LAB CYCLE 1

Experiment No:1
Date:12/12/21

Aim:

Review of python programming and Matrix operations

Data handling and Data Visualization

1. **Review of Python Programming and Matrix Operations**
   Create two 3X4 matrices a and b using numpy arrays. Perform the following operations: Display the number of dimensions of the matrices a and b, find the shape of the matrices aand b, find a+b and a-b, a*b(elementwise), multiply the matrix a and its transpose (matrix multiplicaiton), add the value 10 to all elements of a, find transpose of b,calculate average,mean and standard deviation of the elements of the matrix b, find the maximum element in each column and each row of the matrix a, minimum value of the matrix b, reshape b with dimension 2x6 and find the transpose of a.
   Create a row vector row_a, and a column vector col_a. Create the transpose ofcol_a, calculate the dot product of col_a with itself. Add the vectors row_a and col_a (broadcasting).
   Create a dictionary of data. Convert the dictionary to a feature matrix. Display the feature matrix and its column names.

2. **Data handling**
   You are given the dataset ecom.csv. It contains various properties of E-commerce transactions. Create a pandas dataframe for the dataset. Use appropriate functions to show the shape (number of feature vectors x number of features) of the dataset. Use appropriate slicing functions to show the head and tail ends of the dataset, to display the feature vector corresponding to the row number 180 and to display the set of tuples where mode of shipment is flight and weight is more than 7000 gms. Find the mean, median, mode and variance of the customer rating. Generate descriptive statistics of the numeric features in the dataset.

3. **Data Visualization**
   Create a dataframe from the python dictionary consisting of three attributes and values: age:[12, 14, 3,8, 7, 5,12, 18,21,19], height:[140, 150, 110, 130, 135, 120, 150, 170, 178, 180], and weight:[40, 50, 10, 30, 35, 20, 50, 70, 78, 80]. Draw a scatter plot with age and height on the x and y axis respectively. Draw a bubble plot with age and height on the x and y axis respectively. Draw a density plot for the attribute weight Draw a histogram for the attribute age. Draw a boxplot for the attribute height.

Source Code:

```
In [1]:
#1. Review of Python Programming and Matrix Operations

In [6]:
#Qn.1.1
import numpy as np

In [136…
#create two 3x4 matrices
a=np.array([[1,2,3,0],[0,1,2,1],[2,1,0,3]])
b=np.array([[2,1,1,0],[1,0,1,0],[3,0,2,0]])

In [137…
a

Out[137…
array([[1, 2, 3, 0],
       [0, 1, 2, 1],
       [2, 1, 0, 3]])

In [138…
b

Out[138…
array([[2, 1, 1, 0],
       [1, 0, 1, 0],
       [3, 0, 2, 0]])

In [141…
#Display the number of dimensions of the matrices a and b
print("Dimension of a: ",a.ndim)

Dimension of a:  2

In [144…
print("Dimension of b: ",b.ndim)

Dimension of b:  2

In [146…
#find the shape of the matrices a and b
print("Shape of matrix a: ",a.shape)

Shape of matrix a:  (3, 4)

In [147…
print("Shape of matrix b: ",b.shape)

Shape of matrix b:  (3, 4)

In [212…
#find a+b
add=np.add(a,b)
print("a+b = \n",add)

a+b =
 [[3 3 4 0]
 [1 1 3 1]
 [5 1 2 3]]

In [211…
#find a-b
diff=np.subtract(a,b)
print("a-b = \n",diff)
```

```
 =
 [[-1  1  2  0]
 [-1  1  1  1]
 [-1  1 -2  3]]
```

In [210…

```
#find a*b(elementwise)
print("a*b = \n",a*b)
```

```
a*b =
 [[2 2 3 0]
 [0 0 2 0]
 [6 0 0 0]]
```

In [209…

```
#multiply the matrix a and its transpose (matrix multiplicaiton)
trans_a=a.T
print("AxA' =\n",np.matmul(a,trans_a))
```

```
AxA' =
 [[14  8  4]
 [ 8  6  4]
 [ 4  4 14]]
```

In [208…

```
#add the value 10 to all elements of a
print("a+10 = \n",a+10)
```

```
a+10 =
 [[11 12 13 10]
 [10 11 12 11]
 [12 11 10 13]]
```

In [207…

```
#find transpose of b
trans_b=b.T
print("Transpose of b: \n",trans_b)
```

```
Transpose of b:
 [[2 1 3]
 [1 0 0]
 [1 1 2]
 [0 0 0]]
```

In [172…

```
#calculate average, mean and standard deviation of the elements of the matrix b
print("Average of matrix b: ",np.average(b))
```

```
Average of matrix b:  0.9166666666666666
```

In [173…

```
print("Mean of matrix b: ",np.mean(b))
```

```
Mean of matrix b:  0.9166666666666666
```

In [174…

```
print("Standard deviation of matrix b: ",np.std(b))
```

```
Standard deviation of matrix b:  0.9537935951882998
```

In [175…

```
#find the maximum element in each column of the matrix a
print("Maximum element in each column of matrix a: ",np.max(a,axis=0))
```

```
Maximum element in each column of matrix a:  [2 2 3 3]
```

In [177…

```
#find the maximum element in each row of the matrix a
```

```
print("Maximum element in each row of matrix a: ",np.max(a,axis=1))
```

```
Maximum element in each row of matrix a:  [3 2 3]
```

In [179…

```
#minimum value of the matrix b
print("Minimum value of matrix b: ",np.min(b))
```

```
Minimum value of matrix b:  0
```

In [206…

```
#reshape b with dimension 2x6
print("Matrix b reshaped with dimension 2x6: \n",b.reshape(2,6))
```

```
Matrix b reshaped with dimension 2x6: 
 [[2 1 1 0 1 0]
 [1 0 3 0 2 0]]
```

In [205…

```
#find the transpose of a
trans_a=a.T
print("Transpose of matrix a: \n",trans_a)
```

```
Transpose of matrix a: 
 [[1 0 2]
 [2 1 1]
 [3 2 0]
 [0 1 3]]
```

In [ ]:

```
#################################################################################
```

In [32]:

```
#Qn1.2
```

In [182…

```
#Create a row vector row_a, and a column vector col_a.
row_a=np.array([1,2,3])
print("Row vector row_a= ",row_a)
```

```
Row vector row_a=  [1 2 3]
```

In [204…

```
col_a=np.array([[1],[2],[3]])
 print("Column vector col_a= \n",col_a)
```

```
Column vector col_a= 
 [[1]
 [2]
 [3]]
```

In [184…

```
#Create transpose of col_a
trans_col_a=col_a.T
print("Transpose of col_a: ",trans_col_a)
```

```
Transpose of col_a:  [[1 2 3]]
```

In [188…

```
#calculate the dot product of col_a with itself.
print("Dot product of col_a with itself: ",np.dot(col_a,col_a))
```

In [189…

```
Dot product of row_a with itself:  14
```

In [203…

```
#Add the vectors row_aand col_a (broadcasting).
print("Sum of row_a and col_a: \n",np.add(row_a,col_a))
```

```
Sum of row_a and col_a:
 [[2 3 4]
 [3 4 5]
 [4 5 6]]
```

```
print("Dot product of row_a with itself: ",np.dot(row_a,row_a))
```

In [ ]:

```
###################################################################################
```

In [43]:

```
#Qn1.3
```

In [44]:

```
#Create a dictionary of data
```

In [4]:

```
from sklearn.feature_extraction import DictVectorizer
```

In [5]:

```
 data_dict = [{'Red': 2, 'Blue': 4},
              {'Red': 4, 'Blue': 3},
             {'Red': 1, 'Yellow': 2},
             {'Red': 2, 'Yellow': 2}]
```

In [6]:

```
# Create DictVectorizer object
dictvectorizer = DictVectorizer(sparse=False)
```

In [7]:

```
# Convert dictionary into feature matrix
features = dictvectorizer.fit_transform(data_dict)
```

In [202…

```
# View feature matrix
print("Feature Matrix: \n",features)
```

```
Feature Matrix:
 [[4. 2. 0.]
 [3. 4. 0.]
 [0. 1. 2.]
 [0. 2. 2.]]
```

In [192…

```
# View feature matrix column names
print("Column names: ",dictvectorizer.get_feature_names())
```

```
Column names:  ['Blue', 'Red', 'Yellow']
```

In [45]:

```
###############################################################################
```

In [46]:

```
#Qn_2.Data Handling
```

In [47]:

```
import pandas as pd
import matplotlib.pyplot as plt
```

In [195…

```
ecom=pd.read_csv("ecom.csv")
```

In [197…

```
#functions to show the shape (number of feature vectors x number of features) of the
print("Shape of dataset: ",ecom.shape)
```

```
Shape of dataset:  (10999, 12)
```

In [199…

```
#Use appropriate slicing functions to show the head and tail ends of the dataset
ecom.head()
```

Out[199…

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Prod
0	1	D	Flight	4	2	1
1	2	F	Flight	4	5	2
2	3	A	Flight	2	2	1
3	4	B	Flight	3	3	1
4	5	C	Flight	2	2	1

In [51]:

```
ecom.tail()
```

Out[51]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_t
10994	10995	A	Ship	4	1	
10995	10996	B	Ship	4	1	
10996	10997	C	Ship	5	4	
10997	10998	F	Ship	5	2	
10998	10999	D	Ship	2	5	

In [201…

```
#display the feature vector corresponding to the row number 180
print("feature vector corresponding to the row number 180 \n",ecom.iloc[180])
```

```
feature vector corresponding to the row number 180
```

```
 ID                        181
Warehouse_block              D
Mode_of_Shipment          Ship
Customer_care_calls          4
Customer_rating              1
Cost_of_the_Product        161
Prior_purchases              7
Product_importance      medium
Gender                       F
Discount_offered            18
Weight_in_gms             1294
Reached.on.Time_Y.N          1
Name: 180, dtype: object
```

In [54]:

```
#to display the set of tuples where mode of shipment is flight and weight is more th
ecom.loc[(ecom["Mode_of_Shipment"]=="flight") & (ecom["Weight_in_gms"]>7000)]
```

Out[54]:

ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Produ

In [213…

```
#Find the mean, median, mode and variance of the customer rating.
print("Mean value of Customer rating: ",ecom.Customer_rating.mean())
```

```
Mean value of Customer rating:  2.9905445949631786
```

In [214…

```
print("Median value of Customer rating: ",ecom.Customer_rating.median())
```

```
Median value of Customer rating:  3.0
```

In [216…

```
print("Mode of Customer rating: \n",ecom.Customer_rating.mode())
```

```
Mode of Customer rating: 
 0    3
dtype: int64
```

In [217…

```
print("Variance of Customer rating: ",ecom.Customer_rating.var())
```

```
Variance of Customer rating:  1.9982739259753057
```

In [219…

```
#Generate descriptive statistics of the numeric features in the dataset.
ecom.describe()
```

Out[219…

	ID	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Disc
count	10999.00000	10999.000000	10999.000000	10999.000000	10999.000000	
mean	5500.00000	4.054459	2.990545	210.196836	3.567597	
std	3175.28214	1.141490	1.413603	48.063272	1.522860	
min	1.00000	2.000000	1.000000	96.000000	2.000000	
25%	2750.50000	3.000000	2.000000	169.000000	3.000000	
50%	5500.00000	4.000000	3.000000	214.000000	3.000000	
75%	8249.50000	5.000000	4.000000	251.000000	4.000000	
max	10999.00000	7.000000	5.000000	310.000000	10.000000	

```
In [ ]:
```

```
In [60]:  ###################################################################################
```

```
In [115…  #Qn_3. Data Visualization
          import numpy as np
```

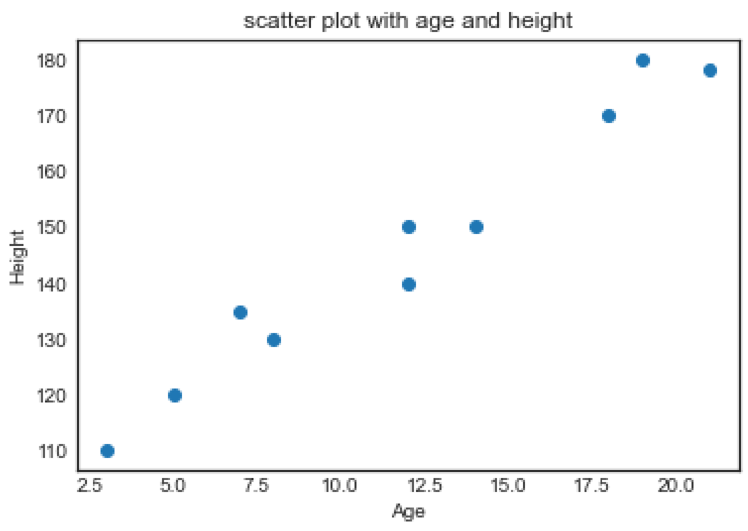
```
In [116…  data={'age':[12,14,3,8,7,5,12,18,21,19],'height':[140,150,110,130,135,120,150,170,17
          df=pd.DataFrame(data)
          df
```

Out[116…

	age	height	weight
0	12	140	40
1	14	150	50
2	3	110	10
3	8	130	30
4	7	135	35
5	5	120	20
6	12	150	50
7	18	170	70
8	21	178	78
9	19	180	80

```
In [165…  #Draw a scatter plot with age and height on the x and y axis respectively
          plt.scatter(df.age,df.height)
          plt.xlabel('Age')
          plt.ylabel('Height')
          plt.title("scatter plot with age and height")
```

```
Out[165…  Text(0.5, 1.0, 'scatter plot with age and height')
```

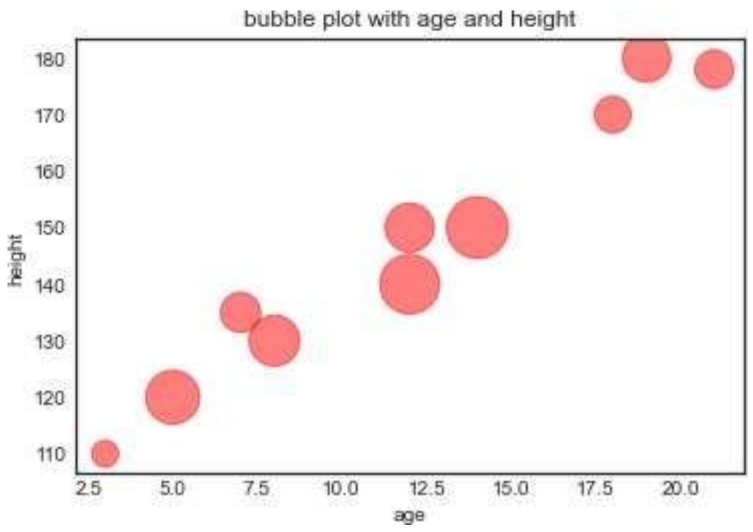


In [164…

```
#Draw a bubble plot with age and height on the x and y axis respectively.
z=np.random.rand(10)
print("bubble plot",df.plot.scatter('age','height',color='red',alpha=0.5,s=z*1000))
plt.title("bubble plot with age and height")
```

Out[164…

```
bubble plot AxesSubplot(0.125,0.125;0.775x0.755)
Text(0.5, 1.0, 'bubble plot with age and height')
```



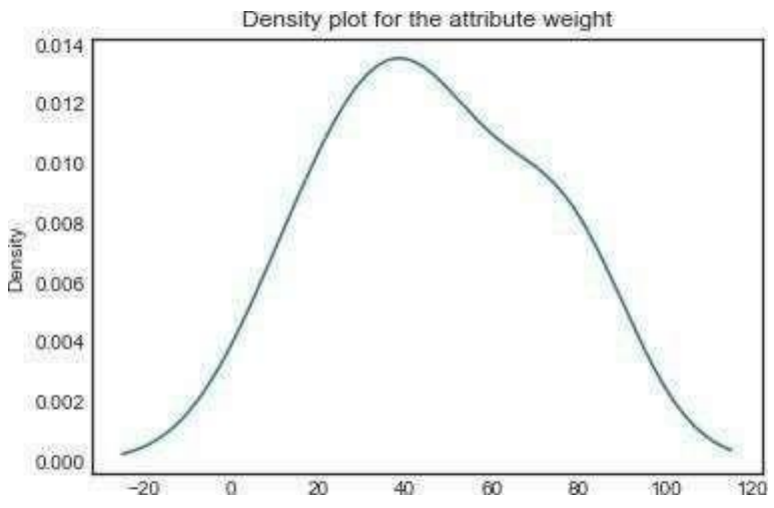
In [163…

```
#Draw a density plot for the attribute weight

print("Density plot for the attribute weight",df.weight.plot.density())
plt.title("Density plot for the attribute weight")
```

Out[163…

```
Density plot for the attribute weight AxesSubplot(0.125,0.125;0.775x0.755)
Text(0.5, 1.0, 'Density plot for the attribute weight')
```



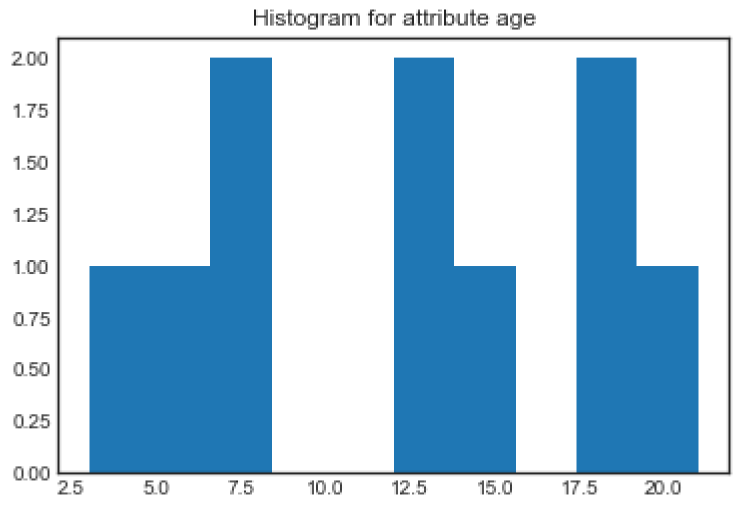
In [155…

```
#Draw a histogram for the attribute age

plt.hist(df.age)
plt.title("Histogram for attribute age")
```

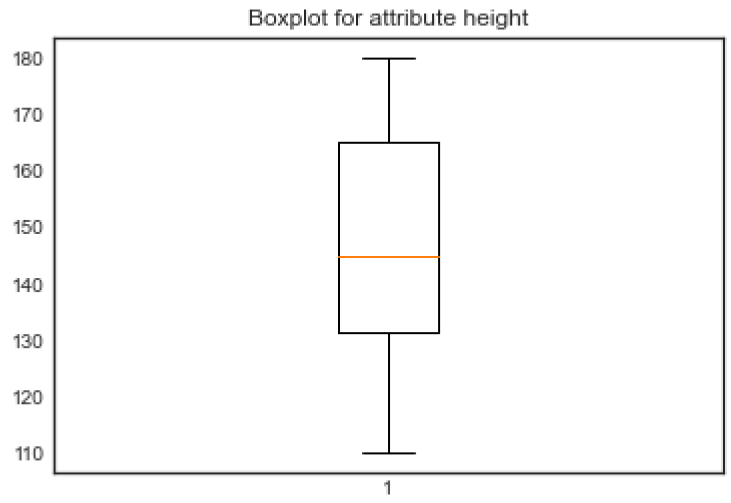
Out[155…

```
Text(0.5, 1.0, 'Histogram for attribute age')
```



```
In [154…
#Draw a boxplot for the attribute height.
plt.boxplot(df.height)
plt.title("Boxplot for attribute height")
```

```
Out[154… Text(0.5, 1.0, 'Boxplot for attribute height')
```



```
In [ ]:
```

```
In [ ]:
```