1. Create a student table with columns Roll.no, Name, age, marks using pandas and do

the following

a. select top 2 rows

b. filter a data based on some condition with mark&gt;80

c. filter in names first name start with  &#39;N&#39; then remaining

import pandas as pd

s1 = pd.DataFrame({

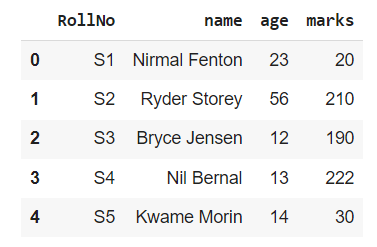
        'RollNo': ['S1', 'S2', 'S3', 'S4', 'S5'],

         'name': ['Nirmal Fenton', 'Ryder Storey', 'Bryce Jensen', 'Nil Bernal', 'Kwame Morin'],

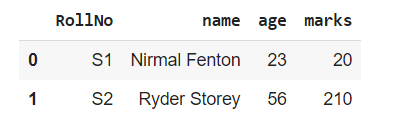
         'age':[23,56,12,13,14],

        'marks': [20, 210, 190, 222, 30]})

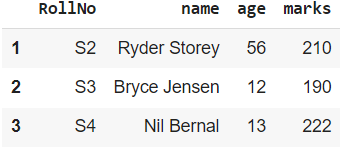
s1



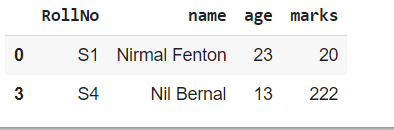
s1.head(2)



s1[s1['marks']>60]



s1[s1['name'].str.startswith('N')]



2. Numpy array creation and basic operations, Initialization ,array indexing

import pandas as pd

import numpy as np

#How to create Series with Mutable index

print(pd.Series(np.array([1,2,3,4,5,6,7]), index=['a','b','c','d','e','f','g']))

#Print all the values of the Series by multiplying them by 2.

print(pd.Series(np.array([1,2,3,4,5,6,7]), index=['a','b','c','d','e','f','g'])\*2)

#Print Square of all the values of the series.

print(pd.Series(np.array([1,2,3,4,5,6,7]), index=['a','b','c','d','e','f','g'])\*\*2)

#Addition of two series

print(pd.Series(np.array([1,2,3,4,5,6,7]))+pd.Series(np.array([11,22,33,44,55,66,77])))

3. Using csv file plot a graph by matplotlib library

import matplotlib.pyplot as plt

# initializing the data

x = [10, 20, 30, 40]

y = [20, 25, 35, 55]

# plotting the data

plt.plot(x, y)

plt.show()

# Adding title to the plot

plt.title("Linear graph")

plt.show()

plt.title("Linear graph", fontsize=25, color="green")

plt.show()

# Adding label on the y-axis

plt.ylabel('Y-Axis')

# Adding label on the x-axis

plt.xlabel('X-Axis')

plt.show()

#Setting the limit of y-axis

plt.ylim(0, 80)

# setting the labels of x-axis

plt.xticks(x, labels=["one", "two", "three", "four"])

plt.show()

# Adding legends

plt.legend(["GFG"])

plt.show()

import matplotlib.pyplot as plt

import numpy

# Create figure() objects

# This acts as a container

# for the different plots

fig = plt.figure()

# Generate line graph

x = numpy.arange(0, 1.414\*2, 0.05)

y1 = numpy.sin(x)

y2 = numpy.cos(x)

# Creating two axes

# add\_axes([xmin,ymin,dx,dy])

axes1 = fig.add\_axes([0, 0, 1, 1])

axes1.plot(x, y1)

axes2 = fig.add\_axes([0, 1, 1, 1])

axes2.plot(x, y2)

# Show plot

plt.show()

import matplotlib.pyplot as plt

x  = [1, 2, 3, 4, 5, 6, 7, 8, 9]

y1 = [1, 3, 5, 3, 1, 3, 5, 3, 1]

y2 = [2, 4, 6, 4, 2, 4, 6, 4, 2]

plt.plot(x, y1, label="line L")

plt.plot(x, y2, label="line H")

plt.plot()

plt.xlabel("x axis")

plt.ylabel("y axis")

plt.title("Line Graph Example")

plt.legend()

plt.show()

import matplotlib.pyplot as plt

# Look at index 4 and 6, which demonstrate overlapping cases.

x1 = [1, 3, 4, 5, 6, 7, 9]

y1 = [4, 7, 2, 4, 7, 8, 3]

x2 = [2, 4, 6, 8, 10]

y2 = [5, 6, 2, 6, 2]

# Colors: https://matplotlib.org/api/colors\_api.html

plt.bar(x1, y1, label="Blue Bar", color='b')

plt.bar(x2, y2, label="Green Bar", color='g')

plt.plot()

plt.xlabel("bar number")

plt.ylabel("bar height")

plt.title("Bar Chart Example")

plt.legend()

plt.show()

import matplotlib.pyplot as plt

import numpy as np

# Use numpy to generate a bunch of random data in a bell curve around 5.

n = 5 + np.random.randn(1000)

m = [m for m in range(len(n))]

plt.bar(m, n)

plt.title("Raw Data")

plt.show()

plt.hist(n, bins=20)

plt.title("Histogram")

plt.show()

plt.hist(n, cumulative=True, bins=20)

plt.title("Cumulative Histogram")

plt.show()

import matplotlib.pyplot as plt

x1 = [2, 3, 4]

y1 = [5, 5, 5]

x2 = [1, 2, 3, 4, 5]

y2 = [2, 3, 2, 3, 4]

y3 = [6, 8, 7, 8, 7]

# Markers: https://matplotlib.org/api/markers\_api.html

plt.scatter(x1, y1)

plt.scatter(x2, y2, marker='v', color='r')

plt.scatter(x2, y3, marker='^', color='m')

plt.title('Scatter Plot Example')

plt.show()

#Bubble chart

import plotly.express as px

df = px.data.iris()

fig = px.scatter(df, x="sepal\_width", y="sepal\_length",

        color="species",

        size='petal\_length',

        hover\_data=['petal\_width'])

fig.show()

#distribution chart

import seaborn as sns

penguins = sns.load\_dataset("penguins")

sns.displot(penguins, x="flipper\_length\_mm")

4.Perform all matrix operation using python (using numpy)

import numpy as np

a = np.array([1, 2, 3])   # Create a rank 1 array

print("type: " ,type(a))            # Prints "<class 'numpy.ndarray'>"

print("shape: " ,a.shape)            # Prints "(3,)"

print(a[0], a[1], a[2])   # Prints "1 2 3"

a[0] = 5                  # Change an element of the array

print(a)                  # Prints "[5, 2, 3]"

b = np.array([[1,2,3],[4,5,6]])    # Create a rank 2 array

print("\n shape of b:",b.shape)                     # Prints "(2, 3)"

print(b[0, 0], b[0, 1], b[1, 0])   # Prints "1 2 4"

a = np.zeros((3,3))   # Create an array of all zeros

print("All zeros matrix:\n  " ,a)              # Prints "[[ 0.  0.]

                      #          [ 0.  0.]]"

b = np.ones((1,2))    # Create an array of all ones

print("\nAll ones matrix:\n  " ,b)              # Prints "[[ 1.  1.]]"

d = np.eye(2)         # Create a 2x2 identity matrix

print("\n identity matrix: \n",d)              # Prints "[[ 1.  0.]

                      #          [ 0.  1.]]"

e = np.random.random((2,2))  # Create an array filled with random values

print("\n random matrix: \n",e)

5. Program to Perform SVD (Singular Value Decomposition) in Python

# Singular-value decomposition

from numpy import array

from scipy.linalg import svd

# define a matrix

A = array([[1, 2], [3, 4], [5, 6]])

print("A: \n%s" %A)

# SVD

U, s, VT = svd(A)

print("\nU: \n%s" %U)

print("\ns: \n %s" %s)

print("\nV^T: \n %s" %VT)

6. Program to implement k-NN classification using any standard dataset available in the public domain and find the accuracy of the algorithm.

import matplotlib.pyplot as plt

from sklearn.neighbors import KNeighborsClassifier

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

from sklearn.metrics import accuracy\_score

import pandas as pd

import numpy as np

from sklearn import tree

from sklearn.datasets import load\_iris

data = load\_iris()

df = pd.DataFrame(data.data, columns=data.feature\_names)

df['target'] = data.target

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(df[data.feature\_names], df['target'], random\_state=42,test\_size=0.1)

clf = KNeighborsClassifier(n\_neighbors = 5)

clf.fit(X\_train, Y\_train)

y\_pred=clf.predict(X\_test)

# comparing actual response values (y\_test) with predicted response values (y\_pred)

from sklearn import metrics

print("KNN model accuracy(in %):", metrics.accuracy\_score(Y\_test, y\_pred)\*100)

7.Program to implement Naive Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm

# load the iris dataset

from sklearn.datasets import load\_iris

iris = load\_iris()

# store the feature matrix (X) and response vector (y)

X = iris.data

y = iris.target

# splitting X and y into training and testing sets

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=42)

# training the model on training set

from sklearn.naive\_bayes import GaussianNB

gnb = GaussianNB()

gnb.fit(X\_train, y\_train)

# making predictions on the testing set

y\_pred = gnb.predict(X\_test)

# comparing actual response values (y\_test) with predicted response values (y\_pred)

from sklearn import metrics

print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy\_score(y\_test, y\_pred)\*100)

8.Program to implement linear and multiple regression techniques using any standard dataset available in the public domain and evaluate its performance.

import numpy as np

from sklearn.linear\_model import LinearRegression

x = [[0, 1], [5, 1], [15, 2], [25, 5], [35, 11], [45, 15], [55, 34], [60, 35]]

y = [4, 5, 20, 14, 32, 22, 38, 43]

x, y = np.array(x), np.array(y)

print(x)

print(y)

model = LinearRegression().fit(x, y)

r\_sq = model.score(x, y)

print(f"coefficient of determination: {r\_sq}")

print(f"intercept: {model.intercept\_}")

print(f"coefficients: {model.coef\_}")

y\_pred = model.predict(x)

print(f"predicted response:\n{y\_pred}")

x\_new = np.arange(10).reshape((-1, 2))

print(x\_new)

y\_new = model.predict(x\_new)

y\_new

9.Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of the algorithm.

import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeClassifier

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

from sklearn.metrics import accuracy\_score

import pandas as pd

import numpy as np

from sklearn import tree

from sklearn.datasets import load\_iris

data = load\_iris()

df = pd.DataFrame(data.data, columns=data.feature\_names)

df['target'] = data.target

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(df[data.feature\_names], df['target'], random\_state=42,test\_size=0.1)

clf = DecisionTreeClassifier(max\_depth=2,random\_state=42)

clf.fit(X\_train, Y\_train)

y\_pred=clf.predict(X\_test)

# comparing actual response values (y\_test) with predicted response values (y\_pred)

from sklearn import metrics

print("Decision tre model accuracy(in %):", metrics.accuracy\_score(Y\_test, y\_pred)\*100)

10. Program to implement k- means clustering technique using any standard dataset available in the public domain

from sklearn.cluster import KMeans

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from matplotlib import pyplot as plt

%matplotlib inline

df = pd.read\_csv("income.csv")

df.head()

scaler = MinMaxScaler()

scaler.fit(df[['Income($)']])

df['Income($)'] = scaler.transform(df[['Income($)']])

scaler.fit(df[['Age']])

df['Age'] = scaler.transform(df[['Age']])

plt.scatter(df.Age,df['Income($)'])

km = KMeans(n\_clusters=3)

y\_predicted = km.fit\_predict(df[['Age','Income($)']])

y\_predicted

df['cluster']=y\_predicted

df.head()

km.cluster\_centers\_

df1 = df[df.cluster==0]

df2 = df[df.cluster==1]

df3 = df[df.cluster==2]

plt.scatter(df1.Age,df1['Income($)'],color='green')

plt.scatter(df2.Age,df2['Income($)'],color='red')

plt.scatter(df3.Age,df3['Income($)'],color='black')

plt.scatter(km.cluster\_centers\_[:,0],km.cluster\_centers\_[:,1],color='purple',marker='\*',label='centroid')

plt.legend()

11. Program to implement simple web crawler using python

import requests

from bs4 import BeautifulSoup

import csv

URL = "[http://www.ajce.in](http://www.ajce.in/)"

r = requests.get(URL)

soup = BeautifulSoup(r.content, 'html5lib')

print(soup.prettify())

import requests

from bs4 import BeautifulSoup

import csv

URL = "[http://www.ajce.in](http://www.ajce.in/)"

r = requests.get(URL)

soup = BeautifulSoup(r.content, 'html5lib')

print(soup.prettify())

from bs4 import BeautifulSoup

import requests

pages\_crawler = []

def crawler(url):

    page = requests.get(url)

    soup = BeautifulSoup(page.text, 'html.parser')

    links = soup.find\_all('a')

    for link in links:

        if 'href' in link.attrs:

            if link['href'].startswith('/wiki') and ':' not in link['href']:

                if link['href'] not in pages\_crawler:

                    new\_link = f"[https://en.wikipedia.org](https://en.wikipedia.org/){link['href']}"

                    pages\_crawler.append(link['href'])

                    try:

                        with open('data.csv', 'a') as file:

                            file.write(f'{soup.title.text}; {soup.h1.text}; {link["href"]}\n')

                        crawler(new\_link)

                    except:

                        continue

crawler('[https://en.wikipedia.org](https://en.wikipedia.org/)')

12. Program to implement scrap of any website

import requests

from bs4 import BeautifulSoup

import csv

URL = "http://www.ajce.in"

r = requests.get(URL)

soup = BeautifulSoup(r.content, 'html5lib')

print(soup.prettify())

13. Program for Natural Language Processing which performs n- grams(Using inbuilt functions)

import nltk

from nltk.util import ngrams

text = "this is a very good book to study";

Ngrams = ngrams(sequence=nltk.wordpunct\_tokenize(text), n=3)

for grams in Ngrams:

 print(grams)

14. Program for Natural Language Processing which perform parts of speech tagging.

import nltk

from nltk.tag import DefaultTagger

exptagger = DefaultTagger('NN')

exptagger.tag\_sents([['Hi', ','], ['How', 'are', 'you', '?']])

import nltk

from nltk.tag import untag

untag([('Tutorials', 'NN'), ('Point', 'NN')])

import nltk

# import all the resources for Natural Language Processing with Python

nltk.download("book")

#Take a sentence and tokenize into words. Then apply a part-of-speech tagger.

sentence = """At eight o'clock on Thursday morning

Arthur didn't feel very good."""

tokens = nltk.word\_tokenize(sentence)

print(tokens)

tagged = nltk.pos\_tag(tokens)

print(tagged)

text ="learn php from guru99 and make study easy".split()

print("After Split:",text)

tokens\_tag = nltk.pos\_tag(text)

print("After Token:",tokens\_tag)