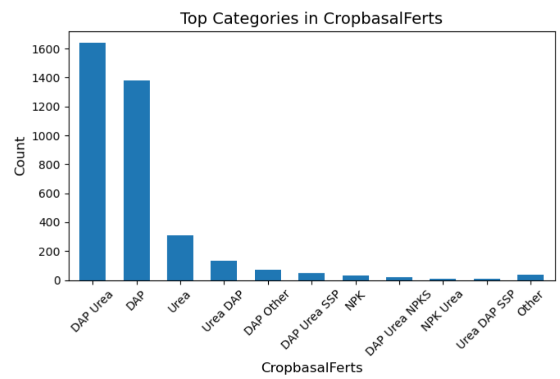


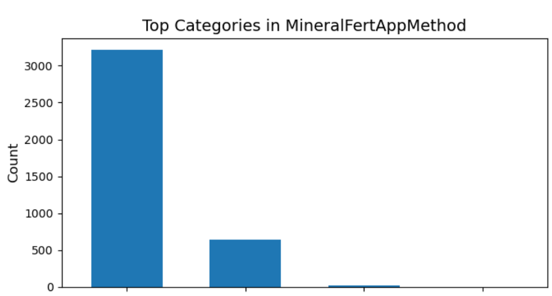

CalendarDate SeedlingA
SeedlingA
Calend
CalendarDat
TransDetFactor

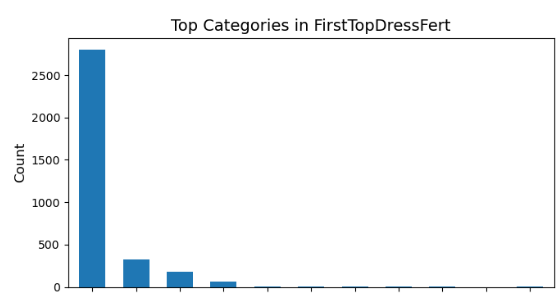

Top Categories in TransplantingIrrigationPowerSource
Count
0
500
1000
1500
2000
2500
Electric
Diesel
Solar
Other
TransplantingIrrigationPowerSource


Top Categories in OrgFertilizers
Count
0
200
400
600
800
1000
1200
Ganaura
FYM
Ganaura FYM
Ganaura FYM Ghanajeevamrit
VermiCompost
Ganaura FYM VermiCompost
FYM Ganaura
Ganaura FYM VermiCompost Ghanajeevamrit
FYM VermiCompost
Ganaura Ghanajeevamrit
Other
OrgFertilizers


Top Categories in PCropSolidOrgFertAppMethod
Count
0
200
400
600
800
1000
1200
1400
1600
SoilApplied
Broadcasting
RootApplication
Spray
Other
PCropSolidOrgFertAppMethod

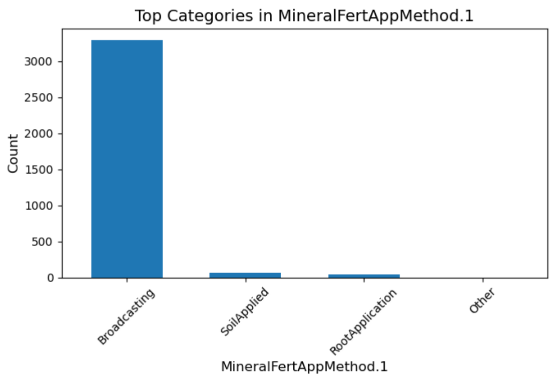

Top Categories in CropbasalFerts
Count
0
200
400
600
800
1000
1200
1400
1600
DAP Urea
DAP
Urea
Urea DAP
DAP Other
DAP Urea SSP
NPK
DAP Urea NPKS
NPK Urea
Urea DAP SSP
Other
CropbasalFerts

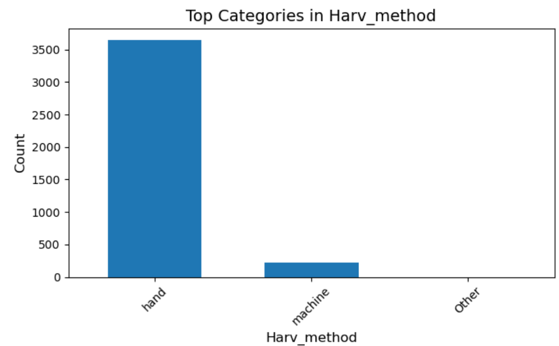

Top Categories in MineralFertAppMethod
Count
0
500
1000
1500
2000
2500
3000

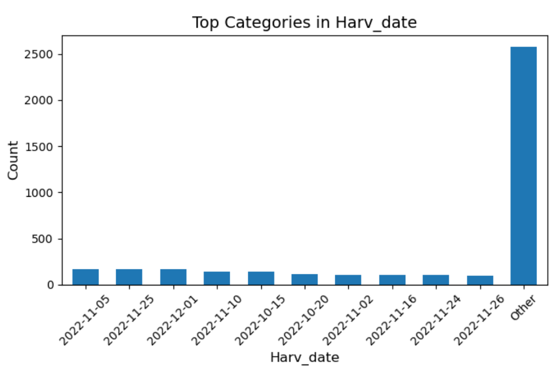

Top Categories in FirstTopDressFert
Count
0
500
1000
1500
2000
2500

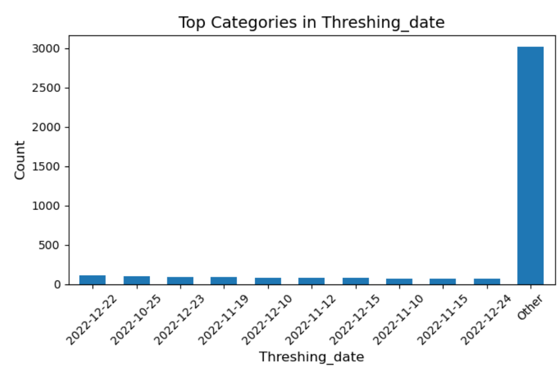

Broadcasting
SoilApplied
RootApplication
Other
MineralFertAppMethod


Urea
DAP Urea
Urea DAP
DAP
NPKS Urea
Urea NPKS
NPK
DAP NPKS
NPK Urea
DAP NPK Urea
Other
FirstTopDressFert

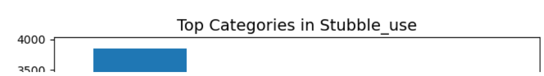

Top Categories in MineralFertAppMethod.1
Count
0
500
1000
1500
2000
2500
3000
Broadcasting
SoilApplied
RootApplication
Other
MineralFertAppMethod.1

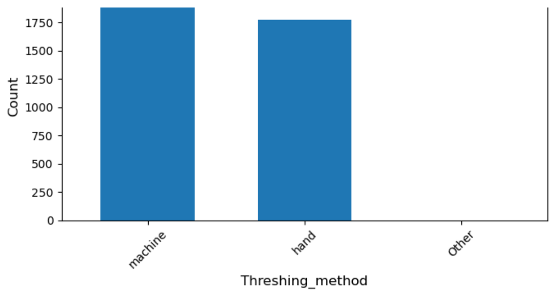

Top Categories in Harv_method
Count
0
500
1000
1500
2000
2500
3000
3500
hand
machine
Other
Harv_method

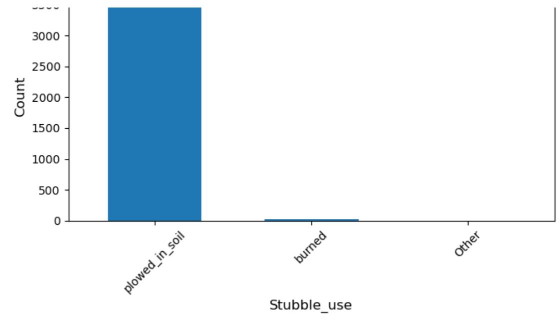

Top Categories in Harv_date
Count
0
500
1000
1500
2000
2500
2022-11-05
2022-11-25
2022-12-01
2022-11-10
2022-10-15
2022-10-20
2022-11-02
2022-11-16
2022-11-24
2022-11-26
Other
Harv_date


Top Categories in Threshing_date
Count
0
500
1000
1500
2000
2500
3000
2022-12-22
2022-10-25
2022-12-23
2022-11-19
2022-12-10
2022-11-12
2022-12-15
2022-11-10
2022-11-15
2022-12-24
Other
Threshing_date


Top Categories in Threshing_method
2000

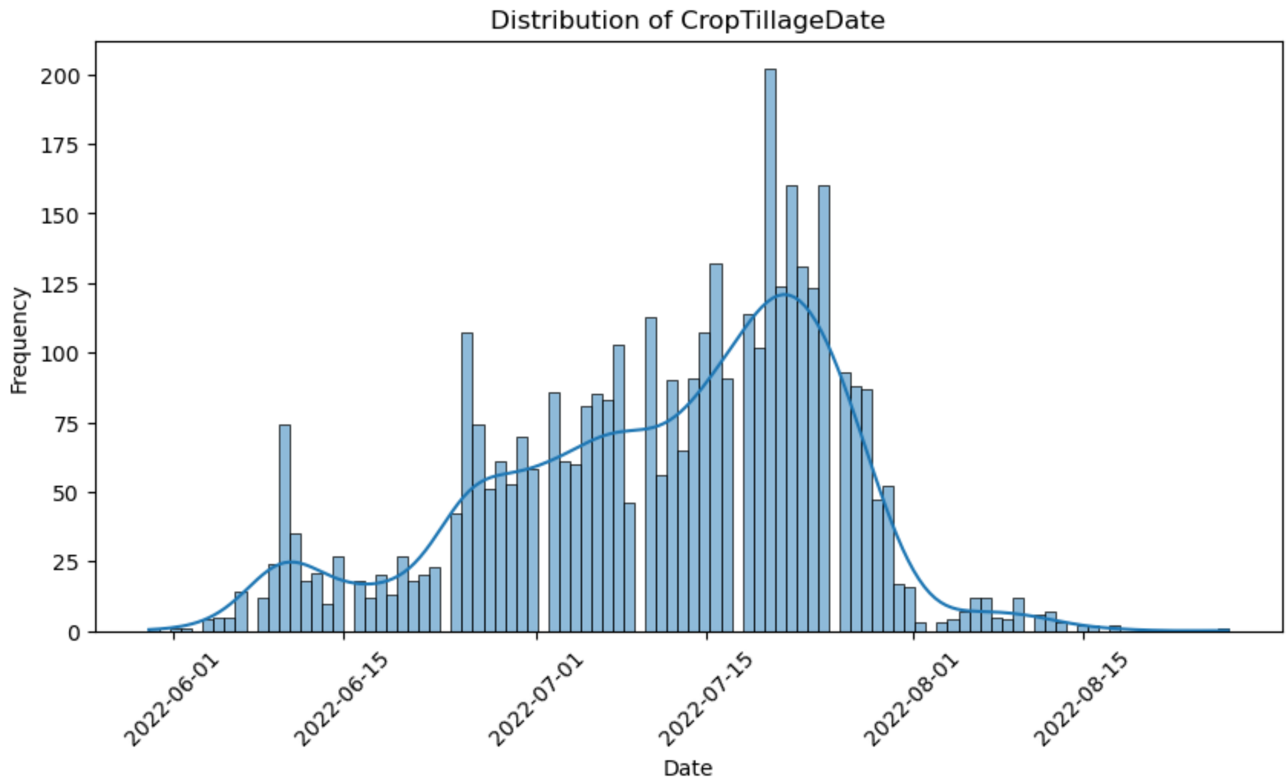

Top Categories in Stubble_use
4000

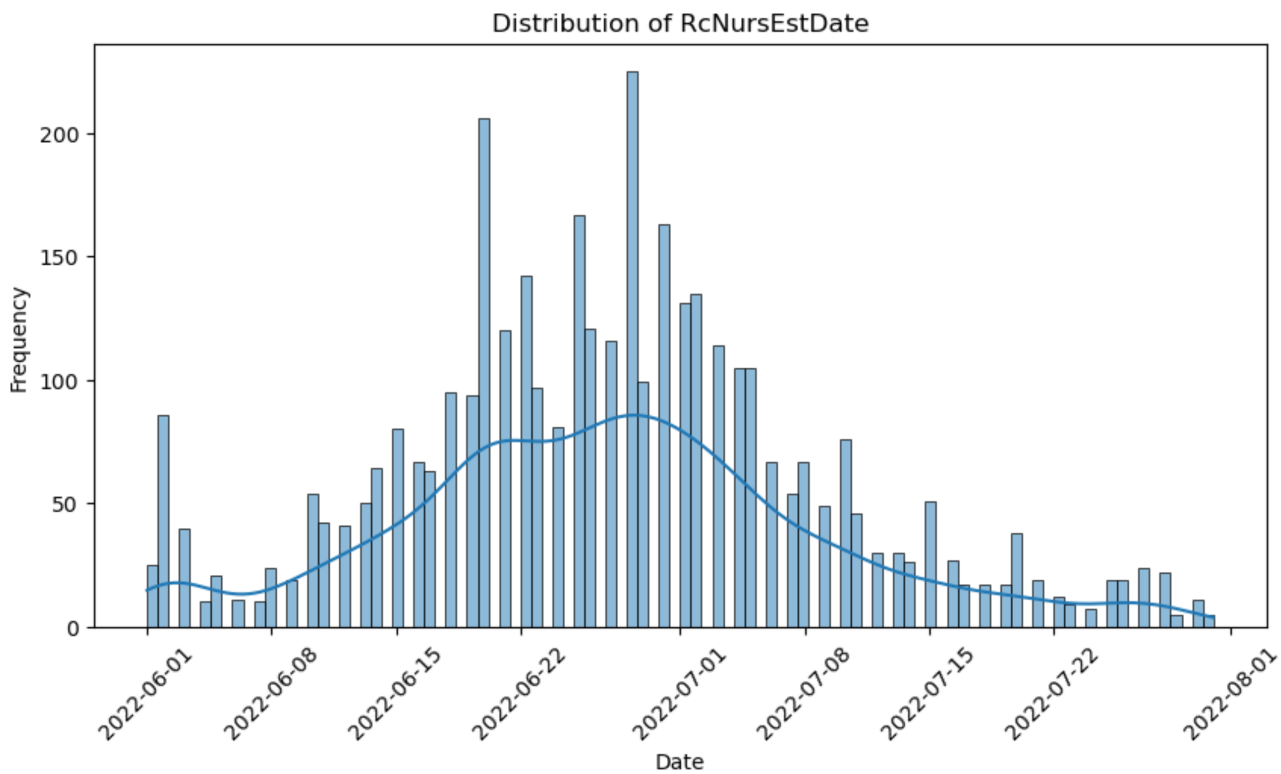


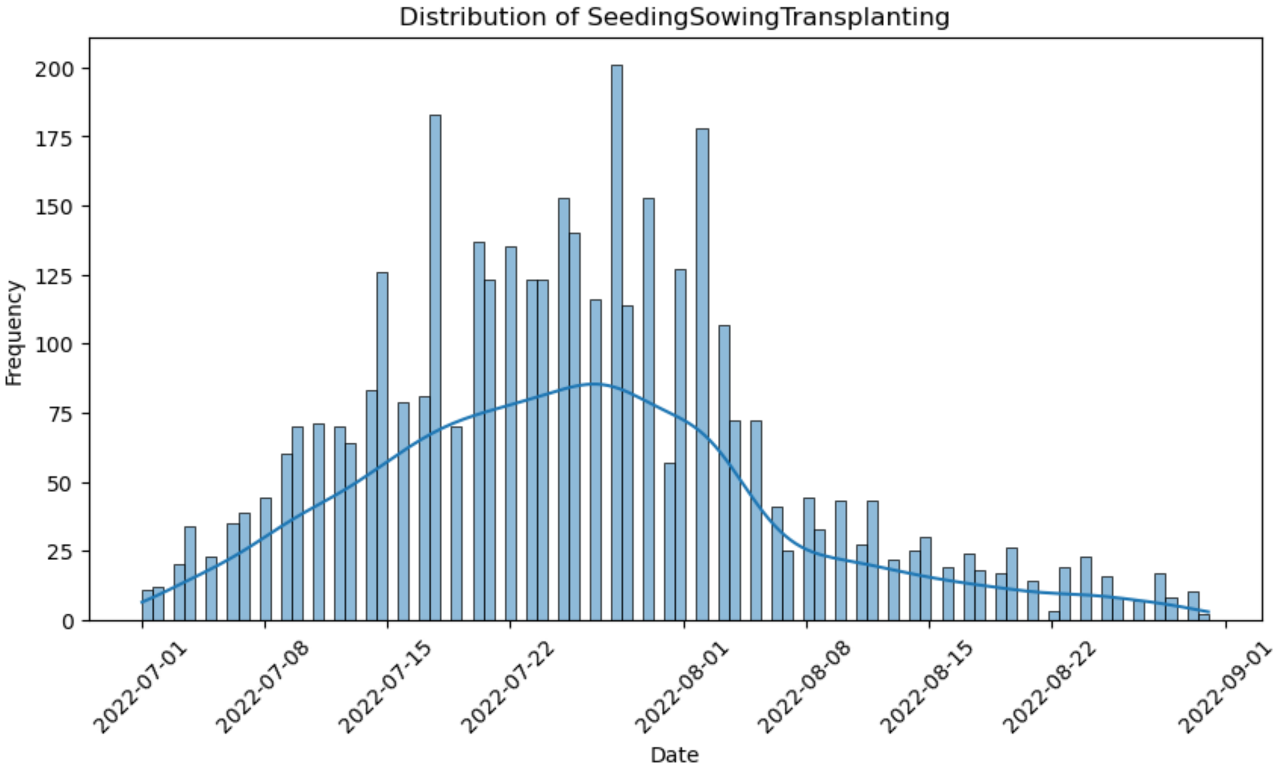


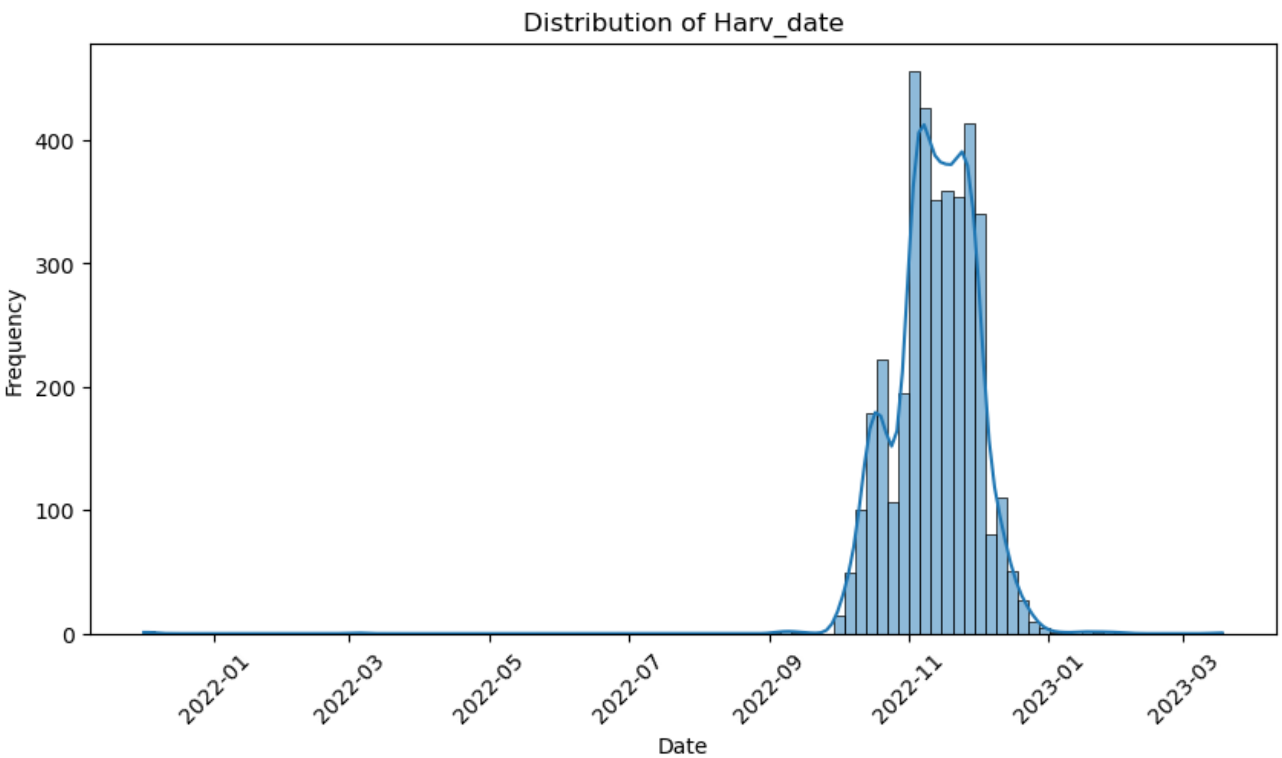
Visualize date columns

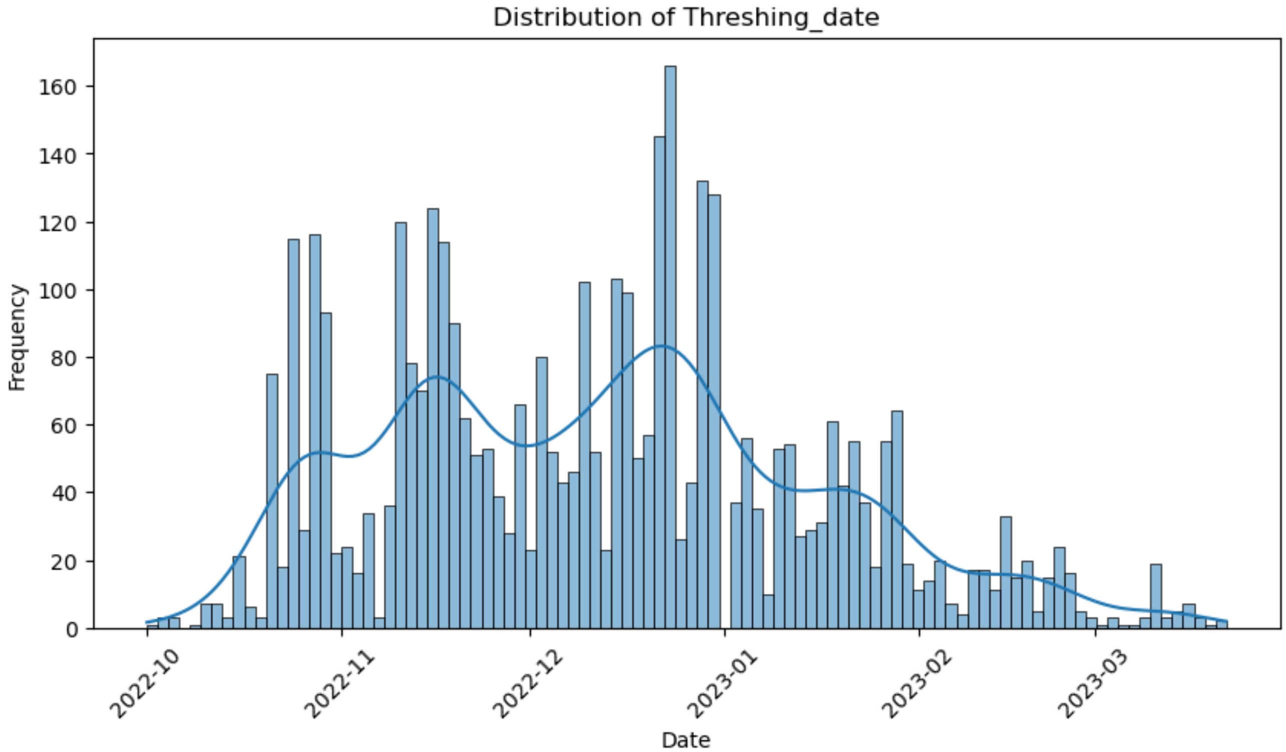
In [30]:
```
# List of date columns
date_columns = ['CropTillageDate', 'RcNursEstDate', 'SeedingSowingTransplant

# Temporarily convert date columns to datetime format
for col in date_columns:
    df[col] = pd.to_datetime(df[col])

# Plot the distribution of each date column
for col in date_columns:
    plt.figure(figsize=(10, 5))
    sns.histplot(df[col].dropna(), bins=100, kde=True)  # Use KDE to see the
    plt.title(f'Distribution of {col}')
    plt.xlabel('Date')
    plt.ylabel('Frequency')
    plt.xticks(rotation=45)
    plt.show()
```


Distribution of CropTillageDate
Frequency
0
25
50
75
100
125
150
175
200
2022-06-01
2022-06-15
2022-07-01
2022-07-15
2022-08-01
2022-08-15
Date


Distribution of RcNursEstDate
Frequency
0
50
100
150
200
2022-06-01
2022-06-08
2022-06-15
2022-06-22
2022-07-01
2022-07-08
2022-07-15
2022-07-22
2022-08-01
Date


Distribution of SeedingSowingTransplanting
Frequency
Date
0
25
50
75
100
125
150
175
200
2022-07-01
2022-07-08
2022-07-15
2022-07-22
2022-08-01
2022-08-08
2022-08-15
2022-08-22
2022-09-01


Distribution of Harv_date
Frequency
Date
0
100
200
300
400
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
2023-03



In [31]:
```python
# Filter rows where 'Harv_date' is before October 1, 2022
df[df['Harv_date'] < '2022-09-01']
```

Out[31]:

	District	Block	CultLand	CropCultLand	LandPreparationMethod
ID					
ID_9P3DV08LL3SX	Jamui	Khaira	50	32	TractorPlough
ID_YTZN9FE7PQUY	Nalanda	Rajgir	30	30	TractorPlough
ID_RL2F5BMVBUAX	Jamui	Khaira	22	16	TractorPlough

In [32]:
```python
df[df['Harv_date'] > '2023-01-01']
```

Out[32]:

ID	District	Block	CultLand	CropCultLand	LandPreparationMe
ID_PX8CNYP9YHPE	Vaishali	Chehrakala	8	6	TractorPl FourWheelTracRota
ID_PATK559888IV	Gaya	Gurua	30	30	TractorPl FourWheelTracRota
ID_ZAQF5TZH58PX	Nalanda	Rajgir	56	42	TractorPl FourWheelTracRota
ID_J7PL485NOPNC	Nalanda	Rajgir	38	28	TractorPl FourWheelTracRota
ID_X2XPIRO6ZYMC	Nalanda	Rajgir	42	32	TractorPl FourWheelTracRota
ID_WIXJZSB1JPAA	Gaya	Gurua	15	15	TractorPl FourWheelTracRota
ID_RCQBDGSCHLXV	Jamui	Jamui	10	10	TractorPl
ID_Z0CHG2VQSJ31	Jamui	Jamui	20	20	TractorPl BullockPl
ID_2KJP9DE2ZBIY	Jamui	Jamui	18	5	WetTillagePud TractorPl BullockPl

In [33]:
```
# Revert to original format if needed (optional)
for col in date_columns:
    df[col] = df[col].astype(str)  # Convert back to string if needed
```

THE END

The rest of this notebook is failed model exploration.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Preprocessing

Split data into training and testing

In [302…

```
# Drop the index column
df = df.reset_index(drop=True)

X = df.drop('Yield', axis=1)  # Features
y = df['Yield']               # Target

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar
```