Name:Bhushan Malpani Div:D15C Roll no: 33

Experiment 01

Aim:

- Load data in Pandas.
- Description of the dataset.
- Drop columns that aren't useful.
- Drop rows with maximum missing values.
- Take care of missing data.
- Create dummy variables.
- Find out outliers (manually)
- standardization and normalization of columns

Data preprocessing

Data preprocessing involves transforming raw data into a structured and meaningful format, making it suitable for analysis. It is a crucial step in data mining, as raw data often contains inconsistencies, missing values, or noise. Ensuring data quality is essential before applying machine learning or data mining algorithms to achieve accurate and reliable results.

Why is Data Preprocessing Important?

Data preprocessing is essential for ensuring the quality and reliability of data before analysis. It helps improve the accuracy and efficiency of machine learning and data mining processes. The key aspects of data quality include:

- **Accuracy:** Ensuring the data is correct and free from errors.
- **Completeness:** Checking for missing or unrecorded data.
- **Consistency:** Verifying that data remains uniform across different sources.
- **Timeliness:** Ensuring the data is up-to-date and relevant.
- **Believability:** Assessing whether the data is reliable and trustworthy.
- **Interpretability:** Making sure the data is clear and easy to understand.

Dataset: Crop Yield Dataset

Name:Bhushan Malpani Div:D15C Roll no: 33

1) Loading Data in Pandas

```python
import pandas as pd

file_path = "/content/crop_yield.csv"
df = pd.read_csv(file_path)

print("Dataset Preview:")
print(df.head().to_string(index=False))
```

```
Dataset Preview:
        Crop  Crop_Year      Season State    Area  Production  Annual_Rainfall  Fertilizer  Pesticide       Yield
    Arecanut       1997 Whole Year  Assam 73814.0       56708           2051.4  7024878.38   22882.34    0.796087
   Arhar/Tur       1997 Kharif      Assam  6637.0        4685           2051.4   631643.29    2057.47    0.710435
 Castor seed       1997 Kharif      Assam   796.0          22           2051.4    75755.32     246.76    0.238333
     Coconut       1997 Whole Year  Assam 19656.0   126905000           2051.4  1870661.52    6093.36 5238.051739
Cotton(lint)       1997 Kharif      Assam  1739.0         794           2051.4   165500.63     539.09    0.420909
```

2)Description of the dataset.

Attribute/Column Name	Data Type	Description
Crop	String	Name of the crop (e.g., Arecanut, Arhar/Tur, Coconut, etc.).
Crop_Year	Float	The year in which the crop was grown.
Season	String	The season in which the crop was cultivated (e.g., Kharif, Whole Year).
State	String	The state in which the crop was grown.
Area	Float	The total area (in hectares) used for cultivation.
Production	Float	The total production of the crop (in metric tons).
Annual_Rainfall	Float	The annual rainfall (in mm) received in the region.
Fertilizer	Float	The amount of fertilizer (in kg) used.
Pesticide	Float	The amount of pesticide (in kg) used.
Yield	Float	The yield (production per unit area) of the crop.

df.info(): Provides an overview of the dataset, including:

- Number of rows and columns.
- Data types of each column (e.g., int, float, object).
- Number of non-null (non-missing) values in each column.

df.describe(): Generates summary statistics for numeric columns, such as:

- count: Number of non-missing values.
- mean: Average value.
- std: Standard deviation.

- min, 25%, 50% (median), 75%, and max: Percentile values.

```
import pandas as pd

file_path = "/content/crop_yield.csv"
df = pd.read_csv(file_path)

print(df.info())
print(df.describe().to_string(index=False))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19689 entries, 0 to 19688
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Crop             19689 non-null  object
 1   Crop_Year        19689 non-null  int64
 2   Season           19689 non-null  object
 3   State            19689 non-null  object
 4   Area             19689 non-null  float64
 5   Production       19689 non-null  int64
 6   Annual_Rainfall  19689 non-null  float64
 7   Fertilizer       19689 non-null  float64
 8   Pesticide        19689 non-null  float64
 9   Yield            19689 non-null  float64
dtypes: float64(5), int64(2), object(3)
memory usage: 1.5+ MB
None
   Crop_Year         Area   Production  Annual_Rainfall   Fertilizer    Pesticide        Yield
19689.000000 1.968900e+04 1.968900e+04     19689.000000 1.968900e+04 1.968900e+04 19689.000000
 2009.127584 1.799266e+05 1.643594e+07      1437.755177 2.410331e+07 4.884835e+04    79.954009
    6.498099 7.328287e+05 2.630568e+08       816.909589 9.494600e+07 2.132874e+05   878.306193
 1997.000000 5.000000e-01 0.000000e+00       301.300000 5.417000e+01 9.000000e-02     0.000000
 2004.000000 1.390000e+03 1.393000e+03       940.700000 1.880146e+05 3.567000e+02     0.600000
 2010.000000 9.317000e+03 1.380400e+04      1247.600000 1.234957e+06 2.421900e+03     1.030000
 2015.000000 7.511200e+04 1.227180e+05      1643.700000 1.000385e+07 2.004170e+04     2.388889
 2020.000000 5.080810e+07 6.326000e+09      6552.700000 4.835407e+09 1.575051e+07 21105.000000
```

3) Drop columns that aren't useful: Columns like Invoice ID may not contribute to analysis (it's often just an identifier). Removing irrelevant columns reduces complexity.

```
import pandas as pd

file_path = "/content/crop_yield.csv"
df = pd.read_csv(file_path)

df = df.drop(['Crop_Year'], axis=1)
df.head()
```

	Crop	Season	State	Area	Production	Annual_Rainfall	Fertilizer	Pesticide	Yield
0	Arecanut	Whole Year	Assam	73814.0	56708	2051.4	7024878.38	22882.34	0.796087
1	Arhar/Tur	Kharif	Assam	6637.0	4685	2051.4	631643.29	2057.47	0.710435
2	Castor seed	Kharif	Assam	796.0	22	2051.4	75755.32	246.76	0.238333
3	Coconut	Whole Year	Assam	19656.0	126905000	2051.4	1870661.52	6093.36	5238.051739
4	Cotton(lint)	Kharif	Assam	1739.0	794	2051.4	165500.63	539.09	0.420909

4)Drop rows with maximum missing values.

df.dropna(thresh=int(0.5 * len(df.columns))):

- Drops rows where more than half the columns have missing (NaN) values.
- thresh=int(0.5 * len(df.columns)): Ensures that a row must have at least 50% non-null values to remain.

df = ...: Updates the DataFrame after dropping rows.
print(df.info()): Confirms that rows with excessive missing values have been removed.

```
import pandas as pd

file_path = "/content/crop_yield.csv"
df = pd.read_csv(file_path)

df = df.dropna(thresh=int(0.5 * len(df.columns)))
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19689 entries, 0 to 19688
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Crop             19689 non-null  object
 1   Crop_Year        19689 non-null  int64
 2   Season           19689 non-null  object
 3   State            19689 non-null  object
 4   Area             19689 non-null  float64
 5   Production       19689 non-null  int64
 6   Annual_Rainfall  19689 non-null  float64
 7   Fertilizer       19689 non-null  float64
 8   Pesticide        19689 non-null  float64
 9   Yield            19689 non-null  float64
dtypes: float64(5), int64(2), object(3)
memory usage: 1.5+ MB
```

5)Take care of missing data.

df.fillna(df.mean()): Replaces missing values (NaN) in numeric columns with the mean of that column.

```
import pandas as pd

file_path = "/content/crop_yield.csv"
df = pd.read_csv(file_path)

df = df.dropna(thresh=int(0.5 * len(df.columns)))

numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].mean())

print(df.isnull().sum())
```

```
Crop              0
Crop_Year         0
Season            0
State             0
Area              0
Production        0
Annual_Rainfall   0
Fertilizer        0
Pesticide         0
Yield             0
dtype: int64
```

6)Create dummy variables.

pd.get_dummies(): Converts categorical variables into dummy variables (binary indicators: 0 or 1).

- Example: The Gender column becomes Gender_Male (1 if Male, 0 otherwise).

columns=['...']: Specifies which columns to convert.
drop_first=True: Avoids the "dummy variable trap" by dropping one dummy variable to prevent multicollinearity.

Name:Bhushan Malpani Div:D15C Roll no: 33

```python
import pandas as pd

file_path = "/content/crop_yield.csv"
df = pd.read_csv(file_path)

df = df.dropna(thresh=int(0.5 * len(df.columns)))

df = pd.get_dummies(df, columns = ['Season'], drop_first = True)
df.head()
```

Crop_Year	State	Area	Production	Annual_Rainfall	Fertilizer	Pesticide	Yield	Season_Kharif	Season_Rabi	Season_Summer	Season_Whole Year	Season_Winter
1997	Assam	73814.0	56708	2051.4	7024878.38	22882.34	0.796087	False	False	False	True	False
1997	Assam	6637.0	4685	2051.4	631643.29	2057.47	0.710435	True	False	False	False	False
1997	Assam	796.0	22	2051.4	75755.32	246.76	0.238333	True	False	False	False	False
1997	Assam	19656.0	126905000	2051.4	1870661.52	6093.36	5238.051739	False	False	False	True	False
1997	Assam	1739.0	794	2051.4	165500.63	539.09	0.420909	True	False	False	False	False

7)Find out outliers (manually)

```python
import pandas as pd

file_path = "/content/crop_yield.csv"
df = pd.read_csv(file_path)

def detect_outliers(df, col):
    Q1 = df[col].quantile(0.25)  # First quartile (25th percentile)
    Q3 = df[col].quantile(0.75)  # Third quartile (75th percentile)
    IQR = Q3 - Q1  # Interquartile range

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    return df[(df[col] < lower_bound) | (df[col] > upper_bound)]

# Detect outliers in the 'Yield' column
outliers = detect_outliers(df, 'Yield')

print(outliers)

if outliers.empty:
    print("No outliers detected.")
else:
    print(f"Outliers detected:\n{outliers}")

print(f"Number of outliers: {len(outliers)}")
```

```
            Crop  Crop_Year       Season      State      Area  Production  \
3        Coconut       1997  Whole Year       Assam   19656.0   126905000
7           Jute       1997  Kharif           Assam   94520.0      904095
14        Potato       1997  Whole Year       Assam   75259.0      671871
21     Sugarcane       1997  Kharif           Assam   31318.0     1287451
54     Sugarcane       1997  Whole Year   Karnataka  308857.0    28999269
...          ...        ...          ...        ...       ...         ...
19618  Sugarcane       2016  Winter          Odisha    5493.0      344294
19636     Potato       2017  Winter          Odisha    3966.0       41812
19647  Sugarcane       2017  Winter          Odisha    3713.0      240245
19665     Potato       2018  Winter          Odisha    4900.0       54455
19676  Sugarcane       2018  Winter          Odisha    6778.0      417672

       Annual_Rainfall   Fertilizer  Pesticide        Yield
3               2051.4   1870661.52    6093.36  5238.051739
7               2051.4   8995468.40   29301.20     9.919565
14              2051.4   7162399.03   23330.29     7.561304
21              2051.4   2980534.06    9708.58    41.896957
54              1266.7  29393920.69   95745.67    91.747368
...                ...          ...        ...          ...
19618           1460.5    841802.25    1922.55    56.160400
19636           1344.4    624407.04    1507.08    11.955455
19647           1344.4    584574.72    1410.94    56.588000
19665           1635.9    794780.00    1715.00    13.017308
19676           1635.9   1099391.60    2372.30    57.584545

[3065 rows x 10 columns]
Outliers detected:
            Crop  Crop_Year       Season      State      Area  Production  \
3        Coconut       1997  Whole Year       Assam   19656.0   126905000
7           Jute       1997  Kharif           Assam   94520.0      904095
14        Potato       1997  Whole Year       Assam   75259.0      671871
21     Sugarcane       1997  Kharif           Assam   31318.0     1287451
54     Sugarcane       1997  Whole Year   Karnataka  308857.0    28999269
...          ...        ...          ...        ...       ...         ...
19618  Sugarcane       2016  Winter          Odisha    5493.0      344294
19636     Potato       2017  Winter          Odisha    3966.0       41812
19647  Sugarcane       2017  Winter          Odisha    3713.0      240245
19665     Potato       2018  Winter          Odisha    4900.0       54455
19676  Sugarcane       2018  Winter          Odisha    6778.0      417672

       Annual_Rainfall   Fertilizer  Pesticide        Yield
3               2051.4   1870661.52    6093.36  5238.051739
7               2051.4   8995468.40   29301.20     9.919565
14              2051.4   7162399.03   23330.29     7.561304
21              2051.4   2980534.06    9708.58    41.896957
54              1266.7  29393920.69   95745.67    91.747368
...                ...          ...        ...          ...
19618           1460.5    841802.25    1922.55    56.160400
19636           1344.4    624407.04    1507.08    11.955455
19647           1344.4    584574.72    1410.94    56.588000
19665           1635.9    794780.00    1715.00    13.017308
19676           1635.9   1099391.60    2372.30    57.584545

[3065 rows x 10 columns]
Number of outliers: 3065
```

Name:Bhushan Malpani Div:D15C Roll no: 33

8) standardization and normalization of columns

Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.

Standardization equation

$$X' = \frac{X - \mu}{\sigma}$$

To standardize your data, we need to import the StandardScalar from the sklearn library and apply it to our dataset.

Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

Normalization equation

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.

- When the value of X is the minimum value in the column, the numerator will be 0, and hence X' is 0
- On the other hand, when the value of X is the maximum value in the column, the numerator is equal to the denominator and thus the value of X' is 1
- If the value of X is between the minimum and the maximum value, then the value of X' is between 0 and 1

To normalize your data, you need to import the MinMaxScalar from the sklearn library and apply it to our dataset.

```python
import pandas as pd
from sklearn.preprocessing import StandardScaler, MinMaxScaler

file_path = "/content/crop_yield.csv"
df = pd.read_csv(file_path)

cols_to_transform = ['Area', 'Production']

standard_scaler = StandardScaler()
minmax_scaler = MinMaxScaler()

df[cols_to_transform] = standard_scaler.fit_transform(df[cols_to_transform])

df[cols_to_transform] = minmax_scaler.fit_transform(df[cols_to_transform])

print(df[cols_to_transform].head())
```

```
       Area    Production
0  0.001453  8.964274e-06
1  0.000131  7.405944e-07
2  0.000016  3.477711e-09
3  0.000387  2.006086e-02
4  0.000034  1.255138e-07
```

Conclusion:

Thus we have understood how to perform data preprocessing which can further be taken into exploratory data analysis and further in the Model preparation sequence.