Data Preprocessing

* Data Cleaning.
* Data Transformation.
* Data Reduction.

The data sets which we are dealing consists of missing values, we need to fill them using various techniques:

import pandas as pd  
df=pd.read\_csv("pima-indians-diabetes.csv")  
cols=["Pregnancies","Glucose","BloodPressure","SkinThickness","Insulin","BMI","DiabetesPedigreeFunction","Age"]  
df[cols] = df[cols].replace({0:np.nan})  
df.isnull().sum()

Fill the Missing Values*.*

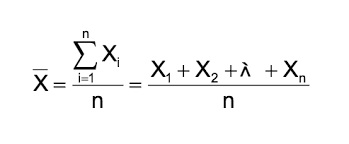
1. Filling with a constant Value like Zero.
2. Mean.
3. Median.
4. Mode.
5. K- Nearest Neighbors.
6. Imputation using Multivariate Imputation by Chained Equation (MICE).
7. Imputation using Deep learning(Datawig).

**Filling our Missing Values using a constant value like “ZERO”**

import pandas as pd  
import numpy as np  
df=pd.read\_csv("diabetes.csv")  
cols = ["Pregnancies","Glucose","BloodPressure","SkinThickness","Insulin","BMI","DiabetesPedigreeFunction","Age"]  
df[cols] = df[cols].replace({np.nan:0})  
df.isnull().sum()

**Filling Missing Values using Mean.**

* It is the arithmetic average of all the values in a given set.
* Calculated as a sum of all numbers divided by the total number of data points



the mean formula

Our First Step in filling the missing values with Mean is Calculating the mean for each column.

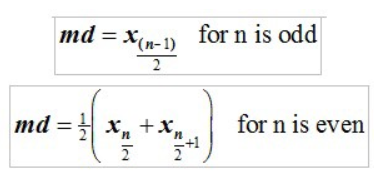
import pandas as pd  
import numpy as np  
df=pd.read\_csv("diabetes.csv")  
df.mean()

Now let us see how to fill the empty values in the dataset with above generated **Mean** values:

print(df.head())  
for index,value in enumerate(cols):  
 df[value]=df[value].replace([0],c[index])print(df.head())

**Filling missing values with Median.**

We can follow the same technique as above but need to replace the mean with the median.



the median formula

To Apply the **Median,** the data should be arranged in either descending or ascending order. The biggest advantage of using the **median function** is that the data-list does not need to be sorted before being sent as a parameter to the median() function.

import pandas as pd  
import numpy as np  
df=pd.read\_csv("diabetes.csv")  
c=df.median()  
print(c)

I removed the missing values before applying the **median()** function itself. In the end, Both Gives us the same result:-

import pandas as pd  
import numpy as np  
df=pd.read\_csv("diabetes.csv")  
cols = ["Pregnancies","Glucose","BloodPressure","SkinThickness","Insulin","BMI","DiabetesPedigreeFunction","Age"]  
for i in cols:  
 a=df[i]  
 a=a.dropna()  
 b=a.sort\_values(axis=0,ascending=True)  
 print(b.median())

let us see how to fill the empty values in the dataset with above generated **Median** values:

print(df.head())  
for index,value in enumerate(cols):  
 df[value]=df[value].replace([0],c[index])  
print(df.head())

**Filling Missing Values with Mode.**

Mode is the most repeated value in a given data set, consisting of Discrete or categorical values. There may also be two modes (bimodal), three modes (trimodal), or four or more modes (multimodal).

**Our data set consists of numerical attributes and each numerical might not be repeated many times, so using mode as our primary technique to fill missing values is not recommended as it might not give desired results.**

import pandas as pd  
import numpy as np  
df=pd.read\_csv("diabetes.csv")  
df=df.replace([0],np.NaN)  
c=df.mode()  
print(c)

`

Now let’s fill the missing values using calculated Mode values.

cols = ["Pregnancies","Glucose","BloodPressure","SkinThickness","Insulin","BMI","DiabetesPedigreeFunction","Age"]  
for i in cols:  
 a = df[i].mode()  
 print(a[0])  
 df[i]=df[i].replace(to\_replace=[0],value=a[0])  
print(df.head(10))

**Filling Missing Values using K- Nearest Neighbors**

Assume **k=10**, means we have to select 10 nearest neighbors surrounding our Green point. To elect those 10 points, we need to calculate the distances from green point to all other possible points in the dataset, For that, we will use mostly **Euclidean distance.** It is given by**:**





The formula for [**Euclidean Distance**](https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/#:~:targetText=Despite%20its%20simplicity%2C%20KNN%20can,based%20on%20their%20expression%20profiles.)

import pandas as pd  
import numpy as np  
import impyute  
from impyute.imputation.cs import fast\_knn   
df=pd.read\_csv("diabetes.csv")  
cols = ["Pregnancies","Glucose","BloodPressure","SkinThickness","Insulin","BMI","DiabetesPedigreeFunction","Age"]  
for i in cols:  
 df[i]=df[i].replace(to\_replace=[0],value=np.NaN)  
k-nn\_impute=fast\_knn(df, k=3)  
pd.DataFrame(k-nn\_impute).to\_csv("file\_k-nn.csv")

**Filling missing values using Multivariate imputation by chained equation(MICE)**

import pandas as pd  
import numpy as np  
import impyute  
from impyute.imputation.cs import mice  
df=pd.read\_csv("diabetes.csv")  
cols = ["Pregnancies","Glucose","BloodPressure","SkinThickness","Insulin","BMI","DiabetesPedigreeFunction","Age"]  
for i in cols:  
 df[i]=df[i].replace(to\_replace=[0],value=np.NaN)  
mice\_imp=mice(df)  
pd.DataFrame(mice\_imp).to\_csv("file\_MICE.csv")

**Filling missing values using Datawig.**

Here we will be using a library **Datawig**, This method works very well with  
categorical and non-numerical features. It is a library that learns Machine  
Learning models using Deep Neural Networks to impute missing values in  
a data frame.

**Datawig** expects you to provide them:

* column name of the column you would like to impute values for (called output\_column below) and some.
* column names that contain values that you deem useful for imputation (called input\_columns below).

import datawig  
import pandas as pd  
import numpy as np  
df=pd.read\_csv("diabetes.csv")  
cols = ["Pregnancies","Glucose","BloodPressure","SkinThickness","Insulin","BMI","DiabetesPedigreeFunction","Age"]  
for i in cols:  
 df[i]=df[i].replace(to\_replace=[0],value=np.NaN)  
df\_train, df\_test = datawig.utils.random\_split(df)  
imputer = datawig.SimpleImputer(  
 input\_columns=[ 'Glucose', 'BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age','Outcome'],output\_column= 'Pregnancies',output\_path = 'imputer\_model')  
imputer.fit(train\_df=df\_train)  
imputed = imputer.predict(df\_test)  
pd.DataFrame(imputed).to\_csv("file\_datawig.csv")