Machine Learning

LAB

Lab #3

Outlier Detection and Removal

Instructor: Saad Rashad

Course Code: AL3002

Semester Fall 2024

Department of Computer Science,
National University of Computer and Emerging Sciences FAST
Peshawar Campus

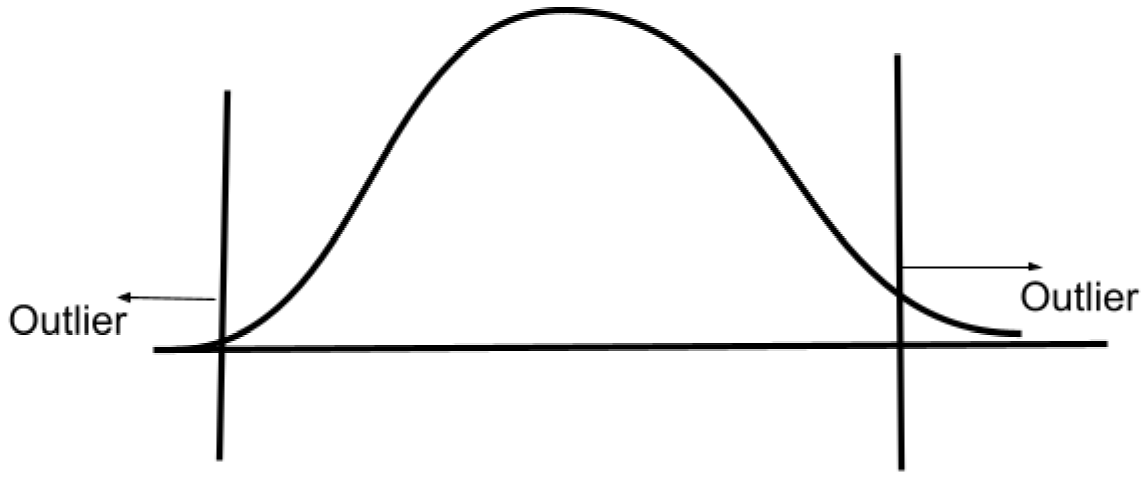
1. **Outliers:**
    - An *outlier* is an observation that lies an abnormal distance from other values in a random sample from a population. In a sense, **this definition leaves it up to the analyst (or a consensus process) to decide what will be considered abnormal.**

3.1 When are Outliers Dangerous?

- They impact statistical measures like mean or variance
- They can affect the strength and direction of correlation between variables, potentially leading to misleading interpretations
- Create a learning bias in the model. Specially those models that deal with weights like linear and logistic regression, Adaboost and Deep learning

3.2 Treatment:

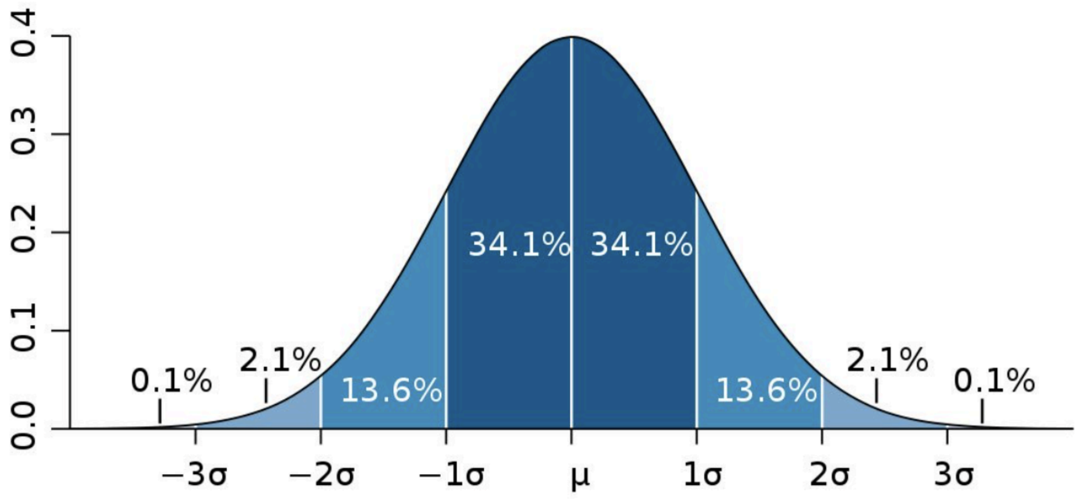
- **Trimming:** Remove the detected outliers from both the ends of the data distribution.
- **Capping:**
    a. Decide on the thresholds that define the acceptable range for your data. Common choices are based on percentiles, such as the 1st and 99th percentiles, or the 5th and 95th percentiles.
    b. Find the data points that fall outside these thresholds. These are considered extreme values.
    c. **Lower Bound:** Replace any value below the lower threshold with the lower threshold value.
    d. **Upper Bound:** Replace any value above the upper threshold with the upper threshold value.

e.


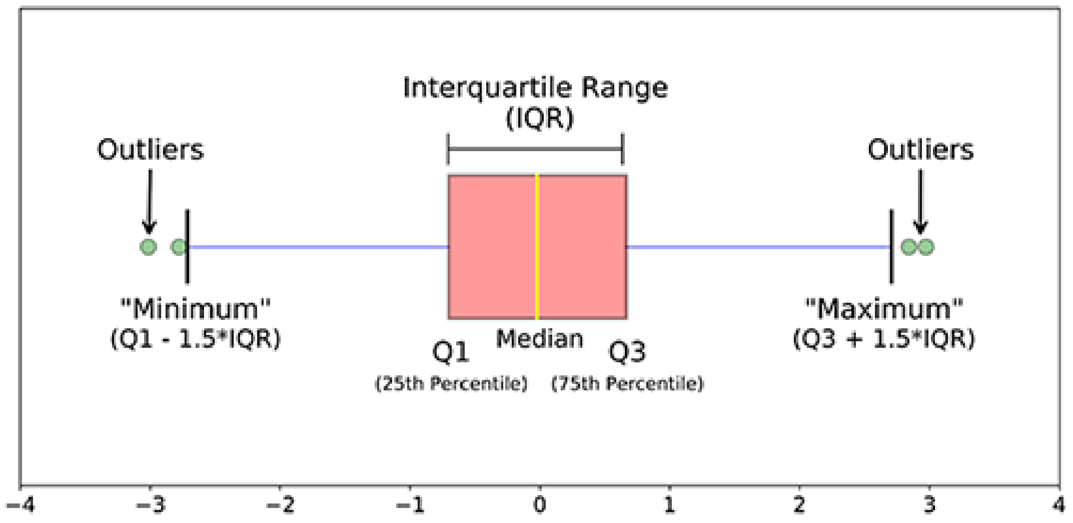
- **Discretization:**
  Divide the range of the continuous variable into equal-width intervals. Each interval represents a discrete category.
  Example: For a variable ranging from 0 to 100, create bins of width 10 (0-10, 10-20, 20-30, etc.).

3.2 How to detect Outliers:

- **Normal Distribution:**
  Z-score:



- **Skewed Distribution:**
  IQR:



3.2.1 Z-Score:

- Zscore is implemented on that column/attribute which is normally distributed or almost normally distributed .
- In normal distribution $\{\mu - \sigma\}\ and\ \{\mu + \sigma\}$ contain 68% of the values, similarly $\{\mu - 2\sigma\}\ and\ \{\mu + 2\sigma\}$ contain 95% of the values and $\{\mu - 3\sigma\}\ and\ \{\mu + 3\sigma\}$ contain 99.7% values. All the other values are considered outliers.

Understanding the Z-Score

Definition: The Z-Score measures how many standard deviations a data point is from the mean of the dataset. The formula is:

$$Z = (X - \mu)/\sigma$$

Where:

- **X** is the value of the data point,
- **μ** is the mean of the data.
- **σ** is the standard deviation of the data.

3.2.2 IQR:

- The **Interquartile Range (IQR)** method is a robust technique for detecting outliers, especially when the data is not normally distributed meaning it is skewed.
- The **Interquartile Range (IQR)** measures the range within which the central 50% of the data falls. It is calculated as:

$$IQR = Q3 - Q1$$

Step to detect outlier using IQR:

- **Calculate Q1 or 25th percentile**
- **Calculate Q3 or 75th percentile**
- **Calculate the IQR**
- **Calculate upper bound**

$$Q3 + 1.5 * IQR$$

- **Calculate Lower bound**

$$Q1 - 1.5 * IQR$$

AL_3002_ML_Lab_3

September 6, 2024

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[2]:
```
import pandas as pd
import numpy as np
```

[3]:
```
csv_file='/content/drive/MyDrive/Colab Notebooks/placement.csv'
df = pd.read_csv(csv_file)
```

[4]:
```
df.shape
```

[4]:
```
(1000, 3)
```

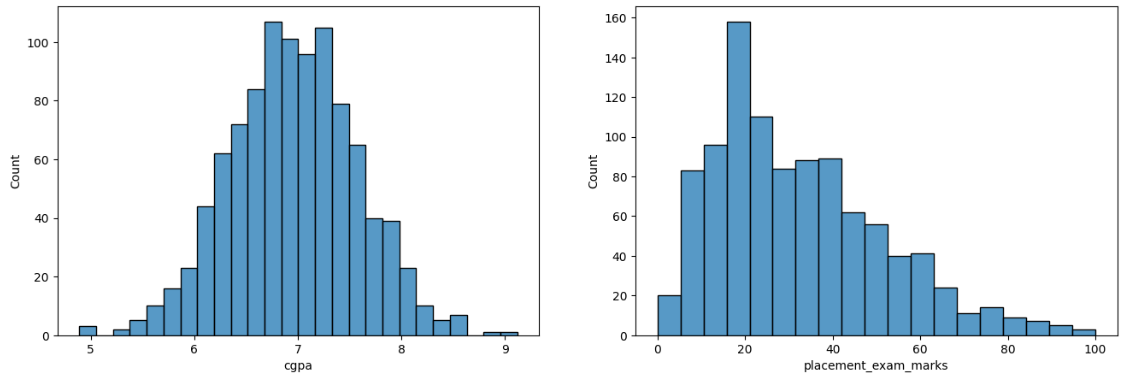
[5]:
```
df.head(10)
```

[5]:
```
   cgpa  placement_exam_marks  placed
0  7.19                  26.0       1
1  7.46                  38.0       1
2  7.54                  40.0       1
3  6.42                   8.0       1
4  7.23                  17.0       0
5  7.30                  23.0       1
6  6.69                  11.0       0
7  7.12                  39.0       1
8  6.45                  38.0       0
9  7.75                  94.0       1
```

[6]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

[7]:
```
plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.histplot(df['cgpa'])

plt.subplot(1,2,2)
sns.histplot(df['placement_exam_marks'])
```

```
plt.show()
```



0.1 Z-score Code:

[8]:
```
print("Mean value of cgpa",df['cgpa'].mean())
print("Std value of cgpa",df['cgpa'].std())
print("Min value of cgpa",df['cgpa'].min())
print("Max value of cgpa",df['cgpa'].max())
```

```
Mean value of cgpa 6.96124
Std value of cgpa 0.6158978751323896
Min value of cgpa 4.89
Max value of cgpa 9.12
```

[36]:
```
# Finding the boundary values
print("Highest allowed",df['cgpa'].mean() + 3*df['cgpa'].std())
print("Lowest allowed",df['cgpa'].mean() - 3*df['cgpa'].std())
```

```
Highest allowed 8.808933625397168
Lowest allowed 5.113546374602832
```

[37]:
```
# Finding the outliers
df[(df['cgpa'] > 8.80) | (df['cgpa'] < 5.11)]
```

[37]:
```
     cgpa  placement_exam_marks  placed
485  4.92                  44.0       1
995  8.87                  44.0       1
996  9.12                  65.0       1
997  4.89                  34.0       0
999  4.90                  10.0       1
```

[38]:
```
# Trimming

new_df = df[(df['cgpa'] < 8.80) & (df['cgpa'] > 5.11)]
```

```
new_df
```

```
[38]:      cgpa  placement_exam_marks  placed
      0    7.19                  26.0       1
      1    7.46                  38.0       1
      2    7.54                  40.0       1
      3    6.42                   8.0       1
      4    7.23                  17.0       0
      ..    ...                   ...     ...
      991  7.04                  57.0       0
      992  6.26                  12.0       0
      993  6.73                  21.0       1
      994  6.48                  63.0       0
      998  8.62                  46.0       1

      [995 rows x 3 columns]
```

```
[39]: # Approach 2

      # Calculating the Zscore

      df['cgpa_zscore'] = (df['cgpa'] - df['cgpa'].mean())/df['cgpa'].std()
```

```
[40]: df[df['cgpa_zscore'] > 3]
```

```
[40]:      cgpa  placement_exam_marks  placed  cgpa_zscore
      995  8.87                  44.0       1     3.099150
      996  9.12                  65.0       1     3.505062
```

```
[41]: df[df['cgpa_zscore'] < -3]
```

```
[41]:      cgpa  placement_exam_marks  placed  cgpa_zscore
      485  4.92                  44.0       1    -3.314251
      997  4.89                  34.0       0    -3.362960
      999  4.90                  10.0       1    -3.346724
```

```
[42]: df[(df['cgpa_zscore'] > 3) | (df['cgpa_zscore'] < -3)]
```

```
[42]:      cgpa  placement_exam_marks  placed  cgpa_zscore
      485  4.92                  44.0       1    -3.314251
      995  8.87                  44.0       1     3.099150
      996  9.12                  65.0       1     3.505062
      997  4.89                  34.0       0    -3.362960
      999  4.90                  10.0       1    -3.346724
```

```
[43]: # Trimming
      new_df = df[(df['cgpa_zscore'] < 3) & (df['cgpa_zscore'] > -3)]
      new_df
```

```
[43]:      cgpa  placement_exam_marks  placed  cgpa_zscore
     0    7.19                  26.0       1     0.371425
     1    7.46                  38.0       1     0.809810
     2    7.54                  40.0       1     0.939701
     3    6.42                   8.0       1    -0.878782
     4    7.23                  17.0       0     0.436371
     ..    ...                   ...     ...          ...
     991  7.04                  57.0       0     0.127878
     992  6.26                  12.0       0    -1.138565
     993  6.73                  21.0       1    -0.375452
     994  6.48                  63.0       0    -0.781363
     998  8.62                  46.0       1     2.693239

     [995 rows x 4 columns]
```

```
[47]: upper_limit = df['cgpa'].mean() + 3*df['cgpa'].std()
      lower_limit = df['cgpa'].mean() - 3*df['cgpa'].std()
      print(upper_limit)
      print(lower_limit)
```

```
8.808933625397168
5.113546374602832
```

```
[49]: df['cgpa'] = np.where(
          df['cgpa']>upper_limit,
          upper_limit,
          np.where(
              df['cgpa']<lower_limit,
              lower_limit,
              df['cgpa']
          )
      )
```

```
[50]: df.shape
```

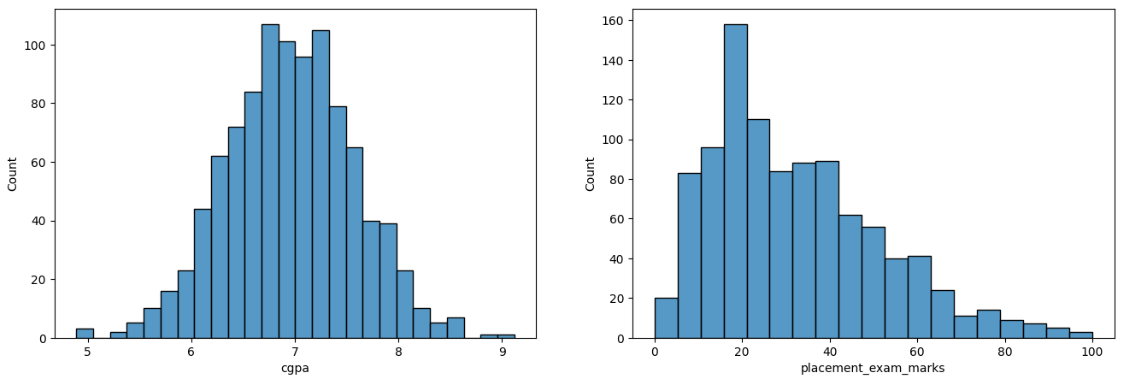
```
[50]: (1000, 4)
```

```
[51]: df['cgpa'].describe()
```

```
[51]: count    1000.000000
      mean        6.961499
      std         0.612688
      min         5.113546
      25%         6.550000
      50%         6.960000
      75%         7.370000
      max         8.808934
      Name: cgpa, dtype: float64
```

0.2 Inter Quartile Range(IQR) Code:

[9]:
```
plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.histplot(df['cgpa'])

plt.subplot(1,2,2)
sns.histplot(df['placement_exam_marks'])

plt.show()
```

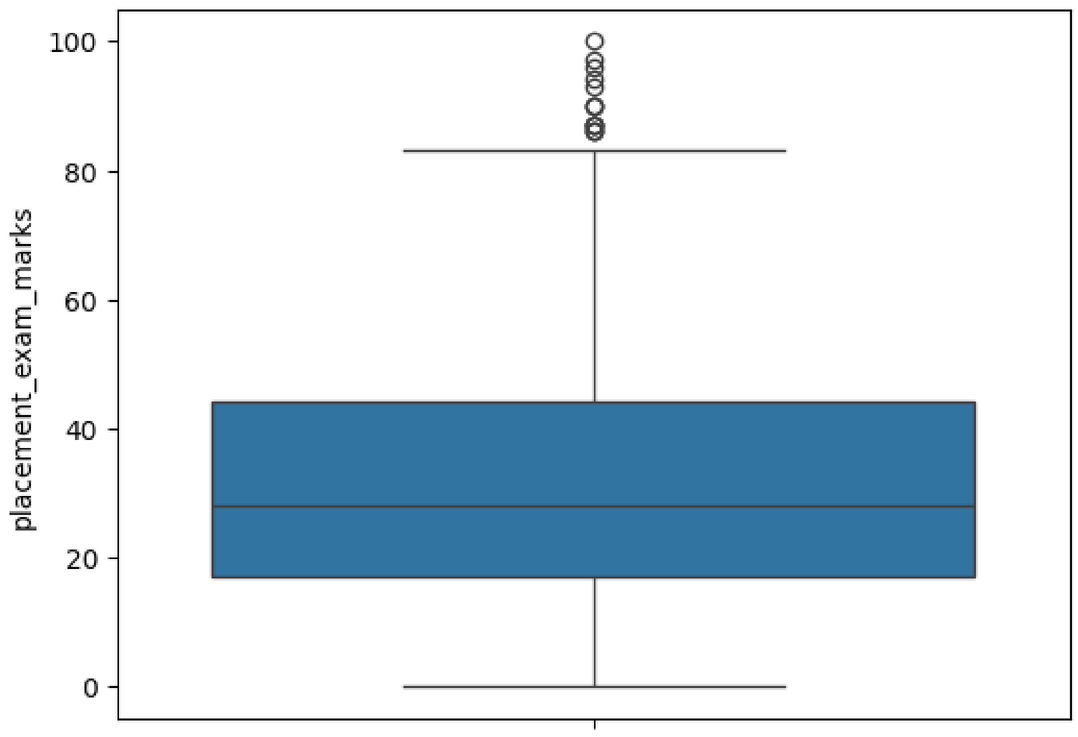


[10]:
```
df['placement_exam_marks'].describe()
```

[10]:
```
count    1000.000000
mean       32.225000
std        19.130822
min         0.000000
25%        17.000000
50%        28.000000
75%        44.000000
max       100.000000
Name: placement_exam_marks, dtype: float64
```

[11]:
```
sns.boxplot(df['placement_exam_marks'])
```

[11]:
```
<Axes: ylabel='placement_exam_marks'>
```



[13]:
```
# Finding the IQR
percentile25 = df['placement_exam_marks'].quantile(0.25)
percentile75 = df['placement_exam_marks'].quantile(0.75)
print(percentile25)
print(percentile75)
```

```
17.0
44.0
```

[14]:
```
iqr = percentile75 - percentile25
iqr
```

[14]:
```
27.0
```

[15]:
```
upper_limit = percentile75 + 1.5 * iqr
lower_limit = percentile25 - 1.5 * iqr
print(upper_limit)
print(lower_limit)
```

```
84.5
-23.5
```

[16]:
```
df[df['placement_exam_marks'] > upper_limit]
```

[16]:
```
     cgpa  placement_exam_marks  placed
9    7.75                  94.0       1
40   6.60                  86.0       1
61   7.51                  86.0       0
134  6.33                  93.0       0
162  7.80                  90.0       0
283  7.09                  87.0       0
290  8.38                  87.0       0
311  6.97                  87.0       1
324  6.64                  90.0       0
630  6.56                  96.0       1
685  6.05                  87.0       1
730  6.14                  90.0       1
771  7.31                  86.0       1
846  6.99                  97.0       0
917  5.95                 100.0       0
```

[17]:
```
df[df['placement_exam_marks'] < lower_limit]
```

[17]:
```
Empty DataFrame
Columns: [cgpa, placement_exam_marks, placed]
Index: []
```

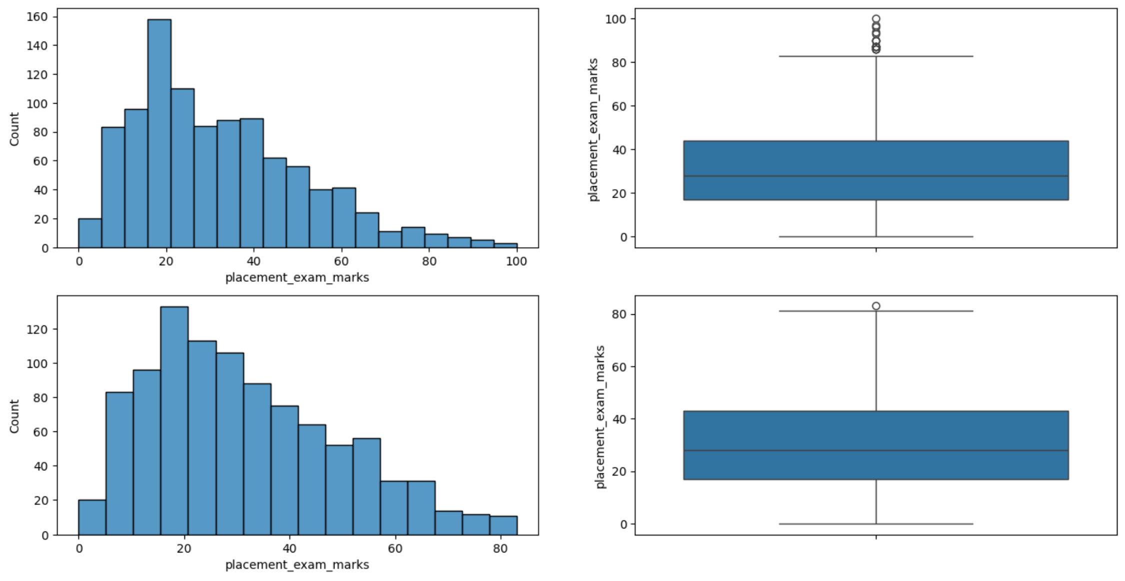
0.3 Trimming

[32]:
```
new_df = df[df['placement_exam_marks'] < upper_limit]
```

[19]:
```
new_df.shape
```

[19]:
```
(985, 3)
```

[20]:
```
# Comparing

plt.figure(figsize=(16,8))
plt.subplot(2,2,1)
sns.histplot(df['placement_exam_marks'])

plt.subplot(2,2,2)
sns.boxplot(df['placement_exam_marks'])

plt.subplot(2,2,3)
sns.histplot(new_df['placement_exam_marks'])

plt.subplot(2,2,4)
sns.boxplot(new_df['placement_exam_marks'])

plt.show()
```



0.4 Capping

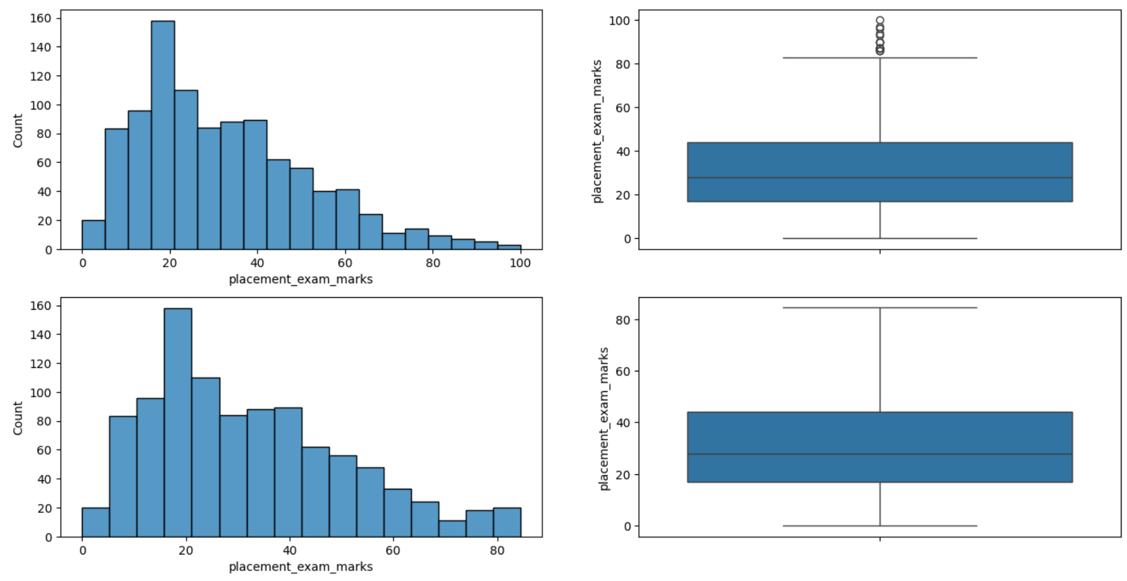
[21]:
```
new_df_cap = df.copy()
#np.where(condtion,true,false)
new_df_cap['placement_exam_marks'] = np.where(
    new_df_cap['placement_exam_marks'] > upper_limit,
    upper_limit,
    np.where(
        new_df_cap['placement_exam_marks'] < lower_limit,
        lower_limit,
        new_df_cap['placement_exam_marks']
    )
)
```

[22]:
```
new_df_cap.shape
```

[22]:
```
(1000, 3)
```

[23]:
```
# Comparing

plt.figure(figsize=(16,8))
plt.subplot(2,2,1)
sns.histplot(df['placement_exam_marks'])

plt.subplot(2,2,2)
sns.boxplot(df['placement_exam_marks'])

plt.subplot(2,2,3)
```

```
sns.histplot(new_df_cap['placement_exam_marks'])

plt.subplot(2,2,4)
sns.boxplot(new_df_cap['placement_exam_marks'])

plt.show()
```



References:

[1]. Hands On Machine Learning with Scikit Learn, Keras and TensorFlow pg.147-pg.156

[2]. https://www.youtube.com/watch?v=Lln1PKgGr_M&list=PLKnIA16_Rmvbr7zKYQuBfsVkjoLcJgxHH&ind