



Courses Recommendation System

Advanced Data Mining Techniques to organize Coursera course

AIE323 || Data Mining

Under supervision of: Prof. Mohamed Abdelaziz

Project Team members

Student ID	Student Name
221100899	Sarah Mostafa Abdelrahman
221101186	Omar Tarek Abo Elela
221100071	Ahmed Abdelrahman Mohamed
221100389	Kareem Wael Mohamed
222102487	Aser Mohamed Ali
223108628	Shrouq Hesham salman

Fall 2024

➤ **Abstract:**

- The rapid growth of online learning platforms such as **Coursera** has resulted in a vast number of available courses, making it challenging for users to select the most relevant ones. This report presents a data mining project focused on developing a **Courses Recommendation System**. The system leverages **clustering algorithms**, **cosine similarity**, and **text preprocessing** to analyze course data and recommend courses tailored to user preferences. Through visualization techniques and **KMeans clustering**, the project successfully demonstrates an efficient way to suggest courses, enhancing the overall user experience.
- This project aims to analyze and organize **Coursera course** data using data mining and machine learning techniques. It involves developing a course recommendation system for Coursera based on text analysis and identifying similarities between courses. Additionally, the project includes data visualization using tools like **Streamlit** and **Matplotlib** to provide an interactive and visual interface that helps users explore patterns and trends within the data.

➤ **Introduction and related work:**

Course recommendation systems aim to **assist users in identifying relevant courses** based on their preferences, prior knowledge, and interests. In this project, we design a system utilizing data mining techniques, including clustering and text analysis, to create an efficient recommendation mechanism. The methodology incorporates advanced preprocessing and clustering algorithms, making it suitable for practical applications in educational platforms.

➤ **Aim:**

To design and implement a Course Recommendation System using clustering and similarity measures to help users identify relevant courses based on their preferences and interests.

➤ Objectives:

- To preprocess course data by applying **tokenization and stemming**.
- To use **KMeans clustering** to group courses with similar content and skills.
- To calculate **cosine similarity** for identifying courses closely related to user preferences.
- To visualize the data for better understanding and evaluation of clustering results.
- To recommend courses efficiently by integrating the similarity matrix and clustering results.

➤ Scope / Applicability:

- **Scope:** The system is applicable to e-learning platforms like Coursera, academic institutions, and corporate training programs.
- **Applicability:**
 - **Students:** Personalized course suggestions for academic and personal growth.
 - **Professionals:** Relevant courses for skill enhancement and career advancement.
 - **Institutions:** Improved course visibility and user engagement.

➤ Purpose:

The primary purpose of this project is to **simplify the course selection process** for users by recommending courses that align with their learning goals. By analyzing course attributes and user preferences, the system provides targeted recommendations, improving user satisfaction and platform engagement.

➤ **Solution (Proposal):**

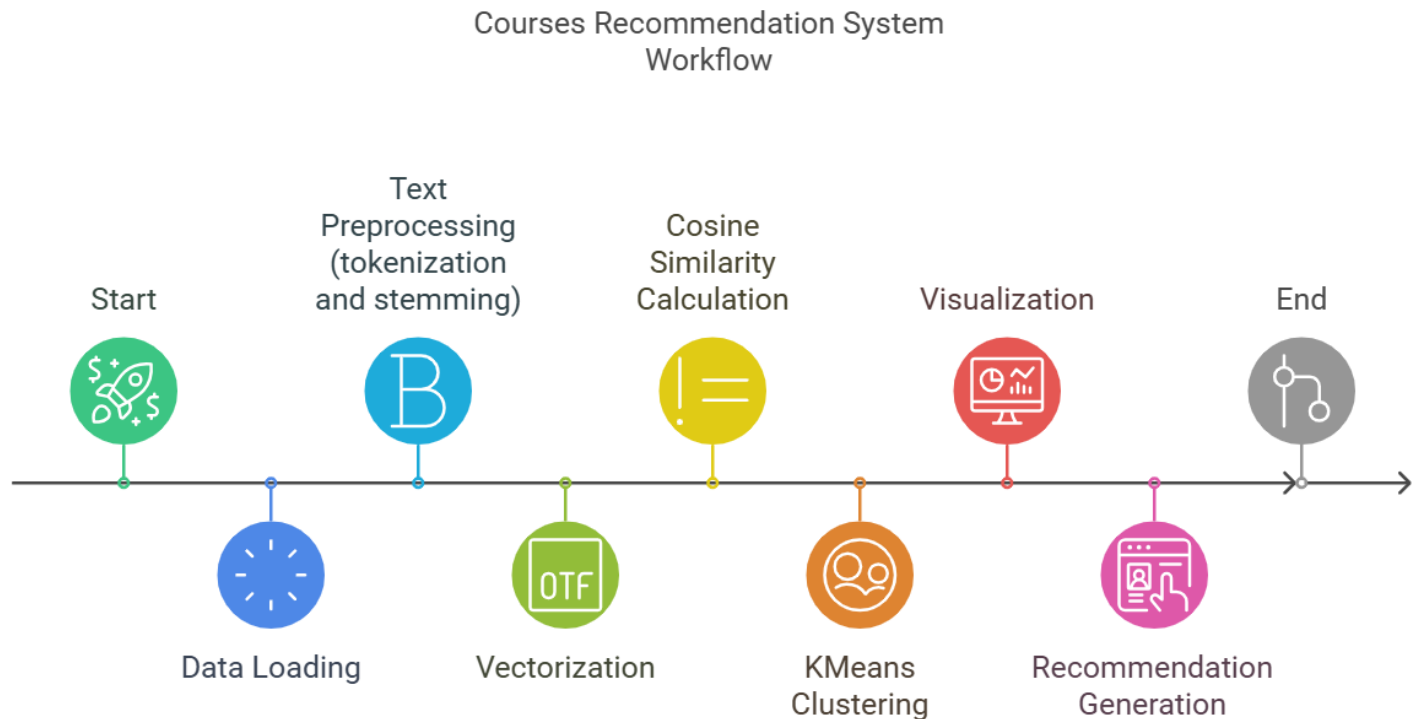
The solution involves building a recommendation system with the following steps:

1. **Data Loading:** Importing course data for analysis.
2. **Text Preprocessing:** Applying tokenization and stemming to clean and normalize textual data.
3. **Feature Engineering:** Combining tokens to create a unified representation of course content.
4. **Vectorization:** Converting textual data into numerical vectors using CountVectorizer.
5. **Cosine Similarity Calculation:** Computing similarity between courses to identify related ones.
6. **KMeans Clustering:** Grouping courses into clusters to enhance recommendation accuracy.
7. **Visualization:** Using plots and heatmaps to evaluate data distribution and clustering results.
8. **Recommendation:** Generating recommendations based on the similarity matrix and clustering outcomes.

➤ **Concepts of Data Mining Used:**

1. **Clustering:** Grouping similar courses using KMeans clustering.
2. **Cosine Similarity:** Measuring similarity between courses based on their vectorized representations.
3. **Text Preprocessing:** Tokenizing and stemming textual data to standardize it for analysis.
4. **Vectorization:** Using CountVectorizer to transform text into numerical data.
5. **Data Visualization:** Applying heatmaps, word clouds, and pairplots for insights into data structure and relationships.

➤ Implementation (Process):



➤ Resources Used:

- **Programming Language:** Python
- **Libraries:** Pandas, NLTK, Scikit-learn, Matplotlib, Seaborn, WordCloud
- **Tools:** Jupyter Notebook, Streamlit for visualization and deployment
- **Data Source:** Coursera course dataset (CSV file containing course names, descriptions, and skills)

➤ Dataset:

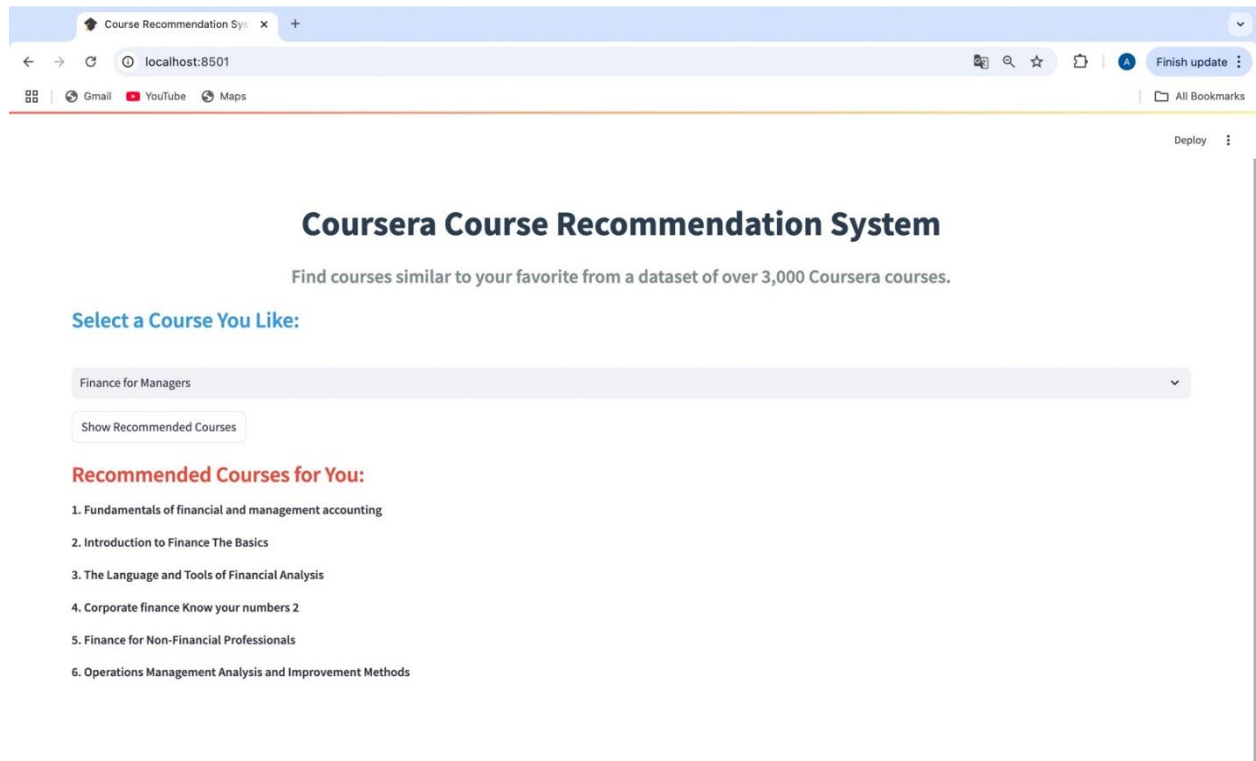
The dataset used in this project is a CSV file named "**Coursera.csv**" containing the following columns:

- **Course Name:** The title of the course.
- **Description:** A brief overview of the course content.
- **Skills:** Key skills taught in the course.

After Cleaning:

6

GUI:



Course Recommendation System

localhost:8501

Finish update

Deploy

Coursera Course Recommendation System

Find courses similar to your favorite from a dataset of over 3,000 Coursera courses.

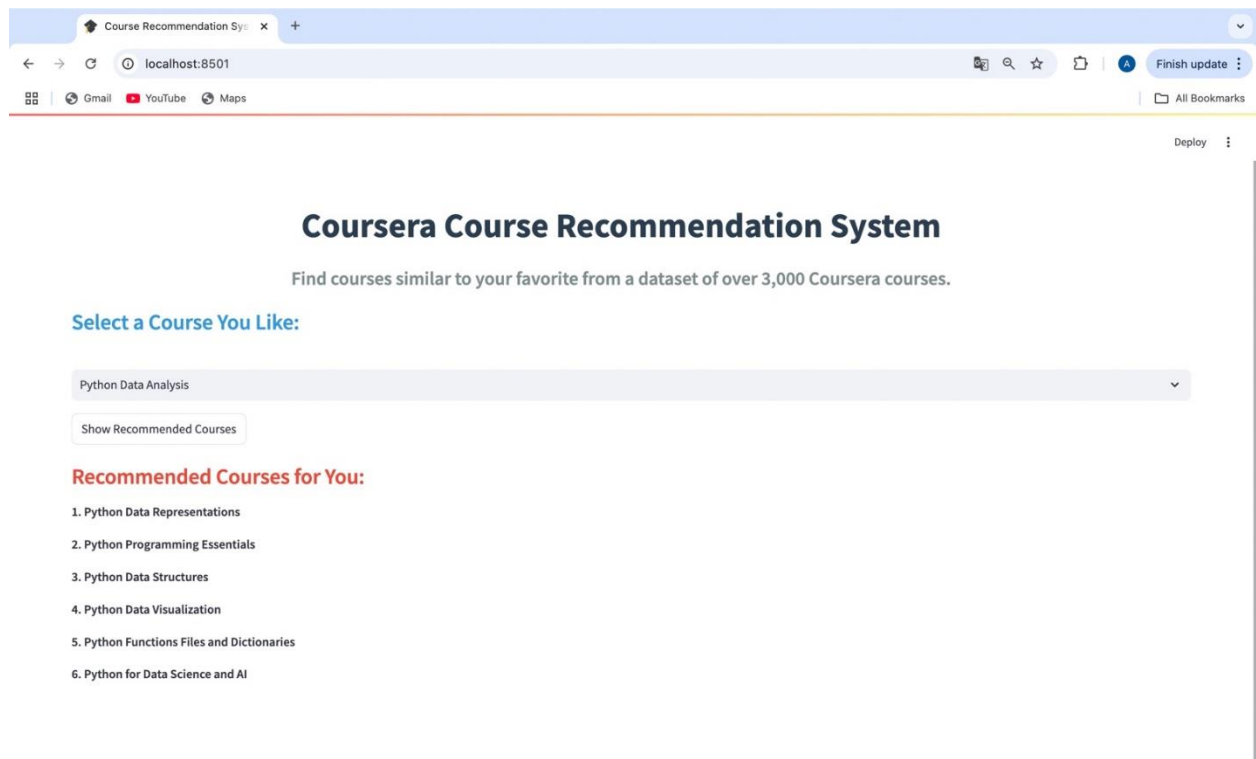
Select a Course You Like:

Finance for Managers

Show Recommended Courses

Recommended Courses for You:

1. Fundamentals of financial and management accounting
2. Introduction to Finance The Basics
3. The Language and Tools of Financial Analysis
4. Corporate finance Know your numbers 2
5. Finance for Non-Financial Professionals
6. Operations Management Analysis and Improvement Methods



Course Recommendation System

localhost:8501

Finish update

Deploy

Coursera Course Recommendation System

Find courses similar to your favorite from a dataset of over 3,000 Coursera courses.

Select a Course You Like:

Python Data Analysis

Show Recommended Courses

Recommended Courses for You:

1. Python Data Representations
2. Python Programming Essentials
3. Python Data Structures
4. Python Data Visualization
5. Python Functions Files and Dictionaries
6. Python for Data Science and AI

Steps:

1. Data Loading:

```
import pandas as pd
data = pd.read_csv("Coursera.csv")
```

2. Text Preprocessing:

- Tokenizing and stemming text data to prepare for analysis.
- Combining processed text into a unified column.

3. Vectorization:

- Converting text data into numerical vectors using CountVectorizer.

4. Cosine Similarity Calculation:

```
from sklearn.metrics.pairwise import cosine_similarity
similarity = cosine_similarity(vectors)
```

5. KMeans Clustering:

```
from sklearn.cluster import KMeans
kmeans_model = KMeans(n_clusters=5, random_state=42)
data['Cluster'] = kmeans_model.fit_predict(vectors)
```

6. Visualization:

- Heatmaps, word clouds, and cluster distribution plots to analyze results.

7. Recommendation Generation:

```
def recommend(course):
    index = data[data['course_name'] == course].index[0]
    distances = sorted(list(enumerate(similarity[index])),
                        reverse=True, key=lambda x: x[1])
    recommended_courses = [data.iloc[i[0]].course_name for i in distances[1:6]]
    return recommended_courses
```


➤ Courses Recommendation System main code:

- Loading the dataset
- Opening the Streamlit GUI window

```
import os
import pickle
import streamlit as st
import pandas as pd

# Load pre-trained data
courses_list = pd.DataFrame(pickle.load(open('course_list.pkl', 'rb')))
similarity = pickle.load(open('similarity.pkl', 'rb'))

# Function to recommend courses
def recommend(course):
    try:
        index = courses_list[courses_list['course_name'] == course].index[0]
        distances = sorted(list(enumerate(similarity[index])), reverse=True,
key=lambda x: x[1])
        recommended_course_names = [courses_list.iloc[i[0]].course_name for i in
distances[1:7]]
        return recommended_course_names
    except IndexError:
        return ["Course not found in the dataset. Please try another."]

# <==== Streamlit UI =====>
st.set_page_config(page_title="Course Recommendation System", layout="wide",
page_icon="📖")

# Page title and subtitle
st.markdown(
    """
    <div style="text-align: center;">
        <h1 style="color: #2c3e50;">Coursera Course Recommendation System</h1>
        <h4 style="color: #7f8c8d;">Find courses similar to your favorite from a
dataset of over 3,000 Coursera courses.</h4>
    </div>
    """,
    unsafe_allow_html=True
)
```

```
# Dropdown for selecting a course
st.markdown("<h3 style='color: #3498db;'>Select a Course You Like:</h3>",
unsafe_allow_html=True)
course_list = courses_list['course_name'].values
selected_course = st.selectbox("", course_list)

# Recommendation button and results display
if st.button('Show Recommended Courses'):
    st.markdown("<h3 style='color: #e74c3c;'>Recommended Courses for You:</h3>",
unsafe_allow_html=True)
    recommended_courses = recommend(selected_course)

    if len(recommended_courses) > 0 and "Course not found" not in
recommended_courses[0]:
        for idx, course in enumerate(recommended_courses):
            st.markdown(f"*{idx + 1}. {course}*")
    else:
        st.markdown(f"<p style='color: red;'>{recommended_courses[0]}</p>",
unsafe_allow_html=True)

# Footer
st.markdown(
    """
    <div style="text-align: center; margin-top: 50px;">
        <p style="color: #bdc3c7;">Copyright &copy; Coursera and respective
course owners.</p>
    </div>
    """,
    unsafe_allow_html=True
)
```

➤ Data Mining Approach:

- Loading the dataset
- Preprocessing the data
- cleaning and clustering data to an Excel file

```
import os
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.cluster import KMeans
import pickle
import warnings
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import word_tokenize
import nltk

# Ensure nltk punkt is downloaded
nltk.download('punkt')

# Suppress warnings about CPU count
os.environ["LOKY_MAX_CPU_COUNT"] = "4" # Set this to the number of cores you
want to use
warnings.filterwarnings("ignore", category=UserWarning,
module="joblib.externals.loky")

# Step 1: Load the dataset
data = pd.read_csv("Coursera.csv")

# Step 2: Preprocess the data

# Tokenization function
def tokenize(text):
    return word_tokenize(text)

# Tokenizing 'Course Name', 'Course Description', and 'Skills'
data['course_name_tokens'] = data['Course Name'].astype(str).apply(lambda x:
tokenize(x.lower()))
data['description_tokens'] = data['Course Description'].astype(str).apply(lambda
x: tokenize(x.lower()))
data['skills_tokens'] = data['Skills'].astype(str).apply(lambda x:
tokenize(x.lower()))
```

```
# Clean 'Skills' column - remove commas and convert to lowercase
data['tags'] = data['skills_tokens'].apply(lambda x: ' '.join(x))

# Clean 'Course Name' column - remove commas, special characters, and convert to lowercase
data['course_name_tokens'] = data['course_name_tokens'].apply(lambda x: [word for word in x if word.isalnum()])
data['description_tokens'] = data['description_tokens'].apply(lambda x: [word for word in x if word.isalnum()])

# Join the tokens back into a string for vectorization
data['tags'] = data['course_name_tokens'].apply(lambda x: ' '.join(x)) + ' ' +
data['description_tokens'].apply(lambda x: ' '.join(x)) + ' ' +
data['skills_tokens'].apply(lambda x: ' '.join(x))

# Rename 'Course Name' to 'course_name' for consistency
data.rename(columns={'Course Name': 'course_name'}, inplace=True)

# Remove duplicates based on course name
data.drop_duplicates(subset=['course_name'], inplace=True)

# Step 3: Create vectors for cosine similarity
cv = CountVectorizer(max_features=5000, stop_words='english') # Initialize vectorizer
vectors = cv.fit_transform(data['tags']).toarray() # Convert tags into vectors

# Step 4: Calculate cosine similarity
similarity = cosine_similarity(vectors) # Compute similarity matrix
pickle.dump(similarity, open('similarity.pkl', 'wb')) # Save similarity matrix

# Step 5: Perform KMeans clustering
kmeans_model = KMeans(n_clusters=5, random_state=42) # Initialize KMeans with 5 clusters
data['Cluster'] = kmeans_model.fit_predict(vectors) # Add cluster assignments to the dataset
pickle.dump(kmeans_model, open('courses.pkl', 'wb')) # Save the clustering model

# Step 6: Save the cleaned and clustered data to an Excel file
output_file = "Cleaned_Coursera_Data.xlsx"
data.to_excel(output_file, index=False) # Save to Excel
print(f"Cleaned and clustered data saved to {output_file}.")

# Step 7: Save course list for future use
pickle.dump(data.to_dict(), open('course_list.pkl', 'wb'))
```

```
print("Course list and similarity data have been saved.")
```

➤ Output:

```
[nltk_data] Downloading package punkt to
[nltk_data] /Users/ahmedrahmo1388/nltk_data...
[nltk_data] Package punkt is already up-to-date!
Cleaned and clustered data saved to Cleaned_Coursera_Data.xlsx.
Course list and similarity data have been saved.
[nltk_data] Downloading package punkt to
[nltk_data] /Users/ahmedrahmo1388/nltk_data...
[nltk_data] Package punkt is already up-to-date!
```

➤ Data Mining Approach:

```
preprocessesing text data by tokenizing and stemming techniques
```

```
import os

import pandas as pd
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import word_tokenize
import nltk

# Download necessary NLTK data
nltk.download('punkt')

# Initialize Porter Stemmer
stemmer = PorterStemmer()

def preprocess_text(text):
    """Tokenize and stem the input text."""
    tokens = word_tokenize(text.lower()) # Tokenize and convert to lowercase
    stemmed_tokens = [stemmer.stem(token) for token in tokens if
token.isalnum()] # Stem and remove non-alphanumeric tokens
    return ' '.join(stemmed_tokens)

# Example dataset (replace with actual data loading)
data = pd.read_csv("Coursera.csv")

# Preprocess relevant columns
```

```
data['Skills'] = data['Skills'].fillna('No Skills Listed').apply(preprocess_text)
data['Course Name'] = data['Course Name'].fillna('No Course Name
Provided').apply(preprocess_text)
```

➤ Output:

```
[nltk_data] Downloading package punkt to
[nltk_data] /Users/ahmedrahmo1388/nltk_data...
[nltk_data] Package punkt is already up-to-date!
```

- **Data Visualization**: Data visualization helps to better understand the structure of the data and the results of the clustering.

The following visualizations are used to analyze the dataset:

- **Missing Data Heatmap**: The Missing Data Heatmap visualizes the missing values in the dataset:
 - • **Yellow** represents **missing data**.
 - • **Dark** areas show where **data is present**.
 - This helps identify columns that require imputation or removal due to excessive missing data.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
from sklearn.cluster import KMeans

# Load your cleaned dataset (assuming it's already cleaned and saved as
'Cleaned_Coursera_Data.xlsx')
data = pd.read_excel("Cleaned.xlsx")
```

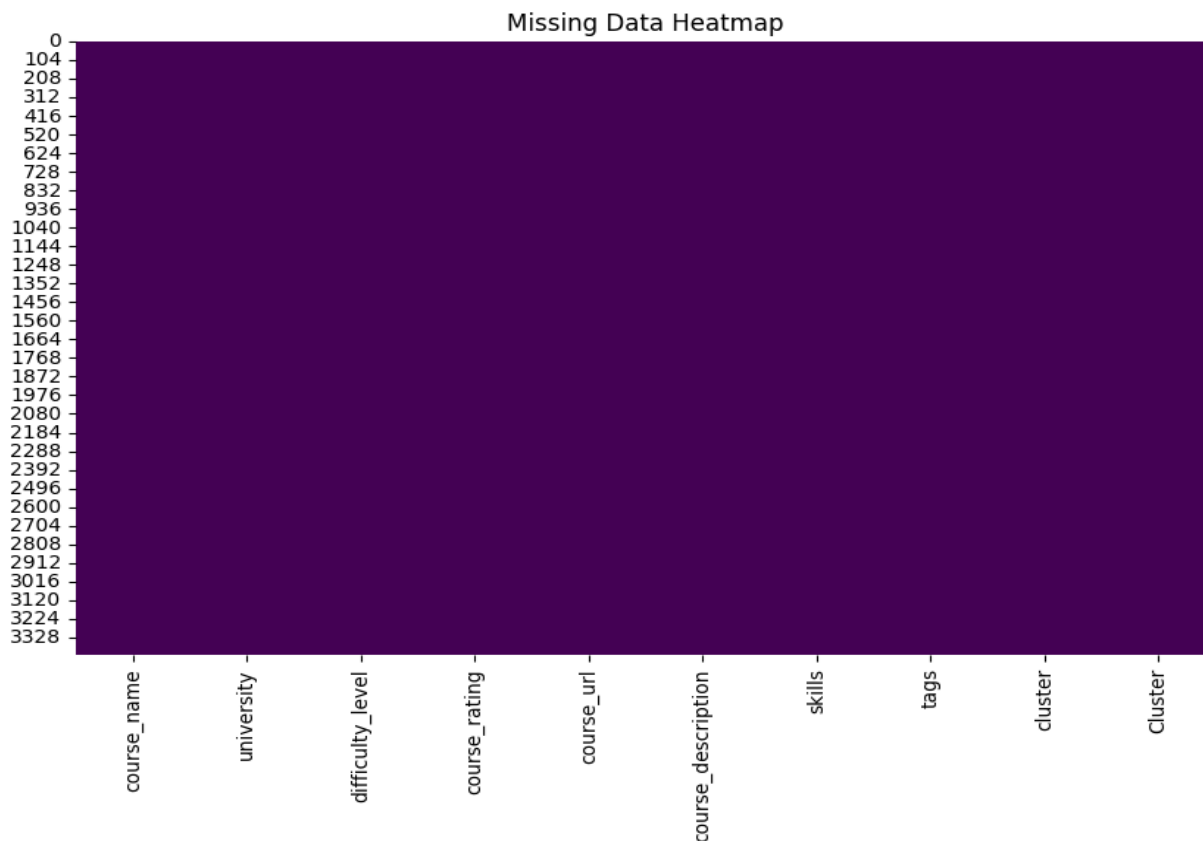


```
# Ensure the data is ready and doesn't have any missing 'Cluster' column
if 'Cluster' not in data.columns:
    # Apply KMeans clustering
    # Example: We'll use numeric columns for clustering, make sure to adjust
    based on your data
    numeric_columns = data.select_dtypes(include=['float64', 'int64']).columns
    if len(numeric_columns) > 0:
        kmeans = KMeans(n_clusters=5, random_state=42)
        data['Cluster'] = kmeans.fit_predict(data[numeric_columns])

# Step 1: Missing Data Visualization - Heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(data.isnull(), cbar=False, cmap='viridis')
plt.title('Missing Data Heatmap')
plt.show()

# Step 2: Distribution Plots for Numeric Data (If applicable)
numeric_columns = data.select_dtypes(include=['float64',
'int64']).columns.tolist()
```

➤ Output:



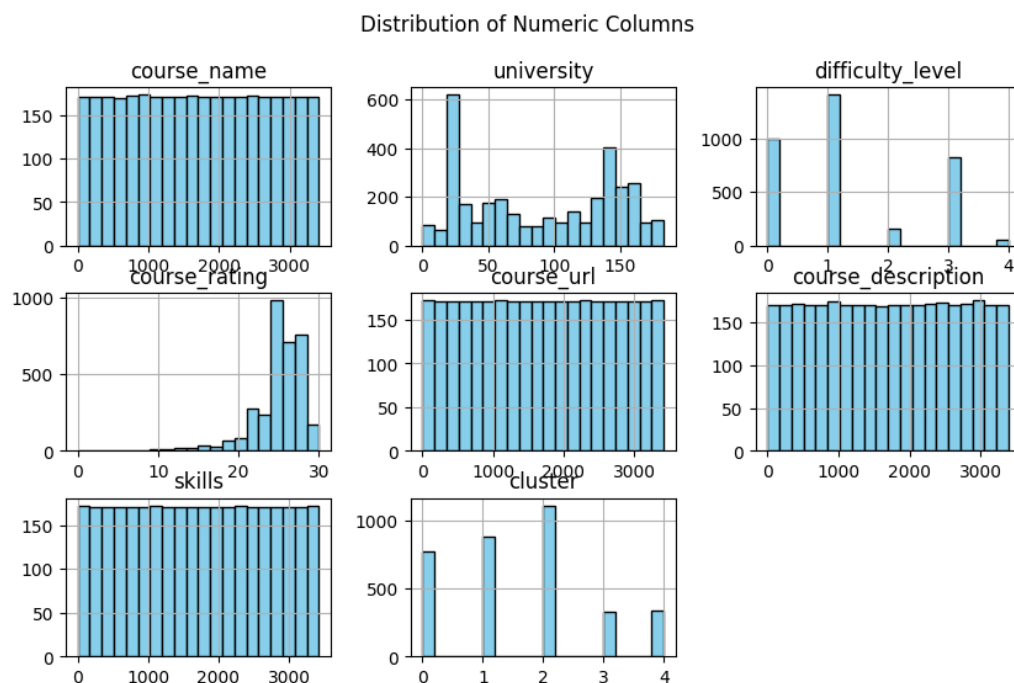
- **Distribution Plots for Numeric Data:** This plot generates histograms for numeric columns (e.g., ratings, duration) to show their distribution. It helps in understanding the range and frequency of numerical data points.

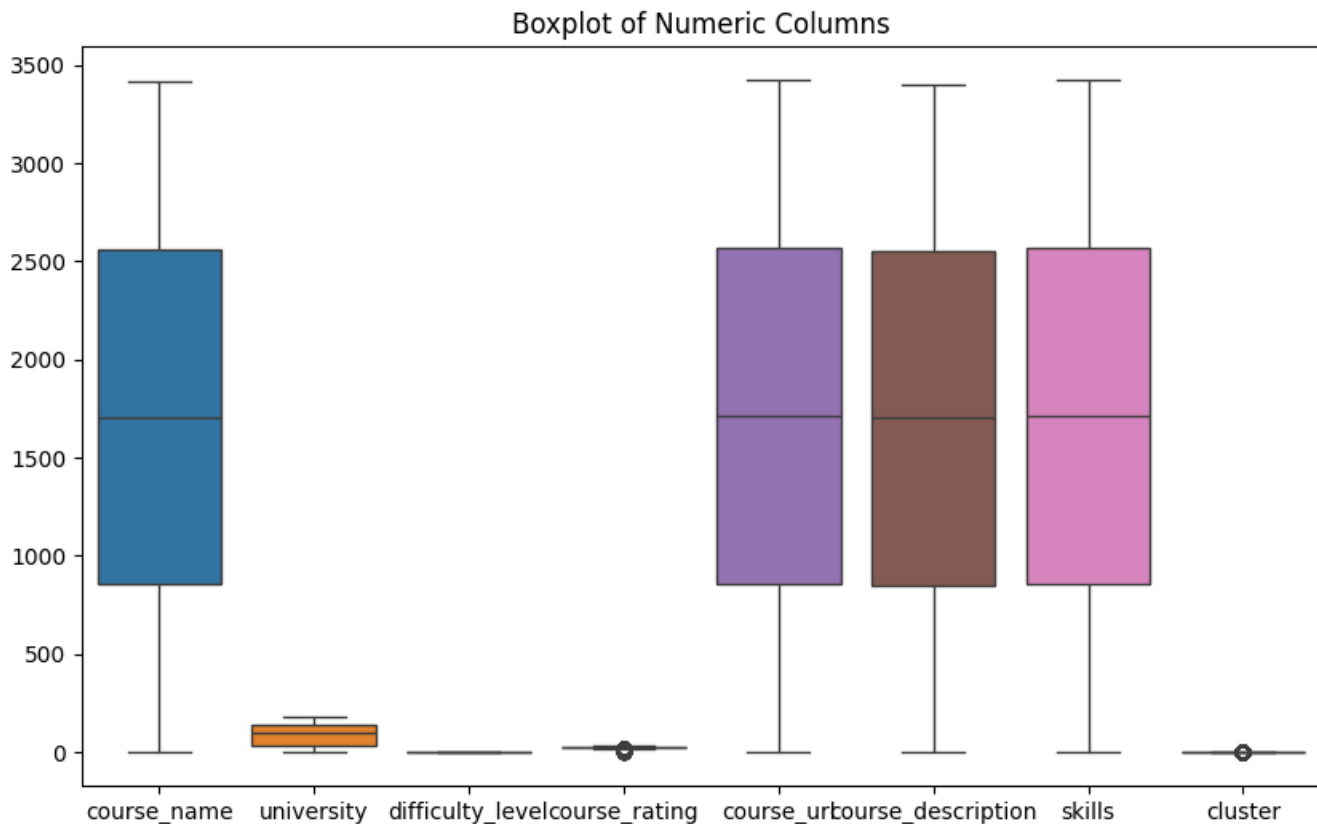
```
# Step 2: Distribution Plots for Numeric Data (If applicable)
numeric_columns = data.select_dtypes(include=['float64',
'int64']).columns.tolist()

if numeric_columns:
    # Histogram for Numeric Data
    data[numeric_columns].hist(figsize=(10, 6), bins=20, color='skyblue',
edgecolor='black')
    plt.suptitle('Distribution of Numeric Columns')
    plt.show()

    # Boxplot for Numeric Data
    plt.figure(figsize=(10, 6))
    sns.boxplot(data=data[numeric_columns])
    plt.title('Boxplot of Numeric Columns')
    plt.show()
```

➤ Output:





- **Word Cloud for Skills:** The Word Cloud visualizes the most frequent terms (skills) across all courses:
- Larger words indicate more frequent terms.
 - It highlights key skills or course topics, giving an overview of what students will learn.

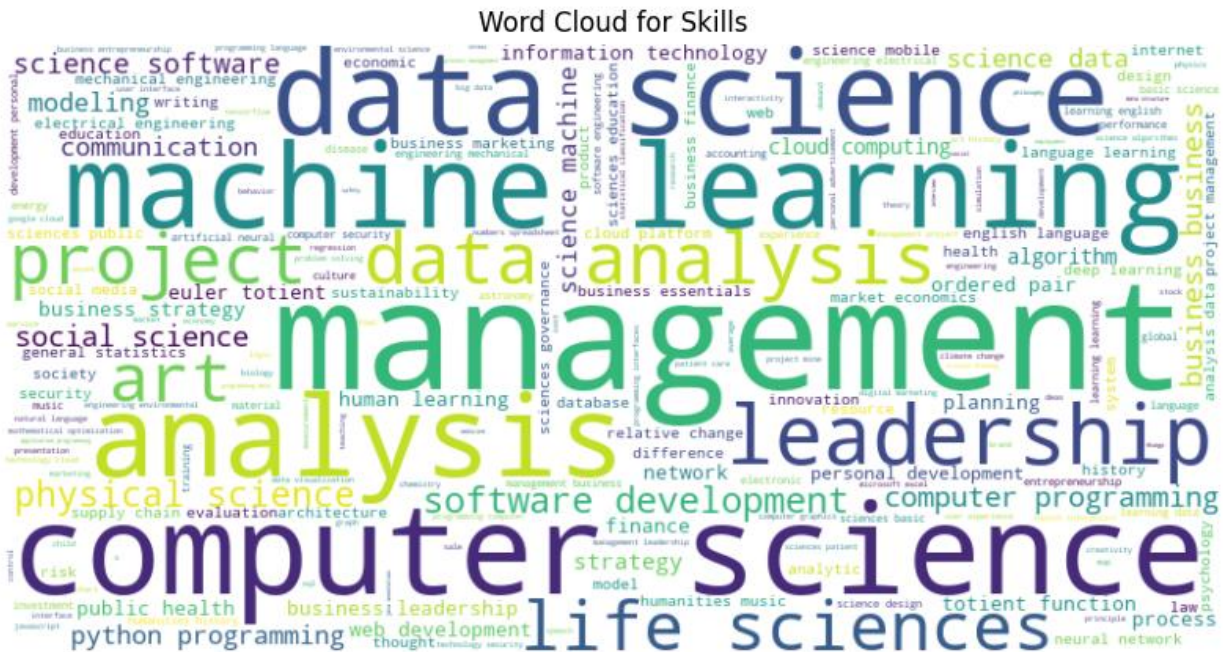
```
# Step 2: Distribution Plots for Numeric Data (If applicable)
numeric_columns = data.select_dtypes(include=['float64',
'int64']).columns.tolist()

if numeric_columns:
    # Histogram for Numeric Data
    data[numeric_columns].hist(figsize=(10, 6), bins=20, color='skyblue',
    edgecolor='black')
    plt.suptitle('Distribution of Numeric Columns')
    plt.show()

    # Boxplot for Numeric Data
    plt.figure(figsize=(10, 6))
    sns.boxplot(data=data[numeric_columns])
    plt.title('Boxplot of Numeric Columns')
```

```
plt.show()
```

 Output:

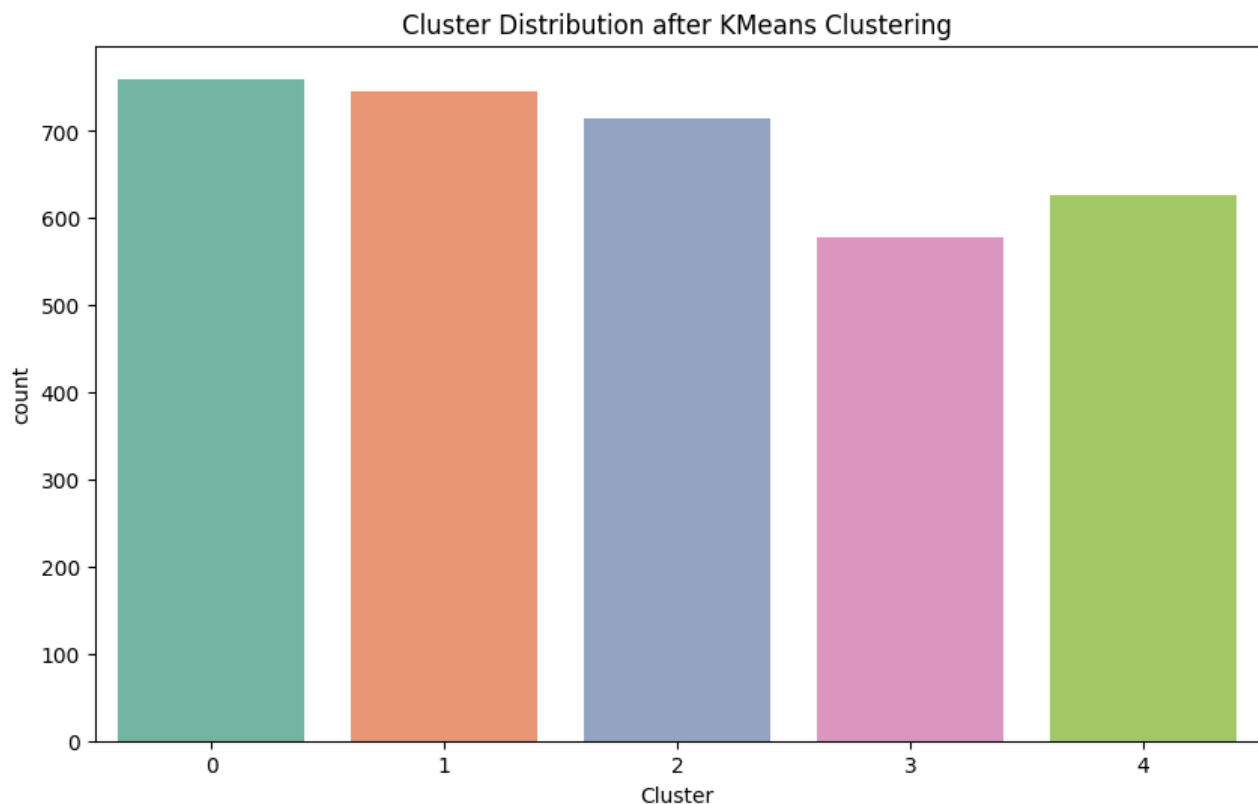


- **Cluster Distribution Countplot:** The Countplot visualizes the distribution of courses across different clusters:
 - It shows how many courses are in each cluster, helping us understand the size and balance of the clusters.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Step 4: Countplot for 'Cluster' (After KMeans clustering)
plt.figure(figsize=(10, 6))
sns.countplot(x='Cluster', data=data, hue='Cluster', palette='Set2',
              legend=False)
plt.title('Cluster Distribution after KMeans Clustering')
plt.show()
```

➤ Output:

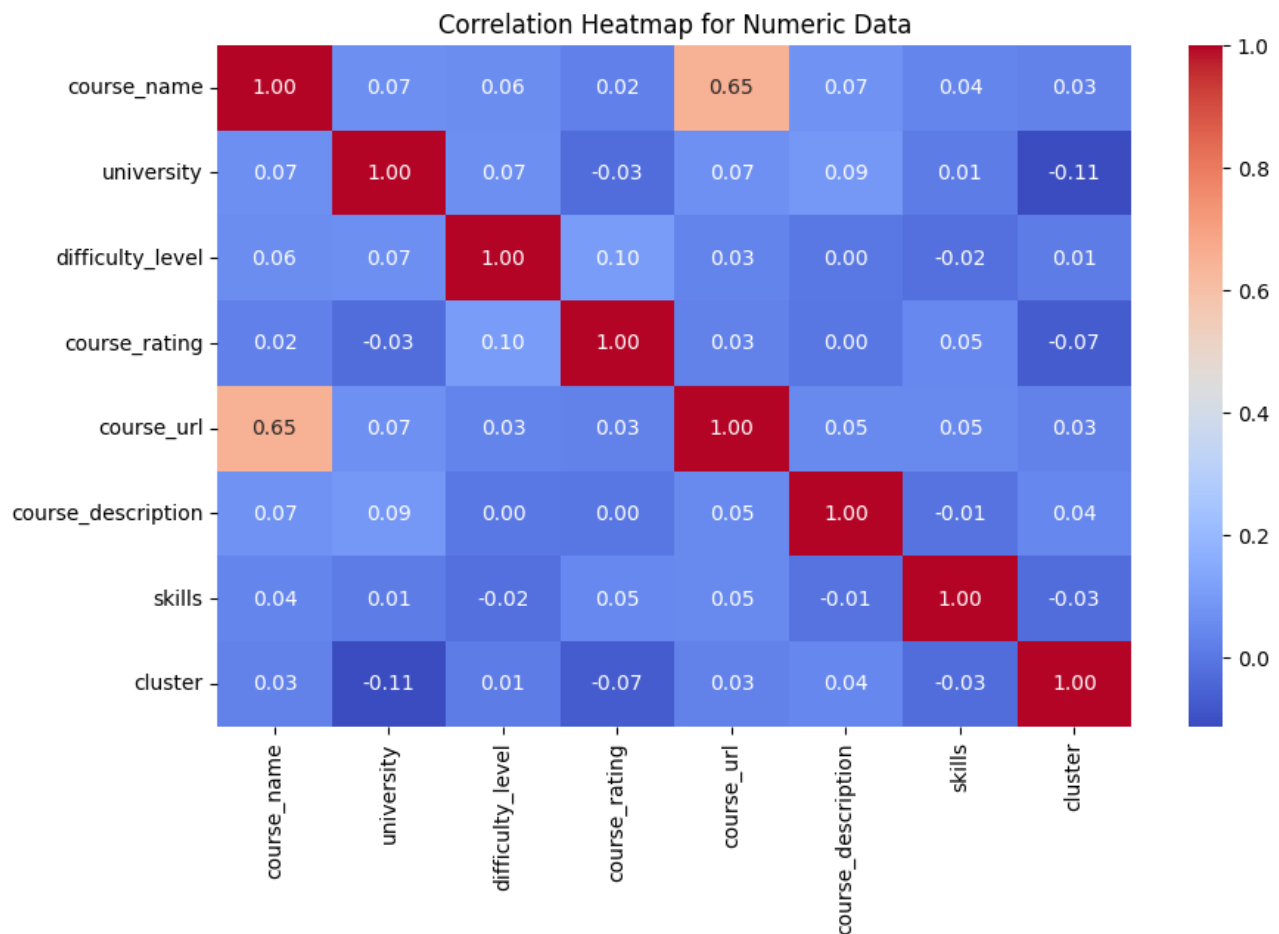


- **Correlation Heatmap for Numeric Data:** The Correlation Heatmap shows the relationships between different numeric features in the dataset (e.g., course duration vs. ratings). It highlights which features are strongly correlated.

Step 5: Correlation Heatmap for Numeric Columns (If available)

```
if numeric_columns:
    plt.figure(figsize=(10, 6))
    correlation_matrix = data[numeric_columns].corr()
    sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
    plt.title('Correlation Heatmap for Numeric Data')
    plt.show()
```

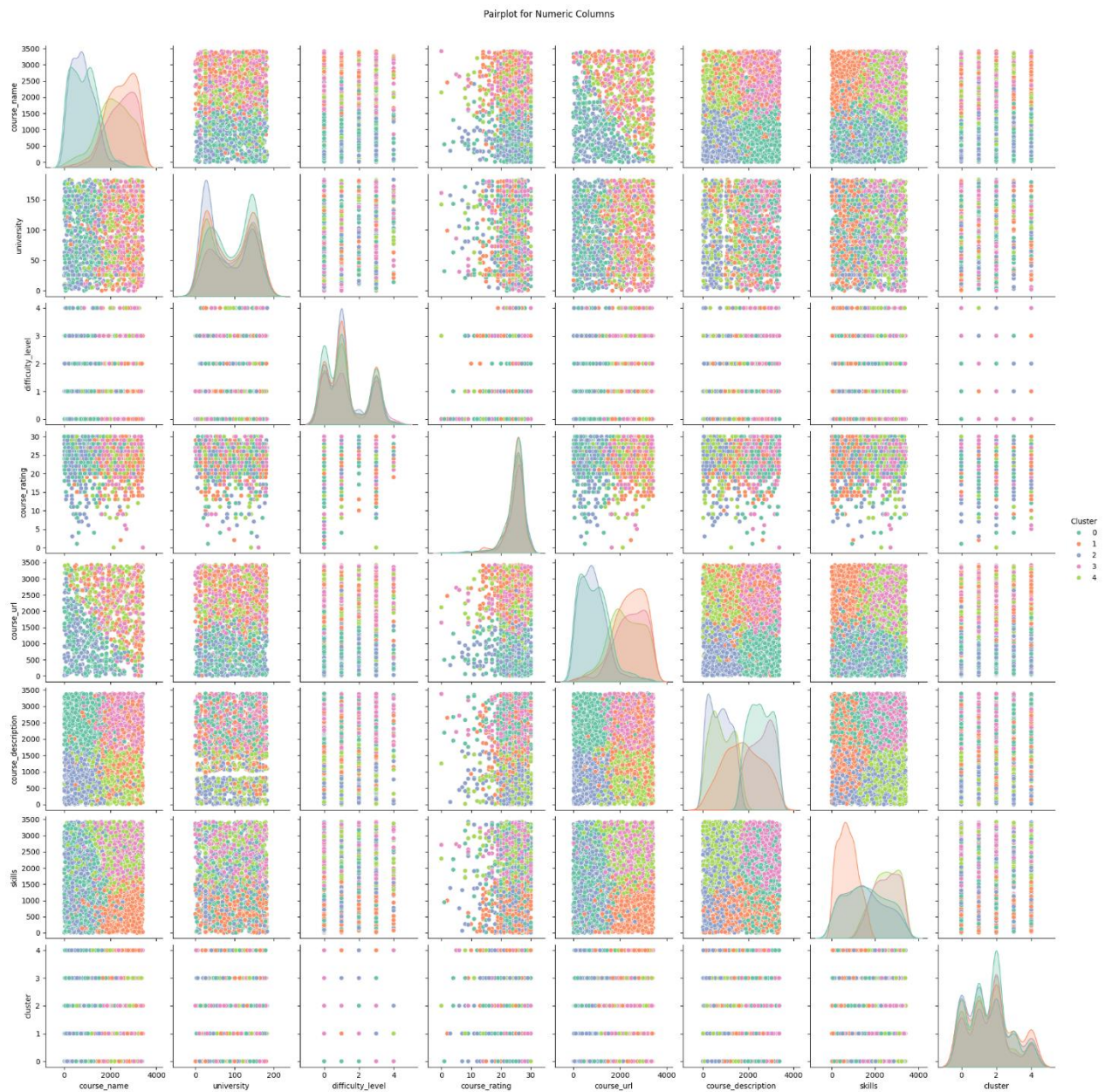
Output:



Pairplot for Numeric Data: The Pairplot visualizes pairwise relationships between numeric features and clusters. This plot allows us to examine the spread of data points within each cluster and check for separability.

```
# Step 6: Pairplot for Numeric Data (If available)
if numeric_columns:
    # Ensure that the 'Cluster' column is included for hue
    if 'Cluster' in data.columns:
        sns.pairplot(data[numeric_columns + ['Cluster']], hue='Cluster',
palette='Set2')
        plt.suptitle('Pairplot for Numeric Columns', y=1.02)
        plt.show()
```


➤ Output:



➤ Conclusion:

The **Course Recommendation System** effectively utilizes data mining techniques such as clustering and cosine similarity to provide personalized course suggestions. The integration of visualization tools enhances the understanding of the dataset and the quality of recommendations. Future improvements could include integrating collaborative filtering for hybrid recommendation models and real-time updates for dynamic datasets.

➤ References:

1. Aggarwal, C. C. (2016). **Recommender Systems: The Textbook**. Springer.
2. Koren, Y., Bell, R., & Volinsky, C. (2009). "Matrix Factorization Techniques for Recommender Systems." *Computer*, 42(8), 30-37.
3. Basu, C., Hirsh, H., & Cohen, W. W. (1998). "Recommendation Systems: A Contextual Overview." *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*.
4. Hunter, J. D. (2007). "Matplotlib: A 2D Graphics Environment." *Computing in Science & Engineering*, 9(3), 90-95.
5. Waskom, M. L. (2021). "Seaborn: Statistical Data Visualization." *Journal of Open Source Software*, 6(60), 3021.
6. Python Software Foundation. (2021). "Matplotlib: Visualization with Python." Retrieved from <https://matplotlib.org>.
7. Lops, P., De Gemmis, M., & Semeraro, G. (2011). Content-based Filtering for Recommender Systems: State of the Art and Trends. In *Recommender Systems Handbook* (pp. 73-105). Springer. o This book chapter explores content-based filtering techniques and how they can be applied to build effective recommender systems.
8. Zhao, Y., & Kumar, R. (2015). Recommender Systems with Deep Learning. *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI 2015)*. o An advanced look at deep learning techniques for enhancing recommender system performance, which could be adapted for online course recommendation.
9. Kusmierczyk, S. M., & Felfernig, A. (2015). User-oriented Filtering and Personalization in Course Recommendation Systems. In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*. o Discusses the application of filtering and personalization techniques for course recommendation systems, focusing on user preferences and course content.