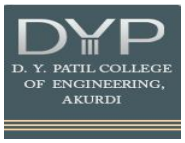
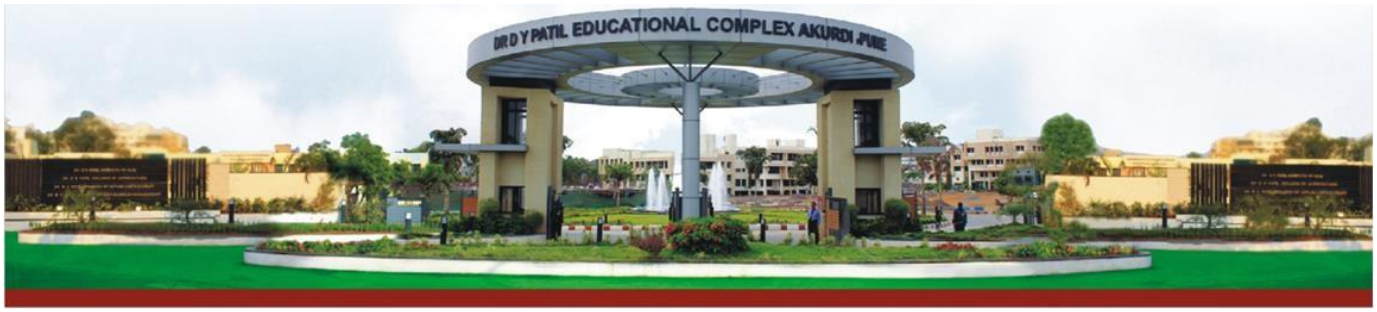



**D. Y. PATIL College Of Engineering,Akurdi,Pune-44**

# Department of Artificial Intelligence and Data Science

# LAB MANUAL
# Computer Laboratory-II
# (BE)
# Semester I

**Prepared by:**
**Dr. Bhagyashree A. Tingare**
**Mrs.Nita J.Mahale**




# Computer Laboratory -II

| Course Code | Course Name | Teaching Scheme (Hrs./ Week) | Credits |
|---|---|---|---|
| 417526 | Computer Laboratory-II: Information Retrieval | 4 | 2 |
| 417526 | Computer Laboratory II: UI /UX Design | 4 | 2 |

## Course Objectives:

- Understand the concepts of information retrieval and web mining
- Understand information retrieval process using standards available tools

## Course Outcomes:

- Understand the concepts of information retrieval and web mining
- Understand information retrieval process using standards available tools

**Operating System recommended:** Practical can be performed on suitable development platform

## Course Objectives:

- To study various tools of UI/UX Design
- To develop skills in creating visually appealing and cohesive user interfaces.
- To learn to conduct usability testing and evaluation
- To understand the role of prototyping in the design process
- To study collaborative features of UI/ UX Tool
- To gain awareness of ethical considerations in UI/UX design

## Course Outcomes:

- Apply user-centered design methodologies
- Create effective user interfaces / user experiences
- Develop proficiency in design tools
- Design for multiple platforms and devices
- Conduct usability testing and analysis
- Develop a portfolio of UI/UX design projects

**Operating System recommended:** Practical can be performed on suitable development platform

# Table of Contents

| | |
|---|---|
| **Lab Assignment No.** | 01 |
| **Title** | Write a program for pre-processing of a text document such as stop word removal, stemming. |
| **Roll No.** | |
| **Class** | BE |
| **Date of Completion** | |
| **Subject** | Computer Laboratory-II |
| **Assessment Marks** | |
| **Assessor's Sign** | |

# ASSIGNMENT No: 01

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

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

## Prerequisite:

Basics of Python

## Software Requirements: Jupyter

## Hardware Requirements:

PIV, 2GB RAM, 500 GB HDD

**Learning Objectives:**

Learn to remove stop words and stemming

## Outcomes:

After completion of this assignment students are able to understand how to remove stop words and stemming

## Theory:

The process of converting data to something a computer can understand is referred to as **pre-processing.** One of the major forms of pre-processing is to filter out useless data. In natural language processing, useless words (data), are referred to as stop words.

What are Stop words?

Stop Words: A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.

We would not want these words to take up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to stop words. NLTK(Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages. You can find them in the nltk_data directory. home/pratima/nltk_data/corpora/stopwords are the directory

address.(Do not forget to change your home directory name)

| Sample text with Stop Words | Without Stop Words |
|---|---|
| GeeksforGeeks – A Computer Science Portal for Geeks | GeeksforGeeks , Computer Science, Portal ,Geeks |
| Can listening be exhausting? | Listening, Exhausting |
| I like reading, so I read | Like, Reading, read |

To check the list of stopwords you can type the following commands in the python shell.

```
import nltk
from nltk.corpus import stopwords

nltk.download('stopwords')
print(stopwords.words('english'))
```

**Output:**

*['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]*

**Note:** You can even modify the list by adding words of your choice in the English .txt. file in the stopwords directory.

Removing stop words with NLTK

The following program removes stop words from a piece of text:

```
from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize
```

```
example_sent = """This is a sample sentence,
                  showing off the stop words filtration."""


stop_words = set(stopwords.words('english'))


word_tokens  =  word_tokenize(example_sent)
  converts the words in word_tokens to lower case and then checks whether
they are present in stop_words or not
filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]
with no lower case conversion
filtered_sentence = []
for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)


print(word_tokens)
print(filtered_sentence)
```

Output:

['This', 'is', 'a', 'sample', 'sentence', ',', 'showing',

'off', 'the', 'stop', 'words', 'filtration', '.']

['This', 'sample', 'sentence', ',', 'showing', 'stop',

'words', 'filtration', '.']

**Performing the Stopwords operations in a file**

In the code below, text.txt is the original input file in which stopwords are to be removed. filteredtext.txt is the output file. It can be done using following code:

import io

from nltk.corpus import stopwords

```
from nltk.tokenize import word_tokenize

word_tokenize accepts

 a string as an input, not a file.

stop_words = set(stopwords.words('english'))

file1 = open("text.txt")

 Use this to read file content as a stream:

line = file1.read()

words = line.split()

for r in words:

   if not r in stop_words:

      appendFile = open('filteredtext.txt','a')

      appendFile.write(" "+r)

      appendFile.close()
```

This is how we are making our processed content more efficient by removing words that do not contribute to any future operations.

Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm reduces the words “chocolates”, “chocolatey”, “choco” to the root word, “chocolate” and “retrieval”, “retrieved”, “retrieves” reduce to the stem “retrieve”. Stemming is an important part of the pipelining process in Natural language processing. The input to the stemmer is tokenized words. How do we get these tokenized words? Well, tokenization involves breaking down the document into different words.

Stemming is a natural language processing technique that is used to reduce words to their base form, also known as the root form. The process of stemming is used to normalize text and make it easier to process. It is an important step in text pre-processing, and it is commonly used in information retrieval and text mining applications.

There are several different algorithms for stemming, including the Porter stemmer, Snowball stemmer, and the Lancaster stemmer. The Porter stemmer is the most widely used algorithm, and it is based on a set of heuristics that are used to remove common suffixes from words. The Snowball stemmer is a more advanced algorithm that is based on the Porter stemmer, but it also supports several other languages in addition to English. The Lancaster stemmer is a more aggressive stemmer and it is less accurate than the Porter stemmer and Snowball stemmer.

Stemming can be useful for several natural language processing tasks such as text classification, information retrieval, and text summarization. However, stemming can also have some negative effects such as reducing the readability of the text, and it may not always produce the correct root form of a word.

Errors in Stemming:

There are mainly two errors in stemming –

over-stemming

under-stemming

Over-stemming occurs when two words are stemmed from the same root that are of different stems. Over-stemming can also be regarded as false-positives. Over-stemming is a problem that can occur when using stemming algorithms in natural language processing. It refers to the situation where a stemmer produces a root form that is not a valid word or is not the correct root form of a word. This can happen when the stemmer is too aggressive in removing suffixes or when it does not consider the context of the word.

Over-stemming can lead to a loss of meaning and make the text less readable. For example, the word "arguing" may be stemmed to "argu," which is not a valid word and does not convey the same meaning as the original word. Similarly, the word "running" may be stemmed to "run," which is the base form of the word but it does not convey the meaning of the original word.

To avoid over-stemming, it is important to use a stemmer that is appropriate for the task and language. It is also important to test the stemmer on a sample of text to ensure that it is producing valid root forms. In some cases, using a lemmatizer instead of a stemmer may be a better solution as it takes into account the context of the word, making it less prone to errors.

Another approach to this problem is to use techniques like semantic role labeling, sentiment analysis, context-based information, etc. that help to understand the context of the text and make the stemming process more precise.

Under-stemming occurs when two words are stemmed from the same root that are not of different stems. Under-stemming can be interpreted as false-negatives. Under-stemming is a problem that can occur when using stemming algorithms in natural language processing. It refers to the situation where a stemmer does not produce the correct root form of a word or does not reduce a word to its base form. This can happen when the stemmer is not aggressive enough in removing suffixes or when it is not designed for the specific task or language.

Under-stemming can lead to a loss of information and make it more difficult to analyze text. For example, the word "arguing" and "argument" may be stemmed to "argu," which does not convey the meaning of the original words. Similarly, the word "running" and "runner" may be stemmed to "run," which is the base form of the word but it does not convey the meaning of the original words.

To avoid under-stemming, it is important to use a stemmer that is appropriate for the task and language. It is also important to test the stemmer on a sample of text to ensure that it is producing

the correct root forms. In some cases, using a lemmatizer instead of a stemmer may be a better solution as it takes into account the context of the word, making it less prone to errors.

Another approach to this problem is to use techniques like semantic role labeling, sentiment analysis, context-based information, etc. that help to understand the context of the text and makethe stemming process more precise.

Applications of stemming :

Stemming is used in information retrieval systems like search

engines. It is used to determine domain vocabularies in domain

analysis.

To display search results by indexing while documents are evolving into numbers and to map documents to common subjects by stemming.

Sentiment Analysis, which examines reviews and comments made by different users about anything, is frequently used for product analysis, such as for online retail stores. Before it is interpreted, stemming is accepted in the form of the text-preparation mean.

A method of group analysis used on textual materials is called document clustering (also known as text clustering). Important uses of it include subject extraction, automatic document structuring, andquick information retrieval.

*Fun Fact*: Google search adopted a word stemming in 2003. Previously a search for “fish” would nothave returned “fishing” or “fishes”.

Some Stemming algorithms are:

Porter’s Stemmer algorithm
It is one of the most popular stemming methods proposed in 1980. It is based on the idea that the suffixes in the English language are made up of a combination of smaller and simpler suffixes. This stemmer is known for its speed and simplicity. The main applications of Porter Stemmer include data mining and Information retrieval. However, its applications are only limited to English words. Also, the group of stems is mapped on to the same stem and the output stem is not necessarily a meaningful word. The algorithms are fairly lengthy in nature and are known to be the oldest stemmer.
Example: EED -> EE means “if the word has at least one vowel and consonant plus EED ending, change the ending to EE” as ‘agreed’ becomes ‘agree’.

Advantage: It produces the best output as compared to other stemmers and it has less error rate.

Limitation: Morphological variants produced are not always real words.

Lovins Stemmer
It is proposed by Lovins in 1968, that removes the longest suffix from a word then the word is recorded to convert this stem into valid words.
Example: sitting -> sitt -> sit

Advantage: It is fast and handles irregular plurals like 'teeth' and 'tooth' etc.

Limitation: It is time consuming and frequently fails to form words from stem.

Dawson Stemmer
It is an extension of Lovins stemmer in which suffixes are stored in the reversed order indexed by their length and last letter.

Advantage: It is fast in execution and covers more suffices.

Limitation: It is very complex to implement.

Krovetz Stemmer
It was proposed in 1993 by Robert Krovetz. Following are the steps:
1) Convert the plural form of a word to its singular form.
2) Convert the past tense of a word to its present tense and remove the suffix 'ing'.
Example: 'children' -> 'child'

Advantage: It is light in nature and can be used as pre-stemmer for other stemmers.

Limitation: It is inefficient in case of large documents.

Xerox Stemmer
Example:

'children' -> 'child'

'understood' -> 'understand'

'whom' -> 'who'

'best' -> 'good'

N-Gram Stemmer
An n-gram is a set of n consecutive characters extracted from a word in which similar words will have a high proportion of n-grams in common.
Example: 'INTRODUCTIONS' for n=2 becomes : I, IN, NT, TR, RO, OD, DU, UC, CT, TI, IO, ON, NS, S

Advantage: It is based on string comparisons and it is language dependent.

Limitation: It requires space to create and index the n-grams and it is not time efficient.

Snowball Stemmer:

When compared to the Porter Stemmer, the Snowball Stemmer can map non-English words too. Since it supports other languages the Snowball Stemmers can be called a multi-lingual stemmer. The Snowball stemmers are also imported from the nltk package. This stemmer is based on a programming language called 'Snowball' that processes small strings and is the most widely used stemmer. The Snowball stemmer is way more aggressive than Porter Stemmer and is also referred to

as Porter2 Stemmer. Because of the improvements added when compared to the Porter Stemmer, the Snowball stemmer is having greater computational speed.

Lancaster Stemmer:

The Lancaster stemmers are more aggressive and dynamic compared to the other two stemmers. The stemmer is really faster, but the algorithm is really confusing when dealing with small words. But they are not as efficient as Snowball Stemmers. The Lancaster stemmers save the rules externally and basically uses an iterative algorithm.

Some more example of stemming for root word "like" include:

-> "likes"

-> "liked"

-> "likely"

-> "liking"

**Errors in Stemming:** There are mainly two errors in stemming – ***Overstemming*** and ***Understemming***. Overstemming occurs when two words are stemmed from the same root that are of different stems. Under-stemming occurs when two words are stemmed from the same root that is not of different stems.

**Applications of stemming are:**

Stemming is used in information retrieval systems like search engines.

It is used to determine domain vocabularies in domain analysis.

Stemming is desirable as it may reduce redundancy as most of the time the word stem and their inflected/derived words mean the same.

Below is the implementation of stemming words using NLTK:

```
from nltk.stem import PorterStemmer

from nltk.tokenize import word_tokenize

ps = PorterStemmer()

 choose some words to be stemmed

words = ["program", "programs", "programmer", "programming", "programmers"]


for w in words:

   print(w, " : ", ps.stem(w))
```

Output:

program : program

programs : program

programmer : program

programming : program

programmers : program

**Code 2:** Stemming words from sentences

```
importing modules
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
ps = PorterStemmer()
sentence = "Programmers program with programming languages"
words = word_tokenize(sentence)
for w in words:
    print(w, " : ", ps.stem(w))
```

**Output :**

Programmers : program

program : program

with : with

programming : program

languages : language

Conclusion: The pre-processing of text data not only reduces the dataset size but also helps us to focus on only useful and relevant data so that the future model would have a large percentage of efficiency. With the help of pre-processing techniques like tokenization, stemming, lemmatization, removing stop-words, and part of speech tag we can remove all the irrelevant text from our dataset and make our dataset ready for further processing or model building.

| | |
|---|---|
| **Lab Assignment No.** | 02 |
| **Title** | Implement a program for retrieval of documents using inverted files. |
| **Roll No.** | |
| **Class** | BE |
| **Date of Completion** | |
| **Subject** | Computer Laboratory-II |
| **Assessment Marks** | |
| **Assessor's Sign** | |

# ASSIGNMENT No: 02

**Title:** Implement a program for retrieval of documents using inverted files.

**Problem Statement:** Implement a program for retrieval of documents using inverted files.

## Prerequisite:

Basics of Python

**Software Requirements:** Jupyter

**Hardware Requirements:**

PIV, 2GB RAM, 500 GB HDD

**Learning Objectives:**

Learn to retrieve documents using inverted files

**Outcomes:**

After completion of this assignment students are able to understand how to retrieve documents using inverted files

**Theory:**

An **Inverted Index** is a data structure used in information retrieval systems to efficiently retrieve documents or web pages containing a specific term or set of terms. In an inverted index, the index is organized by terms (words), and each term points to a list of documents or web pages that contain that term.

Inverted indexes are widely used in search engines, database systems, and other applications where efficient text search is required. They are especially useful for large collections of documents, where searching through all the documents would be prohibitively slow.

An inverted index is an index data structure storing a mapping from content, such as words or number s, 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.

**Example: Consider the following documents.**

Document 1: The quick brown fox jumped over the lazy dog.

Document 2: The lazy dog slept in the sun.

To create an **inverted index** for these documents, we first tokenize the documents into terms, as follows.

Document 1: The, quick, brown, fox, jumped, over, the lazy, dog.

Document 2: The, lazy, dog, slept, in, the, sun.

Next, we create an index of the terms, where each term points to a list of documents that contain that term, as follows.

The -> Document 1, Document 2
Quick -> Document 1
Brown -> Document 1
Fox -> Document 1
Jumped -> Document 1
Over -> Document 1
Lazy -> Document 1, Document 2
Dog -> Document 1, Document 2
Slept -> Document 2
In -> Document 2
Sun -> Document 2

To search for documents containing a particular term or set of terms, the search engine queries the inverted index for those terms and retrieves the list of documents associated with each term. The search engine can then use this information to rank the documents based on relevance to the query and present them to the user in order of importance.
There are two types of inverted indexes:

- **Record-Level Inverted Index:** Record Level Inverted Index contains a list of references to documents for each word.
- **Word-Level Inverted Index:** Word Level Inverted Index additionally contains the positions of each word within a document. The latter form offers more functionality but needs more processing power and space to be created.

Suppose we want to search the texts "hello everyone, " "this article is based on an inverted index, " and "which is **hashmap-like data structure**". If we index by (text, word within the text), the index with a location in the text is:

| | |
|---|---|
| hello | (1, 1) |
| everyone | (1, 2) |
| this | (2, 1) |
| article | (2, 2) |
| is | (2, 3); (3, 2) |
| based | (2, 4) |
| on | (2, 5) |
| inverted | (2, 6) |
| index | (2, 7) |
| which | (3, 1) |
| hashmap | (3, 3) |
| like | (3, 4) |
| data | (3, 5) |
| structure | (3, 6) |

The word "hel o" is in document 1 ("hello everyone") starting at word 1, so has an entry (1, 1), and the word "is" is in documents 2 and 3 at '3rd' and '2nd' positions respectively (here position is based on the word).
The index may have weights, frequencies, or other indicators.

**Steps to Build an Inverted Index**

- **Fetch the Document:** Removing of Stop Words: Stop words are the most occurring and useless words in documents like "I", "the", "we", "is", and "an".

- **Stemming of Root Word:** Whenever I want to search for "cat", I want to see a document that has information about it. But the word present in the document is called "cats" or "catty" instead of "cat". To relate both words, I'**l** chop some part of every word I read so that I could get the "root word". There are standard tools for performing this like "Porter's Stemmer".
- **Record Document IDs:** If the word is already present add a reference of the document to index else creates a new entry. Add additional information like the frequency of the word, location of the word, etc.

**Example:**

| Words | Document |
|---|---|
| ant | doc1 |
| demo | doc2 |
| world | doc1, doc2 |

## Implementing Inverted Index

```
Define the documents
document1 = "The quick brown fox jumped over the lazy dog."
document2 = "The lazy dog slept in the sun."

Step 1: Tokenize the documents
Convert each document to lowercase and split it into wordstokens1
= document1.lower().split()
tokens2 = document2.lower().split()

Combine the tokens into a list of unique terms
terms = list(set(tokens1 + tokens2))

Step 2: Build the inverted index
Create an empty dictionary to store the inverted index
inverted_index = {}

For each term, find the documents that contain itfor
term in terms:
    documents = []
    if term in tokens1:
        documents.append("Document 1")
    if term in tokens2:
        documents.append("Document 2")
    inverted_index[term] = documents

Step 3: Print the inverted index
for term, documents in inverted_index.items():
    print(term, "->", ", ".join(documents))
```

**Advantages of Inverted Index**

- The inverted index is to allow fast full-text searches, at a cost of increased processing when a document is added to the database.
- It is easy to develop.
- It is the most popular data structure used in document retrieval systems, used on a large scale for example in search engines.

**Disadvantages of Inverted Index**

- Large storage overhead and high maintenance costs on updating, deleting, and inserting.
- Instead of retrieving the data in decreasing order of expected usefulness, the records are retrieved in the order in which they occur in the inverted lists.

**Features of Inverted Indexes**

- **Efficient search:** Inverted indexes allow for efficient searching of large volumes of text-based data. By indexing every term in every document, the index can quickly identify all documents that contain a given search term or phrase, significantly reducing search time.
- **Fast updates:** Inverted indexes can be updated quickly and efficiently as new content is added to the system. This allows for near-real-time indexing and searching for new content.
- **Flexibility:** Inverted indexes can be customized to suit the needs of different types of information retrieval systems. For example, they can be configured to handle different types of queries, such as Boolean queries or proximity queries.
- **Compression:** Inverted indexes can be compressed to reduce storage requirements. Various techniques such as delta encoding, gamma encoding, variable byte encoding, etc. can be used to compress the posting list efficiently.
- **Support for stemming and synonym expansion:** Inverted indexes can be configured to support stemming and synonym expansion, which can improve the accuracy and relevance of search results. Stemming is the process of reducing words to their base or root form, while synonym expansion involves mapping different words that have similar meanings to a common term.
- **Support for multiple languages:** Inverted indexes can support multiple languages, allowing users to search for content in different languages using the same system.

**Conclusion: -** This way, Implemented a program for inverted files.

| | |
|---|---|
| **Lab Assignment No.** | 3 |
| **Title** | 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. |
| **Roll No.** | |
| **Class** | BE |
| **Date of Completion** | |
| **Subject** | Computer Laboratory-II |
| **Assessment Marks** | |
| **Assessor's Sign** | |

# ASSIGNMENT No: 03

**Title:** 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.

**Problem Statement:** 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.

## Prerequisite:

Basics of Python

## Software Requirements: Jupyter

## Hardware Requirements:

PIV, 2GB RAM, 500 GB HDD

**Learning Objectives:**

Learn to construct a Bayesian network considering medical data.

## Outcomes:

After completion of this assignment students are able to understand how to construct a Bayesian network considering medical data.

## 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.

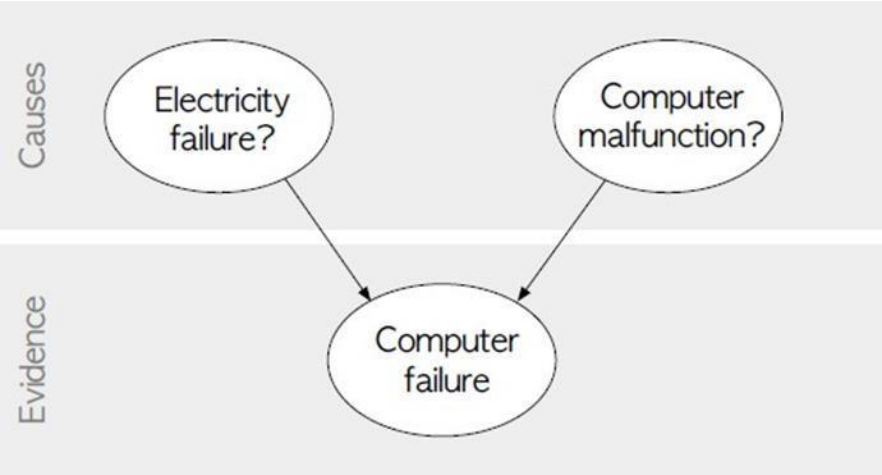



Fig: Directed acyclic graph representing two independent possible causes of a computer failure.

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 diseasein 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 |

**Attribute Information:**

1. age: age in years

2. sex: sex (1 = male; 0 = female)
3. cp: chest pain type
    - Value 1: typical angina
    - Value 2: atypical angina
    - Value 3: non-anginal pain
    - 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
    - Value 0: normal
    - Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
    - 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
    - Value 1: upsloping
    - Value 2: flat
    - Value 3: downsloping
12. ca = number of major vessels (0-3) colored by flourosopy
13. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
14. Heartdisease: It is integer valued from 0 (no presence) to 4. Diagnosis of heart disease(angiographic disease status)

## Some instance from the dataset:

| age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | Heartdisease |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 63 | 1 | 1 | 145 | 233 | 1 | 2 | 150 | 0 | 2.3 | 3 | 0 | 6 | 0 |
| 67 | 1 | 4 | 160 | 286 | 0 | 2 | 108 | 1 | 1.5 | 2 | 3 | 3 | 2 |
| 67 | 1 | 4 | 120 | 229 | 0 | 2 | 129 | 1 | 2.6 | 2 | 2 | 7 | 1 |
| 41 | 0 | 2 | 130 | 204 | 0 | 2 | 172 | 0 | 1.4 | 1 | 0 | 3 | 0 |
| 62 | 0 | 4 | 140 | 268 | 0 | 2 | 160 | 0 | 3.6 | 3 | 2 | 3 | 3 |
| 60 | 1 | 4 | 130 | 206 | 0 | 2 | 132 | 1 | 2.4 | 2 | 2 | 7 | 4 |

**Program:**

```
import numpy as np import csv
import pandas as pd
from pgmpy.models import BayesianModel
from pgmpy.estimators import MaximumLikelihoodEstimator 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('Few examples from the dataset are given below')
print(heartDisease.head())


Model Bayesian Network
Model=BayesianModel([('age','trestbps'),('age','fbs'),
('sex','trestbps'),('exang','trestbps'),('trestbps','heartdise
ase'),('fbs','heartdisease'),('heartdisease','restecg'),
('heartdisease','thalach'),('heartdisease','chol')])

Learning CPDs using Maximum Likelihood Estimators print('\n Learning CPD using Maximum
likelihood estimators') model.fit(heartDisease,estimator=MaximumLikelihoodEstimator)

Inferencing with Bayesian Network print('\n Inferencing with Bayesian Network:')
HeartDisease_infer = VariableElimination(model)

computing the Probability of HeartDisease given Age print('\n 1. Probability of HeartDisease
given Age=30') q=HeartDisease_infer.query(variables=['heartdisease'],evidence
={'age':28})
print(q['heartdisease'])

computing the Probability of HeartDisease given cholesterol print('\n 2. Probability of HeartDisease
given cholesterol=100') q=HeartDisease_infer.query(variables=['heartdisease'],evidence
={'chol':100})
print(q['heartdisease'])
```

***Output:***

```
Few examples from the dataset are given below
```

```
     age   sex cp   trestbps    ...slope ca  thal
heartdisease
0    63     1    1      145       ...   3    0      6               0
1    67     1    4      160       ...   2    3      3               2
2    67     1    4      120       ...   2    2      7               1
3    37     1    3      130       ...   3    0      3               0
4    41     0    2      130       ...   1    0      3               0

[5 rows x 14 columns]

Learning CPD using Maximum likelihood

estimatorsInferencing with Bayesian Network:


1. Probability of HeartDisease given Age=28
```

| heartdisease | phi(heartdisease) |
|---|---|
| heartdisease_0 | 0.6791 |
| heartdisease_1 | 0.1212 |
| heartdisease_2 | 0.0810 |
| heartdisease_3 | 0.0939 |
| heartdisease_4 | 0.0247 |

2. Probability of HeartDisease given cholesterol=100

| heartdisease | phi(heartdisease) |
|---|---|
| heartdisease_0 | 0.5400 |
| heartdisease_1 | 0.1533 |
| heartdisease_2 | 0.1303 |
| heartdisease_3 | 0.1259 |
| heartdisease_4 | 0.0506 |

**Conclusion:** This way constructed a Bayesian network considering medical data.

| | |
|---|---|
| **Lab Assignment No.** | 04 |
| **Title** | Implement Agglomerative hierarchical clustering algorithm using appropriate dataset. |
| **Roll No.** | |
| **Class** | BE |
| **Date of Completion** | |
| **Subject** | Computer Laboratory-II |
| **Assessment Marks** | |
| **Assessor's Sign** | |

# ASSIGNMENT No: 04

**Title:** I+++.

**Problem Statement:** Implement Agglomerative hierarchical clustering algorithm using appropriate dataset

## Prerequisite:

Basics of Python

## Software Requirements: Jupyter

## Hardware Requirements:

PIV, 2GB RAM, 500 GB HDD

**Learning Objectives:**

Learn to Implement Agglomerative hierarchical clustering algorithm using appropriate dataset

## Outcomes:

After completion of this assignment students are able to understand how to Implement Agglomerative hierarchical clustering algorithm using appropriate dataset

## Theory:

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

In machine learning, clustering is the unsupervised learning technique that groups the data based on similarity between the set of data. There are different-different types of clustering algorithms in machine learning. **Connectivity-based clustering:** This type of clustering algorithm builds the cluster based on the connectivity between the data points. Example: Hierarchical clustering

- **Centroid-based clustering:** This type of clustering algorithm forms around the centroids of the data points. Example: K-Means clustering, K-Mode clustering
- **Distribution-based clustering:** This type of clustering algorithm is modeled using statistical distributions. It assumes that the data points in a cluster are generated from a particular probability distribution, and the algorithm aims to estimate the parameters of the distribution to group similar data points into clusters Example: Gaussian Mixture Models (GMM)
- **Density-based clustering:** This type of clustering algorithm groups together data points that are in high-density concentrations and separates points in low-concentrations regions. The basic idea

is that it identifies regions in the data space that have a high density of data points and groups those points together into clusters. Example: DBSCAN(Density-Based Spatial Clustering of Applications with Noise)

**Hierarchical clustering**

Hierarchical clustering is a connectivity-based clustering model that groups the data points together that are close to each other based on the measure of similarity or distance. The assumption is that data points that are close to each other are more similar or related than data points that are farther apart.

A dendrogram, a tree-like figure produced by hierarchical clustering, depicts the hierarchical relationships between groups. Individual data points are located at the bottom of the dendrogram, while the largest clusters, which include all the data points, are located at the top. In order to generate different numbers of clusters, the dendrogram can be sliced at various heights.

The dendrogram is created by iteratively merging or splitting clusters based on a measure of similarity or distance between data points. Clusters are divided or merged repeatedly until all data points are contained within a single cluster, or until the predetermined number of clusters is attained.

We can look at the dendrogram and measure the height at which the branches of the dendrogram form distinct clusters to calculate the ideal number of clusters. The dendrogram can be sliced at this height to determine the number of clusters.

**Types of Hierarchical Clustering**

Basically, there are two types of hierarchical Clustering:

1. Agglomerative Clustering
2. Divisive clustering

**Hierarchical Agglomerative Clustering**

It is also known as the 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 agglomerate pairs of clusters until all clusters have been merged into a single cluster that contains all data.

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

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 dataX

= X.drop('CUST_ID', axis = 1)


Handling the missing values

X.fillna(method ='ffill', inplace = True)
```

Step 3: Preprocessing the data

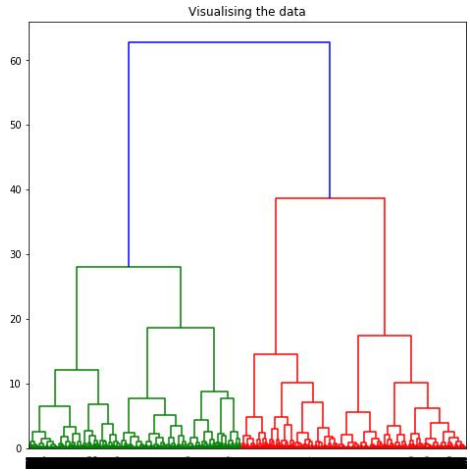
```
Scaling the data so that all the features become comparable
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

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

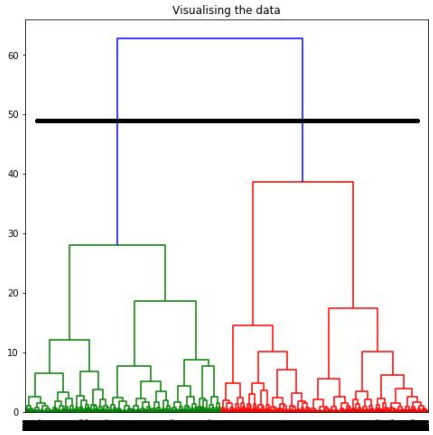
```
pca = PCA(n_components = 2)
X_principal = pca.fit_transform(X_normalized)
X_principal = pd.DataFrame(X_principal)
X_principal.columns = ['P1', 'P2']
```

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

```
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.

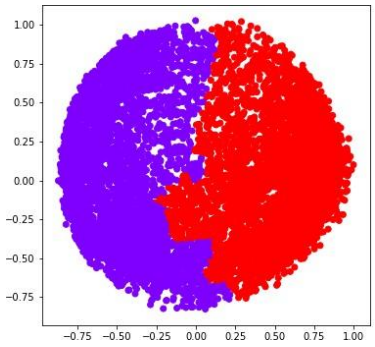



The above image shows that the optimal number of clusters should be 2 for the given data. Step 6: Building and Visualizing the different clustering models for different values of k a) k = 2

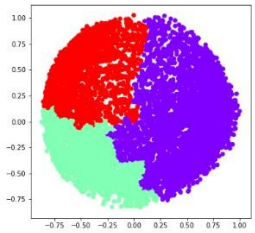
```
ac2 = AgglomerativeClustering(n_clusters = 2)
```

Visualizing the clustering

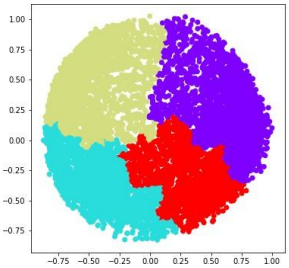
```
plt.figure(figsize =(6, 6))
plt.scatter(X_principal['P1'], X_principal['P2'],
        c = ac2.fit_predict(X_principal), cmap ='rainbow')
plt.show()
```



```
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()
```

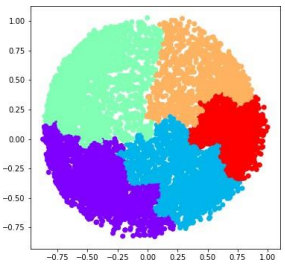


```
ac4 = AgglomerativeClustering(n_clusters = 4)

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

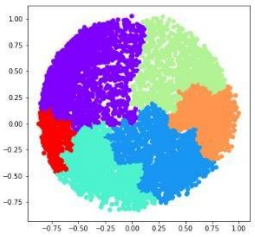


```
ac5 = AgglomerativeClustering(n_clusters = 5)

plt.figure(figsize =(6, 6))
plt.scatter(X_principal['P1'], X_principal['P2'],
```

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

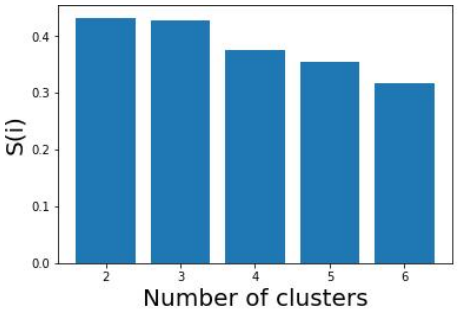


```
ac6 = AgglomerativeClustering(n_clusters = 6)

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.

```
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()
```




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** :- This way Implemented Agglomerative hierarchical clustering algorithm using appropriate dataset

| | |
|---|---|
| **Lab Assignment No.** | 5 |
| **Title** | Implement Page Rank Algorithm. (Use python or beautiful soup for implementation). |
| **Roll No.** | |
| **Class** | BE |
| **Date of Completion** | |
| **Subject** | Computer Laboratory-II |
| **Assessment Marks** | |
| **Assessor's Sign** | |

# ASSIGNMENT No: 5

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

**Problem Statement:** Implement Page Rank Algorithm. (Use python or beautiful soup for implementation).

## Prerequisite:

Basics of Python

## Software Requirements: Jupyter

## Hardware Requirements:

PIV, 2GB RAM, 500 GB HDD

**Learning Objectives:**

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

## Outcomes:

After completion of this assignment students are able to understand how to Implement Page Rank Algorithm. (Use python or beautiful soup for implementation).

## Theory:

PageRank (PR) is an algorithm used by Google Search to rank websites in their search engine results. PageRank was named after Larry Page, one of the founders of Google. PageRank is a way of measurin g the importance of website pages. According to Google:

*PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.*

It is not the only algorithm used by Google to order search engine results, but it is the first algorit hm that was used by the company, and it is the best-known. The above centrality measure is not implemented for multi-graphs.

***Algorithm***

The PageRank algorithm outputs a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. PageRank can be calculated for collectio ns of documents of any size. It is assumed in several research papers that the distributio n is evenly divided among all documents in the collection at the beginning of the computational process. The PageRank computations require several passes, called “iterations”, through the collection to adjust approximate PageRank values to more closely reflect the theoretical true value.

Simplified algorithm

Assume a small universe of four web pages: A, B, C, and D. Links from a page to itself, or multiple outbound links from one single page to another single page, are ignored. PageRank is initialized to the same value for all pages. In the original form of PageRank, the sum of PageRank over all pages was the total number of pages on the web at that time, so each page in this example would have an initial value of 1. However, later versions of PageRank, and the remainder of this section, assume a probability distribution between 0 and 1. Hence the initial value for each page in this example is 0.25.

The PageRank transferred from a given page to the targets of its outbound links upon the next iteration is divided equally among all outbound links.

If the only links in the system were from pages B, C, and D to A, each link would transfer 0.25 PageRank to A upon the next iteration, for a total of 0.75.

Suppose instead that page B had a link to pages C and A, page C had a link to page A, and page D had links to all three pages. Thus, upon the first iteration, page B would transfer half of its existing value, or 0.125, to page A and the other half, or 0.125, to page C. Page C would transfer all of its existing value, 0.25, to the only page it links to, A. Since D had three outbound links, it would transfer one-third of its existing value, or approximately 0.083, to A. At the completion of this iteration, page A will have a PageRank of approximately 0.458.

In other words, the PageRank conferred by an outbound link is equal to the document's own PageRank score divided by the number of outbound links L( ).

In the general case, the PageRank value for any page u can be expressed as:

i.e. the PageRank value for a page u is dependent on the PageRank values for each page v contained in the set Bu (the set containing all pages linking to page u), divided by the number L(v) of links from page v. The algorithm involves a damping factor for the calculation of the PageRank. It is like the income tax which the govt extracts from one despite paying him itself.

Following is the code for the calculation of the Page rank.

```
def pagerank(G, alpha=0.85, personalization=None,
             max_iter=100, tol=1.0e-6, nstart=None, weight='weight',dangling=None):
    """Return the PageRank of the nodes in the graph.

    PageRank computes a ranking of the nodes in the graph G based on
    the structure of the incoming links. It was originally designed as
    an algorithm to rank web pages.

    Parameters
    ----------
    G : graph
      A NetworkX graph. Undirected graphs will be converted to a directed
      graph with two directed edges for each undirected edge.

    alpha : float, optional
      Damping parameter for PageRank, default=0.85.

    personalization: dict, optional
      The "personalization vector" consisting of a dictionary with a
      key for every graph node and nonzero personalization value for each node.
```

```
    By default, a uniform distribution is used.

  max_iter : integer, optional
    Maximum number of iterations in power method eigenvalue solver.

  tol : float, optional
    Error tolerance used to check convergence in power method solver.

  nstart : dictionary, optional
    Starting value of PageRank iteration for each node.

  weight : key, optional
    Edge data key to use as weight.  If None weights are set to 1.

  dangling: dict, optional
    The outedges to be  assigned to any "dangling" nodes, i. e., nodes withou t
    any outedges. The dict key is the node the outedge points to and the dict
    value is the weight of that outedge. By default, dangling nodes are given
    outedges according to the personalization vector (uniform if not
    specified). This must be selected to result in an irreducible transition
    matrix (see notes under google_matrix). It may be common to have the
    dangling dict to be the same as the personalization dict.

  Returns
  -------
  pagerank : dictionary
     Dictionary of nodes with PageRank as value

  Notes
  -----
  The eigenvector calculation is done by the power iteration method
  and has no guarantee of convergence.  The iteration will stop
  after max_iter iterations or an error tolerance of
  number_of_nodes(G) tol has been reached.

  The PageRank algorithm was designed for directed graphs but this
  algorithm does not check if the input graph is directed and will
  execute on undirected graphs by converting each edge in the
  directed graph to two edges.


  """
  if len(G) == 0:
      return {}

  if  not  G.is_directed():
      D = G.to_directed()
  else:
      D = G

    Create a copy in (right) stochastic form
  W =  nx.stochastic_graph(D, weight=weight)
  N = W.number_of_nodes()

    Choose fixed starting vector if not given
```

```
    if nstart is None:
        x = dict.fromkeys(W, 1.0 / N)
    else:
          Normalized nstart vector
        s = float(sum(nstart.values()))
        x = dict((k, v / s) for k, v in nstart.items())

    if personalization is None:

          Assign uniform personalization vector if not given
        p = dict.fromkeys(W, 1.0 / N)
    else:
        missing = set(G) - set(personalization)
        if missing:
            raise NetworkXError('Personalization dictionary '
                                'must have a value for every node. '
                                'Missing nodes %s' % missing)
        s = float(sum(personalization.values()))
        p = dict((k, v / s) for k, v in personalization.items())

    if dangling is None:

          Use personalization vector if dangling vector not specified
        dangling_weights = p
    else:
        missing = set(G) - set(dangling)
        if missing:
            raise NetworkXError('Dangling node dictionary '
                                'must have a value for every node. '
                                'Missing nodes %s' % missing)
        s = float(sum(dangling.values()))
        dangling_weights = dict((k, v/s) for k, v in dangling.items())
    dangling_nodes = [n for n in W if W.out_degree(n, weight=weight) == 0.0]

      power iteration: make up to max_iter iterations
    for _ in range(max_iter):
        xlast = x
        x = dict.fromkeys(xlast.keys(), 0)
        danglesum = alpha * sum(xlast[n] for n in dangling_nodes)
        for n in x:

              this matrix multiply looks odd because it is
            doing a left multiply x^T=xlast^T W
            for nbr in W[n]:
                x[nbr] += alpha * xlast[n] * W[n][nbr][weight]
            x[n] += danglesum * dangling_weights[n] + (1.0 - alpha) * p[n]

          check convergence, l1 norm
        err = sum([abs(x[n] - xlast[n]) for n in x])
        if err < N*tol:
            return x
    raise NetworkXError('pagerank: power iteration failed to converge '
                        'in %d iterations.' % max_iter)
```

The above code is the function that has been implemented in the networkx library.

To implement the above in networkx, you will have to do the following:

```
>>> import networkx as nx
>>>  G=nx.barabasi_albert_graph(60,41)
>>> pr=nx.pagerank(G,0.4)
>>> pr
```

Below is the output, you would obtain on the IDLE after required installations.

```
{0: 0.012774147598875784, 1: 0.013359655345577266, 2: 0.013157355731377924,
3: 0.012142198569313045, 4: 0.013160014506830858, 5: 0.012973342862730735,
 6: 0.012166706783753325, 7: 0.011985935451513014, 8: 0.012973502696061718,
 9: 0.013374146193499381, 10: 0.01296354505412387, 11: 0.013163220326063332,
 12: 0.013368514624403237, 13: 0.013169335617283102, 14: 0.012752071800520563,
15: 0.012951601882210992, 16: 0.013776032065400283, 17: 0.012356820581336275,
18: 0.013151652554311779, 19: 0.012551059531065245, 20: 0.012583415756427995,
 21: 0.013574117265891684, 22: 0.013167552803671937, 23: 0.013165528583400423,
 24: 0.012584981049854336, 25: 0.013372989228254582, 26: 0.012569416076848989,
 27: 0.013165322299539031, 28: 0.012954300960607157, 29: 0.012776091973397076,
 30: 0.012771016515779594, 31: 0.012953404860268598, 32: 0.013364947854005844,
33: 0.012370004022947507, 34: 0.012977539153099526, 35: 0.013170376268827118,
 36: 0.012959579020039328, 37: 0.013155319659777197, 38: 0.013567147133137161,
 39: 0.012171548109779459, 40: 0.01296692767996657, 41: 0.028 089802328702826,
 42: 0.027646981396639115, 43: 0.027300188191869485, 44: 0.02689771667021551,
 45: 0.02650459107960327, 46: 0.025971186884778535, 47: 0.02585262571331937,
48: 0.02565482923824489, 49: 0.024939722913691394, 50: 0.02458271197701402,
 51: 0.024263128557312528, 52: 0.023505217517258568, 53: 0.023724311872578157,
 54: 0.02312908947188023, 55: 0.02298716954828392, 56: 0.02270220663300396,
 57: 0.022060403216132875, 58: 0.021932442105075004, 59: 0.021643288632623502}
```

**Conclusion:-** Thus, this way the centrality measure of Page Rank is calculated for the given graph.

| | |
|---|---|
| **Lab Assignment No.** | 6 |
| **Title** | Design user persona for the users of selected product / system |
| **Roll No.** | |
| **Class** | BE |
| **Date of Completion** | |
| **Subject** | Computer Laboratory-II-UI/UX |
| **Assessment Marks** | |
| **Assessor's Sign** | |

# ASSIGNMENT No: 6

**Title:** Design user persona for the users of selected product / system.

**Problem Statement:** Design user persona for the users of selected product / system.

**Prerequisite:** Basic understanding of user experience (UX) design principles, familiarity with the selected product/system, and access to user data or research findings.

## Software Requirements: Figma Tool

## Hardware Requirements:

PIV, 2GB RAM, 500 GB HDD

**Learning Objectives:**

To identify and categorize key user groups or personas for the selected product/system.

## Outcomes:

Participants will develop user personas that provide a clear and detailed understanding of the selected product/system's target users. These personas will serve as valuable tools for guiding UI/UX design decisions, helping to create user-centered and effective interfaces.

## Theory:

User personas are fictional representations of different user types or segments, based on real user data and research. They include demographic information, behaviors, goals, pain points, and motivations. Creating user personas is a crucial step in user-centered design as they help design teams empathize with users and make informed design choices. By personifying users, designers can better understand their needs and preferences, ultimately leading to improved product/system usability and user satisfaction.

## Steps Involved:

1) Research and Data Collection: Identify common user characteristics, behaviors, and goals related to the product or system.
2) Persona Creation: Create a separate persona for each user segment. Give each persona a name, a photo (stock or custom), and a brief bio to humanize them.
3) Scenarios and Use Cases: Create scenarios or use cases that illustrate how each persona would interact with the product in real-life situations. This can include day-in-the-life stories or specific task workflows.
4) Needs and Expectations: List the specific needs and expectations of each persona.
5) Reference: Throughout the design process, refer back to the personas to ensure that design decisions align with user goals and preferences.

## Conclusion:

Designing user personas is an essential step in the UI/UX design process. It ensures that designers have a deep understanding of their target audience, allowing them to create interfaces that meet user needs and expectations. The personas serve as valuable reference points throughout the design and development phases, helping to align the entire team around user-centric goals.

| | |
|---|---|
| **Lab Assignment No.** | 7 |
| **Title** | Design a wireframe for an online learning platform that includes course listings, video lectures, quizzes, and progress tracking. |
| **Roll No.** | |
| **Class** | BE |
| **Date of Completion** | |
| **Subject** | Computer Laboratory-II-UI/UX |
| **Assessment Marks** | |
| **Assessor's Sign** | |

# ASSIGNMENT No: 7

**Title:** Design a wireframe for an online learning platform that includes course listings, video lectures, quizzes, and progress tracking.

**Problem Statement:** Design a wireframe for an online learning platform that includes course listings, video lectures, quizzes, and progress tracking.

**Prerequisite**: Familiarity with wire framing tools and principles, understanding of online learning platforms, and knowledge of user experience (UX) design basics.

## Software Requirements: Figma Tool

## Hardware Requirements:

PIV, 2GB RAM, 500 GB HDD

**Learning Objectives:**

To identify key features and components required for an effective online learning platform.

## Outcomes:

Participants will produce a wireframe that outlines the layout and functionality of an online learning platform. This wireframe will serve as a blueprint for the platform's interface, ensuring that essential elements like course listings, video lectures, quizzes, and progress tracking are thoughtfully organized for a positive user experience.

## Theory:

Wireframing is a foundational phase in the UI/UX design process, serving as a blueprint for how a digital p product will function and interact with its users. For an online learning platform, wireframes are instrumental in:

1. **Establishing Layout and Structure**: Wireframes map out the arrangement of key components like course listings, video players, quizzes, and progress tracking, ensuring a logical flow and ease of navigation.
2. **Defining Functional Elements**: They help in defining the functional aspects of the platform, such as search functionalities, filters for courses, and interactive elements like buttons and links.
3. **User Interaction Planning**: Wireframes focus on how users will interact with the platform. They highlight user pathways and interactions, such as enrolling in a course, completing quizzes, and tracking progress.

**4.** Testing Usability**: Wireframes allow for early usability testing by providing a clear structure without being influenced by design aesthetics. This helps in identifying usability issues and gathering feedback on the user experience.**

**Steps to follow:**

**1. Define Goals and Requirements**

- **Project Objectives : Clarify the main goals of the online learning platform. For example, is it designed to provide a wide range of courses, focus on interactive learning, or offer certification?**

- **Target Audience : Identify who will use the platform. Are they students, professionals, or hobbyists? Understanding their needs and expectations is crucial.**

- **Key Features : List the essential features required, such as course management, user profiles, payment gateways, and reporting tools.**

**2. Research and User Analysis**

- **User Personas : Develop personas representing different user types based on demographic data, learning preferences, and technology proficiency.**

- **Competitive Analysis : Examine similar platforms to identify best practices, strengths, and areas for improvement.**

- **User Interviews and Surveys : Gather insights from potential users regarding their preferences and pain points related to online learning.**

**3. Content Inventory**

- **Content Types : Categorize content types, including:**

  - **Course Listings : Titles, descriptions, categories, and filters.**

  - **Video Lectures : Player controls, video quality options, and playback features.**

  - **Quizzes and Assignments : Question formats, answer choices, and grading systems.**

  - **Progress Tracking : Dashboards showing course progress, completed modules, and achievements.**

  - **User Profiles : Personal information, enrolled courses, and activity logs.**

  - **Navigation Menus : Main menus, submenus, and links to important sections.**

**4. Create Wireframes**

- **Main Pages : Design wireframes for the core pages:**

  - **Homepage : Layout showcasing featured courses, user login/signup, search bar, and promotional banners.**

  - **Course Listing Page : Display courses with sorting and filtering options. Include essential details like course title, description, instructor, and price.**

  - **Course Detail Page : Provide in-depth information about the course, including syllabus, instructor profile, reviews, and enrollment options.**

  - **User Profile Page : Show user information, enrolled courses, progress, and settings.**

- **Interactive Elements : Include basic interactive elements like buttons, input fields, and dropdowns in the wireframes.**

**5. Course Listings**

- **Layout Design : Determine how courses will be presented. Options include grid view, list view, or a combination of both.**

- **Course Details : Design how each course will be summarized. Include elements like:**

- **Course Title : Clearly visible and engaging.**

- **Course Description : Brief overview highlighting key features and learning outcomes.**

- **Instructor Information : Name, photo, and a short bio.**

- **Ratings and Reviews : Display average ratings and user reviews to build credibility.**

- **Price and Enrollment : Show the price of the course and provide an easy-to-access enrollment button.**

- **Filters and Sorting Options : Allow users to sort and filter courses by categories, price, rating, or popularity**

**6. Detail Interaction Design**

- **User Flows : Map out how users will navigate from one page to another. For example, how a user transitions from the course listing page to the course detail page and then to enrollment.**

- **Microinteractions : Plan for small interactive elements like hover effects, button states, and loading indicators.**

**7. Validate and Iterate**

- **Feedback Sessions : Conduct feedback sessions with stakeholders and potential users to validate the wireframes.**

- **Iterative Design : Refine the wireframes based on feedback, making adjustments to improve usability and meet user needs.**

**8. Prepare for High-Fidelity Design**

- **Design System : Once the wireframes are finalized, create a design system with colors, typography, and component libraries to guide the high-fidelity design phase.**

- **Prototyping : Develop interactive prototypes based on the wireframes to simulate user interactions and test the design before development.**

**By following these expanded steps and incorporating detailed user analysis and content planning, you can create effective wireframes that serve as a solid foundation for your online learning platform's UI/UX design.**

## Conclusion:

Designing a wireframe for an online learning platform is essential for setting the foundation of a user-friendly and efficient interface. It ensures that the platform's key features are logically organized and easily accessible to users, contributing to a positive learning experience. The wireframe serves as a valuable reference point for designers and developers as they proceed with the platform's development and further design iterations.

| | |
|---|---|
| **Lab Assignment No.** | 8 |
| **Title** | Designing a Social Fitness App: Create wireframes and a prototype for a social fitness app that allows users to track workouts, connect with friends, and share progress. Design the user interface for logging exercises, setting goals, and incorporating social features. |
| **Roll No.** | |
| **Class** | BE |
| **Date of Completion** | |
| **Subject** | Computer Laboratory-II-UI/UX |
| **Assessment Marks** | |
| **Assessor's Sign** | |

# ASSIGNMENT No: 8

Title: Designing a Social Fitness App

Problem Statement :
Designing a Social Fitness App involves creating wireframes and a prototype for an application that allows users to track workouts, connect with friends, and share progress. The user interface must facilitate logging exercises, setting fitness goals, and incorporating social features to enhance user engagement and motivation.

Prerequisite:
- Knowledge of wireframing and prototyping tools
- Understanding of fitness app functionality
- Familiarity with user experience (UX) and user interface (UI) design principles

Software Requirements:
- Figma Tool

Hardware Requirements:
- PIV, 2GB RAM, 500 GB HDD

Learning Objectives:
- To design user-friendly and intuitive UI elements for exercise tracking.

Outcomes:
Participants will produce wireframes and a clickable prototype for a social fitness app. The wireframes will provide a visual representation of the app's layout, and the prototype will allow for user interaction and testing. The resulting design will support workout tracking, goal setting, and social engagement within the app.

Theory:

Designing a social fitness app requires a thoughtful approach to user engagement, motivation, and usability. Wireframing and prototyping are essential steps in this process:

1. Wireframing: This step involves creating a basic visual representation of the app's layout and structure without detailed design elements. Wireframes help in:
- Planning Layout: Structuring the arrangement of key features like workout logs, goal setting, and social interactions.
- Identifying Functionality: Defining the functional aspects, such as how users will log exercises, track their progress, and interact with friends.
- Enhancing Usability: Ensuring that the app's structure supports an intuitive and seamless user experience.

2. Prototyping: Prototyping involves creating an interactive model of the app that simulates user interactions. This step is crucial for:

-User Interaction Testing: Validating how users will interact with the app's features and flow.
- Gathering Feedback: Allowing users to provide feedback on the design and functionality, which helps in refining the user experience.
- Design Validation: Ensuring that the app's design meets user needs and expectations before moving to development.

Effective UI elements and social features are critical in a social fitness app. These features enhance user experience by:
- Motivating Users: Through goal-setting, progress tracking, and social interactions.
- Encouraging Engagement: By integrating social features like friend connections, workout challenges, and sharing achievements.

Steps Involved

1. Define Goals and Requirements:
- Project Objectives: Determine the main goals of the social fitness app, such as enhancing user engagement, providing workout tracking, and enabling social interactions.
- Target Audience: Identify the user base, including their fitness levels, preferences, and tech-savviness.
- Key Features: List essential features, such as:
- Workout Tracking: Logging exercises and monitoring progress.
- Goal Setting: Setting and tracking fitness goals.
- Social Features: Connecting with friends, sharing progress, and participating in challenges.

2. Research and User Analysis
- User Personas: Develop personas based on demographic data, fitness goals, and app usage habits.
- Competitive Analysis: Study similar fitness apps to identify successful features and areas for improvement.
- User Interviews and Surveys: Gather insights from potential users to understand their needs, preferences, and pain points.

3. Content Inventory
- List Content Elements:
  - Workout Logs : Exercise types, durations, and intensity levels.
  - Goal Setting : Types of goals (e.g., weight loss, muscle gain) and progress tracking.
  - Social Features : Friend lists, social feeds, and challenge options.
  - User Profiles : Personal information, workout history, and social connections.
  - Navigation Menus : Main navigation options, such as home, workouts, goals, and friends.

4. Create Wireframes
- Main Pages: Design wireframes for core pages:
- Homepage: Layout with featured workouts, user stats, and social feed.
- Workout Log Page: Interface for logging exercises, viewing past workouts, and tracking progress.
- Goal Setting Page: Design for setting and monitoring fitness goals.

- Social Page: Layout for connecting with friends, sharing achievements, and participating in challenges.
- Interactive Elements: Include basic interactive elements like buttons, forms, and navigation links.

5. Design Course Listings:
- Layout Design: Determine how courses or workouts will be displayed:
- Grid or List View: Choose between grid or list view for displaying workouts.
- Course Details: Include details like workout title, description, instructor, duration, and difficulty.
- Filters and Sorting Options: Provide options to filter and sort workouts by type, intensity, or duration.

6. Develop and Test Prototype
- Interactive Prototyping: Create a clickable prototype using Figma to simulate user interactions.
- Usability Testing: Conduct tests with real users to evaluate the app's usability and gather feedback.
- Refinement: Iterate on the design based on feedback to improve user experience and functionality.

Conclusion

Creating wireframes and a prototype for a social fitness app is a vital step in developing a user-friendly and engaging application. By visualizing the app's layout, features, and user interactions, designers can ensure that the app effectively supports workout tracking, goal setting, and social engagement. The clickable prototype allows for testing and validation of the app's user experience, making it an essential part of the app development process.

| | |
|---|---|
| **Lab Assignment No.** | 9 |
| **Title** | Use Figma tool for Improving the User Interface of a Fitness Tracking App: Improve the user interface of an existing fitness tracking app by focusing on simplicity, clarity, and motivational elements. Enhance features like tracking workouts, setting goals, and visualizing progress to create a more engaging and intuitive experience. |
| **Roll No.** | |
| **Class** | BE |
| **Date of Completion** | |
| **Subject** | Computer Laboratory-II-UI/UX |
| **Assessment Marks** | |
| **Assessor's Sign** | |

# ASSIGNMENT No: 9

**Title:** Use Figma tool for Improving the User Interface of a Fitness Tracking App: Improve the user interface of an existing fitness tracking app by focusing on simplicity, clarity, and motivational elements. Enhance features like tracking workouts, setting goals, and visualizing progress to create a more engaging and intuitive experience.

**Problem Statement:** Use Figma tool for Improving the User Interface of a Fitness Tracking App: Improve the user interface of an existing fitness tracking app by focusing on simplicity, clarity, and motivational elements. Enhance features like tracking workouts, setting goals, and visualizing progress to create a more engaging and intuitive experience.

**Prerequisite**: Proficiency in using Figma design tool, familiarity with the existing fitness tracking app's features and functionality, a strong understanding of user interface (UI) design principles, and knowledge of user experience (UX) best practices.

## Software Requirements: Figma Tool

## Hardware Requirements:

PIV, 2GB RAM, 500 GB HDD

**Learning Objectives:**

To apply principles of simplicity, clarity, and motivation to redesign the app's UI.

## Outcomes:

Participants will produce an improved user interface design for the fitness tracking app using Figma. The redesigned UI will prioritize simplicity, clarity, and motivation, resulting in an engaging and intuitive user experience for tracking workouts, setting goals, and visualizing progress.

## Theory: User Interface (UI) Design for Fitness Tracking Apps

User Interface (UI) Design is integral to creating an effective and enjoyable user experience for any application. For a fitness tracking app, the UI plays a crucial role in how users interact with and perceive the app. The key aspects to focus on are:

1. Simplicity : A simple UI ensures that users can navigate the app intuitively without confusion. This involves:

2- Clear Navigation : Simple and intuitive navigation menus and buttons that guide users effortlessly through the app.

3- Minimalist Design : Avoiding unnecessary elements that can clutter the interface and distract users from their main goals.

4- Clarity: Presenting information in a clear and concise manner is essential. This includes:

- Readable Text : Using legible fonts and appropriate sizes for all text elements.
- Organized Layouts : Structuring information logically, with visual hierarchies that highlight important details.

3. **Motivational Elements**: Incorporating features that keep users engaged and motivated. This could involve:
- Achievement Badges: Visual rewards for reaching milestones or completing challenges.
- Progress Tracking: Clear and visually appealing ways to track progress and celebrate successes.
- Social Features: Options to connect with friends, share achievements, and participate in challenges.

4. **Redesign Goals**: The purpose of redesigning the UI is to enhance usability and aesthetics. This involves:
- User-Centric Design: Ensuring that the design addresses user needs and preferences.
- Visual Appeal: Creating a visually pleasing interface that aligns with the fitness and wellness themes, enhancing overall user experience.

Steps Involved in Redesigning the UI for a Fitness Tracking App

1. **Understand the Existing App**
- **Explore the Current Interface** : Use the existing app thoroughly to understand its functionality and user experience.
- **Strengths** : Identify what aspects of the current design work well and are appreciated by users.
- **Weaknesses** : Note areas that are confusing, difficult to navigate, or visually unappealing.
- **Gather User Feedback** : Collect feedback from current users to identify pain points and areas for improvement.
- **Document Findings** : Create a comprehensive list of issues and improvement areas to guide the redesign process.

2. **Competitor Analysis:**
- Research Successful Apps: Analyze leading fitness tracking apps to understand their design approaches and user engagement strategies.
- Design Trends: Identify common UI/UX design trends, such as color schemes, typography, and layout styles.
- Features and Functionality: Evaluate what features are popular and effective in other apps.
- Benchmarking: Compare your app's current design with competitors to identify gaps and opportunities for improvement.

**3. Wireframe the Redesign:**

- Sketch Key Screens: Create wireframes for essential screens of the app, such as the dashboard, workout logging page, progress tracker, and social features.
- Simplify User Journey: Design wireframes to streamline the user flow, making it easier for users to achieve their goals.
- Improve Navigation: Ensure that navigation elements are intuitive and easily accessible.
- Add Motivational Elements: Incorporate spaces for features like achievement badges and progress summaries.
- Iterate: Refine wireframes based on feedback and testing to ensure they meet user needs and improve the overall experience.

**4. Visual Design**

- Develop Design Elements: Translate wireframes into high-fidelity designs, focusing on:
- Color Schemes: Choose colors that align with fitness and wellness themes, enhancing user engagement and visual appeal.
- Typography: Select fonts that are readable and fit the overall design aesthetic.
- Visual Consistency: Maintain a consistent design language across all screens and elements.
- Create Mockups: Develop detailed mockups for each screen, including all visual elements, to provide a clear representation of the final design.

**5. Progress Visualization**

- Redesign Charts and Graphs: Create clear and engaging visualizations to display workout history, achievements, and milestones.
- Chart Types: Use appropriate chart types, such as line graphs for progress over time and pie charts for distribution of workout types.
- Visual Appeal: Ensure that progress visualizations are not only functional but also visually appealing to motivate users.
- Feedback and Iteration: Test progress visualizations with users to ensure they effectively communicate the desired information and iterate based on feedback.

**Conclusion:**

By focusing on simplicity, clarity, and motivational elements, and following the detailed steps of understanding the existing app, performing competitor analysis, wireframing, visual design, and progress visualization, you can create a user-centric and visually appealing fitness tracking app. This redesign will enhance the overall user experience, making it easier for users to track their fitness journey, set and achieve goals, and stay motivated through engaging and intuitive features.

| | |
|---|---|
| **Lab Assignment No.** | 10 |
| **Title** | Product Packaging Mockup: Choose a product and create a mockup of its packaging design. Use a mockup tool that specializes in packaging design or graphic design. Design the product packaging, including the layout, colors, logos, and product visuals. Showcase the packaging design from different angles and perspectives. |
| **Roll No.** | |
| **Class** | BE |
| **Date of Completion** | |
| **Subject** | Computer Laboratory-II-UI/UX |
| **Assessment Marks** | |
| **Assessor's Sign** | |

# ASSIGNMENT No: 10

**Title:** Product Packaging Mockup: Choose a product and create a mockup of its packaging design. Use a mockup tool that specializes in packaging design or graphic design. Design the product packaging, including the layout, colors, logos, and product visuals. Showcase the packaging design from different angles and perspectives.

**Problem Statement:** Product Packaging Mockup: Choose a product and create a mockup of its packaging design. Use a mockup tool that specializes in packaging design or graphic design. Design the product packaging, including the layout, colors, logos, and product visuals. Showcase the packaging design from different angles and perspectives.

**Prerequisite**: Proficiency in a mockup tool specializing in packaging design or graphic design, knowledge of the selected product, a strong grasp of design principles, including layout, color theory, and branding.

## Software Requirements: Figma Tool

## Hardware Requirements:

PIV, 2GB RAM, 500 GB HDD

## Learning Objectives:

To select an appropriate product and understand its target audience and market.

## Outcomes:

Participants will produce a high-quality mockup of a product's packaging design using specialized tools. The packaging design will incorporate layout, colors, logos, and product visuals, and it will be presented from multiple angles, demonstrating its adaptability and visual appeal.

**Theory:** Packaging Design

Packaging Design is a vital component of product marketing and branding. It involves creating packaging that is both visually appealing and functional, ensuring that the product is protected while effectively communicating its value and identity to consumers. Key aspects of effective packaging design include:

- Visual Appeal: The design should attract attention and stand out on the shelves. It should be aesthetically pleasing to draw potential customers to the product.

- Functionality: Packaging must protect the product during transportation and handling. It should be easy to open and use while minimizing waste.

- Brand Alignment: Packaging should reflect the brand’s identity and values. This includes using brand colors, logos, and fonts that align with the overall brand image.

- Target Audience: Understanding the target audience helps in designing packaging that appeals to their preferences and needs. Consider factors such as age, gender, lifestyle, and purchasing behavior.

- Market Positioning: The packaging design should align with the product's positioning in the market. This means creating packaging that reflects the product's price point and competitive positioning.

**Steps Involved in Packaging Design**

**1. Select a Product**

- Product Details: Choose a specific product for which you will design packaging. Consider:

  - Size and Shape: The dimensions and shape of the product will influence the packaging design.

  - Target Audience: Understand who will be buying and using the product.

- Product Usage: Consider how the product will be used and any special requirements for its packaging.

**2. Research and Inspiration**

- Market Research: Study the market to understand consumer preferences and trends. Identify what works well and what doesn't in existing packaging designs.

- Competitor Analysis: Analyze competitors' packaging to see how they position their products and to identify gaps or opportunities.

- Inspiration Sources: Look for design inspiration from various sources, including design magazines, online portfolios, and packaging design websites.

**3. Choose a Mockup Tool**

- Design Tools: Select a graphic design tool or software that specializes in packaging mockups or 3D design. Examples include:

  - Adobe Illustrator: Useful for creating detailed design elements.

  - Adobe Photoshop: Ideal for adding textures and effects.

  - 3D Mockup Tools: Tools like Placeit or Mockup World that allow you to visualize packaging designs in 3D.

**4. Layout and Design**

- Create Layout: Begin the design process by sketching or digitally creating the packaging layout. Consider:

  - Information Hierarchy: Decide the order of information to ensure key details are prominent.

  - Branding Elements: Place logos, taglines, and other branding elements strategically.

  - Visuals and Images: Incorporate product images or graphics that enhance the design.

- Design Principles: Apply design principles such as contrast, alignment, and balance to create an appealing and functional package.

**5. Visualize the Mockup**

- Preview the Design: Use the chosen mockup tool to create a 3D visualization of the packaging design.

- Different Angles: Preview the design from various angles to assess its appearance and functionality.

- Realistic Rendering: Ensure the mockup accurately represents how the final product will look.

- Assess and Refine: Evaluate the mockup for any design flaws or improvements. Make necessary adjustments based on feedback or observations.

# Conclusion

Creating a product packaging mockup is a crucial part of the design process. It ensures that the packaging not only fulfills its practical function but also enhances the product's visual appeal and aligns with branding goals. By using specialized mockup tools or graphic design software, designers can effectively visualize and refine their packaging designs, showcasing them from multiple perspectives to demonstrate their versatility and aesthetics.

---.