**Personalized E-learning Course Recommendation System**

**ABSTRACT:**

A Personalized E-learning Course Recommendation System is designed to enhance the online learning experience by suggesting the most relevant courses to each learner based on their interests, learning history, goals, and skill level. This system uses artificial intelligence and machine learning techniques such as content-based filtering, collaborative filtering, and hybrid recommendation models to analyze learner profiles and course metadata. By personalizing recommendations, the system helps learners discover suitable courses that align with their learning objectives, thereby improving engagement, retention, and completion rates. It also provides an adaptive and scalable framework that can integrate with existing e-learning platforms, ensuring each learner receives a customized and efficient learning journey tailored to their unique needs and preferences.

**INTRODUCTION :**

In recent years, the growth of online education platforms such as Coursera, Udemy, and edX has revolutionized the way people acquire knowledge and skills. With thousands of courses available online, learners often face the challenge of choosing the most suitable courses that align with their interests, goals, and current skill levels. A Personalized E-learning Course Recommendation System addresses this issue by automatically analyzing learner behavior, preferences, and past learning activities to suggest relevant courses. By applying advanced machine learning and data analytics techniques, the system can predict user interests and provide recommendations that enhance the learning experience. Such personalization not only improves learner satisfaction but also increases engagement, course completion rates, and overall learning efficiency. The system bridges the gap between learners and the vast pool of online educational resources, ensuring that each individual receives recommendations tailored specifically to their needs.

**LITERATURE SURVEY**

 **Author:** Yehuda Koren, Robert Bell, and Chris Volinsky  
**Title:** *Matrix Factorization Techniques for Recommender Systems*  
**Description:**  
This paper introduces matrix factorization methods such as Singular Value Decomposition (SVD) for improving recommendation accuracy. It explains how latent factor models can capture hidden relationships between users and items, leading to better personalized recommendations. The approach has been widely used in e-learning systems to model user-course interactions and predict learner preferences effectively.

 **Author:** B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk  
**Title:** *Session-based Recommendations with Recurrent Neural Networks*  
**Description:**  
This research proposes using Recurrent Neural Networks (RNNs) to model user session behavior and make recommendations based on sequential interactions. In e-learning, this approach helps to understand the learning patterns of students over time and suggest the next best course or module dynamically.

 **Author:** Maciej Kula  
**Title:** *Metadata Embeddings for User and Item Cold-Start Recommendations*  
**Description:**  
The study focuses on using metadata embeddings (like text and tags) to handle cold-start problems in recommender systems where new users or courses lack sufficient interaction data. It is particularly useful in e-learning environments where new courses are frequently added, and immediate recommendations are required without prior learner feedback.

 **Author:** Francesco Ricci, Lior Rokach, and Bracha Shapira  
**Title:** *Recommender Systems Handbook*  
**Description:**  
This handbook provides a comprehensive overview of different recommendation system techniques, including collaborative filtering, content-based filtering, and hybrid models. It highlights the design principles, algorithms, and evaluation strategies for building effective personalized systems, making it a foundational reference for e-learning course recommendation development.

 **Author:** Robin Burke  
**Title:** *Hybrid Recommender Systems: Survey and Experiments*  
**Description:**  
Burke’s paper explores hybrid recommendation systems that combine multiple approaches to overcome the limitations of individual methods. In the context of e-learning, hybrid systems are effective in improving accuracy and diversity by combining user behavior data with course content features to deliver more balanced and reliable recommendations.

**EXISTING SYSTEM**

In the existing e-learning platforms, course recommendations are typically based on general popularity, user ratings, or manually selected categories rather than personalized insights. These systems often provide the same set of recommended courses to all learners without considering individual preferences, learning styles, or skill levels. As a result, learners may spend significant time searching for suitable courses that match their interests or career goals. Moreover, traditional recommendation methods rely heavily on static filtering or keyword matching, which fail to adapt dynamically to a learner’s evolving learning behavior or performance. Due to the lack of personalization and contextual understanding, existing systems are less effective in engaging users and often lead to lower course completion rates and learner satisfaction. This highlights the need for an intelligent, data-driven personalized recommendation system that can tailor course suggestions to individual needs**.**

**EXISTING SYSTEM DISADVANTAGES:**

 The recommendations are not personalized, offering the same courses to all users regardless of their interests or learning history.

 Cold-start problem occurs for new users and courses, as the system lacks sufficient interaction data to generate accurate suggestions.

 The system relies mainly on manual search or keyword matching, which limits the accuracy and relevance of recommendations.

 It fails to adapt dynamically to the learner’s changing preferences, performance, or learning goals over time.

**PROPOSED SYSTEM :**

The proposed Personalized E-learning Course Recommendation System aims to overcome the limitations of existing systems by providing intelligent, data-driven, and adaptive course suggestions tailored to each learner’s unique profile. It utilizes advanced machine learning and artificial intelligence techniques such as content-based filtering, collaborative filtering, and hybrid recommendation models to analyze user behavior, learning history, and course content. By combining user preferences, skills, and goals with course metadata, the system delivers personalized recommendations that match the learner’s interests and learning objectives. Additionally, it continuously learns from user interactions — such as enrollments, course completions, and feedback — to refine and improve future recommendations. This dynamic and adaptive approach enhances user engagement, increases course completion rates, and ensures that learners spend their time on courses most relevant to their educational and career growth. The system can be seamlessly integrated with existing e-learning platforms and supports scalability for large datasets and multiple users.

**PROPOSED SYSTEM ADVANTAGES:**

Personalized Learning Experience

Improved Accuracy

Enhanced User Engagement

**HARDWARE & SOFTWARE REQUIREMENTS:**

**HARD REQUIRMENTS :**

* System    :   i3 or above.
* Ram    :   4 GB.
* Hard Disk : 40 GB

**SOFTWARE REQUIRMENTS :**

* Operating system   : Windows8 or Above.
* Coding Language  : python

# SYSTEM STUDY

# FEASIBILITY STUDY

# The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

**TECHNICAL FEASIBILITY**

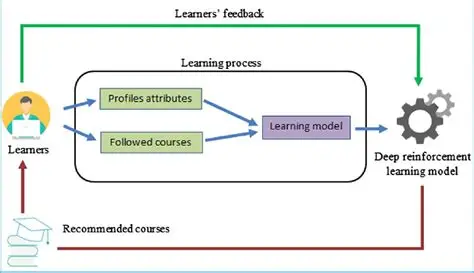
This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system

# SYSTEM DESIGN :

**4.1 SYSTEM ARCHITECTURE:**

****

**4.2UML DIAGRAM’S :**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

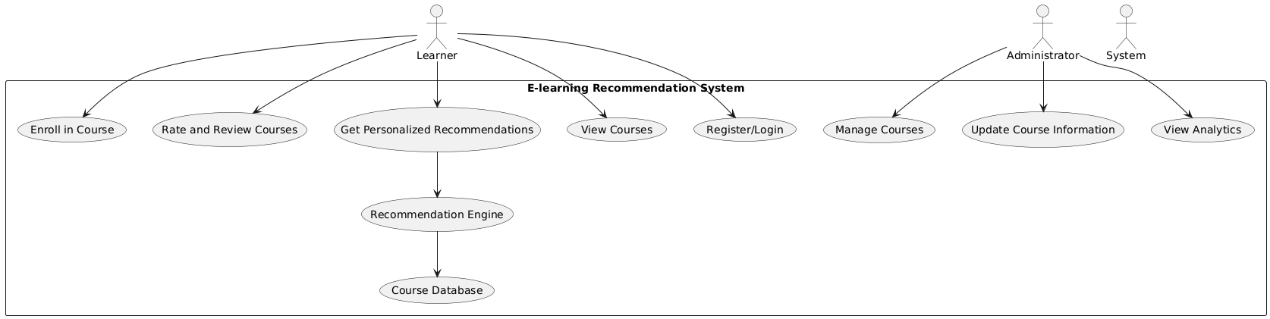
**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.

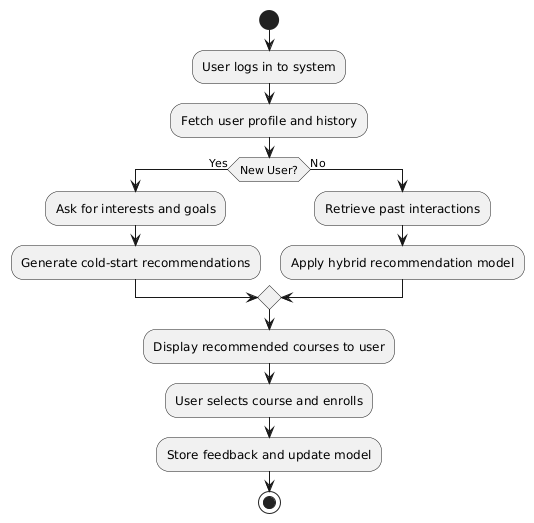
**USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



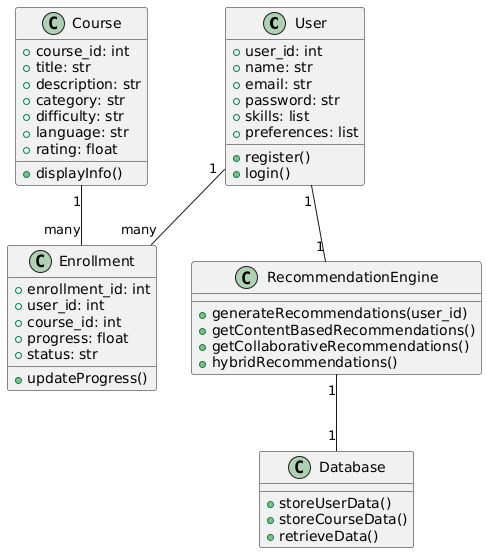
**ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



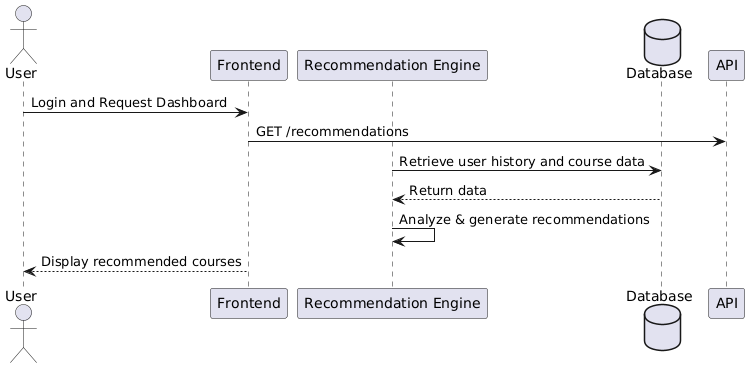
# CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



**SEQUENCE DIAGRAM:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



**INPUT AND OUTPUT DESIGN**

**INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

**OBJECTIVES**

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

**OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the
* Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

**IMPLEMENTATION:**

 **Requirement Analysis:**

* Identify user roles (Learner, Admin).
* Define functional and non-functional requirements.
* Decide tech stack (e.g., Django + React, or FastAPI + Flutter).

 **Database Design & Setup:**

* Create tables: users, courses, enrollments, events, ratings, tags.
* Set up relational DB (PostgreSQL) and caching (Redis).
* Configure object storage for course content (S3/local storage).

 **Backend API Development:**

* Implement authentication (JWT).
* Build endpoints: login/register, course list, recommendations, enrollments, event logging.
* Integrate admin APIs for managing courses and monitoring.

 **Data Collection & Preprocessing:**

* Collect user interactions (views, enrollments, completions, ratings).
* Clean course metadata and text (title, description, syllabus).
* Generate embeddings for courses and users (using SentenceTransformers or TF-IDF).

 **Recommendation Engine Development:**

* Implement content-based filtering using course embeddings and user profiles.
* Implement collaborative filtering (ALS or LightFM) for user-course interaction matrix.
* Combine approaches into a hybrid model; optionally add learning-to-rank.
* Handle cold-start users/courses using metadata and onboarding questions.

 **Frontend Development:**

* Design user dashboard to show personalized recommendations.
* Implement course search, filtering, and category browsing.
* Display “Why this course?” explanation for transparency.

 **Integration & Testing:**

* Connect frontend, backend APIs, and recommendation engine.
* Test recommendation accuracy and API responses.
* Conduct unit, integration, and system testing.

**SOFTWARE ENVIRONMENT :**

**What is Python :**

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

* + [Machine Learning](https://www.geeksforgeeks.org/machine-learning/)
  + GUI Applications (like Kivy, Tkinter, PyQt etc. )
  + Web frameworks like Django (used by YouTube, Instagram, Dropbox)
  + Image processing (like Opencv, Pillow)
  + Web scraping (like Scrapy, BeautifulSoup, Selenium)
  + Test frameworks
  + Multimedia

**Advantages of Python :-**

Let’s see how Python dominates over other languages.

**1. Extensive Libraries**

Python downloads with an extensive library and it *contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more.* So, we don’t have to write the complete code for that manually.

**2. Extensible**

As we have seen earlier, Python can be**extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

## 3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities**to our code in the other language.

## 4. Improved Productivity

The language’s simplicity and extensive libraries render programmers**more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

## 5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

When working with Java, you may have to create a class to print **‘Hello World’**. But in Python, just a print statement will do. It is also quite **easy to learn, understand,** and**code.** This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

## 7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory.** This further aids the readability of the code.

## 8. Object-Oriented

This language supports both the **procedural and object-oriented**programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

## 9. Free and Open-Source

Like we said earlier, Python is **freely available.** But not only can you[**download Python**](https://data-flair.training/blogs/install-python-windows/) for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

## 10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn’t the same with Python. Here, you need to**code only once**, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

## 11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

# Advantages of Python Over Other Languages :

## 1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don’t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

## 2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

**The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.**

## 3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and [**machine learning**](https://data-flair.training/blogs/machine-learning-tutorials-home/), automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

## Disadvantages of Python

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

#### 1. Speed Limitations

We have seen that Python code is executed line by line. But since [Python](https://www.python.org/) is interpreted, it often results in **slow execution**. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

#### 2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the **client-side**. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbonnelle**.

The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

#### 3. Design Restrictions

As you know, Python is **dynamically-typed**. This means that you don’t need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can**raise run-time errors**.

#### 4. Underdeveloped Database Access Layers

Compared to more widely used technologies like **JDBC (Java DataBase Connectivity)** and **ODBC (Open DataBase Connectivity)**, Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

#### 5. Simple

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

## History of Python : -

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python.Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners1, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it."Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

# What is Machine Learning : -

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of *building models of data*.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models *tunable parameters* that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain.Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

# Categories Of Machine Leaning :-

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

*Supervised learning* involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into *classification* tasks and *regression* tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

*Unsupervised learning* involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as *clustering* and *dimensionality reduction.* Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

# Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven’t surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, “to make decisions, based on data, with efficiency and scale”.

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can’t do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

# Challenges in Machines Learning :-

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

**Quality of data** − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

**Time-Consuming task** − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

**Lack of specialist persons** − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

**No clear objective for formulating business problems** − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

**Issue of overfitting & underfitting** − If the model is overfitting or underfitting, it cannot be represented well for the problem.

**Curse of dimensionality** − Another challenge ML model faces is too many features of data points. This can be a real hindrance.

**Difficulty in deployment** − Complexity of the ML model makes it quite difficult to be deployed in real life.

# Applications of Machines Learning :-

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML −

* Emotion analysis
* Sentiment analysis
* Error detection and prevention
* Weather forecasting and prediction
* Stock market analysis and forecasting
* Speech synthesis
* Speech recognition
* Customer segmentation
* Object recognition
* Fraud detection
* Fraud prevention
* Recommendation of products to customer in online shopping

# How to Start Learning Machine Learning?

Arthur Samuel coined the term **“Machine Learning”** in 1959 and defined it as a **“Field of study that gives computers the capability to learn without being explicitly programmed”.**

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to [Indeed](http://blog.indeed.com/2019/03/14/best-jobs-2019/), Machine Learning Engineer Is The Best Job of 2019 with a *344%* growth and an average base salary of **$146,085** per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let’s get started!!!

### How to start learning ML?

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

### Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don’t know these, never fear! You don’t need a Ph.D. degree in these topics to get started but you do need a basic understanding.

#### (a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

#### (b) Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!!  
Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

#### (c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is [Python](https://www.geeksforgeeks.org/python-programming-language/)! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as [Keras](https://keras.io/), [TensorFlow](https://www.tensorflow.org/), [Scikit-learn](https://scikit-learn.org/stable/), etc.

So if you want to learn ML, it’s best if you learn Python! You can do that using various online resources and courses such as [**Fork Python**](https://practice.geeksforgeeks.org/courses/fork-python) available Free on GeeksforGeeks.

### Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It’s best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

#### (a) Terminologies of Machine Learning

* **Model –**A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
* **Feature –**A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
* **Target (Label) –**A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
* **Training –**The idea is to give a set of inputs(features) and it’s expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
* **Prediction –**Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

#### (b) Types of Machine Learning

* **Supervised Learning –**This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
* **Unsupervised Learning –**This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
* **Semi-supervised Learning –**This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
* **Reinforcement Learning –**This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

### Advantages of Machine learning :-

#### 1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

#### 2. No human intervention needed (automation)

With ML, you don’t need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

#### 3. Continuous Improvement

As [**ML algorithms**](https://data-flair.training/blogs/machine-learning-algorithms/) gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

#### 4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

#### 5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

### Disadvantages of Machine Learning :-

#### 1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

#### 2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

#### 3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

#### 4. High error-susceptibility

[Machine Learning](https://en.wikipedia.org/wiki/Machine_learning) is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

**Python Development Steps : -**

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system.  
Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked.Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode.Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

* Print is now a function
* Views and iterators instead of lists
* The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
* There is only one integer type left, i.e. int. long is int as well.
* The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
* Text Vs. Data Instead Of Unicode Vs. 8-bit

**Purpose :-**

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

**Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

**Modules Used in Project :-**

**Tensorflow**

TensorFlow is a [free](https://en.wikipedia.org/wiki/Free_software) and [open-source](https://en.wikipedia.org/wiki/Open-source_software) [software library for dataflow and differentiable programming](https://en.wikipedia.org/wiki/Library_(computing)) across a range of tasks. It is a symbolic math library, and is also used for [machine learning](https://en.wikipedia.org/wiki/Machine_learning) applications such as [neural networks](https://en.wikipedia.org/wiki/Neural_networks). It is used for both research and production at [Google](https://en.wikipedia.org/wiki/Google).‍

TensorFlow was developed by the [Google Brain](https://en.wikipedia.org/wiki/Google_Brain) team for internal Google use. It was released under the [Apache 2.0](https://en.wikipedia.org/wiki/Apache_License) [open-source license](https://en.wikipedia.org/wiki/Open-source_license) on November 9, 2015.

**Numpy**

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. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and [IPython](http://ipython.org/) shells, the [Jupyter](http://jupyter.org/) Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the [sample plots](https://matplotlib.org/tutorials/introductory/sample_plots.html) and [thumbnail gallery](https://matplotlib.org/gallery/index.html).

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

**Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. **Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

**Install Python Step-by-Step in Windows and Mac :**

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

## How to Install Python on Windows and Mac :

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

**Note:** The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your **System Requirements**. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a **Windows 64-bit operating system**. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. [Download the Python Cheatsheet here.](https://myelearninghub.com/python-cheat-sheet/)The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

### Download the Correct version into the system

**Step 1:** Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: [https://www.python.org](https://www.python.org/)

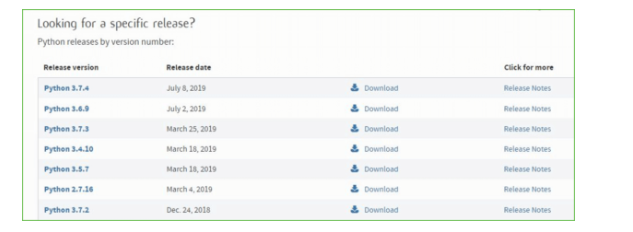


Now, check for the latest and the correct version for your operating system.

**Step 2:** Click on the Download Tab.

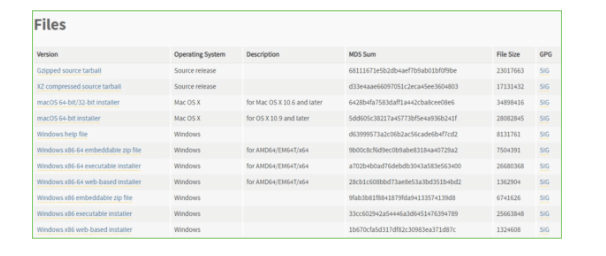
****

**Step 3:** You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

****

**Step 4:** Scroll down the page until you find the Files option.

**Step 5:** Here you see a different version of python along with the operating system.



• To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.

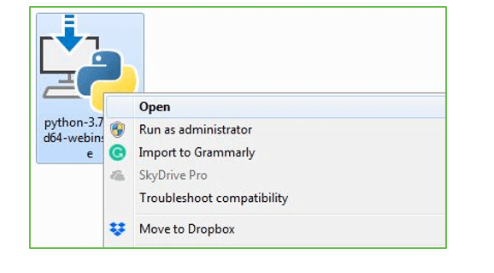
•To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

**Note:** To know the changes or updates that are made in the version you can click on the Release Note Option.

### Installation of Python

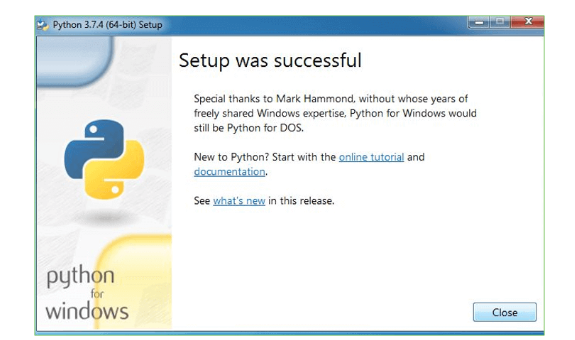
**Step 1:** Go to Download and Open the downloaded python version to carry out the installation process.



**Step 2:** Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.



**Step 3:** Click on Install NOW After the installation is successful. Click on Close.



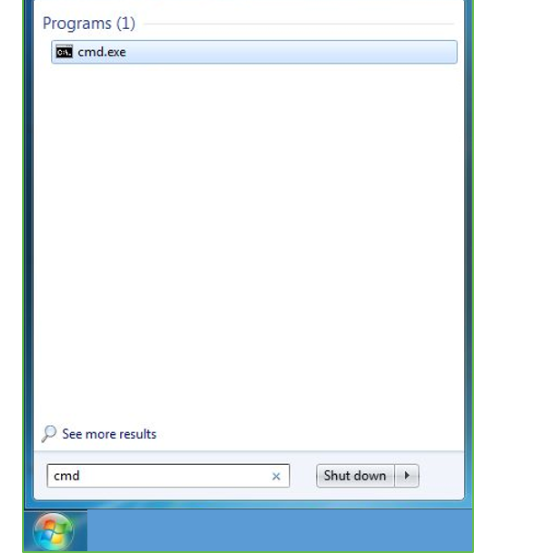
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

**Note:** The installation process might take a couple of minutes.

### Verify the Python Installation

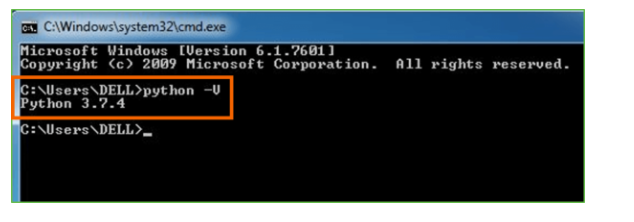
**Step 1:** Click on Start

**Step 2:** In the Windows Run Command, type “cmd”.



**Step 3:** Open the Command prompt option.

**Step 4:** Let us test whether the python is correctly installed. Type **python –V** and press Enter.



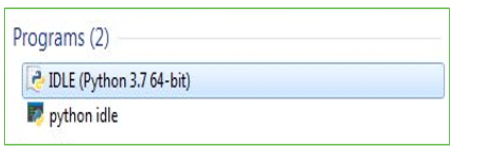
**Step 5:** You will get the answer as 3.7.4

**Note:** If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

### Check how the Python IDLE works

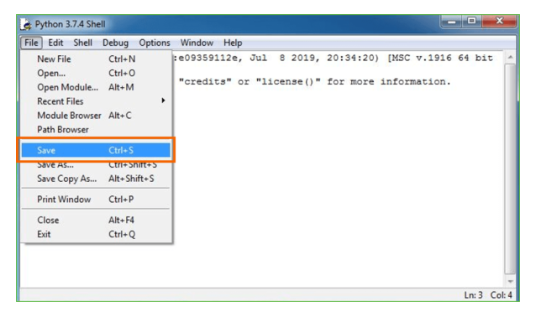
**Step 1:** Click on Start

**Step 2:** In the Windows Run command, type “python idle”.



**Step 3:** Click on IDLE (Python 3.7 64-bit) and launch the program

**Step 4:** To go ahead with working in IDLE you must first save the file. **Click on File > Click on Save**



**Step 5:** Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

**Step 6:** Now for e.g. **enter print**

**SAMPLE CODE:**

**#views code:**

from django.shortcuts import render, redirect

from django.contrib import messages

# Option A: Default User

from django.contrib.auth.models import User

import os

import re

import string   # ✅ Add this line

import pandas as pd

from django.shortcuts import render, redirect

from django.http import HttpResponse

import pandas as pd

import joblib

import os

import re

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

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, classification\_report

# Global variables

df = None

x\_train = x\_test = y\_train = y\_test = model = None

labels = ['Data Science','Marketing','Business','Programming','Design']

def home(request):

return render(request, 'home.html')

def admin\_login(request):

if request.method == 'POST':

uname = request.POST['username']

pwd = request.POST['password']

if uname == 'admin' and pwd == 'admin':

return redirect('admin\_dashboard')

else:

messages.error(request, "Invalid credentials")

return redirect('admin\_login')

return render(request, 'adminlogin.html')

def admin\_dashboard(request):

return render(request, 'admin\_dashboard.html')

import pandas as pd

from django.shortcuts import render, redirect

from django.contrib import messages

df = None  # Global DataFrame (for demo purposes; better to use sessions or DB in production)

def upload\_dataset(request):

global df

if request.method == 'POST' and request.FILES.get('dataset'):

try:

csv\_file = request.FILES['dataset']

df = pd.read\_csv(csv\_file)  # Directly read without saving

messages.success(request, "Dataset uploaded successfully!")

return redirect('admin\_dashboard')  # Redirect to dashboard after upload

except Exception as e:

messages.error(request, f"Error reading file: {str(e)}")

return redirect('upload')

return render(request, 'upload.html')

import os

import re

import string

import joblib

from django.shortcuts import render

from sklearn.preprocessing import LabelEncoder

from sklearn.feature\_extraction.text import TfidfVectorizer

# Global variables

df, X, y, le = None, None, None, None

def preprocess\_data(request):

global df, X, y, le

try:

# --- Load dataset ---

df = pd.read\_csv("online\_course\_recommendation.csv")

# --- Check for missing and duplicate values ---

nulls = df.isnull().sum()

dups = df.duplicated().sum()

# --- Data cleaning ---

df.drop\_duplicates(inplace=True)

df['description'].fillna('', inplace=True)

df['tags'].fillna('', inplace=True)

df['category'].fillna('Unknown', inplace=True)

# --- Text cleaning function ---

def clean\_text(text):

text = str(text).lower()

text = re.sub(f"[{string.punctuation}]", " ", text)

text = re.sub(r"\s+", " ", text).strip()

return text

# --- Combine and clean text columns ---

df['text'] = (df['description'] + " " + df['tags']).apply(clean\_text)

# --- Encode labels ---

le = LabelEncoder()

df['category\_encoded'] = le.fit\_transform(df['category'])

y = df['category\_encoded']

# --- TF-IDF Vectorization ---

os.makedirs('model', exist\_ok=True)

vectorizer\_path = 'model/tfidf\_vectorizer.pkl'

if not os.path.exists(vectorizer\_path):

vectorizer = TfidfVectorizer(stop\_words='english', max\_features=8000)

X = vectorizer.fit\_transform(df['text'])

joblib.dump(vectorizer, vectorizer\_path)

else:

vectorizer = joblib.load(vectorizer\_path)

X = vectorizer.transform(df['text'])

# --- Train-test split ---

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=42, stratify=y

)

# --- Train the model ---

model = LogisticRegression(max\_iter=300)

model.fit(X\_train, y\_train)

# --- Save all components ---

joblib.dump(model, 'model/category\_model.pkl')

joblib.dump(le, 'model/label\_encoder.pkl')

# --- Prepare summary output ---

output\_summary = (

f"✅ Model trained successfully!<br>"

f"Null values before preprocessing: {nulls.to\_dict()}<br>"

f"Duplicate rows removed: {dups}<br>"

f"Final dataset shape: {df.shape}<br>"

f"Vectorizer & Model saved in 'model/' directory."

)

# --- Convert first 10 rows to HTML for display ---

table\_html = df.head(10).to\_html(classes='table table-striped', index=False)

return render(request, 'preprocess.html', {

'result': output\_summary,

'table': table\_html

})

except Exception as e:

return render(request, 'preprocess.html', {

'result': f"❌ Error occurred: {str(e)}",

'table': ""

})

def split\_data(request):

global X\_train, X\_test, y\_train, y\_test, X, y

if X is not None and y is not None:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42, stratify=y)

return render(request, 'split.html', {'result': f"Train shape: {X\_train.shape}, Test shape: {X\_test.shape}"})

else:

return render(request, 'split.html', {'result': "Please preprocess the data first."})

import os

import joblib

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from django.shortcuts import render

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

import joblib, os

def train\_model(request):

global clf, X\_train, X\_test, y\_train, y\_test, le

clf\_path = 'model/LogisticRegression\_Category.pkl'

# Check if the train/test data exists

if 'X\_train' in globals() and X\_train is not None and 'y\_train' in globals() and y\_train is not None:

# Ensure the model directory exists

os.makedirs(os.path.dirname(clf\_path) or '.', exist\_ok=True)

# Train the model if not already saved

if not os.path.exists(clf\_path):

clf = LogisticRegression(max\_iter=300)

clf.fit(X\_train, y\_train)

joblib.dump(clf, clf\_path)

message = "Model trained and saved successfully."

else:

clf = joblib.load(clf\_path)

message = "Model loaded successfully."

# Make predictions on the test set

y\_pred = clf.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred) \* 100

cm = confusion\_matrix(y\_test, y\_pred)

report = classification\_report(y\_test, y\_pred, target\_names=le.classes\_)

return render(request, 'train.html', {

'message': message,

'accuracy': f"{accuracy:.2f}%",

'confusion\_matrix': cm,

'classification\_report': report

})

else:

# If preprocessing/train-test split hasn't been done

return render(request, 'train.html', {

'error': 'Please preprocess and split the data first.'

})

from django.contrib.auth.models import User

from django.contrib import messages

from django.shortcuts import render, redirect

from django.contrib import messages

from django.shortcuts import render, redirect

from django.contrib.auth.hashers import make\_password

from .models import User  # your custom User model

from django.contrib import messages

from django.shortcuts import render, redirect

from django.contrib.auth.hashers import make\_password

from .models import User

from django.shortcuts import render, redirect

from django.contrib import messages

from django.contrib.auth.hashers import make\_password

from .models import User  # Custom User model

def register(request):

if request.method == 'POST':

# Safely get all form fields

name = request.POST.get('name')

email = request.POST.get('email')

phone = request.POST.get('phone')

city = request.POST.get('city')

state = request.POST.get('state')

country = request.POST.get('country')

password = request.POST.get('password')

confirm = request.POST.get('confirm\_password')

# Validate required fields

if not all([name, email, phone, city, state, country, password, confirm]):

messages.error(request, "All fields are required.")

return redirect('register')

if password != confirm:

messages.error(request, "Passwords do not match.")

return redirect('register')

if User.objects.filter(email=email).exists():

messages.error(request, "Email already exists.")

return redirect('register')

# Save user with hashed password

user = User(

name=name,

email=email,

phone=phone,

city=city,

state=state,

country=country,

password=make\_password(password)

)

user.save()

messages.success(request, "Registration successful.")

return redirect('user\_login')

return render(request, 'register.html')

from django.shortcuts import render, redirect

from django.contrib import messages

from django.contrib.auth.hashers import make\_password, check\_password

from .models import User  # Your custom User model

from django.contrib.auth.hashers import check\_password

def user\_login(request):

if request.method == 'POST':

email = request.POST['email']

password = request.POST['password']

try:

user = User.objects.get(email=email)

if check\_password(password, user.password):

request.session['user\_id'] = user.id

return redirect('user\_input')

else:

messages.error(request, "Invalid credentials")

return redirect('user\_login')

except User.DoesNotExist:

messages.error(request, "Invalid credentials")

return redirect('user\_login')

return render(request, 'userlogin.html')

from django.shortcuts import render

import pandas as pd

import joblib

import os

import re, string

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics.pairwise import cosine\_similarity

# Feature columns

FEATURE\_COLUMNS = ['user\_id', 'course\_id', 'description', 'Category', 'tags']

# Course details

course\_info = {

'Data Science': {

'description': 'Covers machine learning, analytics, data processing, and Python skills to build data-driven solutions.',

'skills': ['Python', 'Machine Learning', 'Analytics', 'Data Processing'],

'career\_paths': ['Data Analyst', 'Data Scientist', 'ML Engineer'],

'resources': [

{'type': 'Book', 'title': 'Python for Data Analysis', 'author': 'Wes McKinney'},

{'type': 'Course', 'title': 'Machine Learning A-Z', 'platform': 'Udemy'}

]

},

'Marketing': {

'description': 'Covers marketing strategies, analytics, digital marketing, and business promotion skills.',

'skills': ['Digital Marketing', 'Analytics', 'Strategy', 'SEO'],

'career\_paths': ['Marketing Analyst', 'Digital Marketer', 'Brand Manager'],

'resources': [

{'type': 'Book', 'title': 'Marketing Management', 'author': 'Philip Kotler'},

{'type': 'Course', 'title': 'Digital Marketing Masterclass', 'platform': 'Coursera'}

]

},

'Business': {

'description': 'Covers business strategy, management, analytics, and leadership skills to drive business growth.',

'skills': ['Business Strategy', 'Analytics', 'Leadership', 'Management'],

'career\_paths': ['Business Analyst', 'Project Manager', 'Operations Manager'],

'resources': [

{'type': 'Book', 'title': 'The Lean Startup', 'author': 'Eric Ries'},

{'type': 'Course', 'title': 'Business Analytics Specialization', 'platform': 'Coursera'}

]

},

'Programming': {

'description': 'Covers programming languages, software development, and problem-solving skills for coding careers.',

'skills': ['Python', 'Java', 'Cloud', 'Problem Solving'],

'career\_paths': ['Software Developer', 'Backend Developer', 'Full Stack Developer'],

'resources': [

{'type': 'Book', 'title': 'Clean Code', 'author': 'Robert C. Martin'},

{'type': 'Course', 'title': 'Java Programming Masterclass', 'platform': 'Udemy'}

]

},

'Design': {

'description': 'Covers UX, UI, creative design, and tools for designing applications and products.',

'skills': ['UX Design', 'UI Design', 'Creativity', 'Adobe Tools'],

'career\_paths': ['UI/UX Designer', 'Graphic Designer', 'Product Designer'],

'resources': [

{'type': 'Book', 'title': 'Don’t Make Me Think', 'author': 'Steve Krug'},

{'type': 'Course', 'title': 'User Experience Design Fundamentals', 'platform': 'Skillshare'}

]

}

}

# ---------------------------

# Load models once

# ---------------------------

VECTOR\_PATH = 'model/TFIDF\_Vectorizer.pkl'

MODEL\_PATH = 'model/LogisticRegression\_Category.pkl'

DATA\_PATH = 'online\_course\_recommendation.csv'

vectorizer = joblib.load(VECTOR\_PATH)

model = joblib.load(MODEL\_PATH)

df\_original = pd.read\_csv(DATA\_PATH)

le = LabelEncoder()

le.fit(df\_original['category'])

# ---------------------------

# Text preprocessing

# ---------------------------

def clean\_text(text):

if pd.isna(text):

return ""

text = text.lower()

text = re.sub(f"[{string.punctuation}]", " ", text)

text = re.sub(r"\s+", " ", text).strip()

return text

# ---------------------------

# Django view

# ---------------------------

def user\_input(request):

recommended\_course = None

recommended\_courses = []

predicted\_category = None

error = None

if request.method == 'POST':

description = request.POST.get('description', '')

tags = request.POST.get('tags', '')

if not description and not tags:

error = "Please provide description or tags."

else:

try:

input\_text = clean\_text(f"{description} {tags}")

X\_input = vectorizer.transform([input\_text])

pred\_encoded = model.predict(X\_input)[0]

predicted\_category = le.inverse\_transform([pred\_encoded])[0]

if predicted\_category in course\_info:

recommended\_course = course\_info[predicted\_category].copy()

recommended\_course['Category'] = predicted\_category

# Cosine similarity

df\_original['text'] = (df\_original['description'].fillna('') + " " + df\_original['tags'].fillna('')).apply(clean\_text)

X\_ref = vectorizer.transform(df\_original['text'])

sims = cosine\_similarity(X\_input, X\_ref).flatten()

df\_original['score'] = sims

filtered = df\_original[df\_original['category'] == predicted\_category].copy()

top\_recs = filtered.sort\_values('score', ascending=False).head(5)

recommended\_courses = top\_recs[['course\_id', 'category', 'score', 'description']].to\_dict(orient='records')

except Exception as e:

error = f"Error: {str(e)}"

return render(request, 'user\_input.html', {

'recommended\_course': recommended\_course,

'recommended\_courses': recommended\_courses,

'predicted\_category': predicted\_category,

'error': error

})

def logout(request):

request.session.flush()

return redirect('home')

**SYSTEM TESTING :**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

### TYPES OF TESTS

**Unit testing :**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

***Integration testing***

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

***Acceptance Testing***

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**SCREENSHOTS :**

**CONCLUSION :**

The Personalized E-learning Course Recommendation System provides an intelligent and adaptive approach to enhancing online learning experiences. By leveraging advanced machine learning techniques such as content-based filtering, collaborative filtering, and hybrid models, the system effectively analyzes learner profiles, interactions, and course content to deliver relevant and engaging course suggestions. This personalization not only improves learner satisfaction and engagement but also increases course completion rates and overall learning efficiency. Additionally, the system addresses challenges like cold-start problems and scalability, making it suitable for integration with existing e-learning platforms. With continuous feedback and model updates, the system ensures that learners receive dynamic, accurate, and meaningful recommendations, fostering a more efficient and enjoyable educational journey.

**FUTURE ENHANCEMENT:**

 **Skill-Path Recommendations:** Suggest sequences of courses (micro-learning paths) tailored to specific skills or career goals.

 **Multimodal Content Analysis:** Incorporate video transcripts, slides, quizzes, and assignments into embeddings for richer recommendations.

 **Social & Peer Recommendations:** Enable learners to see courses taken by peers, friends, or cohort members to foster collaborative learning.

 **Adaptive Learning Paths:** Adjust recommended courses based on real-time learner performance, progress, and engagement metrics.

 **Gamification & Rewards:** Introduce badges, points, and leaderboards to encourage course completion and platform interaction.

 **Cross-Platform Integration:** Extend recommendations to mobile apps, email notifications, and calendar scheduling for learning reminders.

 **Explainable AI Features:** Provide detailed reasoning behind each recommended course to increase transparency and trust.

 **Enhanced Analytics Dashboard:** For administrators to track learner progress, course popularity, and recommendation effectiveness.

 **Context-Aware Recommendations:** Factor in time, device, location, and learning context to improve relevance.

 **Multilingual Support:** Recommend courses in preferred languages and localize content for a global audience.

**REFERENCES**

 Koren, Y., Bell, R., & Volinsky, C. (2009). *Matrix Factorization Techniques for Recommender Systems.* IEEE Computer, 42(8), 30–37.

 Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2016). *Session-based Recommendations with Recurrent Neural Networks.* arXiv preprint arXiv:1511.06939.

 Kula, M. (2015). *Metadata Embeddings for User and Item Cold-Start Recommendations.* arXiv preprint arXiv:1507.08439.

 Ricci, F., Rokach, L., & Shapira, B. (2015). *Recommender Systems Handbook.* Springer.

 Burke, R. (2002). *Hybrid Recommender Systems: Survey and Experiments.* User Modeling and User-Adapted Interaction, 12(4), 331–370.

 Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). *Deep Learning Based Recommender System: A Survey and New Perspectives.* ACM Computing Surveys (CSUR), 52(1), 1–38.

 Aggarwal, C. C. (2016). *Recommender Systems: The Textbook.* Springer.