MANJARA CHARITABLE TRUST
RAJIV GANDHI INSTITUTE OF TECHNOLOGY, MUMBAI

Experiment no. 2

Roll no.: 859

Aim: Apply data cleaning techniques (data imputation)

Theory:

A. What is Data Cleaning?

Data cleaning is the process of **detecting, correcting, or removing inaccurate, incomplete, inconsistent, or irrelevant data** from a dataset to improve its quality and reliability for analysis and decision-making.

Dirty data may include:

- Missing values
- Duplicate records
- Incorrect data types
- Outliers
- Inconsistent formatting

High-quality data leads to **accurate analysis, better models, and valid conclusion**

B. DATA CLEANING CYCLE

The data cleaning cycle consists of the following steps:

1. Data Collection

Gather data from different sources such as databases, surveys, or APIs.

2. Data Inspection

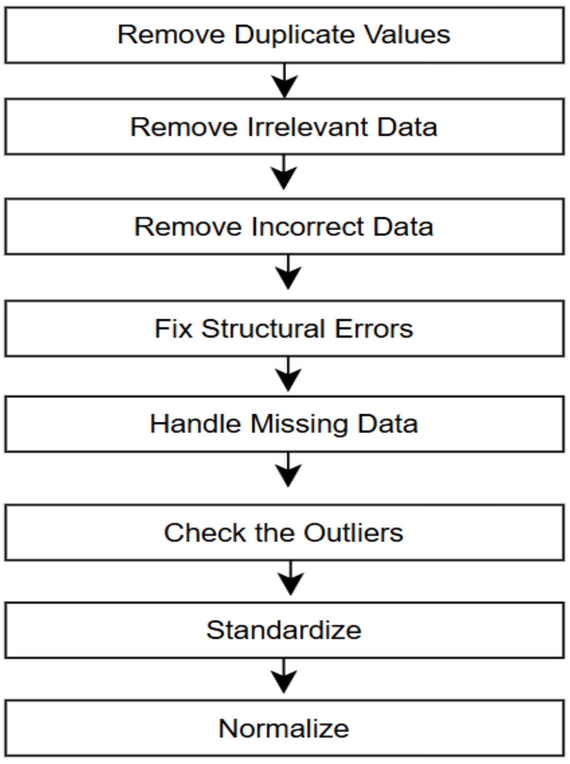
Identify:

- Missing values
- Outliers
- Inconsistencies
- Duplicate entries

3. Data Cleaning

Apply techniques such as:

- Removing duplicates
- Correcting data types
- Handling missing values (imputation)
- Standardizing formats



4. Data Validation

Check whether the cleaned data:

- Meets business rules
- Has logical consistency
- Is complete and accurate

5. Data Storage

Store cleaned data for analysis or modeling.

C. REASONS FOR DATA CLEANING

Data cleaning is necessary because:

- Raw data is often incomplete and inconsistent
- Dirty data leads to wrong insights and poor decisions
- Machine learning models require clean data
- Improves accuracy, reliability, and efficiency
- Saves time during analysis

D. IMPUTATION METHODS

What is Data Imputation?

Data imputation is the process of **replacing missing values with estimated values** rather than removing records.

Common Imputation Methods

- Mean / Median / Mode imputation
- Forward & backward fill
- Regression imputation
- **K-Nearest Neighbors (KNN) imputation**

E. KNN IMPUTATION

What is KNN Imputation?

KNN imputation fills missing values using the **average of the K nearest data points** based on similarity.

How It Works:

1. Select number of neighbors (K)
2. Find K most similar records (distance-based)
3. Compute mean of neighbors
4. Replace missing value

Advantages:

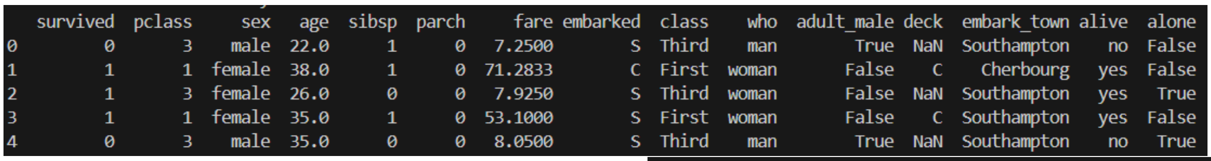
- Uses relationships between variables
- More accurate than mean imputation
- Works well with numerical data

Disadvantages:

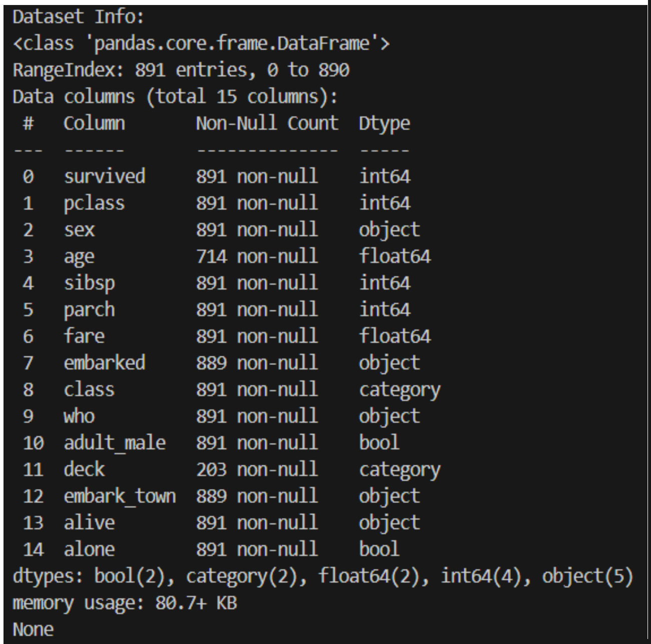
- Computationally expensive
- Sensitive to scaling
- Not suitable for very large datasets

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

df = sns.load_dataset("titanic")
df.to_csv("orig_titanic_dataset.csv", index=False)
print("Dataset Loaded Successfully")
print(df.head())

print("\nDataset Info:")
print(df.info())

print("\nStatistical Summary:")
print(df.describe())

print("\nMissing Values Count:")
print(df.isnull().sum())

df['age'] = df['age'].fillna(df['age'].median())
df['fare'] = df['fare'].fillna(df['fare'].median())
df['embarked'] = df['embarked'].fillna(df['embarked'].mode()[0])
df['deck'] = df['deck'].fillna(df['deck'].mode()[0])
df['embark_town'] = df['embark_town'].fillna(df['embark_town'].mode()[0])

before = df.shape[0]
df.drop_duplicates(inplace=True)
after = df.shape[0]

print(f"\nDuplicates Removed: {before - after}")

categorical_cols = ['sex', 'embarked', 'class', 'who', 'adult_male', 'alone']

for col in categorical_cols:
    df[col] = df[col].astype(str).str.lower()

df['survived'] = df['survived'].astype(int)
df['pclass'] = df['pclass'].astype(int)

Q1 = df['fare'].quantile(0.25)
Q3 = df['fare'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

df['fare'] = np.where(
    df['fare'] > upper_bound,
    upper_bound,
    np.where(df['fare'] < lower_bound, lower_bound, df['fare'])
)

print("\nFinal Missing Values:")
print(df.isnull().sum())

print("\nFinal Dataset Info:")
print(df.info())

print("\nCleaned Dataset Preview:")
print(df.head())

df.to_csv("cleaned_titanic_dataset.csv", index=False)
print("\nCleaned dataset saved as 'cleaned_titanic_dataset.csv'")
```

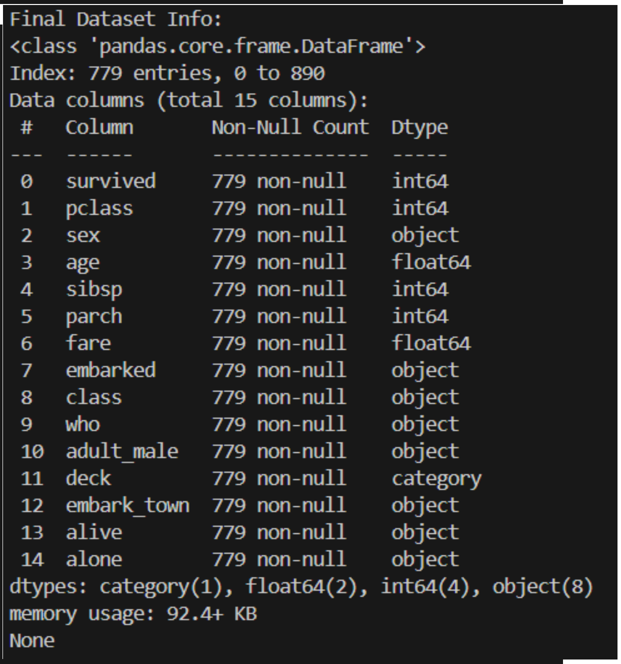
OUTPUT:



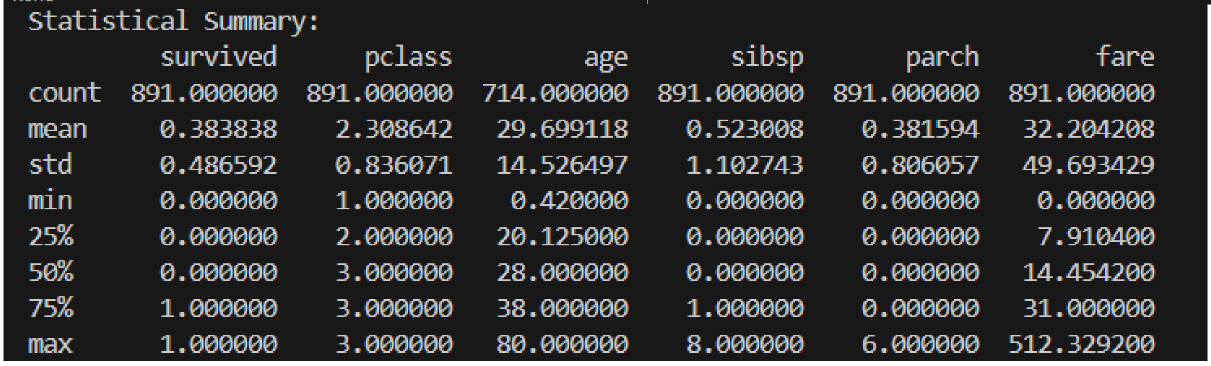
```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class    who  adult_male deck  embark_town alive  alone
0         0       3    male  22.0      1      0   7.2500        S  Third    man        True  NaN  Southampton    no  False
1         1       1  female  38.0      1      0  71.2833        C  First  woman       False    C    Cherbourg   yes  False
2         1       3  female  26.0      0      0   7.9250        S  Third  woman       False  NaN  Southampton   yes   True
3         1       1  female  35.0      1      0  53.1000        S  First  woman       False    C  Southampton   yes  False
4         0       3    male  35.0      0      0   8.0500        S  Third    man        True  NaN  Southampton    no   True
```



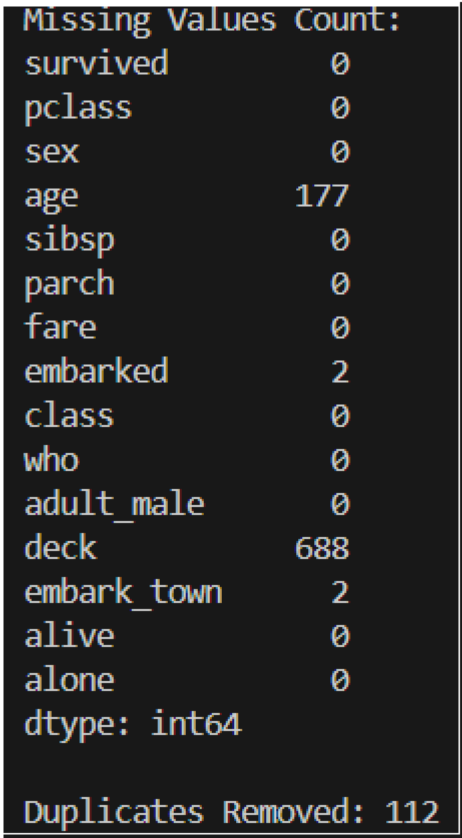
```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
None
```



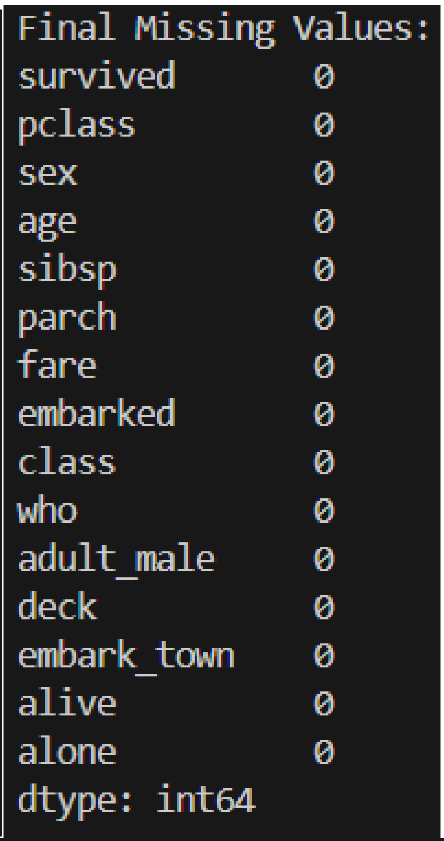
```
Final Dataset Info:
<class 'pandas.core.frame.DataFrame'>
Index: 779 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     779 non-null    int64
 1   pclass       779 non-null    int64
 2   sex          779 non-null    object
 3   age          779 non-null    float64
 4   sibsp        779 non-null    int64
 5   parch        779 non-null    int64
 6   fare         779 non-null    float64
 7   embarked     779 non-null    object
 8   class        779 non-null    object
 9   who          779 non-null    object
 10  adult_male   779 non-null    object
 11  deck         779 non-null    category
 12  embark_town  779 non-null    object
 13  alive        779 non-null    object
 14  alone        779 non-null    object
dtypes: category(1), float64(2), int64(4), object(8)
memory usage: 92.4+ KB
None
```



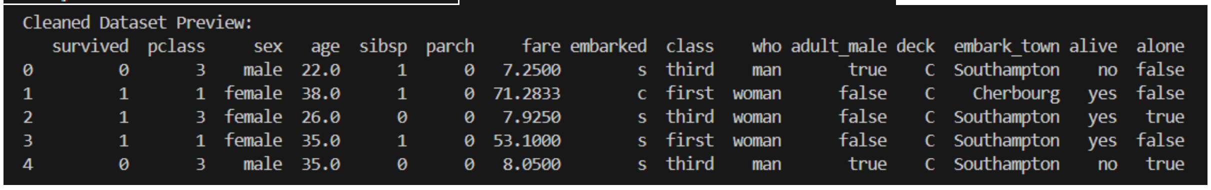
```
Statistical Summary:
         survived      pclass         age       sibsp       parch        fare
count  891.000000  891.000000  714.000000  891.000000  891.000000  891.000000
mean     0.383838    2.308642   29.699118    0.523008    0.381594   32.204208
std      0.486592    0.836071   14.526497    1.102743    0.806057   49.693429
min      0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%      0.000000    2.000000   20.125000    0.000000    0.000000    7.910400
50%      0.000000    3.000000   28.000000    0.000000    0.000000   14.454200
75%      1.000000    3.000000   38.000000    1.000000    0.000000   31.000000
max      1.000000    3.000000   80.000000    8.000000    6.000000  512.329200
```



```
Missing Values Count:
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64

Duplicates Removed: 112
```



```
Final Missing Values:
survived       0
pclass         0
sex            0
age            0
sibsp          0
parch          0
fare           0
embarked       0
class          0
who            0
adult_male     0
deck           0
embark_town    0
alive          0
alone          0
dtype: int64
```



```
Cleaned Dataset Preview:
   survived  pclass     sex   age  sibsp  parch     fare embarked  class    who adult_male deck  embark_town alive  alone
0         0       3    male  22.0      1      0   7.2500        s  third    man       true    C  Southampton    no  false
1         1       1  female  38.0      1      0  71.2833        c  first  woman      false    C    Cherbourg   yes  false
2         1       3  female  26.0      0      0   7.9250        s  third  woman      false    C  Southampton   yes   true
3         1       1  female  35.0      1      0  53.1000        s  first  woman      false    C  Southampton   yes  false
4         0       3    male  35.0      0      0   8.0500        s  third    man       true    C  Southampton    no   true
```

CONCLUSION

Data cleaning is a critical step in data preprocessing.
Among imputation methods, **KNN imputation** is effective because it preserves relationships between variables and provides more accurate estimates compared to simple statistical methods.