**COURSE RECOMMENDATION SYSTEM**

Contents

[**I.** **Abstract** 3](#_Toc108702373)

[**II.** **Introduction** 4](#_Toc108702374)

[**III.** **Related work** 6](#_Toc108702375)

[**IV.** **Methodology** 6](#_Toc108702376)

[I. Data Collection and Cleaning 6](#_Toc108702377)

[II. Exploratory data analysis (EDA) 8](#_Toc108702378)

[III. Natural Language processing 9](#_Toc108702379)

[IV. Clustering 10](#_Toc108702380)

[**V.** **Recommendation platform** 19](#_Toc108702381)

[i. Use Case 1 for recommender application: 21](#_Toc108702382)

[ii. Use Case 2: 24](#_Toc108702383)

[iii. Use Case 3: 27](#_Toc108702384)

[iv. Use Case 4: 28](#_Toc108702385)

[**VI.** **Business model:** 32](#_Toc108702386)

[v. Introduction: 32](#_Toc108702387)

[vi. Problem: 32](#_Toc108702388)

[vii. Objective: 33](#_Toc108702389)

[viii. Vision: 33](#_Toc108702390)

[ix. Future work: 33](#_Toc108702391)

[**VII.** **Result** 34](#_Toc108702392)

[**VIII.** **Conclusion** 36](#_Toc108702393)

[**IX.** **Evaluating with previous work, techniques comparison** 36](#_Toc108702394)

[**X.** **Reference** 38](#_Toc108702395)

# **Abstract**

E-learning Recommender systems were developed to assist learners in navigating a maze of online learning content. This thesis presents a course recommendation approach based on a content filtering technique. In the proposed approach, each user has taken some courses. Each course has a description and objective attributes. The natural language processing NLP technique is used to find similar courses using the description attribute and subdivide them into clusters using k-means. Each delivered cluster contains courses with similar content. We have tried Clustering criteria on both attributes’ objectives and descriptions. For better results, the Silhouette metric had been used to evaluate clustering. Based on that the description attribute will give more accurate results. In the experiment, the inertia function uses the elbow method to optimize number of K clusters. Therefore, the optimal number of k was eight clusters. Finally, we have recommended courses for users using the cosine similarity function, which found courses similar to those users took. for providing a user-friendly environment in which to view the interface and display the abstract of this thesis, there was an integrated application with a machine learning model. In the application, the user could search add and delete courses.

# **Introduction**

Nowadays, an eLearning Recommendation System is widely used to improve learning. eLearning allows us to learn almost anywhere (at home, colleges, schools, universities, while traveling, in the garden), at any time, and on any device (desktop, laptops, iPad, smartphones, etc.) [1]. The Internet and digital learning through eLearning have helped to improve the education system [1]. The Learning Management System is the foundation of eLearning (LMS). Many colleges and institutes began using LMS systems to meet the needs of various departments. Individuals (single users), multi-users, similar users, and hybrid users have all benefited from eLearning. The recommendation system used in several disciplines is an essential tool in online learning environments because it promotes distance learning and allows individual learners to manage their time while focusing on the learning process [2]. E-learning Recommender systems were created to help learners explore many online learning contents [3]. The abundance of information available on the Web and in Digital Libraries, combined with their dynamic and heterogeneous nature, has resulted in constantly rising difficulties in getting what we want when we need it, and in the most efficient manner possible. Researchers used various recommendation techniques to help learners overcome information overload by filtering out irrelevant learning resources and providing them with more personalized content. Individual needs, objectives, and preferences influence learners' learning processes [4]. Different learners have different conceptual understanding, history, proficiency levels, learning preferences, and learning activities [2]. As a result, it is difficult to recommend learning resources to each learner based on their preferences [4]. One of the most challenging aspects of such systems is that user interests, preferences, and needs change over time [4]. There are three main approaches for developing recommendation systems which are collaborative filtering (CF), content-based filtering (CBF), and Hybrid recommender systems [7].

Therefore, many types of research introduce course recommendation engines, their methods, and how they work [4]. One popular technique for recommendation or recommender systems is content-based filtering. Content-based refers to the content or attributes of the items you like [4]. In the content-based approach, the RS recommends items that are similar in terms of content features to those that the target user previously liked [5]. The underlying principle of this approach is based on calculating the similarity of the item features associated with the compared items [5].

At this stage, courses are chosen by detecting similarities between items in the existing course (the active course in which the learner is enrolled) and items in other courses. These course items include the course title, course description, and course reviews [4]. This algorithm is very simple because it needs to compare two lists of items and see how similar they are [8]. In this research case, they have user interests in categories and courses with their categories [9]. For example, the web development course has categories such as informatics, software engineering, and programming, and if the user has informatics, software engineering, and programming in his profile, the Web development course will be recommended [9]. Moreover, A collaborative filtering system suggests items that other users like. As a result, the exploration of new items is ensured by the consideration of other users with similar profiles. User profiles can be created explicitly by asking users about their hobbies and interests, or implicitly by a user's rating [8]. However, the Hybrid recommender systems combine the advantages of the two previous methods [10]. In our research, we are going to use a content-based filtering technique for a course recommendation system. With the help of the Clustering technique in machine learning, we could group data into a finite number of groups based on data similarity [11]. Unsupervised learning techniques are used to classify unlabeled historical data in these methods. Members of the classified groups are similar to one another and dissimilar to members of other groups [11]. While using different algorithms for implementing recommender systems, such as clustering and association rules, better results are achieved [12].

## **Related work**

Many types of research introduce course recommendation engines, their methods, and how they work [1]. One popular technique for recommendation or recommender systems is content-based filtering. Content-based refers to the content or attributes of the items you like [1]. In the content-based approach, the RS recommends items that are similar in terms of content features to those that the target user previously liked [2]. The underlying principle of this approach is based on calculating the similarity of the item features associated with the compared items [2].

At this stage, courses are chosen by detecting similarities between items in the existing course (the active course in which the learner is enrolled) and items in other courses. These course items include the course title, course description, and course reviews [4].

Researchers have compared user profile categories and course categories to determine which categories were shared [4].

This algorithm is very simple because it needs to compare two lists of items and see how similar they are [4]. In this research case, they have user interests in categories and courses with their categories [5]. For example, the web development course has categories such as informatics, software engineering, and programming, and if the user has informatics, software engineering, and programming in his profile, the Web development course will be recommended [5]. In this research, they test the content-based approach on a dataset containing 575 users and 30 courses where learners and courses are divided into categories, they can get the most popular categories that users like by using these categories [5]. They calculate the results in this experiment they extract the common categories of each user from each category and calculate the best probability [5]. The experiment yielded a 75% success rate [5]. It demonstrates that we can find an interesting course for each user [5]. The experiment yielded a 75% success rate when using content-based [5]. Researchers proposed a semantic recommendation algorithm to provide learners with personalized and relevant e-learning content The proposed algorithm takes advantage of extra semantic and intra-semantic relationships between learner requirements and Learning objectives [6]. Using semantic indexing services, the Learning objective is classified based on the concepts that each Learning objective represents [6]. The shortest path algorithm is used to compute the similarity between Learning objective concepts [5]. The most representative concepts are chosen based on the degree of similarity between them [6]. Later, the appropriate Learning objective is recommended to the learner based on the extension of query keywords using semantic relations, based on the learner's needs (expressed in the form of a request query) [6].

In different recommendation phases, various machine learning algorithms and data mining techniques are used for data pre-processing, model learning/training, and validation [3].

The goal of recommendation techniques is to improve the accuracy and performance of target recommendations based on the main objective [3]. These techniques suggest learning content to a learner by analyzing the preferences of other learners in the neighborhood based on their similarities [7]. As a result, similarity computation, like any other recommender system, is an essential component of the e-learning content Recommendation System. A similarity measure, also known as a similarity coefficient, assesses the similarity of two data objects. Pearson Correlation Coefficient (PCC) is the most commonly used similarity measure in most articles [7]. Constrained Pearson correlation employs the median value rather than the average of ratings co-rated by both users; PCC produces the best results when data is normally distributed [7]. The median value on a scale of 1 to 5 is 3. In terms of the user-based algorithm [7]. Research has calculated the similarity between users over a proposed rating matrix to provide better recommendations for users [8]. As a result, it is critical to compare the performance of the recommendation system using various similarity measures relevant to the specific research [8]. The challenge of using a content-based approach is determining user preferences based on the contents of items [10].

Many approaches to solving this problem have been developed in the fields of data mining and machine learning. To recommend some courses to a specific learner [11], for example, a recommender system first obtains all of the courses this user has already taken and then analyses their contents.

Text mining methods, such as the well-known TF-IDF [9], can be used to extract keywords from the text of the course description. A course description can be represented by a multi-dimensional vector after integrating all of the keywords and their weights. Specific clustering algorithms can be used to locate the centers of these vectors, which represent the user’s interests [10]. Researchers propose a machine learning approach for recommending appropriate courses to learners based on their prior learning experiences and performance [12]. The framework uses the k-means clustering algorithm to classify new learners based on their previous performance [13]. The data set was clustered using a clustering algorithm to create a group of similar learners [12]. The mining algorithm was applied to each cluster after the learners had created a common pattern [12]. The system assigns students to groups based on historical data, such as the backgrounds of students who received higher grades in each course [12].

Every time a new student enters the system, the clusters will be used to classify them, and a set of courses will be suggested to them based on frequent pattern mining [12]. Finally, they used a heatmap to visualize the actual score [12].

In this study, they created a course recommender model that considers the students’ characteristics when recommending appropriate courses [13]. Clustering is used in the model to find students who have similar interests and skills [13]. Following the discovery of similar students, fuzzy association rule mining is used to investigate the dependencies between student course selections [13]. Clustering and fuzzy association rules are used to generate appropriate recommendations and a predicted score [13]. The clustering technique was used to identify similar groups of students in this study [13]. This method can be used to find people with similar preferences, skills, and behaviors [13].

# **Methodology**

This study focuses on building a course recommendation engine using the Udemy data set. This method relies on a content-based technique.

## Data Collection and Cleaning

Data collection is the initiative in any scientific research. We have collected the data from the Udemy e-learning platform using APIs. Udemy is from the websites which have their dedicated API. This dedicated API is provided information to anyone interested. We as developers have used the API to fetch the data which we need, as a file to store or to feed the information into our software once we have understood how it works. As long as Udemy’s API is accessible, we got its data.

Each API required us as developers to fully understand and integrate it with our software. Because not all APIs work the same way, using them requires some time and coding knowledge. Data extraction rates may be limited by the API. Some websites may limit the number of requests that can be sent in a given period to avoid overloading the host server. As a result, gathering all of the data has taken a long time.

[We have used Udemy API using the requests module in python](https://stackoverflow.com/questions/65550067/how-to-use-udemy-api-using-requests-module-in-python), we have sent authenticated requests, and provided the ClientID and client secret values as an encoded HTTP Authorization header. We have an import **request ()** library, This Python requests module includes several built-in methods for making HTTP requests to a given URI via GET, POST, PUT, PATCH, or HEAD requests. An HTTP request is intended to either retrieve data from a given URI or push data to a server. It is a request-response protocol that is used between a client and a server.  
In our project, the web browser is an application on a computer that hosts a website. For making a GET request to a specified URI, we have used Python's requests module includes a method called **get ().** We have used the GET method to access information from the indicated server using a URI. The GET method uploads user data that has been encoded to the page request.

We've saved the leads to a data frame and exported them to a CSV file. For these courses, we've downloaded the available reviews. The number of obtainable reviews for a course is 10.000.

Then we move to the following step: data cleaning, which is one of the foremost important steps for preparing data.

1-Tokenization: the first step was to divide the text into tokens. A token can be a symbol, a word, a sentence, or a phrase. We used whitespace as a delimiter to divide our dataset into tokens.

2. Filtering: The tokenized result was filtered to remove meaningless words. The minimum length was used to filter the results. Tokens with less than three characters in length were removed.

3. Stemming: a crucial step in text mining in which words are reduced to their root forms.

4. Cases Transformation: All of the words were finally converted to lowercase.

Throughout the cleaning process, we first transform relevant columns into a list/dictionary of values using **ast.literal\_eval** method in the **transform\_col** method. this method is one of the helper functions used to traverse an abstract syntax tree. This function evaluates an expression node or a string containing a Python literal or container display.  The ast.literal\_eval method can safely evaluate strings containing Python values from unknown sources without requiring us to parse the values. Complex expressions with indexing or operators.

Moreover, we have transformed primary categories and primary subcategories into titles, we transform price and some content information as rating and an average rating of each course given by the user into float. Also, we have transformed published\_time and published\_since\_month in DateTime.

We have an import re (regular expression) library which helps us to remove the HTML tags from the text in Udemy’s data and will return a normal string.

Then we dropped the duplicates within the data. We have filtered the data and kept relevant columns.

Also, we checked the free courses which are labeled free, and change the price to zero. We dropped the missing values. Within the last step, we've got saved the cleaned data in a CSV file.

## Exploratory data analysis (EDA)

Exploratory data analysis EDA investigates the data and summarizes its main characteristic. First, we visualize the average rating for all courses, and the average distribution rating was between 4 and 5 for most of the courses, as shown in Fig. 1.

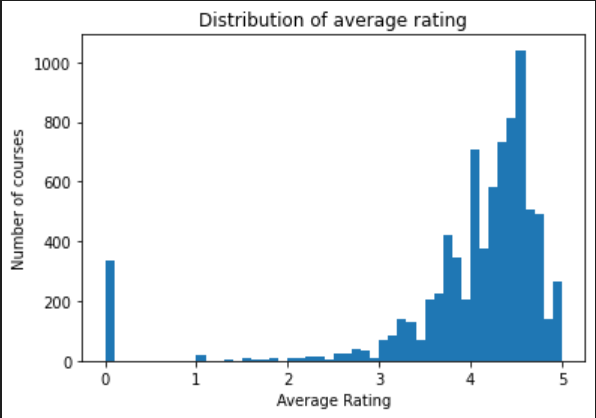


Figure 1: the average rating for all courses

Moreover, we have checked courses with the greatest number of subscribers we print the top 10 courses; we found that the course java-tutorial has 1662635 subscribers, as shown in Fig. 2.

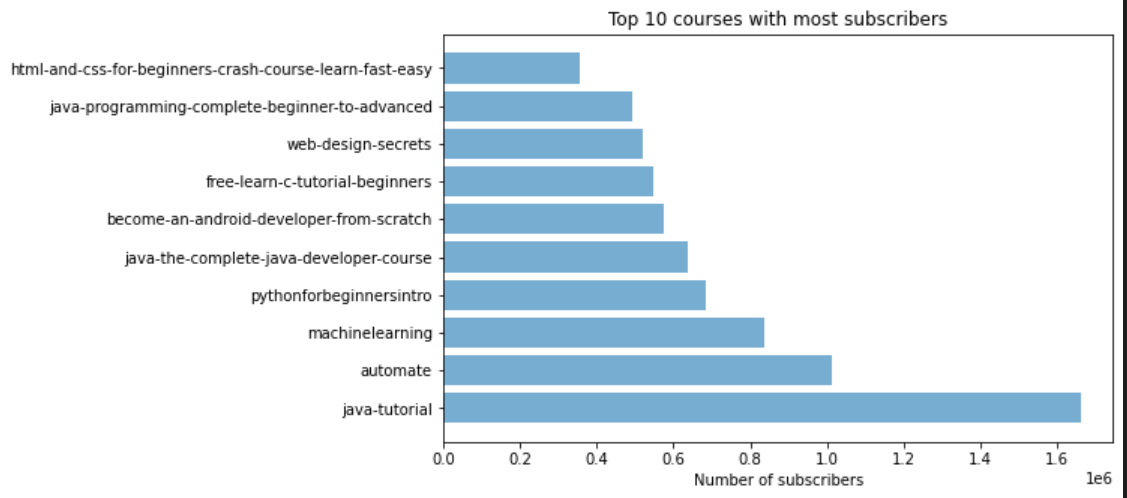


Figure 2: the course java-tutorial has 1662635 subscribers

Also, the number of subscribers and the number of reviews have a positive correlation, as shown in Fig. 3.

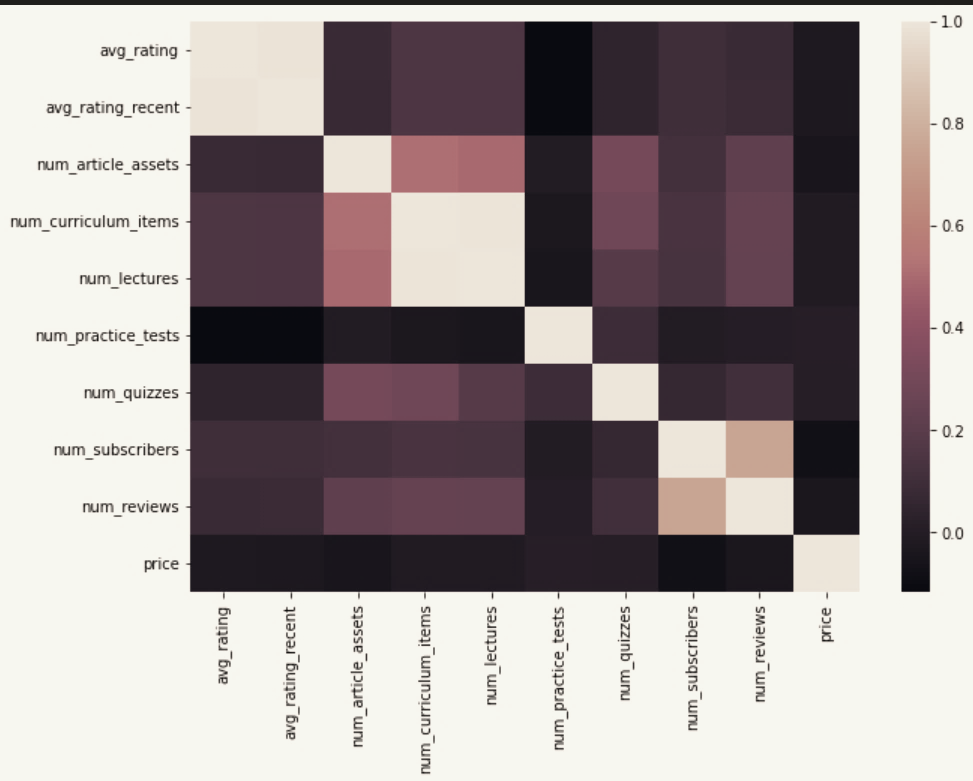


Figure 3: the correlation between the number of subscribers and the number of reviews

## Natural Language processing

In this stage, we were preparing data which are OBJECTIVE and DESCRIPTION attributes for clustering. This data frame would be used for reversing the stemming of the words which is a list. We first create a string from the list items using the function combine-list. We have saved all words with their stemmed consistency within the data frame for stemming. We've removed all duplicates from this data frame because we are only inquisitive about stemmed words (e.g., we treated the words 'analyze' and 'analyzing' as the same words). We've identified StopWords, which contains all of the expressions that haven't been considered within the texts.

Finally, the TF-IDF Vectorizer was applied to the objectives and description attributes: This transformation creates feature vectors from text documents to help select words that are common within the text but uncommon within the corpus.

## Clustering

In this part, we cluster the courses and build a recommender system that depends on the new clusters of description attributes and other course features in the data shown in the following figure.

Clustering methods are unsupervised algorithms that aid in the summarization of information from large amounts of text data by forming various clusters. This method is useful for understanding what your dataset is all about and how you can divide the context of the text in the dataset into different categories.

K-means Clustering is a type of clustering that is used when we do not have labeled data, as we do in our case (means, without defined categories or groups). The goal of this algorithm is to find groups in the data, and the variable K represents the number of groups found. The data were clustered based on high similarity points clustering together and low similarity points clustering separately.

KMeans normally only works with numbers: we need numbers. To obtain numbers, we perform a common procedure known as feature extraction.

The feature we'll be using is TF-IDF, which is a numerical statistic. The term frequency and inverse document frequency are used in this statistic. In a brief, we use statistics to find numerical features. The most frequently occurring words are assigned to feature indices. Yielding a word occurrence frequency (sparse) matrix. The word frequencies are then reweighted using the Inverse Document Frequency (**IDF**) vector, which was collected feature-by-feature across the corpus.

Term frequency **(TF)**is identified as the total number of instances a word appears in a document (i) divided by the total number of words in the document (j).

The inverse document frequency **(IDF)** is the log of the overall number of documents divided by the total number of documents that consist of the word. The logarithm is included to reduce the significance of a very high IDF value.

**TFIDF** is computed by multiplying the term frequency by the inverse document frequency.

The attributes OBJECTIVES and DESCRIPTION were accustomed to clusters the information. Following the creation of those two attributes, the primary section of the system attempts to cluster the courses using the attribute OBJECTIVES. In contrast, the second section uses the course description to construct the clusters.

we started clustering the description attribute. We use the k-means clustering algorithm, we set k=8. Each cluster contains the highest ten commonest words, as shown in Fig. 5.

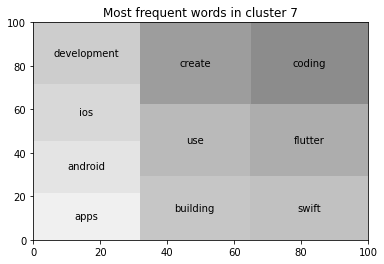


Figure 4: Each cluster contains the highest ten commonest words

We have used inertia, a metric that assesses how well a dataset was clustered using K-Means. It is calculated by calculating the distance between each data point and its centroid, squaring that distance, and summing the squares across one cluster. A good model has both low inertia and a small number of clusters (K). However, this is a tradeoff because as K increases, so does inertia. Use the Elbow method to find the optimal K for a clustering description attribute.

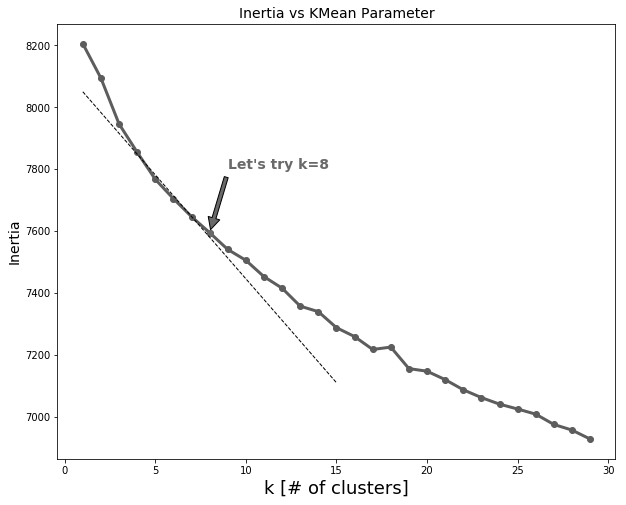


Figure 5: Using the elbow method to achieve the best value of k =8 in K-means

find the point where the decrease in inertia begins to slow. This graph's "elbow" is K=8. We have used the relationship between the number of clusters and the inertia (within-cluster sum-of-squares). According to the elbow method, the line is an arm, and the "elbow" on the arm is the best value of k, as shown in Fig. 5.

We have used the silhouette coefficient metric to measure the accuracy of clustering of description attribute when the number of k=8 and it was **Silhouette Coefficient.**

After that we plot the distribution of the cluster with the top common 6 words and it shows that cluster 1 contains the largest number of courses as shown in the below figure.



Figure 6: visualization top common 8 words

Moreover, we have tried the same steps on objective attributes to decide which attribute we are going to build a recommendation system based on. The number of K=6 depends on the inertia metric using the elbow method as shown in the following fig 8.

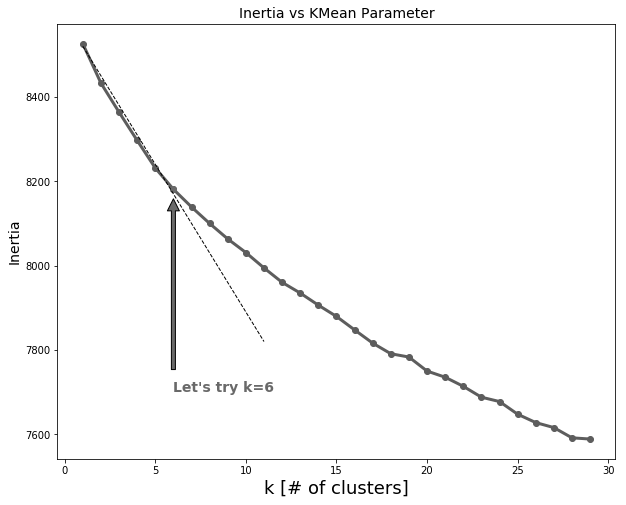


Figure 8: inertia with optimal number of k =6.

We have used the silhouette coefficient metric to measure the accuracy of clustering of Objective attribute when the number of k=6. We have evaluated clustering of objective by using silhouette coefficient **Silhouette Coefficient.**

Then we print the 10 most common frequent words in clusters of objective attributes.

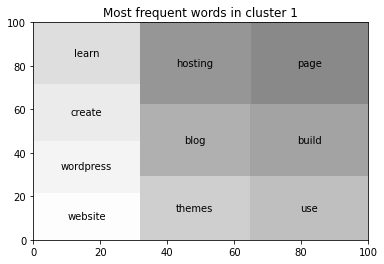


Figure 9: top most frequent words in objective clustering in cluster 1.

We plot the distribution of objective clusters with top 5 words, it shows that cluster 2 contains most of the courses.

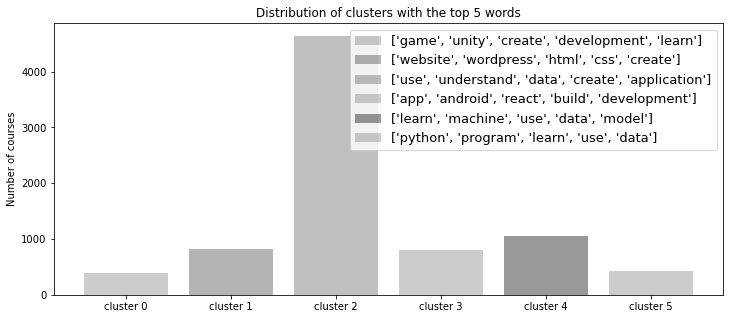


Figure 10: distribution of 6 objective clusters with top 5 words.

we have printed the titles of courses in each cluster from our 6-objective cluster. Then, we evaluated the course descriptions as the criterion of the clustering algorithms after building clusters with the objectives attribute. We have tried to run the same analyses using the description feature and got better-distributed clusters.

Moreover, for evaluation, we have used silhouette metric to evaluate clustering of objective and description attribute, we have found that clustering of description metric will give better results.

Finally, for building the recommender engine, we have used the course features in combination with the k-means clustering result with k=8. We have exported the clustering algorithm in the sav file.

SAV files are data files created by the Statistical Package for the Social Sciences (SPSS). We used this file format because it is categorized into file header and variable descriptor records which we have in our data.

We have normalized features using the StandardScaler () function, which normalizes each feature individually. This function standardizes the data into standard form. To transform and standardize the data, we use fit\_transform () in conjunction with the assigned object. This function basically implies fit first the transform function but done more efficiently when passed together as fit\_transform. The fit function learns the vocabulary and IDF from the dataset and the transform function transforms the document to the document-term matrix.

Therefore, the feature matrix had been normalized as the final step of the preparation because the features are scaled differently Then we saved the normalized data in a data frame which is df\_norm.

Our data is represented by many of attributes where the recommender system does its job depending on them, each of these is recording the frequency of a specific string or phrase in the record. As a result, each column is an object represented by a term-frequency vector. To find the similarity of two vectors in an inner product space is measured by cosine similarity. It is calculated by taking the cosine of the angle between two vectors and determining whether two vectors are pointing in the same general direction. It is frequently used in text analysis to assess document similarity.

**Cosine similarity equation**

**X2**

**X1**

Cosine similarity the angle theta between the two courses determines their similarity. The theta value ranges from 0 to 1. If the theta value is close to one, it is most similar, and if it is close to zero, it is least similar. The course will be recommended if it is close to 1, otherwise, there will be no similarity between them. It will recommend the best courses for the user based on the Cosine similarity. Following the cosine similarity, we used a normalized popular score to find the average cosine similarity.

To calculate Cosine similarity, is done by multiplying the dot product of two vectors. we multiply the vector which is the column of information of course that users take by the records of information of all courses. If the user had taken more than one course, we multiply each vector (column) individually by the matrix of all courses. Then we find the mean of two or more courses and calculate the average cosine similarity between them then the courses will be sorted in a descending order depending on the value of the average cosine similarity. The model will complete in the same manner whenever the number of courses increases. If the user hadn’t taken any courses the system will show all the courses that we have in our data. **Recomended\_for\_user ()** function uses the cosine similarity metric and takes two variables X and Y, where X represents all records of all courses and Y represents taken courses by the user.

Two functions can be used to make course recommendations; the first one is the recommended for user function suggests courses to a user based on their previous courses. The user’s name is the input for this function, and the recommend courses function suggests courses based on the course id of the taken course and course name. This function takes a course id as an argument and searches for courses with similar content to the original.

# **Recommendation platform**

## 

In order, to show our results and test them we connect our model with the flask application, where Flask is a Python web application framework that allows end-users to interact with your Python code (in this case, our ML models) directly from their web browser without the need for any libraries, code files, and so on. This prototype application has been implemented using Html, CSS, SCSS, JavaScript, and Python technologies. The application is divided into Frontend and Backend categories.

-Frontend -All the HTML, and SCSS web pages are mainly responsible for the graphical user interface for the data collection and for representing the values returned after processing of the data, it is responsible for making an environment available to the user to be able to view the interface and display the abstract of this thesis.

-Backend- Consists of Flask and python, which are responsible for processing the collected data, accessing the data source, processing for similar items, and returning the recommendations to the frontend. We made our database using SQLite.

We have file name apps inside this file there’s a templates folder, this folder contains the HTML files (index.html, predict.html) that will be used by our main file (app.py) to generate our application's front end. The templates folder in our application contains 4 files which are the accounts pages, and the home page, which includes files, and layouts. First, the account file has 2 forms which are login and register pages. Login Form with fields Username and password, Form 2 which is register form with fields username which is used to authenticate and email (email address) and password also used to authenticate the email. We have 3 registered error handlers which are 404 Error page not found, 403 error access forbidden, and 500 Internal error.

In rendering pages, the user login with his/her email if an email exists, the user will go to the home page, if the email does not exist then the user has to register with a new account or check his/her email or password. We have object current user where Flask-constructed check Login which can be used to determine whether or not the current request is being executed by an authenticated user. The object is global in scope and can be used in all app controllers and handlers.

Now in the user dashboard, if a user has taken some courses, then the system will recommend new courses with similar content to what he too, however, if the user doesn’t take any courses, then the user could search for courses, he/she from the course list and then add to their dashboard after that system will show recommended or related courses to what they enroll to.

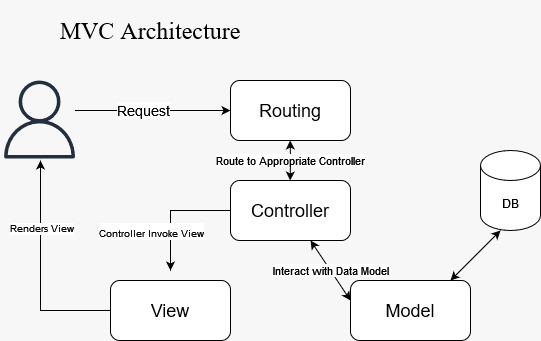


Figure 11: MVC architecture of application.

MVC stands for model, view and controller this is architecture of how course recommendation system works. Each of these parts has a separate function and is designed to manage a certain aspect of an application.

The browser first makes a request to the Controller. To send and receive data, the Controller then engages with the Model. After that, in order to render the data, the Controller communicates with the View. The ultimate presentation is unimportant to the View; what matters is how the information is presented. It will be a dynamic HTML file that renders data in response to commands from the Controller.

The View will then send the Controller its final presentation, and the Controller will then transmit that final data to the user output. The crucial point is that the View and the Model do not ever interact. Only through the Controller do when they interact with one another.

## Use Case 1 for recommender application:

Users enroll in course essentials of machine learning; the system will fetch data from other courses. We have our clusters of description attributes where each cluster contains similar words for each course. We have a description of this course which is [ ***Machine Learning and Data Science Prerequisites: Python's NumPy Stack (V2). I created this course because many students struggle to bridge the gap between machine learning "theory" and writing actual code. "If you can't implement it, you don't understand it," I've always said. This course bridges the knowledge gap by teaching you all of the basic operations required for implementing machine learning and deep learning algorithms. The goal of this course is for you to learn about machine learning algorithms and then implement those algorithms in code using the tools and techniques you've learned. Prerequisites: linear algebra probability Programming in Python"]***

The highlighted words are the most frequent words in cluster 2 as in shown fig.

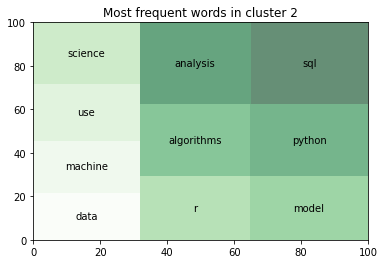
******

Figure 12: Most frequent words in cluster 2

Then the model starts the computation with cosine similarity with essentials of machine learning and all the courses in data, the courses with a higher score or a score close to one will be the recommended courses and system will sort the courses in descending order. The model is integrated with the flask application as we have mentioned before. The user adds a course to his list as shown in fig.

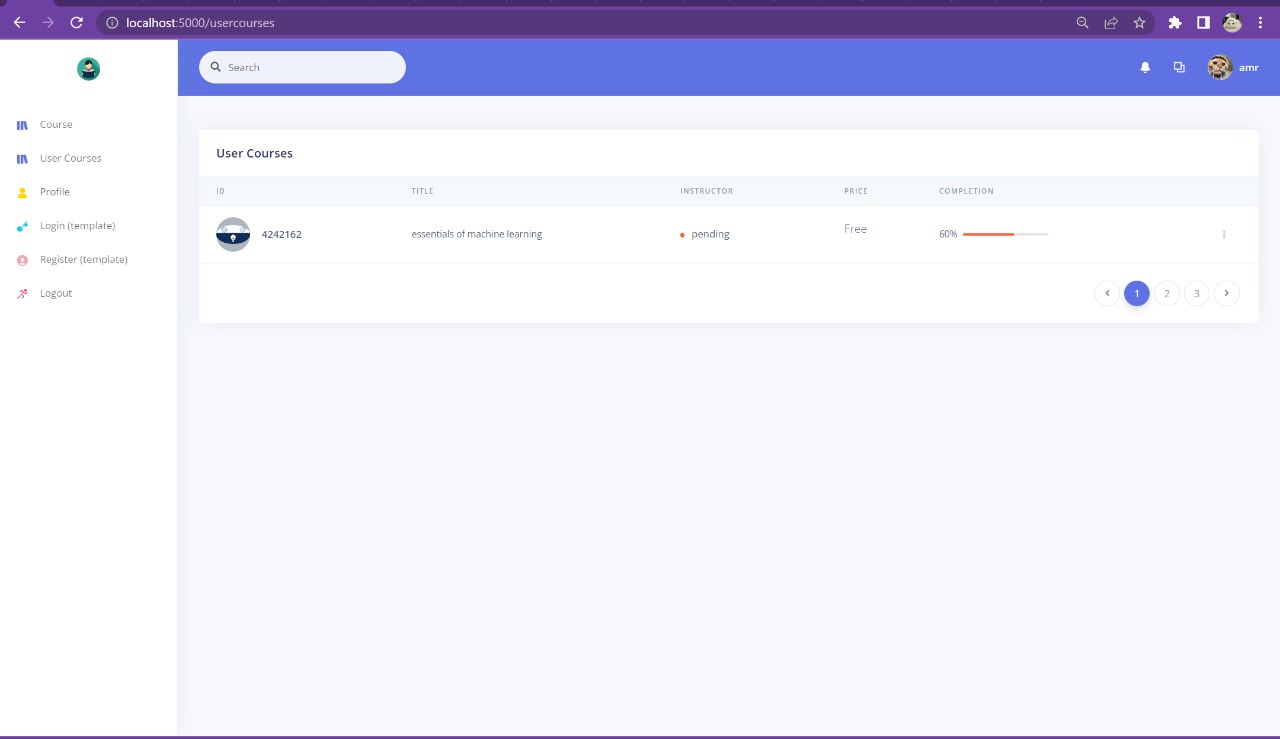


Figure 12: user enrolled to essentials of machine learning

Now, The UI (user interface) shows the results of integrated model with the application. Hence, it will view the recommendations done by the model as shown in next fig. The user could be able to find related courses which he could need to complete His educational career in this field without hours of searching. As shown in the next figure each recommended course has a high cosine similarity score which proves our thesis.

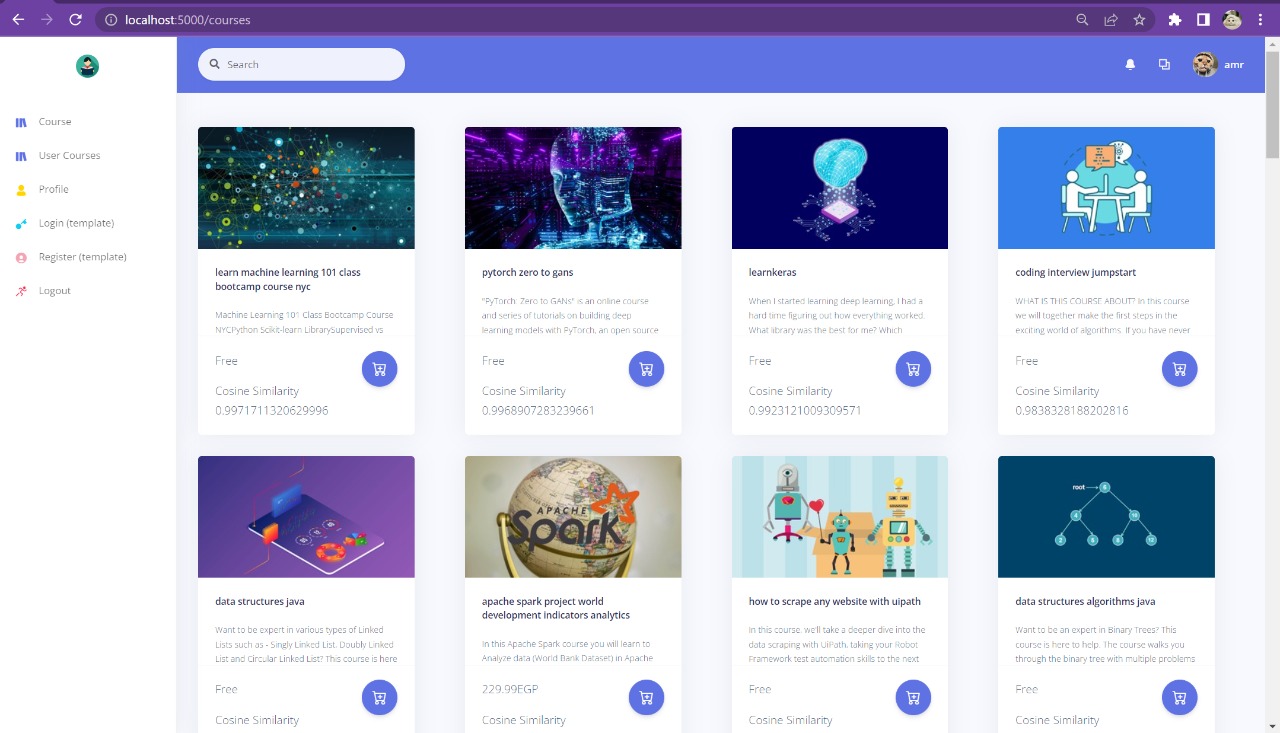


Figure 13: recommended courses for essential of machine learning course

## Use Case 2:

Users enroll in the course create a 2d platformer game with unity; the system will fetch data from other courses. As in the previous, case, we have our clusters of description attributes where each cluster contains similar words for each course. We have a description of this course which is

***[Enroll Now and Learn How to Create Your Own Candy Crush Style Game And Angry Birds Style Game From Scratch Before enrolling in this course, make sure you have prior knowledge of Unity game development and programming. A follow-up to our Mastering 2D Games In Unity: Create 6 Fully Featured Games From Scratch Course!! As a follow-up to a course that already covers the fundamentals of Unity and C#, we are going to get right to the point in this course, so if you are a complete beginner, we recommend that you look at our other course, which will help you understand unity and game development. As is customary in teaching game development, we employ hands-on approaches. In this course, we will create six games from scratch!!! The Circle Pin Game We'll begin with a simple game called Pin The Circle. In the game, we have a rotating circle in which we must shoot all of our needles while avoiding colliding with other needles. We'll make a main menu for the game and add shooting mechanics. spawn a specific number of game objects for a specific level and see how we can easily create levels for our game to make it more difficult for our players.]***

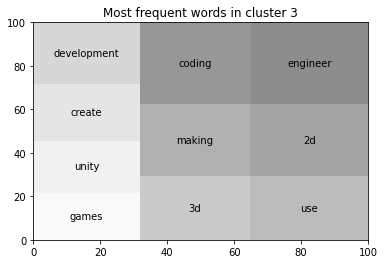
******

Figure 14: Most frequent words in cluster 3

The highlighted words are the most frequent words in cluster 2 as in shown fig.

Here user has enrolled to unity game course as shown in the system.

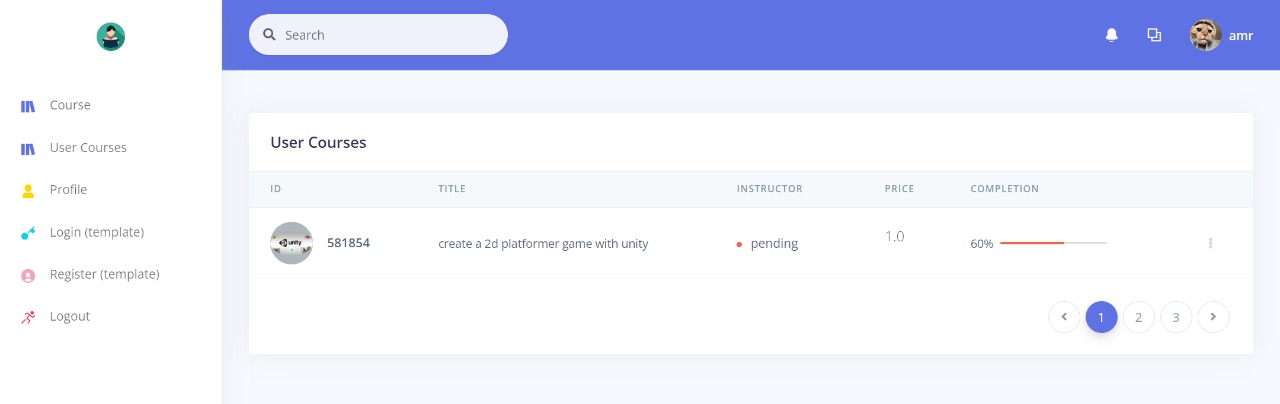


Figure 15: user enrolled to create a 2d platformer game with unity

The system recommends courses which have similar content to the taken course. As shown the cosine similarity has a very high score which is nearly close to 1.

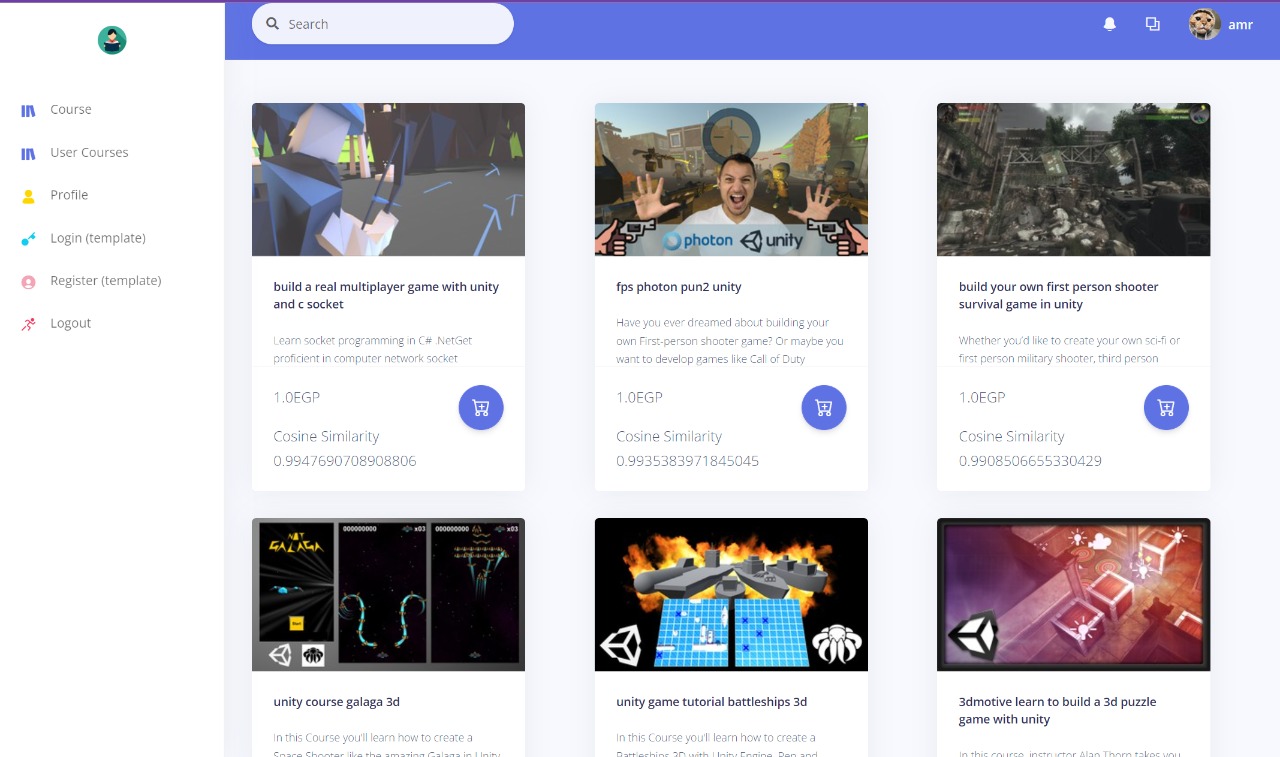


Figure 16: recommended courses to create a 2d platformer game with unity course

## Use Case 3:

In this case, the user has enrolled in 2 different courses with different content. The description of each course has 2 different clusters. The user had enrolled to [create a 2d platformer game with unity](http://localhost:8080/course-details/581854) and enrolled in the essentials of machine learning. The system here will take the mean of cosine similarity of each course which is why the score won’t be high so here we have average cosine similarity. Therefore, the system will recommend courses that have similar content to each of the 2 courses.

In the shown figure the user has enrolled in 2 courses.

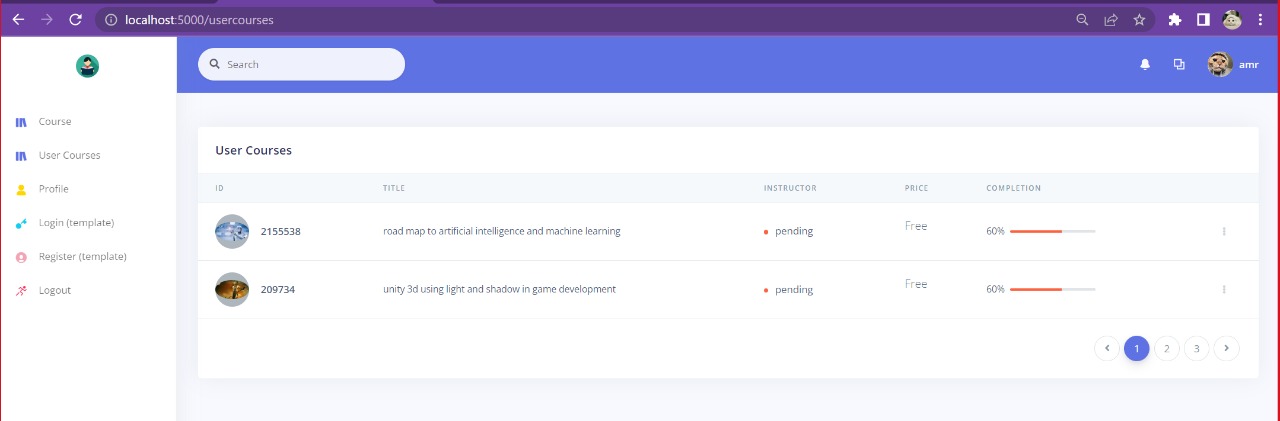


Figure 17: user enrolled into 2 courses from 2 different clusters (different content)

Here in the figure below the system has recommended courses have contents from each course.

And as we can see the cosine similarity is neither high or low because the similarity between two courses is not high.

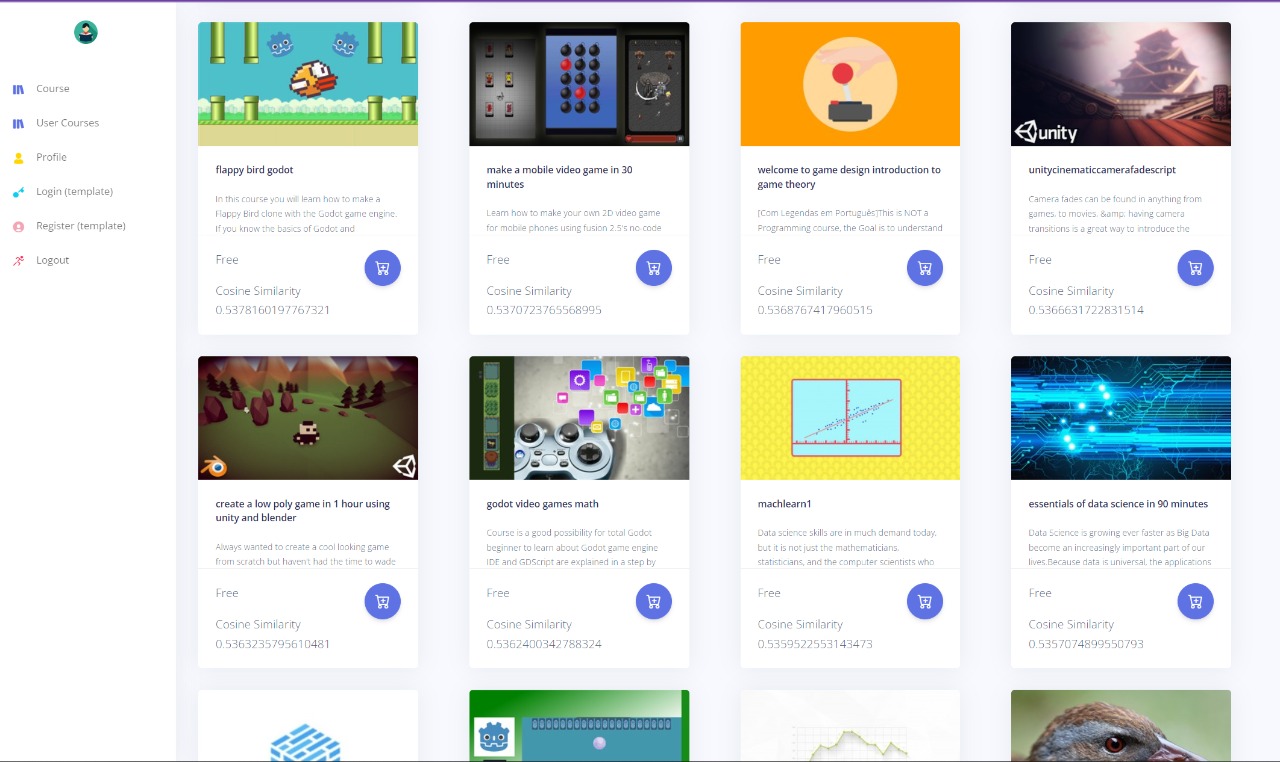


Figure 18: recommended courses for [create a 2d platformer game with unity](http://localhost:8080/course-details/581854) and enrolled in the essentials of machine learning.

## Use Case 4:

In case, the user has enrolled in two courses which are flutterbootcamp and android app development in 2 hours, these two courses are having the same content and their descriptions are from the same cluster.

We have our clusters of description attributes where each cluster contains similar words for each course. We have a part of the description of the flutterbootcamp course which is:

*[ You will have unlimited access to all of the lectures for the rest of your life! This course is backed by a 30-day money-back guarantee! If you are dissatisfied in any way, your money will be refunded. So, what are you holding out for? Learn Flutter in a fun and practical way that will help you advance your career and knowledge! Who should take this course: "Beginner flutter developers interested in developing mobile applications for Android and iOS Developers who want to learn how to create a mobile app using a single code base"]*

Also, we have part of the description of the android app course:

*[ In this short course, I will give you an overview of how to use Android Studio and the Android Virtual Device to start developing Android applications. We will see some of the fundamentals of app development using the Kotlin programming language, such as creating user interface elements, accessing the camera, adding maps to our app, and dealing with user touches. The following is the course content: Introduction to the Android Studio Android Virtual Machine Accessing UI Components in Layout Editor Buttons Views break Functions for Images Object Animator Maps Camera Touch an Object 'Finally, some thoughts’]*

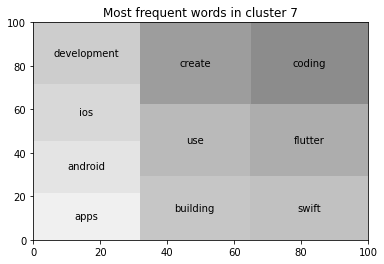


Figure 19: Most frequent words in cluster 7

the user has enrolled in two courses which are flutterbootcamp and android app development in 2 hours and courses had been added to his list.

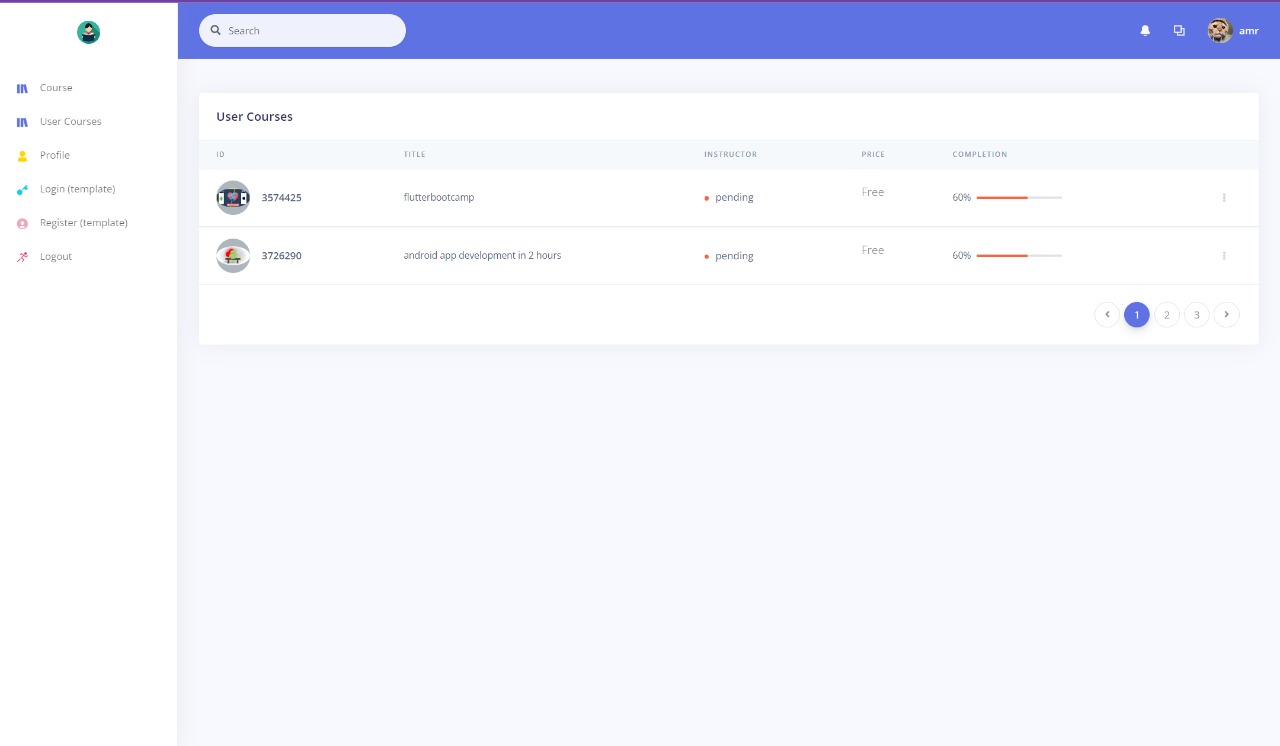


Figure 20: user enroll to 2 courses in the same cluster

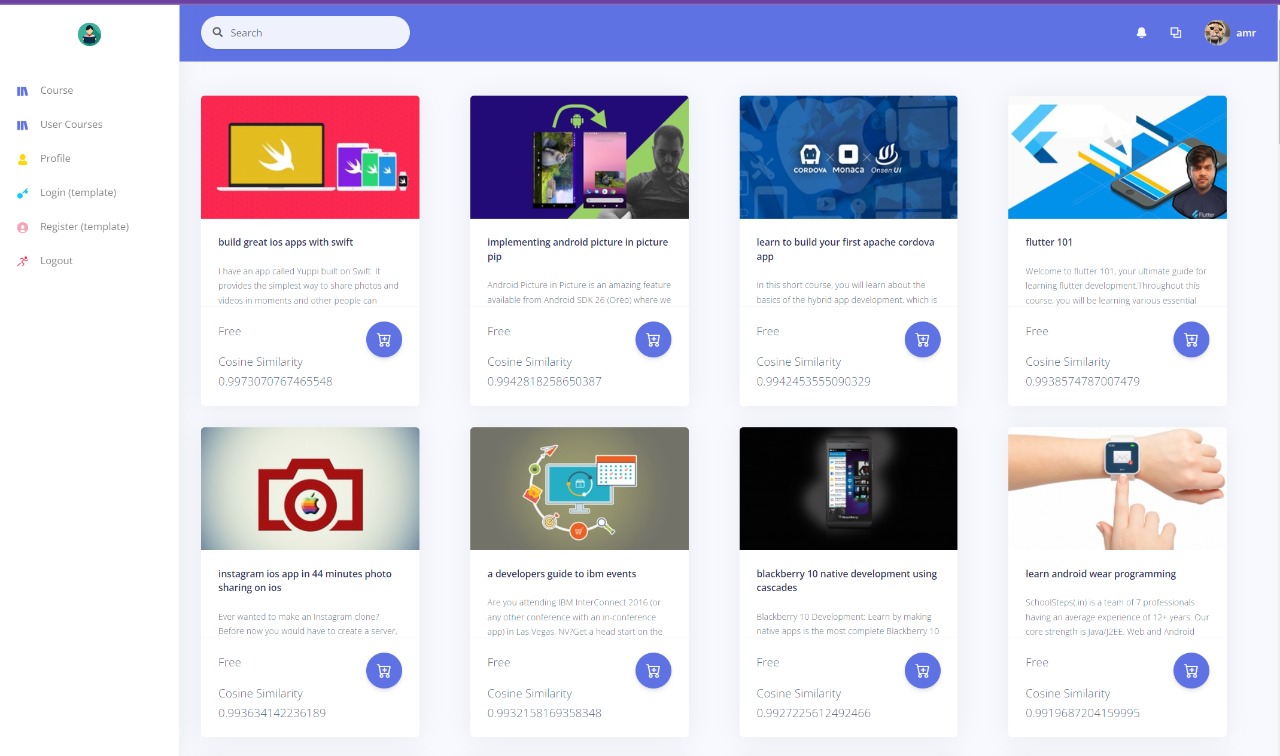


Figure 21: recommended courses for flutterbootcamp and android app development in 2 hours

Here in the figure below the system has recommended courses have contents from each course. And as we can see the cosine similarity is high because the courses are with similar content. Here the score of cosine similarity is near to one.

# **Business model:**

## Introduction:

Using electronic and digital means to deliver courses is a very common tendency, whether for distance learning courses or as enhancements or supplements to what is delivered in classrooms. There are numerous comprehensive online course management systems available today. E-learning is becoming a reality as more learners and educators become attached to technology. With the advancement of the World Wide Web, interest in e-learning and web-based course material delivery has grown. However, very few course management systems include intelligent agents that would allow the course delivery system or suggested learning material to be personalized.

## Problem:

With the increasing number of web applications which is available around the world, it's becoming more difficult to find the right information for a user in a short amount of time. The number of handheld mobile devices is growing, and the majority of business revolves around finding the right data. It is extremely difficult to obtain the required information from web applications without a proper recommender system. Recommender systems are used in web applications to provide relevant data to users based on their preferences and interests. Recommender systems assist us in obtaining the information we demand. It selects only the information that the user requires. In today's world, any system contains a large amount of data. We have to use these data for our benefit. Hence, Recommender systems enable users in selecting items they may require. This output is produced using a variety of artificial intelligence and machine learning techniques. Hence, the need for a recommender engine is a must to recommend courses to the learner, these courses are recommended based on his interests. No system tracks the learning progress of the student and shows a complete report of his progress. No system measures the estimated time of reading to determine the comprehension rate of the learner We will build a course recommender system that should have the capabilities to acquire the student profile to know what his interests are, then recommend a list of courses.

## Objective:

When it comes to sites that are based on monthly subscriptions, such as Netflix, and e-learning websites such as Udemy and Coursera, there has to be a reason why the customer should pay for another month. Most recommender systems are used to ensure that the user always finds something new to watch.

By ensuring that the user receives regular recommendations that are tailored to their preferences, the user is more likely to renew their subscription for another month. The user experience is ultimately what keeps them coming back to you.

## Vision:

A recommender, which is often overlooked as a method of market analysis, can be used to discover user preferences and see what people are most interested in. Businesses can ensure they offer similar products by using user ratings and the number of users watching a show. By offering user something they might be interested in, they are much more likely to use this business on a regular basis. Recommendation system will constantly offer recommendations to customers based on their search results and previous purchases. This is due to the fact that a recommendation system is an excellent way to provide a compelling user experience. Customers will come back to you when we get to know them through content-based approaches. Learning what sells and what doesn't, will provide the primary target audience with exactly what they require. This will quickly result in more sales and profit.

## Future work:

We are going to collect more data in Arabic and improve the recommendation system, we want to make an e-learning system that has English and Arabic content. If we find the appropriate data, we will make a collaborative filtering model

Recommender systems use a technique known as collaborative filtering (CF). Collaborative filtering can be defined in two ways: narrowly and broadly.

In a newer, narrower sense, collaborative filtering is a method of making automatic predictions (filtering) about a user's interests by gathering preferences or taste information from a large number of users (collaborating).

In a broader sense, collaborative filtering is the process of filtering for information or patterns that involve the collaboration of multiple agents, viewpoints, data sources, and so on.

# **Result**

For building a content-based recommendation system, we have collected the data and to work with this data we have applied natural language processing and cleaned the data from tags etc., we have dropped the columns that we don’t need and we kept relevant columns that are recommended for each user are based on.

We have applied k-means clustering to the cleaned data and we were focused on clustering the description and objective of each course, to find the optimal number of K in the description attribute we have used the inertia metric using the elbow method. The optimal number of K’s was 8 clusters. We have applied the same operations to the objective attribute, we found that an optimal number of k for the objective attribute was 6 clusters. Then we applied the silhouette metric to evaluate clustering on objective and description and we found that clustering of description will be more accurate with **Silhouette Coefficient.** Then, we saved these clusters in a sav file.

The method used to determine how similar the courses are is based on their similarities in various characteristics. It represents the cosine of the angle of two vectors projected in a multidimensional space mathematically. The cosine similarity is very useful because it aids in the founding of similar objects.

To create a cosine similarity matrix by using the cosine\_similarity function by Sklearn. Cosine similarity is the metric we are using to see how similar two abstracts are. The cosine\_similarity is a NumPy array containing computed between all abstracts. The similarity matrix when calculated returns a scoring matrix. We take the mean of this score and sort the score of recommended courses in descending manner, these are recommended courses after user Deepak Iyar has taken Microsoft SQL from a to z course with ID 1005698 as shown in the following table.

|  |  |
| --- | --- |
| Published title | Cosine similarity |
| 1. complete-guide-to-automation-with-r-in-2021 | 0.905293 |
| 1. data-visualization-with-python-and-matplotlib | 0.901517 |
| 1. elastic search-and-Kibana | 0.897284 |
| 1. data-structures-and-algorithms-using-java | 0.889786 |

# **Conclusion**

In an e-learning system, research paper recommender systems assist users in finding or obtaining the most relevant courses from a large volume of course papers. To provide recommendations to the intended users, this paper used a content-based filtering technique. Based on the system's findings, incorporating recommendation features in an e-learning program would be beneficial to users. The availability of the contents describing the items and users' profiles of interest resulted in the solution to this problem. Content-based techniques are independent of user ratings but rely on them. This paper also presents an algorithm for providing or recommending recommendations based on the previous learning topics of the user. The algorithm makes use of both the TF-IDF weighing and the cosine similarity measure.

A unified application with a machine learning model was used to give proper a user-friendly environment. This system views the interface and displays the proposal of this thesis. The application enables the users to search for, add, and delete courses. Whenever the user hasn’t enrolled in any courses, the course list will be viewed, however, in case the user has taken courses the system will provide relevant courses to the taken courses. Moreover, users could enroll in new courses with different content and have new recommendations.

# **Evaluating with previous work, techniques comparison**

In the Comparative Study of Recommendation Systems thesis (Lokesh, A. 2019) in the content-based section, they used the cosine similarity function, support vector machine, and TF-IDF technique [33].

Their recommendations were based on the genre of each movie only in our recommendation system, The k-means clustering algorithm is used to improve recommendation accuracy and computational efficiency through efficient learner grouping. we have used k-means clustering to find the most relevant word in the description of each course, we will divide courses into clusters from the same category. Hence, by using k-means clustering the recommendations are more relevant Therefore, the similarity of content and courses will be more accurate. Where using only cosine similarity may recommend courses that are similar but not from the same category Moreover, in our recommendation system we recommend courses based on more than one feature in each course as the average rating of each course, if the user has a certificate or not, the instructional level in that course, how many numbers of lecture includes in the course, number of quizzes, number of assisted articles, price of the course, number of subscribers, sub-categories of course and clusters of the description of each course.

We have normalized the feature matrix and saved clusters of description attributes in the sav file and based on them we have made course recommendations using a cosine similarity metric.

They didn’t integrate their model with an application that could be used by the user and shows the results with the user interface.

Also, here in Content-Based Movie Recommendation System Using Genre Correlation research [31], where they have created a genre data frame with movie IDs as rows in a separate data frame and genres as columns separated by pipeline character. Then they create a list of all the genres available in the dataset. Here they depend on the execrating genre of each movie.

However, in our project, we have clustered descriptions of all courses in the data set to produce more relevant data for the recommendations and normalized feature matrix, so we depend on more than one feature which helps to create more accurate results.

Content-Based Movie Recommendation System Using Genre Correlation research has used Euclidean distance to find similarity

but, in our project, we have used cosine similarity. The cosine similarity is beneficial more than Euclidean distance because, even if two similar documents are separated by the Euclidean distance due to size (for example, the word "python" appeared 50 times in one document and 10 times in another), they can still have a smaller angle between them. The greater the similarity, the smaller the angle [32].

# **Reference**

1. Jyothi, N., et al. (2012). A recommender system assisting instructors in building learning path for personalized learning system. IEEE Fourth International Conference on Technology for Education, Hyderabad, India, 2012.
2. Joy, J., & Pillai, R. V. G. (2021). Review and classification of content recommenders in E-learning environment. *Journal of King Saud University - Computer and Information Sciences*. <https://doi.org/10.1016/j.jksuci.2021.06.009>
3. Itmazi, J. (2010). A Suggested Algorithm of Recommender System to Recommend Learning Objects from Digital Library to Learning Management System. *Asian Journal of Information Technology*, *9*(2), 37–44. <https://doi.org/10.3923/ajit.2010.37.44>
4. Itmazi, J. (2008). Using Recommendation Systems in course management to recommend learning objects. *International Arab Journal for Information Technology*, *5*(3), 234–240.
5. J, B., & B, M. (2021). ONLINE COURSE RECOMMENDATION SYSTEMS. *Natural and Technical Sciences*, *55*(2), 50–56. <https://doi.org/10.47344/sdubnts.v55i2.544>
6. Fraihat, S., & Shambour, Q. (2015). *A Framework of Semantic Recommender System for e-Learning – SOFTWARE*. Jsoftware. *(10)3.*
7. Gazdara, A., & Hidri, L. (2020). A new similarity measure for collaborative filtering-based recommender systems. *Knowledge-Based Systems*, *188*. https://doi.org/10.1016/j.knosys.2019.105058
8. Jeevamol, J., & Renumol, V. (2021). Comparison of Generic Similarity Measures in E-learning Content Recommender System in Cold-Start Condition.*Ieeexplore*,175–179. <https://doi.org/10.1109/IBSSC51096.2020.9332162>.
9. Lops P, Gemmis M D, Semeraro G. (2011) “Content-based Recommender Systems: State of the Art and Trends,” Recommender Systems Handbook. Springer US, 73-105.
10. Noia T D, Mirizzi R, Ostuni V C, et al. (2012) “Linked open data to support content-based recommender systems,” in Proc. International Conference on Semantic Systems. ACM, 1-8.
11. Mu, R. (2018). A Survey of Recommender Systems Based on Deep Learning.*IEEEAccess*,*6*,69009–69022. <https://doi.org/10.1109/access.2018.2880197>
12. Patra, P., Mondal, B., Patra, O., & Mishra, S. (2020). A course recommendation system based on grades. *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*. <https://doi.org/10.1109/ICCSEA49143.2020.9132845>
13. Jafari, S., Shokrollahi, Z., & Asadi, S. (2018). Developing a Course Recommender by Combining Clustering and Fuzzy Association Rules.

*Journal of AI and Data Mining*, *7*(2), 249–262. <https://doi.org/10.22044/jadm.2018.6260.1739>

1. J. Joy and R. V. G, "Comparison of Generic Similarity Measures in E-learning Content Recommender System in Cold-Start Condition," 2020 IEEE Bombay Section Signature Conference (IBSSC), 2020, pp. 175-179, doi: 10.1109/IBSSC51096.2020.9332162.
2. Lops P., de Gemmis M., Semeraro G. (2011) Content-based Recommender Systems: State of the Art and Trends. In: Ricci F., Rokach L., Shapira B., Kantor P. (eds) Recommender Systems Handbook. Springer, Boston, MA. <https://doi.org/10.1007/978-0-387-85820-3_3>
3. Huynh-Ly, T.Nguyen, H.&Thai-Nghe, N.(2016).Methods for building course recommendation systems. 163-168. 10.1109/KSE.2016.7758047.
4. Kulkarni, P . Rai, S & Kale, R. (2020). Recommender System in eLearning: A Survey. 10.1007/978-981-15-0790-8\_13.
5. Souabi, S. (2021). Recommendation Systems on E-Learning and Social Learning. *A Systematic Review. The Electronic Journal of e-Learning*, *19*(5), 432–451. <http://www.ejel.org>
6. Martínez, O. (2020). Using Recommendation System for E-learning Environments at degree level. *International Journal of Artificial Intelligence and Interactive Multimedia*, *1*(2), 67–70. <https://core.ac.uk/display/233058099?utm_source=pdf&utm_medium=banner&utm_campaign=pdf-decoration-v1>
7. J. Joy and R. V. G, "Comparison of Generic Similarity Measures in E-learning Content

Recommender System in Cold-Start Condition," 2020 IEEE Bombay Section Signature Conference (IBSSC), 2020, pp. 175-179, DOI: 10.1109/IBSSC51096.2020.9332162.

1. Joy, J., & Pillai, R. V. G. (2021). Review and classification of content recommenders in E-

learning environment. *Journal of King Saud University - Computer and Information Sciences*. <https://doi.org/10.1016/j.jksuci.2021.06.009\>

1. Jyothi, N., et al. (2012). A recommender system assists instructors in building learning path

for a personalized learning system. IEEE Fourth International Conference on Technology for Education, Hyderabad, India, 2012.

1. Joy, J., & Pillai, R. V. G. (2021). Review and classification of content recommenders in E-

learning environment. *Journal of King Saud University - Computer and Information Sciences*. <https://doi.org/10.1016/j.jksuci.2021.06.009>

1. Huynh-Ly, T.Nguyen, H.&Thai-Nghe, N.(2016).Methods for building course recommendation

systems. 163-168. 10.1109/KSE.2016.7758047.

1. Itmazi, J. (2008). Using Recommendation Systems in course management to recommend

learning objects. *International Arab Journal for Information Technology*, *5*(3), 234–240.

1. J, B., & B, M. (2021). ONLINE COURSE RECOMMENDATION SYSTEMS. *Natural and*

*Technical Sciences*, *55*(2), 50–56. <https://doi.org/10.47344/sdubnts.v55i2.544>

1. Kulkarni, Pradnya & Rai, Sunil & Kale, Rohini. (2020). Recommender System in eLearning:

A Survey. 10.1007/978-981-15-0790-8\_1

1. Patra, P, Mondal, B., Patra, O., & Mishra, S. (2020). A course recommendation system based

on grades. *2020 International Conference on*

*Computer Science, Engineering and Applications (ICCSEA)*.

<https://doi.org/10.1109/ICCSEA49143.2020.9132845>

1. Jafari, S., Shokrollahi, Z., & Asadi, S. (2018). Developing a Course Recommender by

Combining Clustering and Fuzzy Association Rules.

1. *Journal of AI and Data Mining*, *7*(2), 249–262. <https://doi.org/10.22044/jadm.2018.6260.1739>
2. Kunisetti, S. (2022). Content-Based Movie Recommendation System Using Genre Correlation [E-book]. In S. Nalluri (Ed.), *Smart*

*Intelligent Computing and Applications* (Vol. 105, pp. 391–397). Smart Innovation, Systems and Technologies. Retrieved 2020,

from <https://link.springer.com/chapter/10.1007/978-981-13-1927-3_42>

1. Prabhakaran, S. (2022, April 20). *Cosine Similarity - Understanding the math and how it works? (with python)*. Machine Learning Plus.

<https://www.machinelearningplus.com/nlp/cosine-similarity/>

[33] Lokesh, A. (2019). *A Comparative Study of Recommendation Systems*. Western Kentucky University, Western Kentucky.