THE NATIONAL COLLEGE

(AUTONOMOUS)

BASAVANAGUDI, BENGALURU -560004

॥ ಶ್ರದ್ಧಾಹಿ ಪರಮಾ ಗತಿಃ ॥

INTERNSHIP REPORT ON

ARTIFICIAL INTELLIGENCE AND MINI PROJECT AT PANTECH E LEARNING Pvt.Ltd

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF UNDER GRADUATE DEGREE

BACHOLER OF COMPUTER APPLICATION

BY

BHARATH T - U18EZ21S0002
FAKRUDDIN - U18EZ21S0134
RANJITH B - U18EZ21S0021
SURAARCHITH K V - U18EZ21S0063

UNDER THE GUIDENCE OF

Prof. Alakananda K P
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE

YEAR 2023 – 2024

THE NATIONAL COLLEGE

(AUTONOMOUS)

BASAVANAGUDI, BENGALURU - 560004

॥ ಶ್ರದ್ಧಾಹಿ ಪರಮಾ ಗತಿಃ ॥

Certificate

This is to certify that Mr. **BHARATH T [U18EZ21S0002], FAKRUDDIN [U18EZ21S0134], RANJITH B [U18EZ21S0021], SURAARCHITH K V [U18EZ21S0063]** are the student of our college. They have prepared an internship report entitled **"Artificial Intelligence and Mini Project"** at **Pantech e Learning Pvt.Ltd** from **January 2024 to March 2024** towards the partial fulfilment for the award of Sixth semester **Bachelors of Computer Applications** of **The National College, Basavanagudi, Bengaluru - 560004**, during the **year 2023 – 2024**

Prof. Ravi Hegde
Head of the Department
Department of Computer Science

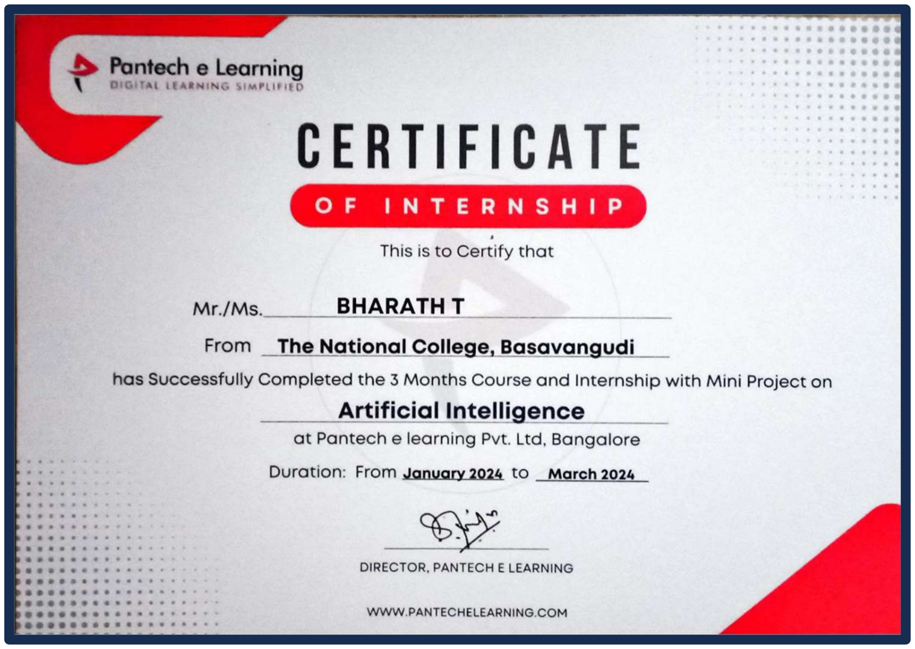
Prof. Alakananda K P
Faculty Co-ordinator
Department of Computer Science

Examiners:

1. ____________________

2. ____________________

CERTIFICATE ISSUED BY THE COMPANY



Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

CERTIFICATE
OF INTERNSHIP

This is to Certify that

Mr./Ms. BHARATH T

From The National College, Basavangudi

has Successfully Completed the 3 Months Course and Internship with Mini Project on

Artificial Intelligence

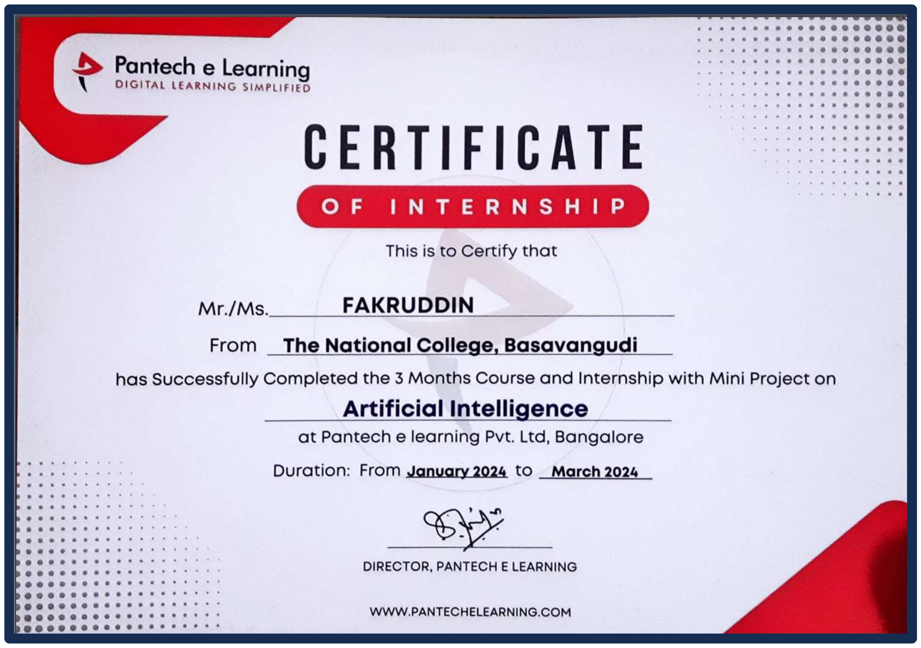
at Pantech e learning Pvt. Ltd, Bangalore

Duration: From January 2024 to March 2024

[illegible]

DIRECTOR, PANTECH E LEARNING

WWW.PANTECHELEARNING.COM



Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

CERTIFICATE
OF INTERNSHIP

This is to Certify that

Mr./Ms. FAKRUDDIN

From The National College, Basavangudi

has Successfully Completed the 3 Months Course and Internship with Mini Project on

Artificial Intelligence

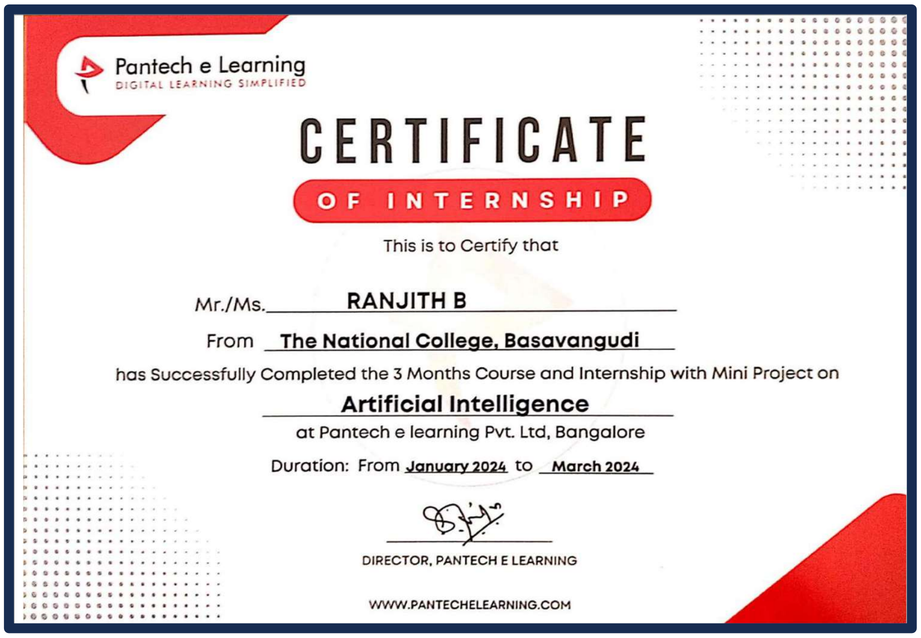
at Pantech e learning Pvt. Ltd, Bangalore

Duration: From January 2024 to March 2024

[illegible]

DIRECTOR, PANTECH E LEARNING

WWW.PANTECHELEARNING.COM



Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

CERTIFICATE
OF INTERNSHIP

This is to Certify that

Mr./Ms. RANJITH B

From The National College, Basavangudi

has Successfully Completed the 3 Months Course and Internship with Mini Project on

Artificial Intelligence

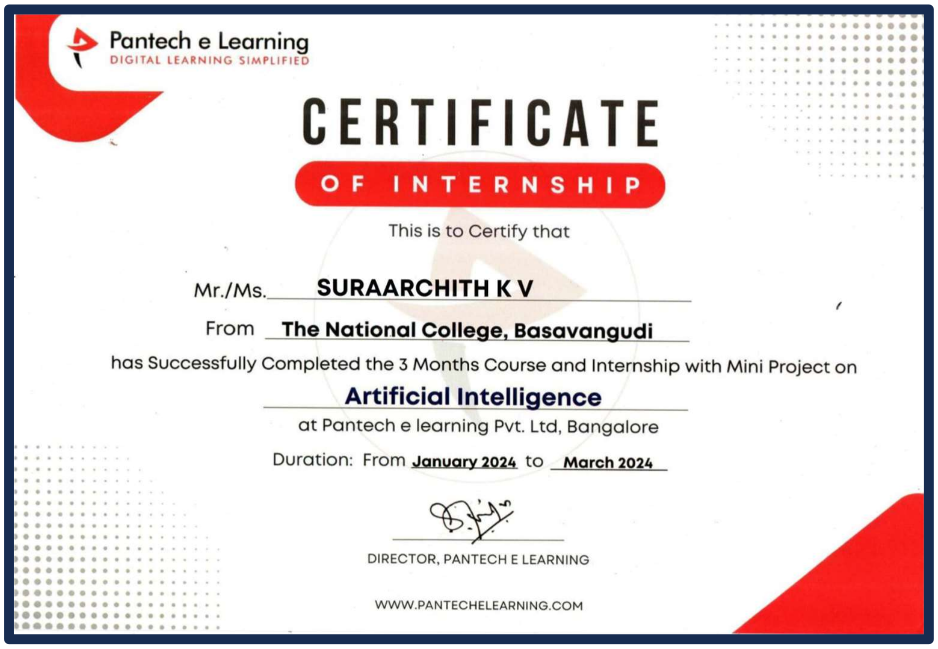
at Pantech e learning Pvt. Ltd, Bangalore

Duration: From January 2024 to March 2024

[illegible]

DIRECTOR, PANTECH E LEARNING

WWW.PANTECHELEARNING.COM



Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

CERTIFICATE
OF INTERNSHIP

This is to Certify that

Mr./Ms. SURAARCHITH K V

From The National College, Basavangudi

has Successfully Completed the 3 Months Course and Internship with Mini Project on

Artificial Intelligence

at Pantech e learning Pvt. Ltd, Bangalore

Duration: From January 2024 to March 2024

[illegible]

DIRECTOR, PANTECH E LEARNING

WWW.PANTECHELEARNING.COM

STUDENT DECLARATION

We **BHARATH T [U18EZ21S0002], FAKRUDDIN [U18EZ21S0134], RANJITH B [U18EZ21S0021], SURAARCHITH K V [U18EZ21S0063]** hereby declare that this internship report entitled **“Artificial Intelligence and Mini Project”** at **Pantech e Learning Pvt.Ltd** from **January 2024 to March 2024** under the supervision and guidance of **Prof. Alakananda K P** of Computer Science Department, The National College, Basavanagudi, Bengaluru – 04.

Date:
Place: Bengaluru

Signature
1.
2.
3.
4.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to Pantech e Learning for providing me the invaluable opportunity to complete my internship on Advanced Python Programming and Mini Project. This experience has been immensely enriching and has significantly contributed to my professional growth.

I extend my sincere thanks to my mentor and the entire team at Pantech e Learning for their constant guidance, support, and encouragement throughout the internship. Their insights and feedback were instrumental in enhancing my understanding and skills in Artificial Intelligence.

I am also thankful to my faculty coordinator Prof. Ravi Hegde, Prof. Alakananda K P and Prof. Sangeetha B K for their unwavering support and guidance throughout this internship. Their advice and encouragement have been crucial in navigating the challenges and making the most of this learning opportunity.

Additionally, I am grateful for the collaborative and positive environment provided by the company, which facilitated learning and professional development. This internship has equipped me with practical skills and knowledge that I am confident will be beneficial in my future endeavours. Thank you once again to everyone at Pantech eLearning and to Prof. Alakananda K P for making this internship a truly valuable and memorable experience.

TABLE OF CONTENTS

SI.NO	INDEX	Pg.No
1	Introduction	1 – 3
2	Company Overview	4
3	Basic Concepts of Artificial Intelligence	5 – 12
4	Machine Learning Lifecycle	13 – 18
5	Basic Concepts of Python	19 – 23
6	Python Data Structures	24 – 41
7	Modules And Packages of Python	42 – 44
8	Natural Language Processing (NLP)	45 – 47
9	Neural Network	48 – 50
10	Bibliography	51
11	Project Work	52 - 57

Introduction

Artificial Intelligence (AI) has emerged as one of the most transformative technologies of the 21st century. It is reshaping industries, enhancing human capabilities, and redefining the way we live and work. This internship report provides a comprehensive overview of AI, highlighting its key features, applications, and potential impact on various sectors. Additionally, it outlines the objectives, experiences, and learnings from my internship in the field of AI.

Overview of Artificial Intelligence

Artificial Intelligence refers to the simulation of human intelligence in machines that are programmed to think and learn like humans. The concept of AI encompasses a broad range of technologies and methodologies, including machineLearning, natural language processing, computer vision, and robotics. AI systems can perform tasks that typically require human intelligence, such as recognizing speech, making decisions, solving problems, and understanding natural language.

Key Features of AI

1. **MachineLearning**: At the core of AI, machine learning involves algorithms that enable systems to learn from and make predictions or decisions based on data. It is divided into supervised learning, unsupervised learning, and reinforcement learning.
2. **Natural Language Processing (NLP)**: NLP allows machines to understand, interpret, and respond to human language. It is used in applications such as chatbots, virtual assistants, and language translation services.
3. **Computer Vision**: This feature enables machines to interpret and make decisions based on visual input from the world. Applications include facial recognition, object detection, and autonomous driving.
4. **Robotics**: AI-powered robots can perform complex tasks in manufacturing, healthcare, and other industries, enhancing efficiency and precision.
5. **Expert Systems**: These are AI programs that mimic the decision-making abilities of a human expert. They are used in fields such as medical diagnosis and financial services.

Applications of AI

AI is being integrated into various industries, revolutionizing processes and creating new opportunities. In healthcare, AI is used for predictive analytics, personalized medicine, and diagnostic imaging. In finance, AI enhances fraud detection, algorithmic trading, and customer service. The automotive industry leverages AI for autonomous vehicles and smart traffic management. Additionally, AI-driven innovations are evident in entertainment, education, agriculture, and more.

Objectives of AI Internship with Mini Project at Pantech eLearning

The internship at Pantech eLearning was designed to provide hands-on experience in the field of Artificial Intelligence (AI), focusing on both theoretical knowledge and practical applications. The primary objective was to enhance understanding and proficiency in AI technologies through a mini project, ensuring a comprehensive learning experience.

Objectives

1. Comprehensive Understanding of AI Concepts
- Goal: To gain a solid foundation in the fundamental concepts of AI, including machine learning, deep learning, natural language processing (NLP), and computer vision.
- Activities: Attending lectures, participating in workshops, and studying relevant literature and online resources.

2. Skill Development in AI Tools and Technologies
- Goal: To develop proficiency in AI programming languages and tools such as Python, TensorFlow, Keras, and PyTorch.
- Activities: Engaging in coding exercises, completing tutorials, and practicing with real-world datasets.

3. Practical Application through a Mini Project
- Goal: To apply theoretical knowledge to a practical AI project, enhancing problem-solving and project management skills.
- Activities: Selecting a mini project relevant to AI, such as image classification, sentiment analysis, or predictive modeling, and completing all phases from data collection to model deployment.

4. Data Handling and Preprocessing
- Goal: To understand the importance of data in AI and develop skills in data preprocessing techniques, including cleaning, normalization, and feature extraction.
- Activities: Working with various datasets, performing data preprocessing tasks, and utilizing libraries like Pandas and NumPy.

5. Model Building and Evaluation
- Goal: To learn the process of building, training, and evaluating machine learning and deep learning models.
- Activities: Implementing models using frameworks like TensorFlow and Keras, tuning hyperparameters, and evaluating model performance using metrics such as accuracy, precision, and recall.

6. Exploring AI Applications
- Goal: To explore various applications of AI across different industries and understand the impact of AI technologies.
- Activities: Studying case studies, analyzing AI applications in fields such as healthcare, finance, and automotive, and discussing future trends and ethical considerations.

7. Enhancing Communication and Collaboration Skills
- Goal: To develop effective communication and collaboration skills essential for working in AI teams and projects.
- Activities: Participating in group discussions, presenting project progress and results, and collaborating with peers and mentors.

8.Gaining Industry Exposure
- Goal: To understand the practical implementation of AI in the industry and stay updated with the latest advancements and trends.
- Activities: Attending webinars, networking with AI professionals, and participating in industry-relevant events and seminars.

Conclusion

The internship at Pantech eLearning aimed to provide a comprehensive and immersive experience in the field of AI. By focusing on both theoretical knowledge and practical applications through a mini project, the objectives were designed to ensure a well-rounded learning experience. The skills and knowledge gained during this internship will serve as a strong foundation for future endeavors in AI, enabling the intern to contribute effectively to this rapidly evolving field.

Company Overview

Pantech e Learning is a leading provider of innovative educational solutions, dedicated to empowering learners and educators worldwide. With a passion for making learning accessible, engaging, and effective, our team of experts has crafted a range of products and services designed to meet the evolving needs of the education sector.

Our Mission

At Pantech e Learning, our mission is to harness the power of technology to create a more inclusive, personalized, and impactful learning experience. We strive to bridge the gap between traditional teaching methods and modern digital tools, providing educators with the resources they need to inspire and motivate their students.

Our Vision

Our vision is to become a trusted partner for educational institutions, educators, and learners, providing them with cutting-edge solutions that foster academic excellence, creativity, and critical thinking. We aim to create a global community of learners who are equipped with the skills, knowledge, and confidence to succeed in an ever-changing world.

Our Values

- Innovation: We believe in harnessing the latest technologies and pedagogical approaches to create innovative learning solutions.
- Accessibility: We are committed to making high-quality education accessible to all, regardless of geographical or socio-economic barriers.
- Collaboration: We foster partnerships with educators, institutions, and industry experts to create a collective impact on education.
- Excellence: We strive for excellence in everything we do, from product development to customer support.

Our Products and Services

Pantech e Learning offers a range of products and services designed to support educators and learners, including:

- Interactive digital content and resources
- Learning management systems and platforms
- Professional development opportunities for educators
- Customized educational solutions for institutions and organizations

By combining our expertise in education, technology, and innovation, we are dedicated to making a positive impact on the lives of learners and educators worldwide.

Basic Concepts of Artificial Intelligence

Introduction and definition of AI and ML:

Artificial intelligence (AI) is revolutionizing the way we interact with technology and transforming various industries. At its core, artificial intelligence involves the development of computer systems that can perform tasks typically requiring human intelligence. This includes problem-solving, decision-making, language understanding, and even visual perception. To grasp the artificial intelligence definition, it's essential to understand that AI encompasses a range of technologies, from machine learning and neural networks to natural language processing and robotics.

By exploring the artificial intelligence meaning, we can appreciate its profound impact on our daily lives and its potential to drive innovation in fields such as healthcare, finance, and transportation. Whether it's through smart assistants, automated customer service, or advanced data analysis, artificial intelligence is paving the way for a smarter and more efficient future.

Machine learning (ML) is a type of artificial intelligence (AI) that allows computers to learn without being explicitly programmed. This article explores the concept of machine learning, providing various definitions and discussing its applications. The article also dives into different classifications of machine learning tasks, giving you a comprehensive understanding of this powerful technology.

What Is Artificial Intelligence?

Artificial Intelligence Definition – Artificial intelligence, commonly known as AI, is a branch of computer science focused on creating systems capable of performing tasks that normally require human intelligence. The artificial intelligence definition involves the development of algorithms and models that enable machines to learn from data, recognize patterns, and make decisions. This field encompasses various subfields, including machine learning, where systems improve their performance through experience, and natural language processing, which allows machines to understand and generate human language.

What is Machine Learning?

Machine learning (ML) is a type of Artificial Intelligence (AI) that allows computers to learn without being explicitly programmed. It involves feeding data into algorithms that can then identify patterns and make predictions on new data. Machine learning is used in a wide variety of applications, including image and speech recognition, natural language processing, and recommender systems.

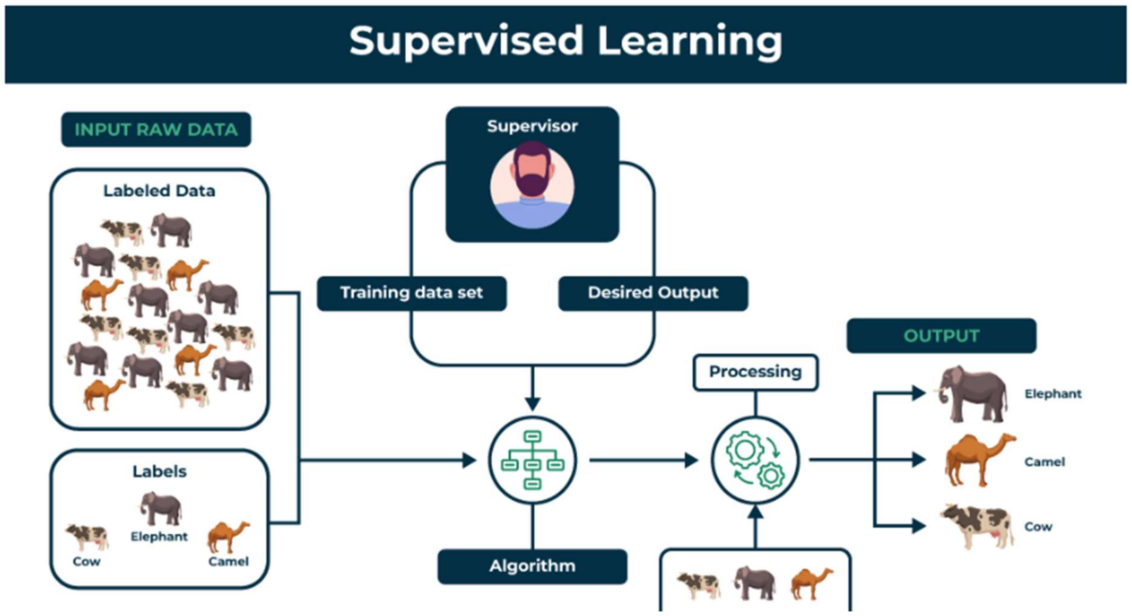
Types of Machine Learning:

There are several types of machine learning, each with special characteristics and applications. Some of the main types of machine learning algorithms are as follows:

- ➢ Supervised Machine Learning
- ➢ Unsupervised Machine Learning
- ➢ Reinforcement Learning

1. Supervised Machine Learning: -

Supervised learning is defined as when a model gets trained on a **"Labelled Dataset"**. Labelled datasets have both input and output parameters. In **Supervised Learning** algorithms learn to map points between inputs and correct outputs. It has both training and validation datasets labelled.



Let's understand it with the help of an example.

Example: Consider a scenario where you have to build an image classifier to differentiate between cats and dogs. If you feed the datasets of dogs and cats labelled images to the algorithm, the machine will learn to classify between a dog or a cat from these labelled images. When we input new dog or cat images that it has never seen before, it will use the learned algorithms and predict whether it is a dog or a cat. This is how **supervised learning** works, and this is particularly an image classification.

There are two main categories of supervised learning that are mentioned below:

- Classification
- Regression

Classification:

Classification deals with predicting **categorical** target variables, which represent discrete classes or labels. For instance, classifying emails as spam or not spam, or predicting whether a patient has a high risk of heart disease. Classification algorithms learn to map the input features to one of the predefined classes.

Here are some classification algorithms:

- Logistic Regression
- Support Vector Machine
- Random Forest
- Decision Tree
- K-Nearest Neighbours (KNN)
- Naïve Bayes

Regression:

Regression, on the other hand, deals with predicting **continuous** target variables, which represent numerical values. For example, predicting the price of a house based on its size, location, and amenities, or forecasting the sales of a product. Regression algorithms learn to map the input features to a continuous numerical value.

Here are some regression algorithms:

- Linear Regression
- Polynomial Regression
- Ridge Regression
- Lasso Regression
- Decision Tree
- Random Forest

Advantages of Supervised Machine Learning:

- **Supervised Learning** models can have high accuracy as they are trained on **labelled data**.
- The process of decision-making in supervised learning models is often interpretable.
- It can often be used in pre-trained models which saves time and resources when developing new models from scratch.

Disadvantages of Supervised Machine Learning:

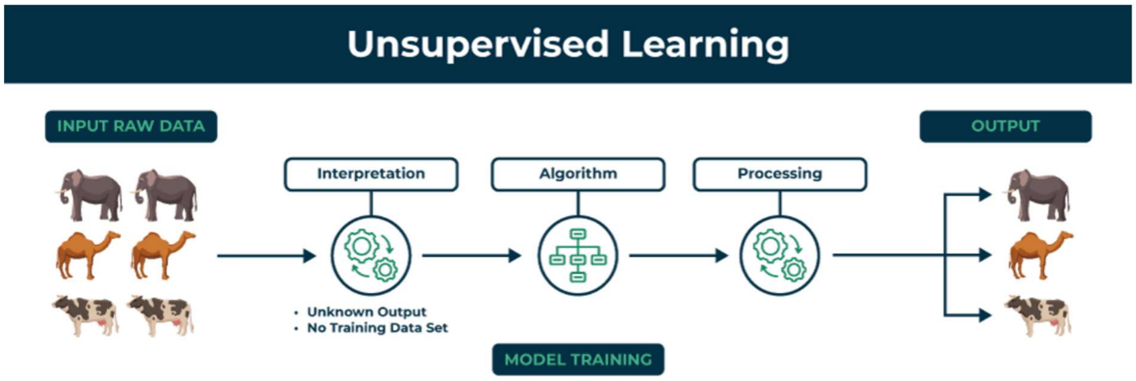
- It has limitations in knowing patterns and may struggle with unseen or unexpected patterns that are not present in the training data.
- It can be time-consuming and costly as it relies on **labelled** data only.
- It may lead to poor generalizations based on new data.

Applications of Supervised Learning:

- **Image classification**: Identify objects, faces, and other features in images.
- **Natural language processing:** Extract information from text, such as sentiment, entities, and relationships.
- **Speech recognition**: Convert spoken language into text.
- **Recommendation systems**: Make personalized recommendations to users.
- **Predictive analytics**: Predict outcomes, such as sales, customer churn, and stock prices.
- **Medical diagnosis**: Detect diseases and other medical conditions.
- **Fraud detection**: Identify fraudulent transactions.

2. Unsupervised Machine Learning: -

Unsupervised learning is a type of machine learning technique in which an algorithm discovers patterns and relationships using unlabelled data. Unlike supervised learning, unsupervised learning doesn't involve providing the algorithm with labelled target outputs. The primary goal of Unsupervised learning is often to discover hidden patterns, similarities, or clusters within the data, which can then be used for various purposes, such as data exploration, visualization, dimensionality reduction, and more.



Let's understand it with the help of an example.

Example: Consider that you have a dataset that contains information about the purchases you made from the shop. Through clustering, the algorithm can group the same purchasing behaviour among you and other customers, which reveals potential customers without predefined labels. This type of information can help businesses get target customers as well as identify outliers.

There are two main categories of unsupervised learning that are mentioned below:

- Clustering
- Association

Clustering:

Clustering is the process of grouping data points into clusters based on their similarity. This technique is useful for identifying patterns and relationships in data without the need for labelled examples.

Here are some clustering algorithms:

- K-Means Clustering Algorithm
- Mean-Shift Algorithm
- DBSCAN Algorithm
- Principal Component Analysis
- Independent component Analysis

Association:

Association rule learning is a technique for discovering relationships between items in a dataset. It identifies rules that indicate the presence of one item implies the presence of another item with a specific probability.

Here are some association rule learning algorithms:

- Apriori Algorithm
- Eclat
- FP-growth Algorithm

Advantages of Unsupervised Machine Learning:

- It helps to discover hidden patterns and various relationships between the data.
- Used for tasks such as **customer segmentation, anomaly detection,** and **data exploration**.
- It does not require labelled data and reduces the effort of data labelling.

Disadvantages of Unsupervised Machine Learning:

- Without using labels, it may be difficult to predict the quality of the model's output.
- Cluster Interpretability may not be clear and may not have meaningful interpretations.
- It has techniques such as autoencoders and dimensionality reduction that can be used to extract meaningful features from raw data.

Applications of Unsupervised Learning:

Here are some common applications of unsupervised learning:

- **Clustering**: Group similar data points into clusters.
- **Anomaly detection**: Identify outliers or anomalies in data.
- **Dimensionality reduction**: Reduce the dimensionality of data while preserving its essential information.

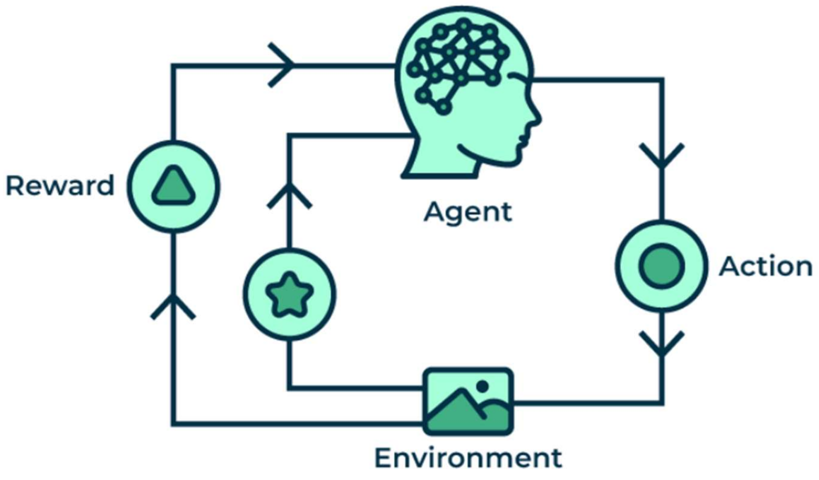
- **Recommendation systems**: Suggest products, movies, or content to users based on their historical behavior or preferences.
- **Topic modelling**: Discover latent topics within a collection of documents.
- **Density estimation**: Estimate the probability density function of data.
- **Image and video compression**: Reduce the amount of storage required for multimedia content.
- **Data preprocessing**: Help with data preprocessing tasks such as data cleaning, imputation of missing values, and data scaling.
- **Market basket analysis**: Discover associations between products.
- **Image segmentation**: Segment images into meaningful regions.
- **Customer behavior analysis**: Uncover patterns and insights for better marketing and product recommendations.
- **Content recommendation**: Classify and tag content to make it easier to recommend similar items to users.
- **Exploratory data analysis (EDA)**: Explore data and gain insights before defining specific tasks.

3. Reinforcement Machine Learning: -

Reinforcement machine learning algorithm is a learning method that interacts with the environment by producing actions and discovering errors. **Trial, error, and delay** are the most relevant characteristics of reinforcement learning. In this technique, the model keeps on increasing its performance using Reward Feedback to learn the behavior or pattern. These algorithms are specific to a particular problem e.g. Google Self Driving car, AlphaGo where a bot competes with humans and even itself to get better and better performers in Go Game. Each time we feed in data, they learn and add the data to their knowledge which is training data. So, the more it learns the better it gets trained and hence experienced.

Here are some of most common reinforcement learning algorithms:

- **Q-learning:** Q-learning is a model-free RL algorithm that learns a Q-function, which maps states to actions. The Q-function estimates the expected reward of taking a particular action in a given state.
- **SARSA (State-Action-Reward-State-Action):** SARSA is another model-free RL algorithm that learns a Q-function. However, unlike Q-learning, SARSA updates the Q-function for the action that was actually taken, rather than the optimal action.
- **Deep Q-learning:** Deep Q-learning is a combination of Q-learning and deep learning. Deep Q-learning uses a neural network to represent the Q-function, which allows it to learn complex relationships between states and actions.



Reinforcement Machine Learning

Let's understand it with the help of examples.

Example: Consider that you are training an AI agent to play a game like chess. The agent explores different moves and receives positive or negative feedback based on the outcome. Reinforcement Learning also finds applications in which they learn to perform tasks by interacting with their surroundings.

Types of Reinforcement Machine Learning

There are two main types of reinforcement learning:

1. **Positive reinforcement**
- Rewards the agent for taking a desired action.
- Encourages the agent to repeat the behavior.
- Examples: Giving a treat to a dog for sitting, providing a point in a game for a correct answer.

2. **Negative reinforcement**
- Removes an undesirable stimulus to encourage a desired behavior.
- Discourages the agent from repeating the behavior.
- Examples: Turning off a loud buzzer when a lever is pressed, avoiding a penalty by completing a task.

Advantages of Reinforcement Machine Learning:

- It has autonomous decision-making that is well-suited for tasks and that can learn to make a sequence of decisions, like robotics and game-playing.
- This technique is preferred to achieve long-term results that are very difficult to achieve.
- It is used to solve a complex problem that cannot be solved by conventional techniques.

Disadvantages of Reinforcement Machine Learning:

- Training Reinforcement Learning agents can be computationally expensive and time-consuming.
- Reinforcement learning is not preferable to solving simple problems.
- It needs a lot of data and a lot of computation, which makes it impractical and costly.

Applications of Reinforcement Machine Learning:

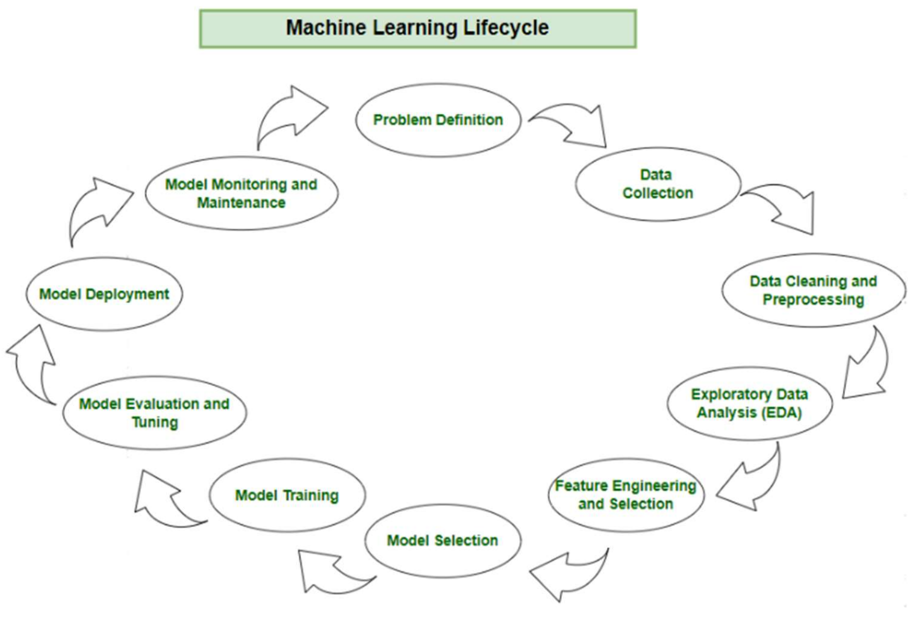
Here are some applications of reinforcement learning:

- **Game Playing**: RL can teach agents to play games, even complex ones.
- **Robotics**: RL can teach robots to perform tasks autonomously.
- **Autonomous Vehicles**: RL can help self-driving cars navigate and make decisions.
- **Recommendation Systems**: RL can enhance recommendation algorithms by learning user preferences.
- **Healthcare**: RL can be used to optimize treatment plans and drug discovery.
- **Natural Language Processing (NLP)**: RL can be used in dialogue systems and chatbots.
- **Finance and Trading**: RL can be used for algorithmic trading.
- **Supply Chain and Inventory Management**: RL can be used to optimize supply chain operations.
- **Energy Management**: RL can be used to optimize energy consumption.
- **Game AI**: RL can be used to create more intelligent and adaptive NPCs in video games.
- **Industrial Control**: RL can be used to optimize industrial processes.
- **Education**: RL can be used to create adaptive learning systems.
- **Agriculture**: RL can be used to optimize agricultural operations.

Machine Learning Lifecycle

The machine learning lifecycle is a process that guides the development and deployment of machine learning models in a structured way. It consists of various steps.
Each step plays a crucial role in ensuring the success and effectiveness of the machine learning solution. By following the machine learning lifecycle, organizations can solve complex problems systematically, leverage data-driven insights, and create scalable and sustainable machine learning solutions that deliver tangible value.

The steps to be followed in the machine learning lifecycle are:


Machine Learning Lifecycle
Problem Definition
Data Collection
Data Cleaning and Preprocessing
Exploratory Data Analysis (EDA)
Feature Engineering and Selection
Model Selection
Model Training
Model Evaluation and Tuning
Model Deployment
Model Monitoring and Maintenance

Machine Learning Lifecycle

Step 1: Problem Definition

Embarking on the machine learning journey involves a well-defined lifecycle, starting with the crucial step of problem definition. In this initial phase, stakeholders collaborate to identify the business problem at hand and frame it in a way that sets the stage for the entire process.
By framing the problem in a comprehensive manner, the team establishes a foundation for the entire machine learning lifecycle. Crucial elements, such as **project objectives, desired outcomes, and the scope of the task**, are carefully delineated during this stage.
Here are the basic features of problem definition:

- **Collaboration:** Work together with stakeholders to understand and define the business problem.
- **Clarity:** Clearly articulate the objectives, desired outcomes, and scope of the task.
- **Foundation:** Establish a solid foundation for the machine learning process by framing the problem comprehensively.

Step 2: Data Collection

Following the precision of problem definition, the machine learning lifecycle progresses to the pivotal stage of data collection. This phase involves the systematic gathering of datasets that will serve as the raw material for model development. The quality and diversity of the data collected directly impact the robustness and generalizability of the machine learning model.

During data collection, practitioners must consider the relevance of the data to the defined problem, ensuring that the selected datasets encompass the necessary features and characteristics. Additionally, factors such as data volume, quality, and ethical considerations play a crucial role in shaping the foundation for subsequent phases of the machine learning lifecycle. A meticulous and well-organized approach to data collection lays the groundwork for effective model training, evaluation, and deployment, ensuring that the resulting model is both accurate and applicable to real-world scenarios.

Here are the basic features of Data Collection:

- **Relevance:** Collect data that is relevant to the defined problem and includes necessary features.
- **Quality:** Ensure data quality by considering factors like accuracy, completeness, and ethical considerations.
- **Quantity:** Gather sufficient data volume to train a robust machine learning model.
- **Diversity:** Include diverse datasets to capture a broad range of scenarios and patterns.

Step 3: Data Cleaning and Preprocessing

With datasets in hand, the machine learning journey advances to the critical stages of data cleaning and preprocessing. Raw data, is often messy and unstructured. Data cleaning involves addressing issues such as missing values, outliers, and inconsistencies that could compromise the accuracy and reliability of the machine learning model.

Preprocessing takes this a step further by standardizing formats, scaling values, and encoding categorical variables, creating a consistent and well-organized dataset. The objective is to refine the raw data into a format that facilitates meaningful analysis during subsequent phases of the machine learning lifecycle. By investing time and effort in data cleaning and preprocessing, practitioners lay the foundation for robust model development, ensuring that the model is trained on high-quality, reliable data.

Here are the basic features of Data Cleaning and Preprocessing:

- **Data Cleaning:** Address issues such as missing values, outliers, and inconsistencies in the data.
- **Data Preprocessing:** Standardize formats, scale values, and encode categorical variables for consistency.
- **Data Quality:** Ensure that the data is well-organized and prepared for meaningful analysis.
- **Data Integrity:** Maintain the integrity of the dataset by cleaning and preprocessing it effectively.

Step 4: Exploratory Data Analysis (EDA)

Now, focus turns to understanding the underlying patterns and characteristics of the collected data. Exploratory Data Analysis (EDA) emerges as a pivotal phase, where practitioners leverage various statistical and visual tools to gain insights into the dataset's structure.

During EDA, patterns, trends, and potential challenges are unearthed, providing valuable context for subsequent decisions in the machine learning process. Visualizations, summary statistics, and correlation analyses offer a comprehensive view of the data, guiding practitioners toward informed choices in feature engineering, model selection, and other critical aspects. EDA acts as a compass, directing the machine learning journey by revealing the intricacies of the data and informing the development of effective and accurate predictive models.

Here are the basic features of Exploratory Data Analysis:

- **Exploration:** Use statistical and visual tools to explore the structure and patterns in the data.
- **Patterns and Trends:** Identify underlying patterns, trends, and potential challenges within the dataset.
- **Insights:** Gain valuable insights to inform decisions in later stages of the machine learning process.
- **Decision Making:** Use exploratory data analysis to make informed decisions about feature engineering and model selection.

Step 5: Feature Engineering and Selection

Feature engineering takes center stage as a transformative process that elevates raw data into meaningful predictors. Simultaneously, feature selection refines this pool of variables, identifying the most relevant ones to enhance model efficiency and effectiveness.

Feature engineering involves creating new features or transforming existing ones to better capture patterns and relationships within the data. This creative process requires domain expertise and a deep understanding of the problem at hand, ensuring that the engineered features contribute meaningfully to the predictive power of the model. On the other hand, feature selection focuses on identifying the subset of features that most significantly impact the model's performance. This dual approach seeks to strike a delicate balance, optimizing the feature set for predictive accuracy while minimizing computational complexity.

Here are the basic features of Feature Engineering and Selection:

- **Feature Engineering:** Create new features or transform existing ones to better capture patterns and relationships.
- **Feature Selection:** Identify the subset of features that most significantly impact the model's performance.
- **Domain Expertise:** Leverage domain knowledge to engineer features that contribute meaningfully to predictive power.
- **Optimization:** Balance feature set for predictive accuracy while minimizing computational complexity.

Step 6: Model Selection

Navigating the machine learning lifecycle requires the judicious selection of a model that aligns with the defined problem and the characteristics of the dataset. Model selection is a pivotal decision that determines the algorithmic framework guiding the predictive

capabilities of the machine learning solution. The choice depends on the nature of the data, the complexity of the problem, and the desired outcomes.
Here are the basic features of Model Selection:

- **Alignment:** Select a model that aligns with the defined problem and characteristics of the dataset.
- **Complexity:** Consider the complexity of the problem and the nature of the data when choosing a model.
- **Decision Factors:** Evaluate factors like performance, interpretability, and scalability when selecting a model.
- **Experimentation:** Experiment with different models to find the best fit for the problem at hand.

Step 7: Model Training

With the selected model in place, the machine learning lifecycle advances to the transformative phase of model training. This process involves exposing the model to historical data, allowing it to learn patterns, relationships, and dependencies within the dataset.
Model training is an iterative and dynamic journey, where the algorithm adjusts its parameters to minimize errors and enhance predictive accuracy. During this phase, the model fine-tunes its understanding of the data, optimizing its ability to make meaningful predictions. Rigorous validation processes ensure that the trained model generalizes well to new, unseen data, establishing a foundation for reliable predictions in real-world scenarios.
Here are the basic features of Model Training:

- **Training Data:** Expose the model to historical data to learn patterns, relationships, and dependencies.
- **Iterative Process:** Train the model iteratively, adjusting parameters to minimize errors and enhance accuracy.
- **Optimization:** Fine-tune the model’s understanding of the data to optimize predictive capabilities.
- **Validation:** Rigorously validate the trained model to ensure generalization to new, unseen data.

Step 8: Model Evaluation and Tuning

Model evaluation involves rigorous testing against validation datasets, employing metrics such as accuracy, precision, recall, and F1 score to gauge its effectiveness.
Evaluation is a critical checkpoint, providing insights into the model’s strengths and weaknesses. If the model falls short of desired performance levels, practitioners initiate model tuning—a process that involves adjusting hyperparameters to enhance predictive

accuracy. This iterative cycle of evaluation and tuning is crucial for achieving the desired level of model robustness and reliability.
Here are the basic features of Model Evaluation and Tuning:

- **Evaluation Metrics:** Use metrics like accuracy, precision, recall, and F1 score to evaluate model performance.

- **Strengths and Weaknesses:** Identify the strengths and weaknesses of the model through rigorous testing.
- **Iterative Improvement:** Initiate model tuning to adjust hyperparameters and enhance predictive accuracy.
- **Model Robustness:** Iterate through evaluation and tuning cycles to achieve desired levels of model robustness and reliability.

Step 9: Model Deployment

Upon successful evaluation, the machine learning model transitions from development to real-world application through the deployment phase. Model deployment involves integrating the predictive solution into existing systems or processes, allowing stakeholders to leverage its insights for informed decision-making.
Model deployment marks the culmination of the machine learning lifecycle, transforming theoretical insights into practical solutions that drive tangible value for organizations.
Here are the basic features of Model Deployment:

- **Integration:** Integrate the trained model into existing systems or processes for real-world application.
- **Decision Making:** Use the model's predictions to inform decision-making and drive tangible value for organizations.
- **Practical Solutions:** Deploy the model to transform theoretical insights into practical solutions that address business needs.
- **Continuous Improvement:** Monitor model performance and make adjustments as necessary to maintain effectiveness over time.

Applications of Machine Learning:

1. **Image Recognition**

Image Recognition is one of the reasons behind the boom one could have experienced in the field of Deep Learning. The task which started from classification between cats and dog images has now evolved up to the level of Face Recognition and real-world use cases based on that like employee attendance tracking.
Also, image recognition has helped revolutionized the healthcare industry by employing smart systems in disease recognition and diagnosis methodologies.

2. **Speech Recognition**

Speech Recognition based smart systems like Alexa and Siri have certainly come across and used to communicate with them. In the backend, these systems are based basically on Speech Recognition systems. These systems are designed such that they can convert voice instructions into text.

One more application of the Speech recognition that we can encounter in our day-to-day life is that of performing Google searches just by speaking to it.

3. **Recommender Systems**

As our world has digitalized more and more approximately all tech giants try to provide customized services to its users. This application is possible just because of the recommender system which can analyse a user's preferences and search history and based on that they can recommend content or services to them.
An example of these services is very common for example YouTube. It recommends new videos and content based on the user's past search patterns. Netflix recommends movies and series based on the interest provided by users when someone creates an account for the very first time.

4. **Fraud Detection**

In today's world, most things have been digitalized varying from buying toothbrushes or making transactions of millions of dollars everything is accessible and easy to use. But with this process of digitization cases of fraudulent transaction and fraudulent activities have increased. Identifying them is not that easy but machine learning systems are very efficient in these tasks.
Due to these applications only whenever the system detects red flags in a user's activity than a suitable notification be provided to the administrator so, that these cases can be monitored properly for any spam or fraud activities.

5. **Self-Driving Cars**

It would have been assumed that there is certainly some ghost who is driving a car if we ever saw a car being driven without a driver but all thanks to machine learning and deep learning that in today's world, this is possible and not a story from some fictional book. Even though the algorithms and tech stack behind these technologies are highly advanced but at the core it is machine learning which has made these applications possible.
The most common example of this use case is that of the Tesla cars which are well-tested and proven for autonomous driving.

6. **Medical Diagnosis**

If you are a machine learning practitioner or even if you are a student then you must have heard about projects like breast cancer Classification, Parkinson's Disease Classification, Pneumonia detection, and many more health-related tasks which are performed by machine learning models with more than 90% of accuracy.
Not even in the field of disease diagnosis in human beings but they work perfectly fine for plant disease-related tasks whether it is to predict the type of disease it is or to detect whether some disease is going to occur in the future.

Basic concepts of Python

What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

Python Syntax compared to other programming languages

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Why Python is Called Interpreted Language

Python is called an interpreted language because it executes code logic directly, line by line, without the need for a separate compilation step. In methods to compiled languages like C or C++, where the source code is translated into machine code before execution, Python code is translated into intermediate code by the Python interpreter.

Python is an Interpreted as well as Compiled language

Python is Interpreted as well as compilation language here, we will understand what does means by interpreted and compilation.

Compilation

In a compiled language, the supply code is translated into gadget code or executable code with the aid of a compiler before execution. The compilation system typically includes more than one ranges, which include lexical analysis, syntax analysis, optimization, and code era. Once the code is compiled, the resulting machine code may be achieved immediately with the aid of the computer's processor.

Compiled languages offer numerous advantages, which include quicker execution pace and higher overall performance because the code has already been translated into gadget instructions. However, compilation introduces a further step within the development system, as developers need to assemble their code earlier than running it. Examples of compiled languages include C and Rust.

Interpretation

In an interpreted language, the supply code is performed line by using line through an interpreter at runtime. The interpreter reads each line of code, interprets it into machine instructions, and executes it straight away. There isn't any separate compilation step, and the supply code stays in its unique form.

Interpreted languages offer benefits inclusive of portability, because the equal source code may be done on unique platforms with out change. Additionally, interpreted languages frequently offer functions together with dynamic typing and runtime introspection, that may make development and debugging easier. Examples of interpreted languages consist of Python, JavaScript, and Ruby.

Why Python is called Interpreted Language?

Python is mostly an interpreted language, even though it consists of elements of each interpretation and compilation. Let's explore Python's execution model to understand why it is called an interpreted language:

Source Code

In Python, the supply code is written in undeniable textual content documents with a .py extension. These documents incorporate human-readable code this is written the use of Python's syntax and language features.

Lexical Analysis
When a Python script is finished, step one is lexical evaluation, also called tokenization. In this step, the interpreter reads the source code and breaks it down into character tokens, together with keywords, identifiers, operators, and literals. This procedure creates a chain of tokens that represent the shape of the code.

Parsing
After lexical evaluation, the interpreter performs parsing, which involves studying the series of tokens to determine the syntactic structure of the code. The parser exams whether the code conforms to the guidelines of the Python language grammar and constructs a parse tree or summary syntax tree (AST) representing the code's structure.

Compilation to Bytecode
Once the parse tree is generated, the interpreter interprets it into an intermediate representation called bytecode. Bytecode is a low-degree, platform-unbiased representation of the source code this is executed by using the Python Virtual Machine (PVM). The compilation to bytecode is accomplished by the Python compiler, also known as the Python bytecode compiler.

Execution with the aid of the Python Virtual Machine (PVM):
The bytecode generated via the compiler is done with the aid of the ***Python Virtual Machine (PVM)***. The PVM is a runtime surroundings that interprets and executes Python bytecode. It consists of a bytecode interpreter, memory manager, garbage collector, and other additives chargeable for dealing with the execution of Python code.

Just-In-Time (JIT) Compilation:
In addition to interpretation, Python also carries elements of compilation thru Just-In-Time (JIT) compilation. JIT compilation is a way in which the interpreter dynamically compiles bytecode into local system code at runtime, optimizing overall performance for regularly done code paths. This allows Python to obtain a balance among the power of interpretation and the overall performance of compilation.

Advantages of Interpretation
Interpreted languages like Python offer numerous advantages:

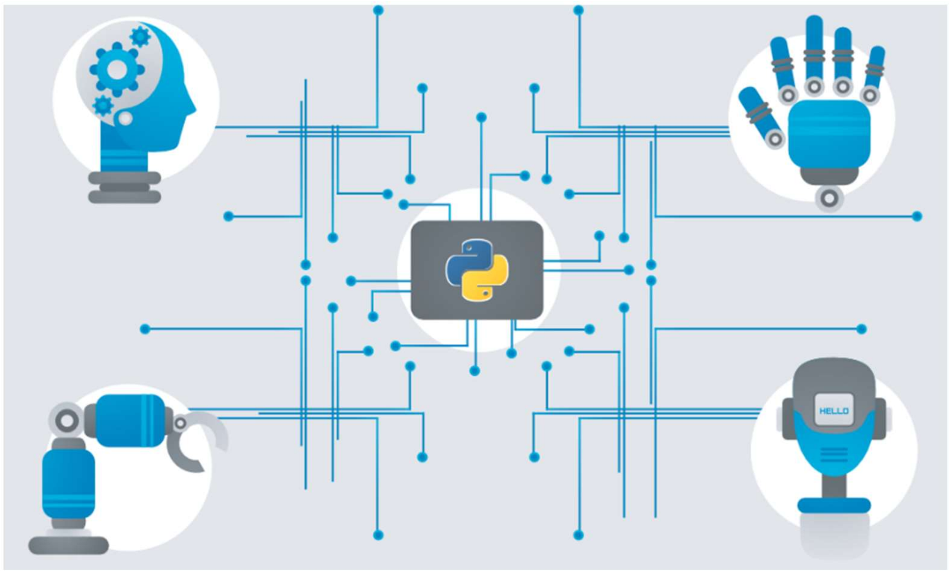
- **Portability:** Interpreted languages are regularly platform-independent, permitting the same source code to run on unique running systems and architectures without change.
- **Rapid Development:** Interpreted languages normally have shorter improvement cycles, as there may be no separate compilation step. Developers can write, test, and execute code faster, facilitating fast prototyping and generation.
- **Dynamic Typing:** Interpreted languages frequently help dynamic typing, permitting variables to exchange their type dynamically at runtime. This flexibility can simplify improvement and make code greater adaptable to converting necessities.

Disadvantages of Interpretation

Interpreted languages like Python offer numerous disadvantages:

- **Slower Execution**: Interpreted languages like Python typically run slower compared to compiled languages because the interpreter translates the source code into intermediate code during runtime. This overhead can result in slower execution speeds, especially for performance-critical applications.
- **Dependency on Interpreter**: Python code requires the presence of the Python interpreter to run, which adds an extra layer of dependency. Users need to have the appropriate version of the interpreter installed on their systems to execute Python programs.
- **Difficulty in Hiding Source Code**: Since Python code is distributed as source files, it can be easier for others to access and view the source code. While tools like obfuscation can be used to make the code less readable, it's inherently more difficult to protect Python code compared to compiled languages.
- **Performance Overhead:** Interpretation can introduce a overall performance overhead compared to compiled languages, as the interpreter must parse, translate, and execute each line of code at runtime. This overhead can result in slower execution speeds for sure sorts of packages.
- **Lack of Optimization:** Interpreted languages may additionally lack a number of the optimization opportunities to be had to compiled languages, inclusive of static analysis and code optimization. This can bring about suboptimal overall performance for overall performance-important applications.

Why is Python the Best-Suited Programming Language for Machine Learning?



Python is currently the most popular programming language for research and development in Machine Learning. The interest in Python for Machine Learning has spiked to an all-new high with other ML languages such as R, Java, Scala, Julia, etc. lagging far behind.So, let's now understand why Python is so popular and consequently why it is best-suited for ML. Some of these reasons for this are given as follows:

1. Python is Easy to Use

Nobody likes excessively complicated things and so the ease of using Python is one of the main reasons why it is so popular for Machine Learning. It is **simple** with an **easily readable syntax** and that makes it well-loved by both seasoned developers and experimental students.

In addition to this, Python is also **supremely efficient**. It allows developers to complete more work using fewer lines of code. The Python code is also easily understandable by humans, which makes it ideal for making Machine Learning models.

2. Python has multiple Libraries and Frameworks

Python is already quite popular and consequently, it has hundreds of different libraries and frameworks that can be used by developers. These libraries and frameworks are really useful in saving time which in turn makes Python even more popular (That's a beneficial cycle!!!).

There are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning. Some of these are given below:

- **Keras** is an open-source library that is particularly focused on experimentation with deep neural networks.
- **TensorFlow** is a free software library that is used for many machine learning applications like neural networks. (They seem to be quite popular!)
- **Scikit-learn** is a free software library for Machine Learning that various classification, regression and clustering algorithms related to this. Also, Scikit-learn can be used in conjugation with NumPy and SciPy.

3. Python has Community and Corporate Support

Python has been around since 1990 and that is ample time to create a **supportive community**. Because of this support, Python learners can easily improve their Machine Learning knowledge, which only leads to increasing popularity.

Also, **Corporate support** is a very important part of the success of Python for ML. Many top companies such as Google, Facebook, Instagram, Netflix, Quora, etc use Python for their products. In fact, Google is single-handedly responsible for creating many of the Python libraries for Machine Learning such as Keras, TensorFlow, etc.

4. Python is Portable and Extensible

This is an important reason why Python is so popular in Machine Learning. A lot of cross-language operations can be performed easily on Python because of its **portable and extensible nature**. There are many data scientists who prefer using Graphics Processing Units (GPUs) for training their ML models on their own machines and the portable nature of Python is well suited for this.

Python Data Structures

Python Lists

- **Python Lists** are just like dynamically sized arrays, declared in other languages (vector in C++ and Array List in Java). In simple language, a list is a collection of things, enclosed in [ ] and separated by commas.
- The list is a sequence data type which is used to store the collection of data. *Tuples* and *String* are other types of sequence data types.

Example of the list in Python:

-Here we are creating a Python **List** using [].

```
Var = ["Geeks", "for", "Geeks"]
print(Var)
```

Output:

```
["Geeks", "for", "Geeks"]
```

Lists are the simplest containers that are an integral part of the Python language. Lists need not be homogeneous always which makes it the most powerful tool in Python. A single list may contain DataTypes like Integers, Strings, as well as Objects. Lists are mutable, and hence, they can be altered even after their creation.

Creating a List in Python:

Lists in Python can be created by just placing the sequence inside the square brackets[]. Unlike Sets, a list doesn't need a built-in function for its creation of a list.

Example 1: Creating a list in Python

```
# Python program to demonstrate
# Creation of List
# Creating a List
List = []
print("Blank List: ")
print(List)
# Creating a List of numbers
List = [10, 20, 14]
print("\nList of numbers: ")
print(List)
# Creating a List of strings and accessing
# using index
List = ["Geeks", "For", "Geeks"]
print("\nList Items: ")
print(List[0])
print(List[2])
```

Output:

```
Blank List:
[]

List of numbers:
[10, 20, 14]

List Items:
Geeks
Geeks
```

Example 2: Creating a list with multiple distinct or duplicate elements

A list may contain duplicate values with their distinct positions and hence, multiple distinct or duplicate values can be passed as a sequence at the time of list creation.

```
# Creating a List with
# the use of Numbers  (Having duplicate values)
List = [1, 2, 4, 4, 3, 3, 3, 6, 5]
print("\nList with the use of Numbers: ")
print(List)

# Creating a List with
# mixed type of values (Having numbers and strings)
List = [1, 2, 'Geeks', 4, 'For', 6, 'Geeks']
print("\nList with the use of Mixed Values: ")
print(List)
```

Output:

```
List with the use of Numbers:
[1, 2, 4, 4, 3, 3, 3, 6, 5]
List with the use of Mixed Values:
[1, 2, 'Geeks', 4, 'For', 6, 'Geeks']
```

Accessing elements from the List

In order to access the list items refer to the index number. Use the index operator [ ] to access an item in a list. The index must be an integer. Nested lists are accessed using nested indexing.

Example 1: Accessing elements from list

```
# Python program to demonstrate
# accessing of element from list

# Creating a List with
# the use of multiple values
List = ["Geeks", "For", "Geeks"]

# accessing a element from the
# list using index number
print("Accessing a element from the list")
print(List[0])
print(List[2])
```

Output:

```
Accessing a element from the list
Geeks
Geeks
```

Example 2: Accessing elements from a multi-dimensional list

```
# Creating a Multi-Dimensional List
# (By Nesting a list inside a List)
List = [['Geeks', 'For'], ['Geeks']]

# accessing an element from the
# Multi-Dimensional List using
# index number
print("Accessing a element from a Multi-Dimensional list")
print(List[0][1])
print(List[1][0])
```

Output:

```
Accessing a element from a Multi-Dimensional list
For
Geeks
```

Negative indexing:

In Python, negative sequence indexes represent positions from the end of the List. Instead of having to compute the offset as in List[len(List)-3], it is enough to just write List[-3]. Negative indexing means beginning from the end, -1 refers to the last item, -2 refers to the second-last item, etc.

Example for negative indexing:

```
List = [1, 2, 'Geeks', 4, 'For', 6, 'Geeks']

# accessing an element using
# negative indexing
print("Accessing element using negative indexing")

# print the last element of list
print(List[-1])

# print the third last element of list
print(List[-3])
```

Output

```
Accessing element using negative indexing
Geeks
For
```

Slicing of a List

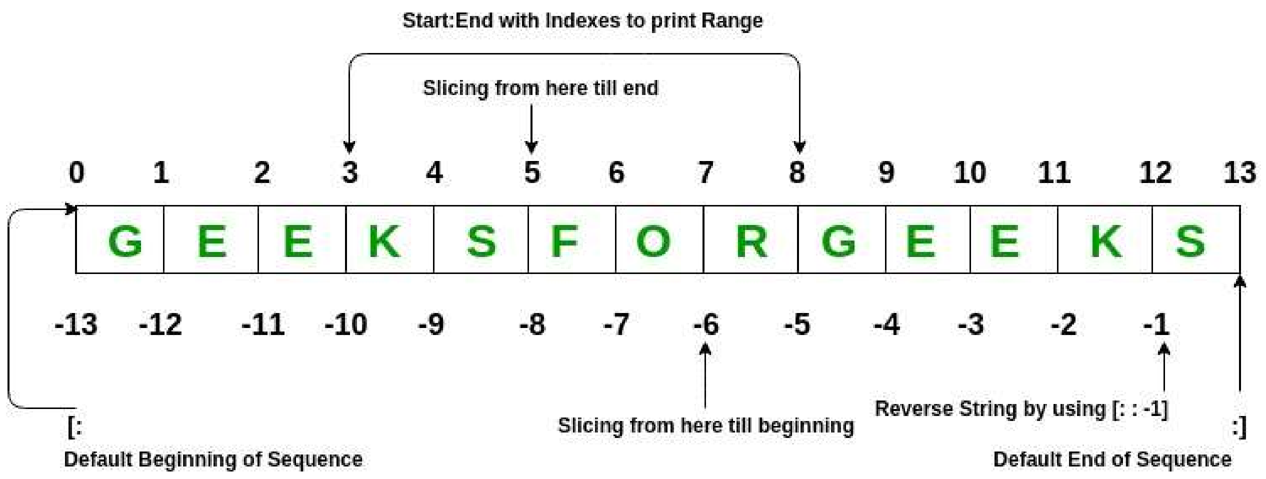
We can get substrings and sublists using a slice. In Python List, there are multiple ways to print the whole list with all the elements, but to print a specific range of elements from the list, we use the Slice operation.

Slice operation is performed on Lists with the use of a colon(:).

- To print elements from beginning to a range, use:
  [: Index]
- To print elements from beginning to negative range, use:
  [:-Index]
- To print elements from a specific Index till the end, use
  [Index:]
- To print elements from a specific negative Index till the end, use
  [-Index:]

- To print the whole list in reverse order, use

[::-1]
Note – To print elements of List from rear-end, use Negative Indexes.


Start:End with Indexes to print Range
Slicing from here till end
0 1 2 3 4 5 6 7 8 9 10 11 12 13
G E E K S F O R G E E K S
-13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
[:
Default Beginning of Sequence
Slicing from here till beginning
Reverse String by using [: : -1]
:]
Default End of Sequence

UNDERSTANDING SLICING OF LISTS:

- pr[0] accesses the first item, 2.
- pr[-4] accesses the fourth item from the end, 5.
- pr[2:] accesses [5, 7, 11, 13], a list of items from third to last.
- pr[:4] accesses [2, 3, 5, 7], a list of items from first to fourth.
- pr[2:4] accesses [5, 7], a list of items from third to fifth.
- pr[1::2] accesses [3, 7, 13], alternate items, starting from the second item.

Example for slicing:

```
# Python program to demonstrate
# Removal of elements in a List
# Creating a List
List = ['G', 'E', 'E', 'K', 'S', 'F', 'O', 'R', 'G', 'E', 'E', 'K', 'S']
print("Initial List: ")
print(List)

# Print elements of a range
# using Slice operation
Sliced_List = List[3:8]
print("\nSlicing elements in a range 3-8: ")
print(Sliced_List)

# Print elements from a
# pre-defined point to end
Sliced_List = List[5:]
print("\nElements sliced from 5th element till the end: ")

print(Sliced_List)
```

```
# Printing elements from
# beginning till end
Sliced_List = List[:]
print("\nPrinting all elements using slice operation: ")
print(Sliced_List)
```

Output:

```
Initial List:
['G', 'E', 'E', 'K', 'S', 'F', 'O', 'R', 'G', 'E', 'E', 'K', 'S']

Slicing elements in a range 3-8:
['K', 'S', 'F', 'O', 'R']

Elements sliced from 5th element till the end:
['F', 'O', 'R', 'G', 'E', 'E', 'K', 'S']

Printing all elements using slice operation:
['G', 'E', 'E', 'K', 'S', 'F', 'O', 'R', 'G', 'E', 'E', 'K', 'S']
```

Negative index List slicing

```
# Creating a List
List = ['G', 'E', 'E', 'K', 'S', 'F','O', 'R', 'G', 'E', 'E', 'K', 'S']
print("Initial List: ")
print(List)

# Print elements from beginning
# to a pre-defined point using Slice
Sliced_List = List[:-6]
print("\nElements sliced till 6th element from last: ")
print(Sliced_List)

# Print elements of a range
# using negative index List slicing
Sliced_List = List[-6:-1]
print("\nElements sliced from index -6 to -1")
print(Sliced_List)

# Printing elements in reverse
# using Slice operation
Sliced_List = List[::-1]

print("\nPrinting List in reverse: ")
print(Sliced_List)
```

Output:

```
Initial List:
['G', 'E', 'E', 'K', 'S', 'F', 'O', 'R', 'G', 'E', 'E', 'K', 'S']

Elements sliced till 6th element from last:
['G', 'E', 'E', 'K', 'S', 'F', 'O']

Elements sliced from index -6 to -1
['R', 'G', 'E', 'E', 'K']

Printing List in reverse:
['S', 'K', 'E', 'E', 'G', 'R', 'O', 'F', 'S', 'K', 'E', 'E', 'G']
```

Python List methods

***Python List Methods** are the built-in methods in lists used to perform operations on Python lists/arrays.*

1. Python append() Method: Adds element to the end of a list.

Syntax: *list.append (element)*

Example:

```
# Adds List Element as value of List.
List = ['Mathematics', 'chemistry', 1997, 2000]
List.append(20544)
print(List)
```

Output

```
['Mathematics', 'chemistry', 1997, 2000, 20544]
```

2. Python insert() Method: Inserts an element at the specified position.

Syntax: *list.insert(<position, element)*

Note: The position mentioned should be within the range of List, as in this case between 0 and 4, else wise would throw IndexError.

Example:

```
List = ['Mathematics', 'chemistry', 1997, 2000]
# Insert at index 2 value 10087
List.insert(2, 10087)
print(List)
```

Output

```
['Mathematics', 'chemistry', 10087, 1997, 2000]
```

3. Python extend() Method: Adds items of an iterable(list, array, string , etc.) to the end of a list.

Syntax: *List1.extend(List2)*

Example:

```
List1 = [1, 2, 3]
List2 = [2, 3, 4, 5]

# Add List2 to List1
List1.extend(List2)
print(List1)

# Add List1 to List2 now
List2.extend(List1)
print(List2)
```

Output

```
[1, 2, 3, 2, 3, 4, 5]
[2, 3, 4, 5, 1, 2, 3, 2, 3, 4, 5]
```

4. Python sum() Method: Calculates the sum of all the elements of the List.

Syntax: *sum(List)*

Example:

```
List = [1, 2, 3, 4, 5]
print(sum(List))
```

Output

```
15
```

5. Python count() Method: Calculates the total occurrence of a given element of the List.

Syntax: *List.count(element)*

Example:

```
List = [1, 2, 3, 1, 2, 1, 2, 3, 2, 1]
print(List.count(1))
```

Output

```
4
```

6. **Python len() Method:** Calculates the total length of the List.

Syntax: *len(list_name)*

Example:

```
List = [1, 2, 3, 1, 2, 1, 2, 3, 2, 1]
print(len(List))
```

Output

```
10
```

7. Python index() Method: Returns the index of the first occurrence. The start and end indexes are not necessary parameters.

Syntax: List.index(element[,start[,end]])

Example:

```
List = [1, 2, 3, 1, 2, 1, 2, 3, 2, 1]
print(List.index(2))
```

Output

```
1
```

8. Python min() Method: Calculates minimum of all the elements of List.

*Syntax: min(iterable, *iterables[, key])*

Example:

```
numbers = [5, 2, 8, 1, 9]
print(min(numbers))
```

Output

```
1
```

9. Python max() Method: Calculates the maximum of all the elements of the List.

*Syntax: max(iterable, *iterables[, key])*

Example:

```
numbers = [5, 2, 8, 1, 9]
print(max(numbers))
```

Output

```
9
```

10. Python sort() Method: Sort the given data structure (both tuple and list) in ascending order.**Key** and **reverse_flag** are not necessary parameter and reverse_flag is set to False if nothing is passed through sorted().

Syntax: list.sort([key,[Reverse_flag]])

Example:

```
List = [2.3, 4.445, 3, 5.33, 1.054, 2.5]
#Reverse flag is set True
List.sort(reverse=True)
```

```
#List.sort().reverse(), reverses the sorted list
print(List)
```

Output

```
[5.33, 4.445, 3, 2.5, 2.3, 1.054]
```

11. Python reverse() Method: reverse() function reverses the order of list.

Syntax: *list. reverse()*

Example:

```
# creating a list
list = [1,2,3,4,5]
#reversing the list
list.reverse()
#printing the list
print(list)
```

Output

```
[5, 4, 3, 2, 1]
```

12. Python pop() Method: Removes an item from a specific index in a list.

Syntax: *list.pop([index])*

Example:

```
List = [2.3, 4.445, 3, 5.33, 1.054, 2.5]
print(List.pop())
```

Output

```
2.5
```

13. Python del() Method: Deletes an element from the list using it's index.

Syntax: *del list.[index]*

Example:

```
List = [2.3, 4.445, 3, 5.33, 1.054, 2.5]
del List[0]
print(List)
```

Output

```
[4.445, 3, 5.33, 1.054, 2.5]
```

14. Python remove() Method: Removes a specific element using it's value/name.

Syntax: *list.remove(element)*

Example :

```
List = [2.3, 4.445, 3, 5.33, 1.054, 2.5]
List.remove(3)
print(List)
```

Output

```
[2.3, 4.445, 5.33, 1.054, 2.5]
```

Python String

A String is a data structure in Python Programming that represents a sequence of characters. It is an immutable data type, meaning that once you have created a string, you cannot change it. Python String are used widely in many different applications, such as storing and manipulating text data, representing names, addresses, and other types of data that can be represented as text.

Create a String in Python:

Strings in Python can be created using single quotes or double quotes or even triple quotes.

Example:

In this example, we will demonstrate different ways to create a Python String. We will create a string using single quotes (‘ ‘), double quotes (” “), and triple double quotes (“”” “””). The triple quotes can be used to declare multiline strings in Python.

```
# Creating a String
# with single Quotes
String1 = 'Welcome to the Geeks World'
print("String with the use of Single Quotes: ")
print(String1)

# Creating a String
# with double Quotes
String1 = "I'm a Geek"
print("\nString with the use of Double Quotes: ")
print(String1)

# Creating a String
# with triple Quotes
String1 = '''I'm a Geek and I live in a world of "Geeks"'''

print("\nString with the use of Triple Quotes: ")
print(String1)
```

Output:

```
String with the use of Single Quotes:
Welcome to the Geeks World
String with the use of Double Quotes:
I'm a Geek
String with the use of Triple Quotes:
I'm a Geek and I live in a world of "Geeks"
```

Python String Methods

Python string methods *is a collection of in-built Python functions that operates on lists.*

Case Changing of Python String Methods

The below Python functions are used to change the case of the strings. Let's look at some Python string methods with examples:

lower(): Converts all uppercase characters in a string into lowercase
upper(): Converts all lowercase characters in a string into uppercase
title(): Convert string to title case
swapcase(): Swap the cases of all characters in a string
capitalize(): Convert the first character of a string to uppercase

Example: Changing the case of Python String Methods

```
# Python3 program to show the
# working of upper() function
text = 'geeKs For geEkS'

# upper() function to convert
# string to upper case
print("\nConverted String:")
print(text.upper())

# lower() function to convert
# string to lower case
print("\nConverted String:")
print(text.lower())

# converts the first character to
# upper case and rest to lower case
print("\nConverted String:")
print(text.title())

# swaps the case of all characters in the string
# upper case character to lowercase and viceversa
print("\nConverted String:")
print(text.swapcase())

# convert the first character of a string to uppercase
print("\nConverted String:")
print(text.capitalize())

# original string never changes
print("\nOriginal String")
print(text)
```

Output

```
Converted String:
GEEKS FOR GEEKS

Converted String:
geeks for geeks

Converted String:
Geeks For Geeks

Converted String:
GEEkS fOR GEeKs

Original String
geeKs For geEkS
```

Python Boolean

Python boolean type is one of the built-in data types provided by Python, which represents one of the two values i.e. True or False. Generally, it is used to represent the truth values of the expressions.

Example

Input:1==1
Output: True

Input:2<1
Output: False

What is the bool() Method in Python?

bool() is a built-in function of Python programming language. It is used to convert any other data type value (string, integer, float, etc) into a boolean data type.
Boolean data type can store only 2 values: **True** and **False.**

False Values: 0, NULL, empty lists, tuples, dictionaries, etc.
True Values: All other values will return true.

bool() Method Syntax: ***bool([x])***

Parameters

x: Any object that you want to convert into a boolean data type.

Return

It can return one of the two values.
It returns True if the parameter or value passed is True.
It returns False if the parameter or value passed is False.

Here are a few cases, in which Python's bool() method returns false. Except these all other values return True.

If a False value is passed.
If None is passed.
If an empty sequence is passed, such as (), [], ", etc.
If Zero is passed in any numeric type, such as 0, 0.0, etc.
If an empty mapping is passed, such as {}.
If Objects of Classes having __bool__() or __len()__ method, returning 0 or False.

Example

In this example, we are checking **the bool()** method of Python with multiple types of variables like Boolean, Integers, None, Tuple, Float, strings, and Dictionary**.**

```
# Python program to illustrate
# built-in method bool()

# Returns False as x is False
x = False
print(bool(x))

# Returns True as x is True
x = True
```

```
print(bool(x))

# Returns False as x is not equal to y
x = 5
y = 10
print(bool(x == y))

# Returns False as x is None
x = None
print(bool(x))

# Returns False as x is an empty sequence
x = ()
print(bool(x))

# Returns False as x is an empty mapping
x = {}
print(bool(x))

# Returns False as x is 0
x = 0.0
print(bool(x))

# Returns True as x is a non empty string
x = 'GeeksforGeeks'
print(bool(x))
```

Output:

```
False
True
False
False
False
False
False
True
```

Tuple Methods

In Python, tuples are immutable sequences, similar to lists, but with the key difference that tuples cannot be modified once created. As a result, tuples have fewer methods compared to lists. Here are the main methods and operations you can perform on tuples in Python:

Tuple Methods:

1. count(value):
   Returns the number of times a specified value appears in the tuple.
   ```
   my_tuple = (1, 2, 2, 3, 4, 2)
   print(my_tuple.count(2))  # Output: 3
   ```

2. index (value, [start, [stop]]):
   Returns the index of the first occurrence of the specified value. You can also specify start and stop indices to search within a subset of the tuple.
   ```
   my_tuple = (1, 2, 3, 4, 3, 5)
   print(my_tuple.index(3))  # Output: 2
   ```

Tuple Operations:

1. Concatenation (+):
   You can concatenate tuples using the + operator.
   ```
   tuple1 = (1, 2, 3)
   tuple2 = (4, 5, 6)
   concatenated_tuple = tuple1 + tuple2
   print(concatenated_tuple)  # Output: (1, 2, 3, 4, 5, 6)
   ```

2. Repetition (*):
   You can create a new tuple by repeating the elements of an existing tuple using the * operator.
   ```
   tuple1 = ('a', 'b')
   repeated_tuple = tuple1 * 3
   print(repeated_tuple)  # Output: ('a', 'b', 'a', 'b', 'a', 'b')
   ```

Tuple Slicing:

Tuple slicing works similarly to list slicing in Python and allows you to access subsets of a tuple using the slice notation start:stop:step.
```
my_tuple = (1, 2, 3, 4, 5)
print(my_tuple[1:4])  # Output: (2, 3, 4)
print(my_tuple[:3])   # Output: (1, 2, 3)
print(my_tuple[2:])   # Output: (3, 4, 5)
print(my_tuple[::-1]) # Output: (5, 4, 3, 2, 1)
```

Other Characteristics

- Tuples support membership tests (in and not in operators).
- They can be nested to create complex data structures.
- Tuples are hashable, which means they can be used as keys in dictionaries and as elements of sets, unlike lists.

Remember, once a tuple is created, you cannot add or remove items from it or modify its contents. If you need to change a tuple, you would typically create a new tuple with the desired elements.

Set Methods

In Python, sets are unordered collections of unique elements. They are mutable, which means you can modify them after creation. Sets support a variety of methods for common operations such as adding or removing elements, combining sets, and performing set operations like intersection and difference. Here are the main methods available for sets in Python

Set Methods:

1. add(element):
   Adds a single element to the set. If the element is already present, the set remains unchanged.
   ```
   my_set = {1, 2, 3}
   my_set.add(4)
   print(my_set)  # Output: {1, 2, 3, 4}
   ```

2. update(*others):
   Adds elements from multiple iterables (sets, lists, tuples, etc.) to the set.
   ```
   my_set = {1, 2, 3}
   my_set.update([4, 5])
   print(my_set)  # Output: {1, 2, 3, 4, 5}
   ```

3. remove(element):
   Removes the specified element from the set. Raises a KeyError if the element is not found.
   ```
   my_set = {1, 2, 3}
   my_set.remove(2)
   print(my_set)  # Output: {1, 3}
   ```
4. pop():
   Removes and returns an arbitrary element from the set. Raises KeyError if the set is empty.
   ```
   my_set = {1, 2, 3}
   ```

```
popped_element = my_set.pop()

print(popped_element, my_set)  # Output: (e.g., 1, {2, 3})
```

5. clear():
   Removes all elements from the set.
   ```
   my_set = {1, 2, 3}
   my_set.clear()
   print(my_set)  # Output: set()
   ```

Dictionary Methods

Dictionaries are mutable collections that store mappings of unique keys to values. Dictionaries are extremely flexible and efficient for retrieving and storing data where each item is accessed by a key rather than by its position in the collection. Here are the main methods and operations available for dictionaries:

Ways to defining dictionary:

1. Using Curly Braces {} and Key-Value Pairs
   The most common and straightforward method is using curly braces {} to define key-value pairs separated by colons :
   ```
   my_dict = {'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
   print(my_dict)  # Output: {'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
   ```

2. Using the dict() Constructor with Iterable of Tuples
   The dict() constructor can also take an iterable of tuples where each tuple consists of a key-value pair.
   ```
   My_dict = dict([('key1', 'value1'), ('key2', 'value2'), ('key3', 'value3')])
   print(my_dict)  # Output: {'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
   ```

3. Empty dictionary
   d={}

Characteristics

- Dictionaries are unordered collections (order of elements may vary).
- Keys in dictionaries must be immutable (strings, numbers, or tuples), but values can be mutable.
- Dictionaries are versatile and used extensively for data storage and retrieval where quick lookup times are essential.

Modules And Packages of Python

Pandas:

Definition:
It is a fast, powerful, flexible and easy to use open-source data analysis and manipulation tool, built on top of the Python programming language.

Why it is used:
Pandas allows us to analyse big data and make conclusions based on statistical theories. Pandas can clean messy data sets, and make them readable and relevant. Relevant data is very important in data science.

Functions of Pandas:
- Pandas read_csv (): This function is used to retrieve data from CSV files in the form of a dataframe.
- Pandas head (): This function is used to return the top n (5 by default) values of a data frame or series.
- Pandas tail (): This method is used to return the bottom n (5 by default) rows of a data frame or series.

NumPy:

Definition:
NumPy stands for Numerical Python, is an open-source Python library that provides support for large, multi-dimensional arrays and matrices.
It also has a collection of high-level mathematical functions to operate on arrays.

Why it is used:
NumPy is a general-purpose array-processing package.
It provides a high-performance multidimensional array object and tools for working with these arrays.
It is the fundamental package for scientific computing with Python.

Functions of Pandas:
- np.array (): This function is used to create an array in NumPy.
- np.arange (): This function is used to create an array with a range of values.
- np.random.rand (): This function is used to create an array with random values between 0 and 1.

Matplotlib:

Definition:

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

Why it is used:

Matplotlib is a popular plotting library in Python used for creating high-quality visualizations and graphs. It offers various tools to generate diverse plots, facilitating data analysis, exploration, and presentation.

Functions of Pandas:

- plt.plot (): This function is used to create line charts, scatter plots, and other types of plots. This function is the backbone of most Matplotlib visualizations.
- plt.xlabel () & plt.ylabel (): These functions are used to label the x-axis and y-axis, respectively. They take a string as input and can also accept optional arguments such as font size, font weight, and colour.
- plt.title (): This function is used to add a title to the plot. It takes a string as input and can also accept optional arguments such as font size, font weight, and colour.

Seaborn:

Definition:

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Why it is used:

Seaborn is an amazing visualization library for statistical graphics plotting in Python.
It provides beautiful default styles and colour palettes to make statistical plots more attractive.
It is built on top matplotlib library and is also closely integrated with the data structures from pandas.

Functions of Pandas:

- distplot (): The distplot function is a versatile tool for visualizing univariate distributions. It plots histograms, kernel density estimates (KDE), rug plots, and can also fit parametric distributions such as normal, exponential, and others.
- heatmap (): The heatmap function creates a colour-encoded matrix representation of a dataset. This function is useful for visualizing the relationship between multiple variables.
- barplot (): The barplot function creates a bar chart that displays the mean of a continuous variable for each category in a categorical dataset. This function is useful for comparing the means of different groups.

Scikit-learn:

Definition:
Scikit-Learn, also known as sklearn is a python library to implement machine learning models and statistical modelling. Through scikit-learn, we can implement various machine learning models for regression, classification, clustering, and statistical tools for analysing these models.

Why it is used:
Through scikit-learn, we can implement various machine learning models for regression, classification, clustering, and statistical tools for analysing these models.

Important modules and packages:
- sklearn.datasets: This module provides various datasets for practicing and testing machine learning algorithms. It includes functions to load standard datasets like the iris dataset, Boston housing dataset, MNIST dataset, etc.
- sklearn.preprocessing: This module provides functions for preprocessing and scaling data. It includes techniques like feature scaling, normalization, label encoding, one-hot encoding, etc.
- sklearn.cluster: This module provides clustering algorithms for unsupervised learning. It includes k-means clustering, DBSCAN, hierarchical clustering, and more.

Natural Language Processing (NLP)

The meaning of NLP is Natural Language Processing (NLP) which is a fascinating and rapidly evolving field that intersects computer science, artificial intelligence, and linguistics. NLP focuses on the interaction between computers and human language, enabling machines to understand, interpret, and generate human language in a way that is both meaningful and useful. With the increasing volume of text data generated every day, from social media posts to research articles, NLP has become an essential tool for extracting valuable insights and automating various tasks.

What is Natural Language Processing?

Natural language processing (NLP) is a field of computer science and a subfield of artificial intelligence that aims to make computers understand human language. NLP uses computational linguistics, which is the study of how language works, and various models based on statistics, machine learning, and deep learning. These technologies allow computers to analyze and process text or voice data, and to grasp their full meaning, including the speaker's or writer's intentions and emotions.

NLP powers many applications that use language, such as text translation, voice recognition, text summarization, and chatbots. You may have used some of these applications yourself, such as voice-operated GPS systems, digital assistants, speech-to-text software, and customer service bots. NLP also helps businesses improve their efficiency, productivity, and performance by simplifying complex tasks that involve language.

Working of Natural Language Processing (NLP)

Working in natural language processing (NLP) typically involves using computational techniques to analyse and understand human language. This can include tasks such as language understanding, language generation, and language interaction.

1. Text Input and Data Collection

- **Data Collection**: Gathering text data from various sources such as websites, books, social media, or proprietary databases.
- **Data Storage**: Storing the collected text data in a structured format, such as a database or a collection of documents.

2. Text Preprocessing

Preprocessing is crucial to clean and prepare the raw text data for analysis. Common preprocessing steps include:

- **Tokenization**: Splitting text into smaller units like words or sentences.
- **Lowercasing**: Converting all text to lowercase to ensure uniformity.
- **Stopword Removal**: Removing common words that do not contribute significant meaning, such as "and," "the," "is."

- **Punctuation Removal**: Removing punctuation marks.
- **Stemming and Lemmatization**: Reducing words to their base or root forms. Stemming cuts off suffixes, while lemmatization considers the context and converts words to their meaningful base form.
- **Text Normalization**: Standardizing text format, including correcting spelling errors, expanding contractions, and handling special characters.

3. Text Representation

- **Bag of Words (BoW)**: Representing text as a collection of words, ignoring grammar and word order but keeping track of word frequency.
- **Term Frequency-Inverse Document Frequency (TF-IDF)**: A statistic that reflects the importance of a word in a document relative to a collection of documents.
- **Word Embeddings**: Using dense vector representations of words where semantically similar words are closer together in the vector space (e.g., Word2Vec, GloVe).

4. Feature Extraction

Extracting meaningful features from the text data that can be used for various NLP tasks.

- **N-grams**: Capturing sequences of N words to preserve some context and word order.
- **Syntactic Features**: Using parts of speech tags, syntactic dependencies, and parse trees.
- **Semantic Features**: Leveraging word embeddings and other representations to capture word meaning and context.

5. Model Selection and Training

Selecting and training a machine learning or deep learning model to perform specific NLP tasks.

- **Supervised Learning**: Using labeled data to train models like Support Vector Machines (SVM), Random Forests, or deep learning models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).
- **Unsupervised Learning**: Applying techniques like clustering or topic modeling (e.g., Latent Dirichlet Allocation) on unlabeled data.
- **Pre-trained Models**: Utilizing pre-trained language models such as BERT, GPT, or transformer-based models that have been trained on large corpora.

6. Model Deployment and Inference

Deploying the trained model and using it to make predictions or extract insights from new text data.

- **Text Classification**: Categorizing text into predefined classes (e.g., spam detection, sentiment analysis).
- **Named Entity Recognition (NER)**: Identifying and classifying entities in the text.
- **Machine Translation**: Translating text from one language to another.
- **Question Answering**: Providing answers to questions based on the context provided by text data.

7. Evaluation and Optimization

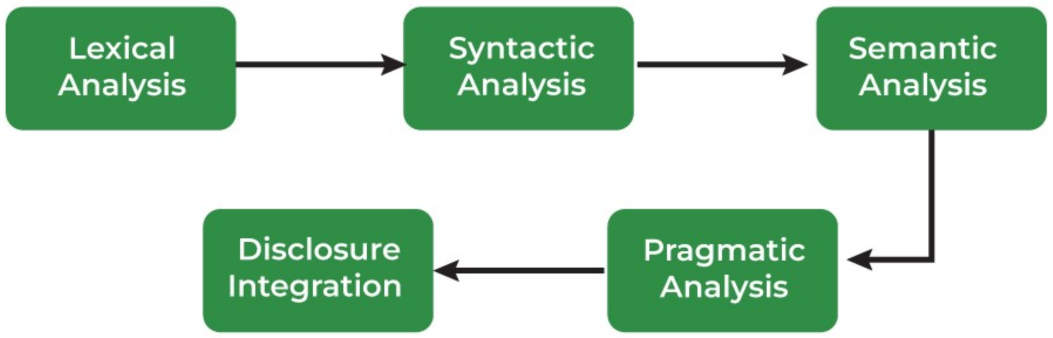
Evaluating the performance of the NLP algorithm using metrics such as accuracy, precision, recall, F1-score, and others.

- **Hyperparameter Tuning**: Adjusting model parameters to improve performance.
- **Error Analysis**: Analyzing errors to understand model weaknesses and improve robustness.

8. Iteration and Improvement

Continuously improving the algorithm by incorporating new data, refining preprocessing techniques, experimenting with different models, and optimizing features.

Phases of Natural Language Processing



NLP Libraries

- NLTK
- Spacy
- Gensim
- fastText
- Stanford toolkit (Glove)
- Apache OpenNLP

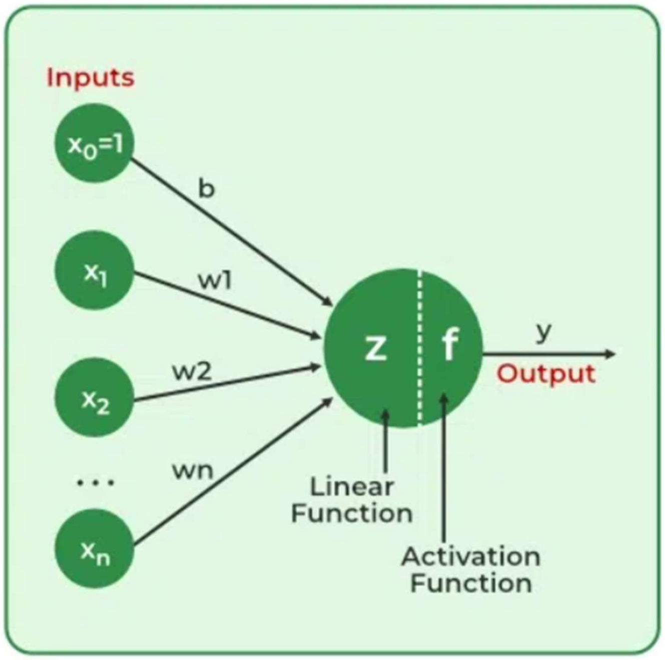
Neural Network

What is a neural network?

Neural Networks are computational models that mimic the complex functions of the human brain. The neural networks consist of interconnected nodes or neurons that process and learn from data, enabling tasks such as pattern recognition and decision making in machine learning. The article explores more about neural networks, their working, architecture and more.

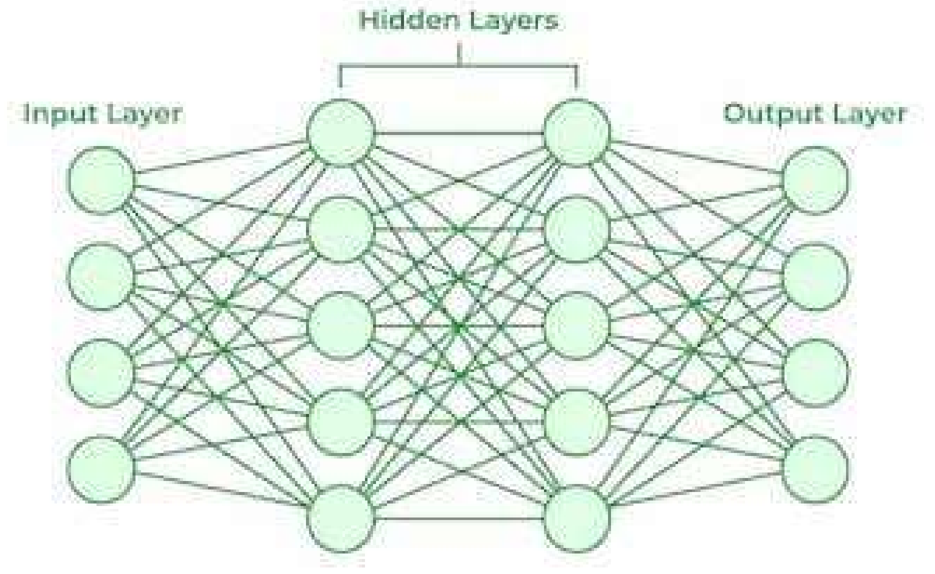
Neural networks extract identifying features from data, lacking pre-programmed understanding. Network components include neurons, connections, weights, biases, propagation functions, and a learning rule. Neurons receive inputs, governed by thresholds and activation functions. Connections involve weights and biases regulating information transfer. Learning, adjusting weights and biases, occurs in three stages: input computation, output generation, and iterative refinement enhancing the network's proficiency in diverse tasks.
These include:

1. The neural network is simulated by a new environment.
2. Then the free parameters of the neural network are changed as a result of this simulation.
3. The neural network then responds in a new way to the environment because of the changes in its free parameters.



Working of a Neural Network

Neural networks are complex systems that mimic some features of the functioning of the human brain. It is composed of an input layer, one or more hidden layers, and an output layer made up of layers of artificial neurons that are coupled. The two stages of the basic process are called backpropagation and forward propagation.



Forward Propagation

- **Input Layer:** Each feature in the input layer is represented by a node on the network, which receives input data.
- **Weights and Connections:** The weight of each neuronal connection indicates how strong the connection is. Throughout training, these weights are changed.
- **Hidden Layers:** Each hidden layer neuron processes inputs by multiplying them by weights, adding them up, and then passing them through an activation function. By doing this, non-linearity is introduced, enabling the network to recognize intricate patterns.
- **Output:** The final result is produced by repeating the process until the output layer is reached.

Backpropagation

- **Loss Calculation:** The network's output is evaluated against the real goal values, and a loss function is used to compute the difference. For a regression problem, the Mean Squared Error (MSE) is commonly used as the cost function.
- **Gradient Descent:** Gradient descent is then used by the network to reduce the loss. To lower the inaccuracy, weights are changed based on the derivative of the loss with respect to each weight.
- **Adjusting weights:** The weights are adjusted at each connection by applying this iterative process, or backpropagation, backward across the network.

- **Training:** During training with different data samples, the entire process of forward propagation, loss calculation, and backpropagation is done iteratively, enabling the network to adapt and learn patterns from the data.
- **Activation Functions:** Model non-linearity is introduced by activation functions like the rectified linear unit (ReLU) or sigmoid. Their decision on whether to "fire" a neuron is based on the whole weighted input.

Types of Neural Networks:

There are *seven* types of neural networks that can be used.

- **Feedforward Networks:** A feedforward neural network is a simple artificial neural network architecture in which data moves from input to output in a single direction. It has input, hidden, and output layers; feedback loops are absent. Its straightforward architecture makes it appropriate for a number of applications, such as regression and pattern recognition.
- **Multilayer Perceptron (MLP):** MLP is a type of feedforward neural network with three or more layers, including an input layer, one or more hidden layers, and an output layer. It uses nonlinear activation functions.
- **Convolutional Neural Network (CNN):** A Convolutional Neural Network (CNN) is a specialized artificial neural network designed for image processing. It employs convolutional layers to automatically learn hierarchical features from input images, enabling effective image recognition and classification. CNNs have revolutionized computer vision and are pivotal in tasks like object detection and image analysis.
- **Recurrent Neural Network (RNN):** An artificial neural network type intended for sequential data processing is called a Recurrent Neural Network (RNN). It is appropriate for applications where contextual dependencies are critical, such as time series prediction and natural language processing, since it makes use of feedback loops, which enable information to survive within the network.
- **Long Short-Term Memory (LSTM):** LSTM is a type of RNN that is designed to overcome the vanishing gradient problem in training RNNs. It uses memory cells and gates to selectively read, write, and erase information.

Bibliography

AI And ML Concepts:

- Artificial Intelligence | An Introduction – GeeksforGeeks (https://www.geeksforgeeks.org/artificial-intelligence-an-introduction/)
- Machine Learning (https://www.javatpoint.com/machine-learning)
- Neural Network (https://www.techtarget.com/searchenterpriseai/definition/neural-network)
- Introduction to Machine Learning With Python by Andreas C.Muller & Sarah Guido

Python Concepts:

- Python Tutorial (**https://www.w3schools.com/python/**)
- Python Tutorial (https://www.tutorialspoint.com/python/index.htm)
- Python For Beginners (https://www.python.org/about/gettingstarted/)
- Python By E.Balaguruswamy

Project Work

Project Title: Image Segmentation and Object Detection Using K-Means

Objective of the Project:

The objective of this project is to perform image segmentation and object detection using K-means clustering. This involves dividing an image into segments based on color similarity and then identifying and isolating a specific object within the image. The focus is on simplifying the image analysis process, enhancing visual understanding, and enabling further image processing tasks like object recognition and tracking.

Brief Description of the Project:

This project utilizes K-means clustering to segment an image into different regions based on color similarity. The process begins by converting the image from BGR to RGB format and then reshaping it into a 2D array where each pixel is represented by its RGB values. By applying K-means clustering, the image is divided into K clusters, each representing a color segment.

The segmented image is then processed to isolate a specific cluster of interest, and a mask is created to highlight the pixels belonging to that cluster. Further, the masked image is converted to the HSV color space for more precise color-based segmentation. The project identifies contours within the masked image to detect the object of interest. Finally, a bounding rectangle is drawn around the detected object to highlight its position within the image.

Algorithms:

- Image Segmentation
- K Means Clustering
- Image Color Space Conversion
- Contour Detection

Explanation about the algorithm in brief:

Image Segmentation:

In computer vision, image segmentation is the process of partitioning an image into multiple segments. The goal of segmenting an image is to change the representation of an image into something that is more meaningful and easier to analyze. It is usually used for locating objects and creating boundaries.
It is not a great idea to process an entire image because many parts in an image may not contain any useful information. Therefore, by segmenting the image, we can make use of only the important segments for processing.

An image is basically a set of given pixels. In image segmentation, pixels which have similar attributes are grouped together. Image segmentation creates a pixel-wise mask for objects in an image which gives us a more comprehensive and granular understanding of the object.

K Means Clustering Algorithm:
K Means is a clustering algorithm. Clustering algorithms are unsupervised algorithms which means that there is no labelled data available. It is used to identify different classes or clusters in the given data based on how similar the data is. Data points in the same group are more similar to other data points in that same group than those in other groups.

K-means clustering is one of the most commonly used clustering algorithms. Here, **k** represents the number of clusters.

Let's see how does K-means clustering work –

1. Choose the number of clusters you want to find which is k.
2. Randomly assign the data points to any of the k clusters.
3. Then calculate the center of the clusters.
4. Calculate the distance of the data points from the centers of each of the clusters.
5. Depending on the distance of each data point from the cluster, reassign the data points to the nearest clusters.
6. Again, calculate the new cluster center.
7. Repeat steps 4,5 and 6 till data points don't change the clusters, or till we reach the assigned number of iterations.

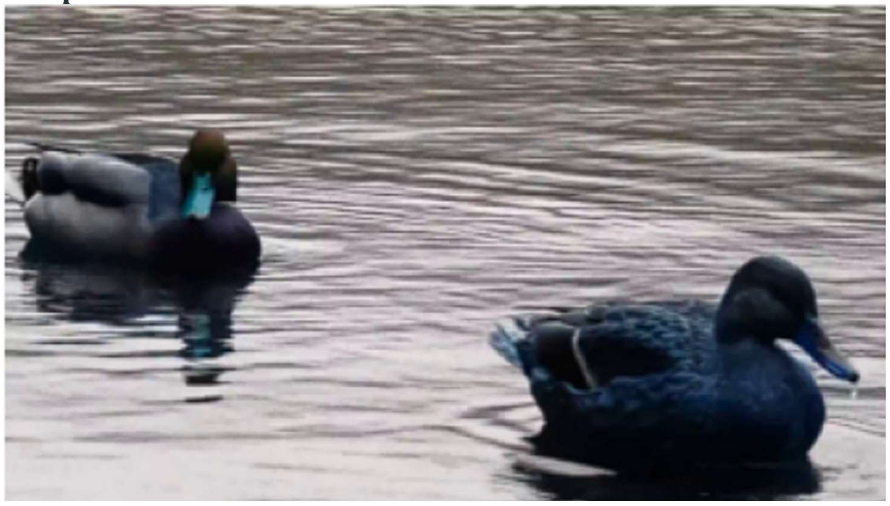
Image Color Space Conversion:
To enhance the segmentation process, the image is converted from BGR (used by OpenCV) to RGB (standard for image processing). After segmentation, the image is also converted to the HSV color space, which separates color information (hue) from intensity information (saturation and value), allowing for more precise color-based filtering.

Contour Detection:
Contours are curves joining all continuous points along a boundary with the same color or intensity. After creating a binary mask to isolate the desired color cluster, contour detection algorithms are applied to identify the boundaries of objects within the mask. The largest contour, assumed to be the object of interest, is then isolated, and a bounding rectangle is drawn around it to mark its position in the image.

Coding and Snapshots:

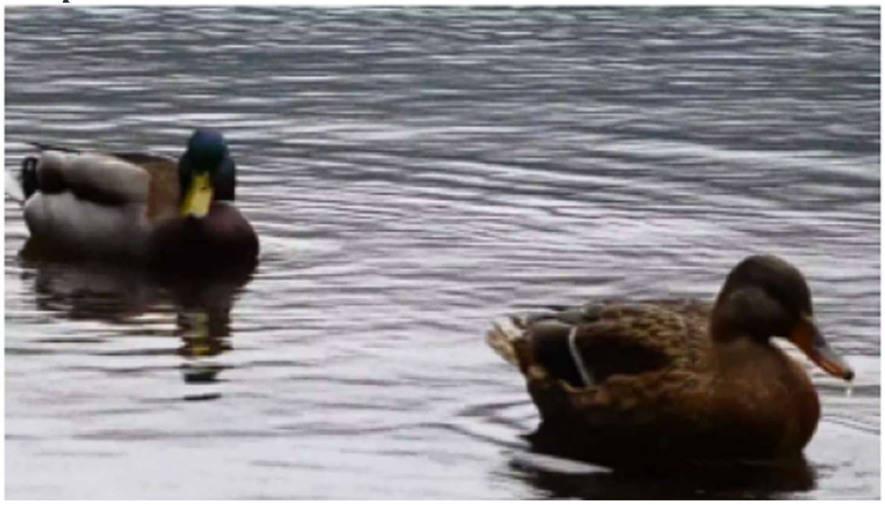
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import cv2

# Having a look at the image
image=cv2.imread('duck.jpg')
plt.imshow(image);
plt.axis('off') ;
```

Output:



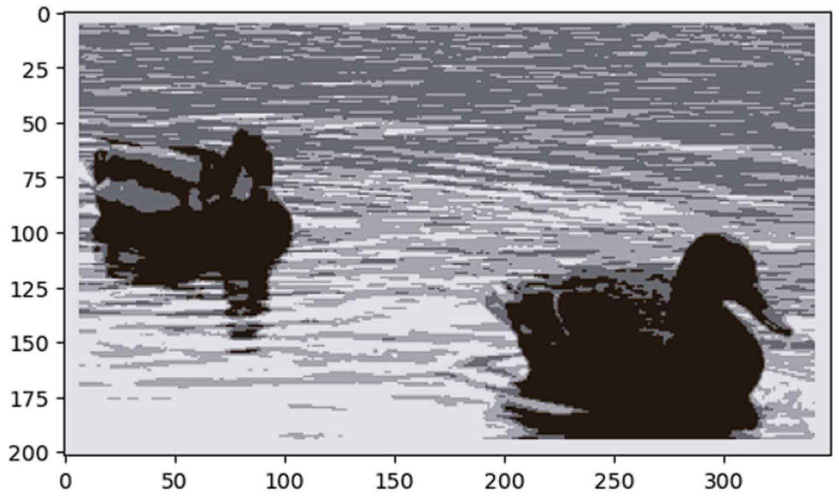
```
# Changing the image from BGR to RGB
image=cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
plt.imshow(image);
plt.axis('off') ;
```

Output:



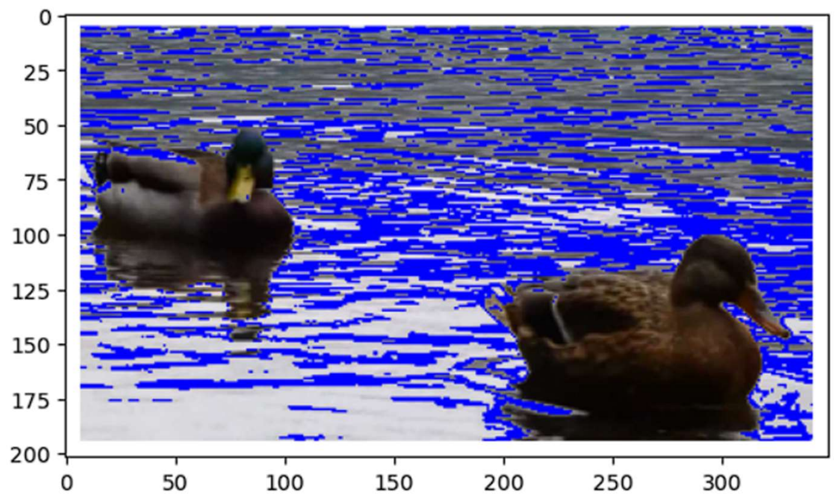
```
# Reshaping the image to 2d
pixel_vals = image.reshape((image.shape[0]*image.shape[1],3))
pixel_vals = np.float32(pixel_vals)
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2)

k = 4
retval, labels, centers = cv2.kmeans(pixel_vals, k, None, criteria, 10,
cv2.KMEANS_RANDOM_CENTERS)
centers = np.uint8(centers)
segmented_data = centers[labels.flatten()]
segmented_image = segmented_data.reshape((image.shape))
labels_reshape = labels.reshape(image.shape[0], image.shape[1])
plt.imshow(segmented_image);
```

Output:



```
BLUE = (0,0,255)
RED = (255,0,0)
cluster = 3
masked_image = np.copy(image)
masked_image[labels_reshape == cluster] = [BLUE]
cv2.imwrite('images/masked.jpg',masked_image)
plt.imshow(masked_image);
```

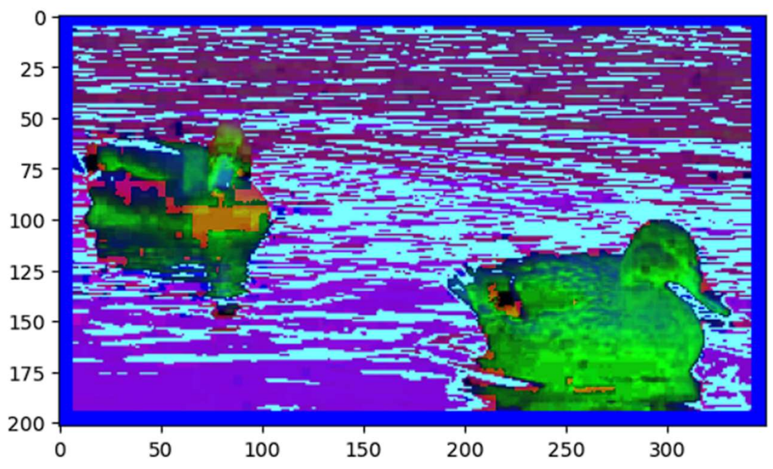
Output:



```
hsv_img = cv2.cvtColor(masked_image, cv2.COLOR_RGB2HSV)
plt.imshow(hsv_img)
```

Output:

```
<matplotlib.image.AxesImage at 0x1fb53808310>
```



```
blue = np.uint8([[[255,0,0]]])
hsv_blue = cv2.cvtColor(blue,cv2.COLOR_BGR2HSV)
print(hsv_blue)
```

Output:

```
[[[120 255 255]]]
```

```
lower_blue = (120, 255, 250)
upper_blue = (120, 255, 255)
COLOR_MIN = np.array([lower_blue], np.uint8)
COLOR_MAX = np.array([upper_blue], np.uint8)

# Threshold the HSV image to get only blue colors
frame_threshed = cv2.inRange(hsv_img, COLOR_MIN, COLOR_MAX)

# Find contours
```

```
contours, hierarchy = cv2.findContours(frame_threshed, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

# Sort contours by area and get the top 3
contours = sorted(contours, key=cv2.contourArea, reverse=True)[:3]

# Draw rectangles around the top 3 contours
pad_w = 3
pad_h = 4
pad_x = 3
pad_y = 4

for cnt in contours:
    x, y, w, h = cv2.boundingRect(cnt)
    cv2.rectangle(image, (x - pad_x, y - pad_y), (x + w + pad_w, y + h + pad_h), (255, 0, 0),
2)

plt.imshow(image)
plt.axis('off')
plt.show()
```

Output: