**Task 1: Data Cleaning & Preprocessing**

**1).Import the dataset and explore basic info (nulls, data types)?**

//program code is done in spyder.

**ANSWER:-**

import pandas as pd

data=pd.read\_csv('Titanic.csv')

print(data)

# View top rows

print(data.head())

# Check data types and non-null counts

print(data.info())

# Check for null values

print(data.isnull().sum())

# Basic statistics (optional)

print(data.describe())

**description:-** This code begins by importing the Pandas library, which is essential for data analysis in Python. It then reads the Titanic dataset from a CSV file named 'Titanic.csv' into a DataFrame called data, which acts like an in-memory table for structured data. The print(data) line displays the entire dataset, although it's usually better to use data.head() to preview just the first five rows for a quick overview of the columns and data types. The data.info() function provides important metadata about the dataset, including the number of non-null entries and data types for each column—helpful for identifying missing values or incorrect data types. To further investigate data quality, data.isnull().sum() counts the number of missing (null) values in each column. Finally, data.describe() offers basic statistical summaries such as mean, standard deviation, min, max, and quartile values for numerical columns, giving a good sense of the data distribution and any potential anomalies.

OUTPUT IN CONSOLE:-

PassengerId Survived Pclass ... Fare Cabin Embarked

0 1 0 3 ... 7.2500 NaN S

1 2 1 1 ... 71.2833 C85 C

2 3 1 3 ... 7.9250 NaN S

3 4 1 1 ... 53.1000 C123 S

4 5 0 3 ... 8.0500 NaN S

.. ... ... ... ... ... ... ...

886 887 0 2 ... 13.0000 NaN S

887 888 1 1 ... 30.0000 B42 S

888 889 0 3 ... 23.4500 NaN S

889 890 1 1 ... 30.0000 C148 C

890 891 0 3 ... 7.7500 NaN Q

[891 rows x 12 columns]

PassengerId Survived Pclass ... Fare Cabin Embarked

0 1 0 3 ... 7.2500 NaN S

1 2 1 1 ... 71.2833 C85 C

2 3 1 3 ... 7.9250 NaN S

3 4 1 1 ... 53.1000 C123 S

4 5 0 3 ... 8.0500 NaN S

[5 rows x 12 columns]

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 891 entries, 0 to 890

Data columns (total 12 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 PassengerId 891 non-null int64

1 Survived 891 non-null int64

2 Pclass 891 non-null int64

3 Name 891 non-null object

4 Sex 891 non-null object

5 Age 714 non-null float64

6 SibSp 891 non-null int64

7 Parch 891 non-null int64

8 Ticket 891 non-null object

9 Fare 891 non-null float64

10 Cabin 204 non-null object

11 Embarked 889 non-null object

dtypes: float64(2), int64(5), object(5)

memory usage: 83.7+ KB

None

PassengerId 0

Survived 0

Pclass 0

Name 0

Sex 0

Age 177

SibSp 0

Parch 0

Ticket 0

Fare 0

Cabin 687

Embarked 2

dtype: int64

PassengerId Survived Pclass ... SibSp Parch Fare

count 891.000000 891.000000 891.000000 ... 891.000000 891.000000 891.000000

mean 446.000000 0.383838 2.308642 ... 0.523008 0.381594 32.204208

std 257.353842 0.486592 0.836071 ... 1.102743 0.806057 49.693429

min 1.000000 0.000000 1.000000 ... 0.000000 0.000000 0.000000

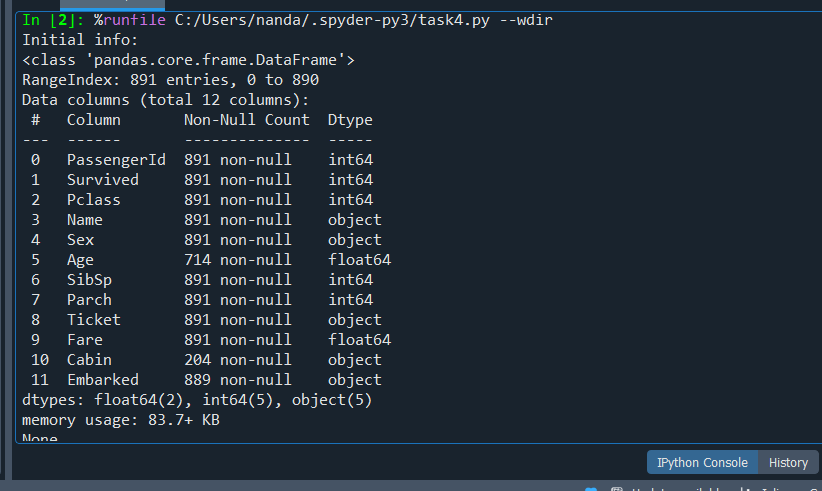
25% 223.500000 0.000000 2.000000 ... 0.000000 0.000000 7.910400

50% 446.000000 0.000000 3.000000 ... 0.000000 0.000000 14.454200

75% 668.500000 1.000000 3.000000 ... 1.000000 0.000000 31.000000

max 891.000000 1.000000 3.000000 ... 8.000000 6.000000 512.329200

[8 rows x 7 columns]



**2)Handle missing values using mean/median/imputation?**

**ANSWER:-**

import pandas as pd

# Load the dataset

df = pd.read\_csv("Titanic.csv")

# Check missing values before imputation

print("Missing values before imputation:\n", df.isnull().sum())

# Handle missing values

df['Age'] = df['Age'].fillna(df['Age'].mean())

df['Fare'] = df['Fare'].fillna(df['Fare'].median())

df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])

df['Cabin'] = df['Cabin'].fillna('Unknown')

# Check missing values after imputation

print("\nMissing values after imputation:\n", df.isnull().sum())

# Show the filled values from each affected column

print("\nFilled 'Age' column (first 10 rows):\n", df['Age'].head(10))

print("\nFilled 'Fare' column (first 10 rows):\n", df['Fare'].head(10))

print("\nFilled 'Embarked' column (first 10 rows):\n", df['Embarked'].head(10))

print("\nFilled 'Cabin' column (first 10 rows):\n", df['Cabin'].head(10))

**description:-** This code loads the Titanic dataset using the Pandas library and focuses on identifying and handling missing values in the data. It starts by reading the "Titanic.csv" file into a DataFrame df and immediately prints the count of missing values in each column using df.isnull().sum(). Then, it performs imputation to fill in the missing data: missing Age values are filled with the mean of the column, Fare values with the median, Embarked with the most frequent value (mode), and Cabin values with the placeholder string 'Unknown'. After these operations, it again prints the updated count of missing values to confirm that they have been handled. Finally, it displays the first 10 entries of each column that was modified, allowing the user to verify that the missing data has been successfully filled. This process helps clean the dataset and prepare it for further analysis or modeling.

**OUTPUT:-**

Missing values before imputation:

PassengerId 0

Survived 0

Pclass 0

Name 0

Sex 0

Age 177

SibSp 0

Parch 0

Ticket 0

Fare 0

Cabin 687

Embarked 2

dtype: int64

Missing values after imputation:

PassengerId 0

Survived 0

Pclass 0

Name 0

Sex 0

Age 0

SibSp 0

Parch 0

Ticket 0

Fare 0

Cabin 0

Embarked 0

dtype: int64

Filled 'Age' column (first 10 rows):

0 22.000000

1 38.000000

2 26.000000

3 35.000000

4 35.000000

5 29.699118

6 54.000000

7 2.000000

8 27.000000

9 14.000000

Name: Age, dtype: float64

Filled 'Fare' column (first 10 rows):

0 7.2500

1 71.2833

2 7.9250

3 53.1000

4 8.0500

5 8.4583

6 51.8625

7 21.0750

8 11.1333

9 30.0708

Name: Fare, dtype: float64

Filled 'Embarked' column (first 10 rows):

0 S

1 C

2 S

3 S

4 S

5 Q

6 S

7 S

8 S

9 C

Name: Embarked, dtype: object

Filled 'Cabin' column (first 10 rows):

0 Unknown

1 C85

2 Unknown

3 C123

4 Unknown

5 Unknown

6 E46

7 Unknown

8 Unknown

9 Unknown

Name: Cabin, dtype: object

**3).Convert categorical features into numerical using encoding?**

**ANSWER:-**

import pandas as pd

# Load the cleaned Titanic dataset

df = pd.read\_csv("Titanic.csv")

# Fill missing values as done before

df['Age'] = df['Age'].fillna(df['Age'].mean())

df['Fare'] = df['Fare'].fillna(df['Fare'].median())

df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])

df['Cabin'] = df['Cabin'].fillna('Unknown')

# Encode 'Sex' using label encoding: male=1, female=0

df['Sex'] = df['Sex'].map({'male': 1, 'female': 0})

# Encode 'Embarked' using one-hot encoding

df = pd.get\_dummies(df, columns=['Embarked'], prefix='Embarked')

# Optional: simplify 'Cabin' to just deck letter (e.g., 'C85' → 'C')

df['Cabin'] = df['Cabin'].apply(lambda x: x[0] if x != 'Unknown' else 'U')

df = pd.get\_dummies(df, columns=['Cabin'], prefix='Cabin')

# Drop non-useful or text-heavy columns (optional)

df.drop(['Name', 'Ticket'], axis=1, inplace=True)

# Display the head of the encoded DataFrame

print(df.head())

**Description:-** This code performs data preprocessing on the Titanic dataset by encoding categorical features into numerical values to make the data suitable for machine learning models. It first reads the dataset and fills any missing values in key columns—Age with the mean, Fare with the median, Embarked with the most common value, and missing Cabin entries with 'Unknown'. It then converts the Sex column into numeric format using label encoding (male as 1 and female as 0). For the Embarked and Cabin columns, it uses one-hot encoding to create separate binary columns for each unique category. Additionally, the Cabin column is simplified by keeping only the first letter (which usually represents the deck level) and replacing 'Unknown' with 'U'. The columns Name and Ticket are dropped since they are either too detailed or not useful for predictive analysis. Finally, it prints the first few rows of the transformed DataFrame to show the result of the encoding process.

**OUTPUT:-**

PassengerId Survived Pclass Sex ... Cabin\_F Cabin\_G Cabin\_T Cabin\_U

0 1 0 3 1 ... False False False True

1 2 1 1 0 ... False False False False

2 3 1 3 0 ... False False False True

3 4 1 1 0 ... False False False False

4 5 0 3 1 ... False False False True

[5 rows x 20 columns]

**4).Normalize/standardize the numerical features?**

**ANSWER:-**

import pandas as pd

import numpy as np

# Step 1: Load the dataset

df = pd.read\_csv("Titanic.csv")

# Step 2: Basic info

print("Initial info:")

print(df.info())

print("\nMissing values before imputation:\n", df.isnull().sum())

# Step 3: Handle missing values

df['Age'] = df['Age'].fillna(df['Age'].mean()) # Use mean for Age

df['Fare'] = df['Fare'].fillna(df['Fare'].median()) # Use median for Fare

df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0]) # Use mode for Embarked

df['Cabin'] = df['Cabin'].fillna('Unknown') # Fill Cabin with 'Unknown'

# Step 4: Check missing values again

print("\nMissing values after imputation:\n", df.isnull().sum())

# Step 5: Encode categorical features

df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})

df['Embarked'] = df['Embarked'].map({'S': 0, 'C': 1, 'Q': 2})

# You can drop columns like Name, Ticket, Cabin if not using them

df = df.drop(columns=['Name', 'Ticket', 'Cabin'])

# Step 6: Normalize numerical features manually (Min-Max Scaling)

numerical\_cols = ['Age', 'Fare', 'SibSp', 'Parch']

for col in numerical\_cols:

df[col] = (df[col] - df[col].min()) / (df[col].max() - df[col].min())

# Step 7: Show final dataset preview

print("\nFinal dataset preview:\n")

print(df.head())

**Description:-** This Python script performs a complete preprocessing pipeline on the Titanic dataset using only built-in Pandas and NumPy functions—making it simple and ideal for basic data analysis tasks without requiring external libraries. It begins by loading the dataset and displaying basic information and missing values. To handle missing data, it fills Age with the mean, Fare with the median, Embarked with the most frequent value (mode), and Cabin with 'Unknown'. Next, it encodes categorical features: converting Sex to numeric (male as 0, female as 1), and mapping embarkation ports S, C, and Q to 0, 1, and 2 respectively. Irrelevant or complex string columns like Name, Ticket, and Cabin are dropped to simplify the dataset. Then, it manually normalizes selected numerical features (Age, Fare, SibSp, Parch) using Min-Max Scaling, bringing their values between 0 and 1. Finally, the script prints a preview of the cleaned and prepared dataset, making it ready for analysis or machine learning modeling.

**OUTPUT:-**

Initial info:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 891 entries, 0 to 890

Data columns (total 12 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 PassengerId 891 non-null int64

1 Survived 891 non-null int64

2 Pclass 891 non-null int64

3 Name 891 non-null object

4 Sex 891 non-null object

5 Age 714 non-null float64

6 SibSp 891 non-null int64

7 Parch 891 non-null int64

8 Ticket 891 non-null object

9 Fare 891 non-null float64

10 Cabin 204 non-null object

11 Embarked 889 non-null object

dtypes: float64(2), int64(5), object(5)

memory usage: 83.7+ KB

None

Missing values before imputation:

PassengerId 0

Survived 0

Pclass 0

Name 0

Sex 0

Age 177

SibSp 0

Parch 0

Ticket 0

Fare 0

Cabin 687

Embarked 2

dtype: int64

Missing values after imputation:

PassengerId 0

Survived 0

Pclass 0

Name 0

Sex 0

Age 0

SibSp 0

Parch 0

Ticket 0

Fare 0

Cabin 0

Embarked 0

dtype: int64

Final dataset preview:

PassengerId Survived Pclass Sex ... SibSp Parch Fare Embarked

0 1 0 3 0 ... 0.125 0.0 0.014151 0

1 2 1 1 1 ... 0.125 0.0 0.139136 1

2 3 1 3 1 ... 0.000 0.0 0.015469 0

3 4 1 1 1 ... 0.125 0.0 0.103644 0

4 5 0 3 0 ... 0.000 0.0 0.015713 0

[5 rows x 9 columns]

**5)Visualize outliers using boxplots and remove them?**

**ANSWER:-**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Load dataset

df = pd.read\_csv("Titanic.csv")

# Handle missing values

df['Age'] = df['Age'].fillna(df['Age'].mean())

df['Fare'] = df['Fare'].fillna(df['Fare'].median())

df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])

df['Cabin'] = df['Cabin'].fillna('Unknown')

# Encode categorical variables

df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})

df['Embarked'] = df['Embarked'].map({'S': 0, 'C': 1, 'Q': 2})

# Drop unnecessary columns

df = df.drop(columns=['Name', 'Ticket', 'Cabin'])

# Plot boxplots using matplotlib

numeric\_cols = ['Age', 'Fare', 'SibSp', 'Parch']

plt.figure(figsize=(12, 6))

for i, col in enumerate(numeric\_cols):

plt.subplot(2, 2, i+1)

plt.boxplot(df[col])

plt.title(f'Boxplot of {col}')

plt.ylabel(col)

plt.tight\_layout()

plt.show()

# Function to remove outliers using IQR

def remove\_outliers\_iqr(dataframe, column):

Q1 = dataframe[column].quantile(0.25)

Q3 = dataframe[column].quantile(0.75)

IQR = Q3 - Q1

lower\_limit = Q1 - 1.5 \* IQR

upper\_limit = Q3 + 1.5 \* IQR

return dataframe[(dataframe[column] >= lower\_limit) & (dataframe[column] <= upper\_limit)]

# Remove outliers from each numerical column

for col in numeric\_cols:

df = remove\_outliers\_iqr(df, col)

# Output cleaned data shape and preview

print("\nShape after outlier removal:", df.shape)

print("\nPreview of cleaned data:\n")

print(df.head())

**Description:-** This Python script begins by loading the Titanic dataset and handling missing values by filling in the mean for Age, the median for Fare, the mode for Embarked, and replacing missing Cabin values with 'Unknown'. It then encodes categorical variables: converting Sex into numerical format (male as 0, female as 1) and mapping Embarked ports (S, C, Q) to 0, 1, and 2 respectively. Unnecessary textual columns like Name, Ticket, and Cabin are dropped to simplify the dataset. The script proceeds to visualize the distribution and detect outliers in numerical columns (Age, Fare, SibSp, and Parch) using boxplots created with Matplotlib. Outliers are then removed using the IQR method, which filters out values lying beyond 1.5 times the interquartile range from Q1 and Q3. Finally, the shape and a preview of the cleaned dataset are printed, ensuring that the data is ready for further analysis or modeling.

**OUTPUT:-**

Shape after outlier removal: (577, 9)

Preview of cleaned data:

PassengerId Survived Pclass Sex ... SibSp Parch Fare Embarked

0 1 0 3 0 ... 1 0 7.2500 0

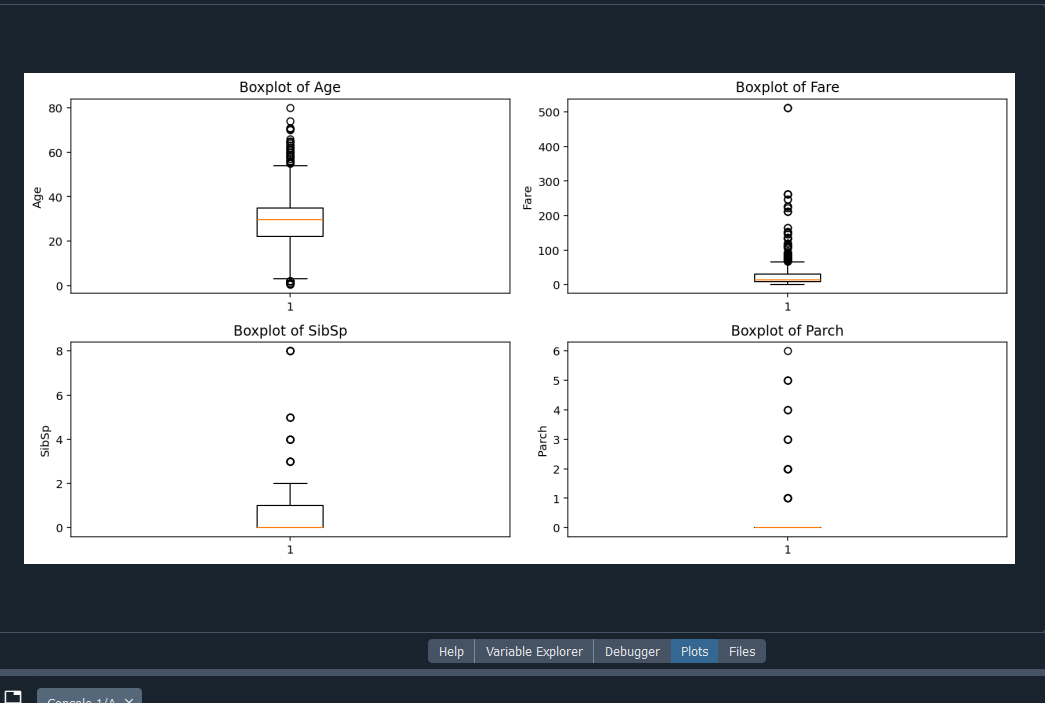
2 3 1 3 1 ... 0 0 7.9250 0

3 4 1 1 1 ... 1 0 53.1000 0

4 5 0 3 0 ... 0 0 8.0500 0

5 6 0 3 0 ... 0 0 8.4583 2

[5 rows x 9 columns]



Some screenshots of the tasks

