Data Preprocessing

```
Data preprocessing is a crucial step in data science that involves transformaion raw data into
a format suitable for analysis .It improves the quality and structure of the dataset to ensure
that models can make accurate predictions.
Here is an extensive overview of data preprocessing ,starting with "Data Cleaning",followed by
other sub-tasks:
```

1 . Data Cleaning

What is Data Cleaning?

```
Data cleaning is the process of detecting and correcting or removing
errors,inconsisitences,and inaccuracies from datasets to improve data quality.It ensures that
the data used for analysis is free of niose ,incomplete values and duplicate entries.
```

Why is Data Cleaning Important?

```
Real-world data is often messy and can contain missing values,error or outliers.Cleaning data
ensures that your analysis or model is built on reliable ,consistent ,and valid data,leading
to more accurate and insightful results.
```

Sub-tasks of Data Cleaning:

1.Handling Missing Data

```
* Missing Data is a common problem that occurs when some observations in the dataset lack a
certain value.
* Why-> Algorithms cannot process missing values directly.
* Methods:
     * Removal:-> Deleting rows or columns with missing data if they are few.
     * Imputation: Replacing missing values with mean ,median ,mode ,or more complex methods
like interpolation.
```

Example:

In [47]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

In [192…
```
df=pd.DataFrame({'Name':['Alice','Srinu','Mani','Sai',None],
                'Age':[25,None,30,22,23],
                'Salary':[50000,60000,None,58000,55000]})
```

In [21]:
```
df.head()
```

Out[21]:

	Name	Age	Salary
0	Alice	25.0	50000.0
1	Srinu	NaN	60000.0
2	Mani	30.0	NaN
3	Sai	22.0	58000.0
4	None	23.0	55000.0

Remove rows with missing data

In [194…
```
df_cleaned=df.dropna()
```

In [196…
```
df_cleaned
```
Out[196…

	Name	Age	Salary
0	Alice	25.0	50000.0
3	Sai	22.0	58000.0

Impute missing values with the mean

In [198…
```
df['Age'].fillna(df['Age'].mean(),inplace=True)
df['Salary'].fillna(df['Salary'].mean(),inplace=True)
df['Name'].fillna(df['Name'].mode()[0],inplace=True)
```
In [70]:
```
df.head()
```
Out[70]:

	Name	Age	Salary
0	Alice	25.0	50000.0
1	Srinu	25.0	60000.0
2	Mani	30.0	55750.0
3	Sai	22.0	58000.0
4	Alice	23.0	55000.0

2.Removing Duplicates

```
* Why : Duplicates data can distort the analysis.
*Real Example : Suppose you are analyzing customer data,and a customer's record appears
multiple times.This can affect the accuarcy of customer segmantation models.
```

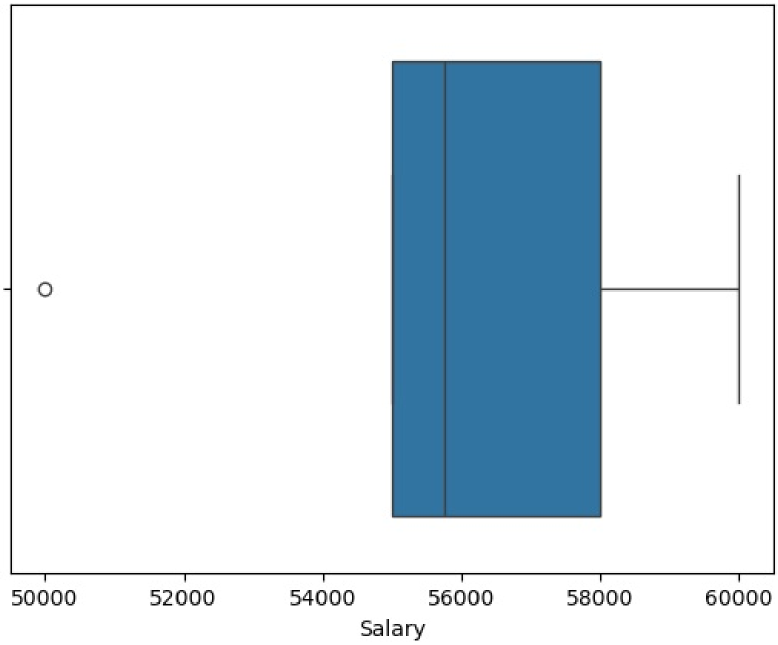
In [72]:
```
df.drop_duplicates(inplace=True)
```
In [84]:
```
df.duplicated().sum()
```
Out[84]: 0

3.Handling Outliers

```
* Why : Outliers can skew statistical models and provide misleading results.
* Method :
         * Removing : Detect and remove outliers based on the IQR(Interquartile Range) or Z-
scores.
         * Transformation: Normalize the data to reduce the impact of outliers.
```

Detecting and handling outliers using Z-score

In [95]:
```
sns.boxplot(data=df,x='Salary')
plt.show()
```



In [93]:
```
from scipy import stats
import numpy as np
df['zscore']=np.abs(stats.zscore(df['Salary']))
df_no_outliers=df[df['zscore']<3] #Removing Outliers
```

In [ ]:
```
Q1=df.quantile(0.25)
Q3=df.quantile(0.75)
IQR=Q3-Q1
lowerbound=Q1-1.5*IQR
upperbound=Q3+1.5*IQR
```

4.Correcting Data Types

```
* why : Incorrect data types (e.g.,numeric values stored as strings) can prevent analysis or
cause compuatational errors.
```

In [102…
```
df['Age']=pd.to_numeric(df['Age'],errors='coerce')
#Coerce invalid types to NaN
```

In [106…
```
df.dtypes
```

Out[106…
```
Name        object
Age        float64
Salary     float64
zscore     float64
dtype: object
```

5.Addressing Inconsistencies

```
* Why :-> Inconsistent data(e.g., different date formats or inconsistence text capitilazation
) can lead to incorrect grouping or analysis.
* Real Example :-> A dataset where "New York" and "new york" are considered different
locations.
```

In [ ]:
```
df['City']=df['City'].str.lower()
```

2. Data Integration

What is Data Integration?

```
Data integration involves combining data from multiple sources into a unfied dataset.It is
essential when you're working with data from different databases or systems.
```

Why is Data Integration Important?

```
In real-world projects ,data often comes from various sources ,such as transactional systems,customer relationship management (CRM) tools,platfroms .Integration these datasets ensures you can anslyse all the data comprehensively.
```

Methods:

1. Merging Datasets:

```
Combining two or more datasets based on a common column(e.g., coustomerId)
```

2.Concatenating Datasets:

```
Stacking datasets on top of one another(i.e adding rows).
```

3.Joining Tables:

```
Combining tables using different join types(inner,outer,left,right)
```

Example:

Merging two datasets

In [122…
```
df_customers=pd.DataFrame({'customer_id':[1,2,3],'name':['vasu','Sai','Mani']})

df_orders=pd.DataFrame({'customer_id':[1,2,3],'order_value':[100,200,300]})
df_merged=pd.merge(df_customers,df_orders ,on='customer_id')
```

In [124…
```
df_merged
```

Out[124…

	customer_id	name	order_value
0	1	vasu	100
1	2	Sai	200
2	3	Mani	300

3.Data Transformation

What is Data Transformation?

```
Data transformation is the process of converting data into a format that is more appropriate for analysis.It involves scaling ,encoding categorical data,and feature engineering.
```

Why is Data Transformation Important?

```
Raw data often need to be normalized or encoding into a format suitable for machine learning alogorithms ,which work best with numeric and scaled data.
```

Sub-tasks of Data Transformation:

1.Normalization and Scaling:

```
* Why:-> Some algorithms (like distance -based models) are sensitive to the scale of feature.
* Methods:->
         * Min-Max Scaling : Rescales data to a range[0,1].
         * Standardization : Rescales data so that it has a mean of 0 and standard deviation
of 1.
```

In [136…
```
from sklearn.preprocessing import MinMaxScaler,StandardScaler
scaler =MinMaxScaler()
df_scaled=scaler.fit_transform(df[['Age','Salary']])
```

In [141…
```
scaler=StandardScaler()
df_scale=scaler.fit_transform(df[['Age','Salary']])
```

2. Encoding Categorical Data

* Why :

```
Machine Learning models cannot process non-numeric data.
```

* Methods:

1. Label Encoding:

```
Converts categories to numeric labels.
```

2. One-Hot-Encoding :

```
Creates binary columns for each category.
```

In [155…
```
df_encoding=pd.get_dummies(df,columns=['Name'])
```

In [157…
```
df_encoding
```

Out[157…

	Age	Salary	zscore	Name_Alice	Name_Mani	Name_Sai	Name_Srinu
0	25.0	50000.0	1.706750	True	False	False	False
1	25.0	60000.0	1.261511	False	False	False	True
2	30.0	55750.0	0.000000	False	True	False	False
3	22.0	58000.0	0.667859	False	False	True	False
4	23.0	55000.0	0.222620	True	False	False	False

3.Feature Engineering

Why :

```
Creating new features based on existing ones can improve model performance
```

Real Example:

```
Creating a total_spent feature from quantity and price.
```

In [167…
```
df=pd.DataFrame({'Items':['biryani','dosa','puri','rice'],
                 'Order_id':[21,23,21,23],
                 'Price':[1000,200,300,200],
                 'Quantity':[10,2,2,5]})
df
```

Out[167…

	Items	Order_id	Price	Quantity
0	biryani	21	1000	10
1	dosa	23	200	2
2	puri	21	300	2
3	rice	23	200	5

In [169…
```
df['total_spent']=df['Quantity']*df['Price']
```

In [171…
```
df
```

Out[171…

	Items	Order_id	Price	Quantity	total_spent
0	biryani	21	1000	10	10000
1	dosa	23	200	2	400
2	puri	21	300	2	600
3	rice	23	200	5	1000

4.Data Reduction

What is Data reduction?

Data reduction techniques aim to reduce the amount of data without losing significant information. this improves the efficiency of the analysis.

Why is Data Reduction Important?

Handling large datasets can be computationally expensive ,so reducing the dataset size helps in faster and more efficient processing.

Methods:

1.Dimentionality Reduction:

Reducing the number of features using methods like Principle Component Component Analysis(PCA).

2.Aggregation:

Summerizing the data (e.g., calculating totals or averages) to reduce the dataset's granually,

Example:

In [200…
```
from sklearn.decomposition import PCA
pca=PCA(n_components=2)
df_reduced=pca.fit_transform(df[['Age','Salary']])
```

In [206…
```
df_reduced
```

Out[206…
```
array([[ 5.74999998e+03, -5.31938675e-01],
       [-4.24999998e+03,  3.93172064e-01],
       [ 4.62555369e-04,  4.99999998e+00],
       [-2.25000027e+03, -2.79185007e+00],
       [ 7.49999812e+02, -2.06938330e+00]])
```

5.Data Discretization

What is Data Discretization?

Data Discretization involves transforming continuous data into dicrete intervals or categories

Why is Data Discretization Important?

Some algorithms work better with discrete value.Discretization can also make the resluts easir to interpret by grouping continuous data into ranges.

Example:

In [ ]:
```
#Binning age into categories
```

In [214…
```
bins=[0,18,35,60,100]
labels=['Child','Young Adult','Adult','Senior']
df['Age_Group']=pd.cut(df['Age'],bins=bins,labels=labels)
```

In [216…
```
df
```

Out[216…

	Name	Age	Salary	Age_Group
0	Alice	25.0	50000.0	Young Adult
1	Srinu	25.0	60000.0	Young Adult
2	Mani	30.0	55750.0	Young Adult
3	Sai	22.0	58000.0	Young Adult
4	Alice	23.0	55000.0	Young Adult

Real-Time Example of Data Preprocessing

```
Imagine we are working with a dataset of customer transactions for a retail company.The
dataset includes customer details,product information ,and purchase history.You aim tp predict
customer churn(weather a customer will stop purchasing).
```

1.Data Cleaning:

```
* Handle missing values in the 'Age' and 'Salary' columns.
* Remove duplicates entries where same transactions is recorded twice.
* Detect and remove outliers in the 'order_value' column.
```

2.Data Integration:

```
* Merge customer demographic data with transactions data.
* Combine external datasets,such as customer feedback survays.
```

3.Data Transformation:

```
* Scale the 'order_value' and 'customer_tenure' columns to ensure they are on a similar scale.
* Encode the 'customer_type' (regular,new,VIP) using one-hot encoding.
```

4.Data Reduction:

```
* Use PCA to reduce the dimentionality of features like 'customer_activity' and
'purchase_history'.
```

5.Data Discretization:

```
    * Group the 'customer_tenure' into bins such as 'new customer','medium tenure' and 'long
tenure'
```

```
This comprehensive preprocessing will prepare the data for machine learning models,ensurig
that is clean,consistent ,and well-structured for analysis.

Let's use a real-world dataset to apply all the preprocessing techniques mentioned above.For
this ,I'll use the famous  'Titanic' dataset,Which contains information about the passengers
on the Titanic ,such as age,sex,class,fare.,etc., and whether they survived or not.This
Dataset is avialable in the Seaborn Library


We'll perform the following steps:

    1.Data Cleaning
    2.Data Integration
    3.Data Transformation
    4.Data Reduction
    5.Data Discretization
```

Step 1. Loading the Titanic Dataset.

In [42]:
```python
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import warnings
warnings.filterwarnings("ignore")
```

In [114…
```python
#Load the Titanic dataset from Seaborn
df=sns.load_dataset('titanic')
#Display the first few records of the dataset.
df.head()
```

Out[114…

	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

In [116…

```
df.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
```

Insights about Dataset

```
The dataset includes the folllowing columns:
    * survived: 1 if the passenger survived,0 otherwise
    * pclass  : Passenger class(1st,2nd,3rd).
    * sex     : Gender
    * age     : Age in years.
    * sibsp   : Number of siblings/spouses aboard.
    * parch   : Number of parents/children aboard.
    * fare    : Passenger fare.
    * embarked: Port of embarkation(C=Cherbourg;Q=Queenstown;S=Southampton)
    * deck    : Deck level
    * embark_town: Town of embarkation.
```

Step 2. Data Cleaning

2.1 Handling Missing Data

```
Let's inspect for missing values and handle them.
```

In [118…

```
#checking for missing values
df.isnull().sum()
```

Out[118…

```
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
```

In [120…
```
#filling missing 'age' values with the mean
df['age'].fillna(df['age'].mean(),inplace=True)
```

In [122…
```
#Dropping columns with too many missing values (like 'deck')
df.drop(columns=['deck'],inplace=True)
```

In [124…
```
#Dropping records with missing 'embarked'
df.dropna(subset=['embarked'],inplace=True)
```

In [126…
```
df.isnull().sum()
```

Out[126…
```
survived       0
pclass         0
sex            0
age            0
sibsp          0
parch          0
fare           0
embarked       0
class          0
who            0
adult_male     0
embark_town    0
alive          0
alone          0
dtype: int64
```

2.2 Removing Duplicates

In [128…
```
#check for duplicates
print(f"Number of duplicates rows:{df.duplicated().sum()}")
```

```
Number of duplicates rows:111
```

In [130…
```
#Removing duplicates rows if found
df.drop_duplicates(inplace=True)
```

In [132…
```
df.duplicated().sum()
```

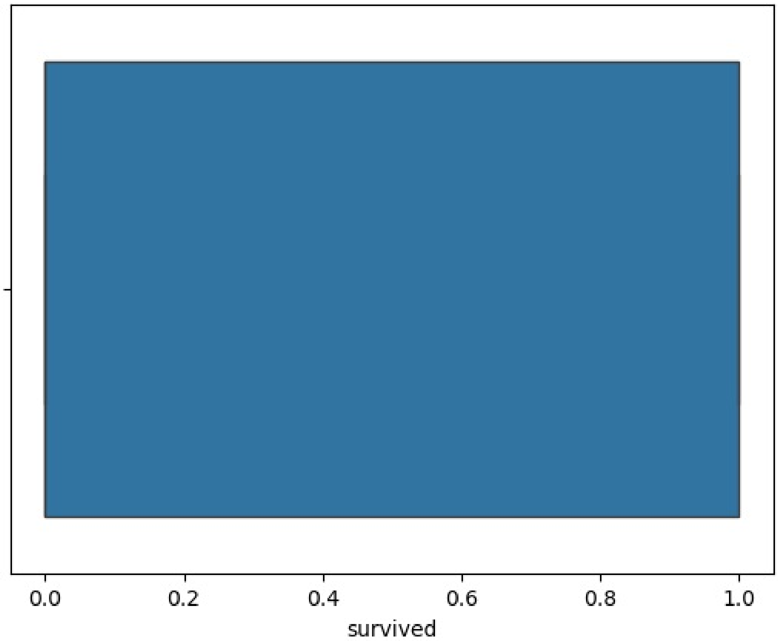
Out[132…
```
0
```

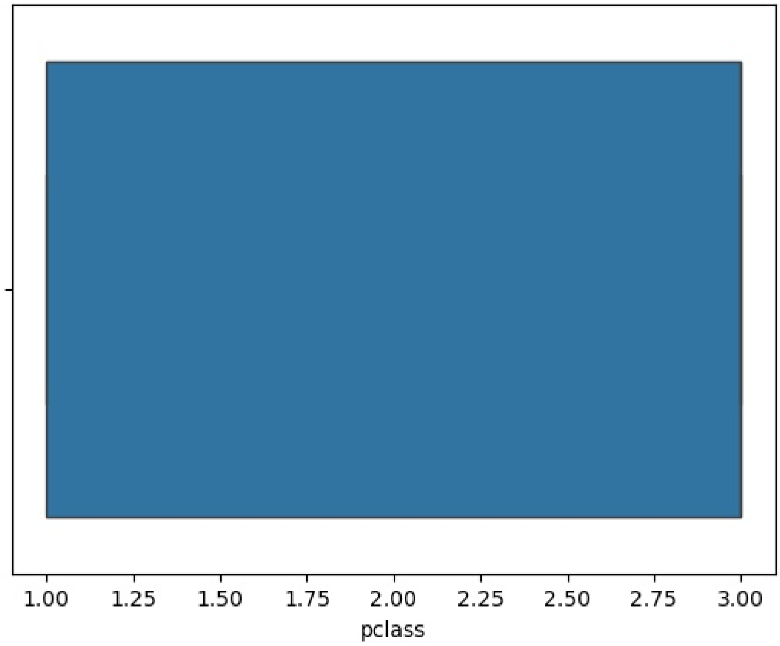
In [134…
```
df.shape
```

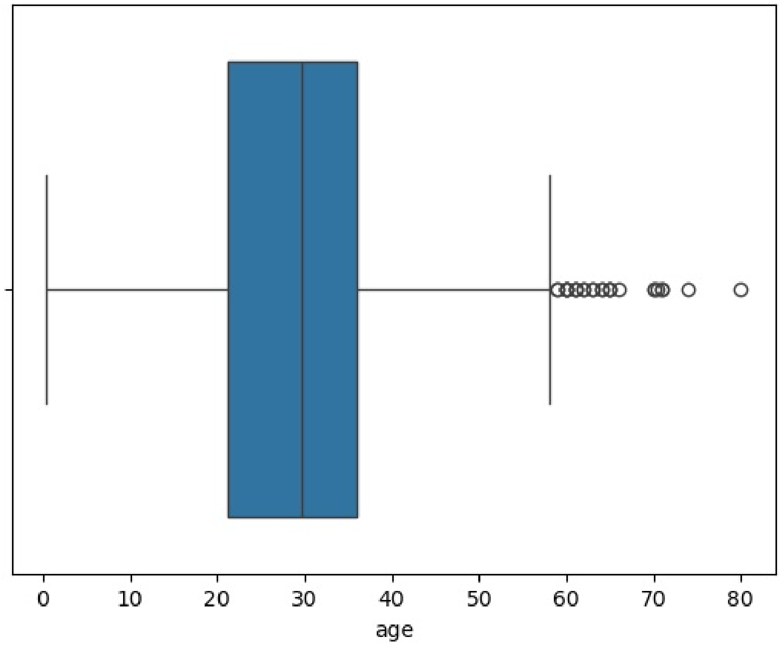
Out[134…
```
(778, 14)
```

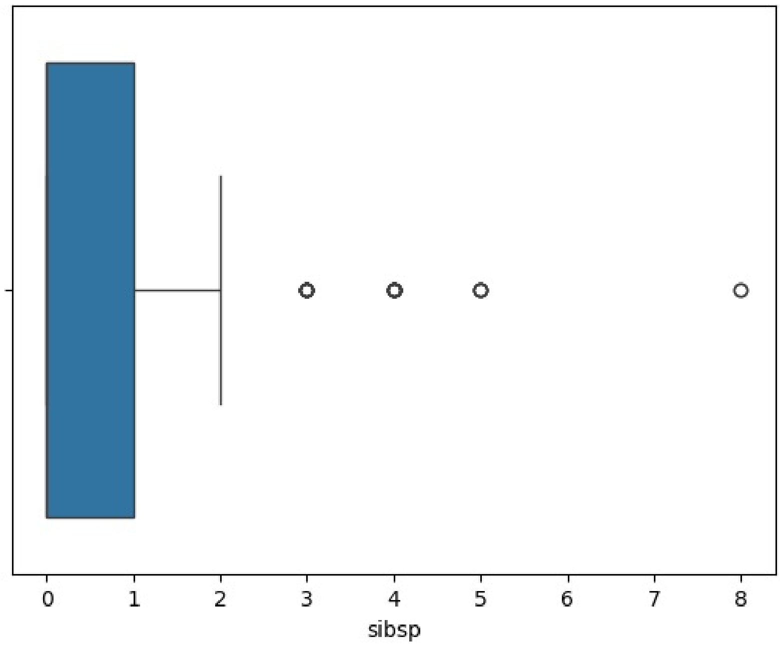
2.3 Handling Outliers

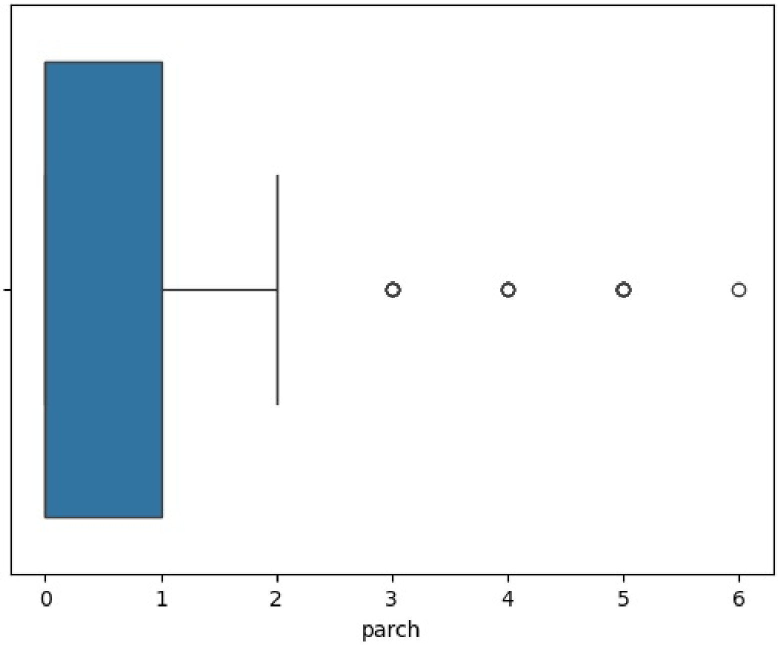
In [136…
```
for i in df.select_dtypes(include='number').columns:
    sns.boxplot(data=df,x=i)
    plt.show()
```

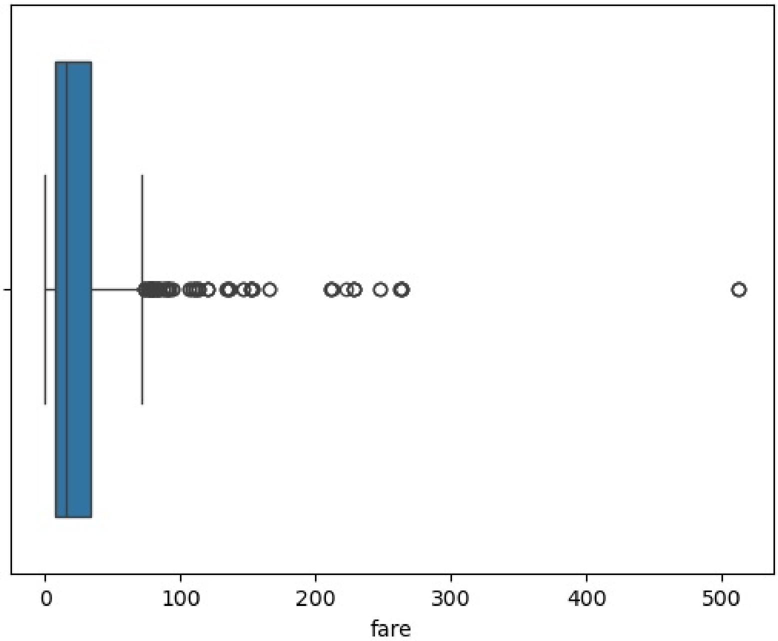



1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
pclass


0
10
20
30
40
50
60
70
80
age


0
1
2
3
4
5
6
7
8
sibsp





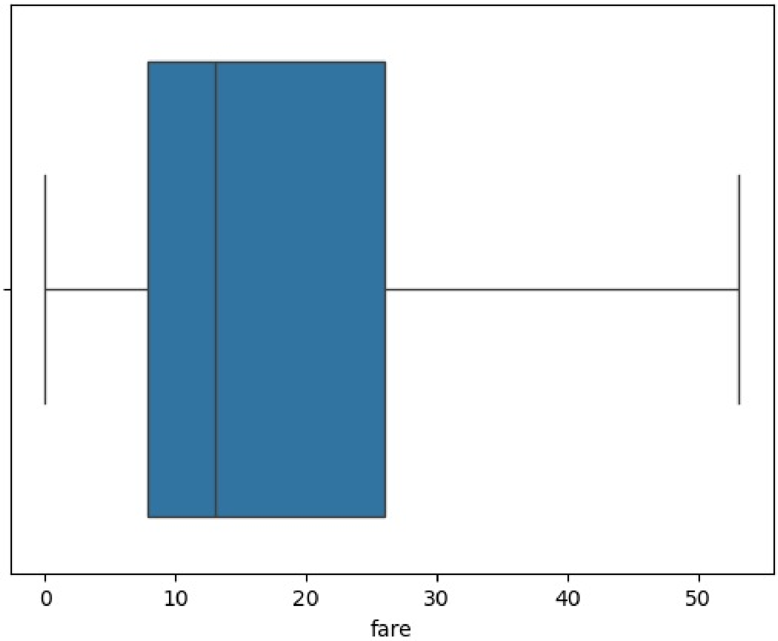
We will use IQR method to detect and remove outliers in the 'fare' column.

In [144…
```
# Detecting outliers using IQR in 'fare'
Q1 = df['fare'].quantile(0.25)
Q3 = df['fare'].quantile(0.75)
IQR = Q3 - Q1

# Removing outliers
df = df[~((df['fare'] < (Q1 - 1.5 * IQR)) | (df['fare'] > (Q3 + 1.5 * IQR)))]

# Check if outliers are removed
print(df.shape)
```

```
(647, 14)
```

In [146…
```
sns.boxplot(data=df,x='fare')
```

Out[146… <Axes: xlabel='fare'>



Step3.Data Integration

For this dataset,we already have a unified table,so there's no need for merging or concatenating additional datasets.However ,if we had more data sources,we would use techniques like merging or joining.

Step4.Data Transformation

4.1 Scaling Numeric Data

We will scale 'fare' and 'age' columns using Min-Max scaling

In [148…

```
from sklearn.preprocessing import MinMaxScaler
#Initialize MinMaxScaler
scaler=MinMaxScaler()

#Scaler 'fare' and 'age'
df[['age','fare']]=scaler.fit_transform(df[['age','fare']])

#verify the scaled columns
df[['age','fare']].head()
```

Out[148…

	age	fare
0	0.271174	0.136535
2	0.321438	0.149247
3	0.434531	1.000000
4	0.434531	0.151601
5	0.367921	0.159290

4.2 Encoding Categorical Data

We will encode the categorical columns (sex,embarked,class) using one-hot encoding.

In [157…

```
df[['sex','embarked','class']].head()
```

Out[157...

	sex	embarked	class
0	male	S	Third
2	female	S	Third
3	female	S	First
4	male	S	Third
5	male	Q	Third

In [159...

```
#One-hot encoding for categorical columns
df=pd.get_dummies(df,columns=['sex','embarked','class'],drop_first=True)
```

In [161...

```
df.head()
```

Out[161...

	survived	pclass	age	sibsp	parch	fare	who	adult_male	embark_town	alive	alone	sex_male	embarked_Q	em
0	0	3	0.271174	1	0	0.136535	man	True	Southampton	no	False	True	False	
2	1	3	0.321438	0	0	0.149247	woman	False	Southampton	yes	True	False	False	
3	1	1	0.434531	1	0	1.000000	woman	False	Southampton	yes	False	False	False	
4	0	3	0.434531	0	0	0.151601	man	True	Southampton	no	True	True	False	
5	0	3	0.367921	0	0	0.159290	man	True	Queenstown	no	True	True	True	

4.3 Feature Engineering

Let's create a new feature 'family_size' by combining 'sibsp' and 'parch' to represent the total number of family members aboard.

In [166...

```
# Create 'family_size' feature

df['family_size']=df['sibsp']+df['parch']
```

In [184...

```
#verify the new feature
df[['sibsp','parch','family_size']].iloc[10:20]
```

Out[184...

	sibsp	parch	family_size
11	0	0	0
12	0	0	0
13	1	5	6
14	0	0	0
15	0	0	0
16	4	1	5
17	0	0	0
18	1	0	1
19	0	0	0
20	0	0	0

Step 5 . Data Reduction

We will use Priciple Component Analysis(PCA) to reduce the dimetionality of the dataset.Before doing that,let's remove the target variable 'survived' and the non-numeric columns

In [188...

```
df.dtypes
```

Out[188…
```
survived          int64
pclass            int64
age             float64
sibsp             int64
parch             int64
fare            float64
who              object
adult_male         bool
embark_town      object
alive            object
alone              bool
sex_male           bool
embarked_Q         bool
embarked_S         bool
class_Second       bool
class_Third        bool
family_size       int64
dtype: object
```

In [200…
```
from sklearn.decomposition import PCA

#Removing non-numeric columns and the target 'survived'
X=df.drop(columns=['survived','who','adult_male','alive','alone','embark_town'])
```

In [202…
```
#performing PCA to reduce dimentionality
pca=PCA(n_components=2)
X_reduced=pca.fit_transform(X)
```

In [208…
```
X_reduced[:5]
```

Out[208…
```
array([[ 0.23489036, -0.72428178],
       [-0.95988096, -0.74332515],
       [ 0.13051776,  1.53370458],
       [-1.02415036, -0.77653925],
       [-1.03846561, -0.93256184]])
```

In [210…
```
df.head()
```

Out[210…

	survived	pclass	age	sibsp	parch	fare	who	adult_male	embark_town	alive	alone	sex_male	embarked_Q	em
0	0	3	0.271174	1	0	0.136535	man	True	Southampton	no	False	True	False	
2	1	3	0.321438	0	0	0.149247	woman	False	Southampton	yes	True	False	False	
3	1	1	0.434531	1	0	1.000000	woman	False	Southampton	yes	False	False	False	
4	0	3	0.434531	0	0	0.151601	man	True	Southampton	no	True	True	False	
5	0	3	0.367921	0	0	0.159290	man	True	Queenstown	no	True	True	True	

Step 6 . Data Discretization

We will discretize the 'age' column into categories such as 'child' and 'Young Adult' ,'Adult','Senior'

In [214…
```
#Binning 'age' into categories
bins=[0,18,35,60,100]
labels=['child','Young Adult','Adult','Senior']
df['age_group']=pd.cut(df['age'],bins=bins,labels=labels)
```

In [216…
```
# Verify the new 'age_group' column
print(df[['age', 'age_group']].head())
```

```
        age age_group
0  0.271174     child
2  0.321438     child
3  0.434531     child
4  0.434531     child
5  0.367921     child
```

Summary

1. Data Cleaning:

We handled missing values, removed duplicates, and dealt with outliers.

2. Data Integration:

We worked with a single dataset, but integration is important when multiple datasets are involved.

3. Data Transformation:

We scaled numeric data, encoded categorical variables, and engineered new features like family_size.

4. Data Reduction:

We applied PCA to reduce the dimensionality of the dataset.

5. Data Discretization:

We binned the age column into categories for better interpretability.

This comprehensive preprocessing workflow ensures that the data is clean, well-structured, and ready for analysis or model building.

In [223…
```
df.head()
```

Out[223…

	survived	pclass	age	sibsp	parch	fare	who	adult_male	embark_town	alive	alone	sex_male	embarked_Q	em
0	0	3	0.271174	1	0	0.136535	man	True	Southampton	no	False	True	False	
2	1	3	0.321438	0	0	0.149247	woman	False	Southampton	yes	True	False	False	
3	1	1	0.434531	1	0	1.000000	woman	False	Southampton	yes	False	False	False	
4	0	3	0.434531	0	0	0.151601	man	True	Southampton	no	True	True	False	
5	0	3	0.367921	0	0	0.159290	man	True	Queenstown	no	True	True	True	

In [ ]:

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js