

**LAB MANUAL OF**

Data Science Lab

Code: ITL601

Class: BE Information Technology Semester: VII (Rev-2019 ‘C’ Scheme)

Lab Incharge H.O.D

Mrs.Manisha Patil Dr. Pradip Mane

Mrs.Vedika Avhad

## College Vision

* To provide an environment to educate, encourage and explore students by facilitating innovative research, entrepreneurship, opportunities and employability to achieve social and professional goals.

## College Mission

* To foster entrepreneurship & strengthen industry institute interaction to enhance career opportunities for the employability of students.
* To encourage collaborations with industries and academic institutes in terms of projects & internships by creating area for Research and Development.
* To build up appropriate moral and ethical skills and to promote holistic development of students through various academic, social and cultural activities

## Department Vision

* To impart quality education in the field of Information Technology to meet the challenging needs of the society and industry.

## Department Mission

* To provide quality education to students by including Problem Solving, Teamwork and Leadership Skills to achieve their goals in the field of Information Technology.
* To develop skilled IT professionals with moral principles and empower them in lifelong learning.
* To educate students for global development including entrepreneurship, employability and the ability to apply technology to real life problems.

## Program Educational Objectives (PEO)

* Graduates will be successful with sound foundation in engineering fundamentals, trending technologies and entrepreneurship.
* Graduates will be able to identify and solve real world problems.
* Graduates will become ingenious and responsible citizens by demonstrating ethics with nurtured professional attitude

## Program Outcomes (POs)

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identity, formulate complex engineering problems reaching substantiated conclusions using principles of Computer Engineering.
3. **Design / development of solutions:** Design / develop solutions for complex engineering problems and design system components or processes that meet the specified needs with

appropriate consideration for the society.

1. **Conduct investigations of complex problems:** Use knowledge for the design of experiments, analysis, interpretation of data, and synthesis of the information to provide valid conclusions.
2. **Modern tool usage:** Create, select and apply appropriate techniques and modern engineering tools, including predictions and modeling to complex engineering activities with an understanding of the limitations.
3. **The engineer and society:** Apply the knowledge to assess social issues and the responsibilities relevant to engineering practices.
4. **Environment and sustainability:** Understand the impact of the professional engineering solutions in social and environmental contexts, and demonstrate the knowledge for sustainable development.
5. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
6. **Individual and teamwork:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
7. **Communication:** Communicate effectively such as being able to comprehend and write effective reports and design documentation, make effective presentations.
8. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management skills and apply the skills to manage projects effectively.
9. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## Program Specific Outcomes (PSO)

* Develop efficient IT based solutions by applying and integrating various domains like Artificial Intelligence, IoT, Computer Networks and Security to solve real time problems.
* Apply technical knowledge in the field of Information Technology to achieve successful career and to pursue higher studies for future endeavors.

## Lab Objectives

**The Lab experiments aims:**

## To apply reasoning for a problem in an uncertain domain.

## To discuss the solution after building a Cognitive application.

## To familiarize the students with the basics of Fuzzy Logic and Fuzzy Systems.

## To familiarize the students with Learning Architectures and Frameworks.

## To define and apply metrics to measure the performance of various learning algorithms..

## To enable students to analyze data science methods for real world problems.

## Lab Outcomes

**On successful completion, of course, learner/student will be able to:**

1. Implement reasoning with uncertainty.
2. Explore use cases of Cognitive Computing
3. Implement a fuzzy controller system.
4. Develop real life applications using learning concepts.
5. Evaluate performance of applications.
6. Implement and analyze applications based on current trends in Data Science

**Prerequisite**: Artificial Intelligence and Data Science-I, Python Programming, Data Mining & Business Intelligence.

## Hardware & Software Requirements:

|  |  |
| --- | --- |
| **Hardware Requirements** | **Software Requirements** |
| PC with following Configuration  1. Intel Core i3/i5/i7  2. 4 GB RAM  3. 500 GB Hard disk | Python, MySQL or Database Software |

**Lab /syllabus:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No.** | **Module** | **Detailed Content** | **Hours** | **LO**  **Mapping** |
| 1 | Uncertainty in AI | 1.Implement Inferencing with Bayesian Network in Python | 02 | LO1 |
| 2 | Cognitive Computing | 2.Building a Cognitive Healthcare application  3.Smarter cities: Cognitive Computing in Government  4.Cognitive computing in Insurance  5.Cognitive computing in Customer Service | 04 | LO2 |
| 3 |  | 6.Implementation of Fuzzy Membership Functions.  7.Implementation of fuzzy set Properties.  8.Design of a Fuzzy control system using Fuzzy tool. | 04 | LO3 |
| 4 | Introduction to Deep Learning | 9.Implementing Deep Learning Applications like  a. Image Classification System  b. Handwritten Digit Recognition System (like MNIST Dataset)  c. Traffic Signs Recognition System.  d. Image Caption Generator | 06 | LO4 |
| 5 | Advanced ML  Classification  Techniques | 10.Implementation of supervised learning algorithm like  a. Ada-Boosting  b. Random forests  11.Evaluation of Classification Algorithms. | 05 | LO4,LO5 |
| 6 | Mini-project on trends and applications in Data Science | 12.Build text/ image/ video/ audio based DS Applications such as  a. Chatbot  b. Document Classification  c. Sentiment Analysis  d. Bounding Box Detection  e. Music/Video Genre Classification | 05 | LO6 |

## Textbooks:

1. Stuart Russell and Peter Norvig, “Artificial Intelligence: A Modern Approach”, Third Edition, Pearson Education.
2. Judith S. Hurwitz, Marcia Kaufman, Adrian Bowles, “Cognitive Computing and Big Data Analytics”, Wiley India,2015.
3. S.N. Sivanandam, S.N. Deepa, “Principles of Soft Computing”, Wiley Publication.
4. Dr. S Lovelyn Rose, Dr. L Ashok Kumar, Dr. D Karthika Renuka, “Deep Learning Using Python”, Wiley India, 2020.
5. B. Uma Maheshwari, R. Sujatha, “Introduction to Data Science Practical Approach with R and Python”, Wiley India, 2021.
6. François Chollet, “Deep Learning with Python”, Manning Publications, 2018.
7. Han J, Kamber M, Pei J, “Data Mining Concepts and Techniques”, Third Edition, Morgan Kaufmann.

## References:

1. Deepak Khemani, “A First Course in Artificial Intelligence”, McGraw Hill Publication.
2. Ethem Alpaydin , “Introduction to Machine Learning”, PHI Learning Pvt. Ltd.
3. Jon Krohn, Grant Beyleveld, Aglae Bassens, “Deep Learning Illustrated: A Visual, Interactive Guide to Artificial Intelligence”, Pearson Education.
4. Prateek Joshi, “Artificial Intelligence with Python”, Packt Publishing.

# List of Experiments

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Experiment Title** | **Lab Outcome** |
| 1 | Implement Inferencing with Bayesian Network in Python | LO1 |
| 2 | Building a Cognitive Healthcare application | LO2 |
| 3 | Smarter cities: Cognitive Computing in Government | LO2 |
| 4 | To study and implement data preprocessing in NLP | LO2 |
| 5 | Implementation of Fuzzy Membership Functions. | LO3 |
| 6 | Implementation of fuzzy set Properties. | LO3 |
| 7 | Design of a Fuzzy control system using Fuzzy tool. | LO3 |
| 8 | Implementing Deep Learning Applications like  a. Image Classification System  b. Handwritten Digit Recognition System (like MNIST Dataset)  c. Traffic Signs Recognition System.  d. Image Caption Generator | LO4 |
| 9 | Implementation of supervised learning algorithm like  a. Ada-Boosting  b. Random forests | LO4,LO5 |
| 10 | Evaluation of Classification Algorithms. | LO4,LO5 |
| 11 | Build text/ image/ video/ audio based DS Applications such as  a. Chatbot  b. Document Classification  c. Sentiment Analysis  d. Bounding Box Detection  e. Music/Video Genre Classification | LO6 |

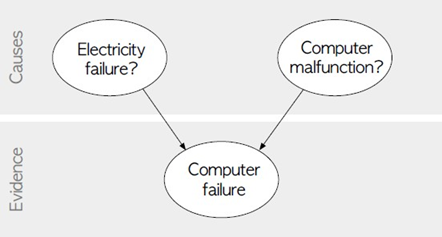
**Experiment No. 1**

**Aim:** Implement Inferencing with Bayesian Network in Python

**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.

****

**Step 1: Defining Things and How They Connect**

Creating a Bayesian network means deciding what things we want to understand and how they connect to each other.

**Step 1.1: Picking Things (Variables)**

In Python, we begin by choosing the things we want to understand. These things are like “variables.” If we’re predicting the weather, our variables could be**“Temperature,” “Rain,” and “Humidity.”**

from pgmpy.models import BayesianNetwork

from pgmpy.factors.discrete import TabularCPD

# Our Empty Bayesian Network

model = BayesianNetwork()

# Picking Our Variables

model.add\_node("Temperature")

model.add\_node("Rain")

model.add\_node("Humidity")

**Step 1.2: Connecting the Dots (Dependencies)**

Next, we need to tell our Python program how our variables are connected. We do this by creating **“Conditional Probability Distributions” (CPDs).**

Think of CPDs like little pieces of information that say how one thing depends on another. For example, we can say, “The chance of rain depends on the temperature.”

# How Variables Depend on Each Other

cpd\_temperature = TabularCPD(variable="Temperature", variable\_card=3, values=[[0.7], [0.2], [0.1]])

cpd\_rain = TabularCPD(

variable="Rain",

variable\_card=2,

values=[[0.4], [0.6]],

evidence=["Temperature"],

evidence\_card=[3],

)

cpd\_humidity = TabularCPD(

variable="Humidity",

variable\_card=2,

values=[[0.8], [0.2]],

evidence=["Rain"],

evidence\_card=[2],

)

# Putting It All Together

model.add\_cpds(cpd\_temperature, cpd\_rain, cpd\_humidity)

# Check if Everything Makes Sense

model.check\_model()

Now our program knows the connections between our variables. We’ve got the foundation of our Bayesian network!

**Step 2: Creating the Bayesian Network**

Creating the actual Bayesian network is simple. We add our variables and their dependencies to the model.

# Creating the Bayesian Network

model = BayesianNetwork()

model.add\_node("Temperature")

model.add\_node("Rain")

model.add\_node("Humidity")

model.add\_cpds(cpd\_temperature, cpd\_rain, cpd\_humidity)

# Making Sure It All Fits Together

model.check\_model()

Now, we have successfully built a Bayesian network in Python. It’s ready for some cool predictions and analysis!

**Step 3: Making Predictions**

In a Bayesian network, we use the information we’ve set up to make predictions or answer questions. We call this**“inference.”**

**Step 3.1: Finding Probabilities**

To find the probability of something happening, we can ask our Bayesian network. For instance, if we want to know the probability of rain when it’s hot, we can use Python to help us:

from pgmpy.inference import VariableElimination

inference = VariableElimination(model)

probability = inference.query(variables=["Rain"], evidence={"Temperature": 2})

print(probability)

This tells us the chances of rain when the temperature is hot.

**Step 3.2: Predicting Values**

Sometimes, we want to predict what’s most likely to happen. For example, if we have some information, we can predict the humidity:

predicted = inference.map\_query(variables=["Humidity"], evidence={"Rain": 1})

print(predicted)

This helps us make educated guesses.

**Step 3.3: Playing with Randomness**

In some cases, we like surprises! We can ask Python to give us random answers from our Bayesian network:

samples = model.sample(size=5)

print(samples)

This is like running simulations and seeing what might happen.

**Step 4: Seeing Is Believing**

Seeing your Bayesian network helps you understand it better. We can create a visual representation using Python.

**Step 4.1: Installing the Right Tools**

To draw our network, we need to install a couple of Python tools:

pip install networkx matplotlib

**Step 4.2: Making It Visual**

Now we can create a picture of our Bayesian network:

import networkx as nx

import matplotlib.pyplot as plt

# Turning Our Bayesian Network into a Picture

graph = model.to\_daft()

# Showing Our Picture

nx.draw(graph, with\_labels=True, node\_size=500, node\_color="skyblue", font\_size=10, font\_color="black")

plt.show()

**Code:**

**Output:**

**Conclusion:** We successfully studied to implement Inferencing with Bayesian Network in Python

# Experiment No: 2

**Aim:** Building a Cognitive Healthcare application (Heart Disease Prediction)

**Theory:**

Designing a typical cognitive application involves basic seven steps as discussed below:

1. Defining the objective
2. Defining Domain
3. Understanding the intended users and their attributes.
4. Defining questions and exploring insights.
5. Acquiring the relevant data sources.
6. Creating and refining the corpora.
7. Training and testing.
8. **Defining the objective:**

**Identify individuals at risk:** Determine who is more likely to develop heart disease in the future.

**Prevent heart disease:** Take proactive measures to reduce the risk of heart disease in at-risk individuals.

**Improve healthcare planning:** Assist healthcare providers in allocating resources and planning interventions for high-risk patients.

**Enhance patient outcomes:** Increase the chances of early detection and timely treatment of heart disease, ultimately saving lives and improving quality of life.

1. **Defining the domain:**

* **Healthcare domain**

**Clinical Data:**

Information collected during medical check-ups.

Includes age, gender, blood pressure, cholesterol levels, and family history.

**Lifestyle Factors:**

Habits and choices that impact heart health.

Includes smoking, alcohol consumption, diet, and physical activity.

**Symptoms and History:**

Patient-reported signs like chest pain, shortness of breath, and past heart-related events.

1. **Understanding the intended users and their attributes:**

**Doctors and Healthcare Professionals:**

Users: Cardiologists, general practitioners, nurses.

Attributes: Medical training and expertise, access to patient data, ability to interpret test results.

**Patients:**

Users: Individuals concerned about their heart health.

Attributes: Personal health history, lifestyle choices, willingness to provide data, basic health literacy.

**Researchers:**

Users: Scientists studying heart disease.

Attributes: Advanced knowledge of medical research, access to large datasets, statistical and analytical skills.

**Healthcare Systems:**

Users: Hospitals, clinics, and healthcare organizations.

Attributes: Access to patient records, resources for data storage and analysis, compliance with privacy regulations.

**Health Tech Companies:**

Users: Developers and data scientists.

Attributes: Technical skills, access to machine learning tools, knowledge of data privacy and security.

**Government and Public Health Agencies:**

Users: Policymakers, epidemiologists.

Attributes: Public health expertise, access to population health data, decision-making authority.

**General Population:**

Users: People interested in heart disease prevention.

Attributes: Basic understanding of health, willingness to adopt preventive measures, access to healthcare resources.

1. **Defining questions and exploring insights:**

**Questions about Heart Disease Prediction:**

What kind of data is used for prediction?

Information about your health, lifestyle, family history, and medical tests.

What are the common risk factors for heart disease?

Smoking, high blood pressure, high cholesterol, obesity, and diabetes.

How do these risk factors affect heart disease prediction?

They increase the likelihood of developing heart disease.

**Insights into Heart Disease Prediction:**

Machine Learning Helps:

Computers use data and math to find patterns that can predict who might get heart disease.

Early Warning Signs:

Predictive models can identify warning signs before symptoms appear.

Personalized Risk Assessment:

Predictions are tailored to each person's unique health data.

Prevention is Key:

Knowing your risk can motivate you to make healthier choices.

Regular Check-Ups Matter:

Regular medical check-ups provide valuable data for prediction.

1. **Acquiring the relevant data sources:**

**Identify Data Needs:**

Determine what data you need for heart disease prediction. This may include medical, lifestyle, and demographic information.

1. **Creating and refining the corpora:**

* **Creating a Corpus:**

**Data Collection:** Gather various types of data related to heart health. This can include medical records, patient information, test results, and lifestyle factors like diet and exercise.

**Organizing Data:** Arrange the collected data in a systematic and organized way. This might involve creating spreadsheets or databases to store the information.

**Data Cleaning:** Check the data for errors or inconsistencies. Remove or correct any inaccuracies to ensure the data is reliable.

**Data Labeling:** Assign labels to the data that indicate whether a person has heart disease or not. This labeling is essential for training machine learning models.

**Data Splitting:** Divide the data into two or more sets, typically a training set, validation set, and testing set. This separation helps evaluate the performance of prediction models.

* **Refining the Corpus:**

**Feature Selection:** Choose the most relevant information or features from the data that are likely to contribute to accurate heart disease prediction. For example, age, cholesterol levels, and blood pressure are often important features.

**Imbalanced Data Handling:** If there's an imbalance in the number of people with and without heart disease, balance the dataset to prevent bias in the prediction model.

**Missing Data Handling:** Address missing data points by filling them in with reasonable values or using imputation techniques.

**Model Training:** Use machine learning or statistical techniques to build predictive models based on the refined dataset. This involves training the model to recognize patterns in the data that indicate the likelihood of heart disease.

1. **Training and testing:**

* **Training the Model:**

**Data Collection:** Gather a large dataset that includes information about people's health, like age, gender, blood pressure, cholesterol levels, and whether they have heart disease or not.

**Data Preparation:** Clean and organize the data, making sure it's in a format the computer can understand. This includes handling missing values and converting text into numbers.

**Splitting Data:** Divide the dataset into two parts - one for training and one for testing. The training data is used to teach the model, while the testing data is used to check how well it has learned.

**Choosing a Model:** Select a machine learning algorithm or model, like decision trees or neural networks, to analyze the training data and learn the patterns that might indicate heart disease.

**Training Process:** Feed the training data into the model and let it learn from the patterns in the data. The model adjusts its internal parameters to make predictions based on the input data.

* **Testing the Model:**

**Data Preparation:** Similar to the training data, clean and format the testing data so it's ready for the model.

**Prediction:** Feed the prepared testing data into the trained model and let it make predictions about whether each person has heart disease or not.

Testing data = 20%

**Conclusion:** Thus, we have successfully build a Cognitive Healthcare application (Heart Disease prediction)

**Experiment No. 3**

**Aim:** Smarter Cities: Cognitive Computing in Government

**Theory:**

This case study explores the implementation of cognitive computing in government initiatives aimed at building smarter cities. Cognitive computing, a subset of artificial intelligence, has the potential to revolutionize urban governance by enabling data-driven decision-making, optimizing city services, and improving the quality of life for citizens. We will examine the experiences of a fictional city named "Techville" to illustrate the practical implications of cognitive computing in government.

**Designing a typical cognitive application involves basic seven steps as discussed below:**

1. Defining the objective
2. Defining the domain
3. Understanding the intended users and their attributes
4. Defining questions and exploring insights.
5. Acquiring the relevant data sources
6. Creating and refining the corpora
7. Training and Testing

**1. Defining the Domain:**

The domain for this multifaceted experiment is "Smart Cities," encompassing a wide array of critical areas, including but not limited to transportation, healthcare, infrastructure development, public safety, and environmental sustainability. These areas collectively form the foundation upon which smarter, more efficient cities can be built.

**2. Understanding the Intended Users and Their Attributes:**

In this research endeavor, our target audience spans across a spectrum of professionals, including government officials, urban planners, data scientists, and policymakers. These individuals bring diverse attributes to the table, ranging from varying levels of expertise in urban planning and data analysis to nuanced insights into the intricacies of government operations.

**3. Defining Questions and Exploring Insights:**

1. How can cognitive computing technologies be effectively applied to enhance urban planning and governance practices?
2. What are the most pressing challenges and impediments when implementing cognitive computing in the public sector?
3. How might cognitive computing bolster public safety and emergency response mechanisms within urban areas?
4. What wealth of data sources and derived insights can be harnessed to facilitate well-informed decision-making for the realization of smarter cities?
5. In what ways can cognitive computing optimize the efficiency and sustainability of public transportation systems in urban locales?
6. What ethical and social considerations are pertinent to the integration of cognitive computing in government operations, and how can potential issues be mitigated?

**4. Acquiring Relevant Data Sources:**

To address the aforementioned questions comprehensively, we will embark on an extensive data acquisition journey, encompassing:

* Government databases and records, providing historical and operational data.
* Urban sensor data, such as traffic patterns, pollution levels, and weather conditions, to monitor and respond to real-time urban challenges.
* Social media data, enabling sentiment analysis and public perception monitoring.
* Geographic information system (GIS) data, facilitating spatial analysis and urban planning.
* Public transportation data, essential for optimizing transportation systems.
* Healthcare and public health data, contributing to urban health management.

**5. Creating and Refining the Corpora:**

The process of creating and refining our datasets involves meticulous data preprocessing and cleaning procedures to ensure data quality and consistency. This may necessitate transformations, normalization, and integration of data from diverse sources to construct a unified, comprehensive corpus suitable for in-depth analysis.

**6. Training and Testing:**

1. Machine Learning Models: We will develop and fine-tune machine learning models to predict urban trends, optimize resource allocation, and empower data-driven decision-making.
2. Natural Language Processing (NLP): Employing cutting-edge NLP techniques, we will analyze textual data, such as government reports, citizen feedback, and sentiment expressed on social media platforms.
3. Cognitive Computing Platforms: Integration of cognitive computing platforms like IBM Watson or Microsoft Azure Cognitive Services will enable the creation of intelligent chatbots, virtual assistants, and recommendation systems to enhance government services and citizen engagement.

* Testing and evaluation will encompass:
* Rigorous cross-validation to assess the robustness and reliability of our models.
* Real-world application scenarios to gauge the practical effectiveness of cognitive computing in government operations.
* Soliciting feedback from government officials and stakeholders to validate the utility and feasibility of our cognitive computing applications.

**Conclusion:** Cognitive computing holds immense potential for transforming government operations and creating smarter cities

# Experiment No. 4

**Aim :** **: To study and implement data preprocessing in NLP**

**Theory:**

Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

The essence of Natural Language Processing lies in making computers understand the natural language. That’s not an easy task though. Computers can understand the structured form of data like spreadsheets and the tables in the database, but human languages, texts, and voices form an unstructured category of data, and it gets difficult for the computer to understand it, and there arises the need for Natural Language Processing. There’s a lot of natural language data out there in various forms and it would get very easy if computers can understand and process that data. We can train the models in accordance with expected output in different ways.

Breaking a complex problem into a number of small problems, making models for each of them and then integrating these models. A similar thing is done in NLP. We can break down the process of understanding English for a model into a number of small pieces.

Natural Language Processing (NLP) is a subfield of computer science and artificial intelligence that deals with the interaction between computers and human languages. The primary goal of NLP is to enable computers to understand, interpret, and generate natural language, the way humans do.

NLP involves a variety of techniques, including computational linguistics, machine learning, and statistical modeling. These techniques are used to analyze, understand, and manipulate human language data, including text, speech, and other forms of communication.

Some of the main applications of NLP include language translation, speech recognition, sentiment analysis, text classification, and information retrieval. NLP is used in a wide range of industries, including finance, healthcare, education, and entertainment, to name a few.

There are three main components:

**1. Tokenization**: Tokenization is about splitting strings of text into smaller pieces, or “Tokens”.Paragraphs can be tokenized into sentences and sentences can be tokenized into words.

**2. Normalization:** Normalization aims to put all text on a level playing field, e.g., converting all characters to lowercase.

**3. Noise Removal:** Noise Removal cleans up the text, e.g., removing extra whitespaces.

Steps in Text pre-processing are:

**● Remove HTML Tags:**

This is useful if we scrap the data from different websites. We might end up having HTML strings as part of our text. Since these tags are not useful for our NLP tasks, it is better to remove them.

**● Remove Extra Whitespaces:**

These are unwanted extra spaces between words. They have no significant meaning; hence it is better to remove them.

**● Convert accented characters:**

Words with accent marks like “latté” and “café” can be converted and standardized to just “latte” and “cafe”,else our NLP model will treat latté and latte as different words even though they are referring to the samething.

**● Expand contractions:**

Contraction is the shortened form of a word like don't stands for do not, and aren't stands for are not. This,we need to expand this contraction in the text data for better analysis. You can easily get the dictionary of contractions on google or create your own and use the `re` module to map the contractions.

● **Remove special characters:**

Special characters typically include any character that is not a letter or number, such as punctuation and whitespace. Removing special characters from a string `result` in a string containing only letters and numbers.

**● Lowercase all texts:**

If the text is in the same case, it is easy for a machine to interpret the words because the lower case and uppercase are treated differently by the machine. So, we need to make the text in the same case and the most preferred case is a lowercase to avoid such problems.

**● Convert number words to numeric form:**

This step involves the conversion of number words to numeric form e.g., seven to 7 to standardize text.

**● Remove numbers:**

While analyzing text for certain applications, the numbers in the string are not of importance. In such cases,we prefer to remove numbers and work on the words only.

**● Remove stop words:**

Stop words are the most commonly occurring words in a text which do not provide any valuable information,Stop words like they, there, this, where, etc. are some of the stopwords. NLTK library is a common library that is used to remove stopwords and includes approximately 180 stopwords which it removes. If we want to add any new word to a set of words, then it is easy using the add method.

● **Stemming:**

Stemming is a process to reduce the word to its root stem for example run, running, runs, runed derived from the same word as run. Basically, stemming do is remove the prefix or suffix from words like ing, s, es, etc.NLTK library is used to stem the words. The stemming technique is not used for production purposes because it is not a very efficient technique and most of the time it stems unwanted words. So, to solve the problem another technique came into the market as Lemmatization, there are various types of stemming algorithms like porter stemmer, and snowball stemmer. Porter stemmer is widely used present in the NLTK library

**● Lemmatization:**

Lemmatization is similar to stemming, used to stem the words into root words but differs in working Actually, Lemmatization is a systematic way to reduce the words into their lemma by matching them with a language dictionary.

**● Sentence Tokenization:** Splitting sentences in the paragraph

**● Word Tokenization:**Splitting words in a sentence.

### Text Lowercase:

We can use text\_lowercase(input\_str)function to convert the text to lowercase.

### Remove numbers:

We can either remove numbers or convert the numbers into their textual representations.   
We can use regular expressions to remove the numbers. 

We can also convert the numbers into words. This can be done by using the inflect library.

### Remove punctuation:

Step 1: import library string which includes attribute string.Punctuation which includes many built in punctuation characters

Step 2: Use .maketrans() method which takes three arguments, the first two of which are [empty strings](https://datagy.io/python-check-empty-string/), and the third is the list of punctuation we want to remove. This tells the function to replace all punctuation with none.

Step 3:pass above string to translate function.

### Remove whitespaces:

We can use the join and split function to remove all the white spaces in a string.

### Stemming:

Step 1: import files like word\_tokenize, PorterStemmer

Step 2:Tokenize the words.

Step 3: Pass the tokenize words to stemmer.stem function.

### Lemmatization:

Step 1: import files like WordNetLemmatizer, word\_tokenize

Step 2:Tokenize the words.

Step 3: Pass the tokenize words to lemmatizer.lemmatize function.

**Code:**

**Output:**

**Conclusion:** We successfully studied to implement data preprocessing in NLP

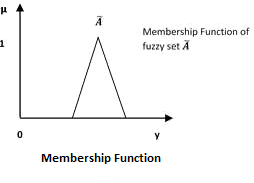
# Experiment No. 5

**Aim:** Implementation of Fuzzy membership functions.

**Theory:**

Fuzzy membership function

We already know that fuzzy logic is not logic that is fuzzy but logic that is used to describe fuzziness. This fuzziness is best characterized by its membership function. In other words, we can say that membership function represents the degree of truth in fuzzy logic.



List of membership functions that can be used as LMF and UMF functions:

|  |  |  |
| --- | --- | --- |
| **Membership function** | **Description** | **List of parameters** |
| zero\_mf | All zero membership function | None |
| singleton\_mf | Singleton membership function | Singleton’s center and height |
| const\_mf | Constant membership function | Constant membership function’s height |
| tri\_mf | Triangular membership function | The left end, the center, the right end, and the height of the triangular membership function |
| ltri\_mf | Left triangular membership function | The right end, the center, and the height of the triangular membership function |
| rtri\_mf | Right triangular membership function | The left end, the center, and the height of the triangular membership function |
| trapezoid\_mf | Trapezoidal membership function | The left end, the left center, the right center, the right end, and the height of the trapezoidal membership function |
| gaussian\_mf | Gaussian membership function | The center, the standard deviation, and the height of the gaussian membership function |
| gauss\_uncert\_mean\_umf | Gaussian with uncertain mean UMF | The lower limit of mean, the upper limit of mean, the standard deviation, and the height of the Gaussian membership function |
| gauss\_uncert\_mean\_lmf | Gaussian with uncertain mean LMF | The lower limit of mean, the upper limit  of mean, the standard deviation, and the height of the Gaussian membership function |
| gauss\_uncert\_std\_umf | Gaussian with uncertain standard deviation UMF | The center, the lower limit of std., the upper limit of std., and the height of the gaussian membership function |
| gauss\_uncert\_std\_lmf | Gaussian with uncertain standard deviation LMF | The center, the lower limit of std., the upper limit of std., and the height of the gaussian membership function |

It must be noticed that the parameters of the introduced functions are passed as a list with items mentioned, respectively.

Creating an IT2FS

The constructor function of the IT2FS class has six parameters, listed as below:

domain: The universe of discourse of the interval type 2 fuzzy set

umf: The UMF of the interval type 2 fuzzy set

umf\_params: The parameters of the UMF function

lmf: The LMF of the interval type 2 fuzzy set

lmf\_params: The parameters of the LMF function

check\_set: Boolean with default False value. If it is set as True then while the set is creating the condition of LMF(x)<=UMF(x) is checked for each x in domain.

Example 1

The first example demonstrates how to create an IT2FS with trapezoid LMF and triangular UMF functions:

from pyit2fls import IT2FS, trapezoid\_mf, tri\_mf from numpy import linspace

mySet = IT2FS(linspace(0., 1., 100),

trapezoid\_mf, [0, 0.4, 0.6, 1., 1.],

  tri\_mf, [0.25, 0.5, 0.75, 0.6])

In the first line, the IT2FS class and trapezoid and triangular membership functions are imported from the toolkit. Also in the second line, from the NumPy, the linspace is imported for creating the domain of the set. Then, using the IT2FS, the interval type 2 fuzzy set, named mySet, is created.

numpy.linspace:

The NumPy linspace function (sometimes called np.linspace) is a tool in Python for creating numeric sequences.It’s somewhat similar to the NumPy arange function, in that it creates sequences of evenly spaced numbers structured as a NumPy array.

Matplotlib Pyplot:

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

**Code:**

**Output:**

**Conclusion:**

**Experiment No. 6**

**Aim:** Design of Fuzzy control system using Fuzzy set properties.

# Theory:

# The well-known operations which can be performed on fuzzy sets are the operations of union, intersection, complement, algebraic product and algebraic sum. In addition to these operations, new operations called "bounded-sum" and In addition to these operations, new operations called "bounded-sum" and "bounded-difference" are introduced.

# 1.Union of fuzzy sets

# Union of fuzzy sets consists of every element that falls into either set. The value of the membership value is will be the largest membership value of the element in either set.

# Let A(x) and B(x) are two fuzzy sets for all x ∈ X,

# Union of fuzzy sets is denoted by (AUB)(x) and the membership function value is determined as follows

# µ (AUB)(x)= max{µA(x),µB(x)}

# Note: Union is analogous to logical OR operation.

# 

# Fig 6.1 Union of fuzzy sets

# 2. Intersection of fuzzy set

# Inter section of a fuzzy sets define how much of the element belongs to both sets. May have different degrees of membership in each set. The degree of membership is the lower membership in both sets of each element.

# Let A(x) and B(x) are two fuzzy sets, the intersection of is denoted by (A∩B)(x) and the membership function value is given as follows

# µ (A∩B)(x)= min{µA(x),µB(x)}

# Intersection is analogous to logical AND operation.

# 

# Fig 6.2 Intersection of fuzzy sets

# 3.Complement of fuzzy set

# The complement is the opposite of the set. The complement of a fuzzy set is denoted by Ā(x) and is defined with respect to the universal set X as follows: Ā(x) = 1- A(x) for all x ϵ X

# 

# Fig 6.3 Complement of fuzzy set

# 4.Algebraic product of fuzzy sets

# The Algebraic product of two fuzzy sets A(x) and B(x) for all x ∈ X, is denoted by A(x).B(x) and defined as follows

# µ A.B(x)= {(x, µA(x).µB(x)), x ϵ X }

# 5.Algebraic sum of two fuzzy sets

# The Algebraic sum of two fuzzy sets A(x) and B(x) for all x ∈ X, is denoted by

# A(x) + B(x) and defined as follows

# µA+B(x)= {(x,µA+B(x), x ϵ X } Where µA+B(x) = µA(x) +µB(x) - µA(x).µB(x)

# 6.Bounded sum of two fuzzy sets

# The bounded sum of two fuzzy sets A(x) and B(x) for all x ∈ X, is denoted by A(x)⊕ B(x)

# and defined as follows = µ{ A⊕ B (x), x ∈ X }

# Where µA⊕ B(x)= min[1,µA(x)+µB(x)]

# 7.Bounded difference of two fuzzy sets

# The Algebraic difference of two fuzzy sets A(x) and B(x) for all x ∈ X, is denoted by A(x)-B(x) and defined as follows

# µAOB(x)= max[0,µA(x)-µB(x)]

**Code:**

**Output:**

**Conclusion:**Thus, we have studied and designed the fuzzy control system using fuzzy set properties.

# Experiment No. 7

**Aim:** Design of a Fuzzy control system using Fuzzy tool

# Theory: Fuzzy Inference System is the key unit of a fuzzy logic system having decision making as its primary work. It uses the “IF…THEN” rules along with connectors “OR” or “AND” for drawing essential decision rules.

# ARCHITECTURE

# 1. RULE BASE: It contains the set of rules and the IF-THEN conditions provided by the experts to govern the decision-making system, on the basis of linguistic information. Recent developments in fuzzy theory offer several effective methods for the design and tuning of fuzzy controllers. Most of these developments reduce the number of fuzzy rules.

# 2. FUZZIFICATION: It is used to convert inputs i.e. crisp numbers into fuzzy sets. Crisp inputs are basically the exact inputs measured by sensors and passed into the control system for processing, such as temperature, pressure, rpm’s, etc.

# 3. INFERENCE ENGINE: It determines the matching degree of the current fuzzy input with respect to each rule and decides which rules are to be fired according to the input field. Next, the fired rules are combined to form the control actions.

# 4. DEFUZZIFICATION: It is used to convert the fuzzy sets obtained by the inference engine into a crisp value. There are several defuzzification methods available and the best-suited one is used with a specific expert system to reduce the error.

# 

# Membership function A graph that defines how each point in the input space is mapped to a membership value between 0 and 1. Input space is often referred to as the universe of discourse or universal set (u), which contains all the possible elements of concern in each particular application.

# Design of Fuzzy Controllers

# Steps in Designing Fuzzy Logic Control

# Identify the system input variable ,their ranges and membership functions.

# Identify the output variables,their ranges and membership functions

# Identify rules that describes the relations of the input to outputs.

# Determine the defuzzifier method of combining fuzzy rules into system outputs.

**Code:**

**Output:**

**Conclusion:** We have successfully studied to design of a Fuzzy control system using Fuzzy tool.

# Experiment No. 8

**Aim:** Implementing Deep Learning Applications like Handwritten Digit Recognition System

**Theory:** The handwritten digit recognition is the ability of computers to recognize human handwritten digits. It is a hard task for the machine because handwritten digits are not perfect and can be made with many different flavors. The handwritten digit recognition is the solution to this problem which uses the image of a digit and recognizes the digit present in the image.

**The MNIST dataset**

This is probably one of the most popular datasets among machine learning and deep learning enthusiasts. The [MNIST dataset](http://yann.lecun.com/exdb/mnist/) contains 60,000 training images of handwritten digits from zero to nine and 10,000 images for testing. So, the MNIST dataset has 10 different classes. The handwritten digits images are represented as a 28×28 matrix where each cell contains grayscale pixel value.

## Step 1:Import the libraries and load the dataset

First, we are going to import all the modules that we are going to need for training our model. The Keras library already contains some datasets and MNIST is one of them. So we can easily import the dataset and start working with it. The **mnist.load\_data()** method returns us the training data, its labels and also the testing data and its labels.

import kerasfrom keras.datasets

import mnist from keras.models

import Sequential from keras.layers

import Dense, Dropout, Flatten from keras.layers

import Conv2D, MaxPooling2D from keras

import backend as K

# the data, split between train and test sets(x\_train, y\_train), (x\_test, y\_test) =mnist.load\_data() print(x\_train.shape, y\_train.shape)

## Step 2:Preprocess the data

The image data cannot be fed directly into the model so we need to **perform some operations and process the data** to make it ready for our neural network. The dimension of the training data is

(60000,28,28). The CNN model will require one more dimension so we reshape the matrix to shape (60000,28,28,1).

x\_train = x\_train.reshape(x\_train.shape[0], 28, 28, 1)

x\_test = x\_test.reshape(x\_test.shape[0], 28, 28, 1)

input\_shape = (28, 28, 1)

# convert class vectors to binary class matrices

y\_train=keras.utils.to\_categorical(y\_train,num\_classes)

y\_test = keras.utils.to\_categorical(y\_test, num\_classes) x\_train = x\_train.astype('float32')

x\_test = x\_test.astype('float32') x\_train /= 255

x\_test /= 255

print('x\_train shape:', x\_train.shape) print(x\_train.shape[0], 'train samples') print(x\_test.shape[0], 'test samples')

## Step 3:Create the model

Now we will **create our CNN model** in Python data science project. A CNN model generally consists of convolutional and pooling layers. It works better for data that are represented as grid structures, this is the reason why CNN works well for image classification problems. The dropout layer is used to deactivate some of the neurons and while training, it reduces offer fitting of the model. We will then compile the model with the Adadelta optimizer.

batch\_size = 128

num\_classes = 10

epochs = 10

model = Sequential()

model.add(Conv2D(32,kernel\_size=(3,3),activation='relu',input\_shape=input\_shape)) model.add(Conv2D(64,(3, 3),activation='relu'))

model.add(MaxPooling2D(pool\_size=(2,2)

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(256,activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(num\_classes,activation='softmax'))

model.compile(loss=keras.losses.categorical\_crossentropy,optimizer=keras.optimizers.Adadelta(),metrics=['accuracy'])

## Step 4:Train the model

The **model.fit() function** of Keras will start the training of the model. It takes the training data,validation data,epochs,and batch size.It takes some time to train the model. After training, we save the weights and model definition in the ‘mnist.h5’ file.

hist=model.fit(x\_train,y\_train,batch\_size=batch\_size,epochs=epochs,verbose=1,validation\_data=x\_test, y\_test))

print("The model has successfully trained") model.save('mnist.h5')

print("Saving the model as mnist.h5")

## Step 5:Evaluate the model

score = model.evaluate(x\_test, y\_test, verbose=0) print('Test loss:', score[0])

print('Test accuracy:', score[1])

## Step 6:Create GUI to predict digits

Now for the GUI, we have created a new file in which we build an interactive window to draw digits on canvasand with a button, we can recognize the digit. The Tkinter library comes in the Python standard library. We have created a function predict\_digit()that takes the image as input and then uses the trained model to predict the digit.

Then we create the App classwhich is responsible for building the GUI for our app. We create a canvas where we can draw by capturing the mouse event and with a button, we trigger the predict\_digit() function and display the results.

Here’s the full code for our gui\_digit\_recognizer.py file:

# Code:

**Output:**

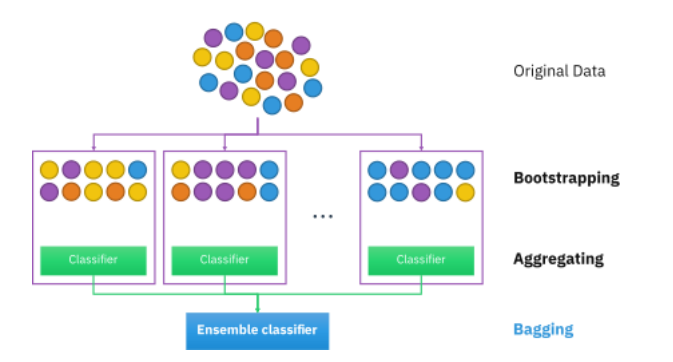
**Conclusion:** Thus we have successfully Implemented Handwritten Digit Recognition System using Deep learning .

## Experiment No. 9

**Aim:** Implementation of supervised learning algorithm like Ada –boosting and bagging.

**Theory:**

Bagging: A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregates their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it. Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N examples(or data) from the original training dataset – where N is the size of the original training set. The training set for each of the base classifiers is independent of each other. Many of the original data may be repeated in the resulting training set while others may be left out. Bagging, also known as Bootstrap Aggregation, is the ensemble technique used by random forest.Bagging chooses a random sample/random subset from the entire data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as row sampling. This step of row sampling with replacement is called bootstrap. Now each model is trained independently, which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting, is known as aggregation.

****

Bagging reduces overfitting (variance) by averaging or voting, however, this leads to an increase in bias, which is compensated by the reduction in variance though.

Steps to Perform Bagging

• Consider there are n observations and m features in the training set. You need to select a random sample from the training dataset without replacement

• A subset of m features is chosen randomly to create a model using sample observations

• The feature offering the best split out of the lot is used to split the nodes

• The tree is grown, so you have the best root nodes

• The above steps are repeated n times.

It aggregates the output of individual decision trees to give the best prediction

**Random Forest Algorithm**

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, **"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

**The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.**



One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables, as in the case of regression, and categorical variables, as in the case of classification. It performs better for classification and regression tasks. In this tutorial, we will understand the working of random forest and implement random forest on a classification task.

**Code:**

**Output:**

**Conclusion:** Thus we have successfully implemented Ada –boosting and bagging supervised algorithm.

**Experiment No. 10**

**Aim**:Evaluation of Classification Algorithms.

**Theory:**The Supervised Machine Learning algorithm can be broadly classified into Regression and Classification Algorithms. In Regression algorithms, we have predicted the output for continuous values, but to predict the categorical values, we need Classification algorithms.

## What is the Classification Algorithm?

The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups. Such as, **Yes or No, 0 or 1, Spam or Not Spam, cat or dog,** etc. Classes can be called as targets/labels or categories.

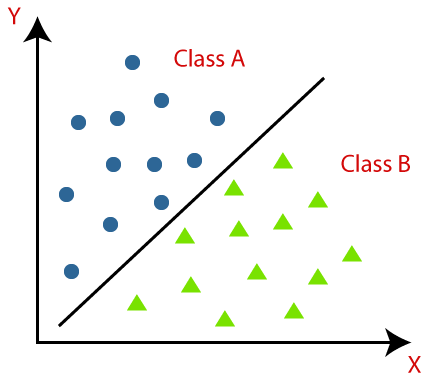
In classification algorithm, a discrete output function(y) is mapped to input variable(x).

y=f(x), where y = categorical output

The best example of an ML classification algorithm is **Email Spam Detector**.

The main goal of the Classification algorithm is to identify the category of a given dataset, and these algorithms are mainly used to predict the output for the categorical data.

Classification algorithms can be better understood using the below diagram. In the below diagram, there are two classes, class A and Class B. These classes have features that are similar to each other and dissimilar to other classes.



The algorithm which implements the classification on a dataset is known as a classifier. There are two types of Classifications:

**Binary Classifier:** If the classification problem has only two possible outcomes, then it is called as Binary Classifier.

**Examples:** YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc.

**Multi-class Classifier:** If a classification problem has more than two outcomes, then it is

Called as Multi-class Classifier.

**Example:** Classifications of types of crops, Classification of types of music.

## Types of ML Classification Algorithms:

Classification Algorithms can be further divided into the Mainly two category:

**Linear Models**

* + Logistic Regression
  + Support Vector Machines

**Non-linear Models**

* + K-Nearest Neighbours
  + Kernel SVM
  + Naïve Bayes
  + Decision Tree Classification
  + Random Forest Classification

## Evaluating a Classification model:

Once our model is completed, it is necessary to evaluate its performance; either it is a Classification or Regression model. So for evaluating a Classification model, we have the following ways:

**1. Log Loss or Cross-Entropy Loss:**

It is used for evaluating the performance of a classifier, whose output is a probability value between the 0 and 1.For a good binary Classification model, the value of log loss should be near to 0.The value of log loss increases if the predicted value deviates from the actual value.The lower log loss represents the higher accuracy of the model.For Binary classification, cross-entropy can be calculated as:

**2. Confusion Matrix:**

The confusion matrix provides us a matrix/table as output and describes the performance of the model.It is also known as the error matrix.The matrix consists of predictions result in a summarized form, which has a total number of correct predictions and incorrect predictions. The matrix looks like as below table:

|  |  |  |
| --- | --- | --- |
|  | **Actual Positive** | **Actual Negative** |
| Predicted Positive | True Positive | False Positive |
| Predicted Negative | False Negative | True Negative |

From the confusion matrix, we can find the following metrics

**Accuracy:**Accuracy is used to measure the performance of the model. It is the ratio of Total correct instances to the total instances.

**Accuracy =(TP+TN)/(TP+FP+TN+FN)**

**Precision:**Precision is a measure of how accurate a model’s positive predictions are. It is defined as the ratio of true positive predictions to the total number of positive predictions made by the model.

**Precision=TP/(TP+FP)**

**Recall:**Recall measures the effectiveness of a classification model in identifying all relevant instances from a dataset. It is the ratio of the number of true positive (TP) instances to the sum of true positive and false negative (FN) instances.

**Recall= TP/(TP+FN)**

**F1-Score:**F1-score is used to evaluate the overall performance of a classification model. It is the harmonic mean of precision and recall,

**F1-Score=(2\*Precision\*Recall)/ (Precision+Recall)**

**3. AUC-ROC curve:**ROC curve stands for **Receiver Operating Characteristics Curve** and AUC stands for **Area Under the Curve**.

It is a graph that shows the performance of the classification model at different thresholds.To visualize the performance of the multi-class classification model, we use the AUC-ROC Curve.The ROC curve is plotted with TPR and FPR, where TPR (True Positive Rate) on Y-axis and FPR(False Positive Rate) on X-axis.

Steps to

Step 1:Import the necessary libraries like sklearn.dataset,sklearn.model\_selection,sk.tree,sklearn\_metrics,seborn,matplotlib.pyplot

Step 2: Load the dataset

Step 3Train the model

Step 4: do the preduction

Step 5:compute the confusion matrix

Step 6:Plot the confusion matrix.

Step 7:Find precision, recall,F1\_score

**Code:**

**Output:**

**Conclusion:** Thus we have successfully studiedevaluation of different classifiers.

**Experiment No. 11**

**Aim**: Build text/ image/ video/ audio based DS Applications such as

a. Chatbot

b. Document Classification

c. Sentiment Analysis

d. Bounding Box Detection

e. Music/Video Genre Classification

**Theory:**

1. Introduction

1.1 Brief overview of the project

1.2 Objectives and goals

2. Problem Statement

2.1 Clearly define the problem you aim to solve

2.2 Explain why it's important or relevant

3. Data Collection

3.1 Source of data (text, images, videos, audio)

3.2 Data format and structure

3.3 Any preprocessing steps applied

4. Exploratory Data Analysis

4.1 Statistical analysis of the data

4.2 Visualizations to understand patterns and relationships

If the mini project is text based

5. Text-Based Data Analysis

5.1 Natural Language Processing (NLP) techniques if dealing with textual data

5.2 Tokenization, stemming, or lemmatization

5.3 Sentiment analysis or topic modeling if applicable

If the mini project is image based

5. Image-Based Data Analysis

5.1 Image preprocessing (resizing, normalization)

5.2 Convolutional Neural Networks (CNNs) for feature extraction

5.3 Object detection or classification tasks

If the mini project is video based

5. Audio Based Data Analysis

5.1 Frame extraction and preprocessing

5.2 Temporal analysis or motion detection

5.3 Video classification or action recognition models

If the mini project is voice based

5. Audio Based Data Analysis

5.1 Audio feature extraction (MFCC, spectrograms)

5.2 Speech recognition or emotion analysis if applicable

5.3 Building audio classification models

6. Model Buildind

6.1 Choose appropriate machine learning or deep learning models

6.2 Train/test split and model training

6.3 Hyperparameter tuning if necessary

7.Evaluation

7.1 Metrics for assessing model performance

7.2 Comparison of different models if applicable

7.3 Discuss any challenges faced during model evaluation

8.Result and Discussion

8.1 Present the results of your analysis

8.2 Discuss the implications and insights gained

8.3 Address any limitations of the model or data

9.Conclusion

10. Summarize key findings

11. Discuss potential future work or improvements

12.References

13.Cite relevant papers, articles ,or libraries used in project

14.Code