1. Discuss the need of multiple imputation and its practical difficulties involved when it is applied to MNAR type dataset.

**Multiple imputation**(MI) is a way to deal with [nonresponse bias](https://www.statisticshowto.com/what-is-bias/#Nonresponse) — missing research data that happens when people fail to respond to a survey. The technique allows you to analyze incomplete data with regular data analysis tools like a [t-test](https://www.statisticshowto.com/probability-and-statistics/t-test/) or [ANOVA](https://www.statisticshowto.com/probability-and-statistics/hypothesis-testing/anova/). Impute means to “fill in.” With singular imputation methods, the [mean](https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/mean-median-mode/#mean), [median](https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/mean-median-mode/#median), or some other statistic is used to impute the missing values. However, using single values carries with it a level of [uncertainty](https://www.statisticshowto.com/uncertainty-in-statistics/)about which values to impute. **Multiple imputation narrows uncertainty about missing values by calculating several different options (“imputations”).** Several versions of the same data set are created, which are then combined to make the “best” values.

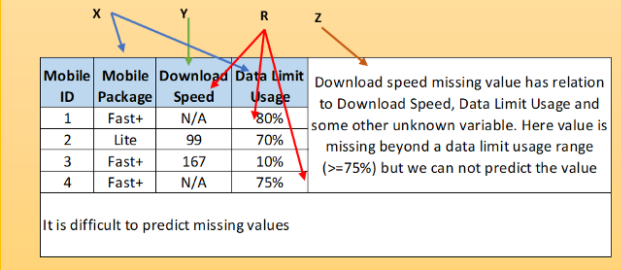
## **Advantages of Multiple Imputation**

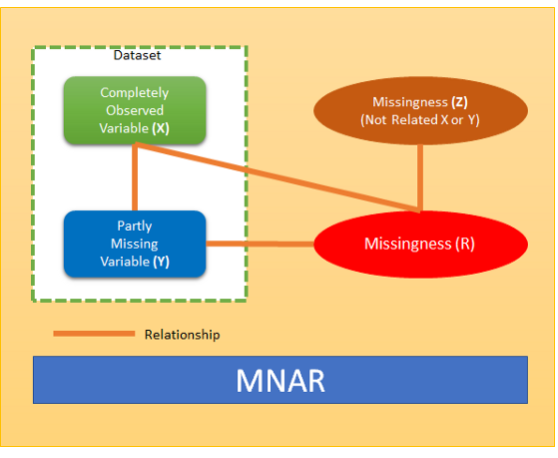
Used correctly, MI can:

* **Reduce**[**bias**](https://www.statisticshowto.com/what-is-bias/). “Bias” refers to errors that creep into your analysis.
* **Improve**[**validity**](https://www.statisticshowto.com/reliability-validity-definitions-examples/)**.** Validity simply means that a test or instrument is accurately measuring what it’s supposed to. For example, when you create a test or questionnaire for depression, you want the questions to actually measure depression and not something else (like anxiety).
* **Increase**[**precision**](https://www.statisticshowto.com/accuracy-and-precision/)**.** Precision is how close two or more measurements are to each other.
* Result in [**robust**](https://www.statisticshowto.com/robust-statistics/)**statistics**, which are resistant to [outliers](https://www.statisticshowto.com/statistics-basics/find-outliers/)(very high or very low data points).

# **Missing not at Random (MNAR)**

If the data characters do not meet those of MCAR or MAR, they fall into the category of missing not at random (MNAR). When data are **missing, not at random**, the missingness is specifically related to what is missing, e.g. a person does not attend a drug test because the person took drugs the night before. A person did not take an English proficiency test due to his poor English language skill. The cases of MNAR data are problematic. The only way to obtain an unbiased estimate of the parameters in such a case is to model the missing data, but that requires proper understanding and domain knowledge of the missing variable. The model may then be incorporated into a more complex one for estimating the missing values. A pictorial view of MNAR is below where missingness directly relates**to variable Y**. It can have other relationships (X & Z).





1. Explain in detail any four techniques used to identify outliers in a dataset with an example.

## **What is an outlier?**

An outlier is a piece of data that is an abnormal distance from other points. In other words, it’s data that lies **outside the other values** in the set. If you had Pinocchio in a class of children, the length of his nose compared to the other children would be an outlier.  
In this set of random numbers, 1 and 201 are outliers:  
1, 99, 100, 101, 103, 109, 110, 201  
“1” is an extremely low value and “201” is an extremely high value.

## **Types of outliers**

Outliers can be of two kinds: **univariate** and **multivariate**. Univariate outliers can be found when looking at a distribution of values in a single feature space. Multivariate outliers can be found in a n-dimensional space (of n-features). Looking at distributions in n-dimensional spaces can be very difficult for the human brain, that is why we need to train a model to do it for us.

Detecting outliers

Detecting outliers is of major importance for almost any quantitative discipline (ie: Physics, Economy, Finance, Machine Learning, Cyber Security). In machine learning and in any quantitative discipline the quality of data is as important as the quality of a prediction or classification model

Some of the most popular methods for outlier detection are:

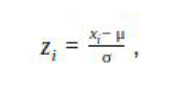
* Z-Score or Extreme Value Analysis (parametric)
* Probabilistic and Statistical Modeling (parametric)
* Linear Regression Models (PCA, LMS)
* Proximity Based Models (non-parametric)
* Information Theory Models
* High Dimensional Outlier Detection Methods (high dimensional sparse data)

**Four Outlier Detection Techniques**

**Z-Score**

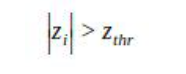
Z-score is a parametric outlier detection method in a one or low dimensional feature space.

This technique assumes a Gaussian distribution of the data. The outliers are the data points that are in the tails of the distribution and therefore far from the mean. How far depends on a set threshold zthr for the normalized data points zi calculated with the formula:

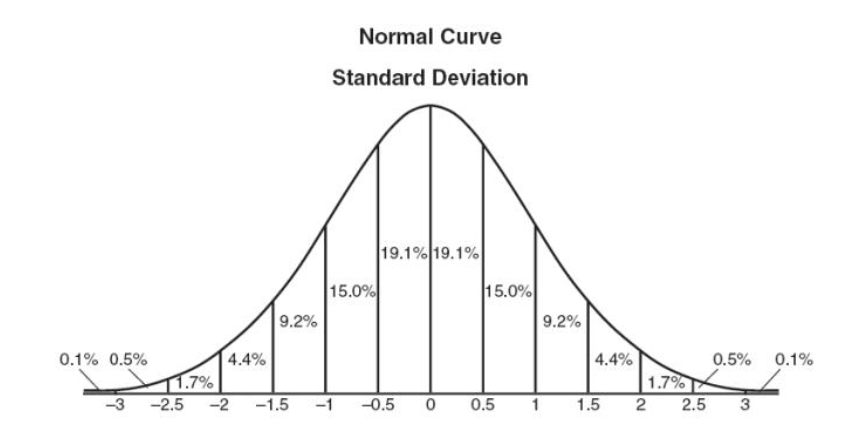


where xi is a data point, μ is the mean of all xi and is the standard deviation of all xi.

An outlier is then a normalized data point which has an absolute value greater than zthr. That is



Commonly used zthr values are 2.5, 3.0 and 3.5.



**DBSCAN**

This technique is based on the [DBSCAN](https://en.wikipedia.org/wiki/DBSCAN)clustering method. DBSCAN is a non-parametric, density based outlier detection method in a one or multi dimensional feature space.

In the DBSCAN clustering technique, all data points are defined either as *Core Points*, *Border Points* or *Noise Points*.

Dbscan then defines different classes of points:

* **Core point**: **A** is a core point if its neighborhood (defined by ɛ) contains at least the same number or more points than the parameter MinPts.
* **Border point**: **C** is a border point that lies in a cluster and its neighborhood does not contain more points than MinPts, but it is still ‘density reachable’ by other points in the cluster.
* **Outlier**: **N** is an outlier point that lies in no cluster and it is not ‘density reachable’ nor ‘density connected’ to any other point. Thus this point will have “his own cluster”

Outlier detection thus depends on the required number of neighbors *MinPts*, the distance ℇ and the selected distance measure, like Euclidean or Manhattan.

Two points p and q are density-connected if there is a point **o** such that both **p** and **q** are density-reachable from **o**. Density-connectedness is symmetric.

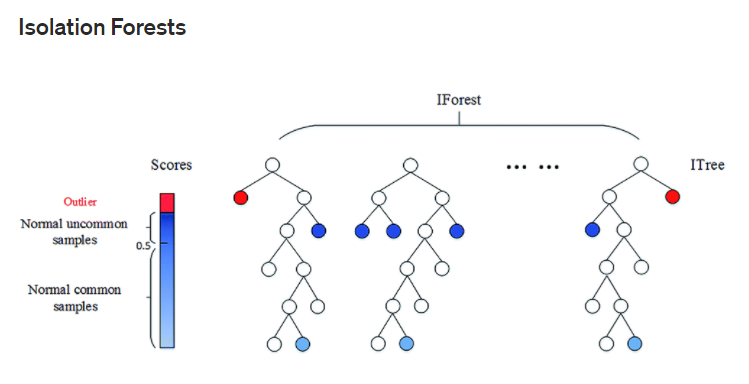
A cluster satisfies two properties:

* All points within the cluster are mutually density-connected.
* If a point is density-reachable from any point of the cluster, it is part of the cluster as well.

The complexity of dbscan is of O(n log n), it and effective method with medium sized data sets. Feeding data to the model is easy when using Scikit learn’s implementation. After fitting dbscan to the data clusters can be extracted and each sample is assigned to a cluster. Dbscan estimates the number of clusters by itself, there is no need to specify the number of desired clusters, it is an unsupervised machine learning model.

**Isolation Forest**

This is a non-parametric method for large datasets in a one or multi dimensional feature space.



An important concept in this method is the isolation number.

The isolation number is the number of splits needed to isolate a data point. This number of splits is ascertained by following these steps:

* A point “a” to isolate is selected randomly.
* A random data point “b” is selected that is between the minimum and maximum value and different from “a”.
* If the value of “b” is lower than the value of “a”, the value of “b” becomes the new lower limit.
* If the value of “b” is greater than the value of “a”, the value of “b” becomes the new upper limit.
* This procedure is repeated as long as there are data points other than “a” between the upper and the lower limit.
* It requires fewer splits to isolate an outlier than it does to isolate a non-outlier, i.e. an outlier has a lower isolation number in comparison to a non-outlier point. A data point is therefore defined as an outlier if its isolation number is lower than the threshold.
* The threshold is defined based on the estimated percentage of outliers in the data, which is the starting point of this outlier detection algorithm.

**Numeric Outlier**

This is the simplest, nonparametric outlier detection method in a one dimensional feature space. Here outliers are calculated by means of the *IQR* (InterQuartile Range).

The first and the third [quartile](https://en.wikipedia.org/wiki/Quartile)(*Q1, Q3*) are calculated. An outlier is then a data point xi that lies outside the interquartile range. That is:

Equation

Using the interquartile multiplier value *k*=1.5, the range limits are the typical upper and lower whiskers of a box plot

## **Conclusions:**

**Z-Score pros:**

* It is a very effective method if you can describe the values in the feature space with a gaussian distribution. (Parametric)
* The implementation is very easy using pandas and scipy.stats libraries.

**Z-Score cons:**

* It is only convenient to use in a low dimensional feature space, in a small to medium sized dataset.
* Is not recommended when distributions can not be assumed to be parametric.

**Dbscan pros:**

* It is a super effective method when the distribution of values in the feature space can not be assumed.
* Works well if the feature space for searching outliers is multidimensional (ie. 3 or more dimensions)
* Sci-kit learn’s implementation is easy to use and the documentation is superb.
* Visualizing the results is easy and the method itself is very intuitive.

**Dbscan cons:**

* The values in the feature space need to be scaled accordingly.
* Selecting the optimal parameters eps, MinPts and metric can be difficult since it is very sensitive to any of the three params.
* It is an unsupervised model and needs to be re-calibrated each time a new batch of data is analyzed.
* It can predict once calibrated but is strongly not recommended.

**Isolation Forest pros:**

* There is no need of scaling the values in the feature space.
* It is an effective method when value distributions can not be assumed.
* It has few parameters, this makes this method fairly robust and easy to optimize.
* Scikit-Learn’s implementation is easy to use and the documentation is superb.

**Isolation Forest cons:**

* The Python implementation exists only in the development version of Sklearn.
* Visualizing results is complicated.
* If not correctly optimized, training time can be very long and computationally expensive.

This article was published as a part of the [*Data Science Blogathon*](https://datahack.analyticsvidhya.com/contest/data-science-blogathon-8/)

## **Introduction**

In my previous article, I talk about the theoretical concepts about outliers and trying to find the answer to the question: **“When we have to drop outliers and when to keep outliers?”**.

To gain a better understanding of this article, firstly you have to read that [**article**](https://www.analyticsvidhya.com/blog/2021/05/why-you-shouldnt-just-delete-outliers/) and then proceed with this so that you have a clear idea about the outlier analysis in Data Science Projects.

In this article, we will try to give the answer to the following questions along with the **Python**implementation,

**How to treat outliers?**

**How to detect outliers?**

**What are the techniques for outlier detection and removal?**

## **Let’s get started**

### How to treat outliers?

**Trimming:** It excludes the outlier values from our analysis. By applying this technique our data becomes thin when there are more outliers present in the dataset. Its main advantage is its **fastest**nature.

**Capping:**In this technique, wecap our outliers data and make the limit i.e, above a particular value or less than that value, all the values will be considered as outliers, and the number of outliers in the dataset gives that capping number.

**For Example,** if you’re working on the income feature, you might find that people above a certain income level behave in the same way as those with a lower income. In this case, you can cap the income value at a level that keeps that intact and accordingly treat the outliers.

**Treat outliers as a missing value:**Byassuming outliers as the missing observations, treat them accordingly i.e, same as those of missing values.

You can refer to the missing value article [here](https://www.analyticsvidhya.com/blog/2021/04/how-to-handle-missing-values-of-categorical-variables/)

**Discretization:** In this technique, by making the groups we include the outliers in a particular group and force them to behave in the same manner as those of other points in that group. This technique is also known as **Binning**.

You can learn more about discretization [**here**](https://www.analyticsvidhya.com/blog/2021/05/complete-guide-on-encode-numerical-features-in-machine-learning/).

### How to detect outliers?

**For Normal distributions:**Use empirical relations of Normal distribution.

– The data points which fall below ***mean-3\*(sigma)***or above ***mean+3\*(sigma)***are outliers.

where mean and sigma are the **average value** and **standard deviation** of a particular column.

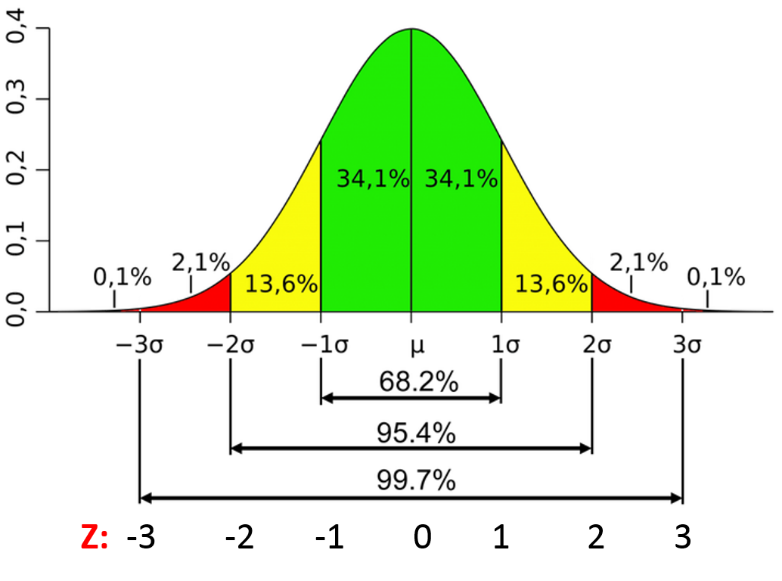


Fig. Characteristics of a Normal Distribution

**Image Source:**[link](https://www.google.com/url?sa=i&url=https%3A%2F%2Fsphweb.bumc.bu.edu%2Fotlt%2FMPH-Modules%2FPH717-QuantCore%2FPH717-Module6-RandomError%2FPH717-Module6-RandomError5.html&psig=AOvVaw1eKfXnP7Ru-TD4iOy9goG2&ust=1621166587664000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCOCY0qjSy_ACFQAAAAAdAAAAABAD)

**For Skewed distributions:** Use Inter-Quartile Range (IQR) proximity rule.

– The data points which fall below **Q1 – 1.5 IQR**or above**Q3 + 1.5 IQR** are outliers.

where Q1 and Q3 are the **25th** and **75th percentile** of the dataset respectively, and IQR represents the inter-quartile range and given by Q3 – Q1.

Fig. IQR to detect outliers

**Image Source:**[link](https://www.google.com/url?sa=i&url=https%3A%2F%2Fnaysan.ca%2F2020%2F06%2F28%2Finterquartile-range-iqr-to-detect-outliers%2F&psig=AOvVaw0ZWFP0wK-QTvhTrcOxkjOs&ust=1621166715671000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCIDp4ePSy_ACFQAAAAAdAAAAABAP)

**For Other distributions:**Usepercentile-based approach.

**For Example,**Data points that are far from 99% percentile and less than 1 percentile are considered an outlier.

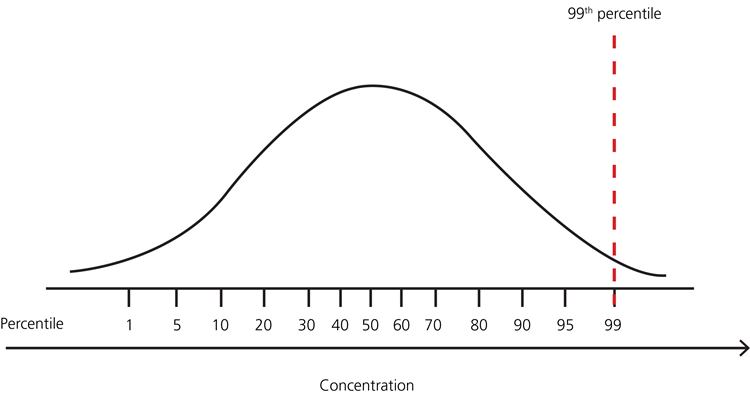


Fig. Percentile representation

**Image Source:**[link](https://www.google.com/url?sa=i&url=https%3A%2F%2Facutecaretesting.org%2Fen%2Farticles%2Freference-intervals-and-percentiles-implications-for-the-healthy-patient&psig=AOvVaw2SFi0dyL_MlQrD91M8NwPy&ust=1621166884005000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCNjJ2LXTy_ACFQAAAAAdAAAAABAJ)

### Techniques for outlier detection and removal:

**Z-score treatment :**

**Assumption**– The features are normally or approximately normally distributed.

**Step-1: Importing Necessary Dependencies**

import numpy as np

import pandas as pd

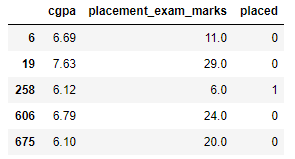
import matplotlib.pyplot as plt

import seaborn as sns

**Step-2: Read and Load the Dataset**

df = pd.read\_csv('placement.csv')

df.sample(5)



**Step-3: Plot the Distribution plots for the features**

import warnings

warnings.filterwarnings('ignore')

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

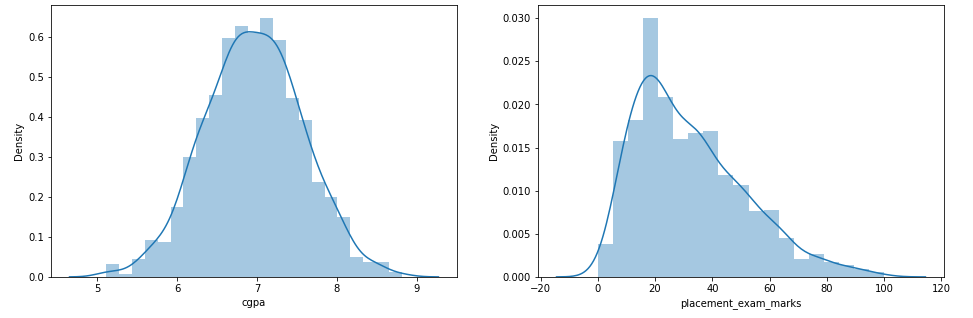
plt.subplot(1,2,1)

sns.distplot(df['cgpa'])

plt.subplot(1,2,2)

sns.distplot(df['placement\_exam\_marks'])

plt.show()



**Step-4: Finding the Boundary Values**

print("Highest allowed",df['cgpa'].mean() + 3\*df['cgpa'].std())

print("Lowest allowed",df['cgpa'].mean() - 3\*df['cgpa'].std())

**Output:**

Highest allowed 8.808933625397177

Lowest allowed 5.113546374602842

**Step-5: Finding the Outliers**

df[(df['cgpa'] > 8.80) | (df['cgpa'] < 5.11)]

**Step-6: Trimming of Outliers**

new\_df = df[(df['cgpa'] < 8.80) & (df['cgpa'] > 5.11)]

new\_df

**Step-7: Capping on Outliers**

upper\_limit = df['cgpa'].mean() + 3\*df['cgpa'].std()

lower\_limit = df['cgpa'].mean() - 3\*df['cgpa'].std()

**Step-8: Now, apply the Capping**

df['cgpa'] = np.where(

df['cgpa']>upper\_limit,

upper\_limit,

np.where(

df['cgpa']<lower\_limit,

lower\_limit,

df['cgpa']

)

)

**Step-9: Now see the statistics using “Describe” Function**

df['cgpa'].describe()

**Output:**

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

***This completes our Z-score based technique!***

**IQR based filtering :**

Used when our data distribution is skewed.

**Step-1: Import necessary dependencies**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

**Step-2: Read and Load the Dataset**

df = pd.read\_csv('placement.csv')

df.head()

**Step-3: Plot the distribution plot for the features**

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

plt.subplot(1,2,1)

sns.distplot(df['cgpa'])

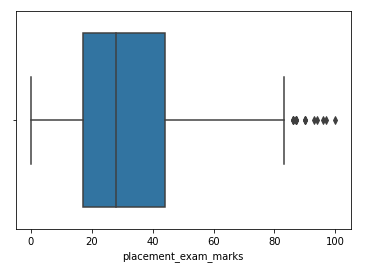
plt.subplot(1,2,2)

sns.distplot(df['placement\_exam\_marks'])

plt.show()

**Step-4: Form a Box-plot for the skewed feature**

sns.boxplot(df['placement\_exam\_marks'])



**Step-5: Finding the IQR**

percentile25 = df['placement\_exam\_marks'].quantile(0.25)

percentile75 = df['placement\_exam\_marks'].quantile(0.75)

**Step-6: Finding upper and lower limit**

upper\_limit = percentile75 + 1.5 \* iqr

lower\_limit = percentile25 - 1.5 \* iqr

**Step-7: Finding Outliers**

df[df['placement\_exam\_marks'] > upper\_limit]

df[df['placement\_exam\_marks'] < lower\_limit]

**Step-8: Trimming**

new\_df = df[df['placement\_exam\_marks'] < upper\_limit]

new\_df.shape

**Step-9: Compare the plots after trimming**

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'])

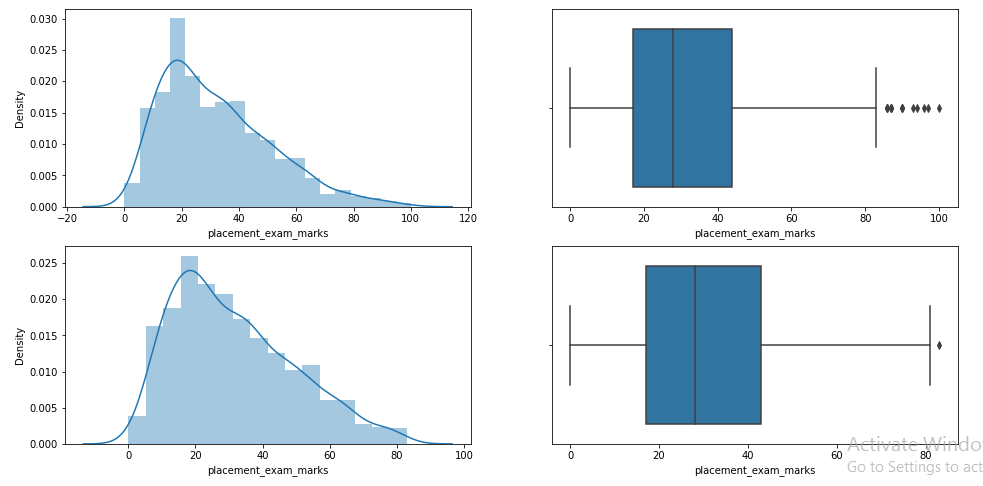
plt.subplot(2,2,3)

sns.distplot(new\_df['placement\_exam\_marks'])

plt.subplot(2,2,4)

sns.boxplot(new\_df['placement\_exam\_marks'])

plt.show()



**Step-10: Capping**

new\_df\_cap = df.copy()

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']

)

)

**Step-11: Compare the plots after capping**

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'])

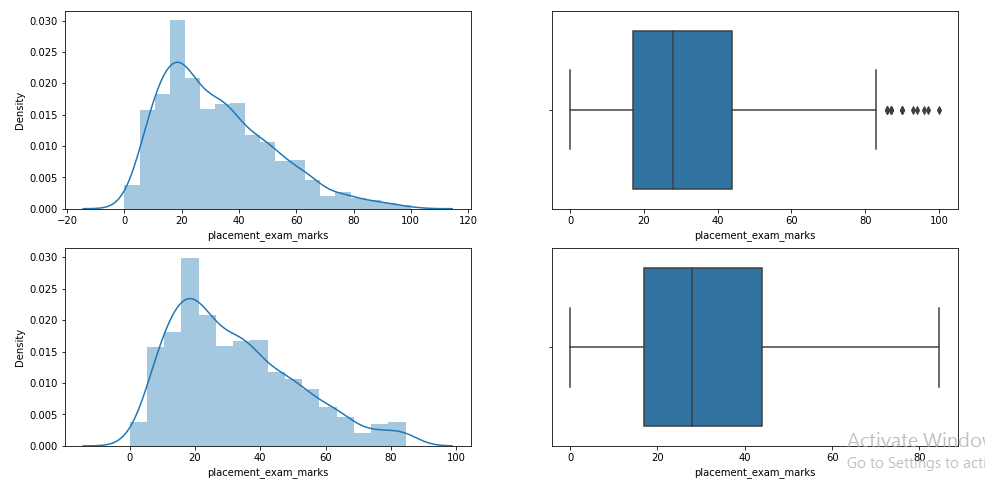
plt.subplot(2,2,3)

sns.distplot(new\_df\_cap['placement\_exam\_marks'])

plt.subplot(2,2,4)

sns.boxplot(new\_df\_cap['placement\_exam\_marks'])

plt.show()



***This completes our IQR based technique!***

**Percentile :**

– This technique works by setting a particular threshold value, which decides based on our problem statement.

– While we remove the outliers using capping, then that particular method is known as **Winsorization**.

– Here we always maintain**symmetry**on both sides means if remove 1% from the right then in the left we also drop by 1%.

**Step-1: Import necessary dependencies**

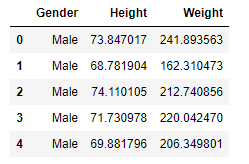
import numpy as np

import pandas as pd

**Step-2: Read and Load the dataset**

df = pd.read\_csv('weight-height.csv')

df.sample(5)

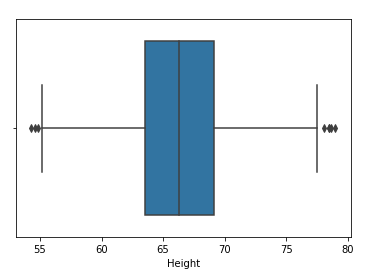


**Step-3: Plot the distribution plot of “height” feature**

sns.distplot(df['Height'])

**Step-4: Plot the box-plot of “height” feature**

sns.boxplot(df['Height'])



**Step-5: Finding upper and lower limit**

upper\_limit = df['Height'].quantile(0.99)

lower\_limit = df['Height'].quantile(0.01)

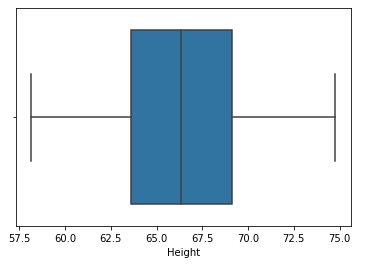
**Step-7: Apply trimming**

new\_df = df[(df['Height'] <= 74.78) & (df['Height'] >= 58.13)]

**Step-8: Compare the distribution and box-plot after trimming**

sns.distplot(new\_df['Height'])

sns.boxplot(new\_df['Height'])



**Winsorization :**

**Step-9: Apply Capping(Winsorization)**

df['Height'] = np.where(df['Height'] >= upper\_limit,

upper\_limit,

np.where(df['Height'] <= lower\_limit,

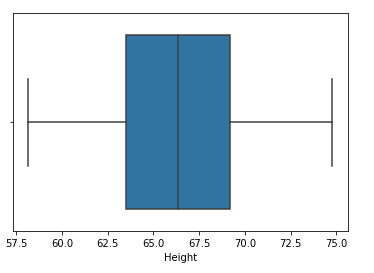
lower\_limit,

df['Height']))

**Step-10: Compare the distribution and box-plot after capping**

sns.distplot(df['Height'])

sns.boxplot(df['Height'])



***This completes our percentile-based technique!***