Machine Learning for Data Science and Analytics ECE461

Department of Electrical and Computer Engineering
University of Thessaly

Final Project on Recommendation Systems

Kavvathas Spyridon **02928**
Lagomatis Ilias **03005**
Petridis Christos **03086**
{skavvathas, ilagomatis, cpetridis}@uth.gr

TEAM G

Fall 2023

1 Abstract

A recommender system serves as a form of information filtering mechanism that predicts what things a user might like and provides recommendations based on those predictions. In this project, we present a comprehensive exploration of machine learning techniques applied to the domain of online course recommendation using datasets that are based on Coursera courses. Leveraging diverse algorithms such as **Singular Value Decomposition (SVD)**, **Non-Negative Matrix Factorization (NMF)**, **Term Frequency-Inverse Document Frequency (TF-IDF)**, **Jaccard similarity**, **K-means**, **K-Nearest Neighbors (KNN)**, and **Neural Networks**, we strategically embed collaborative filtering and content-based methodologies within three distinct datasets, each algorithm exclusively applied to a specific dataset.

Our investigation delves into the landscape of collaborative filtering, aiming to enhance the accuracy and efficiency of course recommendations by analyzing user preferences and behaviors. Simultaneously, we explore content-based machine learning approaches, focusing on the intrinsic characteristics of courses to tailor recommendations based on course content and user profiles.

By trying out these methods on different sets of data, we are not just figuring out how well they work but also dealing with the specific difficulties that come with the many courses on Coursera. This project is important because it helps make personalized learning better, considering what different users like. It might also have an impact on how recommendation systems are designed and improved on big educational platforms such as Coursera.

2 Introduction

In the rapidly evolving domain of online education, personalizing learning experiences has become crucial. Our project aims to develop a sophisticated recommender system for Coursera. Coursera is an online platform that offers a wide range of courses and specializations taught by instructors from top universities and organizations around the world. This system leverage advanced machine learning techniques to suggest courses that align with individual learners' preferences, learning styles, and educational needs.

For our project, we have sourced three distinct datasets, each offering a unique perspective on Coursera's educational offerings. These datasets are: `Coursera.csv`, `Coursera courses.csv`, and `Coursera reviews.csv`. Our selection process involved an extensive online search focused on identifying datasets specifically related to Coursera. A significant part of our research was conducted on Kaggle, a renowned platform known for its comprehensive collection of datasets.

To enhance the accessibility and user-friendliness of our project, we have developed a sophisticated user interface. This interface plays a crucial role in enabling users to easily interact with and comprehend the various machine learning algorithms implemented in our system. For the creation of this interface, we have utilized Streamlit, a highly regarded Python framework known for its efficiency and capability to produce engaging interfaces. Streamlit's intuitive design and flexibility have allowed us to construct a user interface that is not only visually appealing but also straightforward to navigate, thus significantly enhancing the user experience.

Regarding the input for our algorithms, it varies depending on the specific machine learning technique employed. For instance, we use the `Coursera.csv` file as the input when applying the **TF-IDF** and **Jaccard Similarity** algorithms. These techniques are instrumental in recommending courses that are similar to the user's selected course, providing personalized suggestions based on content similarity. Additionally, the same `Coursera.csv` dataset serves as the input for the K-means algorithm, a clustering technique. The **K-means** algorithm also aims to identify courses similar to the user's choice, but it does so through a different approach, grouping courses into clusters based on their features. In addition, **K-Nearest Neighbor** is used for content based and collaborative filtering recommendations, as we apply this algorithm both in the ratings dataset `Coursera reviews.csv` and in the dataset with course content `Courses.csv`.

Furthermore, we have implement Matrix Factorization techniques, **SVD** and **NMF**, utilizing the `Coursera courses.csv` and `Coursera reviews.csv` datasets. The primary objective is to generate course recommendations that are finely tuned to each user's preferences, as determined by their historical course ratings. The output produced by these Matrix Factorization techniques is a carefully curated list featuring the most relevant and highly recommended courses for each individual user.

3 Literature Review

To begin with, we have make a review of the current literature in the topic of the courses recommendations. Firstly, we have selected the paper named **"A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields"**[4]. This paper covers many topics, algorithms and techniques of the field of recommendation systems. It discusses **various filtering methods like Content-Based, Collaborative, and Hybrid systems**, detailing their operational principles, strengths, and limitations. It also covers different recommendation techniques such as Text Mining, KNN, Clustering, Matrix Factorization, and Neural Networks, describing their functionality and application in the recommendation domain. The paper emphasizes the importance of qualitative evaluation metrics like RMSE, Precision, Recall, and AUC to assess the performance of these systems. Moreover, it provides insights into the trends and research progress in recommendation systems from 2010 onwards, underlining their growing complexity and sophistication due to advances in data mining and machine learning technologies.

Next, we found very interesting a very famous paper with approximately 6,000 citations named **"Neural collaborative filtering"**[3]. This paper presents an in-depth study of recommendation systems using deep neural networks for **collaborative filtering** on implicit feedback data. The authors propose a framework called Neural Collaborative Filtering (NCF), which generalizes matrix factorization techniques and employs a multi-layer perceptron to capture the non-linear user-item interactions. The framework is designed to enhance the performance of recommendation systems significantly. They introduce a neural matrix factorization model that combines linearity with non-linearity for improved recommendation quality. The paper also includes extensive experiments on real-world datasets that demonstrate the superiority of the NCF framework over state-of-the-art methods, particularly emphasizing the advantages of deep learning for collaborative filtering tasks.

Moreover, we examined the **content-based** technique and we selected the paper named "Hybrid Recommender Systems based on Content Feature Relationship"[1]. It introduces a new hybrid recommender system that incorporates the relationship between content features to improve recommendation accuracy. It proposes a method for extracting **content feature relationships, integrating these into a collaborative filtering algorithm to better address the cold-start problem**, and developing a content-based recommendation system. The results on a movie dataset demonstrate significant improvements in accuracy, novelty, and diversity of recommendations compared to existing methods. The research highlights the utility of considering content feature relationships in recommendation systems.

We found very useful to examine ensemble methods and we believe that the paper named "Presentation of a recommender system with ensemble learning and graph embedding: a case on MovieLens"[5] is a straightforward one. It outlines a novel recommender system utilizing **ensemble learning and graph embedding**, specifically tested on the MovieLens dataset. It emphasizes improving recommendation accuracy by classifying users based on **ensemble learning techniques, fuzzy rules, and decision trees.** Furthermore, it incorporates heterogeneous knowledge graphs and embedding vectors to model user behavior more intricately. The system demonstrates improved performance in terms of precision and user-behavior analysis over traditional methods.

To gain insight into the impact of **deep learning on recommendation systems**, we select the paper named "Wide & Deep Learning for Recommender Systems"[2]. It combines linear models for memorization of feature interactions with neural networks for generalization of unseen interactions. This hybrid approach was implemented and evaluated on the Google Play app store and showed significant improvements in app acquisition rates over models using only wide or deep components. The framework, open-sourced in TensorFlow, highlights the effectiveness of combining memorization and generalization for recommendation tasks.

4 Dataset and Features

As mentioned above, our work is built upon three distinct datasets.: `Coursera.csv`, `Coursera courses.csv`, and `Coursera reviews.csv`.

Firstly we will analyse the `Coursera.csv` file, which is a rich collection of data on Coursera courses, covering everything from the course name and university to the level of difficulty. This dataset comprises 3,532 rows and 7 columns:

Name	Data type
CourseName	string
University	string (Beginner, Advanced etc.)
DifficultyLevel	string
CourseRating	decimal (1-5)
CourseURL	string
CourseDescription	string
Skills	string

Table 1: `Coursera.csv` description

In addition, the `Coursera reviews.csv` dataset complements our analysis, offering insights into user reviews for Coursera courses. This dataset comprises 1,454,711 rows and 4 columns:

Name	Description
userID	identifier for users
course id	identifier for each course
rating	user-assigned ratings ranging from 1 to 5, reflecting their satisfaction with the course

Table 2: `Coursera reviews.csv` description

To identify the specific course corresponding to the reviews in the `Coursera reviews.csv` dataset, we utilize the `course id` column, allowing us to link each reviewer's rating to the respective course name in the `Coursera courses.csv` dataset. `Coursera courses.csv`, containing 623 rows and 4 columns:

Name	Description
Course Name	Describes the name/title of the course, providing a clear identification.
Institution	Specifies the educational institution offering the course, offering insights into the course's academic origin.
Course URL	Represents the web address for the course on Coursera, facilitating easy access to additional details.
Course ID	Serves as a unique alphanumeric identifier for each course, allowing for efficient cross-referencing with other datasets.

Table 3: `Coursera courses.csv` description

5 Methods

5.1 Singular Value Decomposition

Matrix factorization techniques, such as **Singular Value Decomposition (SVD)**, play a crucial role in recommender systems. In the context of collaborative filtering, where user-item interaction data is represented as a matrix, SVD offers a powerful approach for extracting latent features and uncovering underlying patterns. Unlike eigen-decomposition, which is applicable to square matrices, SVD extends this concept to rectangular matrices. Given a **user-item matrix A** of dimensions $m \times n$, SVD factorizes it into three matrices: U, Σ, and V^T. Here, U is an $m \times m$ matrix capturing information about the users, Σ is an $m \times n$ diagonal matrix containing singular values, and V^T is an $n \times n$ matrix representing item features. The singular values in Σ are analogous to eigenvalues, and the left and right singular vectors in U and V provide insights into the relationships between users and items. The orthogonality of U and V ensures that these vectors are mutually independent, enhancing the interpretability of the decomposition. The resulting factorization

$$A = U\Sigma V^T$$

serves as a foundation for collaborative filtering in recommender systems, helping to predict user preferences and generate personalized recommendations based on the uncovered latent features.

5.2 Non-Negative Matrix Factorization

Non-Negative Matrix Factorization (NMF) is a matrix decomposition technique that has found widespread application in various fields, including image processing, text mining, and recommender systems. In this project, we apply it in our recommender system. Unlike traditional matrix factorization methods, NMF enforces non-negativity constraints on the factorized matrices, ensuring that both the input matrix and the resulting factorized matrices contain only non-negative values. Given an input matrix V of dimensions $m \times n$, NMF aims to find non-negative matrices W $(m \times r)$ and H $(r \times n)$, where r is the chosen rank, such that

$$V \approx WH$$

Here, W captures the basis vectors, and H represents the coefficients corresponding to these basis vectors.

5.3 TF-IDF

TF-IDF is a numerical statistic used in information retrieval and text mining to evaluate the importance of a word in a document relative to a collection of documents (a corpus). TF-IDF combines two metrics: term frequency (TF) and inverse document frequency (IDF).

Term Frequency (TF):

$$TF = \frac{Number\ of\ times\ term\ appears\ in\ a\ document}{Total\ number\ of\ terms\ in\ the\ document}$$

Inverse Document Frequency (IDF):

$$IDF = \log\left(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ containing\ the\ term}\right)$$

The TF-IDF of a term is calculated by multiplying TF and IDF scores.

$$TF - IDF = TF \times IDF$$

TF-IDF assigns higher weights to terms that are frequent within a document but relatively rare across the entire corpus, aiding in identifying important words in a document and enabling tasks like document retrieval, classification, and clustering.

5.4 Jaccard Similarity

The **Jaccard similarity** or Jaccard index is a statistic method used for comparing the similarity and diversity of sample sets. Mathematically, the Jaccard similarity, J, between two sets A and B can be expressed as:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Here, $|A \cap B|$ represents the number of elements common to both sets (the intersection), and $|A \cup B|$ represents the number of elements in either set (the union).

The Jaccard similarity ranges from 0 to 1:

- A value of 0 indicates that the two sets have no elements in common.
- A value of 1 indicates that the two sets are identical.

5.5 KNN

The **k-Nearest Neighbors (KNN)** algorithm is a supervised machine learning algorithm used for classification and regression tasks. It's a simple and intuitive algorithm that is based on the idea that similar data points tend to belong to the same class or have similar numeric values. KNN is a non-parametric and lazy learning algorithm that makes predictions by considering the majority class (for classification) or the average (for regression) of the k-nearest data points in the feature space. For a given data point in the testing set, KNN identifies the k-nearest neighbors from the training set based on a distance metric (commonly Euclidean distance).

5.6 K-Means

K-means algorithm is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

5.7 Neural Network

Neural networks can be used in many different ways in any field and especially in the field of recommendation systems. There is no one-fits-all solution in the dynamic landscape of neural network-based recommendation systems. In our case, the **neural network has been implemented primarily for content-based recommendations** in the context of online courses. Through the network's ability for **dimensionality reduction and feature learning**, we extract valuable insights from course descriptions and skills. By generating course embeddings, we condense the essential course characteristics into a lower-dimensional space, facilitating the calculation of cosine similarities. This empowers us to provide users with personalized course recommendations based on content similarity, enhancing their learning experience.

6 Experiments and Results

6.1 Singular Value Decomposition

Singular Value Decomposition is used for **courses collaborative filtering recommendations** in our project. We apply this algorithm to the ratings dataset `Coursera reviews.csv` and we aim to take in the end recommended courses for every user. We have choose to take the 10 first recommendationsof courses.

To begin, we preprocess our dataset, represented as a DataFrame containing user ratings for various courses, and transform it into a user-item matrix. This matrix captures the interactions between users and courses, with any missing values filled in as zeros. Subsequently, we employ NumPy and the SVD technique to factorize the user-item matrix into three matrices: U (user-feature), Σ (singular values), and Vtx (course-feature). By retaining only the top k singular values and associated vectors, we effectively reduce the dimensionality of the data.

This k value is called latent factor and is an unobservable variable or hidden feature that captures underlying patterns or characteristics in data, often used in matrix factorization techniques to model complex relationships between entities. We select the k that give us the smaller Mean Squared Error (k = 100), after we test some different k values.

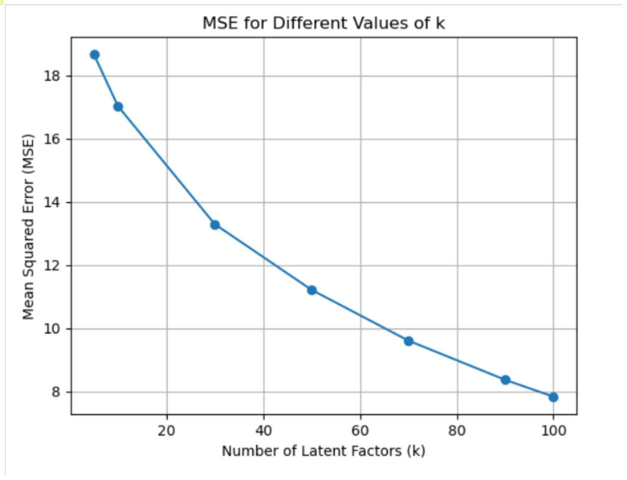


Figure 1: Different MSEs for different K

The resulting reduced matrices are then used to approximate the original user-item matrix. This approximation, obtained through SVD, serves as the foundation for generating course recommendations for each user, with the final recommendations being based on the top courses extracted from this approximation. The recommendations are subsequently saved for users to explore, providing them with a personalized selection of courses based on their past ratings and preferences. This implementation showcases how SVD plays a vital role in enhancing the quality and accuracy of our Coursera course recommendation system, ultimately enhancing the user experience.

6.2 Non-Negative Matrix Factorization

In this project, **Non-Negative Matrix Factorization** is used for **collaborative filtering recommendations**. We apply this algorithm to the ratings dataset `Coursera reviews.csv` and we aim to take in the end, 10 recommended courses for a specific user.

Initially, we preprocess our dataset, structured as a DataFrame containing user ratings for courses, transforming it into a user-item matrix while ensuring missing values are filled in as zeros. Subsequently, utilizing scikit-learn, we implement NMF with 100 latent factors and initialize it with random values. Through Stochastic Gradient Descent (SGD), the model is fitted to the user-item matrix, resulting in two factorized matrices: W (user-feature) and H (course-feature). These matrices are instrumental in reconstructing the original user-item matrix, *R_approx_sgd*. By utilizing this approximation, we generate highly personalized course recommendations for individual users, facilitating the discovery of 10 top-rated courses tailored to each user's unique preferences and past interactions, thereby enriching their Coursera learning journey.

6.3 TF-IDF

In our project, we have also employed the **TF-IDF (Term Frequency-Inverse Document Frequency)** method coupled with Cosine Similarity to enhance our content-based recommendation system. First, we gathered information from different parts of the `Coursera.csv` dataset, like the `Course Name`, `Difficulty Level`, `Course Description`, and `Skills`. To keep things consistent, we made all the texts lowercase. Then, we applied the TF-IDF method separately to the following columns: **1)** `Course Description`**, 2)** `Course Name`**, and 3)** `Skills`. After that, we combined these columns into one for better results. The goal was to make our recommendation system better by taking important information from different parts of the dataset. Thus, we have 4 different applications of this method. **In the content-based scenario, having more information that explains each course would undoubtedly result in better recommendations for similar courses.**

Once the text was vectorized, we used Cosine Similarity to measure the likeness between courses (vectors). This similarity metric, by comparing the cosine of the angle between TF-IDF vectors, efficiently identified how closely related the courses were in terms of their content and thematic elements. A similarity matrix was generated, capturing the degree of similarity between each pair of courses.

To provide practical recommendations, for any given course, we extracted the most similar courses based on their cosine similarity scores. These recommendations were ranked, prioritizing courses with higher similarity scores, and thus, more relevance. This TF-IDF and Cosine Similarity-based approach significantly refined our recommendation system, enabling more precise and content-focused course suggestions.

6.4 Jaccard Similarity

For the Jaccard Similarity algorithm we have use the `Coursera.csv` in order to employ our content based recommendation. The process began with a meticulous preprocessing of the data, focusing particularly on the `Skills` column. This step was crucial to accurately represent the unique skill sets associated with each course. Following this, the skills of each course were compared against every other, calculating their Jaccard Similarity - a measure based on the intersection and union of their skill sets. This computation resulted in a comprehensive similarity matrix, encapsulating the likeness between all pairs of courses in terms of their skills. This matrix then served as the foundation for our recommendation engine, enabling us to identify and suggest courses that share similar skill sets, thereby providing tailored recommendations to learners based on their interests and existing skill profile.

6.5 K-Nearest Neighbors

K-Nearest Neighbors algorithm (KNN) is a very simple, non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another. **In our case we use k-NN for content based & collaborative filtering recommendations. We apply this algorithm both in the ratings dataset `Coursera reviews.csv` and in the dataset with course content `Courses.csv`**.

When it comes to the courses-based implementation (`Courses.csv`), the purpose is to recommend the top-K "Neighbors" of a liked course. For example, if the user has rated course A with 5 out of 5, then we search in the model who are the K closest neighbors at this course and these are the recommendations. We use TF-IDF Vectorizer to convert text data into numeric.

When it comes to the ratings-based implementation `Coursera reviews.csv`, the concept is to find similar users based on their rating. Again, using the ratings from each user we aim to suggest similar content to similar users. Here is an example to make it more clear. Let's say that there are 26 courses (A, B, C, D, ..., Y, Z).

Course	Rating
course A	5
course C	5
course F	5
course J	5

(a) Ratings for user U1

Course	Rating
course A	5
course C	5
course F	5
course J	5
course W	5

(b) Ratings for user U2

Table 4: Collaborative filtering using K-NN algorithm

Based on user U2 we should recommend course W to user U1. However, there is a problem here because if there are K users with the exact same ratings (or less) as user U2 we cannot recommend more than 1 course to user U1.
NOTE: We should also set a similarity threshold

6.6 K-Means

The K-means algorithm was employed in the `Coursera.csv` file to enhance the recommendation system for online courses. Initially, the dataset was preprocessed, combining relevant features such as `Course Name` and `Skills` to create a consolidated `combined features` column. Subsequently, a Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer was applied to convert the textual data into a numerical format, emphasizing the significance of unique terms across the dataset while considering their frequency. The optimal number of clusters (k) was determined using the Elbow method, and a range of k values was explored to identify the point of diminishing returns in terms of inertia. The chosen k value, in this case, was set to 180. Below we can see the plot of Elbow Method :

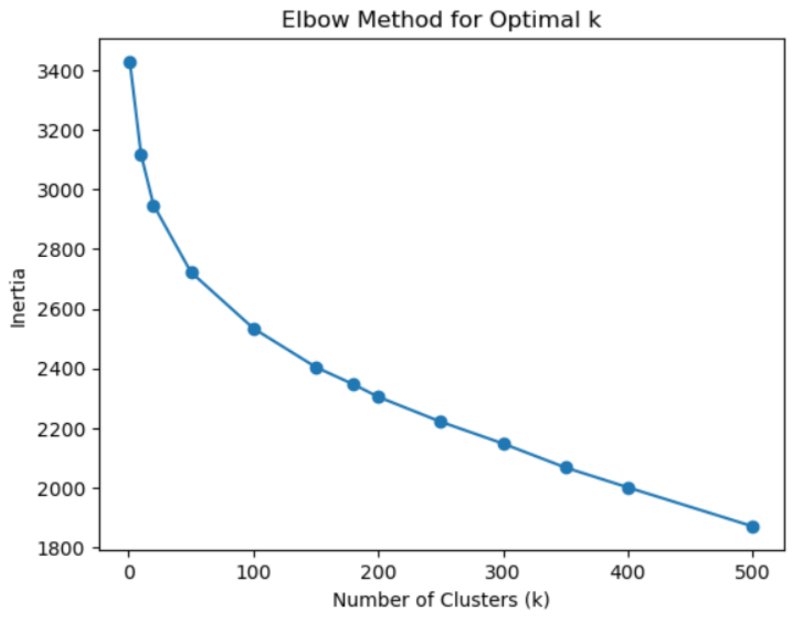


Figure 2: Elbow Method for optimal k

The K-means algorithm was then executed with the selected k, and a query course's TF-IDF vector was used to predict its cluster. Finally, courses within the same cluster were identified and presented as similar items to the queried course, providing users with relevant and clustered course recommendations.

6.7 Neural Network

Our neural network follows the Autoencoder architecture 3 for generating course embeddings and the way it works is very interesting. An autoencoder is designed to learn efficient representations of the input data, typically for the purpose of dimensionality reduction or feature learning.

Autoencoders are trained by minimizing the difference between the input and the output, which is known as the reconstruction loss (in our case it is the mean squared error).

Our neural network

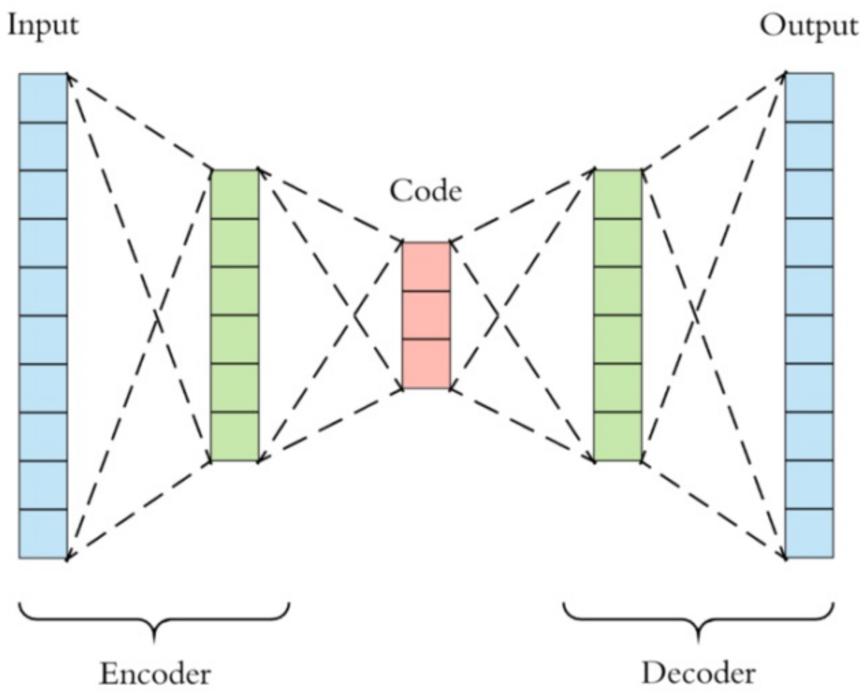
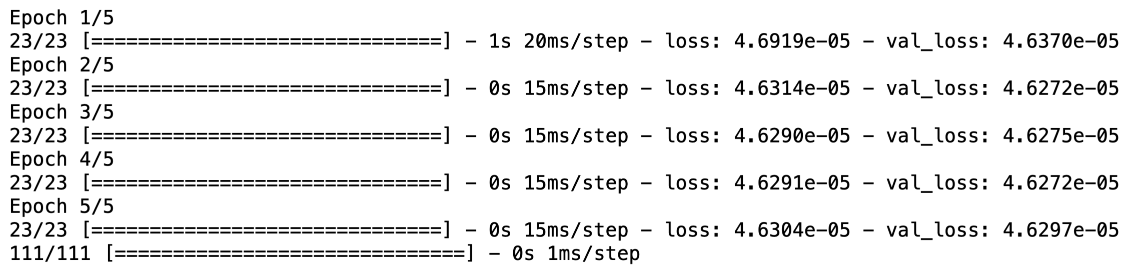


Figure 3: Architecture of Autoencoders

- First of all, the TF-IDF (Term Frequency-Inverse Document Frequency) Vectorizer is used to convert the text data (`Course Description + Skills`) in `combined_features` into a numeric form that can be processed by the neural network, giving importance to more meaningful words and reducing the weight of common words (`tfidf_matrix`) .
- After a lot of experiments, the neural network finally consists of three hidden layers with 128, 64, and 32 neurons, respectively, each using the ReLU activation function. **In order to prevent overfitting** we add Dropout layers with a dropout rate of 0.2 after the first and second hidden layer. The output layer has the same number of neurons as the input features, and it uses a linear activation function. This suggests that the model is performing a regression task, trying to predict the input features themselves. The model is compiled using the `tf.keras.optimizers.legacy.Adam` optimizer and mean squared error as the loss function. It is noteworthy that the above experiments have been implemented in an Apple M1 Pro and therefore we should use the `tf.keras.optimizers.legacy.Adam`.
- Autoencoder neural networks require **embeddings extracting.** Therefore, our neural network initializes a new sequential model and incorporates the first three layers of the trained model (Fig.7). These layers are then used to predict embeddings for the entire dataset (`tfidf_matrix`) after normalizing it. The result holds the lower-dimensional representations of the new data, encapsulating meaningful patterns learned by the initial model. This process is valuable for feature learning, where the extracted embeddings capture essential information about the input data's structure.



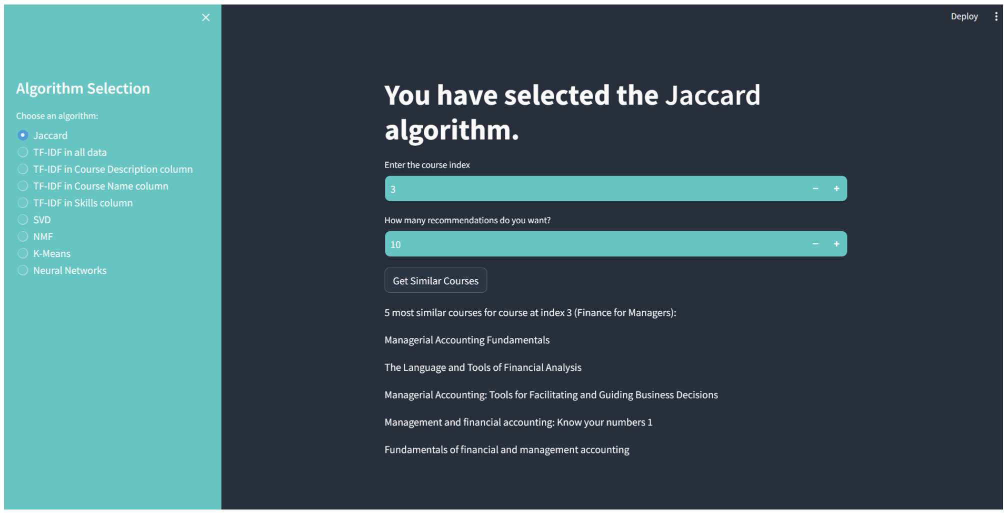
```
Epoch 1/5
23/23 [==============================] - 1s 20ms/step - loss: 4.6919e-05 - val_loss: 4.6370e-05
Epoch 2/5
23/23 [==============================] - 0s 15ms/step - loss: 4.6314e-05 - val_loss: 4.6272e-05
Epoch 3/5
23/23 [==============================] - 0s 15ms/step - loss: 4.6290e-05 - val_loss: 4.6275e-05
Epoch 4/5
23/23 [==============================] - 0s 15ms/step - loss: 4.6291e-05 - val_loss: 4.6272e-05
Epoch 5/5
23/23 [==============================] - 0s 15ms/step - loss: 4.6304e-05 - val_loss: 4.6297e-05
111/111 [==============================] - 0s 1ms/step
```

Figure 4: Training process of our neural network

In a typical machine learning task, such as classification or regression, we would only use `predict` on the test set, because we are trying to evaluate the model's performance on unseen data. However, the goal in this scenario is different. We are training a neural network to learn a meaningful representation (embedding) of each course based on its features (extracted from course descriptions and skills). Once the network is trained, we use it to transform all of our course data (both training and test) into this new feature space.

7 User Interface

To enhance the accessibility and user-friendliness of our project, we have developed a user interface for our project. This interface plays a crucial role in enabling users to easily interact with and comprehend the various machine learning algorithms implemented in our system. For the creation of this interface, we have utilized Streamlit. Streamlit is a fast way to build and share data apps as it turns data scripts into shareable web apps in minutes, all in pure Python with no front-end experience required.
On the left side of the frontend screen, users are presented with a list of available algorithms for course recommendations. The options include Jaccard algorithm, TF-IDF, SVD method, NMF method, K-means, KNN, and Neural Network.
When a user selects the Jaccard algorithm, the interface displays a screenshot showcasing its functionality. Users can choose a specific course for which they want recommendations and indicate the desired number of recommended courses. Upon clicking the `Get Similar Courses` button, the system generates and displays the recommended courses based on the Jaccard algorithm.



Algorithm Selection

Choose an algorithm:

- [✓ Jaccard]
- TF-IDF in all data
- TF-IDF in Course Description column
- TF-IDF in Course Name column
- TF-IDF in Skills column
- SVD
- NMF
- K-Means
- Neural Networks

You have selected the Jaccard algorithm.

Enter the course index: 3

How many recommendations do you want?: 10

5 most similar courses for course at index 3 (Finance for Managers):

Managerial Accounting Fundamentals

The Language and Tools of Financial Analysis

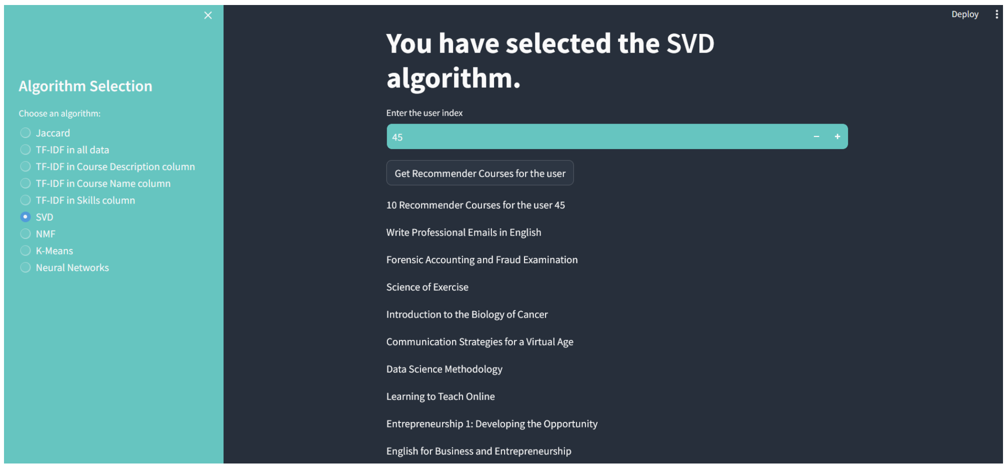
Managerial Accounting: Tools for Facilitating and Guiding Business Decisions

Management and financial accounting: Know your numbers 1

Fundamentals of financial and management accounting

Figure 5: Screenshot of the app's user interface, Jaccard similarity is selected

The process for the remaining algorithms (Singular Value Decomposition, Non-Negative Matrix Factorization, TF-IDF for diffrent inputs, Jaccard Similarity, K-Nearest Neighbors, K-Means, and Neural Networks) we've implemented is quite similar to the Jaccard Similarity. Below, we present additional screenshots depicting the functionality of some of these algorithms:



You have selected the SVD algorithm.

Enter the user index

45

10 Recommender Courses for the user 45

Write Professional Emails in English

Forensic Accounting and Fraud Examination

Science of Exercise

Introduction to the Biology of Cancer

Communication Strategies for a Virtual Age

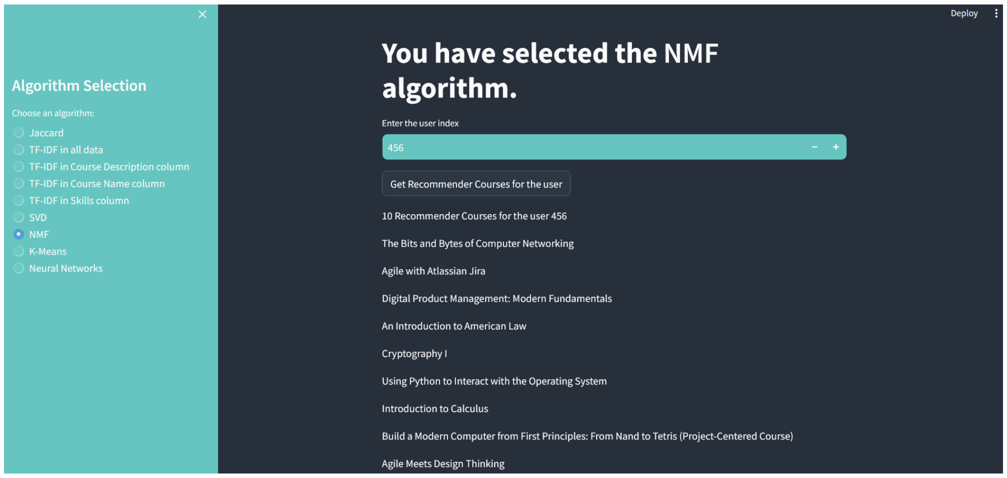
Data Science Methodology

Learning to Teach Online

Entrepreneurship 1: Developing the Opportunity

English for Business and Entrepreneurship

Figure 6: Screenshot of the app's user interface, SVD is selected, users can choose the index of the specific user for whom they want to receive personalized course recommendations, 10 recommended courses appears.



You have selected the NMF algorithm.

Enter the user index

456

10 Recommender Courses for the user 456

The Bits and Bytes of Computer Networking

Agile with Atlassian Jira

Digital Product Management: Modern Fundamentals

An Introduction to American Law

Cryptography I

Using Python to Interact with the Operating System

Introduction to Calculus

Build a Modern Computer from First Principles: From Nand to Tetris (Project-Centered Course)

Agile Meets Design Thinking

Figure 7: Screenshot of the app's user interface, NMF is selected, users can choose the index of the specific user for whom they want to receive personalized course recommendations, 10 recommended courses appears.

8 Conclusion and Future Work

In conclusion, our research and experimentation have showcased the effectiveness of diverse machine learning algorithms in enhancing the recommendation system for online courses on platforms like Coursera. Through the utilization of techniques such as **Singular Value Decomposition** (SVD), **Non-Negative Matrix Factorization** (NMF), **TF-IDF** coupled with Cosine Similarity, **Jaccard Similarity**, **K-Nearest Neighbors** (KNN), **K-Means**, and **Neural Networks**, we have successfully addressed the challenges associated with both collaborative and content-based filtering. Each algorithm contributes uniquely to the recommendation process, providing users with personalized and relevant course suggestions based on their preferences, past interactions, and skill sets. Moreover, **the development of a user-friendly interface using Streamlit** ensures a seamless interaction for users, making the recommendation system accessible and comprehensible.

While our current study has laid a solid foundation for advanced course recommendation systems, there are several avenues for future exploration and improvement. Firstly, incorporating **hybrid approaches** that combine the strengths of collaborative and content-based filtering could potentially yield more robust recommendations. Additionally, exploring **advanced deep learning architectures** and models for feature learning and representation extraction could further enhance the system's ability to capture intricate patterns within course data. Addressing scalability and real-time recommendation challenges, especially in large-scale platforms, would be crucial for practical deployment. As the field of machine learning continues to evolve, our work opens avenues for continual refinement and innovation in the domain of online course recommendations.

References

[1] Ehsan Aslanian, Mohammadreza Radmanesh, and Mahdi Jalili. Hybrid recommender systems based on content feature relationship. *IEEE Transactions on Industrial Informatics*, pages 1–1, 2016.

[2] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems. page 7–10, 2016.

[3] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. page 173–182, 2017.

[4] Hyeyoung Ko, Suyeon Lee, Yoonseo Park, and Anna Choi. A survey of recommendation systems: Recommendation models, techniques, and application fields. *Electronics*, 11, 2022.

[5] Kamal Berahmand Saman Forouzandeh and Mehrdad Rostami. Presentation of a recommender system with ensemble learning and graph embedding: a case on movielens. *Springer Nature 2020, Multimedia Tools and Applications*, 2020.