**Name: Upas Nath**

**Roll No: 47**

**Batch: MCA-B**

**Date: 1-09-2022**

**DATA SCIENCE LAB**

**Experiment No.: 1**

**Aim**

Aim: To implement

(a) Matrix operations (using vectorization),

(b) transformation using python and

(c) SVD using Python.

**Procedure**

**a)1 Matrix operations**

import numpy as np

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

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

print("shape: %s" %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((2,2)) # Create an array of all zeros

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

# [ 0. 0.]]"

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

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

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

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

# [ 0. 1.]]"

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

print("\n random matrix: \n%s"%e)

**a)2**

#vectorized sum

print("Vectorized sum example\n")

x = np.array([[1,2],[3,4]])

print("x:\n %s" %x)

print("sum: %s"%np.sum(x))  # Compute sum of all elements; prints "10"

print("sum axis = 0: %s" %np.sum(x, axis=0))  # Compute sum of each column; prints "[4 6]"

print(" sum axis = 1: %s" %np.sum(x, axis=1))  # Compute sum of each row; prints "[3 7]"

#matrix dot product

a = np.arange(10000)

b = np.arange(10000)

dp = np.dot(a,b)

print("Dot product: %s\n" %dp)

#outer product

op = np.outer(a,b)

print("\n Outer product: %s\n" %op)

#elementwise product

ep = np.multiply(a, b)

print("\n Element Wise product: %s \n" %ep)

**b) Matrix transformation**

import numpy as np

x = np.array([[1,2], [3,4]])

print("Original x: \n%s " %x)    # Prints "[[1 2]

            #          [3 4]]"

print("\nTranspose of x: \n%s" %x.T)  # Prints "[[1 3]

            #          [2 4]]"

**c) SVD using 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)

**Output Screenshot**

type: <class 'numpy.ndarray'>

shape: 3

1 2 3

[5 2 3]

shape of b: (2, 3)

1 2 4

All zeros matrix:

[[0. 0.]

[0. 0.]]

All ones matrix:

[[1. 1.]]

identity matrix:

[[1. 0.]

[0. 1.]]

random matrix:

[[0.19450703 0.60931147]

[0.71466669 0.22569737]]

a.1

Vectorized sum example

x:

[[1 2]

[3 4]]

sum: 10

sum axis = 0: [4 6]

sum axis = 1: [3 7]

Dot product: 333283335000

Outer product: [[ 0 0 0 ... 0 0 0]

[ 0 1 2 ... 9997 9998 9999]

[ 0 2 4 ... 19994 19996 19998]

...

[ 0 9997 19994 ... 99940009 99950006 99960003]

[ 0 9998 19996 ... 99950006 99960004 99970002]

[ 0 9999 19998 ... 99960003 99970002 99980001]]

Element Wise product: [ 0 1 4 ... 99940009 99960004 99980001]

**b**

Original x:

[[1 2]

[3 4]]

Transpose of x:

[[1 3]

[2 4]]

**C**

A:

[[1 2]

[3 4]

[5 6]]

U:

[[-0.2298477 0.88346102 0.40824829]

[-0.52474482 0.24078249 -0.81649658]

[-0.81964194 -0.40189603 0.40824829]]

s:

[9.52551809 0.51430058]

V^T:

[[-0.61962948 -0.78489445]

[-0.78489445 0.61962948]]

**DATA SCIENCE LAB**

**Name: Rizwan**

**Roll No:22**

**Batch: MCA-B**

**Date: 1-09-2022**

**Experiment No.: 2**

**Aim**

Programs using matplotlib / plotly / bokeh / seaborn for data visualisation.

**Procedure and Output**

**Histogram**

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)

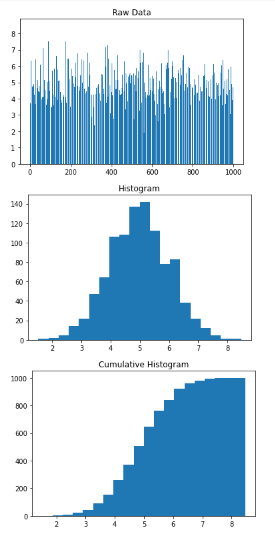
plt.title("Histogram")

plt.show()

plt.hist(n, cumulative=True, bins=20)

plt.title("Cumulative Histogram")

plt.show()



**Distribution Chart**

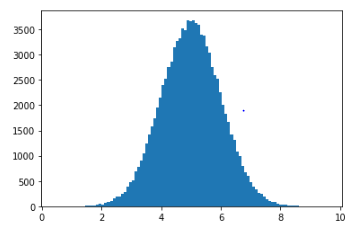
import numpy

import matplotlib.pyplot as plt

x = numpy.random.normal(5.0, 1.0, 100000)

plt.hist(x, 100)

plt.show()



**Bubble Chart**

import matplotlib.pyplot as plt

import numpy as np

# create data

x = np.random.rand(40)

y = np.random.rand(40)

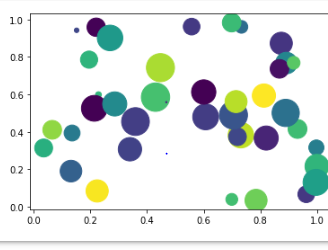
z = np.random.rand(40)

colors = np.random.rand(40)

# use the scatter function

plt.scatter(x, y, s=z\*1000,c=colors)

plt.show()



**Scatter Plot**

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

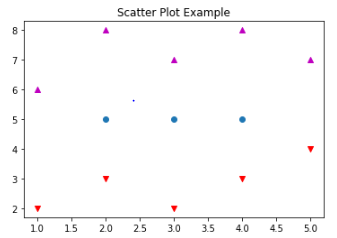
plt.scatter(x1, y1)

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

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

plt.title('Scatter Plot Example')

plt.show()



**Line graph**

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()

**output**



**Bar chart**

import matplotlib.pyplot as plt

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]

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

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

plt.plot()

plt.xlabel("bar number")

plt.ylabel("bar height")

plt.title("Bar Chart Example")

plt.legend()

plt.show()

**output**



**Box plot**

plt.figure()

plt.suptitle("Boxplot for X vs Y split into 5 bins")

ax = plt.gca()

df2.boxplot(showmeans=True)

# Rotate x axis text values

for tick in ax.get\_xticklabels():

    tick.set\_rotation(30)

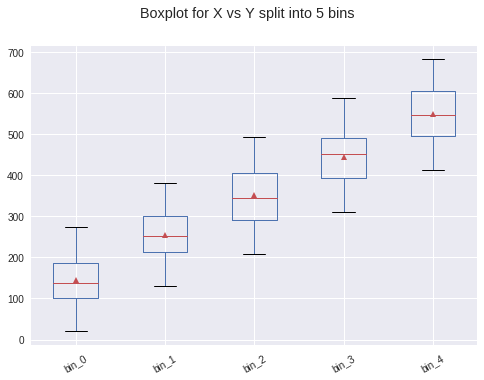
print("\nIn the boxplot below, the box extends from the lower to upper quartile values of the data, with a line at the median.\n \

The whiskers extend from the box to show the range of the data. The triangle indicates the mean value.\n")

**output**

In the boxplot below, the box extends from the lower to upper quartile values of the data, with a line at the median.

 The whiskers extend from the box to show the range of the data. The triangle indicates the mean value.



**Name: Rizwan**

**Roll No:22**

**Batch: MCA-B**

**Date:01-09-2022**

**DATA SCIENCE LAB**

**Experiment No.: 3**

**Aim**

 Programs to handle data using pandas.

**Question**

**Q1 - Pandas  Series**

1. How to create Series with nd array
2. How to create Series with Mutable index
3. Creating a series from a Dictionary
4. Print all the values of the Series by multiplying them by 2.
5. Print Square of all the values of the series.
6. Print all the values of the Series that are greater than2
7. Addition of two series
8. Print the first and last 5 elements of a series
9. Print the values from index 0 to 5
10. Selection Using loc, iloc index label
11. Retrieve subsets of data using slicing

**Q2 Dataframe**

1. create Dataframe From Series
2. DataFrame from List of Dictionaries
3. Display the first 5 rows of data frame
4. Select the last two columns of the data frame
5. Add two data frames
6. Demonstrate deletion, and renaming of columns
7. Demonstrate concat, Merge operations in data frame
8. Write a Pandas program to join the two given dataframes along rows and assign all data

**Test Data:**

student\_data1:

  student\_id              name  marks

0         S1  Danniella Fenton    200

1         S2      Ryder Storey    210

2         S3      Bryce Jensen    190

3         S4         Ed Bernal    222

4         S5       Kwame Morin    199

student\_data2:

  student\_id              name  marks

0         S4  Scarlette Fisher    201

1         S5  Carla Williamson    200

2         S6       Dante Morse    198

3         S7    Kaiser William    219

4         S8   Madeeha Preston    201

**Procedure and Output**

#1.How to create Series with nd array

import pandas as pd

import numpy as np

arr=np.array([10,15,18,22])

s = pd.Series(arr)

print(s)

**out put**

0    10

1    15

2    18

3    22

dtype: int64

2.How to create Series with Mutable index

import pandas as pd

import numpy as np

arr=np.array(['a','b','c','d'])

s=pd.Series(arr,

index=['first','second','third','fourth'])

print(s)

**out put**

first     a

second    b

third     c

fourth    d

dtype: object

3Creating a series from a Dictionary

import pandas as pd

s={'name':'hardik','iplteam':'mi','runs':100}

p=pd.Series(s)

print(p)

**out put**

name       hardik

iplteam        mi

runs          100

dtype: object

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

import pandas as pd

p=pd.Series([1,2,3,4,5])

print(p)

print("multlipling all values in series by 2")

print(p\*2)

**out put**

0    1

1    2

2    3

3    4

4    5

dtype: int64

multlipling all values in series by 2

0     2

1     4

2     6

3     8

4    10

dtype: int64

5.Print Square of all the values of the series

import pandas as pd

p=pd.Series([1,2,3,4,5])

print('..............................................')

print("square of all values")

print(p\*\*2)

 ')

0    1

1    2

2    3

3    4

4    5

dtype: int64

square of all values

0     1

1     4

2     9

3    16

4    25

dtype: int64

6 Print all the values of the Series that are greater than2

import pandas as pd

p=pd.Series([1,2,3,4,5])

print("when the value greater than 2")

print(p[p>2])

print('..............................................')

out put

when the value greater than 2

2    3

3    4

4    5

dtype: int64

7.Addition of two series

import pandas as pd

s1=pd.Series([1,2,3,4,5],index=['a','b','c','d','e'])

s2=pd.Series([1,2,3,4,5],index=['a','b','c','d','e'])

print(s1)

print(s2)

print(s1+s2)

**out put**

a    1

b    2

c    3

d    4

e    5

dtype: int64

a    1

b    2

c    3

d    4

e    5

dtype: int64

a     2

b     4

c     6

d     8

e    10

dtype: int64

8. Print the first and last 5 elements of a series

import pandas as pd

import numpy as np

arr=np.array([10,12,23,3,4,56,57,6,7])

s=pd.Series(arr)

print(s.head(5))

**output**

0    10

1    12

2    23

3     3

4     4

dtype: int64

9. Print the values from index 0 to 5

import pandas as pd

import numpy as np

arr=np.array([10,12,23,3,4,56,57,6,7])

s=pd.Series(arr)

print(s.head(6))

**output**

0    10

1    12

2    23

3     3

4     4

5    56

dtype: int64

10.Selection Using loc, iloc index label

import pandas as pd

import numpy as np

arr=np.array([10,12,23,3,4,56,57,6,7])

s=pd.Series(arr)

print(s)

print(s.loc[:2])

print(s.iloc[3:4])

**output**

0    10

1    12

2    23

3     3

4     4

5    56

6    57

7     6

8     7

dtype: int64

0    10

1    12

2    23

dtype: int64

3    3

dtype: int64

11.Retrieve subsets of data using slicing

import pandas as pd

import numpy as np

arr=np.array([10,12,23,3,4])

s=pd.Series(arr,index=['A','B','C','D','E'])

print(s)

print(s[::-1])

**output**

A    10

B    12

C    23

D     3

E     4

dtype: int64

E     4

D     3

C    23

B    12

A    10

dtype: int64

1.create Dataframe From Series

import pandas as pd

s = pd.Series(['a','b','c','d'])

df=pd.DataFrame(s)

print(df)

output

  0

0  a

1  b

2  c

3  d

2 DataFrame from List of Dictionaries

import pandas as pd

l=[{'Name':'sachin','city':'kerala'},

   {'Name':'virat','city':'tamilnadu'}]

d=pd.DataFrame(l)

print(d)

output

Name       city

0  sachin     kerala

1. virat  tamilnadu

3.Display the first 5 rows of data frame

import pandas as pd

empdata = {'empid':[1,2,3,4,5,6],'ename':['Vimal','Sachin','Bav','Kumar','Ravy','Sunil']}

df=pd.DataFrame(empdata)

print(df)

print(df.head(5))

output

empid   ename

0      1   Vimal

1      2  Sachin

2      3     Bav

3      4   Kumar

4      5    Ravy

5      6   Sunil

   empid   ename

0      1   Vimal

1      2  Sachin

2      3     Bav

3      4   Kumar

1. 5    Ravy

4.Select the last two columns of the data frame

import pandas as pd

empdata = {'empid':[1,2,3,4,5,6], 'ename':['Vimal','Sachin','Bav','Kumar','Ravy','Sunil']}

df=pd.DataFrame(empdata)

print(df)

df.loc[0:5]

print(df.tail(2))

output

empid   ename

0      1   Vimal

1      2  Sachin

2      3     Bav

3      4   Kumar

4      5    Ravy

5      6   Sunil

   empid  ename

4      5   Ravy

5      6  Sunil

6. Demonstrate deletion, and renaming of columns

 import pandas as pd dic1= {'id':['1','2','3','4','5'],'value1':['A','C','E','G','I'],'value2':['B','D','F','H','J']} dic2= {'id':['2','3','6','7','8'],'value1':['K','M','O','Q','S'],'value2':['L','N','P','R','T']} dic3= {'id':['1','2','3','4','5','7','8','9','10','11'],'value3':[12,13,14,15,16,17,15,12,13,23]} df1=pd.DataFrame(dic1) df2=pd.DataFrame(dic2) df3=pd.concat([df1,df2]) df4=pd.DataFrame(dic3) df5=pd.merge(df3,df4,on='id') print(df5)

id value1 value2 value3 0 1 A B 12 1 2 C D 13 2 2 K L 13 3 3 E F 14 4 3 M N 14 5 4 G H 15 6 5 I J 16 7 7 Q R 17 8 8 S T 15

7 Demonstrate concat, Merge operations in data frame

import pandas as pd

s= pd.Series([10,20,30,40])

df=pd.DataFrame(s)

df.columns=['List1']

df['List2']=40

df1=df.drop('List2',axis=1)

df2=df.drop(index=[2,3],axis=0)

print(df)

print(" After deletion::")

print(df1)

print (" After row deletion::")

print(df2)

output

List1  List2

0     10     40

1     20     40

2     30     40

3     40     40

 After deletion::

   List1

0     10

1     20

2     30

3     40

 After row deletion::

   List1  List2

0     10     40

1. 20     40

8.Write a Pandas program to join the two given dataframes along rows and assign all data

**Test Data:**

student\_data1:

  student\_id              name  marks

0         S1  Danniella Fenton    200

1         S2      Ryder Storey    210

2         S3      Bryce Jensen    190

3         S4         Ed Bernal    222

4         S5       Kwame Morin    199

student\_data2:

  student\_id              name  marks

0         S4  Scarlette Fisher    201

1         S5  Carla Williamson    200

2         S6       Dante Morse    198

3         S7    Kaiser William    219

4         S8   Madeeha Preston    201

Out put

Original DataFrames:

  student\_id              name  marks

0         S1  Danniella Fenton    200

1         S2      Ryder Storey    210

2         S3      Bryce Jensen    190

3         S4         Ed Bernal    222

4         S5       Kwame Morin    199

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

  student\_id              name  marks

0         S4  Scarlette Fisher    201

1         S5  Carla Williamson    200

2         S6       Dante Morse    198

3         S7    Kaiser William    219

4         S8   Madeeha Preston    201

Join the said two dataframes along rows:

  student\_id              name  marks

0         S1  Danniella Fenton    200

1         S2      Ryder Storey    210

2         S3      Bryce Jensen    190

3         S4         Ed Bernal    222

4         S5       Kwame Morin    199

0         S4  Scarlette Fisher    201

1         S5  Carla Williamson    200

2         S6       Dante Morse    198

3         S7    Kaiser William    219

4         S8   Madeeha Preston    201

**Name: Rizwan**

**Roll No:22**

**Batch:MCA-B**

**Date:12-09-2022**

**DATA SCIENCE LAB**

**Experiment No.: 4**

**Aim**

Perform Z-score normalization, Min-max normalization

**Procedure and Output**

1. Z-score Normalization

import pandas as pd

import numpy as np

import scipy.stats as stats

data = np.array([6, 7, 7, 12, 13, 13, 15, 16, 19, 22])

print("\n Data before aplying z-score operation\n",data) # z-score normalization

new\_data=stats.zscore(data)

print("Normalized Data are:\n",new\_data)

**Output**

Data before aplying z-score operation

[ 6 7 7 12 13 13 15 16 19 22]

Normalized Data are:

[-1.39443338 -1.19522861 -1.19522861 -0.19920477 0. 0.

0.39840954 0.5976143 1.19522861 1.79284291]

2. Min-max Normalization

from numpy import asarray

from sklearn.preprocessing import MinMaxScaler

# define data

data = asarray([[100, 0.001],

[8, 0.05],

[50, 0.005],

[88, 0.07],

[4, 0.1]])

print("\n before normalization\n",data)

# define min max scaler #min max normalization's another example

scaler = MinMaxScaler()

# transform data

print("\n After applying transformation")

scaled = scaler.fit\_transform(data)

print(scaled)

Output

before normalization

[[1.0e+02 1.0e-03]

[8.0e+00 5.0e-02]

[5.0e+01 5.0e-03]

[8.8e+01 7.0e-02]

[4.0e+00 1.0e-01]]

After applying transformation

[[1. 0. ]

[0.04166667 0.49494949]

[0.47916667 0.04040404]

[0.875 0.6969697 ]

[0. 1. ]]

**Name: Rizwan**

**Roll No:22**

**Batch: MCA-B**

**Date:15-09-2022**

**DATA SCIENCE LAB**

**Experiment No.: 5**

**Aim**

 Implement KNN Algorithm using python

**Procedure**

from math import sqrt

# calculate the Euclidean distance between two vectors

def euclidean\_distance(row1, row2):

distance = 0.0

for i in range(len(row1)-1):

distance += (row1[i] - row2[i])\*\*2

return sqrt(distance)

# Locate the most similar neighbors

def get\_neighbors(train, test\_row, num\_neighbors):

distances = list()

for train\_row in train:

dist = euclidean\_distance(test\_row, train\_row)

distances.append((train\_row, dist))

distances.sort(key=lambda tup: tup[1])

neighbors = list()

for i in range(num\_neighbors):

neighbors.append(distances[i][0])

return neighbors

# Make a classification prediction with neighbors

def predict\_classification(train, test\_row, num\_neighbors):

neighbors = get\_neighbors(train, test\_row, num\_neighbors)

output\_values = [row[-1] for row in neighbors]

prediction = max(set(output\_values), key=output\_values.count)

return prediction

# Test distance function

dataset = [[2.7810836,2.550537003,0],

[1.465489372,2.362125076,0],

[3.396561688,4.400293529,0],

[1.38807019,1.850220317,0],

[3.06407232,3.005305973,0],

[7.627531214,2.759262235,1],

[5.332441248,2.088626775,1],

[6.922596716,1.77106367,1],

[8.675418651,-0.242068655,1],

[7.673756466,3.508563011,1]]

prediction = predict\_classification(dataset, dataset[0], 3)

print('Expected %d, Got %d.' % (dataset[0][-1], prediction))

**Output**

Expected 0, Got 0.