Assignment No 1



#### In [2]:

*# Importing all the required python libraries*

**import** pandas **as** pd

**import** numpy **as** np

In [3]:

*# Loading dataset into pandas data frame*

df**=**pd.read\_csv("iris.csv") df

#### Out[3]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **sepal\_length** | | **sepal\_width** | **petal\_length** | **petal\_width** | **species** |
| **0** | NaN | 3.5 | 1.4 | 0.2 | setosa |
| **1** | 4.9 | NaN | 1.4 | 0.2 | setosa |
| **2** | 4.7 | 3.2 | NaN | 0.2 | setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| **4** | 5.0 | NaN | 1.4 | 0.2 | setosa |
| **...** | ... | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | virginica |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | virginica |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | virginica |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | virginica |

In [4]:

*# Data Preprocessing # Descibe Function* df.describe()

#### Out[4]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **sepal\_length** | **sepal\_width** | **petal\_length** | **petal\_width** |
| **count** | 146.000000 | 146.000000 | 146.000000 | 146.000000 |
| **mean** | 5.858219 | 3.045890 | 3.823288 | 1.224658 |
| **std** | 0.832508 | 0.432654 | 1.743878 | 0.756905 |
| **min** | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| **25%** | 5.100000 | 2.800000 | 1.600000 | 0.325000 |
| **50%** | 5.800000 | 3.000000 | 4.400000 | 1.300000 |
| **75%** | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| **max** | 7.900000 | 4.400000 | 6.900000 | 2.500000 |

In [5]:

*# Check dimension of data frame*

df.shape

#### Out[5]:

(150, 5)

#### In [6]:

df.head()

Out[6]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **sepal\_length** | **sepal\_width** | **petal\_length** | **petal\_width** | **species** |
| **0** NaN | 3.5 | 1.4 | 0.2 | setosa |
| **1** 4.9 | NaN | 1.4 | 0.2 | setosa |
| **2** 4.7 | 3.2 | NaN | 0.2 | setosa |
| **3** 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| **4** 5.0 | NaN | 1.4 | 0.2 | setosa |

#### In [7]:

df.tail()

Out[7]:

**sepal\_length sepal\_width petal\_length petal\_width species**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | virginica |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | virginica |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | virginica |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | virginica |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 | virginica |

#### In [8]:

df.head(2)

Out[8]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **sepal\_length** | **sepal\_width** | **petal\_length** | **petal\_width** | **species** |
| **0** NaN | 3.5 | 1.4 | 0.2 | setosa |
| **1** 4.9 | NaN | 1.4 | 0.2 | setosa |

#### In [9]:

df.tail(2)

Out[9]:

**sepal\_length sepal\_width petal\_length petal\_width species**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 virginica |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 virginica |

#### In [10]:

*# Types of variables*

df.dtypes

Out[10]:

#### sepal\_length float64 sepal\_width float64 petal\_length float64 petal\_width float64 species object dtype: object

In [11]:

*# Check for missing values in data frame*

df.isnull()

#### Out[11]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **sepal\_length** | | **sepal\_width** | **petal\_length** | **petal\_width** | **species** |
| **0** | True | False | False | False | False |
| **1** | False | True | False | False | False |
| **2** | False | False | True | False | False |
| **3** | False | False | False | False | False |
| **4** | False | True | False | False | False |
| **...** | ... | ... | ... | ... | ... |
| **145** | False | False | False | False | False |
| **146** | False | False | False | False | False |
| **147** | False | False | False | False | False |
| **148** | False | False | False | False | False |
| **149** | False | False | False | False | False |

150 rows × 5 columns

#### In [12]:

|  |  |
| --- | --- |
| df.isnull().sum  Out[12]: | () |
| sepal\_length | 4 |
| sepal\_width | 4 |
| petal\_length | 4 |
| petal\_width | 4 |
| species  dtype: int64 | 0 |
| In [13]: |  |

df.isnull().sum().sum()

Out[13]:

#### 16

In [14]:

*# Fill null value* df**=**df.fillna(value**=**0) df

#### Out[14]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **sepal\_length** | | **sepal\_width** | **petal\_length** | **petal\_width** | **species** |
| **0** | 0.0 | 3.5 | 1.4 | 0.2 | setosa |
| **1** | 4.9 | 0.0 | 1.4 | 0.2 | setosa |
| **2** | 4.7 | 3.2 | 0.0 | 0.2 | setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| **4** | 5.0 | 0.0 | 1.4 | 0.2 | setosa |
| **...** | ... | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | virginica |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | virginica |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | virginica |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | virginica |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 | virginica |

150 rows × 5 columns

#### In [15]:

df**=**pd.read\_csv("iris.csv") df**=**df.fillna(method**=**'pad') df

Out[15]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **sepal\_length** | | **sepal\_width** | **petal\_length** | **petal\_width** | **species** |
| **0** | NaN | 3.5 | 1.4 | 0.2 | setosa |
| **1** | 4.9 | 3.5 | 1.4 | 0.2 | setosa |
| **2** | 4.7 | 3.2 | 1.4 | 0.2 | setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| **4** | 5.0 | 3.1 | 1.4 | 0.2 | setosa |
| **...** | ... | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | virginica |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | virginica |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | virginica |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | virginica |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 | virginica |

#### 150 rows × 5 columns

In [16]:

df**=**pd.read\_csv("iris.csv") df**=**df.fillna(method**=**'ffill') df

#### Out[16]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **sepal\_length** | | **sepal\_width** | **petal\_length** | **petal\_width** | **species** |
| **0** | NaN | 3.5 | 1.4 | 0.2 | setosa |
| **1** | 4.9 | 3.5 | 1.4 | 0.2 | setosa |
| **2** | 4.7 | 3.2 | 1.4 | 0.2 | setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| **4** | 5.0 | 3.1 | 1.4 | 0.2 | setosa |
| **...** | ... | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | virginica |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | virginica |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | virginica |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | virginica |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 | virginica |

150 rows × 5 columns

#### In [17]:

df**=**pd.read\_csv("iris.csv") df**=**df.fillna(method**=**'bfill') df

Out[17]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **sepal\_length** | | **sepal\_width** | **petal\_length** | **petal\_width** | **species** |
| **0** | 4.9 | 3.5 | 1.4 | 0.2 | setosa |
| **1** | 4.9 | 3.2 | 1.4 | 0.2 | setosa |
| **2** | 4.7 | 3.2 | 1.5 | 0.2 | setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| **4** | 5.0 | 3.9 | 1.4 | 0.2 | setosa |
| **...** | ... | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | virginica |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | virginica |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | virginica |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | virginica |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 | virginica |

#### 150 rows × 5 columns

In [18]:

*# Fill null values with mean* df**=**pd.read\_csv("iris.csv") mean\_value**=**df['sepal\_length'].mean()

df['sepal\_length'].fillna(value**=**mean\_value,inplace**=True**) df

#### Out[18]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **sepal\_length** | | **sepal\_width** | **petal\_length** | **petal\_width** | **species** |
| **0** | 5.858219 | 3.5 | 1.4 | 0.2 | setosa |
| **1** | 4.900000 | NaN | 1.4 | 0.2 | setosa |
| **2** | 4.700000 | 3.2 | NaN | 0.2 | setosa |
| **3** | 4.600000 | 3.1 | 1.5 | 0.2 | setosa |
| **4** | 5.000000 | NaN | 1.4 | 0.2 | setosa |
| **...** | ... | ... | ... | ... | ... |
| **145** | 6.700000 | 3.0 | 5.2 | 2.3 | virginica |
| **146** | 6.300000 | 2.5 | 5.0 | 1.9 | virginica |
| **147** | 6.500000 | 3.0 | 5.2 | 2.0 | virginica |
| **148** | 6.200000 | 3.4 | 5.4 | 2.3 | virginica |
| **149** | 5.900000 | 3.0 | 5.1 | 1.8 | virginica |

150 rows × 5 columns

#### In [19]:

*# Fill null values with median* df**=**pd.read\_csv("iris.csv") median\_value**=**df['sepal\_length'].median()

df['sepal\_length'].fillna(value**=**median\_value,inplace**=True**) df

Out[19]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **sepal\_length** | | **sepal\_width** | **petal\_length** | **petal\_width** | **species** |
| **0** | 5.8 | 3.5 | 1.4 | 0.2 | setosa |
| **1** | 4.9 | NaN | 1.4 | 0.2 | setosa |
| **2** | 4.7 | 3.2 | NaN | 0.2 | setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| **4** | 5.0 | NaN | 1.4 | 0.2 | setosa |
| **...** | ... | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | virginica |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | virginica |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | virginica |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | virginica |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 | virginica |

#### 150 rows × 5 columns

In [20]:

*# Fill null values with mode* df**=**pd.read\_csv("iris.csv") mode\_value**=**df['sepal\_length'].mode()

df['sepal\_length'].fillna(value**=**mode\_value,inplace**=True**) df

#### Out[20]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **sepal\_length** | | **sepal\_width** | **petal\_length** | **petal\_width** | **species** |
| **0** | 5.0 | 3.5 | 1.4 | 0.2 | setosa |
| **1** | 4.9 | NaN | 1.4 | 0.2 | setosa |
| **2** | 4.7 | 3.2 | NaN | 0.2 | setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| **4** | 5.0 | NaN | 1.4 | 0.2 | setosa |
| **...** | ... | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | virginica |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | virginica |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | virginica |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | virginica |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 | virginica |

150 rows × 5 columns

#### In [21]:

*# Data formatting and data normalization*

df.dtypes

Out[21]:

#### sepal\_length float64 sepal\_width float64 petal\_length float64 petal\_width float64 species object dtype: object

In [22]:

|  |  |  |
| --- | --- | --- |
| *# Changing data type of column*  *# Before changing data type all the null value must be handled properly else it will cre*  mean\_value**=**df['sepal\_length'].mean() df['sepal\_length'].fillna(value**=**mean\_value,inplace**=True**) df['sepal\_length']**=**df['sepal\_length'].astype(int)  df.dtypes | | |
|  |  |  |

#### Out[22]:

sepal\_length int32

#### sepal\_width float64 petal\_length float64 petal\_width float64

species object dtype: object

#### In [23]:

mean\_value**=**df['sepal\_length'].mean() df['sepal\_length'].fillna(value**=**mean\_value,inplace**=True**)

df['sepal\_length']**=**df['sepal\_length'].round(0).astype(int) df.dtypes

Out[23]:

#### sepal\_length int32

sepal\_width float64 petal\_length float64 petal\_width float64

#### species object dtype: object

In [24]:

*# Converting categorical variables into quantitative variables* df['species'].replace({'setosa':1,'versicolor':2,'virginica':3},inplace**=True**) df

#### Out[24]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **sepal\_length** | | **sepal\_width** | **petal\_length** | **petal\_width** | **species** |
| **0** | 5 | 3.5 | 1.4 | 0.2 | 1 |
| **1** | 4 | NaN | 1.4 | 0.2 | 1 |
| **2** | 4 | 3.2 | NaN | 0.2 | 1 |
| **3** | 4 | 3.1 | 1.5 | 0.2 | 1 |
| **4** | 5 | NaN | 1.4 | 0.2 | 1 |
| **...** | ... | ... | ... | ... | ... |
| **145** | 6 | 3.0 | 5.2 | 2.3 | 3 |
| **146** | 6 | 2.5 | 5.0 | 1.9 | 3 |
| **147** | 6 | 3.0 | 5.2 | 2.0 | 3 |
| **148** | 6 | 3.4 | 5.4 | 2.3 | 3 |
| **149** | 5 | 3.0 | 5.1 | 1.8 | 3 |

150 rows × 5 columns

4/25/23, 7:02 PM DSBDAASS02-DATAWRANGLING2.ipynb - Colaboratory

ASSIGNMENT 2

import pandas as pd

import numpy as np

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

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| df | **Gender** | **Math Score** | **reading score** | **writing score** | **Placement Score** | **Placement Count** | **Region** |
|  | **0** Female | 75 | 65 | 55 | 70 | 2 | pune |
|  | **1** Male | 60 | 65 | 90 | 80 | 2 | pimpri |
|  | **2** Female | 78 | 78 | 90 | 81 | 1 | Mumbai |
|  | **3** Male | 67 | 90 | 95 | 82 | 2 | pune |
|  | **4** Female | 89 | 98 | 72 | 83 | 3 | pune |
|  | **5** Male | 90 | 69 | 82 | 84 | 3 | pune |
|  | **6** Male | 98 | 73 | 99 | 85 | 3 | pune |
|  | **7** Female | 79 | 98 | 99 | 86 | 1 | pune |
|  | **8** Male | 80 | 75 | 76 | 87 | 2 | pune |
|  | **9** Female | 81 | 87 | 80 | 88 | 3 | Banglore |

from sklearn import preprocessing

label\_encoder = preprocessing.LabelEncoder()

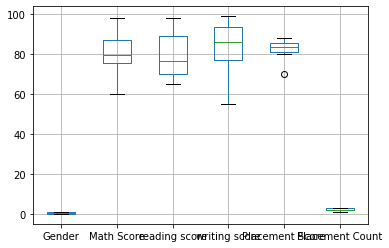
df['Gender']= label\_encoder.fit\_transform(df['Gender']) df['Gender'].unique()

array([0, 1])

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| df | **Gender** | **Math Score** | **reading score** | **writing score** | **Placement Score** | **Placement Count** | **Region** |
|  | **0** 0 | 75 | 65 | 55 | 70 | 2 | pune |
|  | **1** 1 | 60 | 65 | 90 | 80 | 2 | pimpri |
|  | **2** 0 | 78 | 78 | 90 | 81 | 1 | Mumbai |
|  | **3** 1 | 67 | 90 | 95 | 82 | 2 | pune |
|  | **4** 0 | 89 | 98 | 72 | 83 | 3 | pune |
|  | **5** 1 | 90 | 69 | 82 | 84 | 3 | pune |
|  | **6** 1 | 98 | 73 | 99 | 85 | 3 | pune |
|  | **7** 0 | 79 | 98 | 99 | 86 | 1 | pune |
|  | **8** 1 | 80 | 75 | 76 | 87 | 2 | pune |
|  | **9** 0 | 81 | 87 | 80 | 88 | 3 | Banglore |

import matplotlib.pyplot as plt boxplot = df.boxplot()

plt.show()



import scipy.stats as stats df['Math Score'].mean()

https://colab.research.google.com/drive/1np8OXHpe8VRqRmAfKJM6ahhfZivXqn3O?authuser=2#printMode=true 1/2

4/25/23, 7:02 PM DSBDAASS02-DATAWRANGLING2.ipynb - Colaboratory

mean = df['Math Score'].mean()

df['Math Score'].std()

std = df['Math Score'].std()

zscores = stats.zscore(df['Math Score']) print(zscores)

0 -0.447295

1 -1.874831

2 -0.161787

3 -1.208647

4 0.885072

5 0.980241

6 1.741594

7 -0.066618

8 0.028551

9 0.123720

Name: Math Score, dtype: float64

threshold = 0 outlier = []

for i in df['Math Score']: z = (i-mean)/std

if z > threshold:

outlier.append(i)

print('outlier in dataset is', outlier)

outlier in dataset is [89, 90, 98, 80, 81]

[Colab paid product](https://colab.research.google.com/signup?utm_source=footer&utm_medium=link&utm_campaign=footer_links)s - [Cancel contracts her](https://colab.research.google.com/cancel-subscription)e



https://colab.research.google.com/drive/1np8OXHpe8VRqRmAfKJM6ahhfZivXqn3O?authuser=2#printMode=true 2/2

ASSIGNMENT 3

import pandas as pd import numpy as np

import matplotlib.pyplot as plt

df = pd.read\_csv('/content/HR.csv') df.head()

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Age** | **Attrition** | **BusinessTravel** | **DailyRate** | **Department** | **DistanceFromHome** | **Educati** |
| **0** 41 | Yes | Travel\_Rarely | 1102 | Sales | 1 |  |

#### Research &

|  |  |  |  |
| --- | --- | --- | --- |
| **1** 49 | No | Travel\_Frequently | 279 |
| **2** 37 | Yes | Travel\_Rarely | 1373 |
| **3** 33 | No | Travel\_Frequently | 1392 |
| **4** 27 | No | Travel\_Rarely | 591 |

Development 8

[Research &](#_TOC_250001)

Development 2

[Research &](#_TOC_250000)

Development 3

Research &

Development 2

5 rows × 35 columns



df.columns

#### Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',

'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount', 'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate', 'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',

#### 'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked', 'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',

'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',

#### 'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',

'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion', 'YearsWithCurrManager'],

#### dtype='object')

#mean of monthly income

df.loc[:,"MonthlyIncome"].mean()

6502.931292517007

# mean of age

df.loc[: , "Age"].mean()

#### 36.923809523809524

#Meadian of monthly income

df.loc[: , "MonthlyIncome"].median()

4919.0

#Meadian of age

df.loc[: , "Age"].median()

#### 36.0

#Mode of monthly income

df.loc[: , "MonthlyIncome"].mode()

0 2342

#### Name: MonthlyIncome, dtype: int64

#mode of age

df.loc[: , "Age"].mode()

0 35

#### Name: Age, dtype: int64

#STD of monthly income

df.loc[: , "MonthlyIncome"].std()

4707.956783097994

#STD of age

df.loc[: , "Age"].std()

#### 9.135373489136732

arr1 = np.array(df["MonthlyIncome"]) arr2 = np.array(df['Age'])

print("Income " , arr1) print("Age" , arr2)

Income [5993 5130 2090 ... 6142 5390 4404]

#### Age [41 49 37 ... 27 49 34]

#MAX income

print(max(arr1))

#min income

print(min(arr1))

19999

#### 1009

#MAX age

print(max(arr2))

#min age

print(min(arr2))

60

#### 18

T

df["BusinessTravel"].replace({"Travel\_Rarely" : 1 , "Travel\_Frequently" : 0} , inplace = df["Attrition"].replace({"Yes" : 1 , "No" : 0} , inplace = True)

df.head()

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Age** | **Attrition** | **BusinessTravel** | **DailyRate** | **Department** | **DistanceFromHome** | **Educatio** |
| **0** 41 | 1 | 1 | 1102 | Sales | 1 |  |

#### 5 rows × 35 columns



#### Research &

#### Development 8

|  |  |  |  |
| --- | --- | --- | --- |
| **1** 49 | 0 | 0 | 279 |
| **2** 37 | 1 | 1 | 1373 |
| **3** 33 | 0 | 0 | 1392 |
| **4** 27 | 0 | 1 | 591 |

#### Research &

#### Development 2

#### Research &

#### Development 3

#### Research &

#### Development 2

#### [Colab paid products](https://colab.research.google.com/signup?utm_source=footer&utm_medium=link&utm_campaign=footer_links) - [Cancel contracts here](https://colab.research.google.com/cancel-subscription)















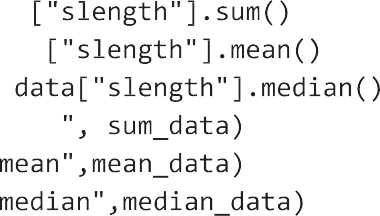
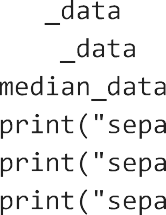








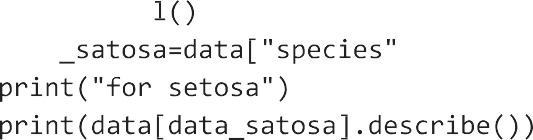






















































































## Assignment no - 4

import pandas as pd import numpy as np

import matplotlib.pyplot as plt import seaborn as sns

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

##### Unnamed:

**0**

##### crim zn indus chas nox rm age dis rad tax ptratio black lstat me

**0** 1 0.00632 18.0 2.31 0 0.538 6.575 65.2 4.0900 1 296 15.3 396.90 4.98 24

**1** 2 0.02731 0.0 7.07 0 0.469 6.421 78.9 4.9671 2 242 17.8 396.90 9.14 21

**2** 3 0.02729 0.0 7.07 0 0.469 7.185 61.1 4.9671 2 242 17.8 392.83 4.03 34

**3** 4 0.03237 0.0 2.18 0 0.458 6.998 45.8 6.0622 3 222 18.7 394.63 2.94 33

**4** 5 0.06905 0.0 2.18 0 0.458 7.147 54.2 6.0622 3 222 18.7 396.90 5.33 36

**...** ... ... ... ... ... ... ... ... ... ... ... ... ... ...

**501** 502 0.06263 0.0 11.93 0 0.573 6.593 69.1 2.4786 1 273 21.0 391.99 9.67 22

**502** 503 0.04527 0.0 11.93 0 0.573 6.120 76.7 2.2875 1 273 21.0 396.90 9.08 20

**503** 504 0.06076 0.0 11.93 0 0.573 6.976 91.0 2.1675 1 273 21.0 396.90 5.64 23

**504** 505 0.10959 0.0 11.93 0 0.573 6.794 89.3 2.3889 1 273 21.0 393.45 6.48 22

**505** 506 0.04741 0.0 11.93 0 0.573 6.030 80.8 2.5050 1 273 21.0 396.90 7.88 11

506 rows × 15 columns

df.head()

##### Unnamed:

**0**

##### crim zn indus chas nox rm age dis rad tax ptratio black lstat medv

**0** 1 0.00632 18.0 2.31 0 0.538 6.575 65.2 4.0900 1 296 15.3 396.90 4.98 24.0

**1** 2 0.02731 0.0 7.07 0 0.469 6.421 78.9 4.9671 2 242 17.8 396.90 9.14 21.6

**2** 3 0.02729 0.0 7.07 0 0.469 7.185 61.1 4.9671 2 242 17.8 392.83 4.03 34.7

**3** 4 0.03237 0.0 2.18 0 0.458 6.998 45.8 6.0622 3 222 18.7 394.63 2.94 33.4

**4** 5 0.06905 0.0 2.18 0 0.458 7.147 54.2 6.0622 3 222 18.7 396.90 5.33 36.2

df.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 506 entries, 0 to 505

Data columns (total 15 columns):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # |  | Column | Non-Null Count |  | Dtype |
| 0 |  | Unnamed: 0 | 506 non-null |  | int64 |
| 1 |  | crim | 506 non-null |  | float64 |
| 2 |  | zn | 506 non-null |  | float64 |
| 3 |  | indus | 506 non-null |  | float64 |
| 4 |  | chas | 506 non-null |  | int64 |
| 5 |  | nox | 506 non-null |  | float64 |
| 6 |  | rm | 506 non-null |  | float64 |
| 7 |  | age | 506 non-null |  | float64 |
| 8 |  | dis | 506 non-null |  | float64 |
| 9 |  | rad | 506 non-null |  | int64 |
| 10 |  | tax | 506 non-null |  | int64 |
| 11 |  | ptratio | 506 non-null |  | float64 |
| 12 |  | black | 506 non-null |  | float64 |
| 13 |  | lstat | 506 non-null |  | float64 |
| 14 |  | medv | 506 non-null |  | float64 |

dtypes: float64(11), int64(4) memory usage: 59.4 KB

#null values must be removed before using regression df.isna().sum()

Unnamed: 0 0

crim 0

zn 0

indus 0

chas 0

nox 0

rm 0

age 0

dis 0

rad 0

tax 0

ptratio 0

black 0

lstat 0

medv 0

dtype: int64

target\_features = "medv"

#seperate object from target feature y = df[target\_features]

#sepperate object for input features x = df.drop(target\_features, axis=1)

x.head()

y.head()

|  |  |
| --- | --- |
| 0 | 24.0 |
| 1 | 21.6 |
| 2 | 34.7 |
| 3 | 33.4 |
| 4 | 36.2 |

Name: medv, dtype: float64

#Spliting data for training and testing

#Here, 20% data used for testing and 80% data used for training from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size = 0.2,random\_state=2)

from sklearn.linear\_model import LinearRegression regression = LinearRegression()

regression.fit(x\_train,y\_train)



▾ LinearRegression

LinearRegression()

train\_score=round(regression.score(x\_train,y\_train)\*100,2) print('Train score of linear regression',train\_score)

y\_pred = regression.predict(x\_test)

Train score of linear regression 72.91

from sklearn.metrics import r2\_score

score=round(r2\_score(y\_test,y\_pred)\*100,2) print('r\_2 score',score)

r\_2 score 78.1

round(regression.score(x\_test,y\_test)\*100,2)

78.1

from sklearn import metrics

print("Mean absolute error on test data of linear regression",metrics.mean\_absolute\_error(y\_test,y\_pred)) print("Mean squared error on test data of linear regression",metrics.mean\_squared\_error(y\_test,y\_pred))

print("Root mean squared error on test data of linear regression",np.sqrt(metrics.mean\_squared\_error(y\_test,y\_pred)))

Mean absolute error on test data of linear regression 3.0812603233002447 Mean squared error on test data of linear regression 18.321720821929564

Root mean squared error on test data of linear regression 4.280387928906627

df1=pd.DataFrame({'Actual':y\_test,'Predicted':y\_pred,'Variance':y\_test-y\_pred}) df1.head()

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Actual** | **Predicted** | **Variance** |
| **463** | 20.2 | 22.935008 | -2.735008 |
| **152** | 15.3 | 21.334270 | -6.034270 |
| **291** | 37.3 | 33.643417 | 3.656583 |
| **183** | 32.5 | 31.381211 | 1.118789 |
| **384** | 8.8 | 3.218861 | 5.581139 |

 [Colab paid product](https://colab.research.google.com/signup?utm_source=footer&utm_medium=link&utm_campaign=footer_links)s - [Cancel contracts her](https://colab.research.google.com/cancel-subscription)e

## Assignment no - 5

import pandas as pd import numpy as np

import matplotlib.pyplot as plt

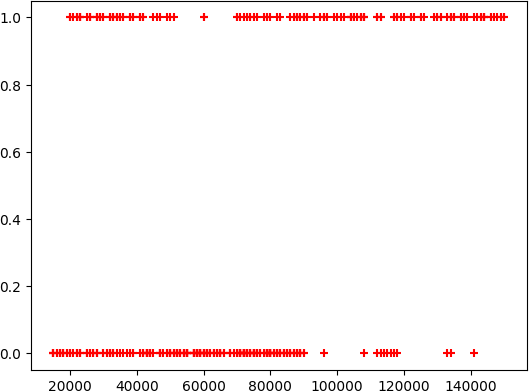
df = pd.read\_csv("Social\_Network\_Ads.csv") df

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **User ID** | **Gender** | **Age** | **EstimatedSalary** | **Purchased** |
| **0** | 15624510 | Male | 19 | 19000 | 0 |
| **1** | 15810944 | Male | 35 | 20000 | 0 |
| **2** | 15668575 | Female | 26 | 43000 | 0 |
| **3** | 15603246 | Female | 27 | 57000 | 0 |
| **4** | 15804002 | Male | 19 | 76000 | 0 |
| **...** | ... | ... | ... | ... | ... |
| **395** | 15691863 | Female | 46 | 41000 | 1 |
| **396** | 15706071 | Male | 51 | 23000 | 1 |
| **397** | 15654296 | Female | 50 | 20000 | 1 |
| **398** | 15755018 | Male | 36 | 33000 | 0 |
| **399** | 15594041 | Female | 49 | 36000 | 1 |

400 rows × 5 columns

plt.scatter(df.EstimatedSalary,df.Purchased,marker= '+',color='red')

<matplotlib.collections.PathCollection at 0x7efbf4164670>



# input

x = df[['Age','EstimatedSalary']] y = df['Purchased']

from sklearn.model\_selection import train\_test\_split

xtrain, xtest, ytrain, ytest = train\_test\_split( x, y, test\_size = 0.1,random\_state = 0)

from sklearn.preprocessing import StandardScaler sc\_x = StandardScaler()

xtrain = sc\_x.fit\_transform(xtrain) xtest = sc\_x.transform(xtest)

print (xtrain[0:10, :])

[[-1.05714987 0.53420426]

[ 0.2798728 -0.51764734]

[-1.05714987 0.41733186]

[-0.29313691 -1.45262654]

[ 0.47087604 1.23543867]

[-1.05714987 -0.34233874]

[-0.10213368 0.30045946]

[ 1.33039061 0.59264046]

[-1.15265148 -1.16044554]

[ 1.04388575 0.47576806]]

# check the lenth of taining len(xtrain)

len(xtest)

40

# check the traing values of eacg element xtrain

xtest

array([[-0.77064501, 0.50498616],

[-0.00663206, -0.57608354],

[-0.29313691, 0.15436896],

[-0.77064501, 0.27124136],

[-0.29313691, -0.57608354],

[-1.05714987, -1.45262654],

[-0.67514339, -1.59871705],

[-0.19763529, 2.17041787],

[-1.91666444, -0.05015774],

[ 0.85288251, -0.78061024],

[-0.77064501, -0.60530164],

[-0.96164825, -0.42999304],

[-0.10213368, -0.42999304],

|  |  |
| --- | --- |
| [ 0.08886956, | 0.21280516], |
| [-1.7256612 , | 0.47576806], |
| [-0.57964177, | 1.38152917], |
| [-0.10213368, | 0.21280516], |
| [-1.82116282, | 0.44654996], |
| [ 1.61689547, | 1.76136447], |

[-0.29313691, -1.39419034],

[-0.29313691, -0.66373784],

[ 0.85288251, 2.17041787],

[ 0.2798728 , -0.54686544],

[ 0.85288251, 1.03091197],

[-1.43915634, -1.21888174],

|  |  |
| --- | --- |
| [ 1.04388575, | 2.08276357], |
| [-0.96164825, | 0.50498616], |
| [-0.86614663, | 0.30045946], |
| [-0.10213368, -0.22546634], | |
| [-0.57964177, | 0.47576806], |
| [-1.63015958, | 0.53420426], |
| [-0.10213368, | 0.27124136], |
| [ 1.8078987 , -0.28390254], | |

[-0.10213368, -0.48842924],

[-1.34365472, -0.34233874],

[-1.91666444, -0.51764734],

[-1.53465796, 0.32967756],

[-0.38863853, -0.78061024],

[-0.67514339, -1.04357314],

[ 1.04388575, -0.98513694]])

# create a model of logistic regression

from sklearn.linear\_model import LogisticRegression model = LogisticRegression()

model.fit(xtrain,ytrain)



▾ LogisticRegression

LogisticRegression()

model.predict(xtest)

array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,

0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1])

model.score(xtest,ytest)

0.95

## Assignment No - 6



import pandas as pd import numpy as np

import matplotlib.pyplot as plt import seaborn as sns

df=pd.read\_csv("/content/Iris.csv") df

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Id** | **SepalLengthCm** | **SepalWidthCm** | **PetalLengthCm** | **PetalWidthCm** | **Species** |
| **0** | 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| **...** | ... | ... | ... | ... | ... | ... |
| **145** | 146 | 6.7 | 3.0 | 5.2 | 2.3 | Iris-virginica |
| **146** | 147 | 6.3 | 2.5 | 5.0 | 1.9 | Iris-virginica |
| **147** | 148 | 6.5 | 3.0 | 5.2 | 2.0 | Iris-virginica |
| **148** | 149 | 6.2 | 3.4 | 5.4 | 2.3 | Iris-virginica |
| **149** | 150 | 5.9 | 3.0 | 5.1 | 1.8 | Iris-virginica |

150 rows × 6 columns

df.isnull().sum()

Id 0

SepalLengthCm 0

SepalWidthCm 0

PetalLengthCm 0

PetalWidthCm 0

Species 0

dtype: int64



#Removing null values

columns=['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm'] for col in columns:

df[col]=df[col].fillna(df[col].mean())

df

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Id** | **SepalLengthCm** | **SepalWidthCm** | **PetalLengthCm** | **PetalWidthCm** | **Species** |
| **0** | 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| **...** | ... | ... | ... | ... | ... | ... |
| **145** | 146 | 6.7 | 3.0 | 5.2 | 2.3 | Iris-virginica |
| **146** | 147 | 6.3 | 2.5 | 5.0 | 1.9 | Iris-virginica |
| **147** | 148 | 6.5 | 3.0 | 5.2 | 2.0 | Iris-virginica |
| **148** | 149 | 6.2 | 3.4 | 5.4 | 2.3 | Iris-virginica |
| **149** | 150 | 5.9 | 3.0 | 5.1 | 1.8 | Iris-virginica |

150 rows × 6 columns

df.isnull().sum()

Id 0

SepalLengthCm 0

SepalWidthCm 0

PetalLengthCm 0

PetalWidthCm 0

Species 0

dtype: int64



#Converting categorical values ot numeric values from sklearn import preprocessing

label\_encoder = preprocessing.LabelEncoder()

df['Species']= label\_encoder.fit\_transform(df['Species']) df['Species'].unique()

df

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Id** | **SepalLengthCm** | **SepalWidthCm** | **PetalLengthCm** | **PetalWidthCm** | **Species** |
| **0** | 1 | 5.1 | 3.5 | 1.4 | 0.2 | 0 |
| **1** | 2 | 4.9 | 3.0 | 1.4 | 0.2 | 0 |
| **2** | 3 | 4.7 | 3.2 | 1.3 | 0.2 | 0 |
| **3** | 4 | 4.6 | 3.1 | 1.5 | 0.2 | 0 |
| **4** | 5 | 5.0 | 3.6 | 1.4 | 0.2 | 0 |
| **...** | ... | ... | ... | ... | ... | ... |
| **145** | 146 | 6.7 | 3.0 | 5.2 | 2.3 | 2 |
| **146** | 147 | 6.3 | 2.5 | 5.0 | 1.9 | 2 |
| **147** | 148 | 6.5 | 3.0 | 5.2 | 2.0 | 2 |
| **148** | 149 | 6.2 | 3.4 | 5.4 | 2.3 | 2 |
| **149** | 150 | 5.9 | 3.0 | 5.1 | 1.8 | 2 |

150 rows × 6 columns

y=df['Species']

x=df.drop('Species',axis=1)

#Spliting data for training and testing

#Here, 20% data used for testing and 80% data used for training from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size = 0.2,random\_state=2)

# import the class

from sklearn.naive\_bayes import GaussianNB gaussian = GaussianNB()

gaussian.fit(x\_train, y\_train)



▾ GaussianNB

GaussianNB()

y\_pred = gaussian.predict(x\_test) y\_pred

array([0, 0, 2, 0, 0, 2, 0, 2, 2, 0, 0, 0, 0, 0, 1, 1, 0, 1, 2, 1, 1, 1,

2, 1, 1, 0, 0, 2, 0, 2])

from sklearn.metrics import accuracy\_score,precision\_score,recall\_score accuracy = accuracy\_score(y\_test,y\_pred)

precision = precision\_score(y\_test, y\_pred,average='micro') recall = recall\_score(y\_test, y\_pred,average='micro')

print('recall' , recall)

print('precision' , precision) print('accuracy' , accuracy)

recall 1.0

precision 1.0

accuracy 1.0

from sklearn.metrics import precision\_score,confusion\_matrix,accuracy\_score,recall\_score cm = confusion\_matrix(y\_test, y\_pred)

cm

|  |  |  |
| --- | --- | --- |
| array([[14, | 0, | 0], |
| [ 0, | 8, | 0], |
| [ 0, | 0, | 8]]) |

df = pd.DataFrame({'Real Values' : y\_test , 'Predicated Values' : y\_pred}) df.head()



|  |  |  |
| --- | --- | --- |
|  | **Real Values** | **Predicated Values** |
| **6** | 0 | 0 |
| **3** | 0 | 0 |
| **113** | 2 | 2 |
| **12** | 0 | 0 |
| **24** | 0 | 0 |

[Colab paid product](https://colab.research.google.com/signup?utm_source=footer&utm_medium=link&utm_campaign=footer_links)s - [Cancel contracts her](https://colab.research.google.com/cancel-subscription)e

0s completed at 8:53 PM

## Assignment no - 7

1 import nltk

1. nltk.download('stopwords')
2. nltk.download('words')
3. nltk.download('wordnet')
4. nltk.download('averged\_perception\_tagger')
5. nltk.download('punkt')

[nltk\_data] Downloading package stopwords to /root/nltk\_data... [nltk\_data] Unzipping corpora/stopwords.zip.

[nltk\_data] Downloading package words to /root/nltk\_data... [nltk\_data] Unzipping corpora/words.zip.

[nltk\_data] Downloading package wordnet to /root/nltk\_data...

[nltk\_data] Error loading averged\_perception\_tagger: Package

[nltk\_data] 'averged\_perception\_tagger' not found in index [nltk\_data] Downloading package punkt to /root/nltk\_data...

[nltk\_data] Unzipping tokenizers/punkt.zip. True

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. import pandas as pd 2. import numpy as np | | | | | | | | | | | | |
| 1 | sent= | "They | told | that | thier | eges | are | 20 | 23 | and | 27 | respectively" |
| 1 | add=[] | | | | | | | | | | | |
| 1. for word in sent.split(): 2. if word.isdigit(): 3. add.append(int(word)) | | | | | | | | | | | | |
| 1 print ("Ave", sum(add)/len(add)) | | | | | | | | | | | | |

Ave 23.333333333333332

1 from nltk.tokenize import word\_tokenize, sent\_tokenize

1 sent= "Hello all! how are you? Welcome to pun "

1 sent\_tokenize(sent)

['Hello\xa0all!\xa0how\xa0are\xa0you?', 'Welcome\xa0to\xa0pun']

1 word\_tokenize(sent)

['Hello', 'all', '!', 'how', 'are', 'you', '?', 'Welcome', 'to', 'pun']

1 from nltk.tokenize import SpaceTokenizer 2 tk=SpaceTokenizer()

3 tk.tokenize(sent)

['Hello\xa0all!\xa0how\xa0are\xa0you?\xa0Welcome\xa0to\xa0pun', '']

1 sent='Hello all!\tHow are u?\tto pune'

1 print(sent)

Hello all! How are u? to pune

1. s1='ctas','catlike','catty','cat'
2. s2='stemmer','stemming','stemmed','stem'

|  |
| --- |
| 3 s3='fishing','fished','fisher','fish' |
| 1 from nltk.stem import PorterStemmer |
| 1 ps=PorterStemmer() |
| 1 ps.stem(s3[0]) |

'fish'

1. for word in s4:
2. ps=PorterStemmer()
3. print(ps.stem(word))

argu argu argu argu

|  |
| --- |
| 1 # lemmatization |
| 1 word='playing' |
| 1 from nltk.stem import WordNetLemmatizer |
| 1. wnl=WordNetLemmatizer() 2. print(wnl.lemmatize(word,'n')) # noun 3 print(wnl.lemmatize(word,'v')) # verb   4 print(wnl.lemmatize(word,'a')) # adjective 5 print(wnl.lemmatize(word,'r')) # adverb |

playing play

playing

playing

1 word='went'

1. wnl=WordNetLemmatizer()
2. print(wnl.lemmatize(word,'n')) # noun 3 print(wnl.lemmatize(word,'v')) # verb

4 print(wnl.lemmatize(word,'a')) # adjective 5 print(wnl.lemmatize(word,'r')) # adverb

went go

went went

1 # POS tagging

1 from nltk import pos\_tag

1. import nltk
2. nltk.download('averaged\_perceptron\_tagger')

[nltk\_data] Downloading package averaged\_perceptron\_tagger to [nltk\_data] /root/nltk\_data...

[nltk\_data] Unzipping taggers/averaged\_perceptron\_tagger.zip.

True

1 sents='Rajgad (literal meaning Ruling Fort) is a hill fort situated in the Pune district of Maha

1 print(sents)

Rajgad (literal meaning Ruling Fort) is a hill fort situated in the Pune district of Maharashtra, India. Formerly known as Murumdev

1 nltk.download('omw-1.4')

1 words=word\_tokenize(sents)

[nltk\_data] Downloading package omw-1.4 to /root/nltk\_data... True

1 pos\_tag(words)

[('Rajgad', 'NNP'),

('(', '('),



('literal', 'JJ'),

('meaning', 'NN'),

('Ruling', 'NNP'),

('Fort', 'NNP'),

(')', ')'),

('is', 'VBZ'),

('a', 'DT'),

('hill', 'NN'),

('fort', 'NN'),

('situated', 'VBN'),

('in', 'IN'),

('the', 'DT'),

('Pune', 'NNP'),

('district', 'NN'),

('of', 'IN'),

('Maharashtra', 'NNP'),

(',', ','),

('India', 'NNP'),

('.', '.'),

('Formerly', 'RB'),

('known', 'VBN'),

('as', 'IN'),

('Murumdev', 'NNP'),

(',', ','),

('the', 'DT'),

('fort', 'NN'),

('was', 'VBD'),

('the', 'DT'),

('capital', 'NN'),

('of', 'IN'),

('the', 'DT'),

('Maratha', 'NNP'),

('Empire', 'NNP'),

('under', 'IN'),

('the', 'DT'),

('rule', 'NN'),

('of', 'IN'),

('Shivaji', 'NNP'),

('for', 'IN'),

('almost', 'RB'),

('26', 'CD'),

('years', 'NNS'),

(',', ','),

('afterwhich', 'IN'),

('the', 'DT'),

('capital', 'NN'),

('was', 'VBD'),

('moved', 'VBN'),

('to', 'TO'),

('the', 'DT'),

('Raigad', 'NNP'),

('Fort', 'NNP'),

('.', '.'),

('[', 'CC'),

('1', 'CD'),

(']', 'NN'),

1 tags=pos\_tag(words)

1. for word in tags:
2. if word[1].startswith('V'):
3. print(word[0])

is

situated known

was was

moved

discovered

called were

used build

fortify needed

1. # spell correction
2. from textblob import TextBlob
3. t=TextBlob('computoor')
4. print(t.correct())

computer

1. t=TextBlob('nead')
2. print(t.correct())

head

1



## Assignment no - 8

import pandas as pd import numpy as np

import matplotlib.pyplot as plt import seaborn as sns

df1 = sns.load\_dataset('titanic') df1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **survived** | **pclass** | **sex** | **age** | **sibsp** | **parch** | **fare** | **embarked** | **class** | **who** | **adult\_male** | **deck** | **e** |
| **0** | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | S |
| **1** | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C |  |
| **2** | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | S |
| **3** | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | S |
| **4** | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN | S |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |  |
| **886** | 0 | 2 | male | 27.0 | 0 | 0 | 13.0000 | S | Second | man | True | NaN | S |
| **887** | 1 | 1 | female | 19.0 | 0 | 0 | 30.0000 | S | First | woman | False | B | S |
| **888** | 0 | 3 | female | NaN | 1 | 2 | 23.4500 | S | Third | woman | False | NaN | S |
| **889** | 1 | 1 | male | 26.0 | 0 | 0 | 30.0000 | C | First | man | True | C |  |
| **890** | 0 | 3 | male | 32.0 | 0 | 0 | 7.7500 | Q | Third | man | True | NaN |  |

891 rows × 15 columns

df = pd.DataFrame(df1) df

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **survived** | **pclass** | **sex** | **age** | **sibsp** | **parch** | **fare** | **embarked** | **class** | **who** | **adult\_male** | **deck** | **e** |
| **0** | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | S |
| **1** | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C |  |
| **2** | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | S |
| **3** | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | S |
| **4** | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN | S |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |  |
| **886** | 0 | 2 | male | 27.0 | 0 | 0 | 13.0000 | S | Second | man | True | NaN | S |
| **887** | 1 | 1 | female | 19.0 | 0 | 0 | 30.0000 | S | First | woman | False | B | S |
| **888** | 0 | 3 | female | NaN | 1 | 2 | 23.4500 | S | Third | woman | False | NaN | S |
| **889** | 1 | 1 | male | 26.0 | 0 | 0 | 30.0000 | C | First | man | True | C |  |
| **890** | 0 | 3 | male | 32.0 | 0 | 0 | 7.7500 | Q | Third | man | True | NaN |  |

891 rows × 15 columns

df.describe()

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **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 |

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

df.columns

Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',

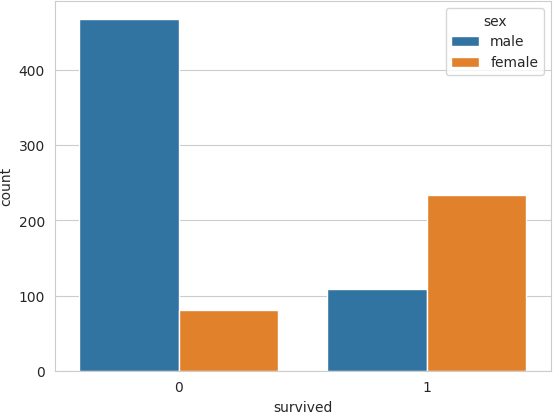
'embarked', 'class', 'who', 'adult\_male', 'deck', 'embark\_town', 'alive', 'alone'],

dtype='object')

sns.set\_style('whitegrid')

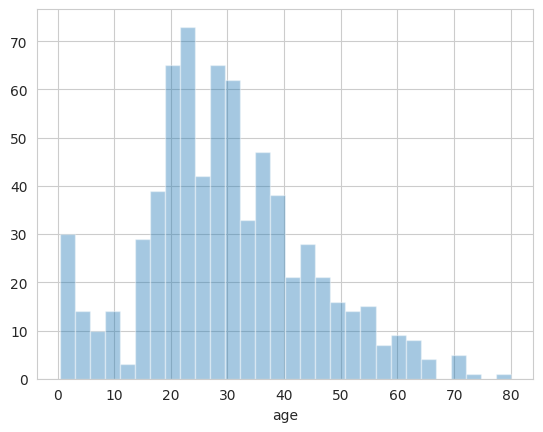
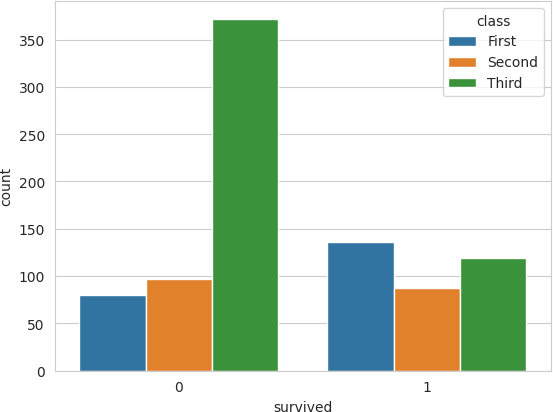
sns.countplot(x='survived', data=df, hue='sex')

<Axes: xlabel='survived', ylabel='count'>



sns.countplot(x='survived', data=df, hue='class')

<Axes: xlabel='survived', ylabel='count'>



<ipython-input-26-49f290fce869>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

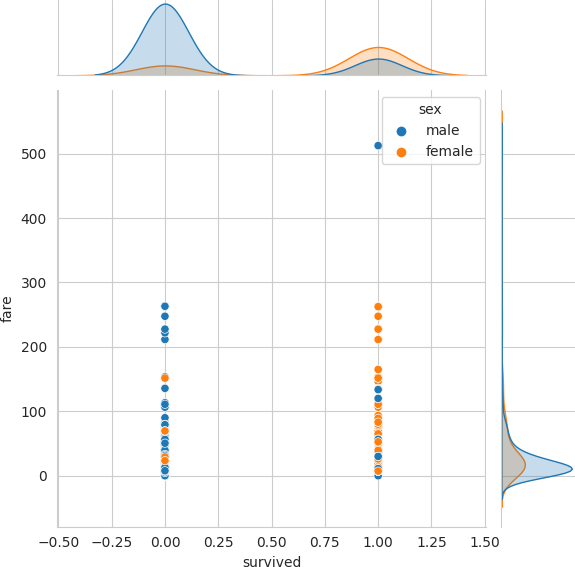
sns.distplot(df['age'].dropna() , kde = False , bins=30)

<Axes: xlabel='age'>

sns.distplot(df['age'].dropna() , kde = False , bins=30)

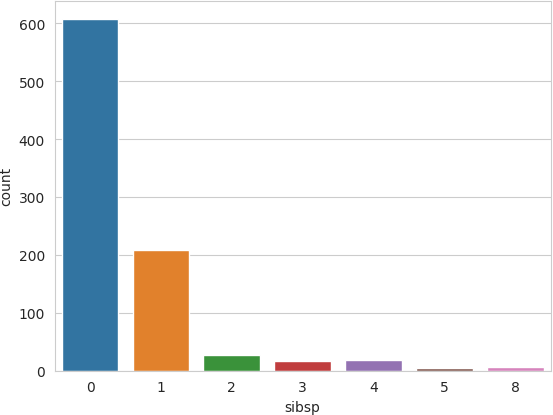
sns.jointplot(x = 'survived' , y = 'fare' , data = df , hue = 'sex')

<seaborn.axisgrid.JointGrid at 0x7f5720172460>

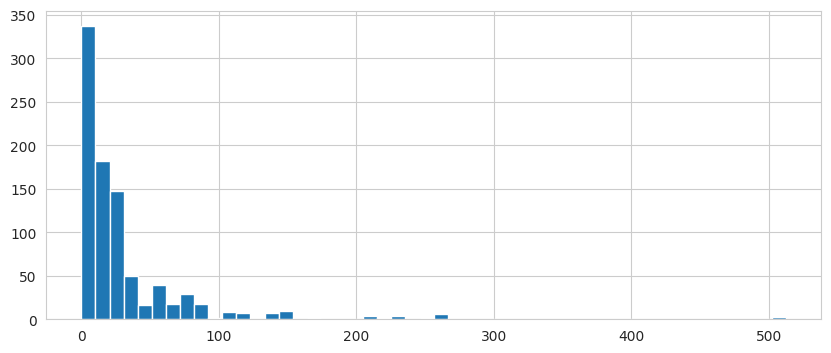


sns.countplot(x='sibsp' , data= df)

<Axes: xlabel='sibsp', ylabel='count'>



df['fare'].hist(bins = 50 , figsize = (10 , 4))

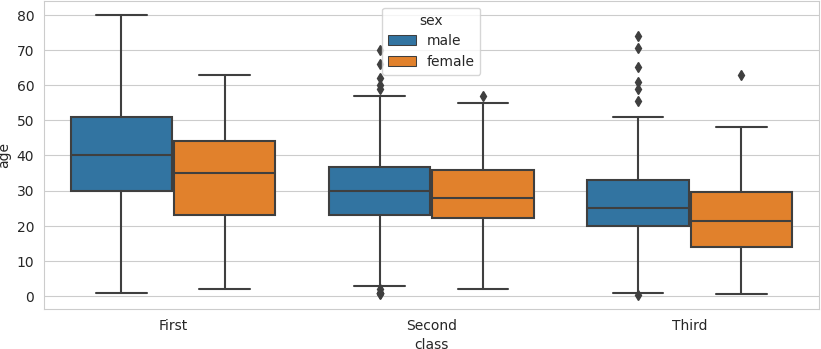


<Axes: >

plt.figure(figsize= (10 , 4))

sns.boxplot(x = 'class' , y = 'age' , data = df , hue = 'sex')

<Axes: xlabel='class', ylabel='age'>

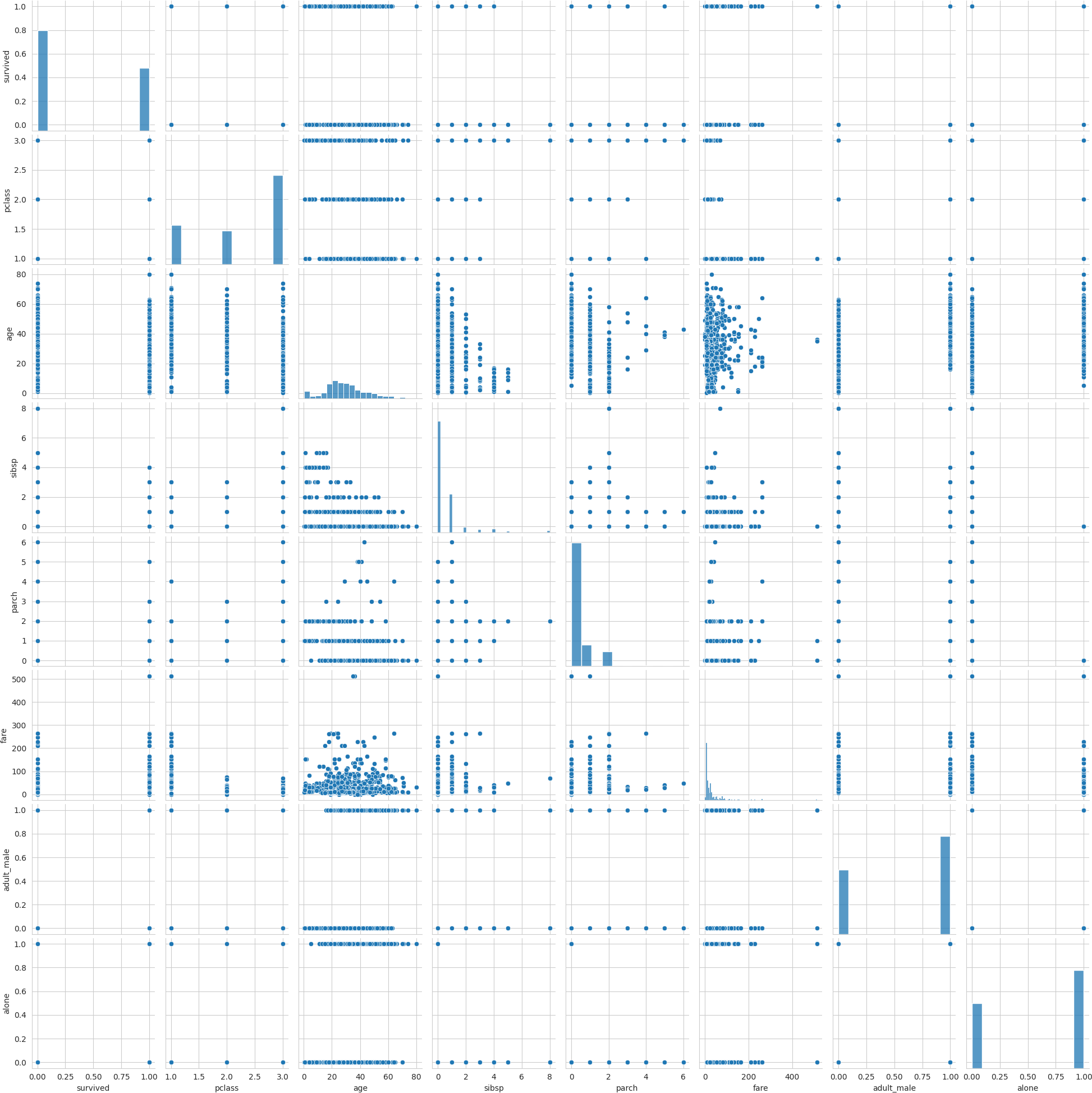


sns.pairplot(df)

< array\_function

< array\_function

internals>:180: RuntimeWarning: Converting input from bool to <class 'numpy.uint8' internals>:180: RuntimeWarning: Converting input from bool to <class 'numpy.uint8'

<seaborn.axisgrid.PairGrid at 0x7f571d9af430>

df['fare'].max()

512.3292

df['fare'].min()

0.0

Assignment No 9



import numpy as np import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

df1 = sns.load\_dataset('titanic') df1

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **survived** | **pclass** | **sex** | **age** | **sibsp** | **parch** | **fare** | **embarked** | **class** | **who** | **a** |
| **0** | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man |  |
| **1** | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman |  |
| **2** | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman |  |
| **3** | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman |  |
| **4** | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man |  |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |  |
| **886** | 0 | 2 | male | 27.0 | 0 | 0 | 13.0000 | S | Second | man |  |
| **887** | 1 | 1 | female | 19.0 | 0 | 0 | 30.0000 | S | First | woman |  |
| **888** | 0 | 3 | female | NaN | 1 | 2 | 23.4500 | S | Third | woman |  |
| **889** | 1 | 1 | male | 26.0 | 0 | 0 | 30.0000 | C | First | man |  |
| **890** | 0 | 3 | male | 32.0 | 0 | 0 | 7.7500 | Q | Third | man |  |

891 rows × 15 columns

df = pd.DataFrame(df1)

df.head()

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **survived** | **pclass** | **sex** | **age** | **sibsp** | **parch** | **fare** | **embarked** | **class** | **who** | **adul** |  |
| **0** 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man |  |  |
| **1** 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman |  |  |
| **2** 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman |  |  |
| **3** 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman |  |  |
| **4** 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| df.describe() | | | | | | | | | | | |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **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 |

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

df.columns

Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',

'embarked', 'class', 'who', 'adult\_male', 'deck', 'embark\_town', 'alive', 'alone'],

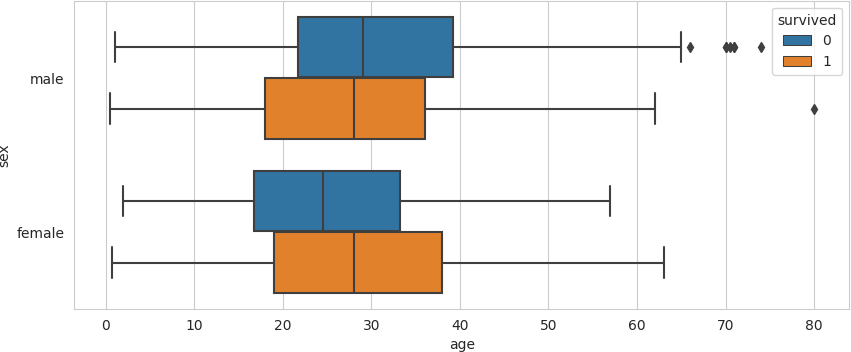
dtype='object')

sns.set\_style('whitegrid')

plt.figure(figsize = (10 , 4))

sns.boxplot(x = 'age' , y = 'sex' , data= df , hue = 'survived')

<Axes: xlabel='age', ylabel='sex'>



 [Colab paid product](https://colab.research.google.com/signup?utm_source=footer&utm_medium=link&utm_campaign=footer_links)s - [Cancel contracts her](https://colab.research.google.com/cancel-subscription)e

## Assignment no - 10



import numpy as np import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

df1 = pd.read\_csv('/content/Iris.csv') df1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Id** | **SepalLengthCm** | **SepalWidthCm** | **PetalLengthCm** | **PetalWidthCm** | **Species** |
| **0** | 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| **...** | ... | ... | ... | ... | ... | ... |
| **145** | 146 | 6.7 | 3.0 | 5.2 | 2.3 | Iris-virginica |
| **146** | 147 | 6.3 | 2.5 | 5.0 | 1.9 | Iris-virginica |
| **147** | 148 | 6.5 | 3.0 | 5.2 | 2.0 | Iris-virginica |
| **148** | 149 | 6.2 | 3.4 | 5.4 | 2.3 | Iris-virginica |
| **149** | 150 | 5.9 | 3.0 | 5.1 | 1.8 | Iris-virginica |

150 rows × 6 columns



df = pd.DataFrame(df1) df.head()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Id** | **SepalLengthCm** | **SepalWidthCm** | **PetalLengthCm** | **PetalWidthCm** | **Species** |
| **0** 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |

df.describe()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Id** | **SepalLengthCm** | **SepalWidthCm** | **PetalLengthCm** | **PetalWidthCm** |
| **count** | 150.000000 | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| **mean** | 75.500000 | 5.843333 | 3.054000 | 3.758667 | 1.198667 |
| **std** | 43.445368 | 0.828066 | 0.433594 | 1.764420 | 0.763161 |
| **min** | 1.000000 | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| **25%** | 38.250000 | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| **50%** | 75.500000 | 5.800000 | 3.000000 | 4.350000 | 1.300000 |
| **75%** | 112.750000 | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| **max** | 150.000000 | 7.900000 | 4.400000 | 6.900000 | 2.500000 |

df.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 150 entries, 0 to 149

Data columns (total 6 columns):

# Column Non-Null Count Dtype

1. Id 150 non-null int64
2. SepalLengthCm 150 non-null float64
3. SepalWidthCm 150 non-null float64
4. PetalLengthCm 150 non-null float64
5. PetalWidthCm 150 non-null float64
6. Species 150 non-null object dtypes: float64(4), int64(1), object(1)

memory usage: 7.2+ KB

df.columns

Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm', 'Species'],

dtype='object')

df['SepalLengthCm'].max()

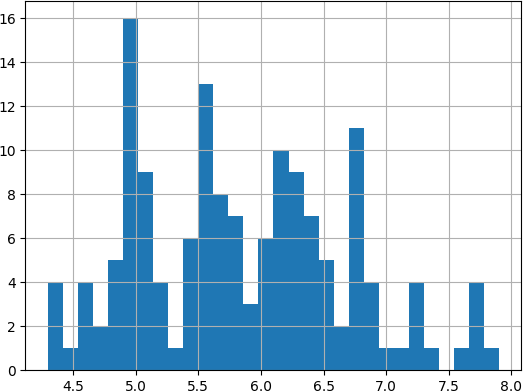
7.9

df['SepalLengthCm'].min()

4.3

df['SepalLengthCm'].hist(bins = 30)

<Axes: >



df['PetalLengthCm'].max()

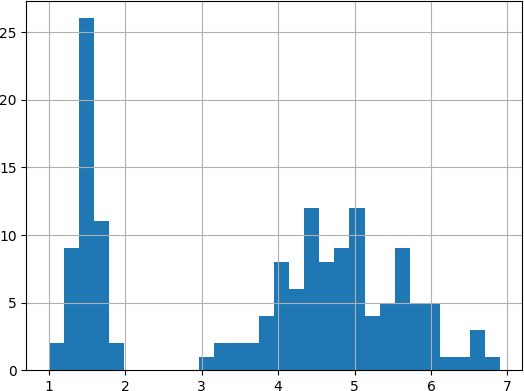
6.9

df['PetalLengthCm'].min()

1.0

df['PetalLengthCm'].hist(bins = 30)

<Axes: >



df['PetalWidthCm'].max()

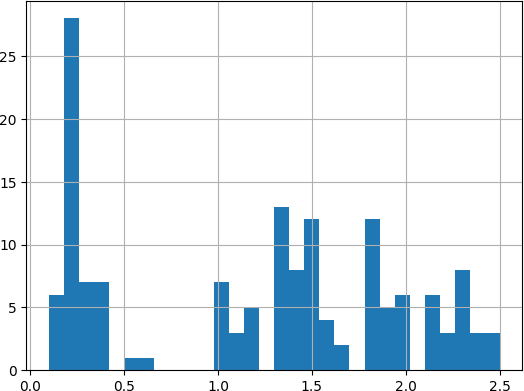
2.5

df['PetalWidthCm'].min()

0.1

df['PetalWidthCm'].hist(bins = 30)

<Axes: >



df['SepalWidthCm'].max()

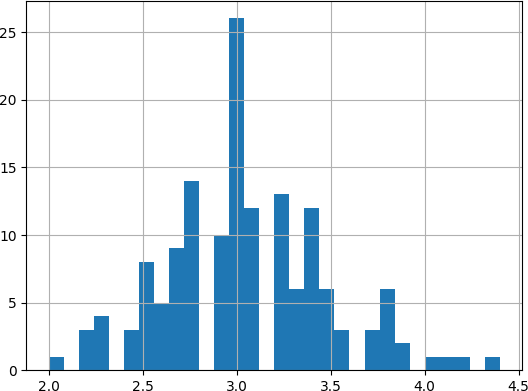
4.4

df['SepalWidthCm'].min()

2.0

df['SepalWidthCm'].hist(bins = 30)

<Axes: >



df['Species'].value\_counts()

Iris-setosa 50

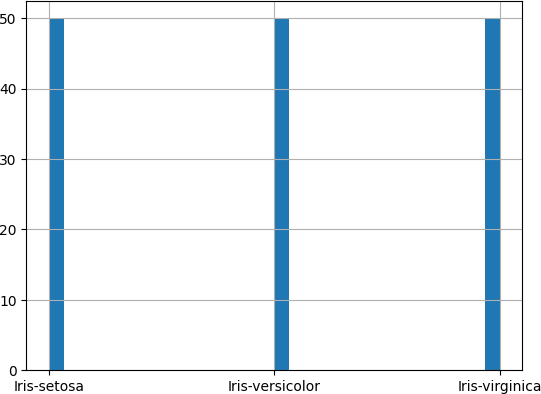
Iris-versicolor 50

Iris-virginica 50

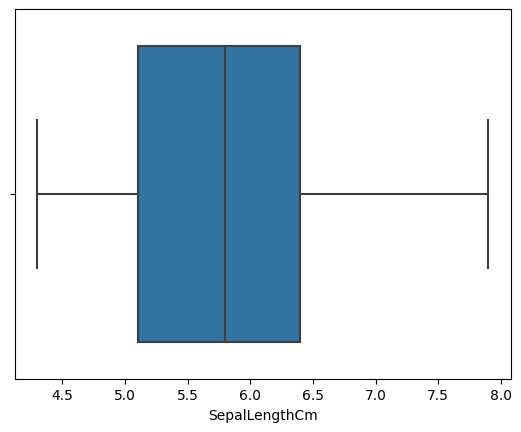
Name: Species, dtype: int64

df['Species'].hist(bins = 30)

<Axes: >



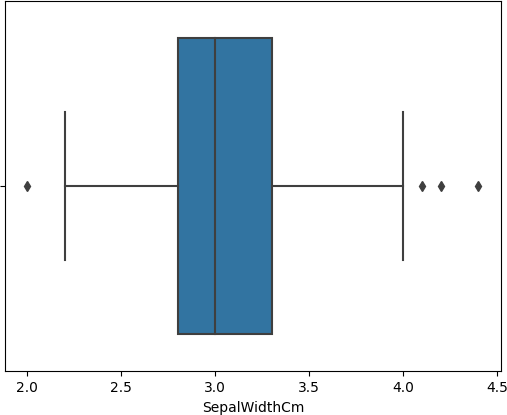
sns.boxplot(x = 'SepalLengthCm' ,data = df)



<Axes: xlabel='SepalLengthCm'>

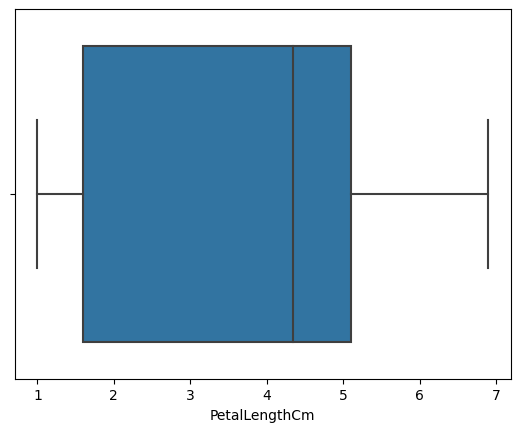
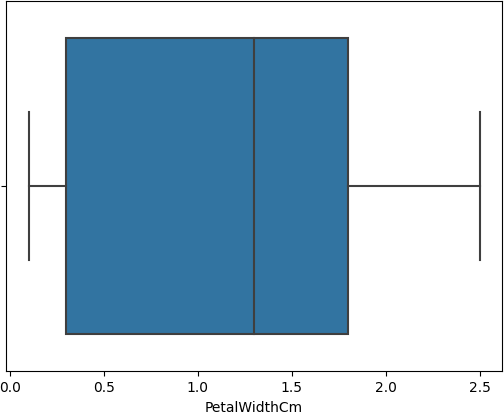
sns.boxplot(x = 'SepalWidthCm' , data = df)

<Axes: xlabel='SepalWidthCm'>



sns.boxplot(x = 'PetalWidthCm' , data = df)

<Axes: xlabel='PetalWidthCm'>



<Axes: xlabel='PetalLengthCm'>

sns.boxplot(x = 'PetalLengthCm', data = df)



0s completed at 7:23 PM

Assignment No 11

**Steps to install Hadoop: Step 1)** mkdir words

**Step 2)** Download hadoop-core-1.2.1.jar, which is used to compile and execute the MapReduce program. Visit the following

**link**

<http://mvnrepository.com/artifact/org.apache.hadoop/hadoop-core/1.2.1>

**Step 3)** Put that downloaded jar file into words folder.

**Step 4)** Implement WordCount.java program.

**Step 5)** Create input1.txt on home directory with some random text

**Step 6)** go on words path then compile

javac -classpath /home/vijay/words/hadoop-core-1.2.1.jar /home/vijay/words/WordCount.java

**Step 7)** jar -cvf words.jar -c words/ .

**Step 8)** cd .. then use following commands hadoop fs -mkdir /input

hadoop fs -put input1.txt /input hadoop fs -ls /input

hadoop jar /home/vijay/words/words12.jar WordCount /input/input1.txt /out321 hadoop fs -ls /out321

hadoop fs -cat /out321/part-r-00000

### (Otherwise check in Browsing HDFS -> Utilities -> Browse the file System -> /)

**Java Code for word count:**

import java.io.IOException; import java.util.\*;

import org.apache.hadoop.conf.\*; import org.apache.hadoop.fs.\*; import org.apache.hadoop.conf.\*; import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.\*;

import org.apache.hadoop.mapreduce.lib.input.\*; import org.apache.hadoop.mapreduce.lib.output.\*; import org.apache.hadoop.util.\*;

public class WordCount extends Configured implements Tool

{

public static void main(String args[]) throws Exception

{

int res = ToolRunner.run(new WordCount(), args); System.exit(res);

}

public int run(String[] args) throws Exception

{

Path inputPath = new Path(args[0]); Path outputPath = new Path(args[1]);

Configuration conf = getConf();

Job job = new Job(conf, this.getClass().toString()); job.setJarByClass(WordCount.class);

FileInputFormat.setInputPaths(job, inputPath); FileOutputFormat.setOutputPath(job, outputPath);

job.setJobName("WordCount");

job.setMapperClass(Map.class); job.setCombinerClass(Reduce.class);

job.setReducerClass(Reduce.class); job.setMapOutputKeyClass(Text.class); job.setMapOutputValueClass(IntWritable.class); job.setOutputKeyClass(Text.class); job.setOutputValueClass(IntWritable.class); job.setInputFormatClass(TextInputFormat.class); job.setOutputFormatClass(TextOutputFormat.class);

return job.waitForCompletion(true) ? 0 : 1;

}

public static class Map extends Mapper<LongWritable, Text, Text, IntWritable>

{

private final static IntWritable one = new IntWritable(1); private Text word = new Text();

public void map(LongWritable key, Text value, Mapper.Context context) throws IOException, InterruptedException

{

String line = value.toString();

StringTokenizer tokenizer = new StringTokenizer(line); while (tokenizer.hasMoreTokens())

{

word.set(tokenizer.nextToken()); context.write(word, one);

}

}

}

public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable>

{

public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException

{

int sum = 0;

for(IntWritable value : values)

{

sum += value.get();

}

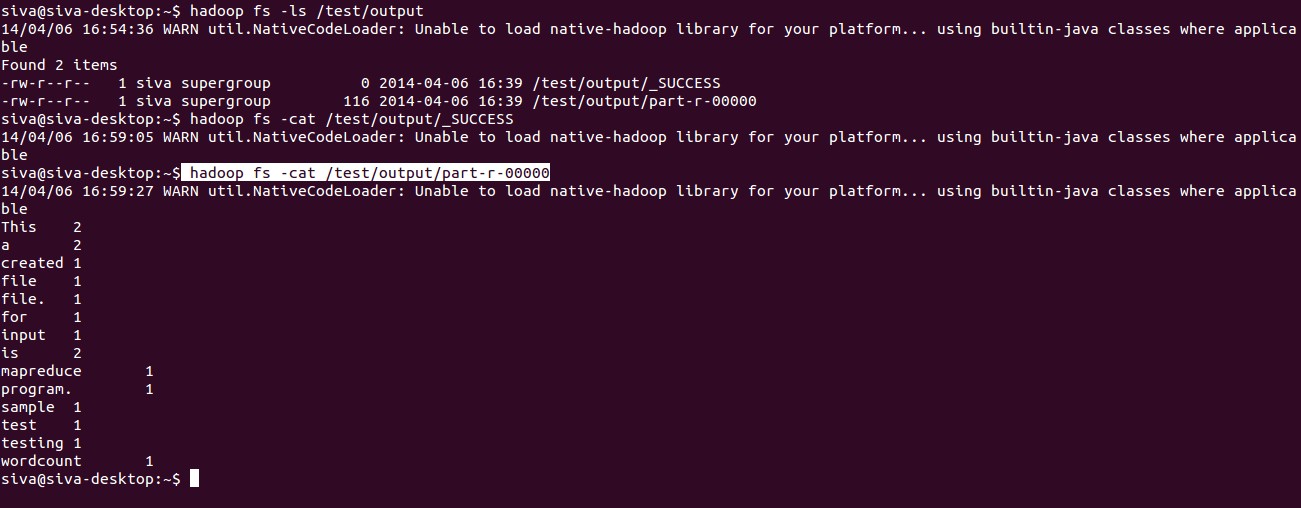
context.write(key, new IntWritable(sum));

}

}

}

**Output:**



## Sample\_Weather.txt file

**Assignment No 12**

|  |  |  |  |
| --- | --- | --- | --- |
| Date, | Temperature, | DewPoint, | WindSpeed |
| 2022-05-01, | 72, | 54, | 5.1 |
| 2022-05-02, | 68, | 55, | 6.5 |
| 2022-05-03, | 71, | 56, | 7.0 |
| 2022-05-04, | 74, | 57, | 7.3 |
| 2022-05-05, | 65, | 54, | 6.2 |
| 2022-05-06, | 63, | 52, | 5.7 |
| 2022-05-07, | 72, | 54, | 5.3 |

## WeatherDataProcessor.Java File

**import** java.io.BufferedReader; **import** java.io.FileReader; **import** java.io.IOException;

**public class** WeatherDataProcessor {

**public static void** main(String[] args) **throws** IOException { String filePath = "E:\\TE\\Sem 6\\DS&BDA

LAB\\DSBDAL\_Assignment12\_TCOB03\\src\\sample\_weather.txt";

// Open the file for reading

BufferedReader reader = **new** BufferedReader(**new** FileReader(filePath));

// Skip the header line

String line = reader.readLine();

// Initialize the sum and count variables

**double** sumTemperature = 0.0; **double** sumDewPoint = 0.0; **double** sumWindSpeed = 0.0; **int** count = 0;

// Read each line of the file

**while** ((line = reader.readLine()) != **null**) {

// Split the line into fields String[] fields = line.split(",");

// Parse the values

**double** temperature = Double.*parseDouble*(fields[1]); **double** dewPoint = Double.*parseDouble*(fields[2]); **double** windSpeed = Double.*parseDouble*(fields[3]);

// Add the values to the sum variables sumTemperature += temperature;

sumDewPoint += dewPoint; sumWindSpeed += windSpeed;

// Increment the count variable count++;

}

// Calculate the averages

**double** avgTemperature = sumTemperature / count; **double** avgDewPoint = sumDewPoint / count; **double** avgWindSpeed = sumWindSpeed / count;

// Print the averages

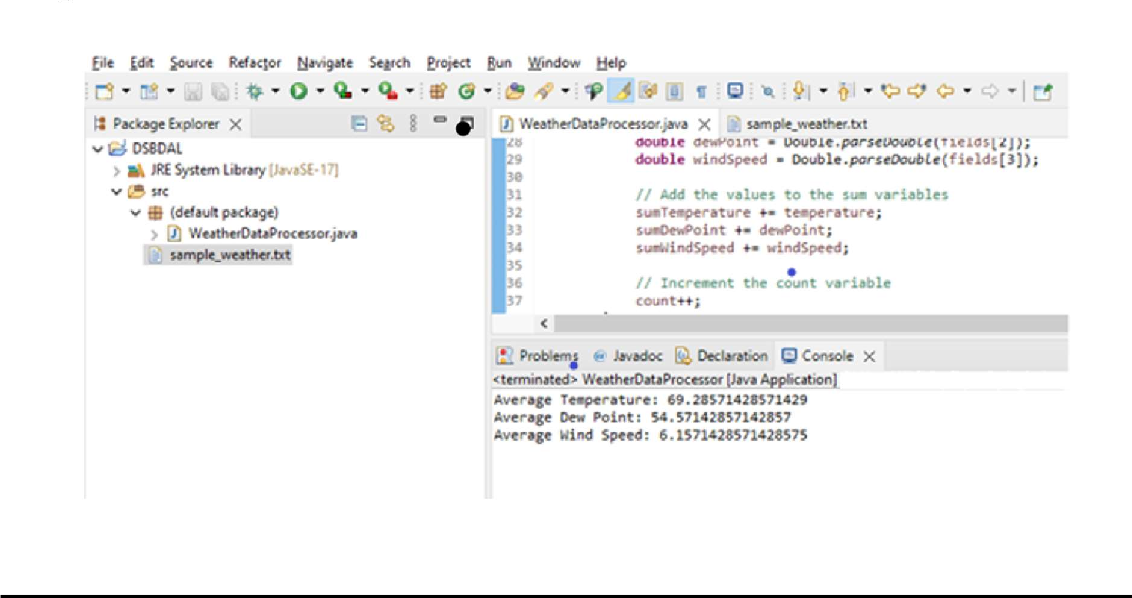
System.***out***.println("Average Temperature: " + avgTemperature); System.***out***.println("Average Dew Point: " + avgDewPoint); System.***out***.println("Average Wind Speed: " + avgWindSpeed);

// Close the file reader.close();

}

}

**Output :**



# Assignment No 13

**Step 1)** java -version

**Step 2)** Install **Scala** from the apt repository by running the following commands to search for scala and install it.

sudo apt search scala ⇒ Search for the package sudo apt install scala ⇒ Install the package

**Step 3)** To verify the installation of **Scala**, run the following command. scala -version

# 2) Apache Spark Framework Installation

Apache Spark is an open-source, distributed processing system used for **big data workloads**. It utilizes in-memory caching, and optimized query execution for fast analytic queries against data of any size.

**Step 1)** Now go to the official Apache Spark download page and grab the latest version (i.e. 3.2.1) at the time of writing this article. Alternatively, you can use the wget command to download the file directly in the terminal.

wget https://apachemirror.wuchna.com/spark/spark-3.2.1/spark-3.2.1-bin-hadoop2.7.tgz

**Step 2)** Extract the Apache Spark tar file. tar -xvzf spark-3.1.1-bin-hadoop2.7.tgz

**Step 3)** Move the extracted **Spark** directory to **/opt** directory. sudo mv spark-3.1.1-bin-hadoop2.7 /opt/spark

**Configure Environmental Variables for Spark**

**Step 4)** Now you have to set a few environmental variables in **.profile** file before starting up the

spark.

echo "export SPARK\_HOME=/opt/spark" >> ~/.profile

echo "export PATH=$PATH:/opt/spark/bin:/opt/spark/sbin" >> ~/.profile echo "export PYSPARK\_PYTHON=/usr/bin/python3" >> ~/.profile

**Step 5)** To make sure that these new environment variables are reachable within the shell and available to Apache Spark, it is also mandatory to run the following command to take recent changes into effect.

source ~/.profile

**Step 6)** ls -l /opt/spark

### Start Apache Spark in Ubuntu

**Step 7)** Run the following command to start the **Spark** master service and slave service. start-master.sh

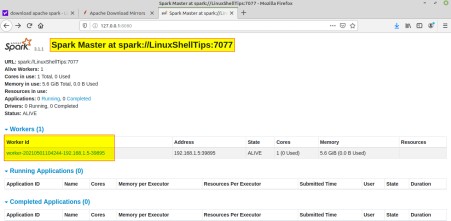
start-workers.sh spark://localhost:7077

(if workers not starting then remove and install openssh:

sudo apt-get remove openssh-client openssh-server sudo apt-get install openssh-client openssh-server)

**Step 8)** Once the service is started go to the browser and type the following URL access spark page. From the page, you can see my master and slave service is started.

http://localhost:8080/



**Step 9)** You can also check if **spark-shell** works fine by launching the **spark-shell**

command. Spark-shell

sudo apt install snapd snap find “intellij”

sudo snap install intellij-idea-community - - classic

### Start Intellij IDE community Edition Source Code:

**/\* Sample Code to print Statement \*/**

object ExampleString {

def main(args: Array[String]) {

//declare and assign string variable "text"

val text : String = "You are reading SCALA programming language.";

//print the value of string variable "text"

println("Value of text is: " + text);

}

}

### /\*\*Scala program to find a number is positive, negative or positive.\*/

object ExCheckNumber {

def main(args: Array[String]) {

/\*\*declare a variable\*/ var number= (-100);

if(number==0){ println("number is zero");

}

else if(number>0){

println("number is positive");

}

else{

println("number is negative");

}

}

}

### /\*Scala program to print your name\*/

object ExPrintName {

def main(args: Array[String]) { println("My name is Mike!")

}

}

### /\*\*Scala Program to find largest number among two numbers.\*/

object ExFindLargest {

def main(args: Array[String]) { var number1=20;

var number2=30; var x = 10;

if( number1>number2){

println("Largest number is:" + number1);

}

else{

}

}

println("Largest number is:" + number2);