INDEX

PART - B		
WEEK NO	**TOPIC NAME**	**PAGE NO**
1	Generate Association Rules using the Apriori algorithm	
2	Generate Association Rules using the F-P Growth algorithm	
3	K-Means Clustering	
4	Hierarchical Clustering	
5	DB CAN Clustering	
6	Web Scraping	

PART - A

Week-1

Extract data from different le formats and display the summary statistics.

Extracting data from csv file

A CSV is a comma-separated values le, which allows data to be saved in a tabular format. CSVs look like a garden-variety spreadsheet but with a .csv extension.

CSV les can be used with most any spreadsheet program, such as Microsoft Excel or Google Spreadsheets.

There are 2 ways to extract data from a csv file.They are

1.Pandas read_csv

2.CSV Reader

Using pandas read_csv

```
import pandas as pd
```

```
from google.colab import files
uploaded=files.upload()
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving clever.csv to clever.csv
```

```
dfc=pd.read_csv("clever.csv")
dfc
```

	Sl.No	Age	Marks	Final	IQ
0	1	19	87.0	89.0	87
1	2	18	78.0	82.0	91
2	3	17	NaN	NaN	101
3	4	20	NaN	NaN	76
4	5	18	47.0	55.0	74

```
dfc.describe
```

```
<bound method NDFrame.describe of    Sl.No  Age  Marks  Final   IQ
0      1   19   87.0   89.0   87
1      2   18   78.0   82.0   91
2      3   17    NaN    NaN  101
3      4   20    NaN    NaN   76
4      5   18   47.0   55.0   74>
```

Reading csv file using csv reader

```
import csv
```

```
from google.colab import files
uploaded=files.upload()
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving home.csv to home.csv
```

```
#The type of file is "_io.TextIOWrapper" which is a file object that is returned by the open() method.
file=open("home.csv")
type(file)
```

```
_io.TextIOWrapper
```

```
csvreader=csv.reader(file)


# Extract the field names
header=[]
header=next(csvreader)
header
```

```
    ['area', 'price']
```

```
rows=[]
for i in csvreader:
  rows.append(i)
rows[:5]
```

```
    [['2600', '550000'],
     ['3000', '565000'],
     ['3200', '610000'],
     ['3600', '680000'],
     ['4000', '725000']]
```

Implementing the above code using with() statement:

Syntax: with open(filename, mode) as alias_filename:

Modes:

'r' - to read an existing file,

'w' – to create a new file if the given file doesn't exist and write to it,

'a' – to append to existing file content,

'+' – to create a new file for reading and writing

```
file.close()
```

Extracting data from JSON file

JSON (JavaScript Object Notation) is an open standard file format for sharing data that uses human-readable text to store and transmit data. JSON files are stored with the .json extension. JSON requires less formatting and is a good alternative for XML. JSON is derived from JavaScript but is a language-independent data format. The generation and parsing of JSON is supported by many modern programming languages.

JSON data is written in key/value pairs. The key and value are separated by a colon(:) in the middle with the key on the left and the value on the right. Different key/value pairs are separated by a comma(,).

```
import json


data='{"Rohit":45,"Kohli":18,"Dhoni":7,"Hardik":33,"Kishan":23}'


#Now we will use the loads() function from 'json' module to load the json data from the variable. We store the json data
json_load=(json.loads(data))
#We will use the dumps() function from the 'json' module to convert the python string to a JSON object
print(json.dumps(json_load,indent= 5))
```

```
    {
         "Rohit": 45,
         "Kohli": 18,
         "Dhoni": 7,
         "Hardik": 33,
         "Kishan": 23
    }
```

```
for i in json_load:
  print("%s : %d" % (i,json_load[i]))
```

```
    Rohit : 45
    Kohli : 18
    Dhoni : 7
    Hardik : 33
    Kishan : 23
```

Extracting data from XML file

XML or eXtensible Markup Language is a markup language that defines rules for encoding data/documents to be shared across the Internet. Like JSON, XML is also a text format that is human readable. Its design goals involved strong support for various human languages (via Unicode), platform independence, and simplicity. XMLs are widely used for representing data of varied shapes and sizes.

XMLs are widely used as configuration formats by different systems, metadata, and data representation format for services like RSS, SOAP, and many more.

XML is a language with syntactic rules and schemas defined and refined over the years.

The most import components of an XML are as follows:

• **Tag:** A markup construct denoted by strings enclosed with angled braces (“<” and “>”).

• **Content:** Any data not marked within the tag syntax is the content of the XML file/object.

• **Element:** A logical construct of an XML. An element may be defined with a start and an end tag with or without attributes, or it may be simply an empty tag.

• **Attribute:** Key-value pairs that represent the properties or attributes of the element in consideration. These are enclosed within a start or an empty tag

Xml files can be extracted in two ways

1) ElementTree

2) Minidom

```
from google.colab import files
uploaded=files.upload()
```

```
Choose Files No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving books.xml to books.xml
```

```
#Using Element Tree:
import xml.etree.ElementTree as ET
mytree=ET.parse('books.xml')
myroot=mytree.getroot()
print(mytree)
print(myroot)
```

```
<xml.etree.ElementTree.ElementTree object at 0x7f6f5f36b590>
<Element 'catalog' at 0x7f6f5f35bd70>
```

```
#Using Minidom:
import xml.dom.minidom as dm
df=dm.parse('books.xml')
print(df.nodeName)
print(df.firstChild.tagName)
```

```
#document
catalog
```

```
myroot[0].tag
```

```
'book'
```

```
for j in myroot:
    print(j.attrib)
```

```
{'id': 'bk101'}
{'id': 'bk102'}
{'id': 'bk103'}
{'id': 'bk104'}
{'id': 'bk105'}
{'id': 'bk106'}
{'id': 'bk107'}
{'id': 'bk108'}
{'id': 'bk109'}
```

```
print('Underlying tags of book are : ')
for j in myroot[0]:
  print(j.tag,end='  ')
```

```
Underlying tags of book are :
author  title  genre  price  publish_date  description
```

```
for x in myroot:
  for j in x:
    print(j.text)
  print('\n')
```

```
    Gambardella, Matthew
    XML Developer's Guide
    Computer
    44.95
    2000-10-01
    An in-depth look at creating applications
          with XML.


    Ralls, Kim
    Midnight Rain
    Fantasy
    5.95
    2000-12-16
    A former architect battles corporate zombies,
          an evil sorceress, and her own childhood to become queen
          of the world.


    Corets, Eva
    Maeve Ascendant
    Fantasy
    5.95
    2000-11-17
    After the collapse of a nanotechnology
          society in England, the young survivors lay the
          foundation for a new society.


    Corets, Eva
    Oberon's Legacy
    Fantasy
    5.95
    2001-03-10
    In post-apocalypse England, the mysterious
          agent known only as Oberon helps to create a new life
          for the inhabitants of London. Sequel to Maeve
          Ascendant.


    Corets, Eva
    The Sundered Grail
    Fantasy
    5.95
    2001-09-10
    The two daughters of Maeve, half-sisters,
          battle one another for control of England. Sequel to
          Oberon's Legacy.


    Randall, Cynthia
    Lover Birds
    Romance
    4.95
    2000-09-02
    When Carla meets Paul at an ornithology
          conference, tempers fly as feathers get ruffled.
```

Week-2

Write a program that extracts the words (features) used in a sentence.

Text Pre-Processing Before feature engineering, we need to pre-process, clean, and normalize the text like we mentioned before. There are multiple pre-processing techniques, some of which are quite elaborate. We will not be going into a lot of details in this section but we will be covering a lot of them in further detail in a future chapter when we work on text classification and sentiment analysis. Following are some of the popular pre-processing techniques.

• Text tokenization and lower casing

• Removing special characters

• Contraction expansion

• Removing stopwords

• Correcting spellings

• Stemming

• Lemmatization

```
import numpy as np
import pandas as pd
import re
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
words_pattern = '[a-zA-Z0-9]+'
```

```
corpus="Rohit Gurunath Sharma is an Indian international cricketer, who is the current captain of the Indian national cr
```

```
re.findall(words_pattern, corpus)
```

```
['Rohit',
 'Gurunath',
 'Sharma',
 'is',
 'an',
 'Indian',
 'international',
 'cricketer',
 'who',
 'is',
 'the',
 'current',
 'captain',
 'of',
 'the',
 'Indian',
 'national',
 'cricket',
 'team',
 'In',
 'the',
 'Indian',
 'Premier',
 'League',
 'he',
 'captains',
 'Mumbai',
 'Indians',
 'and',
 'is',
 'a',
 'right',
 'handed',
 'opening',
 'batsman',
 'and',
```

```
 'an',
 'occasional',
 'right',
 'arm',
 'off',
 'break',
 'bowler',
 'He',
 'plays',
 'for',
 'Mumbai',
 'in',
 'domestic',
 'cricket']
```

WordPunctTokenizer():

With the help of nltk.tokenize.WordPunctTokenizer() method, we are able to extract the tokens from string of words or sentences in the form of Alphabetic and Non-Alphabetic character by using tokenize.WordPunctTokenizer() method.

```
wpt=nltk.WordPunctTokenizer()
stop_words=nltk.corpus.stopwords.words('english')
print(stop_words)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yours
```

```
def normalize_document(doc):
  doc=re.sub(r'[^a-zA-Z0-9\s]',' ',doc,re.I)
  doc=doc.lower()
  doc=doc.strip()
  # Tokenize Document
  tokens=wpt.tokenize(doc)
  # filter stopwords out of document
  filtered_tokens=[token for token in tokens if token not in stop_words]
  # re-create document from filtered tokens
  doc=" ".join(filtered_tokens)
  return doc


corpus=np.array([corpus])


#The np.vectorize(...) function helps us run the same function over all elements of a numpy array
#instead of writing a loop. We will now use this function to pre-process our text corpus
norm_corpus=np.vectorize(normalize_document)
norm_corpus
```

```
<numpy.vectorize at 0x7f9f4375f6d0>
```

```
normalize_corpus = np.vectorize(normalize_document)
normalize_corpus
```

```
<numpy.vectorize at 0x7f9f437d74d0>
```

```
norm_corpus=normalize_corpus(corpus)


norm_corpus
```

```
array(['rohit gurunath sharma indian international cricketer current captain indian national cricket team indian premier league
captains mumbai indians right - handed opening batsman , occasional right - arm break bowler . plays mumbai domestic cricket .'],
      dtype='<U246')
```

BAG OF WORDS MODEL

Bag of Words Model

This is perhaps one of the simplest yet effective schemes of vectorizing features from unstructured text. The core principle of this model is to convert text documents into numeric vectors. The dimension or size of each vector is N where N indicates all possible distinct words across the corpus of documents. Each document once transformed is a numeric vector of size N where the values or weights in the vector indicate the frequency of each word in that specific document.

CountVectorizer():

CountVectorizer is a great tool provided by the scikit-learn library in Python. It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text. This is helpful when we have multiple such texts, and we wish to convert each word in each text into vectors (for using in further text analysis).

CountVectorizer creates a matrix in which each unique word is represented by a column of the matrix, and each text sample from the document is a row in the matrix. The value of each cell is nothing but the count of the word in that particular text sample.

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(min_df=0., max_df=1.)
cv_matrix=cv.fit_transform(norm_corpus)
print(type(cv_matrix))
cv_matrix=cv_matrix.toarray()
cv_matrix
```

```
<class 'scipy.sparse.csr.csr_matrix'>
array([[1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 3, 1, 1, 1, 2, 1, 1, 1, 1, 1,
        2, 1, 1, 1]])
```

The output represents a numeric term frequency based feature vector for each document like we mentioned before. To understand it better, we can represent it using the feature names and view it as a dataframe

```
vocab=cv.get_feature_names()
pd.DataFrame(cv_matrix,columns=vocab)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function get_fea
  warnings.warn(msg, category=FutureWarning)
```

	arm	batsman	bowler	break	captain	captains	cricket	cricketer	current	domestic	...	mumbai	n
0	1	1	1	1	1	1	2	1	1	1	...	2	

1 rows × 26 columns

Bag of N-Grams Model

An n-gram is basically a collection of word tokens from a text document such that these tokens are contiguous and occur in a sequence. Bi-grams indicate n-grams of order 2 (two words), Tri-grams indicate n-grams of order 3 (three words), and so on. We can easily extend the bag of words model to use a bag of n-grams model to give us n-gram based feature vectors

```
bv=CountVectorizer(ngram_range=(2,2))
bv_matrix=bv.fit_transform(norm_corpus)
bv_matrix=bv_matrix.toarray()
bv_matrix
```

```
array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1]])
```

The output represents a numeric term frequency based feature vector for each document like we mentioned before. To understand it better, we can represent it using the feature names and view it as a dataframe

```
vocab=bv.get_feature_names()
pd.DataFrame(bv_matrix,columns=vocab)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function get_fea
  warnings.warn(msg, category=FutureWarning)
```

	arm break	batsman occasional	bowler plays	break bowler	captain indian	captains mumbai	cricket team	cricketer current	current captain	domestic cricket	...	nat cr
0	1	1	1	1	1	1	1	1	1	1	...	

1 rows × 30 columns

TF-IDF Model

TF-IDF stands for Term Frequency-Inverse Document Frequency, which uses a combination of two metrics in its computation, namely: term frequency (tf) and inverse document frequency (idf). This technique was developed for ranking results for queries in search engines and now it is an indispensable model in the world of information retrieval and text analytics

```
corpus = ['The sky is blue and beautiful.',
'Love this blue and beautiful sky!',
'The quick brown fox jumps over the lazy dog.',
'The brown fox is quick and the blue dog is lazy!',
'The sky is very blue and the sky is very beautiful today',
'The dog is lazy but the brown fox is quick!']


labels = ['weather', 'weather', 'animals', 'animals', 'weather', 'animals']
corpus = np.array(corpus)
corpus_df = pd.DataFrame({'Document': corpus,'Category': labels})
corpus_df = corpus_df[['Document', 'Category']]
corpus_df
```

	Document	Category
0	The sky is blue and beautiful.	weather
1	Love this blue and beautiful sky!	weather
2	The quick brown fox jumps over the lazy dog.	animals
3	The brown fox is quick and the blue dog is lazy!	animals
4	The sky is very blue and the sky is very beaut...	weather
5	The dog is lazy but the brown fox is quick!	animals

```
normalize_corpus = np.vectorize(normalize_document)
norm_corpus=normalize_corpus(corpus)
norm_corpus
```

```
array(['sky blue beautiful', 'love blue beautiful sky',
       'quick brown fox jumps lazy dog', 'brown fox quick blue dog lazy',
       'sky blue sky beautiful today', 'dog lazy brown fox quick'],
      dtype='<U30')
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
tv = TfidfVectorizer(min_df=0., max_df=1., use_idf=True)
tv_matrix = tv.fit_transform(norm_corpus)
tv_matrix = tv_matrix.toarray()
vocab = tv.get_feature_names()
pd.DataFrame(np.round(tv_matrix, 2), columns=vocab)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function get_fea
  warnings.warn(msg, category=FutureWarning)
```

	beautiful	blue	brown	dog	fox	jumps	lazy	love	quick	sky	today
0	0.60	0.52	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.60	0.00
1	0.46	0.39	0.00	0.00	0.00	0.00	0.00	0.66	0.00	0.46	0.00
2	0.00	0.00	0.38	0.38	0.38	0.54	0.38	0.00	0.38	0.00	0.00
3	0.00	0.36	0.42	0.42	0.42	0.00	0.42	0.00	0.42	0.00	0.00
4	0.36	0.31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.72	0.52
5	0.00	0.00	0.45	0.45	0.45	0.00	0.45	0.00	0.45	0.00	0.00

Week - 3

Write a program for edge detection to extract edge based features from a sample image.

Edge detection is a very old problem in computer vision which involves detecting the edges in an image to determine object boundary and thus separate the object of interest. One of the most popular technique for edge detection has been Canny Edge detection which has been the go-to method for most of the computer vision researchers and practitioners.

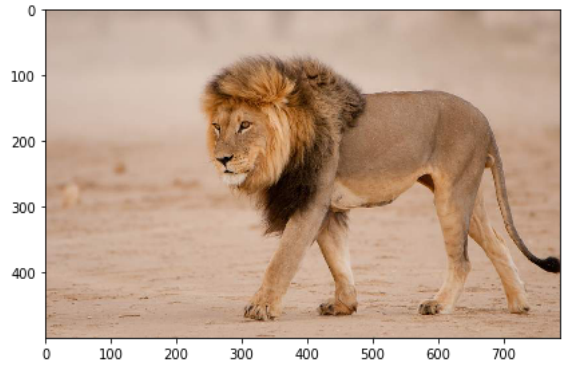
```
import skimage
from skimage import io
import pandas as pd
import matplotlib.pyplot as plt
```

```
from google.colab import files
uploaded=files.upload()
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Lion.jpg to Lion (1).jpg
```

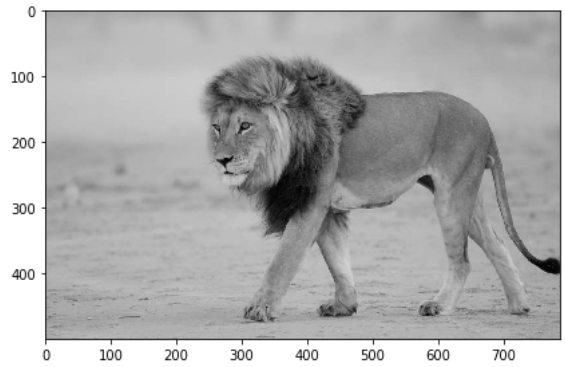
```
#Reading and dsiplaying image
lion = io.imread('Lion.jpg')
io.imshow(lion)
```

```
<matplotlib.image.AxesImage at 0x7f6e218e4290>
```




```
#converting to gray scale image using rgb2gray
from skimage.color import rgb2gray
lgs = rgb2gray(lion)
io.imshow(lgs)
```

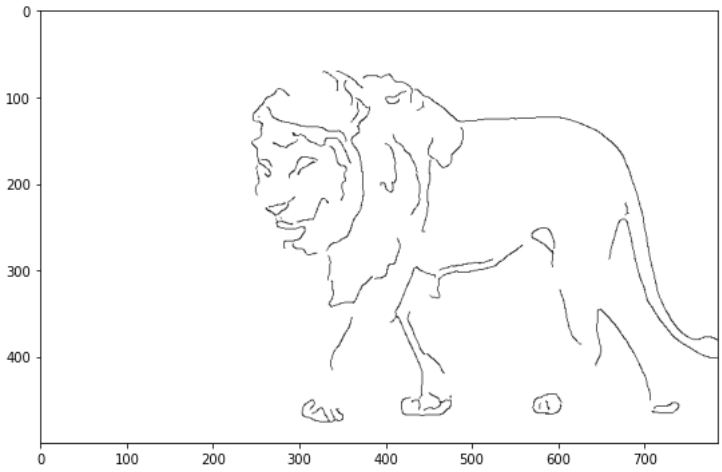
```
<matplotlib.image.AxesImage at 0x7f6e1ec0b810>
```




```
from skimage.feature import canny
lion_edges = canny(lgs, sigma=3)
fig = plt.figure(figsize = (20,15))
```

```
ax1 = fig.add_subplot(1,2, 1)
ax1.imshow(lion_edges, cmap='binary')
```

```
<matplotlib.image.AxesImage at 0x7f6e0b4eef90>
```



Week - 4

Write a program to extract SURF/SIFT feature descriptors from a sample image

The scale-invariant feature transform (SIFT) is a computer vision algorithm to detect, describe, and match local features in images.

SIFT keypoints of objects are first extracted from a set of reference images and stored in a database. An object is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature vectors.

```
import cv2 as cv
from skimage import io
from·google.colab.patches·import·cv2_imshow


from google.colab import files
uploaded=files.upload()
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Lion.jpg to Lion.jpg
```

```
image = cv.imread("Lion.jpg")
cv2_imshow(image)
```

```
#convert to grayscale image
gray_scale = cv.cvtColor(image, cv.COLOR_BGR2GRAY)


#initialize SIFT object
sift = cv.xfeatures2d.SIFT_create()


#detect keypoints
keypoints, descriptors= sift.detectAndCompute(image, None)
```

```
#draw keypoints
sift_image = cv.drawKeypoints(image, keypoints, None)


cv2_imshow(sift_image)
cv.waitKey(0)
```

Week - 5

Write a program to perform Exploratory Data Analysis on real time datasets using the following approaches:

a) Univariate Analysis

b) Multivariate Analysis

c) Visualization using correlation matrix

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import datasets
iris=datasets.load_iris()

df=pd.DataFrame(iris.data,columns=iris.feature_names)
df['species']=iris.target
df
```

	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

```
df.loc[df['species']==0,'species']='Setosa'
df.loc[df['species']==1,'species']='Versicolor'
df.loc[df['species']==2,'species']='Virginica'


df.columns=['Sepal_length','Sepal_width','Petal_length','Petal_width','class']

df.tail(9)
```

	Sepal_length	Sepal_width	Petal_length	Petal_width	class
141	6.9	3.1	5.1	2.3	Virginica
142	5.8	2.7	5.1	1.9	Virginica

UNIVARIATE ANALYSIS

Univariate analysis explores each variable in a data set, separately. It looks at the range of values, as well as the central tendency of the values. It describes the pattern of response to the variable. It describes each variable on its own. Descriptive statistics describe and summarize data.

```
#calculate mean of 'Sepal_width'
df['Sepal_width'].mean()
```

```
3.0573333333333337
```

```
#calculate median of 'Sepal_width'
df['Sepal_width'].median()
```

```
3.0
```

```
#calculate standarad deviation of 'Sepal_width'
df['Sepal_width'].std()
```

```
0.4358662849366982
```

```
#create frequency table for 'Petal_length'
df['Sepal_length'].value_counts()
```

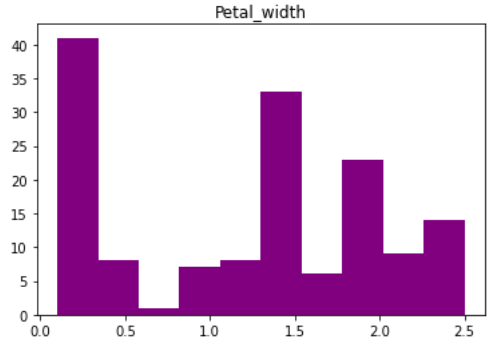
```
5.0    10
5.1     9
6.3     9
5.7     8
6.7     8
5.8     7
5.5     7
6.4     7
4.9     6
5.4     6
6.1     6
6.0     6
5.6     6
4.8     5
6.5     5
6.2     4
7.7     4
6.9     4
4.6     4
5.2     4
5.9     3
4.4     3
7.2     3
6.8     3
6.6     2
4.7     2
7.6     1
7.4     1
7.3     1
7.0     1
7.1     1
5.3     1
4.3     1
4.5     1
7.9     1
Name: Sepal_length, dtype: int64
```

```
import matplotlib.pyplot as plt

df.boxplot(column=['Petal_width'], grid=False, color='orange')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe8c89ae750>
```

```
import matplotlib.pyplot as plt

df.hist(column=['Petal_width'], grid=False, color='purple')
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fe8c898b110>]],
      dtype=object)
```

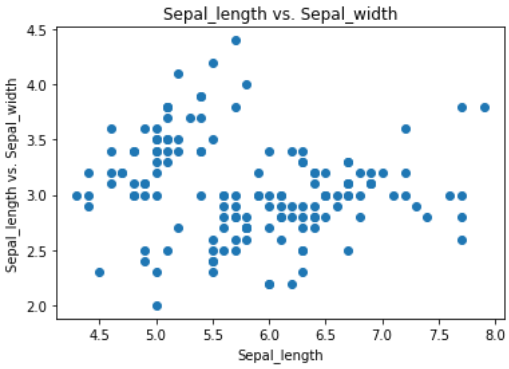


BIVARIATE ANALYSYIS

The analysis of two specific variables to determine the empirical relationship present between them is referred to as bivariate analysis and it is considered to be one of the simplest forms of quantitative analysis.

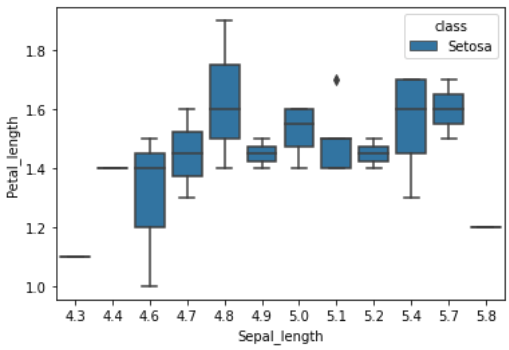
```
import matplotlib.pyplot as plt
#create scatterplot of Sepal_length vs. Sepal_width
plt.scatter(df.Sepal_length, df.Sepal_width)
plt.title('Sepal_length vs. Sepal_width')
plt.xlabel('Sepal_length')
plt.ylabel('Sepal_length vs. Sepal_width')
```

```
Text(0, 0.5, 'Sepal_length vs. Sepal_width')
```



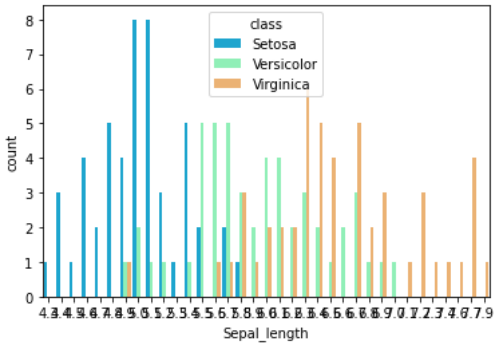

```
df1=df.head(30)
sns.boxplot(x='Sepal_length', y='Petal_length', hue='class', data=df1)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe8c887ab50>
```




```
sns.countplot(x='Sepal_length', hue='class', data=df, palette='rainbow')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe8cd1de290>
```



MULTIVARIATE ANALYSIS

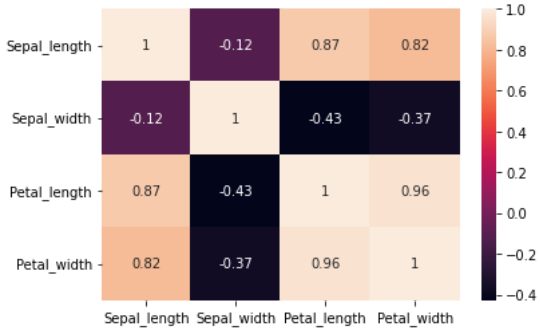
Multivariate analysis (MVA) is a Statistical procedure for analysis of data involving more than one type of measurement or observation. It may also mean solving problems where more than one dependent variable is analyzed simultaneously with other variables.

```
df.corr()
```

	Sepal_length	Sepal_width	Petal_length	Petal_width
Sepal_length	1.000000	-0.117570	0.871754	0.817941
Sepal_width	-0.117570	1.000000	-0.428440	-0.366126
Petal_length	0.871754	-0.428440	1.000000	0.962865
Petal_width	0.817941	-0.366126	0.962865	1.000000

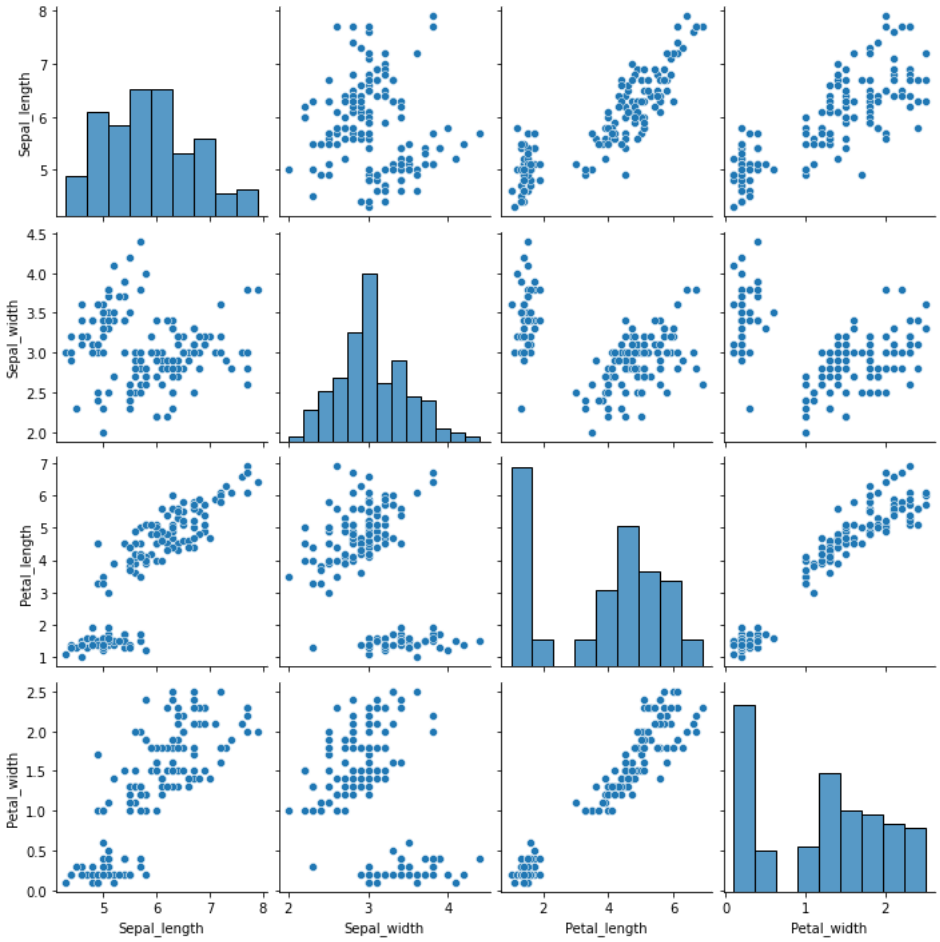
```
sns.heatmap(df.corr(),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe8c8501a50>
```




```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7fe8c8425050>
```



Week - 6

Write a program to perform any of the following Dimensionality Reduction techniques on real time datasets.

a) Principal Component Analysis

b) Single value Decomposition

c) Linear Discriminant Analysis

PRINCIPAL COMPONENT ANALYSIS

Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation that converts a set of correlated variables to a set of uncorrelated variables. PCA is the most widely used tool in exploratory data analysis and in machine learning for predictive models. Moreover, PCA is an unsupervised statistical technique used to examine the interrelations among a set of variables. It is also known as a general factor analysis where regression determines a line of best fit.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.datasets import load_iris
iris = load_iris()
df = pd.DataFrame(iris['data'], columns = iris['feature_names'])
df.head()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 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
dtypes: float64(4)
memory usage: 4.8 KB
```

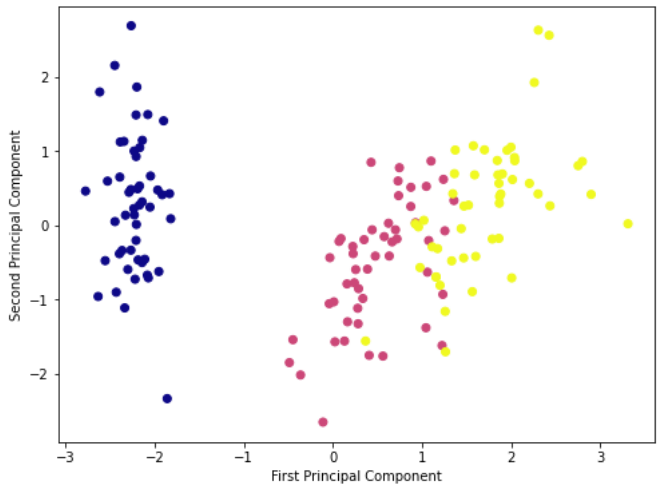
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
scaled_data = sc.fit_transform(df)
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
pca.fit(scaled_data)
x_pca = pca.transform(scaled_data)

x_pca.shape
```

```
(150, 2)
```

```
plt.figure(figsize =(8, 6))
plt.scatter(x_pca[:, 0], x_pca[:, 1], c = iris['target'], cmap ='plasma')

plt.xlabel('First Principal Component')
plt.ylabel('Second Principal Component')
```

```
Text(0, 0.5, 'Second Principal Component')
```

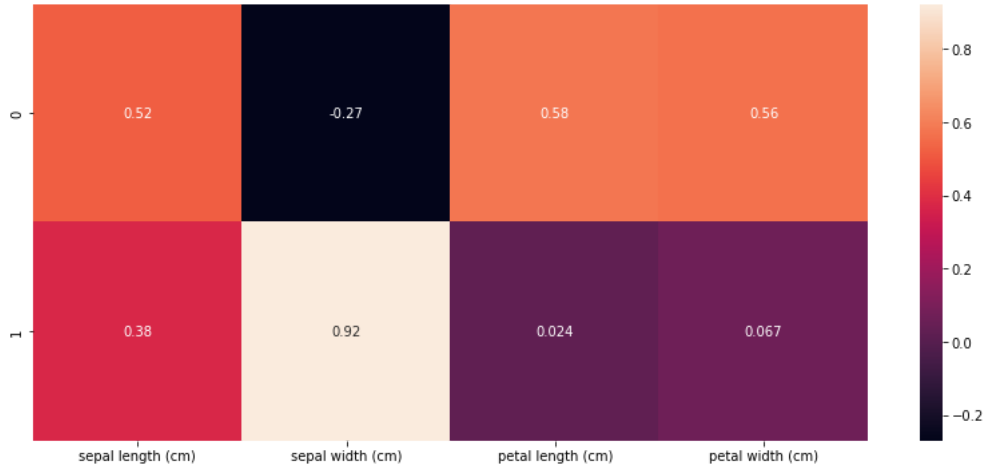



```
pca.components_
```

```
array([[ 0.52106591, -0.26934744,  0.5804131 ,  0.56485654],
       [ 0.37741762,  0.92329566,  0.02449161,  0.06694199]])
```

```
df_comp = pd.DataFrame(pca.components_, columns = iris['feature_names'])

plt.figure(figsize =(14, 6))
sns.heatmap(df_comp, annot = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f30a1e0df90>
```



SINGULAR VALUE DECOMPOSITION

The Singular Value Decomposition (SVD) of a matrix is a factorization of that matrix into three matrices. It has some interesting algebraic properties and conveys important geometrical and theoretical insights about linear transformations. It also has some important applications in data science. In this article, I will try to explain the mathematical intuition behind SVD and its geometrical meaning.

The SVD of mxn matrix A is given by the formula : A = UWV^{T}

where:

U: mxn matrix of the orthonormal eigenvectors of AA^{T}

.

VT: transpose of a nxn matrix containing the orthonormal eigenvectors of A^{T}A.

W: a nxn diagonal matrix of the singular values which are the square roots of the eigenvalues of A^{T}A .

```
import numpy as np
from scipy.linalg import svd
```

```
X = np.array([[3, 3, 2], [2,3,-2]])
print(X,'\n')

U, singular, V_transpose = svd(X)
print("U: ",U)
print("\nSingular array : \n",singular)
print("\nV^{T} : \n",V_transpose)
```

```
[[ 3  3  2]
 [ 2  3 -2]]

U:  [[-0.7815437 -0.6238505]
 [-0.6238505  0.7815437]]

Singular array :
 [5.54801894 2.86696457]

V^{T} :
 [[-0.64749817 -0.7599438  -0.05684667]
 [-0.10759258  0.16501062 -0.9804057 ]
 [-0.75443354  0.62869461  0.18860838]]
```

LINEAR DISCRIMINANT ANALYSIS

Linear Discriminant Analysis (LDA) is one of the commonly used dimensionality reduction techniques in machine learning to solve more than two-class classification problems. It is also known as Normal Discriminant Analysis (NDA) or Discriminant Function Analysis (DFA).

Linear Discriminant analysis is one of the most popular dimensionality reduction techniques used for supervised classification problems in machine learning. It is also considered a pre-processing step for modeling differences in ML and applications of pattern classification. Play Video

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix
```

```
from sklearn.datasets import load_iris
iris = load_iris()
dataset = pd.DataFrame(iris['data'], columns = iris['feature_names'])
```

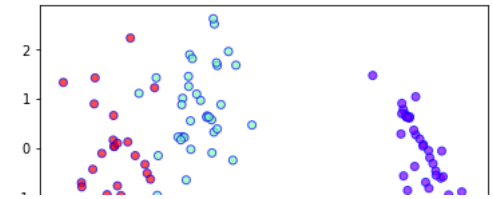
```
X = dataset
y = iris['target']
```

```
sc = StandardScaler()
X = sc.fit_transform(X)
le = LabelEncoder()
y = le.fit_transform(y)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
lda = LinearDiscriminantAnalysis(n_components=2)
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)
```

```
plt.scatter(X_train[:,0], X_train[:,1], c=y_train, cmap='rainbow', alpha=0.7, edgecolors='b')
```

```
<matplotlib.collections.PathCollection at 0x7f30a1d4ee50>
```




```
from sklearn.svm import SVC
classifier = SVC(kernel='linear', random_state=0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
print('Accuracy : ' + str(accuracy_score(y_test, y_pred)))
```

```
Accuracy : 0.9777777777777777
```

```
conf_m = confusion_matrix(y_test, y_pred)
print(conf_m)
```

```
[[14  0  0]
 [ 0 14  1]
 [ 0  0 16]]
```

PART - B

Week - 1

Write a program to generate Association Rules using the Apriori algorithm.

Apriori algorithm refers to the algorithm which is used to calculate the association rules between objects. It means how two or more objects are related to one another. In other words, we can say that the apriori algorithm is an association rule leaning that analyzes that people who bought product A also bought product B.

The primary objective of the apriori algorithm is to create the association rule between different objects. The association rule describes how two or more objects are related to one another. Apriori algorithm is also called frequent pattern mining. Generally, you operate the Apriori algorithm on a database that consists of a huge number of transactions.

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
# 'P' = 'Pizza' , 'B' = 'Burger' , 'I' = 'Icecream' , 'K' = 'Ketchup' , 'C' = 'Coke' , 'G' = 'Garlic' , 'B' = 'Bread'
dataset = [['Pizza','Burger','Icecream','Ketchup'],
           ['Pizza','Burger'],
           ['Pizza','Coke','Garlic'],
           ['Garlic','Coke'],
           ['Garlic','Ketchup'],
           ['Pizza','Coke','Icecream','Ketchup']]

te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
print(df)
# Building the model
frq_items = apriori(df, min_support = 0.05, use_colnames = True)
print(frq_items)

# Collecting the inferred rules in a dataframe
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False, False])
print(rules.head())
```

```
    Burger   Coke  Garlic  Icecream  Ketchup  Pizza
0    True  False   False      True     True   True
1    True  False   False     False    False   True
2   False   True    True     False    False   True
3   False   True    True     False    False  False
4   False  False    True     False     True  False
5   False   True   False      True     True   True
     support                        itemsets
0   0.333333                        (Burger)
1   0.500000                          (Coke)
2   0.500000                        (Garlic)
3   0.333333                       (Icecream)
4   0.500000                        (Ketchup)
5   0.666667                          (Pizza)
6   0.166667               (Icecream, Burger)
7   0.166667                (Ketchup, Burger)
8   0.333333                  (Burger, Pizza)
9   0.333333                   (Garlic, Coke)
10  0.166667                 (Icecream, Coke)
11  0.166667                  (Ketchup, Coke)
12  0.333333                    (Coke, Pizza)
13  0.166667                 (Ketchup, Garlic)
14  0.166667                   (Garlic, Pizza)
15  0.333333                (Icecream, Ketchup)
16  0.333333                  (Icecream, Pizza)
17  0.333333                   (Ketchup, Pizza)
18  0.166667         (Icecream, Ketchup, Burger)
19  0.166667           (Icecream, Burger, Pizza)
20  0.166667            (Ketchup, Burger, Pizza)
21  0.166667               (Garlic, Coke, Pizza)
22  0.166667          (Icecream, Ketchup, Coke)
23  0.166667            (Icecream, Coke, Pizza)
24  0.166667             (Ketchup, Coke, Pizza)
25  0.333333          (Icecream, Ketchup, Pizza)
```

```
26  0.166667  (Icecream, Ketchup, Burger, Pizza)
27  0.166667    (Icecream, Ketchup, Coke, Pizza)
                antecedents       consequents  antecedent support  \
20         (Ketchup, Burger)        (Icecream)            0.166667
42           (Ketchup, Coke)        (Icecream)            0.166667
60          (Ketchup, Pizza)        (Icecream)            0.333333
61                (Icecream)  (Ketchup, Pizza)            0.333333
67  (Ketchup, Burger, Pizza)        (Icecream)            0.166667

    consequent support   support  confidence  lift  leverage  conviction
20            0.333333  0.166667         1.0   3.0  0.111111         inf
42            0.333333  0.166667         1.0   3.0  0.111111         inf
60            0.333333  0.333333         1.0   3.0  0.222222         inf
61            0.333333  0.333333         1.0   3.0  0.222222         inf
67            0.333333  0.166667         1.0   3.0  0.111111         inf
```

Week - 2

Write a program to generate Association Rules using the FP-Growth algorithm.

```
import sys
!{sys.executable} -m pip install fpgrowth
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting fpgrowth
  Downloading fpGrowth-1.0.0.tar.gz (2.1 kB)
Building wheels for collected packages: fpgrowth
  Building wheel for fpgrowth (setup.py) ... done
  Created wheel for fpgrowth: filename=fpGrowth-1.0.0-py3-none-any.whl size=2866 sha256=67b9531e780f75a0195b828f9cbecc2bc82b28cebf0
  Stored in directory: /root/.cache/pip/wheels/64/33/72/1991a9117d1813325c4ef85597ba8ece8c4780adc240bd0b0f
Successfully built fpgrowth
Installing collected packages: fpgrowth
Successfully installed fpgrowth-1.0.0
```

```
%pip install mlxtend --upgrade
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: mlxtend in /usr/local/lib/python3.7/dist-packages (0.14.0)
Collecting mlxtend
  Downloading mlxtend-0.21.0-py2.py3-none-any.whl (1.3 MB)
     |████████████████████████████████| 1.3 MB 5.2 MB/s
Requirement already satisfied: scipy>=1.2.1 in /usr/local/lib/python3.7/dist-packages (from mlxtend) (1.7.3)
Requirement already satisfied: numpy>=1.16.2 in /usr/local/lib/python3.7/dist-packages (from mlxtend) (1.21.6)
Requirement already satisfied: pandas>=0.24.2 in /usr/local/lib/python3.7/dist-packages (from mlxtend) (1.3.5)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from mlxtend) (57.4.0)
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.7/dist-packages (from mlxtend) (1.0.2)
Requirement already satisfied: matplotlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from mlxtend) (3.2.2)
Requirement already satisfied: joblib>=0.13.2 in /usr/local/lib/python3.7/dist-packages (from mlxtend) (1.2.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0.0->mlxtend) (2.
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0.0->mlxtend) (1.4.4
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0.0->mlxtend) (0.11.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.0.1->matplotlib>=3.0
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.24.2->mlxtend) (2022.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib>=3.0.0->ml
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=1.0.2->mlxtend) (
Installing collected packages: mlxtend
  Attempting uninstall: mlxtend
    Found existing installation: mlxtend 0.14.0
    Uninstalling mlxtend-0.14.0:
      Successfully uninstalled mlxtend-0.14.0
Successfully installed mlxtend-0.21.0
```

The idea behind the FP Growth algorithm is to find the frequent itemsets in a dataset while being faster than the Apriori algorithm. The Apriori algorithm basically goes back and forth to the data set to check for the co-occurrence of products in the data set.

Steps of the FP Growth Algorithm

Step 1 — Counting the occurrences of individual items.

Step 2— Filter out non-frequent items using minimum support.

Step 3— Order the itemsets based on individual occurrences.

Step 4— Create the tree and add the transactions one by one.

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import fpgrowth
dataset = [['Pizza','Burger','Icecream','Ketchup'],
           ['Pizza','Burger'],
           ['Pizza','Coke','Garlic'],
           ['Garlic','Coke'],
           ['Garlic','Ketchup'],
           ['Pizza','Coke','Icecream','Ketchup']]

te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
print(df)
```

```
fpgrowth(df, min_support=0.3)
fpgrowth(df, min_support=0.3, use_colnames=True)
```

```
   Burger   Coke  Garlic  Icecream  Ketchup  Pizza
0    True  False   False      True     True   True
1    True  False   False     False    False   True
2   False   True    True     False    False   True
3   False   True    True     False    False  False
4   False  False    True     False     True  False
5   False   True   False      True     True   True
```

	support	itemsets
0	0.666667	(Pizza)
1	0.500000	(Ketchup)
2	0.333333	(Icecream)
3	0.333333	(Burger)
4	0.500000	(Garlic)
5	0.500000	(Coke)
6	0.333333	(Ketchup, Pizza)
7	0.333333	(Ketchup, Icecream)
8	0.333333	(Pizza, Icecream)
9	0.333333	(Ketchup, Pizza, Icecream)
10	0.333333	(Burger, Pizza)
11	0.333333	(Coke, Garlic)
12	0.333333	(Coke, Pizza)

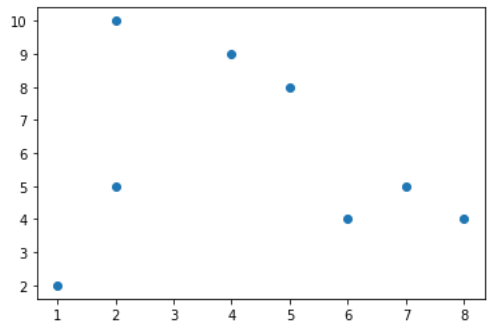
Week - 3

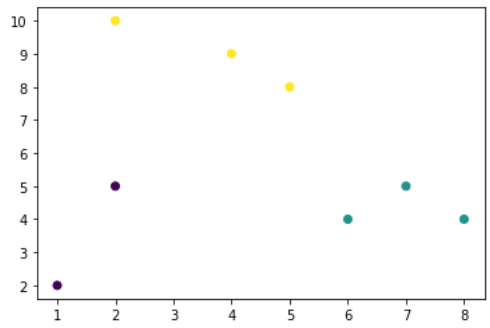
Write a program to implement K-means clustering algorithm.

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

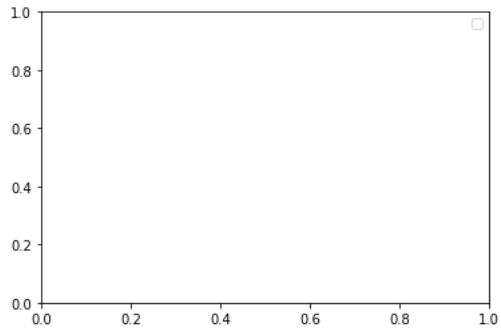
It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

x = [2,2,8,5,7,6,1,4]
y = [10,5,4,8,5,4,2,9]

plt.scatter(x, y)
plt.show()
data = list(zip(x, y))
kmeans = KMeans(n_clusters=3)
kmeans.fit(data)

plt.scatter(x, y, c=kmeans.labels_)
plt.show()
plt.legend()
```






```
WARNING:matplotlib.legend:No handles with labels found to put in legend.
<matplotlib.legend.Legend at 0x7f092b7d4610>
```



Week - 4

Write a program to implement hierarchical clustering algorithms.

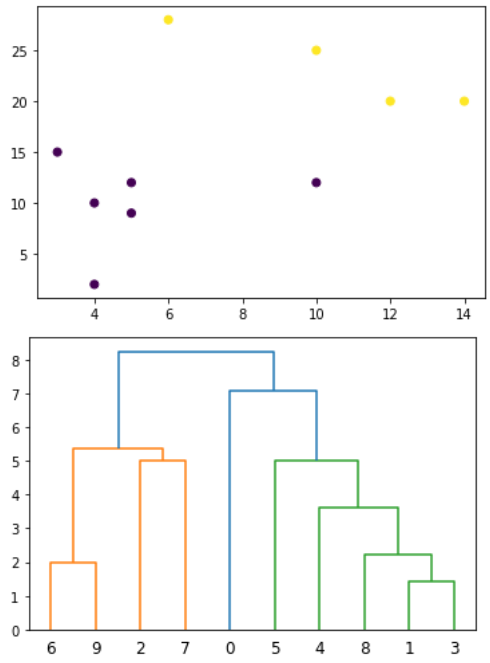
In data mining and statistics, hierarchical clustering analysis is a method of cluster analysis that seeks to build a hierarchy of clusters i.e. tree-type structure based on the hierarchy.

Basically, there are two types of hierarchical cluster analysis strategies –

1. Agglomerative Clustering: Also known as bottom-up approach or hierarchical agglomerative clustering (HAC). A structure that is more informative than the unstructured set of clusters returned by flat clustering. This clustering algorithm does not require us to prespecify the number of clusters. Bottom-up algorithms treat each data as a singleton cluster at the outset and then successively agglomerates pairs of clusters until all clusters have been merged into a single cluster that contains all data.
2. Divisive clustering: Also known as a top-down approach. This algorithm also does not require to prespecify the number of clusters. Top-down clustering requires a method for splitting a cluster that contains the whole data and proceeds by splitting clusters recursively until individual data have been split into singleton clusters.

Linkage: Single is MIN and Complete is MAX

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage
data=[(4, 2), (5, 9), (10, 25), (4, 10), (3, 15), (10, 12), (14, 20), (6, 28), (5, 12),(12,20)]
hierarchical_cluster = AgglomerativeClustering(n_clusters=2, affinity='euclidean', linkage='single')
labels = hierarchical_cluster.fit_predict(data)
x=[i[0] for i in data]
y=[i[1] for i in data]
plt.scatter(x,y,c=labels)
plt.show()
dendrogram(linkage(data, method='single', metric='euclidean'))
plt.show()
```



Week - 5

Write a program to implement the DBSCAN clustering algorithm.

DBSCAN Algorithm :

- DBSCAN is a popular density-based data clustering algorithm. To cluster data points, this algorithm separates the high-density regions of the data from the low-density areas. Unlike the K-Means algorithm, the best thing with this algorithm is that we don't need to provide the number of clusters required prior.
- DBSCAN algorithm group points based on distance measurement, usually the *Euclidean distance* and *the minimum number of points*. An essential property of this algorithm is that it helps us track down the outliers as the points in low-density regions; hence it is not sensitive to outliers as is the case of K-Means clustering.

DBSCAN algorithm works with two parameters.

These parameters are:

- **Epsilon (Eps):** This is the least distance required for two points to be termed as a neighbor. This distance is known as Epsilon (Eps). Thus we consider Eps as a threshold for considering two points as neighbors, i.e., if the distance between two points is utmost Eps, then we consider the two points to be neighbors.
- **MinPoints:** This refers to the minimum number of points needed to construct a cluster. We consider MinPoints as a threshold for considering a cluster as a cluster. A cluster is only recognized if the number of points is greater than or equal to the MinPts.

Types of data points in a DBSCAN clustering

After the DBSCAN clustering is complete, we end up with three types of data points as follows:

Core: This is a point from which the two parameters above are fully defined, i.e., a point with at least Minpoints within the Eps distance from itself.

Border: This is any data point that is not a core point, but it has at least one Core point within Eps distance from itself.

Noise: This is a point with less than Minpoints within distance Eps from itself. Thus, it's not a Core or a Border.

```
from sklearn.cluster import DBSCAN
import numpy as np

X = np.array([[1, 2], [2, 5], [2, 3],[7, 8], [8, 10], [25, 80]])

clustering = DBSCAN(eps=3, min_samples=2).fit(X)
print(clustering.labels_)
```

```
[ 0  0  0  1  1 -1]
```

Week - 6

Write a program to extract data from a web page.

What is Web Scraping:

Web scraping, web harvesting, or web data extraction is an automated process of collecting large data(unstructured) from websites. The user can extract all the data on particular sites or the specific data as per the requirement. The data collected can be stored in a structured format for further analysis.

Steps involved in web scraping:

1.Find the URL of the webpage that you want to scrape

2.Select the particular elements by inspecting

3.Write the code to get the content of the selected elements

4.Store the data in the required format

The popular libraries/tools used for web scraping are:

Selenium – a framework for testing web applications

BeautifulSoup – Python library for getting data out of HTML, XML, and other markup languages

Pandas – Python library for data manipulation and analysis

Step 1: Find the URL of the webpage that you want to scrape

Step 2: Select the particular elements by inspecting

Step 3: Write the code to get the content of the selected elements

```
# #Step 1.1: Defining the Base URL, Query parameters
# base_url="https://www.consumeraffairs.com/food/dominos.html"
# query_parameter="?page="+str(i)


#importing libraries
import pandas as pd
import requests
from bs4 import BeautifulSoup as bs


all_pages_review=[]
def scrapper():
```

```
  for i in range(1,6):
    pagewise_reviews=[]
    base_url="https://www.consumeraffairs.com/food/dominos.html"
    query_parameter="?page="+str(i)
    url=base_url+query_parameter  #Construct the URL
    response=requests.get(url)  #Send HTTP request to the URL using requests an
    soup=bs(response.content,'html.parser')
    rev_div=soup.findAll("div",attrs={"class","rvw-bd"})
    for j in range(len(rev_div)):     # finding all the p tags to fetch only th
      pagewise_reviews.append(rev_div[j].find("p").text)
    for k in range(len(pagewise_reviews)):
      all_pages_review.append(pagewise_reviews[k])
  return all_pages_review

#Driver Code
reviews=scrapper()
reviews_df=pd.DataFrame({'Review':reviews})
print(reviews_df)
reviews_df.to_csv('reviews.txt')
```

```
                                                Review
0    OH, this restaurant is So worth it. I mean lis...
1    I ordered stuffed cheesy bread for a change an...
2    Assistant was very Professional and understood...
3    Ordered a veggie lover's pizza without onion f...
4    Just another good experience with the Domino's...
..                                                 ...
125  I ordered pizza from Domino's and the delivery...
126  Pre-ordered 6 pizzas for delivery at 6:30 pm. ...
127  I was calling Domino's to ask them a question ...
128  I ordered a chicken caesar salad for delivery ...
129  We gone to Domino's Pizza near grand mall. Fir...

[130 rows x 1 columns]
```

```
reviews_df.head()
```

	Review
0	OH, this restaurant is So worth it. I mean lis...
1	I ordered stuffed cheesy bread for a change an...
2	Assistant was very Professional and understood...
3	Ordered a veggie lover's pizza without onion f...
4	Just another good experience with the Domino's...

```
all_pages_review=[]
def scrapper():
  for i in range(1,6):
    pagewise_reviews=[]
```

```
    base_url="https://www.consumeraffairs.com/food/pizza-hut.html"
    query_parameter="?page="+str(i)
    url=base_url+query_parameter  #Construct the URL
    response=requests.get(url)  #Send HTTP request to the URL using requests an
    soup=bs(response.content,'html.parser')
    rev_div=soup.findAll("div",attrs={"class","rvw-bd"})
    for j in range(len(rev_div)):     # finding all the p tags to fetch only th
      pagewise_reviews.append(rev_div[j].find("p").text)
    for k in range(len(pagewise_reviews)):
      all_pages_review.append(pagewise_reviews[k])
  return all_pages_review

#Driver Code
reviews=scrapper()
reviews1_df=pd.DataFrame({'Review':reviews})
print(reviews1_df)
reviews1_df.to_csv('reviews.txt')
```

```
                                                Review
0    I Order from our Local Pizza Hut on Average ab...
1    The pizza was amazing but when the pizza guy a...
2    The team with the Pizza Hut in LaBelle couldn'...
3    So I ordered this pizza from Pizza Hut, man it...
4    We called to order 10 orders of breadsticks an...
..                                                 ...
125  Their P'zone is ALL dough and next to NOTHING ...
126  I went to visit my granddaughter in Grove Okla...
127  I have reached out to Pizza Hut directly via p...
128  We get wings plain ran through the oven twice....
129  I haven't ordered from Pizza Hut in a while, s...

[130 rows x 1 columns]
```

```
reviews1_df.head()
```

	Review
0	I Order from our Local Pizza Hut on Average ab...
1	The pizza was amazing but when the pizza guy a...
2	The team with the Pizza Hut in LaBelle couldn'...
3	So I ordered this pizza from Pizza Hut, man it...
4	We called to order 10 orders of breadsticks an...