Implementing Artificial Intelligence Algorithms To Develop Recommender System For Personalized Education

By
Haseeb Sultan
Registration No 00000317658
Fall-2019-MS-Systems Engineering

Supervisor
Dr Mehak Rafiq

A thesis submitted in partial fulfilment of the Masters of Systems Engineering degree requirements.

In

School of Interdisciplinary Engineering and Science,

National University of Sciences and Technology

Islamabad, Pakistan.

April 2023

Thesis Acceptance Certificate

Certified that the final copy of the MS thesis entitled “Implementing Artificial Intelligence algorithms to develop recommender system for personalized education", written by Haseeb Sultan (Registration No 00000317658), has been reviewed by the undersigned, found to be complete in all respects and accordance with NUST Statutes/Regulations, is free of plagiarism, errors, and mistakes, and is accepted as partial fulfilment for the award of an MS degree. It is also certified that necessary changes suggested by the scholar's GEC members have been incorporated into the thesis.

Signature: ____________________

Name of Advisor: Dr. Mehak Rafiq

Date: ____________________

Signature (HoD): ____________________

Date: ____________________

Signature (Dean/Principal): ____________________

Date: ____________________

Dedication

Foremost to Almighty Allah for giving me the willpower and strength to complete my dissertation and to my family for their endless love, support, and encouragement throughout my pursuit of education. I hope this achievement will fulfil the dream they envisioned for me.

Certificate of Originality

Author Name: Haseeb Sultan

Signature: _____________

Acknowledgements

First, I want to thank and praise Almighty Allah, who has enabled me to understand, think and complete this case report work in His great mercy and benevolence. My heartiest praise goes to the holiest man in the whole galaxy, Almighty’s beloved, Hazrat Muhammad (Peace Be upon Him), humanity's reformer.

I want to express my sincere gratitude and thanks to my supervisor Dr Mehak Rafiq for showing me support and guidance for my Post Grad study; she helped me through all the time of study and writing of this report; without Dr Mehak’s constant support, and mentorship this would not be possible. I also want to express gratitude towards my GEC members, Dr Muhammad Tariq Saeed and Dr Shehzad Rasool, for their utmost support and guidance throughout the project

I want to extend a special express gratitude to the Data Analytics lab members Syed Muhammad Ameer Ghaznavi, Farhan Bashir, Mohsin Ahmed, Nimrah Ghias, Huzaifa Arshad, Mehar Masood, Mashaal Shah, and last but not least, Zunaira Said for their support, discussions, compassion, and respect. The time I spent with them will always be cherished.

Table of Contents

List of Figures

List of Tables

Abstract

Recommender systems have been widely used in various domains, including e-commerce, social media, and entertainment. In recent times, the education domain has also witnessed significant attention towards the use of recommender systems because personalized recommendations based on behavior patterns and preferences can enhance students' learning experience.

This research focuses on developing a recommender system for education using the publicly available EdNet dataset comprising over 8 million student interactions with online learning platforms. The approach involved several steps, starting with analysing of student behavior patterns. Subsequently, students were divided into groups based on their performance as, the top performers, average and low performing. This was done by analysing the time taken to attempt all questions and how many questions they got right. However, processing the massive dataset with many users and interactions was challenging. To address this, a Min-Max scaler was utilized to scale the dataset, and principal component analysis (PCA) was applied to reduce the dimensionality of the data.

The K-Means algorithm was employed to identify distinct clusters within the student population, based on their academic trajectories, with the aim of grouping together students who exhibited similar study paths. Initially, 50 clusters were utilized, resulting in a sum of squared errors (SSE) score of 6.74, which closely approached the ideal value. Subsequently, the optimal number of clusters was determined through the implementation of an elbow plot, resulting in the selection of 90 clusters, which effectively facilitated the grouping of students into cohorts based on shared characteristics. Additionally, a K-Nearest Neighbour (KNN) machine learning algorithm, in conjunction with a cosine matrix, was employed to construct a recommender system using the clusters derived from the K-Means algorithm. To validate the recommender system, students' path was tested by inputting different user and assessing the path provided. Specifically, the system takes the input of a user ID for whom recommendations are sought and provides a list of the top 6 performing users whose learning paths match that of the input user. This yielded an SSE score of 3.15, which closely approximated the ideal scenario. As a result, tailored study paths were devised for each cohort.

In conclusion, a recommender system has been developed that utilizes the KNN algorithm to generate recommendations for students based on their learning trajectories. The recommender system suggested in this study could be used by educational platforms to recommend personalized content to students based on their learning history and behaviour. This could result in a more engaging and effective learning experience for students at universities and colleges.

Chapter 1: Introduction

This chapter briefly describes the motivation behind selecting and exploring the problem of recommender systems, their different types, and the rise and acceptance of online recommendation systems. It also summarizes the core problem, which is positive or negative recommendations, its effect on online industries and its importance.

1.1 Recommender Systems

Recommender systems are algorithms that present users with appropriate items (which movies to watch, text to read, products to buy or anything else depending on industries). During the previous few decades, recommender systems have become more prevalent due to the growth of websites like YouTube, Amazon, Netflix, and many more. In some businesses, recommender systems are crucial because, when used well, they can generate significant revenue or serve as a means of differentiating oneself from rivals. For example, we can use the fact that a few years ago, Netflix organized a competition (dubbed the "Netflix prize") to create a recommender system that outperformed its algorithm as evidence of the significance of recommender systems. The winner received a prize of $1 million. As a result, Recommender systems are becoming a necessary part of our daily online activities, whether in e-commerce (suggest to buyers articles that may interest them) or online advertising (suggest to users the proper content, matching their tastes) or finding the relevant entertainment videos/audios.

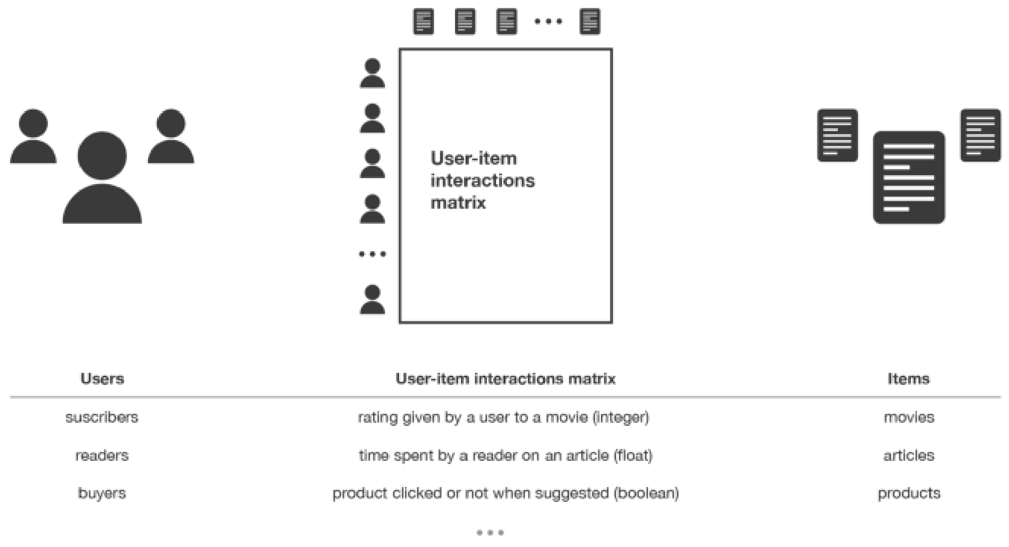
The collaborative and content-based paradigms are the two primary paradigms of recommender systems. The many forms of collaborative filtering, including user-user, item-item, matrix factorization, and content-based filtering approaches, will then be discussed in the following sections, along with how they operate. Lastly, the evaluation of a recommender system.

1.2 Categories of Recommender Systems (RS)

The objective of a recommender system is to provide personalized recommendations to users. Two main categories of techniques, collaborative filtering and content-based methods, are commonly employed to achieve this objective. A brief overview of these two primary paradigms will be presented before delving into the details of specific algorithms.

1.2.1 Collaborative filtering methods

To generate new recommendations, collaborative techniques for recommender systems rely only on the historical interactions between users and things that have been recorded. The so-called "user-item interactions matrix" contains these interactions.



Users	User-item interactions matrix	Items
suscribers	rating given by a user to a movie (integer)	movies
readers	time spent by a reader on an article (float)	articles
buyers	product clicked or not when suggested (boolean)	products
	...	

Figure 1: Collaborative filtering method

The fundamental tenet of collaborative approaches is that prior interactions between users and items are sufficient to identify similar users or items and to provide predictions based on these estimated proximities.

The two sub-categories of collaborative filtering algorithms are commonly referred to as memory-based and model-based techniques. Memory-based techniques are primarily based on nearest neighbours' searches and work directly with values of recorded encounters, assuming there is no model (for example, find the neighbouring users from a user of interest and suggest the most popular items among these neighbours). Model-based techniques assume that a "generative" model underlies the user-item interactions and seeks to identify them to develop new predictions.

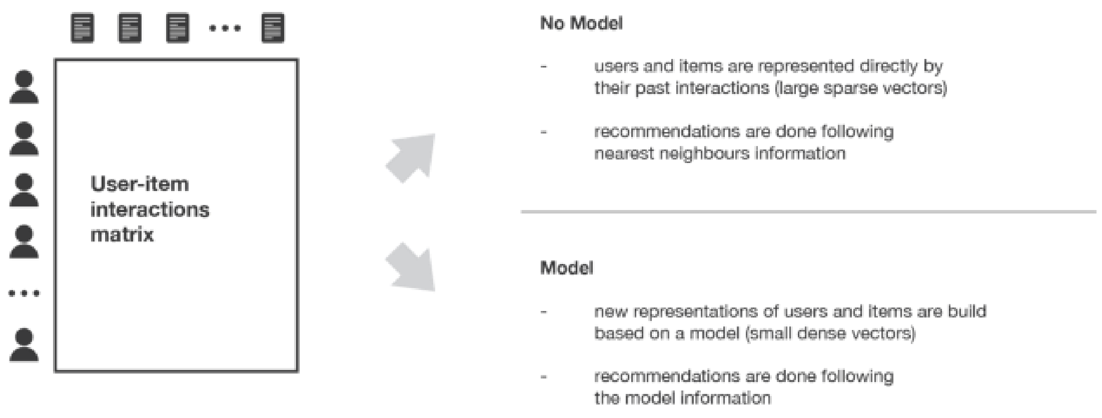

User-item interactions matrix
No Model
- users and items are represented directly by their past interactions (large sparse vectors)
- recommendations are done following nearest neighbours information
Model
- new representations of users and items are build based on a model (small dense vectors)
- recommendations are done following the model information

Figure 2 : Overview of Collaborative filtering methods

The real benefit of collaborative techniques is that they may be applied in various circumstances because they do not need to know anything about the users or the products. Additionally, fresh interactions recorded over time add new information and increase the system's effectiveness for a fixed set of users and goods. The more users engage with items, the more accurate new recommendations become.

Collaborative filtering, however, suffers from the "cold start problem" because it only considers past interactions when making recommendations, and it illustrates that it is impossible to recommend anything to new users or a novel item to any users, and many users or items have too few interactions to handle effectively. This problem can be solved in a variety of ways, such as by recommending random items to new users or new items to new users (random strategy), widespread items to new users or new items to most active users (maximum anticipation approach), a set of different items to different users (exploratory strategy), or, finally, by using a non-collaborative method for the early stages of the user's or the item's life.

1.2.1.1 Memory-based collaborative methods

The main distinctions between user-user and item-item techniques lie in their utilization of data from the user-item interaction matrix and their reliance on model-based assumptions when generating recommendations. User-user filtering involves recommending items based on the preferences of similar users, while item-item filtering recommends items based on their similarity to items that the user has liked or interacted with. Both approaches do not make any model-based assumptions and solely rely on historical user-item interaction data for generating recommendations.

The user-user technique attempts to generally find users with the most alike "interactions profile" (nearest neighbours) to suggest goods that are the most well-liked amongst these neighbours (and that are "new" to our user). As it represents users based on their interactions with products and assesses distances between users, this method is regarded as "user-centred."

In the process of generating recommendations for a specific user, the user can be represented as a row or a vector in the interaction matrix, capturing their interactions with other items. Subsequently, a "similarity" score can be calculated between the target user and all other users in the system. This similarity score serves as a measure of resemblance or affinity between the preferences or behaviors of the target user and other users. Various techniques such as cosine similarity, Pearson correlation, or Jaccard similarity, among others, can be employed to compute this similarity score. The similarity score is used to identify users with similar interests or preferences, and serves as a basis for generating recommendations for the target user. Users with similar interactions on the same items will be considered close according to the chosen similarity metric. Once the similarity to each user has been computed, the k-nearest-neighbours to the target user can be identified. Then, the most well-liked items among these neighbours, which the target user has not yet interacted with, can be recommended.

When calculating user similarity, the number of "common interactions" (the number of items both users have considered) should be considered. It is essential to prevent a situation where a user with 100 interactions in common with the reference user is considered "closer" to the reference user than another user with 98% agreement on those interactions. Therefore, they are considered similar when two users have engaged with several everyday objects similarly, such as by rating them similarly or spending a comparable amount of time interacting with them.

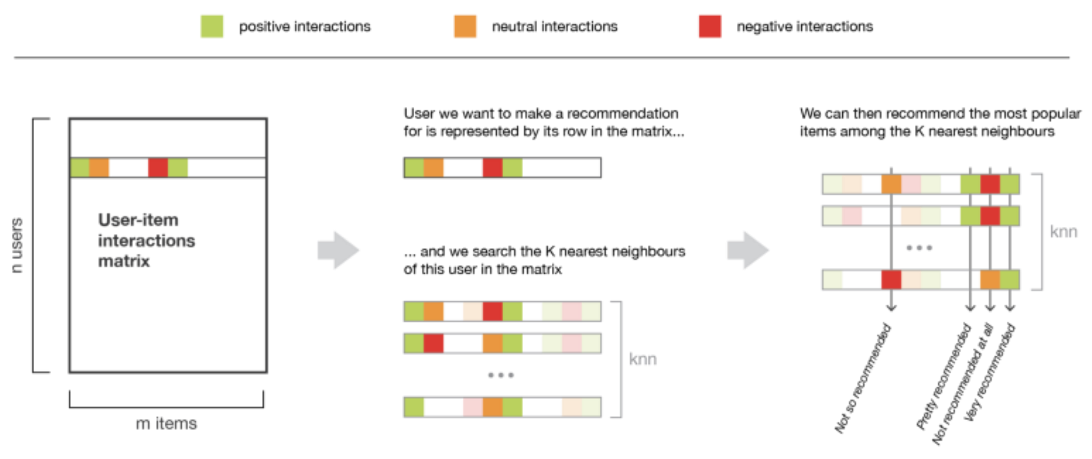

positive interactions
neutral interactions
negative interactions
n users
User-item interactions matrix
m items
User we want to make a recommendation for is represented by its row in the matrix...
... and we search the K nearest neighbours of this user in the matrix
knn
We can then recommend the most popular items among the K nearest neighbours
knn
Not so recommended
Pretty recommended
Not recommended at all
Very recommended

Figure 3: Illustration of User-User recommendations method

The concept behind the item-item method is to locate things comparable to those the user has already "positively" interacted with to offer a new recommendation to them. When most users who have interacted with two objects did so in a similar manner, it is when two items are said to be comparable. This approach is "item-centred" since it represents items based on user interactions and assesses distances between those objects.

When recommending items to a specific user, the item the user has liked the most is considered the first consideration. This item, along with all other items, is represented by its vector of interactions with each user, which can be represented as "its column" in the interaction matrix. Next, the degree of similarity between the "best item" and all other items is calculated. The k-nearest-neighbours to the chosen "best item" that is new to the user of interest are then identified and suggested to the user after similarities have been calculated. This process can also be repeated for other items the user has liked to obtain more relevant recommendations. In this scenario, items similar to multiple of the user's favourite products can be suggested.

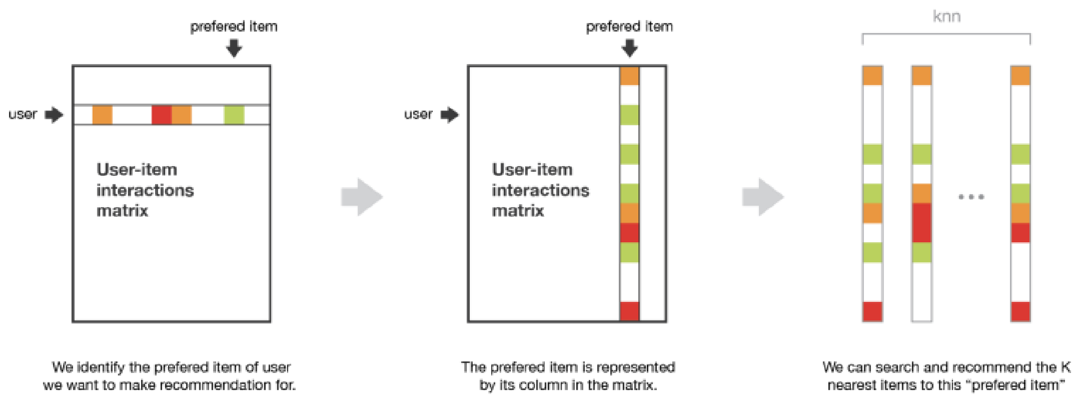

prefered item
user
User-item interactions matrix
We identify the prefered item of user we want to make recommendation for.
prefered item
user
User-item interactions matrix
The prefered item is represented by its column in the matrix.
knn
We can search and recommend the K nearest items to this "prefered item"

Figure 4 : Illustration of Item-Item recommendations method

The user-user approach aims to find users who have had similar interactions with products. As most users only interact with a limited number of items, this approach is highly sensitive to recorded interactions, resulting in high variance. However, as the final recommendations are primarily based on interactions observed between users who are similar to the user of interest, the results are more personalized and have lower bias.

In contrast to the user-user technique, the item-item technique in collaborative filtering aims to identify items that exhibit similar interaction patterns with users. The neighborhood search in the item-item approach is less sensitive to single interactions, as items generally have a higher number of interactions from many users, resulting in lower variance. However, as a trade-off, this approach incorporates interactions from all types of users, even those who may have dissimilar preferences compared to the reference user, which can make the recommendations less individualized and more biased.

As a result, while the item-item technique may be more resilient in generating recommendations due to its reliance on a larger pool of interactions, it may also be less personalized compared to the user-user approach. The item-item approach takes into consideration interactions from diverse users, including those who may have significantly different preferences from the target user, resulting in a less individualized recommendation approach. Nonetheless, the item-item technique can still be a valuable approach in certain scenarios, offering robustness and effectiveness in generating recommendations for users.

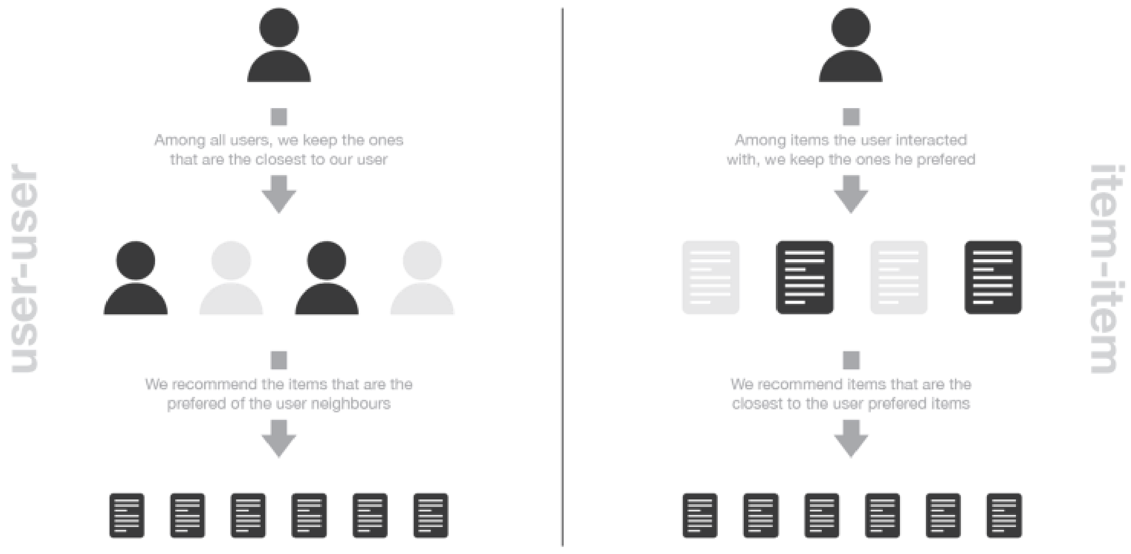

user-user
Among all users, we keep the ones that are the closest to our user
We recommend the items that are the prefered of the user neighbours
item-item
Among items the user interacted with, we keep the ones he prefered
We recommend items that are the closest to the user prefered items

Figure 5: Difference between User-User & Item-Item

1.2.1.2 Model-based collaborative methods

Model-based collaborative techniques in recommender systems solely focus on user-item interactions and assume a latent model that captures these interactions. One example of such techniques is matrix factorization, which involves breaking down the large and sparse user-item interaction matrix into smaller and denser matrices. This results in a user-factor matrix that is multiplied by a factor-item matrix, which contains item representations. The goal of this approach is to capture underlying patterns and relationships between users and items, and use them to generate recommendations. Matrix factorization and other model-based collaborative techniques leverage mathematical models to infer latent factors or representations from the interaction data, and then use these learned representations to make recommendations to users. These approaches can be effective in handling large datasets and dealing with sparsity in user-item interactions, but they do require assumptions about the underlying latent model and may not always capture all nuances of user preferences and item characteristics. Nonetheless, they are widely used and have shown promising results in various recommendation scenarios.

Matrix factorization is based on the assumption that a low-dimensional latent space of features exists, which can represent both users and items and calculate the dot product of corresponding dense vectors to determine how a user and an item interact.

Using the example of a user-movie rating matrix, one can assume that specific characteristics can describe and differentiate movies well, and these attributes can also be used to describe user preferences (high values for features that the user likes, low values for features the user does not like). However, these features do not need to be explicitly included in the model as they can be done in content-based approaches. Instead, the system will determine these valuable qualities and create representations of users and items. The extracted features have mathematical significance, but they do not have an intuitive interpretation as they are learned and not given, and thus, they are difficult or impossible to understand for humans. However, it is common for structures to emerge from this kind of algorithm that is remarkably like the sort of intuitive decomposition a human being could imagine.

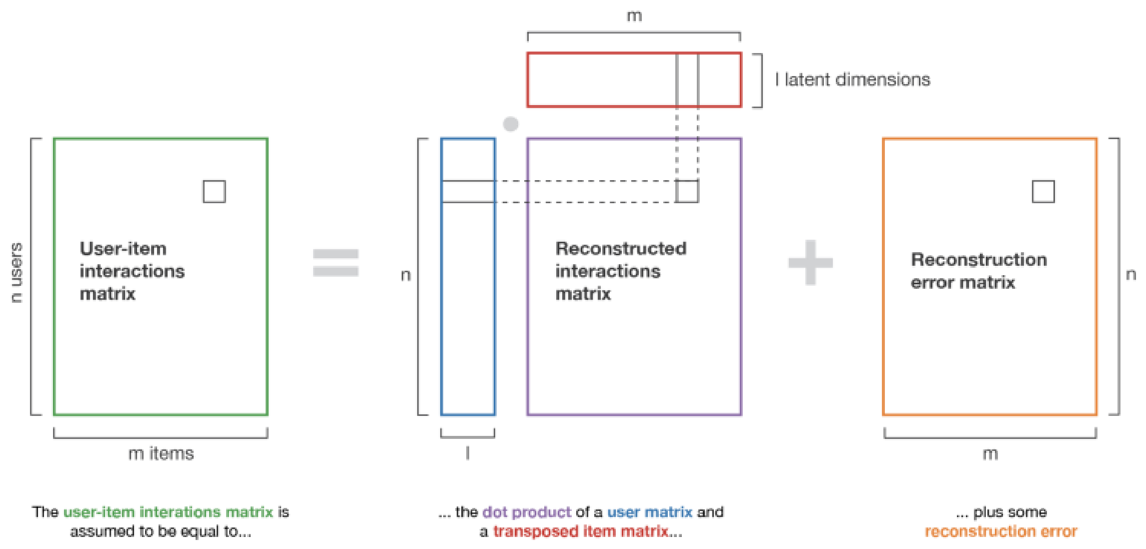

m
l latent dimensions
n users
User-item interactions matrix
m items
n
Reconstructed interactions matrix
l
Reconstruction error matrix
n
m
The user-item interations matrix is assumed to be equal to...
... the dot product of a user matrix and a transposed item matrix...
... plus some reconstruction error

Figure 6:Overview of Matrix factorization

1.2.2 Content-based filtering methods

Content-based filtering recommender system (CBF-RS) approaches differ from collaborative methods as they utilize additional information about users or items beyond just user-item interactions. Content-based filtering involves making recommendations based on keywords, attributes, or other relevant information associated with items in a database, and matching them to a user's profile. This approach focuses on the content characteristics of items, such as genre, keywords, actors, or directors for movies, or genre, author, and publication date for books, in order to identify items that are similar to a user's preferences. By analyzing the content features of items and comparing them with a user's preferences, content-based filtering can generate personalized recommendations. One advantage of content-based filtering is that it can provide recommendations for users with unique or specific preferences, even in scenarios where user-item interactions may be limited or sparse. However, a limitation of content-based filtering is that it relies heavily on the availability and quality of content information, and may not be effective in capturing complex user preferences or serendipitous recommendations. Nonetheless, content-based filtering is a widely used approach in recommendation systems and has been applied in various domains, including movies, music, books, and news articles, among others. The user profile is created based on data derived from a user's actions, such as purchases, ratings, downloads, items searched for on a website, and clicks on product links.

For instance, in the case of recommending accessories to a user who has just purchased a smartphone on a website and has previously bought smartphone accessories, the user profile would contain keywords such as the smartphone manufacturer, make, and model. Additionally, it would indicate that the user has previously bought phone holders with sleeves for credit cards. Based on this information, the recommender system may suggest similar phone holders for the new phone with added features such as an RFID-blocking fabric layer to help prevent unauthorized credit card scanning. In this scenario, the user would expect recommendations for similar phone holders, but the RFID blocking feature would be an unexpected yet valuable recommendation.

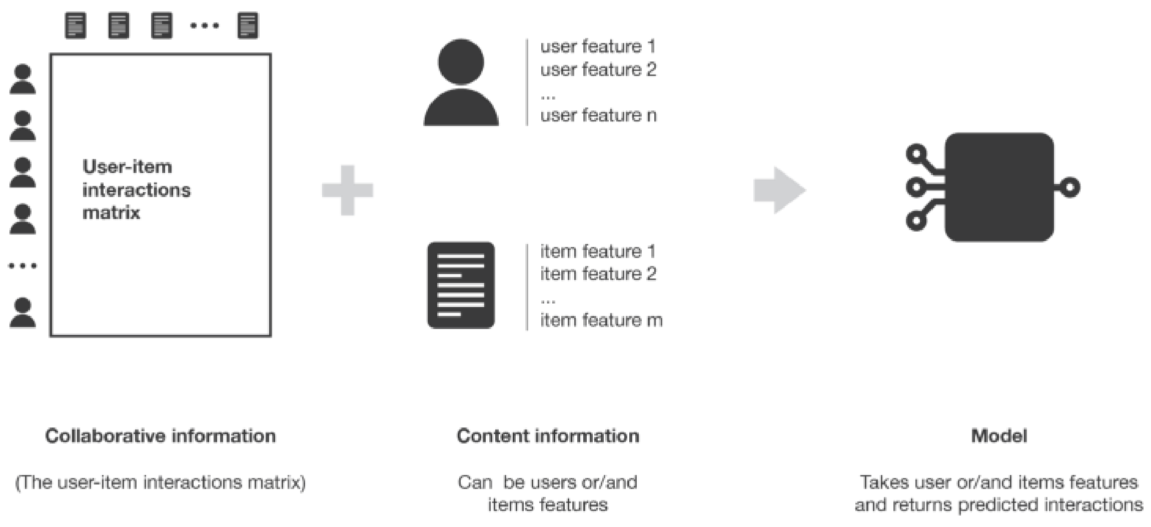

User-item interactions matrix
user feature 1
user feature 2
...
user feature n
item feature 1
item feature 2
...
item feature m
Collaborative information
(The user-item interactions matrix)
Content information
Can be users or/and items features
Model
Takes user or/and items features and returns predicted interactions

Figure 7: Overview of content-based methods

Compared to collaborative approaches, content-based solutions are far less susceptible to the cold start issue since new individuals or things may be described by their qualities (content), allowing for the creation of pertinent ideas for these new entities. This disadvantage will presumably only affect new users or products with previously undiscovered features, but once the system is well-trained, this has little to no probability of occurring.

1.2.2.1 The content-based recommender system concept

In content-based techniques, the recommendation problem can be approached as a classification or regression problem. In the classification approach, the task is to predict whether a user will "like" or "dislike" an item, while in the regression approach, the task is to predict the rating that a user will give to an item. In both cases, a model is constructed based on the available features of users or items, often referred to as the "content" in content-based techniques.

The classification approach involves training a model to classify items as liked or disliked by a user, based on the features or content associated with those items. For example, in a movie recommendation system, the features could include genre, actors, directors, or keywords, and the model would be trained to predict whether a user would like or dislike a movie based on these features. Similarly, in the regression approach, the model is trained to predict the rating that a user would give to an item, based on the content features of that item.

The content features used in content-based techniques are typically extracted from item metadata, such as textual descriptions, keywords, or other relevant attributes. These features are used to represent the content characteristics of items and are utilized to make recommendations based on their similarity to a user's preferences. Content-based techniques are particularly useful in scenarios where explicit user-item interaction data is limited or unavailable, and can provide recommendations based on the inherent content properties of items. However, like any approach, content-based techniques also have limitations, such as the reliance on the quality and availability of content information, and the potential for limited coverage of diverse user preferences. Nonetheless, content-based techniques have been widely used in recommendation systems and have demonstrated effectiveness in various domains, including movies, music, books, and news articles, among others.

A method is item-centred if the classification or regression is focused on user features. In this case, modelling, optimization, and calculations are performed "per item." To answer the question "What is the chance for each user to like this item?" (or for regression, "What is the rating given by each user to this item?"), one model is created and learned per item based on user features.

In an item-centred approach, since many people have interacted with each item, the model associated with each item is trained on data pertaining to that item. Therefore, these models tend to be relatively robust. However, even though each user's interactions are considered when building the model, as users may have diverse preferences despite sharing some characteristics or features, the method is less individualized and more biased than the user-centred method.

On the other hand, the approach is user-centred when dealing with item features because modelling, optimization, and computations are all carried out "per user." Using the attributes of the objects, a single model is trained for each user to determine the likelihood that the user will find each item appealing (or for regression, "What is the rating this user has assigned to each item?"). The model created is more individualized than its item-centred counterpart because it only considers interactions from the considered user. However, as a user often interacts with only a few items, the model produced is less reliable than an item-centred one.

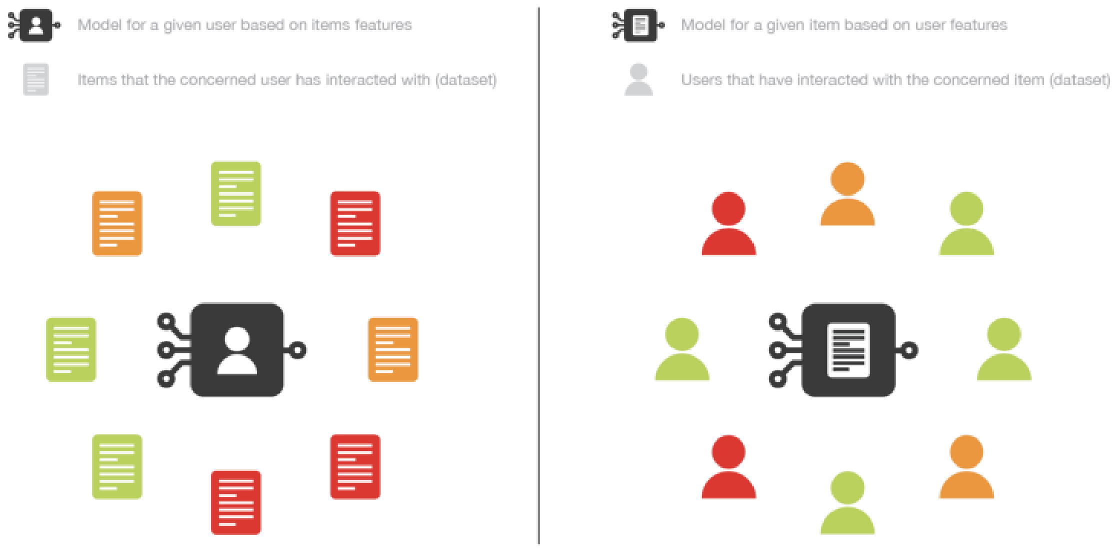

Model for a given user based on items features
Items that the concerned user has interacted with (dataset)
Model for a given item based on user features
Users that have interacted with the concerned item (dataset)

Figure 8: Difference between Item-Centered & User-centered Content-based methods

1.2.2.2 Item-Centred Content-based recommender system

In the first scenario, the classification is item-centred. The objective is to train a Bayesian classifier for each item, utilizing user attributes as inputs, with the output being either "like" or "dislike". Finally, the task is to perform the necessary computation to complete the classification process.

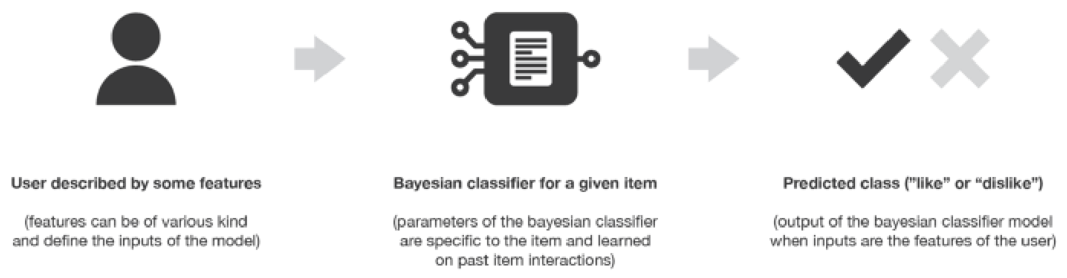

User described by some features
(features can be of various kind and define the inputs of the model)
Bayesian classifier for a given item
(parameters of the bayesian classifier are specific to the item and learned on past item interactions)
Predicted class ("like" or "dislike")
(output of the bayesian classifier model when inputs are the features of the user)

Figure 9: Item-Centered method

1.2.2.3 User-Centred Content-based recommender system

In the following scenario, the focus is on user-centred regression. The objective is to train a straightforward linear regression for each user, utilizing item attributes as inputs, with the output being the item's rating.

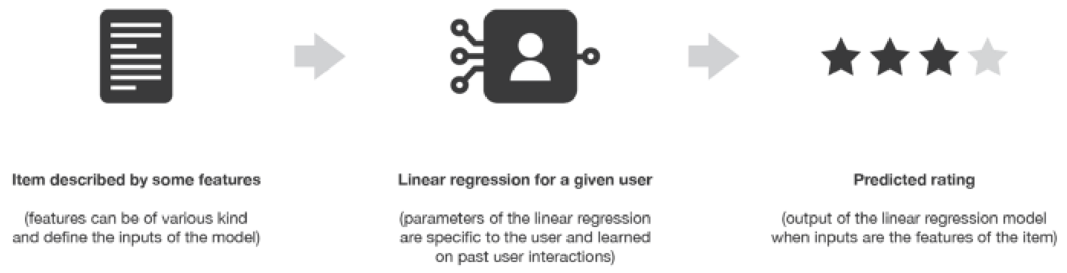

Item described by some features
(features can be of various kind and define the inputs of the model)
Linear regression for a given user
(parameters of the linear regression are specific to the user and learned on past user interactions)
Predicted rating
(output of the linear regression model when inputs are the features of the item)

Figure 10: User-Centered method

1.3 Models bias & Variances differences

In collaborative approaches that rely on a memory-based algorithm, The algorithms directly interact with user-item interactions; for instance, users are represented by their interactions with things, and suggestions are generated using the closest neighbour search on these representations. Therefore, though they have a high variance, these approaches presumably have a low bias.

Using the representation of users and the objects, the model used in model-based collaborative approaches is trained to reconstruct the values of user-item interactions. Then, new proposals might be made using this model as a guide. However, the model's latent representations of people and items have a mathematical meaning that can be challenging for humans to comprehend. Therefore, this method potentially has a more significant bias but a lower variance than other methods due to the assumption of a (free) model for user-item interactions.

In content-based techniques, the model is furnished with content that delineates the representation of users or items. For instance, users are depicted by specific features, and the model endeavors to discern the type of user profile that exhibits a preference or aversion towards a particular item. Similar to model-based collaborative techniques, an implicit assumption is made about the user-item interaction model. However, due to the predefined representations of users or items, the model operates under more constraints, resulting in higher bias but lower variance.

Content-based techniques rely on the content features of users or items, such as their attributes, characteristics, or metadata, to establish user profiles or item representations. These features are employed to define the preferences or interests of users, or the inherent properties of items. The model then utilizes these predefined representations to make recommendations based on the similarity between user profiles and item representations.

As a consequence of the predetermined representations, content-based techniques tend to exhibit higher bias, as the model is constrained by the provided content features. However, this also leads to lower variance, as the model's predictions are less influenced by idiosyncrasies of the user-item interaction data..

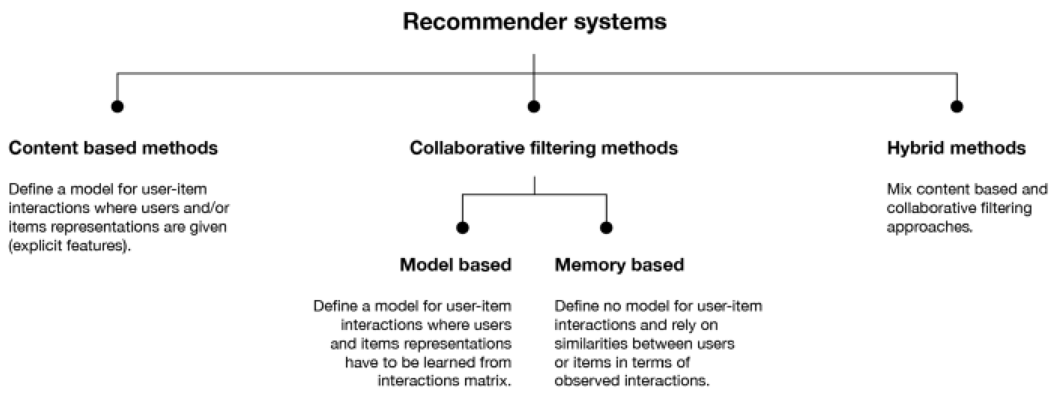

Recommender systems
Content based methods
Define a model for user-item interactions where users and/or items representations are given (explicit features).
Collaborative filtering methods
Hybrid methods
Mix content based and collaborative filtering approaches.
Model based
Define a model for user-item interactions where users and items representations have to be learned from interactions matrix.
Memory based
Define no model for user-item interactions and rely on similarities between users or items in terms of observed interactions.

Figure 11: Summary of differences between recommender systems

1.3.1 Evaluation of Recommender Systems(RS)

In order to determine the most appropriate machine learning algorithm for our specific circumstances, it is necessary to assess the performance of our recommender systems. This evaluation can be categorized into two main approaches: one that is based on clearly specified metrics and the other that primarily relies on user feedback and satisfaction estimation. In the realm of recommender systems, these two types of evaluation methods are commonly used to assess the effectiveness of the algorithms employed:

- Metric evaluations
- Human evaluations

1.3.1.1 Metric-based Recommender Systems Evaluation

The quality of outputs generated by these techniques can be evaluated conventionally using measurement metrics by “mean square error” or “sum of squared errors (SSE)” if the recommender system is based on a model that generates numerical values as rating predictions or matching probabilities. A model is evaluated on the interactions not used for training, which are a portion of the available interactions.

1.3.1.2 Human-based Recommender Systems Evaluation

In addition to models that provide accurate recommendations, other desirable characteristics can be considered when constructing a recommender system, such as diversity and explainability of recommendations. However, as the collaborative aspect suggests, it is essential to avoid trapping users in an "information confinement region."

The term "serendipity" is often used to describe a model's ability to produce diverse recommendations, and a way to measure this can be to assess the distance between suggested items.

Another critical factor in the success of recommendation algorithms is explainability. It has been demonstrated that users lose trust in the recommender system if they do not understand why a particular item was recommended. Therefore, if we create an easily understandable model, we may provide a brief justification for a recommendation (e.g., "you liked this item, you may be fascinated by this one," or "those who liked this item also liked this one").

1.4 The motivation for the study

This research aims to enhance the effectiveness of digital learning, assessments, and provide recommendations for students to review their weaker subject areas before taking final exams. Despite being the 5th most populous nation, Pakistan falls behind developed nations in terms of AI technology. To remain competitive, it is crucial for our country to make swift improvements to our educational system in order to reach a higher standard. The motivation behind choosing this topic is to ensure equal access to online education which has become the norm globally. It is firmly believed that now is the time to improve, advance, and secure digital learning systems. By making it more efficient, individuals will have the opportunity to enroll in desired institutions, acquire new skills and connect with reputable educators and researchers, ultimately benefiting them in the long run.

1.5 Research Questions

Following are the few vital questions for this research project

- How can online education be made more efficient and effective?
- How can interactions of students over time be identified?
- How can the gap between developed and developing countries in terms of AI technology be bridged in the field of education?
- How can personalized learning experiences be provided to students through the use of AI?
- How can a recommendation system be developed to function as a personalized AI-based education system?
- Will this system replace or integrate with existing education systems?
- How can the existing education system be enhanced or transformed to keep pace with the rapidly changing technological landscape?

1.6 Research Objectives

The research objectives are as follows:

- Analyzing the student's-based dataset and establishing the different pre-processing approaches to categorize the correction of interactions
- Data is passed through machine learning and AI-based models to classify consumers.
- Evaluation of machine learning algorithm through metric-based evaluation
- Developing a recommender system

The first target was achieved by performing an in-depth examination of the data using the most effective data analytics tools and techniques (data preprocessing). The secondary goal was achieved by applying and discussing the essential aspects of recommender system modelling based on the results of machine learning techniques.

1.7 Structure of thesis

Chapter 1 introduces recommender systems, including their types and a general overview. Chapter 2 conducts a literature review of recommender systems and their types, subtypes, and limitations. Chapter 3 outlines the project's methodology, including gathering data, data preprocessing, and developing a model. Chapter 4 presents the results of the steps performed in Chapter 3 methodology. Finally, Chapter 5 concludes the project with a summary of the findings and potential future research areas.

Chapter 2: Literature Review

This chapter will examine contributions made by other recommender systems researchers by offering an overview of their work. It begins with a brief history of recommender systems, followed by some examples from other businesses, before narrowing it down to recommendations.

2.1 Artificial Intelligence-Education Related Work

The 21st century is often regarded as an era of artificial intelligence and technology. Technology today plays a crucial role in every facet of life; thus, this research will focus on how technological advancement can help online education worldwide. As COVID-19 is the new standard, online education plays a vital role in our world to educate everyone. There are millions of sources which offer online education, but each has its flaws.

In recent years, breakthroughs have been made in artificial intelligence technology, which has brought about changes in people's lives. However, in all facets of life, emerging innovations evolve quickly. For example, artificial intelligence and education came into being, and education has been constantly reshaped in its form.

These reference papers [1] [2] present the connotations, features and critical technologies of applications for artificial intelligence education based on the current situation of the application of artificial intelligence in education in China,

To sort out and reflect on the advanced strategies of applications for artificial intelligence education in the intelligent era, to provide development [3] [4] Knowledge tracing, the long-standing challenge of Artificial Intelligence in Education (AIEd)., is the job of modelling the knowledge state of a student through their learning experiences over time.

Since knowledge of a student's knowledge state is a critical step for many issues of interest, including the recommendation of the learning path, prediction of scores and drop-out prediction, knowledge tracing is considered one of AIEd's most fundamental problems. Data-driven models that understand the dynamic nature of student behaviours from interaction data have become a famous formula for knowledge tracing, with advances in data science and the increasing availability of Interactive Educational

Systems (IESs) [5] [6] [7] [8] [9]. Although several datasets, such as ASSISTments [10] [11], Junyi Academy [12] [13], Synthetic [14] and STATICS [11], are available to the public and widely used by AIEd researchers, they are not large enough to leverage the full potential of data-driven models. Furthermore, the data they collect is limited to question-solving activities.

2.2 Recommender Systems Related Work

In theory, the proliferation of digital content should offer greater opportunities for users to discover content that aligns with their individual needs. However, traditional information systems users may still face challenges with information overload, as only a small fraction of the vast amount of available content is likely to be relevant to their specific interests, as supported by previous research [15].

According to research findings, students often experience information overload when confronted with a vast array of options to select from when making course selections [16]. Information overload is a phenomenon that occurs when individuals are exposed to an overwhelming amount of information that exceeds their cognitive processing capacity. The integration of advanced features in educational technologies has facilitated students' access to a wide range of information resources in diverse formats, resulting in a more complex and potentially productive information environment. However, this proliferation of information has also imposed a burden of information overload on students, as evidenced by previous studies [17]. In today's digital era, there has been a rapid surge in the amount of course-related information that is readily accessible to students [18].The process of sourcing this information from numerous websites is arduous and time-consuming. An effective search for relevant course-related information should encompass details on the course content, educational institution, as well as career prospects pertaining to the particular course subject. Therefore, assisting students in selecting from a wide range of available courses that best suit their requirements poses a significant challenge [19].

In view of the abundance of information available, it is crucial for students to engage in comprehensive search, organization, and utilization of resources to align their individual goals, interests, and current level of knowledge with their course selections. This approach ensures that students are able to make informed decisions and optimize their learning experience by selecting courses that are most relevant to their needs and aspirations.

The process of selecting suitable courses can be time-consuming for students, as it involves navigating multiple platforms, searching for available courses, thoroughly reviewing each course syllabus, and then making an informed decision. The abundance of information available has highlighted the need for guidance to assist students in selecting, organizing, and effectively utilizing resources that align with their objectives, interests, and current knowledge. Bendakir and Aïmeur [22] have identified two major challenges faced by students in pursuing education: the vast array of courses to choose from and a lack of clarity regarding the optimal course sequence to follow.

The process of selecting a course can be arduous and intricate. Online resources have facilitated quick access to information about universities and their courses [23]. However, the sheer availability of course information on university websites does not guarantee that students possess the cognitive capacity to effectively evaluate all the available options [24]. Instead, they often encounter the challenge of "information overload" [25], where the abundance of information overwhelms their ability to process and make informed decisions.

Information retrieval systems have evolved to incorporate artificial intelligence techniques that were initially developed in research. Among these, recommender systems (RS) have emerged as a promising approach to information filtering, aiding users in identifying the most relevant items [27]. Although there are several online systems that allow users to search for and locate courses, none of them provide personalized recommendations that offer comprehensive information on specifically relevant courses.

This research is motivated by the need to address the issue of information overload for users during the process of selecting a university course. The abundance of education information available on the internet in various formats poses a significant challenge in extracting relevant information that aligns with users' search queries [28]. Furthermore, the heterogeneity of course information and individual user needs further complicates the decision-making process. A promising approach to addressing this heterogeneity is through the measurement of the ontology hierarchy structure of item concepts [29].

Furthermore, providing comprehensive information about a satisfactory course option that aligns with a user's preferences presents another challenge. The variation in individual tastes and preferences can significantly impact their satisfaction levels. For instance, when searching for a university course degree, a user must gather relevant information not only about the course subject content but also the university's reputation, facilities provided, career prospects, and more. As such, establishing a comprehensive framework that can extract and integrate information from diverse sources and align this data in a unified form is a key motivation for this research.

Recommender systems have emerged as a promising approach for information filtering, offering assistance to users in identifying the most suitable items [31]. Through the analysis of individual user needs, these systems generate specific recommendations that are tailored to their preferences [32]. These systems have found successful implementation in diverse domains, ranging from e-commerce, news, movies, music, research papers, to course materials. In the field of education, recommender systems have been utilized for various purposes, including e-learning applications, academic advice, and course material recommendations. Currently, there are several online systems available for finding and searching for courses, which utilize different tools, such as the incorporation of users' prior knowledge of courses [33], keyword-based queries, collaborative filtering (CF) [34], data mining and association rules [33], and content-based filtering (CBF) models [35]. Despite the strengths of existing course recommendation systems, they also have certain significant limitations, including:

One limitation of current course recommendation systems is that models based primarily on keywords may not effectively address the specific needs and preferences of individual users during the recommendation process.

Another limitation of current course recommendation systems is that although some models utilize collaborative filtering techniques such as association rules and decision trees algorithms, the lack of historical data can make it challenging to generate accurate recommendations. This is particularly true for new students who may not have sufficient information available to the system to generate personalized recommendations.

Content-based filtering models usually focus on recommending items based on similar attributes, such as course subject or syllabus, without taking into account other factors such as user feedback or preferences. The similarity calculation is usually based on features such as keywords, topics, or course attributes, which are then weighted and averaged to determine the similarity score. However, these models do not consider the user's interaction with the deployed system, such as their feedback or ratings of previously recommended items. This can result in recommendations that are not tailored to the user's individual needs and preferences.

One notable limitation of existing course recommendation systems is their inability to furnish students with extensive insights regarding the most suitable course offerings. For instance, students need to be informed about the potential career paths associated with a particular course and require pertinent details about the facilities offered by the educational institution providing the course. The current models fail to account for these essential aspects, thus highlighting a critical gap in the current state-of-the-art. As such, there is a need to develop a more comprehensive framework that can provide students with a holistic understanding of the course offerings and their corresponding career prospects.

In order to assist students in selecting an appropriate course, it is important to categorize their needs and areas of interestOne potential solution to overcome the limitation of historical data availability is by developing methods that integrate data from multiple heterogeneous sources. This would allow for the rapid extraction of valuable course-related information, encompassing various aspects such as course content, university reputation, facilities, and career prospects. By integrating data from diverse sources, the recommendation system can enhance its ability to provide comprehensive and relevant recommendations to users, even when historical data is limited or unavailable. By doing so, students can be provided with comprehensive and personalized recommendations that take into account their specific needs and interests. Such an approach would help to address the limitations of current models, which often fail to provide students with a comprehensive understanding of the most relevant courses and their potential career paths [23].

The aforementioned concerns have instigated the development of a novel approach that intends to tackle the challenge of information overload and obtain an extensive comprehension of recommended items. Nonetheless, before implementing this approach, two research problems need to be addressed. Firstly, the integration of all existing information concerning courses, which includes course sections, job projections, and user interests, and establishing a connection between these related pieces of data. Secondly, recommending the most pertinent courses to fulfill the individual requirements of users using the integrated information.

2.3 Recommender Systems as Solution

Recommender systems, also referred to as RS, are software programs that utilize diverse information, such as items, users, and their interactions, to predict user preferences and recommend items accordingly. The quality of recommendations generated by RS can be influenced by various factors, which may vary depending on the application domain and can significantly impact the system's performance. Martinez and Lhadj [36] have identified several key factors that play a crucial role in determining the quality of recommendations in a recommender system. A summary of these factors can be found in Table 1.

Factor	**Description**
User Aspect	It is important to consider all essential characteristics of users, including their background, demographics, and language
Personal Aspect	The personal factor comprises several key aspects, including user behavior, their willingness to accept or discard recommendations, level of awareness, present mood, inspiration, confidence, perception, honesty, privacy concerns, awareness of other available possibilities, bookmarked preferences, individual requirements, and collaboration weight
Recommendation Aspect	These aspects include but are not limited to the quality, reliability, quantifiability, and weight of the suggestion, as well as its reliability, classification, and date with time. Additionally, it is recommended to provide an explanation about why a specific resource is being suggested and who the contributors are

Resource Aspect	In order to improve the quality of recommendations made by a system, several factors related to the content of items can be considered. These include the use of a thesaurus or taxonomy to categorize items, as well as the inclusion of tags, keywords, scores, assessments, synopsis, contributors, date, and number of divisions. By incorporating such information, the system can better understand the characteristics of each item and how it relates to user preferences.
System Aspect	This factor includes approachability, serviceability, considerations, objective, preliminary data, data analysis procedures, design, manner, and graphical interface. The accessibility and usability of the system are essential in ensuring that users can easily navigate and interact with the system

Table 1: Factors that influence the recommendation for recommender systems

The utilization of recommendation system(RS) filtering techniques is commonly categorized into four main methods. The first approach is the content-based filtering (CBF) method, which recommends items based on the user's previous preferences [37]. The second approach is the collaborative filtering (CF) method, which recommends items based on the preferences of users with similar needs or interests [38]. The third approach is knowledge-based filtering, which recommends items based on domain knowledge that meets the users' needs and preferences [39]. Finally, a hybrid recommendation system approach combines two or more recommendation methods to overcome the limitations of each approach [40] [41] [24]. The following subsections provide a description of these techniques, and their respective advantages and shortcomings are examined.

2.4 Types of Recommender Systems

Recommender systems are classified into several types, and their limitations are discussed below:

2.4.1 Collaborative filtering system Recommender System (CFS-RS)

Collaborative filtering is a widely utilized technique in recommender systems that is proposed to adopt the issue of information overload. The approach is based on the idea of aiding users in making decisions by relying on the thoughts of other individuals who share similar concerns. As such, the technique requires a large user community to gather and analyze vast amounts of data concerning user behavior and characteristics.

The cold start problem is a well-known challenge faced by collaborative filtering systems in situations where there is inadequate data available regarding a particular user's preferences [43]. To address this issue, the system may provide the top-rated item as a recommendation.

Collaborative filtering system recommender system (cfs-rs) is a widely recognized and extensively implemented technique in recommender systems. The technique has gained immense popularity since 1990, leading to the development of numerous recommender systems based on collaborative filtering in academia and business. The applications of these systems are vast and varied, and include suggesting courses, books, web pages, and more [44]; [45]; [46]; [47]; [48]

According to previous researchers, There exist multiple collaborative filtering RS algorithms that can be utilized to produce recommendations. However, it is worth noting that collaborative filtering algorithms are typically classified into two main categories, namely memory-based and model-based approaches, as identified by previous researchers [49].

2.4.1.1 Memory-based

Memory-based collaborative filtering RS relies on user-item ranking data to evaluate the relationship between users or items, which is then used to generate recommendations [50]. This approach is widely employed in commercial systems due to its flexibility in accommodating different types of products. However, a limitation of memory-based collaborative filtering is that the similarity metric is solely based on rating data and does not take into account the underlying reasons behind the ratings, such as the factors that contribute to a good or bad rating.

As a result, it is feasible for two users to express liking for the same item but for different reasons. In this thesis, Chapter 3 presents a novel approach that incorporates this additional dimension into the recommender system. The commonly used memory-based method, known as the neighbors' method, is categorized into two types: user-based and item-based. In the user-based method, recommendations are calculated based on the similarity between users in their consumption patterns. Specifically, the preferences of similar users and neighbors are considered when generating recommendations for a target user [51].

In the process of forming neighbors, it is essential to compute the similarity between the target user and all other users. There are several algorithms available for measuring user similarities, including Pearson's correlation [36] and cosine-based approach. These algorithms are commonly used in memory-based collaborative filtering to quantify the similarity between users and facilitate the generation of recommendations [52].

2.4.1.2 Model-based

Model-based collaborative filtering(MB-CF), on the other hand, utilizes the user-item ranking information to construct a predictive machine learning(ML) model, which is then utilized to make recommendations for unrated items. Developing a model involves utilizing various statistical and machine-learning algorithms [53]. These methods are often inspired by machine learning techniques such as Bayesian networks, artificial neural networks [54], and latent factor models. These models enable the system to capture underlying patterns and correlations within the data, allowing for accurate prediction of user preferences and generation of recommendations for items that have not been rated by the user [55].

2.4.1.3 Cold Start Problem

One challenge faced by these procedures is the "cold start problem" [56], which arises when there is insufficient data available for a new user. The cold start dilemma occurs when the recommender system(RS) lacks adequate information to make accurate predictions or recommendations for users or items that do not have enough data. This can pose a limitation, as the system may struggle to provide relevant recommendations for newfound users or items until sufficient data is collected to make meaningful assumptions and predictions.

For example, when a user(s) visits a particular recommender system(RS) for the first time, none of the items in the system would have been rated by that user. As a result, the recommender system lacks information about the user's preferences and dislikes. Similarly, the same issue arises when a new item is added to the system, as there may be little or no data available on the item's ratings or user preferences. This "cold start" scenario can pose a challenge for the recommender system in accurately generating recommendations until sufficient data is collected to make informed predictions for new users or items.

As no ratings or preferences are available for a newly added item, the recommender system lacks the information to determine its similarity with other items. Consequently, the system is unable to generate recommendations for the element until a sufficient amount of users have rated it. However, one of the strengths of collaborative filtering is that it does not depend on any prior knowledge about the items in the database. The matrix of users' ratings serves as the primary input for the collaborative filtering algorithm, enabling it to generate recommendations solely based on user-item interactions and preferences. This ability to operate without prior knowledge about the items makes collaborative filtering a versatile approach for generating recommendations in various domains.

2.4.2 Content-Based filtering Recommender System(CBF-RS)

Content-Based filtering Recommender System(CBF-RS) is a conventional approach employed to address information overload challenges [57]. This method recommends items to users based on the characteristics or features of items that have been previously evaluated by the user. However, one limitation of content-based filtering is its tendency towards over-specialization, as it may not be able to recommend unexpected or novel items to users. This means that users may only receive recommendations for items that are similar to those they have rated in the past, potentially limiting the discovery of new items. This challenge, commonly referred to as the serendipity problem, arises due to the inability of content-based filtering to recommend items that deviate from the user's known preferences or past interactions, and it can impact the system's ability to provide diverse and novel recommendations to users.

In contrast to collaborative filtering, which relies on other users' preferences, content-based filtering uses the characteristics of items that a user has liked or interacted with to generate recommendations [41]; [57]. This approach analyzes the content or features of items that have been previously rated by the user, and identifies suitable matches involving the user's profile and the item properties to make recommendations. One notable advantage of content-based filtering system(CB-FS) methods is their ability to effectively handle the new item problem, wherein recommendations are needed for items with no user feedback. Unlike collaborative filtering algorithms, which require user ratings or interactions for items in order to make recommendations, content-based methods can generate recommendations for new items that do not have any user feedback available. This makes content-based filtering an attractive option for addressing the challenges associated with recommending new or less-rated items to users.

Moreover, content-based filtering has gained popularity in generating recommendations for information items based on user interactions such as marking or buying specific objects of interest [58]. The method then recommends items that are most similar to the user's favored items. Content-based filtering relies on detailed information about the items in the database and the user profile, which describes the user's preferences. However, unlike collaborative filtering, content-based filtering does not require a substantial convergence of users for generating recommendations. Instead, it leverages the content or features of items and the user's profile to make personalized recommendations, making it a viable option even in scenarios where user data may be limited.

Every content-based recommender system aims to achieve the following three goals:

- to analyze the explanations and records associated with items in the database. This involves extracting relevant features, such as keywords, genre, or other attributes, from item descriptions or documents to understand the content and characteristics of the items.
- to build a comprehensive profile of the user's preferences based on their interactions with items. This profile captures the user's preferences in terms of the content or features of items, which are inferred from their liked or rated items.
- the user's favorite items with other items in the database based on their content similarity. This involves measuring the similarity between the features of the user's favorite items and the features of other items in the database, and identifying items that are most similar to the user's preferences

The architecture of this recommender system was published in [58]. The authors describe the three main components that contribute to achieving the aforementioned goals:

- Content analyzer component of the system is responsible for extracting structured and relevant information, typically in the form of keywords, from texts for further processing.
- Profile learner component collects information about user interests, such as their item selections, ratings, and feedback, utilizing machine learning algorithms [60], and creates a user profile. This profile is used to capture the user's preferences and interests for generating personalized recommendations.
- Filtering component of the recommender system utilizes the user profile to recommend relevant items by matching the user's profile with corresponding items in the database. This process involves using various similarity metrics, such as cosine similarity [61], to determine the similarity between the user profile and items in the database. Based on the similarity measures, a ranked list of items is generated, and these items are suggested to the user as recommendations.

Content-based filtering system(CB-FS) recommendation have garnered attention from diverse domains, including data retrieval, and machine learning. For instance, early proposals for web recommendations incorporated term-weighting models from information retrieval. These models are utilized to represent the content or features of items, such as keywords or attributes, in a quantitative manner. Term-weighting models assign weights to different terms or features based on their relevance or importance to the items, enabling the algorithm to analyze and compare items effectively [62].

Nevertheless, the existing content-based filtering approaches may not accurately model the data. Utilizing ontologies to model the data can result in higher quality modeling and, subsequently, more appropriate recommendations. Improved-modeled items can capture more nuanced user(s) preferences, leading to more accurate similarities and recommendations. By incorporating ontologies, which provide a structured representation of domain knowledge, content-based filtering algorithms can enhance their modeling abilities and produce more precise recommendations [63].

Semantic web technologies(SWT) have been proposed as methods for content-based recommendations in various domains, such as news recommendations [64], as well as movie and music recommendations that leverage Linked Open Data. These approaches utilize the semantic web's rich representation of data and knowledge to enhance the recommendation process. By leveraging Linked Open Data, which is a vast collection of interconnected and semantically annotated data, these methods can enrich the content-based filtering approach with additional contextual information, leading to more refined and context-aware recommendations. Such approaches demonstrate the potential of semantic web technologies in enhancing content-based recommendation systems [65].

Mooney and Roy [66] employed Bayesian classifiers as a technique for book recommendations within the domain of machine learning. Similarly, Pazzani and Billsus [67] utilized a diverse range of techniques including Bayesian classifiers, clustering algorithm(s), decision trees algorithm(s), and artificial neural networks(ANN) for website recommendations. These machine learning techniques were applied to analyze user preferences and item characteristics, and to make recommendations based on learned patterns and associations. The use of such advanced machine learning techniques in recommendation algorithms showcases the application of cutting-edge methodologies for improving the accuracy and effectiveness of content-based recommendation systems.

2.4.2.1 Limitations of Content-based filtering system(CB-FS)

Content-based recommender systems(CB-RS) have numerous restrictions recognized in the literature [68]. Among these limitations, the most relevant ones include:

• **Limited content analysis limitation:** Content-based recommendations(CB-RS) are limited by the features that are clearly correlated with the items to be endorsed. For instance, in the case of course recommendations, content-based approaches can only rely on information such as the course title, course fee, and university name, which may not capture the complete set of factors that influence a user's preference for a course.

The effectiveness of content-based recommendation techniques relies heavily on the availability of descriptive data. It is necessary to have a good set of features that can either be automatically parsed by a computer or manually extracted with ease from the content. However, fulfilling these requirements can be challenging in many cases. For instance, obtaining content in a format that can be easily parsed or manually extracting relevant features may pose difficulties, limiting the effectiveness of content-based recommendation approaches.

Automatic feature extraction can be particularly challenging in certain domains, where the content is not easily amenable to automated parsing. For example, multimedia data/information, such as images, videos, and audios, pose greater difficulties in applying automatic feature extraction methods compared to text content. Manual assignment of features to a model may also not be practical in such cases [69]. While recent attempts have highlighted the need for further investigation in this direction, the challenges associated with automatic feature extraction from multimedia data remain significant.

One recent trend observed in the field of content-based recommender systems is the utilization of external knowledge sources, such as ontology-based ones, to enrich the content representation. For example, the Explicit Semantic Analysis (ESA) approach, which was introduced in [70], employs an indexing technique that incorporates content obtained from Wikipedia articles. Additionally, there have been early efforts to integrate content-based filtering based on ontology with techniques for knowledge infusion, as proposed in [71]. The objective of these approaches is to enhance the content representation by incorporating external knowledge sources, which could potentially result in improved accuracy and relevance of recommendations.

• **Content over-specialization:** Content-based filtering(CB-FS) based on ontology involves retrieving items that have high similarity scores against a specific user profile, using the ontology as a knowledge source. However, one limitation of content-based techniques is that they are unable to recommend items that are substantially different from what the user has previously seen or liked. This can result in a lack of serendipity or novelty in the recommendations, as they are solely based on the user's past preferences and may not introduce the user to new or unexpected items.

To address this limitation, it may be necessary to introduce an factor of arbitrariness in the suggestions. For example, in cases where a user has no prior experience or preference for a particular genre or type of content, such as ambient music, traditional content-based filtering may not provide any recommendations in that category. By incorporating randomness or serendipity into the recommendation process, users can be exposed to new and unexpected items, even if they have not shown explicit preference for similar content in the past. This can help broaden the scope of recommendations and enhance the overall recommendation experience for users, enabling them to discover new interests and expand their horizons beyond their known preferences [72].

Alternative approaches, such as the one implemented in Daily Learner [73], propose filtering out items not only if they are too altered from the user's favorites, but also if they are too closely related to something the user has previously watched. This approach aims to strike a balance between novelty and relevance in recommendations. By filtering out items that are too similar to previously viewed items, the system encourages diversity in recommendations and prevents over-exposure to similar content. This approach can help users discover a wider range of items while still maintaining relevance to their preferences, and can be particularly useful in scenarios where users seek both novelty and relevance in their recommendations.

Furthermore, in [74], a set of five severance measures is delivered to assess whether a document viewed to be relevant contains novel knowledge. These measures assess the extent to which the content of a document is redundant with respect to other documents in the system. This can help identify documents that offer unique or novel information, rather than duplicating content that has already been presented to the user. By incorporating redundancy measures into the recommendation process, systems can better ensure that recommended items are not only relevant but also provide valuable and novel information to users.

• **Cold start:** To ensure that a content-based recommender system accurately understands users' preferences and provides reliable recommendations, it typically requires users to rate a sufficient number of items. This allows the system to gather enough data on each user's preferences and behavior to generate meaningful recommendations. Without an adequate number of ratings from users, the system may lack the necessary information to accurately profile and understand individual preferences, which can result in less reliable recommendations. Therefore, user engagement and participation in rating items are crucial for the effectiveness and accuracy of content-based recommender systems.

The advent of Web 2 , 3 and public platforms has revolutionized the landscape of user outlining in content-based recommender systems. With the wealth of information that users share on social media platforms, including comments, posts, tags, and data from social networks, it is now possible to leverage this data as a preliminary point to develop and model user profiles incrementally. This means that instead of solely relying on users to rate items, recommender systems can tap into the data users have already provided on social platforms, allowing for a more comprehensive and dynamic user profiling approach. This enables the system to adapt and update user profiles over time, taking into account the changing preferences and behavior of users, and potentially leading to more accurate and personalized recommendations. The availability of user-generated data from social platforms has opened up new opportunities for improving the performance and usability of content-based recommender systems by leveraging the rich information available on the web.

Social media-based user profiling has emerged as a recent trend in this area [75]. It is important to note that content-based(CB) techniques do not rely on the accurate content of items. For instance, in the context of course recommendation, a content-based filtering system(CB-FS) typically does not examine the actual content of the course. Instead, it may rely on descriptors such as course title, keywords, or metadata, such as genre or author, at best. The textual content related to items, generated by users, blogs, or other sources, is often considered as the "content" for content-based filtering methods. These methods may also incorporate semantic analysis using ontologies to derive meaningful information from the textual content.

2.4.3 Knowledge-Based filtering system

These techniques such as case-based reasoning, ontology-based systems, and rule-based systems are often used for explicit representation of knowledge in recommendation systems. These systems leverage past problem-solving experiences, ontologies, or rules to make inferences about a user's interests and preferences. For example, a case-based system may use past cases (i.e., previously solved problems) as a core foundation of knowledge to resolve a new problem, while an ontology-based system may utilize domain-specific ontologies to represent and reason about knowledge for recommendation purposes. Rule-based systems, on the other hand, use predefined rules to make recommendations based on user preferences and other relevant factors.

An exemplary illustration of the knowledge-based method (KB-FS) can be found in the work of Burke [41]. In his research, the model was specifically proposed to assist users in finding restaurants that align with their preferences through the use of interactive dialogue. Users were able to refine their search queries based on their interests and modify the retrieved suggestions until they arrived at a suitable option. Another knowledge-based approach involves employing semantic similarity to make recommendations to users. Ontologies have been widely applied in various recommender systems to mitigate content heterogeneity and enhance content retrieval. For instance, in a study by [77], promising outcomes were achieved in dealing with content heterogeneity by utilizing subsumption hierarchies to generalize user profiles.

Additionally, the idea of the semantic web(SWT) has been employed to enhance e-learning. For instance, in a study by Yang et al. [78], a semantic(SWT) recommender system approach was proposed for use in e-learning, aiming to assist students in defining appropriate learning objectives. Furthermore, the system could provide suggestions to teachers for newfound resources that could be accepted to enrich the course curriculum.

The approach described in [48] was developed with an extension for query keywords, incorporating semantic relations and ontology reasoning. This personalized ontology-based recommendation system, similar to the approaches mentioned earlier, represents items and user profiles to deliver personalized services utilizing semantic web applications.

The evaluation results demonstrate that the utilization of semantics-based methods in the recommender system leads to improved accuracy in the recommendations.

Additionally, an ontology-based recommendation system can effectively address the issue of the cold start problem, which arises when there is insufficient information from the past to make accurate recommendations [79]. This problem often occurs when dealing with new users who have not provided enough ratings or preferences, making it challenging to generate reliable recommendations. Therefore, a proposed e-learning personalization model based on ontology takes into account the past preference history of learners to recommend suitable learning objectives, overcoming the limitations of the cold start problem.

Similar to traditional systems, this ontology-based recommendation system also faces challenges with the new user problem, where limited information is available for new users, and it is constrained to recommending learning objectives only [80]. However, the utilization of ontology structures can greatly enhance the accuracy of the system [49]. For example, the use of "IS-A" relations in ontology, where concepts are organized in a hierarchical tree with associations represented as "IS-A" relationships, can improve the measurement of semantic similarity. This hierarchical representation can potentially result in more accurate calculations of similarity between concepts, mitigating some of the limitations of the system.

The knowledge-based (KB_FS) recommender system has both benefits and shortcomings [81]. On the positive side, it does not suffer from the cold start problem, as it can make recommendations to new users based on superficial knowledge of their preferences. This means that users do not need to rate or purchase a large number of items for the system to specify meaningful recommendations.

On the negative side, the knowledge-based (KB_FS) recommender system faces scalability problems. Calculating similarities for a large case base requires more time and effort compared to other prevailing recommendation methods. Additionally, the knowledge-based (KB_FS) method relies heavily on including information about items, users, and functional knowledge to generate recommendations, which can be a weakness of the system.

2.4.4 Hybrid-Based filtering system

Hybrid recommender systems combine multiple methods, as stated above, to conquer their difficulties and leverage their strengths [41]. For example, collaborative filtering(CB-FS) techniques may struggle with the novel-item problem, where unrated items cannot be recommended. However, content-based methods can utilize available information about new item characteristics. One approach to hybridization is to select different methods and permit each to produce a independent ranked list, which is then merged into a final list [82]. Other hybrid approaches involve one primary model that incorporates a secondary model to address limitations such as the cold start crisis or to enhance user profiles [83].

Burke [41] proposed a taxonomy for hybrid recommender systems, categorizing them into the following seven categories:

• **Weighted:** One approach in hybrid recommender systems is to combine recommendation component scores numerically, often using an additive formula. For example, a straight sequence of recommendation results, such as in P-Tango [82], is a commonly used method. In this approach, equal weights are initially assigned to collaborative and content-based recommenders(CB-FS), but the weighting is dynamically improved based on users' feedback on predictions, allowing for continuous refinement of the recommendation process.

• **Switching:** In some hybrid recommender systems, a particular component is chosen and applied from the available recommendation components based on certain conditions. For example, the recommendation process may start with a content-based (CB-FS) recommender, but switch to a collaborative one when the confidence level in the recommendations even now presented is not enough, as demonstrated in the work of Billsus and Pazzani [84]. However, this switching hybrid approach does not completely avoid the ramp-up problem, as both the collaborative and content-based (CB-FS) systems may still face the another user problem.

• **Hybrid:** Another approach in hybrid recommender systems involves merging and presenting multiple ranked lists of recommendations together into a single ranked list. This method, known as list merging, is implemented in the PTV system [85]. Different recommender systems produce their recommendations, and these recommendations are combined and presented as a unified ranked list using list merging techniques.

• **Feature Combination**: The hybrid recommender systems in this category typically consist of two different recommendation components: contributing and actual recommenders. The actual recommender's functioning relies on the data that has been modified or enriched by the contributing recommender. This can involve leveraging features or information from one source and incorporating it into the recommendations generated by the other component's source.

• **Feature Augmentation**: The main difference between this category and the feature combination hybrid is that in this category, the contributing recommender provides novel characteristics or features that are then used by the actual recommender, making it more flexible. The contributing recommender enriches the data that is used by the actual recommender, allowing for a more dynamic and adaptable approach to generating recommendations.

• **Cascade**: In the cascade hybrid category, different recommendation components are assigned priorities, and the lower-priority recommenders act as tiebreakers over those with higher precedence. Typically, one recommendation technique generates a set of candidates, and the second system filters and re-ranks this list to generate the final list of recommendations. This approach can be more efficient compared to the weighted approach where all methods are applied to all items, as it allows for a more selective and efficient use of different recommendation components based on their priorities.

• **Meta-level:** The utilization of the output model from one recommender as input for another recommender is a characteristic of a meta-level hybrid approach, as described by Burke [41]. In contrast, feature augmentation hybrids involve the use of a discovered model to produce additional characteristics for input to a second system. In a meta-level mixture, the entire model produced by one recommender is used as input for another recommender, allowing for a more comprehensive integration of different recommendation components

2.4.5 Recommendation Systems in Higher Education Domain System

The recommendation problem has widespread applications across various domains, with particular prominence in e-commerce and entertainment. Many advanced recommendation systems are designed to assist users in making informed decisions about which products to purchase or utilize. Effective implementation of e-commerce recommenders has been reported in relevant literature [86], showcasing the practical relevance and effectiveness of such systems in real-world scenarios.

The success of recommender systems in other domains, as cited earlier, has also inspired the completion of such systems in the educational domains [87]. In the context of education, the goal is to facilitate knowledge acquisition among learners, while educators provide support throughout the learning process. The application of recommender systems in the educational domain holds great promise in enhancing the learning experience and promoting effective learning outcomes for students.

In other words, the distinct characteristics of different domains impact the design of recommender systems [88]. In the case of the education domain, the objective of recommending items, whether directly or indirectly, is to improve the administrative procedure and aid in the collection of appropriate courses for students. In the educational context, a recommendation system serves as an intellectual mediator that suggests various alternatives to students, taking into consideration their academic performance and other personal information, based on the actions of other students with similar characteristics [89]. It is worth noting that while enrolling in a course is a necessary step, the crucial aspect lies in the decision-making process that precedes it [90]. Recommender systems in the education domain play a significant role in assisting students in making informed choices about their course selections, ultimately contributing to a more effective and personalized learning experience.

In this section, the literature on recommendation systems in the education domain will be reviewed, focusing on the element of the items recommended and the various methods used for item recommendations. It should be noted that recommender systems in education differ significantly from those in e-commerce, as they need to take into account not only the preferences of scholars or educators for specific knowledge materials, but also how these materials can help them achieve their learning goals [91]. Hence, in Table 2.2, a comparison has been made to highlight the key distinctions and factors between a universal-purpose recommendation system and an scholastic recommendation system.

Difference factor	**General recommendation system**	**Educational recommendation system**
Goal-factor	In industries such as e-commerce, users often search for products based on specific characteristics and price ranges [88].	Educational recommender systems are designed to assist users, or groups of users, in finding suitable learning resources and activities that optimize the

		realization of learning targets and the development of competencies in a more efficient manner [88]
Context-factor	Many recommender systems commonly utilize factors such as networks and peer information, as evidenced in various studies ([92]; [36]; [93]; [94]).	The perspective of educational recommender systems is educationally related and should take into consideration factors such as pre- and post-requisites, timeframe, instructional design ([36]; [93]), and social networks ([95]).
User Inclined-factor	Recommender systems are primarily based on user judgments, peculiar preferences, or user's likes and disgusts [93]; [96].	Recommender systems in the educational domain are highly influenced by pedagogical factors such as learning history, knowledge, preferences, processes, strategies, styles, patterns, activities, feedback, misconceptions, weaknesses, progress, and expertise ([97]; [98]).

Table 2: Difference between the factors in a general recommender system and an educational recommender system

Numerous recommender systems(RS) have been projected for the education realm, catering to various stakeholders such as scholars, instructors, or educational advisors. The recommendable items in this domain include educational resources, universities, and data related to courses, topics, student accomplishment, and field of education. These recommender systems are designed to facilitate teaching and academic guidance in the education domain.

In this thesis, the proposed methodology was evaluated with a focus on reducing information overloading for students who need to make decisions about university courses. The aim was to provide them with the most appropriate recommendations to streamline their decision-making process in selecting the right university course.

In the context of this thesis, the term "course" refers to a program of studies, such as undergraduate or postgraduate programs. Various types of research have been conducted, employing different methods and algorithms, to recommend courses to students. For example, Sandvig and Burke proposed the Academic Advisor Course Recommendation Engine (AACORN), which employed a case-based solving methodology that utilized past cases to solve new problems [30]. Their system leveraged course histories and past students' experiences as the foundation for assisting future students in making informed decisions about course selection.

Furthermore, it has been observed that the career aspirations of students play a crucial role in their course selection decision-making process [99]. Farzan and Brusilovsky substantiated this notion through their reported course recommendation system, which was based on an adaptive community [21]. In their approach, they utilized a social steering method to analyze students' assessments of their career targets, with the primary objective of obtaining specific feedback implicitly as part of their natural interaction with the system. This approach recognized the significance of considering students' intended future career as a vital factor in providing relevant course recommendations.

Artificial Intelligence techniques offer the potential to enhance human decision-making and reasoning processes, reducing uncertainty in active learning and promoting lifelong learning mechanisms [100]. Hence, the challenge for recommender systems in the education domain is to gain a deeper understanding of students' interests and their learning goals [24]. One approach to address this challenge is through association mining-based recommender systems, which leverage data on student performance and similarity with other students to recommend learning tasks that are most suitable for individual learners [101]. Furthermore, a course recommendation system has been proposed to assess the similarity between university course programs and students' profiles, providing tailored recommendations based on individual needs and preferences.

The proposed basis in this thesis is broad, as it combines content-based(CB-FS) and collaborative filtering techniques with an ontology-based approach to address the data overload issue. The framework utilizes a hierarchical ontology to map course profiles with user (student) profiles, enabling a better understanding of recommended items. Two novel methods are developed to extract and integrate data from multiple sources, aligning the data appropriately for ontology mapping. This enhances the system's ability to provide comprehensive recommendations.

Moreover, the proposed approach addresses the new user difficulty by computing the ontology comparison between users' profiles based on their ratings for each item. This enables the system to provide personalized recommendations even for new users who may not have a long history of interactions with the system. Finally, the proposed recommender system evaluates the ontology similarity between the item and users' profiles in a hierarchical manner, ensuring that the recommended courses match the students' requirements and align with their intended program of study before enrollment. This approach enhances the accuracy and relevance of the recommendations provided by the system.

2.4.5.1 Course Recommender system

Previous research has indicated that students often face information overload when it comes to choosing a course [23]. Furthermore, the amount of course-related information available to students has been rapidly increasing over time. This abundance of information can overwhelm students and make the decision-making process challenging and complex. As a result, there is a need for effective recommender systems that can assist students in making informed decisions by providing personalized and relevant recommendations tailored to their individual preferences, needs, and career goals. The proposed framework in this thesis aims to address this issue by combining content-based and collaborative filtering techniques with an ontology-based approach to reduce information overload and provide accurate recommendations for university courses.

The plethora of available information has led to a growing need for assistance in helping students choose, organize, and effectively utilize resources that align with their objectives, interests, and existing knowledge [21]. With the increasing availability of course-related information, students may struggle to sift through the vast amount of data and identify the most relevant and suitable resources for their specific needs. As such, there is a demand for recommender systems that can offer personalized and targeted recommendations, taking into consideration students' individual characteristics, preferences, and goals. The proposed framework in this thesis aims to address this need by utilizing a comprehensive approach that integrates content-based and collaborative filtering techniques with ontology-based mapping to assist students in making informed decisions and effectively utilizing available resources in line with their academic objectives.

According to Dr Cable, a British politician who served as the Secretary of State for Business, applying to university is a significant decision, and it is important to ensure that all students, regardless of their background, have access to key facts to help them make informed choices [citation]. Bendakir and Aïmeur also highlight that students pursuing education face challenges related to the multitude of courses available and a lack of knowledge about which courses to take and in what sequence [citation]. Furthermore, the heterogeneity of course information and the individual needs of users can make the decision process complex and laborious [citation]. In line with this, Wendy Hodgkiss, a careers adviser with the 'Which? University' organization, advises against making impulsive decisions and encourages students to conduct thorough research to ensure that they are choosing the relevant university and course [102]. The abundance of course options, lack of knowledge about course sequencing, and personal preferences make the decision-making process for students overwhelming and demanding. Therefore, the development of effective course recommendation systems that can provide tailored and relevant recommendations to students can greatly assist them in making informed decisions about their educational pathways.

Amer and Jamal's research demonstrated that students' course choice decisions are influenced by their background and personal or career interests [103]. However, simply providing more course information on university websites does not guarantee that students will have the cognitive ability to evaluate all the options as alternatives. In fact, it often leads to a problem known as "information overload" [23; 104; 105; 106]. Even with the availability of search tools, evaluating all the course alternatives can be challenging for students. Therefore, the automated identification of relevant courses that match students' needs is a pressing problem [90]. The development of effective course recommendation systems that can automatically and accurately match students' interests, background, and career goals with suitable courses can provide valuable assistance in addressing this issue and supporting students in making informed decisions about their course choices.

Numerous online systems have been developed to facilitate students in discovering and comparing various courses offered by different universities, including UCAS, UK course finder, unistats, and comparetheuni. However, these tools have been acknowledged to primarily rely on previous users' knowledge of courses or keyword-based queries as their underlying mechanisms. Consequently, when students submit a course query, they may receive a large number of results in a random order, leading to information overload and potential irrelevance of the recommendations.

Significant advancements have been made in the field of course recommendation systems, with the objective of assisting students in identifying appropriate courses. Notably, noteworthy achievements have been accomplished in the development of a course recommendation system that employs a collaborative filtering approach [107].

Furthermore, Course Agent is a notable contribution in the field of course recommendation systems. It utilizes a community-based approach and employs social navigation techniques to generate course recommendations based on a student's estimation of their career goals [21]. This method implicitly collects explicit student feedback as part of their natural interactions with the system. However, beyond its recommendation capabilities, Course Agent also serves as a valuable course administration system, storing information about the courses chosen by students and facilitating communication with their advisors, thus providing additional benefits to students..

Sandvig and Burke have introduced a novel course recommendation system called AACORN, which employs a case-based reasoning approach [30]. The system leverages the past experiences and course histories of previous students as a starting point for advising course selection, and utilizes a widely used metric in bioinformatics called the edit distance. However, AACORN requires input of a partial history of courses taken by a student before it can generate practical recommendations.

The RARE recommender system, as proposed by [101], combines association rules with user preference data to provide relevant course recommendations. The system utilizes accurate data from the Department of Computer Science at the Universite de Montreal to analyze the past behavior of students in terms of their course choices. By formalizing implicit association rules, RARE is able to predict recommendations for new students. Furthermore, the system also proposes a solution to address the cold start problem, which is a common challenge in recommender systems.

PEL-IRT, or Personalized E-Learning system, as described in [108], applies item response theory to recommend suitable course material to students by taking into consideration both the difficulty of the course material and the student's ability. When using PEL-IRT, students have the option to choose course categories and units, and they can also search for course material using relevant keywords. Once the course material is suggested to students and they have reviewed the information, the system prompts them to answer two questionnaires. The explicit feedback provided by students through these questionnaires is then utilized by PEL-IRT to re-evaluate their abilities and further customize the recommendation of course material, taking into account the difficulty level that best matches their abilities.

Recent academic research has highlighted the significance of personalization as a key factor in enhancing the accuracy of recommendations and information retrieval [23]; [109]. For instance, Punj and Moore [110] demonstrated that recommendation agents that possess the capability to filter and integrate information, as well as provide feedback, have a more significant influence on user decision-making compared to agents that are only aware of alternative options. Moreover, it has been observed that the outcome of relevant course recommendations is enhanced by integrating valuable information from multiple sources, such as job sites, social networks, and other relevant educational data sources [23]. The Course Recommender System incorporates various collaborative filtering algorithms, including user-based [112] and item-based [111], to provide comprehensive and customized recommendations to user

The Course Recommender System [113] is based on a variation of the commonly used item-based collaborative filtering algorithm. The system serves as a module recommender, aiming to streamline and enhance the online module selection process by providing recommendations for optional modules to students based on their core module selections. By leveraging historical enrolment data, the system has demonstrated promising performance in terms of recall and coverage. Recent research has also explored the application of course recommender systems in niche areas, such as civil engineering professional courses [114] and physical education courses at universities [115]. This indicates the potential of course recommender systems to cater to specific domains and provide tailored recommendations to meet the unique needs of different disciplines and industries.

Through a comprehensive review of the existing literature, it is evident that recommendation technology has been successfully applied in the field of education to facilitate teaching and learning processes. Given the significance and importance of education, the assistance of a recommendation system has the potential to enhance the efficiency and effectiveness of learners in real-world educational scenarios. By providing personalized recommendations, these systems can aid in guiding learners towards relevant and appropriate learning resources, courses, and modules that align with their individual needs and preferences. This can lead to improved learning outcomes, increased learner engagement, and enhanced overall educational experiences. The utilization of recommendation systems in education has the potential to positively impact the quality of education and contribute to the advancement of the field.

The studies mentioned above emphasize the significance of course recommendation systems in supporting students in finding suitable courses. However, it is observed that existing systems predominantly rely on a single data source, such as student profiles, course histories, or university records, to provide course recommendations. In contrast, the proposed research in this thesis aims to develop a novel course recommendation system by integrating course information from multiple diverse data sources. This approach involves incorporating data from university websites, job websites, social networks, and other relevant educational data sources to generate more comprehensive and informed course recommendations for students. By leveraging data from various sources, the proposed system aims to enhance the accuracy, relevance, and reliability of course recommendations, ultimately assisting students in making well-informed decisions about their educational paths. This multi-source integration approach has the potential to bring about novel insights and opportunities for course recommendation systems in the field of education.

The proposed course recommendation system aims to offer students personalized recommendations that align with their individual needs, interests, and career aspirations, thereby supporting their decision-making process. Additionally, the system aims to address the issue of information overload and heterogeneity by utilizing ontology-based data integration techniques to streamline the search results of relevant courses. This necessitates the development of software that can automatically filter out irrelevant choices, gather pertinent information about the available options, and present users with a curated list of more suitable alternatives that best align with their needs and preferences.

Despite the significant impact and usefulness of course recommendation systems, there are certain limitations that need to be addressed. Some of these limitations include:

One limitation of current course recommendation systems is that models based solely on keywords may not effectively address the individual needs of users during the recommendation process. While some models may utilize collaborative filtering and data mining techniques, such as association rules and decision trees, there is often a lack of historical information available, making it challenging to use this approach. For example, new students who are interested in using the system may not be able to generate relevant recommendations due to the lack of sufficient information about their preferences and interests.

A limitation of content-based filtering models is that current approaches typically focus on specific subject recommendations rather than providing recommendations for entire university courses. Additionally, the similarity calculation in these models often relies on weighted averages of features, which may not accurately capture the nuanced preferences of users. Furthermore, these models do not take into account user interactions with the system, such as user ratings of recommended items, which could provide valuable feedback for improving the recommendation accuracy..

Another limitation of current models is that they may not provide comprehensive information about the most relevant course for a student. For instance, students often need to know not only the content of the course, but also the potential career prospects associated with it. Furthermore, students may require information about the quality of facilities offered by the educational institution that offers the course. These aspects are crucial considerations for students in making informed decisions about their course selections, but current models may not adequately address these needs.

The needs and interests of students can be categorized to facilitate appropriate course recommendations. Methods that integrate data from multiple heterogeneous sources can be developed to quickly provide valuable course-related information [23]. The use of an ontology can enable users to obtain precise knowledge about the courses [116]. By establishing relationships between relevant information on the internet, such as course modules, job opportunities, and user interests, ontology provides a vocabulary of classes and properties that describe a domain and promote knowledge sharing [117]. Semantic descriptions of courses and student profiles can be applied to enable qualitative and quantitative analysis of match, along with necessary information about courses and student interests, which is crucial for refining the course selection process.

In this thesis, a novel hybrid filtering system is proposed, which combines both content-based and collaborative filtering methods while utilizing an ontology to address the challenge of information overload in building an effective recommendation system. The proposed approach leverages an ontology for data extraction and integration from multiple data sources, using ontology-based metadata for data integration. The system employs a combination of model-based and memory-based ontology in collaborative filtering to provide high-quality recommendations.

To tackle the issue of the "new user problem", user profiling based on ontology, item ontology, semantic similarity between ontologies, and the proposed KNN (K-nearest neighbors) algorithm are employed in the collaborative filtering aspect. This approach aims to overcome the challenge of providing accurate recommendations for new users who may have limited historical data by utilizing ontologies to profile users and items, calculate semantic similarity between ontologies, and leverage the KNN algorithm for recommendation purposes.

In the content-based filtering aspect, both item-based ontology and semantic similarity are employed to address the challenge of the "new item cold start problem". Additionally, a hierarchy concept similarity approach is utilized to enhance the accuracy of semantic similarity. This approach measures the degree of "IS-A" relationship between the two nodes of the item ontology, resulting in a more precise recommendation list for the target user. By leveraging item-based ontology, semantic similarity, and hierarchy concept similarity, the proposed approach aims to overcome the limitations of the new item cold start problem in content-based filtering and provide more relevant recommendations for users.

2.4.5.2 Course Recommender system Limitations

The implementation of effective education recommender systems presents several challenges that need to be addressed. This section provides a summary of the main challenges identified in the literature and outlines potential strategies to overcome them.

Information overload: The issue at hand pertains to the overwhelming amount of pedagogical content available on the internet, which is widely distributed across various networks [118]. This results in information overload, making it challenging for students to identify and evaluate high-quality learning resources [98]; [114].

Lack of structure in the data: A significant challenge and crucial aspect of a recommender system is the structuring of data [36]. In social learning environments, for instance, information is often categorized into a single category, limiting the options available to users [97]. Moreover, social networks lack a predefined structure, which hinders the interoperability among recommender systems as the lack of structure prevents the reuse of information in other systems [119].

New user and cold start problem: The issue of new resource or user cold start arises when there are no ratings available for newly added resources or when a new user has not yet provided any ratings for items [96]. Possible approaches to overcome this challenge includes:

1) One approach to address the cold start problem is to designate a knowledge provider as the initial starter, who can contribute by providing ratings or other relevant information [120]. Subsequent users can then build upon this initial knowledge, further elaborating and enhancing the recommendation system.

2) Another possible solution to tackle the cold start problem is to employ artificial learners [96]. These machine learning algorithms can be trained on existing data or other

sources of information to generate initial recommendations for new users or items, even in the absence of explicit ratings. Artificial learners can leverage their ability to infer patterns and relationships from data to make informed recommendations, mitigating the limitations of sparse or missing data in the early stages of a recommender system's implementation.

3) Another approach to address the cold start problem is to utilize information related to the completion of activities and similar preferences [87]. For instance, by tracking users' interactions with the system, such as their completion of activities, engagement in discussions, or expressed preferences, the system can gather valuable data to infer their interests and preferences. This information can then be used to generate initial recommendations for new users or items, enhancing the accuracy and relevance of the recommendations despite the lack of historical ratings or user profiles.

Cognitive overload: This challenge pertains to the effort required in selecting valuable resources or assigning accurate ratings [120]. It often arises when dealing with raw data, when users are unable to formulate the right questions, when pedagogical resources lack proper definition by experts [121], when resources are not adequately classified, or when summaries, keywords, or other descriptors are absent [122]. To overcome this challenge, content analysis techniques such as data mining can be employed to identify relevant keywords or structures within the resources. These techniques can help in automatically extracting meaningful information from the resources, enabling better understanding and categorization of the content, and ultimately improving the accuracy and relevance of the recommendations.

Quality of the recommendation and trust: Another challenge arises when users lack trust in the recommender system and its recommendations. The probability of users taking action based on the recommendations may be too low [123]. To address this, it is recommended to define the quality of recommendations [92], differentiate between precise and relevant recommendations, minimize biased recommendations, and provide transparency on the origin of recommendations and the process of adding new items [123]. It is essential to establish trust by making it clear to users where the recommendations come from [120] and how new items are added, thereby ensuring transparency and credibility in the recommendation process

2.5 Literature Gap

The literature covering most of the studies is based on labelled data, with insufficient information on whether a student is considered correct. Although there are studies labelling students as correct or incorrect, how this study differs from others is as follows:

This study proposed three ways to define users as correct and incorrect. Secondly, machine learning models analyse students' correctness and incorrectness. Lastly, a recommender system is to be developed to make recommendations

Chapter 3: Research Methodology

This chapter provides an overview of the overall approach and methods used in the project to achieve the research objectives. It started by introducing the concept of a recommender system, the importance of recommendations, and past work related to it. It then explained the importance of data acquisition as the first step in the technique, which is crucial for building an accurate model. The chapter also described cleaning and preprocessing the data, feature engineering, and using machine learning algorithms such as K-means and KNN to cluster students and develop a recommender system. Finally, it highlighted the importance of considering the time and memory efficiency of the algorithm when selecting the best solution and discussed the importance of validation and evaluation of the model created.

Lastly, the chapter also mentioned how this kind of system could be tailored to online education, providing personalized recommendations of learning paths to students and helping teachers to provide personalized tutoring, which can generate feedback to improve the recommendations. The results and conclusion will be discussed in the upcoming chapter, which will help to understand the impact of the model on the field.

3.1 Methodology Overview

The project is motivated by the COVID-19 pandemic, which forced the world to adapt to online education to continue the learning process. In addition, the project aims to address the challenges distance learning students face regarding keeping track of resources and finding the most appropriate learning path for themselves.

The proposed recommender system for the online education system helps students by providing personalized recommendations based on their interactions and performance on the platform, which can assist students in making intelligent decisions and optimizing their learning journey. The general flow and steps taken during this research are shown in Figure 12, which illustrates the overall process, starting from data acquisition and preprocessing to the development and evaluation of the recommender system.

It is worth mentioning that the proposed system is an addition to the existing educational system, not a replacement, and it should work alongside the teachers, students and other educational resources. The system should be validated and evaluated by experts in the field to ensure that it satisfies the institution's goals and is aligned with the pedagogy and curriculum.

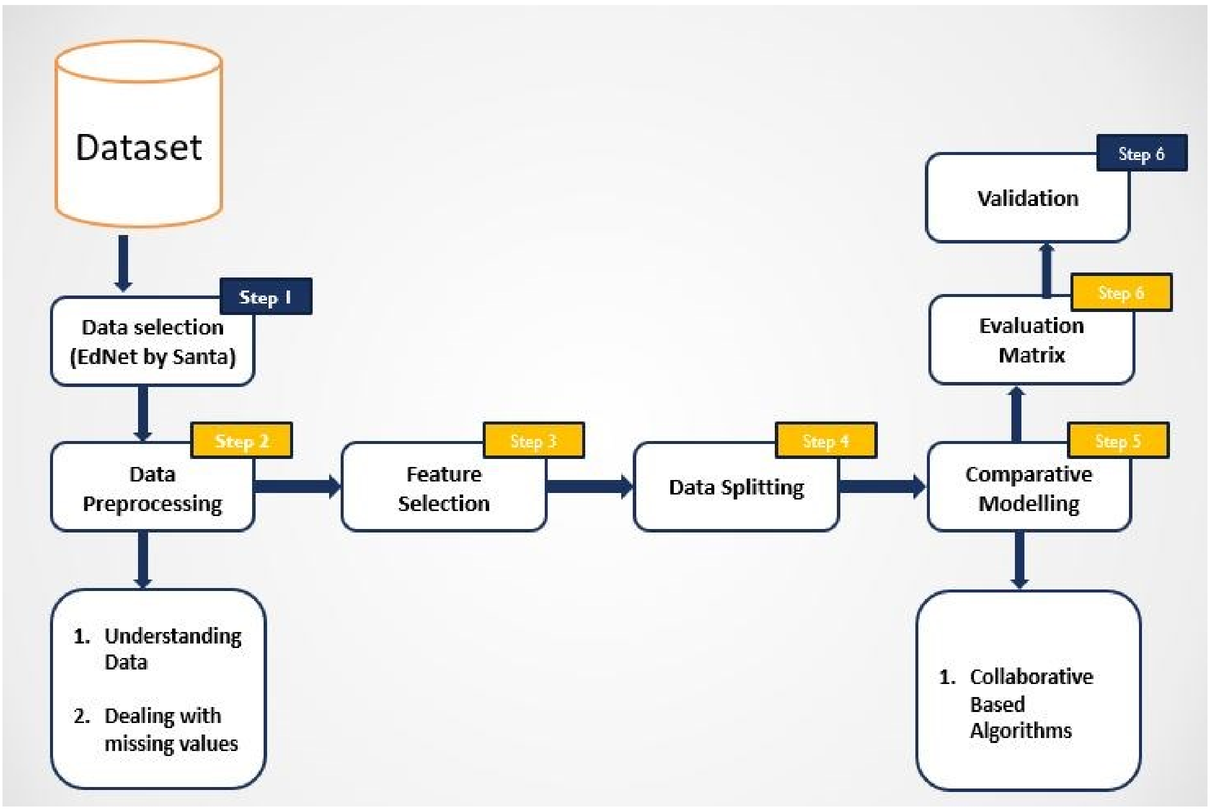


Figure 12 : Illustration of Methodology

3.2 Data Acquisition

The dataset used in the project was acquired from EdNet, a publicly available dataset of student-system interactions collected over two years by Santa. Santa is an AI-based tutoring service that runs on multiple podiums such as iOS, Android, and the web, and it has a user base of over 780,000 students based in Korea.

The EdNet dataset contains almost three years' worth of student interactions, totalling 19.4 GB of data, which consists of almost 1.8 million individual user files upon downloading and decompressing the data. The publicly available dataset can be easily accessed from the project's official GitHub repository.

As it stated that it is based in Korea and the dataset is from Santa AI-based tutoring service, it may be possible that the sample data and feature may not be fully applicable to other education systems or countries;

However, it can be used as a reference or inspiration to other researchers and practitioners in online education, student modelling and personalized learning.

3.2.1 Data Characteristics

The EdNet dataset contains various features that capture different aspects of student actions and interactions with the system. Some of the major categories of the dataset include:

- **Learning activities:** This category contains information about what topics students have covered, which questions they answered, and their responses.
- **Timing data:** This category captures students' time on specific learning modules and questions.
- **Question data:** This category contains information about the specific questions students encountered.
- **Student profile:** This category includes information about the students, such as their age, gender, and primary education level.
- **Bundle data:** This category includes information about the specific learning modules that students interacted with, including the number of bundles read and their percentage
- **Evaluative data:** This category contains information about students' performance, like the percentage of questions answered correctly and the time spent on average for each question.

The dataset is rich and detailed, making it valuable for researchers and practitioners in student modelling and personalized learning. With this dataset, it is possible to investigate various research questions related to online learning and education, student behaviour, student performance, and clustering and recommendation. Here are some major categories of the EdNet dataset shown and described below

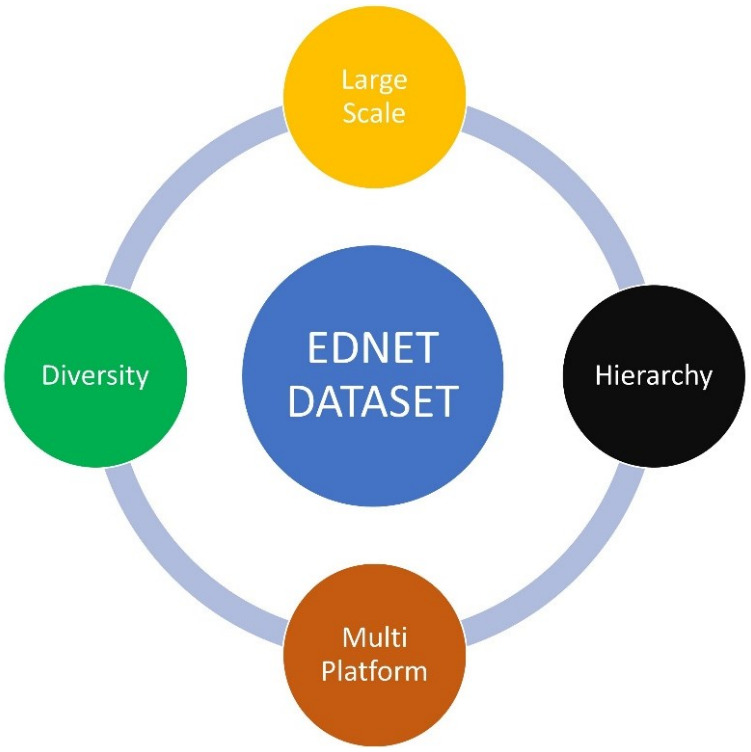


Figure 13: Illustration of characteristics of the dataset

3.2.1.1 Large Scale

The EdNet dataset contains a large amount of data collected from many students using the Santa AI tutoring tool. The dataset comprises nearly 1.4 million interactions collected from almost 1.8 million students since 2017, with each student generating around 442 interactions using Santa across various platforms. In addition, the dataset includes 13,169 problems or questions from 1,021 lectures consumed approximately 95 million times.

The dataset is one of the most extensive education datasets publicly available regarding the number of students, interactions, and types of interactions, which makes it a valuable resource for researchers and practitioners in online education and student modelling. In addition, it provides a rich and detailed dataset to investigate various research questions related to student behaviour, student performance, and personalized learning.

3.2.1.2 Diversity

EdNet dataset is known for its diversity, providing a wide range of information about students' interactions with the Santa AI tutoring system; the dataset includes data on various learning activities, such as watching lectures, reading explanations, attempting questions, repeating questions and even the time frame of logging in and out. This diversity of data allows us to investigate different aspects of student behaviour and performance, providing a complete picture of students' interactions with the system.

It is indeed one of the richest datasets in the field of online education systems; it has a lot of diverse data which covers student's learning journey from start to finish, providing data on how the students are engaging with the system and the contents, how well are they learning, where are they struggling and what is their learning pace. With this dataset, we can develop better and more effective personalized recommendations for students to improve their learning experience and understand their behaviour and performance.

3.2.1.3 Multi-Platform

Santa is a multi-platform AI tutoring system that students can access using various devices, such as personal computers, smartphones, and even AI speakers, and it allows students to access the system and continue their learning journey regardless of where they are or what device they have access. The EdNet dataset reflects this diversity of access points, as it contains data points from mobile and desktop platforms.

The dataset becomes even more valuable as it captures students' interactions with the system across different platforms, providing a complete picture of students' learning journey, which can provide insights into how students interact with the system in different contexts, how platform-specific features are used, and the impact of device choice on students' learning experiences.

3.2.1.4 Hierarchy

The EdNet dataset is organized in a hierarchical structure, with five different levels, each named "KT1", "KT2", "KT3", "KT4", and "Content", respectively. Each level represents a different granularity of data, with increasing levels of detail as moving down the hierarchy.

The "KT1" level represents the most general information, such as the number of students and problems. The "KT2" level provides more detailed information, such as the amount of correct and incorrect answers for each problem, the average time spent on each problem, and the number of attempts. The "KT3" level includes even more detailed information, such as the student's performance on each problem over time and the specific answers given by the student. The "KT4" level provides the most detailed information, such as the exact time and date when each action was taken and the time taken for each action. Finally, the "Content" level includes the actual content of the problem and the corresponding explanation.

This hierarchical data structure makes it easy to access the data with various levels of granularity based on the research question. It allows us to understand the data better and quickly analyze the student's behaviour and performance.

The EdNet dataset is divided into five levels with different data granularity. The "KT1", "KT2", "KT3", "KT4", and "Content" levels each provide different information about students' interactions with the Santa AI tutoring system.

However, a standard feature across all these levels is; The dataset is divided by students, identified by user_id, in the form of CSV files, and it only contains interactions with the system across all the modules users have engaged, which allows for easy access and analysis of data for specific students and their interactions with the system. In addition, the user_id is a unique identifier for each student, and it helps to link all the data of one student together and makes it easy to track the student's journey.

This organization of the data also allows us to efficiently study the student-specific interactions, behaviours and progress with the system, which can give the researcher a deeper understanding of how each student engages with the system and their unique strengths and weaknesses. Also, it allows researchers to understand how the students are progressing and compare their performance with other students.

3.3 Python Libraries

Python is a popular programming language in data science, and many open-source libraries are available for use in data science projects. These libraries are built, maintained, and supported by a large community of developers, which makes it easier for researchers and data scientists to find solutions to problems they encounter.

This study uses different libraries for data manipulation, cleaning, visualizing and model building. Table 3 might provide the details of the libraries and their specific application, such as Pandas for data manipulation, Numpy for numerical computation, Matplotlib and Seaborn for data visualization, Sklearn for machine learning models.

These libraries are compelling and easy to use and provide the researcher with a wide range of tools to analyze and visualize the data. In addition, some specific libraries like scikit-learn and K-Means can be used to train and evaluate the Machine Learning models. Overall, Python provides an excellent ecosystem for data analysis and machine learning projects, and its libraries are designed to make data analysis easy and efficient.

Library	Usage
Pandas	Data analysis begins with importing the data files, followed by data cleaning, integrating several datasets into one, statistical analysis, and much more.
Numpy	It makes it possible to work with multi-dimensional arrays effectively. This library is the foundation upon which Pandas, Matplotlib, and Scikit-Learn are based.
Matplotlib	It is a tool that is used for data visualization and plotting.
Seaborn	Visualization package that is built on top of Matplotlib to generate aesthetically appealing graphs
Plotly	Plotly facilitates various languages and offers a high level of customization and interactivity.
Datetime	This module supplies classes that are used for accessing and manipulating data and time
feather	Feather format is an alternative file format to store the dataset. It offers more

	extraordinary read & write speeds than the traditional CSV or excel file format.
Sklearn	It is used to anticipate customer churn because it has established many machine-learning algorithms, pre-processing techniques, performance indicators, and many other things.

Table 3: Python Libraries used in this project

3.4 Fundamental Analysis and General Trends

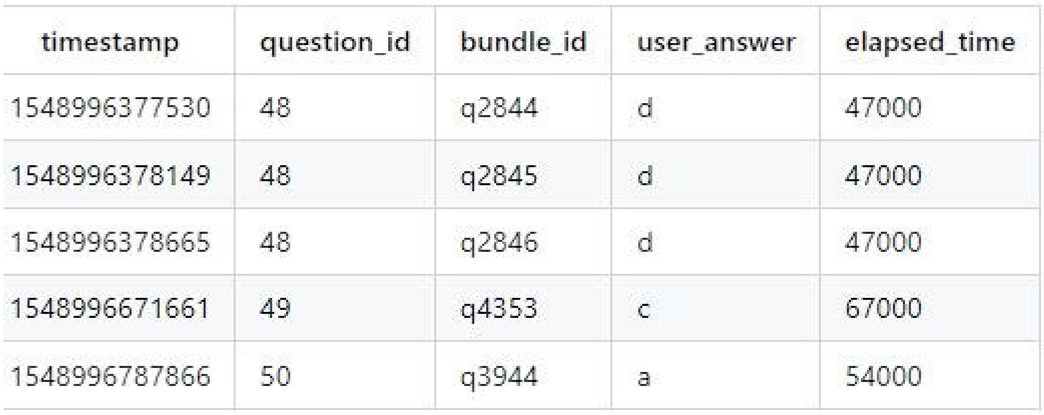
The EdNet dataset comprises five different sub-datasets, each with a different level of granularity and containing different types of information about the students' interactions with the Santa AI tutoring system.

The KT1 dataset contains basic information such as the user_id, the timestamp, and the student's actions. The KT2 dataset provides more detailed information, such as the specific problems (questions) the student attempted and their performance on those problems. The KT3 dataset adds information about the student's learning history, such as how many times they attempted a problem and how many times they got it correct. The KT4 dataset provides even more detailed information, including the specific modules the student interacted with and how much time they spent on each one. Finally, the Content dataset provides information about the specific content of each module, such as the topics covered and the number of questions included.

Each of these sub-datasets provides a different perspective on the student's interactions with the system, and together they provide a comprehensive picture of each student's learning journey. Using these datasets, researchers can analyze student-specific interactions, behaviours, and progress and gain a deeper understanding of how each student engages with the system, their unique strengths and weaknesses, and how they are progressing.

3.4.1 KT1 SUBDATASET

First, the preview of sub-dataset KT1:



timestamp	question_id	bundle_id	user_answer	elapsed_time
1548996377530	48	q2844	d	47000
1548996378149	48	q2845	d	47000
1548996378665	48	q2846	d	47000
1548996671661	49	q4353	c	67000
1548996787866	50	q3944	a	54000

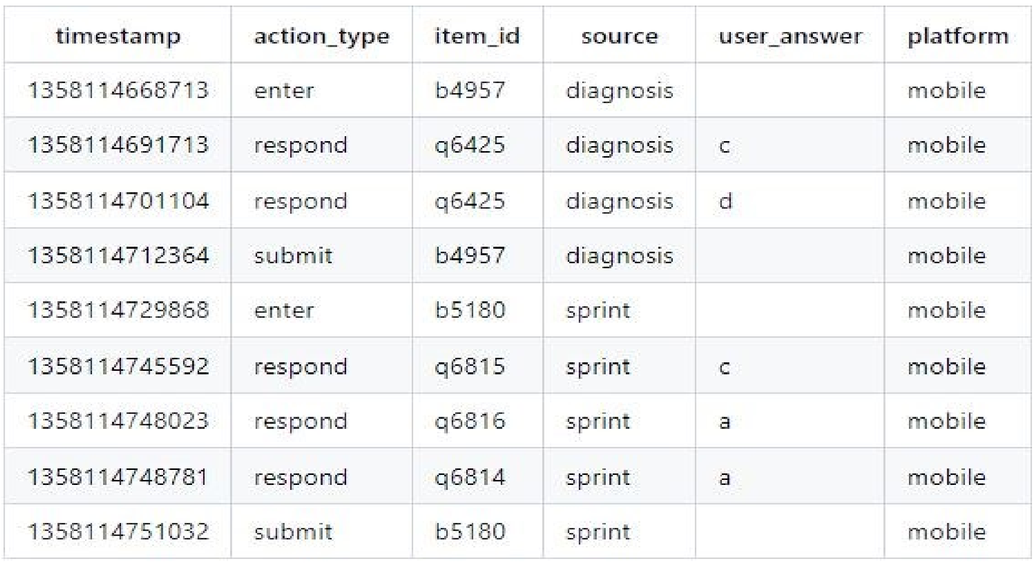
Figure 14: Preview of first sub-dataset(KT1)

Description:

- **Timestamp**: is the instant the question was presented to the user and is represented as a Unix timestamp in milliseconds
- **Question_id**: this represents the question that was presented to a student
- **Bundle_id**: this represents from which specific bundle that question belongs. In standard terms, these can also be known as chapters.
- **User_answer**: The student's answer to a specific question was presented and recorded as a character between a and d inclusively.

3.4.2 KT2 SUBDATASET

Secondly, the preview of the sub-dataset KT2:



timestamp	action_type	item_id	source	user_answer	platform
1358114668713	enter	b4957	diagnosis		mobile
1358114691713	respond	q6425	diagnosis	c	mobile
1358114701104	respond	q6425	diagnosis	d	mobile
1358114712364	submit	b4957	diagnosis		mobile
1358114729868	enter	b5180	sprint		mobile
1358114745592	respond	q6815	sprint	c	mobile
1358114748023	respond	q6816	sprint	a	mobile
1358114748781	respond	q6814	sprint	a	mobile
1358114751032	submit	b5180	sprint		mobile

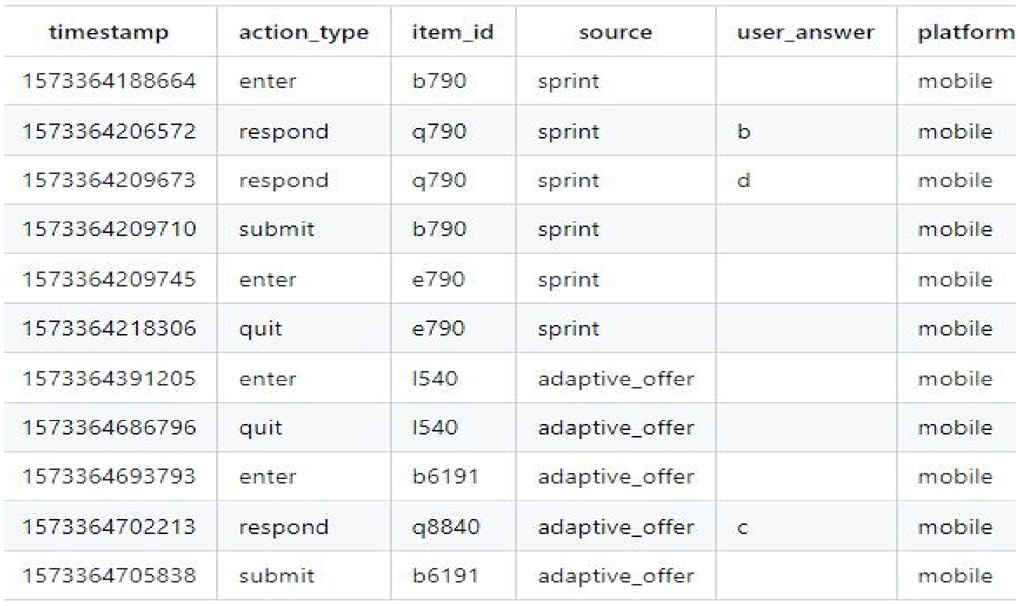
Figure 15: Preview of first sub-dataset(KT2)

Description:

- **action_type**: this column contains three values:
- **enter**: this is documented when a student receives and views a question bundle through SantaUI
- **respond:** this is documented when the student answers from a to d respectively; a student can change their answer to the same question multiple times; the last response a student gives would be considered the last response.
- **submit**: this is documented when a student submits the final answer to the given question from the bundle
- **item_id**: this column contains two values (bundle integer or question integer); for KT2-only, the IDs of questions and bundles are documented; whenever the action type is "enter" or "submit", it is documented as bundle id, but whenever the action type is "respond" it records as question id.
- **Source:** this column represents where the student is watching a lecture on solving a question in SANTAUI.
- Sprint: This entry represents students choosing a part that they want to study; after selecting the specific path, they can only answer questions to specific questions of that bundle unless they change their path
- Diagnosis: this represents when a user enters the SantaUI application for the first time, they need to solve several questions for diagnosis (as per the author of the dataset)
- **User_answer:** this column only records values from a to d, respectively, whenever the action type is "respond".
- **Platform:** This column represents which specific device students use to access the SANTAUI, which contains mobile & web.

3.4.3 KT3 SUBDATASET

Here is the preview of the sub-dataset KT3:



timestamp	action_type	item_id	source	user_answer	platform
1573364188664	enter	b790	sprint		mobile
1573364206572	respond	q790	sprint	b	mobile
1573364209673	respond	q790	sprint	d	mobile
1573364209710	submit	b790	sprint		mobile
1573364209745	enter	e790	sprint		mobile
1573364218306	quit	e790	sprint		mobile
1573364391205	enter	l540	adaptive_offer		mobile
1573364686796	quit	l540	adaptive_offer		mobile
1573364693793	enter	b6191	adaptive_offer		mobile
1573364702213	respond	q8840	adaptive_offer	c	mobile
1573364705838	submit	b6191	adaptive_offer		mobile

Figure 16: Preview of first sub-dataset(KT3)

Description:

As mentioned above, that student has various interactions and may participate in multiple learning activities aside from solving questions, such as watching lectures provided by SANTAUI or reading through explanations of a specific question by the expert teacher's recorded comments also provided by SANTAUI.

Reading explanations:

When the student attempts the given question, corresponding explanations are provided, and the source column will record as "sprint" or "review". Of course, they can also re-read the explanations of the questions; in that case, the source column will have "my_note".

As the student enters or exits the explanation view in SANTAUI, the item_id will record as "enter" or "quit".

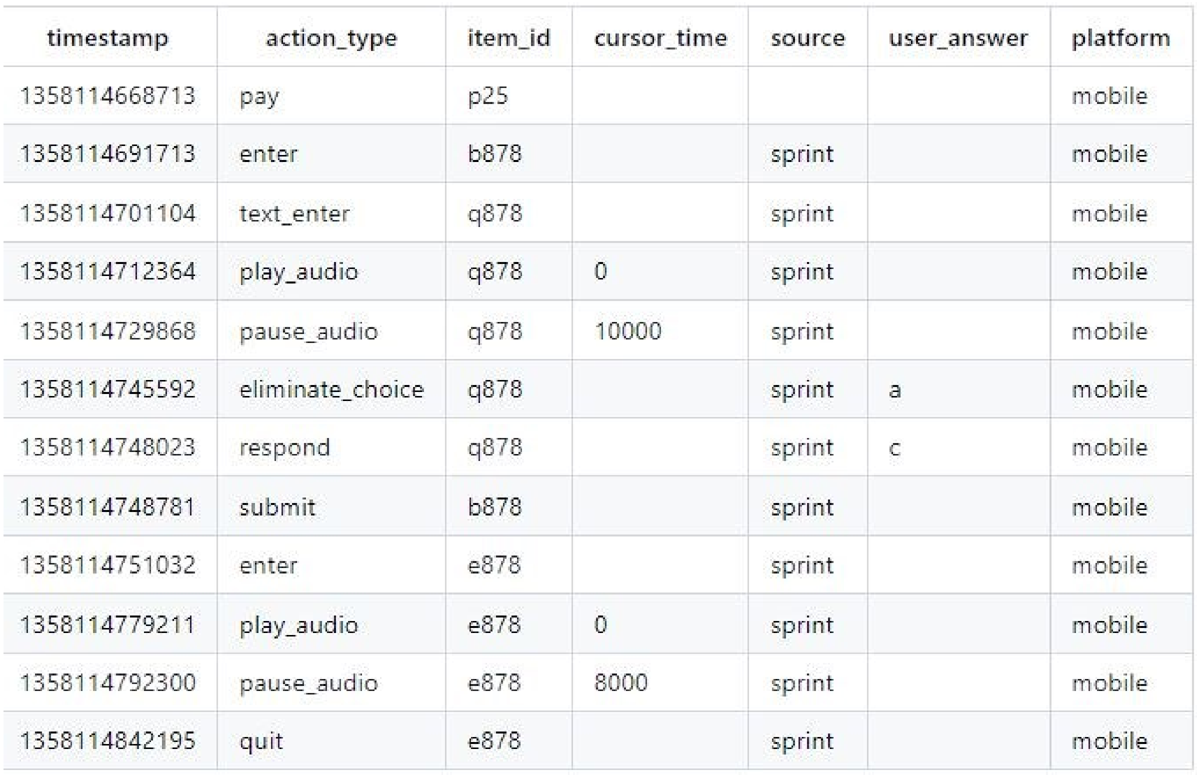
Watching lectures:

Whenever a student watches a lecture, source column entries have two outcomes: "archive" and "adaptive_offer". "Archive" is watching a recorded lecture presented in the SANTAUI, adaptive_offer is SANTAUI recommending the following lectures based on the student's path.

Action_type would register as "enter" and "quit" with lectureID when the student starts watching or stops the lecture.

3.4.4 KT4 SUBDATASET

Here is the preview of the sub-dataset KT4:



timestamp	action_type	item_id	cursor_time	source	user_answer	platform
1358114668713	pay	p25				mobile
1358114691713	enter	b878		sprint		mobile
1358114701104	text_enter	q878		sprint		mobile
1358114712364	play_audio	q878	0	sprint		mobile
1358114729868	pause_audio	q878	10000	sprint		mobile
1358114745592	eliminate_choice	q878		sprint	a	mobile
1358114748023	respond	q878		sprint	c	mobile
1358114748781	submit	b878		sprint		mobile
1358114751032	enter	e878		sprint		mobile
1358114779211	play_audio	e878	0	sprint		mobile
1358114792300	pause_audio	e878	8000	sprint		mobile
1358114842195	quit	e878		sprint		mobile

Figure 17: Preview of first sub-dataset(KT4)

Description:

This sub-dataset contains a complete list of actions that SANTAUI records. These actions are added to KT3; following are the actions are “erase_choice”, “undo_erase_choice”, “play_audio”, “pause_audio”, “play_video”, “pause_video”, “pay”, “refund”, “enroll_coupon.”

Erase_choice, undo_erase_choice: A student can erase his choice while attempting a question and undo_erase choice.

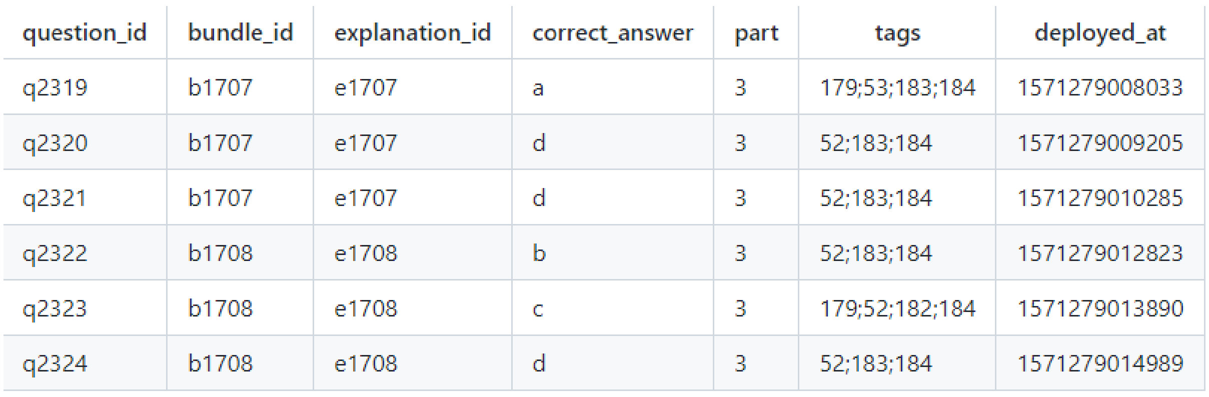
Play_audio, pause_audio, play_video, pause_video: As the lectures and the explanations provided by expert teachers are recordings saved in SANTAUI, a student can select any of these available actions, and while choosing these options, a cursor_time is recorded in the cursor_time column when the students play or pause the media.

Pay, refund: SANTAUI offers a free trial to every student who wishes to get an idea of the UI. In a free trial, the student is provided ten questions; if they hope to continue, they need to pay to access all the contents. Refunds may also apply to several specific situations.

Enroll_coupon: this entry is recorded when a student pays for full access to SANTAUI contents and applies valid coupons during the payment period.

3.4.4.1 *CONTENT SUBDATASET*

Here is the preview of the sub-dataset Content:



question_id	bundle_id	explanation_id	correct_answer	part	tags	deployed_at
q2319	b1707	e1707	a	3	179;53;183;184	1571279008033
q2320	b1707	e1707	d	3	52;183;184	1571279009205
q2321	b1707	e1707	d	3	52;183;184	1571279010285
q2322	b1708	e1708	b	3	52;183;184	1571279012823
q2323	b1708	e1708	c	3	179;52;182;184	1571279013890
q2324	b1708	e1708	d	3	52;183;184	1571279014989

Figure 18: Preview of first sub-dataset (Content)

Description:

Last but not least, this sub-dataset contains a CSV file named **questions; as** per the above table, it has **questions_id, bundle_id, explanation_id, part, tags, deployed_at**

question_id: This column enlists all the question numbers in the form of q{integer}

bundle_id: This column enlists all the bundle numbers in the form of b{integer}. A single bundle can contain many questions, from a minimum of 1 to a maximum being 5.

Explanation_id: This column enlists an expert teacher's corresponding explanations for each bundle. Bundle_id and explanation_id are the same for every question which is provided

Correct_answer: This column enlists all the correct answers to each question; it is recorded as a character between a,b,c and d.

Part: This column enlists the different parts of a question; it contains numeric values from 1 to 7 inclusively.

Tags: This column contains all the annotated tags by expert teachers for their understanding.

Deployed_at: specifies the moment at which each question in Santa begins to be provided as a Unix timestamp in milliseconds

3.5 Data Pre-processing

Data preprocessing is a critical step in machine learning, preparing the data for analysis and modelling. Its goal is to clean, reformat, and transform the dataset so that it is in a format that the machine learning algorithm can easily use.

Data cleaning is the procedure of removing irrelevant, duplicate, or inconsistent data from the dataset. This step is critical to ensure that the dataset is accurate and free of errors.

Data reduction is the procedure of reducing the dimensionality of the dataset by selecting a subset of features that are most relevant to the analysis. This step helps reduce the algorithm's computational cost and helps visualise the data effectively. Data transformation is converting data from one format to another, such as converting text data to numerical data and normalizing or standardizing data.

Data scaling is the process of scaling the data so that all features are in the same range. This step is essential because specific machine learning algorithms are sensitive to the scale of the data, and having features in different ranges can affect the algorithm's performance.

All these preprocessing steps are essential to ensure that the dataset is in a format that can be easily understood and used by the machine learning algorithm to increase the accuracy of the predictions/recommendations, as displayed in Figure 14.

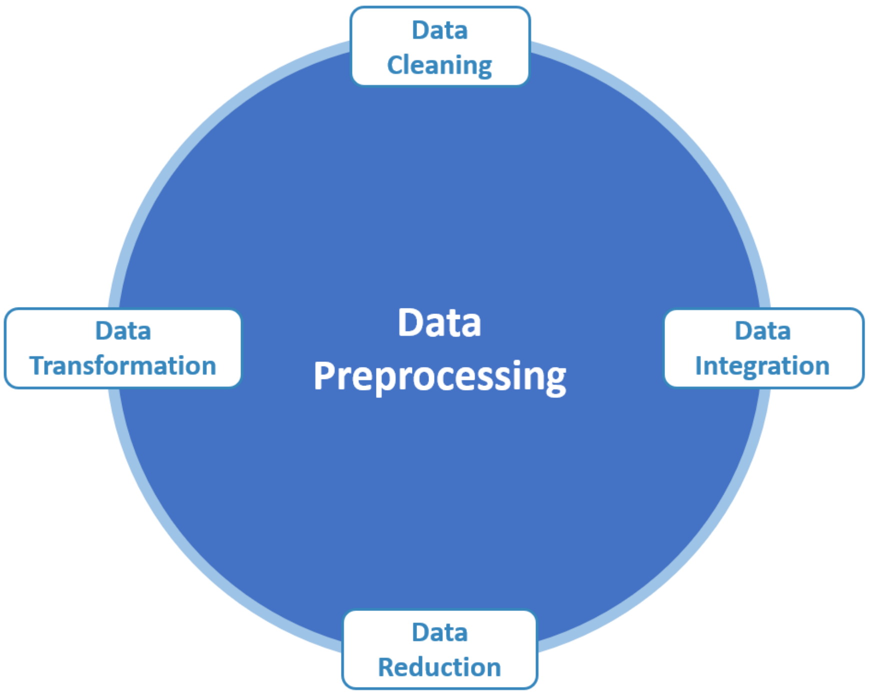


Figure 19: Steps that are taken during the pre-processing phase

3.5.1 Data Cleaning

Data cleaning is a crucial step in the data preprocessing phase, and it is crucial to the overall success of a machine learning project. Irregularities in the data, such as duplicate data points, missing values, and incorrect formatting, can significantly affect the performance of a machine-learning model. These issues can introduce bias into the model and cause the model to make inaccurate predictions. For example, duplicate data points can cause the model to over-represent specific data, resulting in a bias in the predictions. In addition, missing values can cause the model not to be able to make predictions for specific data points, and incorrect formatting can cause the model to misinterpret the data.

By identifying and addressing these issues during the data cleaning phase, we can ensure that the data is accurate, unbiased and in a suitable format, which can help build a more robust and accurate machine learning model.

3.5.1.1 Handle Missing Values

missing values can significantly impact a machine-learning model's performance and should not be ignored. Discarding observations with missing values is one option, but this can result in a loss of valuable data and can cause a reduction in the size of the dataset, making it harder to train an accurate model.

Another option is to fill in the missing values with a substitute value, such as the mean or median of the other values in the dataset, commonly known as imputation. Finally, interpolation and data regression techniques can also be applied to the dataset to fill in the missing values.

However, one should be careful while dealing with the missing values as it can cause data integrity issues if done incorrectly; therefore, applying these techniques after a thorough understanding of the problem and how the missing values affect the integrity of the data.

An alternative way to deal with missing values is to use robust models such as decision tree-based models, Random Forest and Gradient Boosting Machine Learning. These models can handle missing values and give reasonably good predictions.

3.5.1.2 Duplicate Data Removal

Additionally, duplicate data is present; this data should be removed before the data is provided to the machine learning algorithm. The duplication of interactions performed by students will be removed to remove the biases and to achieve an actual percentage evaluation of each student

3.5.1.3 Outliers Detection

Outliers can significantly affect the performance of machine learning algorithms and can lead to incorrect predictions. Therefore, it is essential to identify and handle outliers in the data. There are various methods to identify and handle outliers, such as the Z-score method, the Interquartile range method and more advanced methods, such as clustering-based methods. Once identified, there are several ways to handle outliers, such as removing them, replacing them with the mean or median value of the feature or transforming the feature to make it more robust to outliers. The choice of method depends on the data's nature and the analysis's goals.

3.5.2 Data Reduction

Data reduction is a process that involves identifying and removing unnecessary data to make the data more manageable for analysis and modelling. Data Reduction can be achieved through various techniques, such as feature selection and dimensionality reduction. Feature selection involves choosing a subset of features from the original dataset most relevant to the task. Dimensionality reduction practices such as PCA (Principal Component Analysis) can reduce the number of features by identifying and removing redundancy in the data. By reducing the data, the model can run more efficiently, and the results can be more interpretable.

3.5.3 Data Integration

Combining data from multiple sources or subsets can provide a more comprehensive view of the problem being analyzed, but it can also increase the complexity of the data preprocessing step. The data may also have different formats and structures, which must be addressed before it can be combined. Therefore, the analyst needs to understand each dataset's characteristics and carefully plan how they will be combined to ensure that the resulting data is accurate and valuable for analysis. Additionally, combining data from multiple sources can also introduce privacy and security concerns, which should be considered.

3.5.4 Feature Engineering

Feature engineering is a crucial step in the data science pipeline because it can significantly impact the performance of machine learning models. It involves extracting relevant information from raw data and transforming it into features that the model can use to make predictions. This process includes feature selection, feature scaling, and feature transformation. As a result, the model can be more accurate and efficient by carefully selecting and transforming the features. It also helps to reduce the dimensionality of the data, which can improve the performance of some algorithms and make it easier to visualize the data. Overall, feature engineering helps to improve the interpretability and predictive power of machine learning models.

3.5.4.1 Merging the dataset

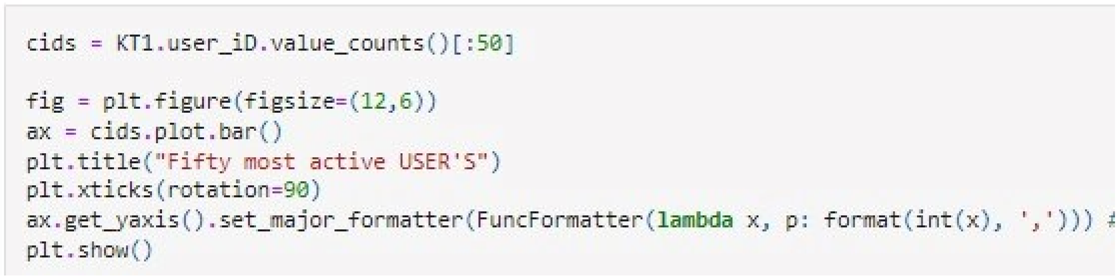
Since there were 1.6 million users' CSV single files, thus, to create a data frame, the whole dataset merged CSVs into a single data frame. This process involves reading all the individual CSV files, combining them into a single data frame, and then merging them into a large dataset. This step is necessary to make analysing and working with the data more accessible. It also allows for more efficient processing when applying machine learning algorithms to the data. The merged dataset can then be preprocessed and cleaned before being passed to the machine-learning model for further analysis.

3.5.4.2 Writing the data frame

Feather format is a lightweight binary format for storing data frames designed to be fast and efficient for reading and writing. Because it is binary, it can be read and written much faster than traditional text-based formats like CSV or Excel, making it well-suited for large datasets like this project. Additionally, because it is designed to be lightweight, it reduces the data frame's size, making it easier to store and share. The feather format is well suited for storing data frames for machine learning and data analysis tasks because it is fast, efficient, and easy to use. Exporting the data frame was causing performance issues as the data frame was almost 10.0 Gb being written as a single CSV or excel file, which was a time-consuming procedure as 10.0 Gb and was not an ideal approach. Thus, exporting the data frame into a feather format significantly reduced the size of the data frame; not only the feather format reduced the file size significantly size it offered more extraordinary read-write speeds as compared to traditional excel or CSV format

3.5.4.3 Top 50 Active Users

The code snippet below will select the dataset's top 50 users with the most interactions. By counting the number of interactions for each user and then sorting the data by the number of interactions in descending order, the top 50 users with the most interactions can be found. This information can help determine which users are most engaged with the platform and may have more valuable information for the analysis.



```
cids = KT1.user_iD.value_counts()[:50]

fig = plt.figure(figsize=(12,6))
ax = cids.plot.bar()
plt.title("Fifty most active USER'S")
plt.xticks(rotation=90)
ax.get_yaxis().set_major_formatter(FuncFormatter(lambda x, p: format(int(x), ','))) #
plt.show()
```

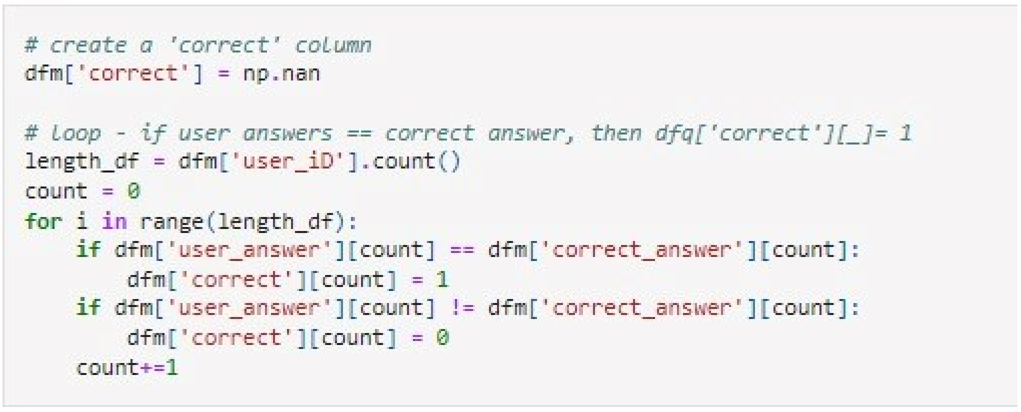
3.5.4.4 Target Column "Correct."

The dataset had several interactions different users had; Now, the approach was to determine the correctness and incorrectness of users' interactions. Thus, a new column was needed to indicate that when a user attempts a question, it is correct or incorrect.

This column will have a binary indication that is 1 and 0. If the attempted question by any user is correct, then the value would be “1”; otherwise, it would be “0” in case of incorrect.

As mentioned above, there is a “Questions CSV” in the “content sub-dataset”, which is an answer key sheet for all the questions.

Now there is a need to create a correct column in the primary data frame and check the attempted questions with the answer key. Below is the code snippet for creating a “correct” column, checking each interaction, and filling up the values in the “correct” column. This information can then be used to analyze the students' performance and identify trends in their interactions. In addition, this new column will allow us to calculate the accuracy of each student.



```
# create a 'correct' column
dfm['correct'] = np.nan

# Loop - if user answers == correct answer, then dfq['correct'][_]= 1
length_df = dfm['user_iD'].count()
count = 0
for i in range(length_df):
    if dfm['user_answer'][count] == dfm['correct_answer'][count]:
        dfm['correct'][count] = 1
    if dfm['user_answer'][count] != dfm['correct_answer'][count]:
        dfm['correct'][count] = 0
    count+=1
```

3.5.4.5 Questions Correctness/Incorrectness

After identifying the total number of questions attempted by each user, the next step was to find the number of correct and incorrect answers for each user. To achieve this, the "Correct" column, created in the previous step, filtered the data and counted the number of correct and incorrect answers for each user.

This code groups the data by the "user_id" column and counts the number of times each value appears in the "Correct" column. The resulting data is then unstacked and transformed into a new data frame with columns for "Correct" and "Incorrect" answers and rows for each user. This way, we can summarise the number of interactions of users to see each user how many questions they attempted and the count for correctness and incorrectness.

```
df_total_questions_attempted = dfm.groupby(['user_iD'], as_index=False , sort=False)['question_id'].count()

df_total_questions_attempted = df_total_questions_attempted.rename(columns={'question_id': 'Total Questions Attempted'})

df_total_questions_attempted.head()
```

After figuring out the total number of questions attempted, it is time to identify how many questions are correct and incorrect. Below is the code snippet to check the count for “Corrected” and “InCorrected.”

```
df_correct = dfm[dfm['correct']==1].groupby(['user_iD'], as_index=False, sort=False)['question_id'].count()

df_correct = df_correct.rename(columns={'question_id': 'Corrected'})

df_correct.head()
```

```
df_incorrect = dfm[dfm['correct']==0].groupby(['user_iD'], as_index=False, sort=False)['question_id'].count()

df_incorrect = df_incorrect.rename(columns={'question_id': 'Incorrected'})

df_incorrect.head()
```

3.5.4.6 Users Percentages

The "Percentage" column is calculated by dividing the number of correctly answered questions by the total number of questions attempted by a user and multiplying the result by 100 to get the percentage. The percentage represents the user's performance on the questions they attempted, which allows for a more accurate comparison of the performance of different users, as the total number of questions attempted can vary significantly between users.

Below is the code snippet for calculating the percentages of each user

```
dfs = [df_total_questions_attempted, df_correct, df_incorrect]
df_final = reduce(lambda left,right: pd.merge(left,right,on='user_iD'), dfs)
```

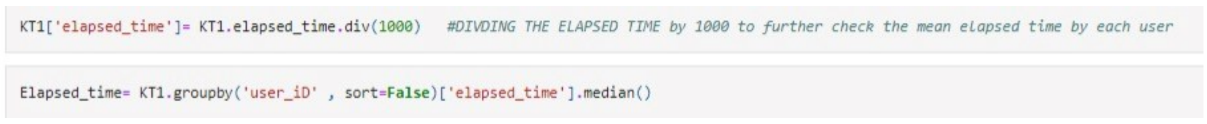
```
df_final['Percentage'] = (df_final['Corrected'] / df_final['Total Questions Attempted']) *100
df_final.head()
```

3.5.4.7 Users elapsed time

With all the interactions and information gathered in terms of attempts, correctness, incorrectness and percentages, Furthermore to identify on average each user how much time in seconds it takes to answer all the questions they attempted. As per the source dataset, the elapsed time of each attempt was given in milliseconds.

First, convert from milliseconds to seconds, then take an average of all interactions to identify each user's average time.

Below is the code snippet for finding out elapsed time



```
KT1['elapsed_time']= KT1.elapsed_time.div(1000)   #DIVDING THE ELAPSED TIME by 1000 to further check the mean elapsed time by each user
```

```
Elapsed_time= KT1.groupby('user_iD' , sort=False)['elapsed_time'].median()
```

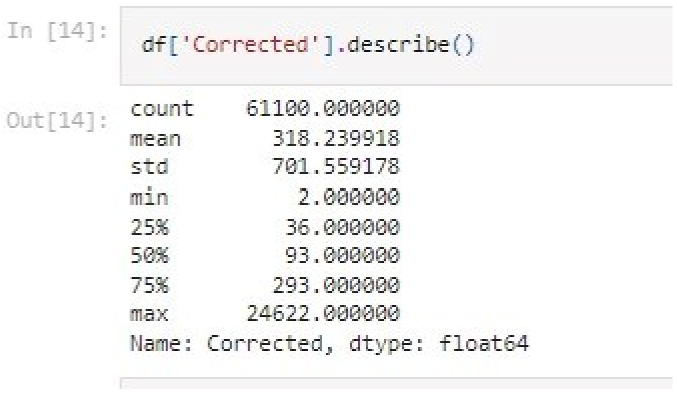
3.5.4.8 Users Statistics:

Additionally, all interactions can be combined into a single data frame, such as "Total Questions Attempted", "Corrected", "Incorrected", "Percentage", and "Elapsed time". These statistics provide an overview of all users in a single table format data frame, detailing the number of total questions attempted, the number of correct answers, the number of incorrect answers, the percentage of correct answers out of the total attempted, and the average elapsed time taken per question for each user.

3.5.4.9 Describing Statistics:

Now the aim was to determine the statistics of users based on a corrected column to see the minimum number, maximum number and, on average, how many questions are corrected.

Below is the snapshot of the “Corrected” column describing the feature used in the python language.



```
In [14]:  df['Corrected'].describe()

Out[14]:  count    61100.000000
          mean       318.239918
          std        701.559178
          min          2.000000
          25%         36.000000
          50%         93.000000
          75%        293.000000
          max      24622.000000
          Name: Corrected, dtype: float64
```

Figure 20: Description of Student's Correct Answers

Here we can identify that the minimum number of corrected answers is 2, which indicates some users only corrected only two questions; this also determines the maximum number of corrected answers is 24622, which indicates that specific users exist, which has the highest amount of correctness.

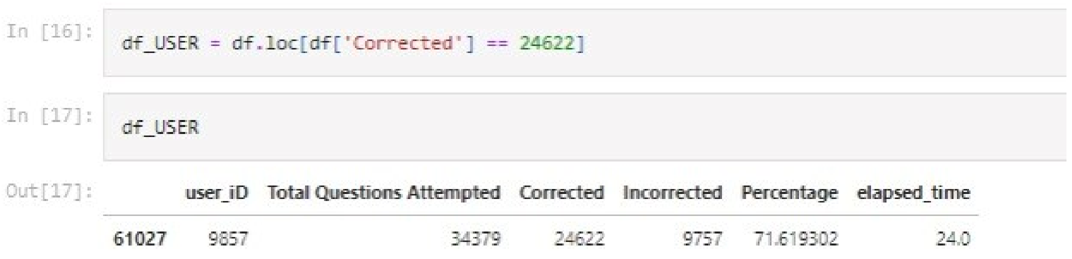
On Average, i.e., 50% of the users correct 93 questions, and 75% correct 293 questions.

3.5.4.10 Top Performing Users:

With all the information that was determined from describing the statistics, the next step is to identify which users are the top users whose percentages are greater or equal to 95%. Nevertheless, unfortunately, there were only 13 out of all 780K users whose percentages were greater or equal to 95%.

3.5.4.11 Identifying Top Performing Users:

With all the information that was determined from describing the statistics, the next step is to identify which users have attempted the maximum number of questions correctly. Below is the code snippet to find out those specific users.



```
In [16]:  df_USER = df.loc[df['Corrected'] == 24622]

In [17]:  df_USER

Out[17]:
```

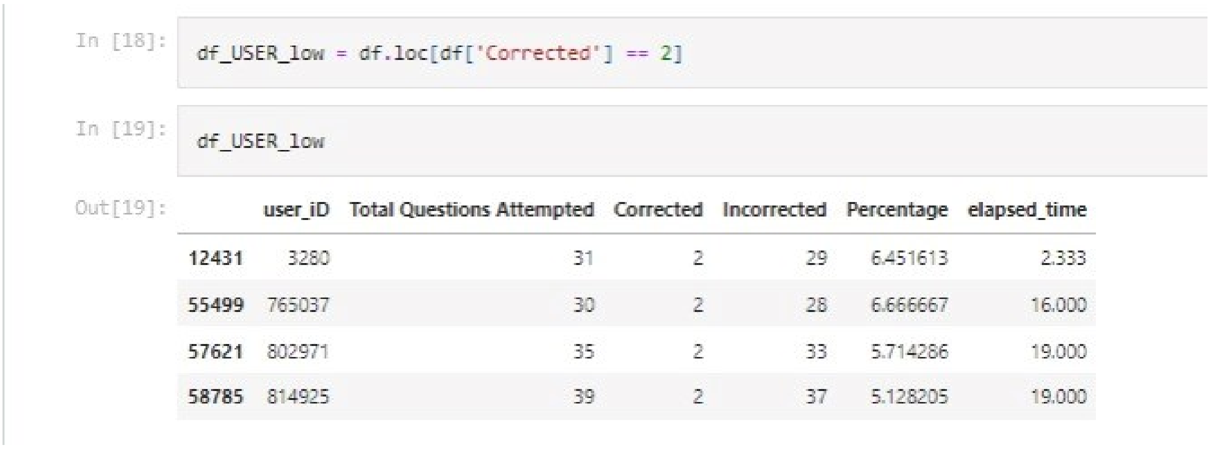
	user_iD	Total Questions Attempted	Corrected	Incorrected	Percentage	elapsed_time
61027	9857	34379	24622	9757	71.619302	24.0

Figure 21: Identification of the user who has corrected the most questions

Only one user attempted a total of 34379 questions, 24622 were corrected, and 24622 were incorrect, which indicates the percentage of the user to be 71.6%, and the average time spent on each question is 24 seconds.

3.5.4.12 Identifying Low-Performing Users:

With all the information that was determined from describing the statistics, the next step is to identify which users have attempted the minimum number of questions correctly. Below is the code snippet to find out those specific users.



```
In [18]: df_USER_low = df.loc[df['Corrected'] == 2]

In [19]: df_USER_low

Out[19]:
```

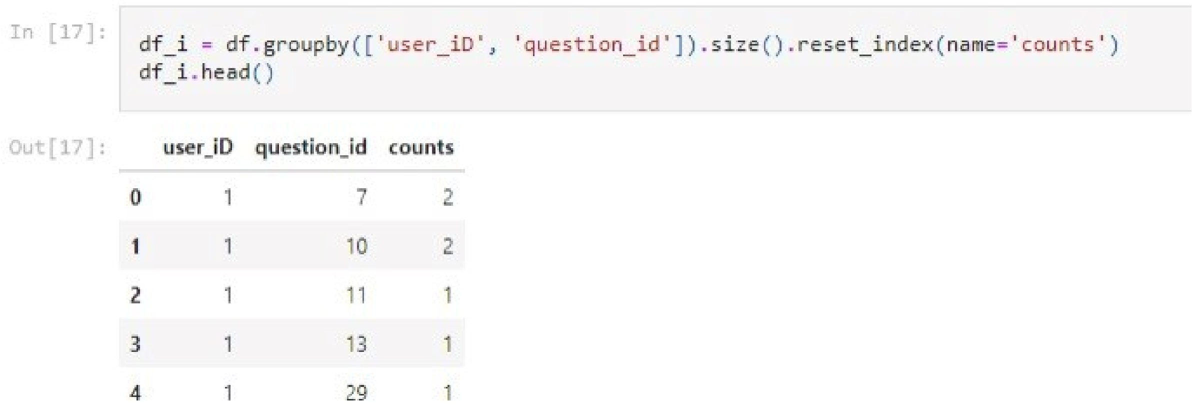
	user_iD	Total Questions Attempted	Corrected	Incorrected	Percentage	elapsed_time
12431	3280	31	2	29	6.451613	2.333
55499	765037	30	2	28	6.666667	16.000
57621	802971	35	2	33	5.714286	19.000
58785	814925	39	2	37	5.128205	19.000

Only four users attempted a different number of questions, but only 2 out of all the attempted questions were corrected.

3.5.4.13 Repeated Questions in the data frame:

This dataset is gathered with the help of the SANTA UI application via mobile and web applications. Thus, users could attempt questions previously shown to that specific user during the previous login, which increases the percentage of a user who knew the answer already to that specific question.

Below is the code snippet to find out to each user how many questions they have attempted twice or more and which specific question.



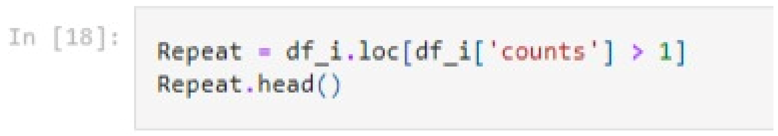
```
In [17]: df_i = df.groupby(['user_iD', 'question_id']).size().reset_index(name='counts')
         df_i.head()
```

```
Out[17]:    user_iD  question_id  counts
         0        1            7       2
         1        1           10       2
         2        1           11       1
         3        1           13       1
         4        1           29       1
```

Figure 22: Identification of the repeated questions count attempted by the user

Here it indicates that user one has attempted questions # 7 and 10 twice, and this list continues. Furthermore, the data frame also includes those questions that were only attempted once; thus, to get a clearer view, eliminate those questions that were only attempted once by each user.

Below is the code snippet

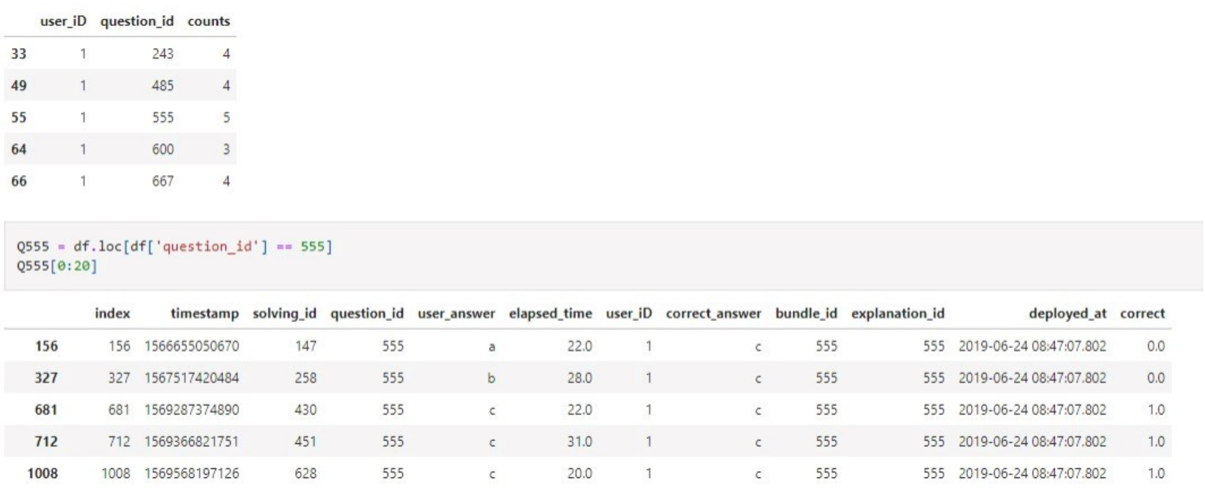


```
In [18]: Repeat = df_i.loc[df_i['counts'] > 1]
         Repeat.head()
```

Figure 23: Identification of attempted repeated questions attempted by the users

Now all the actual repeated questions have come to light from each user. Below is the example of user # 1, which attempted question # 555 five times. The first two interactions indicate that the answer provided by the user was incorrect. Still, the last three times, the user attempted the question correctly, which may affect the percentage.

As the correct answer to question # 555 was “c”, The user chose “a” the first time and “b” the second time, which determines to be incorrect. Then, for a third, fourth and final time, the user chose option “c”, which appears to be correct.



	user_iD	question_id	counts
33	1	243	4
49	1	485	4
55	1	555	5
64	1	600	3
66	1	667	4

```
Q555 = df.loc[df['question_id'] == 555]
Q555[0:20]
```

	index	timestamp	solving_id	question_id	user_answer	elapsed_time	user_iD	correct_answer	bundle_id	explanation_id	deployed_at	correct
156	156	1566655050670	147	555	a	22.0	1	c	555	555	2019-06-24 08:47:07.802	0.0
327	327	1567517420484	258	555	b	28.0	1	c	555	555	2019-06-24 08:47:07.802	0.0
681	681	1569287374890	430	555	c	22.0	1	c	555	555	2019-06-24 08:47:07.802	1.0
712	712	1569366821751	451	555	c	31.0	1	c	555	555	2019-06-24 08:47:07.802	1.0
1008	1008	1569568197126	628	555	c	20.0	1	c	555	555	2019-06-24 08:47:07.802	1.0

Figure 24: Identification of repeated question attempts and analysis in which time the answer matches the correct one

3.5.4.14 Removal of Repeated Questions in the data frame:

As many users have attempted multiple questions repeatedly thus, all the repetitive questions to each specific user have been dropped and kept the interaction where the user correctly answers the specific question. The same questions attempted by each user was removed based on the "user_iD", "question_id", and "correct" column from the primary data frame. Below is the code snippet

```
df['correct'].eq(1) & ~df.drop_duplicated(['user_iD','question_id','correct'])
```

```
mask = df['correct'].eq(1) & ~df.duplicated(['user_iD','question_id','correct'])
```

```
out = df[mask]
```

3.5.4.9 Updated Data frame

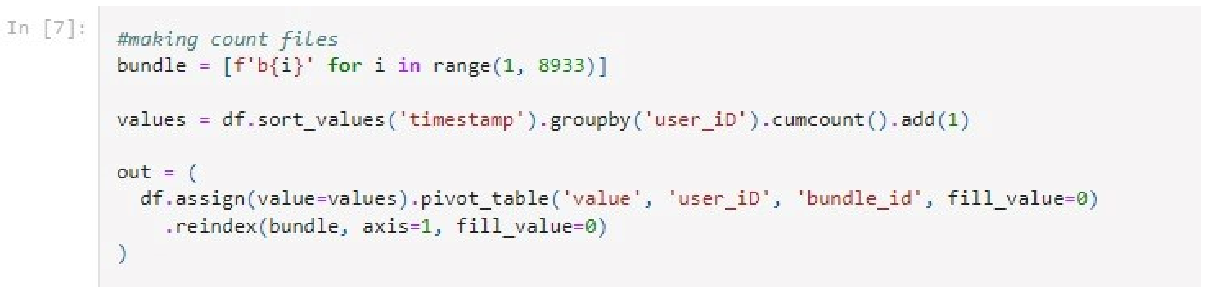
After the removal of repeated questions attempted by each specific user, most of the users lie in the bracket whose percentages are greater or equal to 80%, which concludes a data frame of almost 28 thousand users. As mentioned at the start of the methodology, there are 8932 bundles stated in the "bundle id" column. Each bundle contains 1, 2, 3, 4 and 5 questions.

3.5.4.15 Pivot Table Data frame:

All the incorrect attempts were also removed to create a data frame where all the users who belong to 80 percent or more categories are the top students and find out the trends of bundles order studied.

With the help of "bundle id", "timestamp", and "user_iD", The aim was to create a pivot table data frame which would indicate each user in what order they studied the specific bundle.

If a user does not study a specific bundle(s), it will place as 0; otherwise, it will fill up with the ascending order of bundles studied by each user. Below is the code snippet

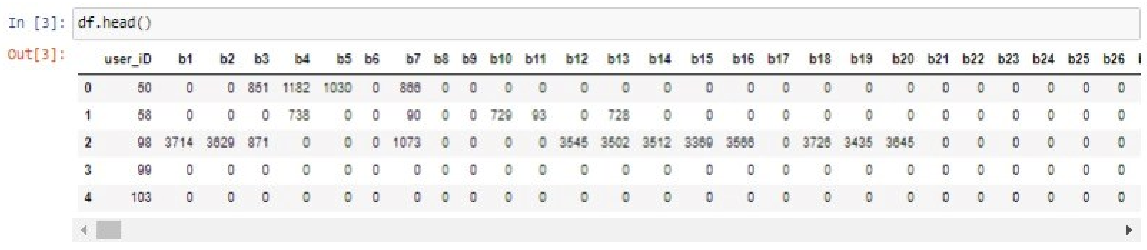


```
In [7]:  #making count files
         bundle = [f'b{i}' for i in range(1, 8933)]

         values = df.sort_values('timestamp').groupby('user_iD').cumcount().add(1)

         out = (
           df.assign(value=values).pivot_table('value', 'user_iD', 'bundle_id', fill_value=0)
             .reindex(bundle, axis=1, fill_value=0)
         )
```

Below is the preview of the pivot table data frame.

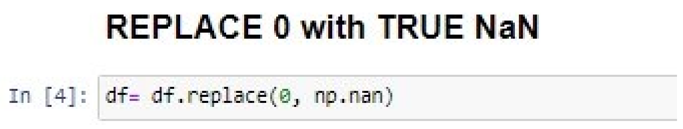


```
In [3]: df.head()
Out[3]:
   user_iD    b1    b2   b3    b4    b5  b6    b7  b8  b9  b10  b11   b12   b13   b14   b15   b16  b17   b18   b19   b20  b21  b22  b23  b24  b25  b26
0       50     0     0  851  1182  1030   0   866   0   0    0    0     0     0     0     0     0    0     0     0     0    0    0    0    0    0    0
1       58     0     0    0   738     0   0    90   0   0  729   93     0   728     0     0     0    0     0     0     0    0    0    0    0    0    0
2       98  3714  3629  871     0     0   0  1073   0   0    0    0  3545  3502  3512  3369  3588    0  3726  3435  3645    0    0    0    0    0    0
3       99     0     0    0     0     0   0     0   0   0    0    0     0     0     0     0     0    0     0     0     0    0    0    0    0    0    0
4      103     0     0    0     0     0   0     0   0   0    0    0     0     0     0     0     0    0     0     0     0    0    0    0    0    0    0
```

The dataframe indicates each user in ascending order of bundles studied.

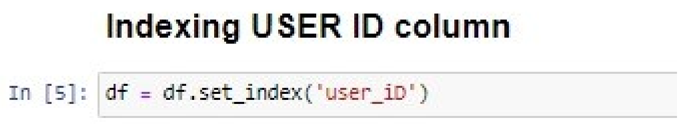
3.5.4.15 Replacing Values in the data frame:

As the pivot table indicates many zeros in the data frame, All the zeros were replaced with "True-NAN" and indexed the user id for the other process so that the user id column would not get affected. Below is the code snippet



REPLACE 0 with TRUE NaN

```
In [4]: df= df.replace(0, np.nan)
```

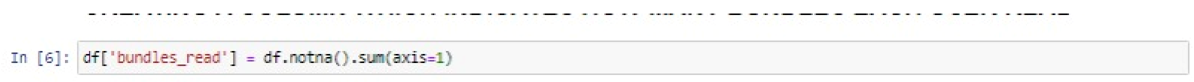


Indexing USER ID column

```
In [5]: df = df.set_index('user_iD')
```

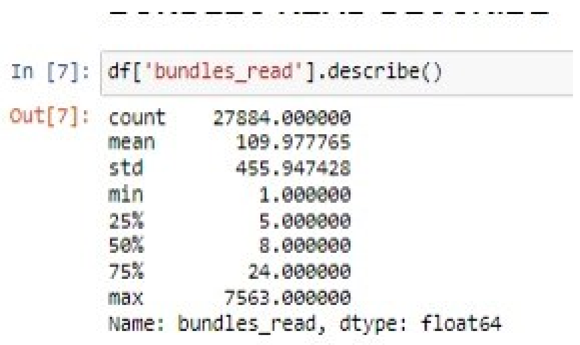
3.5.4.16 Updated Data frame Description:

Indexing the "user_id" was necessary to calculate the total number of bundles read by each user out of the total bundles; thus, a new column was created to store the count of total bundles read by each user. Below is the code snippet



```
In [6]: df['bundles_read'] = df.notna().sum(axis=1)
```

Since the bundle's read column was created, it contains the total number of bundles read by each user out of the total bundles, which is 8932.

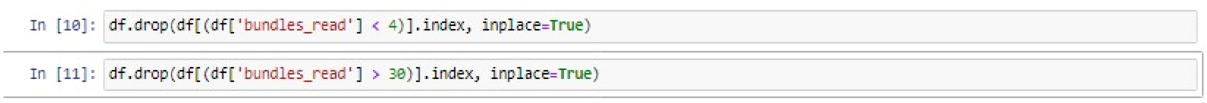


```
In [7]: df['bundles_read'].describe()
Out[7]: count    27884.000000
        mean       109.977765
        std        455.947428
        min          1.000000
        25%          5.000000
        50%          8.000000
        75%         24.000000
        max       7563.000000
        Name: bundles_read, dtype: float64
```

As the output indicates, most users only read between 5 and 24 bundles out of all the total bundles.

3.5.4.17 Removal of Users:

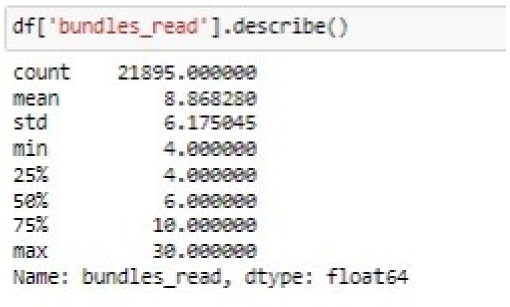
The total number of bundles in the data frame is 8932, and upon further investigation, it indicates most of the users only read between 5 and 25, respectively, out of all 8932. Thus, all the users were dropped who had read less than four and greater than 30 bundles. Below is the code snippet.



```
In [10]: df.drop(df[(df['bundles_read'] < 4)].index, inplace=True)

In [11]: df.drop(df[(df['bundles_read'] > 30)].index, inplace=True)
```

Here is also the detailed description of the "bundles_read" now



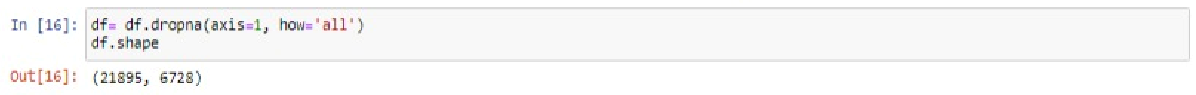
```
df['bundles_read'].describe()

count    21895.000000
mean         8.868280
std          6.175045
min          4.000000
25%          4.000000
50%          6.000000
75%         10.000000
max         30.000000
Name: bundles_read, dtype: float64
```

This description indicates that the minimum number of bundles read is 4, the maximum number of "bundles_read" is 30, and fifty percent of the users have read ten bundles. It also indicates the total number of users left in the data frame in the count as 21895 out of 27884 users.

3.5.4.18 Removal of Bundles:

The total number of bundles is 8932, there are still such bundles which no user has read, and it only contains values as "NaN", as it was replaced by 0 earlier. Thus, removing such bundles was also necessary as it does not play any part in all student interaction. Below is the code snippet and shape of a newly formed dataset



```
In [16]: df= df.dropna(axis=1, how='all')
         df.shape

Out[16]: (21895, 6728)
```

As this indicates, now the user's count is 21895, and the remaining bundles are 6728 out of 8932, meaning any user did not read 2204 bundles throughout the time.

3.5.5 Features Scaling

The goal of feature scaling is to normalize the variety of independent variables or features of data. Feature Scaling is done by transforming the data to fit within a specific scale, like 0-1 or -1 to 1. It is vital because machine learning algorithms typically work better when the input features have a consistent scale and distribution.

There are several standard techniques for feature scaling, such as:

- **Min-Max Scaling**: This method scales the data to a specific range, such as 0-1. Each value is scaled to 0 and 1, where 0 is the minimum value and 1 is the maximum value.
- **Standardization**: This method scales the data to standard normal distributions with a mean of 0 and a standard deviation of 1. Each value is transformed, so the new value has a mean of 0 and a standard-deviations of 1.
- **Normalization**: This method scales the data to a unit norm. Each value is transformed such that the sum of squares of the transformed values equals 1.

It is important to note that feature scaling should be applied after the data has been cleaned, preprocessed and transformed. In this project, the type of feature scaling used is not specified; thus, any of the above methods or other methods depend on the use case and data characteristics.

3.5.5.1 Min Max Scaler

Min-max scaling is used to scale the values of a particular feature or variable to a fixed range, usually between 0 and 1. Min Max Scaler is useful when the range of values for that variable is quite extensive and would otherwise have an outsized influence on any models or calculations that use it. In this case, the formula is:

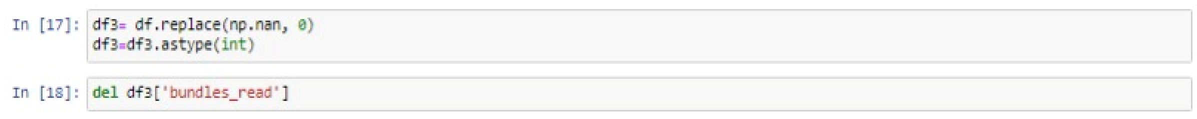
$$m = \frac{x - xmin}{xmax - xmin}$$

Where:

- m is our new value
- x is the original cell value
- xmin is the minimum value of the column
- xmax is the maximum value of the column

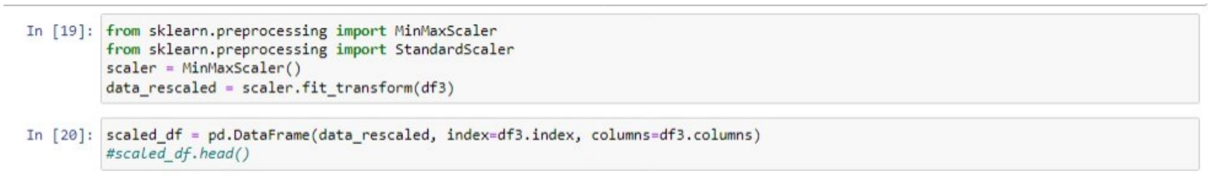
For Scaling, “MinMaxScaler” was applied because it scales all the dataset features in the range of either (0, 1) or if the features include negative values (-1, 1). In this case, the dataframe did not contain any negative values; thus, “MinMaxScaler” scaled all the features with 0 and 1. Before applying the min max scaler, all the “True-NaN” were replaced with 0, and the count of the "bundles_read" column was removed as it provides no further information or plays any role anymore.

Below are the code snippets



```
In [17]: df3= df.replace(np.nan, 0)
         df3=df3.astype(int)

In [18]: del df3['bundles_read']
```



```
In [19]: from sklearn.preprocessing import MinMaxScaler
         from sklearn.preprocessing import StandardScaler
         scaler = MinMaxScaler()
         data_rescaled = scaler.fit_transform(df3)

In [20]: scaled_df = pd.DataFrame(data_rescaled, index=df3.index, columns=df3.columns)
         #scaled_df.head()
```

3.5.5.2 Principal Component Analysis (PCA):

Principal component analysis, or PCA, is a dimensionality reduction technique often used to reduce the dimensionality of large data sets by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

Reducing the number of variables in a dataset set naturally comes at the expense of accurateness. Still, the trick in dimensionality reduction is to trade a little accuracy for simplicity. In addition, reduced data sets are easier to explore and visualize, making analyzing datasets much easier and quicker for machine learning procedures deprived of extraneous variables to process.

To sum up, the knowledge of PCA is unpretentious — condense the number of data variables while preserving as much information as possible.

After scaling, a dataset's dimensionality was still considered significant enough to be processed in the machine learning model. Thus, a technique named PCA was applied. PCA is a technique often used in machine learning and data science to reduce the dimensionality of a dataset by transforming large sets of variables into smaller ones that still contain most of the information of a large dataset. To summarize, PCA reduces the number of dataset variables while preserving as much information as conceivable. Thus, PCA was applied to the scaled dataset, which was scaled earlier by “MinMaxScaler”.

3.6 Machine Learning Model

This dataset indicates that this belongs to unsupervised learning. K-Means is a widely used clustering algorithm that partitions a dataset into K clusters where each data point fits the cluster with the nearest mean. The objective of the K-means procedure is to minimize the sum of the squared distances amongst each data point(s) and the mean of the cluster it belongs to. In this project, the K-means algorithm was applied to the PCA-transformed dataset to group users based on their behaviour or interactions within the learning platform.

Additionally, the K-Nearest Neighbors (KNN) algorithm is another unsupervised learning method that can be used for Recommender systems. The KNN algorithm determines the k number of closest points (neighbours) to a given data point and then makes predictions based on the most known target attributes of those k nearest neighbours. Identifying similar patterns in the dataset can make predictions for new data points.

In this project, The K-means algorithm was used for clustering the dataset and KNN was used for the recommendation system. The goal is to cluster the users in similar groups and make recommendations based on those groups, which is done by identifying each user's behaviour.

3.6.1 K-Means Algorithm:

The k-means algorithm is one of the most naive and popular algorithm unsupervised machine learning and data science algorithms. The real challenge in this algorithm is to set a value of K. The value of K indicates the total number of clusters to be made. K-means works by iteratively dividing the data into k clusters based on the distance of each data point from the centroid of each cluster. The algorithm starts by randomly initializing the centroids of the clusters, then assigns each data point to the cluster whose centroid it is nearby. It then calculates the new centroid for each cluster based on the data points assigned. This procedure is repeated until the centroids no longer change or until a particular stopping criterion is met.

One way to set the value of K is to use the elbow method, which plots the value of the cost function (e.g., the sum of squared distances amongst data points and cluster centroids) against the number of clusters and chooses the point at which the cost function starts to decrease at a slower rate as the number of clusters increases.

Data science and machine learning practitioners can use K-means to identify patterns and group similar data points together, which is more interpretable and valuable for further analysis.

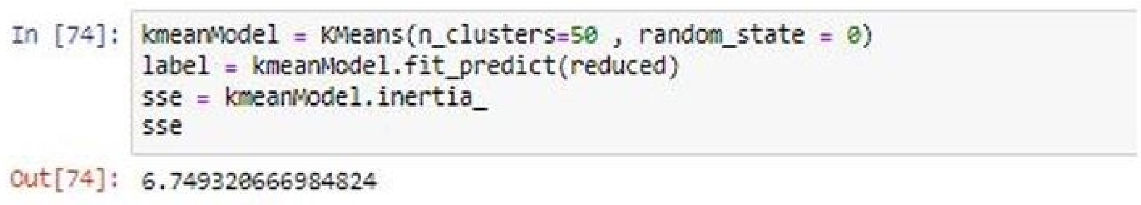
3.6.1.1 1st Integration:

The k-means algorithm randomly initialises K centroids, where K is the number of clusters to be made. Then, each data point is assigned to the cluster with the closest centroid. After that, the centroids are recomputed as the mean of all data points assigned to that cluster. This process is repeated until the cluster assignments no longer change or a maximum number of iterations is reached. The final result is K clusters, each containing a set of data points.

The value of K is a crucial parameter in the K-means algorithm, as it determines the number of clusters to be made. A standard method for choosing the value of K is to use the elbow method, which involves plotting the validation score (such as SSE) against different values of K and choosing the value of K where the validation score starts to flatten out.

In the case of this project, the value of K is set to 50 as an arbitrary value. The first iteration will indicate the number of set clusters, and the validation score as the SSE score will indicate the machine learning model's performance. The choice of K = 50 is arbitrary, and the results must be checked with other values of K to ensure that the results are optimal. SSE (Sum of Squared Errors) measures how far the data points in a cluster are from the centroid (mean) of that cluster. In K-means, the goal is to minimize the SSE to get the best clustering results. A low SSE value indicates that the data points in the clusters are very close to the centroid, which means that the clusters are well-defined.

Therefore, a low SSE score, like 6.74, is desirable in this case; it indicates that the clustering results are promising, which indicates that the algorithm has been able to group the data points into 50 clusters, so data points in each cluster are similar and close to the centroid of that cluster. Below is the code snippet

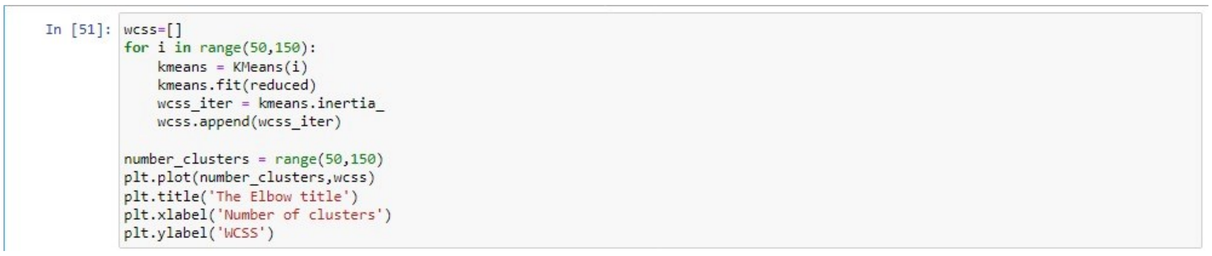


```
In [74]: kmeanModel = KMeans(n_clusters=50 , random_state = 0)
         label = kmeanModel.fit_predict(reduced)
         sse = kmeanModel.inertia_
         sse

Out[74]: 6.749320666984824
```

3.6.1.2 K-Means Elbow Method:

The K-Means elbow method can help determine the finest number of clusters for the K-Means algorithm. By providing a range of possible values for K and evaluating the algorithm's performance using a metric like the SSE score for each value, the method can help identify the point where the algorithm's performance plateaus, commonly referred to as the "elbow" of the plot. This elbow point is considered the optimal value of K for the algorithm, as adding more clusters beyond that point typically does not significantly improve performance. In this case, the range was provided from 50 to 150 clusters to determine the optimal value of K for this specific dataset, which will help optimize the clustering and best representation of data. Below is the code snippet



```
In [51]: wcss=[]
         for i in range(50,150):
             kmeans = KMeans(i)
             kmeans.fit(reduced)
             wcss_iter = kmeans.inertia_
             wcss.append(wcss_iter)

         number_clusters = range(50,150)
         plt.plot(number_clusters,wcss)
         plt.title('The Elbow title')
         plt.xlabel('Number of clusters')
         plt.ylabel('WCSS')
```

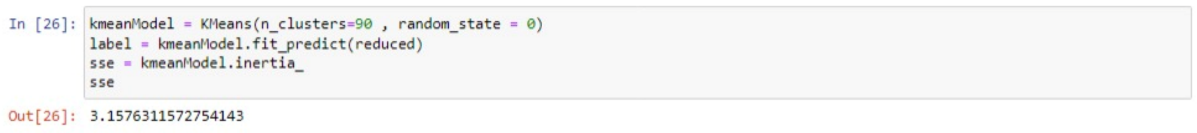
3.6.1.3 K-Means Elbow Method Plot:

The K-Elbow Visualizer is a technique used to determine the optimal value of K when applying the K-means algorithm. The technique runs K-means clustering on the dataset for a range of values of K and then computes an average score for all clusters for each value of K. By default, the distortion score, the sum of square distances from each point to its assigned centre, is computed. Other metrics, such as the silhouette score, or the calinski_harabasz score, can also be used. When these overall metrics for each model are plotted, it is possible to determine the best value for K visually. The "elbow" on the curve represents the inflexion point where the model fits the best, and in this case, the elbow is observed at K = 90, indicating that the optimal number of clusters for this dataset is 90.

The elbow method returns the output as a plot (discussed in the next chapter); the elbow was observed at the K-value on 90

3.6.1.2 2nd Integration:

Finding the optimal value for K in the K-means algorithm to be 90, the second iteration was done with the value of K to be set as 90 to determine the SSE score. Below is the code snippet

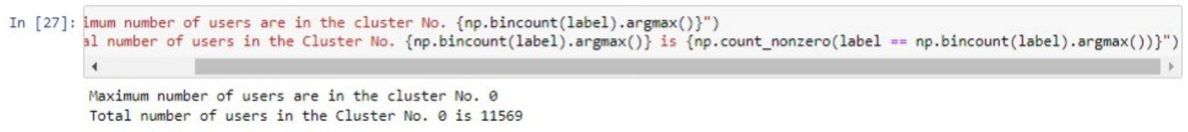


```
In [26]: kmeanModel = KMeans(n_clusters=90 , random_state = 0)
         label = kmeanModel.fit_predict(reduced)
         sse = kmeanModel.inertia_
         sse
Out[26]: 3.1576311572754143
```

It is great to see that the SSE score improved from the initial value of 6.74 to 3.15 after adjusting the number of clusters to 90. A lower SSE score typically indicates that the clusters are more tightly grouped and have a better fit for the data. Therefore, using 90 clusters for this dataset could lead to more accurate and meaningful results than using a different number of clusters.

3.6.1.4 Maximum number of users in a cluster:

It is important to note that clustering algorithms like K-means aim to group similar data points in the same cluster. Therefore, it is not necessary or meaningful to say that one cluster contains the "most users", as the goal is not to group data points by quantity but by similarity. Additionally, it is essential to consider the interpretation of the results, the final number of clusters obtained, the method used, if it makes sense to the problem, if the clustering provides valuable insights into the data, and if it is worth following the next step. To determine which specific cluster contains most of the users and which cluster number, below is the code snippet and results



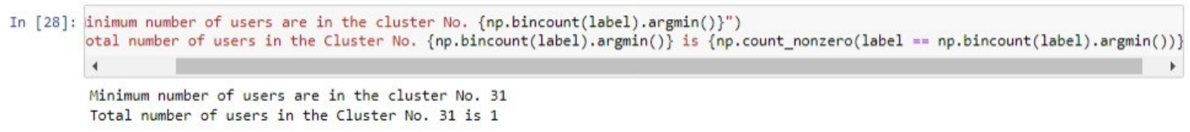
```
In [27]: imum number of users are in the cluster No. {np.bincount(label).argmax()}")
         al number of users in the Cluster No. {np.bincount(label).argmax()} is {np.count_nonzero(label == np.bincount(label).argmax())}")

         Maximum number of users are in the cluster No. 0
         Total number of users in the Cluster No. 0 is 11569
```

The above output indicates the first cluster, which is Cluster # 0, contains most of the users with a count of 11569 out of almost 22 thousand users

3.6.1.5 Minimum number of users in a cluster:

To determine which specific cluster contains the minimum number of users and which cluster number is, below is the code snippet and results



```
In [28]: inimum number of users are in the cluster No. {np.bincount(label).argmin()}")
         otal number of users in the Cluster No. {np.bincount(label).argmin()} is {np.count_nonzero(label == np.bincount(label).argmin())}

         Minimum number of users are in the cluster No. 31
         Total number of users in the Cluster No. 31 is 1
```

The above output indicates that cluster, which is Cluster # 31 contains the lowest number of users, as it only contains one user.

3.6.2 Recommender System Algorithm:

A recommendation engine is a type of machine learning that typically deals with ranking or rating systems found on websites like YouTube and Netflix and online retailers such as Amazon and Daraz. These systems also consider the behaviour of individual users. In this case, as we have a dataset that is user-based, the recommendation system will analyze similar trends among different users and make suggestions to both new and existing users

3.6.2.1 KNN Recommender System Algorithm:

The K-Nearest Neighbors (KNN) algorithm is a popular choice for building a recommendation system. The basic idea behind this algorithm is that it finds the k-number of users most similar to a given user based on their past interactions or behaviour. These similar users are then used to make recommendations to the given user.

The first step in developing a recommendation system using KNN is determining user similarity, which can be done using various metrics such as cosine similarity or Euclidean distance. Once the similarity between users is calculated, the k-nearest neighbours are found for each user. These nearest neighbours are then used to make recommendations to the given user.

In addition to the KNN algorithm, other techniques, such as collaborative filtering and matrix factorization, can also be used to build recommendation systems. Collaborative filtering is based on the idea that users with similar preferences will likely have similar preferences in the future. On the other hand, Matrix factorisation is a technique used to decompose a large sparse matrix into smaller and denser matrices. These matrices can then be used to make recommendations.

Overall, the choice of algorithm for a recommendation system depends on the problem's specific requirements and the data availability. The KNN algorithm is a simple and effective method for building a recommendation system. Still, other techniques, such as collaborative filtering and matrix factorization, may also be considered depending on the context.

3.6.2.2 KNN Algorithm Metrics:

The KNN algorithm, or K-Nearest Neighbors, is a technique used in machine learning to determine the similarity between data points by considering their distance. Various metrics can be used to calculate the distance between data points, and each is suited for different data types and use cases. Some standard metrics used in KNN include Euclidean distance, Manhattan distance, and Minkowski distance. Euclidean distance is often used for continuous variables, Manhattan distance for discrete variables, and Minkowski distance for a combination of both. Other metrics like cosine similarity, correlation distance, and hamming distance can also be used based on the use case and data characteristics.

KNN METRICS	USE CASE
Minkowski Distance	Normal Vector Space

Manhattan Distance	Taxi-Cab Distance or City Block Distance
Euclidean Distance	Distance between two straight lines in Euclidean space
Cosine Distance	Similarities between two vectors
Jaccard Distance	Similar to Cosine, but it looks for where two vectors approach the value 1
Hamming Distance	Comparing two binary data strings

Table 4: KNN Algorithm Metrics

3.6.2.3 KNN Metric Cosine Distance:

The cosine similarity metric measures the cosine of the angle between two vectors in a multidimensional space. It is often used when working with text data, as it can measure the similarity of two text documents by treating them as vectors in a high-dimensional space. In this case, the KNN algorithm would use the cosine similarity metric to determine the similarity between users based on their interactions with the web application and other sensor data and make recommendations to new or existing users based on that similarity.

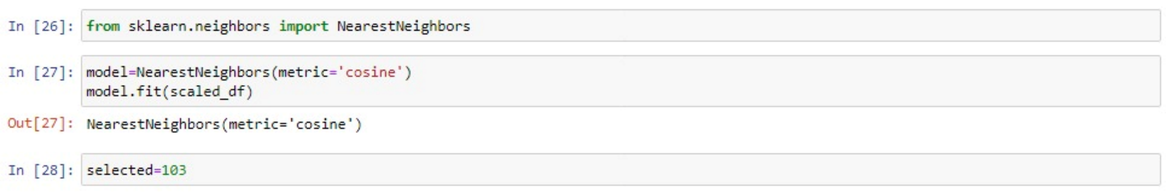
It also works well in high-dimensional spaces because it is less affected by the curse of dimensionality. It also does not require any parameters to be set, which makes it easy to use and interpret. For example, the formula for the cosine metric is given below:

$$\cos\theta = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\cdot\|\vec{b}\|}$$

Thus, the KNN machine learning algorithm is used to build a recommender system with cosine metric as a distance calculator.

3.6.2.4 KNN Recommender System Machine Learning modelling:

KNN algorithm to the scaled dataset using the cosine metric to measure similarity between vectors. It uses scaled data to look for similar user trends and make recommendations based on those trends. This process allows for personalized recommendations for each user based on their past interactions and behaviours. The code snippet of the KNN machine learning algorithm was applied to the scaled dataset, which was scaled with “MinMaxScaler” to look for similar trends and eventually recommend a specific user to the other.



```
In [26]: from sklearn.neighbors import NearestNeighbors

In [27]: model=NearestNeighbors(metric='cosine')
         model.fit(scaled_df)

Out[27]: NearestNeighbors(metric='cosine')

In [28]: selected=103
```

The KNN algorithm, using the cosine metric, compares the vector of the chosen user (in this case, user 103) to the vectors of all other users in the dataset. It then identifies the users most similar to user 103 based on the cosine similarity measure and recommends them as similar users. This recommendation can be helpful in several applications, such as making personalized content or product recommendations to users on a website or app. The KNN algorithm can be used for various tasks, including recommendation systems, classification, and regression. In this case, it was used to find similar trends among different users in the dataset and make recommendations to a new or existing user. The algorithm was applied to a scaled dataset, which was scaled using the MinMaxScaler. The KNN algorithm uses a distance metric to determine the similarity between observations. In this case, the cosine similarity metric was used, which compares the direction of vectors to determine similarity. The machine learning model returned six similar users to the selected user (user # 103) as recommendations. It is significant to note that the recommendations provided by the machine learning model are based on the data provided in the dataset. Therefore, the accuracy of the recommendations may vary depending on the quality and relevance of the data. In the next chapter, it would be beneficial to analyze the recommendations provided by the model and evaluate their effectiveness.

Chapter 4: Result and Discussion

This chapter describes the methodology and results of a research project aimed at creating a machine learning-based recommendation system for an online education system. This project provides an in-depth discussion and personal analysis of the processes used to analyze student interactions. The primary objective of this project venture is to classify similar students' workflows, grouping them into clusters and utilizing these clusters to develop a recommendation system. Two machine learning models cluster students based on their typical patterns and create a recommendation system. Additionally, the algorithm's time and memory requirements are considered to create an efficient solution that balances accuracy with performance.

4.1 Students Patterns & Analysis

The data utilized in this research study comprises a record of student interactions, including timestamps. For a data scientist, one of the first steps when working with this type of data is to gain a general understanding of the student's behaviours, such as when and what they are studying, as well as the impact of external factors. This initial step is essential to identify the essential information before moving on to a more specific or targeted analysis.

4.1.1 Top 50 Active Users

The dataset in question contains information on the number of students and their interactions to identify which specific users have the highest interactions throughout the dataset collected by SANTA UI. To visualize the top 50 active users, a histogram was plotted with the specific user ID on the X-axis and the number of interactions on the Y-axis, which allows for a clear and straightforward representation of the distribution of interactions among the different users.

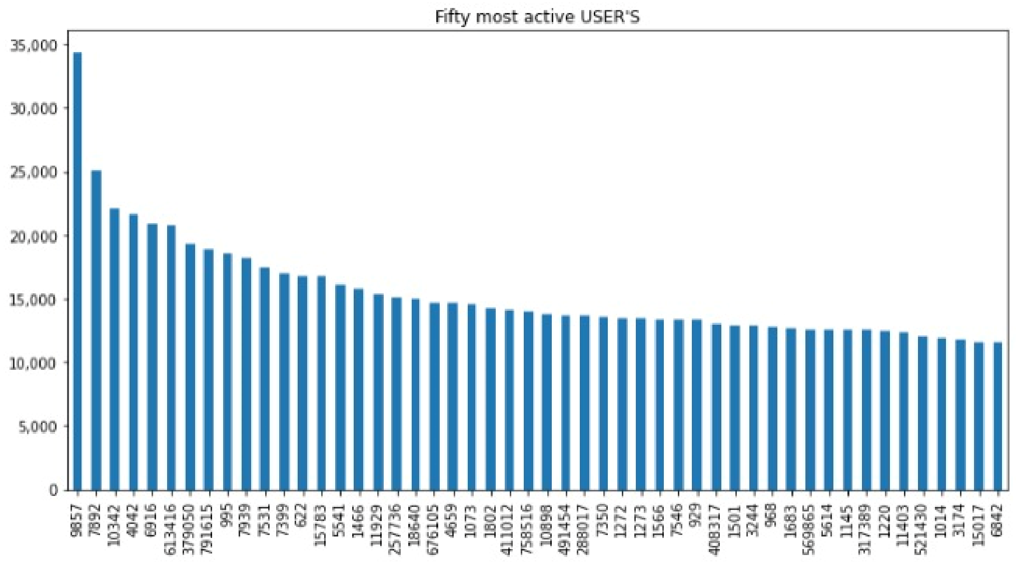


Figure 25: Histogram of Top 50 Active Users

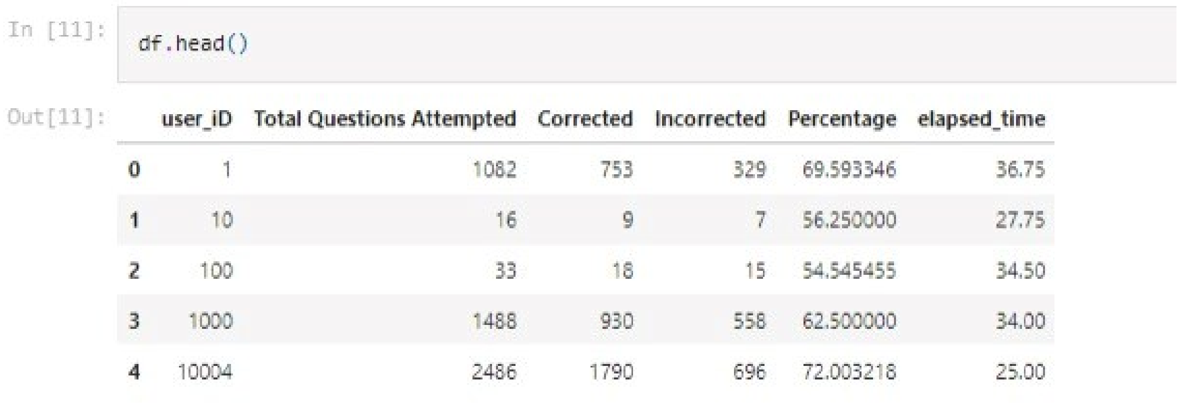
The histogram allows us to identify the user with the highest number of interactions. In this example, the user has attempted almost 35000 questions throughout the data collection period. This information provides insights into user engagement with the system. It is essential to point out that this histogram would help determine the students' engagement and identify outliers. Still, it might not be enough to determine the student's success or the system's effectiveness.

4.1.2 Users Data Statistics

The dataset provides information on the performance of individual students, including the number of questions attempted, the number correctly answered, and the number answered incorrectly. To quantify the students' performance, a percentage of correct answers is calculated using the traditional mathematical formula of

$$\frac{\text{Corrected}}{\text{Total Questions Attempted}} \text{x}100$$

The statistics provide a standardised and straightforward measure of student performance across the dataset. This percentage is a standard metric for measuring multiple-choice questions performance. However, it is essential to note that it may not be the best metric for all subjects or evaluation methodologies. Still, it is a valuable tool for quickly analysing the dataset performance.



```
In [11]:  df.head()

Out[11]:
   user_iD  Total Questions Attempted  Corrected  Incorrected  Percentage  elapsed_time
0        1                       1082        753          329   69.593346         36.75
1       10                         16          9            7   56.250000         27.75
2      100                         33         18           15   54.545455         34.50
3     1000                       1488        930          558   62.500000         34.00
4    10004                       2486       1790          696   72.003218         25.00
```

Figure 26: Each User's statistics for correctness, incorrectness, respective percentages and on elapsed time on average on each question

Lastly, the dataset also includes information on the elapsed time for each student, indicating the average number of seconds spent on each question attempted. This information can provide insight into the student's study habits and engagement with the material. For example, User 1 has spent an average of 36.75 seconds on the 1082 questions attempted. In addition, this information can be used with the percentage of correct answers to try to identify the factors related to success and engagement.

4.1.3 Top Performing Users

With all the information gathered, the next step is to identify the top students, which in this case are those whose percentages of correct answers are greater than or equal to 95%. The dataset shows that only 13 students fall into this category. A histogram plot is created with this information, with the user ID on the X-axis and the percentage of correct answers on the Y-axis, which allows for an easy and intuitive representation of the distribution of performance among the top-performing students.

It is significant to point out that even though this process would be an excellent way to identify top-performing students and outliers, it is not the only metric to evaluate the system's success or the students. Other factors such as engagement, effort and learning may need to be considered to have a more holistic perspective of student performance.

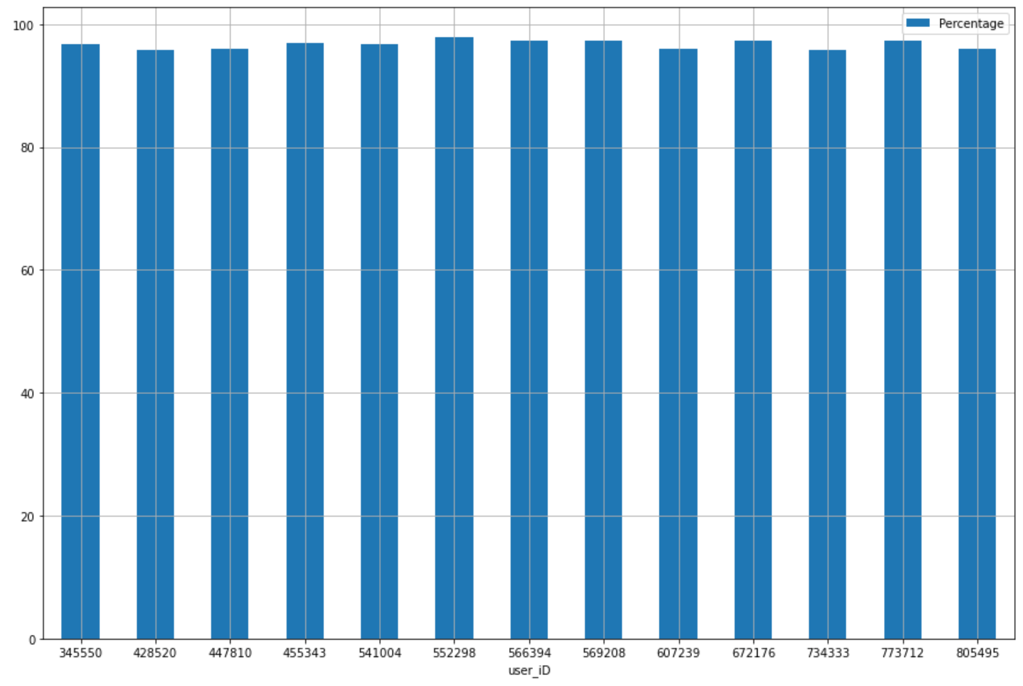


Figure 27: Histogram of Top Performing Users

4.1.4 Top Users vs Questions Attempted

As previously discussed, the dataset was analyzed to identify students with percentages of correct answers greater than or equal to 95%. An additional analysis that can be done with this information is a histogram frequency plot that illustrates the relationship between the number of questions attempted and the percentage of correct answers. The x-axis represents the total number of questions attempted. At the same time, the y-axis shows the frequency of students with a certain percentage of correct answers for a given number of questions attempted. These stats can help identify patterns and trends in student performance and look for possible relationships between the number of questions attempted and the quality of the performance.

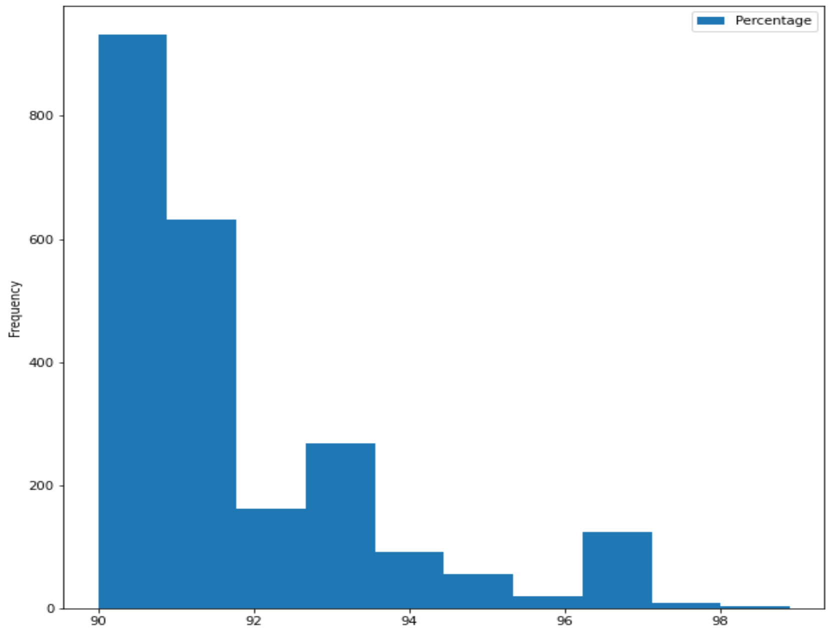


Figure 28: Frequency Plot between Users & Questions

Here it indicates that almost 900 questions are attempted by some users, which have 90 percent and less than 100 for the percentage of 97 and above.

4.1.5 Users vs Percentages

The histogram frequency plot illustrates a relationship between the number of questions attempted and the percentage of correct answers. One possible pattern that can be observed from the plot is that students who attempted between 900 and 1000 questions may have a percentage of correct answers ranging from 90% to 97% and above. This information can indicate that there might be an optimal point in the number of questions attempted for the students to perform at the best level. However, this correlation does not imply a causal relationship; other factors, such as students' prior knowledge, study habits, and strategies, should also be considered. Additionally, as this is only a part of the data, more analysis and larger sample size are needed to have a more robust conclusion.

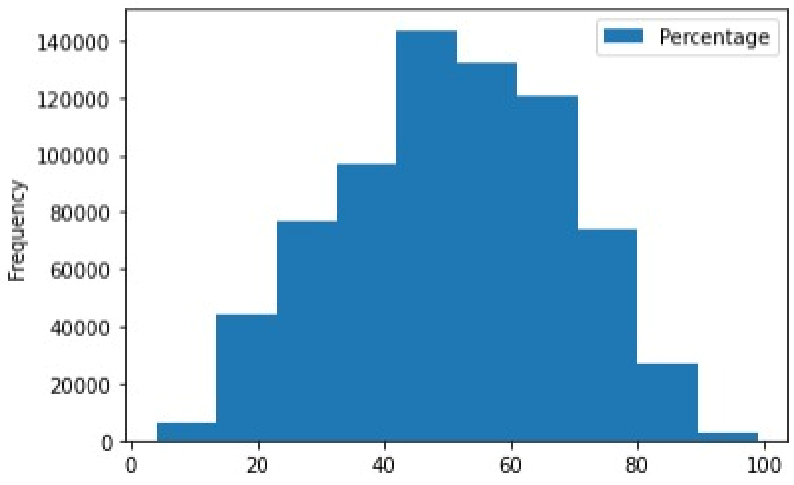


Figure 29: Frequency plot between Users with Percentages

4.1.6 Users vs Bundles Read

The dataset includes an additional column named "bundles read," which indicates the total number of bundles read by each user out of 8932 total bundles. This column could be used further to analyze students' engagement with the online education system. A histogram frequency plot of this column could help identify student behaviour patterns. For example, the plot may show that most students read less than 1000 bundles, with a few outliers reading between 2000 to 3000 bundles. Outliers, in this case, are the users far from the rest of the sample and may or may not be significant. However, in this case, it could be argued that reading so many bundles might not represent a typical student, and therefore these users can be removed from the analysis.

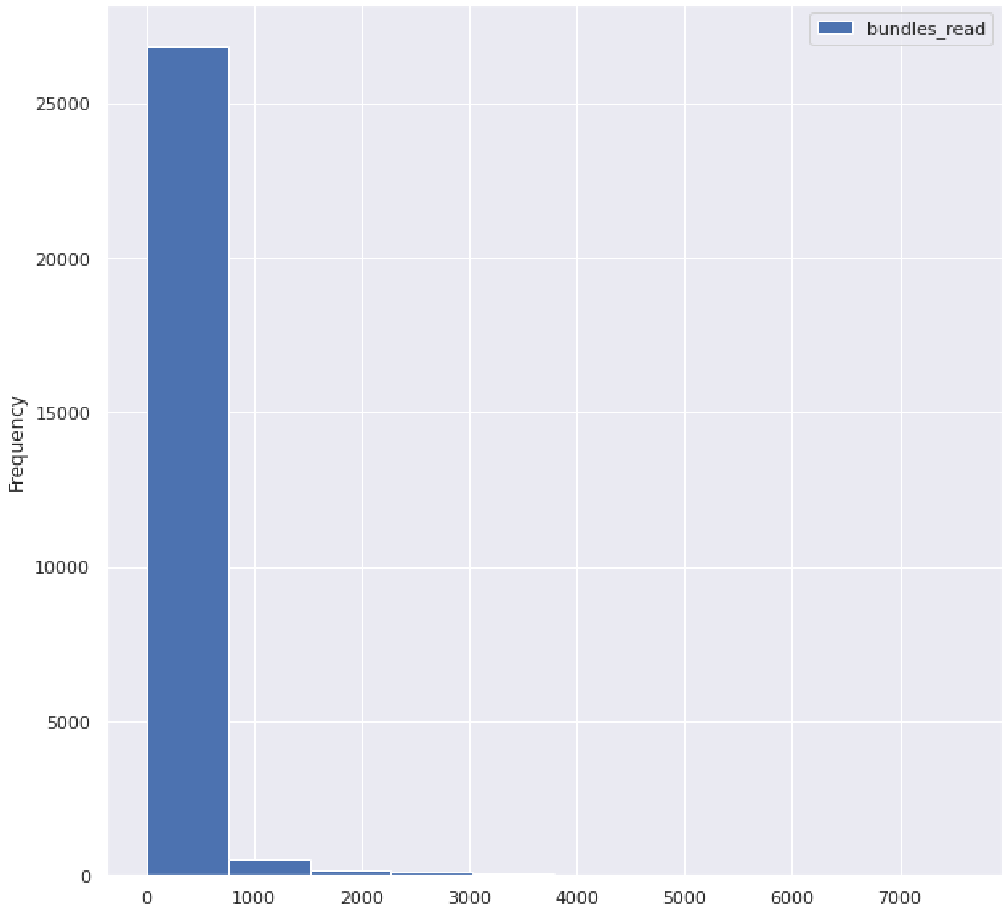


Figure 30: Frequency plot between Users with Total Bundles Read

4.1.6.1 Reduced Users vs Bundles Read:

The "bundles read" column in the dataset contains the count of the total number of bundles read by each user out of the total number of available bundles, which is 8932. This column provides additional information on the engagement and study habits of the students in the online education system. The information below can be used to understand how students interact with the system and how many resources each student is accessing.

Count	27844.0000
Mean	109.977765
Std	455.947428
Min	1.000000

25%	5.00000
50%	8.00000
75%	24.00000
Max	7563.0000

Table 5: Identification of quartiles bundles read by students

As the output indicates, most users only read a small number of bundles, specifically between 5 and 25 out of the total 8932 bundles. To further refine the analysis, users who have read less than four bundles or more than 30 bundles are removed from the dataset. This filter is applied to exclude users who might not represent the average student, which can bias the results. After this filtering, a new histogram frequency plot is generated, showing a more focused distribution of the remaining users in a narrower range of read bundles. It is essential to point out that removing data in this way may help to reduce noise and make it easier to identify patterns.

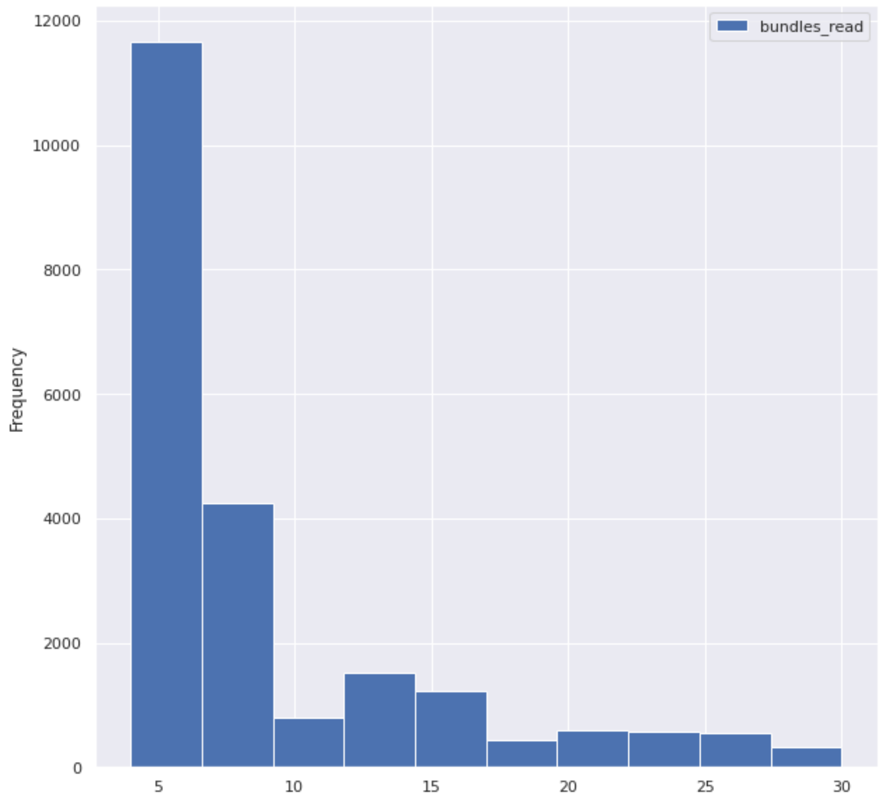


Figure 31: Frequency plot between Users with Modified Bundles_Read

Here is also the detailed description of the "bundles_read" now

Count	21895
Mean	8.868280
Std	6.175045
Min	4.000000
25%	4.00000
50%	6.00000
75%	10.00000
Max	30.0000

Table 6: Identification of quartiles after the un-read bundles read by students

This information suggests that after applying the filter and removing users who have read less than four bundles or more than 30 bundles, the remaining dataset still has a wide range of bundles read. For example, where the minimum number of bundles read is 4, the maximum is 30, and the median or 50 percent of the users read ten bundles. Additionally, the total number of users remaining in the dataset after this filtering is 21895 out of the initial 27884 users.

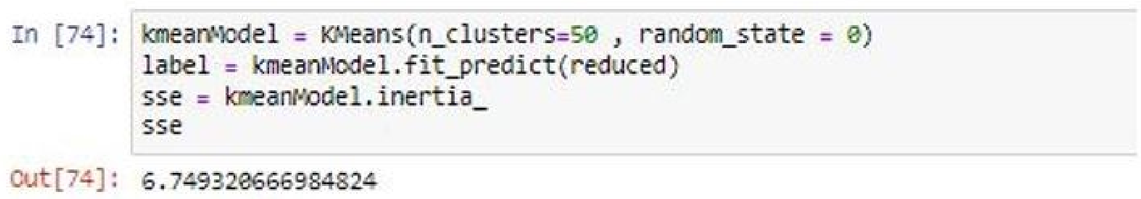
It is worth noting that these numbers will vary depending on the parameters set to filter; they could be adjusted to different values depending on the research goals. With this new data frame, the next step would be to analyze this information to identify patterns and gain insights into the students' behaviour.

4.2 K-Means Clustering

K-means is a popular unsupervised machine-learning algorithm that is often used for clustering. One way it can be applied in this case is to group the remaining users in the dataset into "clusters" based on their similarities, such as the number of bundles read.

One way to do this is by setting the number of clusters to 50 as an arbitrary value. However, it is important to note that the value of k is often a parameter chosen before running the algorithm and should be carefully selected.

The process of selecting the best value of k is usually referred to as the "elbow method" or "silhouette method", which consists in running the algorithm for different values of k and comparing the results. The goal is to find a k that yields the best trade-off between computational cost, interpretability, and performance. For example, with an arbitrary value of 50, the algorithm's performance will be evaluated to see if it produces sensible clusters and if this value of k needs to be adjusted.



```
In [74]: kmeanModel = KMeans(n_clusters=50 , random_state = 0)
         label = kmeanModel.fit_predict(reduced)
         sse = kmeanModel.inertia_
         sse
Out[74]: 6.749320666984824
```

4.2.1 SSE (Sum of Squared Errors) Score:

Cluster analysis is a statistical technique that groups similar consumers or respondents into distinct market segments or clusters. The goal of cluster analysis is to minimize the sum of squared errors (SSE) amongst the consumers in each cluster and the cluster centroid (the centre of the cluster).

In this use case, the SSE score of 6.749 suggests that the consumers in the same cluster are relatively similar, which is considered a good score for a sizeable unsupervised dataset, as it means that the algorithm finds tightly packed clusters. First, however, it is essential to validate the algorithm's performance by looking for the optimal value of k.

The elbow method is a common technique for determining the optimal number of clusters. It consists of plotting the SSE for different values of k and looking for the "elbow point", the value of k where the decrease in SSE starts to level off. This point is often considered the optimal value of k. Therefore, plotting the elbow plot for K-means and re-calculating the SSE score for different k values would better understand the optimal value for k in this use case. Furthermore, it would allow us to see if the SSE improves as “k” increases or if it stays stable after a certain point, indicating the optimal number of clusters.

4.2.2 K-Means Elbow Plot:

The elbow method returns an output as a plot, and the point where the rate of decrease in the SSE slows down is considered the elbow point. The value of k at this point is considered the optimal value of k for the given dataset.

The elbow method indicates an optimal value of k = 90 for this dataset, which suggests that using 90 clusters to segment the data is likely to produce the best balance between the number of clusters and the similarity of consumers within each cluster. As a result, the SSE score should be re-calculated with the new optimal value of k = 90 to ensure that it produces a better cluster solution than k = 50 and that the results are more meaningful and interpretable.

It is important to note that the elbow method is one way to determine the best number of clusters, but different methods may have different outcomes. Therefore, it is also essential to validate the results with other metrics, such as silhouette or Davies-Bouldin index and try to understand the meaningfulness and interpretability of the clusters.

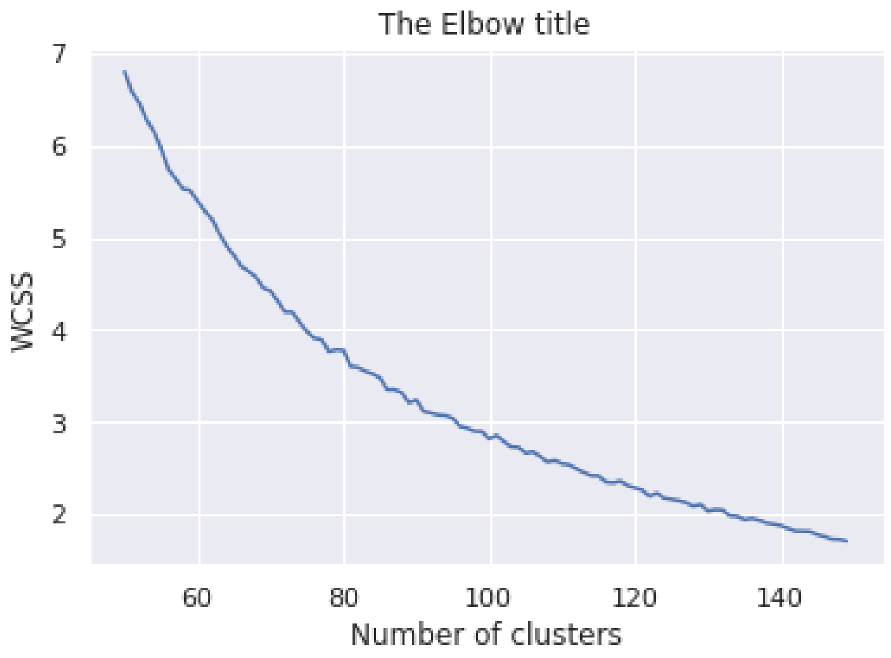


Figure 32: K-Means Elbow Plot

4.2.3 2nd Iteration SSE (Sum of Squared Errors) Score:

After determining the optimal value for the number of clusters using the elbow method, a second iteration of the K-means algorithm was conducted with the number of clusters set to 90. The new SSE score of 3.15 suggests that the new cluster solution is significantly better than the previous iteration. The score is much closer to zero and almost half the difference from the previous SSE score of 6.749.

This new score indicates that K-means's machine learning algorithm can partition the data more meaningfully and similar group students in fewer clusters.

It is worth noting that it is essential to validate these results with other metrics and with a comparison to ground truth (if available) to determine the model's effectiveness and the interpretability of the clusters.

Additionally, it is essential to remember that clustering is an unsupervised problem, and there is no guarantee that the results will be optimal; the objective is to find a "good enough" solution that helps to achieve the research goals.

4.3 Recommender System:

KNN (K-Nearest Neighbors) is another popular machine learning algorithm that can be used for developing a recommender system by finding the most similar users to a selected user based on their interactions with the system.

In this case, an arbitrary value for the number of recommended users was set to 6, which means the algorithm will recommend six similar users to a selected user. The value of k can be set to a higher or lower value depending on the research goals and the specific use case. A higher k will increase the diversity of the recommended users but also might include less similar users. A lower k will provide more similar users but might include less diverse options.

For example, suppose the number of recommending users was set to 6, and we want to see recommendations for user 103. In that case, the KNN algorithm will find the six nearest neighbours (most similar users) to user 103 based on their interactions with the system and recommend them to user 103.

It is essential to point out that KNN is a simple and robust algorithm, but it can be sensitive to the scaling of the features and the choice of the distance metric used. Additionally, it is essential to validate the results and ensure that the recommendations are helpful for the user.

In this case, the variable "selected" is assigned a value of 103, representing a specific user in the dataset. Then, the KNN algorithm is applied to this user to find the nearest neighbours (most similar users) based on their interactions with the system. In this case, the nearest neighbours' value is set to 6, which means that the algorithm will recommend six similar users to the selected user 103.

The value of 6 is arbitrary and can be adjusted depending on the research goals and the specific use case. For example, the larger the value of nearest neighbours, the more recommendations will be provided to the selected user, which might include less similar users. Conversely, a smaller value of nearest neighbours will provide fewer recommendations and might include more similar users.

The KNN algorithm was implemented, and how it can be used to recommend similar high-performing users to the selected user 103. It is essential to point out that the recommendations generated by KNN should be validated and tested to ensure they are helpful for the user.

The KNN algorithm uses the selected user's features and trends to find the most similar users in the dataset, in this case, user 103. Then, based on these similarities, the algorithm provides a list of six recommended users likely to have similar interactions with the system and can serve as guides for the selected user.

From these recommendations, it is possible to investigate the recommended users' study habits, such as the order of the bundles they read and the resources they accessed, and use this information to help the selected user to optimize their study path and succeed in the respective course.

It is worth noting that providing similar users as recommendations could help the selected user perform similarly. Still, it has not been guaranteed, as every student is unique and has different ways of learning. Moreover, it is essential to consider other factors, such as user preferences, the context and the final goal.

Chapter 5 Conclusions

Recommender systems have become an integral part of our lives in recent years. They are used in many applications, including online education, e-commerce, and social media. These systems use advanced algorithms and large datasets to provide personalized recommendations to users based on their preferences and interactions with the system.

In the case of online education, a recommender system can help students to discover new learning resources and optimize their study path. It can do so by analyzing the interactions of millions of students with the platform and grouping them into clusters based on their similarities.

Then, providing recommendations to a specific student based on the patterns of the other students in the same cluster can be very helpful for students, who can learn from the experience and progress of similar students and optimize their performance.

Furthermore, recommendation systems have a wide range of usage. For instance, Amazon uses it to recommend new products to customers based on their browsing and purchase history. Netflix uses it to suggest new shows and movies to users based on their viewing history. YouTube uses it to recommend new videos to users based on their watch history, and it is a crucial element for the success of platforms like these, as it helps retain the user by providing personalized content.

5.1 Future Aspects:

The recommender system developed in this project can be personalized for each educational institution's Learning Management System (LMS) to recommend a learning path for any course to all the students. It can also indicate specific chapters of a topic in a course that a student needs to re-learn before attempting a final exam based on their performance evaluation of that chapter.

This project can be very beneficial for the students and teachers, as it can help provide personalized tutoring to each student and generate feedback to improve the recommendations. In addition, as the machine learning algorithm learns and adapts, it can provide increasingly precise recommendations.

In addition, this kind of system can also provide valuable insights for educational institutions and course designers to understand how their students interact with the materials and where the struggles or opportunities lie. For instance, it could detect that a specific type of problem or activity is helping some students to learn better than others and could be applied to other students.

It is worth noting that a recommender system like this would need a significant amount of data to train on. Feature engineering and model selection would be necessary to get accurate recommendations. Additionally, the system should be validated and evaluated by experts in the field to ensure that it satisfies the institution's goals.

References

[1] Ya Wang, Yanmei Yang, Research on the Application of Artificial Intelligence in Education, 2020 15th International Conference on Computer Science & Education (ICCSE). https://doi.org/10.1164/rccm.201504-0781OC

[2] Huiyan Li, Hao Wang, Research on the Application of Artificial Intelligence in Education, 2020 15th International Conference on Computer Science & Education (ICCSE). https://doi.org/10.1109/ICCSE49874.2020.9201743

[3] Olaf Zawacki-Richter, Victoria I. Marín, Melissa Bond & Franziska Gouverneur, Systematic review of research on artificial intelligence applications in higher education, International Journal of Educational Technology in Higher Education volume 16, Article: 39 (2019) https://doi.org/ 10.1186/s41239-019-0171-0

[4] Stefan A. D. Popenici & Sharon Kerr, Research and Practice in Technology Enhanced Learning volume 12, Article: 22 (2017) https://doi.org/ 10.1186/s41039-017-0062-8

[5] Huang, Z., Yin, Y., Chen, E., Xiong, H., Su, Y., Hu, G., et al.: Ekt: Exercise aware knowledge tracing for student performance prediction. IEEE Transactions on Knowledge and Data Engineering (2019) https://doi.org/10.1109/TKDE.2019.2924374

[6] Lee, Y., Choi, Y., Cho, J., Fabbri, A.R., Loh, H., Hwang, C., Lee, Y., Kim, S.W., Radev, D.: Creating a neural pedagogical agent by jointly learning to review and assess. arXiv preprint arXiv:1906.10910 (2019) https://arxiv.org/abs/1906.10910

[7] Pandey, S., Karypis, G.: A self-attentive model for knowledge tracing. arXiv preprint arXiv:1907.06837 (2019) https://arxiv.org/abs/1907.06837

[8] Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L.J., SohlDickstein, J.: Deep knowledge tracing. In: Advances in neural information processing systems. pp.505–513 (2015). https://dl.acm.org/doi/10.5555/2969239.2969296

[9] Zhang, J., Shi, X., King, I., Yeung, D.Y.: Dynamic key-value memory networks for knowledge tracing. In: Proceedings of the 26th international conference on World Wide Web. pp. 765–774. International World Wide Web Conferences Steering Committee (2017) https://doi.org/10.1145/3038912.3052580

[10] Feng, M., Heffernan, N., Koedinger, K.: Addressing the assessment challenge with an online system that tutors as it assesses. User Modeling and User-Adapted Interaction 19(3), 243–266 (2009) https://doi.org/10.1007/s11257-009-9063-7

[11] Pardos, Z.A., Baker, R.S., San Pedro, M.O., Gowda, S.M., Gowda, S.M.: Affective states and state tests: Investigating how affect and engagement during the school year predict end-of-year learning outcomes. Journal of Learning Analytics 1(1),107–128 (2014) https://doi.org/10.18608/jla.2014.11.6

[12] Youngduck Choi, Youngnam Lee, Dongmin Shin, Junghyun Cho, Seoyon Park Seewoo Lee, Jineon Baek, Chan Bae, Byungsoo Kim, Jaewe Heo In International Conference on Artificial Intelligence in Education AIED 2020:

Artificial Intelligence in Education pp 69-73 https://link.springer.com/chapter/10.1007/978-3-030-52240- 7_13

[13] Chang, H.S., Hsu, H.J., Chen, K.T.: Modeling exercise relationships in e-learning: A unified approach. In: EDM. pp. 532–535 (2015) https://pslcdatashop.web.cmu.edu/DownloadPaper?fileName=edm_submission _sho rt.pdf&fileId=7868&datasetId=1275

[14] Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L.J., SohlDickstein, J.: Deep knowledge tracing. In: Advances in neural information processing systems. pp. 505–513 (2015) https://dl.acm.org/doi/10.5555/2969239.2969296

[15] D.G.F.M. Bollen, B.P. Knijnenburg, M.C. Willemsen, M.P. Graus: Understanding choice overload in recommender systems: Human-Technology Interaction: 2010: 10.1145/1864708.1864724

[16] Huang, Chen, & Chen: Promoting in-depth reading experience and acceptance: Design and assessment of Tablet reading interfaces: June 2014: DOI:10.1080/0144929X.2012.759625

[17] Kalyuga, S. (2011). Cognitive load theory: How many types of load does it really need? Educational Psychology Review, 23(1), 1–19. https://doi.org/10.1007/s10648-010-9150-7

[18] Kiratijuta Bhumichitr, S. Channarukul, July 2017, 14th International Joint Conference on Computer Science and Software Engineering (JCSSE): Recommender Systems for university elective course recommendation, DOI:10.1109/JCSSE.2017.8025933

[19] Huang, Zhan, Zhang, & Yang, 2017, Education Quarterly Reviews, DOI: 10.31014/aior.1993.03.01.119

[20] Apaza, Cervantes, Quispe, & Luna, 2014, Online Courses Recommendation based on LDA,

[21] Farzan & Brusilovsky, 2006, Social Navigation Support in a Course Recommendation System, DOI:10.1007/11768012_11

[22] Bendakir & Aımeur, 2006, Using Association Rules for Course Recommendation,

[23] Huang et al., 2013, Applying Learning Analytics for the Early Prediction of Students' Academic Performance in Blended Learning

[24] Ibrahim, Yang, & Ndzi, 2019, Ontology-Based Personalized Course Recommendation Framework, DOI:10.1109/ACCESS.2018.2889635

[25] Z. Zhang, Zhou, & Zhang, 2010, Information overload in the information age: a review of the literature from business administration, business psychology, and related disciplines with a bibliometric approach and framework development

[26] Garcia, Sebastia, & Onaindia, 2011, On the design of individual and group recommender systems for tourism, DOI:10.1016/j.eswa.2010.12.143

[27] Jannach, Zanker, Felfering, & Friedrich, 2011 , recommender Systems – An Introduction

[28] Alimam & Seghiouer, 2013, Recommendation of personalized RSS feeds based on ontology approach and multi-agent system in web 2.0

[29] Bach & Dieng-Kuntz, 2005, Conceptual Graphs for Semantic Web Applications, DOI: 10.1007/11524564_2
[30] Sandvig & Burke, 2005, Defending recommender systems: detection of profile injection attacks
[31] Jannach et al., 2011, Recommender Systems: An Introduction, DOI:10.1017/CBO9780511763113
[32] Ren, Zhang, Cui, Deng, & Shi, 2015, PERSONALIZED FINANCIAL NEWS RECOMMENDATION ALGORITHM BASED ON ONTOLOGY
[33] H. Zhang, Yang, Huang, & Zhan, 2017, Deep Learning Based Recommender System: A Survey and New Perspectives, DOI:10.1145/3285029
[34] Carballo, 2014, Masters' Courses Recommendation: Exploring Collaborative Filtering and Singular Value Decomposition with Student Profiling
[35] Lotfy & Salama, 2014, Review of Recommender Systems Algorithms Utilized in Social Networks based e-Learning Systems & Neutrosophic System

[36] Martinez & Lhadj, 2013, Educational Recommender Systems: A Pedagogical-Focused Perspective, DOI:10.1007/978-3-319-00375-7_8
[37] Lops, de Gemmis, & Semeraro, 2011, Content-based Recommender Systems: State of the Art and Trends, DOI:10.1007/978-0-387-85820-3_3
[38] Kim, 2013, A Collaborative Filtering Based Personalized TOP-K Recommender System for Housing
[39] Ruotsalo & Hyvönen, 2007, Methods and applications for ontology-based recommender systems
[40] Adomavicius & Tuzhilin, 2005, A Well-Built Hybrid Recommender System for Agricultural Products in Benue State of Nigeria
[41] Burke, 2002, Hybrid Recommender Systems: Survey and Experiments
[42] Kaminskas & Bridge, 2014, Measuring Surprise in Recommender Systems
[43] Adibi & Ladani, 2013, A collaborative filtering recommender system based on user's time pattern activity, DOI: 10.1109/IKT.2013.6620074
[44] Cui & Chen, 2009, An Online Book Recommendation System Based on Web Service, DOI:10.1109/FSKD.2009.328
[45] Herlocker, Konstan, & Riedl, 2000, Recommender systems: from algorithms to user experience
[46] Linden et al., 2003, Amazon.com recommendations: item-to-item collaborative filtering
[47] Ray & Sharma, 2016, Explanations in recommender systems: an overview, DOI:10.1504/IJBIS.2016.10000276
[48] Ren et al., 2015, Recommender Systems: Introduction and Challenges, DOI:10.1007/978-1-4899-7637-6_1
[49] Bagherifard, Rahmani, Nilashi, & Rafe, 2017, Performance Improvement for Recommender Systems Using Ontology, DOI:10.1016/j.tele.2017.08.008
[50] Zhao & Shang, 2010, User-Based Collaborative-Filtering Recommendation Algorithms on Hadoop, DOI:10.1109/WKDD.2010.54
[51] Desrosiers & Karypis, 2011, A Comprehensive Survey of Neighborhood-Based Recommendation Methods, DOI:10.1007/978-0-387-85820-3_4

[52] Chang, Lin, & Chen, 2016, A review on Recommender Systems for course selection in higher education, DOI 10.1088/1757-899X/1098/3/032039
[53] Kunal Shah, 2017, Recommender systems: An overview of different approaches to recommendations, DOI:10.1109/ICIIECS.2017.8276172
[54] Salakhutdinov, Mnih, & Hinton, 2007, Restricted Boltzmann Machines for Collaborative Filtering
[55] Koren, Bell, & Volinsky, 2009, Matrix Factorization Techniques for Recommender System DOI: 10.1109/MC.2009.263s,
[56] Schein, Popescul, Ungar, & Pennock, 2002, CROC: A New Evaluation Criterion for Recommender Systems
[57] Pazzani & Billsus, 2007, Recommender Systems Book
[58] Lops et al., 2011, Content-based Recommender Systems: State of the Art and Trends, DOI: 10.1007/978-0-387-85820-3_3
[59] Ruthven & Lalmas, 2003, Recommender System Performance Evaluation and Prediction: An Information Retrieval Perspective
[60] Han, Kamber, & Pei, 2012 , Data Mining Book
[61] Zezula, Dohnal, & Amato, 2006, Recommender Systems An Introduction Book
[62] Balabanović & Shoham, 1997, Content-Based, Collaborative Recommendation, DOI:10.1145/245108.245124
[63] Maidel, Shoval, Shapira, & Taieb-Maimon, 2010, A Recommendation System for Browsing of Multimedia Collections in the Internet of Things, DOI: 10.1007/978-3-642-34952-2_16
[64] Cantador et al., 2008; Kumar, 2012, Cross-domain recommender systems: A survey of the State of the Art
[65] Ostuni, Di Noia, Mirizzi, & Di Sciascio, 2014, A Linked Data Recommender System Using a Neighborhood-Based Graph Kernel, DOI: 10.1007/978-3-319-10491-1_10
[66] Mooney & Roy, 1999, Content-based book recommending using learning for text categorization, DOI:10.1145/336597.336662
[67] Pazzani & Billsus, 1997, Learning and Revising User Profiles: The Identification of Interesting Web Sites. Machine Learning
[68] Burke, 2002; Ekstrand, Harper, Willemsen, & Konstan, 2014, Letting Users Choose Recommender Algorithms: An Experimental Study, doi.org/10.1145/2792838.2800195
[69] Li, Ogihara, & Li, 2009, Music Recommendation Based on Acoustic Features and User Access Patterns, DOI:10.1109/TASL.2009.2020893
[70] Markovitch & Markovitch, 2006, Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis
[71] Musto, 2015, Semantics-Aware Content-Based Recommender Systems, DOI:10.1007/978-1-4899-7637-6_4
[72] Maes & Sharadanand, 1995, Social Information Filtering: Algorithms for Automating "Word of Mouth"
[73] Billsus & Pazzani, 2000, Learning collaborative information filters
[74] Zhang, Callan, & Minka, 2002, Interaction and Personalization of Criteria in Recommender Systems, DOI: 10.1007/978-3-642-13470-8_18

[75] Donghui Yang, 2016, A social recommender system by combining social network and sentiment similarity: A case study of healthcare, doi.org/10.1177/0165551516657712
[76] Middleton, De Roure, & Shadbolt, Ontological User Profiling in Recommender Systems, 10.1145/963770.963773
[77] Obeid, Lahoud, El Khoury, & Champin, 2018, Ontology-based Recommender System in Higher Education
[78] Yang, Sun, Wang, & Jin, 2010, Semantic Web-Based Personalized Recommendation System of Courses Knowledge Research, DOI:10.1109/ICICCI.2010.54
[79] Zhou, Yang, & Zha, 2011, Research Problems in Recommender systems, 10.1088/1742-6596/1717/1/012002
[80] Ambikapathy, 2011 , ICEL 2018 13th International Conference on e-Learning
[81] Adomavicius & Tuzhilin, 2005b; Sieg, Mobasher, & Burke, 2010, Context-Aware Recommender Systems, DOI: 10.1007/978-0-387-85820-3_7
[82] Claypool et al., 1999, Combining content-based and collaborative filters in an online newspaper. In: Proceedings of the SIGIR-99 workshop on recommender systems: algorithms and evaluation, Berkeley
[83] Pazzani, 1999, Learning collaborative information filters. In: Proceedings of the fifteenth international conference on machine learning (ICML-98), Madison. Morgan Kaufmann, San Francisco, pp 46–54
[84] Billsus & Pazzani, 2000, Learning collaborative information filters. In: Proceedings of the fifteenth international conference on machine learning (ICML-98), Madison. Morgan Kaufmann, San Francisco, pp 46–54
[85] Smyth & Cotter, 2000, PTV: Intelligent, personalized TV guides. In Twelfth conference on innovative applications of artificial intelligence, Austin, Texas (pp. 957–964)
[86] Sarwar, Karypis, Konstan, & Riedl, 2000, Combining collaborative filtering with personal agents for better recommendations. In Proceedings of the sixteenth national conference on artificial intelligence (AAAI-99), Orlando, Florida (pp. 439–446).
[87] Manouselis, Vuorikari, & Assche, 2010, Recommender Systems for Learning
[88] Drachsler, Hummel, & Koper , Recommender Systems in Technology Enhanced Learning , DOI:10.1007/978-0-387-85820-3_12
[89] Park, 2017, Deep hybrid recommender systems via exploiting document context and statistics of items, doi.org/10.1016/j.ins.2017.06.026
[90] Ibrahim et al., 2018, A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques, doi.org/10.1016/j.eswa.2017.09.058
[91] Bozo, Alarcón, & Iribarra, 2010, Recommending Learning Objects According to a Teachers' Contex Model, DOI: 10.1007/978-3-642-16020-2_39
[92] J. Hu & Zhang, 2008, Research on entropy-based collaborative filtering algorithm and personalized recommendation in e-commerce
[93] Santos & Boticario, 2009, Requirements for Semantic Educational Recommender Systems in Formal E-Learning Scenarios, doi.org/10.3390/a4030131

[94] T. Tang & Mccalla, 2004, Utilizing Artificial Learners to Help Overcome the Cold-Start Problem in a Pedagogically-Oriented Paper Recommendation System, DOI: 10.1007/978-3-540-27780-4_28

[95] Q. Yang et al., 2010, Semantic Web-Based Personalized Recommendation System of Courses Knowledge Research, DOI: 10.1109/ICICCI.2010.54

[96] T. Y. Tang & McCalla, 2009b, Recommender Systems in Technology Enhanced Learning

[97] Abel et al., 2008, A Rule-Based Recommender System for Online Discussion Forums, DOI: 10.1007/978-3-540-70987-9_4

[98] Gasparini, Lichtnow, Pimenta, & Oliveira, 2009, Quality Ontology for Recommendation in an Adaptive Educational System, DOI:10.1109/INCOS.2009.11

[99] Huang, 2017, Deep Matrix Factorization Models for Recommender Systems

[100] Bobadilla, Ortega, Hernando, & Gutiérrez, 2013, Recommender systems survey, doi.org/10.1016/j.knosys.2013.03.012

[101] Noakes et al., 1968, 5 Ontogeny of Behavior and Concurrent Developmental Changes in Sensory Systems in Teleost Fishes

[102] Ge, Chen, Peng, & Li, 2012, An ontology-based method for personalized recommendation, DOI: 10.1109/ICCI-CC.2012.6311204

[103] Al-Badarenah & Alsakran, 2016, An Automated Recommender System for Course Selection, DOI:10.14569/IJACSA.2016.070323

[104] Obeid et al., 2018, Ontology-based Recommender System in Higher Education, doi.org/10.1145/3184558.3191533

[105] Shrivastav & Hiltz, 2013, Information Overload in Technology-based Education: a Meta-Analysis

[106] Sieg et al., 2010, Ontological Approach in Knowledge-Based Recommender System to Develop the Quality of E-learning System

[107] Carballo, 2014; Ray & Sharma, 2011, An Ontology-based Hybrid Approach to Course Recommendation in Higher Education

[108] Chen, Lee, & Chen, 2005, Link prediction approach to collaborative filtering, DOI: 10.1145/1065385.1065415

[109] Salahli, Özdemir, & Yaşar, 2013, Concept-Based Approach for Adaptive Personalized Course Learning System, DOI:10.5539/ies.v6n5p92

[110] Punj & Moore, 2007, Smart versus knowledgeable online recommendation agents, doi.org/10.1002/dir.20089

[111] Sarwar et al., 2001, Sparsity, scalability, and distribution in recommender systems

[112] Chen He, 2011, Interactive recommender systems: A survey of the state of the art and future research challenges and opportunities, doi.org/10.1016/j.eswa.2016.02.013

[113] Mahony & Smyth, 2007, A Recommender System for On-line Course Enrolment: An Initial Study

[114] Zhang, 2009, CARES: a ranking-oriented CADAL recommender system, doi.org/10.1145/1555400.1555432

[115] Liu, Wang, Liu, & Yang, 2010, A survey of matrix completion methods for recommendation systems, 10.26599/BDMA.2018.9020008

[116] Wang & Sapporo, 2006, Language Models of Collaborative Filtering, DOI: 10.1007/978-3-642-04769-5_19

[117] Hongji Yang, Zhan Cui, & Brien, 1999, Extracting ontologies from legacy systems for understanding and re-engineering, DOI:10.1109/CMPSAC.1999.812512

[118] Wang, 2008, Website browsing aid: A navigation graph-based recommendation system, doi.org/10.1016/j.dss.2007.05.006

[119] Nikos Manouselis, Drachsler, Verbert, Santos, & Konstan, 2014, Recommender systems for technology enhanced learning: Research trends and applications, 10.1007/978-1-4939-0530-0

[120] Rafaeli, Dan-Gur, & Barak, 2005, Social Recommender Systems: Recommendations in Support of E-Learning.

[121] DeLong, Desikan, & Srivastava, 2006, USER: User-Sensitive Expert Recommendations for Knowledge-Dense Environments, DOI: 10.1007/11891321_5

[122] Yang, Huang, Tsai, Chung, & Wu, 2009, An Automatic Multimedia Content Summarization System for Video Recommendation,

[123] Schulz et al., 2001, An Architecture for Behavior-Based Library Recommender Systems