**IMPLEMENTATION OF PREPROCESSING TECHNIQUES**

Ex. No. 3

**Aim:**

To implement the preprocessing techniques in data cleaning using python.

**Preprocessing Techniques:**

Preprocessing is a crucial step in the data science pipeline, as it helps in preparing raw data for analysis and modelling. Effective preprocessing can significantly improve the performance and accuracy of machine learning models.

**Data Cleaning:**

* 1. **Handling Missing Values:**
     + **Imputation:** Replace missing values with a statistical measure like mean, median, or mode.
     + **Deletion:** Remove rows or columns with missing values.
  2. **Noise Filtering:**
* **Outlier Detection and Removal:** Identify and handle outliers that can distort the analysis and modelling process.

**Detecting Outliers:**

1. Box-Plot
2. Inter-quartile Range
3. Z-score

**Box-Plot:**

**Algorithm:**

1. Import required libraries:

- numpy as np

- pandas as pd

- matplotlib.pyplot as plt

- seaborn as sns

2. Read the 'insurance.csv' file and store it in a DataFrame named insurance:

- insurance = pd.read\_csv('insurance.csv')

3. Generate summary statistics for the insurance DataFrame:

- summary\_statistics = insurance.describe()

- Display summary\_statistics

4. Create a new figure with a size of 4x8 inches:

- plt.figure(figsize=(4, 8))

5. Plot a boxplot for the 'bmi' column of the insurance DataFrame:

- sns.boxplot(y=insurance.bmi)

**Code:**

# import the required library

import numpy as np

import pandas as pd

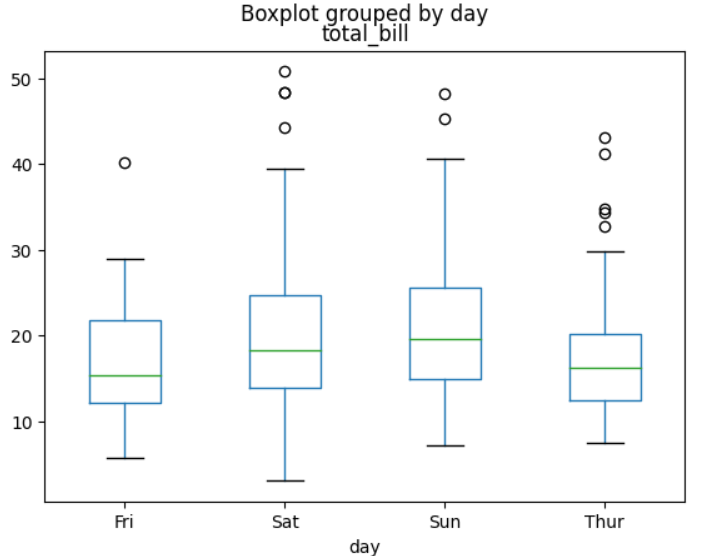
import matplotlib.pyplot as plt

# load the dataset

df = pd.read\_csv("tips.csv")

df.boxplot(by ='day', column =['total\_bill'], grid = False)

**Output:**

****

**Inter-quartile Range:**

**Algorithm:**

1. Import required libraries:

- numpy as np

- pandas as pd

- matplotlib.pyplot as plt

- seaborn as sns

2. Read the 'insurance.csv' file and store it in a DataFrame named insurance:

- insurance = pd.read\_csv('insurance.csv')

3. Define a function named outlier\_IQR to calculate outliers using Interquartile Range (IQR):

1. Calculate Q1, Q3, and IQR.

2. Define the threshold and calculate lower and upper bounds.

3. Identify and print the number of lower and upper outliers.

4. Return the total number of outliers.

4. Calculate and display outliers for the 'bmi' column of the insurance DataFrame:

- outlier\_IQR(insurance, 'bmi')

5. Create a new figure with a size of 10x8 inches:

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

6. Plot a distribution plot for the 'bmi' column of the insurance DataFrame with 25 bins:

- sns.distplot(insurance.bmi, bins=25)

7. Create bands to identify outliers using plt.axvspan():

- Create red bands for values less than the lower bound and greater than the upper bound.

**Code:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

 # import sales data

s = pd.read\_csv('tips.csv')

s

# defining the outlier function for interquartile range

def outlier\_IQR(df, column):

    global lower, upper

    Q1 = np.quantile(df[column], 0.25) # first quartile

    Q3 = np.quantile(df[column], 0.75) # third quartile

    IQR = Q3 - Q1 # inter - quartile range

    threshold = 1.5 \* IQR # defining the threshold

    lower = Q1 - threshold

    upper = Q3 + threshold

    lower\_bound = df[df[column] < lower]

    upper\_bound = df[df[column] > upper]

    #printing IQR, threshold, lower bound, upper bound and total number of outlier

    print('IQR is:', IQR)

    print('Threshold is:', threshold)

    print('Lower bound is:', lower)

    print('Upper bound is:', upper)

    return print('total number of outliers are:', lower\_bound.shape[0] + upper\_bound.shape[0])

# getting the IQR, threshold, lower bound, upper bound, total outliers

outlier\_IQR(s, 'tip')

# plotting figure

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

sns.distplot(s.tip, bins = 25)

# creting band to identify the outliers

# plt.axvspan : function sets the vertical rectangle across the axes of the plot

plt.axvspan(xmin = lower, xmax = s.tip.min(), alpha = 0.2, color = 'red')

plt.axvspan(xmin = upper, xmax = s.tip.max(), alpha = 0.2, color = 'red')

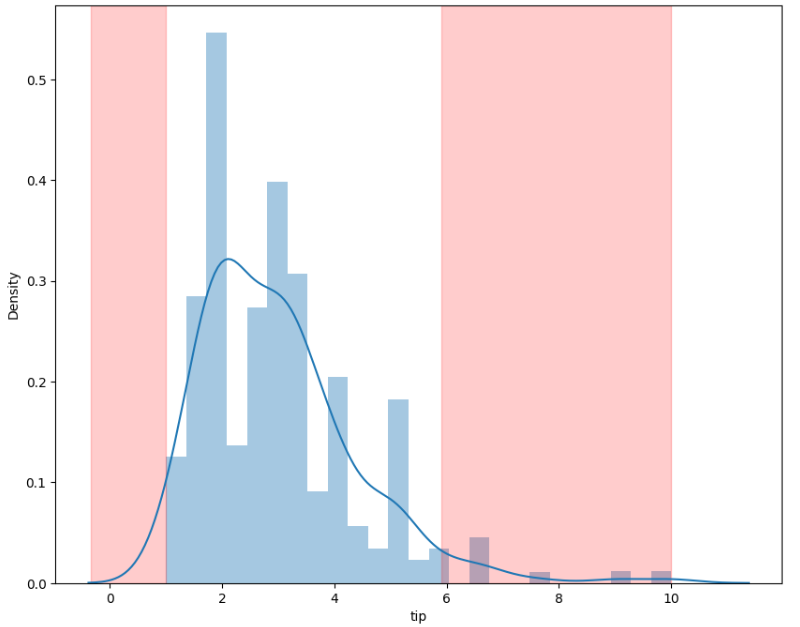
**Output:**

**IQR is: 1.5625**

**Threshold is: 2.34375**

**Lower bound is: -0.34375**

**Upper bound is: 5.90625**

**Total number of outliers are: 9**

**Z-score:**

**Algorithm:**

1. Import required libraries:

- numpy as np

- pandas as pd

- matplotlib.pyplot as plt

- seaborn as sns

2. Read the 'insurance.csv' file and store it in a DataFrame named insurance:

- insurance = pd.read\_csv('insurance.csv')

3. Define a function named outlier\_zscore to calculate outliers using Z-Score:

1. Initialize empty lists for outliers and z-scores.

2. Set z-score threshold to 3.

3. Calculate mean and standard deviation of the input column.

4. Iterate through each value to calculate z-score and identify outliers.

5. Print the total number of outliers.

6. Return the total number of outliers.

4. Calculate and display outliers for the 'bmi' column of the insurance DataFrame:

- outlier\_zscore(insurance.bmi)

5. Create a new figure with a size of 10x5 inches:

- plt.figure(figsize=(10, 5))

6. Plot a distribution plot for the z\_score list:

- sns.distplot(z\_score)

7. Create a red band to identify outliers using plt.axvspan():

- Create a band starting from the z-score threshold of 3 extending up to the maximum z-score.

**Code:**

# import library:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# import sales data

s = pd.read\_csv('tips.csv')

s

# defining the outlier function for interquartile range

def outlier\_zscore(df\_column):

global outlier, z\_score

outlier = []

z\_score = []

threshold = 3

mean = np.mean(df\_column)

std = np.std(df\_column)

for i in df\_column:

zscore = (i - mean)/std

z\_score.append(zscore)

if np.abs(zscore) > threshold:

outlier.append(i)

return print("total number of outliers", len(outlier))

# getting total number of outlier

outlier\_zscore(s.tip)

# plotting figure

plt.figure(figsize = (10,5))

sns.distplot(z\_score)

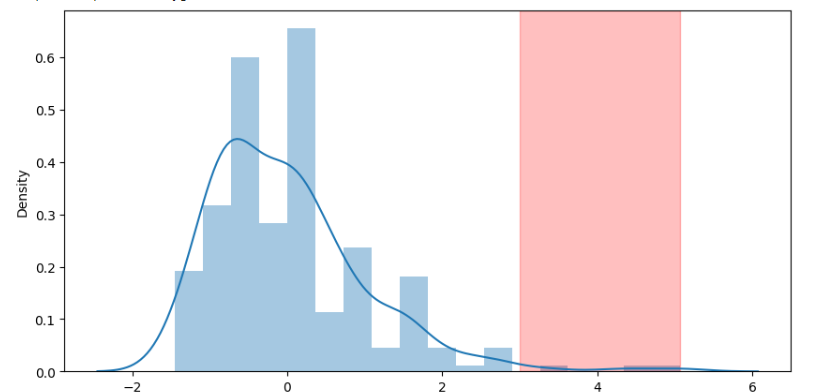
# creting band to identify the outliers

# plt.axvspan : function sets the vertical rectangle across the axes of the plot

plt.axvspan(xmin = 3 ,xmax = max(z\_score),alpha = 0.25, color ='red')

**Output:**

total number of outliers : 3



**Missing Values:**

**Algorithm:**

1. Import the pandas library with the alias pd.

2. Read data from the CSV file located at '/content/gdrive/MyDrive/Colab Notebooks/Studentdata.csv':

- Use pd.read\_csv('/content/gdrive/MyDrive/Colab Notebooks/Studentdata.csv').

- Store the read data in a DataFrame named data.

3. Print the content of the data DataFrame:

- Use print(data).

**Code:**

import pandas as pd

data=pd.read\_csv('/content/gdrive/MyDrive/Colab Notebooks/Studentdata.csv')

print(data)

**Output:**

Name Id Mark1 Mark2 Mark3 Total

0 shan 58.0 78.0 NaN 90.0 240.0

1 karan NaN NaN NaN NaN 800.0

2 petchi 90.0 NaN 78.0 80.0 NaN

3 dinesh 56.0 67.0 NaN NaN 400.0

4 ravi 12.0 NaN 56.0 NaN NaN

5 karthi NaN NaN NaN NaN NaN

**Missing values of data fill using mean, median, mode, standard deviation:**

**Algorithm:**

1. Fill missing values in the 'Id' column:

- Calculate the mean of the 'Id' column.

- Use fillna() to fill missing values in 'Id' with the calculated mean value.

2. Fill missing values in the 'Mark1' column:

- Calculate the median of the 'Mark1' column.

- Use fillna() to fill missing values in 'Mark1' with the calculated median value.

3. Fill missing values in the 'Mark2' column:

- Calculate the mean of the 'Mark2' column.

- Use fillna() to fill missing values in 'Mark2' with the calculated mean value.

4. Fill missing values in the 'Mark3' column:

- Calculate the standard deviation of the 'Mark3' column.

- Use fillna() to fill missing values in 'Mark3' with the calculated standard deviation value.

5. Fill missing values in the 'Total' column:

- Calculate the mean of the 'Total' column.

- Use fillna() to fill missing values in 'Total' with the calculated mean value.

**Code:**

data['Id']=data['Id'].fillna(data['Id'].mean())

data['Mark1']=data['Mark1'].fillna(data['Mark1'].median())

data['Mark2']=data['Mark2'].fillna(data['Mark2'].mean())

data['Mark3']=data['Mark3'].fillna(data['Mark3'].std())

data['Total']=data['Total'].fillna(data['Total'].mean())

print(data)

**Output:**

Name Id Mark1 Mark2 Mark3 Total

0 shan 58.0 78.0 67.0 90.000000 240.0

1 Asir 54.0 72.5 67.0 7.071068 800.0

2 Karan 90.0 72.5 78.0 80.000000 480.0

3 petchi 56.0 67.0 67.0 7.071068 400.0

4 ravi 12.0 72.5 56.0 7.071068 480.0

5 karthi 54.0 72.5 67.0 7.071068 480.0

**Rubrics:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Problem**  **Understanding**  **(10)** | **Implementation**  **(20)** | **Viva**  **(10)** | **Time Management**  **(10)** | **Total**  **(50)** |
|  |  |  |  |  |

**Result:**

Thus the implementation of the preprocessing techniques in data cleaning using python were successfully executed. The output was verified.