Q1. Write a Python program to display the current date and time.
Source Code:
```
import datetime as dt
now=dt.datetime.now()
print("Current date & time : "+now.strftime("%Y-%m-%d %H:%M:%S"))
```
Output: Current date & time : 2023-04-09 06:52:55

Q2. Write a Python program that calculates the area of a circle based on the radius entered by the user
Source Code:
```
from math import pi
r=float(input("Enter the radius: "))
a=pi*r**2
ans=round(a,2)
print("Area of the circle is: "+str(a))
print("Area of the circle is: "+str(ans))
```
Output:
Enter the radius: 4
Area of the circle is: 50.26548245743669
Area of the circle is: 50.27

Q3. Write a Python program that accepts the user's first and last name and prints them in reverse order with a space between them.
Source Code:
```
fname = input("Input your First Name : ")
lname = input("Input your Last Name : ")
print ("Hello " + lname + " " + fname)
```
Output:
Input your First Name : Saurab
Input your Last Name : Negi
Hello Negi Saurab

Q4. Write a Python program to display the first and last colors from the following list.
Source Code:
```
color_list = ["Red","Green","White" ,"Black"]
color_list = ["Red","Green","White" ,"Black"]
print( "%s %s"%(color_list[0],color_list[-1]))
```
Output: Red Black

Q5. Write a Python program that accepts an integer (n) and computes the value of n+nn+nnn.
Source Code:
```
a = int(input("Input an integer : "))
n1 = int( "%s" % a )
n2 = int( "%s%s" % (a,a) )
n3 = int( "%s%s%s" % (a,a,a) )
print (n1+n2+n3)
```
Output:
Input an integer : 4
492

Q6. Create a Numpy array object.

Source Code:

```
arr = np.array([1, 2, 3, 4, 5])
print(arr)
print(type(arr))
import numpy as np
```

Output:

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

Q7. Create a 2-D array containing two arrays with the values 1,2,3 and 4,5,6.

Source Code:

```
import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6]])
print(arr)
```

Output:

```
[[1 2 3]
[4 5 6]]
```

Q8.Create a 3-D array with two 2-D arrays, both containing two arrays with the values 1,2,3 and 4,5,6.

Source Code:

```
import numpy as np
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
print(arr)
```

Output:

```
[[[1 2 3]
[4 5 6]]
[[1 2 3]
[4 5 6]]]
```

Q9. Check the dimensions of the arrays.

Source Code:

```
import numpy as np
a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```

Output:

```
0
1
2
3
```

Q10. Access the element on the first row, second column.

Source Code:

```
import numpy as np
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print('2nd element on 1st row: ', arr[0, 1])
```

Output: 2nd element on 1st row: 2

Q11. Access the element on the 2nd row, 5th column.
Source Code:
```
import numpy as np
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print('5th element on 2nd row: ', arr[1, 4])
```
Output:
5th element on 2nd row: 10

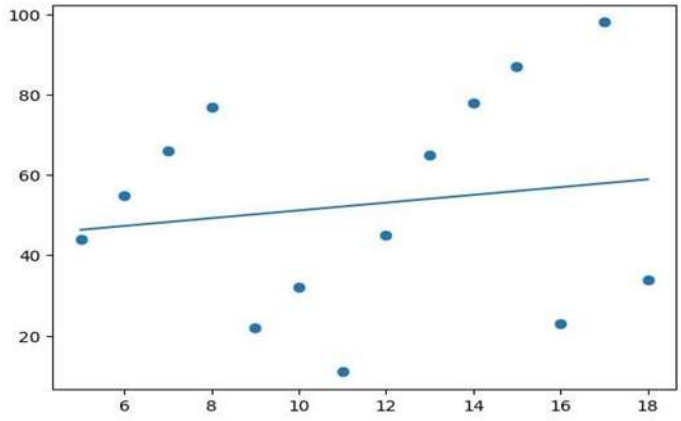
Q12. Descriptive Analysis.
Source Code:
```
import scipy as sp
import numpy as np
nums=np.random.randint(1,20,size=(1,18))[0]
print("Data :", nums)
print("Get Descriptive Statistics")
print("Mean :",np.mean(nums))
print("Median :",np.median(nums))
from scipy import stats, optimize, interpolate
print("Mode :",sp.stats.mode(nums,keepdims=True))
print('Standard Deviation :',np.std(nums))
print('Variance :',np.var(nums))
print('Skew :',sp.stats.skew(nums))
print('Kurtosis :',sp.stats.kurtosis(nums))
```
Output:
```
Data : [11 7 12 13 17 16 17 9 2 13 3 8 7 7 18 18 18 16]
get descriptive stats
Mean : 11.777777777777779
Median : 12.5
Mode : ModeResult(mode=array([7]), count=array([3]))
Standard Deviation : 5.126787553359701
Variance : 26.28395061728395
Skew : -0.35142484023194304
Kurtosis : -1.107948765410717
```

Q13. Write a program for linear regression for a given dataset & predict the value of y when n=10.
Source Code:
```
from scipy import stats
x=[5,6,7,8,9,10,11,12,13,14,15,16,17,18]
y=[44,55,66,77,22,32,11,45,65,78,87,23,98,34]
slope, intercept, r, p, std_err=stats.linregress(x,y)
def myfunc(x):
return slope*x +intercept
s=myfunc(10)
print(s)
```
Output:
51.1956043956044

Q14. Write a program for visualizing the slope on given input using matlplotlib.
Source Code:

```
import matplotlib.pyplot as plt
mymodel=list(map(myfunc,x))
plt.scatter(x,y)
plt.plot(x,mymodel)
plt.show()
```

Output:



Q15. Write a program for linear regression & print the coefficient for the calculated slope.
Source Code:

```
from sklearn.linear_model import LinearRegression
a = [[1], [2], [3], [4], [5]]
b = [[2], [4], [5], [4], [5]]
reg = LinearRegression()
# Train the model using the input data
reg.fit(a, b)
# Print the coef icient
print(reg.coef_)
```

Output: [[0.6]]

Q16. Write a program to print training and test data from datasets.
Source Code:

```
from sklearn.linear_model import Lasso
from sklearn import datasets
import numpy as np
diabetes= datasets.load_diabetes()
X_train=diabetes.data[:310]
Y_train=diabetes.target[:310]
X_test=diabetes.data[310:]
Y_test=diabetes.target[310:]
print(X_train)
print(Y_train)
print(X_test)
print(Y_test)
```

Output:

```
[[ 0.03807591  0.05068012  0.06169621 ... -0.00259226  0.01990749
  -0.01764613]
 [-0.00188202 -0.04464164 -0.05147406 ... -0.03949338 -0.06833155
  -0.09220405]
 [ 0.08529891  0.05068012  0.04445121 ... -0.00259226  0.00286131
  -0.02593034]
 ...
 [ 0.06713621  0.05068012 -0.03099563 ...  0.03430886  0.02337142
   0.08176444]
 [ 0.00175052 -0.04464164 -0.046085   ... -0.06938329 -0.0611758
  -0.07977773]
 [-0.00914709  0.05068012  0.00133873 ... -0.00259226  0.02671684
   0.08176444]]
[151.  75. 141. 206. 135.  97. 138.  63. 110. 310. 101.  69. 179. 185.
 118. 171. 166. 144.  97. 168.  68.  49.  68. 245. 184. 202. 137.  85.
 131. 283. 129.  59. 341.  87.  65. 102. 265. 276. 252.  90. 100.  55.
  61.  92. 259.  53. 190. 142.  75. 142. 155. 225.  59. 104. 182. 128.
  52.  37. 170. 170.  61. 144.  52. 128.  71. 163. 150.  97. 160. 178.
  48. 270. 202. 111.  85.  42. 170. 200. 252. 113. 143.  51.  52. 210.
  65. 141.  55. 134.  42. 111.  98. 164.  48.  96.  90. 162. 150. 279.
  92.  83. 128. 102. 302. 198.  95.  53. 134. 144. 232.  81. 104.  59.
 246. 297. 258. 229. 275. 281. 179. 200. 200. 173. 180.  84. 121. 161.
  99. 109. 115. 268. 274. 158. 107.  83. 103. 272.  85. 280. 336. 281.
 118. 317. 235.  60. 174. 259. 178. 128.  96. 126. 288.  88. 292.  71.
 197. 186.  25.  84.  96. 195.  53. 217. 172. 131. 214.  59.  70. 220.
 268. 152.  47.  74. 295. 101. 151. 127. 237. 225.  81. 151. 107.  64.
 138. 185. 265. 101. 137. 143. 141.  79. 292. 178.  91. 116.  86. 122.
  72. 129. 142.  90. 158.  39. 196. 222. 277.  99. 196. 202. 155.  77.

 191.  70.  73.  49.  65. 263. 248. 296. 214. 185.  78.  93. 252. 150.
  77. 208.  77. 108. 160.  53. 220. 154. 259.  90. 246. 124.  67.  72.
 257. 262. 275. 177.  71.  47. 187. 125.  78.  51. 258. 215. 303. 243.
  91. 150. 310. 153. 346.  63.  89.  50.  39. 103. 308. 116. 145.  74.
  45. 115. 264.  87. 202. 127. 182. 241.  66.  94. 283.  64. 102. 200.
 265.  94. 230. 181. 156. 233.  60. 219.  80.  68. 332. 248.  84. 200.
  55.  85.  89.  31. 129.  83. 275.  65. 198. 236. 253. 124.  44. 172.
 114. 142.]
[[-0.00551455 -0.04464164  0.06492964 ...  0.00072884 -0.01811369
   0.03205916]
 [ 0.09619652 -0.04464164  0.04013997 ...  0.03615391  0.01255119
   0.02377494]
 [-0.07453279 -0.04464164 -0.02345095 ... -0.03949338 -0.03845972
  -0.03007245]
 ...
 [ 0.04170844  0.05068012 -0.01590626 ... -0.01107952 -0.04688253
   0.01549073]
 [-0.04547248 -0.04464164  0.03906215 ...  0.02655962  0.04452873
  -0.02593034]
 [-0.04547248 -0.04464164 -0.0730303  ... -0.03949338 -0.00422151
   0.00306441]]
[109. 180. 144. 163. 147.  97. 220. 190. 109. 191. 122. 230. 242. 248.
 249. 192. 131. 237.  78. 135. 244. 199. 270. 164.  72.  96. 306.  91.
 214.  95. 216. 263. 178. 113. 200. 139. 139.  88. 148.  88. 243.  71.
  77. 109. 272.  60.  54. 221.  90. 311. 281. 182. 321.  58. 262. 206.
 233. 242. 123. 167.  63. 197.  71. 168. 140. 217. 121. 235. 245.  40.
  52. 104. 132.  88.  69. 219.  72. 201. 110.  51. 277.  63. 118.  69.
 273. 258.  43. 198. 242. 232. 175.  93. 168. 275. 293. 281.  72. 140.
 189. 181. 209. 136. 261. 113. 131. 174. 257.  55.  84.  42. 146. 212.
 233.  91. 111. 152. 120.  67. 310.  94. 183.  66. 173.  72.  49.  64.
  48. 178. 104. 132. 220.  57.]
```

Q17. Write a Python Program to implement linear regression on the iris dataset.

Source Code:

```
# Import Dataset from sklearn
from sklearn.datasets import load_iris
# Load Iris Data
iris = load_iris()
import numpy as np
import pandas as pd
# Creating pd DataFrames
iris_df = pd.DataFrame(data= iris.data, columns= iris.feature_names)
target_df = pd.DataFrame(data= iris.target, columns= ['species'])
def converter(specie):
if specie == 0:
return 'setosa' elif specie == 1:
return 'versicolor' else:
return 'virginica' target_df['species'] = target_df['species'].apply(converter)
# Concatenate the DataFrames
iris_df = pd.concat([iris_df, target_df], axis= 1)
iris_df.describe()
iris_df.info()
import seaborn as sns
sns.pairplot(iris_df, hue= 'species')
# Converting Objects to Numerical dtype
iris_df.drop('species', axis= 1, inplace= True)
target_df = pd.DataFrame(columns= ['species'], data= iris.target)
iris_df = pd.concat([iris_df, target_df], axis= 1)
iris_df.head()
# Variables
X= iris_df.drop(labels= 'sepal length (cm)', axis= 1)
y= iris_df['sepal length (cm)']
from sklearn.model_selection import train_test_split
# Splitting the Dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.33, random_state= 101)
from sklearn import linear_model
# Instantiating LinearRegression() Model
lr = linear_model.LinearRegression()
# Training/Fitting the Model
lr.fit(X_train, y_train)
# Making Predictions
lr.predict(X_test)
pred = lr.predict(X_test)
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
# Evaluating Model's Performance
iris_df.loc[5]
d = {'sepal length (cm)' : [5.4],
'sepal width (cm)' : [3.9],
'petal length (cm)' : [1.7],
'petal width (cm)' : [0.4],
'species' : 0}
test_df = pd.DataFrame(data= d)
test_df
pred = lr.predict(X_test)
```

print('Predicted Sepal Length (cm):', pred[0])
print('Actual Sepal Length (cm):', 5.4)

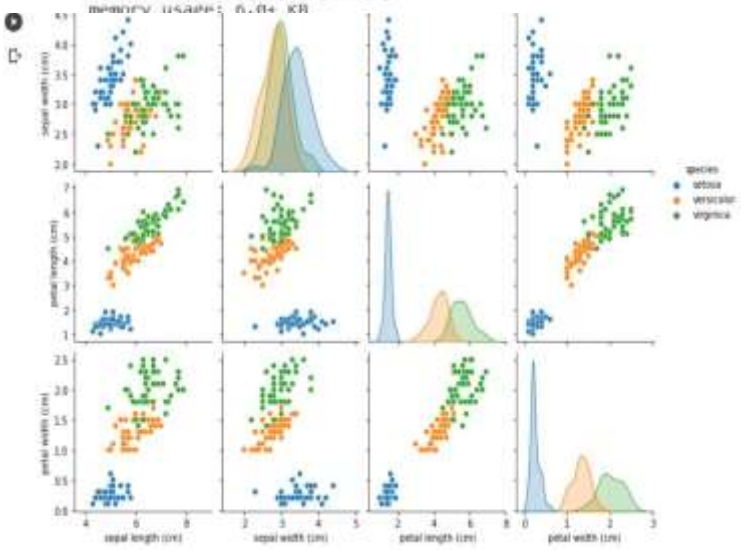
Output:

[5]

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
iris_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   species            150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```



	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

```
[21] d = {'sepal length (cm)' : [5.4],
         'sepal width (cm)' : [3.9],
         'petal length (cm)' : [1.7],
         'petal width (cm)' : [0.4],
         'species' : 0}
     test_df = pd.DataFrame(data= d)
     test_df
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.4	3.9	1.7	0.4	0

```
pred = lr.predict(X_test)
print('Predicted Sepal Length (cm):', pred[0])
print('Actual Sepal Length (cm):', 5.4)
```

```
Predicted Sepal Length (cm): 5.461145872156033
Actual Sepal Length (cm): 5.4
```

Q18. Write a Python Program to implement the decision trees on the iris dataset.
Source Code:

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
# Reading the Iris.csv file
data = load_iris()
# Extracting Attributes / Features
X = data.data
# Extracting Target / Class Labels
y = data.target
# Import Library for splitting data
from sklearn.model_selection import train_test_split
# Creating Train and Test datasets
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state = 50, test_size = 0.25)
# Creating Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf.fit(X_train,y_train)
# Predict Accuracy Score
y_pred = clf.predict(X_test)
print("Train data accuracy:",accuracy_score(y_true = y_train, y_pred=clf.predict(X_train)))
print("Test data accuracy:",accuracy_score(y_true = y_test, y_pred=y_pred))
```

Output:

```
DecisionTreeClassifier()
```

```
[28] # Predict Accuracy Score
     y_pred = clf.predict(X_test)
     print("Train data accuracy:",accuracy_score(y_true = y_train, y_pred=clf.predict(X_train)))
     print("Test data accuracy:",accuracy_score(y_true = y_test, y_pred=y_pred))
```

```
Train data accuracy: 0.7626666666666667
Test data accuracy: 0.48
```

Q19.Write a Python Program to implement the decision trees on the loan dataset.
Source Code:

```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score
data = pd.read_csv("/content/Loan payments data.csv")
data['effective_date'] = pd.to_datetime(data['effective_date'])
data['dayofweek'] = data['effective_date'].dt.dayofweek
data['weekend'] = data['dayofweek'].apply(lambda x: 1 if (x>3) else 0)
from sklearn import preprocessing
# Extracting Attributes / Features
data['Gender'].replace(to_replace=['male','female'], value=[0,1],inplace=True)
X = data[['Principal','terms','age','Gender','weekend']]
X= preprocessing.StandardScaler().fit(X).transform(X)
# Extracting Target / Class Labels
y = data['loan_status'].values
# Import Library for splitting data
from sklearn.model_selection import train_test_split
# Creating Train and Test datasets
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state = 50, test_size = 0.25)
# Creating Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf.fit(X_train,y_train)
# Predict Accuracy Score
y_pred = clf.predict(X_test)
print("Train data accuracy:",accuracy_score(y_true = y_train, y_pred=clf.predict(X_train)))
print("Test data accuracy:",accuracy_score(y_true = y_test, y_pred=y_pred))
```

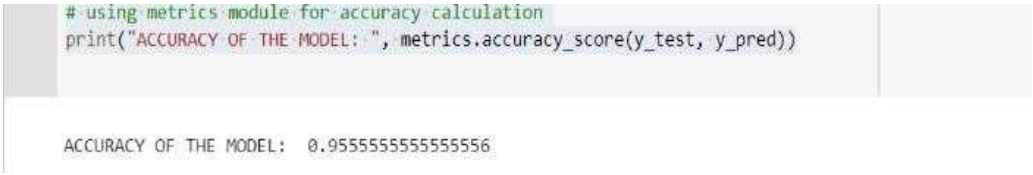
Output:

Q20.Write a Python Program to implement the random forest
Source Code

```
# importing required libraries
# importing Scikit-learn library and datasets package
from sklearn import datasets
# Loading the iris plants dataset (classification)
iris = datasets.load_iris()
print(iris.target_names)
print(iris.feature_names)
# dividing the datasets into two parts i.e. training datasets and test datasets
X, y = datasets.load_iris( return_X_y = True)
# Splitting arrays or matrices into random train and test subsets
from sklearn.model_selection import train_test_split
```

```
# i.e. 70 % training dataset and 30 % test datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30)
# importing random forest classifier from assemble module
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
# creating dataframe of IRIS dataset
data = pd.DataFrame({'sepallength': iris.data[:, 0], 'sepalwidth': iris.data[:, 1],'petallength': iris.da
ta[:, 2], 'petalwidth': iris.data[:, 3],'species': iris.target})
# creating a RF classifier
clf = RandomForestClassifier(n_estimators = 100)
# Training the model on the training dataset
# fit function is used to train the model using the training sets as parameters
clf.fit(X_train, y_train)
# performing predictions on the test dataset
y_pred = clf.predict(X_test)
# metrics are used to find accuracy or error
from sklearn import metrics
print()
# using metrics module for accuracy calculation
print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_test, y_pred))
```

Output:



Q21.Write a python code to classify the given dataset using support vector machine.
Source Code:

```
import pandas as pd
df=pd.read_csv(r'C:\Users\Omen\Desktop\printout\iris.csv')
x=df.iloc[:,0:4]
y=df.iloc[:,4]
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)
from sklearn.svm import SVC
model=SVC(kernel='linear')
model.fit(x_train,y_train)
model.predict(x_test)
```

Output:

```
array(['Versicolor', 'Setosa', 'Setosa', 'Virginica', 'Setosa', 'Setosa',
       'Virginica', 'Setosa', 'Setosa', 'Virginica', 'Versicolor',
       'Virginica', 'Virginica', 'Virginica', 'Virginica', 'Versicolor',
       'Virginica', 'Virginica', 'Setosa', 'Versicolor', 'Virginica',
       'Versicolor', 'Setosa', 'Setosa', 'Versicolor', 'Setosa',
       'Virginica', 'Virginica', 'Virginica', 'Virginica'], dtype=object)
```

Q22. Write a python program to design a chatbot with a static database to interact with the patients.

Source Code:

```
import random
greetings = ["Hello!", "Hi there!", "Welcome!", "Greetings!"]
common_questions = ["What is your name?", "How can I help you today?", "What symptoms are you experiencing?", "Do you have any allergies?", "Are you currently taking any medications?",]
responses = ["I'm sorry, I'm just a chatbot and cannot provide medical advice. It's best to consult with a healthcare professional.", "Please consult with a doctor for proper diagnosis and treatment.", "It's important to seek medical attention for your condition.", "I recommend reaching out to a healthcare professional to discuss your concerns.", ]
def get_random_greeting():
return random.choice(greetings)
# Function to respond to user input
def respond(user_input):
if user_input.endswith("?"):
return random.choice(responses)
else:
return random.choice(common_questions)
# Main chat loop
def chat():
print(get_random_greeting())
while True:
user_input = input(">")
if user_input.lower() == "exit":
break
print(respond(user_input))
chat()
```

Output:

```
Welcome!
>HELLO
Do you have any allergies?
>YES
Do you have any allergies?
>hello
Are you currently taking any medications?
```

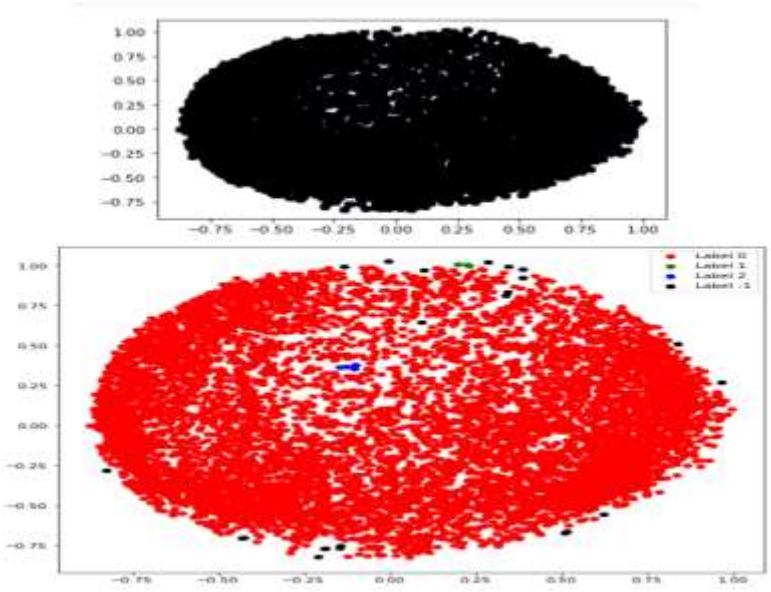
Q23. Write a Python Program to implement the DBScan for recommendation system.

Source Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import normalize
from sklearn.decomposition import PCA
X = pd.read_csv('/content/CC GENERAL.csv')
# Dropping the CUST_ID column from the data
X = X.drop('CUST_ID', axis = 1)
# Handling the missing values
X.fillna(method ='ffill', inplace = True)
# Scaling the data to bring all the attributes to a comparable level
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Normalizing the data so that
# the data approximately follows a Gaussian distribution
X_normalized = normalize(X_scaled)
# Converting the numpy array into a pandas DataFrame
X_normalized = pd.DataFrame(X_normalized)
pca = PCA(n_components = 2)
X_principal = pca.fit_transform(X_normalized)
X_principal = pd.DataFrame(X_principal)
X_principal.columns = ['P1', 'P2']
print(X_principal.head())
# Numpy array of all the cluster labels assigned to each data point
db_default = DBSCAN(eps = 0.0375, min_samples = 3).fit(X_principal)
labels = db_default.labels_
# Building the label to colour mapping
colours = {}
colours[0] = 'r' colours[1] = 'g' colours[2] = 'b' colours[-1] = 'k' # Building the colour vector for each data point
cvec = [colours[label] for label in labels]
# For the construction of the legend of the plot
r = plt.scatter(X_principal['P1'], X_principal['P2'], color ='r');
g = plt.scatter(X_principal['P1'], X_principal['P2'], color ='g');
b = plt.scatter(X_principal['P1'], X_principal['P2'], color ='b');
k = plt.scatter(X_principal['P1'], X_principal['P2'], color ='k');
# Plotting P1 on the X-Axis and P2 on the Y-Axis
# according to the colour vector defined
plt.figure(figsize =(9, 9))
plt.scatter(X_principal['P1'], X_principal['P2'], c = cvec)
# Building the legend
plt.legend((r, g, b, k), ('Label 0', 'Label 1', 'Label 2', 'Label -1'))
plt.show()
db = DBSCAN(eps = 0.0375, min_samples = 50).fit(X_principal)
labels1 = db.labels_
colours1 = {}
colours1[0] = 'r' colours1[1] = 'g' colours1[2] = 'b' colours1[3] = 'c' colours1[4] = 'y' colours1[5] = 'm' colours1[-1] =
'k' cvec = [colours1[label] for label in labels]
colors = ['r', 'g', 'b', 'c', 'y', 'm', 'k' ]
r = plt.scatter(
```

```
X_principal['P1'], X_principal['P2'], marker ='o', color = colors[0])
g = plt.scatter(
X_principal['P1'], X_principal['P2'], marker ='o', color = colors[1])
b = plt.scatter(
X_principal['P1'], X_principal['P2'], marker ='o', color = colors[2])
c = plt.scatter(
X_principal['P1'], X_principal['P2'], marker ='o', color = colors[3])
y = plt.scatter(
X_principal['P1'], X_principal['P2'], marker ='o', color = colors[4])
m = plt.scatter(
X_principal['P1'], X_principal['P2'], marker ='o', color = colors[5])
k = plt.scatter(
X_principal['P1'], X_principal['P2'], marker ='o', color = colors[6])
plt.figure(figsize =(9, 9))
plt.scatter(X_principal['P1'], X_principal['P2'], c = cvec)
plt.legend((r, g, b, c, y, m, k), ('Label 0', 'Label 1', 'Label 2', 'Label 3', 'Label 4',
'Label 5', 'Label -1'), scatterpoints = 1, loc ='upper left', ncol = 3, fontsize = 8)
plt.show()
```

Output :



Q. Write a python program to read a CSV file and print the shape of the file. import pandas as pd
data=pd.read_csv("/content/drive/MyDrive/csvfiles/credit_risk.csv") data.shape

Output:

```
(1000, 21)
```

Q. Divide the given dataset in four necessary components for training and testing the ML model. Define the splitting.

import pandas as pd
from
sklearn.model_selection
import train_test_split
data=pd.read_csv("//content/drive/MyDrive/csvfiles/computers.csv")
x_train,x_test,y_train,y_test=train_test_split(data["Units"],data["Minutes"],test_size=0.2,ran dom_state=0)
x_train.shape, x_test.shape, y_train.shape, y_test.shape]

X_train=X_train/255
X_test=X_test/255
X_train[0]

Output:

```
((11,), (3,), (11,), (3,))
```

Q. Write a python code to convert images of the dataset into normalize dataset print the normalize data for the first image of the dataset.
import tensorflow as tf from tensorflow
import keras import matplotlib.pyplot as plt %matplotlib inline
import numpy as np

(X_train,y_train), (X_test,y_test)=keras.datasets.mnist.load_data()

X_train=X_train/255
X_test=X_test/255
X_train[0]

Output:

```
array([[0.        , 0.        , 0.        , 0.        , 0.
        0.        , 0.        , 0.        , 0.        , 0.
        0.        , 0.        , 0.        , 0.        , 0.
        0.        , 0.        , 0.        , 0.        , 0.
```

```
  0.        , 0.        , 0.01176471, 0.07058824, 0.07058824,
  0.07058824, 0.49411765, 0.53333333, 0.68627451, 0.10196078,
  0.65098039, 1.        , 0.96862745, 0.49803922, 0.        ,
  0.        , 0.        , 0.        ],
 [0.        , 0.        , 0.        , 0.        , 0.        ,
```

Q.- Write a python code to convert the normalize data of the images of the given dataset into a flattened vector.

```
import tensorflow as tf from tensorflow
import keras
import   matplotlib.pyplot as plt
%matplotlib inline
import numpy as np

(X_train,y_train), (X_test,y_test)=keras.datasets.mnist.load_data()

X_train=X_train/255
X_test=X_test/255

X_train_flattened=X_train.reshape(len(X_train),28*28)
X_test_flattened=X_test.reshape(len(X_test),28*28) X_train_flattened.shape
```

```
(60000, 784)
```

X_train_flattened[0]

Output:

```
array([  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   3,  18,  18,  18,
       126, 136, 175,  26, 166, 255, 247, 127,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170, 253,
       253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253,
       253, 253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,
```

Q- Write a python a python code to create a neural network for the classification of MNIST dataset.

```
model=keras.Sequential([
keras.layers.Dense(10, input_shape=(784,), activation='sigmoid')])   model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
model.fit(X_train_flattened, y_train, epochs=5) model.evaluate(X_test_flattened,
y_test)
```

```
Epoch 1/5
1875/1875 [==============================] - 4s 2ms/step - loss: 2.1879 - accuracy: 0.5917
Epoch 2/5
1875/1875 [==============================] - 4s 2ms/step - loss: 1.9767 - accuracy: 0.7074
Epoch 3/5
1875/1875 [==============================] - 4s 2ms/step - loss: 1.7901 - accuracy: 0.7372
```

model.evaluate(X_test_flattened, y_test)

```
313/313 [==============================] - 1s 1ms/step - loss: 30.1229 - accuracy: 0.6946
[30.122894287109375, 0.694599986076355]
```

y_predicted=model.predict(X_test_flattened) y_predicted_labels=[np.argmax(i) for i in y_predicted] cm=tf.math.confusion_matrix(labels=y_test, predictions=y_predicted_labels) cm

```
313/313 [==============================] - 1s 2ms/step
<tf.Tensor: shape=(10, 10), dtype=int32, numpy=
array([[974,   0,   2,   4,   0,   0,   0,   0,   0,   0],
       [  3, 917, 102,  61,   0,   0,  13,   0,  39,   0],
       [420,   0, 585,   4,   7,   0,   9,   4,   3,   0],
       [522,   0, 141, 332,   2,   0,   0,   8,   5,   0],
       [190,   0,  79,  48, 581,   0,  32,   4,  48,   0],
       [740,   1,  25,  37,   4,  30,   4,   3,  48,   0],
       [544,   2, 322,   8,   1,   1,  80,   0,   0,   0],
       [190,   5,  92,  90,  54,   0,   3, 572,  22,   0],
       [260,   0, 127, 261,   7,   4,  12,  11, 292,   0],
       [156,   1,  22,  89, 447,   3,   2, 133, 151,   5]],
```

Q.- Write a python program to classify the given dataset using decision tree classification.

from sklearn.tree import DecisionTreeClassifier from sklearn.model_selection import train_test_split from sklearn import metrics
import pandas as pd

data=pd.read_csv("credit_risk.csv") data.head()

	over_draft	credit_usage	credit_history	purpose	current_balance	Average_Credit_Balance	employment	location	personal_status	o
0	<0	6	critical/other existing credit	radio/tv	1169	no known savings	>=7	4	male single	
1	0<=X<200	48	existing paid	radio/tv	5951	<100	1<=X<4	2	female div/dep/mar	

X=data.columns.drop('class')
Y=data['class']
X=pd.get_dummies(data[X])
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=0)

```
clf = DecisionTreeClassifier() clf.fit(X_train,y_train)

y_pred = clf.predict(X_test)

train_acc = clf.score(X_train,y_train) test_acc =
clf.score(X_test,y_test) print("Train Accuracy: ",train_acc,"\nTest
Accuracy: ",test_acc)
```

```
Train Accuracy:  1.0
Test Accuracy:  0.7
```

Q- Write a python program to classify the Given CSV file using SVM classifier.

```
from sklearn import metrics from
sklearn.svm import SVC
from sklearn.model_selection import train_test_split import
pandas as pd

data=pd.read_csv("iris.csv")
data.Species.unique()
```

```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

```
data.head()
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa

```
X=data.columns.drop('Species')
Y=data['Species']
X=pd.get_dummies(data[X])
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.3)


clf=SVC(kernel='linear') clf.fit(X_train,y_train)
y_predict=clf.predict(X_test)

print("Accuracy:",metrics.accuracy_score(y_test,y_predict))
```

```
Accuracy: 0.9555555555555556
```

Q.- Write a python program to classify the given dataset using logistic regression.

```
import pandas as pd from sklearn.linear_model import LogisticRegression from sklearn.model_selection import
train_test_split

data=pd.read_csv("bank-additional-full.csv")
X=data[["age","duration","campaign"]]
Y=data["y"]
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.30,random_state=0)
print(x_train.shape) print(y_train.shape) print(x_test.shape) print(y_test.shape)

model=LogisticRegression()
# Model Fitting model.fit(x_train,y_train) # Checking accuracy
print("Model Train Score: ",model.score(x_train,y_train)) print("Model
Test Score: ",model.score(x_test,y_test))
```

Output:

```
(28831, 3)
(28831,)
(12357, 3)
(12357,)
Model Train Score:  0.8927543269397523
Model Test Score:  0.8937444363518653
```

Q- Write a python program to Design a chatbot with a static database to interact with the patients.

```
db={
    "Hello": "Hello, how can I help you?",
    "How are you?": "I'm am good. How are you?",
    "Services": "Schedule appointment, Billing, Medication. What service you like?",
"Schedule appointment": "Appointent scheduled for 1pm, Aug 19,2023 (ID: 957395)",
    "Billing": "Your billing amount (including GST) is: $1250.00",
    "Medication": "Fetching details..."    } def chatbot():
print("Chatbot: Hi. How can I assist you today?")
while True:
        user_input = input("User: ")          if
user_input in db:
print("Chatbot:",
db[user_input])
else:           break

chatbot()
```

```
Chatbot: Hi. How can I assist you today?
User: Hello
Chatbot: Hello, how can I help you?
User: Services
Chatbot: Schedule appointment, Billing, Medication. What service you like?
User: Schedule appointment
Chatbot: Appointent scheduled for 1pm, Aug 19,2023 (ID: 957395)
User: bye
```

Q.- Write a program to classify the fashion MNIST dataset using neural networks.

```
import tensorflow
as tf from
tensorflow import
keras import
matplotlib.pyplot as plt
%matplotlib inline import
numpy as np
(X_train, y_train) , (X_test, y_test) =
keras.datasets.fashion_mnist.load_data()

X_train.shape, y_train.shape, X_test.shape,y_test.shape
len(X_train) len(X_test)
X_train[0].shape X_train[0]
X_train=X_train/255
X_test=X_test/255
X_train_flattened=X_train.reshape(len(X_train),28*28) X_test_flattened=X_test.reshape(len(X_test),28*28)
model=keras.Sequential([ keras.layers.Dense(10, input_shape=(784,), activation='sigmoid')])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
model.fit(X_train_flattened, y_train,
epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.5997 - accuracy: 0.7983
Epoch 2/5
1875/1875 [==============================] - 2s 1ms/step - loss: 0.4617 - accuracy: 0.8426
Epoch 3/5
```

```
model.evaluate(X_test_flattened, y_test)
```

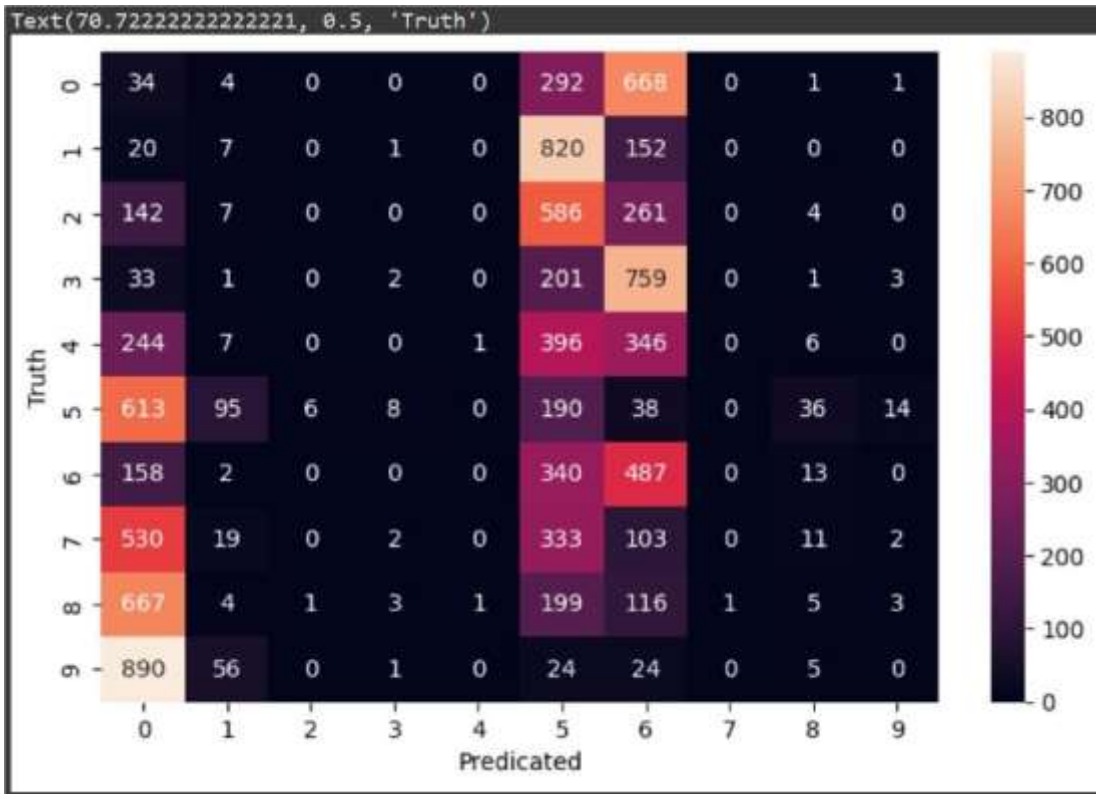
```
313/313 [==============================] - 0s 1ms/step - loss: 0.4572 - accuracy: 0.8403
[0.4571601450443268, 0.8403000235557556]
```

```
y_predicted=model.predict(X_test_flattened) y_predicted_labels=[np.argmax(i) for i in
y_predicted] cm=tf.math.confusion_matrix(labels=y_test,
predictions=y_predicted_labels) cm
```

```
313/313 [==============================] - 0s 866us/step
<tf.Tensor: shape=(10, 10), dtype=int32, numpy=
array([[796,   6,  19,  62,   9,   0,  92,   0,  16,   0],
       [  2, 959,   4,  26,   5,   0,   2,   0,   2,   0],
       [ 14,   5, 770,  11, 150,   1,  42,   0,   7,   0],
       [ 17,  18,  23, 864,  45,   0,  28,   0,   5,   0],
       [  0,   3, 119,  30, 803,   0,  40,   0,   5,   0],
       [  0,   0,   0,   1,   0, 931,   0,  48,   2,  18],
       [123,   4, 168,  51, 174,   0, 457,   0,  23,   0],
       [  0,   0,   0,   0,   0,  34,   0, 949,   0,  17],
       [  3,   1,  11,  11,   4,   5,  10,   4, 951,   0],
       [  0,   0,   0,   0,   0,  21,   0,  55,   1, 923]], dtype=int32)>
```

plt.matshow(X_train[1])

import seaborn as sn plt.figure(figsize =(10,7))
sn.heatmap(cm,annot=True,fmt='d')
plt.xlabel('Predicated')
plt.ylabel('Truth')



Q.-Write a program to carry out clustering using K-means also find the value of k using elbow method (Density Base).

import numpy as np def k_means(X,K,max_iters=10): m,n=X.shape
#Initialize centroids randomly centroids =
X[np.random.choice(m,K,replace=False),:] print(centroids)

for i in range(max_iters):

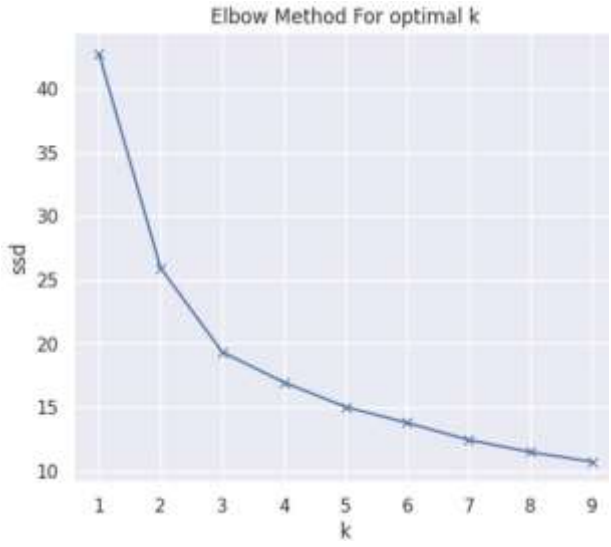
distances = np.sqrt(((X-centroids[:,np.newaxis])**2).sum(axis=2))
labels=np.argmin(distances,axis=0)

for j in range(K):
centroids[j] = X[labels == j].mean(axis=0)

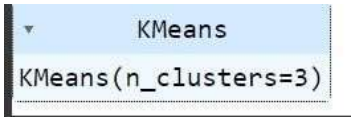
return centroids,labels
X=np.random.randn(100,2) centroids,labels =
k_means(X,K=3)
print("labels") print(labels)
print("centroids") print(centroids)

```
[[-0.61932384 -0.10502437]
 [-0.40905091  0.81148857]
 [ 0.35705637 -0.27520999]]
labels
[2 0 1 2 2 1 1 0 2 0 0 1 2 2 0 2 1 2 1 1 2 0 2 2 1 0 1 1 2 2 0 2 2 2 2 2 1
 2 0 0 2 2 2 2 0 0 0 2 1 2 0 1 0 2 1 2 2 2 2 2 2 2 1 1 0 0 1 1 1 1 1 0 1 0
 1 2 1 2 2 1 2 2 2 1 0 1 2 2 0 2 2 2 0 2 0 2 2 0 2 1]
centroids
[[-0.99352762 -0.45279521]
 [-0.24287917  1.21542654]
 [ 0.90326041 -0.27528874]]
```

```
import numpy as np import pandas
as pd import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from sklearn.cluster import KMeans   data =
pd.read_csv("/content/drive/MyDrive/csvfiles/Country-data.csv") data

from sklearn.preprocessing import MinMaxScaler scalar=MinMaxScaler()

scaled_data= scalar.fit_transform(data.drop('country',axis=1))

scaled_df=pd.DataFrame(data = scaled_data,columns=data.columns[1:])
scaled_df['country']=data['country'] scaled_df

data = scaled_df.drop('country',axis=1)

#calculate sum of squared distances

ssd = []
K=range(1,10);   for   k   in   K:
km=KMeans(n_clusters=k)
km=km.fit(data)
ssd.append(km.inertia_)

plt.figure(figsize=(6,5)) plt.plot(K,ssd,'bx-')
plt.xlabel('k') plt.ylabel('ssd')
plt.title('Elbow Method For optimal k') plt.show()
```



```
kmean = KMeans(n_clusters =3)
kmean.fit(data)
```

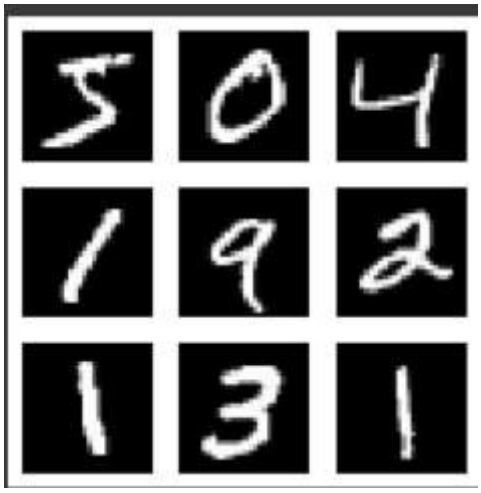



```
pred = kmean.labels_ print(pred)
```

```
[1 0 0 1 0 0 0 2 2 0 0 0 0 0 0 2 0 1 0 0 0 0 0 2 0 1 1 0 1 2 0 1 1 0 0 0 1
 1 1 0 1 0 2 2 2 0 0 0 0 1 1 0 0 2 2 1 1 0 2 1 2 0 0 1 1 0 1 0 2 0 0 0 1 2
 2 2 0 2 0 0 1 1 2 0 1 0 0 1 1 0 0 2 0 1 1 0 0 1 2 1 0 0 0 0 0 0 1 0 1 0 2
 2 1 1 2 0 1 0 0 0 0 0 2 2 0 0 1 0 0 1 0 0 1 2 2 2 1 0 2 2 0 0 1 0 2 2 0 1
 0 1 1 0 0 0 0 1 0 2 2 2 0 0 0 0 0 1 1]
```

Q.- Write a program to carry out data augmentation for a given set of images.

```
import os   import tensorflow as tf   from
tensorflow.keras.datasets import mnist import
matplotlib.pyplot as plt
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator import
numpy as np

(X_train,y_train),(X_test,y_test)=mnist.load_data()

X_train.shape,X_test.shape

X_train = X_train.reshape(X_train.shape[0],28,28,1)
X_test = X_test.reshape(X_test.shape[0],28,28,1)
#change the type to float
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

X_train.shape,X_test.shape

data_generator = ImageDataGenerator()


for X_batch,y_batch in data_generator.flow(X_train,y_train,batch_size =9,shuffle=False):
  fig, ax = plt.subplots(3,3,figsize=(4,4))    for i in
range(3):       for j in range(3):          ax[i][j].axis('off')
      ax[i][j].imshow(X_batch[i*3 + j].reshape(28,28),cmap=plt.get_cmap('gray'))    plt.show()
break;
```



```
data_generator = ImageDataGenerator(
rescale=1./255,        rotation_range=30,
zoom_range=0.2,        shear_range=0.2,
      height_shift_range=0.2,
)

for X_batch,y_batch in data_generator.flow(X_train,y_train,batch_size =9,shuffle=False):
   fig, ax = plt.subplots(3,3,figsize=(4,4))     for i in
range(3):        for j in range(3):            ax[i][j].axis('off')
         ax[i][j].imshow(X_batch[i*3 + j].reshape(28,28),cmap=plt.get_cmap('gray'))      plt.show()
break;
```