**Experiment 3: Data Cleaning**

**Objective** :To learn and implement the data cleaning techniques

**Time Required** : 3 hrs

**Programming Language** : Python

**Software Required** : Anaconda

**Introduction**

Data cleaning is the process of identifying and correcting errors, inconsistencies, and inaccuracies in a dataset. It is an essential step in data preparation and analysis, as it ensures that the data is accurate, consistent, and complete.

The process of data cleaning typically involves several steps, including:

* This involves assessing the quality and completeness of the data, identifying any missing or incorrect values, and verifying that the data is in the correct format.
* This involves correcting any errors or inconsistencies in the data, such as misspellings, formatting errors, or incorrect data types.
* This involves converting data into a consistent format, such as converting dates into a standard format or converting categorical data into numerical data.
* This involves combining data from multiple sources, resolving any inconsistencies or conflicts in the data, and creating a unified dataset.

Overall, the goal of data cleaning is to ensure that the data is accurate, consistent, and complete, and to reduce the risk of errors and inaccuracies in the analysis process.

Bad data could be:

1. Empty cells
2. Data in wrong format
3. Wrong data
4. Duplicates

The dataset that we are going to use is ‘rawdata.csv’. It has following characteristics:

* The data set contains some empty cells ("Date" in row 22, and "Calories" in row 18 and 28).
* The data set contains wrong format ("Date" in row 26).
* The data set contains wrong data ("Duration" in row 7).
* The data set contains duplicates (row 11 and 12).

**Step 1: Load and view dataset**

**Task:** Load and view the dataset provided after importing important libraries.

**Step 2: Dealing with empty cells**

As empty cells can potentially give a wrong result while analyzing data, so to deal with the empty cells, we would be performing the following operations:

* 1. ***Remove Rows***

One way to deal with empty cells is to remove rows that contain empty cells by using the method dropna(). Since data sets can be very big, and removing a few rows will not have a big impact on the result.

**Task:** Remove all the empty cells in dataset provided

By default, the dropna() method returns a new DataFrame, and will not change the original. If you want to change the original DataFrame, use the inplace = True argument.

* 1. ***Replace empty values***

Another way of dealing with empty cells is to insert a *new* value instead by using method fillna(). This way you do not have to delete entire rows just because of some empty cells.

**Task:** Replace the empty values with 150

***Replace only for a specified Columns***

In above methods, we are replacing all empty cells in the whole Data Frame. To only replace empty values for one column, specify the *column name* for the DataFrame.

*index=False* specifies that the index column should not be included in the output file.

**Task:** Replace the empty values in ‘Calories’ with 130.

***Replace Using Mean, Median, or Mode***

A common way to replace empty cells, is to calculate the mean, median or mode value of the column. Pandas uses the mean() median() and mode() methods to calculate the respective values for a specified column:

1. **Mean:**

Mean = the average value (the sum of all values divided by number of values).

1. **Median:**

Median = the value in the middle, after you have sorted all values ascending.

1. **Mode:**

Mode = the value that appears most frequently.

**Tasks:**

1. Calculate the Mean of ‘Calories’ and replace the missing values with it.
2. Calculate the Median of ‘Maxpulse’ and replace the missing values with it.
3. Calculate the mode of ‘Pulse’ and replace the missing values with it.

**Step 3: Dealing with data of wrong format**

As cells with data of wrong format can make it difficult, or even impossible, to analyze data. To fix it, you have two options: remove the rows, or convert all cells in the columns into the same format.

1. ***Convert Into a Correct Format***

In our Data Frame, we have two cells with the wrong format. Check out row 22 and 26, the 'Date' column should be a string that represents a date.

You will see that the empty date in row 22 got a NaT (Not a Time) value, in other words an empty value. One way to deal with empty values is simply removing the entire row.

**Task:** Remove the entire row 22

1. ***Removing Rows***

The result from the converting in the example above gave us a NaT value, which can be handled as a NULL value, and we can remove the row by using the dropna() method.

**Step 4: Dealing with wrong data**

"Wrong data" does not have to be "empty cells" or "wrong format", it can just be wrong, like if someone registered "199" instead of "1.99". Sometimes you can spot wrong data by looking at the data set, because you have an expectation of what it should be.

In our data set, you can see that in row 7, the duration is 450, but for all the other rows the duration is between 30 and 60. It doesn't have to be wrong but taking in consideration that this is the data set of someone's workout sessions, we conclude with the fact that this person did not work out in 450 minutes.

**a) Replacing Values**

One way to fix wrong values is to replace them with something else. In our test data, it is most likely a typo, and the value should be "45" instead of "450".

**Task:** Insert the value "45" instead of "450" in row 7.

For small data sets you might be able to replace the wrong data one by one, but not for big data sets. To replace wrong data for larger data sets you can create some rules, e.g. set some boundaries for legal values, and replace any values that are outside of the boundaries.

**Task:** Loop through all values in the ‘Duration’ column. If the value is higher than 120, set it to 120.

**b) Removing Rows**

Another way of handling wrong data is to remove the rows that contains wrong data. This way you do not have to find out what to replace them with, and there is a good chance you do not need them to do your analyses.

**Task:** Delete rows where "Duration" is higher than 120.

**Step 5: Dealing with duplicates**

Duplicate rows are rows that have been registered more than one time. By looking at our data set, we can assume that rows 11 and 12 are duplicates.

To discover duplicates, we can use the duplicated() method. The duplicated() method returns a Boolean values for each row:

**Task:** Remove duplicates using the drop\_duplicates() method.

**Final Lab Task:**

**For given dataset ‘diabetes.csv’, perform data cleaning techniques. After applying the data cleaning methods, carry out fitting of data with a Regression Model and compute its accuracy. The code for fitting with Logistic Regression is as follows:**

# Split Data into a training set and test set

from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.3, random state=1)

Note: Here x includes the features or independent variables where y includes the dependent variable variables or label.

# Fitting with Logistic Regression

from sklearn.linear\_model import LogisticRegression

lr = LogisticRegression()

lr.fit(x\_train,y\_train)

y\_pred = lr.predict(x\_test)

# Import scikit-learn metrics module for accuracy calculation

From sklearn import metrics

# Compute Model Accuracy

print(“Accuracy: “, metrics.accuracy\_score(y\_test, y\_pred)\*100)