```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestRegressor

# Load the dataset
df = pd.read_csv("crop.csv")

# Convert Crop_Year column to datetime
df['Crop_Year'] = pd.to_datetime(df['Crop_Year'], format='%Y')

print(df.head())

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

# Check for null values in the dataset
print("\nNumber of null values:\n\n", df.isnull().sum())

# Check for the data types of each column
print("\nData types:\n\n", df.dtypes)

# Get the count of unique values in the categorical columns
state_df = df['State'].value_counts()
crop_df = df['Crop'].value_counts()
season_df = df['Season'].value_counts()

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

# Group the data by state and season, and sum the production for each group
state_season_data = df.groupby(['State', 'Season'])['Production'].sum()

# Convert the grouped data into a DataFrame, and reset the index
state_season_df = state_season_data.to_frame().reset_index()

# Pivot the DataFrame to create a matrix of production values for each state and
season
state_season_matrix = state_season_df.pivot(index='State', columns='Season',
values='Production')

print(state_season_matrix)

# Create a horizontal bar chart of the production values for each state
ax = state_season_matrix.plot(kind='barh', figsize=(10, 20), stacked=True)

# Set the chart title and axis labels
ax.set_title('Crop Production by State and Season')
ax.set_xlabel('Production')
ax.set_ylabel('State')

# Show the chart
plt.show()
```

```
# Group the data by crop type and sum the production for each group
crop_data = df.groupby('Crop')['Production'].sum()

# Sort the crop data in descending order of production
crop_data = crop_data.sort_values(ascending=False)

print(crop_data)

# Create a histogram of the production values for each crop
ax = crop_data.plot(kind='bar', figsize=(10, 5))

# Set the chart title and axis labels
ax.set_title('Crop Production by Type')
ax.set_xlabel('Crop Type')
ax.set_ylabel('Production')

# Show the chart
plt.show()

# Group the data by State and sum the production for each group
state_data = df.groupby('State')['Production'].sum()

# Sort the state data in descending order of production
state_data = state_data.sort_values(ascending=False)

print(state_data)

# Create a histogram of the production values for each crop
ax = state_data.plot(kind='bar', figsize=(10, 5))

# Set the chart title and axis labels
ax.set_title('State Production by Name')
ax.set_xlabel('State Name')
ax.set_ylabel('Production')

# Show the chart
plt.show()

# Group the data by Year and sum the production for each group
time_data = df.groupby('Crop_Year')['Production'].sum()

print(time_data)

# Create a histogram of the production values for each year
ax = time_data.plot(kind='bar', figsize=(10, 5))

# Set the chart title and axis labels
ax.set_title('Crop Production by Year')
ax.set_xlabel('Year')
ax.set_ylabel('Production')

# Show the chart
plt.show()

# Convert categorical features to numerical using LabelEncoder
label_encoder = LabelEncoder()
df['State'] = label_encoder.fit_transform(df['State'])
df['District'] = label_encoder.fit_transform(df['District'])
```

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

# Plot histograms of the features to check for their distribution
df.hist(bins=50, figsize=(20,15))
plt.suptitle('Histograms of features', fontsize=16)

# Scatter plot of Area vs Production
df.plot(kind='scatter', x='Area', y='Production')
plt.title('Area vs Production')
plt.xlabel('Area')
plt.ylabel('Production')
plt.show()

# Box plot to check for outliers
df.boxplot(column=['Area','Production','Yield'], figsize=(10,8))
plt.title('Box plot of Area, Production, and Yield')
plt.show()

# The correlation matrix
corr_matrix = df.corr()
print('\nThe correlation matrix:\n\n', corr_matrix)
plt.figure(figsize=(15,10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation matrix')
plt.show()
```