## A Project report on

**“AI-Powered Techniques for Automated Article Summarization to Enhance Information Retrieval”**

Submitted in partial fulfillment of the requirement for the award of the degree of

## BACHELOR OF TECHNOLOGY

**in**

## ARTIFICIAL INTELLIGENCE & DATA SCIENCE

**By**

## P. Layavardhan 21701A3036

**A. Ahamad 21701A3001**

## K. Ram Charan 21701A3053

**P. Lokesh 21701A3038**

*Under the esteemed guidance of*

**Mr. Y. Venkata Subbaiah, M.Tech**

Assistant Professor

Department of AI&DS, AITS.



Submitted to

# Department of Artificial Intelligence & Data Science

## Annamacharya Institute of Technology and Sciences

## (An Autonomous Institution)

(Affiliated to J.N.T. University, Anantapur)

New Boyanapalli, Rajampet-516126 Annamaiah (Dt), A.P.

## 2024-2025

## (An Autonomous Institution)

(Affiliated to J.N.T. University, Anantapur)

New Boyanapalli, Rajampet-516126 Annamaiah (Dt), A.P.



**CERTIFICATE**

This is to certify that the project report entitled **“AI-Powered Techniques for Automated Article Summarization to Enhance Information Retrieval”** is submitted by

## P. Layavardhan 21701A3036

**A. Ahamad 21701A3001**

## K. Ram Charan 21701A3053

**P. Lokesh 21701A3038**

in partial fulfillment of the requirements for the award of Degree of **Bachelor of Technology** in “**Artificial Intelligence and Data Science**” for the academic year 2024-25.

### Signature of Guide: Signature of HOD:

Mr. Y. Venkata Subbaiah, M.Tech Dr. P. Phanindra Kumar Reddy, M.Tech, Ph.D.

Assistant Professor, Associate Professor,

Department of AI&DS, Head of the Department,

AITS, Artificial Intelligence & Data Science,

Rajampet. AITS,

Rajampet.

## (An Autonomous Institution)

(Affiliated to J.N.T. University, Anantapur)

New Boyanapalli, Rajampet-516126 Annamaiah (Dt), A.P.



**CERTIFICATE**

This is to certify that the project report entitled **“AI-Powered Techniques for Automated Article Summarization to Enhance Information Retrieval” is** submitted by

## P. Layavardhan 21701A3036

**A. Ahamad 21701A3001**

## K. Ram Charan 21701A3053

**P. Lokesh 21701A3038**

in partial fulfillment of the requirements for the award of Degree of **Bachelor of Technology** in “**Artificial Intelligence and Data Science**” is a record of bonafide work carried out by her during the academic year 2024-25.

Project viva-voce held on :

Internal Examiner External Examiner

## (An Autonomous Institution)

(Affiliated to J.N.T. University, Anantapur)

New Boyanapalli, Rajampet-516126 Annamaiah (Dt), A.P.



**ANTI-PLAGIARISM CERTIFICATE**

This is to certify that the project report entitled **“AI-Powered Techniques for Automated Article Summarization to Enhance Information Retrieval”** is submitted by

## P. Layavardhan 21701A3036

**A. Ahamad 21701A3001**

## K. Ram Charan 21701A3053

**P. Lokesh 21701A3038**

in partial fulfillment of the requirements for the award of Degree of **Bachelor of Technology** in “**Artificial Intelligence and Data Science**”. Course contains the plagiarism of which is within the acceptable limits.

### Date: Dean / Coordinator

**Research & Development Cell Date:**

**DECLARATION**

We hereby declare that the project report entitled **“AI-Powered Techniques for Automated Article Summarization to Enhance Information Retrieval”** under the guidance of **Mr. Y. Venkata Subbiah M.Tech Assistant Professor**, Department of Artificial Intelligence and Data Science is submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Artificial Intelligence and Data Science.

This is a record of bonafide work carried out by me and the results embodied in this project report have not been reproduced or copied from any source. The results embodied in this project report have been submitted to any other University or institute for the Award of any other Degree or Diploma.

## PROJECT ASSOCIATES

## P. Layavardhan

**A. Ahamad**

## K. Ram Charan

## P. Lokesh

***ACKNOWLEDGEMENT***

*We endeavor of a long period can be successful only with the advice of many well- wishers. We take this opportunity to express my deep gratitude and appreciation to all those who encouraged me for the successful completion of the project work.*

*Our heartfelt thanks to our Guide, Mr.* ***Y. Venkata Subbaiah*** *M.Tech Assistant Professor in Department of Artificial Intelligence and Data Science, Annamacharya Institute of Technology and Sciences, Rajampet, for his valuable guidance and suggestions in analyzing and testing throughout the period, till the end of the project work completion.*

*We wish to express sincere thanks and gratitude to* ***Dr. P. Phanindra Kumar Reddy****, Head of the Department of Artificial Intelligence and Data Science, for his encouragement and facilities that were offered to us for carrying out this project.*

*We take this opportunity to offer gratefulness to our Principal* ***Dr. S.M.V. Narayana****, for providing all sorts of help during the project work.*

*We are very much thankful to* ***Dr. C. Gangi Reddy****, Honorary Secretary of the Annamacharya Educational Trust, for his help in providing good facilities in our college.*

*We would express our sincere thanks to all faculty members of* ***Artificial Intelligence & Data Science Department, batch-mates, friends*** *and* ***lab-technicians****, who have helped us to complete the project work successfully.*

*Finally, we express our sincere thanks to* ***our parents*** *who has provided their heartfelt support and encouragement in the accomplishment to complete this project successfully.*

## PROJECT ASSOCIATES

P. Layavardhan

A. Ahamad

K. Ram Charan

P. Lokesh

***ABSTRACT***

*With the speed at which information is accumulating in this day and age, it is more important than ever to be able to quickly extract and understand vast amounts of textual content. The deluge of articles, research papers, and reports presents a formidable obstacle for individuals and organizations looking to quickly and precisely extract relevant information. Conventional techniques for summarizing texts, which frequently depend on human labor or simplistic computational methods, can be laborious and devoid of the subtlety necessary to effectively distill the core of intricate texts. Our project "AI-Powered Article Summarization Techniques" uses cutting-edge machine learning and natural language processing (NLP) technology to improve the summarizing process in order to address this difficulty. We create an intelligent summarizing system that produces succinct, coherent, and contextually relevant summaries of lengthy articles by utilizing state-of-the-art models like transformer-based topologies and pre-trained language models. This AI-powered solution is made to examine the original text's structure and content, highlighting important details and concepts while maintaining the original context and meaning. The resulting summaries enhance better informed decision-making, expedite information retrieval, and enable faster understanding.*

***KEYWORDS****: AI-Powered Summarization, Natural Language Processing (NLP), Text Summarization, Transformer Models, Pre-trained Language Models, Intelligent Summarizing System, Information Extraction and Textual Content Analysis*

# TABLE OF CONTENTS

**CHAPTER PAGE NO**

# ABSTRACT

1. INTRODUCTION 1-2
   1. [Objective 1](#_bookmark0)
   2. Defining Text Summarization 2
2. [LITERATURE SURVEY 3-5](#_bookmark1)
3. [**SYSTEM ANALYSIS** 6-8](#_bookmark2)
   1. Existing System 6
      1. [Disadvantages 6](#_bookmark3)
   2. [Proposed System 7](#_bookmark4)
      1. [Advantages 7](#_bookmark5)
   3. [Modules in Proposed System 8-9](#_bookmark6)
      1. [Data Ingestion 7](#_bookmark7)
      2. [Data Preprocessing 8](#_bookmark8)
      3. [Exploratory Data Analysis 8](#_bookmark9)
      4. [Feature Extraction: Correlations 8](#_bookmark10)
4. SYSTEM REQUIREMENTS SPECIFICATIONS 9-11
   1. [Software Requirements 10](#_bookmark11)
   2. Hardware Requirements & Feasibility study 10-11
5. [SYSTEM DESIGN 12-29](#_bookmark12)
   1. [Architecture design 13](#_bookmark13)
   2. Data flow diagram 14-18
   3. [Introduction to UML Diagrams 19-20](#_bookmark14)
   4. [UML Notations 20-21](#_bookmark15)
   5. [UML Diagrams 22-29](#_bookmark16)
      1. [Use case Diagram 22-23](#_bookmark17)
      2. [Class Diagram 23-24](#_bookmark18)
      3. [Sequence Diagram 24-26](#_bookmark19)
      4. [Collaboration Diagram 27](#_bookmark20)
      5. [Deployment Diagram 28](#_bookmark21)
      6. [Activity Diagram 29](#_bookmark22)
6. SYSTEM CODING & IMPLEMENTATION 30-63
   1. Domain Specification 30-39
      1. [Machine Learning 30](#_bookmark23)
      2. Machine Learning Vs. Technical Programming 30-31
      3. Inferring 31-32
      4. Machine Learning Algorithms 33
      5. Machine Learning Algorithm and where they are used? 33
      6. Application of Machine Learning 34-35
      7. Deep Learning 37-38
      8. [Reinforcement Learning 38](#_bookmark24)
      9. Application examples of Deep Learning Applications 39
   2. Difference between Machine Learning and Deep Learning 40-41
   3. Artificial Intelligence 42-43

6.3.1 Types of Artificial Intelligence 43-45

6.3.2 Application examples of Artificial Intelligence 46-50

* 1. Libraries 41-42
     1. Tensor Flow 42-44
     2. Keras 45-56
     3. Numpy 50-54
     4. Pandas 51-54
     5. Matplotlib & Seaborn 55-60
     6. Scikit-Learn 60-62
     7. Streamlit- Web Application 61-62
  2. Overview of Python Programming Language 45-46
  3. [History of Python 47](#_bookmark25)
  4. Python features 47-48
  5. Applications of Python 50-52
  6. Visual Studio Code Navigator 49-50

6.8. Code 50-61

1. SYSTEM TESTING 62-66
   1. Testing Methods 63-64
      1. Functional Testing 64-65
      2. [Integration Testing 65](#_bookmark26)
   2. Test Cases 65-66
2. **RESULTS 67-72**

## CONCLUSION AND FUTURE ENHANCEMENTS 73-74

**BIBLIOGRAPHY PLAGARISM 75-76**

**REPORT JOURNAL PUBLICATION**

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **Fig. No.** | **Figures** | **Page No.** |
| 5.1. | Architecture diagram | 13 |
| 5.2. | Data flow Diagram | 14-16 |
| 5.5.1. | Use Case Diagram | 23 |
| 5.5.2. | Class Diagram | 24 |
| 5.5.3. | Sequence Diagram | 25 |
| 5.5.4. | Collaboration Diagram | 27 |
| 5.5.5. | Deployment Diagram | 28 |
| 5.5.6. | Activity Diagram | 29 |
| 6.1.2.1 | Traditional Programming | 30 |
| 6.1.2.2 | Learning phase | 31 |
| 6.1.3.1 | Machine Learning Algorithms | 32 |
| 6.1.6.1 | Deep Learning Architecture | 32 |
| 6.1.6.2 | Neural Networks | 39 |
| 6.1.7.1 | Reinforcement Learning | 40 |
| 6.2.1 | Analysis of AI, Machine Learning & DL | 43 |
| 6.3.1 | Types of Artificial Intelligence | 47 |
| 8.1. | Dataset of Content | 69 |
| 8.2 | Dataset of Content | 69 |
| 8.3 | Full text token counts | 70 |
| 8.4 | Summary text token counts | 70 |
| 8.5. | Rouge Scores | 71 |
| 8.6. | Rouge Scores for Summarization | 71 |
| 8.7. | Home Page | 72 |
| 8.8 | Predictions Page | 72 |
| 8.9 | Text Summarization | 73 |
| 8.10 | Article Summarization | 73 |

# CHAPTER 1

**INTRODUCTION**

1. **INTRODUCTON**

## Objective

This work's main goal is to improve information retrieval by automating article summarizing using cutting-edge AI approaches. The project intends to improve the efficiency of information collection and understanding by producing high-quality, succinct summaries of text and documents using Google's T5 (Text-To-Text Transfer Transformer) architecture. Because of the T5 model's capacity to comprehend and provide natural language summaries, long content may be condensed into a more precise and contextually appropriate form, facilitating users' rapid acquisition of crucial information. The project's main goal is to improve summarizing skills while also utilizing streamlit to create an intuitive user interface. Users may submit documents and text into this interface, which effectively generates summaries. The goal is to offer a smooth and engaging user experience so that users may efficiently access and make use of the condensed material. Through the integration of cutting-edge natural language processing and user-friendly interface design, the project aims to provide a strong solution for improving information retrieval and assisting with well-informed decision-making.

## Defining The Text Summarization

Condensing a text or document into a shorter version while maintaining its main ideas and information is known as summarization. It seeks to make the information more readable and accessible for readers by reducing the amount of content without sacrificing its main points. When dealing with long papers, news stories, and scholarly studies, summaries may be very helpful since they offer a concise synopsis of the key ideas and topics. Extractive and abstractive summarization are the two main categories. In order to produce a summary, extractive summarizing entails choosing and removing important words, phrases, or passages from the original text. Although the original text's phrasing and organization are preserved, this approach occasionally produces summaries that are disjointed or illogical. However, abstractive summarization creates new words and phrases to capture the main ideas of the source material, which frequently leads to summaries that are more logical and seamless.

To improve the quality and accuracy of the summaries, modern summarizing approaches frequently make use of cutting-edge machine learning and natural language processing (NLP) models. Deep learning techniques are utilized by these models, including Google's T5 (Text-To-Text Transfer Transformer), to comprehend the subtleties, meaning, and context of the text. These algorithms are able to provide summaries that accurately convey the original content's meaning in a condensed manner since they have been trained on extensive datasets. A primary obstacle in the process of summarizing is striking a balance between the informative and succinct aspects. A summary should be simply readable and concise, yet comprehensive enough to contain all necessary details. This calls for complex algorithms that can recognize and rank the text's most significant passages while removing unnecessary or unimportant ones.

Evaluation metrics are essential for determining how well summarization models perform. The efficacy and correctness of summaries are frequently assessed using metrics like BLEU (Bilingual Evaluation Understudy) and ROUGE (Recall-Oriented Understudy for gisting Evaluation). While BLEU compares the text's quality with reference translations, ROUGE analyzes the overlap of n-grams between the produced summary and reference summaries. These metrics aid in measuring the effectiveness of a summarizing model in producing summaries that are both meaningful and coherent. It's critical to assess the usefulness and accessibility of summary technologies in addition to the quality of summaries. Summarization models may be made much more useful with user interfaces that make it simple to interact with them. One example of an interactive platform that makes it easy to integrate summarizing capabilities into multiple applications is streamlit, which lets users enter text or documents and obtain summaries. Summarization is used in many different fields, such as decision support systems, content aggregation, and information retrieval. By offering succinct previews, summarization in information retrieval assists users in rapidly determining the significance of search results. It facilitates the synthesis of data from several sources into a coherent summary in content aggregation, supporting thorough analysis and reporting.

# CHAPTER 2

**LITERATURE SURVEY**

**2. LITERATURE SURVEY**

A thorough review of the literature on text summarization demonstrates the abundance of research on the subject, including a wide range of application areas, datasets, and approaches.

**[2.1] Brown, Tom B., et al. "Language Models are Few-Shot Learners" (2020):** In this work, a cutting-edge language model called GPT-3 is introduced. It can handle a variety of NLP jobs with little task-specific training. By comprehending context and producing coherent text in a variety of settings, the model's design makes use of a vast number of factors to achieve impressive performance in text production, translation, and summarization.

**[2.2] Raffel, Colin, et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer" (2020):** The T5 model, which combines several NLP tasks into a single text-to-text framework, is presented in this study. T5 demonstrates the efficacy of transfer learning in NLP by achieving excellent performance in text summarization, translation, and question answering by training on a variety of text-based tasks and turning them into a text generation issue.

**[2.3] Liu, Yichao, et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach" (2019):** The T5 model, which combines several NLP tasks into a single text-to-text framework, is presented in this study. T5 demonstrates the efficacy of transfer learning in NLP by achieving excellent performance in text summarization, translation, and question answering by training on a variety of text-based tasks and turning them into a text generation issue.

**[2.4] Narayan, Shashi, et al. "Ranking Sentences for Extractive Summarization" (2018):** In order to investigate extractive summarizing techniques, this study ranks phrases according to how relevant they are to the core idea. The method enhances the quality and coherence of extractive summaries by combining neural network-based embeddings with a ranking algorithm to identify the most informative phrases.

**[2.5] See, Abigail, et al. "Get to The Point: Summarization with Pointer-Generator Networks" (2017):** The Pointer-Generator Network, which blends sequence-to-sequence learning with pointers networks that handle both extraction and generation, is the method for abstractive summarization that the study suggests. Because of this hybrid approach, the model is able to solve problems such as out-of-vocabulary terms and provide more precise and fluent summaries.

**[2.6] Paulus, Jason, et al. "A Deep Attentive Model for Abstractive Summarization" (2018):** In order to provide abstractive summaries, this work presents a deep attentive model that makes advantage of attention processes. The model offers advantages over conventional summary algorithms in terms of readability and informativeness by concentrating on collecting significant features and preserving coherence in produced summaries.

**[2.7] Zhang, Yao, et al. "Hippocampus: A Novel Framework for Summarizing Knowledge Graphs" (2021):** The study offers Hippocampus, a framework for condensing important information into brief summaries through the extraction and consolidation of knowledge graphs. By improving the readability and accessibility of intricate knowledge graphs, this method promotes improved information retrieval.

**[2.8] Xie, Liwei, et al. "Abstractive Summarization with Reinforcement Learning" (2019):** In order to maximize summary quality, this approach optimizes abstractive summarization using reinforcement learning. Summarization becomes more efficient and interesting when it is framed as a reinforcement learning issue. This helps the model learn to provide summaries that optimize rewards based on relevance and fluency.

**[2.9] Liu, Pengfei, et al. "Text Summarization with Pretrained Language Models" (2020):** The study looks at text summarization using pretrained language models, namely BERT and GPT-2. It shows that optimizing these models for summarizing tasks leads to notable performance gains, underscoring the advantages of using pretrained representations to produce high-caliber summaries.

**[2.10] Gormley, Ian, et al. "Summarization of Documents using Multi-Scale Attention Mechanisms" (2021):** In order to generate thorough summaries, this study presents a multi-scale attention system for document summarizing that gathers data at different granularities. By strengthening the model's capacity to concentrate on various levels of detail, the method raises the general caliber and applicability of the summaries that are produced.

# CHAPTER 3

**SYSTEM ANALYSIS**

**3. SYSTEM ANALYSIS**

**3.1 Existing System**

Extractive and abstractive summarizing are the two basic methods currently used by article summarization systems. Key words, phrases, or passages are directly chosen and extracted from the source text using extractive summarization systems. To determine which passages in the text are the most informative, these systems employ graph-based techniques, sentence ranking algorithms, and clustering. To create summaries, methods such as Text-Rank and Lex-Rank, for instance, create graphs of phrase similarities and rate them. Although extractive approaches are good at maintaining the original terminology, their dependence on direct textual extraction can occasionally result in summaries that are neither coherent or fluid. On the other hand, abstractive summarization algorithms produce summaries by comprehending and rewording the material instead of just extracting parts of it. These systems frequently use sophisticated neural network models, such as transformer-based models and sequence-to-sequence architectures, to generate summaries that seem more logical and realistic. Large-scale preparing and fine-tuning on summarizing tasks are used by models such as GPT-3 and T5 to produce summaries that capture the main ideas of the original text in a succinct manner. Although abstractive approaches are more readable and flexible, they might be more difficult to implement and need a lot of processing power for inference and training.

### Disadvantages:

* + - * Lack of Coherence
      * Over-reliance on Source Text
      * Generation Quality
      * Complexity and Resources
      * Difficulty in Handling Long Documents

## 

## Proposed System:

The suggested article summary method makes use of cutting-edge AI approaches to improve the effectiveness and precision of text and document summaries. This system combines many summarizing tasks into a single text generation issue by using the T5 (Text-To-Text Transfer Transformer) architecture, which enables it to produce coherent and contextually appropriate summaries. T5's design uses a sequence-to-sequence foundation to interpret and analyze the information, allowing it to handle a variety of text summarizing scenarios. The model is guaranteed to capture the subtleties and important details needed for efficient summarization because of its training on large datasets. By using this method, the system hopes to generate high-quality summaries that are more accessible and succinct while preserving the core of the original information. Apart from its ability to summarize, the suggested system has an easy-to-use interface created using streamlit. With this interface, users may effortlessly input text and data and obtain outputs that are summarized, making interaction easier. The user experience is improved by streamlit's simple design, which offers a multimedia interface for creating summaries without the need for technical know-how. T5's integration with the Stream lit UI guarantees that users may effectively utilize cutting-edge summarizing technology, making it a useful tool for enhancing decision-making and information retrieval. By combining complex AI models with user-friendly interfaces, this approach seeks to overcome the shortcomings of current summarization systems and provide a more reliable and user-focused solution.

### Advantages

* + - * Enhanced accuracy and reliability
      * Enhanced Coherence
      * Unified Framework
      * High Quality Summaries
      * User-Friendly Interface
      * Efficient Processing

## Modules in Proposed System

### Data Ingestion

Data ingestion is the process of moving data from numerous sources to an appropriate medium of storage so that the company may access, utilize, and analyze it. The destination is typically a database, content store, data mart, or warehouse. Spreadsheets, databases, internal software, SaaS data, and even content that has been grabbed from the internet can all be considered sources. Data intake is the key component of any analytics infrastructure. Data that is easily accessible and dependable is essential for analytics and next-generation reporting systems. Numerous models or architectures can be used as the basis for the design of a particular data ingestion layer. Data can be ingested in a variety of ways.

### Data Preprocessing

Data preprocessing is one data mining technique used to transform unprocessed data into an efficient and useful format. There are two stages this data goes through. The act of locating and fixing (or eliminating) erroneous or incomplete records from a dataset is known as data preprocessing, preparation, or cleaning. It involves determining whether parts of the data are wrong, incomplete, or irrelevant and then changing, replacing, or erasing the coarse or filthy data.

1. **Data cleaning**: It is imperative that the data be free of errors and superfluous information. As a result, the data gets scrubbed up before proceeding to the following stage. Finding and removing duplicate items, improper formatting, and missing values are all part of data cleaning. The practice of repairing or eliminating inaccurate, corrupted, improperly formatted, duplicate, or incomplete data from a dataset is known as data cleaning. Combining data from many sources increases the likelihood that some information may be duplicated or incorrectly tagged.

2. **Data Transformation:** Using mathematics, data transformation transforms datasets into representations that are helpful for data mining. This arranges the multitude of records rationally, allowing us to comprehend the material on a deeper level. Transformation includes attribute selection, normalization, and standardization. The process of transforming data from one format such as an Excel spreadsheet, XML document, or database file into another is known as data transformation. A raw data source is usually transformed into a format that has been cleaned, verified, and is suitable for use.

### Exploratory Data Analysis

The process of employing visual aids like bar diagrams and scatter plots to help consumers understand information more thoroughly is known as exploratory data analysis, or EDA. This enables us to identify data trends more precisely and modify our study as necessary. The yearly trends graph is dependent upon and monitors the yearly areas under cultivation and harvest. Despite global output, China's imports rose over the preceding 12 years, while its exports saw a dramatic fall in 2009. This shows that, despite having a significant and over time need for soybeans, China is becoming more reliant on global supplies to meet its needs.

### Feature Extraction: Correlations

Now let's look at the relationships that exist between different numerical characteristics. A correlation coefficient, which goes from -1 to 1, shows how much the values of two distinct attributes move together. When two variables, like a child's age and height, develop together, there is a positive association. When there is a negative correlation between two variables, like study hours and party attendance, it means that when one grows, the other lowers. Correlations near -1 or 1 show strong associations. A weaker association is indicated by values closer to zero. Zero indicates that there is no relationship.

Making an outline for the information that will be kept in a database is known as data modeling. A machine learning system is trained to predict categories from features as part of the modeling process, adjusted for business needs, and validated with holdout data. A trained model, which may be used for inference or to make predictions about fresh data points, is the result of modeling. Because modeling utilizes uniform inputs and is independent of other stages of the procedure for machine learning, we can make changes to the prediction issue without completely reworking our code. We could create more label timings and matching characteristics and add them to the framework if the needs of the business alter. Following their use, models are assessed for the root average square error is used to evaluate the correctness of models once they are implemented. Regressors used in the prediction regression approach: Support Vector Regression (SVR) uses random forests as Regressor Kernel functions. The linear regression technique Choice Tree One type of regression is regression. Since there are several categories involved in this activity, we use the following metrics:  R2 rating: The regression's r2-score is the proportion of the tested tuples that the regressor appropriately classifies. Root Mean Square Error: The accuracy of each model is measured and the corresponding root median square error is evaluated.

# CHAPTER 4

**SYSTEM REQUIREMENTS SPECIFICATION**

# 4. SYSTEM REQUIREMENTS SPECIFICATION

A thorough explanation of the tasks that a system must do is provided by software requirements specifications (SRS), often referred to as software system demands specifications. This section's use cases explain how the program communicates with users. Beyond the usage case, the SRS also includes non-functional requirements. Criteria that restrict design or execution are known as non-functional specifications (e.g., performance engineering requirements, quality standards, or design restrictions).

## Software Requirements

* + - Operating System : Windows 11
    - Server-side Script : Python
    - IDE : Visual Studio Code
    - Libraries used : Numpy, Pandas, Keras
    - Visualization : Matplotlib
    - Dataset : Articles Text Dataset

## Hardware Requirements

* + - Processor : I5/Intel Processor
    - RAM : 4GB (min)
    - Hard Disk : 160GB
    - Key Board : Standard Windows Keyboard

## Feasibility Study:

### Technical Feasibility

**Data Accessibility:** The ease with which people may receive and use data from several sources is referred to as data accessibility. This entails obtaining historical records, environmental factors, and real-time text summarization from databases, sensors, and open sources in order to summarize text. Efficient data management systems, appropriate integration of heterogeneous data sources.

**Computational Resources:** Machine learning model training for the text summarization can be computationally demanding, necessitating the use of potent hardware resources like GPUs or TPUs. Nevertheless, scalable infrastructure for model deployment and training is provided by cloud computing platforms like Amazon Web Services and Google Cloud Platform.

**Model Complexity:** Optimization and design of the model architecture. Nonetheless, pre-trained model libraries like Py-Torch and TensorFlow Affordability provide frameworks for developing and putting into practice complicated models readily available.

**Financial Feasibility:**

**Cost of Computing Resources:** Systems for cloud computing are flexible and scalable, but they have costs associated with compute, storage, and data transfer. The project's feasibility is dependent on available funds and the ability to effectively manage and optimize cloud expenses.

**Open-source Tools and Libraries:** Using open-source machine learning frameworks and libraries can significantly reduce development costs. A number of popular libraries, such scikit-learn and TensorFlow, are supported by large developer communities and are open source.

### Operation Feasibility:

**Integration with Current Systems**: It is crucial to consider if text summarization technologies can be integrated with the platforms and practices that are in place now. APIs and SDKs enable the seamless integration of social networking sites, news aggregation websites, and content management systems.

**User adoption and adoption:** Users' adoption of text summarization systems may vary depending on characteristics including ease of use, accuracy, and transparency. Usability and acceptability may be increased by using user feedback techniques and user-centric design concepts. Legal and Ethical Viability.

### Legal And Ethical Feasibility:

**Data Privacy and Compliance:** Respecting data privacy rules, such the CCPA and GDPR, is essential when analyzing user data to generate text summarization. Operating in an ethical and compliant manner requires maintaining the necessary data anonymization and consent processes.

**Implications for Ethics**: Fairness, prejudice, and accountability are just a few of the ethical issues that should be taken into account while developing and implementing text summarization systems. Ensuring openness in decision-making and mitigating algorithmic biases are crucial for upholding ethical integrity and user confidence.

### Market Feasibility:

**Demand for Text Summarization Solutions:** There is a need for reliable text summarization tools. Surveys and market research may be used to determine the size, level of competition, and prospects for commercialization of the market.

**Use Cases and the Target Audience:** There are several application cases in a variety of disciplines for the suggested automatic article summarizing system that makes use of the T5 model and streamlit UI. Students as well as researchers can use the method in the academic setting to swiftly summarize long articles, research papers, and academic publications. This feature speeds up literature studies and aids in the synthesis of data from many sources, greatly cutting down on the amount of time needed to extract important conclusions and insights. To help with more effective knowledge acquisition and distribution, researchers may enter lengthy scientific publications and obtain succinct summaries that emphasize the most important features. Professionals in business industries who must keep up with market research, industry studies, and company documents may find the summary tool to be quite helpful. The technique may be used by executives, economists, and decision-makers to quickly summarize extensive studies, saving them the time and effort of reading through lengthy documentation in order to make well-informed judgments. This use case is very helpful for increasing output and simplifying information processing in hectic work settings when prompt insights are essential.

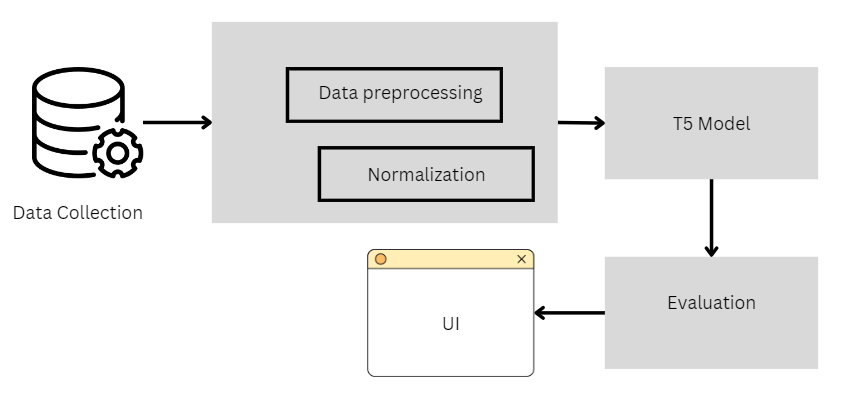
The approach is also helpful for media workers and content producers that deal with a lot of news stories, blogs, and other types of media content on a daily basis. It is simpler for content creators to choose and deliver information to their viewers when they use the summarizing tool to quickly and effectively create summaries of news pieces, feature articles, or press releases. This use case encourages the production of more approachable and interesting content, assisting media professionals in more efficiently providing readers with pertinent news and information. All things considered, the system meets the requirements of a wide range of users by offering a reliable and approachable method for organizing and analyzing large amounts of text.

# CHAPTER 5

**SYSTEM DESIGN**

# 5. SYSTEM DESIGN

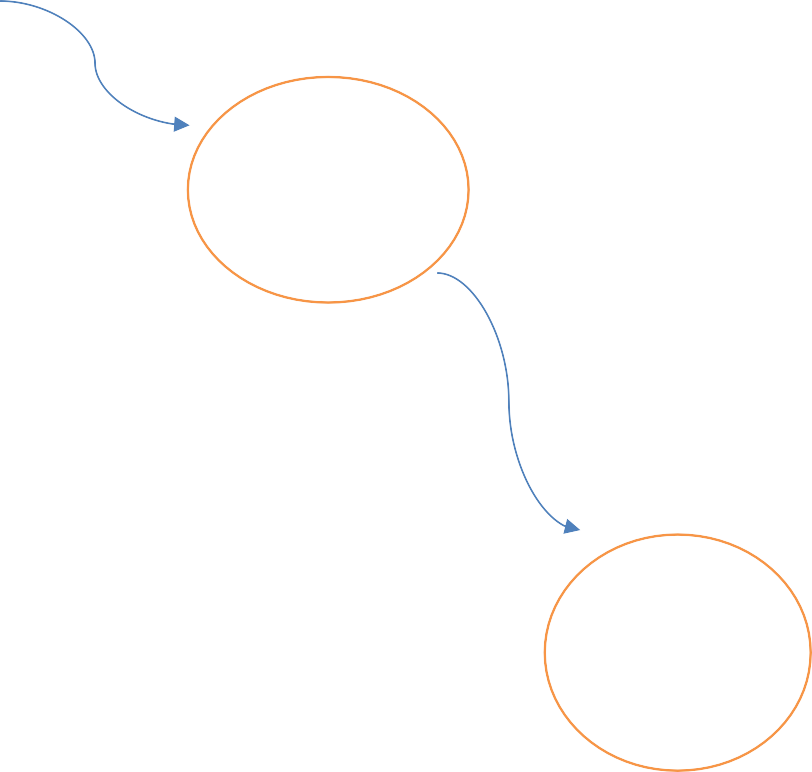
## 5.1. ARCHITECTURE DESIGN

****

### Figure 5.1: Architecture diagram

In order to provide an effective summarizing solution, the suggested automated article summarization system's architecture design combines the T5 (Text-To-Text Transfer Transformer) paradigm with an intuitive streamlit interface. The T5 model, which uses a transformer-based design to handle incoming text or documents, is at the center of the architecture. By transforming what is input into a text-to-text format, the model enables it to comprehend the context and important details included in the material and provide succinct and logical summaries. To verify that the incoming data is compatible with the model, the system preprocesses it, normalizing the text and tokenizing it. Following processing, the text is sent into the T5 model, which generates the output summary.

### 5.2 Data Flow Diagram Level - 0



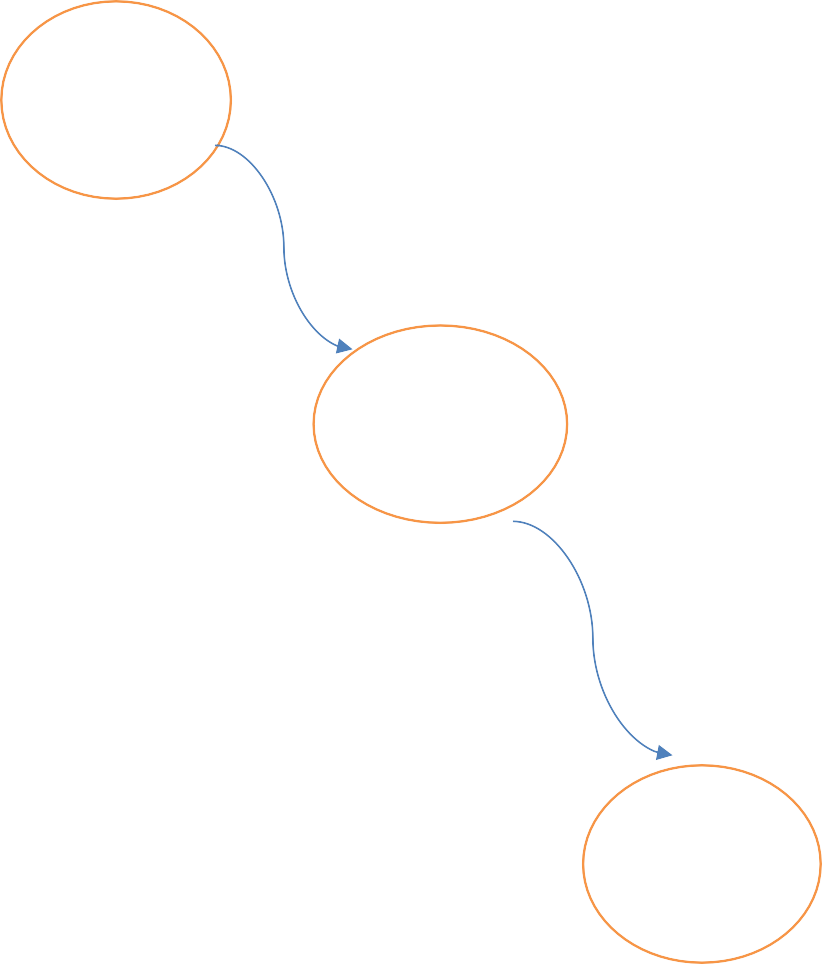
Dataset Collection

Pre- processing

T5 model

Trained & Testing dataset

**Level -1**



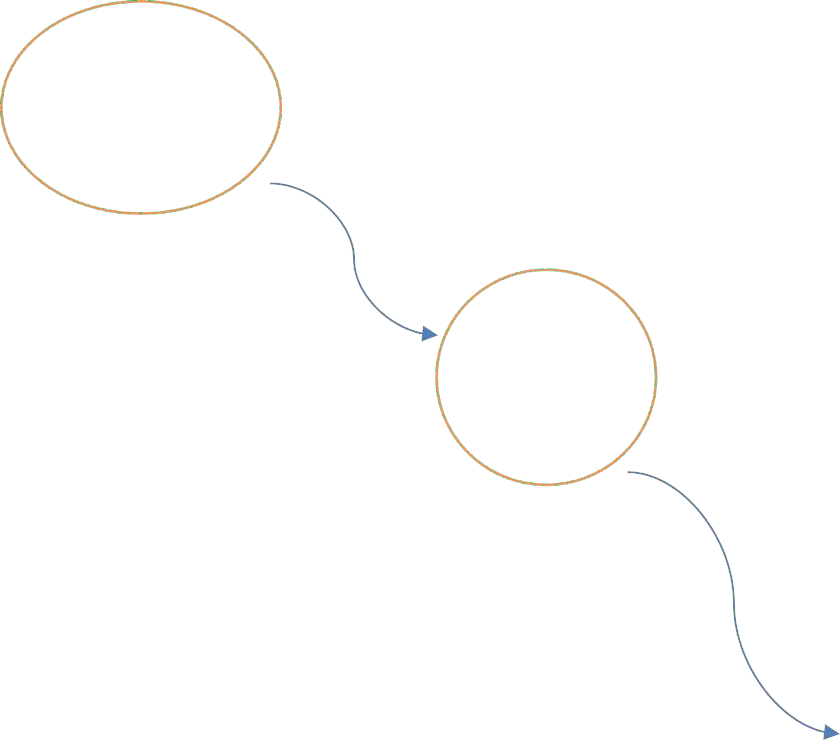
Dataset collection

Pre- processing

Feature Extraction

Apply Algorithm

### Level - 2



Classify the dataset

Accuracy of Result

User Interface

Finalize the accuracy of text summary

A data flow diagram (DFD) is a graphical depiction that shows how data moves through a system, emphasizing the interconnections, data storage, and procedures. It offers an easy-to-understand method of visualizing the flow of data through several processing steps, from input to output. Data flows, which depict the transfer of data between elements, data stores, which contain data, processes, which alter data, and external entities, which interact with the system but are beyond its scope, are the usual components of DFDs. DFDs aid in comprehending the operation of the system, spotting any inefficiencies, and enhancing the overall design by outlining these components.

DFDs are primarily used to give a high-level overview of a system's data flow, which facilitates analysis and communication of the data processing process. They provide insights into the interactions between various components and the management of data, making them useful tools for both system design and analysis. Developers, analysts, and project managers are among the stakeholders that benefit from DFDs since they reduce complicated systems into more digestible visual representations. Better decision-making, improved team communication, and the ability to see where adjustments or improvements are needed are all made possible by this clarity. DFDs are also a helpful documentation tool, storing design specifications and system needs for later usage and upkeep.

The robotic article summarization system's Data Flow Diagram (DFD) shows how data moves through the system's many parts and how input is converted into output. The front end, where users communicate with the system by uploading documents or entering text, is shown at the top of the diagram. The main point of input for data into a system is represented by this first contact. These inputs are captured and processed by the streamlit-developed user interface, which makes sure the data is structured appropriately for subsequent processing. The preprocessing module receives the data flow from this component and is in charge of getting the input data ready for summary. The incoming text or document goes through a number of changes during the preprocessing stage to make sure it is compatible with the T5 model. Tokenization, which divides the text into digestible sections or tokens, and the normalization process, which transforms the material into a standardized format, are both included in this phase. Any necessary text maintenance, such as eliminating special characters or fixing formatting errors, is also taken care of by the preprocessing module. The T5 model-powered core summarization engine receives the cleaned and designated data after that.

The primary task of creating summaries is carried out by the T5 model, which is at the center of the summary process. Using its transformer-based design, it interprets the preprocessed text and extracts important information and context. The model draws on its substantial training to produce a logical and succinct synopsis that encapsulates the main ideas of the source material. In order to get high-quality output, this summarization procedure requires intricate calculations and data manipulations, making use of the model's deep learning capabilities. The output data returns to the user interface once the T5 model has produced the summary. This part is in charge of giving the user a legible and accessible format for the results. Users may examine and make use of the condensed versions of their original writings by using the Streamlit interface, which shows the created summaries. Users may also request fresh summaries or edit their input through the user interface, resulting in an ongoing cycle of feedback and interactivity.

The DFD also has feedback features that allow users to comment on how well the summaries are written. The system gathers this input, which may be utilized to gradually enhance the model's performance or optimize the summarization procedure. It is possible to examine the feedback data to find areas that need work, which might result in iterative improvements to the user interface elements or summarization methods. All things considered, the DFD records every stage of the data lifecycle in the summarizing system, from the first user input to the feedback and final output. It demonstrates how preprocessing, model processing, and user interaction can all work together seamlessly to ensure that the system effectively converts unprocessed text into insightful summaries while preserving user involvement and pleasure.

## Introduction To UML Diagrams

Software is becoming more and more strategic, thus the industry is looking for ways to streamline text summarization, improve quality, reduce costs, and speed up time to market. Among these methods are modular technology, visual programming, frameworks, and patterns. As a business expands, it looks for methods to manage the size and reach of its systems. minimize the intricacy of them. They are aware of the problems related to load balancing, fault tolerance, collaboration, replication, and physical distribution. While the Internet has simplified many jobs, it has also made other structural problems worse. To meet these needs, the Unified Modeling Language (UML) was developed. Systems design, to put it simply, is the process of building a system's architecture, parts, elements, interfaces, and data in order to achieve certain objectives. Throughout the project, eight fundamental UML diagrams were explained.

* + - Use Case Diagram
    - Class Diagram
    - Activity Diagram
    - Sequence Diagram
    - Collaboration Diagram
    - State chart Diagram
    - Component Diagram
    - Deployment Diagram

### GOALS

1. Make available to users a ready-to-use, expressive visual modeling language that enables them to create and share meaningful models.
2. Provide mechanisms for extendibility and specialization in order to broaden the scope of the core concepts.
3. Refrain from using specific programming languages or development processes.
4. Lay the groundwork for a formal understanding of the modeling language.
5. The following are the primary goals of the UML design:
6. Encourage the growth of the market for OO tools.
7. Help with the implementation of higher-level development concepts like collaborations, frameworks, patterns, and components.
8. Implement best practices**.**

## UML Notations

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **SYMBOL**  **NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Initial Activity |  | This diagram depicts the flows initial point or  activity. |
| 2. | Final Activity |  | A bull’s eye icon marks the  conclusion of the  activity graphic. |
| 3. | Activity |  | Represented by a  rectangle with a rounded edge. |
| 4. | Decision |  | One that requires  decision-making. |
| 5. | Use Case |  | Explain how a user and a system communicate. |
| 6. | Actor |  | A function a user has in relation to the system. |

|  |  |  |  |
| --- | --- | --- | --- |
| 7. | Object |  | A Real -Time entity. |
| 8. | Message |  | To communicate  between the lives of object. |
| 9. | State |  | It depicts events t  occur during an objects lifetime. |
| 10. | Initial State |  | Represents the  objects initial state. |
| 11. | Final State |  | Represents the objects final state. |
| 12. | Transition |  | Label the transition with the event that triggered it and the action that result  from it . |
| 13. | Class |  | A group of items with similar structures and  behaviours. |
| 14. | Association |  | Relationship between classes. |
| 15. | Generalization |  | Relationship between more general class and a  more specific class. |

**Fig. UML Notations**

## UML Diagrams

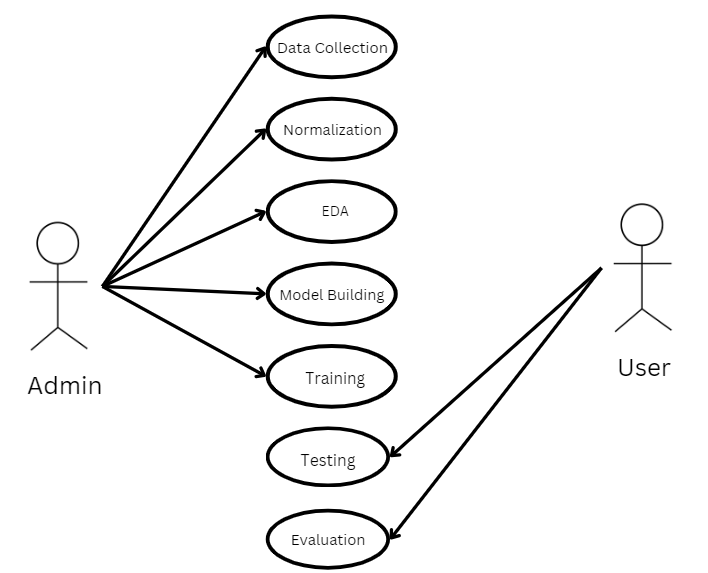
### Use Case Diagram

One type of behavioral diagram that results from use-case research is the use case diagram, which is an illustration of how software engineering uses the Unified Modeling Language (UML). Its objective is to illustrate the players in a system, together with the goals (expressed as use cases) and any dependencies between them. A use case diagram's primary objective is to display which system operations are carried out for each actor. The roles that the actors in the system play are evident. Use cases are employed throughout the need elicitation along with evaluation phase to demonstrate the system's capabilities. Use scenarios are employed to explain how the technology functions while not in use. Actors are external to the system, whereas use cases are located within it. In the case diagrams, which is an actor diagram, a device border divides a set of use scenarios. The program Understanding the behavior of the element requires a diagram.

* + - 1. Sequences highlight the relationship to outside circumstances.
      2. This covers both the performer's job and the system.
      3. Actors can portray people or a building.

Software system artifacts may be specified, visualized, built, and documented using the standard language UML.

* The Object Management Group (OMG) developed UML, and in January 1997, the OMG received a draft of the UML 1.0 standard.
* OMG is working tirelessly to create a true industry standard.
* Unified Modeling Language is what UML stands for the visual language UML is used to create software blueprints. A text summarization system's use case diagram shows how various players interact with the features of the system. The primary actors in this condensed example are "User," "Admin," and "System." Users may browse the analysis findings, contribute news stories for verification and analysis, and comment on how accurate the classifications are by interacting with the system. In the interim, administrators' duties also include user account management, system performance monitoring, and system updates with new datasets or algorithms. In order to maintain security and privacy, the "User Authentication" use case makes sure that administrators and users authenticate before using the system. All things considered, the use case diagram offers a concise synopsis of the features and interactions among different players inside the system.

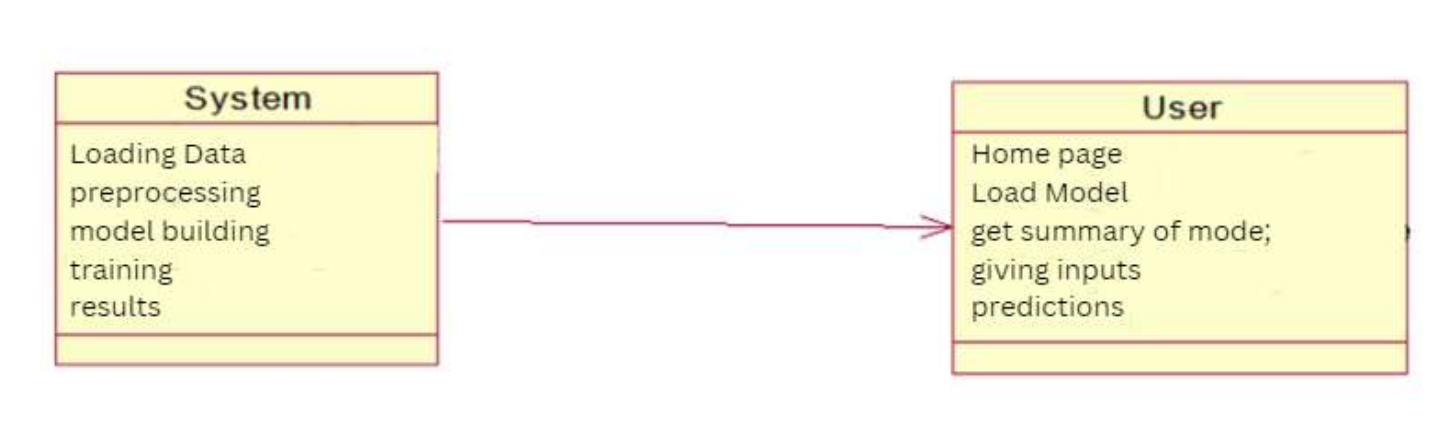


### Figure 5.5.1: Use Case Diagram

### Class Diagram

Software engineers use class diagrams, a sort of static structural diagram in the Unified Modeling Language (UML), to display the classes, attributes, and connections among the classes that comprise a system.

It is employed in analysis to display the details of the system. Architecture looks at the class structure to see if any classes need to be separated or if there are too many functions in them. There are links established between the classes. Developers utilize the Class Diagram as one technique for creating classes. A class diagram is a collection of linked objects with the same semantics—a set of shared properties, operations, relationships, and connections that are connected to each other. A class in a production is a large collection of objects.

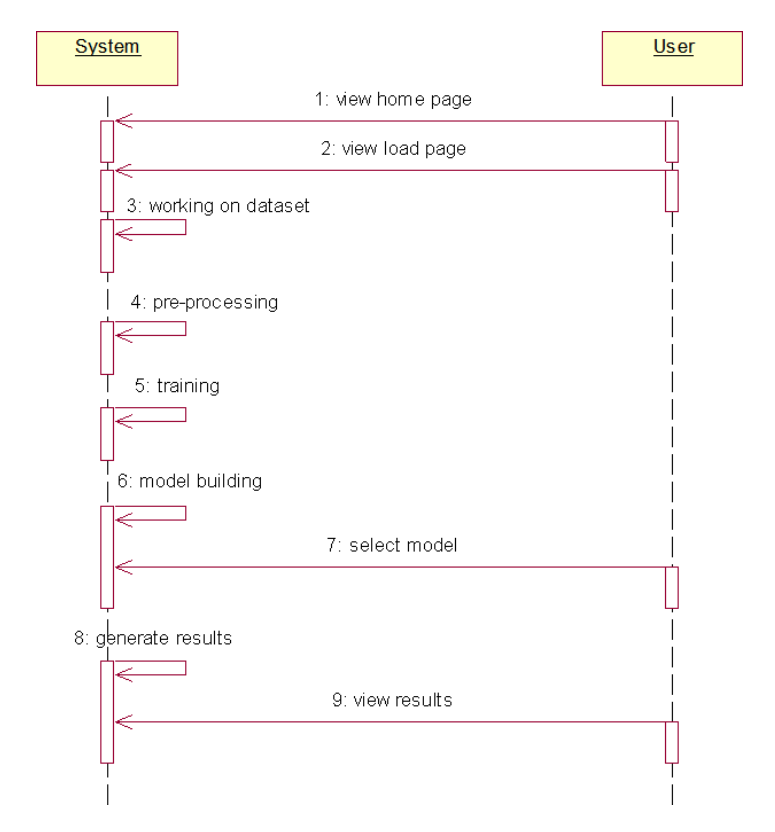


### Figure 5.5.2: Class Diagram

A class diagram is a kind of static structural diagram in the Unified Modeling language that shows the relationships, functions, and interactions between objects to represent the structure of a system. The class diagram is the fundamental component of object-oriented modeling. The classes described in include image, create dataset, pre-processing, segmentation, and classification along with the associated attributes, procedures, and inter class connections.

### Sequence Diagram

A sequence diagram is a form of interaction diagram used in the Unified Modelling Language (UML) that shows the relationships and order of actions. It is referred to as a message sequence chart. Timing diagrams, occurrence-trace diagrams, and depictions of event contexts are examples of sequence diagrams. A sequence diagram may also be referred to as an event scenario or an event diagram. Sequence diagrams display the interactions between the various parts of a system. Software engineers and entrepreneurs often use these diagrams to understand and express specifications for both new and existing systems. An interaction diagram where the time of message transmission is highlighted. The objects involved in a relationship are represented in different ways depending on how long they live and the signals they send or organize over time.



### Figure 5.5.3. Sequence Diagram

An interaction diagram called a sequence diagram shows the sequential communications that are delivered and received between items or system components.

These diagrams are crucial for illustrating the dynamic behavior of a system, helping to clarify the flow of control and data over time. Sequence diagrams are versatile and can be used to model various scenarios such as use case scenarios, system architecture, collaboration between objects, and concurrency or parallelism in execution paths. They serve as valuable tools for system designers, developers, and stakeholders to understand and communicate the dynamic interactions within a system, aiding in the analysis, design, and implementation phases of software development. Sequence diagrams offer a detailed view of how objects interact in a particular scenario, making them invaluable during system design and development. Each object's lifeline on the diagram illustrates its existence over time, while the arrows between lifelines represent messages passed between objects. These messages can include method calls, responses, or signals, providing a clear depiction of the order of interactions.

Furthermore, sequence diagrams can highlight important aspects of system behavior, such as exception handling, loops, and conditional branching. By incorporating these elements, sequence diagrams can accurately capture the complexities of real-world system interactions, aiding developers in understanding the system's behavior under various conditions. Moreover, sequence diagrams are not only beneficial for developers but also for stakeholders and end-users. They provide a visual representation of system behavior that is easily understandable, facilitating communication and collaboration between different project stakeholders. Additionally, sequence diagrams can serve as documentation for future reference, helping in system maintenance, debugging, and enhancement efforts. In summary, sequence diagrams are powerful tools for modeling and understanding the dynamic behavior of systems. They provide a detailed, chronological view of object interactions, making them indispensable during system design, development, and maintenance. Their ability to capture complex system behaviors in a clear and concise manner makes them an essential asset for software engineers and stakeholders alike.

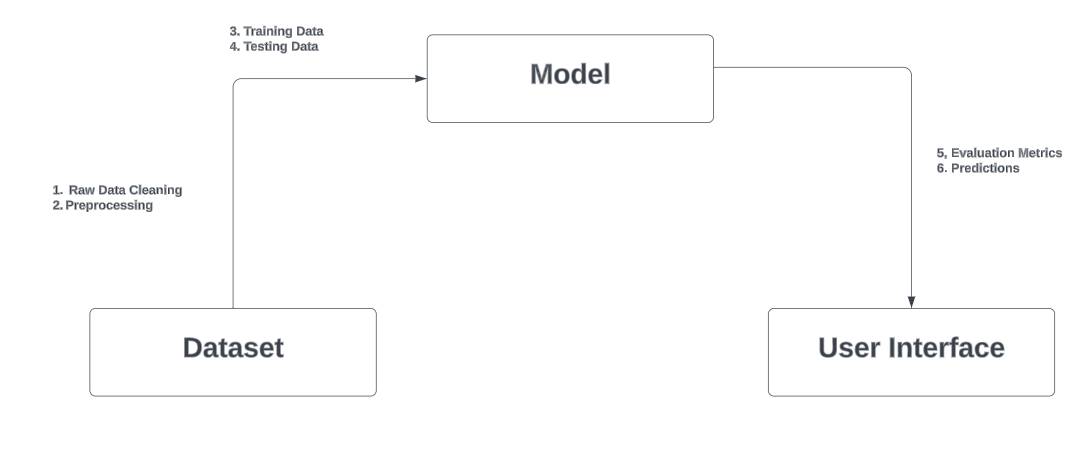
One can also refer to a sequence diagram as an event diagram or an event scenario. Sequence diagrams show how a system's components interact with one another. The requirements for both new and current systems are frequently described and understood by entrepreneurs and software engineers using these diagrams. Each object's lifeline on the diagram illustrates its existence over time, while the arrows between lifelines represent messages passed between objects. These messages can include method calls, responses, or signals, of the order of interactions.

These diagrams are crucial for illustrating the dynamic behavior of a system, helping to clarify the flow of control and data over time. Sequence diagrams are versatile and can be used to model various scenarios such as use case scenarios, system architecture, collaboration between objects, and concurrency or parallelism in execution paths. They serve as valuable tools for system designers, developers, and stakeholders to understand and communicate the dynamic interactions within a system, aiding in the analysis, design, and implementation phases of software development. Sequence diagrams offer a detailed view of how objects interact in a particular scenario, making them invaluable during system design and development. Each object's lifeline on the diagram illustrates its existence over time, while the arrows between lifelines represent messages passed between objects. These messages can include method calls, responses, or signals, providing a clear depiction of the order of interactions. Each object's lifeline on the diagram illustrates its existence over time, while the arrows between lifelines represent messages passed between objects. These messages can include method calls, responses, or signals, of the order of interactions.

Furthermore, sequence diagrams can highlight important aspects of system behavior, such as exception handling, loops, and conditional branching. By incorporating these elements, sequence diagrams can accurately capture the complexities of real-world system interactions, aiding developers in understanding the system's behavior under various conditions. Moreover, sequence diagrams are not only beneficial for developers but also for stakeholders and end-users. They provide a visual representation of system behavior that is easily understandable, facilitating communication and collaboration between different project stakeholders. Additionally, sequence diagrams can serve as documentation for future reference, helping in system maintenance, debugging, and enhancement efforts. One can also refer to a sequence diagram as an event diagram or an event scenario. Sequence diagrams show how a system's components interact with one another. The requirements for both new and current systems are frequently described and understood by entrepreneurs and software engineers using these diagrams.

### Collaboration Diagram

In a collaboration diagram, the technique's call sequence is denoted by a numbering scheme, as in the example below. The sequence in which the techniques are invoked is indicated by the number. The order management system is also used to define the collaboration diagram. The method calls have a sequence diagram-like appearance. The collaboration diagram describes the object organization, but the sequence graph does not. This is the distinction. A collaboration diagram illustrates the interactions and communication between different actors or parts of a system. This kind of graphic would illustrate how various parts collaborate to process and assess news stories in a text summarization system. To illustrate, a rudimentary version would comprise the "News Article" object and actors like "Admin," "User," and "System." The process of analysis begins when the user uploads a news story to the system. This is how the diagram's visual representation of the communication flow looks. The system parses the text, extracts features, and classifies the material during analysis using internal components or algorithms. When the analysis is complete, the system sends the user the analysis findings. Additionally, the user can provide input to the system.



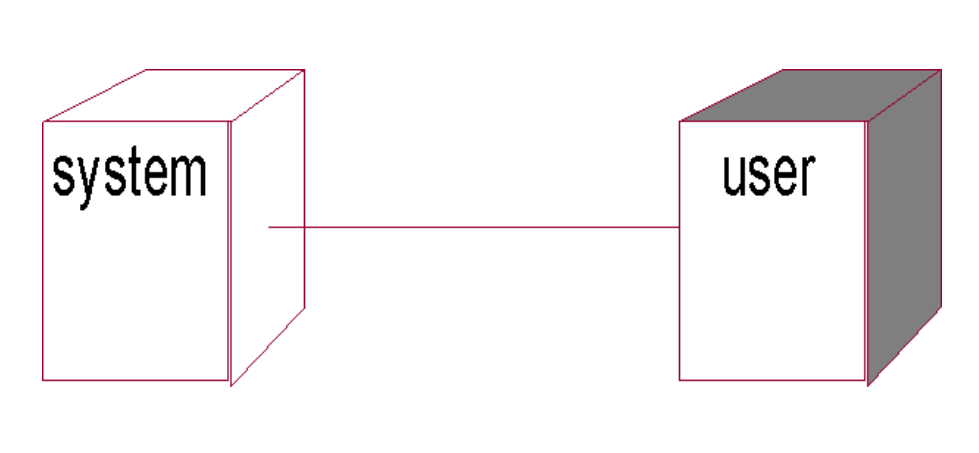
### Figure 5.5.4. Collaboration Diagram

### Deployment Diagram

Deployment diagrams are used to illustrate both the software and the hardware components of a deployment. Component diagrams and deployment diagrams are quite similar. Deployment diagrams, which are used to explain components, provide representations of the component' deployment in hardware.

The main focus of UML is on a system's software artifacts. But the purpose of these two specific graphics is to draw attention to the software and hardware elements. While most UML charts are used to handle logical components, deployment diagrams are intended to focus on the physical structure of a system. Diagrams are used by the system engineers for deployment. Deployment diagrams serve the following purposes, to summarize:

* Think about how a system's hardware is organized.
* Explain the hardware elements that are deployed in order to run software components.
* Tell us about the runtime processing nodes.



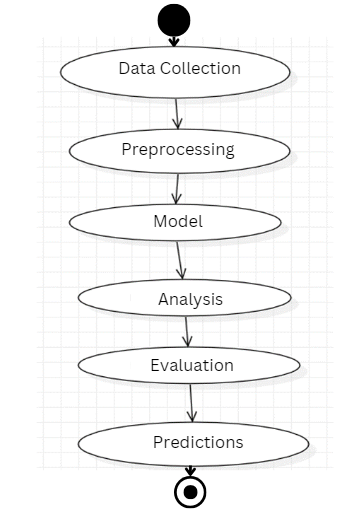
### Figure 5.5.5: Deployment Diagram

### Activity Diagram

Activity diagrams show the work flows of changing tasks and activities with choice, iteration, and concurrency. In the Unified Modified Language, activity flowcharts may be used to depict in detail the operational and commercial procedures of system components.

An activity diagram shows the control flow in its entirety. An activity diagram is comparable to a flowchart with distinct states. The activity diagram allows you to monitor the order in which your system's operations are taking place. States resemble activities, although activities are somewhat more nuanced. They are state because they occur and then proceed without interruption to the next state. The "diamond" conditional branch is stateless and chooses which activity to transition to based on a feature. Activity Diagram consists of

* Action states.
* Transition.
* Objects.
* Contains Fork, Join and branching relations along with flow Chart symbols.



**Figure 5.5.6: Activity Diagram**

# CHAPTER 6

**SYSTEM CODING & IMPLEMENTATION**

# 6.SYSTEM CODING AND IMPLEMENTATION

## Domain Specification

### Machine Learning

A machine learning system is one that can pick up knowledge from examples by improving itself and doesn't require explicit programming. The concept that a machine can independently learn from the data (i.e., example) and provide correct results is the breakthrough. To forecast an outcome, machine learning uses statistical methods and data. Corporate uses this output to get practical insights. Bayesian predictive modeling and data mining are strongly connected to machine learning. The machine takes in data and uses an algorithm to provide responses. Making recommendations is a common machine-learning problem. All movie or series recommendations for Netflix users are predicated on their past usage history. Unsupervised learning is being used by IT businesses to enhance. A branch of artificial intelligence (AI) known as "machine learning" is concerned with creating statistical models and algorithms that let computers analyze, learn from, and make decisions based on data. In contrast to traditional programming, which includes explicitly coding human rules and logic, machine learning entails building systems that are able to recognize patterns, anticipate outcomes, and get better with time. These systems use a variety of methods, such as supervised learning in which a model is trained on labeled data to produce predictions or unsupervised learning in which a model finds patterns and correlations in unlabeled data to learn from past data.

Fundamental to machine learning is the notion that systems are capable of self-learning and self-adaptation based on their interactions with data. Large amounts of data are fed into a machine learning model as part of this learning process, and the model employs statistical and mathematical techniques to identify patterns and correlations in the data. In supervised learning, for example, a model is trained with input-output pairs from the dataset, enabling it to understand the link between the inputs and the related outputs. On the other hand, unsupervised learning methods without predetermined labels or categories, including clustering and dimensionality reduction, assist in revealing underlying patterns and relationships in data. Machine learning has a wide range of transformational applications that affect many different disciplines and sectors.

### Machine Learning vs. Traditional Programming

Machine learning is very different from traditional programming. In conventional programming, a programmer works with an industry specialist to determine the regulations and develop software for that industry. Every rule has a logical basis, and the machine will carry out the output in accordance with the logical assertion. More rules must be developed as the system gets more complicated. It can easily become too costly to maintain.

### DATA OUTPUT

**RULES**

### 6.1.2.1 Traditional Programming

COMPUTER

**How is machine learning implemented?**

The brain that does all the learning is machine learning. Machine learning is comparable to human learning. unstructured formats like text or images. Following data collection, the next step is data preprocessing, where you clean and organize the data. This typically involves handling missing values, removing outliers, scaling features, and encoding categorical variables. Subsequently, you engage in feature engineering, selecting, extracting, or transforming features to make the data suitable for training ML models. With the preprocessed data ready, you move on to selecting the appropriate ML algorithm(s) based on factors such as the problem type, data size, and desired outcomes. Popular algorithms include linear regression, decision trees, support vector machines, and neural networks. Once chosen, you train the selected model(s) using the prepared data, during which the model learns patterns and relationships within the data. After training, you evaluate the model's performance on a separate dataset to gauge its ability to generalize to unseen data. Evaluation metrics vary based on the problem type, including accuracy, precision, recall, and F1-score for classification, or RMSE for regression tasks. Hyperparameter tuning comes next, where you adjust the model's hyperparameters to optimize its performance. Hyperparameters, like learning rate and number of hidden layers, influence the learning process but are not directly learned during training.

Once satisfied with the model's performance, you deploy it to make predictions on new data. Deployment can take various forms, such as integrating the model into applications, creating APIs, or deploying it on cloud platforms. Monitoring and maintenance are crucial post-deployment, involving continuous performance monitoring and periodic retraining with new data to ensure the model remains accurate and relevant over time. Just like a human, the computer struggles to understand if its feed contains an example that hasn't been seen before. Learning and inference are machine learning's main goals. The computer gains knowledge by identifying patterns. The data is what led to this finding. Selecting the right data to feed the computer is an essential skill for a data scientist. A feature vector is a set of properties used to solve an issue. A feature vector can be seen as a subset of data applied to an issue. The machine turns this discovery into a model by applying sophisticated algorithms to simplify reality. As a result, the data are described and condensed into a model during the learning step. A feature vector is a set of properties used to solve an issue. A feature vector can be seen as a subset of data applied to an issue. The machine turns this discovery into a model by applying sophisticated algorithms to simplify reality. As a result, the data are described and condensed into a model during the learning step.

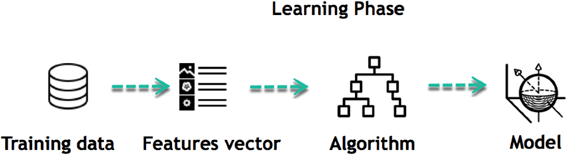


Fig. 6.1.2.2 Learning Phase

For example, the computer is attempting to comprehend the connection between an individual's income and the probability of dining at a fine dining establishment. It appears that the algorithm discovers a favorable correlation between income and dining at upscale restaurants.

### Inferring

Once the model is constructed, its effectiveness may be evaluated using previously unseen data. The updated data are converted into a vector of features, run through the model, and provide a forecast. All of this is what makes machine learning so wonderful. Retraining the model or updating the rules are not necessary. The model that has already been trained can be used to draw conclusions from fresh data.

|  |  |  |
| --- | --- | --- |
| **Algorithm Name** | **Description** | **Type** |
| **Linear regression** | Finds a way to correlate each  feature to the output to help predict future values. A variable's value can be predicted using linear regression analysis based on the value of another variable. | Regression |
| **Logistic regression** | Extension of linear regression that's used for classification tasks. The output variable is binary (e.g., only black or white) rather than continuous (e.g., an infinite list of potential  colors) | Classification |
| **Decision tree** | Highly interpretable  classification or regression model that splits data-feature. One non-parametric supervised learning approach that is used for both regression and classification applications is the decision tree. | Regression and Classification |

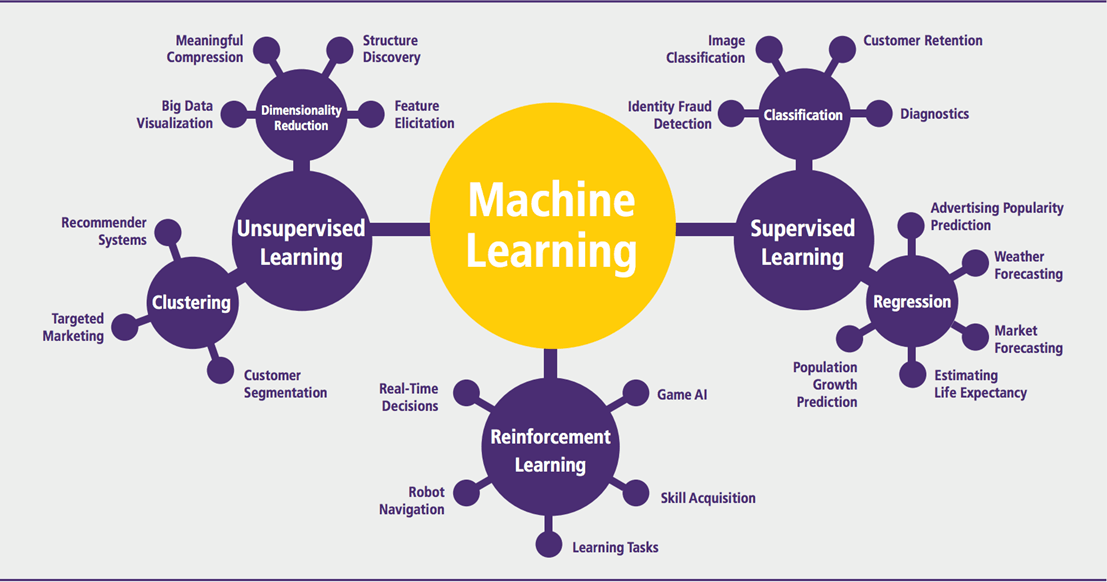
|  |  |  |
| --- | --- | --- |
| **Naïve bayes** | The Bayesian method is a classification method that makes use of the Bayesian theorem. The theorem updates the prior knowledge of an event with the independent probability of each feature that  can affect the event. | Regression Classification |
| **Support vector machine** | Support Vector Machine, or SVM, is typically used for the classification task. SVM algorithm finds a hyperplane that optimally divided the classes. It is best used with a  non-linear solver. | Regression (not very common) Classification |
| **Random forest** | The algorithm is built upon a decision tree to improve the accuracy drastically. Random forest generates many times simple decision trees and uses the 'majority vote' method to decide on which label to return. For the classification task, the final prediction will be the one with the most vote; while for the regression task, the average prediction of all the trees is the  final prediction. | Regression Classification |

The following points may be used to sum up the simple life of machine learning programs:

1. Specify the query.
2. Gather information.
3. Display the data.
4. Develop the algorithm
5. Evaluate the algorithm
6. Get input.
7. Streamline the algorithm
8. Repetition 4–7 until desired outcomes are obtained
9. Create a forecast using the model. The algorithm applies its learned conclusions to new data sets as soon as it becomes proficient at making the correct decisions.

### Machine Learning Algorithms and Where They Are Used?

The algorithm is built upon a decision tree to improve the accuracy drastically. Random forest generates many times simple decision trees and uses the 'majority vote' method to decide on which label to return. For the classification task, the final prediction will be the one with the most vote; while for the regression task, the average prediction of all the trees is the final prediction. The algorithm is built upon a decision tree to improve the accuracy drastically. Random forest generates many times simple decision trees and uses the 'majority vote' method to decide on which label to return. For the classification task, the final prediction will be the one with the most vote; while for the regression task, the average prediction of all the trees is the final prediction. There are several varieties of machine learning algorithms, and each is appropriate for a particular use case. When labeled data is available, supervised learning algorithms like Support Vector Machines (SVM) and Linear Regression are frequently employed for tasks like email classification into non-spam and spam categories or house price prediction. In order to find hidden patterns and structures in unlabeled data, unsupervised learning methods like Principal Component Analysis (PCA) and K-Means Clustering are used. This makes them ideal for dimensionality reduction and market segmentation.



### Fig. 6.1.3.1 Machine Learning Algorithms

Machine learning can be grouped into two broad learning tasks: Supervised and Unsupervised. There are many other algorithms.

## Supervised Algorithm:

Supervised instruction an algorithm learns the link between given inputs and provided outputs by using human feedback and training data a practitioner might forecast sales using input data such as marketing expenses and weather forecasts. When the output data is known, supervised learning may be used. New data will be predicted by the program. By training on a dataset that contains both input data and associated output labels, supervised learning algorithms are made to model connections between those two elements. By using this method, the computer is able to create a predictive model based on labeled data by learning from a dataset whose expected outputs are known. In order to teach the algorithm to learn the mapping from inputs to outputs, a large number of input-output pairs are fed to it during the training phase. In the case of email spam detection, for instance, the algorithm is trained on a dataset of emails that have been classified as "spam" or "not spam."

There are two categories of supervised learning:

* Classification task
* Regression task

**Classification:** Let's say you wish to determine a customer's gender for an ad. You will begin compiling information from your client database on height, weight, occupation, pay, shopping basket, etc. You are aware of the gender of every client you have; they can only be male or female. Assigning a likelihood of being male or female (i.e., the label) based on the information (i.e., characteristics you

have collected) is the classifier's goal. Once the algorithm has been trained to identify between male and female, additional data may be used to forecast. For example, you would like to know if the new information you received from an unidentified consumer is male or female. If 70% of the classifier's predictions are male.

Assigning incoming data to predetermined groups or classes based on its attributes is the aim of classification, a sort of supervised learning. A labeled dataset, in which each input is linked to a distinct class label, is used to train a model. To predict if a patient has a certain condition, for example, a classification algorithm may be trained using patient data, such as symptoms and medical history. In order to produce precise predictions on fresh, unobserved data, the algorithm must first understand the connections between the characteristics and the class labels. Classification challenges come in many different forms: multi-class classification (like identifying different animal species in photos) or binary classification (like determining if an image is spam or not).

Different algorithms are employed for categorization, each having advantages and being better suited for particular kinds of issues. Decision Trees are simple to understand but may be prone to overfitting since they provide a tree-like model of decisions based on feature values. Support Vector Machines (SVMs) are useful in high-dimensional spaces because they can identify the best hyperplane to divide classes with the largest margin. When dealing with binary classification issues, logistic regression is employed to calculate the likelihood of a class given its input features. By lowering variance and bias, ensemble approaches like Random Forests and Gradient Boosting Machines combine many models to improve accuracy and resilience. A number of variables, including the size of the dataset, the difficulty of the challenge, and the requirement for interpretability.

**Regression:** Regression analysis is done when the result is a continuous value. For example, a financial analyst could have to project a stock's value based on a variety of factors, such as equity, past stock performance, and the macroeconomic index. The goal of the system's training is to minimize inaccuracy in the stock price estimation.

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **Description** | **Type** |
| **K-means clustering** | Puts data into some groups (k) that each contains data with similar characteristics (as determined by the model, not in advance by humans) | Clustering |
| **Gaussian mixture model** | A generalization of k-means clustering that provides more flexibility in the size and shape of groups (clusters) | Clustering |
| **Hierarchical clustering** | Splits clusters along a hierarchical tree to form a classification system.  Can be used for Cluster loyalty- card customer | Clustering |
| **Recommender system** | Help to define the relevant data  for making a recommendation | Clustering |

Unsupervised education Without being provided a clear output variable, an algorithm investigates input data in unsupervised learning (e.g., studies customer demographic data to detect trends. In artificial intelligence, machine learning that takes place in the absence of human supervision is known as unsupervised learning. Unsupervised machine learning models, in contrast to supervised learning, are given unlabeled data and let to find patterns and insights on their own without explicit direction or instruction.

## APPLICATION OF MACHINE LEARNING Enhancement:

Machine learning, which helps people with daily chores on a personal or professional level without granting total control over the result. Numerous applications of this kind of machine learning exist, including software solutions, data analysis, and virtual assistants.

**Goal Automation:**

Machine learning, which functions completely on its own in any sector without the assistance of a person. For instance, robots in industrial facilities carry out crucial process steps, financial business The financial business is becoming more and more interested in machine learning. ML is mostly used by banks to identify patterns in data, but it is also used to stop fraud.

### Government Organization:

To oversee utilities and public safety, the government uses machine learning. Consider China, a country with a high degree of facial recognition. Artificial intelligence is used by the authorities to stop Jaywalkers. One of the earliest industries to employ machine learning for image identification was the healthcare sector Marketing: Due to easy access to data, marketing makes extensive use of AI. In the days before big data, researchers used sophisticated mathematical techniques like Bayesian analysis to calculate a customer's worth.

## Example of Application of Machine Learning in Supply Chain

An illustration of how machine learning is used in supply chains excellent results for visual pattern recognition are produced by machine learning, which opens up a wide range of possible applications in physical inspection and maintenance throughout the whole supply chain network. In the varied sample, unsupervised learning may find similar patterns quickly. The machine may then carry out a quality check on shipments that have wear and damage across the logistics hub. IBM's Watson platform, for example, has the ability to identify damage to shipping containers. Watson tracks, reports, and makes suggestions in real-time using a combination of visual and system-based data. A stock manager's evaluation and inventory forecasting in the previous year were mostly based on the principal technique. Better forecasting approaches (an improvement of 20 to 30 percent over standard forecasting tools) have been adopted when big data and machine learning are combined.

### Deep Learning

Deep learning is a branch of machine learning that specializes in algorithms that are modeled after the architecture and operation of the neural networks found in the human brain. Through the use of numerous layers of linked nodes (neurons) in artificial neural networks, the goal is to automatically learn representations of data. Deep learning's capacity to automatically extract complex patterns and features from massive volumes of raw data without the need for human feature engineering is one of its main advantages. This is accomplished by training the neural network on labeled data, which teaches the network to identify patterns and provide predictions by repeatedly modifying its internal parameters. In a number of fields, including computer vision, natural language processing, speech recognition, and reinforcement learning, deep learning has shown impressive promise. For instance, deep learning models in computer vision have attained cutting-edge results in tasks like object identification, picture production, and image categorization. Similar to this, deep learning models have proven invaluable in natural language processing for tasks like text production, sentiment analysis, and machine translation.

All things considered, deep learning has completely changed the area of artificial intelligence by making it possible to create extremely precise and adaptable models that can handle complicated tasks and data. Because of its broad use, technology is still advancing, and new applications in a variety of sectors are being made possible. Neural network topologies fall into various categories under deep learning, and each is best suited for a particular set of tasks and data. The most basic kind of neural networks are called feedforward neural networks (FNNs), which function best for simple classification and regression tasks since data travels from input to output through hidden layers in a single direction. Convolutional Neural Networks (CNNs) are very useful for image and video identification because they use convolutional layers to recognize features and spatial hierarchies. CNNs are particularly good at processing grid-like input, like pictures. For applications like language modeling and time series prediction, recurrent neural networks (RNNs) and more sophisticated versions, such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs), are optimized for sequential input and are capable of capturing temporal relationships. Generative Adversarial Networks (GANs) are frequently employed in picture production and data augmentation. They are composed of two competing networks, a discriminator and a generator, that cooperate to produce realistic synthetic data.

Deep learning operations are a collection of methods and procedures that let neural networks learn from and forecast complicated data. The training and operation of deep neural networks depend heavily on these activities, which include forward propagation, activation functions, loss functions, backpropagation, and optimization. The first stage of forward propagation is when the neural network processes incoming data in1 order to generate predictions. In each network neuron, this procedure involves multiplying each input characteristic by a weight and adding a bias term to the result. After then, the data are run through an activation function, which adds non-linearity to the model and enables it to learn increasingly intricate patterns. In order to make predictions and assess the network's performance during training, forward propagation is necessary.

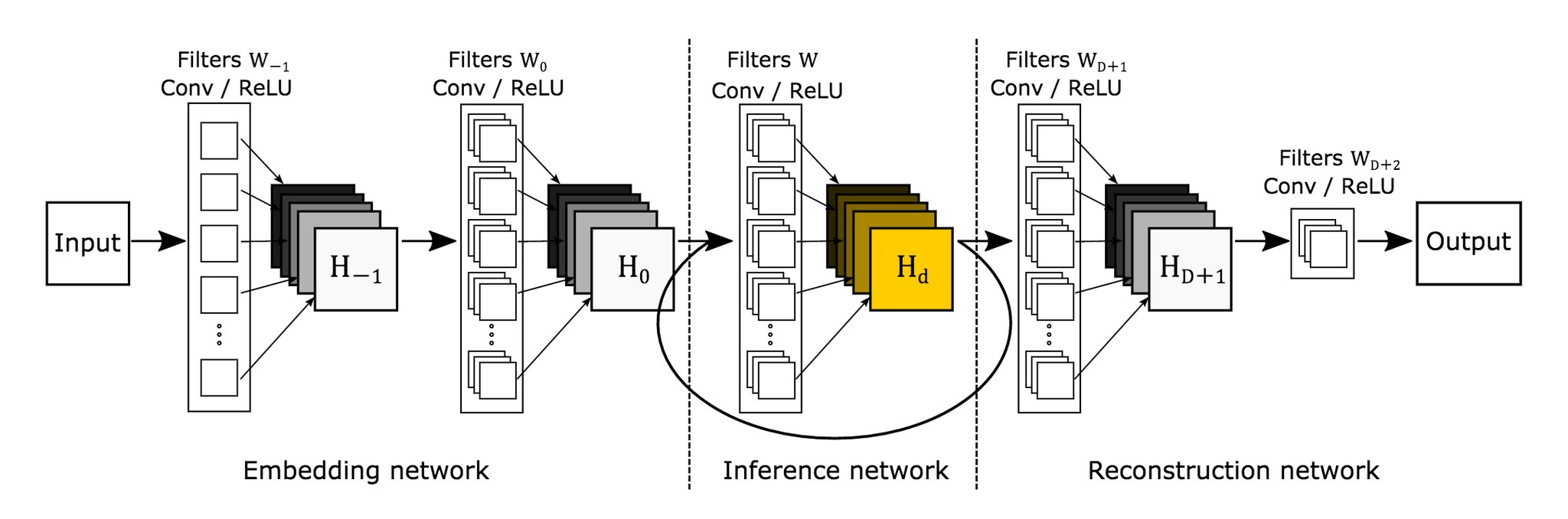


Fig.6.1.6.1 Deep Learning Architecture

Each neuron's output in a layer of a neural network is subjected to mathematical functions called activation functions. They contribute to the final output and decide whether a neuron should be triggered. Typical activation functions include the Rectified Linear Unit (ReLU), which effectively introduces non-linearity and mitigates the vanishing gradient problem since it outputs zero for negative inputs and the input value itself for positive inputs. In binary classification assignments, the Sigmoid function is frequently utilized, since it squashes output to a range between 0 and 1. In multi-class classification issues, on the other hand, the Softmax function normalizes output to a probability distribution over many classes. The discrepancy between the target values and the network's predictions is measured using loss functions. They offer an indicator of the model's performance and direct the process of optimization to increase accuracy.

Mean Squared Error (MSE), which computes the average squared difference between predicted and actual data, is frequently used for regression tasks. Cross-Entropy Loss, which penalizes inaccurate predictions more severely, is used to quantify the performance of classification models in tasks involving categorization. The nature of the issue and the kind of output the network is generating determine the loss function to use. The technique of changing the network's weights to reduce the loss function is known as backpropagation. It entails applying the calculus chain rule to determine the gradient of the loss function with regard to each weight. These gradients show which way to alter the weights in order to minimize error. Through the use of backpropagation, weights are modified as the error gradients are propagated from the output layer backward through the network to the input layer. The parameters of the model must be learned and improved through this iterative process.

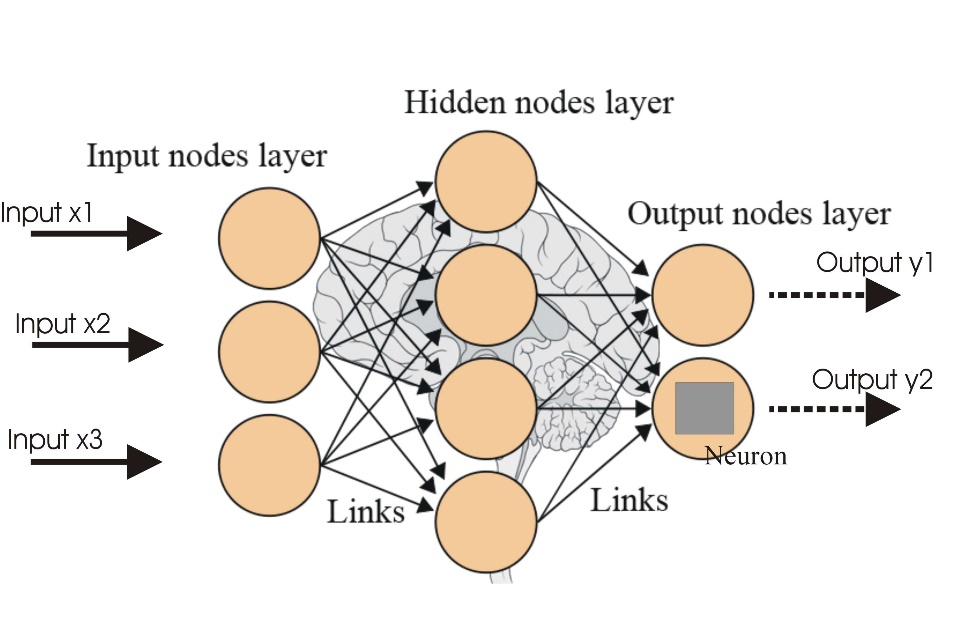


Fig.6.1.6.2 Neural Networks

To determine the ideal collection of weights that minimizes the loss function, optimization procedures are utilized. Stochastic Gradient Descent (SGD) is the most widely used optimization approach. It updates weights according to the gradient of the loss function with respect to each weight. Adaptive learning rates and momentum are included into SGD variants like Adam (Adaptive Moment Estimation) and RMSprop (Root Mean Square Propagation) to increase convergence and performance. In order to effectively train deep learning models and make sure they generalize well to new, unknown data, optimization strategies are essential.

## Reinforcement Learning

By taking actions and seeing the outcomes of those actions, an agent learns how to behave in an environment through reinforcement learning, a feedback-based machine learning approach. The agent receives positive feedback for each successful activity and negative feedback or a penalty for each unsuccessful one. Unlike supervised learning, reinforcement learning uses feedbacks to autonomously learn the agent without the need for labeled data. The agent can only learn by experience because there isn't any labeled data.

Recursion-based learning (RL) addresses a particular class of problems, such as robotics and game play, where the objective is long-term and decision-making is sequential. The agent investigates and engages with its surroundings on its own. In reinforcement learning, an agent's main objective is to increase performance by obtaining the greatest number of positive rewards. Through a process of trial and error, the agent gains knowledge and improves its performance of the job. Thus, it can be said that "Reinforcement learning is a type of machine learning technique where a computer program that is an intelligent agent interacts with the outside world and learns to act within that." One example of reinforcement learning is the process by which a robotic dog learns to move his arms.

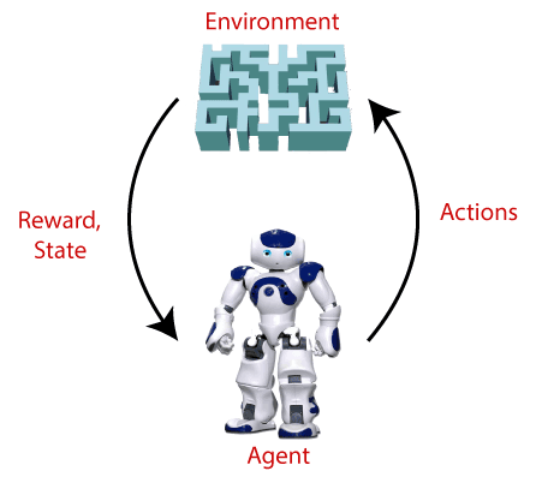


Fig.6.1.7.1 Reinforcement Learning

A dynamic area of machine learning called reinforcement learning (RL) mimics how animals and people pick up skills via trial and error. Through interaction and investigation, RL agents learn how to accomplish certain objectives in their surroundings. Throughout this process, actions are taken in various stages, consequences are assessed, and behavior is modified to optimize cumulative reward over time.RL stands apart for its emphasis on long-term returns, which frequently necessitate forgoing short-term earnings in favor of larger future rewards. The exploration-exploitation problem revolves around this trade-off, in which agents have to decide how best to maximize learning by combining known activities (exploitation) with novel actions (exploration). There are two types of RL algorithms: model-based and model-free techniques. Methods based on models learn a dynamic model of the environment to help in planning. Using neural networks to estimate complicated functions and learn representations from unprocessed sensory input, deep reinforcement learning (DRL) blends reinforcement learning (RL) with deep learning.

Notwithstanding its achievements, RL still has drawbacks such as sample inefficiency, instability, and the requirement for meticulous hyperparameter adjustment. Scholars are presently investigating several methods to tackle these obstacles, such as enhanced exploration tactics, target networks, and experience replay. Numerous sectors, including robots, autonomous cars, banking, healthcare, and more, have found extensive uses for reinforcement learning. Its capacity to pick up intricate behaviors and adjust to changing surroundings makes it an effective tool for handling challenges in the real world. All things considered, reinforcement learning signifies a profound change in the way that computers learn and interact with the outside environment. This holds fascinating prospects for creating intelligent systems that are able to make decisions on their own and adapt to their surroundings.

By contributing something, positive reinforcement learning increases the likelihood that the predicted behavior will recur. Positive effects are observed in the agent's behavior, leading to a rise in its strength. Long-term changes can be maintained by this kind of reinforcement, although excessive positive reinforcement can produce an excess of states that lessen the effects. The reverse of positive reinforcement is negative reinforcement learning, which makes it more likely that the particular behavior will recur by avoiding the unfavorable circumstance. Depending on the circumstance and conduct, it may be more successful than positive reinforcement; nonetheless, it only offers reinforcement for the bare minimum of performance.

### Applications/ Examples of Deep Learning Applications

**AI in Finance:** In order to reduce costs, save time, and generate value, the financial technology sector has already started implementing AI. Deep learning is transforming the loan industry by using more precise credit rating. Artificial intelligence (AI) may be used by credit decision-makers to develop thorough credit lending applications that take applicants' ability and character into account, enabling a quicker, more precise risk assessment. An AI solution is provided to credit creators by Underwrite, a fintech business. Underwrite.ai uses AI to identify the applicant with the best possibility of repaying a loan. Their approach is significantly more effective than traditional ones.

**AI in HR**: Sportswear juggernaut Under Armour uses AI to modernize the application process and change hiring. The hiring procedure at Under Armour's retail locations is actually reduced by 35%. 2012 saw a rise in interest in the popularity of Under Armour. On average, they got 30,000 resumes every month. Reading through each application and beginning the screening and interview process was taking too long. The duration of time required for hiring and integrating additional staff members impacted Under Armour's ability to maintain fully staffed, scaled, and functional retail stores. Under Armour's tools weren't particularly useful, even with all of the "must-have" HR technology in place at the time, including transactional solutions for sourcing, applying, monitoring, and onboarding.

**Artificial Intelligence in Marketing:** AI is a helpful tool for handling customization and customer support problems. Thanks to improved speech recognition in call center management and call routing, which is made possible by the use of AI technology, customers may enjoy a more seamless experience. For example, utilizing deep learning analysis of audio, computers may identify a consumer's emotional tone. If a consumer is not responding well to the AI chatbot, the system may switch the conversation over to real human operators. Beyond the three industries and enterprises listed above, artificial intelligence is widely used in many other sectors.

## Difference Between Machine Learning and Deep Learning

|  |  |  |
| --- | --- | --- |
| **S. No** | **Machine Learning** | **Deep Learning** |
| **1** | Machine Learning is a superset of Deep Learning | Deep Learning is a subset of Machine Learning |
| **2** | The data represented in Machine Learning is quite different compared to Deep Learning as it uses structured data | The data representation used in Deep Learning is quite different as it uses neural networks (ANN). |
| **3** | Machine Learning is an evolution of AI. | Deep Learning is an evolution of Machine Learning. Basically, it is how deep is the machine learning. |
| **4** | Machine learning consists of thousands of data points. | Big Data: Millions of data points. |
| **5** | Outputs: Numerical Value, like classification of the score. | Anything from numerical values to free-form elements, such as free text and sound. |
| **6** | Uses various types of automated algorithms that turn to model functions and predict future action from data. | Uses a neural network that passes data through processing layers to, interpret data features and relations. |
| **7** | Algorithms are detected by data analysts to examine specific variables in data sets. | Algorithms are largely self-depicted on data analysis once they’re put into production. |
| **8** | Machine Learning is highly used to stay in the competition and learn new things. | Deep Learning solves complex machine-learning issues. |
| **9** | Training can be performed using the CPU (Central Processing Unit). | A dedicated GPU (Graphics Processing Unit) is required for training. |
| **10** | More human intervention is involved in getting results. | Although more difficult to set up, deep learning requires less intervention once it is running. |
| **11** | Machine learning systems can be swiftly set up and run, but their effectiveness may be constrained. | Although they require additional setup time, deep learning algorithms can produce results immediately (although the quality is likely to improve over time as more data becomes available). |
| **12** | Its model takes less time in training due to its small size. | A huge amount of time is taken because of very big data points. |

### When To Use ML Or DL?

|  |  |  |
| --- | --- | --- |
|  | **Machine learning** | **Deep learning** |
| **Training dataset** | Small | Large |
| **Choose features** | Yes | No |
| **Number of algorithms** | Many | Few |
| **Training time** | Short | Long |

The table below lists the main differences between machine learning and deep learning. Machine learning uses less data to train the algorithm than deep learning does. Deep learning requires a massive and diverse data set in order to identify the underlying structure. Additionally, machine learning provides a faster-learning model. The most advanced deep learning architecture may need a few days to a week to train. The accuracy of deep learning is better than that of machine learning. The neural network learned how to choose important characteristics, so you don't need to know which features best reflect the data. It is up to you to decide what features to include in the model when it comes to machine learning. For instance, algorithms are able to determine the emotional tone of a consumer using deep-learning analysis of audio. The technology has the ability to divert a discussion to actual human operators, who take over if the customer is not reacting well to the AI chatbot.AI is extensively utilized in many other fields and businesses besides the three mentioned above.

While both machine learning (ML) and deep learning (DL) are subsets of artificial intelligence, their approaches to learning from data and degree of complexity are where they diverge most. In order to learn from data, machine learning algorithms use both conventional statistical techniques and characteristics that have been manually created. A kind of machine learning called deep learning makes use of multi-layered neural networks, or "deep networks," to automatically extract hierarchical characteristics from unprocessed input. Deep learning models, like transformers and convolutional neural networks, can extract and learn complex patterns directly from big datasets without the need for explicit feature engineering, in contrast to typical machine learning. Because DL can automatically find complex patterns and representations in large-scale, high-dimensional data, such text, audio, and pictures, it performs better than machine learning (ML) at simpler tasks with smaller datasets.

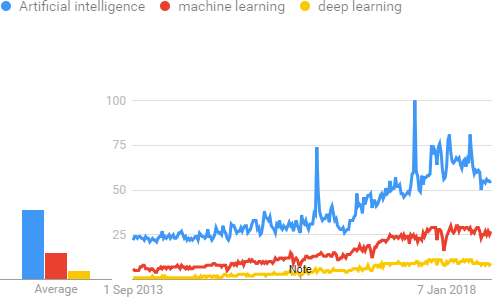


Fig.6.2.1 Analysis of AI, Machine Learning & Deep Learning

In the science of computational intelligence, artificial intelligence (AI), machine learning (ML), and deep learning (DL) reflect several abstraction levels, each with unique properties and uses. The most general name for any method that makes it possible for computers to carry out activities that ordinarily require human ability is artificial intelligence (AI). Artificial Intelligence encompasses a broad spectrum of techniques, ranging from probabilistic frameworks and heuristic algorithmic approaches to rule-based and expert systems. The main objective of artificial intelligence (AI) is to develop systems that can simulate or reproduce cognitive processes like thinking, problem-solving, and decision-making.

AI includes both conventional methods, like symbolic AI, which applies rules and logic, and contemporary methods, like machine learning (ML) and deep learning (DL), which concentrate on using data to learn and make predictions. ML techniques entail teaching models on past data to find trends and connections so they can anticipate or make judgments based on fresh data. Whereas conventional AI systems mostly depend on explicit programming and pre-established rules, machine learning models get increasingly and more effective over time as they are exposed to larger amounts of data. ML approaches include reinforcement learning, which teaches models by making mistakes; supervised learning, which trains models on labeled data; and unsupervised learning, which finds hidden patterns in unlabeled data.

### Artificial Intelligence

Today's globe is seeing rapid technological advancement, with new technologies emerging on a daily basis. Here, artificial intelligence is one of the burgeoning fields of computer science that is poised to bring about a new global revolution through the creation of intelligent machines. These days, artificial intelligence permeates everything. It is actively working on a wide range of subfields, from broad to specialized, including painting, performing music, chess, theorem proofing, self-driving automobiles, and more. One of the exciting and all-encompassing areas of computer science with a bright future is artificial intelligence (AI). AI has the potential to make a machine function like a human. The term artificial intelligence is derived from the term intelligence and artificial, which indicate "man-made" and "thinking power," respectively. As a result, AI stands for "a man-made thought power." Thus, AI may be defined as: It is an area of computing by which we may build intelligent devices which are capable of acting like individuals, think like beings, and able to make judgments.

When a machine is capable of human based functions like learning, thinking, and problem solving, it is said to possess artificial intelligence. The amazing thing about artificial intelligence is that, unlike preprogrammed machines, it is possible to build a machine with preprogrammed algorithms that can function on its own. Prior to studying artificial intelligence (AI), we need understand its significance and the reasons behind our need to learn about it. The key reasons to educate yourself about AI are as follows: AI may be used to develop software and hardware that can accurately and quickly address a variety of real-world problems, including those involving marketing, traffic, health, and other concerns. You may design your own virtual assistant using AI, much like Cortana, Google Assistant, Siri, and so on. AI may be used to create robots that can operate in environments where human existence may be at jeopardy. Other new technology, gadgets, and opportunities are made possible by AI. Artificial intelligence (AI) is a collection of technologies that provide computers the capacity to carry out a wide range of sophisticated tasks, such as data analysis, recommendation making, speech and text comprehension, and vision. Although it's a large field with many different approaches, here's a summary to get you started:

**History of Artificial Intelligence:**

The thought of imbuing non-living objects with intelligence has always piqued people's interest. Greek myths from antiquity had gods building intelligent devices, and Egyptian engineers moved statues. Thinkers who used symbols to explain how human thought functions, such as Aristotle and Ramon Llull, set the foundation for artificial intelligence. The development of modern computers began in the late 1800s and early 1900s. In the 1830s, Ada Lovelace and Charles Babbage created the first machines with programming capabilities. John Von Neumann developed the concept of storing computer programs in the 1940s. Walter Pitts and Warren McCulloch began developing the fundamentals of neural networks at the same time.

Modern computers were invented in the 1950s, enabling researchers to explore machine intelligence. In computer science, Alan Turing's Turing test gained significant attention. At a 1956 Dartmouth College conference, the first artificial intelligence program, the Logic Theorist, was unveiled. This is when the phrase "artificial intelligence" was first used. The ensuing years, known as "AI Winters," saw both prosperous and unsuccessful years for AI. We reached the boundaries of machine power and complexity in the 1970s and 1980s. But the excitement level returned in the late 1990s. There was more data and computers ran quicker. It was a significant event when IBM's Deep Blue defeated chess champ Garry Kasparov in 1997.

A new era in computer vision, language processing, and machine learning began in the 2000s. Cool new goods and services resulted from this. AI witnessed a boom in the 2010s with the introduction of voice assistants and self-driving automobiles. The field of generative AI, which creates creative content, has also begun to grow. The 2020s saw a surge in interest in generative AI, thanks to projects like ChatGPT-3 and Google's Bard. When you give these models an assignment, like as an essay or a piece of art, they may produce all kinds of new things. Recall that this technology is still in its infancy and that there are issues to be resolved, such as ensuring that it doesn't fabricate information.

### 6.3.1 Types of Artificial Intelligence

There are various methods to classify artificial intelligence, but the two primary ones are capabilities and functioning.

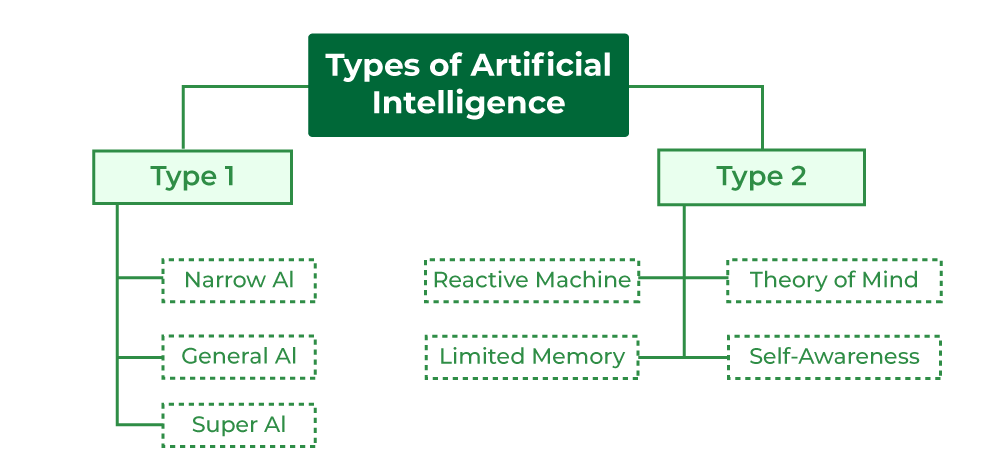


Fig.6.3.1 Types of Artificial Intelligence

Narrow AI, often referred to as Weak AI, is comparable to a specialist in the field of artificial intelligence. Think of it as a virtual expert that is intelligently focused on completing a single task. Consider Apple's Siri as an example. When it comes to voice commands and question responding, it is rather intelligent, but other than that, it is not very intelligent. Narrow AI functions within very specific bounds; if you ask it to do something outside of its comfort zone, it may not respond as you would expect. These days, artificial intelligence (AI) is used in everything from self-driving cars to smartphone picture identification. Example of narrow AI is Watson from BM. Though it is a supercomputer, it remains a specialist, combining machine learning, natural language processing, and expert systems.

The ultimate form of artificial intelligence is called "strong artificial intelligence," or "general AI." Imagine a system that was as efficient as a person at any intellectual job. The goal of general artificial intelligence (AI) is to build machines that can learn and think like people, but the catch is that no such system currently exists. Researchers from all over the world are putting a lot of effort and time into making it a reality, but it's a difficult road. AI is elevated to a whole new level by super AI. It's the ultimate state of machine intelligence, where robots are more intelligent than humans in every way.

Even though artificial intelligence is so broad and dependent on many other variables, it is not only a component of computer science. In order to construct AI, we must first understand how intelligence is made up. Intelligence is an intangible component of the human brain that is made up of a variety of skills including reasoning, learning, problem-solving, perception, linguistic understanding, etc. Artificial intelligence needs the following disciplines in order for a machine or program to accomplish the aforementioned goals: Biology, Mathematics, Mentality, Sociology, Computer Science and Neurons Data. It would be nothing short of revolutionary to reach such a degree of artificial intelligence, and it's a task that remains to be solved.

The simplest type of artificial intelligence is represented by reactive machines. These machines are aware just of the here and now; they are not guided by memories or experiences from the past. They just consider the immediate situation, reacting as best they can according to their training. Reactive machines include Google's AlphaGo, a master at the ancient game of Go, and IBM's Deep Blue, a chess-playing computer. Machines with limited memory can retain certain information or prior experiences, but only temporarily. They make judgments and navigate situations using this stored information. Self-driving automobiles are a prime example of this kind of artificial intelligence. To enable safe navigation on the road, these automobiles retain up-to-date information like as speed limits, distances, and neighboring car speeds.

Mental Theory Artificial Intelligence is still a research and development field. These artificial intelligence (AI) systems seek to mimic human emotions, thoughts, and social interactions. Although this kind of AI is still a ways off, scientists are making great progress in building devices that are able to comprehend and communicate with people on a more profound, emotional level. Self-Recognition The frontier of artificial intelligence to come is AI. These machines will have consciousness, feelings, and self-awareness, making them incredibly clever. They will surpass the capacity of the human intellect. It's important to remember, though, that Self-Awareness AI is still only a theoretical idea and has not yet been implemented in practice. Reaching this degree of AI would represent a significant advancement in knowledge and technology.

### 6.3.2 Applications of Artificial Intelligence

**Health-Care:** By improving diagnosis, tailoring therapy, and forecasting patient outcomes, artificial intelligence is transforming the healthcare industry. Advanced algorithms analyze medical pictures, such as X-rays and MRIs, with great accuracy to detect illnesses like cancer at early stages. AI-driven systems, such as IBM Watson Health, aid clinicians in identifying diseases and providing treatment choices based on detailed analysis of patient data and medical literature. Additionally, predictive models aid in the identification of those who may be at risk for certain illnesses, facilitating proactive treatment and prevention. AI-based customized medicine and genomics applications adjust medications based on each patient's unique genetic profile, increasing effectiveness and minimizing side effects.

**Finance:** Artificial intelligence (AI) algorithms are employed in the financial sector for automated trading, risk management, and fraud detection. Transaction patterns are analyzed by machine learning models to spot anomalies and highlight possible fraud. Artificial intelligence (AI)-powered systems, like those employed by JPMorgan Chase and other big banks, forecast market trends and help in investing strategy decision-making. In addition to analyzing a variety of variables outside of standard credit scores, algorithms also evaluate credit risk, offering a more comprehensive evaluation of a person's or company's creditworthiness. Robo-advisors employ AI to monitor client portfolios and provide individualized investing recommendations.

**Retail:** AI improves the shopping experience by managing inventories, making tailored suggestions, and providing customer support. AI is used by e-commerce sites such as Amazon to examine past browsing and purchase behavior and provide customized product suggestions that increase consumer happiness and revenue. Artificial intelligence (AI)-powered chatbots and virtual assistants answer client questions and offer round-the-clock assistance, increasing service effectiveness. AI also improves inventory control by anticipating demand, which lowers the likelihood of stockouts and overstocks. In order to maximize income and profitability, dynamic pricing algorithms modify prices in response to variables including demand, competition, and inventory levels.

**Transportation:** With improvements in logistics, traffic control, and driverless cars, artificial intelligence is revolutionizing transportation. In an effort to increase efficiency and safety, self-driving cars which have been created by firms like Waymo and Tesla use artificial intelligence (AI) to traverse roadways, identify impediments, and make driving choices. Real-time traffic data is analyzed by AI-powered traffic management systems to optimize signal timings and lessen congestion. In order to reduce fuel consumption and delivery times, AI algorithms in logistics optimally route delivery vehicles by taking into account variables like traffic patterns and delivery windows.

**Agriculture:** Crop monitoring, automated machinery, and precision farming are some of the uses of AI in agriculture. AI-equipped drones examine soil and crop health, giving farmers useful information for better planting and harvesting. AI-powered tools track meteorological trends and anticipate insect outbreaks, enabling farmers to take preventative action. Artificial Intelligence (AI) is used by automated tractors and harvesters to do operations like planting, weeding, and harvesting more efficiently and with less labor expense. AI algorithms also evaluate sensor data to optimize fertilization and irrigation, increasing agricultural yields and sustainability.

**Education:** AI improves education through administrative automation, intelligent teaching systems, and individualized learning. With the use of AI, adaptive learning platforms may customize tests and instructional materials to meet the requirements of specific students, allowing them to advance at their own speed. Personalized feedback and help are offered by intelligent tutoring systems, which target certain learning difficulties. AI-powered solutions free up teachers' time to concentrate more on teaching by automating administrative duties like scheduling and grading. Furthermore, students who are at danger of falling behind are identified by AI-driven analytics, allowing for prompt interventions and support.

**Entertainment:** AI improves audience engagement, recommendation systems, and content development in the entertainment sector. AI algorithms are used by streaming services such as Netflix and Spotify to examine user viewing and listening habits and provide tailored content suggestions. AI-driven technologies help with content production by producing art, music, and even screenplays, giving authors and artists creative support. AI also evaluates audience comments and social media to determine how popular movies, TV series, and other material is; this information helps producers and artists decide what to do next.

**Manufacturing:** AI enhances industrial processes via automation, quality control, and predictive maintenance. By using AI to evaluate equipment data and anticipate breakdowns before they happen, predictive maintenance solutions lower maintenance costs and downtime. AI-driven quality control systems use computer vision to check items for flaws, guaranteeing high standards and cutting down on waste. Automation boosts production uniformity and efficiency by using AI-powered robots to quickly and precisely complete jobs like welding, packing, and assembly. AI also improves supply chain management by controlling inventory levels and demand predictions.

**Customer Service:** Through sentiment analysis, chatbots, and virtual assistants, artificial intelligence improves customer service. Virtual assistants and chatbots respond quickly to consumer queries, taking care of common queries and problems all day long. Artificial intelligence (AI)-powered sentiment analysis systems track social media and customer reviews to measure consumer happiness and pinpoint areas in need of development. By examining prior contacts and preferences, AI systems also personalize consumer encounters by making customized recommendations and solutions. AI also automates support and ticketing routines, which expedites service delivery and enhances response times.

## 6.4 Libraries

Libraries are collections of prewritten Python code that offer a variety of features and tools to help developers do particular jobs more quickly and effectively. Libraries make it easy to reuse and share functionality by encapsulating code into modules, classes, and functions all without requiring the rewriting of complicated algorithms from begin. Python libraries greatly speed up development and increase productivity for a wide range of activities, including web development, machine learning, data analysis, and more. Python's core distribution comes with standard libraries, which offer a wide range of features that are often helpful in a variety of applications. These libraries provide modules for system interactions (sys, subprocess), data serialization (json, pickle), and file handling (os, shutil). Additionally, there are tools for network connections (socket), regular expressions (re), and more in the standard library. Python programming has a strong foundation thanks to these built-in libraries, which guarantee that routine tasks may be completed without the need to install extra packages.

Through the provision of specific functionality that may not be present in the standard library, third-party libraries enhance Python's capabilities. Examples include data manipulation and analysis libraries like NumPy, which provides support for large, complex arrays and matrices along with mathematical algorithms to operate on these arrays, and Pandas, which offers robust data functions and structures for working with structured data. Another well-liked package that makes interacting with online services and APIs simpler is called Requests. It streamlines HTTP requests. Package managers such as pip are commonly used to install third-party libraries, which enable developers to expand the capability of Python as required. Python's environment for data analysis, machine learning, and scientific computing depends heavily on scientific and numerical libraries. SciPy extends NumPy with a set of high-level commands and methods for data manipulation and visualization. Users may create machine learning models and algorithms with the help of scikit-learn, which offers tools for dimensionality reduction, clustering, regression, and classification. Renowned deep learning libraries TensorFlow and PyTorch facilitate the creation of intricate machine learning models and neural networks by providing a wealth of features for model construction and training.

## 6.4.1 Tensorflow

Google Brain created the open-source deep learning toolkit TensorFlow, which makes it easier to create and implement machine learning models. From straightforward linear regressions to intricate neural networks and extensive machine learning pipelines, it is made to tackle a wide range of jobs. Building, training, and deploying machine learning models is made easier for academics and developers by TensorFlow's vast collection of tools, libraries, and community resources. Because of its scalability and versatility, it is a preferred option for both commercial and scholarly applications. TensorFlow functions fundamentally using a computational graph paradigm in which dataflow graphs are used to describe calculations. These graphs show the mathematical operations as nodes and the data flow between them as edges. TensorFlow can now leverage distributed computing and parallelism to optimize the calculation of complicated models thanks to this approach. Utilizing computational graphs also makes it easier to carry out operations on several types of hardware accelerators, including CPUs, GPUs, and TPUs, which improves machine learning projects' performance and scalability.

Keras is a high-level API included in TensorFlow that makes neural network construction and training easier. Keras offers a user-friendly interface for building deep learning models with functional or sequential APIs. Both novices and seasoned practitioners may use it since it abstracts away many of the difficulties associated with defining and training neural networks. With Keras, users can experiment with multiple architectures, quickly prototype models, and incorporate different kinds of layers, activation functions, and optimization methods with ease. TensorFlow provides TensorFlow Extended (TFX), a collection of tools and frameworks made to handle the whole lifetime of machine learning models, for ready for manufacturing machine learning pipelines. Data encouragement transformation, model training, assessment, and serving are all included in TFX components. It guarantees that models are trained, evaluated, and deployed in a uniform and dependable way by supporting end-to-end processes. TensorFlow Model Analysis (TFMA), TensorFlow Data Validation (TFDV), and other components are integrated with TFX to manage model assessment, performance monitoring, and data preparation.

A more portable version of TensorFlow called TensorFlow Lite is intended for the deployment of machine learning models on mobile and edge devices. It offers tools that are specialized for converting and executing models on contexts with limited resources, like embedded systems, cellphones, and Internet of Things devices. To guarantee effective execution, TensorFlow Lite supports a wide range of hardware accelerators, such as custom accelerators and mobile GPUs. It may be used for real-time inference on low-power devices since it comes with a model optimization toolbox that helps lower latency and model size. A JavaScript library called TensorFlow.js enables users to run and train artificial intelligence models on Node.js or in web browsers. The client-side capabilities of TensorFlow are expanded by this library, allowing for real-time estimation of models and training in applications for the web. With the help of TensorFlow.js, machine learning models can be easily integrated into interactive online apps, allowing browser-based processing capability to be utilized for tasks like object identification, picture classification, and the processing of natural languages. For increased flexibility and control, TensorFlow lets users create bespoke training loops in addition to offering high-level APIs for model training. This functionality comes in very handy for implementing non-standard training procedures, unique loss functions, and advanced training methodologies.

TensorFlow's gradient computation functions and low-level operations allow users to construct training loops that are customized for particular applications or research requirements. Pre-trained models and other reusable machine learning components may be found in TensorFlow Hub. It speeds up development and experimentation by making it simple for users to exchange and reuse machine learning models across various projects. For a variety of applications, such as text embeddings, feature extraction, and picture classification, TensorFlow Hub offers a large selection of pre-trained models. With little effort, these models may be included into TensorFlow processes, utilizing cutting-edge methods and minimizing the requirement for in-depth training from start. A vibrant and dynamic community exists for TensorFlow, offering assistance and development via forums, guides, and open-source contributions. Many more tools and frameworks, such Tensor-Board for representation, TensorFlow Federated for federated instruction, and TensorFlow Quantum for nuclear machine learning, are part of the TensorFlow ecosystem. This vast ecosystem encourages creativity and teamwork, giving users the opportunity to take advantage of a wide range of tools and knowledge to develop their machine learning initiatives.

TensorFlow uses a number of strategies, including distributed computing, parallelism, and automatic differentiation, to maximize performance. Hardware accelerators like GPUs and TPUs, which greatly speed up machine learning model training and inference, are supported. TensorFlow also contains tools for benchmarking and debugging, helping users to find and fix performance issues. This focus on efficiency guarantees that TensorFlow can handle large-scale and complicated models effectively, making it ideal for both research and production applications. Numerous deployment methods are supported by TensorFlow, such as edge devices, on-premises servers, and cloud-based services. TensorFlow Serving is a dedicated library that offers a scalable and adaptable serving infrastructure for the deployment of machine learning models in commercial settings. It guarantees that models may be serviced effectively and consistently by supporting features like versioning, batching, and monitoring. TensorFlow is a flexible option for implementing machine learning apps at scale because of its scalability across many platforms and devices.

## 6.4.2 Keras

A high-level neural network API called Keras was created to make deep learning model development and training easier. It was first created as a stand-alone library, but it has since been included as an official high-level API in TensorFlow. Keras is renowned for its modular design and easy-to-use interface, which enable users to rapidly prototype, test, and implement intricate neural network structures. It offers a user-friendly interface for working with neural networks, making machine learning and deep learning accessible to both novices and experts. Keras prioritizes extension, modularity, and simplicity. The foundation of the library is its straightforward and consistent API, which makes it simple to create and test neural network models. Complex structures may be implemented with less code because to Keras' high-level abstractions for building layers, models, and optimizers. Deep learning model development and optimization depend heavily on quick experimentation and iteration, which is made possible by the design's encouragement of understandable and transparent code. The Sequential API and the Functional API are the two primary Keras APIs that are used to generate neural network models. A simple method for creating models layer by layer in a linear stack is the Sequential API.

Simple feedforward networks, in which the input moves through the layers consecutively, are the perfect fit for it. Contrarily, the Functional API is more adaptable and well-suited for creating intricate systems with shared layers, non-sequential data flows, and multi-input/output models. More control over the network topology may be achieved in the building of complex models thanks to the Functional API. Through its applications module, Keras offers access to a wide range of pre-trained models. These models, which may be utilized for extraction of features or fine-tuning on new tasks, include VGG16, ResNet, and Inception. They have been trained on huge datasets like ImageNet. When dealing with minimal data, pre-trained models may greatly accelerate development and increase performance. In addition to enabling users to adapt and alter these models to meet unique requirements, Keras supports loading these models with weights that have been trained beforehand.

Neural network designs may be quickly and simply constructed by combining the many pre-built layers that Keras gives. These include of common layers such pooling layers, recurrent (LSTM, GRU), convolutional (Conv2D), and dense (completely connected). Every layer in Keras represents a particular process and may be tailored using many options, including regularization strategies, activation functions, and the quantity of units. In order to enable users to build specific processes or components as needed, Keras also permits the construction of custom layers. Keras's fit, evaluate, and forecast techniques streamline the model training and assessment process. The fit technique manages the definition of the dataset, number of epochs, batch size, and optimization methods for the purpose of training models. Different metrics and loss functions are supported by Keras, and they may be tailored to the specific task at hand. While the predict technique is used to provide predictions from the trained model, the evaluate method offers a means to evaluate the model's performance on a validation or test dataset. By streamlining the training and assessment process, these techniques facilitate the monitoring and modification of model performance.

Several built-in optimizers in Keras, including SGD, Adam, and RMSprop, make it easier to optimize the parameters of the model during training. Weight adjustments and loss function minimization are mostly dependent on optimizers, and Keras offers an intuitive interface for choosing and tuning them. Furthermore, Keras provides a range of callbacks that enable users to track training progress and apply customized actions during distinct training phases. Features like preserving model checkpoints, halting training early based on performance, and viewing training metrics are made possible by callbacks like Model Checkpoint, Early Stopping, and Tensor Board.

Now that Keras is a part of TensorFlow, it can take advantage of its backend to achieve scalability and efficient processing. Through this integration, TensorFlow's cutting-edge capabilities such as support for GPU and TPU speed, distributed school, and deployment tools—are certain to benefit Keras models. TensorFlow's low-level APIs and additional elements, such TensorFlow Model Analysis (TFMA) and Data Validation (TFDV), may be easily integrated with Keras models, offering a complete environment for creating and implementing machine learning solutions. Keras boasts a thriving ecosystem and community that support and continue to improve the platform. The library has a huge amount of third-party resources, documentation, and tutorials accessible, and it is extensively used in universities as well as businesses. The Keras community actively works to create new tools and extensions, exchange best practices, and improve the library. This cooperative setting guarantees that Keras stays current with the most recent developments in deep learning and offers users helpful tools and assistance.

The deployment tools provided by TensorFlow may be used to set up Keras models in different contexts. Keras-trained models may be deployed on servers, mobile devices, and online apps by exporting them in formats that are compatible with TensorFlow Serving, TensorFlow Lite, and TensorFlow.js. TensorFlow Lite optimizes models for mobile and edge devices, whereas TensorFlow Serving offers a scalable and adaptable serving infrastructure. TensorFlow.js facilitates the instantaneous and interactive deployment of models within web browsers, hence allowing machine learning applications. Because Keras is extendable and configurable, users may create and include unique components into their models. Custom layers, loss functions, and optimizers may be created by users, giving them the freedom to use cutting-edge methods or modify already-established ones to suit particular requirements. Because of the modular design of the library, users may easily integrate these unique components and experiment with developing new deep learning architectures and techniques. One important aspect that makes Keras a strong and flexible tool for a variety of machine learning applications is its flexibility. A high-level machine learning package called Keras offers an intuitive user interface for creating, honing, and implementing neural network models. Because of its emphasis on extensibility, modularity, and simplicity, it is both powerful and approachable for both novice and seasoned practitioners. Keras uses cutting-edge computational tools and resources through its integration with TensorFlow, providing a broad range of uses from research to production.

## 6.4.3 Numpy

The core library for numerical computation in Python is called NumPy (Numerical Python), and it supports arrays, matrices, and a wide range of mathematical functions for working with these structures. Designed to overcome the shortcomings of Python's built-in list data structures for mathematical and scientific computations, NumPy presents nd-array, a powerful array object that facilitates the effective storing and handling of massive datasets. NumPy is an essential part of the scientific Python ecosystem because of its array-based design, which forms the basis of many scientific computing and data analysis packages in Python. The nd-array object, an array with multiple dimensions that enables vectorized operations that is, the ability to conduct operations on whole arrays without the need for explicit loops the foundation of NumPy. Through the use of low-level, optimized C and Fortran implementations, this feature improves performance. Because NumPy arrays are homogeneous that is, all of their members must be of the same type efficient computation and memory use are made possible. Large arrays can be processed fast for operations like element-wise addition, multiplication, and logical comparisons. Additionally, NumPy offers broadcasting, a method that enables arithmetic and other operations across arrays with suitable dimensions on arrays of various shapes without explicitly replicating data.

A wide range of mathematical functions are available in NumPy to facilitate intricate computations. These include more complex operations like polynomial computations and statistical measures, as well as more fundamental ones like logarithms, exponentials, and trigonometric functions. Strong support for linear algebra functions, including as matrix multiplication, eigenvalue decomposition, and singular value decomposition, is also provided by the library. These features are necessary for many applications in data analysis, machine learning, and scientific computing. Many numerical and algebraic problems need the use of functions for solving linear systems, calculating determinants, and performing decomposition, all of which are included in the linalg module of NumPy. Numpy. random, a robust module for producing random numbers in NumPy, offers utilities for sampling from different probability distributions and creating random numbers. For applications like statistical simulations, Monte Carlo techniques, and stochastic processes, this module offers operations like generating random samples from uniform, normal, and binomial distributions.

NumPy is made to work easily with other Python-based scientific computing frameworks. It forms the basis for libraries like Pandas, which provides data processing and analysis capabilities using DataFrames based on NumPy arrays, and SciPy, which expands on NumPy to give additional scientific and engineering tasks. Because of NumPy's and these libraries' compatibility, data analysis workflows are unified and robust, allowing users to take use of both NumPy's effective array operations and the specialized features offered by other libraries. Additionally, NumPy arrays are easily interfaceable with deep learning and machine learning frameworks like TensorFlow and PyTorch, as well as scikit-learn.

NumPy's effectiveness and efficiency while processing numerical data is one of its main advantages. Because NumPy's array operations are carried out in C and Fortran rather than Python, it can perform far faster than Python's built-in list operations. Because of its compact structures for data, which minimize latency and lower the amount of resource needed for huge datasets, the library also offers effective memory management. Due to its ability to perform difficult numerical tasks with efficiency and its support for arrays of dimensions and vectorized operations, NumPy is a recommended choice for projects involving large-scale data management and calculation. In conclusion, NumPy is a fundamental component of Python's numerical computing environment, providing a robust array object and an extensive set of mathematical functions for effective calculation. Its emphasis on efficiency, support for random number generation, and interaction with other scientific computing libraries make it a vital tool for machine learning, data analysis, and scientific research. NumPy gives users a strong basis for working with numerical data, making it simple to carry out intricate computations and create sophisticated algorithms. A vast set of high-level mathematical functions to work on huge, multi-dimensional arrays and matrices, as well as support for these arrays, are provided by the NumPy library for the Python programming language.

## 6.4.4 Pandas

With Pandas, you can easily handle and analyze structured data with its robust, adaptable, and user-friendly data structures. Pandas is an open-source Python data analysis and manipulation package. It is based on NumPy and easily combines with other scientific Python packages in the ecosystem. The Series and Data-Frame are the two primary data structures in Pandas. Pandas are an essential tool for data scientists, analysts, and engineers because of these structures' ability to work well with disparate data types and offer a broad variety of operations for data wrangling, exploration, and analysis. The Series and Data-Frame are the two main types of data structures in Pandas. A one-dimensional named array called a series may carry any kind of data, including texts, floats, and integers. It resembles a single row in a Data-Frame or a column in a spreadsheet. Similar to a spreadsheet or SQL table, a Data-Frame is a two-dimensional in nature labeled data structure containing columns that may be of various kinds. Effective data manipulation and analysis, such as filtering, grouping, merging, and reshaping, are made possible by it. Data access and manipulation are made simple with the use of straightforward indexing and selection procedures thanks to the labeling in both Series and Data-Frame.

Pandas is particularly good at cleaning and manipulating data, two essential processes in the pipeline for data analysis. It offers an extensive range of missing data handling capabilities, such as those for identifying, adding, and removing missing values. Pandas makes it simple to do data transformation operations like pivoting, concatenating, merging, and joining, allowing users to mix and reorganize data from many sources. Strong data aggregation and grouping facilities are also included in the library, enabling users to effectively compute summary statistics, apply functions to grouped data, and carry out sophisticated aggregations. Pandas has strong date handling and time series data support, which is especially helpful for financial analysis, forecasting, and data including temporal components. Tools for creating date ranges, resampling time series data, and executing time-based indexing and slicing are all included in the library. Users may simply execute tasks including moving, rotating windows, and computing time differences in addition to working with various frequencies and time zones. Pandas' integrated datetime feature makes it easy to work with time series data and integrate it with other time-based data analysis projects.

Although Pandas lacks powerful data visualization features, it works well with visualization libraries such as Matplotlib and Seaborn. Using the features of these libraries, users may quickly build line plots, scatter plots, histograms, and other types of charts straight from Pandas Data-Frames and Series. Pandas facilitates data sharing and interaction with other tools and systems by supporting exporting data to several formats, including CSV, Excel, and SQL databases. It is convenient to engage with several data sources and communicate findings when one can read and write data in diverse forms. Although Pandas is intended to handle data well, excessively big datasets may cause performance issues. The library optimizes processes using effective data structures and algorithms, although users may need to think about using other tools or methods for very big datasets. Pandas can analyze larger-than-memory datasets because to its integration with libraries like dask for out-of-core and parallel computation. These interfaces guarantee that Pandas is a useful tool for both small- and large-scale data analysis jobs by enabling users to carry out distributed calculations and work with big data more successfully. In conclusion, Pandas is a strong and adaptable Python library for data analysis and manipulation that provides effective data structures and a wide range of tools for working with structured data. Its support for time series and interaction with visualization tools improve its capabilities for thorough data analysis, and its Series and Data-Frame objects make complicated data operations easier.

Pandas continues to be an essential component of data science and analytics, offering users the capability they need to carry out complex data operations and obtain insightful knowledge, even in the face of performance issues with very big datasets. Pandas is a data manipulation and analysis software package designed for the Python programming language. It provides functions and data structures specifically for working with time series and numerical tables. Released under the three-clause BSD license, it is free software. Using statistical theories, pandas enables us to examine large amounts of data and draw conclusions. Pandas can tidy up unstructured data sets, making them more meaningful and understandable. In data science, pertinent data is crucial. Pandas is a robust and adaptable Python package that makes data manipulation jobs easier. Working with tabular data in spreadsheets or SQL tables is a great fit for pandas. When working with structured data in Python, data analysts, scientists, and engineers need the Pandas package.

## 6.4.5 Matplotlib & Seaborn

Two crucial Python libraries for data visualization are Matplotlib and Seaborn. They work well together and offer strong capabilities for making a variety of plots and charts. Although Seaborn builds upon Matplotlib to offer a more advanced interface and other functionality, Matplotlib remains the core Python charting package. When combined, they provide a full set of tools for data visualization, letting users produce visually appealing and educational visualizations for study and presentation. Plotting libraries like Matplotlib are popular and adaptable tools that provide a great deal of control over the look and behavior of charts. Matplotlib is a tool for producing static, animated, and interactive visualizations. It was developed by John D. Hunter in 2003. The pyplot package, which provides a plot creation interface akin to MATLAB, is the central component of Matplotlib. Using basic functions and parameters, users may create a wide range of plots, such as line plots, scatter plots, bar charts, histograms, and more. Plots may be highly customized with Matplotlib's versatility, including changing the colors, markers, line styles, axis labels, and legends.

The capacity of Matplotlib to produce intricate figures with several subplots is one of its advantages. Within a single figure, users can organize many plots in a grid pattern using the subplot function and its variations. This capability is very helpful when comparing several data sets or displaying various facets of the same data. Individual plot layouts, figure characteristics, and subplot size and spacing may all be modified by users. Moreover, Matplotlib allows the creation of multi-panel figures with common colorbars and axes, offering an adaptable method for structuring visualizations. The low-level API of Matplotlib offers a wide range of options for sophisticated charting methods. Plot components may be precisely controlled by users through direct interaction with the Axes and Figure objects, enabling them to design bespoke plots. Plots with specific characteristics, such contour plots, polar plots, and 3D plots, may be made using this feature. The mplot3d toolbox from Matplotlib expands its capabilities to 3D charting and provides tools for making three-dimensional surface plots, wireframes, and scatter plots. Additionally supported by the library are annotation, text rendering, and the addition of unique shapes or lines to plots.

Building upon Matplotlib, Seaborn is a high-level data visualization framework that offers an easier-to-use and more aesthetically beautiful interface for producing statistical visuals. Seaborn, created by Michael Waskom, makes it easier to create intricate visualizations by providing functions that manage routine operations with little to no coding. With Seaborn's good Pandas integration, users may plot straight from DataFrames and Series. Plot types include scatter plots, bar plots, violin plots, and pair plots, among others, all of which are intended to show distinct facets of data and statistical correlations. When it comes to producing statistical graphs that highlight correlations and distributions within data, Seaborn is an expert. It offers tools for using rug plots, kernel density plots, and histograms to visualize data distributions. Furthermore, Seaborn provides tools like scatter plots with regression lines and heatmaps of correlation matrices for displaying correlations between variables. Through the creation of a matrix of scatter plots and histograms, Seaborn's pairplot function facilitates the exploration of pairwise correlations between numerous variables. Seaborn is a useful tool for hypothesis testing and exploratory data analysis because of these features.

Seaborn places a strong emphasis on aesthetics and offers pre-existing themes and color schemes that improve a plot's visual attractiveness. Plots may be readily customized by users by switching between several themes, such as ticks, whitegrid, and darkgrid. Users are assisted in selecting the proper colors for various data types and plots by Seaborn's color palettes, which include divergent, sequential, and category palettes. Although Seaborn provides a great degree of customization, users may also incorporate Matplotlib customization options, fusing the comprehensive control of Matplotlib with the visual attractiveness of Seaborn. The process of building visualizations from structured data is streamlined by Seaborn's seamless connection with Pandas DataFrames and Series. Pandas objects may be sent directly to Seaborn charting routines, saving users from having to manually convert their data. With this connection, users may effectively plot intricate datasets and take advantage of Pandas' data manipulation features in conjunction with Seaborn's visualization tools. For example, users may create charts that show how Pandas was used for data grouping, aggregation, and filtering, producing visualizations that faithfully depict the underlying data.

## 6.4.6 Scikit-Learn

A popular open-source machine learning framework for Python called Scikit-Learn offers an extensive toolkit for creating and assessing machine learning models. Building upon NumPy, SciPy, and Matplotlib, Scikit-Learn is a component of the scientific Python environment that provides a uniform interface for a range of machine learning applications and methods. Because of its straightforward, dependable, and effective design, it is a vital resource for novices as well as seasoned data science and machine learning professionals. The main characteristic of Scikit-Learn is its dependable and intuitive API, which makes implementing machine learning algorithms easier. The classes and methods that comprise the library are well-defined and tailored to specific tasks including clustering, regression, classification, and dimensionality reduction. A consistent technique for training models, generating predictions, and assessing performance is provided via the common interface's methods like fit, predict, transform, and score. Because of this consistency, users may experiment with different techniques and move between different algorithms without having to get used to switching interfaces.

Scikit-Learn offers a wide variety of algorithms for classification and regression applications in supervised learning. The collection contains methods for classification, including k-Nearest Neighbors (k-NN), Random Forests, Gradient Boosting, and Support Vector Machines (SVM). Based on input information, these algorithms are used to predict category outcomes. In order to model the connection between input features and continuous target variables, Scikit-Learn provides regression techniques such as Linear Regression, Ridge Regression, Lasso Regression, and Polynomial Regression. Additionally, the library offers metrics for model evaluation, including F1 score, mean squared error, recall, accuracy, and precision, as well as R-squared. Unsupervised learning, in which the objective is to find hidden patterns or groups in data without specified labels, is another area in which Scikit-Learn offers capabilities. To divide data into discrete groups according to similarity, the library contains clustering methods including k-Means, DBSCAN, and Hierarchical Clustering. Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE) are two dimensionality reduction approaches that assist in lowering the amount of features while maintaining the crucial structure of the data. These methods are helpful for preparing high-dimensional data for additional analysis or modeling by displaying it.

## 6.4.7 Streamlit-Web Application

An open-source framework called Streamlit makes it easier to create engaging and eye-catching online apps for data science and machine learning. Its main objective is to simplify the process of creating web apps that enable users to engage with data models and visualizations for data scientists and machine learning engineers without having a deep understanding of web development tools. Because of its emphasis on usability and simplicity, Streamlit ha Fundamentally, Streamlit offers a simple API for writing little to no coding web apps. Using Streamlit's built-in functions, developers write their apps in pure Python and utilize them to construct user interfaces, show data, and communicate with other components. The framework manages the rendering and underlying web server, freeing up developers to write application logic and manipulate data. By eliminating the requirement for HTML, CSS, and JavaScript, Streamlit's method simplifies the development process and makes it understandable to people without prior experience with web programming. s gained popularity as a tool for quickly developing and implementing data-driven applications.

A variety of interactive widgets that improve user experience and make data analysis easier are part of Streamlit's capabilities. Users may interact with data and change parameters in real time by integrating widgets like sliders, buttons, text inputs, and file uploaders into apps with ease. A data scientist may, for instance, utilize a button to start data processing or a slider to change a machine learning model's threshold. Streamlit offers real-time feedback and engagement by dynamically updating the user interface and recalculating findings in response to user inputs. Support for data visualization is one of Streamlit's other important features. The framework allows developers to generate rich, interactive visualizations right within their applications by integrating smoothly with popular Python libraries like Matplotlib, Plotly, and Altair. Through user-friendly and interactive visual representations of data, Streamlit's API facilitates the display of charts, graphs, and other visual components, allowing users to derive insights from the data. This connection improves the user experience overall by making it easier to integrate sophisticated visuals into online apps.

Applications that are streamlit are made to be extremely responsive and to update automatically in real time as users interact with the interface. Reactive programming, a feature of Streamlit, guarantees that modifications to widgets or input variables automatically update the application's output, enabling this dynamic behavior. Simple Python functions may be used by developers to provide the logic of the application, and Streamlit handles state management and UI refreshes. With this method, consumers may have a smooth, engaging experience and quickly see the results of their activities. Applications developed using Streamlit are easy to deploy and require little configuration. Streamlit has a number of deployment choices, such as hosting apps locally, deploying to cloud services like AWS or Heroku, and utilizing Streamlit Sharing, which is Streamlit's platform for application hosting and sharing. Data scientists and developers may easily share their apps with others for internal usage, client demos, or public access due to the simplicity of deployment.

Applications may communicate with other services and retrieve data from several sources by integrating external APIs and data sources with Streamlit. With the help of this feature, developers may create apps that work with live data, analyze data in real time, and interface with other platforms and tools. A Streamlit app, for example, may retrieve data from a web API, analyze it, and show the outcomes in an interactive element or visualization, giving users access to current data and insights. Streamlit's emphasis on efficiency and simplicity is one of its advantages. The framework reduces the amount of boilerplate code needed to construct apps and is designed to be user-friendly. With just a few lines of code, developers can produce effective online apps, while Streamlit takes care of the technical aspects of web development in the background. This emphasis on simplicity speeds up the development process and facilitates faster iterations by enabling data scientists and machine learning practitioners to rapidly prototype and test their ideas. Moreover, Streamlit has tools for handling session data and application state. The integrated caching algorithms in Streamlit enable developers to store and retrieve data effectively, enhancing application speed and eliminating needless calculations. Moreover, Streamlit offers user session management features, which enable apps to save information during user interactions and deliver a customized experience. The overall dependability and functionality of Streamlit apps are enhanced by these characteristics.

## 6.5 Overview of Python Programming Language

Python is a popular high-level interpreted programming language that is easy to learn and has a lot of adaptability. Python, which Guido van Rossum created in the late 1980s, is now among the most widely used programming languages in the world. Its simple syntax promotes readability and clarity, making it easier to learn and understand quickly. Python's interactive mode allows for real- time experimentation with language features, a its interpreted nature facilitates quick development and testing. The language offers both type safety and flexibility at the same time because of its robust typing features and dynamic typing. High-level data structures like lists, dictionaries, tuples its interpreted nature facilitates quick development and testing. The language offers both type safety and flexibility at the same time because of its robust typing features and dynamic typing. High-level data structures like lists, dictionaries, tuple Python's success is due to its intuitive syntax, which puts an emphasis on readability and expressiveness and makes it possible for programmers to produce clear, succinct code.

Its "batteries-included" approach eliminates the need for external dependencies and accelerates development by offering a sizable standard library covering a wide variety of applications. Furthermore, Python's dynamic nature facilitates quick prototyping and experimentation, which makes it a great language for developers of all skill levels. Its accessibility is enhanced by its rich documentation and vibrant community, which offer a wealth of learning and troubleshooting tools. Python's flexibility also includes support for several programming paradigms, which gives developers the freedom to select the one that best fits the needs of their particular project. Python provides reliable solutions for a wide range of use cases, including creating web applications using frameworks like Django or Flask, analyzing data with tools like Pandas and NumPy, and creating machine learning models with TensorFlow or PyTorch. Because of its cross-platform interoperability, Python programming may be deployed and collaborated upon more easily across many operating systems. Python continues to be at the vanguard of contemporary software development, propelling innovation and enabling solutions across sectors with its constantly expanding ecosystem and ongoing development. Python's interactive mode allows for real-time experimentation with language features, a its interpreted nature facilitates quick development and testing.

The language offers both type safety and flexibility at the same time because of its robust typing features and dynamic typing. High-level data structures like lists, dictionaries, tuples its interpreted nature facilitates quick development and sets that are built into Python make typical programming tasks and data processing easier. Moreover, the vast Python standard library provides modules for a variety of tasks, decreasing reliance on other libraries. Efficient memory allocation and deallocation are ensured by automatic garbage collection memory management, decreasing Python's platform independence promotes portability by enabling code to operate smoothly on several operating systems. Python is incredibly versatile because of its vast ecosystem and active community. Thousands of third-party packages are available for Python, covering a wide range of fields. With support for procedural, object-oriented, and functional programming, Python is a versatile programming language that can be used for a multitude of tasks, ranging from web development and automation to scientific computing, artificial intelligence, and scripting. All things considered, Python's readability, strength, and wide range of applications make it a great option for developers and businesses looking for a versatile and strong programming language for a variety of tasks and applications.

Python's success is due to its intuitive syntax, which puts an emphasis on readability and expressiveness and makes it possible for programmers to produce clear, succinct code. Its "batteries- included" approach eliminates the need for external dependencies and accelerates development by offering a sizable standard library covering a wide variety of applications. Furthermore, Python's dynamic nature facilitates quick prototyping and experimentation, which makes it a great language for developers of all skill levels. Its accessibility is enhanced by its rich documentation and vibrant community, which offer a wealth of learning and troubleshooting tools. Because of its cross-platform interoperability. Python programming may be deployed and collaborated upon more easily across many operating systems. Python continues to be at the vanguard of contemporary software development, propelling innovation and enabling solutions across sectors with its constantly expanding ecosystem and ongoing development.

### 6.5. History Of Python

Guido van Rossum created Python in the late 1980s as an alternative to the ABC programming language. Its design philosophy was centered on the readability, simplicity, and maintainability of the code. Here is a quick synopsis of Python's past: Origin (late 1980s): In December 1989, while employed at the Centrum Wiskunde & Informatica (CWI) in the Netherlands, Guido van Rossum, a Dutch programmer, started working on Python. His goal was to develop a language that matched the simplicity of ABC with the strength of C/C++. First Release (1991): In February 1991, Python 0.9.0, the language's initial release, was made available. Modules, functions, and exception handling were provided in this release.

Early Growth (1990s): Python's readability and simplicity of use propelled it to steady popularity throughout the 1990s. 2008–2009: The switch to Python 3: In December 2008, Python 3.0, sometimes referred to as Python 3000 or "Py3k," was made available. In order to fix inconsistencies and enhance overall design, this significant upgrade included backward-incompatible language modifications. Due to compatibility issues, the switch from Python 2 to Python 3 was made gradually; Python 2 will continue to receive support until its end of life in January 2020. Progress (2010s): Python's widespread adoption in this decade was fueled by its adaptability and large library and framework ecosystem. The language had significant updates during this time, with Python 3.4 (2014), 3.5 (2015), 3.6 (2016), 3.7 (2018), and 3.8 (2019) being among the most notable releases.

Between the 2010s and 2020s, Python 3 adoption occurred. In spite of early opposition, As of right now, Python is still one of the most widely used programming languages in the world. It powers a variety of applications in many different fields, such as scientific computing, web development, data analysis, machine learning, and artificial intelligence. Its readability, adaptability, and simplicity continue to draw in developers and businesses alike, assuring its continued relevance in the ever- changing world of technology. Python began as a side project for Guido van Rossum and has since grown into a sophisticated programming language with a thriving community and vast ecosystem. Its history is proof of the effectiveness of pragmatism and simplicity in software design, which has made it a favorite among developers for creating cutting-edge and significant solutions.

## Python Features

Python is a strong and flexible programming language that has a large library and framework ecosystem in addition to being easy to learn and understand. Here are a few of Python's main attributes:

**Simple to Learn and Read:** Python's syntax is intended to be understandable and readable by both novice and seasoned programmers. The mental effort needed to comprehend and create code is lessened by its simple and uniform syntax. Python is an interpreted language, which means that commands are carried out line by line. This feature enables quick development and testing. Users may experiment with language features in real-time and execute code interactively with its interactive mode.

**Strong Typing and Dynamic Typing:** Python is dynamically typed, which eliminates the need for explicit typing by determining variable types at runtime. Strong typing implies that actions on variables take into account the type of the variable, which indicates that variables indeed have a type. Because of dynamic typing, a variable's type is only known at runtime. When conducting operations, types must be compatible with the operand due to strong typing.

**Data Structures:** Python comes with built-in support for a number of high-level data structures, including sets, dictionaries, tuples, and lists, which make it easier to handle and manipulate data and simplify typical programming tasks. The list, set, tuple, and dictionary data structures are the fundamental ones in Python. Every data structure is distinct in its own right. Data structures are "containers" that classify and arrange information based on their nature. The mutability and order of the data structures are different.

**Large Standard Library**: The extensive standard library that comes with Python provides modules and functions for a variety of activities, such as database access, file input and output, networking, web development, and more. Development is made easier and requires fewer third-party dependencies thanks to this large library.

**Object-Oriented Programming (OOP):** Python is compatible with OOP concepts, which let programmers use classes and objects to write modular, reusable code. Procedural and functional programming are two more programming paradigms that it supports. OOP uses four fundamental principles encapsulation, inheritance, polymorphism, and abstraction to enable objects to communicate with one another. These four OOP concepts allow objects to cooperate and interact with one another to build strong applications.

**Cross platform Compatibility:** Because Python is platform-independent, code written in it can execute on any system that can run the Python interpreter, such as Windows, macOS, Linux, and operating systems that resemble Unix. The capacity of software programs to function across many hardware platforms, operating systems, and devices is known as cross-platform compatibility. In the modern digital world, where consumers depend on a variety of devices and operating systems to access apps and data, its significance has grown.

**Community and Ecosystem:** A sizable and vibrant group of programmers create libraries, frameworks, and other tools for Python, which in turn benefits the ecosystem. Thousands of third-party packages that expand Python's capabilities for a variety of fields, such as web development, data science, machine learning It is a general-purpose programming language. Because of its adaptability and flexibility, it may be used for a variety of tasks, including web development, scientific computing, artificial intelligence, scripting, and automation. ease because of its user-friendly interface. Furthermore, by enabling users to establish and maintain isolated environments with various package versions, Anaconda Navigator streamlines environment management and guarantees repeatability and compatibility across projects. Anaconda Navigator improves the development experience by integrating.

## Visual Studio Code Navigator

The Visual Studio Code integrated programming environment (IDE) has a strong feature called Visual Studio Code (VS Code) Navigator, which is intended to improve navigation and expedite the coding process. This essential IDE function gives developers a simple and effective method to explore and manage their codebase. By integrating a number of features that facilitate rapid access to files, symbols, and code places, VS Code Navigator helps developers increase productivity and save time spent on navigational activities. VS Code Navigator's capability to give a summary of the project's file structure is one of its primary features. Developers can explore and open files with ease thanks to the Explorer pane in VS Code Navigator, which offers an organized overview of the project files and folders. The file look, file opening up and folder expansion functions supported by this pane make it easier for developers to find the files they need fast. Developers may easily navigate through complicated project structures by utilizing the Explorer window, which saves time spent looking for specific files and increases productivity.

VS Code Navigator provides capabilities for navigating symbols inside a file in addition to file navigation. The functions, classes, and variables found in the currently open file are represented in an organized manner by the Outline view. With this approach, developers may choose symbols from a list and easily navigate to them, making code editing and review more efficient. The Outline view facilitates developers' understanding of the code structure and allows them to identify certain sections without having to navigate through the whole file by giving a comprehensive overview of all the symbols in a file. Additionally, VS Code Navigator has strong search features that improve code exploration. Developers may search the whole source for text, including file names, content, and symbols, using the Search window. The ability to locate certain code snippets, variables, or functions across many files is a very helpful feature. Developers can now execute sophisticated queries and easily make mass changes thanks to advanced search features like search and replace and regular expressions.

The Go to Definition feature of VS Code Navigator is another important tool that developers may use to rapidly browse to the definition of a symbol, such a function or variable, right from where the symbol is used in the code. Developers are able to quickly get to the place where a symbol is defined by either using a keyboard shortcut or right-clicking on the symbol and choosing "Go to Definition". This feature helps developers understand the specifics of how functions and classes are implemented, which helps them make well-informed updates and guarantees that the changes are compatible with the rest of the codebase. Developers may now navigate the codebase more easily by going straight to a certain symbol using the Go to Symbol functionality. Developers have the ability to search for and choose symbols by name, including variables, functions, and methods, by using the "Go to Symbol" command or keyboard shortcut. Large codebases are a good fit for this functionality since manual navigating would take a lot of time. It increases productivity and lowers the risk of mistakes by allowing developers to swiftly reach pertinent areas of the code.

Developers may now navigate the codebase more easily by going straight to a certain symbol using the Go to Symbol functionality. Developers have the ability to search for and choose symbols by name, including variables, functions, and methods, by using the "Go to Symbol" command or keyboard shortcut. Large codebases are a good fit for this functionality since manual navigating would take a lot of time. It increases productivity and lowers the risk of mistakes by allowing developers to swiftly reach pertinent areas of the code. The usefulness of VS Code Navigator is further increased by its interface with version control systems. The IDE's Source Control window gives developers a visual view of changes, commits, and branch information, enabling them to manage their version control process. With the help of this connection, developers can easily explore their codebase and stay informed about version history and changes. It also makes code review, merging, and dispute resolution easier. VS Code Navigator provides functionality to organize and transition between several open files and editors, which is useful for developers working on complicated projects. The split-view feature and editor tabs allow developers to work on numerous files at once and examine various code portions side by side. This feature is necessary for efficiently handling big codebases, carrying out code reviews, and editing several files at once.

## Code

!nvidia-smi

!pip install --quiet pytorch\_lightning

!pip install --quiet transformers

!pip install --quiet seaborn

!pip install --quiet wget

import pytorch\_lightning as pl

import json

import pandas as pd

import numpy as np

import torch

from torch.utils.data import Dataset, DataLoader

from pytorch\_lightning.callbacks import ModelCheckpoint

from pytorch\_lightning.loggers import TensorBoardLogger

from sklearn.model\_selection import train\_test\_split

from termcolor import colored

import textwrap

from transformers import AdamW, T5ForConditionalGeneration, T5TokenizerFast as T5Tokenizer

from tqdm.auto import tqdm

import seaborn as sns

from pylab import rcParams

import matplotlib.pyplot as plt

from matplotlib import rc

sns.set(style='whitegrid',palette='muted',font\_scale=1.2)

rcParams['figure.figsize'] = 16, 6

pl.seed\_everything(42)

url = 'news\_summary.csv'

df = pd.read\_csv(url,encoding='latin')

df.columns

df = df[["text","ctext"]]

df.head()

df.columns

df.columns = ["summary", "text"]

df = df.dropna()

df.head()

df.shape

train\_df, test\_df = train\_test\_split(df,test\_size=0.1)

train\_df.shape,test\_df.shape

class NewsSummaryDataset(Dataset):

def \_\_init\_\_(

self,

data : pd.DataFrame,

tokennizer : T5Tokenizer,

text\_max\_token\_len : 512,

summary\_max\_token\_len : 128):

self.tokennizer = tokennizer

self.data = data,

self.text\_max\_token\_len = text\_max\_token\_len

self.summary\_max\_token\_len = summary\_max\_token\_len

def \_\_len\_\_(self):

return len(self.data)

def \_\_getitem\_\_(self, index : int):

data\_row = self.data[0].iloc[index]

text = data\_row["text"]

text\_encoding = self.tokennizer(

text,

max\_length = self.text\_max\_token\_len,

padding = "max\_length",

truncation = True,

return\_attention\_mask = True,

return\_tensors = "pt"

)

summary\_encoding = self.tokennizer(

data\_row["summary"],

max\_length = self.summary\_max\_token\_len,

padding = "max\_length",

truncation = True,

return\_attention\_mask = True,

return\_tensors = "pt" # Return PyTorch tensors

)

labels = summary\_encoding["input\_ids"]

labels[labels==0] = -100

return dict(

text = text,

summary = data\_row["summary"],

text\_input\_ids = text\_encoding["input\_ids"].flatten(),

text\_attention\_mask = text\_encoding["attention\_mask"].flatten(),

labels = labels.flatten(),

labels\_attention\_mask = summary\_encoding["attention\_mask"].flatten()

)

class NewsSummaryDataModule(pl.LightningDataModule):

def \_\_init\_\_(

self,

train\_df : pd.DataFrame,

test\_df : pd.DataFrame,

tokenizer: T5Tokenizer,

batch\_size : int = 8,

text\_max\_token\_len : int = 512,

summary\_max\_token\_len :int = 128

):

super().\_\_init\_\_()

self.train\_df = train\_df

self.test\_df = test\_df

self.batch\_size = batch\_size

self.tokenizer = tokenizer

self.text\_max\_token\_len = text\_max\_token\_len

self.summary\_max\_token\_len = summary\_max\_token\_len

# LightningModule.setup(stage=None)

# Called at the beginning of fit (train + validate), validate, test, or predict.

def setup(self, stage=None):

self.train\_dataset = NewsSummaryDataset(

self.train\_df,

self.tokenizer,

self.text\_max\_token\_len,

self.summary\_max\_token\_len)

self.test\_dataset = NewsSummaryDataset(

self.test\_df,

self.tokenizer,

self.text\_max\_token\_len,

self.summary\_max\_token\_len)

def train\_dataloader(self):

return DataLoader(

self.train\_dataset,

batch\_size=self.batch\_size,

shuffle= True,

num\_workers=2

)

def val\_dataloader(self):

return DataLoader(

self.test\_dataset,

batch\_size=self.batch\_size,

shuffle= False,

num\_workers=2

)

def test\_dataloader(self):

return DataLoader(

self.test\_dataset,

batch\_size=self.batch\_size,

shuffle= False,

num\_workers=2

)

MODEL\_NAME = "t5-base"

tokenizer = T5Tokenizer.from\_pretrained(MODEL\_NAME)

text\_token\_counts = []

summary\_token\_counts = []

for \_, row in train\_df.iterrows():

text\_token\_count = len(tokenizer.encode(row["text"]))

text\_token\_counts.append(text\_token\_count)

summary\_token\_count = len(tokenizer.encode(row["summary"]))

summary\_token\_counts.append(summary\_token\_count)

fig, (ax1,ax2) = plt.subplots(1,2)

sns.histplot(text\_token\_counts,ax=ax1)

ax1.set\_title("Full text token counts")

sns.histplot(summary\_token\_counts,ax=ax2)

ax2.set\_title("Summary text token counts")

N\_EPOCHS = 10

BATCH\_SIZE = 16

data\_module = NewsSummaryDataModule(train\_df,test\_df,tokenizer,batch\_size=BATCH\_SIZE)

class NewsSummaryModel(pl.LightningModule):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.model = T5ForConditionalGeneration.from\_pretrained(MODEL\_NAME,return\_dict=True)

def forward(self,input\_ids,attention\_mask,decoder\_attention\_mask, labels=None):

output = self.model(

input\_ids,

attention\_mask = attention\_mask,

labels = labels,

decoder\_attention\_mask = decoder\_attention\_mask

)

return output.loss, output.logits

def training\_step(self, batch, batch\_idx):

input\_ids = batch["text\_input\_ids"]

attention\_mask = batch["text\_attention\_mask"]

labels = batch["labels"]

labels\_attention\_mask = batch["labels\_attention\_mask"]

loss, outputs = self(

input\_ids = input\_ids,

attention\_mask = attention\_mask,

decoder\_attention\_mask = labels\_attention\_mask,

labels = labels

)

self.log("train\_loss",loss,prog\_bar=True,logger=True)

return loss

def validation\_step(self, batch, batch\_idx):

input\_ids = batch["text\_input\_ids"]

attention\_mask = batch["text\_attention\_mask"]

labels = batch["labels"]

labels\_attention\_mask = batch["labels\_attention\_mask"]

loss, outputs = self(

input\_ids = input\_ids,

attention\_mask = attention\_mask,

decoder\_attention\_mask = labels\_attention\_mask,

labels = labels

)

self.log("val\_loss",loss,prog\_bar=True,logger=True)

return loss

def test\_step(self, batch, batch\_idx):

input\_ids = batch["text\_input\_ids"]

attention\_mask = batch["text\_attention\_mask"]

labels = batch["labels"]

labels\_attention\_mask = batch["labels\_attention\_mask"]

loss, outputs = self(

input\_ids = input\_ids,

attention\_mask = attention\_mask,

decoder\_attention\_mask = labels\_attention\_mask,

labels = labels

)

self.log("test\_loss",loss,prog\_bar=True,logger=True)

return loss

def configure\_optimizers(self):

return AdamW(self.parameters(),lr = 0.0001)

model = NewsSummaryModel()

import torch

torch.cuda.is\_available()

checkpoint\_callback = ModelCheckpoint(

dirpath="checkpoints",

filename="best-checkpoint",

save\_top\_k=1,

verbose=True,

monitor="val\_loss",

mode="min"

)

logger = TensorBoardLogger("lightning\_loss",name="news-summary")

trainer = pl.Trainer(

logger=logger,

callbacks=[checkpoint\_callback],

max\_epochs=N\_EPOCHS,

accelerator="auto",

enable\_progress\_bar=True

)

trainer.fit(model,datamodule = data\_module)

trained\_model = NewsSummaryModel.load\_from\_checkpoint(trainer.checkpoint\_callback.best\_model\_path)

trained\_model.freeze()

def summarize\_text(text):

device = trained\_model.device # Get the device of the trained model

text\_encoding = tokenizer(

text,

max\_length=512,

padding="max\_length",

truncation=True,

return\_attention\_mask=True,

add\_special\_tokens=True,

return\_tensors="pt"

)

# Move input tensors to the same device as the trained model

text\_encoding = {key: value.to(device) for key, value in text\_encoding.items()}

generated\_ids = trained\_model.model.generate(

input\_ids=text\_encoding["input\_ids"],

attention\_mask=text\_encoding["attention\_mask"],

max\_length=150,

num\_beams=2,

repetition\_penalty=2.5,

length\_penalty=1.0,

early\_stopping=True

)

# Move generated\_ids back to CPU if it was on GPU

generated\_ids = generated\_ids.cpu() if device.type == 'cuda' else generated\_ids

preds = [tokenizer.decode(gen\_id, skip\_special\_tokens=True, clean\_up\_tokenization\_spaces=True)

for gen\_id in generated\_ids]

return " ".join(preds)

sample\_row = test\_df.iloc[0]

text = sample\_row["text"]

model\_summary = summarize\_text(text)

text

model\_summary

sample\_row["summary"]

text = "Vedanta's chairman Anil Agarwal earlier this week announced the biggest investment of ₹1.54 lakh crore for setting up the country's first-ever semiconductor chip plant in Gujarat."

summarize\_text(text)

from nltk.translate.bleu\_score import corpus\_bleu

from nltk.translate.bleu\_score import sentence\_bleu

reference\_summaries = test\_df["summary"].tolist() # Ground truth summaries

generated\_summaries = [summarize\_text(text) for text in test\_df["text"]]

# Compute ROUGE scores

rouge\_scores = corpus\_bleu(reference\_summaries, generated\_summaries)

print("ROUGE Score:", rouge\_scores)

choice = st.sidebar.selectbox("Select your choice", ['Home',"Summarize Text", "Summarize Document"])

if choice == "Home":

st.title("Article Summarization to Enhance Information Retrieval")

#make image center

st.image("1.png", use\_column\_width=True)

if choice == "Summarize Text":

st.subheader("Text Summarization to Enhance Information Retrieval")

input\_text = st.text\_area("Enter your text here")

if input\_text is not None:

if st.button("Summarize Text"):

col1, col2 = st.columns([1, 1])

with col1:

st.markdown("\*\*Your Input Text\*\*")

st.info(input\_text)

# CHAPTER 7

**SYSTEM TESTING**

# SYSTEM TESTING

## Testing Methods

Software testing is the systematic process of evaluating a software program or system to ensure that it meets requirements and performs as intended. Its primary goals are to validate requirements, find defects, limit defect leaking, and improve overall quality. Unit, integration, system, and acceptance testing are among the several stages of the software development lifecycle during which testing may occur. A range of techniques and approaches, such as automated testing, exploratory testing, and manual testing, can be used to conduct effective testing. Additionally, several testing methodologies focus on distinct aspects of the software, including non-functional testing, security, usability, regression, and performance testing. Ultimately, software testing ensures that end users obtain software products that are trustworthy, high-quality, and defect-free, which makes it an essential part of software development. Software testing has evolved into a continuous process that is integrated throughout the development lifecycle, mostly due to Agile and DevOps methodologies.

In order to identify and fix errors as soon as possible and save money and effort on future problem-solving, this approach strongly emphasizes early testing jobs including unit and integration testing. By arranging testing efforts in accordance with the likelihood and impact of likely errors, risk-based testing optimizes test coverage while utilizing minimal resources. Exploratory testing gives testers the opportunity to analyze program behavior in real time and uncover issues that haven't been discovered before, which helps them become more creative and adaptive. Test automation is crucial for optimizing coverage, reducing manual work, and speeding testing processes, especially for regression testing and repetitive tasks. Metrics and reporting, which provide information on testing progress, quality, and areas for improvement, serve as a guide for data-driven decision-making. Furthermore, by evaluating software from the perspective of the end user, user experience testing ensures that it is useable, accessible, and accepted by them. By employing these strategies, organizations may increase the effectiveness, efficiency, and reliability of their software testing processes, which will ultimately lead to the delivery of higher-quality products to customers.

One kind of software testing called functional testing confirms that every feature of the software program works as intended. Its main objective is to validate the software's inputs, outputs, and behavior against the functional specifications in order to verify the software's functioning. Assuring that the software fulfills the functional requirements provided by the stakeholders and correctly completes the intended tasks is the main goal of functional testing. It entails testing the software's many components, such as databases, integrations, user interfaces, and APIs, to make sure everything functions as it should.

Testing strategies are methodical techniques for finding and fixing flaws in software systems to guarantee their quality and dependability. Unit testing, integration testing, system testing, and acceptance testing are examples of common approaches. Unit testing checks that each software unit operates as intended when it is isolated, concentrating on specific parts or features. In order to make sure that combined components operate appropriately when integrated, integration testing looks at how various parts or modules interact with one another. System testing entails evaluating the overall functionality, efficiency, and conformance to requirements of the software system as a whole. Acceptance testing determines if the software satisfies business requirements and is prepared for deployment. It is usually carried out by end users or stakeholders.

In the course of the software testing lifecycle, additional techniques including usability testing, testing for performance, and regression testing are just as important as these fundamental procedures. Regression testing maintains program stability over time by ensuring that new code modifications do not negatively impact current functionality. Performance testing finds possible bottlenecks or limits by assessing the software's scalability, speed, and responsiveness under many circumstances. The goal of usability testing is to evaluate the software's intuitiveness and user-friendliness from the viewpoint of its intended users. When combined, these approaches offer a thorough framework for enhancing dependability, confirming software quality, and providing a satisfying user experience.

There are many software stack levels at which functional testing may be carried out, including: Unit testing is the process of independently testing separate software modules or units to ensure proper operation.

**Integration testing:** Examining how things work together System testing is the process of putting the complete system to the test in order to verify that its functionality and behavior match the functional requirements. Verifying the interactions between various software modules or components is the main goal of integration testing, a crucial stage in the software testing lifecycle. By addressing potential problems that may develop during the integration of individual components, this testing technique makes sure that combined pieces of code function together as intended. Integration testing seeks to find errors pertaining to data flow, communication, and functioning at the module borders by examining the interfaces and interactions of integrated components. This procedure is an essential step in verifying the overall coherence and dependability of the software program by assisting in making sure the integrated system operates as intended and satisfies the criteria.

**Acceptance Testing:** Examining if the program satisfies end users' wants and requirements by testing it from their point of view. Black-box, white-box, and gray-box testing are three functional testing methodologies that concentrate on various facets of the software's functioning. In order to provide thorough coverage, test cases are usually written using functional requirements, use cases, and user stories as a basis. They often include a variety of situations, inputs, and edge cases and to identify any mistakes. The completed program is assessed against predetermined objectives and business goals during the critical acceptance testing phase of the process of developing software to see if it satisfies the acceptance standards established by the stakeholders or end users. This kind of testing, which is usually carried out by the client or end users, is concerned with making sure the software serves the intended function and offers the required functionality from the user's point of view. Acceptance testing is the last stage of validation before software is released into production. It can take many different forms, such as contract acceptance testing, user acceptance testing (UAT), and alpha and beta testing. Verifying that the program meets user specifications and requirements and is prepared for release is its main objective.

### Common Types of Functional Testing Include:

**Smoke testing:** examining the application's fundamental features to make sure it is reliable and prepared for more testing. Testing to make sure that new additions or modifications don't adversely affect already-existing functionality is known as regression testing. A subset of fundamental tests are run at the initial stage of software testing, known as "smoke testing" or "build verification testing," to see if the software build is stable enough for additional, more thorough testing. Smoke testing's main goal is to rapidly determine whether an application's basic features are operating as intended following a fresh build or deployment. In order to make sure that the build is not really defective, this testing usually includes crucial areas of the program, such as initialization, core activities, and vital workflows.

**User Acceptance Testing (UAT):** end-user testing to confirm that the program satisfies their needs and expectations. GUI testing is the process of examining how responsive, intuitive, and visually consistent the graphical user interface. The last stage of software testing is called User Acceptance Testing (UAT), during which stakeholders or end users assess the program to make sure it satisfies their needs and expectations prior to going live. During this testing phase, the main goal is to confirm that the software meets user and business requirements and functions as intended in real-world circumstances.

**Application programming interface (API) testing**: is the process of testing APIs to make sure they meet requirements and provide the anticipated results. By comparing an application's functionality to specified criteria, functional testing helps to ensure that software is reliable and high-quality, which in turn increases customer happiness and product confidence. Application programming interface (API) testing is a subset of software testing that assesses the functionality, performance, dependability, and security of APIs. In order to verify that an API satisfies its requirements and produces the desired results, this testing technique entails making calls to the API and examining the answers. API testing confirms that the API operates as intended, handles a variety of inputs appropriately, and interfaces easily with other software elements or systems.

### Functional Testing

One kind of software testing called functional testing confirms that every feature of the software program works as intended. Its main objective is to validate the software's inputs, outputs, and behavior against the functional specifications in order to verify the software's functioning. Assuring that the software fulfills the functional requirements provided by the stakeholders and correctly completes the intended tasks is the main goal of functional testing. It entails testing the software's many components, such as databases, integrations, user interfaces, and APIs, to make sure everything functions as it should. There are many software stack levels at which functional testing may be carried out, including: Unit testing is the process of independently testing separate software modules or units to ensure proper operation.

**Integration testing**: Examining how things work together System testing is the process of putting the complete system to the test in order to verify that its functionality and behavior match the functional requirements. By looking at the interfaces and interactions of integrated components, integration testing looks for faults related to data flow, communication, and functionality at the module boundaries. This process helps to ensure that the integrated system works as planned and meets the requirements, which is a crucial step in confirming the overall coherence and reliability of the software program.

**Acceptance Testing:** Examining if the program satisfies end users' wants and requirements by testing it from their point of view. Black-box, white-box, and gray-box testing are three functional testing methodologies that concentrate on various facets of the software's functioning. In order to provide thorough coverage, test cases are usually written using functional requirements, use cases, and user stories as a basis. They often include a variety of situations, inputs, and edge cases. The final step of verification before software is put into production is acceptance testing. It can come in a variety of shapes and sizes, including alpha and beta testing, contract acceptance testing, and user acceptability testing (UAT). Its primary goal is to confirm that the software satisfies user needs and standards and is ready for distribution.

### Integration Testing

The goal of integration testing, a type of software testing, is to confirm how various software application modules, components, or subsystems interact and work together. In order to make sure that these units function together as planned, it seeks to identify any flaws and problems that could develop from their interactions. Validating interfaces and interactions between different software components and confirming that they communicate and operate as intended when integrated are the main objectives of integration testing. It assists in locating problems with integration, such as incompatibilities between modules, data inconsistencies, interface mistakes, and communication breakdowns.

**Integration testing can be performed at different levels of granularity, including**: Testing the interaction between separate modules or components to make sure they integrate properly and perform as intended when combined is known as component integration testing. System integration testing involves putting various subsystems or components through their paces to make sure they work well together and that the final product satisfies all criteria. Several methods, including incremental, bottom-up, and top-down integration testing, may be used to do integration testing.

**Top-Down Integration Testing:** This method involves testing the highest-level components or modules first, then integrating and testing lower-level modules over time. To mimic the behavior of unintegrated components, stubs or drivers can be employed.

**Bottom-Up Integration Testing:** This method integrates higher-level components progressively after testing the lowest-level modules or components. Test cases are created during integration testing using the system architecture, interface definitions, and integration standards as a guide. To make sure the integrated system operates appropriately in a variety of situations and settings, these test cases validate data flows, boundary conditions, and scenarios. Stubs, drivers and mocks are often used in integration testing to isolate certain components for testing or to mimic the behavior of not-yet-integrated components. By verifying the interaction and integration of various components, integration testing helps to ensure the interoperability, stability, and reliability of software systems. This lowers the likelihood of integration-related errors and guarantees the overall quality of the software.

### Test Cases

**Test Case for Excel Sheet Verification:**

Here in machine learning we are dealing with dataset which is in excel sheet format so if any test case we need means we need to check excel file. Later on classification will work on the respective columns of dataset. This method integrates higher-level components progressively after testing the lowest-level modules or components. Test cases are created during integration testing using the system architecture, interface definitions, and integration standards as a guide. To make sure the integrated system operates appropriately in a variety of situations and settings, these test cases validate data flows, boundary conditions, and scenario tubs. This method integrates higher-level components progressively after testing the lowest-level modules or components. Test cases are created during integration testing using the system architecture, interface definitions, and integration standards as a guide. To make sure the integrated system operates appropriately in a variety of situations and settings, these test cases validate data flows, boundary conditions, and scenarios Stubs.

System integration testing involves putting various subsystems or components through their paces to make sure they work well together and that the final product satisfies all criteria. Several methods, including incremental, bottom-up, and top-down integration testing, may be used to do integration testing. This method integrates higher-level components progressively after testing the lowest-level modules or components. Test cases are created during integration testing using the system architecture, interface definitions, and integration standards as a guide. Test cases are comprehensive and precise collections of variables or circumstances created to verify that a certain feature of a software program performs as intended. Every test case describes a series of actions, inputs, and anticipated results to confirm that a feature or capability operates as intended in a variety of circumstances. Test cases are crucial to structured testing because they provide thorough application coverage, assist find flaws or requirements deviations, and serve as a foundation for consistent and repeatable testing. Clearly stated test cases enable team members communicate with each other, aid in debugging, and preserve the dependability and quality of the program.

**Test Case 1:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case** | **Expecting Behavior** | **Exhibiting Behavior** | **Result** |
| 1 | Data Collection | Action Performed | Action Performed | Pass |
| 2 | Check for Null Values | Action Performed | Action Performed | Pass |
| 3 | Outliers Detected | Error | Action Performed | Fail |
| 4 | Checked for Missing Values after preprocessing | Error | Action Performed | Fail |
| 5 | Get Results | Action Performed | Action Performed | Pass |
| 6 | Predictions | Action Performed | Action Performed | Pass |

# CHAPTER 8

**RESULTS**

# RESULTS

Compared to conventional approaches, the T5 model's findings for text summarizing show a notable improvement in producing cohesive and succinct summaries. Because of its capacity to comprehend and process text in a text-to-text format, the T5 model is able to generate summaries that faithfully capture the main ideas of the source material while maintaining readability and flow. The resulting summaries successfully convey the main ideas and important details from the original content and are frequently more coherent and contextually relevant. This is especially true for complicated papers, where it may be difficult for classic extractive techniques to keep a logical flow or smoothly combine disparate kinds of information. A quantitative assessment of the summary outcomes, employing measures like ROUGE and BLEU scores, underscores the T5 model's efficacy even more. High accuracy in collecting significant material is shown by the ROUGE scores, which quantify the overlap of n-grams between the produced summary and reference summaries. The model's capacity to generate accurate and fluid summaries is further demonstrated by the BLEU score, which evaluates the quality of the generated summaries based on n-gram precision. These metrics highlight the model's ability to produce summaries that are both useful and uphold the high standards of quality and consistency associated with sophisticated summarizing systems.

User comments on the results of the summary validate the model's usefulness. Many users have noted that the T5 model's summaries are more relevant and readable than those created by more conventional extraction techniques, and they are very helpful for rapidly comprehending complicated materials. The user experience is further improved by the interactive streamlit interface, which makes summaries easy to create and examine. Overall, the findings show that the suggested approach significantly raises the quality of summaries, which makes it a useful tool for effective information retrieval and understanding.

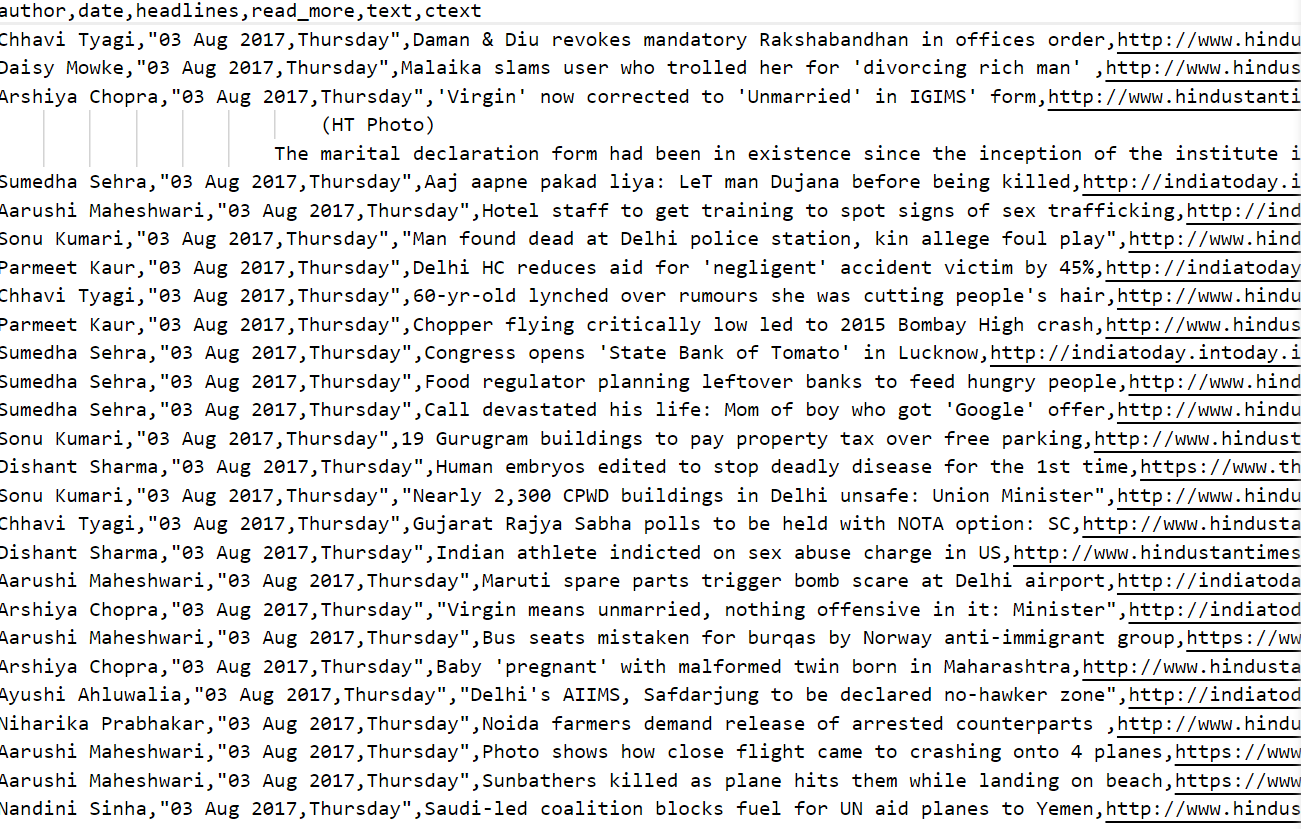
****

Fig.8.1 Dataset of Content

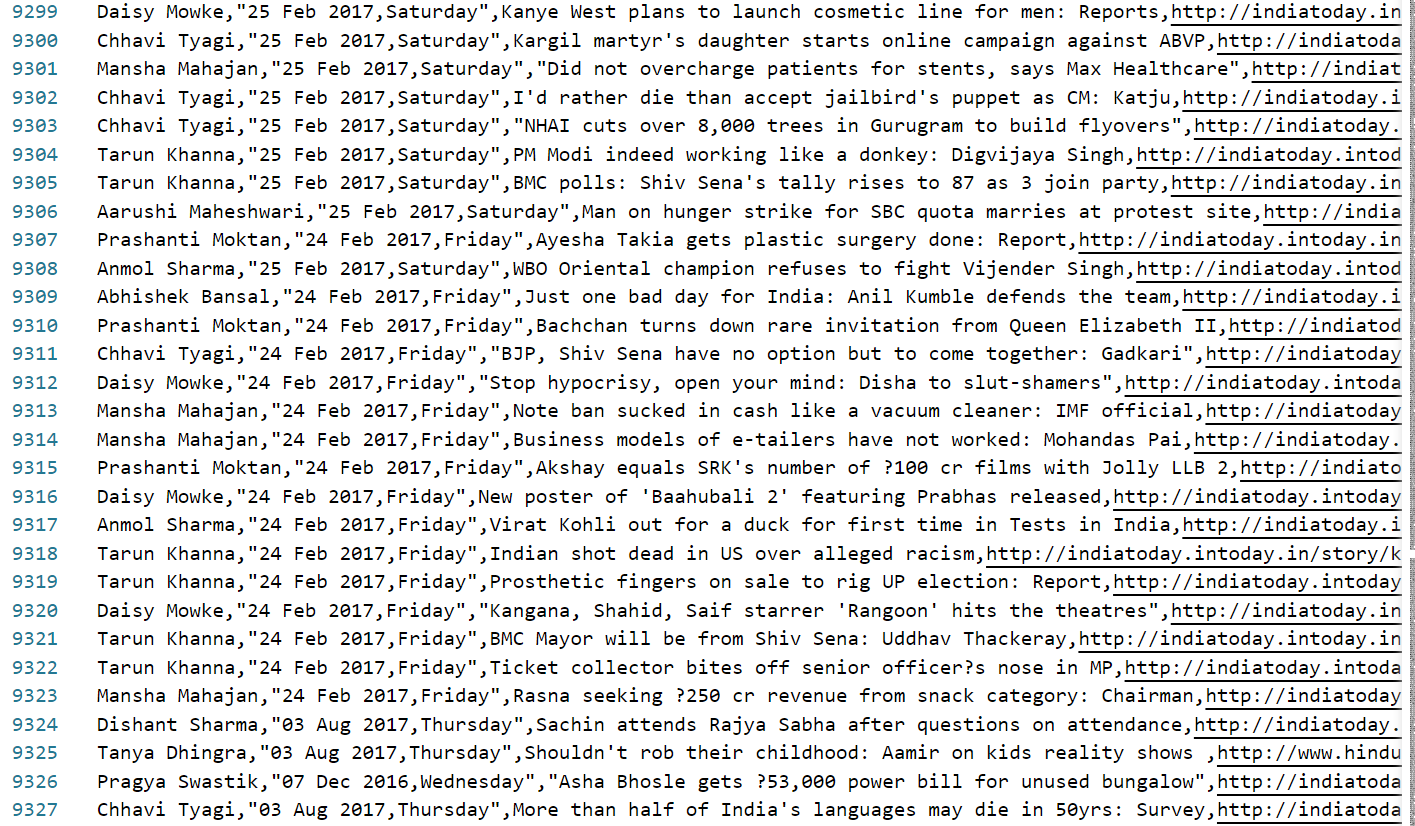


Fig.8.2 Dataset of Content

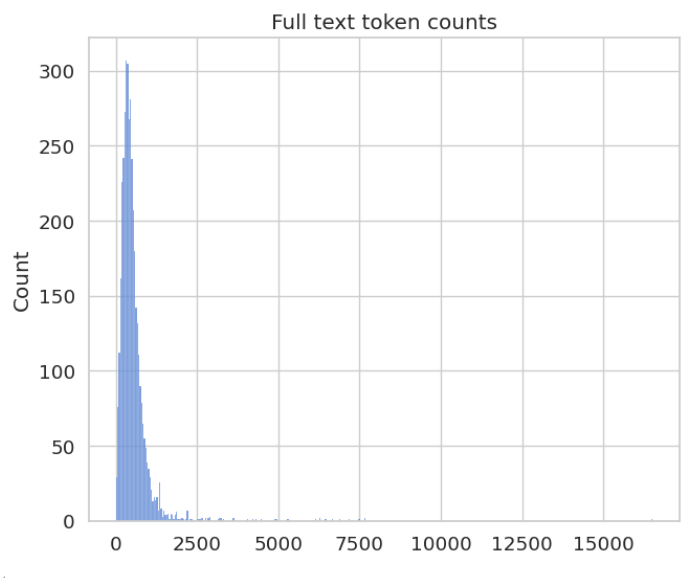
****

Fig.8.3 Full text token counts

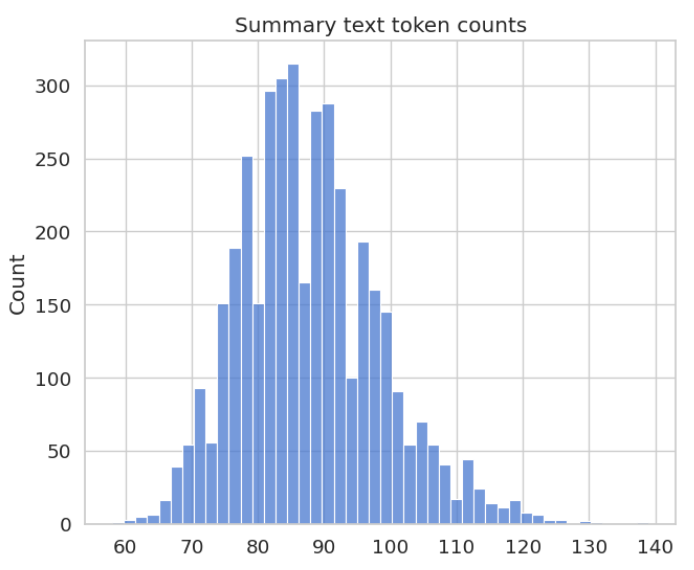


Fig.8.4 Summary text token counts

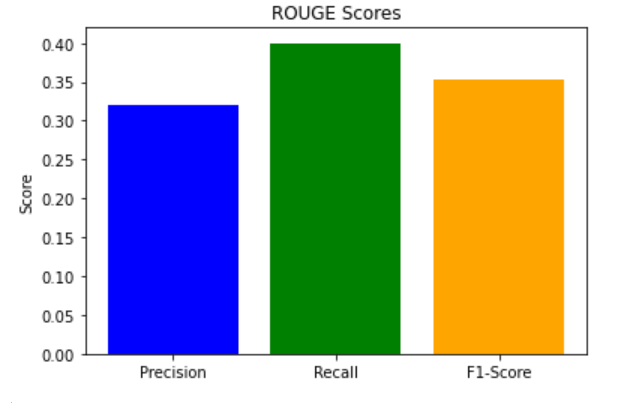
****

Fig.8.5 Rouge Scores

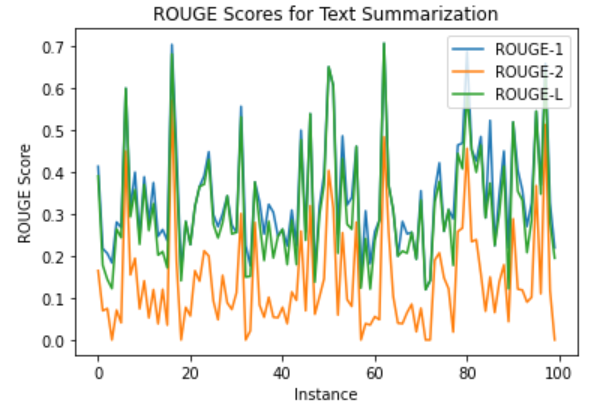


Fig.8.6 Rouge Scores for Summarization

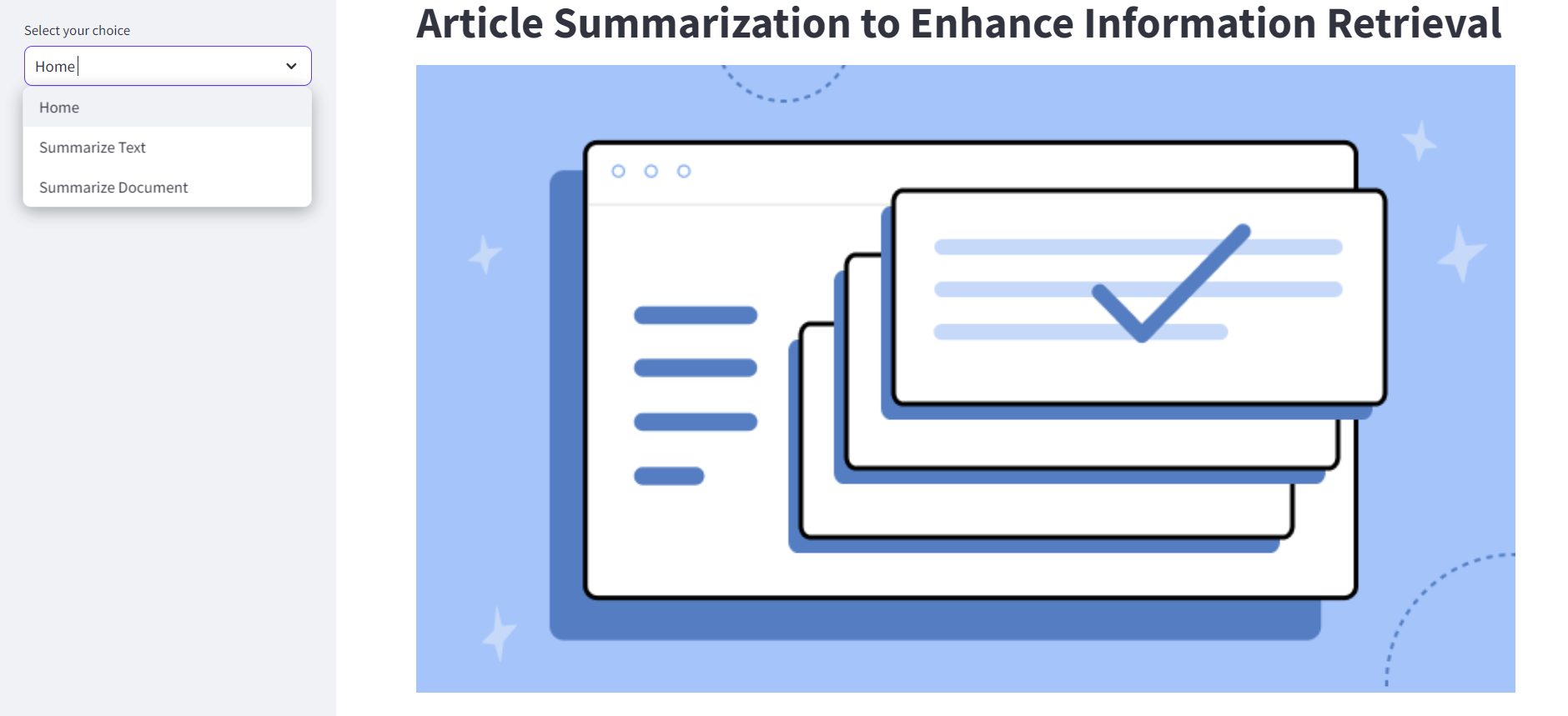


Fig.8.7 Home Page

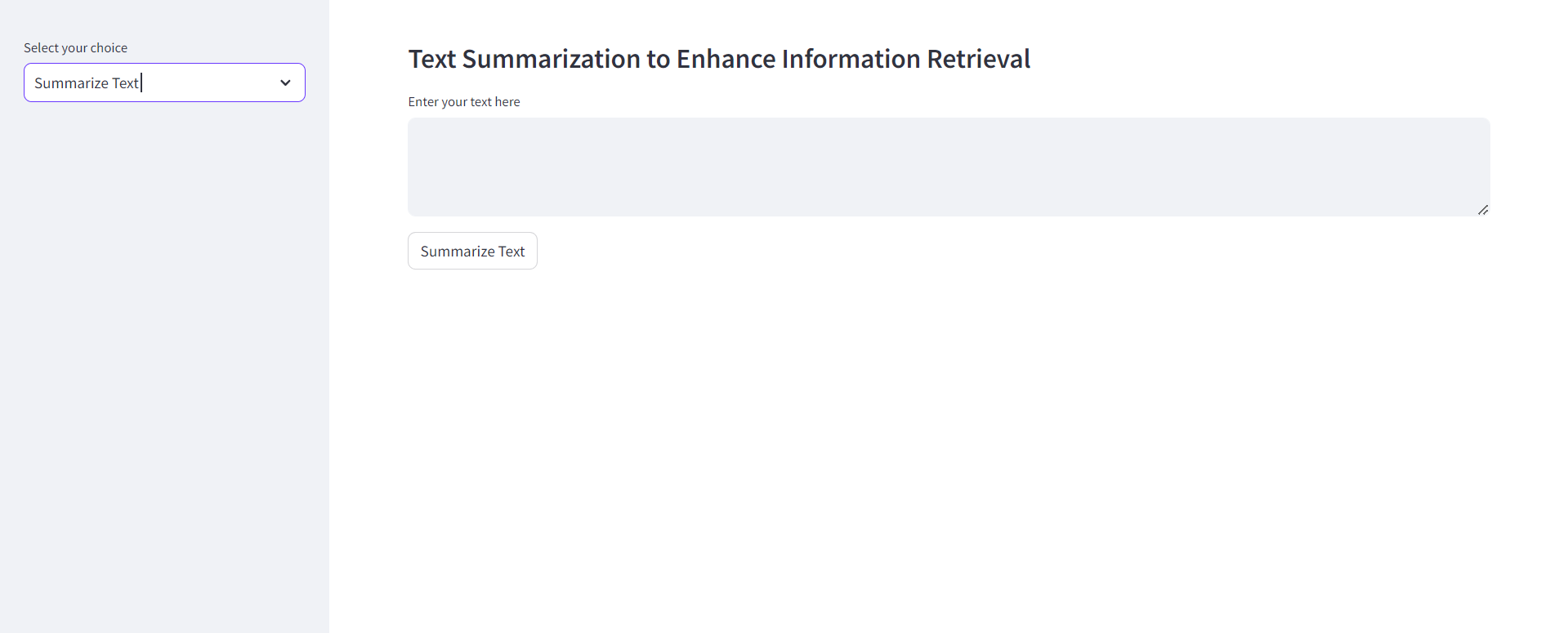
****

Fig.8.8 Predictions Page

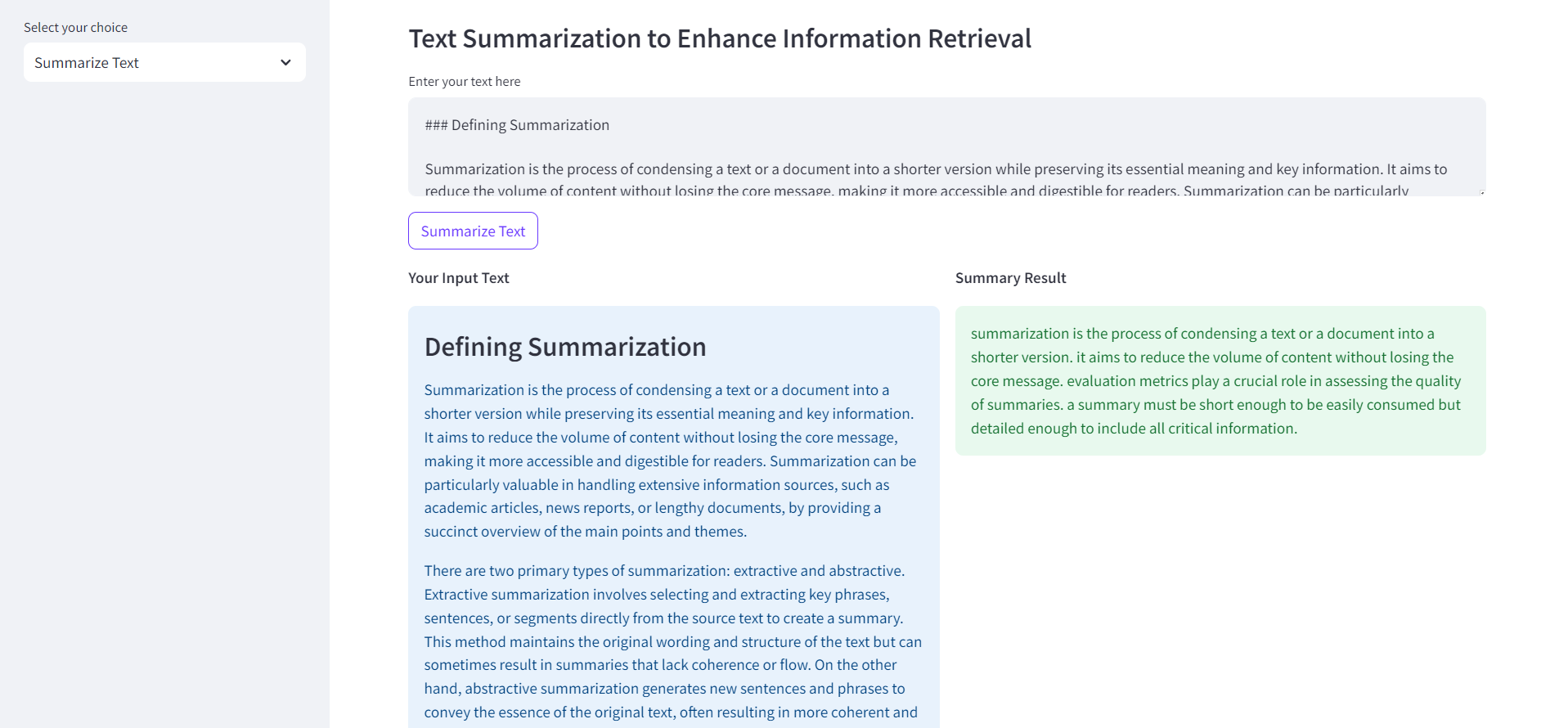


Fig.8.9 Text Summarization



Fig.8.10 Article Summarization

**CHAPTER 9**

**CONCLUSION**

**9.CONCLUSION**

In conclusion, by utilizing cutting-edge AI approaches, the suggested automated article summarizing system successfully overcomes the difficulties involved in processing enormous volumes of text. Through the integration of the T5 (Text-To-Text Transfer Transformer) paradigm with an intuitive streamlit interface, the system offers a reliable means of producing cohesive, high-quality summaries. Because of its transformer-based design, the T5 model is able to preserve readability and contextual relevance while encapsulating the spirit of the source material. By doing away with the drawbacks of intricate abstractive models and conventional extractive techniques, this method produces summaries that are both readable and educational. Users obtain accurate and succinct summaries because to the smooth preprocessing and summary processes, which greatly improves their capacity to absorb and comprehend large amounts of material.

Furthermore, the addition of an interactive user interface created using streamlit significantly improves the system by facilitating the usage of cutting-edge summarization technologies. Smooth interactions are facilitated by the interface, which offers choices for feedback and revision and makes it simple for users to submit information and obtain summaries. In addition to enhancing user experience, this approach facilitates ongoing refinement of the summary procedure through user feedback. All things considered, the system is a complete solution that blends cutting edge natural language processing methods with realistic, user-centered design, opening doors for improved decision-making and more effective information retrieval in a variety of fields.

**FUTURE ENHANCEMENT**

By adding cutting-edge features and methods, future improvements to the automated article summary system can concentrate on raising the caliber and adaptability of the summaries produced. The incorporation of multi-modal summarization capabilities is one possible improvement. Text summarization might be enhanced to provide richer, more contextually relevant summaries by integrating it with other types of data, such audio or pictures. For example, using visual data from papers or multimedia information might offer a more thorough comprehension of the subject matter, resulting in summaries that encompass both the visual and textual aspects of the content. Expanding the system's capacity to manage and process many forms of input data would be necessary to implement this multi-modal approach, which would increase the system's application and efficacy. The application of adaptive summarizing techniques which customize summaries according on user requirements and preferences is another area for future research.

This might entail improvements to the model design, including using distributed computing strategies or implementing transformer variations that are more efficient. Improved scalability would allow the system to handle expanding datasets and rising user demand, guaranteeing its continuous applicability and efficacy across a range of scenarios. By taking care of these issues, the system may develop to satisfy users' growing needs and offer even more benefits in the field of automated summarization. Through integration of input particular to each user and learning from unique usage patterns, the system may provide alternatives for customized summary. Users can tailor summaries to the system's needs by, for instance, designating the amount of information or focusing on certain areas of the content they are interested in. By analyzing user interactions and preferences, machine learning algorithms may be used to constantly improve the summarization process and produce outputs that are more relevant and tailored.

# BIBLIOGRAPHY

1. Day, Min-Yuh, and Chao-Yu Chen. "Artificial intelligence for automatic text summarization." *2018 IEEE International Conference on Information Reuse and Integration (IRI)*. IEEE, 2018.
2. Neto, Joel Larocca, Alex A. Freitas, and Celso AA Kaestner. "Automatic text summarization using a machine learning approach." *Advances in Artificial Intelligence: 16th Brazilian Symposium on Artificial Intelligence, SBIA 2002 Porto de Galinhas/Recife, Brazil, November 11–14, 2002 Proceedings 16*. Springer Berlin Heidelberg, 2002.
3. Sekaran, Kaushik, et al. "Design of optimal search engine using text summarization through artificial intelligence techniques." *TELKOMNIKA (Telecommunication Computing Electronics and Control)* 18.3 (2020): 1268-1274.
4. Nazari, Narges, and M. A. Mahdavi. "A survey on automatic text summarization." *Journal of AI and Data Mining* 7.1 (2019): 121-135.
5. Abualigah, Laith, et al. "Text summarization: a brief review." *Recent Advances in NLP: the case of Arabic language* (2020): 1-15.
6. Yousefi-Azar, Mahmood, and Len Hamey. "Text summarization using unsupervised deep learning." *Expert Systems with Applications* 68 (2017): 93-105.
7. Gong, Yihong, and Xin Liu. "Generic text summarization using relevance measure and latent semantic analysis." *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. 2001.
8. Batra, Pooja, et al. "A review: Abstractive text summarization techniques using NLP." *2020 International Conference on Advances in Computing, Communication & Materials (ICACCM)*. IEEE, 2020.
9. Kumar, Yogesh, Komalpreet Kaur, and Sukhpreet Kaur. "Study of automatic text summarization approaches in different languages." *Artificial Intelligence Review* 54.8 (2021): 5897-5929.
10. Cheng, Ruijia, et al. "Mapping the design space of human-ai interaction in text summarization." *arXiv preprint arXiv:2206.14863* (2022).
11. Yadav, Arun Kumar, et al. "Extractive text summarization using deep learning approach." *International Journal of Information Technology* 14.5 (2022): 2407-2415.
12. Merine, Regina, and Saptarshi Purkayastha. "Risks and benefits of AI-generated text summarization for expert level content in graduate health informatics." *2022 IEEE 10th International Conference on Healthcare Informatics (ICHI)*. IEEE, 2022.
13. Subbalakshmi, Chatti, Piyush Kumar Pareek, and M. V. Narayana. "A Gravitational Search Algorithm Study on Text Summarization Using NLP." *International Conference on Artificial Intelligence and Data Science*. Cham: Springer Nature Switzerland, 2021.
14. Belkebir, Riadh, and Ahmed Guessoum. "A supervised approach to arabic text summarization using adaboost." *New Contributions in Information Systems and Technologies: Volume 1*. Springer International Publishing, 2015.
15. Tas, Oguzhan, and Farzad Kiyani. "A survey automatic text summarization." *PressAcademia Procedia* 5.1 (2007): 205-213.
16. Dar, Zaema, et al. "Advanced Generative AI Methods for Academic Text Summarization." *2024 IEEE 3rd International Conference on Computing and Machine Intelligence (ICMI)*. IEEE, 2024.
17. El-Kassas, Wafaa S., et al. "Automatic text summarization: A comprehensive survey." *Expert systems with applications* 165 (2021): 113679.
18. Mridha, Muhammad Firoz, et al. "A survey of automatic text summarization: Progress, process and challenges." *IEEE Access* 9 (2021): 156043-156070.
19. Lai, Vivian, et al. "An exploration of post-editing effectiveness in text summarization." *arXiv preprint arXiv:2206.06383* (2022).
20. Abu Nada, Abdullah M., et al. "Arabic text summarization using arabert model using extractive text summarization approach." (2020).