**PRACTICAL – 3**

**AIM**: Write a code to read data from the different file formats like JSON, HTML, and CSV files and check for missing data and outlier values and handle them.

**THEORY**

• Different file format

CSV (or Comma Separated Value) file is the most common type of file that a data scientist will ever work with. These files use a “,” as a delimiter to separate the values and each row in a CSV file is a data record.

HTML ( Hyper Text Markup Language) the standard markup language for creating Web pages. It describes the structure of a Web page and consists of a series of elements. It’s elements tell the browser how to display the content.

JSON (JavaScript Object Notation) files are lightweight and human-readable to store and exchange data. It is easy for machines to parse and generate these files and are based on the JavaScript programming language. JSON files store data within {} similar to how a dictionary stores it in Python.

• Missing and Outlier value treatment

Data Cleaning is the process of finding and correcting the inaccurate/incorrect data that are present in the dataset. Missing values are usually represented in the form of Nan or null or None in the dataset. an Outlier is an observation in a given dataset that lies far from the rest of the observations. That means an outlier is vastly larger or smaller than the remaining values in the set. If our dataset is small, we can detect the outlier by just looking at the dataset.

**SOURCE CODE AND OUTPUT**

1. **`Important labour for data cleaning**

import numpy as np

import pandas as pd

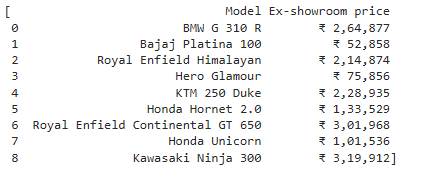
import matplotlib.pyplot as plt

import seaborn as sns

1. **Read data from the different file formats like JSON, HTML, and CSV files**
2. **HTML file**

Bikes\_df = pd.read\_html("/content/sample\_data/Best Bikes in India - April 2022 \_ Top 10 Bikes - BikeWale.html")

Bikes\_df



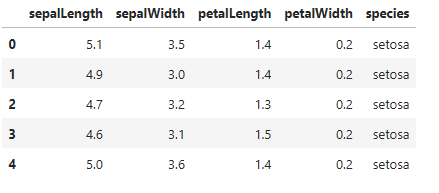
1. **JSON file**

iris\_df = pd.read\_json("/content/sample\_data/iris.json")

iris\_df.shape

(150, 5)

iris\_df.head()



1. **CSV file**

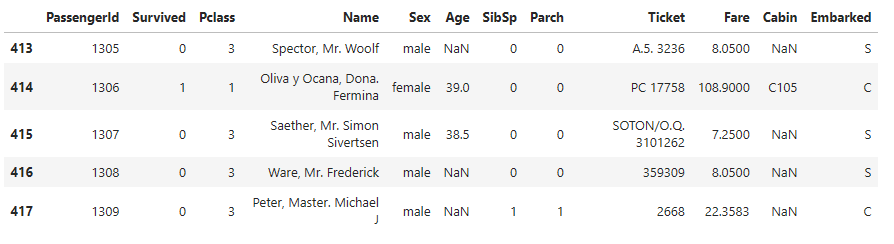
tit\_df = pd.read\_csv("/content/sample\_data/Titanic dataset.csv")

tit\_df.shape

(418, 12)

tit\_df.head()

tit\_df.tail()

****

tit\_df["Survived"].unique()

array([0, 1])

tit\_df["Survived"].value\_counts()

0 266

1 152

Name: Survived, dtype: int64

tit\_df["Pclass"].value\_counts()

3 218

1 107

2 93

Name: Pclass, dtype: int64

tit\_df['Pclass'].unique()

array([3, 2, 1])

tit\_df["Sex"].value\_counts()

male 266

female 152

Name: Sex, dtype: int64

tit\_df.isnull().sum()

PassengerId 0

Survived 0

Pclass 0

Name 0

Sex 0

Age 86

SibSp 0

Parch 0

Ticket 0

Fare 1

Cabin 327

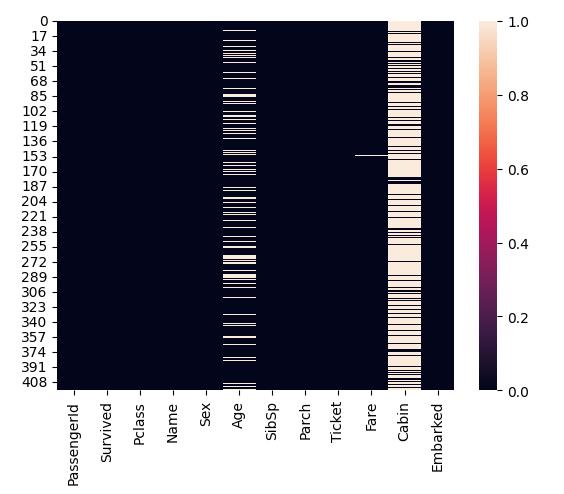
Embarked 0

dtype: int64

**# To check the null value in tit\_dif the white line in coloumn (Cabin,Fare,Age)**

sns.heatmap(tit\_df.isnull())

plt.show()

****

**# No. of rows**

tit\_df.shape[0]

418

**#No. of columns**

tit\_df.shape[1]

12

**# Number of rows in the DataFrame**

tit\_df.isnull().sum()\*100/tit\_df.shape[0]

PassengerId 0.000000

Survived 0.000000

Pclass 0.000000

Name 0.000000

Sex 0.000000

Age 20.574163

SibSp 0.000000

Parch 0.000000

Ticket 0.000000

Fare 0.239234

Cabin 78.229665

Embarked 0.000000

dtype: float64

# **Display the data types**

tit\_df.dtypes

PassengerId int64

Survived int64

Pclass int64

Name object

Sex object

Age float64

SibSp int64

Parch int64

Ticket object

Fare float64

Cabin object

Embarked object

dtype: object

**# Count of unique values**

tit\_df["Age"].value\_counts()

21.0 17

24.0 17

22.0 16

30.0 15

18.0 13

..

76.0 1

28.5 1

22.5 1

62.0 1

38.5 1

Name: Age, Length: 79, dtype: int64

**# Find the mode**

tit\_df["Age"].mode()

0 21.0

1 24.0

Name: Age, dtype: float64

**#Filiing nul values of age in average/mean of age permenant**

m = np.round(tit\_df["Age"].mean())

m

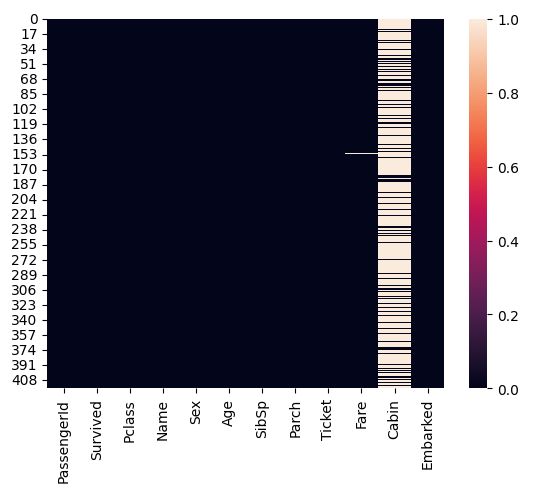
30.0

tit\_df["Age"].fillna(m, inplace=True)

**# To cheack the null value in tit\_dif the white line in coloumn (Cabin,Fare,Age)**

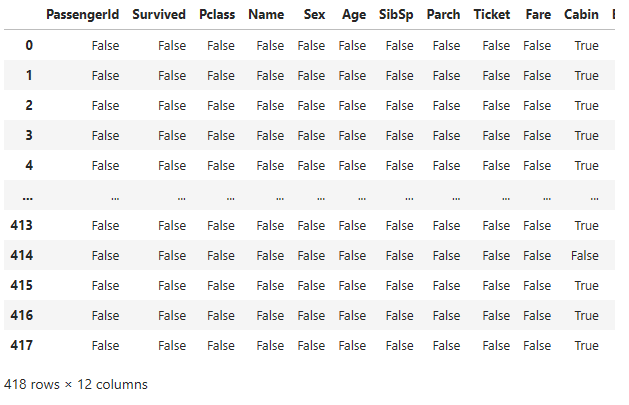
sns.heatmap(tit\_df.isnull())

plt.show()

****

**# Create a Boolean mask**

tit\_df.isnull()

****

**# Obtain a count of unique**

tit\_df["Fare"].value\_counts()

7.7500 21

26.0000 19

13.0000 17

8.0500 17

7.8958 11

..

7.8208 1

8.5167 1

78.8500 1

52.0000 1

22.3583 1

Name: Fare, Length: 169, dtype: int64

tit\_df["Fare"].mode()

0 7.75

Name: Fare, dtype: float64

n = tit\_df['Fare'].mode()

n

0 7.75

Name: Fare, dtype: float64

tit\_df["Fare"].fillna(" 7.75", inplace=True)

tit\_df.isnull().sum()

PassengerId 0

Survived 0

Pclass 0

Name 0

Sex 0

Age 0

SibSp 0

Parch 0

Ticket 0

Fare 0

Cabin 327

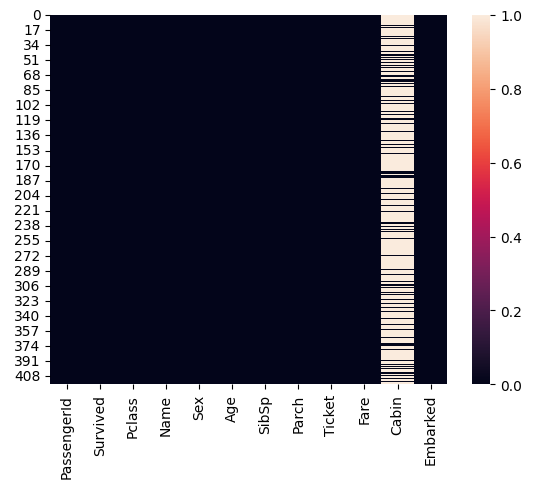
Embarked 0

dtype: int64

**# to check the null value in tit\_dif the white line in column (Cabin,Fare,Age)**

sns.heatmap(tit\_df.isnull())

plt.show()

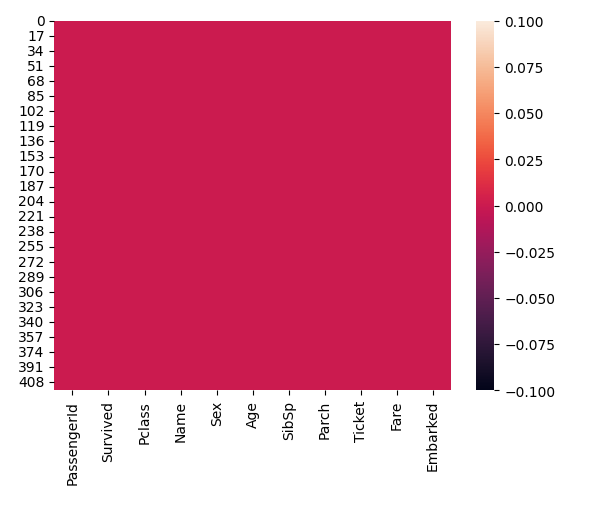
****

**# To delete cabin column permanently from given dataset**

tit\_df.drop('Cabin',axis=1, inplace=True)

sns.heatmap(tit\_df.isnull())

plt.show()

****

**#To check if there is any duplicate rows**

tit\_df.duplicated().sum()

0