Rinex Internship Report

An Internship Report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

(Specialization in Artificial Intelligence & Machine Learning)

Submitted by

Dev Varma Rajasagi (VU21CSEN0300066)

RINEX

02 May, 2024 to 05 July, 2024

DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

GITAM (Deemed to be University)

VISAKHAPATNAM

2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM SCHOOL OF TECHNOLOGY GITAM (Deemed to be University)

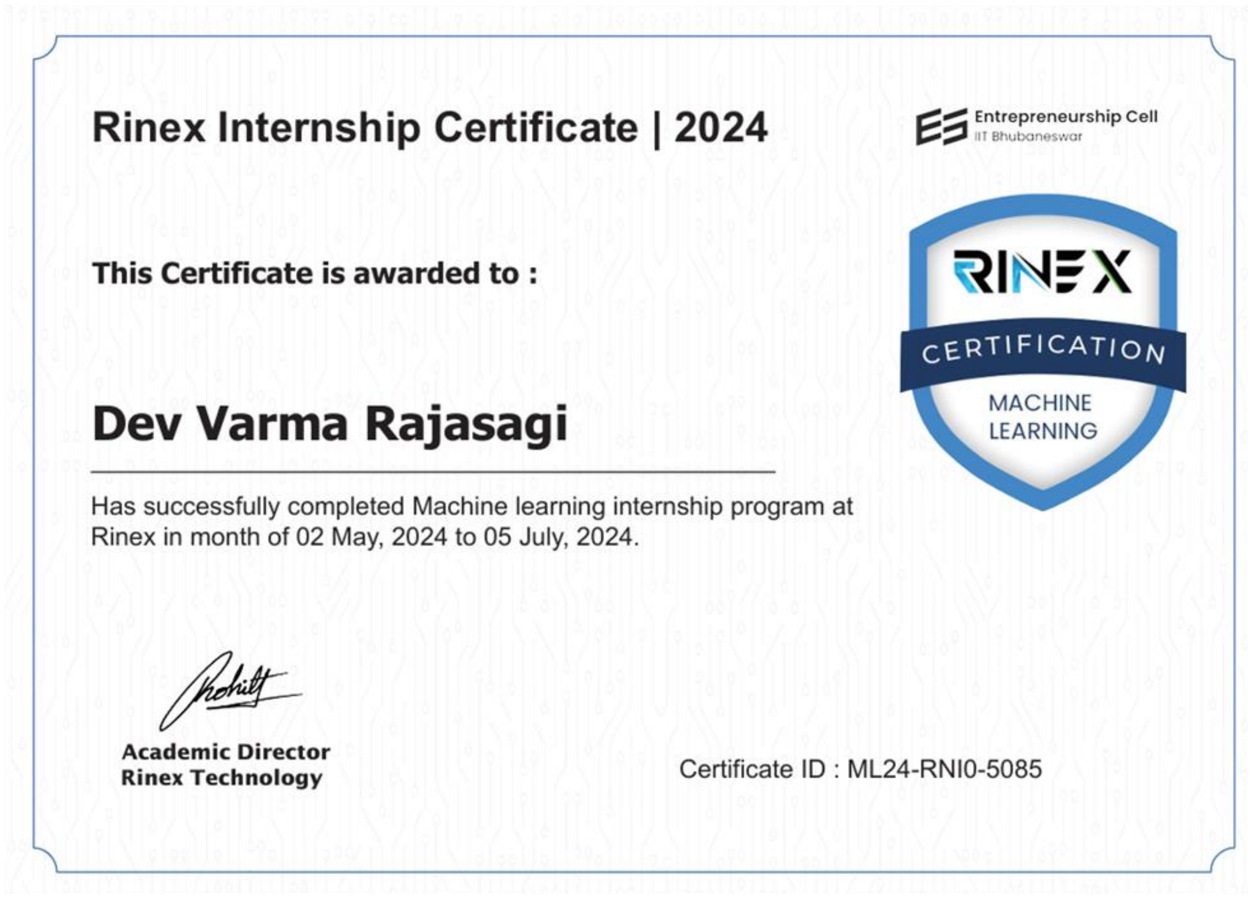
DECLARATION

I hereby declare that the internship report entitled “Rinex Internship Report” is an original work done in the Department of Computer Science and Engineering, GITAM School of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date: 10/10/2024

Registration No	Name	signature
VU21CSEN0300066	Dev Varma Rajasagi	

Internship Certificate



Rinex Internship Certificate | 2024

Entrepreneurship Cell
IIT Bhubaneswar

This Certificate is awarded to :

RINEX
CERTIFICATION
MACHINE LEARNING

Dev Varma Rajasagi

Has successfully completed Machine learning internship program at Rinex in month of 02 May, 2024 to 05 July, 2024.

[illegible]

Academic Director
Rinex Technology

Certificate ID : ML24-RNI0-5085

Acknowledgement

I would like to express my sincere gratitude to Rinex for providing me with the opportunity to intern in the field of Machine Learning. This internship has been an invaluable experience that has deepened my understanding of practical applications and reinforced my knowledge of machine learning concepts. I am especially grateful to my mentors and colleagues at Rinex, whose guidance, support, and insights were instrumental throughout my internship. Their encouragement and expertise not only enhanced my technical skills but also inspired me to approach problem-solving with creativity and critical thinking. Thank you once again to Rinex for this enriching experience.

Table of Contents

Image/Diagram Index

Abstract

This report outlines my machine learning project completed during my internship with Rinex. The aim of the project was to create a Course Recommendation System utilizing my skills in python, data analysis, machine learning, and data science. Course Recommendation System aims to improve user experience on online learning platforms by providing personalized course suggestions. The project's objective was to design a recommendation engine that assists users in identifying relevant courses based on their past interactions, preferences, and popular trends. A hybrid approach has been used, combining popularity-based recommendations and content-based recommendations. This hybrid approach allows the model to strike a balance between suggesting popular courses and those based on individual interests and history.

For the popularity-based recommendations, a weighted scoring system was used, taking into account the number of subscribers and reviews for each course. Meanwhile, the content-based recommendation model analyzed course titles and subjects, and processed it through text preprocessing and vectorization techniques. The similarity scores between courses were then calculated using cosine similarity, allowing the system to suggest similar courses to users. Overall, Course Recommendation System has the ability to improve user experience by giving effective and accurate personalized course recommendations.

1. Introduction

In the technological age, where there is a sea of online courses available for educational purposes. It is difficult for users to understand what courses they should take up. Users have the burden of filtering through thousands of courses to find the one that matches their ideas and desires. This huge number of options can be overwhelming for the users, and can cause great confusion. This is where a personalized course recommendation system can come in and help the users streamline this process, and ease the confusion. Choosing the right courses can be based on personal interests, values, skill levels, and goals. Thus, the course recommendation should take into account these various factors and give accurate and effective course recommendations. This system must make the time-consuming and difficult process of selecting courses easier. Recommendation systems play a vital role on e-learning platforms, offering users personalized suggestions tailored to their preferences and past activities. This internship project focuses on developing one such **C**ourse Recommendation System designed to enhance user experience by providing personalized course recommendations.

Throughout this internship, my role involved data preprocessing, applying recommendation algorithms, and designing a user-friendly interface. This experience not only strengthened my technical skills but also deepened my understanding of user-centered design in recommendation systems. This report outlines the techniques, results, and future scope of the project, providing a comprehensive overview of the practical and technical knowledge gained during the internship.

1.1. Internship Objectives

- Develop a Functional Recommendation System: Build a system that recommends courses based on popularity and content similarity, enhancing user experience by suggesting relevant courses.
- Apply Data Preprocessing Techniques: Gain hands-on experience in cleaning, preprocessing, and transforming raw data for analysis and model building.
- Explore Machine Learning and Data Science Concepts: Implement foundational AI techniques, such as vectorization and similarity measurement, in a real-world context.
- Design a User-Friendly Interface: Create a simple, interactive GUI for easy user interaction with recommendations, utilizing skills in both backend processing and frontend design.
- Strengthen Problem-Solving and Analytical Skills: Tackle challenges in data handling, feature engineering, and algorithm selection to refine technical problem-solving abilities.

1.2. Problem Statement

With the increasing volume of online courses, users often struggle to identify courses that meet their learning objectives. This project aims to develop a Course Recommendation System that suggests relevant courses to users based on popularity and content similarity. By analyzing key metrics such as subscriber count and review count, the system will recommend high-quality courses in an accessible, user-friendly format, addressing the challenge of content discovery in online learning platforms.

1.3. Technologies Used

- **Python**: The primary programming language used for the entirety of this project, due to its effectiveness with GUI design and data-processing capabilities.
- **Jupyter Notebook**: An interactive tool for live code, data visualization, and documentation.
- **Pandas**: Used for data manipulation, cleaning, and preprocessing tasks to ensure a structured dataset for analysis.
- **Neattext**: A text preprocessing library for removing stop words and special characters from course titles, improving text analysis quality.
- **Scikit-learn**: Utilized for vectorizing text data with CountVectorizer and calculating **cosine similarity** for content-based recommendations.
- **Tkinter**: Python’s built-in GUI library, used to build an interactive application for displaying recommendations and handling user inputs.

1.4. Key Concepts

- **Data Collection and Preprocessing**: This step is crucial for preparing raw data for use in machine learning models. In an AI/ML sense, preprocessing transforms data to a consistent format, improves quality, and enhances feature relevance, all of which are essential for building accurate models. For text-based data, techniques like removing stop words and special characters help simplify input, reduce noise, and improve the relevance of features in the dataset.

- **Popularity-Based Recommendation**: This is a simple form of recommendation using aggregate metrics like user engagement to identify high-interest items. While not a complex ML model, it lays a foundation by providing baseline recommendations based on known behaviors. This approach is often combined with other machine learning algorithms in hybrid recommendation systems.
- **Content-Based Filtering Using Vectorization and Similarity Measurement**: A content-based filtering approach uses machine learning techniques to understand relationships between items based on their content features. Here, text fields like course_title and subject are transformed into numerical data using vectorization (specifically CountVectorizer), which creates a matrix representation that can be analyzed for similarities. Cosine similarity is then used to measure the closeness between courses in this vectorized space. This enables the AI to identify and recommend courses with similar characteristics, providing users with personalized suggestions.
- **Natural Language Processing (NLP) for Text Analysis**: NLP techniques are applied to preprocess course titles, ensuring that irrelevant words do not skew similarity calculations. These preprocessing steps, often the initial stages of NLP in recommendation systems, make the textual data cleaner and more meaningful, enhancing the machine learning model's ability to detect and recommend relevant courses.

2. Requirement Specification

The Requirement Specification for the Course Recommendation System includes both Functional and Non-Functional Requirements to ensure the project meets user needs and performs effectively. Specific software and tools are also required to design a system that meets the objectives set by the problem.

2.1. Functional Requirements

The system's core functionalities involve data collection, preprocessing, recommendation generation, and user interface design. Data collection includes loading the course dataset, cleaning the data by removing duplicates and handling any missing values, and performing text preprocessing on course titles for more accurate analysis. The recommendation functionality includes a Popularity-Based Recommendation System, where a popularity score is calculated for each course, and the system sorts courses by this score to suggest the most popular options. In the Content-Based Recommendation System, course descriptions are enhanced by combining title and subject fields, and the system calculates similarity scores using vectorization and cosine similarity to recommend courses similar to a selected course. A Graphical User Interface (GUI) allows users to interact with the system, selecting courses and receiving recommendations based on both popularity and content.

2.2. Non-Functional Requirements

The project also includes several non-functional requirements to ensure it meets usability, performance, and scalability standards. Usability is prioritized through a user-friendly GUI that features clear prompts and intuitive navigation, making course selection and recommendation viewing simple and efficient. Performance is essential, as the system should quickly compute popularity and similarity scores to provide recommendations with minimal delay. Scalability is also considered; the recommendation logic is designed to adapt to larger datasets or additional features, accommodating future growth or new data fields. To maintain the system's reliability and adaptability, maintainability is ensured by modular, well-documented code, facilitating future updates or modifications as needed.

2.3 Software and Tools

The project utilizes Python as the primary programming language, alongside several key libraries: Pandas for data manipulation, Neattext for text cleaning, and Scikit-Learn for vectorization (CountVectorizer) and similarity measurement (cosine similarity). Tkinter is employed to create the GUI for user interaction. The dataset used for this recommendation system is the Kaggle Udemy Courses Dataset, which provides detailed information on course titles, subjects, and other essential fields for building the recommendation logic.

3. Methodology

The Course Recommendation System employs two main recommendation strategies: Popularity-Based Recommendations and Content-Based Recommendations. The popularity-based approach highlights courses that have high user engagement, such as a large number of subscribers and reviews, while the content-based approach analyzes similarities in course titles and subjects to suggest relevant courses aligned with users' previous choices. This project was implemented in Python, utilizing libraries like Pandas for data handling, Scikit-Learn for text processing and vectorization, and Tkinter for building an interactive graphical user interface (GUI). The Kaggle dataset used contained fields such as course_title, num_subscribers, num_reviews, and subject, providing the foundation for creating meaningful recommendations.

3.1 Data Collection and Preprocessing

The project used a Kaggle dataset of Udemy courses, containing fields such as course_title, subject, num_subscribers, and num_reviews. Data preprocessing was essential to ensure accuracy, so duplicates were removed, and any missing values were handled. Additionally, the course_title field was cleaned using the neattext library, which removed stop words and special characters to improve the quality of text analysis and enable more effective recommendations.

3.2 Popularity-Based Recommendation System

A popularity score was calculated for each course using the formula: Popularity Score = 0.6 * num_subscribers + 0.4 * num_reviews. Courses were then sorted based on these popularity scores in descending order, with the top-ranked courses selected as popular recommendations for users.

3.3 Content-Based Recommendation System

To enable content-based recommendations, a new feature called title_subject was created by combining the course_title and subject fields, enriching each course's description. This combined feature was then vectorized using the CountVectorizer function from Scikit-Learn, which transformed it into a matrix representation for textual analysis. Cosine similarity was used to compute similarity scores between courses within this matrix, which allowed for personalized recommendations based on course descriptions.

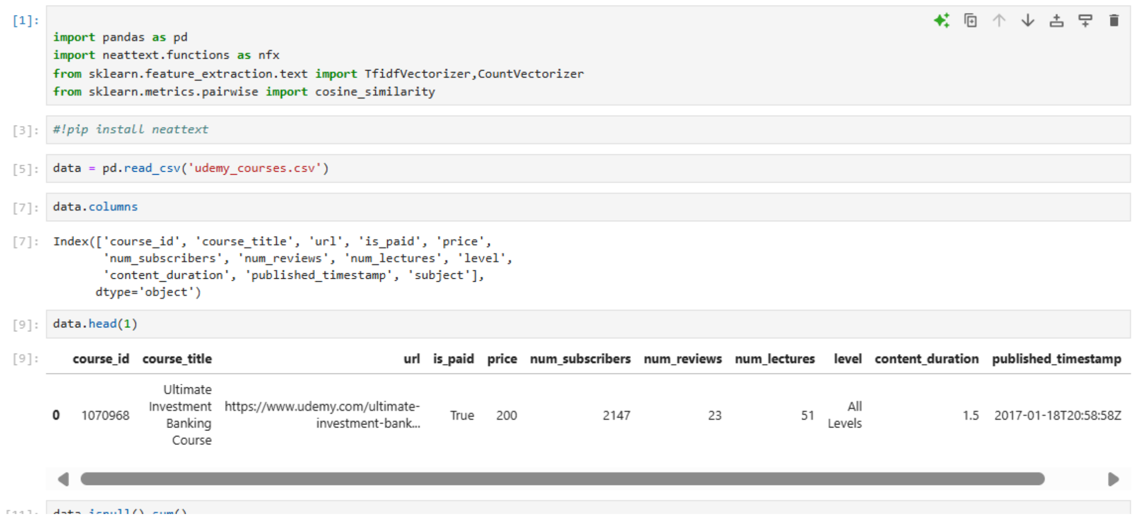
3.4 Graphical User Interface (GUI)

A simple and interactive GUI was created using Tkinter, allowing users to select a course and view recommendations based on popularity and similarity. The GUI interface displayed both popularity-based and similarity-based course recommendations, providing users with a versatile and user-friendly selection process.

3.5 Testing and Evaluation

To ensure the quality of recommendations, manual testing was performed to check the relevance and accuracy of the suggested courses, identifying any inconsistencies or outliers. Feedback was also gathered from users who interacted with the system to evaluate whether the recommendations met their expectations. This structured approach enabled the Course Recommendation System to effectively combine popularity and similarity-based methods, resulting in a practical and user-centered recommendation tool.

4. Code Snippets



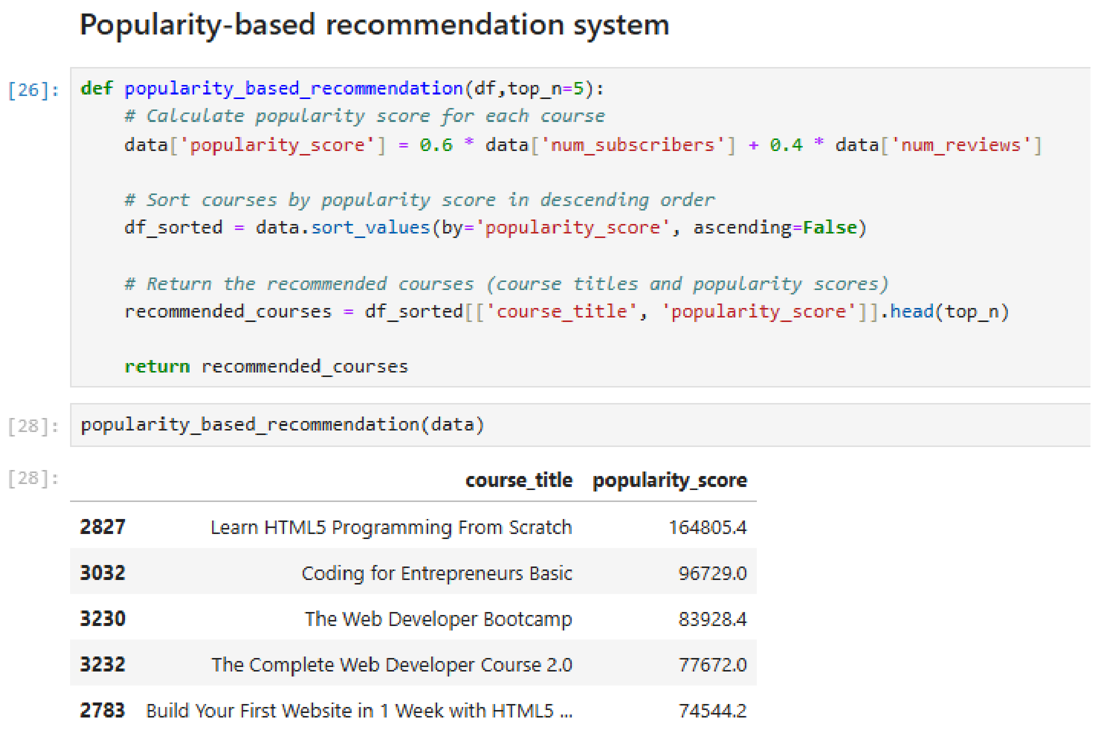
```
[1]:
import pandas as pd
import neattext.functions as nfx
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity

[3]: #!pip install neattext

[5]: data = pd.read_csv('udemy_courses.csv')

[7]: data.columns

[7]: Index(['course_id', 'course_title', 'url', 'is_paid', 'price',
        'num_subscribers', 'num_reviews', 'num_lectures', 'level',
        'content_duration', 'published_timestamp', 'subject'],
       dtype='object')

[9]: data.head(1)
```

[9]:

	course_id	course_title	url	is_paid	price	num_subscribers	num_reviews	num_lectures	level	content_duration	published_timestamp
0	1070968	Ultimate Investment Banking Course	https://www.udemy.com/ultimate-investment-bank...	True	200	2147	23	51	All Levels	1.5	2017-01-18T20:58:58Z

```
[11]: data.isnull().sum()
```

Figure 1.1:- Importing Required Libraries and Dataset



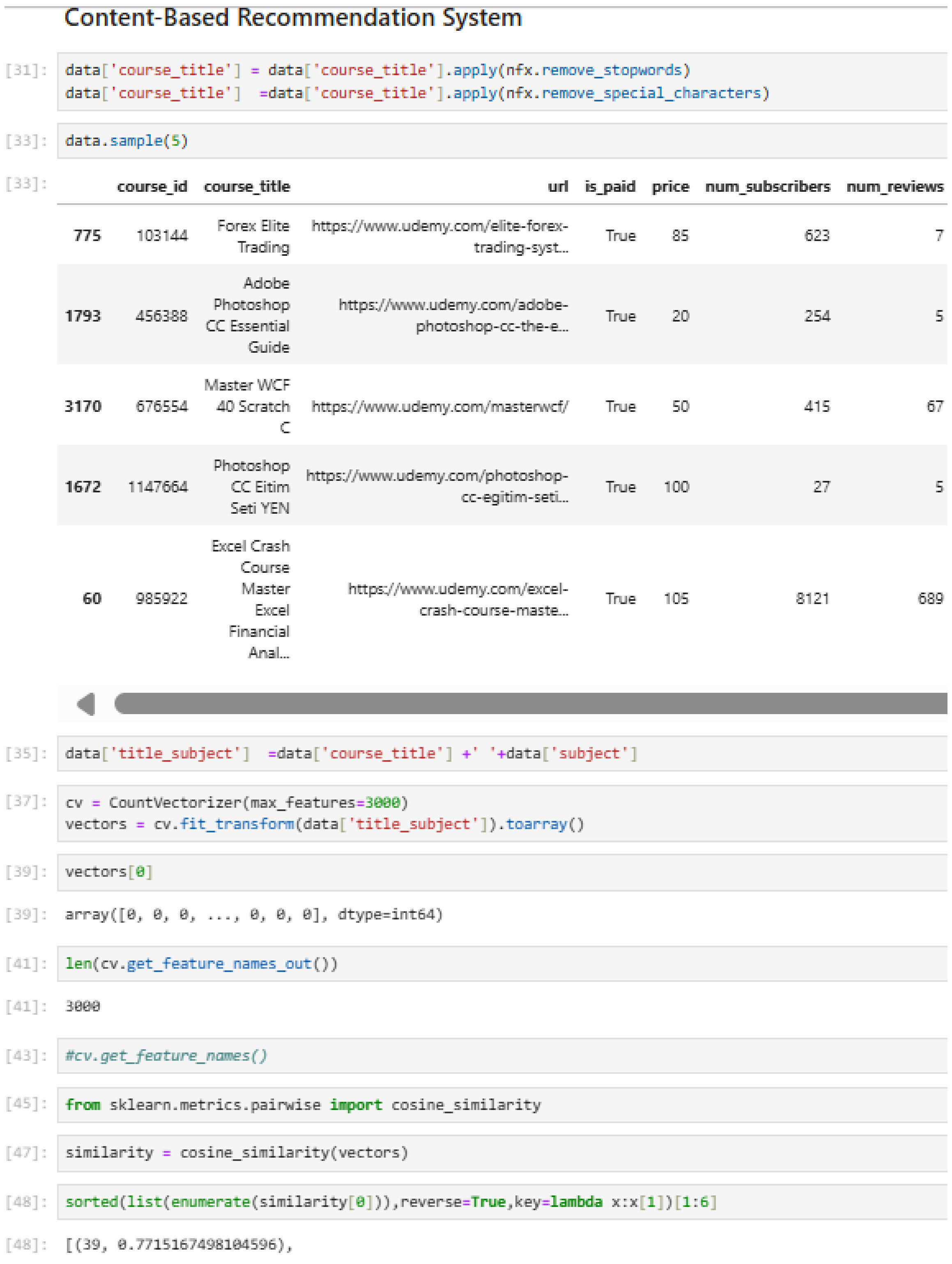
Popularity-based recommendation system

```
[26]: def popularity_based_recommendation(df,top_n=5):
          # Calculate popularity score for each course
          data['popularity_score'] = 0.6 * data['num_subscribers'] + 0.4 * data['num_reviews']

          # Sort courses by popularity score in descending order
          df_sorted = data.sort_values(by='popularity_score', ascending=False)

          # Return the recommended courses (course titles and popularity scores)
          recommended_courses = df_sorted[['course_title', 'popularity_score']].head(top_n)

          return recommended_courses

[28]: popularity_based_recommendation(data)
```

[28]:

	course_title	popularity_score
2827	Learn HTML5 Programming From Scratch	164805.4
3032	Coding for Entrepreneurs Basic	96729.0
3230	The Web Developer Bootcamp	83928.4
3232	The Complete Web Developer Course 2.0	77672.0
2783	Build Your First Website in 1 Week with HTML5 ...	74544.2

Figure 1.2:- Popularity-Based Filtering



Content-Based Recommendation System

```
[31]: data['course_title'] = data['course_title'].apply(nfx.remove_stopwords)
      data['course_title']  =data['course_title'].apply(nfx.remove_special_characters)
```

```
[33]: data.sample(5)
```

[33]:

	course_id	course_title	url	is_paid	price	num_subscribers	num_reviews
775	103144	Forex Elite Trading	https://www.udemy.com/elite-forex-trading-syst...	True	85	623	7
1793	456388	Adobe Photoshop CC Essential Guide	https://www.udemy.com/adobe-photoshop-cc-the-e...	True	20	254	5
3170	676554	Master WCF 40 Scratch C	https://www.udemy.com/masterwcf/	True	50	415	67
1672	1147664	Photoshop CC Eitim Seti YEN	https://www.udemy.com/photoshop-cc-egitim-seti...	True	100	27	5
60	985922	Excel Crash Course Master Excel Financial Anal...	https://www.udemy.com/excel-crash-course-maste...	True	105	8121	689

```
[35]: data['title_subject']  =data['course_title'] +' '+data['subject']
```

```
[37]: cv = CountVectorizer(max_features=3000)
      vectors = cv.fit_transform(data['title_subject']).toarray()
```

```
[39]: vectors[0]
```

```
[39]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

```
[41]: len(cv.get_feature_names_out())
```

```
[41]: 3000
```

```
[43]: #cv.get_feature_names()
```

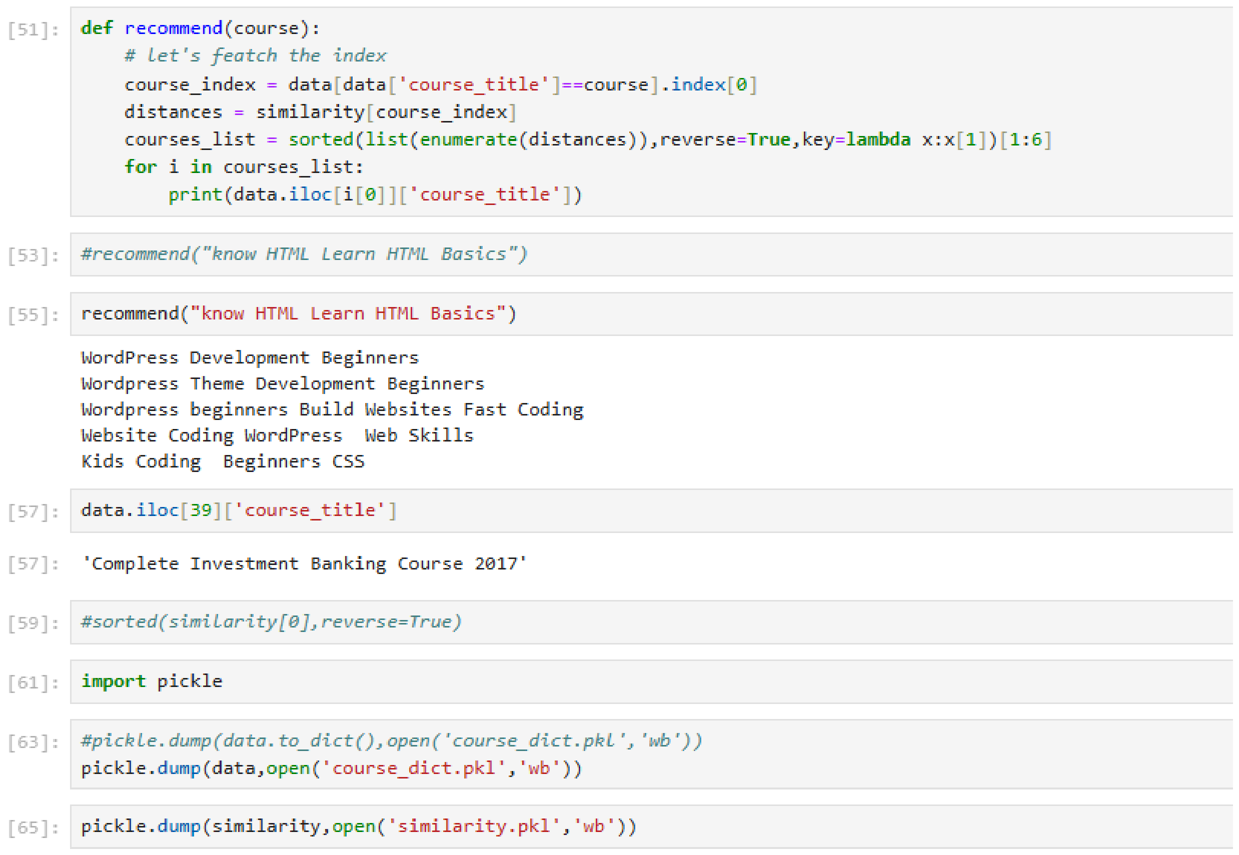
```
[45]: from sklearn.metrics.pairwise import cosine_similarity
```

```
[47]: similarity = cosine_similarity(vectors)
```

```
[48]: sorted(list(enumerate(similarity[0])),reverse=True,key=lambda x:x[1])[1:6]
```

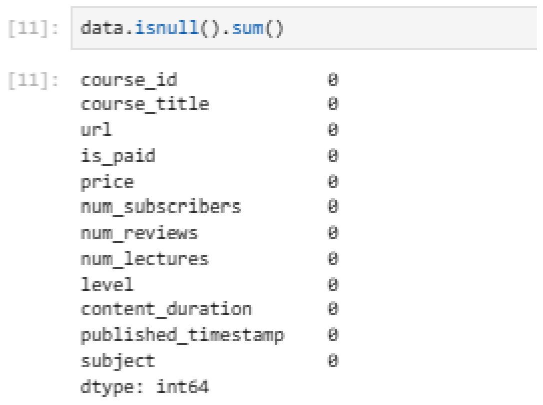
```
[48]: [(39, 0.7715167498104596),
```

Figure 1.3:- Content-Based Filtering



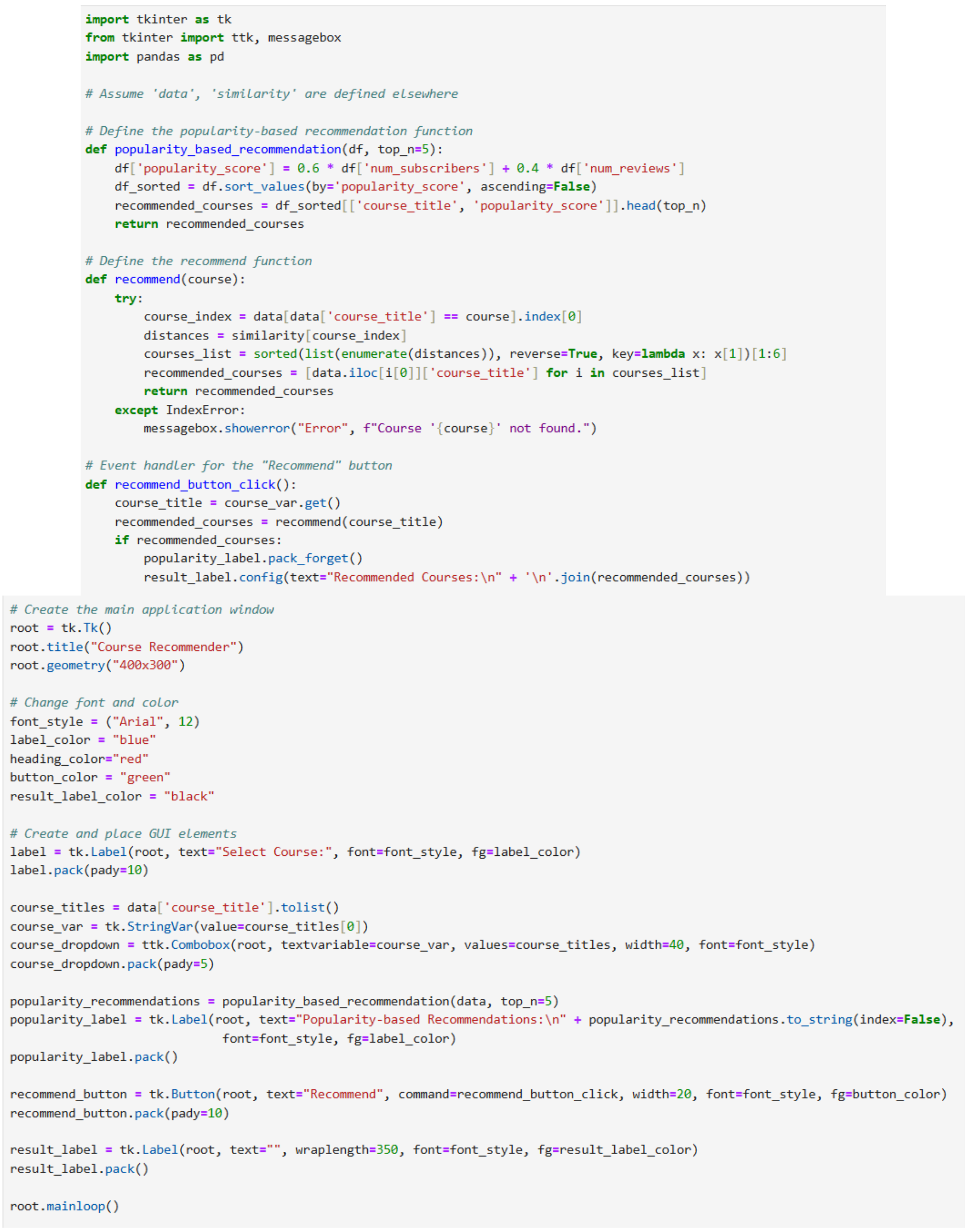
```
[51]: def recommend(course):
          # let's featch the index
          course_index = data[data['course_title']==course].index[0]
          distances = similarity[course_index]
          courses_list = sorted(list(enumerate(distances)),reverse=True,key=lambda x:x[1])[1:6]
          for i in courses_list:
              print(data.iloc[i[0]]['course_title'])

[53]: #recommend("know HTML Learn HTML Basics")

[55]: recommend("know HTML Learn HTML Basics")

      WordPress Development Beginners
      Wordpress Theme Development Beginners
      Wordpress beginners Build Websites Fast Coding
      Website Coding WordPress  Web Skills
      Kids Coding  Beginners CSS

[57]: data.iloc[39]['course_title']

[57]: 'Complete Investment Banking Course 2017'

[59]: #sorted(similarity[0],reverse=True)

[61]: import pickle

[63]: #pickle.dump(data.to_dict(),open('course_dict.pkl','wb'))
      pickle.dump(data,open('course_dict.pkl','wb'))

[65]: pickle.dump(similarity,open('similarity.pkl','wb'))
```

Figure 1.4:- Recommend Function and Dumping Model for Applications



```
[11]: data.isnull().sum()

[11]: course_id              0
      course_title           0
      url                    0
      is_paid                0
      price                  0
      num_subscribers        0
      num_reviews            0
      num_lectures           0
      level                  0
      content_duration       0
      published_timestamp    0
      subject                0
      dtype: int64
```

Figure 1.5:- Dataset Labels



```python
import tkinter as tk
from tkinter import ttk, messagebox
import pandas as pd

# Assume 'data', 'similarity' are defined elsewhere

# Define the popularity-based recommendation function
def popularity_based_recommendation(df, top_n=5):
    df['popularity_score'] = 0.6 * df['num_subscribers'] + 0.4 * df['num_reviews']
    df_sorted = df.sort_values(by='popularity_score', ascending=False)
    recommended_courses = df_sorted[['course_title', 'popularity_score']].head(top_n)
    return recommended_courses

# Define the recommend function
def recommend(course):
    try:
        course_index = data[data['course_title'] == course].index[0]
        distances = similarity[course_index]
        courses_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6]
        recommended_courses = [data.iloc[i[0]]['course_title'] for i in courses_list]
        return recommended_courses
    except IndexError:
        messagebox.showerror("Error", f"Course '{course}' not found.")

# Event handler for the "Recommend" button
def recommend_button_click():
    course_title = course_var.get()
    recommended_courses = recommend(course_title)
    if recommended_courses:
        popularity_label.pack_forget()
        result_label.config(text="Recommended Courses:\n" + '\n'.join(recommended_courses))
```

```python
# Create the main application window
root = tk.Tk()
root.title("Course Recommender")
root.geometry("400x300")

# Change font and color
font_style = ("Arial", 12)
label_color = "blue"
heading_color="red"
button_color = "green"
result_label_color = "black"

# Create and place GUI elements
label = tk.Label(root, text="Select Course:", font=font_style, fg=label_color)
label.pack(pady=10)

course_titles = data['course_title'].tolist()
course_var = tk.StringVar(value=course_titles[0])
course_dropdown = ttk.Combobox(root, textvariable=course_var, values=course_titles, width=40, font=font_style)
course_dropdown.pack(pady=5)

popularity_recommendations = popularity_based_recommendation(data, top_n=5)
popularity_label = tk.Label(root, text="Popularity-based Recommendations:\n" + popularity_recommendations.to_string(index=False),
                            font=font_style, fg=label_color)
popularity_label.pack()

recommend_button = tk.Button(root, text="Recommend", command=recommend_button_click, width=20, font=font_style, fg=button_color)
recommend_button.pack(pady=10)

result_label = tk.Label(root, text="", wraplength=350, font=font_style, fg=result_label_color)
result_label.pack()

root.mainloop()
```

Figure 1.6:- Graphical User Interface Design

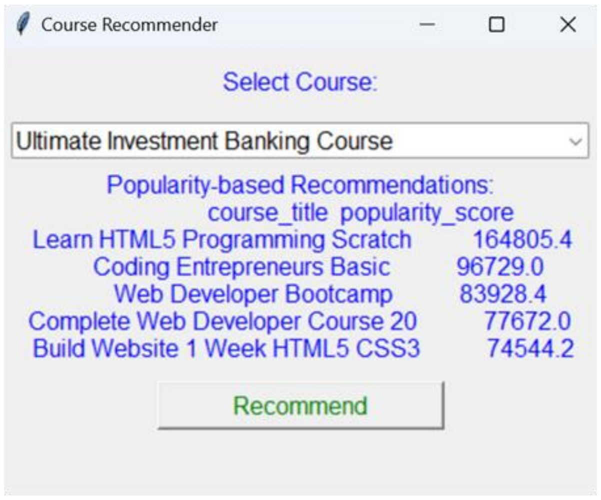
5. Results

The Course Recommendation System proved to be a valuable tool in delivering personalized and relevant recommendations, guiding users toward discovering new courses that aligned with their interests and learning goals. The popularity-based recommendation model successfully highlighted courses with high user engagement by leveraging metrics such as the number of subscribers and reviews. This approach helped users quickly identify widely favored courses, which often serve as a strong starting point for those exploring popular or highly rated content in a particular subject area.

Additionally, the content-based recommendation model enhanced the system's ability to suggest diverse options by identifying courses with similar topics or themes. By analyzing the course titles and subject fields, the content-based approach effectively grouped courses with overlapping themes, allowing the system to recommend courses based on users’ past interactions and preferences. This not only enriched the variety of recommendations but also added a layer of personalization, making the recommendations feel more relevant to each user.

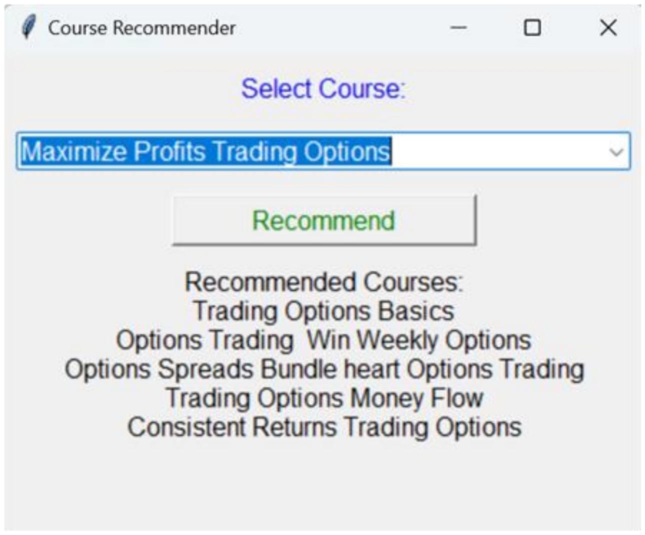
Feedback from users indicated that the system provided highly tailored recommendations that aligned with their interests and helped them find courses they might not have discovered otherwise. Users appreciated the balance between popular, high-engagement courses and those closely related to their unique interests, which made the system both efficient and enjoyable to use. Overall, the Course Recommendation

System demonstrated its effectiveness in offering a seamless and personalized user experience, with both popularity-based and content-based models contributing to a holistic and satisfying recommendation process.



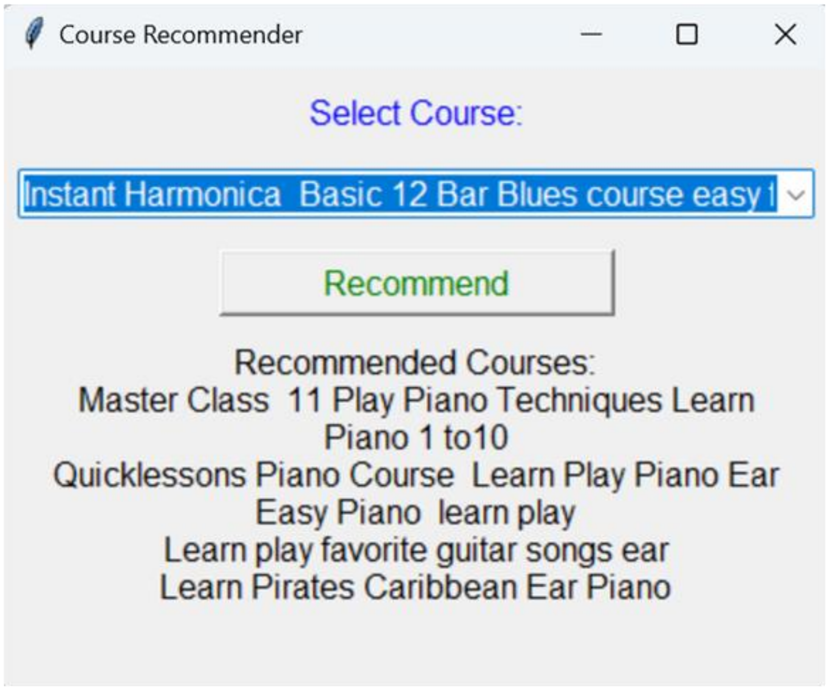
```
Popularity-based Recommendations:
                    course_title  popularity_score
Learn HTML5 Programming Scratch       164805.4
   Coding Entrepreneurs Basic         96729.0
     Web Developer Bootcamp           83928.4
Complete Web Developer Course 20       77672.0
Build Website 1 Week HTML5 CSS3        74544.2
```

Figure 1.7:- GUI



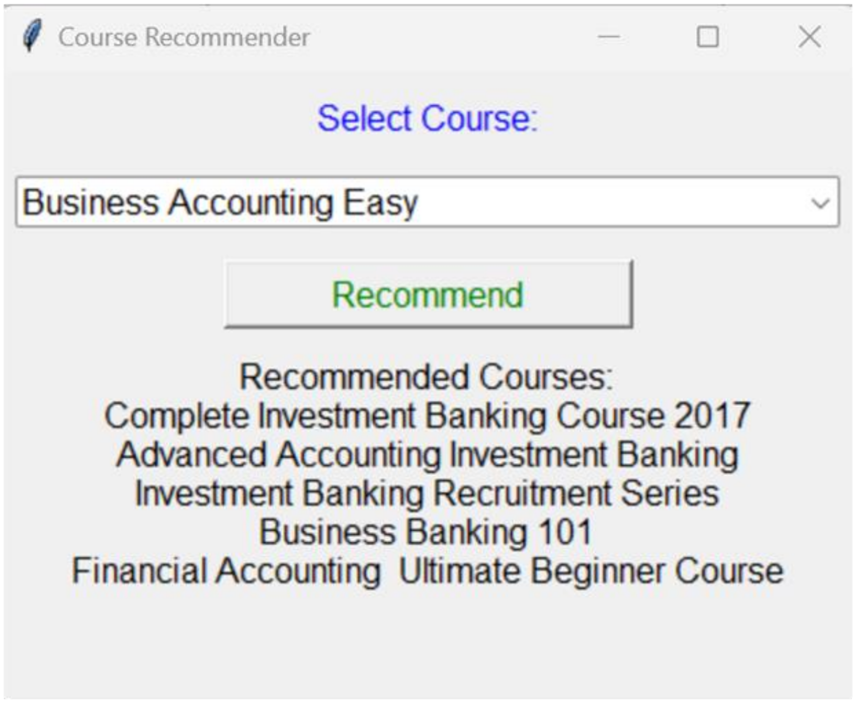
```
Recommended Courses:
Trading Options Basics
Options Trading  Win Weekly Options
Options Spreads Bundle heart Options Trading
Trading Options Money Flow
Consistent Returns Trading Options
```

Figure 1.8:- Sample Output 1



```
Instant Harmonica  Basic 12 Bar Blues course easy t
Recommended Courses:
Master Class  11 Play Piano Techniques Learn
Piano 1 to10
Quicklessons Piano Course  Learn Play Piano Ear
Easy Piano  learn play
Learn play favorite guitar songs ear
Learn Pirates Caribbean Ear Piano
```

Figure 1.9:- Sample Output 2



```
Business Accounting Easy
Recommended Courses:
Complete Investment Banking Course 2017
Advanced Accounting Investment Banking
Investment Banking Recruitment Series
Business Banking 101
Financial Accounting  Ultimate Beginner Course
```

Figure 2.0:- Sample Output 3

6. Conclusion

The development of the Course Recommendation System offered deep insights into the practical applications of machine learning and data processing in recommendation systems. Throughout this project, critical components such as data preprocessing, algorithm selection, and user-centered design emerged as essential aspects of creating a system capable of providing accurate, relevant, and personalized course recommendations. Data preprocessing was especially significant, as it underscored the value of handling missing values, eliminating duplicates, and structuring text data to ensure the recommendations' reliability and precision. Algorithm selection, including popularity-based and content-based filtering, emphasized how different models cater to unique recommendation goals, such as highlighting popular courses versus identifying closely related content.

Beyond immediate functionality, this project emphasized the importance of user-centered design to ensure that the recommendations aligned with user expectations and provided a smooth, intuitive experience. Implementing a simple yet effective GUI allowed users to interact easily with the system, making the process of discovering relevant courses enjoyable and efficient.

This project strengthened my skills in machine learning, data science, and application design, building a robust foundation for tackling more advanced work. I gained hands-on experience with data handling, NLP-based text processing, and user interface development—skills that are increasingly valuable in today's data-driven landscape. Looking ahead, there is considerable scope for expanding this system's capabilities. One promising direction for future improvements is the integration of collaborative filtering techniques, which would add a new dimension of personalization by considering the preferences of similar users, not just course attributes.

Further, incorporating user feedback loops would enable dynamic system updates based on user satisfaction and behavior, making recommendations more accurate over time. This adaptive approach, combined with advanced recommendation techniques, could increase the relevance and diversity of courses suggested to users.

In a broader sense, the Course Recommendation System can be extended beyond educational platforms. Similar frameworks could be applied in various domains, such as recommending job openings, books, movies, or even e-commerce products, where personalized, user-focused suggestions significantly enhance the customer experience. By continuously refining and scaling the model, this system has the potential to grow into a highly adaptable recommendation engine applicable across different fields, offering targeted and impactful recommendations.