→ dated 26.5.25

Internship

→ Data Preprocessing

a) Import necessary library

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

b) Reading the dataset

```
df=pd.read_csv("filename.csv")
#head (printing first 5 rows)
df.head()
#tail (last 5 row's)
df.tail()
```

c) Sanity Check of data.

```
df.shape
df.info()
df.isnull().sum()
```

If there is a missing values more than 50% in a column, we need to do mean, median etc.... instead of deleting the rows or column. (But mostly we do this even if we have less than 1% of Null values in that Column.

```
df.isnull().sum()/df.shape[0] *100
```

#finding duplicates

```
df.duplicated().sum()
```

#identify garbage values.

Since the garbage values like special characters is exist as string. so object type

```
for i in df.select_dtypes(include="object").columns:
    print(df[i].value_counts())
    print("***"*10)
```

→ It gives us the count of each unique value in that column. Even if it special characters it count that and give in output.

Exploratory data analysis

```
df.describe().T
```

→ get the description in numbers.

```
df.describe(include="object")
```

→ to get the description in string not in numbers

#histogram to understand the distribution

→ we can understand how the data distri

```
import warnings
warnings.filterwarnings("ignore")   } don't give warnings.

for i in df.select_dtypes(include="number").columns:
    sns.histplot(data=df, x=i)
    plt.show()
```

Boxplot for outliers

```
for i in select_dtypes(include="number").columns:
    sns.boxplot(data=df, x=i)
    plt.show()
```

Note:

=> We use Histogram to understand distribution of data.

=> We use Boxplot to understand outliers of data.

Now we find any bivariate relationship
we can find it using scatterplot.

```
for i in ['Year', ...]:      -> independent variable (input) ; Target should n't be present here
    sns.scatterplot(data=df, x=i, y='Life expectancy')
    plt.show()
```

'Life expectancy' → Target variable → result we want

Note:

=) Positive Relationship → dots go up together.

=) Negative Relationship → dots go down together.

=) No Relationships → dots are scattered randomly.

Correlation with heatmap to interpret the relation and multicordinality.

```
s: df.select_dtypes(include="number").corr()
plt.figsize((15,15))
sns.heatmap(s, annot=True)
```

Missing treatment → We don't do this for target variable.

Mean or Median

```
for i in ["BMI", ....., ""]:
    df[i].fillna(df[i].median(), inplace=True)
```

(or)

```
df[i].fillna(df[i].mean(), inplace=True)
```

→ columns contains numbers as type not string

for i in [...]: → categorical column

df[i].fillna(df[i].mode()[0], inplace=True) → column that contains categorical data

KNNImputer

from sklearn.impute import KNNImputer

imputer = KNNImputer(n_neighbors=3) → looking at 3 closest other columns. rows that have similar values. Uses their average to fill this column

for i in df.select_dtypes(include='number').columns:

df[i] = impute.fit_transform(df[[i]])

Outlier Treatment → We do this only for continuous, numerical data. column

=> Not for target variable, categorical and discrete variable → counted (x)

(✓) measured

def whisker(col):

q1, q3 = np.percentile(col, [25, 75])

iqr = q3 - q1

lw = q1 - 1.5 * iqr

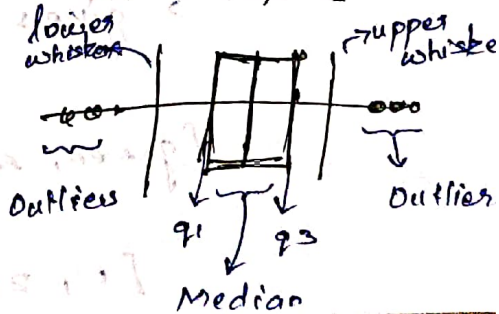
uw = q3 + 1.5 * iqr

return lw, uw

lw → Lower whisker

uw → Upper whisker

In Boxplot



for i in [_____]:

lw, uw = winsor(df[i])

df[i] = np.where(df[i] < lw, lw, df[i])

df[i] = np.where(df[i] > uw, uw, df[i])

Duplicate & garbage value treatment

df.drop_duplicates()

For garbage value treatment

↳ we have multiple ways (search web)

Easy way is replacing it with median or mode of that column only (not for all columns)

Encoding of data (for categorical (text) data into numbers)

One hot encoding → create seperate column for each categorical data [order doesn't matter]

pd.get_dummies(data = df, columns = ["Country", "Status"], drop_first = True)

Label encoding

↳ gives each category one number

like

[green, red, green] → [1, 2, 1]

[order matter]

(like grade.

↳ ordinal data

Normalization (Only for number type column)

Normalization

- Output Range: 0 to 1
- Formula: $\frac{x - min}{max - min}$

  x - current value from that column
  min → min. value of that column
  max → max. " "
- When to use
  - When you need fixed range.

Standardization

- Mean → 0, Std. Deviation → 1
- Formula → $\frac{x - mean}{Std.\ Deviation}$
- When to use
  - When data has outliers