INTERNSHIP TRAINING REPORT
ON
“MACHINE LEARNING USING PYTHON PROGRAMMING”
AT

NATIONAL INSTITUTE OF ELECTRONICS AND INFORMATION TECHNOLOGY ROPAR

SUBMITTED BY

RIYA THAKUR & SHAGUN THAUR

Students of

JAWAHAR LAL NEHRU GOVERNMENT ENGINEERING COLLEGE SUNDERNAGAR

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

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE ENGINEERING (AI & ML)

DURATION FROM : **3RD JUNE 2024** TO **12 JULY 2024**

Contents

4 CHAPTER 1: AI ECOSYSTEM

1.1 INTRODUCTION

An AI ecosystem is a **network of people, organizations, and technologies that work together to advance the field of artificial intelligence (AI**). This can include AI researchers and engineers, AI-focused companies, academic institutions, and government agencies. The goal of an AI ecosystem is to create an environment that fosters innovation, collaboration, and the sharing of ideas and resources

An AI ecosystem is a dynamic and evolving environment that plays a critical role in driving innovation and progress in the field of AI.The goal of an AI ecosystem is to foster innovation, collaboration, and the sharing of ideas and resources to drive progress in AI.The AI ecosystem is a dynamic network of organizations, individuals, and technologies that work together to advance artificial intelligence (AI). Key components include AI researchers and engineers, AI-focused companies and startups, academic institutions, government agencies, cloud computing providers, and organizations that develop AI tools and infrastructure.

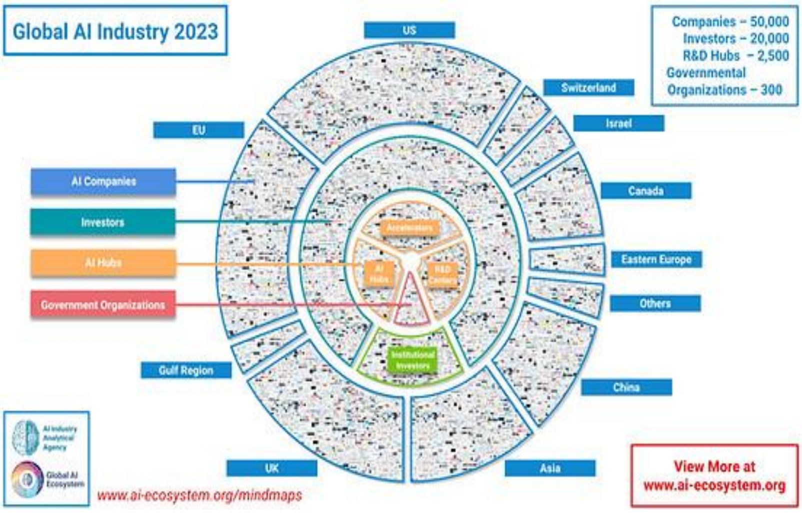


Figure 1:global AI ecosystem

The goal of the AI ecosystem is to foster innovation, collaboration, and the sharing of ideas and resources to drive progress in AI. This includes early algorithms, expert systems, neural networks, and recent breakthroughs in areas like natural language processing and computer vision.

While AI offers many benefits, there are also concerns about potential misuse and overreliance on the technology that need to be addressed. TheAI ecosystem plays a critical role in shaping the development and application of AI technologies.

1.2 COMPONENTS OF AI ECOSYSREM

The AI ecosystem refers to the **set of knowledge, tools, and practices related to artificial intelligence** (AI) that have become integral to scientific research.

The key components of the AI ecosystem include:

Cloud Infrastructure: Cloud and data center providers that offer the connectivity and computing resources to power AI systems.

Data Collectors: Entities that collect, curate, annotate, and label data to improve its quality and utility for AI.

Model Developers: Those who build, test, and train AI models, including foundation models, extended models, and specialized models.

Application Developers: Entities that design and develop AI-powered solutions for users.

AI Building Tools and Platforms: Platforms that allow users to build their own AI solutions with little to no expertise.

Integrators: Entities that bring together AI solutions and incumbent digital systems to work as a whole.

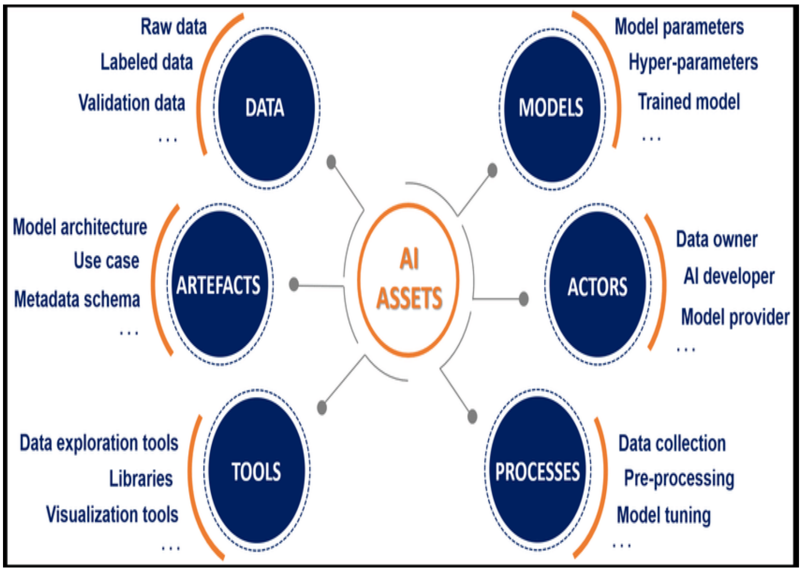


Figure 2: assets of AI ecosystem

Deployers and Users: Those that deploy and use AI products and services.

Processors, Memory, and Storage: Hardware components like CPUs, GPUs, FPGAs, and ASICs that power AI systems.

Natural Language Processing (NLP): A key technology that enables AI to understand and generate human language.

Machine Learning and Deep Learning: Foundational AI techniques for training models to perform tasks.

Computer Vision: AI capabilities for processing and understanding visual information.

Predictive Analytics: AI-powered capabilities for forecasting and decision-making.

The AI ecosystem is a complex, interconnected system of these various components that must work together to enable the development and deployment of effective AI solutions.

1.3 WORKING OF AI ECOSYSTEM

AI ecosystems are complex, interconnected systems that leverage artificial intelligence technologies to drive innovation, optimize processes, and create value. Here's a high-level overview of how AI ecosystems work:

- **AI Model Development**

AI ecosystems rely on vast amounts of data from various sources, such as IoT sensors, social media, and enterprise systems. This data needs to be collected, cleaned, and preprocessed before it can be used for AI applications.

- **Data Collection and Preprocessing**

Data scientists and machine learning engineers use the preprocessed data to develop and train AI models using techniques like deep learning, natural language processing, and computer vision. These models are designed to perform specific tasks, such as predicting customer behavior, detecting fraud, or optimizing supply chains.

- **AI Model Deployment**

Once the AI models are trained, they need to be deployed into production environments where they can be used to make real-time decisions and take actions. This often involves integrating the models with existing enterprise systems and applications.

- **AI-Powered Applications**

AI-powered applications leverage the deployed AI models to deliver intelligent capabilities to end-users. These applications can be used across various domains, such as customer service, marketing, sales, operations, and finance. Examples include chatbots, predictive maintenance systems, and personalized recommendation engines.

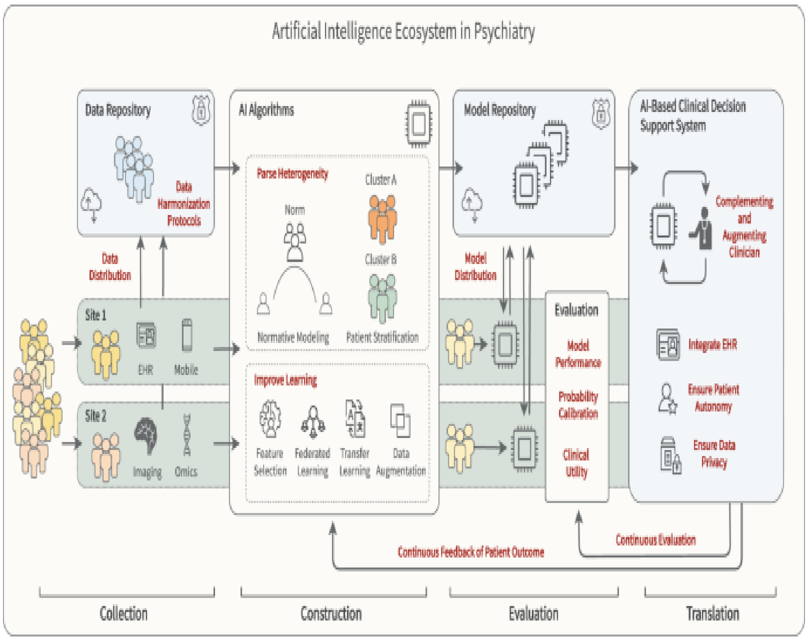


Figure 3:AI ecosystem in psychiatry

- **Continuous Learning and Improvement**

AI ecosystems are designed to learn and improve over time as they process more data and receive feedback from users. This involves continuously monitoring the performance of AI models, retraining them when necessary, and deploying updated versions to production.

- **Collaboration and Partnerships**

Building and maintaining an AI ecosystem requires collaboration among various stakeholders, including technology providers, domain experts, data scientists, and end-users. Partnerships with cloud providers, software vendors, and academic institutions can help organizations access the necessary resources and expertise to build and scale their AI capabilities.

- **Governance and Ethics**

As AI ecosystems become more complex and pervasive, it's crucial to have robust governance frameworks in place to ensure that they are being used in an ethical, transparent, and accountable manner. This includes establishing guidelines for data privacy, model fairness, and human oversight.

By leveraging these key components, organizations can build and deploy AI ecosystems that deliver tangible business value while mitigating risks and challenges. However, building and maintaining an AI ecosystem is not an easy task and requires significant investment in people, processes, and technology.

1.4 TOOLS OF AI ECOSYSTEM

An AI tool is a software application that uses artificial intelligence algorithms to perform specific tasks and solve problems.

The AI ecosystem encompasses a wide range of tools and technologies that enable the development and deployment of artificial intelligence systems. Some key components of the AI ecosystem include:

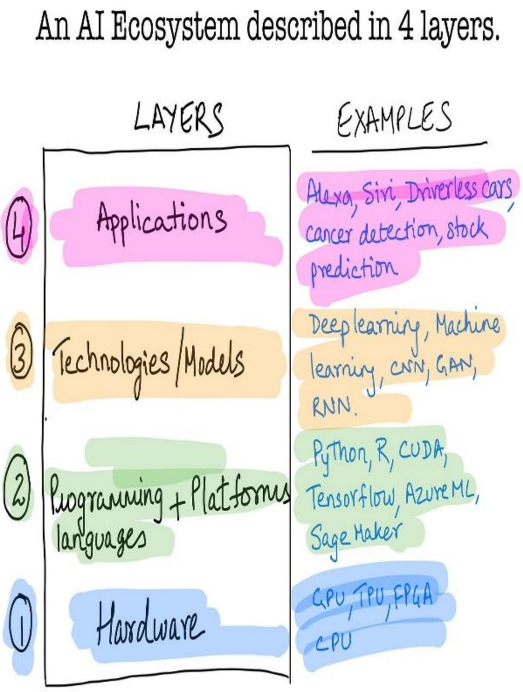

An AI Ecosystem described in 4 layers.
LAYERS
EXAMPLES
4
Applications
Alexa, Siri, Driverless cars, cancer detection, stock prediction
3
Technologies/Models
Deep learning, Machine learning, CNN, GAN, RNN.
2
Programming + Platforms languages
Python, R, CUDA, Tensorflow, Azure ML, Sage Maker
1
Hardware
GPU, TPU, FPGA, CPU

Figure 4: description of AI using 4 layers

AI Foundation Models: These are large, pre-trained AI models that can be adapted to a variety of tasks, such as natural language processing, computer vision, and scientific research. The Department of Energy (DOE) is actively developing AI foundation models for its science, energy, and security missions.

AI Development Platforms: Tools like the ecosystem.AI platform provide low-code, easy-to-use environments for building, deploying, and managing AI-powered applications without extensive coding knowledge.

AI Analytics and Insights: Platforms like the Global AI Ecosystem offer interactive databases, analytics, and insights on the AI industry, including information on companies, investors, research hubs, and AI leaders.

AI Collaboration and Community: The Global AI Ecosystem also provides collaborative environments like Slack and Telegram groups where AI professionals can connect, share knowledge, and participate in the broader AI community.

AI Talent Marketplaces: The AI Jobs Marketplace connects AI experts with companies seeking AI talent, facilitating the growth of the AI workforce.

Overall, the AI ecosystem encompasses a diverse set of tools, platforms, and resources that enable the development, deployment, and adoption of AI technologies across various industries and applications.

1.5 BENEFITS OF AI ECOSYSTEM

- **Improved Business Efficiency**

AI helps ensure 24-hour service availability and delivers consistent performance throughout the day, leading to improved business efficiency. AI-powered systems and automation can free human workers to focus on more creative and strategic activities, increasing productivity and enabling faster decisions.

- **Handling Big Data**

AI can process and analyze large volumes of real-time data quickly and accurately, helping businesses and data scientists extract valuable insights and make better decisions. AI drives predictive analytics, allowing algorithms to analyze historical data to anticipate future trends, behavior, and outcomes.

- **Reducing Human Error**

AI is improving autonomous systems like vehicles, drones, and robots, helping them operate better in hazardous environments and perform tasks without human intervention. In healthcare, AI can assist in diagnosing diseases, predicting outbreaks, and discovering potential drug candidates.

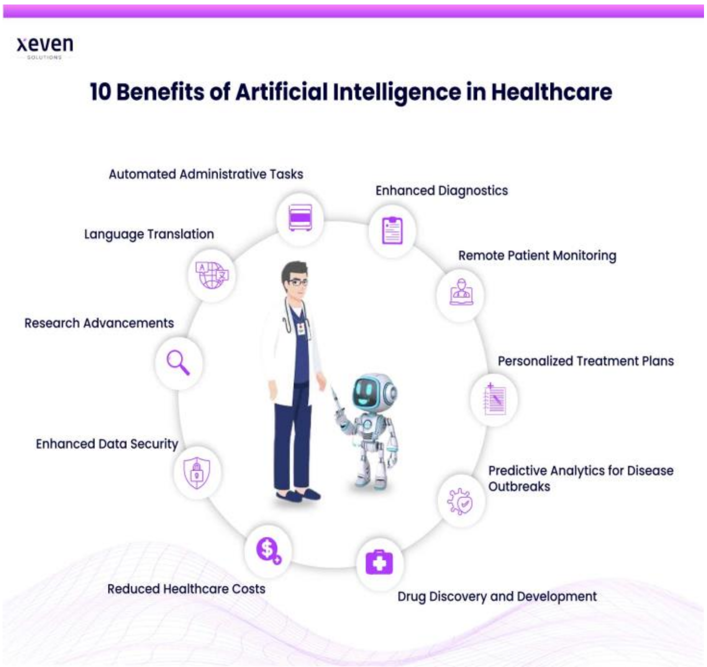

xeven
SOLUTIONS
10 Benefits of Artificial Intelligence in Healthcare
Automated Administrative Tasks
Enhanced Diagnostics
Language Translation
Remote Patient Monitoring
Research Advancements
Personalized Treatment Plans
Enhanced Data Security
Predictive Analytics for Disease Outbreaks
Reduced Healthcare Costs
Drug Discovery and Development

Figure 5: benefits of AI ecosystem

- **Fraud Detection and Cybersecurity**

AI can detect unusual patterns and behaviors, aiding in fraud detection and enhancing cybersecurity measures by identifying potential threats.

- **Tokenization of Assets and Data**

AI tokens help tokenize assets and data securely and transparently, allowing owners to have full control and monetize their data and assets.

- **New Funding Opportunities**

AI tokens have created novel means to fund AI projects and technology advancements through token sales, ICOs, crowdfunding, and other fundraising methods.

- **Enhanced User Experience**

AI tokens use machine learning to analyse past data and offer personalized recommendations and services tailored to users' requirements and preferences.

However, the adoption of AI also comes with challenges such as high costs, job displacement, and ethical conerns that need to be addressed.

5 CHAPTER 2 :PYTHON PROGRAMMING

2.1 INRTODUCTION

Early programming languages were highly specialized, relying on mathematical notation and similarly obscure syntax. Throughout the 20th century, research in compiler theory led to the creation of high-level programming languages, which use a more accessible syntax to communicate instructions. The evolution of programming languages has continued to the present day, with the introduction of languages like Python, Java, C#, and Swift, each addressing specific needs and use cases in the rapidly advancing field of computer science.

Figure 6:GUIDO VAN ROSSUM (inventor of python)

PYTHON…

It is a widely used general-purpose, high level programming language.

It was created by **Guido van Rossum in 1991** and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently. There are two major Python versions: **Python 2 and Python 3**. Both are quite different.

It is used for:

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

2.2 WHY PYTHON???

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

Some silent features of python are…

- **Simplicity and Readability**

  Python's clean and intuitive syntax, with its emphasis on readability, makes it an excellent choice for beginners and experienced developers alike. The language's English-like structure and lack of complex constructs contribute to its ease of use.

- **Large Standard Library and Ecosystem**

  Python comes with a vast standard library that provides a wide range of functionality, reducing the need for developers to write boilerplate code or rely on external libraries for common tasks. Additionally, Python has a large and active community that has developed thousands of libraries and frameworks for various domains, such as data analysis, machine learning, and scientific computing.

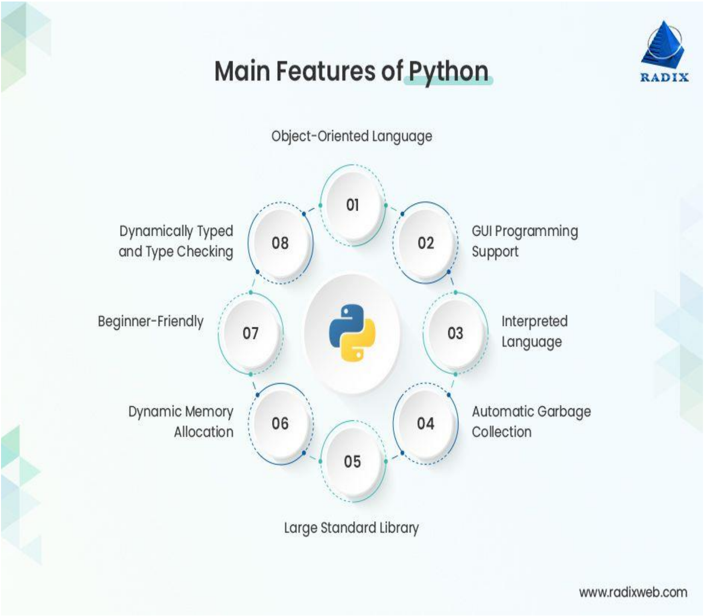


Figure 7:feature of python

- **Versatility**

  Python is a general-purpose language that can be used for a wide range of applications, from web development and data analysis to machine learning and artificial intelligence. This versatility makes it a valuable tool in many industries and domains.

- **Interpreted Nature**

  Python is an interpreted language, which means it can execute code directly without the need for a separate compilation step. This makes it faster to write and test programs, especially during the development phase.

- **Data Science and Machine Learning Libraries**

Python has become a dominant language in the fields of data science and machine learning, with popular libraries like NumPy, Pandas, and TensorFlow providing powerful tools for data manipulation, analysis, and model development.

- **Hands-On Learning**

  Many online courses and tutorials offer hands-on experience with Python for data science and AI, allowing learners to practice and apply what they learn through interactive exercises and projects using tools like Jupyter Notebook . These features, combined with a large and active community, make Python a popular choice for both beginners and experienced programmers looking to work in the fields of data science and artificial intelligence.

2.3 APPLICATIONS OF PYTHON

Python plays an essential role in the development of many applications like:

- **Web Applications**

  We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautifulSoup, Feedparser, etc. One of Python web-framework named Django is used on Instagram. Python provides many useful frameworks, and these are given below:

  - Django and Pyramid framework(Use for heavy applications)
  - Flask and Bottle (Micro-framework)
  - Plone and Django CMS (Advance Content management)

- **Desktop GUI Applications**

The GUI stands for the Graphical User Interface, which provides a smooth interaction to any application. Python provides a Tk GUI library to develop a user interface

- **Software Development**

Python is useful for the software development process. It works as a support language and can be used to build control and management, testing, etc.

  - SCons is used to build control.
  - Buildbot and Apache Gumps are used for automated continuous compilation and testing.
  - Round or Trac for bug tracking and project management.

- **Scientific and Numeric**

Python language is the most suitable language for Artificial intelligence or machine learning. It consists of many scientific and mathematical libraries, which makes easy to solve complex calculations.Implementing machine learning algorithms require complex mathematical calculation.

- **Python libraries**

Python has many libraries for scientific and numeric such as Numpy, Pandas, Scipy, Scikit-learn, etc. If you have some basic knowledge of Python, you need to import libraries on the top of the code. Few popular frameworks of machine libraries are given below.

  - SciPy
  - Scikit-learn
  - NumPy
  - Pandas
  - Matplotlib

- **Business Applications**

Business Applications differ from standard applications. E-commerce and ERP are an example of a business application. This kind of application requires extensively, scalability and readability, and Python provides all these features.

Oddo is an example of the all-in-one Python-based application which offers a range of business applications. Python provides a Tryton platform which is used to develop the business application.

- **Enterprise Applications**

Python can be used to create applications that can be used within an Enterprise or an Organization. Some real-time applications are

- Tryton
- Picalo, etc.

- **Image Processing Application**

Python contains many libraries that are used to work with the image. The image can be manipulated according to our requirements. Some libraries of image processing are given below.

- OpenCV
- Pillow
- SimpleITK

2.4 PYTHON USING COLAB

2.4.1 INTRODUCTION TO COLAB

Google Colaboratory, or Colab, is an as-a-service version of Jupyter Notebook that enables you to write and execute Python code through your browser. Jupyter Notebook is a free, open source creation from the Jupyter Project.

With Colab we can harness the full power of popular Python libraries to analyze and visualize data. The code cell below uses numpy to generate some random data, and uses matplotlib to visualize it. To edit the code, just click the cell and start editing

Google Colab is a free, cloud-based Jupyter notebook environment that allows you to write, run, and share Python code directly in your web browser.

What is Google Colab?

Google Colab is a **version of the popular Jupyter** Notebook that runs in the cloud, allowing you to write and execute Python code without needing to install any software on your local machine.

Colab notebooks combine executable code, text, images, and other rich media in a single document that can be easily shared and collaborated on.

Colab provides access to free GPU and TPU resources that can significantly speed up machine learning and deep learning model training.

Getting Started with Colab

To start using Colab, simply go to the Colab website at colab.research.google.com and create a new notebook.

Colab notebooks contain code cells where you can write and execute Python code, as well as text cells for adding documentation and explanations.

You can add new code and text cells, run code, and share your Colab notebooks just like a regular Jupyter Notebook.

Overall, Google Colab provides an easy-to-use, cloud-based environment for writing, running, and sharing Python code, making it an excellent tool for students, data scientists, and AI researchers alike.

Getting Started

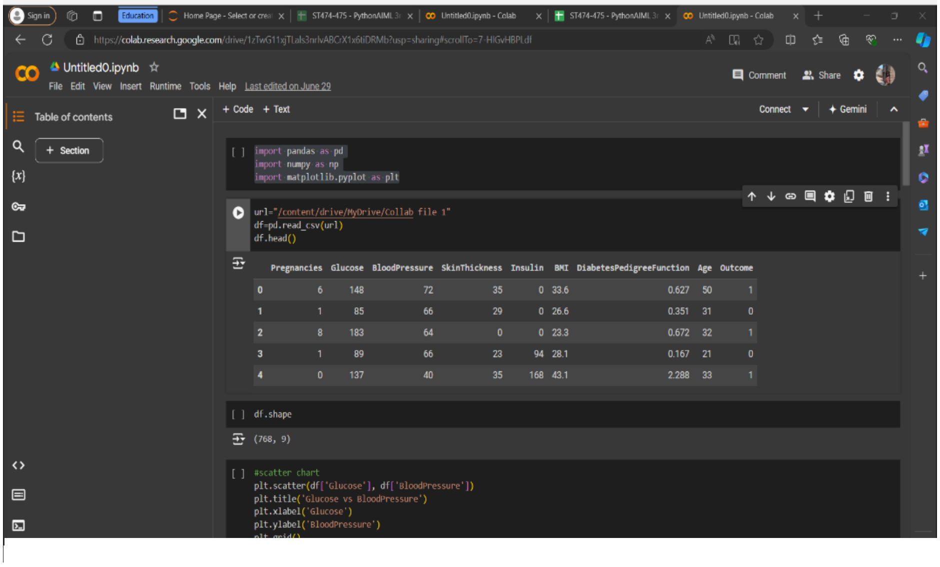
- Sign In: To start using Google Colab, you need to log in with your Google account credentials. Go to the Colab website at colab.research.google.com.
- Create a New Notebook: Once logged in, click on the "File" menu and select "New notebook" to create a new notebook.

PLOTTING DIFFERENT TYPES OF GRAPH IN COLAB USING RAW DATA

For plotting any graph the first two steps are always same.

- Importing the packages

- Using the correct code for plotting graph



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

```
url="/content/drive/MyDrive/Collab file 1"
df=pd.read_csv(url)
df.head()
```

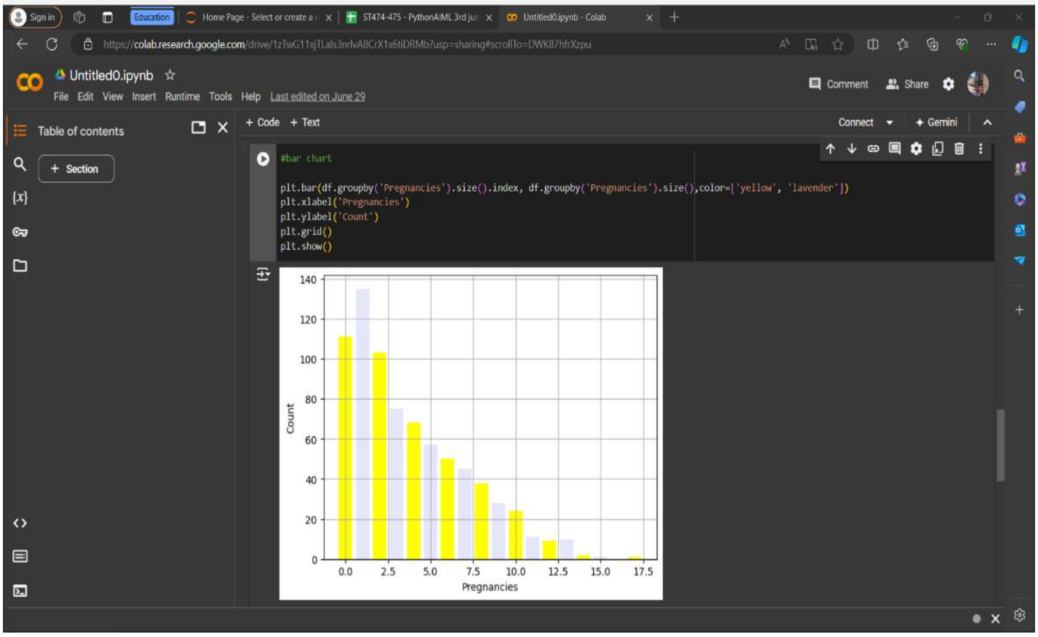
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  DiabetesPedigreeFunction  Age  Outcome
0            6      148             72             35        0  33.6                     0.627   50        1
1            1       85             66             29        0  26.6                     0.351   31        0
2            8      183             64              0        0  23.3                     0.672   32        1
3            1       89             66             23       94  28.1                     0.167   21        0
4            0      137             40             35      168  43.1                     2.288   33        1
```

```
df.shape
```

```
(768, 9)
```

```
#scatter chart
plt.scatter(df['Glucose'], df['BloodPressure'])
plt.title('Glucose vs BloodPressure')
plt.xlabel('Glucose')
plt.ylabel('BloodPressure')
plt.grid()
```

colab 1:importing the packages



```
#bar chart

plt.bar(df.groupby('Pregnancies').size().index, df.groupby('Pregnancies').size(),color=['yellow', 'lavender'])
plt.xlabel('Pregnancies')
plt.ylabel('Count')
plt.grid()
plt.show()
```

colab 2: plotting of bar graph

Features of google colab

- **Free Access to GPUs and TPUs**

Colab provides free access to powerful GPUs and TPUs, which are essential for machine learning and deep learning tasks.

- **No Setup Required:**

Colab runs in the cloud, eliminating the need for users to set up their own development environment.

- **Collaborative Editing**

Multiple users can work on the same Colab notebook simultaneously, making it a useful tool for collaborative projects.

- **Integration with Google Drive**

Colab is integrated with Google Drive, allowing users to save their work directly to their Google Drive account.

- **Pre-installed Libraries**

Colab comes pre-installed with popular Python libraries such as TensorFlow, PyTorch, and Matplotlib, among others.

- **Writing and Executing Code**

You can write and execute Python code in the notebook cells. The results are displayed within the notebook.

- **Uploading Files**

You can upload files to Colab using the file browser or by writing code to upload files. For example, you can use the files.upload() function in Python.

- **Sharing Notebooks**

Colab notebooks can be easily shared by providing a link to the notebook. Others can view or edit the code in real-time.

- **Additional Resources**
    - Support for Other Languages

In addition to Python, Colab also supports other languages like R and Julia through its notebook environment.

- Accessibility

  Colab works with most major browsers and is most thoroughly tested with the latest versions of Chrome, Firefox, and Safari.

2.4.2 LIMITATIONS OF COLAB

Overall, Google Colab is a versatile and accessible platform for Python coding, particularly useful for machine learning, data science, and education, thanks to its cloud-based nature, free access to powerful computing resources, and collaborative features. But it also have some limitations such as:

- **Resource Limits**

While Colab provides free access to powerful computing resources, there are usage limits to ensure that resources are available to as many users as possible.

- **Restricted Activities**

Certain activities such as file hosting, media serving, and cryptocurrency mining are disallowed to prevent abuse of the free resources.

2.4.3 PYTHON USING JUPYTER NOTEBOOK

What is Jupyter Notebook?

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, visualizations, and rich text. It was originally developed as the IPython Notebook, but was later expanded to support multiple programming languages, including Python, R, and Julia.

Jupyter Notebook provides an interactive computing environment that combines code execution, rich text, mathematics, plots, and multimedia in a single document.

Getting Started with Jupyter Notebook…

- To use Jupyter Notebook, you first need to install it. The easiest way is to install the Anaconda distribution of Python, which includes Jupyter Notebook.
- Once installed, you can start the Jupyter Notebook server by running the jupyter notebook command in your terminal or command prompt.
- This will open a web browser window with the Jupyter Notebook dashboard, where you can create new notebooks or open existing ones.

Using Jupyter Notebook

A Jupyter Notebook consists of a sequence of cells, which can contain code, text (using Markdown formatting), equations (using LaTeX), visualizations, and more. You can execute the code in each cell by clicking the "Run" button or pressing Shift+Enter.

Jupyter Notebook provides a rich set of features, including tab completion, code highlighting, and the ability to include rich media such as images, videos, and interactive plots.

Notebooks can be saved, shared, and even published as HTML or PDF documents.

Benefits of Jupyter Notebook

- Allows for interactive, exploratory programming and data analysis
- Supports a wide range of programming languages and scientific computing libraries
- Enables easy sharing and collaboration on computational work
- Provides a unified environment for writing code, documenting, and visualizing results
- Overall, Jupyter Notebook is a powerful tool for data analysis, machine learning, and scientific computing, making it a popular choice among researchers, data scientists, and educators.

6 CHAPTER 3: NUMPY & PANDAS

3.1 INTRODUCTION TO NUMPY

What is NumPy?

NumPy is a powerful open-source Python library for numerical computing. It provides support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

NumPy was created in 2005 by Travis Oliphant as a combination of the Numeric and Numarray libraries, and has since become a fundamental package for scientific computing with Python.

Key features of NumPy…

- Efficient N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities
- Can be used as an efficient multi-dimensional container of arbitrary data

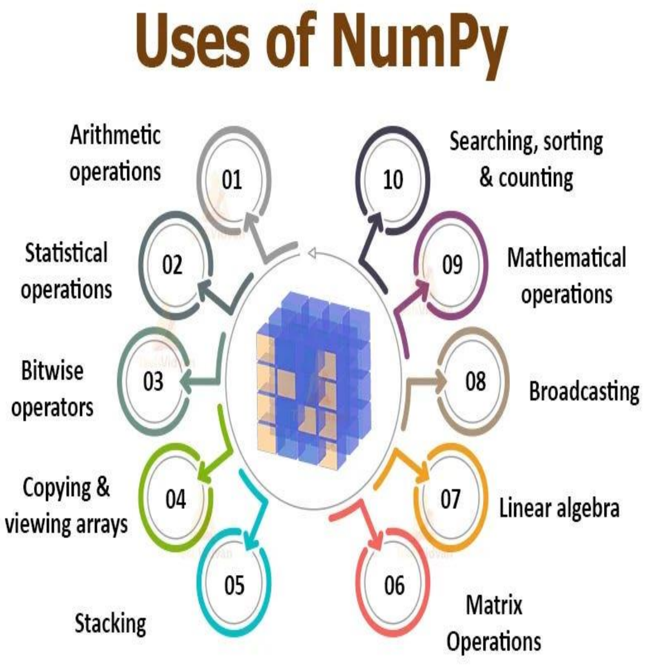


Figure 8:uses of numpy

Why Use NumPy?

- NumPy arrays are much faster and more memory-efficient than standard Python lists for numerical operations.
- NumPy is heavily used in data science, machine learning, image processing, and other scientific computing domains.
- NumPy integrates well with other popular libraries like Pandas, SciPy, and Matplotlib.
- It provides a wide range of mathematical functions and routines for working with arrays.

Overall, NumPy is an essential library for scientific computing and data analysis in Python, providing powerful array objects and a rich set of supporting functions.

Advantages of NumPy…

- **Powerful N-dimensional arrays**

Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today**.**

- **Numerical computing tools**

NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

- **Open source**

Distributed under a liberal BSD license, NumPy is developed and maintained publicly on GitHub by a vibrant, responsive, and diverse community.

- **Interoperable**

NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.

- **Performant**

The core of NumPy is well-optimized C code. Enjoy the flexibility of Python with the speed of compiled code.

- **Easy to use**

NumPy's high level syntax makes it accessible and productive for programmers from any background or experience level.

3.1.1 HOW PANDA AND NUMPY ARE RELATED TO EACH OTHER

The relation between Pandas and NumPy is fundamental and essential for data manipulation and analysis in Python. Here are the key points:

Integration and Dependence

- **Pandas is Built on NumPy**

Pandas is built on top of NumPy, leveraging its high-performance numerical computations to handle large datasets efficiently. This means that Pandas uses NumPy arrays as its underlying data structure.

- **NumPy is a Dependency**

NumPy is a dependency of Pandas, meaning that Pandas relies on NumPy for its core functionality. This integration allows Pandas to perform vectorized operations efficiently.

Similarities and Differences

- **Data Structures**

Both libraries provide data structures for efficient data handling. NumPy offers N-dimensional arrays, while Pandas provides DataFrames and Series, which are optimized for structured data.

- **Operations**

  Both libraries support various operations, but Pandas is designed to handle structured data with labels, making it more suitable for data analysis and manipulation tasks.

- **Vectorized Operations**

Both libraries support vectorized operations, which allow for efficient processing of large datasets by applying the same operation to all elements of an array or DataFrame.

Use Cases

- **Data Analysis**

Pandas is widely used for data analysis, data cleaning, and data manipulation. It provides tools for handling missing data, grouping data, and performing various statistical operations.

- **Machine Learning**

Pandas is often used in machine learning workflows to prepare data for models by cleaning, transforming, and preprocessing data.

- **Scientific Computing**

NumPy is used in scientific computing for tasks such as linear algebra, Fourier transforms, and random number generation

3.2 INRODUCTION TO PANDAS

What is Pandas?

Pandas is a powerful open-source Python library for data manipulation and analysis.It provides easy-to-use data structures and data analysis tools for working with structured (tabular, multidimensional, potentially heterogeneous) and time series data.**Pandas is built on top of NumPy** and provides high-performance, easy-to-use data structures and data analysis tools for Python.

Key Features of Pandas…

- Efficient DataFrame object with default and customized indexing
- Tools for loading data into in-memory data objects from different file formats
- Data alignment and integrated handling of missing data
- Reshaping and pivoting of datasets
- Label-based slicing, indexing and subsetting of large data sets
- Columns can be inserted and deleted from DataFrame
- Group by functionality for performing split-apply-combine operations on data sets
- High performance merging and joining of data
- Time Series functionality

Why Use Pandas?

Pandas is a powerful and flexible library that simplifies the process of data manipulation, analysis, and exploration in Python, making it an essential tool for data scientists, analysts, and researchers.

- Pandas is fast and efficient for working with large datasets
- It provides powerful data manipulation and analysis capabilities
- Pandas integrates well with other Python libraries like NumPy, Matplotlib, and Scikit-learn
- It is widely used in data science, finance, economics, statistics, and more.
- Powerful Data Manipulation and Analysis
- Pandas provides flexible and efficient data structures like DataFrames and Series for working with tabular, multidimensional, and potentially heterogeneous data.
- It offers a wide range of functions and methods for cleaning, transforming, and analyzing data, making it a powerful tool for data science and machine learning tasks.
- Efficient Handling of Large Datasets
- Intuitive and Readable Syntax
- Pandas has a clear and concise syntax that is easy to read and understand, even for beginners.
- This makes Pandas code more maintainable and collaborative, as it's easier for others to follow and build upon.

Pandas is a powerful and flexible library that simplifies the process of data manipulation, analysis, and exploration in Python, making it an essential tool for data scientists, analysts, and researchers.

Getting Started with Pandas

- Install Pandas using pip: pip install pandas
- Import Pandas in your Python script: import pandas as pd
- Create a DataFrame from a list, dictionary, or CSV file
- Explore and manipulate the data using Pandas' functions and methods

Overall, Pandas is an essential tool for data scientists working with structured data in Python, providing powerful yet easy-to-use data manipulation and analysis capabilities.

ADVANTAGES OF PANDAS…

- **Efficient Data Handling**

Pandas provides highly optimized data structures like DataFrames and Series that can efficiently handle large datasets, making it a powerful tool for working with big data.

- **Flexible Data Representation**

Pandas offers flexible and streamlined data representation, making it easier to analyze and understand complex data.

- **Less Coding, More Work**

Pandas allows you to accomplish more with less code compared to using just base Python, saving time and effort.

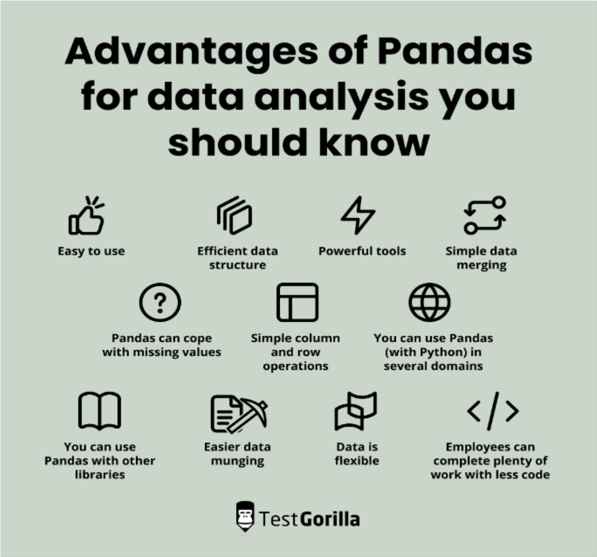

Advantages of Pandas for data analysis you should know
Easy to use
Efficient data structure
Powerful tools
Simple data merging
Pandas can cope with missing values
Simple column and row operations
You can use Pandas (with Python) in several domains
You can use Pandas with other libraries
Easier data munging
Data is flexible
Employees can complete plenty of work with less code
TestGorilla

Figure 9:advantages of pandas

- **Extensive Feature Set**

Pandas provides a wide range of features and functions for data manipulation, filtering, grouping, and analysis, giving you powerful tools to work with your data.

- **Integration with Other Libraries**

Pandas integrates seamlessly with other popular Python libraries like NumPy, SciPy, Matplotlib, and Scikit-learn, enabling you to build robust data analysis and machine learning pipelines.

- **Intuitive Syntax**

Pandas has a clear and concise syntax that is relatively easy to learn and understand, especially for those familiar with Excel-like data manipulation.

- **Versatile I/O**

Pandas can read and write data from/to a variety of formats, including CSV, Excel, SQL databases, and more, making it a versatile tool for data scientists and analysts.

- **Time Series Support**

Pandas provides robust functionality for working with time series data, including tools for handling time-stamped data, resampling, and calculating rolling statistics.

- **Vibrant Community**

Pandas has a large and active user community, providing ample resources, tutorials, and support for both beginners and experienced users.

7 CHAPTER 4: MACHINE LEARNING

4.1 INTRODUCTION TO M.L.

What is machine learning?

It is define as a subset of AI (artificial intelligence). Machine Learning is the ability of a machine a to learn something new on the basis of its past experience and the environment around it. focuses on the using data and algorithms to enable AI to imitate the way that humans learn, gradually improving its accuracy.

4.1.1 STEPS INVOLVE IN MACHINE LEARNING ARE

- **Collecting data**

machines initially learn from the data that you give them. It is of the utmost importance to collect reliable data so that your machine learning model can find the correct patterns. The quality of the data that you feed to the machine will determine how accurate your model is. If you have incorrect or outdated data, you will have wrong outcomes or predictions which are not relevant.

- **Preparing the Data:**

After you have your data, you have to prepare it. You can do this by :

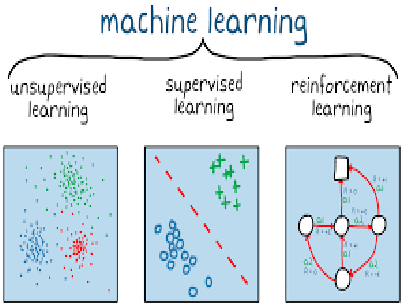
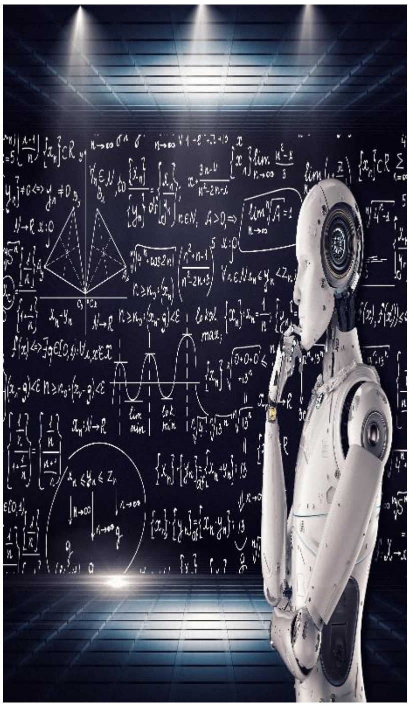


Figure 10: types of ml algo

  - Cleaning the data to remove unwanted data, missing values, rows, and columns, duplicate Putting together all the data you have and randomizing it. This helps make sure that data is evenly distributed.
  - values, data type conversion, etc. You might even have to restructure the dataset and change the rows and columns or index of rows and columns.
  - Splitting the cleaned data into two sets - a training set and a testing set. The training set is the set your model learns from. A testing set is used to check the accuracy of your model after training.



- **Choosing a Model**:

A machine learning model determines the output you get after running a machine learning algorithm on the collected data. It is important to choose a model which is relevant to the task at hand. Over the years, scientists and engineers developed various models suited for different tasks like speech recognition, image recognition, prediction, etc. Apart from this, you also have to see if your model is suited for numerical or categorical data and choose accordingly.

- **Training the Model**:

Training is the most important step in machine learning. In training, you pass the prepared data to your machine learning model to find patterns and make predictions. It results in the model learning from the data so that it can accomplish the task set. Over time, with training, the model gets better at predicting.

- **Evaluating the Model:**

Evaluation of the model is done by testing the performance of the model on previously unseen data. The unseen data used is the testing set that you split our data into earlier. If testing was done on the same data which is used for training, you will not get an accurate measure, as the model is already used to the data, and finds the same patterns in it.

- **Parameter Tuning**:

This step involves the accuracy of performance of the model. This is done by tuning the parameters present in your model. Parameters are the variables in the model that the programmer generally decides. At a particular value of your parameter, the accuracy will be the maximum. Parameter tuning refers to finding these values.

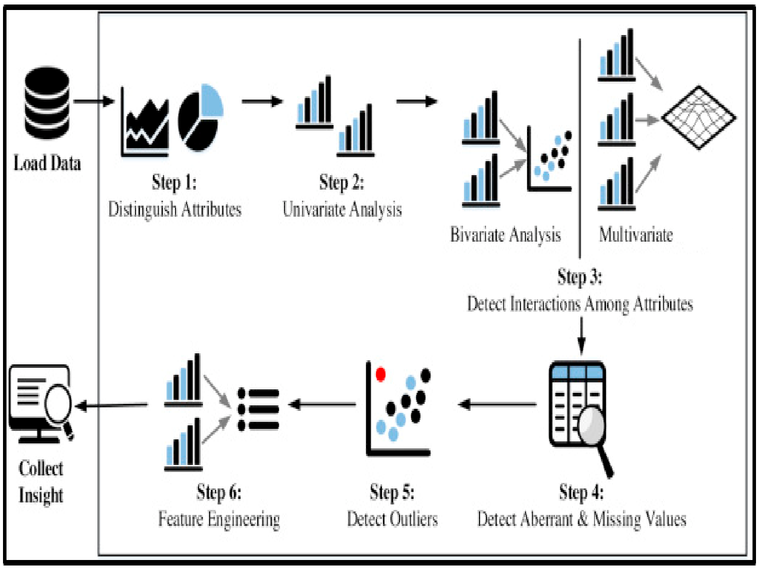


Figure 11:steps in machine learning process

- **Making Predictions**

In the end, you can use your model on unseen data to make predictions accurately.

4.2TYPES OF MACHINE LEARNING

Basically machine learning is divided into three main categories. These are:

4.2.1SUPERVISED LEARNING

What is Supervised Learning?

Supervised learning is a category of machine learning that uses labeled datasets to train algorithms to predict outcomes and recognize patterns. These algorithms are given labeled training to learn the relationship between the input and the outputs. Supervised machine learning algorithms make it easier for organizations to create complex models that can make accurate predictions.

The data used in supervised learning is labeled — meaning that it contains examples of both inputs (called features) and correct outputs (labels). The algorithms analyze a large dataset of these training pairs to infer what a desired output value would be when asked to make a prediction on new data.

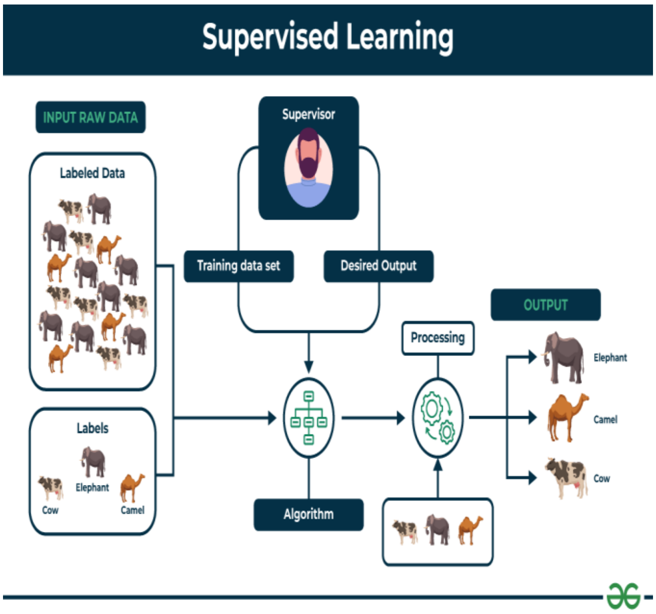


Figure 12:supervised learning representation

- **Types of supervised learning**

Supervised learning in machine learning is generally divided into two categories: classification and regression.

1.**Classification**:Classification algorithms are used to group data by predicting a categorical label or output variable based on the input data. Classification is used when output variables are categorical, meaning there are two or more classes.EXAMPLE:-One of the most common examples of classification algorithms in use is the spam filter in your email inbox. A supervised learning model is trained to predict whether an email is spam or not with a dataset that contains labeled examples of both spam and legitimate emails. The algorithm extracts information about each email, including the sender, the subject line, body copy, and more. It then uses these features and corresponding output labels to learn patterns and assign a score that indicates whether an email is real or spam.

2.**Regression:**Regression algorithms are used to predict a real or continuous value, where the algorithm detects a relationship between two or more variables. EXAMPLE:-A common example of a regression task might be predicting a salary based on work experience. For instance, a supervised learning algorithm would be fed inputs related to work experience (e.g., length of time, the industry or field, location, etc.) and the corresponding assigned salary amount. After the model is trained, it could be used to predict the average salary based on work experience.

4.2.2UNSUPERVISED MACHINE LEARNING

What is unsupervised learning?

Unsupervised learning in artificial intelligence is a type of machine learning that learns from data without human supervision. Unlike supervised learning, unsupervised machine learning models are given unlabeled data and allowed to discover patterns and insights without any explicit guidance or instruction.

How does unsupervised learning work?

the model is given raw, unlabeled data and has to infer its own rules and structure the information based on similarities, differences, and patterns without explicit instructions on how to work with each piece of data.

Unsupervised learning algorithms are better suited for more complex processing tasks, such as organizing large datasets into clusters. They are useful for identifying previously undetected patterns in data and can help identify features useful for categorizing data.

Imagine that you have a large dataset about weather. An unsupervised learning algorithm will go through the data and identify patterns in the data points. For instance, it might group data by temperature or similar weather patterns.

Unsupervised machine learning algorithms

there are three types of unsupervised learning tasks: clustering, association rules, and dimensionality reduction.

1.Clustering: Clustering is a technique for exploring raw, unlabeled data and breaking it down into groups (or clusters) based on similarities or differences. It is used in a variety of applications, including customer segmentation, fraud detection, and image analysis. Clustering algorithms split data into natural groups by finding similar structures or patterns in uncategorized

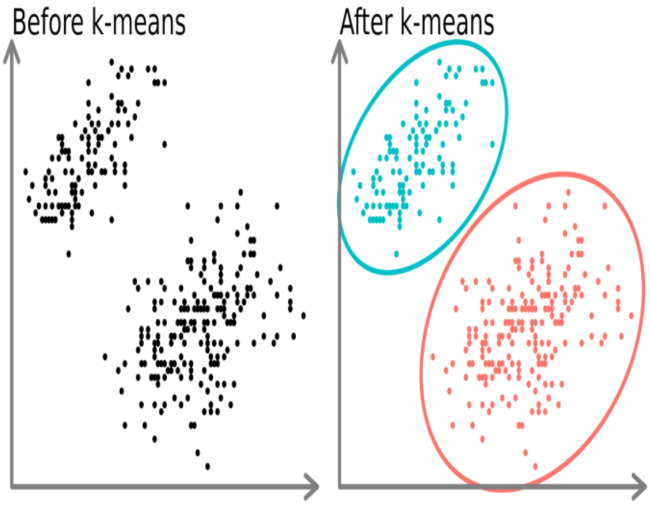


Figure 13:after and before data during clustering

data. **Exclusive clustering** ,**Overlapping clustering** ,**Hierarchical clustering** ,**Probabilistic clustering etc.** are different types of clustering.

2.Association: Association rule mining is a rule-based approach to reveal interesting relationships between data points in large datasets. Unsupervised learning algorithms search for frequent if-then associations—also called rules—to discover correlations and co-occurrences within the data and the different connections between data objects. Association rules are also often used to organize medical datasets for clinical diagnoses. Using unsupervised machine learning and association rules can help doctors identify the probability of a specific diagnosis by comparing relationships between symptoms from past patient cases.

3.Dimensionality reduction: Dimensionality reduction is an unsupervised learning technique that reduces the number of features, or dimensions, in a dataset. More data is generally better for machine learning, but it can also make it more challenging to visualize the data.

Dimensionality reduction extracts important features from the dataset, reducing the number of irrelevant or random features present. This method uses principle component analysis (PCA) and singular value decomposition (SVD) algorithms to reduce the number of data inputs without compromising the integrity of the properties in the original data.

Real-world unsupervised learning examples

the most common use cases helping businesses explore large volumes of data quickly.

- **Anomaly detection:** Unsupervised clustering can process large datasets and discover data points that are atypical in a dataset.
- **Recommendation engines:** Using association rules, unsupervised machine learning can help explore transactional data to discover patterns or trends that can be used to drive personalized recommendations for online retailers.
- **Customer segmentation:** Unsupervised learning is also commonly used to generate buyer persona profiles by clustering customers' common traits or purchasing behaviors. These profiles can then be used to guide marketing and other business strategies.
- **Fraud detection:** Unsupervised learning is useful for anomaly detection, revealing unusual data points in datasets. These insights can help uncover events or behaviors that deviate from normal patterns in the data, revealing fraudulent transactions or unusual behavior like bot activity.
- **Natural language processing (NLP)**: Unsupervised learning is commonly used for various NLP applications, such as categorizing articles in news sections, text translation and classification, or speech recognition in conversational interfaces.
- **Genetic research:** Genetic clustering is another common unsupervised learning example. Hierarchical clustering algorithms are often used to analyze DNA patterns and reveal evolutionary relationships.

4.2.3REINFORCEMENT MACHINE LEARNING

What is reinforcement learning?

Reinforcement learning (RL) is a machine learning (ML) technique that trains software to make decisions to achieve the most optimal results. It mimics the trial-and-error learning process that humans use to achieve their goals. Software actions that work towards your goal are reinforced, while actions that detract from the goal are ignored.

RL algorithms use a reward-and-punishment paradigm as they process data. They learn from the feedback of each action and self-discover the best processing paths to achieve final outcomes. The algorithms are also capable of delayed gratification. The best overall strategy may require short-term sacrifices, so the best approach they discover may include some punishments or backtracking along the way. RL is a powerful method to help artificial intelligence (AI) systems achieve optimal outcomes in unseen environments.

What are the benefits of reinforcement learning?

There are many benefits to using reinforcement learning (RL). However, these three often stand out.

- **Excels in complex environments**

RL algorithms can be used in complex environments with many rules and dependencies. In the same environment, a human may not be capable of determining the best path to take, even with superior knowledge of the environment. Instead, model-free RL algorithms adapt quickly to continuously changing environments and find new strategies to optimize results.

- **Requires less human interaction**

In traditional ML algorithms, humans must label data pairs to direct the algorithm. When you use an RL algorithm, this isn't necessary. It learns by itself. At the same time, it offers mechanisms to integrate human feedback, allowing for systems that adapt to human preferences, expertise, and corrections.

- **Optimizes for long-term goals**

RL inherently focuses on long-term reward maximization, which makes it apt for scenarios where actions have prolonged consequences. It is particularly well-suited for real-world situations where feedback isn't immediately available for every step, since it can learn from delayed rewards.

For example, decisions about energy consumption or storage might have long-term consequences. RL can be used to optimize long-term energy efficiency and cost. With appropriate architectures, RL agents can also generalize their learned strategies across similar but not identical tasks.

Example of RL are:

- **Marketing personalization**

RL can customize suggestions to individual users based on their interactions. This leads to more personalized experiences. For example, an application may display ads to a user based on some demographic information. With each ad interaction, the application learns which ads to display to the user to optimize product sales.

- **Optimization challenges**

Traditional optimization methods solve problems by evaluating and comparing possible solutions based on certain criteria. In contrast, RL introduces learning from interactions to find the best or close-to-best solutions over time.

- **Financial predictions**

The dynamics of financial markets are complex, with statistical properties that change over time. RL algorithms can optimize long-term returns by considering transaction costs and adapting to market shifts.

How does reinforcement learning work?

The learning process of reinforcement learning (RL) algorithms is similar to animal and human reinforcement learning in the field of behavioral psychology. For instance, a child may discover that they receive parental praise when they help a sibling or clean but receive negative reactions when they throw toys or yell. Soon, the child learns which combination of activities results in the end reward.

An RL algorithm mimics a similar learning process. It tries different activities to learn the associated negative and positive values to achieve the end reward outcome.

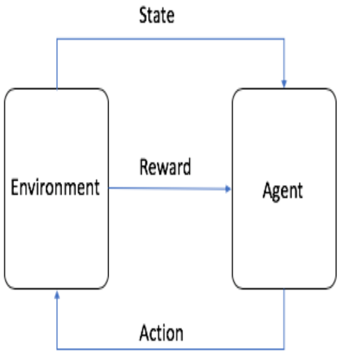


Figure 14: diagramatical representation of RL

Key concepts of RL are:

In reinforcement learning, there are a few key concepts to familiarize yourself with:

- The agent is the ML algorithm (or the autonomous system)
- The environment is the adaptive problem space with attributes such as variables, boundary values, rules, and valid actions
- The action is a step that the RL agent takes to navigate the environment
- The state is the environment at a given point in time
- The reward is the positive, negative, or zero value—in other words, the reward or punishment—for taking an action
- The cumulative reward is the sum of all rewards or the end value
- Algorithm basics

8 CHAPTER 5: PROJECT

TOPIC-SALARY PREDICTION

5.1INTRODUCTION ABOUT PROJECT

In this project, we try to predict the salary of employees on the basis of some elements like age , education etc.

The goal of this project is to build a machine learning model that can accurately predict a person's salary based on factors like their years of experience, education level, job title, and location.

The prediction of salary is done with the help of machine learning .

5.2DATASET

- **What do you mean by dataset?**

Data sets are effective tools for tracking and analyzing important information. Compiling related information into data sets can also help streamline analysis and evaluation processes

Our dataset contains information about the salaries of employees at a company. Each row represents a different employee, and the columns include information such as age, gender, education level, job title, years of experience, and salary.

Link of dataset: https://drive.google.com/file/d/1KA39AEPh3UWbicg0nDw4NLGZZpCIq9wz/view?usp=drivesdk

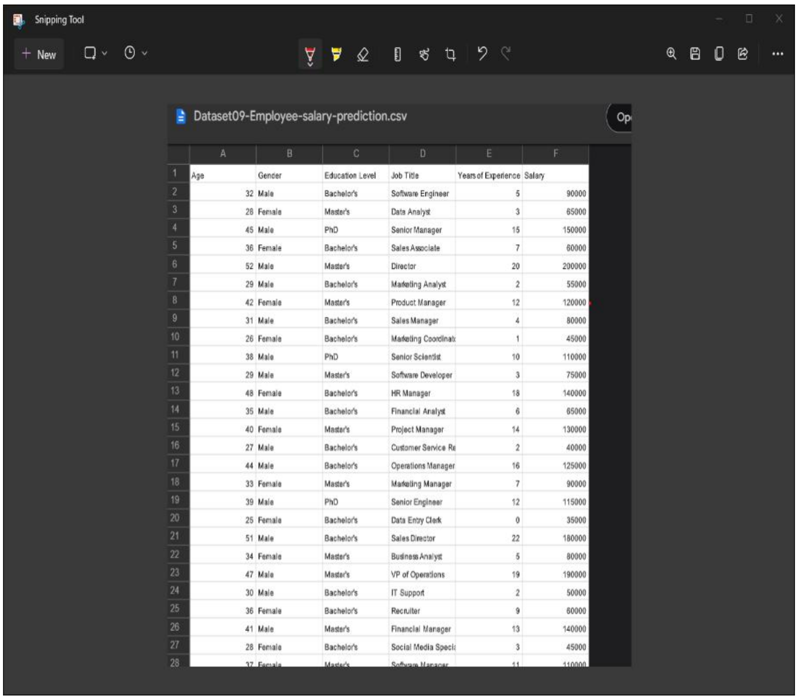
Columns:

Age:
This column represents the age of each employee in years. The values in this column are numeric.

Gender
This column contains the gender of each employee, which can be either male or female. The values in this column are categorical.

Education Level:
This column contains the educational level of each employee, which can be high school, bachelor's degree, master's degree, or PhD. The values in this column are categorical.

Job Title:
This column contains the job title of each employee. The job titles can vary depending on the company and may include positions such as manager, analyst, engineer, or administrator. The values in this column are categorical.



Dataset09-Employee-salary-prediction.csv

	A	B	C	D	E	F
1	Age	Gender	Education Level	Job Title	Years of Experience	Salary
2	32	Male	Bachelor's	Software Engineer	5	90000
3	28	Female	Master's	Data Analyst	3	65000
4	45	Male	PhD	Senior Manager	15	150000
5	36	Female	Bachelor's	Sales Associate	7	60000
6	52	Male	Master's	Director	20	200000
7	29	Male	Bachelor's	Marketing Analyst	2	55000
8	42	Female	Master's	Product Manager	12	120000
9	31	Male	Bachelor's	Sales Manager	4	80000
10	26	Female	Bachelor's	Marketing Coordinat	1	45000
11	38	Male	PhD	Senior Scientist	10	110000
12	29	Male	Master's	Software Developer	3	75000
13	48	Female	Bachelor's	HR Manager	18	140000
14	35	Male	Bachelor's	Financial Analyst	6	65000
15	40	Female	Master's	Project Manager	14	130000
16	27	Male	Bachelor's	Customer Service Re	2	40000
17	44	Male	Bachelor's	Operations Manager	16	125000
18	33	Female	Master's	Marketing Manager	7	90000
19	39	Male	PhD	Senior Engineer	12	115000
20	25	Female	Bachelor's	Data Entry Clerk	0	35000
21	51	Male	Bachelor's	Sales Director	22	180000
22	34	Female	Master's	Business Analyst	5	80000
23	47	Male	Master's	VP of Operations	19	190000
24	30	Male	Bachelor's	IT Support	2	50000
25	36	Female	Bachelor's	Recruiter	9	60000
26	41	Male	Master's	Financial Manager	13	140000
27	28	Female	Bachelor's	Social Media Speci	3	45000
28	37	Female	Master's	Software Manager	11	110000

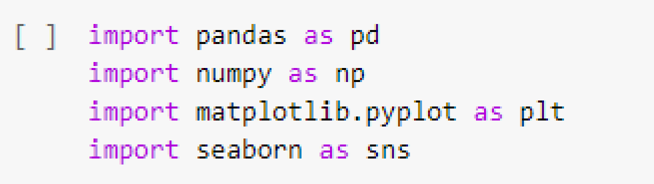
Figure 15: salary prediction dataset

Experience:
This column represents the number of years of work experience of each employee. The values in this column are numeric.

Salary:
This column represents the annual salary of each employee in US dollars. The values in this column are numeric and can vary depending on factors such as job title, years of experience, and education level.

5.3PRE-PROCESSING DATA

- **IMPORTING THE PACKAGES**

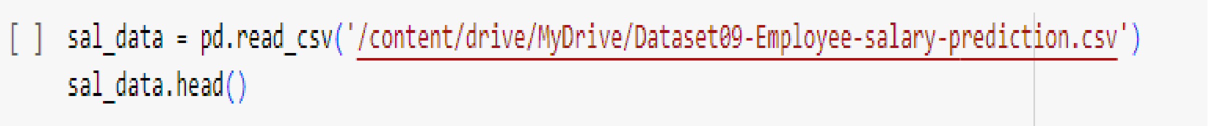


```
[ ]  import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
```

colab 3:importing packages

Here packages are imported; matplotlib and seaborn is used for data visualisation whereas panda and numpy are used for data manipilation.

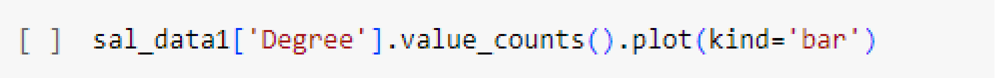
- **READING THE CSV FILE**



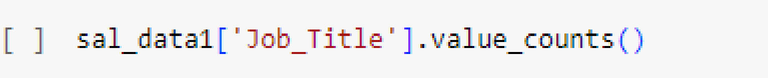
```
[ ]  sal_data = pd.read_csv('/content/drive/MyDrive/Dataset09-Employee-salary-prediction.csv')
     sal_data.head()
```

Reading the csv file help us to process the data.

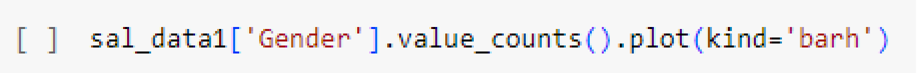
- **VISUALISING THE DATA**



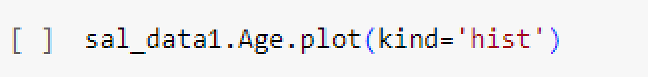
```
[ ]  sal_data1['Degree'].value_counts().plot(kind='bar')
```



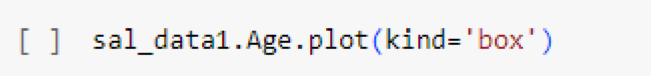
```
[ ]  sal_data1['Job_Title'].value_counts()
```



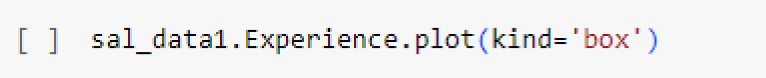
```
[ ]  sal_data1['Gender'].value_counts().plot(kind='barh')
```



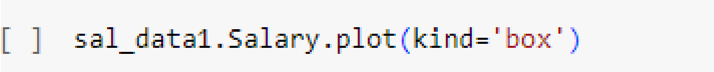
```
[ ]  sal_data1.Age.plot(kind='hist')
```



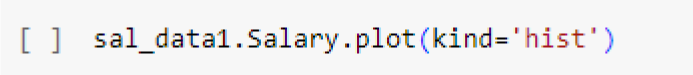
```
[ ]  sal_data1.Age.plot(kind='box')
```



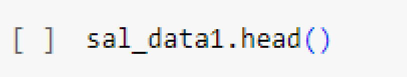
```
[ ]  sal_data1.Experience.plot(kind='box')
```



```
[ ]  sal_data1.Salary.plot(kind='box')
```



```
[ ]  sal_data1.Salary.plot(kind='hist')
```



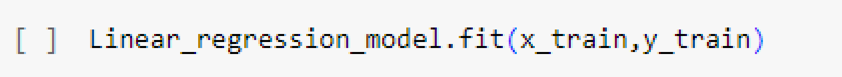
```
[ ]  sal_data1.head()
```

- **MACHINE LEARNING**

- model training model using linear regression

- **What is linear regression?**

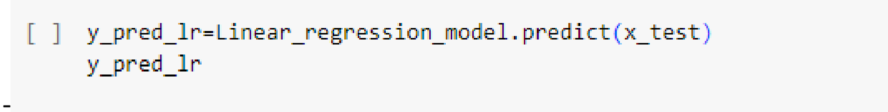
Linear Regression is a key data science tool for predicting continuous outcomes. This shows the linear relationship between tow variables(one is dependent and the other is independent)



```
[ ]  Linear_regression_model.fit(x_train,y_train)
```

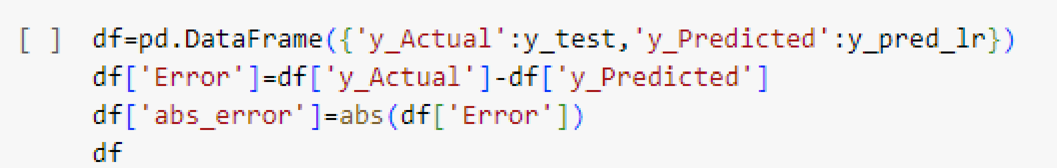
colab 4: using linear regression algo.

Here fit() helps with trainning the model by using trainned data.



```
[ ]  y_pred_lr=Linear_regression_model.predict(x_test)
     y_pred_lr
```

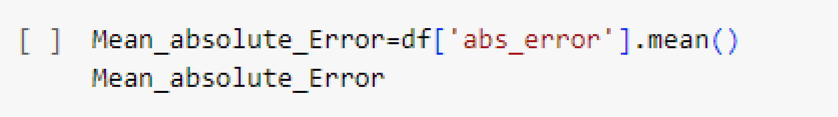
-



```
df=pd.DataFrame({'y_Actual':y_test,'y_Predicted':y_pred_lr})
df['Error']=df['y_Actual']-df['y_Predicted']
df['abs_error']=abs(df['Error'])
df
```

Here is the dataframe of the actual values and the predicted value.

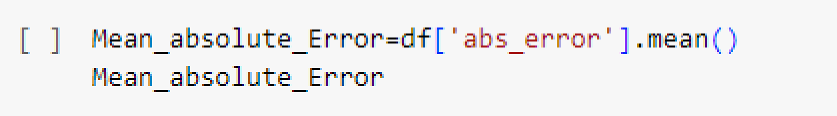
And we are also calculating the difference between these two values which are mentioned above.



```
Mean_absolute_Error=df['abs_error'].mean()
Mean_absolute_Error
```

Now calculating the absolute error.

- model evaluation



```
Mean_absolute_Error=df['abs_error'].mean()
Mean_absolute_Error
```

5.4FRONT END

- **HTM/CSS/JAVASRIPT**

- **WEB SERVER DEVELOPMENT USING FLASK**

5.5PROJECT WORKTHROUGH

PROJECT LIKS:

COLAB:

https://colab.research.google.com/drive/17jPS3wxZEVFJxGbUnZJyvoUxW6ksF7qW?usp=drive_link

GITHUP:

DATASET:

https://drive.google.com/file/d/1KA39AEPh3UWbicg0nDw4NLGZZpCIq9wz/view?usp=sharing

THANK YOU