Data Preprocessing

- **Loading the data:**

df = pd.read_csv("path for the .csv file to be uploaded")

df = pd.read_excel("path for the .xlsx file")

- **Checking the data:**

df.shape - To check the number of rows and columns

df.columns.values - What are the column names?, Sometimes import doesn't consider column names while importing

df.head(n) - First n observations of data

df.tail(n) - Last n observations of the data

df.dtypes - Data types of all variables

df.describe() - Summary of all variables

df['custId'].describe() - Summary of a variable

df.columnname.value_counts() - Get frequency table for a given variable

table(df$columnnanme) - Get frequency tables for categorical variable

sum(df.columnname.isnull()) - Missing value count in a variable

df.sample(n=10) - Take a random sample of size 10

- **Missing values:**

i) isnull().sum() - count missing values per column.

ii) dropna() - drop rows with missing values.

iii) dropna(axis=1) - drop columns that has atleast one NaN.

iv) dropna(how="all") - drop rows where all rows are NaN.

v) dropna(thresh=4) - drop rows that have atleast 4 NaN.

- **Duplicate values:**

i) **full data**

```
df=df.duplicated()
sum(df) - some int value
df_uniq=df.drop_duplicates()
```

ii) **Column wise**

```
df=df.column_name.duplicated()
sum(df) - some int value
df_uniq=df.drop_duplicates(['column_name'])
```

- **Subsetting:**

We can subset our data based on the column names,values and other attributes based on the requirement.

Example -

- Select first 1000 rows only

  ```
  bank_data1 = bank_data.head(1000)
  ```

- Select only four columns “Cust_num” “age” “default” and “balance”

  ```
  bank_data2 = bank_data[["Cust_num", "age","default","balance"]]
  ```

- Select 20,000 to 40,000 observations along with four variables “Cust_num” “job” “marital” and “education”

  ```
  bank_data3 = bank_data[["Cust_num", "job","marital","education"]][20000:40000]
  ```

- Select 5000 to 6000 observations drop “poutcome“ and “y”

  ```
  bank_data4=bank_data.drop(['poutcome','y'], axis=1)[5000:6000]
  ```

- 5)bank_subset1=bank_data[(bank_data['age']>40) & (bank_data['loan']=="no")]

- 6)bank_subset2=bank_data[(bank_data['age']>40) | (bank_data['loan']=="no"
- 7)bank_subset3= bank_data[(bank_data['age']>40) & (bank_data['loan']=="no") | (bank_data['marital']=="single")]

- <u>**Sorting :**</u>

df=df.sort('column_name',ascending=False)

df=df.sort_index(axis=1, ascending=True)

df.sort_values(by='column_name',ascending=False)

- <u>**Joining or Merging:**</u>

➢ INNER JOIN

inner_df = pd.merge(Table1, Table2, on='column_name in both tables', how='inner')

➢ OUTER JOIN

outer_df = pd.merge(Table1, Table2, on='column_name in both tables', how='outer')

➢ LEFT JOIN

left_df = pd.merge(Table1, Table2, on='column_name in both tables', how='left')

➢ RIGHT JOIN

right_df = pd.merge(Table1, Table2, on='column_name in both tables', how='right')

<u>**Splitting into training, testing and validation sets:**</u>

X=df_data.iloc[:,1:].values (All columns and rows except first one)

Y=df_data.iloc[:,0].values (First column and all rows)

X-train,X-test,Y_train,Y_test = train_test_split(X,Y,test_size=0.3,random_state=0)