**Lab # 9**

**Dataset Preprocessing and Scaling techniques**

**OBJECTIVE**

Checking the data set for missing values and outliers. Implementing Normalization and Standardization techniques to scale the values.

**THEORY**

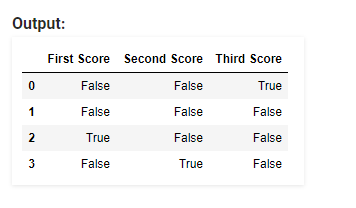
## **WORKING WITH MISSING DATA:**

Missing Data can occur when no information is provided for one or more items or for a whole unit. Missing Data is a very big problem in real life scenario. Missing Data can also refer to as NA (Not Available) values in pandas.

**Checking for missing values using isnull()**

In order to check null values in Pandas DataFrame, we use isnull() function, this function returns data frame of Boolean values which are True for NaN values.

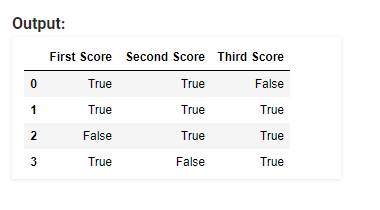
*# importing pandas as pd***import** pandas **as** pd  
  
*# importing numpy as np***import** numpy **as** np  
  
*# dictionary of lists*dict = {**'First Score'**: [100, 90, np.nan, 95],  
 **'Second Score'**: [30, 45, 56, np.nan],  
 **'Third Score'**: [np.nan, 40, 80, 98]}  
  
*# creating a dataframe from list*df = pd.DataFrame(dict)  
  
*# using isnull() function*print(df.isnull())



**Checking for missing values using notnull()**

In order to check null values in Pandas Dataframe, we use notnull() function, this function returns dataframe of Boolean values which are False for NaN values.

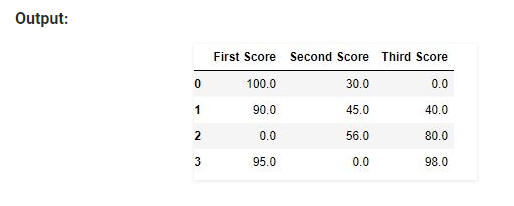
*# importing pandas as pd***import** pandas **as** pd  
  
*# importing numpy as np***import** numpy **as** np  
  
*# dictionary of lists*dict = {**'First Score'**: [100, 90, np.nan, 95],  
 **'Second Score'**: [30, 45, 56, np.nan],  
 **'Third Score'**: [np.nan, 40, 80, 98]}  
  
*# creating a dataframe using dictionary*df = pd.DataFrame(dict)  
  
*# using notnull() function*print(df.notnull())



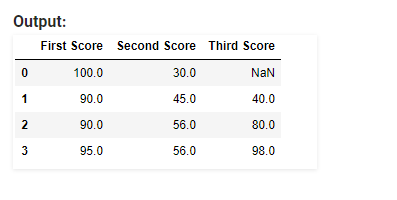
**Filling missing values using fillna(), replace() and interpolate()**

In order to fill null values in datasets, we use fillna(), replace() and interpolate() functions, these functions replace NaN values with some value of their own. All these functions help in filling null values in datasets of a DataFrame. Interpolate() function is basically used to fill NA values in the dataframe but it uses various interpolation techniques to fill the missing values rather than hard-coding the values.

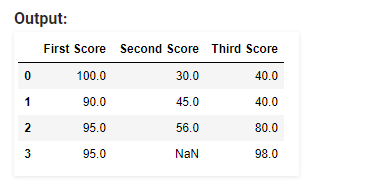
*# importing pandas as pd***import** pandas **as** pd  
  
*# importing numpy as np***import** numpy **as** np  
  
*# dictionary of lists*dict = {**'First Score'**: [100, 90, np.nan, 95],  
 **'Second Score'**: [30, 45, 56, np.nan],  
 **'Third Score'**: [np.nan, 40, 80, 98]}  
  
*# creating a dataframe from dictionary*df = pd.DataFrame(dict)  
  
*# filling missing value using fillna()*print(df.fillna(0))



*# filling missing values with the previous ones using fillna()*print(df.fillna(method=**'pad'**))



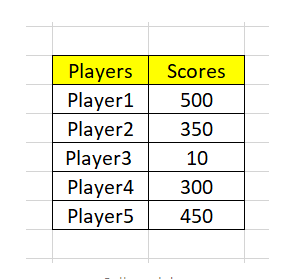
*# filling missing values with the next ones using fillna()*print(df.fillna(method=**'bfill'**))



*# will replace Nan value in dataframe with value 0*print(df.replace(to\_replace=np.nan, value=0))

*# to interpolate the missing values*print(df.interpolate(method=**'linear'**, limit\_direction=**'forward'**))

## OUTLIERS



An **outlier** is an observation point that is distant from other observations.

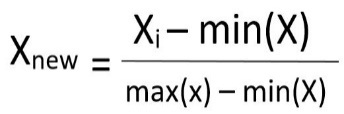
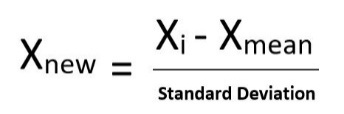
As you can see from the above collected data that all other players scored 300+ except Player3 who scored 10. This figure can be just a typing **mistake**or it is showing the **variance** in your data and indicating that Player3 is performing very bad so, needs improvements. In case of a mistake, we can simply interpolate that value.

## **FEATURE SCALING USING NORMALIZATION AND STANDARDIZATION:**

Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units. If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values.

**Example:** If an algorithm is not using feature scaling method then it can consider the value 3000 meter to be greater than 5 km but that’s actually not true and in this case, the algorithm will give wrong predictions. So, we use Feature Scaling to bring all values to same magnitudes and thus, tackle this issue.

**Techniques to perform Feature Scaling**

* Min-Max Normalization: This technique re-scales a feature or observation value with distribution value between 0 and 1.  
  
* Standardization: It is a very effective technique which re-scales a feature value so that it has distribution with 0 mean value and variance equals to 1.  
  

**CODE: (Scaling through Min-Max Normalization)**

import numpy as np

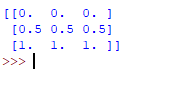
from sklearn import preprocessing

x = np.array([[1,2,3],[4,5,6],[7,8,9]])

minmax = preprocessing.MinMaxScaler(feature\_range=(0,1))

print(minmax.fit(x).transform(x))

**OUTPUT:**



**CODE: (Scaling through Standardization)**

import numpy as np

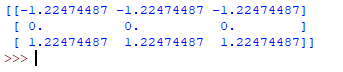
from sklearn import preprocessing

x = np.array([[1,2,3],[4,5,6],[7,8,9]])

standard = preprocessing.StandardScaler().fit(x)

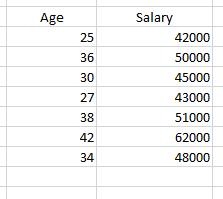
print(standard.transform(x))

**OUTPUT:**



**Lab Tasks:**

1. Write a python code to fill all the null values in Gender column of employees.csv with “No Gender”. Print the first 10 to 30 rows of the data frame for visualization.
2. Write a python code to scale the values of features (Age and Salary) using Min-Max Normalization technique. Verify your answers by applying the formula mentioned above.
3. Write a python code to scale the values of features (Age and Salary) using Standardization technique. Verify your answers by applying the formula mentioned above.



1. Given this dictionary, create a dataframe from dictionary and interpolate the missing values using backward interpolation. Hint: use interpolate().

dict = {**'First Score'**: [100, 90, np.nan, 95],  
 **'Second Score'**: [30, 45, 56, np.nan],  
 **'Third Score'**: [np.nan, 40, 80, 98]}