**Data Cleaning**

# Data Type

Numerical Date Mixed Categorical

Number Name

# Definition of Data Cleaning

Data Cleaning is the process to preparing data for data analysis / ML /DL by removing and modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted.

# Most important steps for Data Cleaning

Handling Missing Data

Outlier Detection and Handling

Data Scaling and Transformation

Handling Duplicates

Dealing with Inconsistent Data

# Find Null values

dataset.shape (rows,columns)

dataset.isnull().sum() (null value in dataset)

dataset.isnull().sum().sum() (total number of null value in dataset)

dataset.isnull().sum().sum()/dataset.shape[0]\*100 (percentage of null value in dataset)

dataset.isnull().sum().sum()/(dataset.shape[0]\*dataset.shape[1])\*100(percentage of dataset)

# graphical represent null values

sns.heatmap(dataset.isnull())

plt.show()

# Working on null value

if the null value is greater than 50 then not working on dataset. And if the null value is less than 50 then working on the dataset. Second is check the column null value if the percentage of null value is greater than 50 or if you keep that important content in the column you can fill null value then drop the column otherwise you can work on it.

# HANDLING MISSING VALUES (Dropping)

## Delete Data

Delete column: dataset.drop(columns=[“column\_name”], inplace=Ture)

Delete all data by row: dataset.dropna(inplace=True)

# HANDLING MISSING VALUES (Imputing Category Data)

data["Gender"].fillna(method = "ffill" or “bfill”) fill data forward or backword

data["Gender"].fillna(data["Gender"].mode()[0], inplace=True) mostly use this for numberic

for i in dataset.select\_dtypes(include=”object”).columns: fill categorical data

data[ i ].fillna(data["Gender"].mode()[0], inplace=True) implement all datatype

# HANDLING MISSING VALUES (Scikit-learn)

(dataset.select\_dtypes(include="float64").columns)

from sklearn.impute import SimpleImputer

si = SimpleImputer(strategy="mean")

ar = si.fit\_transform(dataset[['Age']])

# What IS ENCODING

When we take categorical data and convert into numerical data there used is encoding

**Why we use encoding in data science**

We use encoding in data science to convert categorical data into numerical format so that machine learning algorithms can process it effectively.

## One Hot Encoding and Dummy Variables

Use there one and two value present in whole column, like Gender (male, female) or Married

## Using Pandas

## pd.get\_dummies(dataset[["Gender",]])

pd.get\_dummies(en\_data)

## using scikit-learn

en\_data = dataset[["Gender"]]

from sklearn.preprocessing import OneHotEncoder

ohe = OneHotEncoder(drop="first")

dataset["Gender"]/ar = ohe.fit\_transform(en\_data).toarray()

pd.DataFrame(ar, columns=["Gender\_Male"])

# Label Encoding

Label encoding use in nominal like (cow, boy, cat, parrot, haroon)

df = pd.DataFrame({"name": ["moonSolution","cat","dog","cow","haroon"]})

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

df["en\_name"] = le.fit\_transform(df["name"])

# Ordinal Encoding

**Using scikit-learn** .unique() check column value

from sklearn.preprocessing import OrdinalEncoder

oe = OrdinalEncoder() #categories= value/varaiable

dataset["Address"] = oe.fit\_transform(dataset[["Address"]])

**Using Map Function** (better from using scikit-learn)

df = pd.DataFrame({"size": ["s","m","l","xl","m","s","xl","s","l","m"]})

ord\_data = {"xl":0,"l":1,"m":2,"s":3}

df["size\_en"] = df["size"].map(ord\_data)

# Handle Outlers

## Using IQR Method

Q1 = dataset["Age"].quantile(0.25)

Q3 = dataset["Age"].quantile(0.75)

IQR = Q3 - Q1

# Define the outlier boundaries

min\_range = Q1 - 1.5 \* IQR

max\_range = Q3 + 1.5 \* IQR

# Filter out outliers

new\_data = dataset[(dataset["Age"] >= min\_range) & (dataset["Age"] <= max\_range)]

# Using Z-Score

## Direct method

min\_range = dataset["AccountBalance"].mean() - (3\*dataset["AccountBalance"].std())

max\_range = dataset["AccountBalance"].mean() + (3\*dataset["AccountBalance"].std())

n\_data = dataset[dataset["AccountBalance"] <= max\_range]

sns.boxplot(n\_data["AccountBalance"])

plt.show()

## Z-Score Method formula = x - µ / α

z\_score = (dataset["AccountBalance"] - dataset["AccountBalance"].mean()) / (dataset["AccountBalance"].std())

dataset["z\_score"] = z\_score

c = dataset[dataset["z\_score"]<3]

sns.boxplot(c["AccountBalance"])

plt.show()

# FEATURE SCALING (Standardization)

It is a very effective technique which re-scales a feature value so that it has distribution with 0 mean value and variance is equal to 1.

Xnew = Xi  - Xmean / standard deviation

from sklearn.preprocessing import StandardScaler

ss = StandardScaler()

ss.fit(dataset[["Age", "ZipCode"]])

dataset[["Age", "ZipCode"]] = pd.DataFrame(ss.transform(dataset[["Age", "ZipCode"]]))

# FEATURE SCALING (Normalization)

It is a scaling technique in which values are shifted or rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

Xnew = Xi  - min(X) / max(X) – main(X)

from sklearn.preprocessing import MinMaxScaler

ms = MinMaxScaler()

ms.fit(dataset[["AccountBalance"]])

dataset[["AccountBalance"]] = pd.DataFrame(ms.transform(dataset[["AccountBalance"]]))

# Handle Duplicated Data

## Find duplicated data

Dataset.duplicated()

Dataset.duplicated().sum()

## Drop Duplicated data

Dataset.drop\_duplicates(inplace=True)

# Replace and Data Type Change

dataset["PhoneNumber"].value\_counts()

## Replace

dataset["PhoneNumber"].replace("previous\_value", "new\_value", inplace=True)

## change data type

dataset["PhoneNumber"] = dataset["PhoneNumber"].astype("int64")

# Function Transformation

When you feel like your data is important and not remove outlier, data are convert non normal distribution to normal distribution

from sklearn.preprocessing import FunctionTransformer

ft = FunctionTransformer(func=np.log1p) / (func = lambda x : x\*\*2)

ft.fit(dataset[["Age"]])

ft.transform(dataset[["Age"]])