Understanding Outliers: Impact, Detection, and Remedies

Introduction

Outliers are data points that significantly deviate from the average or typical values within a dataset. These observations, though rare, can greatly influence statistical analyses and machine learning models if not properly addressed. In this blog, we will explore what outliers are, why they can be dangerous, their effects on machine learning models, and effective methods to detect and treat outliers.

What are Outliers?

Outliers are data points that lie far away from the majority of the data, either above or below the expected range. They can arise due to various reasons, such as measurement errors, experimental anomalies, or truly exceptional observations. Outliers can distort statistical analyses, affecting the accuracy and reliability of the results.

When are Outliers Dangerous?

Outliers can be particularly dangerous when they exert a disproportionate influence on the analysis or modeling results. They can skew the statistical measures of central tendency, such as the mean and median, leading to biased estimates. In regression analysis, outliers can significantly affect the slope and intercept of the regression line, distorting the relationship between variables. Outliers can also impact clustering algorithms by affecting the distance metrics and the formation of clusters.

Which machine learning models are more sensitive to outliers, and which ones are less affected by them?

Outliers can have a varying impact on different machine learning models. Some models are more sensitive to outliers, while others are more robust and less affected by extreme values. Here's a breakdown:

Models Affected by Outliers:

a. Linear Regression: Linear regression models can be significantly affected by outliers, as they heavily rely on minimizing the squared differences between predicted and actual values. Outliers can introduce a high level of error and result in biased coefficient estimates.

b. K-means Clustering: K-means clustering can be influenced by outliers, as they can distort the distances between data points and the formation of clusters. Outliers can be mistakenly assigned to clusters or result in the creation of separate clusters.

c. Support Vector Machines (SVM): SVM models can be sensitive to outliers, especially in cases where the margin between classes is narrow. Outliers lying near the decision boundary can affect the placement of the boundary, leading to misclassifications.

Models Less Affected by Outliers:

a. Decision Trees: Decision trees are relatively robust to outliers. The hierarchical splitting process focuses on finding optimal splits based on impurity measures, such as Gini index or entropy, rather than on the exact values of individual data points.

b. Random Forests: Random forests, which are an ensemble of decision trees, are also less affected by outliers. The averaging of multiple trees helps mitigate the impact of outliers, as the errors introduced by outliers tend to average out.

c. Naive Bayes: Naive Bayes models are generally less sensitive to outliers since they make strong independence assumptions between features. Outliers might not disrupt the probabilistic calculations as much as in other models.

How can outliers be treated?

Outliers can be addressed using various techniques depending on the specific goals of the analysis and the nature of the data. It is important to choose an appropriate approach to maintain the integrity of the data while mitigating the impact of outliers.

Trimming: Trimming involves removing a certain percentage of extreme values from both ends of the data distribution. This approach discards the outliers entirely, which can reduce their influence on statistical analyses. By trimming the dataset, the extreme values are eliminated, and the analysis focuses on the majority of the data. However, it's essential to carefully consider the percentage to be trimmed to avoid removing too much data.

Capping: Capping, also known as Winsorization, sets a predefined threshold beyond which the outlier values are replaced with the nearest acceptable value within that threshold. Capping prevents the complete removal of outliers and instead modifies their values to align them with the nearby observations. This approach helps control the impact of outliers while retaining their presence in the dataset. Capping can be done symmetrically, by capping both lower and upper extremes, or asymmetrically if there is a specific directionality to the outliers.

What are the approaches used for outlier detection?

Detecting outliers in different types of distributions is crucial for accurate data analysis. Several methods can be employed based on the distribution characteristics. Let's explore the techniques and formulas used to detect outliers in various distributions:

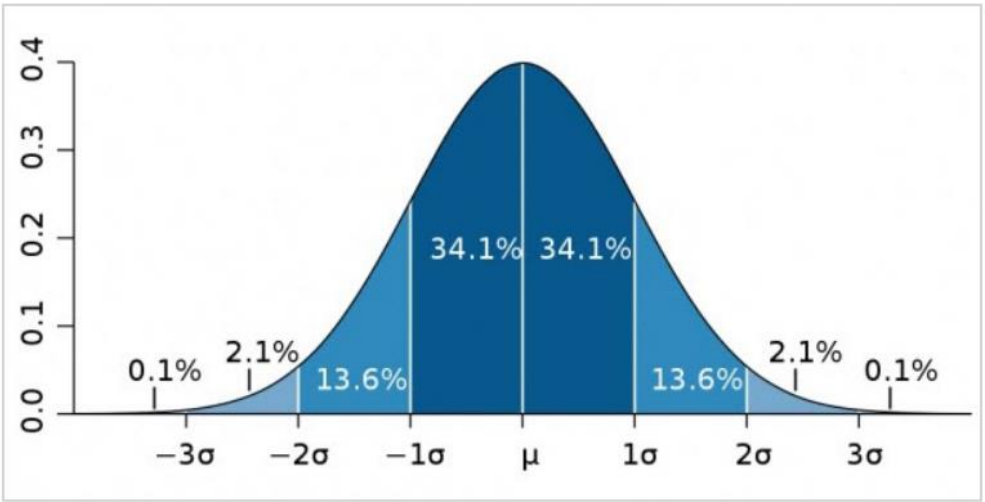
Normal Distribution:

Normal distribution, also known as the Gaussian distribution or bell curve, is a statistical concept that describes a specific pattern of data distribution. In a normal distribution, the data is symmetrically distributed around the mean, creating a characteristic bell-shaped curve.

In a dataset that follows a normal distribution, the following rules apply to the bell curve:

1. The mean of the data represents the center of the bell curve and is also the highest point on the curve.
2. Approximately 68.2% of the data points fall within one standard deviation of the mean, covering the range from (Mean - Standard Deviation) to (Mean + Standard Deviation).
3. Roughly 95.5% of the data points lie within two standard deviations of the mean, encompassing the range from (Mean - 2 * Standard Deviation) to (Mean + 2 * Standard Deviation).
4. Almost 99.7% of the data points fall within three standard deviations of the mean, covering the range from (Mean - 3 * Standard Deviation) to (Mean + 3 * Standard Deviation).

These rules, often referred to as the empirical rule or the 68-95-99.7 rule, provide a guideline for understanding the distribution of data in a normal distribution and help identify the expected range of values around the mean.



AI Insights

In a normal distribution, outliers can be detected using the z-score approach. The formula for calculating the z-score is: z = (x - μ) / σ Here, z represents the z-score, x is the data point, μ is the mean, and σ is the standard deviation. Typically, a z-score greater than a certain threshold (e.g., ±2 or ±3) indicates an outlier.

Let’s Perform outlier detection and removal using Python code.

Import the necessary libraries and dataset. Here I am using Placement dataset.

```
In [1]:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [2]:
df = pd.read_csv('placement.csv')
```

```
In [3]:
df.sample(5)
```

Out[3]:

	cgpa	placement_exam_marks	placed
269	6.47	16.0	0
300	7.36	52.0	1
842	7.33	14.0	1
746	6.72	25.0	1
119	7.00	39.0	0

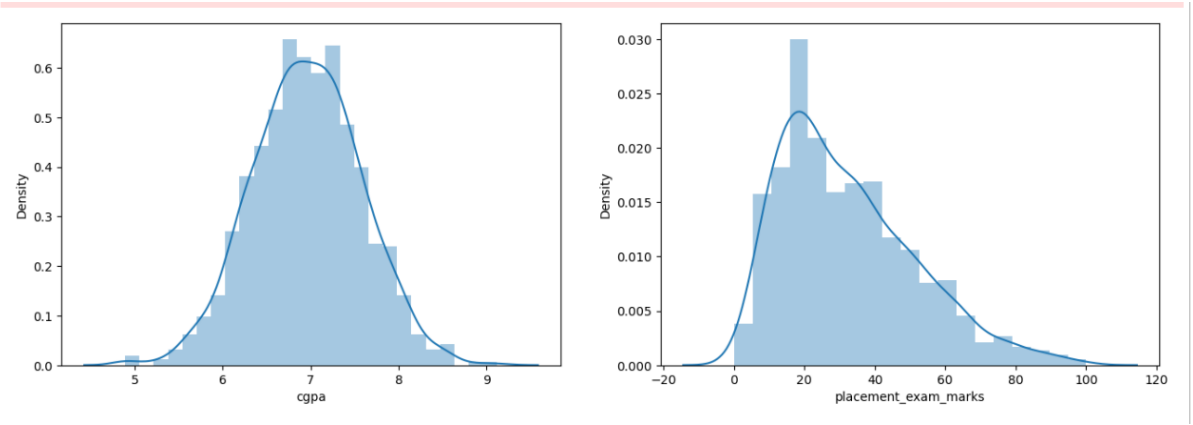
The dataset consists of 1000 rows and 3 columns.

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

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

Create a plot to visualize the distribution of data in the columns.

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

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

plt.show()
```



Upon observing the data, it becomes evident that the "cgpa" column follows a normal distribution, while the "placement_exam_marks" column exhibits right-skewness. As a result, the z-score approach is applicable only to the "cgpa" column.

Calculate the minimum, maximum, mean, and standard deviation for the "cgpa" column.

In [6]:
```
print('The min value in CGPA column', df['cgpa'].min())
print('The max value in CGPA column', df['cgpa'].max())
print('The max value in CGPA column', df['cgpa'].mean())
print('The max value in CGPA column', df['cgpa'].std())
```

```
The min value in CGPA column 4.89
The max value in CGPA column 9.12
The max value in CGPA column 6.96124000000001
The max value in CGPA column 0.6158978751323894
```

Determine the boundary values.

In [7]:
```
#finding boundary value
upper_limit=df['cgpa'].mean() + 3*df['cgpa'].std()
lowest_limit=df['cgpa'].mean() - 3*df['cgpa'].std()
print('highest boundary :',upper_limit)
print('lowest boundary :', lowest_limit)
```

```
highest boundary : 8.808933625397177
lowest boundary : 5.113546374602842
```

The highest boundary is determined to be 8.8089, while the lowest boundary is identified as 5.1135.

AI Insights

Let's identify the outliers in our dataset by considering all values above 8.8 and below 5.11 as outliers.

In [8]:
```
# Finding the outliers

df[(df['cgpa'] > upper_limit) | (df['cgpa'] <lowest_limit)]
```

Out[8]:

	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

.

Upon examining our dataset, we observe that there are three rows with values below the lower boundary and two rows with values above the highest boundary.

To address the outliers, we will employ the trimming approach and eliminate all rows with outliers in our dataset.

In [9]:
```
# Trimming

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

Out[9]:

	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 × 3 columns

In [10]:
```
new_df.shape
```

Out[10]: (995, 3)

AI Insights

After applying the trimming approach, the dataset has been reduced to 995 rows and 3 columns, excluding the rows that contained outliers.

In [11]:
```
# Approach 2

# Calculating the Zscore

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

In [12]:
```
df[(df['cgpa_zscore'] > 3) | (df['cgpa_zscore'] < -3)]
```

Out[12]:

	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

Let's proceed with the second approach, which involves applying the capping method to address the outliers in the dataset.

To handle outliers using the capping method, we will replace all values above the upper limit with the upper limit value and all values below the lower limit with the lower limit value.

In [17]:
```
df['cgpa'] = np.where(
    df['cgpa']>upper_limit,
    upper_limit,
    np.where(
        df['cgpa']<lowest_limit,
        lowest_limit,
        df['cgpa']
    )
)
```

It is observed that the shape of the dataset remains unchanged after implementing the capping method, indicating that the number of rows and columns in the dataset remains the same.

In [18]:
```
df.shape
```

Out[18]: (1000, 4)

Now, let's examine the minimum and maximum values of the dataset after applying the capping method. The maximum value should correspond to the highest boundary, while the minimum value should align with the lowest boundary.

AI Insights

```
In [19]:
1 print('The min value in CGPA column', df['cgpa'].min())
2 print('The max value in CGPA column', df['cgpa'].max())
```

```
The min value in CGPA column 5.113546374602842
The max value in CGPA column 8.808933625397177
```

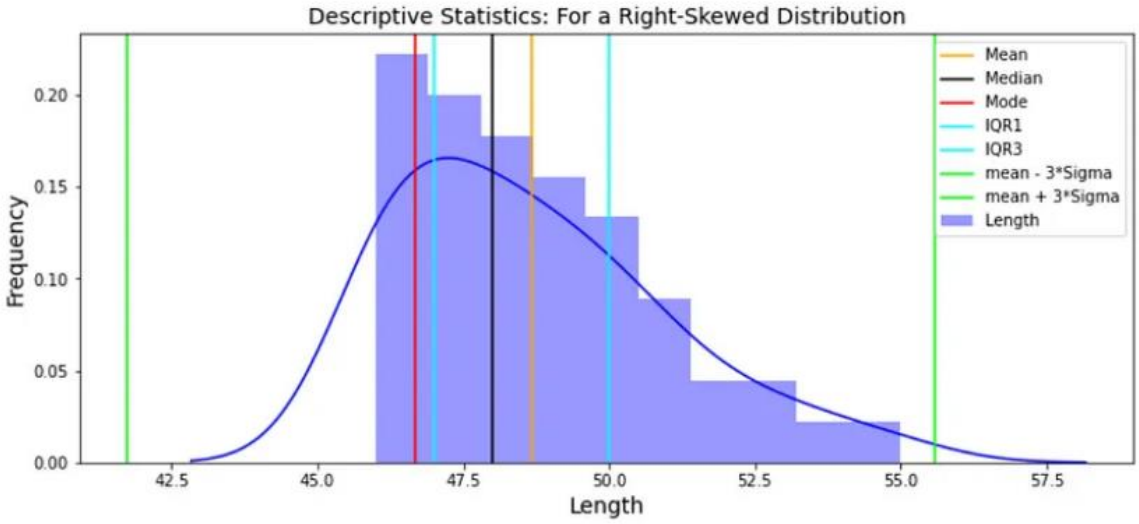
Skewed Distribution (IQR method): A skewed distribution refers to a type of data distribution where the values are not evenly distributed around the mean. In a skewed distribution, the data tends to be concentrated towards one side, resulting in a long tail on either the left or right side of the distribution.

Right-skewed data, also known as positively skewed data, refers to a type of data distribution where the majority of the values are concentrated towards the left (lower values) of the distribution, while a few larger values are present on the right (higher values) side. This results in a long tail extending towards the right.

In a right-skewed distribution:

- The mean is usually greater than the median.
- The median is closer to the lower end of the data range.
- The mode, or the most frequently occurring value, tends to be smaller than the median.

Visually, a right-skewed distribution appears asymmetric, with a stretched or elongated tail on the right-hand side. The bulk of the data is situated towards the left, indicating a prevalence of smaller values, while a few extreme values on the right contribute to the skewness.



AI Insights

To treat outliers in a rightly skewed distribution using the Interquartile Range (IQR) method, follow these steps:

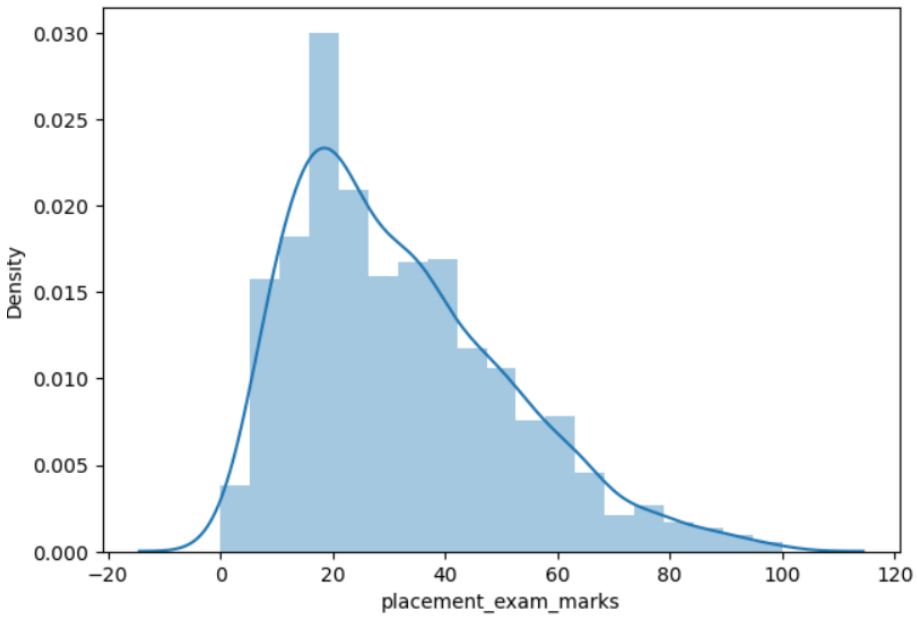
1. Calculate the first quartile (Q1) and third quartile (Q3) of the dataset.
2. Calculate the IQR by subtracting Q1 from Q3, i.e., IQR = Q3 - Q1.
3. Identify the lower limit by subtracting 1.5 times the IQR from Q1, i.e., Lower Limit = Q1 - 1.5 * IQR.
4. Identify the upper limit by adding 1.5 times the IQR to Q3, i.e., Upper Limit = Q3 + 1.5 * IQR.
5. Values below the lower limit or above the upper limit are considered outliers.
6. Treat the outliers by either removing them from the dataset or replacing them with a suitable value.

Let’s Perform outlier detection and removal of rightly skewed data using Python code.

Continuing from the previous scenario, we will work with the same data frame and focus on addressing outliers in the "placement_exam_marks" column, which exhibits a right-skewed distribution.

n [27]:

```
plt.figure(figsize=(16,5))


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

plt.show()
```



AI Insights

Calculate the minimum, maximum, mean, and standard deviation for the "placement_exam_marks" column.

In [6]:
```
print('The min value in placement_exam_marks column', df['placement_exam_marks'].min())
print('The max value in placement_exam_marks column', df['placement_exam_marks'].max())
print('The max value in placement_exam_marks column', df['placement_exam_marks'].mean())
print('The max value in placement_exam_marks column', df['placement_exam_marks'].std())
```

```
The min value in placement_exam_marks column 0.0
The max value in placement_exam_marks column 100.0
The max value in placement_exam_marks column 32.225
The max value in placement_exam_marks column 19.13082233892108
```

We will proceed with the steps outlined earlier to handle outliers in the "placement_exam_marks" column, considering its right-skewed distribution.

Calculate 25th and 75th Percentile ,IQR , upperlimit and lowerlimit.

[7]:
```
percentile25=df['placement_exam_marks'].quantile(0.25)
print(percentile25)
```

```
17.0
```

[8]:
```
percentile75=df['placement_exam_marks'].quantile(0.75)
print(percentile25)
```

```
17.0
```

[9]:
```
iqr= percentile75 - percentile25
print(iqr)
```

```
27.0
```

[10]:
```
upperlim= percentile25 + 1.5*iqr
lowerlim= percentile25 - 1.5*iqr

print('Upper limit is :', upperlim)
print('Lower limit is :', lowerlim)
```

```
Upper limit is : 57.5
Lower limit is : -23.5
```

Let's examine the dataset to identify any values in the "placement_exam_marks" column that exceed the upper and lower limit.

AI Insights

```
[11]: 1 df[df['placement_exam_marks'] > upperlim]
```

Out[11]:

	cgpa	placement_exam_marks	placed
9	7.75	94.0	1
25	6.28	58.0	1
40	6.60	86.0	1
42	7.46	71.0	1
43	7.85	63.0	0
...	...	...	...
966	6.24	72.0	1
967	7.35	59.0	0
987	6.77	62.0	0
994	6.48	63.0	0
996	9.12	65.0	1

114 rows × 3 columns

Upon analysis, it is evident that there are 114 rows in the dataset where the values in the "placement_exam_marks" column exceed the upper limit. However, there are no rows with values below the lower limit in the dataset.

While trimming may not be the ideal option in this case due to the significant loss of 10% of the data, we will still perform trimming as an exercise to understand how it can be implemented. However, it is important to note that capping is generally considered a better approach as it allows us to retain more data and crucial information.

In [17]:
```
#Trimming

new_df= df[(df['placement_exam_marks'] <= upperlim) & (df['placement_exam_marks'] >= lowerlim)]
new_df
```

Out[17]:

	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
...	...	...	...
993	6.73	21.0	1
995	8.87	44.0	1
997	4.89	34.0	0
998	8.62	46.0	1
999	4.90	10.0	1

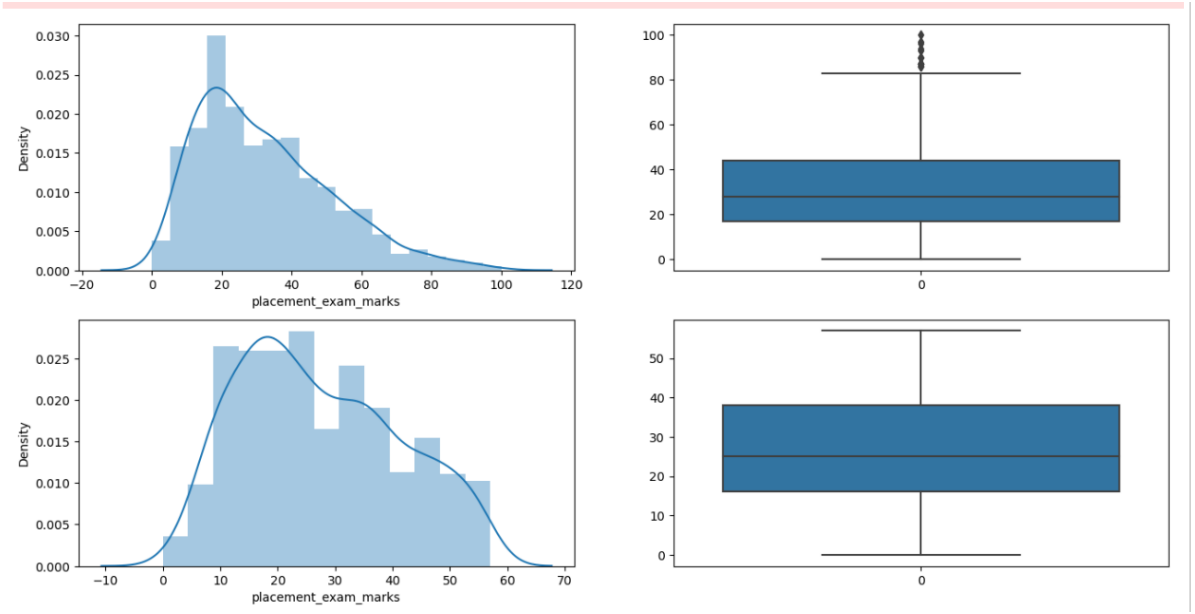
886 rows × 3 columns

Trimming involves the removal of all data points that fall below the lower limit and above the upper limit. After applying the trimming approach, the data frame is reduced to 886 rows and 3 columns, reflecting the removal of the outlier data points.

Let's generate visualizations to compare the plots before and after the removal of outliers.

In [18]:
```
# Comparing the plots

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

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

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

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

plt.show()
```

We can clearly see the outliers are removed.



Considering the suitability of the situation, we will apply the capping method. This involves replacing all values above the upper limit with the upper limit value and all values below the lower limit with the lower limit value. This approach is more appropriate since it ensures that the dataframe shape remains unchanged, and we retain all the important information without any loss.

In [22]:
```
#Capping

cap_df=df.copy()
cap_df['placement_exam_marks']=np.where(df['placement_exam_marks'] > upperlim, upperlim,
                np.where(df['placement_exam_marks'] < lowerlim, lowerlim,
                 df['placement_exam_marks']
                        ))

```

In [23]:
```
cap_df
```

Out[23]:

	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
...	...	...	...
995	8.87	44.0	1
996	9.12	57.5	1
997	4.89	34.0	0
998	8.62	46.0	1
999	4.90	10.0	1

1000 rows × 3 columns

In [25]:
```
print('The min value in placement_exam_marks column', cap_df['placement_exam_marks'].min())
print('The max value in placement_exam_marks column', cap_df['placement_exam_marks'].max())
```

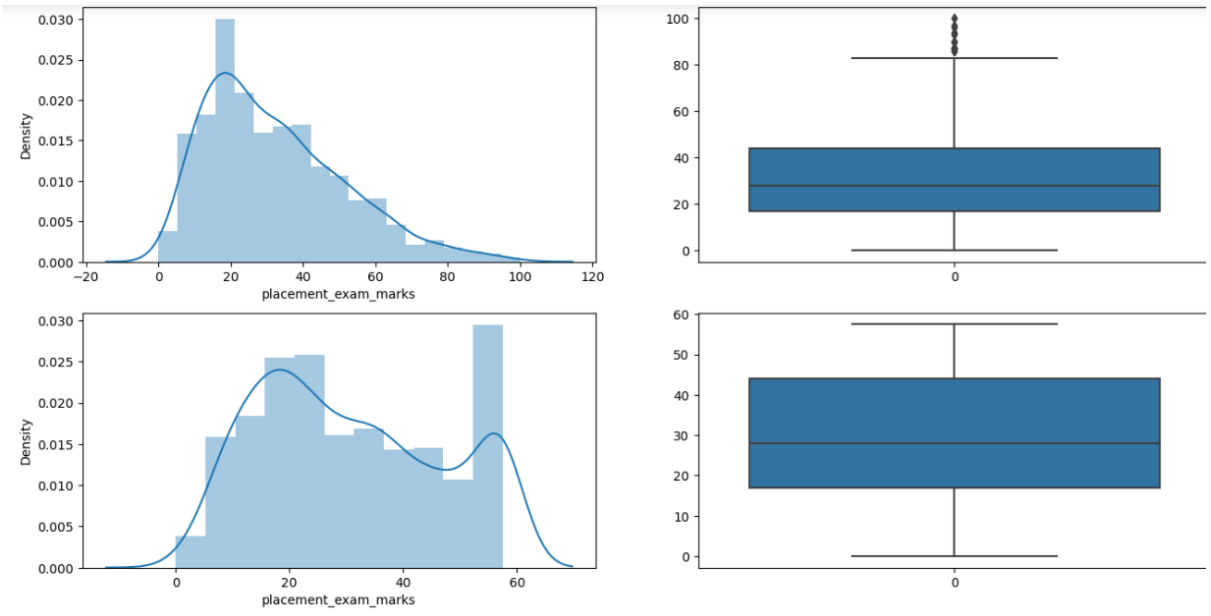
```
The min value in placement_exam_marks column 0.0
The max value in placement_exam_marks column 57.5
```

AI Insights

It is apparent that the maximum value in the dataset corresponds to the upper limit, while the minimum value corresponds to the lower limit. This indicates that all the outliers have been effectively replaced through the capping process.

Let's visually examine the dataset to confirm the removal of outliers and the application of capping by plotting the data before and after the process.

In [26]:
```
plt.figure(figsize=(16,8))
plt.subplot(2,2,1)
sns.distplot(df['placement_exam_marks'])

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

plt.subplot(2,2,3)
sns.distplot(cap_df['placement_exam_marks'])

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

plt.show()
```



AI Insights

Percentile Approach :

The percentile method is a technique used to treat outliers by identifying and capping extreme values based on a specified percentage threshold. It involves calculating the threshold values based on percentiles and replacing any data points that exceed these thresholds with the corresponding threshold values.

The process of using the percentile method to handle outliers typically involves the following steps:

1. Determine the percentage threshold: Select a percentage value that represents the threshold for extreme values. Commonly used thresholds are 95th percentile (5% of data points are considered outliers) or 99th percentile (1% of data points are considered outliers). The choice of threshold depends on the specific dataset and the desired level of outlier removal.
2. Calculate the threshold values: Use the chosen percentile to calculate the upper and lower thresholds. For example, if the 95th percentile is chosen, the upper threshold would be the value below which 95% of the data falls, and the lower threshold would be the value above which 95% of the data falls.
3. Identify and replace outliers: Iterate through the dataset and identify any data points that exceed the upper or lower threshold. Replace these outliers with the corresponding threshold value.

By applying the percentile method, extreme values that lie beyond the selected threshold are effectively replaced with more representative values. This helps mitigate the impact of outliers on the data analysis and modeling process, allowing for more robust and reliable results.

Let’s Perform outlier detection and removal of outliers using Percentile approach in Python code.

We will work with a new dataset called "heightweight" and import the required libraries. Let's determine the shape of the dataframe, which provides information about the number of rows and columns in the dataset.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df = pd.read_csv('weight-height.csv')
```

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

Out[4]:

	Gender	Height	Weight
0	Male	73.847017	241.893563
1	Male	68.781904	162.310473
2	Male	74.110105	212.740856
3	Male	71.730978	220.042470
4	Male	69.881796	206.349801

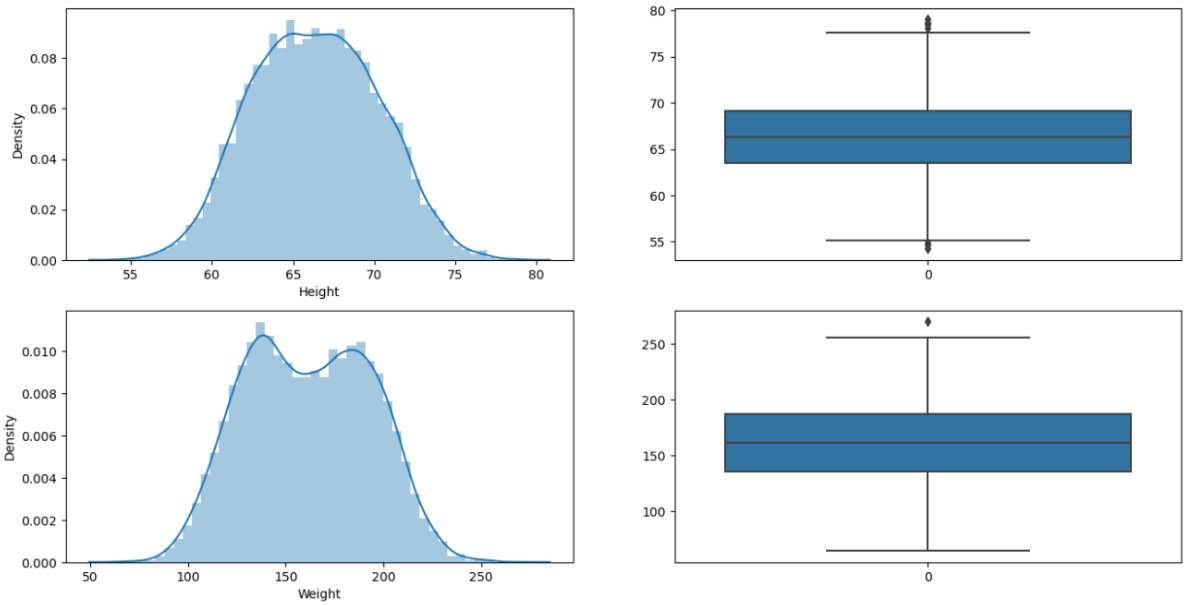
In [5]:
```
df.shape
```

Out[5]:
```
(10000, 3)
```

We will generate visualizations to examine the "height" and "weight" columns and identify any potential outliers present in the dataset.

In [7]:
```
plt.figure(figsize=(16,8))
plt.subplot(2,2,1)
sns.distplot(df['Height'])

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

plt.subplot(2,2,3)
sns.distplot(df['Weight'])

plt.subplot(2,2,4)
sns.boxplot(df['Weight'])

plt.show()
```

AI Insights



It is evident from the visualizations that the "height" column contains a higher number of outliers compared to the "weight" column. Therefore, we will focus our attention on addressing outliers specifically in the "height" column.

To detect outliers in the "height" column, we will utilize the 99th percentile and 1st percentile as the boundary values. Any data points above the 99th percentile or below the 1st percentile will be considered outliers.

In [8]:
```
upper_limit = df['Height'].quantile(0.99)
lower_limit = df['Height'].quantile(0.01)
print('The upper limit', upper_limit)
print('The lower limit', lower_limit)
```

```
The upper limit 74.7857900583366
The lower limit 58.13441158671655
```

Upon analysis, we have identified that there are 100 rows in the dataset with values exceeding the 99th percentile, as well as 100 rows with values falling below the 1 percentile. These rows are considered outliers in the dataset.

AI Insights

In [8]:
```
1 df[(df['Height'] > upper_limit)]
```

Out[8]:

	Gender	Height	Weight
23	Male	75.205974	228.761781
190	Male	76.709835	235.035419
197	Male	75.944460	231.924749
202	Male	75.140821	224.124271
215	Male	74.795375	232.635403
...	...	...	...
4565	Male	75.690384	223.587548
4569	Male	77.547186	242.041173
4701	Male	76.732446	241.686601
4721	Male	75.330847	240.440816
4825	Male	74.975231	214.495489

100 rows × 3 columns

In [9]:
```
1 df[(df['Height'] < lower_limit)]
```

Out[9]:

	Gender	Height	Weight
5026	Female	56.547975	84.872124
5074	Female	56.159458	90.815256
5122	Female	57.103869	93.506316
5123	Female	56.445685	96.640245
5162	Female	57.961936	112.226984
...	...	...	...
9761	Female	56.975279	90.341784
9825	Female	55.979198	85.417534
9895	Female	57.740192	93.652957
9904	Female	57.028857	101.202551
9978	Female	57.375759	114.192209

100 rows × 3 columns

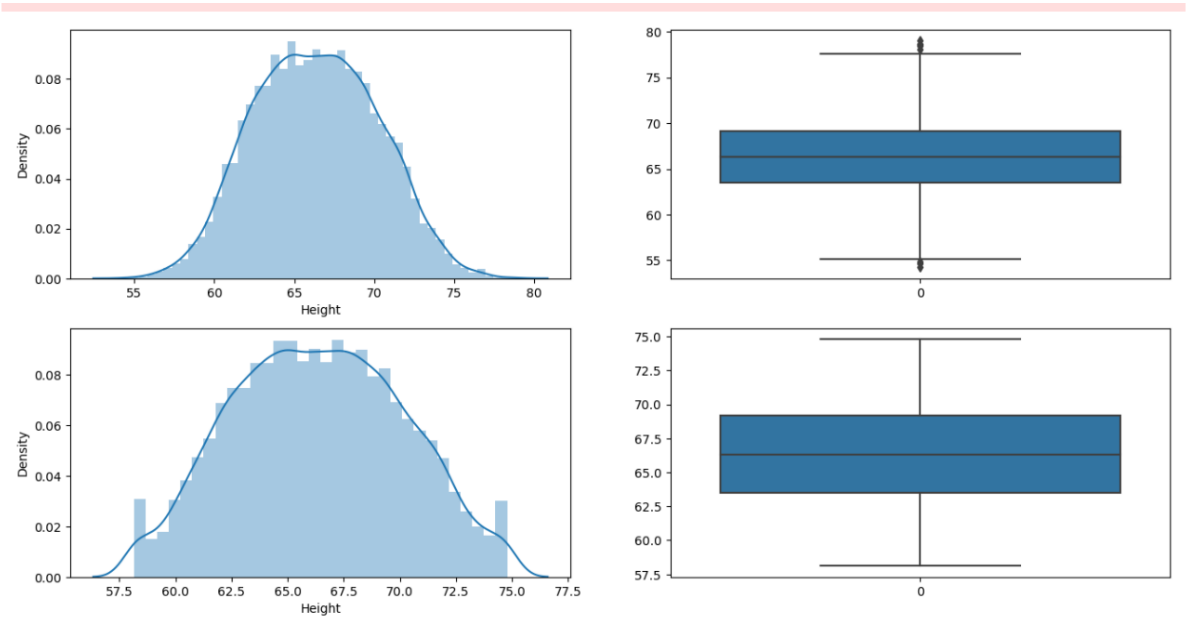
AI Insights

Since we have previously explored the trimming method for handling outliers, I will directly proceed with applying capping to the "height" column. This involves replacing all values above the 99th percentile with the upper limit and all values below the 1 percentile with the lower limit.

In [10]:
```
# Capping
cap_df=df.copy()
cap_df['Height'] = np.where(cap_df['Height'] >= upper_limit,
        upper_limit,
        np.where(cap_df['Height'] <= lower_limit,
        lower_limit,
        cap_df['Height']))
```

Let's generate visualizations to compare the plots before and after the removal of outliers.

In [11]:
```
plt.figure(figsize=(16,8))
plt.subplot(2,2,1)
sns.distplot(df['Height'])

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

plt.subplot(2,2,3)
sns.distplot(cap_df['Height'])

plt.subplot(2,2,4)
sns.boxplot(cap_df['Height'])

plt.show()
```



AI Insights

Conclusion

Outliers can significantly impact statistical analyses and machine learning models, distorting results and hindering accurate predictions. Understanding what outliers are, when they are dangerous, and how to handle them is crucial for robust data analysis. By employing techniques such as z-score, IQR, and percentile approaches and further trimming and capping, outliers can be effectively treated, improving the quality and reliability of the analysis and enhancing the performance of machine learning models.

The code can be found here

https://github.com/taherafirdose/100-days-of-Machine-Learning/tree/master/Handling%20Outliers