**Data Wrangling II**

Create an “Academic performance” dataset of students and perform the following operations using Python.

1. Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use any of the suitable techniques to deal with them.

2. Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them.

3. Apply data transformations on at least one of the variables. The purpose of this transformation should be one of the following reasons: to change the scale for better understanding of the variable, to convert a non-linear relation into a linear one, or to decrease the skewness and convert the distribution into a normal distribution.

1. Import required modules.

2. **Reading the dataset in a dataframe using Pandas**

df = pd.read\_csv("Path of file ")

**3. Describe the given data**

print(df. describe())

**4. Display first 10 rows of data**

print(df.head(10))

5.

1. **Missing values In Pandas**

**missing data is represented by two values:**

1. **None**: None is a Python singleton object that is often used for missing data in Python code.
2. **NaN** :NaN (an acronym for Not a Number), is a special floating-point value recognized by all systems

 isnull()

**Syntax: Pandas.isnull(“DataFrame Name”) or DataFrame.isnull()  
Parameters: Object to check null values for  
Return Type: Dataframe of Boolean values which are True for NaN values**

 notnull()

**Syntax:**Pandas.notnull(“DataFrame Name”) or DataFrame.notnull()  
**Parameters:**Object to check null values for  
**Return Type:**Dataframe of Boolean values which are False for NaN values

1. **Cleaning Empty Cells**
2. Remove rows that contain empty cells by dropna().If you want to change the original DataFrame, use the inplace = True argument.
3. Replace Empty Values by fillna()
4. Replace Only For Specified Columns
5. Replace Using Mean, Median, or Mode

# 2. Cleaning Data of Wrong Format

## 1. Convert date Into a Correct Format

df['Date'] = pd.to\_datetime(df['Date'])

1. **Cleaning wrong data**
2. One way to fix wrong values is to replace them with something else.

df.loc[7, 'Duration'] = 45

Replace value in row 7 duration coloumn to 45

1. Loop through all values in the "Duration" column.

If the value is higher than 120, set it to 120:

for x in df.index:  
  if df.loc[x, "Duration"] > 120:  
    df.loc[x, "Duration"] = 120

1. Removing rows

for x in df.index:  
  if df.loc[x, "Duration"] > 120:  
    df.drop(x, inplace = True)

4.Removing duplicates

The duplicated() method returns True for every row that is a duplicate, othwerwise False.

Removing duplicates :

Remove all duplicates:

df.drop\_duplicates(inplace = True)

What are outlier ?

-Outliers can be unusually and extremely different from most of the data points existing in our sample.

-create biased results

Types of outliers :

* **Univariate**

A univariate outlier is a data point that consists of an extreme value on one variable.

* **multivariate**.

A multivariate outlier is a combination of unusual scores on at least two variables.

**Most common causes of outliers on a data set:**

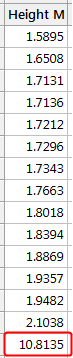
* Data entry errors (human errors)
* Measurement errors (instrument errors)
* Experimental errors (data extraction or experiment planning/executing errors)
* Intentional (dummy outliers made to test detection methods)
* Data processing errors (data manipulation or data set unintended mutations)
* Sampling errors (extracting or mixing data from wrong or various sources)
* Natural (not an error, novelties in data)

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)

## Sorting Your Datasheet to Find Outliers

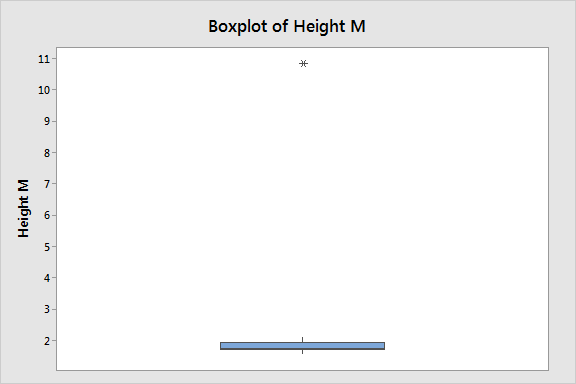
df.sort\_values("city08")

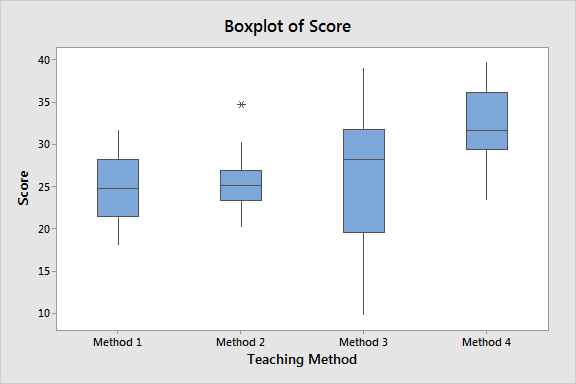


## Graphing Your Data to Identify Outliers

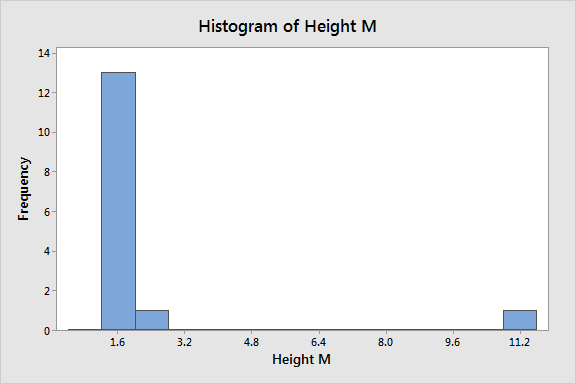
Boxplots, histograms, and scatterplots can highlight outliers.

import seaborn as sns  
sns.boxplot(x=boston\_df['Height M'])





sns.histplot(data=tips, x="total\_bill", color="lime")



import matplotlib.pyplot as plt

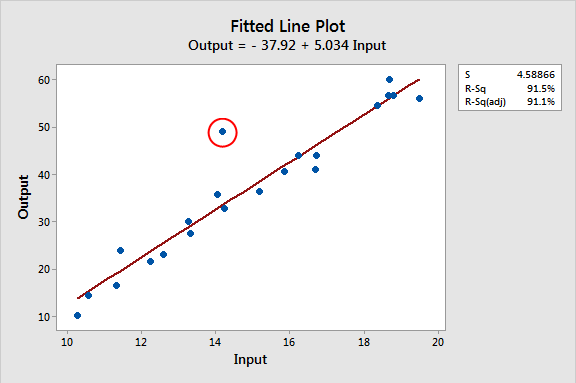
import seaborn as sns

import pandas as pd

df = pd.read\_csv('worldHappiness2016.csv')

sns.scatterplot(data = df, x = "Economy (GDP per Capita)", y = "Happiness Score")

plt.show()



## Using Z-scores to Detect Outliers

Z-scores are the number of standard deviations above and below the mean that each value falls.

For example, a Z-score of 2 indicates that an observation is two standard deviations above the average while a Z-score of -2 signifies it is two standard deviations below the mean.

A Z-score of zero represents a value that equals the mean.

To calculate the Z-score for an observation,

take the raw measurement, subtract the mean, and divide by the standard deviation. Mathematically, the formula for that process is the following:

z-score equation

from scipy import stats

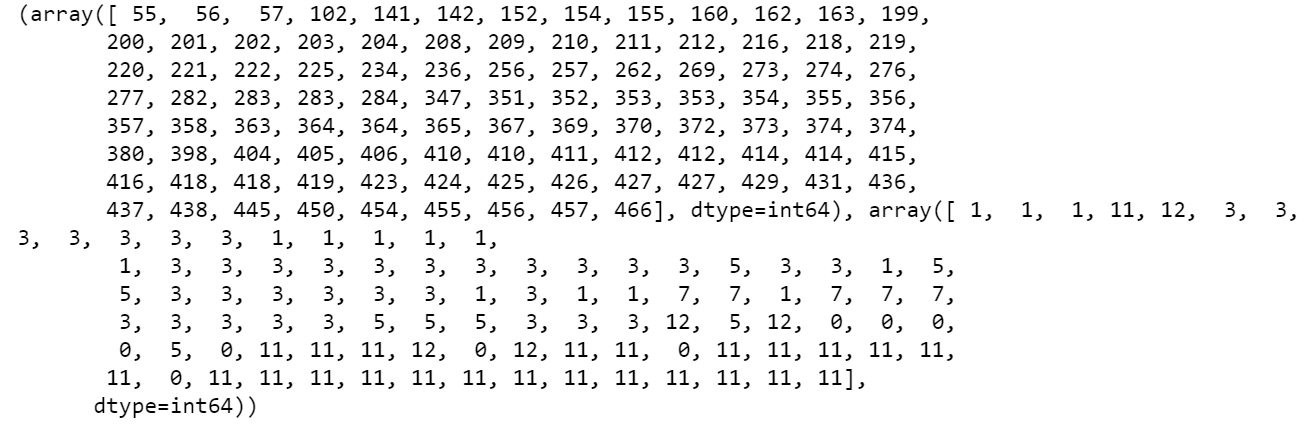
import numpy as np

z = np.abs(stats.zscore(boston\_df))

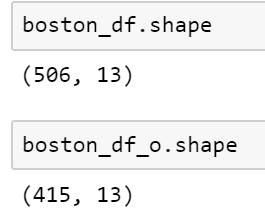
print(z)

threshold = 3

print(np.where(z > 3))



boston\_df\_o = boston\_df\_o[(z < 3).all(axis=1)]



What is Feature scaling ?

Scaling data is the process of increasing or decreasing the magnitude according to a fixed ratio , in simpler words you change the size but not the shape of the data .

Example the column with the name height will have data in cm (centimetre ) and column with weight will have data in Kg(kilogram).

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature\_range=(5, 10))

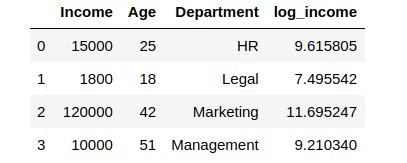
df\_scaled[col\_names] = scaler.fit\_transform(features.values)

df\_scaled

## Log Transform

The Log Transform is one of the most popular Transformation techniques. It is primarily used to convert a [skewed distribution](https://www.analyticsvidhya.com/blog/2020/07/what-is-skewness-statistics/?utm_source=blog&utm_medium=Feature_Transformation_and_Scaling_Techniques) to a normal distribution/less-skewed distribution.

In this transform, we take the log of the values in a column and use these values as the column instead.



df['log\_income'].plot.hist(bins = 5)

df['log\_income'] = np.log(df['Income'])

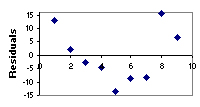
# We created a new column to store the log values

## A Transformation Example

The table shows data for independent and dependent variables - x and y, respectively.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **x** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **y** | 2 | 1 | 6 | 14 | 15 | 30 | 40 | 74 | 75 |

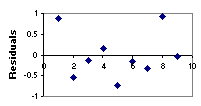
When we apply a linear regression to the untransformed raw data, the [residual plot](https://stattrek.com/statistics/dictionary.aspx?definition=Residual%20plot) shows a non-random pattern (a U-shaped curve), which suggests that the data are nonlinear.



Suppose we repeat the analysis, using a quadratic model to transform the dependent variable. For a quadratic model, we use the square root of y

The table below shows the transformed data we analyzed.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **x** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **yt** | 1.41 | 1.00 | 2.45 | 3.74 | 3.87 | 5.48 | 6.32 | 8.60 | 8.66 |



skewness is the measure of how much the probability distribution of a random variable deviates from the [normal distribution](https://www.analyticsvidhya.com/blog/2020/04/statistics-data-science-normal-distribution/?utm_source=blog&utm_medium=what-is-skewness-statistics).

normal distribution is the probability distribution without any skewness.

