DPU

Dr. D. Y. Patil Institute of Technology
Pimpri Pune-411018

Department of Artificial Intelligence and Data Science

Laboratory Manual

Savitribai Phule Pune University

Fourth Year of Artificial Intelligence and Data Science (2020 Course)

Subject Code: 417526

Computer Laboratory II

Prepared by

Mrs. Sonali B. Gavali

Academic Year

2023-2024

Dr. D. Y. Patil Institute of Technology
Pimpri Pune-411018

Department of Artificial Intelligence and Data Science

Vision of the Institute

- Empowerment through knowledge

Mission of the Institute

- Developing human potential to serve the Nation
- Dedicated efforts for quality education.
- Yearning to promote research and development.
- Persistent endeavor to imbibe moral and professional ethics.
- Inculcating the concept of emotional intelligence.
- Emphasizing extension work to reach out to the society.
- Treading the path to meet the future challenges.

Vision of the Department

- To produce globally competent engineers in the field of Artificial Intelligence and Data Science with human values

Mission of the Department

- To develop students with a sound understanding in the area of Artificial Intelligence, Machine Learning and Data Science.
- To enable students to become innovators, researchers, entrepreneurs and leaders globally.
- Equip the department with new advancement in high performance equipments and software to carrying out research in emerging technologies in AI and DS.
- To meet the pressing demands of the nation in the areas of Artificial Intelligence and Data Science.

Computer Laboratory II
417526

Teaching Scheme	Credit Scheme	Examination Scheme and Marks
Practical: 04 Hours/Week	02	Term Work: 50 Marks Practical: 50 Marks

Course Objectives:
To understand the concepts of information retrieval and web mining and Understand information retrieval process using standards available tools.

Course Outcomes:
On completion of the course, learner will be able to–
CO1: Apply various tools and techniques for information retrieval and web mining.
CO2: Evaluate and analyze retrieved information.

Guidelines for Instructor's Manual
Lab Assignments: Following is a list of suggested laboratory assignments for reference. Laboratory Instructors may design a suitable set of assignments for their respective courses at their level. Beyond curriculum assignments, the mini-project is also included as a part of laboratory work. The Inclusion of a few optional assignments that are intricate and/or beyond the scope of the curriculum will surely be a valuable addition for the students and it will satisfy the intellectuals within the group of learners and will add to the perspective of the learners. For each laboratory assignment, it is essential for students to draw/write/generate flowcharts, algorithms, test cases, mathematical models, Test data sets, and comparative/complexity analysis (as applicable).

Guidelines for Student's Laboratory Journal
Program codes with sample output of all performed assignments are to be submitted as a softcopy. The use of DVDs or similar media containing student programs maintained by the Laboratory Incharge is highly encouraged. For reference one or two journals may be maintained with program prints in the Laboratory. As a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers as part of write-ups and program listing to journals may be avoided. Submission of journal/ term work in the form of softcopy is desirable and appreciated.

Guidelines for Laboratory / Term Work Assessment
Term work is a continuous assessment that evaluates a student's progress throughout the semester. Term work assessment criteria specify the standards that must be met and the evidence that will be gathered to demonstrate the achievement of course outcomes. Categorical assessment criteria for the term work should establish unambiguous standards of achievement for each course outcome. They should describe what the learner is expected to perform in the laboratories or on the fields to show that the course

outcomes have been achieved. It is recommended to conduct an internal monthly practical examination as part of continuous assessment.

Guidelines for Laboratory Conduction

Following is a list of suggested laboratory assignments for reference. Laboratory Instructors may design a suitable set of assignments for respective courses at their level. Beyond curriculum assignments and mini-project may be included as a part of laboratory work. The instructor may set multiple sets of assignments and distribute them among batches of students. It is appreciated if the assignments are based on real-world problems/applications. The Inclusion of a few optional assignments that are intricate and/or beyond the scope of the curriculum will surely be a value addition for the students and it will satisfy the intellectuals within the group of learners and will add to the perspective of the learners. For each laboratory assignment, it is essential for students to draw/write/generate flowcharts, algorithms, test cases, mathematical models, Test data sets, and comparative/complexity analysis (as applicable). Batch size for practical and tutorials may be as per guidelines of authority.

Any 5 from List of Assignments.

e-Books:

1. http://nlp-iiith.vlabs.ac.in/

Practical No.	Assignment to be covered
Any 5 from List of Assignments	
1	Write a program for pre-processing of a text document such as stop word removal, stemming.
2	Implement a program for retrieval of documents using inverted files.
3	Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using the standard Heart Disease Data Set (You can use Java/Python ML library classes/API.
4	Implement e-mail spam filtering using text classification algorithm with appropriate dataset.
5	Implement Agglomerative hierarchical clustering algorithm using appropriate dataset.
6	Implement Page Rank Algorithm. (Use python or beautiful soup for implementation).
7	Build the web crawler to pull product information and links from an e-commerce website.

Assignment 1

Problem Statement:

Write a program for pre-processing of a text document such as stop word removal, stemming.

Objective:

To understand the concepts of information retrieval and web mining

Theory:

Text data derived from natural language is unstructured and noisy. Text preprocessing involves transforming text into a clean and consistent format that can then be fed into a model for further analysis and learning.

Text preprocessing techniques may be general so that they are applicable to many types of applications, or they can be specialized for a specific task. For example, the methods for processing scientific documents with equations and other mathematical symbols can be quite different from those for dealing with user comments on social media.

However, some steps, such as sentence segmentation, tokenization, spelling corrections, and stemming, are common to both.

Here's what you need to know about text preprocessing to improve your natural language processing (NLP).

The NLP Preprocessing Pipeline

A natural language processing system for textual data reads, processes, analyzes, and interprets text. As a first step, the system preprocesses the text into a more structured format using several different stages. The output from one stage becomes an input for the next—hence the name "preprocessing pipeline."

An NLP pipeline for document classification might include steps such as sentence segmentation, word tokenization, lowercasing, stemming or lemmatization, stop word removal, and spelling correction. Some or all of these commonly used text preprocessing stages are used in typical NLP systems, although the order can vary depending on the application.

Segmentation

Segmentation involves breaking up text into corresponding sentences. While this may seem like a trivial task, it has a few challenges. For example, in the English language, a period normally indicates the end of a sentence, but many abbreviations, including "Inc.," "Calif.," "Mr.," and "Ms.," and all fractional numbers contain periods and introduce uncertainty unless the end-of-sentence rules accommodate those exceptions.

Tokenization

The tokenization stage involves converting a sentence into a stream of words, also called “tokens.” Tokens are the basic building blocks upon which analysis and other methods are built.

Many NLP toolkits allow users to input multiple criteria based on which word boundaries are determined. For example, you can use a whitespace or punctuation to determine if one word has ended and the next one has started. Again, in some instances, these rules might fail. For example, *don’t*, *it’s*, etc. are words themselves that contain punctuation marks and have to be dealt with separately.

Change Case

Changing the case involves converting all text to lowercase or uppercase so that all word strings follow a consistent format. Lowercasing is the more frequent choice in NLP software.

Spell Correction

Many NLP applications include a step to correct the spelling of all words in the text.

Stop-Words Removal

"Stop words" are frequently occurring words used to construct sentences. In the English language, stop words include *is*, *the*, *are*, *of*, *in,* and *and*. For some NLP applications, such as document categorization, sentiment analysis, and spam filtering, these words are redundant, and so are removed at the preprocessing stage.

Stemming

The term *word stem* is borrowed from linguistics and used to refer to the base or root form of a word. For example, *learn* is a base word for its variants such as *learn, learns, learning,* and *learned.*

Stemming is the process of converting all words to their base form, or stem. Normally, a lookup table is used to find the word and its corresponding stem. Many search engines apply stemming for retrieving documents that match user queries. Stemming is also used at the preprocessing stage for applications such as emotion identification and text classification.

Lemmatization

Lemmatization is a more advanced form of stemming and involves converting all words to their corresponding root form, called “lemma.” While stemming reduces all words to their stem via a lookup table, it does not employ any knowledge of the parts of speech or the context of the word. This means stemming can’t distinguish which meaning of the word *right* is intended in the sentences “Please turn right at the next light” and “She is always right.”

The stemmer would stem *right* to *right* in both sentences; the lemmatizer would treat *right* differently based upon its usage in the two phrases.

A lemmatizer also converts different word forms or inflections to a standard form. For example, it would convert *less* to *little*, *wrote* to *write*, *slept* to *sleep*, etc.

A lemmatizer works with more rules of the language and contextual information than does a stemmer. It also relies on a dictionary to look up matching words. Because of that, it requires more processing power and time than a stemmer to generate output. For these reasons, some NLP applications only use a stemmer and not a lemmatizer.

Text Normalization

Text normalization is the preprocessing stage that converts text to a canonical representation. A common application is the processing of social media posts, where input text is shortened or words are spelled in different ways. For example, *hello* might be written as *hellooo* or *something* might appear as *smth*, and different people might choose to write *real time, real-time,* or *realtime.* Text normalization cleans the text and ideally replaces all words with their corresponding canonical representation. In the last example, all three forms would be converted to *realtime*. Many text normalization stages also replace emojis in text with a corresponding word. For example, *:-)* is replaced by *happy face*.

Parts of Speech Tagging

One of the more advanced text preprocessing techniques is parts of speech (POS) tagging. This step augments the input text with additional information about the sentence's grammatical structure. Each word is, therefore, inserted into one of the predefined categories such as a noun, verb, adjective, etc. This step is also sometimes referred to as grammatical tagging.

Conclusion:

By using above steps, we have performed pre-processing of a text document such as stop word removal, stemming successfully.

Oral Questions

1. What are the different NLTK libraries?
2. How to remove stop words from the file?
3. What is mean by stemming?
4. What is mean by Lemmatization?

Assignment 2

Problem Statement:

Implement a program for retrieval of documents using inverted files.

Objective:

1. Evaluate and analyse retrieved information
2. To study Indexing, Inverted Files and searching with the help of inverted file

Theory:

An inverted index is an index data structure storing a mapping from content, such as words or numbers, to its locations in a document or a set of documents. In simple words, it is a HashMap like data structure that directs you from a word to a document or a web page.

Creating Inverted Index

We will create a **Word level inverted index** that is it will return the list of lines in which the word is present. We will also create a dictionary in which key values represent the words present in the file and the value of a dictionary will be represented by the list containing line numbers in which they are present. To create a file in Jupiter notebook, use magic function:

```
%%writefile file.txt

This is the first word.

This is the second text, Hello! How are you?

This is the third, this is it now.
```

This will create a file named file.txt will the following content.

To read file:

- Python3

```
# this will open the file

file = open('file.txt', encoding='utf8')

read = file.read()

file.seek(0)

read
```

```
 # to obtain the

# number of lines

# in file

line = 1

for word in read:

    if word == '\n':

        line += 1

print("Number of lines in file is: ", line)

 # create a list to

# store each line as

# an element of list

array = []

for i in range(line):

    array.append(file.readline())

 array
```

```
Number of lines in file is: 3

['This is the first word.\n',

'This is the second text, Hello! How are you?\n',

'This is the third, this is it now.']
```

Functions used:

- **Open:** It is used to open the file.
- **read:** This function is used to read the content of the file.
- **seek(0):** It returns the cursor to the beginning of the file.

Remove punctuation:
Python3

```
punc = '''!()-[]{};:'"\, <>./?@#$%^&*_~'''

for ele in read:
    if ele in punc:
        read = read.replace(ele, " ")
  read
 # to maintain uniformity
read=read.lower()
read
```

Output:

```
'this is the first word \n
this is the second text hello how are you \n
this is the third this is it now '
```

Tokenize the data as individual words:

Apply linguistic preprocessing by converting each word in the sentences into tokens. Tokenizing the sentences help with creating the terms for the upcoming indexing operation.

Python3

```
def tokenize_words(file_contents):
    """
    Tokenizes the file contents.
     Parameters
    ----------
    file_contents : list
        A list of strings containing the contents of the file.
         Returns
    -------
    list
        A list of strings containing the contents of the file tokenized.

    """
    result = []
     for i in range(len(file_contents)):
        tokenized = []
         # print("The row is ", file_contents[i])
         # split the line by spaces
        tokenized = file_contents[i].split()
         result.append(tokenized)
     return result
```

Clean data by removing stopwords:

Stop words are those words that have no emotions associated with it and can safely be ignored without sacrificing the meaning of the sentence.

Python3

```
from nltk.tokenize import word_tokenize
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
 for i in range(1):
    # this will convert
    # the word into tokens
    text_tokens = word_tokenize(read)
 tokens_without_sw = [
    word for word in text_tokens if not word in stopwords.words()]
 print(tokens_without_sw)
```

Output:

```
['first', 'word', 'second', 'text', 'hello', 'third']
```

Create an **inverted index:**
Python3

```
dict = {}
 for i in range(line):
    check = array[i].lower()
    for item in tokens_without_sw:
         if item in check:
            if item not in dict:
                dict[item] = []
             if item in dict:
                dict[item].append(i+1)
 dict
```

Output:

```
{'first': [1],
'word': [1],
'second': [2],
'text': [2],
'hello': [2],
'third': [3]}
```

Conclusion:

By this way, we can perform retrieval of documents using inverted files.

Oral Questions?

1. What is mean by inverted index?
2. What are steps for creation of Inverted Index?
3. What are built in functions used for index creation?

Assignment 3

Problem Statement:

Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using the standard Heart Disease Data Set (You can use Java/Python ML library classes/API.

Objective:

1. Evaluate and analyse retrieved information.
2. To study Bayesian network model.

Theory:

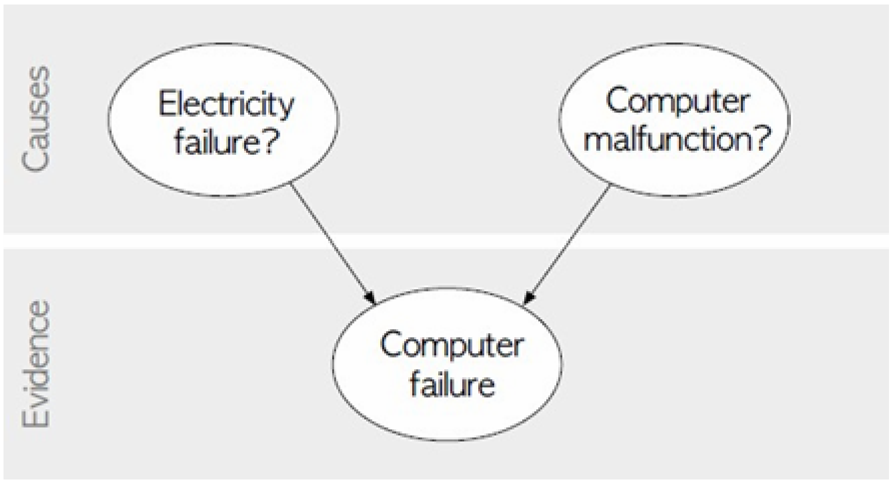
A Bayesian network is a directed acyclic graph in which each edge corresponds to a conditional dependency, and each node corresponds to a unique random variable.

Bayesian network consists of two major parts: a directed acyclic graph and a set of conditional probability distributions

- The directed acyclic graph is a set of random variables represented by nodes.
- The conditional probability distribution of a node (random variable) is defined for every possible outcome of the preceding causal node(s).

For illustration, consider the following example. Suppose we attempt to turn on our computer, but the computer does not start (observation/evidence). We would like to know which of the possible causes of computer failure is more likely. In this simplified illustration, we assume only two possible causes of this misfortune: electricity failure and computer malfunction.

The corresponding directed acyclic graph is depicted in below figure.



The goal is to calculate the posterior conditional probability distribution of each of the possible unobserved causes given the observed evidence, i.e., P [Cause | Evidence].

Data Set:

Title: Heart Disease Databases

The Cleveland database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date. The “Heartdisease” field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4.

Database:	0	1	2	3	4	Total
Cleveland:	164	55	36	35	13	303

<u>Attribute Information:</u>

1. age: age in years
2. sex: sex (1 = male; 0 = female)
3. cp: chest pain type
   1. Value 1: typical angina
   2. Value 2: atypical angina
   3. Value 3: non-anginal pain
   4. Value 4: asymptomatic
4. trestbps: resting blood pressure (in mm Hg on admission to the hospital)
5. chol: serum cholestoral in mg/dl
6. fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
7. restecg: resting electrocardiographic results
   1. Value 0: normal
   2. Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
   3. Value 2: showing probable or definite left ventricular hypertrophy by Estes’ criteria
8. thalach: maximum heart rate achieved
9. exang: exercise induced angina (1 = yes; 0 = no)
10. oldpeak = ST depression induced by exercise relative to rest
11. slope: the slope of the peak exercise ST segment
    1. Value 1: upsloping
    2. Value 2: flat
    3. Value 3: downsloping
12. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
13. Heartdisease: It is integer valued from 0 (no presence) to 4.

Some instance from the dataset:

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	Heart disease
63	1	1	145	233	1	2	150	o	2.3	3	o	6	o
67	1	4	160	286	o	2	108	1	1.5	2	3	3	2
67	1	4	120	229	o	2	129	1	2.6	2	2	7	1

41	o	2	130	204	o	2	172	o	1.4	1	o	3	o
62	o	4	140	268	o	2	160	o	3.6	3	2	3	3
60	1	4	130	206	o	2	132	1	2.4	2	2	7	4

Python Program to Implement and Demonstrate Bayesian network using pgmpy Machine Learning

```
import numpy as np

import pandas as pd

import csv

from pgmpy.estimators import MaximumLikelihoodEstimator

from pgmpy.models import BayesianModel

from pgmpy.inference import VariableElimination

#read Cleveland Heart Disease data

heartDisease = pd.read_csv('heart.csv')

heartDisease = heartDisease.replace('?',np.nan)

#display the data

print('Sample instances from the dataset are given below')

print(heartDisease.head())

print('\n Attributes and datatypes')

print(heartDisease.dtypes)

#Model Bayesian Network
```

```
model=BayesianModel([('age','heartdisease'),('sex','heartdisease'),('exang','heartdisease'),('cp','heartdisea
se'),('heartdisease','restecg'),('heartdisease','chol')])

#Learning CPDs using Maximum Likelihood Estimators

print('\nLearning CPD using Maximum likelihood estimators')

model.fit(heartDisease,estimator=MaximumLikelihoodEstimator)

# Inferencing with Bayesian Network

print('\n Inferencing with Bayesian Network:')

HeartDiseasetest_infer = VariableElimination(model)

#computing the Probability of HeartDisease given Age

print('\n 1. Probability of HeartDisease given evidence= restecg')

q1=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'restecg':1})

print(q1)

#computing the Probability of HeartDisease given cholesterol

print('\n 2. Probability of HeartDisease given evidence= cp ')

q2=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'cp':2})

print(q2)
```

Output

```
Learning CPD using Maximum likelihood estimators

 Inferencing with Bayesian Network:

 1. Probability of HeartDisease given evidence= restecg

   +-----------------+---------------------+
   | heartdisease    |   phi(heartdisease) |
   +=================+=====================+
   | heartdisease(0) |              0.1012 |
   +-----------------+---------------------+
   | heartdisease(1) |              0.0000 |
   +-----------------+---------------------+
   | heartdisease(2) |              0.2392 |
   +-----------------+---------------------+
   | heartdisease(3) |              0.2015 |
   +-----------------+---------------------+
   | heartdisease(4) |              0.4581 |
   +-----------------+---------------------+
2. Probability of HeartDisease given evidence= cp

   +-----------------+---------------------+
   | heartdisease    |   phi(heartdisease) |
   +=================+=====================+
   | heartdisease(0) |              0.3610 |
   +-----------------+---------------------+
   | heartdisease(1) |              0.2159 |
   +-----------------+---------------------+
   | heartdisease(2) |              0.1373 |
   +-----------------+---------------------+
   | heartdisease(3) |              0.1537 |
   +-----------------+---------------------+
   | heartdisease(4) |              0.1321 |
   +-----------------+---------------------+
```

Conclusion:

In this way, we have successfully constructed a Bayesian network considering medical data. We have Use this created model to demonstrate the diagnosis of heart patients using the standard Heart Disease Data Set

Oral Question:

1. What is Bayesian network model?
2. What are the attributes used in Heart Disease Data Set?
3. What is pgmpy?

Assignment 4

Problem Statement:

Implement e-mail spam filtering using text classification algorithm with appropriate dataset.

Objective:

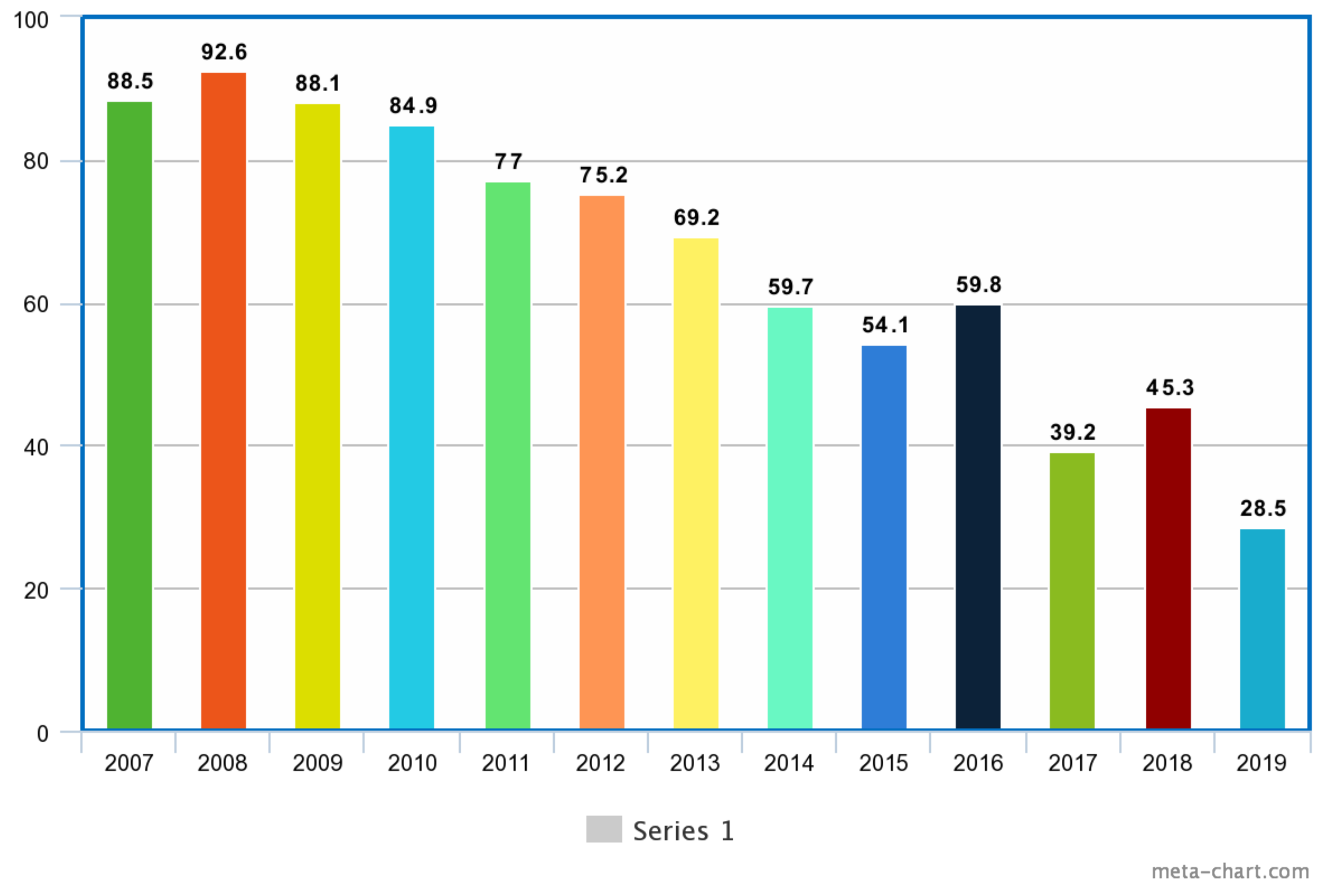
1. Basic concepts of Spam Filtering.
2. To study KNN algorithm.

Theory:

In the new era of technical advancement, electronic mail (emails) has gathered significant users for professional, commercial, and personal communications. In 2019, on average, every person received **130 emails each day,** and overall, **296 Billion** emails were sent that year.

Because of the high demand and huge user base, there is an upsurge in unwanted emails, also known as spam emails. At times, more than **50% of the total emails were spam**. Even today, people lose millions of dollars to fraud every day.

But, in the figure shown below, it can be observed that the quantity of such emails has decreased significantly after 2016 because of the evolution of the software that can detect these spam emails and can filter them out.



Percentage of emails marked as Spam (Source: Statista)

Key takeaways from this article

- What are the different methods used to segregate incoming emails into a spam or non-spam categories?
- Steps to implement a Spam-classifier using the k-NN algorithm.
- How to evaluate the performance of the model formed?
- Use-case of Gmail, Outlook, and Yahoo. How do these companies classify emails?
- Possible interview questions on this machine learning application.

Many several techniques are present in the market to detect spam emails. If we want to classify broadly, there are five different techniques based on which algorithms decide whether any mail is spam.

Content-Based Filtering Technique

Algorithms analyze words, the occurrence of words, and the distribution of words and phrases inside the content of emails and segregate them into spam and non-spam categories.

Case Base Spam Filtering Method

Algorithms trained on well-annotated spam/non-spam marked emails try to classify incoming emails into two categories.

Heuristic or Rule-Based Spam Filtering Technique

Algorithms use pre-defined rules as regular expressions to give a score to the messages in the emails. They segregate emails into spam and non-spam categories based on the scores generated.

The Previous Likeness Based Spam Filtering Technique

Algorithms extract the incoming mails' features and create a multi-dimensional space vector and draw points for every new instance. Based on the KNN algorithm, these new points get assigned to the closest class of spam and non-spam.

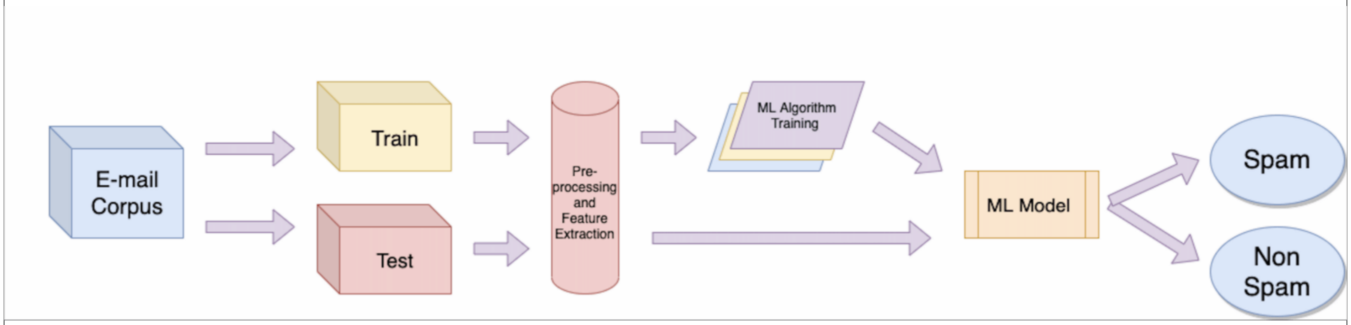
Adaptive Spam Filtering Technique

Algorithms classify the incoming emails into various groups and, based on the comparison scores of every group with the defined set of groups, spam and non-spam emails got segregated.

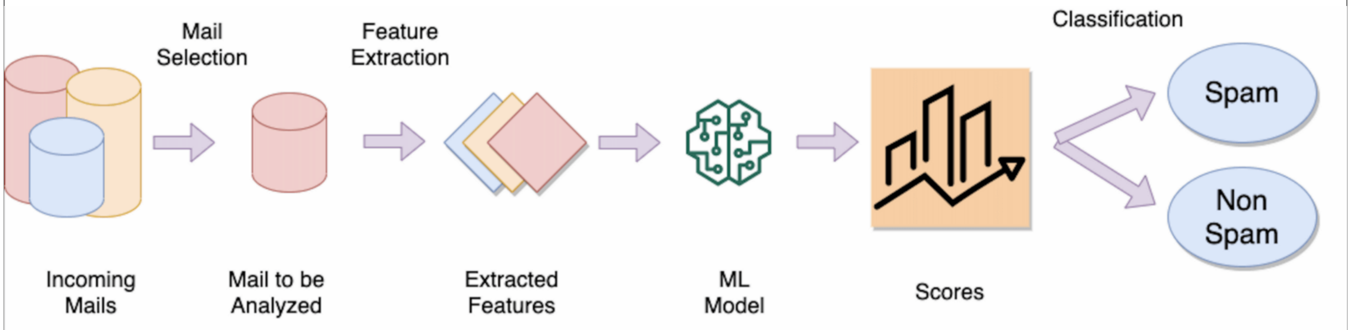
This article will give an idea for implementing content-based filtering using one of the most famous spam detection algorithms, K-Nearest Neighbour (KNN)**.**

K-NN based algorithms are widely used for clustering tasks. Let's quickly know the entire architecture of this implementation first and then explore every step. Executing these **five steps,** one after the other, will help us implement our spam classifier smoothly.

Training Testing Phase



New Email Classification



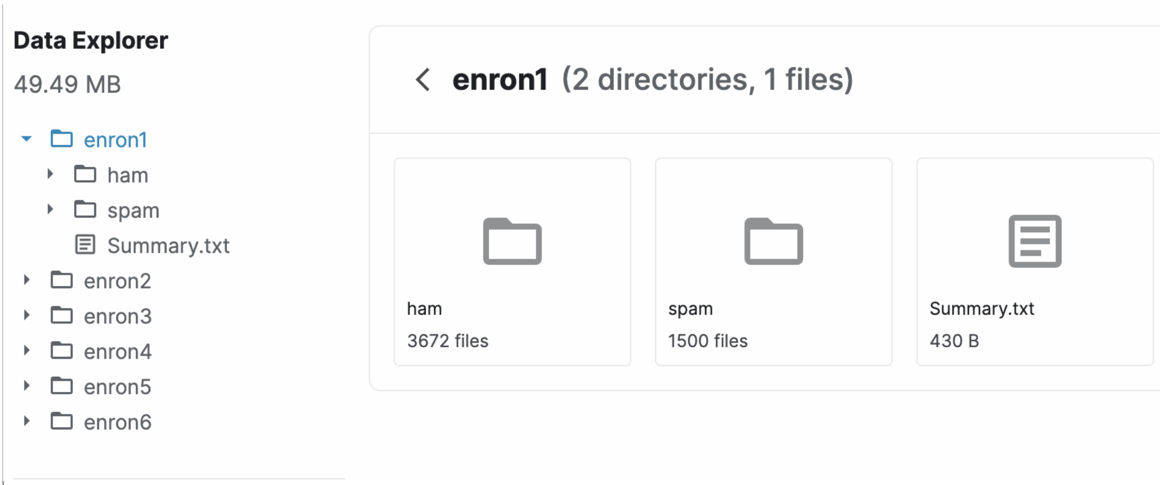
Step 1: E-mail Data Collection

The dataset contained in a corpus plays a crucial role in assessing the performance of any spam filter. Many open-source datasets are freely available in the public domain. The two datasets below are widely popular as they contain many emails.

1. Enron corpus datasets (Created in 2006 and having 55% spam emails)
2. Trec 2007 dataset (Created in 2007 and having 67% spam emails)

Train/Test Split: Split the dataset into train and test datasets but make sure that both sets balance the numbers of ham and spam emails (ham is a fancy name for non-spam emails).

Enron Corpus Dataset on Kaggle



Step 2: Pre-processing of E-mail content

As the dataset is in text format, we will need the pre-processing of text data. At this step, we mainly perform **tokenization** of mail. Tokenization is a process where we break the content of an email into words and transform big messages into a sequence of representative symbols termed **tokens**. These tokens are extracted from the email body, header, subject, and image.

Extracting words from images (For a simple implementation, this can be ignored)

These days, senders have options to attach inline images to the mail. These emails can be categorized as spam, not based on their mail content but on the images' content. This was not an easy task until google came up with the open-source library Tesseract**.** This library extracts the words from images automatically with certain accuracy. But still, Times New Roman and Captcha words are challenging to read automatically.

Step 3: Feature Extraction and Selection

After pre-processing, we can have a large number of words. Here we can maintain a database that contains the frequency of the different words represented in each column. These attributes can be categorized on another basis, like:

- **Essential attributes:** Frequency of repeated words, Number of semantic discrepancies, an Adult content bag of words, etc.
- **Additional Attributes:** Sender account features like Sender country, IP address, email, age of Sender, Number of replies, Number of recipients, and website address. **Note:** These web addresses are converted in the word format only. For example, https://www.google.com/ can be converted to "HTTP google." Sometimes these processes are called Normalization.
- **Less important attributes:** Geographical distance between sender and receiver, Sender's date of birth, Account lifespan, Sex of Sender, and Age of the recipient.

You must be clear that the more the number of attributes, → more the time complexity of the model.

These attributes can be tremendous, so techniques like Stemming, noise removal, and stop-word removal can be used. One of the famous stemming algorithms is the **Porter-Stemmer Algorithm.** Some general things that we do in stemming are:

- Removing suffixes (-ed, -ing, -full, -ness, etc.)
- Removing prefixes (Un-, Re-, Pre-, etc.)

List of stop words

```
["i", "me", "my","him", "his", "himself", "she", “her", “hers", "herself",

"it", "its", "itself", "they", "them", "their", "theirs","we",

"myself", our", “ours", "ourselves", "you", "your",

"yours", "yourself", "yourselves", "he",

"themselves", "what", "which", "who", "whom", "this",

"that", "these", "those", "am", "is", "are", "was", "were", "be",

an", "the", "and", "but", "if", "or",

"because", “as", “until", "while", "of", "at", "by", "for",

"with", "about", "against", "between", "into", "through",

"been", "being", "have", "has", "had", "having", "do", "does",

"did", "doing", "a", "out", "on", "off", "over", “under”,

"both", "each", "few","during", "before", "after", "above",

"below", "to", "from", "up", "down",

"again", "further", "then", "once", "here", "there", "when",
```

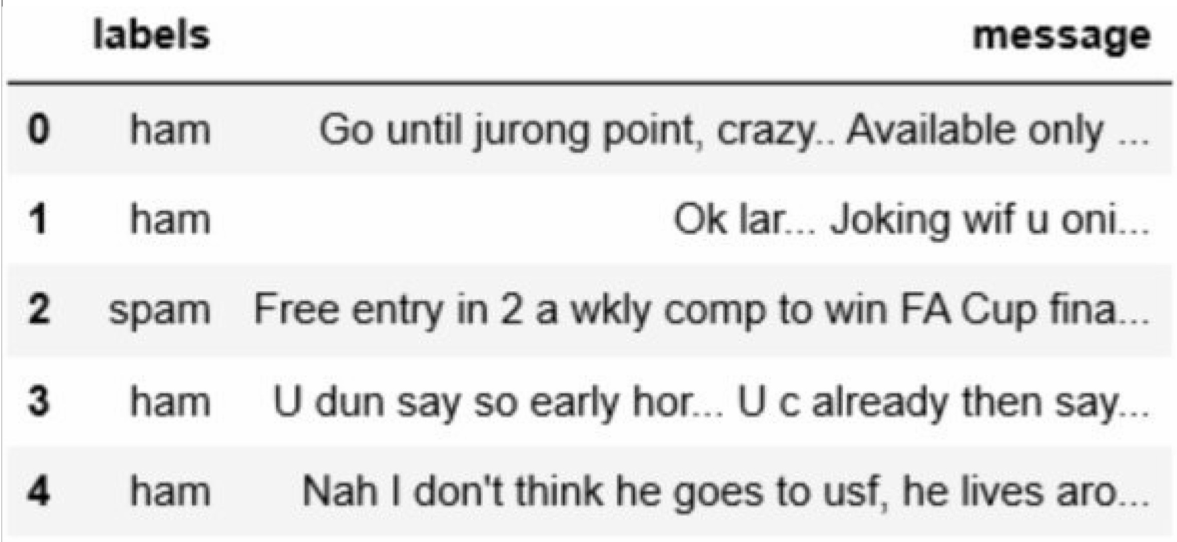
"where", "why' “how' "all", "any",

"more", "most", “other", "some", "such", "no", “nor”,

"not", “only", "own", "same", "so", "than", "too", "very",

"can", "will", "just", "don", "should", “now"]

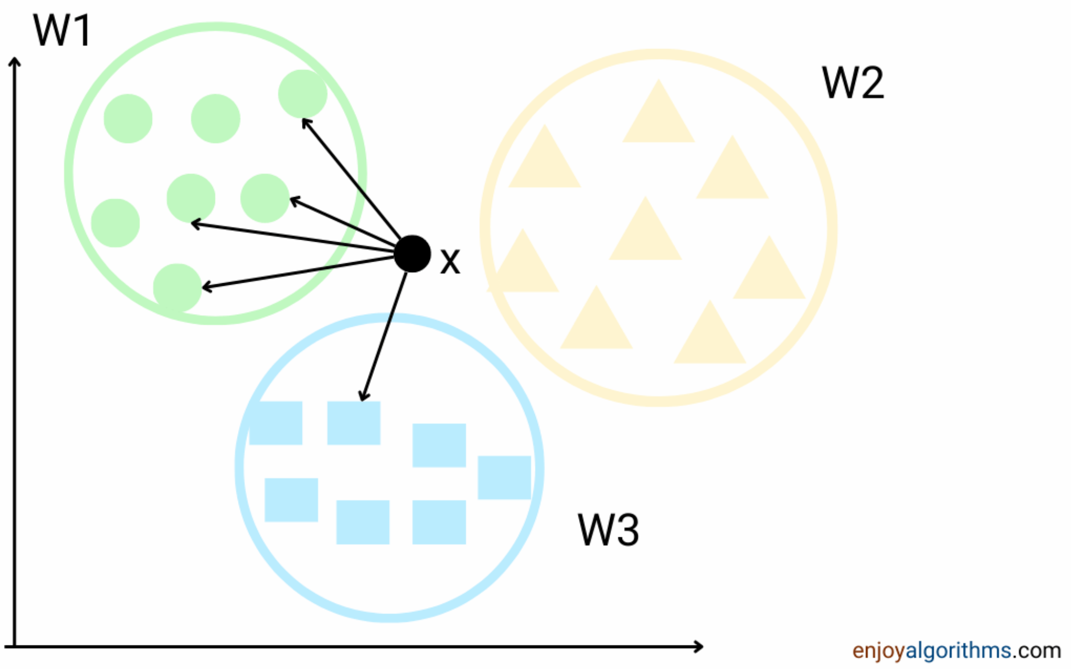
Example dataset format



	labels	message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Step 4: KNN (K-Nearest Neighbour) Implementation

Similar to the Nearest Neighbour algorithm, the K-Nearest Neighbour algorithm serves the purpose of clustering. Still, instead of giving just one nearest instance, it looks at the closest K instances to the new incoming instance. K-NN classifies the new instances based on the frequency of those K instances. The value of K is considered to be a hyperparameter that needs tuning. To tune this, one can take one of the famous Hit and Trial approaches, where we try some K values and then check the model's performance.



To find the nearest instance, one can use the Euclidean distance. One can use the Scikit-learn library to implement the K-NN algorithm for this task.

$$Euclidean = \sqrt{\Sigma_i^k (X_i - Y_i)^2}$$

Step 5: Performance Analysis

Our algorithm is ready, so we must check the model's performance. **Even a single missed important message may cause a user to reconsider the value of spam filtering.** So we must be sure that our algorithm will be as close to 100% accurate. But some researchers feel that more than accuracy as the evaluation parameter for spam classification is needed.

According to the below table (also known as the confusion matrix), we must evaluate our spam classification model based on **four different parameters.**

- **Accuracy : (TP + TN)/(TP + FP + FN + TN)**
- **Precision: TP / (TP + FP)**
- **Sensitivity: TP / (TP + FN)**
- **Specificity: TN / (TN + FP)**

More advanced algorithms are available for this classification, but you can easily achieve more than 90% accuracy using k-NN-based implementation.

Gmail, Yahoo, and Outlook Case Study

Gmail

Google data centers use thousands of rules to filter spam emails. They provide the weightage to different parameters; based on that, they filter the emails. Google's spam classifier is said to be a **state of an art technique** that uses various techniques like Optical character recognition, linear regression, and a combination of multiple neural networks.

Yahoo

Yahoo mail is the world's first free webmail service provider, with more than 320 million active users. They have their filtering techniques to categorize emails. Yahoo's basic methods are URL filtering, email content, and user spam complaints. **Unlike Gmail, Yahoo filter emails messages by domain and not the IP address.** Yahoo also provides users with custom-filtering options to send mail directly to junk folders.

Outlook

Microsoft-owned mailing platform widely used among professionals. In 2013, Microsoft renamed Hotmail and Windows Live Mail to Outlook. At present, outlook has more than 400 Million active users. Outlook has its distinctive feature based on which it filters every incoming mail. Based on their official website, they have provided the list of spam filters they use to send any mail in the junk folder, which includes :

- Safe Senders list
- Safe Recipients list
- Blocked Senders list
- Blocked Top-Level Domains list
- Blocked Encodings list

Conclusion

In terms of the Number of spam emails sent daily and the Number of money people lose every day because of these spam scams, Spam-filtering becomes the primary need for all email-providing companies. This article discussed the complete process of spam email filtering using advanced machine learning technologies. We also have closed one possible way of implementing our spam classifier using one of the most famous algorithms, k-NN. We also discussed the case studies of well-known companies like Gmail, Outlook, and Yahoo to review how they use ML and AI techniques to filter such spammers.

Oral Questions

- What is Porter Stemmer's Algorithm?
- Why did you use the k-NN algorithm for this problem?

- Is this supervised learning or unsupervised learning?
- What are the different algorithms that can replace k-NN here?
- What steps can be taken to improve accuracy further?

Assignment 5

Problem Statement:

Implement Agglomerative hierarchical clustering algorithm using appropriate dataset.

Objective:

- Evaluate and analyse retrieved information using Page Ranking algorithm.
- To study Random Walk.

Theory:

Prerequisites:
Agglomerative Clustering Agglomerative Clustering is one of the most common hierarchical clustering techniques.
Dataset – Credit Card Dataset.

Assumption: The clustering technique assumes that each data point is similar enough to the other data points that the data at the starting can be assumed to be clustered in 1 cluster.

Step 1: Importing the required libraries

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.decomposition import PCA

from sklearn.cluster import AgglomerativeClustering

from sklearn.preprocessing import StandardScaler, normalize

from sklearn.metrics import silhouette_score

import scipy.cluster.hierarchy as shc
```

Step 2: Loading and Cleaning the data

- Python3

```
# Changing the working location to the location of the file

cd C:\Users\Dev\Desktop\Kaggle\Credit_Card

 X = pd.read_csv('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)
```

Step 3: Preprocessing the data

- Python3

```
# Scaling the data so that all the features become comparable

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)
```

Step 4: Reducing the dimensionality of the Data

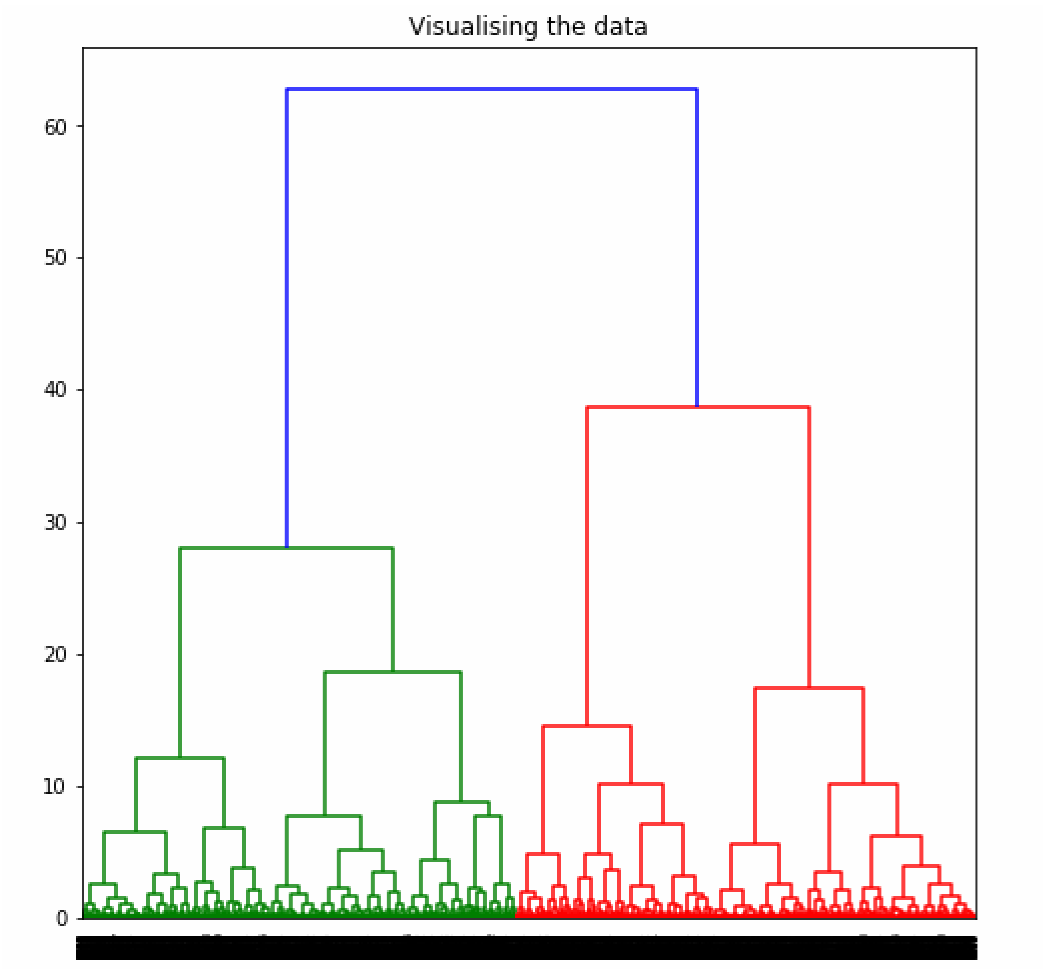
- Python3

```
pca = PCA(n_components = 2)

X_principal = pca.fit_transform(X_normalized)

X_principal = pd.DataFrame(X_principal)

X_principal.columns = ['P1', 'P2']
```

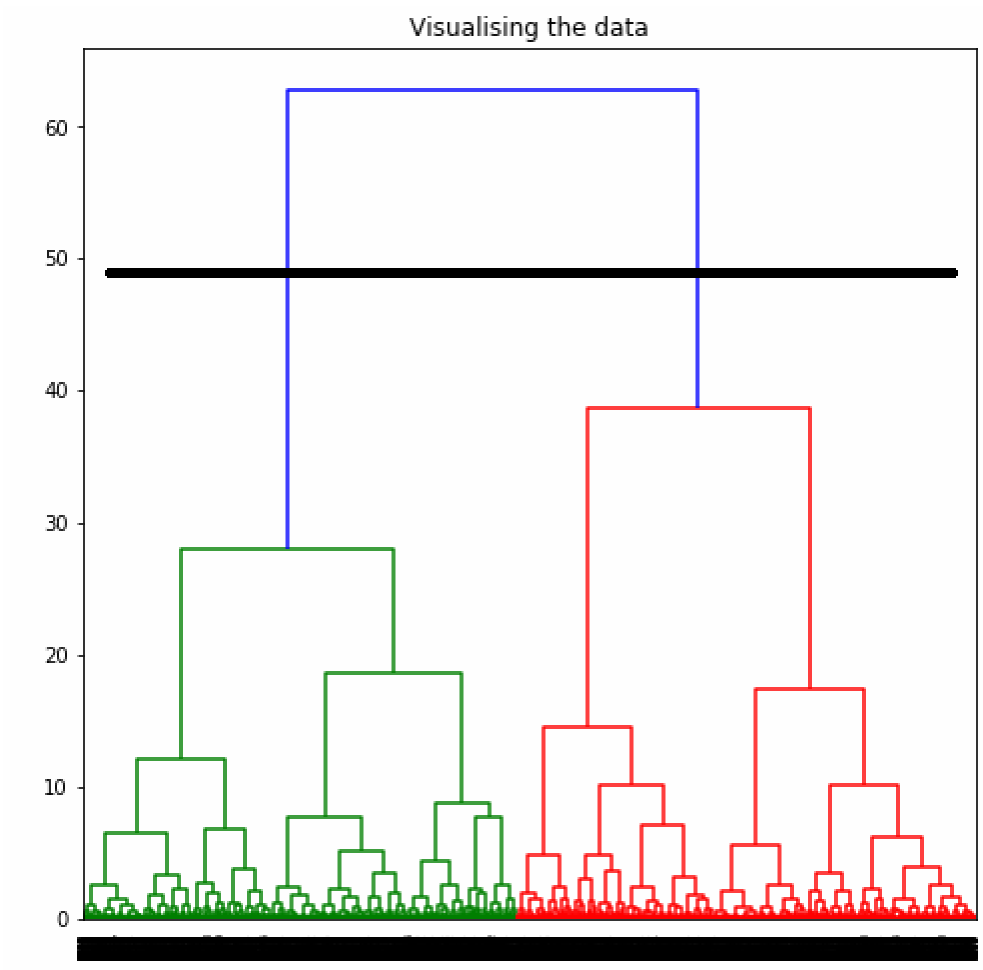
Dendrograms are used to divide a given cluster into many different clusters. **Step 5: Visualizing the working of the Dendrograms**

- Python3

```
plt.figure(figsize =(8, 8))

plt.title('Visualising the data')

Dendrogram = shc.dendrogram((shc.linkage(X_principal, method ='ward')))
```



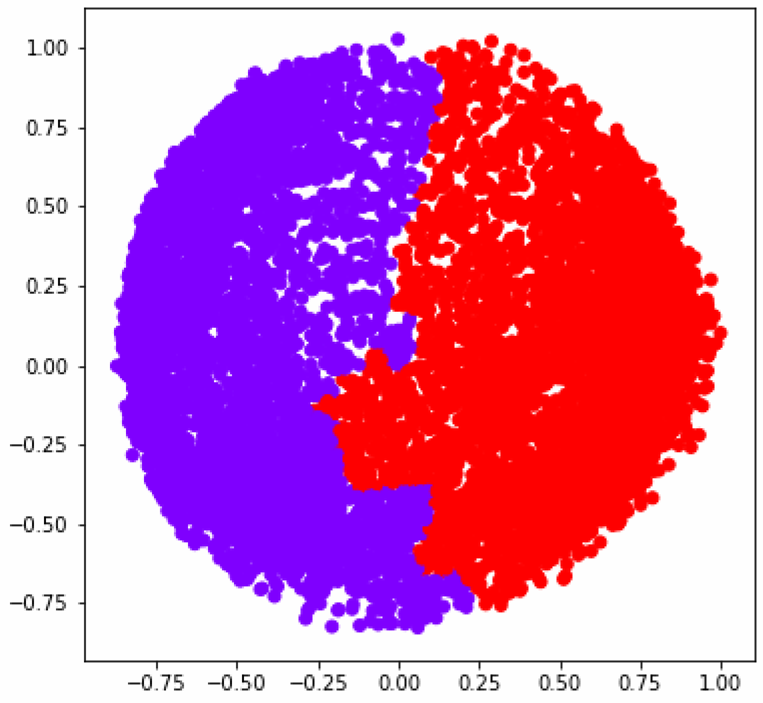
To determine the optimal number of clusters by visualizing the data, imagine all the horizontal lines as being completely horizontal and then after calculating the maximum distance between any two horizontal lines, draw a horizontal line in the maximum distance calculated.



The above image shows that the optimal number of clusters should be 2 for the given data. **Step 6: Building and Visualizing the different clustering models for different values of k** a) **k = 2**

- Python3

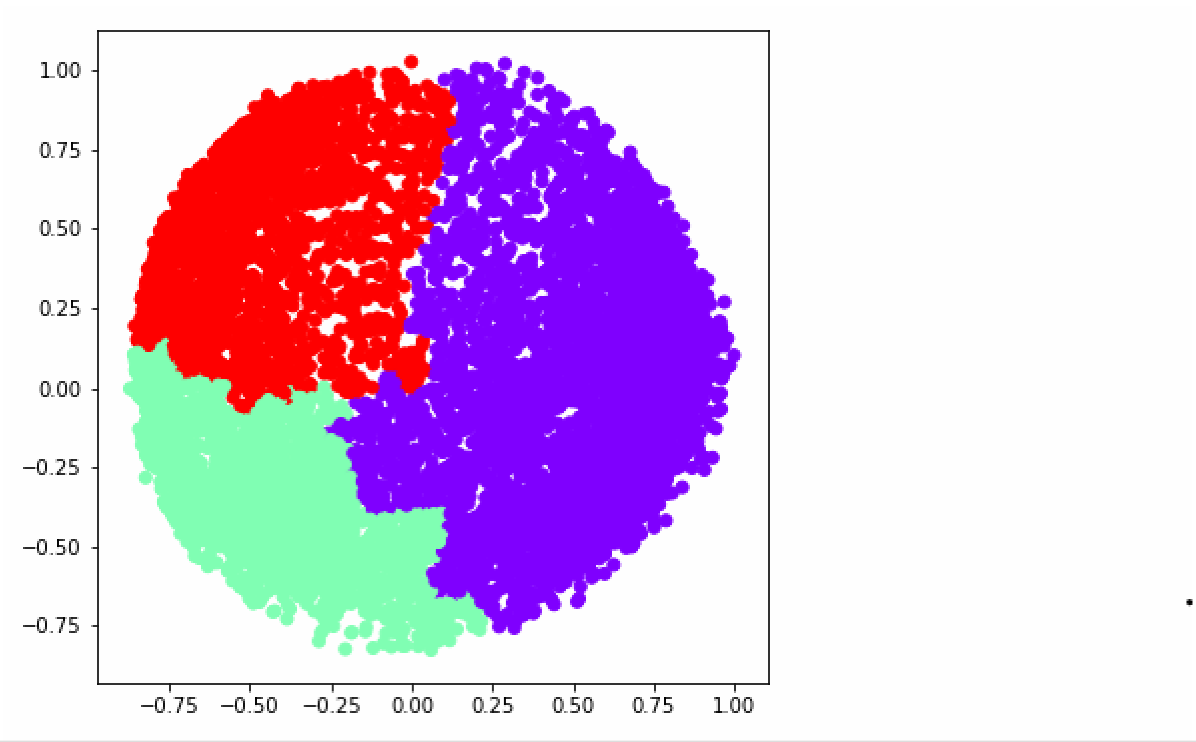
```
ac2 = AgglomerativeClustering(n_clusters = 2)

 # Visualizing the clustering

plt.figure(figsize =(6, 6))

plt.scatter(X_principal['P1'], X_principal['P2'],

          c = ac2.fit_predict(X_principal), cmap ='rainbow')

plt.show()
```



b) **k = 3**

- Python3

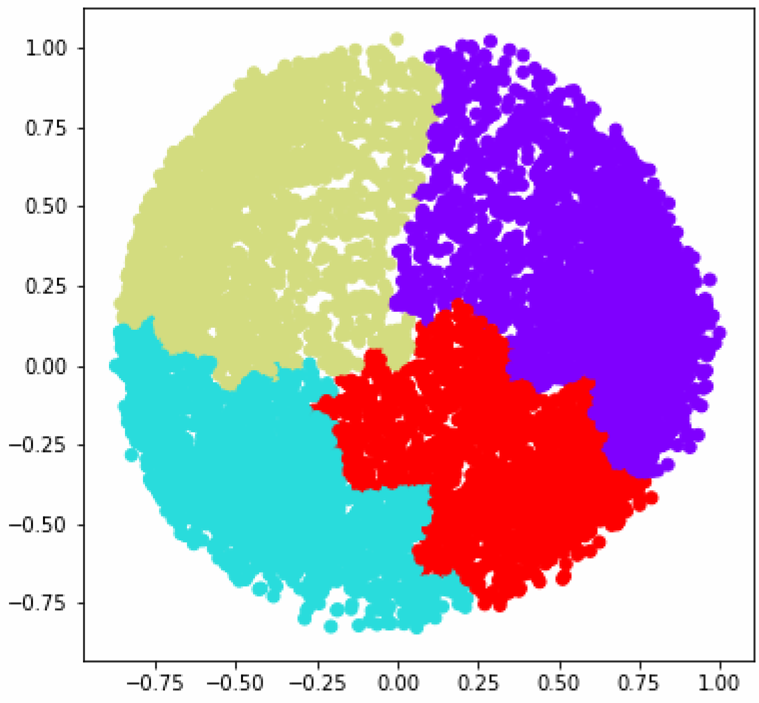
```
ac3 = AgglomerativeClustering(n_clusters = 3)


plt.figure(figsize =(6, 6))
plt.scatter(X_principal['P1'], X_principal['P2'],
            c = ac3.fit_predict(X_principal), cmap ='rainbow')
plt.show()
```



c) **k = 4**

- Python3

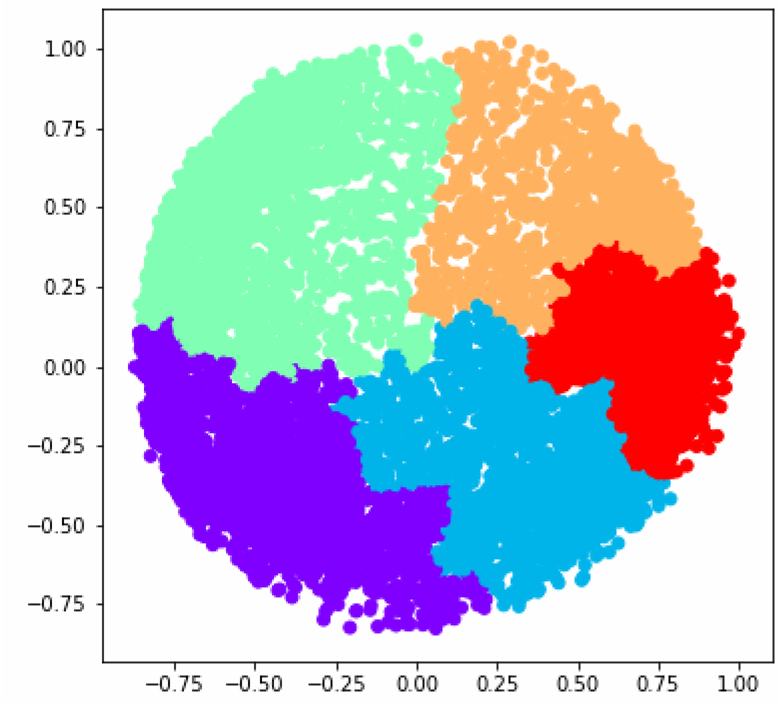
```
ac4 = AgglomerativeClustering(n_clusters = 4)


plt.figure(figsize =(6, 6))
plt.scatter(X_principal['P1'], X_principal['P2'],
            c = ac4.fit_predict(X_principal), cmap ='rainbow')
plt.show()
```



d) **k = 5**

- Python3

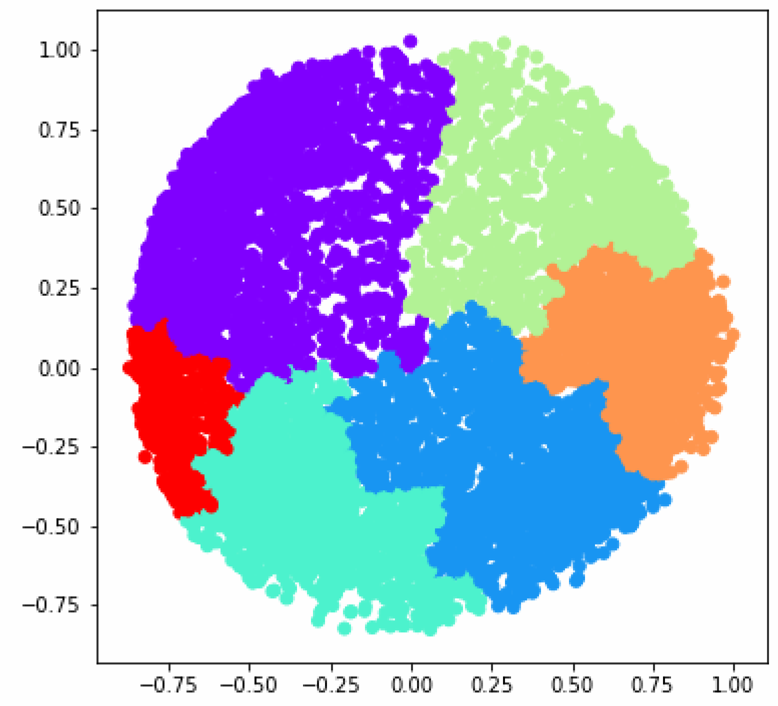
```
ac5 = AgglomerativeClustering(n_clusters = 5)


plt.figure(figsize =(6, 6))

plt.scatter(X_principal['P1'], X_principal['P2'],

           c = ac5.fit_predict(X_principal), cmap ='rainbow')

plt.show()
```



e) **k = 6**

- Python3

```
ac6 = AgglomerativeClustering(n_clusters = 6)


plt.figure(figsize =(6, 6))
plt.scatter(X_principal['P1'], X_principal['P2'],
            c = ac6.fit_predict(X_principal), cmap ='rainbow')
plt.show()
```

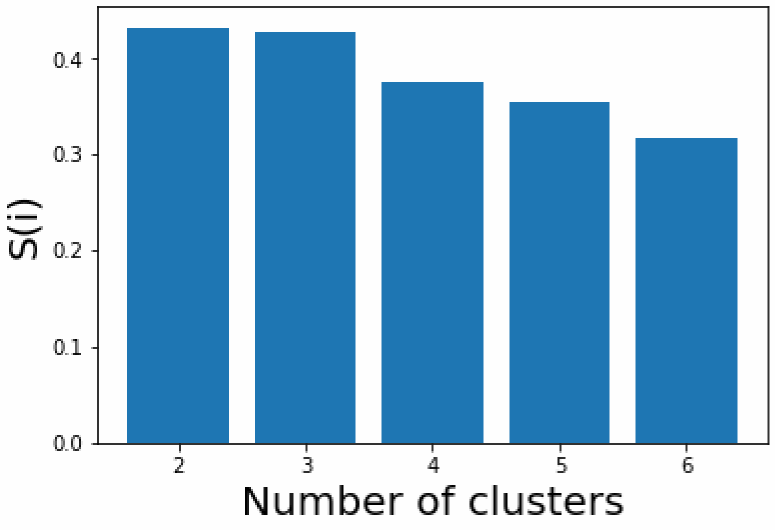


We now determine the optimal number of clusters using a mathematical technique. Here, We will use the **Silhouette Scores** for the purpose.

Step 7: Evaluating the different models and Visualizing the results.

- Python3

```
k = [2, 3, 4, 5, 6]


# Appending the silhouette scores of the different models to the list

silhouette_scores = []

silhouette_scores.append(

        silhouette_score(X_principal, ac2.fit_predict(X_principal)))

silhouette_scores.append(

        silhouette_score(X_principal, ac3.fit_predict(X_principal)))

silhouette_scores.append(

        silhouette_score(X_principal, ac4.fit_predict(X_principal)))

silhouette_scores.append(

        silhouette_score(X_principal, ac5.fit_predict(X_principal)))

silhouette_scores.append(
        silhouette_score(X_principal, ac6.fit_predict(X_principal)))


# Plotting a bar graph to compare the results

plt.bar(k, silhouette_scores)

plt.xlabel('Number of clusters', fontsize = 20)

plt.ylabel('S(i)', fontsize = 20)

plt.show()
```



Thus, with the help of the silhouette scores, it is concluded that the optimal number of clusters for the given data and clustering technique is 2.

Conclusion:

In this way, we have successfully completed implementation of Agglomerative hierarchical clustering algorithm using appropriate dataset.

Oral Questions:

1. What is mean by Agglomerative hierarchical clustering algorithm?
2. Which is the readymade function available to build Agglomerative hierarchical clustering algorithm?
3. Tell me the steps to implement Agglomerative hierarchical clustering algorithm.
4. Applications of Agglomerative hierarchical clustering algorithm.

Assignment 6

Problem Statement:

Implement Page Rank Algorithm. (Use python or beautiful soup for implementation)

Objective:

- Evaluate and analyze retrieved information using Page Ranking algorithm.
- To study Random Walk.

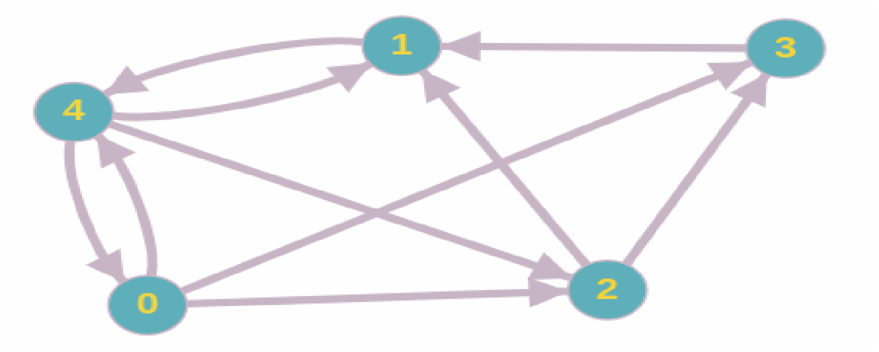
Theory:

Prerequisite: Page Rank Algorithm and Implementation, Random Walk.

In Social Networks page rank is a very important topic. Basically, page rank is nothing but how webpages are ranked according to its importance and relevance of search. All search engines use page ranking. Google is the best example that uses page rank using the web graph.

Random Walk

The web can be represented like a directed graph where nodes represent the web pages and edges form links between them. Typically, if a node (web page) ***i*** is linked to a node ***j***, it means that ***i*** refers to ***j***.



Example of a directed graph

We have to define what is the importance of a web page. As a first approach, we could say that it is the total number of web pages that refer to it. If we stop to these criteria, the importance of these web pages that refer to it is not taken into account. In other words, an important web page and a less important one has the same weight. Another approach is to assume that a web page spread its importance equally to all web pages it links to. By doing that, we can then define the score of a node ***j*** as follows:

$$r_j = \sum_{i \to j} \frac{r_i}{d_i}$$

where r_i is the score of the node ***i*** and d_i its out-degree.

From the example above, we can write this linear system:

$$\begin{cases} r_0 = \frac{r_4}{3} \\ r_1 = \frac{r_2}{2} + \frac{r_4}{3} + r_3 \\ r_2 = \frac{r_0}{3} + \frac{r_4}{3} \\ r_3 = \frac{r_2}{2} + \frac{r_0}{3} \\ r_4 = \frac{r_0}{3} + r_1 \end{cases}$$

By passing the right-hand side of this linear system into the left-hand side, we get a new linear system that we can solve by using Gaussian elimination. But this solution is limited for small graphs. Indeed, as this kind of graphs are sparse and Gauss elimination modifies the matrix when performing its operations, we lose the sparsity of the matrix and it would take more memory space. In the worst case, the matrix can no longer be stored.

Markov Chain and PageRank

Since a Markov Chain is defined by an initial distribution and a transition matrix, the above graph can be seen as a Markov chain with the following transition matrix:

$$P = \begin{pmatrix} 0 & 0 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & \frac{1}{2} & 1 & \frac{1}{3} \\ \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & 1 & 0 & 0 & 0 \end{pmatrix}$$

Transition matrix corresponding to our example

We notice that ***P*** transpose is row stochastic which is a condition to apply Markov chain theorems.
For the initial distribution, let's consider that it is equal to :

$$\pi = \left(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}, \frac{1}{n}\right)$$

where ***n*** is the total number of nodes. This means that the random walker will choose randomly the initial node from where it can reach all other nodes.

At every step, the random walker will jump to another node according to the transition matrix. the probability distribution is then computed for every step. This distribution tells us where the random walker is likely to be after a certain number of steps. The probability distribution is computed using the following equation:

$$\pi^{(t+1)} = P\pi^{(t)}$$

A stationary distribution of a Markov chain is a probability distribution ***π*** with ***π = Pπ***. This means that the distribution will not change after one step. It is important to note that not all Markov chains admit a stationary distribution.

If a Markov chain is strongly connected, which means that any node can be reached from any other node, then it admits a stationary distribution. It is the case in our problem. So, after an infinitely long walk, we know that the probability distribution will converge to a stationary distribution ***π***.

All we have to do is solving this equation:

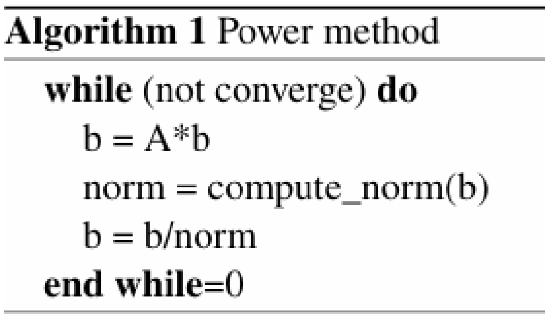
$$\pi = P\pi$$

We notice that ***π*** is an eigenvector of the matrix ***P*** with the eigenvalue ***1***. Instead of computing all eigenvectors of ***P*** and select the one which corresponds to the eigenvalue ***1***, we use the ***Frobenius-Perron*** theorem.

According to ***Frobenius-Perron*** theorem, if a matrix ***A*** is a square and positive matrix (all its entries are positive), then it has a positive eigenvalue ***r***, such as $|\lambda| < r$, where λ is an eigenvalue of A. The eigenvector ***v*** of ***A*** with eigenvalue ***r*** is positive and is the unique positive eigenvector.

In our case, the matrix ***P*** is positive and square. The stationary distribution ***π*** is necessarily positive because it is a probability distribution. We conclude that ***π*** is the dominant eigenvector of ***P*** with the dominant eigenvalue ***1***.

To compute ***π***, we use the power method iteration which is an iterative method to compute the dominant eigenvector of a given matrix ***A***. From an initial approximation of the dominant eigenvector ***b*** that can be initialized randomly, the algorithm will update it until convergence using the following algorithm:



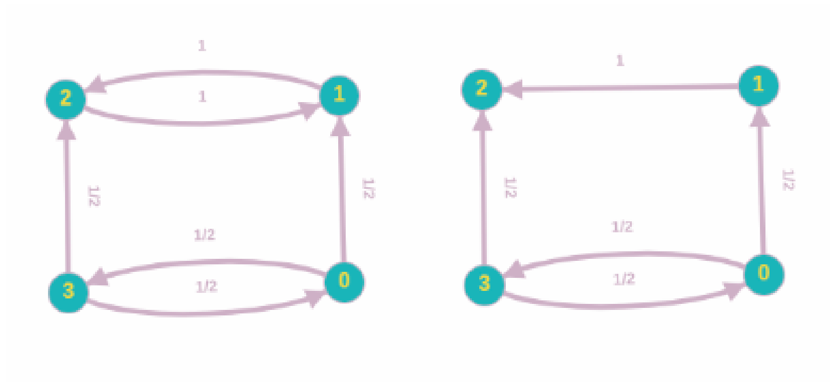
```
Algorithm 1 Power method
    while (not converge) do
        b = A*b
        norm = compute_norm(b)
        b = b/norm
    end while=0
```

Power method algorithm

As mentioned before, the probability distribution at time ***t*** defines the probability that the walker will be in a node after ***t*** steps. It means that the higher the probability, the more important is the node. We can then rank our web pages according to the stationary distribution we get using the power method.

Teleportation and damping factor

In the web graph, for example, we can find a web page ***i*** which refers only to web page ***j*** and ***j*** refers only to ***i***. This is what we call ***spider trap problem***. We can also find a web page which has no outlink. It is commonly named ***Dead end***.



Dead ends and spider traps illustration

In the case of a spider trap, when the random walker reaches the node ***1*** in the above example, he can only jump to node ***2*** and from node ***2***, he can only reach node ***1***, and so on. The importance of all other nodes will be taken by nodes ***1*** and ***2***. In the above example, the probability distribution will converge to ***π = (0, 0.5, 0.5, 0)***. This is not the desired result.

In the case of Dead ends, when the walker arrives at node ***2***, it can't reach any other node because it has no outlink. The algorithm cannot converge.

To get over these two problems, we introduce the notion of ***teleportation***.

Teleportation consists of connecting each node of the graph to all other nodes. The graph will be then complete. The idea is with a certain probability ***β***, the random walker will jump to another node according to the transition matrix ***P*** and with a probability ***(1-β)/n***, it will jump randomly to any node in the graph. We get then the new transition matrix ***R***:

$$R = \beta P + (1 - \beta)) v e^T$$

where ***v*** is a vector of ones, and ***e*** a vector of ***1/n***

$$e^T = (\tfrac{1}{n}, \ldots, \tfrac{1}{n}).$$

$$v = (1, \ldots, 1)^T ;$$

β is commonly defined as the ***damping factor***. In practice, it is advised to set ***β*** to ***0.85***.

By applying teleportation in our example, we get the following new transition matrix:

$$R = \begin{pmatrix} \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1}{3}\beta + \frac{1-\beta}{5} \\ \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1}{2}\beta + \frac{1-\beta}{5} & \beta + \frac{1-\beta}{5} & \frac{1}{3}\beta + \frac{1-\beta}{5} \\ \frac{1}{3}\beta + \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1}{3}\beta + \frac{1-\beta}{5} \\ \frac{1}{3}\beta + \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1}{2}\beta + \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} \\ \frac{1}{3}\beta + \frac{1-\beta}{5} & \beta + \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} \end{pmatrix}$$

Our new transition matrix

The matrix ***R*** has the same properties than ***P*** which means that it admits a stationary distribution, so we can use all the theorems we saw previously.

That's it for the PageRank algorithm. I hope you understood the intuition and the theory behind the PageRank algorithm. Please, do not hesitate to leave comments or share my work.

Random Walk Method – In the random walk method we will choose 1 node from the graph uniformly at random. After choosing the node we will look at its neighbors and choose a neighbor uniformly at random and continue these iterations until convergence is reached. After N iterations a point will come after which there will be no change In points of every node. This situation is called convergence.

Algorithm: Below are the steps for implementing the Random Walk method.

Create a directed graph with N nodes.

Now perform a random walk.

Now get sorted nodes as per points during random walk.

At last, compare it with the inbuilt PageRank method.

Below is the python code for the implementation of the points distribution algorithm.

Output:

```
PageRank using Random Walk Method

[ 9 10  4  6  3  8 13 14  0  7  1  2  5 12 11]

PageRank using inbuilt pagerank method
```

```
9, 10, 6, 3, 4, 8, 13, 0, 14, 7, 1, 2, 5, 12, 11,
```

Conclusion:

By this way, we have successfully Implemented Page Rank Algorithm.

Oral Questions-

- What is mean by Random Walk?
- How you have implemented random walk method?
- What is mean by Page Rank Algorithm?