DATA SCIENCE LAB

Name: Manya Madhu

Roll No:17

Batch: MCA-B

Date: 24-08-2022

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: Manya Madhu

Roll No:17

Batch: MCA-B

Date: 24-08-2022

Experiment No.: 2

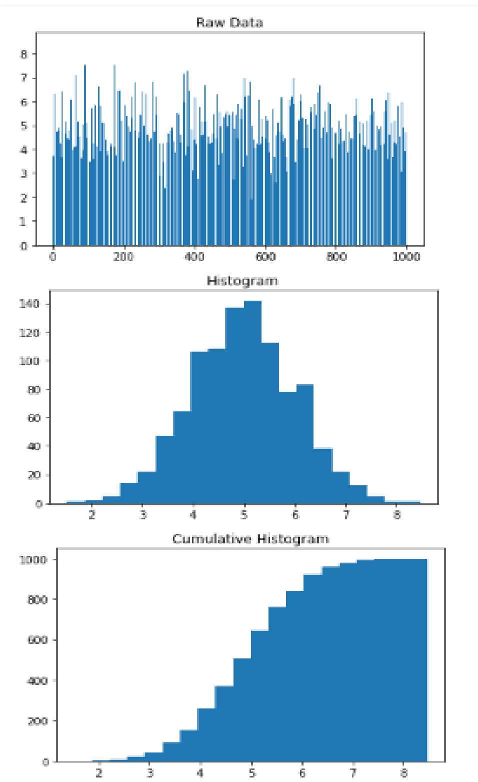
Aim

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

Procedure and Output

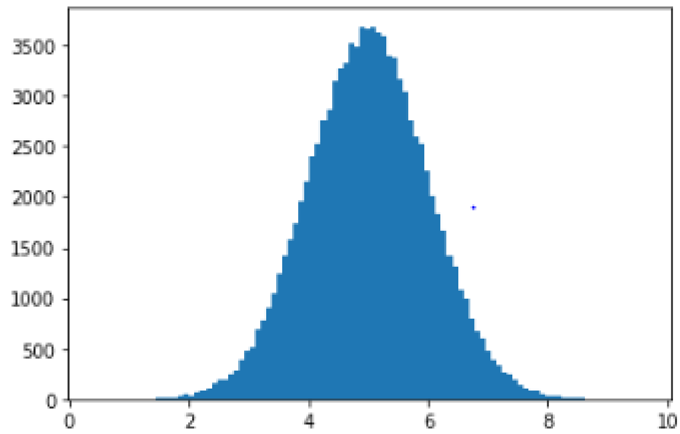
Histogram

```
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()
```



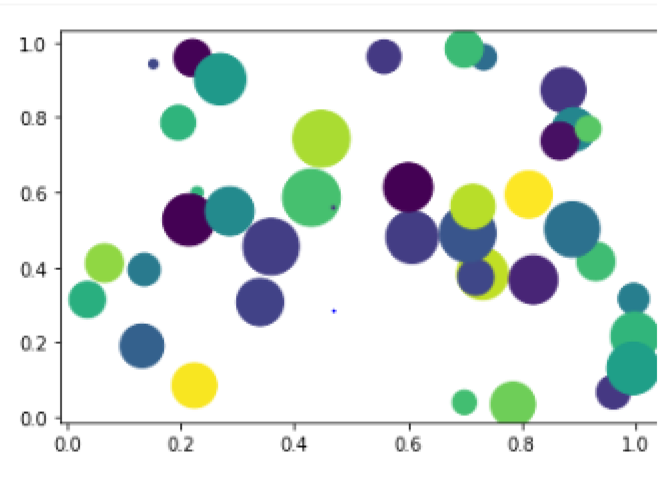
Distribution Chart

```
import numpy
import matplotlib.pyplot as plt

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

plt.hist(x, 100)
plt.show()
```



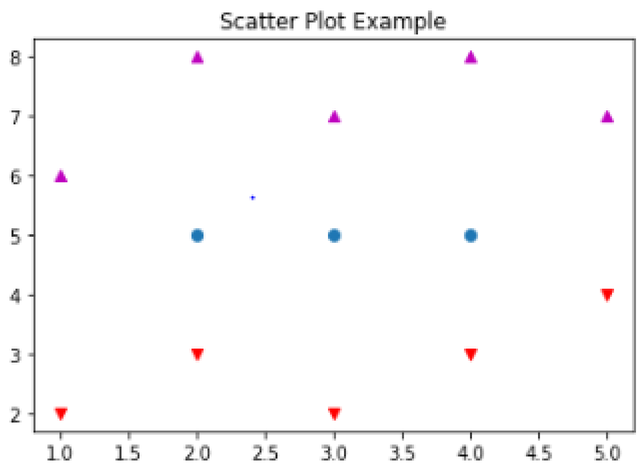
Bubble Chart

```
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

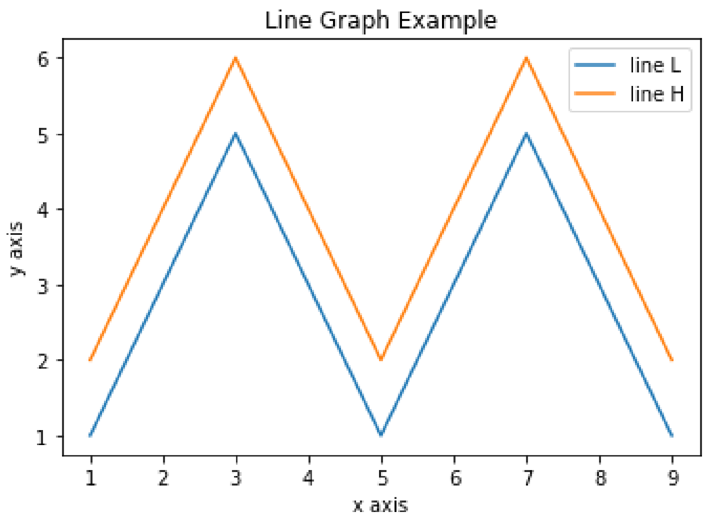
```
import matplotlib.pyplot as plt

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()
```

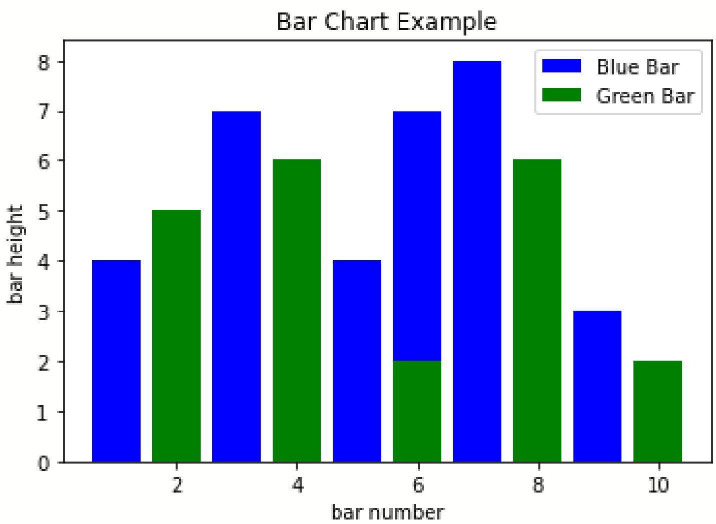
output



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 in
dicates 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.
```