Course: Artificial Intelligence and Machine Learning Code: 20CS51I

WEEK - 10: Preparing the Data

- **Dealing with:**
  - i. **Missing Values**
  - ii. **Categorical Values**
  - iii. **Labeled Encoding**
  - iv. **One hot coding**

Session No. 5

Missing Values:

Missing data is defined as the values or data that is not stored (or not present) for some variable/s in the given dataset.

Below is a sample of the missing data from the Titanic dataset. You can see the columns 'Age' and 'Cabin' have some missing values.

PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	male	22	1	0	A/5 21171	7.25		S
2	1	1	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	female	26	0	0	STON/O2. 3101282	7.925		S
4	1	1	female	35	1	0	113803	53.1	C123	S
5	0	3	male	35	0	0	373450	8.05		S
6	0	3	male		0	0	330877	8.4583		Q

How is Missing Value Represented in The Dataset?

In the dataset, blank shows the missing values.

In Pandas, usually, missing values are represented by **NaN**.

It stands for **Not a Number**.

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

Types Of Missing Value

Formally the missing values are categorized as follows:

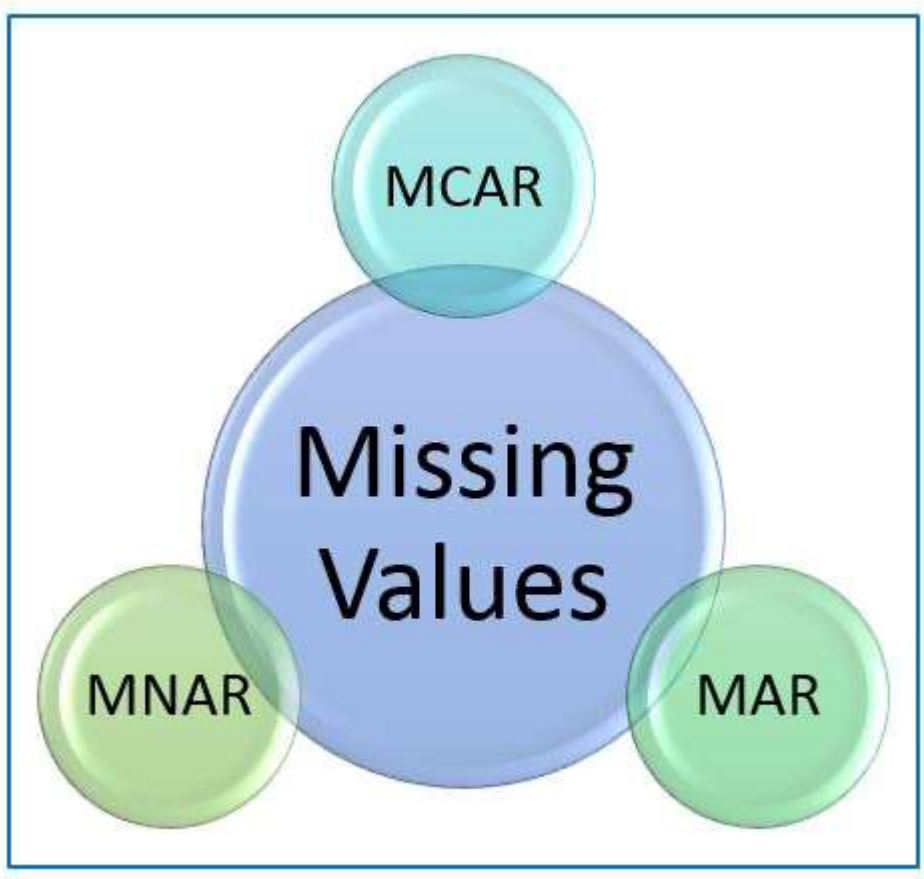

Figure 1 - Different Types of Missing Values in Datasets

Missing Completely At Random (MCAR)

- In MCAR, the probability of data being missing is the same for all the observations.
- In this case, there is no relationship between the missing data and any other values observed or unobserved (the data which is not recorded) within the given dataset.
- That is, missing values are completely independent of other data. There is no pattern.
- In the case of MCAR, the data could be missing due to human error, some system/equipment failure, loss of sample, or some unsatisfactory technicalities while recording the values.
- For Example, suppose in a library there are some overdue books. Some values of overdue books in the computer system are missing. The reason might be a human error like the librarian forgot to type in the values. So, the missing values of overdue books are not related to any other variable/data in the system.
- It should not be assumed as it's a rare case. The advantage of such data is that the statistical analysis remains unbiased.

Missing At Random (MAR):

- Missing at random (MAR) means that the reason for missing values can be explained by variables on which you have complete information as there is some relationship between the missing data and other values/data.
- In this case, the data is not missing for all the observations. It is missing only within sub-samples of the data and there is some pattern in the missing values.

Missing Not at Random (MNAR):

- Missing values depend on the unobserved data.
- If there is some structure/pattern in missing data and other observed data can not explain it, then it is Missing Not at Random (MNAR).
- If the missing data does not fall under the MCAR or MAR then it can be categorized as MNAR.

Handling Missing Values:

Handle Missing Data with Deletion:

In the list-wise deletion method, you remove a record or observation in the dataset if it contains some missing values.

You can implement list-wise deletion in Python by simply using the Pandas .dropna method like this:

```
df.dropna(axis=1, inplace=True)
```

Handle Missing Data with Imputation:

- Another frequent general method for dealing with missing data is to fill in the missing value with a substituted value.
- **This method entails replacing the missing value with a specific value. To use it, you need to have domain knowledge of the dataset. You use this to populate the MAR and MCAR values.**

To implement it in Python, you use the .fillna method in Pandas like this:

```
df.fillna(inplace=True)
```

Regression imputation:

The regression imputation method includes creating a model to predict the observed value of a variable based on another variable. Then you use the model to fill in the missing value of that variable

Simple Imputation:

This method involves utilizing a numerical summary of the variable where the missing value occurred (that is using the feature or variable's central tendency summary, such as mean, median, and mode).

You use this method in the MCAR category. And you implement it in Python using the **SimpleImputer** transformer in the Scikit-learn library.

```
from sklearn.impute import SimpleImputer
#Specify the strategy to be the median class
fea_transformer = SimpleImputer(strategy="median")
values = fea_transformer.fit_transform(df[["Distance"]])
pd.DataFrame(values)
```

Mean Imputation:

- Using Imputer to fill the nun values with the Mean. missing_values : In this we have to place the missing values and in pandas it is 'NaN'.
- strategy: In this we have to pass the strategy that we need to follow to impute in missing value it can be mean, median, most_frequent or constant. By default, it is mean.

```
miss_mean_imputer = Imputer(missing_values='NaN', strategy='mean', axis=0)
```

ii) Categorical Values:

What is Categorical Data?

- Categorical data is a type of data that is used to group information with similar characteristics, while numerical data is a type of data that expresses information in the form of numbers.
- Most Machine Learning algorithms cannot work with categorical data and needs to be converted into numerical data. Sometimes in datasets, we encounter columns that contain categorical features (string values) for example parameter *Gender* will have categorical parameters like *Male*, *Female*.
- These labels have no specific order of preference and also since the data is string labels, machine learning models misinterpreted that there is some sort of hierarchy in them.
- One approach to solve this problem can be label encoding where we will assign a numerical value to these labels for example *Male* and *Female* mapped to *0* and *1*. But this can add bias in our model as it will start giving higher preference to the *Female* parameter as 1>0 and ideally both labels are equally important in the dataset

Categorical variables can be divided into two categories:

- Nominal: no particular order
- Ordinal: there is some order between values

iii) Labeled Encoding

In label encoding, each category is assigned a value from 1 through N where N is the number of categories for the feature. There is no relation or order between these assignments.

Example:

```
data = pd.DataFrame({ 'gender' : ['Male', 'Female', 'Male', 'Female',
'Female'],
                      'class' : ['A','B','C','D','A'],
                       'city' :
['Delhi','Gurugram','Delhi','Delhi','Gurugram'] })
data.head()

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
Label encoder takes no arguments
le_class = le.fit_transform(data[["class"]])
```

One hot coding:

- In this technique, the categorical parameters will prepare separate columns for both Male and Female labels.
- So, wherever there is Male, the value will be 1 in Male column and 0 in Female column, and vice-versa.
- Let's understand with an example: Consider the data where fruits and their corresponding categorical values and prices are given.

Fruit	Categorical value of fruit	Price
apple	1	5
mango	2	10
apple	1	15
orange	3	20

The output after one-hot encoding of the data is given as follows,

apple	mango	orange	price
1	0	0	5
0	1	0	10
1	0	0	15
0	0	1	20

Code: Python code implementation of Manual One-Hot Encoding Technique Loading the data

```
# Program for demonstration of one hot encoding

# import libraries
import numpy as np
import pandas as pd

# import the data required
data = pd.read_csv("employee_data.csv")
print(data.head())
```

Output:

	Emploee_ID	Gender	Remarks
0	45	Male	Nice
1	78	Female	Good
2	56	Female	Great
3	12	Male	Great
4	7	Female	Nice

Checking for the labels in the categorical parameters

```
print(data['Gender'].unique())
print(data['Remarks'].unique())
```

Output:

```
array(['Male', 'Female'], dtype=object)
array(['Nice', 'Good', 'Great'], dtype=object)
```

Checking for the label counts in the categorical parameters

```
data['Gender'].value_counts()
data['Remarks'].value_counts()
```

Output:

```
Female    7
Male      5
Name: Gender, dtype: int64

Nice     5
Great    4
Good     3
Name: Remarks, dtype: int64
```