# 1.LECTURE ON CUSTOMER SUCCESS

04 NOVEMBER 2023

MR. SHAH IMRAN IBRAHIM SHAH

SENIOR MANAGER, CUSTOMER EXPERIENCE & INSIGHTS (P),
WILEY APAC SERVICES LLP

## ABSTRACT

In the landscape of contemporary business, customer success stands as a pivotal paradigm that transcends conventional customer support, fostering proactive engagement, and ensuring sustained value realization for clients. This report navigates the multifaceted terrain of customer success, elucidating its diverse dimensions, from the fundamental principles to the implementation strategies adopted across industries. It encompasses the holistic approach to cultivating enduring relationships, driving customer satisfaction, and achieving mutual growth between businesses and their clientele. By delving into the core tenets, case studies, and evolving trends, this report illuminates the significance of customer success as a cornerstone in today's competitive market, shedding light on its transformative impact on businesses and customer experiences.

## INTRODUCTION

Customer success is the effort a business undertakes to help its customers be most successful, both with its product and in their own business operations.

However, it is no longer sufficient to assume that the company as a whole will take on customer success management; for your customers to shine, you'll need someone (or a team) to be wholly focused on it. Dedicated customer success teams take a proactive, data-led approach to helping customers more effectively use a product.

Depending on the structure and maturity of the team, it may handle everything from trial user engagement through renewal. This comprehensive approach helps businesses reach several top-level goals, including:

- Increasing renewal sales and revenue.
- Inspiring customer loyalty and retention.
- Boosting lifetime customer value and annual recurring revenue (ARR).
- Reducing churn.

Customer success increases the likelihood that users will stick around by maximizing their mastery of the product. For subscription-based businesses, that's a vital component of growing monthly recurring revenue (MRR). For companies that don't follow that particular model, the value of customer success shows itself with leading product insights and word-of-mouth marketing.

However, customer success experiences overlap with other customer-facing functions, such as customer support, customer experience, and even account management. As easy as it is to talk about what customer success is, it's equally important to distinguish what it isn't.

**Customer success vs. customer support**

Many customer success teams report through the same structure as support or service and exhibit the same customer focus, but there is at least one key difference.

Another key difference is the metrics that customer support and customer success use to determine success. Customer support uses metrics like:

- Time to response
- Customer satisfaction
- Handle time
- First response resolution

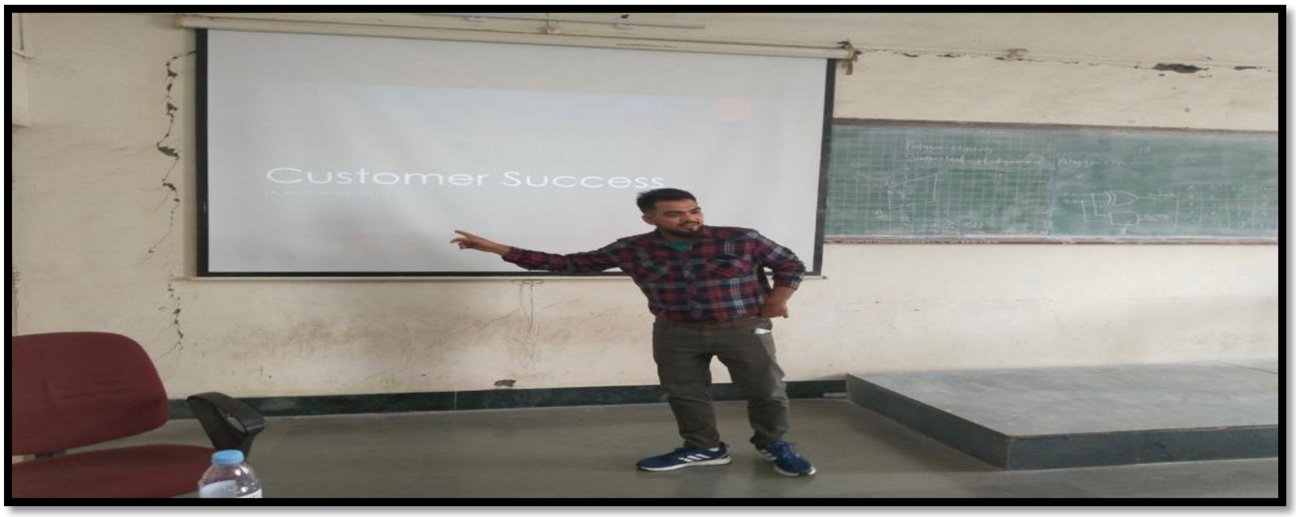
Customer success teams typically take a different approach and use metrics like:

- Expansion rate
- Churn rate
- Average MRR
- Customer health

While there may be some overlap between teams, customer service is primarily focused on metrics at an email level, whereas customer success works at a relationship or lifecycle level

GPS Map Camera
Google
Jalgaon, Maharashtra, India
computer department, Government college of engineering, Ramanand Nagar, Jalgaon, Maharashtra 425001, India
Lat 21.001936°
Long 75.549232°
04/11/23 03:31 PM GMT +05:30

Customer Success


Customer Success

## CUSTOMER SUCCESS CHALLENGES & SOLUTIONS

**1. Onboarding and training new customers**
One of the biggest challenges for customer success teams is onboarding and training new customers. This process can be time-consuming and resource-intensive, and it can also be challenging to ensure that customers retain the information they need to succeed with your product. One solution to this problem is to create a comprehensive and interactive onboarding program that includes a mix of online resources, in-person training, and ongoing support. Another solution is to use technology, such as online tutorials and training modules, to make the onboarding process more efficient and effective.

**2. Keeping customers engaged**
Another major challenge for customer success teams is keeping customers engaged with your product. This is particularly important for businesses that sell subscriptions, as customer churn can significantly impact revenue. One solution to this problem is to create a customer engagement program that includes regular check-ins, personalized content, and incentives for customers to continue using your product. Another solution is to leverage data and analytics to identify at-risk customers and intervene before they churn.

**3. Dealing with support tickets**
Customer success teams often must deal with a high volume of support tickets, which can be overwhelming and time-consuming. One solution to this problem is investing in customer service software to help manage and prioritize tickets. Another solution is to hire more support staff or outsource customer service to a third-party provider.

**4. Managing customer expectations**
**5. Addressing customer complaints**

## CONCLUSION

The landscape of modern business is inexorably tethered to the pivotal paradigm of customer success. As this report illuminates, customer success transcends the traditional confines of customer service, embodying a proactive and holistic approach aimed at not just addressing needs but exceeding expectations. It stands as the linchpin for sustainable growth, fostering enduring relationships that drive mutual value creation between businesses and their clientele.
In closing, customer success is not merely a departmental function but an organizational philosophy that propels businesses towards sustained growth and success. By nurturing enduring relationships, delivering value, and adapting to dynamic customer landscapes, businesses not only thrive but pave the way for a future where customer-centricity remains the cornerstone of success.

# 2. LECTURE ON JOURNEY TO MASTER MACHINE LEARNING

07 NOVEMBER 2023

MR. VAIBHAV TULSYAN
MACHINE LEARNING TECH LEAD AND SENIOR SOFTWARE DEVELOPER, GOOGLE

## ABSTRACT

The expedition to master machine learning is an immersive odyssey through the intricacies of data, algorithms, and their applications in crafting predictive models and intelligent systems. This report encapsulates the multifaceted voyage, tracing the trajectory from foundational principles to sophisticated implementations, elucidating the pivotal milestones, challenges, and strategies intrinsic to this transformative journey.

Beginning with a foundational comprehension, this report navigates the foundational concepts, elucidating the fundamental algorithms, and delineating the landscape of supervised, unsupervised, and reinforcement learning paradigms. It explores the symbiotic relationship between theory and practice, elucidating the diverse learning pathways—be it through structured academic courses or the autodidactic pursuit fostered by online resources, interactive platforms, and immersive projects.

## INTRODUCTION

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.
Over the last couple of decades, the technological advances in storage and processing power have enabled some innovative products based on machine learning, such as Netflix's recommendation engine and self-driving cars.

Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, and to uncover key insights in data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase. They will be required to help identify the most relevant business questions and the data to answer them.

Now talking about the purpose of this report then this report is made to describe or to depict the learning path how the tech enthusiastic nerds can land into the field of this vast machine learning world. There are a lot of opportunities for the geeks who are really interested in developing something new, something unique that can help the world to automate, predict and thus making a better perspective for a person to view the world. There can be a lot of ways they can innovate like by making "IRON MAN's goggles" which can do a lot of interesting stuffs like scanning, getting  x rays filters, or they can build future predicting machines means there are a lot of possibilities and many more stuff to be created using machine learning and AI.

# CONCEPTS

**Classification of Machine Learning**

Machine learning implementations are classified into four major categories, depending on the nature of the learning “signal” or “response” available to a learning system which are as follows:

**A. Supervised learning:**

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. The given data is labeled. Both classification and regression problems are supervised learning problems.

Example — Consider the following data regarding patients entering a clinic . The data consists of the gender and age of the patients and each patient is labeled as “healthy” or “sick”.

| Gender | Age | Label |
|---|---|---|
| M | 48 | sick |
| M | 67 | sick |
| F | 53 | healthy |
| M | 49 | sick |
| F | 32 | healthy |
| M | 34 | healthy |

**B. Unsupervised learning:**

Unsupervised machine learning is a type of algorithm used to draw inferences from datasets consisting of input data without labeled responses. In unsupervised learning algorithms, classification or categorization is not included in the observations. Example: Consider the following data regarding patients entering a clinic. The data consists of the gender and age of the patients.

| Gender | Age |
|---|---|
| M | 48 |
| M | 67 |
| F | 53 |
| M | 49 |

As a kind of learning, it resembles the methods humans use to figure out that certain objects or events are from the same class, such as by observing the degree of similarity between objects. Some recommendation systems that you find on the web in the form of marketing automation are based on this type of learning.

.

**C. Reinforcement learning:**

Reinforcement learning is the problem of getting an agent to act in the world so as to maximize its rewards.

A learner is not told what actions to take as in most forms of machine learning but instead must discover which actions yield the most reward by trying them. For example — Consider teaching a dog a new trick: we cannot tell him what to do, what not to do, but we can reward/punish it if it does the right/wrong thing.

When watching the video, notice how the program is initially clumsy and unskilled but steadily improves with training until it becomes a champion.

**D. Semi-supervised learning:**
Where an incomplete training signal is given: a training set with some (often many) of the target outputs missing. There is a special case of this principle known as Transduction where the entire set of problem instances is known at learning time, except that part of the targets are missing. Semi-supervised learning is an approach to machine learning that combines small labeled data with a large amount of unlabeled data during training. Semi-supervised learning falls between unsupervised learning and supervised learning.

Categorizing based on Required Output
Another categorization of machine-learning tasks arises when one considers the desired output of a machine-learned system:

**Classification**: When inputs are divided into two or more classes, the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This is typically tackled in a supervised way. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are “spam” and “not spam”.

**Regression**: Which is also a supervised problem, A case when the outputs are continuous rather than discrete.

**Clustering**: When a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.
Machine Learning comes into the picture when problems cannot be solved using typical approaches. ML algorithms combined with new computing technologies promote scalability and improve efficiency. Modern ML models can be used to make predictions ranging from outbreaks of disease to the rise and fall of stocks.

## ALGORITHMS

From classification to regression, here are seven algorithms you need to know:

**1. Linear regression**
Linear regression is a supervised learning algorithm used to predict and forecast values within a continuous range, such as sales numbers or prices.
Originating from statistics, linear regression performs a regression task, which maps a constant slope using an input value (X) with a variable output (Y) to predict a numeric value or quantity.
Linear regression uses labelled data to make predictions by establishing a line of best fit, or 'regression line', that is approximated from a scatter plot of data points. As a result, linear regression is used for predictive modelling rather than categorisation.

**2. Logistic regression**
Logistic regression, or 'logit regression', is a supervised learning algorithm used for binary classification, such as deciding whether an image fits into one class.
Originating from statistics, logistic regression technically predicts the probability that an input can be categorised into a single primary class. In practice, however, this can be used to group outputs into one of two categories ('the primary class' or 'not the primary class'). This is achieved

by creating a range for binary classification, such as any output between 0-.49 is put in one group, and any between .50 and 1.00 is put in another.
As a result, logistic regression in machine learning is typically used for binary categorisation rather than predictive modelling.

**3. Naive Bayes**
Naive Bayes is a set of supervised learning algorithms used to create predictive models for either binary or multi-classification. Based on Bayes' theorem, Naive Bayes operates on conditional probabilities, which are independent of one another but indicate the likelihood of a classification based on their combined factors.
For example, a programme created to identify plants might use a Naive Bayes algorithm to categorise images based on particular factors, such as perceived size, colour, and shape. While each of these factors is independent, the algorithm would note the likelihood of an object being a particular plant using the combined factors.

**4. Decision tree**
A decision tree is a supervised learning algorithm used for classification and predictive modelling.
Resembling a graphic flowchart, a decision tree begins with a root node, which asks a specific question of the data and then sends it down a branch depending on the answer. These branches each lead to an internal node, which asks another question of the data before directing it toward another branch, depending on the answer. This continues until the data reaches an end node, also called a leaf node, that doesn't branch any further.
Decision trees are common in machine learning because they can handle complex data sets with relative simplicity.

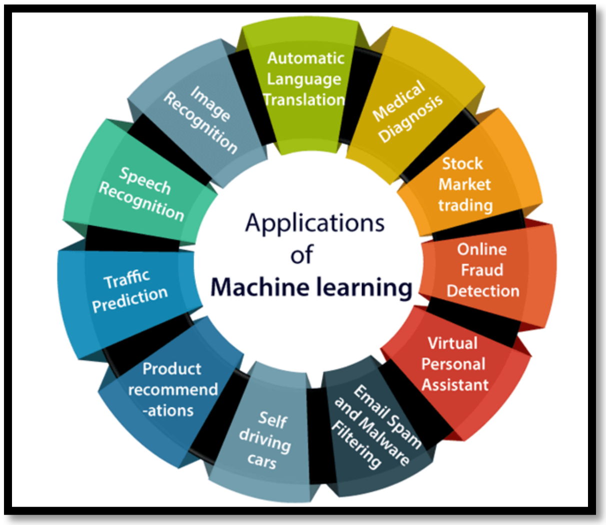
**5. Random forest algorithm**
A random forest algorithm uses an ensemble of decision trees for classification and predictive modelling.
In a random forest, many decision trees (sometimes hundreds or even thousands) are each trained using a random sample of the training set (a method known as 'bagging'). Afterwards, the algorithm puts the same data into each decision tree in the random forest and tallys their end results. The most common result is then selected as the most likely outcome for the data set.
Although they can become complex and require significant time, random forests correct the common problem of 'overfitting' that can occur with decision trees. Overfitting is when an algorithm coheres too closely to its training data set, which can negatively impact its accuracy when introduced to new data later.
**6. K-nearest neighbour (KNN) algorithm**
A K-nearest neighbour is a supervised learning algorithm for classification and predictive modelling.
True to its name, KNN algorithms classify an output by its proximity to other outputs on a graph. For example, if an output is closest to a cluster of blue points on a graph rather than a cluster of red points, it would be classified as a member of the blue group. This approach means that KNN algorithms can classify known outcomes or predict the value of unknown ones.

## APPLICATIONS OF MACHINE LEARNING

Machine learning is a buzzword for today's technology, and it is growing very rapidly day by day. We are using machine learning in our daily life even without knowing it such as Google Maps, Google assistant, Alexa, etc. Below are some most trending real-world applications of Machine Learning:

1. Image Recognition:
2. Speech Recognition
3. Traffic prediction:
4. Product recommendations:
5. Self-driving cars:
6. Email Spam and Malware Filtering:
7. Virtual Personal Assistant:
8. Online Fraud Detection:
9. Stock Market trading:

**Fig. Applications of Machine**



Applications of Machine learning
Automatic Language Translation
Medical Diagnosis
Stock Market trading
Online Fraud Detection
Virtual Personal Assistant
Email Spam and Malware Filtering
Self driving cars
Product recommend -ations
Traffic Prediction
Speech Recognition
Image Recognition

## MAJOR CHALLENGES FACED BY MACHINE LEARNING PROFESSIONALS

**1. Poor Quality of Data**

Data plays a significant role in the machine learning process. One of the significant issues that machine learning professionals face is the absence of good quality data. Unclean and noisy data can make the whole process extremely exhausting. We don’t want our algorithm to make inaccurate or faulty predictions. Hence the quality of data is essential to enhance the output. Therefore, we need to ensure that the process of data preprocessing which includes removing outliers, filtering missing values, and removing unwanted features, is done with the utmost level of perfection.

**2. Underfitting of Training Data**

This process occurs when data is unable to establish an accurate relationship between input and output variables. It simply means trying to fit in undersized jeans. It signifies the data is too simple to establish a precise relationship. To overcome this issue:

- Maximize the training time
- Enhance the complexity of the model
- Add more features to the data
- Reduce regular parameters
- Increasing the training time of model

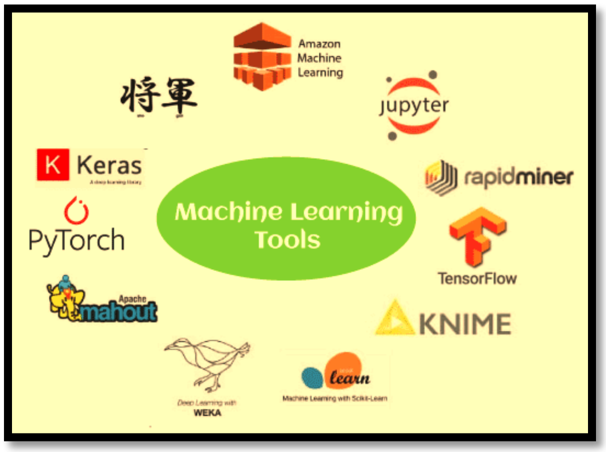
**3. Overfitting of Training Data**

Overfitting refers to a machine learning model trained with a massive amount of data that negatively affect its performance. It is like trying to fit in Oversized jeans. Unfortunately, this is one of the significant issues faced by machine learning professionals. This means that the algorithm is trained with noisy and biased data, which will affect its overall performance. Let's understand this with the help of an example. Let's consider a model trained to differentiate between a cat, a rabbit, a dog, and a tiger. The training data contains 1000 cats, 1000 dogs, 1000 tigers, and 4000 Rabbits. Then there is a considerable probability that it will identify the cat as a rabbit. In this example, we had a vast amount of data, but it was biased; hence the prediction was negatively affected.

We can tackle this issue by:

- Analyzing the data with the utmost level of perfection
- Use data augmentation technique
- Remove outliers in the training set
- Select a model with lesser features

**4. Lack of Training Data**



The most important task you need to do in the machine learning process is to train the data to achieve an accurate output. Less amount training data will produce inaccurate or too biased predictions. Let us understand this with the help of an example. Consider a machine learning algorithm similar to training a child. One day you decided to explain to a child how to distinguish between an apple and a watermelon. You will take an apple and a watermelon and show him the difference between both based on their color, shape, and taste. In this way, soon, he will attain perfection in differentiating between the two. But on the other hand, a machine-learning algorithm needs a lot of data to distinguish. For complex problems, it may even require millions of data to be trained. Therefore we need to ensure that Machine learning algorithms are trained with sufficient amounts of data.

## MACHINE LEARNING TOOLS

**1. TensorFlow**

TensorFlow is one of the most popular open-source libraries used to train and build both machine learning and deep learning models. It provides a JS library and was developed by Google Brain Team. It is much popular among machine learning enthusiasts, and they use it for building different ML applications. It offers a powerful library, tools, and resources for numerical computation, specifically for large scale machine learning and deep learning projects. It enables data scientists/ML developers to build and deploy machine learning applications efficiently. For training and building the ML models, TensorFlow provides a high-level Keras API, which lets users easily start with TensorFlow and machine learning.

**2. PyTorch**

PyTorch is an open-source machine learning framework, which is based on the Torch library. This framework is free and open-source and developed by FAIR(Facebook's AI Research lab). It is one of the popular ML frameworks, which can be used for various applications, including computer vision and natural language processing. PyTorch has Python and C++ interfaces; however, the Python interface is more interactive. Different deep learning software is made up on top of PyTorch, such as PyTorch Lightning, Hugging Face's Transformers, Tesla autopilot, etc.

It specifies a Tensor class containing an n-dimensional array that can perform tensor computations along with GPU support.

**3. Google Cloud ML Engine**

While training a classifier with a huge amount of data, a computer system might not perform well. However, various machine learning or deep learning projects requires millions or billions of training datasets. Or the algorithm that is being used is taking a long time for execution. In such a case, one should go for the Google Cloud ML Engine. It is a hosted platform where ML developers and data scientists build and run optimum quality machine, learning models. It provides a managed service that allows developers to easily create ML models with any type of data and of any size.

**4. Amazon Machine Learning (AML)**

Amazon provides a great number of machine learning tools, and one of them is Amazon Machine Learning or AML. Amazon Machine Learning (AML) is a cloud-based and robust machine learning software application, which is widely used for building machine learning models and making predictions. Moreover, it integrates data from multiple sources, including Redshift, Amazon S3, or RDS.

## FUTURE SCOPE OF MACHINE LEARNING

The scope of Machine Learning is not limited to the investment sector. Rather, it is expanding across all fields such as banking and finance, information technology, media & entertainment, gaming, and the automotive industry. As the Machine Learning scope is very high, there are some areas where researchers are working toward revolutionizing the world for the future. Let us discuss them in detail.

## CONCLUSION

The voyage to master machine learning is an exhilarating and ever-evolving expedition that transcends the boundaries of data and algorithms, forging a path toward transformative innovation. Throughout this comprehensive exploration, we've traversed the foundational bedrock of machine learning, unraveling its fundamental concepts, algorithms, and diverse learning pathways.

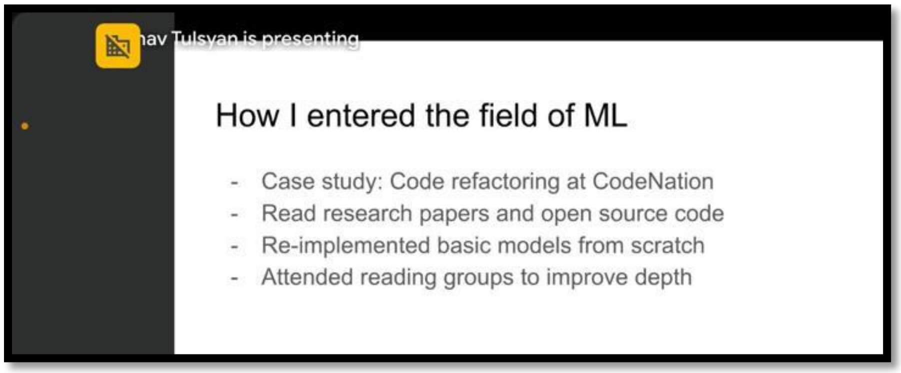
# 3. LECTURE ON GENERATIVE AI

14 NOVEMBER 2023

MR. Ajay T. SHAJU
S7 AD

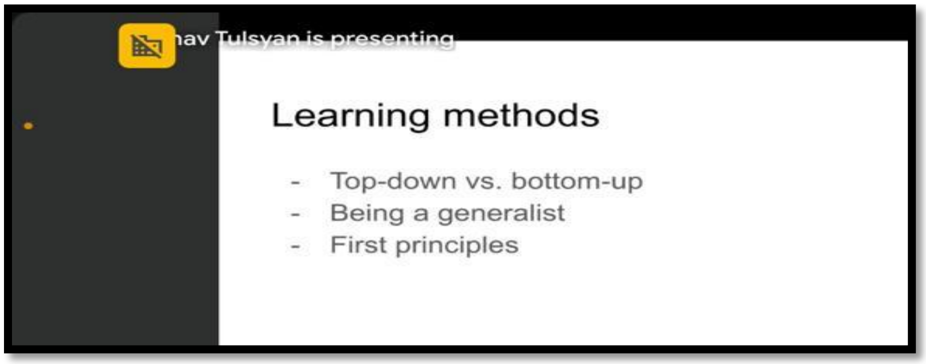
## ABSTRACT

The term "generative AI" refers to computational techniques that are capable of generating seemingly new, meaningful content such as text, images, or audio from training data. The widespread diffusion of this technology with examples such as Dall-E 2, GPT-4, and Copilot is currently revolutionizing the way we work and communicate with each other. In this article, we provide a conceptualization of generative AI as an entity in socio-technical systems and provide examples of models, systems, and applications. Based on that, we introduce limitations of current generative AI and provide an agenda for Business & Information Systems Engineering (BISE) research. Different from previous works, we focus on generative AI in the context of information systems, and, to this end, we discuss several opportunities and challenges that are unique to the BISE community and make suggestions for impactful directions for BISE research.



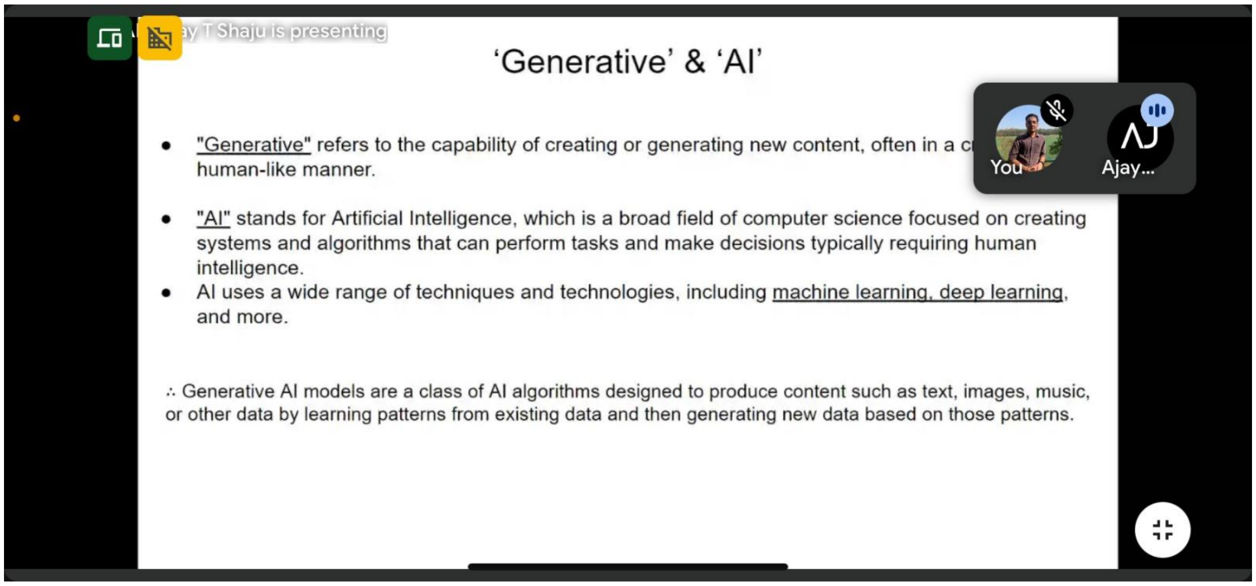
How I entered the field of ML

- Case study: Code refactoring at CodeNation
- Read research papers and open source code
- Re-implemented basic models from scratch
- Attended reading groups to improve depth



Learning methods

- Top-down vs. bottom-up
- Being a generalist
- First principles



## 'Generative' & 'AI'

- "Generative" refers to the capability of creating or generating new content, often in a c human-like manner.
- "AI" stands for Artificial Intelligence, which is a broad field of computer science focused on creating systems and algorithms that can perform tasks and make decisions typically requiring human intelligence.
- AI uses a wide range of techniques and technologies, including machine learning, deep learning, and more.

∴ Generative AI models are a class of AI algorithms designed to produce content such as text, images, music, or other data by learning patterns from existing data and then generating new data based on those patterns.

## INTRODUCTION

Generative AI is a type of artificial intelligence technology that can produce various types of content, including text, imagery, audio and synthetic data. The recent buzz around generative AI has been driven by the simplicity of new user interfaces for creating high-quality text, graphics and videos in a matter of seconds.

The technology, it should be noted, is not brand-new. Generative AI was introduced in the 1960s in chatbots. But it was not until 2014, with the introduction of generative adversarial networks, or GANs -- a type of machine learning algorithm -- that generative AI could create convincingly authentic images, videos and audio of real people.

Two additional recent advances that will be discussed in more detail below have played a critical part in generative AI going mainstream: transformers and the breakthrough language models they enabled. Transformers are a type of machine learning that made it possible for researchers to train ever-larger models without having to label all of the data in advance. New models could thus be trained on billions of pages of text, resulting in answers with more depth. In addition, transformers unlocked a new notion called attention that enabled models to track the connections between words across pages, chapters and books rather than just in individual sentences. And not just words: Transformers could also use their ability to track connections to analyze code, proteins, chemicals and DNA.

The rapid advances in so-called large language models (LLMs) -- i.e., models with billions or even trillions of parameters -- have opened a new era in which generative AI models can write engaging text, paint photorealistic images and even create somewhat entertaining sitcoms on the fly. Moreover, innovations in multimodal AI enable teams to generate content across multiple types of media, including text, graphics and video. This is the basis for tools like Dall-E that automatically create images from a text description or generate text captions from images.

These breakthroughs notwithstanding, we are still in the early days of using generative AI to create readable text and photorealistic stylized graphics. Early implementations have had issues with accuracy and bias, as well as being prone to hallucinations and spitting back weird answers. Still, progress thus far indicates that the inherent capabilities of this generative AI could fundamentally change enterprise technology how businesses operate. Going forward, this technology could help write code, design new drugs, develop products, redesign business processes and transform supply chains.

## WHAT ARE USE CASES FOR GENERATIVE AI?

Some of the use cases for generative AI include the following:

- Implementing chatbots for customer service and technical support.
- Deploying deepfakes for mimicking people or even specific individuals.
- Improving dubbing for movies and educational content in different languages.
- Writing email responses, dating profiles, resumes and term papers.
- Creating photorealistic art in a particular style.
- Improving product demonstration videos.
- Suggesting new drug compounds to test.
- Designing physical products and buildings.
- Optimizing new chip designs.
- Writing music in a specific style or tone.

## WHAT ARE THE BENEFITS OF GENERATIVE AI?

Some of the potential benefits of implementing generative AI include the following:

- Automating the manual process of writing content.
- Reducing the effort of responding to emails.
- Improving the response to specific technical queries.
- Creating realistic representations of people.
- Summarizing complex information into a coherent narrative.
- Simplifying the process of creating content in a particular style.

## WHAT ARE THE LIMITATIONS OF GENERATIVE AI?

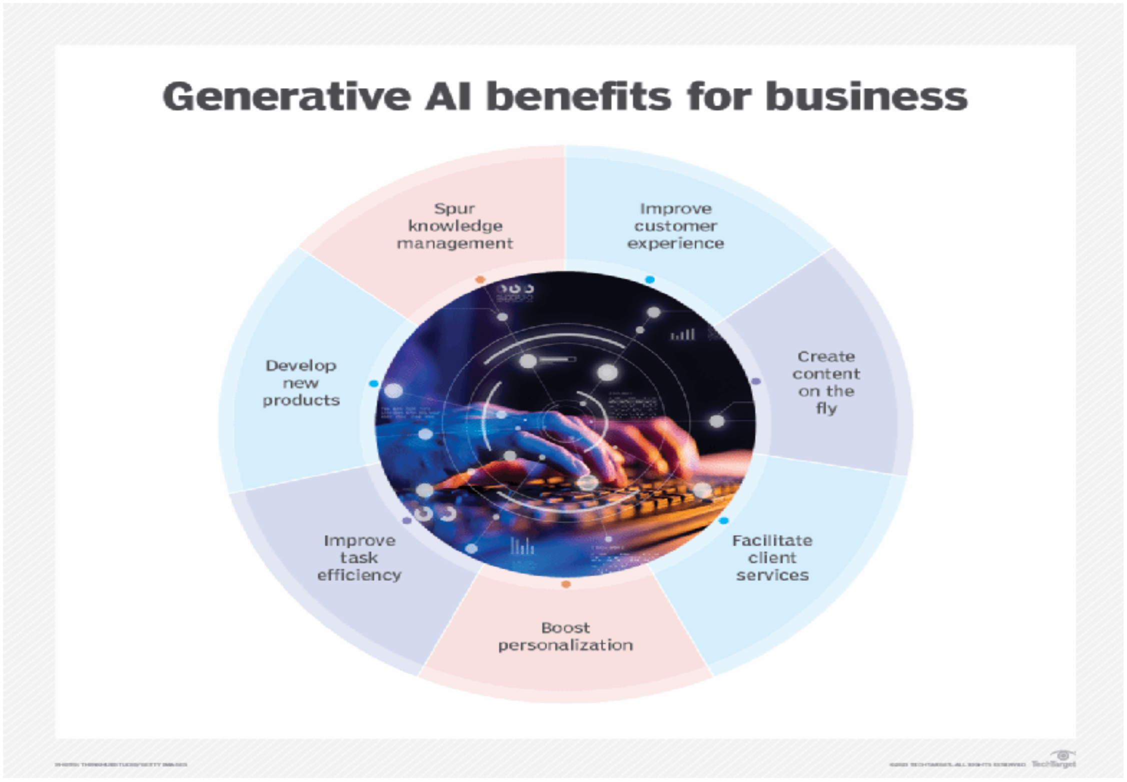
Here are some of the limitations to consider when implementing or using a generative AI app:

- It does not always identify the source of content.
- It can be challenging to assess the bias of original sources.
- Realistic-sounding content makes it harder to identify inaccurate information.
- It can be difficult to understand how to tune for new circumstances.
- Results can gloss over bias, prejudice and hatred.

## THE FUTURE OF GENERATIVE AI

The incredible depth and ease of ChatGPT spurred widespread adoption of generative AI. To be sure, the speedy adoption of generative AI applications has also demonstrated some of the difficulties in rolling out this technology safely and responsibly. But these early implementation issues have inspired research into better tools for detecting AI-generated text, images and video.

Generative AI will continue to evolve, making advancements in translation, drug discovery, anomaly detection and the generation of new content, from text and video to fashion design and music. As good as these new one-off tools are, the most significant impact of generative AI in the future will come from integrating these capabilities directly into the tools we already use.
Grammar checkers, for example, will get better. Design tools will seamlessly embed more useful recommendations directly into our workflows. Training tools will be able to automatically identify best practices in one part of an organization to help train other employees more efficiently. These are just a fraction of the ways generative AI will change what we do in the near-term.


Generative AI benefits for business
Spur knowledge management
Improve customer experience
Create content on the fly
Facilitate client services
Boost personalization
Improve task efficiency
Develop new products

## CONCLUSION

Generative AI represents a transformative technology with immense potential across diverse domains. However, its responsible development and deployment necessitate addressing ethical challenges, ensuring transparency, fairness, and accountability in its use.

# 4. LECTURE ON TIME & SPACE COMPLEXITY ANALAYSIS OF ALGORITHM

21 NOVEMBER 2023

PROF. SARJU S

ASSISTANT PROFESSOR, DEPARTMENT OF CSE, SJCET

## ABSTRACT

The analysis of time and space complexity in algorithms constitutes a pivotal cornerstone in the realm of computer science and software engineering. This report delves into the intricate study of algorithmic efficiency, illuminating the significance of evaluating computational resources—time and memory—utilized by algorithms. Exploring fundamental concepts and methodologies, it navigates through the understanding of Big O notation, elucidating how it quantifies algorithmic performance and scalability. Moreover, this report elucidates the symbiotic relationship between time and space complexities, offering insights into strategies for optimizing algorithms to achieve optimal performance and mitigate resource constraints, thereby laying a foundational understanding crucial for algorithm design and problem-solving in diverse computational domains.

## INTRODUCTION

**Algorithm Analysis**

Analysis of efficiency of an algorithm can be performed at two different stages, before implementation and after implementation, as

1. A priori analysis − This is defined as theoretical analysis of an algorithm. Efficiency of algorithm is measured by assuming that all other factors e.g. speed of processor, are constant and have no effect on implementation.
2. A posterior analysis − This is defined as empirical analysis of an algorithm. The chosen algorithm is implemented using programming language. Next the chosen algorithm is executed on target computer machine. In this analysis, actual statistics like running time and space needed are collected.

**Algorithm Complexity**

Suppose X is treated as an algorithm and N is treated as the size of input data, the time and space implemented by the Algorithm X are the two main factors which determine the efficiency of X.

1. Time Factor − The time is calculated or measured by counting the number of key operations such as comparisons in sorting algorithm.
2. Space Factor − The space is calculated or measured by counting the maximum memory space required by the algorithm.

**Space Complexity**

Space complexity of an algorithm represents the amount of memory space needed the algorithm in its life cycle.

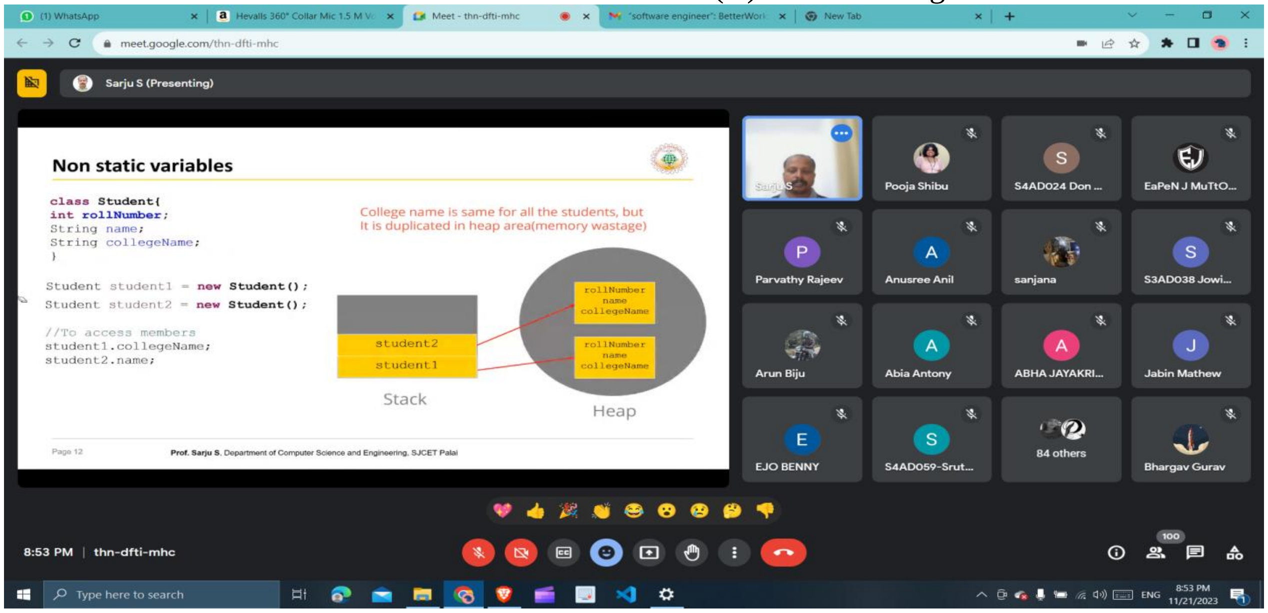
Space needed by an algorithm is equal to the sum of the following two components

A fixed part that is a space required to store certain data and variables (i.e. simple variables and constants, program size etc.), that are not dependent of the size of the problem.
A variable part is a space required by variables, whose size is totally dependent on the size of the problem. For example, recursion stack space, dynamic memory allocation etc.
Space complexity S(p) of any algorithm p is S(p) = A + Sp(I) Where A is treated as the fixed part and S(I) is treated as the variable part of the algorithm which depends on instance characteristic I. Following is a simple example that tries to explain the concept

```
Algorithm
SUM(P, Q)
Step 1 - START
Step 2 - R ← P + Q + 10
Step 3 - Stop
```

Here we have three variables P, Q and R and one constant. Hence S(p) = 1+3. Now space is dependent on data types of given constant types and variables and it will be multiplied accordingly.

**Time Complexity**
Time Complexity of an algorithm is the representation of the amount of time required by the algorithm to execute to completion. Time requirements can be denoted or defined as a numerical function t(N), where t(N) can be measured as the number of steps, provided each step takes constant time.
For example, in case of addition of two n-bit integers, N steps are taken. Consequently, the total computational time is t(N) = c*n, where c is the time consumed for addition of two bits. Here, we observe that t(N) grows linear



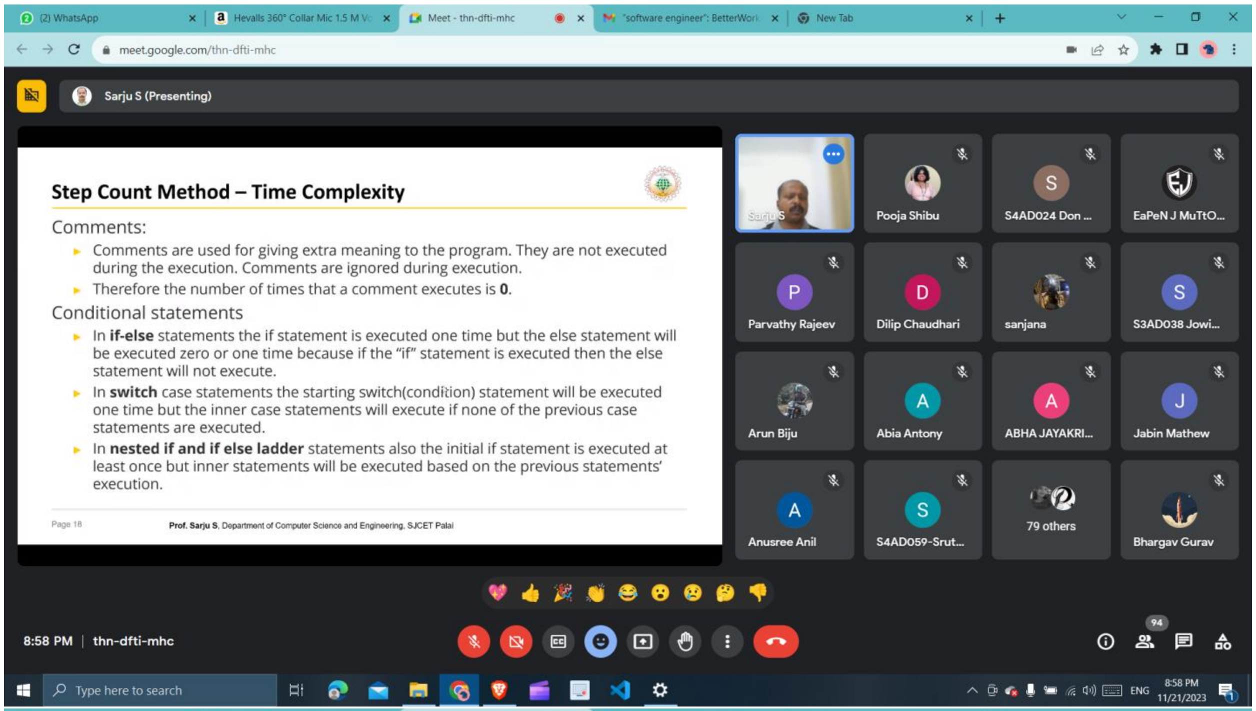
Non static variables

```
class Student{
int rollNumber;
String name;
String collegeName;
}

Student student1 = new Student();
Student student2 = new Student();

//To access members
student1.collegeName;
student2.name;
```

College name is same for all the students, but It is duplicated in heap area(memory wastage)

Page 12 Prof. Sarju S, Department of Computer Science and Engineering, SJCET Palai

ly as input size increases.



**Step Count Method – Time Complexity**

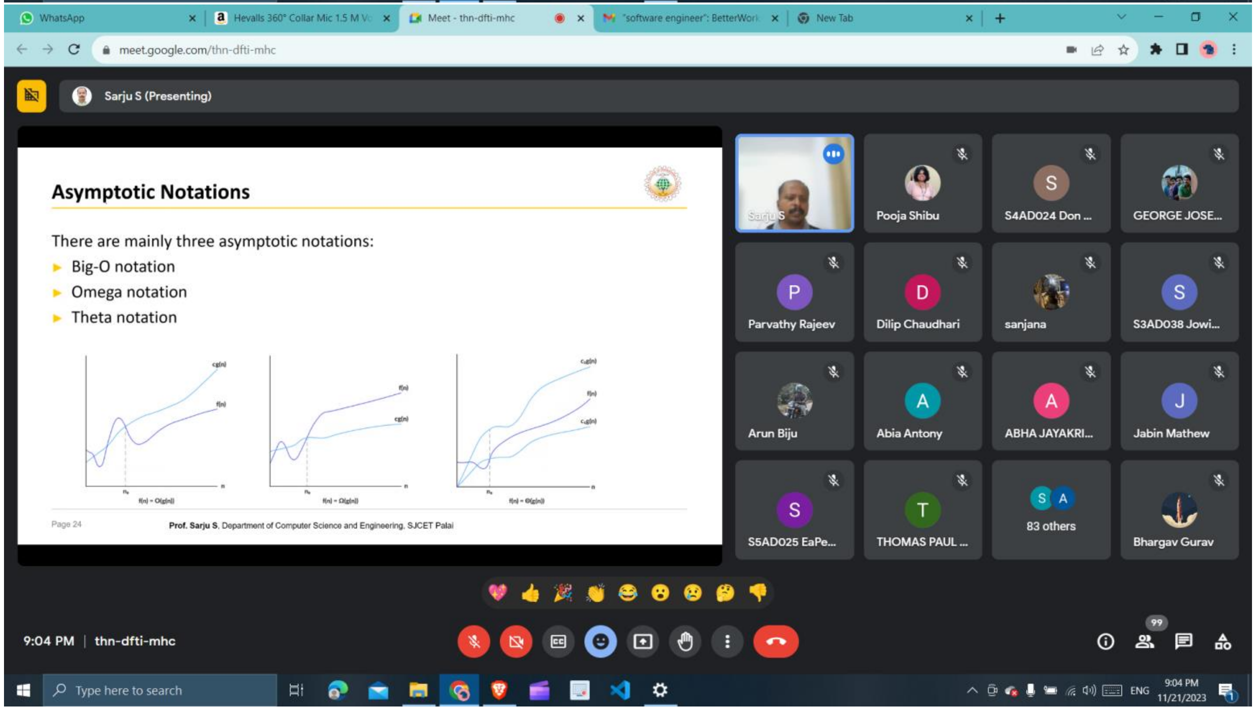
Comments:

- Comments are used for giving extra meaning to the program. They are not executed during the execution. Comments are ignored during execution.
- Therefore the number of times that a comment executes is **0**.

Conditional statements

- In **if-else** statements the if statement is executed one time but the else statement will be executed zero or one time because if the "if" statement is executed then the else statement will not execute.
- In **switch** case statements the starting switch(condition) statement will be executed one time but the inner case statements will execute if none of the previous case statements are executed.
- In **nested if and if else ladder** statements also the initial if statement is executed at least once but inner statements will be executed based on the previous statements' execution.

Page 18 **Prof. Sarju S**, Department of Computer Science and Engineering, SJCET Palai



**Asymptotic Notations**

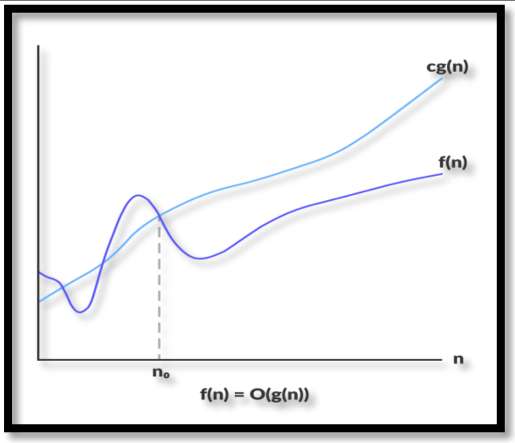
There are mainly three asymptotic notations:

- Big-O notation
- Omega notation
- Theta notation

Page 24 **Prof. Sarju S**, Department of Computer Science and Engineering, SJCET Palai

## ASYMPTOTIC ANALYSIS: BIG-O NOTATION AND MORE

The efficiency of an algorithm depends on the amount of time, storage and other resources required to execute the algorithm. The efficiency is measured with the help of asymptotic notations.



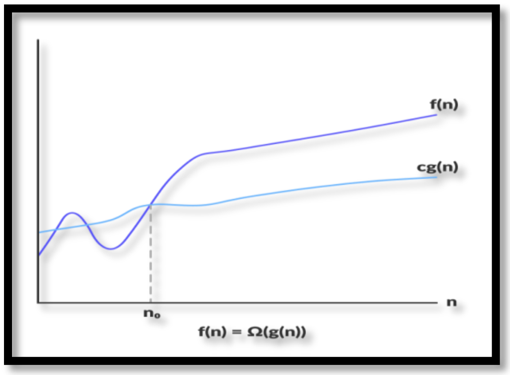
An algorithm may not have the same performance for different types of inputs. With the increase in the input size, the performance will change.
The study of change in performance of the algorithm with the change in the order of the input size is defined as asymptotic analysis.

**Asymptotic Notations**
Asymptotic notations are the mathematical notations used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value.
For example: In bubble sort, when the input array is already sorted, the time taken by the algorithm is linear i.e. the best case.
But, when the input array is in reverse condition, the algorithm takes the maximum time (quadratic) to sort the elements i.e. the worst case.
When the input array is neither sorted nor in reverse order, then it takes average time. These durations are denoted using asymptotic notations.
There are mainly three asymptotic notations:

1. Big-O notation
2. Omega notation
3. Theta notation

**Big-O Notation (O-notation)**
Big-O notation represents the upper bound of the running time of an algorithm. Thus, it gives the worst-case complexity of an algorithm.

Big-O gives the upper bound of a function
O(g(n)) = { f(n): there exist positive constants c and n0
such that 0 ≤ f(n) ≤ cg(n) for all n ≥ n0 }
The above expression can be described as a function f(n) belongs to the set O(g(n)) if there exists a positive constant c such that it lies between 0 and cg(n), for sufficiently large n.



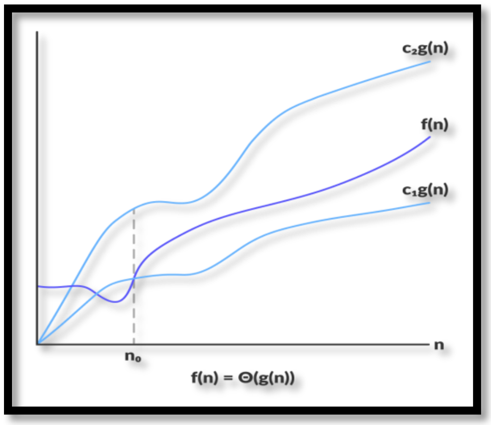
For any value of n, the running time of an algorithm does not cross the time provided by O(g(n)).
Since it gives the worst-case running time of an algorithm, it is widely used to analyze an algorithm as we are always interested in the worst-case scenario.

**Omega Notation (Ω-notation)**
Omega notation represents the lower bound of the running time of an algorithm. Thus, it provides the best case complexity of an algorithm.
Omega gives the lower bound of a function
Ω(g(n)) = { f(n): there exist positive constants c and n0
such that 0 ≤ cg(n) ≤ f(n) for all n ≥ n0 }

The above expression can be described as a function f(n) belongs to the set Ω(g(n)) if there exists a

positive constant c such that it lies above cg(n), for sufficiently large n.
For any value of n, the minimum time required by the algorithm is given by Omega Ω(g(n)).

**Theta Notation (Θ-notation)**
Theta notation encloses the function from above and below. Since it represents the upper and the lower bound of the running time of an algorithm, it is used for analyzing the average-case complexity of an algorithm.



Theta bounds the function within constants factors
For a function g(n), Θ(g(n)) is given by the relation:
Θ(g(n)) = { f(n): there exist positive constants c1, c2 and n0
such that 0 ≤ c1g(n) ≤ f(n) ≤ c2g(n) for all n ≥ n0
}
The above expression can be described as a function f(n) belongs to the set Θ(g(n)) if there exist positive constants c1 and c2 such that it can be sandwiched between c1g(n) and c2g(n), for sufficiently large n.
If a function f(n) lies anywhere in between c1g(n) and c2g(n) for all n ≥ n0, then f(n) is said to be asymptotically tight bound.

## PRACTICAL APPLICATIONS AND EXAMPLES

Real-World Applications:
**1. Search Engines:**
Case Study: Google's PageRank algorithm is a prime example of an efficient algorithm. By analyzing the link structure of the web, PageRank determines the relevance of web pages, enabling Google to provide efficient search results. Its time complexity is optimized to swiftly navigate through vast amounts of data, ranking pages based on their importance.

**2. Sorting Algorithms in Databases:**
Case Study: Database management systems utilize sorting algorithms like Quicksort and Mergesort to efficiently arrange data. These algorithms offer optimal time complexities, allowing for faster retrieval and manipulation of data, thus enhancing the overall performance of the database.

**3. Compression Algorithms:**
Case Study: ZIP compression uses algorithms like Huffman coding or Lempel-Ziv-Welch (LZW) to efficiently compress files. These algorithms optimize space complexity, reducing file

sizes while maintaining or even improving the quality of the data. This enables quicker file transfers and saves storage space.

**4. Network Routing:**
Case Study: Routing algorithms in networking, such as Dijkstra's algorithm, help find the shortest path between nodes in a network efficiently. By analyzing network topology and optimizing time complexities, these algorithms enable routers to efficiently transmit data packets, reducing latency and congestion.

## OPTIMIZATION STRATEGIES

Here are several optimization strategies used in algorithm design:

1. Algorithmic Selection.
2. Data Structures.
3. Divide and Conquer.
4. Dynamic Programming.
5. Greedy Algorithms.
6. Pruning Techniques.
7. Approximation Algorithms.
8. Parallelism and Concurrency.
9. Preprocessing and Caching.
10. Space-Time Trade-offs.

## CONCLUSION

In conclusion, the exploration into the realm of time and space complexity analysis unveils the critical essence of efficiency in algorithmic design. Through the lenses of Big O notation, this report illuminated the means to gauge and quantify the performance of algorithms, facilitating a nuanced understanding of their scalability and resource utilization. The interplay between time and space complexities emerged as a pivotal consideration, guiding strategies for optimizing algorithms to strike a balance between computational resources, ultimately fostering the creation of more efficient and scalable solutions.

## REFERENCES

1. ChatGPT
2. GeeksForGeeks
3. JavatPoint

# 5. LECTURE ON OPEN SOURCE AND GSOC

21 NOVEMBER 2023

MR. PAVAN JOSHI

SOFTWARE ENGINEER, ALBY

## ABSTRACT

Open source software development has been revolutionizing the tech industry, fostering collaboration, innovation, and accessibility. Google Summer of Code (GSoC) stands as a testament to this synergy, providing a platform for budding developers to contribute meaningfully to open-source projects. This report delves into the profound impact GSoC has had on the open-source ecosystem, analyzing its role in fostering mentorship, nurturing talent, and accelerating project growth.

Additionally, the report evaluates the challenges faced by GSoC, such as diversity and inclusivity concerns, and proposes strategies to address these issues. It also discusses the evolving landscape of open-source software development and the role GSoC plays in adapting to these changes, including its impact on emerging technologies and trends.

## INTRODUCTION

### Open source

#### What is open source?

Open-source software is software whose source code—the instructions that define what the software does—is published and freely available. The opposite of open source is closed source. Source code is human-readable, and software developers create software by writing it. To run software on a device, though, the source code has to be transformed into a form that's mostly unreadable to humans. That unreadable form is what you get when you download an app. Open-source software is often developed in a collaborative manner, and considered a public good, free for anybody to use. The Brave Browser, Linux operating system, and OpenOffice are examples of open-source software.

#### Why does it matter whether software is open source?

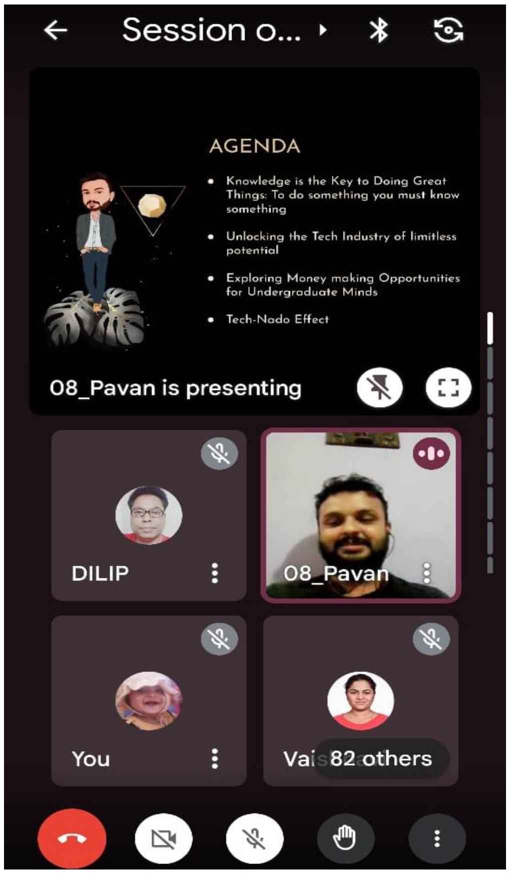
With a closed-source app, it takes very specific expertise—which even many professional software developers don't have—to be able to understand exactly what that app is doing. Even for people with that expertise, it takes a lot of time and effort to fully analyze a closed-source app. It's much easier to understand what an open-source app is doing.

That means it's easier to hide nefarious behavior, or even malware, in a closed-source app. An app may be stealing your private information, and if that app isn't open source, you'll probably never know.

Another benefit of open-source software is that the community at large can contribute to it in various ways. Other developers can contribute new features and bug fixes, or inspect for security problems. Anyone can copy and modify the software to suit their own needs.

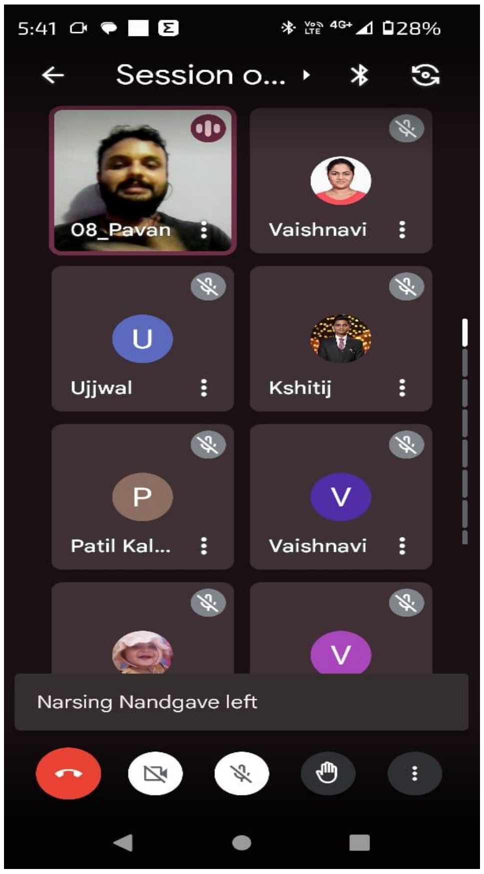
## What are some examples of open-source software?

The Brave browser is one good example. It follows a common practice in the open-source community, called "forking." This is when a project is started as a copy of another open-source project, rather than starting from scratch. Brave is based on Chromium, an open-source Web browser that also forms the basis of Chrome and Edge. Another open-source project is the Linux operating system, which powers a wide range of devices.



AGENDA

- Knowledge is the Key to Doing Great Things: To do something you must know something
- Unlocking the Tech Industry of limitless potential
- Exploring Money making Opportunities for Undergraduate Minds
- Tech-Nodo Effect



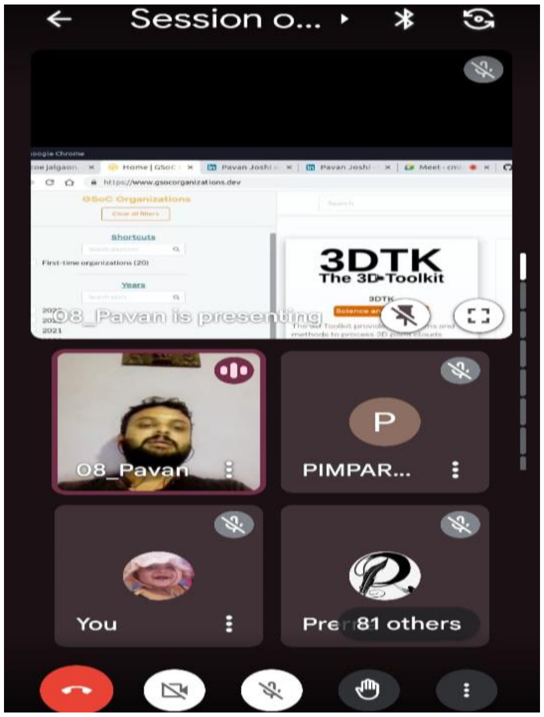
Narsing Nandgave left

## Why do people and companies give away their source code?

It may not be obvious why anyone would give away their source code, rather than making money from it. But there are several reasons to publish (or open) source code:

- To contribute back to the community. Lots of companies use other open-source software, including for business-critical purposes, and publishing some of their own source code can be a way of giving back in kind.
- To receive community contributions, such as bug reports and improvements.
- To let others adapt the code however they need to.
- To be transparent, and show that there's no nefarious behavior being hidden. This is especially important for browser extensions.

Companies can open-source their software and still make money from it, such as by selling support contracts for it. Individuals who maintain open-source software can also accept payment in return for prioritizing specific features or providing support.



## GSOC ( Google Summer of Code )

Google Summer of Code is a global, online mentoring program focused on introducing new contributors to open source software development. GSoC contributors work on a 12+ week programming project with the guidance of mentors from their open source organization.

Since 2005, the Google Summer of Code program has connected 19,000+ new open source contributors from 112 countries with 18,000+ mentors from 133 countries. Google Summer of Code has produced over 43 million lines of code for 800+ open source organizations.

During Google Summer of Code, participating contributors are paired with mentors from open source organizations, gaining exposure to real-world software development techniques. Contributors will learn from experienced open source developers while writing code for real-world projects! A small stipend is provided as an incentive.

Participating organizations use the program to identify and bring in new, excited developers. Many of those new developers will continue to contribute to their new communities and open source long after GSoC is over.

### How It Works

#### Contributors

Potential GSoC contributors contact the mentor organizations they want to work with and write a project proposal based on ideas the organization has suggested. Once accepted, GSoC contributors spend a few weeks becoming familiar with the community norms and codebase while determining expected milestones with their mentor for the summer. GSoC contributors then spend 12+ weeks coding on their projects.

#### Organizations

Open source projects apply to be mentor organizations. Once accepted, organizations discuss possible ideas with contributors and choose the proposals they wish to mentor for the summer. They provide mentors to help guide each contributor through the program.

**Mentors**
Community members and committers already active in the mentoring organizations can choose to mentor a contributor project. Mentors and GSoC contributors work together to determine appropriate goals for the program period. Mentor interaction is a vital part of the program.

**Full Program Timeline**
**Organization Application Period**

Open source organizations can submit their applications to be mentor organizations for GSoC.

**Organizations Announced**
Potential GSoC contributors discuss project ideas with accepted mentor organizations.

**Contributor Application Period**
Potential GSoC contributors can register and submit their proposals to the mentor organizations that interest them.

**Proposal Review Period**
Organizations review and select contributor proposals.

**Contributor Projects Announced**
Accepted GSoC contributors are paired with a mentor and start planning their projects and milestones.

**Community Bonding**
GSoC contributors spend 3 weeks learning about their organization's community and preparing for their coding project.

**Coding Period**
GSoC contributors work on their Google Summer of Code projects.

**Evaluations**
Mentors and GSoC contributors submit their evaluations of one another.

**GSoC contributors Submit Code and Final Evaluations**
GSoC contributors submit their code, project summaries, and final evaluations of their mentors.

**Mentors Submit Final Evaluations**
Mentors review their GSoC contributor code samples and determine if they should pass the Google Summer of Code Program.

**Results Announced**
GSoC contributors are notified of the pass/fail status of their Google Summer of Code projects.

## CHALLENGES AND OPPORTUNITIES IN GSOC

**Challenges :**

- High turnover
- Overwhelming noise
- Lack of visibility
- Tech-driven approaches
- Performance degradation in applications that use OpenMP

**Some opportunities that participants may have include:**

- Exposure to open source projects
- Learning about open source culture and community
- Interacting with experienced developers
- Building a network of mentors and programmers
- Internship opportunities
- Stipends of up to US\$3,000

## FUTURE OF OPEN SOURCE

The future of open source software is poised for an intriguing evolution, driven by several key trends and factors shaping its trajectory.

- Collaboration and Interconnectivity.
- Sustainability and Community Growth.
- AI and Automation Integration.
- Security and Compliance.
- Ethical and Responsible AI.
- Decentralization and Web3.
- Continued Corporate Involvement.

## CONCLUSION

The future of open source software is a landscape of innovation, collaboration, and responsibility. Its evolution involves harnessing emerging technologies, nurturing vibrant communities, fortifying security measures, and navigating ethical considerations. As open source continues to underpin technological advancements, its adaptability and commitment to collaboration will be pivotal in shaping the digital future.

## REFERENCES

1. ChatGPT
2. GeeksForGeeks
3. Intellipat
4. JavatPoint