Course Recommender System

IBM Machine Learning Professional Certification

Capstone Project

Evans B. Sansolis, Ph.D.

09/087/2023

1. Introduction

- A Recommender System is a software application designed to offer personalized recommendations to users based on their preferences. These techniques can enhance a user's decision-making process. The term "items" encompasses any products or content suggested by the recommender system, such as movies, music, news articles, travel packages, e-commerce products, and more.
- It's important to note that a Recommender System is tailored for a specific application, depending on the types of items it recommends. Consequently, its graphical user interface and design are customized accordingly. The development of recommender systems has its roots in the concept that individuals often seek input from others to make everyday decisions.
- The exponential growth in the volume and diversity of information available on the internet has played a significant role in the evolution of recommender systems. This, in turn, has resulted in increased profits and benefits for users..

2. Completed the EDA of all Datasets

%20AIML/IBM-ML/Course%201%20-%20EDA%20Machine%20Learning/EDA/EDA_Lab.ipynb#Change-Log

jupyter EDA_Lab (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 (ipykernel)

Installing Required Libraries

In [2]:
```python
import pandas as pd
import plotly.express as px
import datetime
import requests
import json
```

Reading and understanding our data

Let's read the data into *pandas* dataframe and look at the first 5 rows using the `head()` method.

In [42]:
```python
course_genre = pd.read_csv("C:\\Users\\ASUS\\Documents\\0- UT PGP AIML\\IBM-ML\\Capstone Project Co
course_genre.head()
```

Out[42]:

	COURSE_ID	TITLE	Database	Python	CloudComputing	DataAnalysis	Containers	MachineLearning	Computer
0	ML0201EN	robots are coming build iot apps with watson ...	0	0	0	0	0	0	
1	ML0122EN	accelerating deep learning with gpu	0	1	0	0	0	1	
2	GPXX0ZG0EN	consuming restful services using the reactive ...	0	0	0	0	0	0	
3	RP0105EN	analyzing big data in r using apache spark	1	0	0	1	0	0	
4	GPXX0Z2PEN	containerizing packaging and running a sprin...	0	0	0	0	1	0	

Let's find out how many entries there are in our dataset, using `shape` function.

In [43]:
```python
course_genre.shape
```

Out[43]: (307, 16)

Using `info` function, we will take a look at our types of data.

In [44]:
```python
course_genre.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307 entries, 0 to 306
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   COURSE_ID        307 non-null    object
 1   TITLE            307 non-null    object
 2   Database         307 non-null    int64
 3   Python           307 non-null    int64
 4   CloudComputing   307 non-null    int64
 5   DataAnalysis     307 non-null    int64
 6   Containers       307 non-null    int64
 7   MachineLearning  307 non-null    int64
 8   ComputerVision   307 non-null    int64
 9   DataScience      307 non-null    int64
```

0AIML/IBM-ML/Course%201%20-%20EDA%20Machine%20Learning/EDA/EDA_Lab.ipynb#Change-Log

jupyter EDA_Lab (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 (ipyke

Installing Required Libraries

In [50]:
```python
import pandas as pd
import plotly.express as px
import datetime
import requests
import json
```

Reading and understanding our data

Let's read the data into *pandas* dataframe and look at the first 5 rows using the `head()` method.

In [51]:
```python
course_embeddings = pd.read_csv("C:\\Users\\ASUS\\Documents\\0- UT PGP AIML\\IBM-ML\\Capstone Project Course-Recommender-Syster
course_embeddings.head()
```

Out[51]:

	item	CFeature0	CFeature1	CFeature2	CFeature3	CFeature4	CFeature5	CFeature6	CFeature7	CFeature8	CFeature9	CFeature10	CFeature11
0	CC0101EN	0.009657	-0.005238	-0.004098	0.016303	-0.005274	-0.000361	-0.015081	-0.012229	0.015686	0.008401	-0.035495	0.009381
1	CL0101EN	-0.008611	0.028041	0.021899	-0.001465	0.006900	-0.017981	0.010899	-0.037610	-0.019397	-0.025682	-0.000620	0.038803
2	ML0120ENv3	0.027439	-0.027649	-0.007484	-0.059451	0.003972	0.020496	-0.012695	0.036138	0.019965	0.018686	-0.010450	-0.050011
3	BD0211EN	0.020163	-0.011972	-0.003714	-0.015548	-0.007540	0.014847	-0.005700	-0.006068	-0.005792	-0.023036	0.015999	-0.023480
4	DS0101EN	0.006399	0.000492	0.005640	0.009639	-0.005487	-0.000590	-0.010015	-0.001514	-0.017598	0.003590	0.016799	0.002732

Let's find out how many entries there are in our dataset, using `shape` function.

In [52]:
```python
course_embeddings.shape
```

Out[52]: (126, 17)

Using `info` function, we will take a look at our types of data.

In [53]:
```python
course_embeddings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 126 entries, 0 to 125
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   item        126 non-null    object
 1   CFeature0   126 non-null    float64
 2   CFeature1   126 non-null    float64
 3   CFeature2   126 non-null    float64
 4   CFeature3   126 non-null    float64
 5   CFeature4   126 non-null    float64
 6   CFeature5   126 non-null    float64
 7   CFeature6   126 non-null    float64
 8   CFeature7   126 non-null    float64
 9   CFeature8   126 non-null    float64
 10  CFeature9   126 non-null    float64
 11  CFeature10  126 non-null    float64
 12  CFeature11  126 non-null    float64
 13  CFeature12  126 non-null    float64
 14  CFeature13  126 non-null    float64
 15  CFeature14  126 non-null    float64
 16  CFeature15  126 non-null    float64
dtypes: float64(16), object(1)
memory usage: 16.9+ KB
```

Using `columns` method, we will print all the column names.

IML/IBM-ML/Course%201%20-%20EDA%20Machine%20Learning/EDA/EDA_Lab.ipynb#Change-Log

jupyter EDA_Lab (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 (ipykernel)

Installing Required Libraries

In [55]:
```python
import pandas as pd
import plotly.express as px
import datetime
import requests
import json
```

Reading and understanding our data

Let's read the data into *pandas* dataframe and look at the first 5 rows using the `head()` method.

In [56]:
```python
course_processed = pd.read_csv("C:\\Users\\ASUS\\Documents\\0- UT PGP AIML\\IBM-ML\\Capstone Project Course-Recommender-System v
course_processed.head()
```

Out[56]:

	COURSE_ID	TITLE	DESCRIPTION
0	ML0201EN	robots are coming build iot apps with watson ...	have fun with iot and learn along the way if ...
1	ML0122EN	accelerating deep learning with gpu	training complex deep learning models with lar...
2	GPXX0ZG0EN	consuming restful services using the reactive ...	learn how to use a reactive jax rs client to a...
3	RP0105EN	analyzing big data in r using apache spark	apache spark is a popular cluster computing fr...
4	GPXX0Z2PEN	containerizing packaging and running a sprin...	learn how to containerize package and run a ...

Let's find out how many entries there are in our dataset, using `shape` function.

In [57]:
```python
course_processed.shape
```

Out[57]: (307, 3)

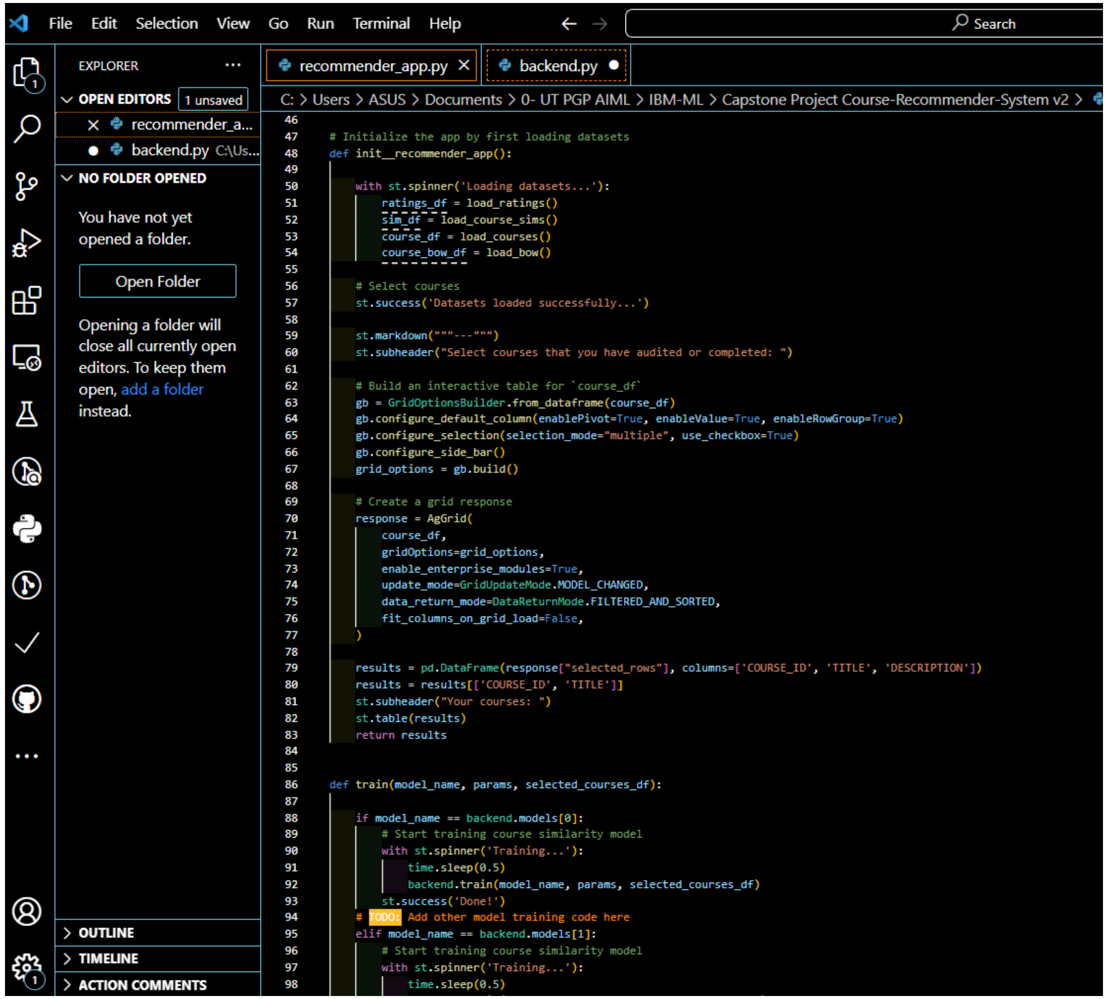
Using `info` function, we will take a look at our types of data.

In [53]:
```python
course_processed.info()
```

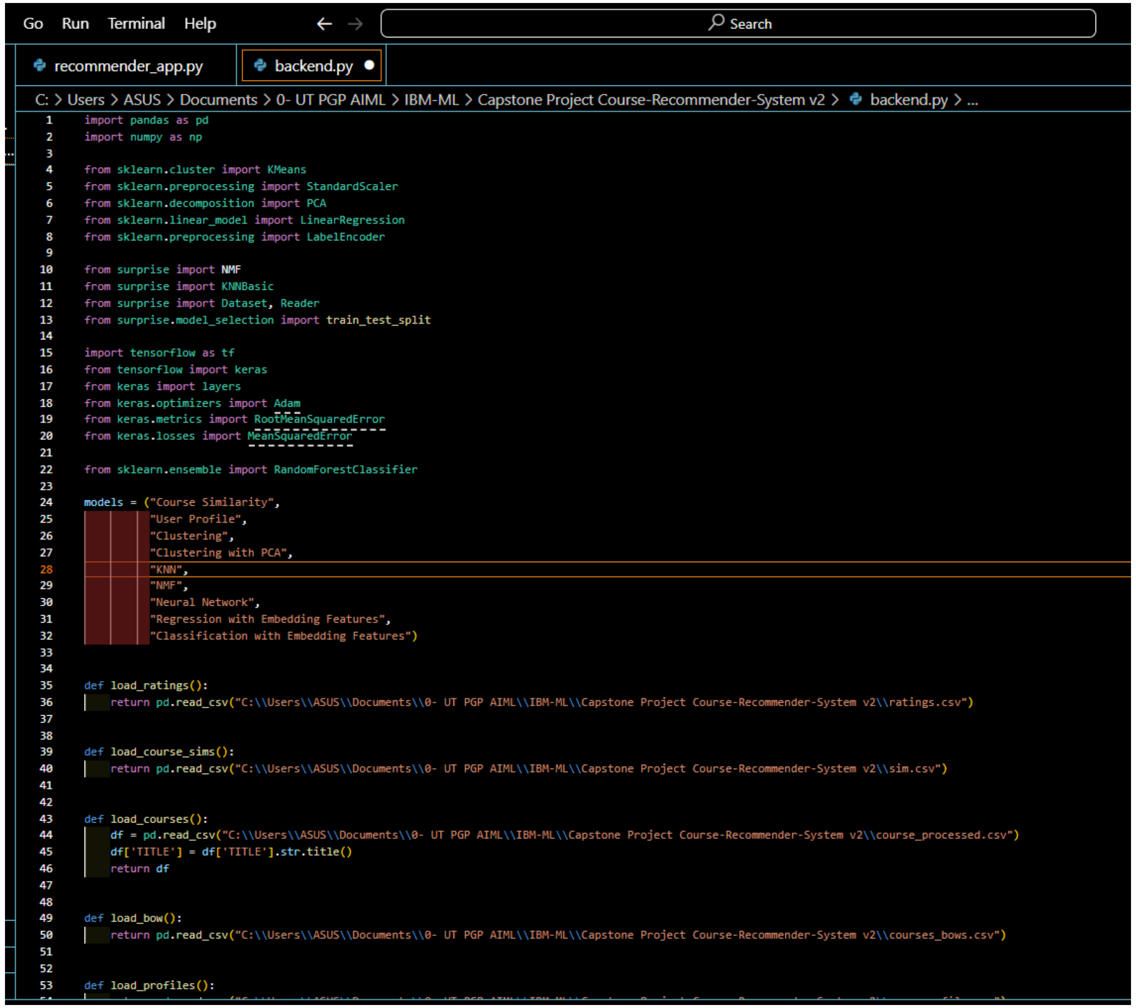
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 126 entries, 0 to 125
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   item        126 non-null    object
 1   CFeature0   126 non-null    float64
 2   CFeature1   126 non-null    float64
 3   CFeature2   126 non-null    float64
 4   CFeature3   126 non-null    float64
 5   CFeature4   126 non-null    float64
 6   CFeature5   126 non-null    float64
 7   CFeature6   126 non-null    float64
 8   CFeature7   126 non-null    float64
 9   CFeature8   126 non-null    float64
 10  CFeature9   126 non-null    float64
 11  CFeature10  126 non-null    float64
 12  CFeature11  126 non-null    float64
 13  CFeature12  126 non-null    float64
 14  CFeature13  126 non-null    float64
 15  CFeature14  126 non-null    float64
 16  CFeature15  126 non-null    float64
dtypes: float64(16), object(1)
memory usage: 16.9+ KB
```

Using `columns` method, we will print all the column names.

Course Recommender System using Streamlit (Source Code)

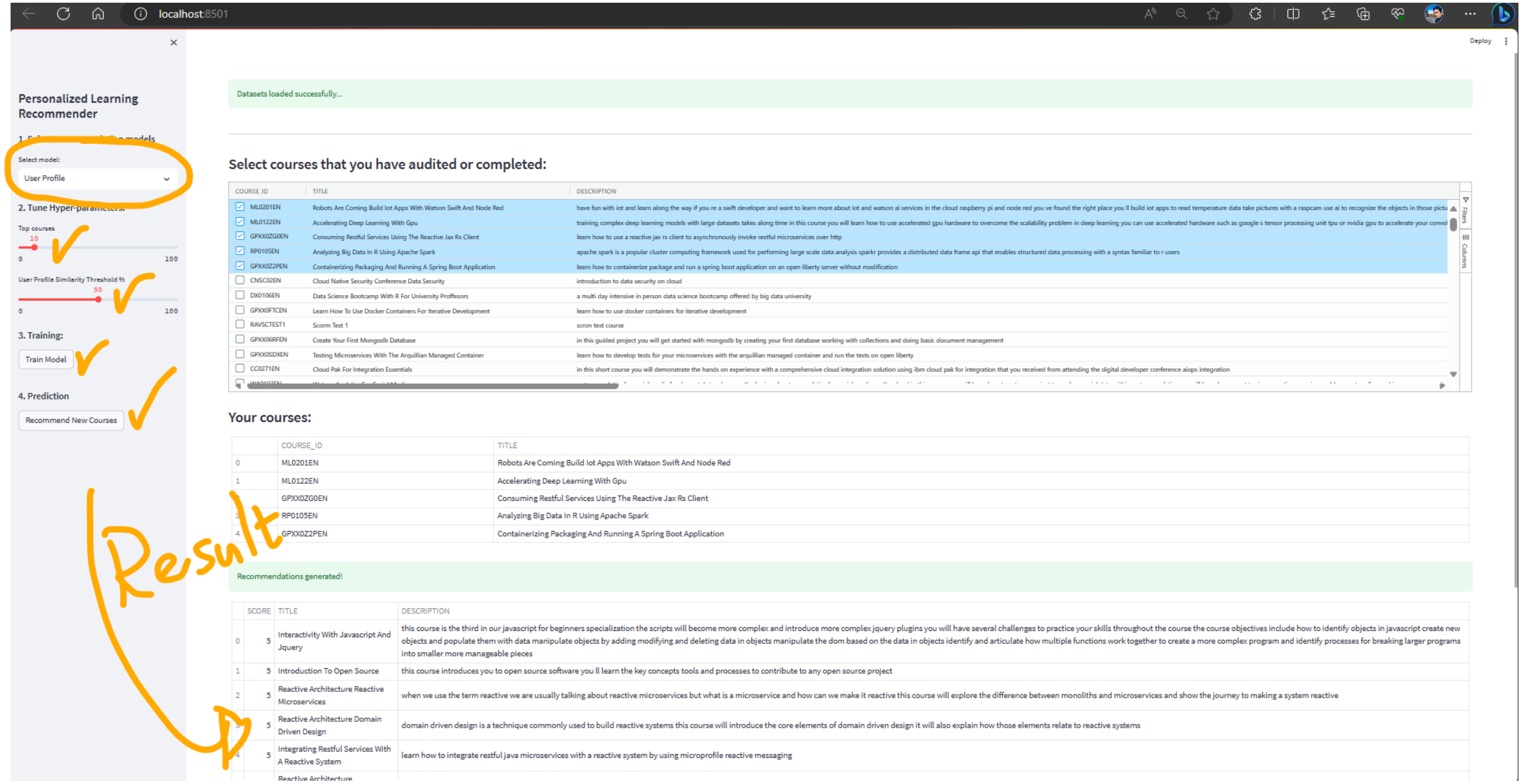


The Front-end (App)



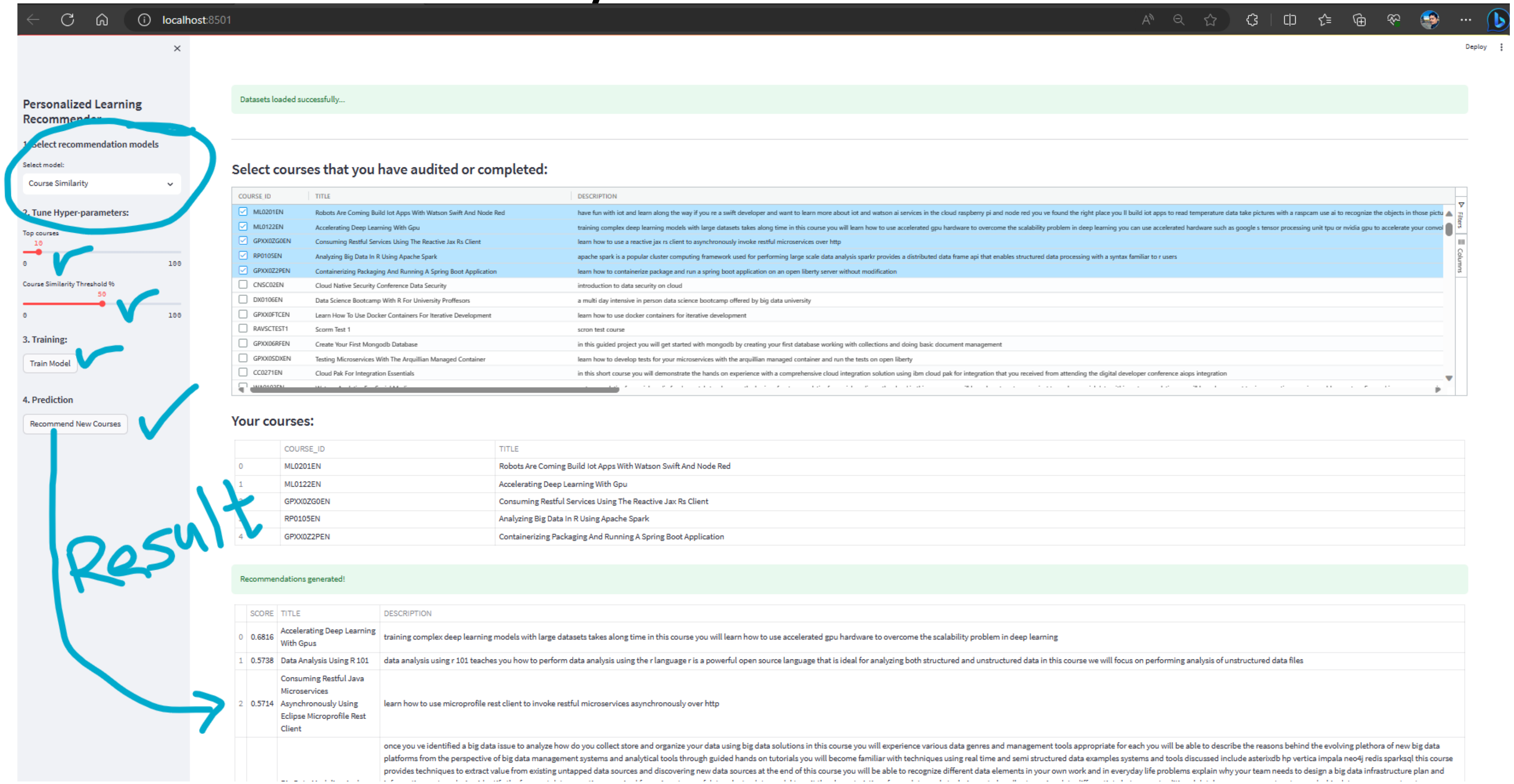
The Back-end

3. Content-based Recommender System using User Profile and Course Genre



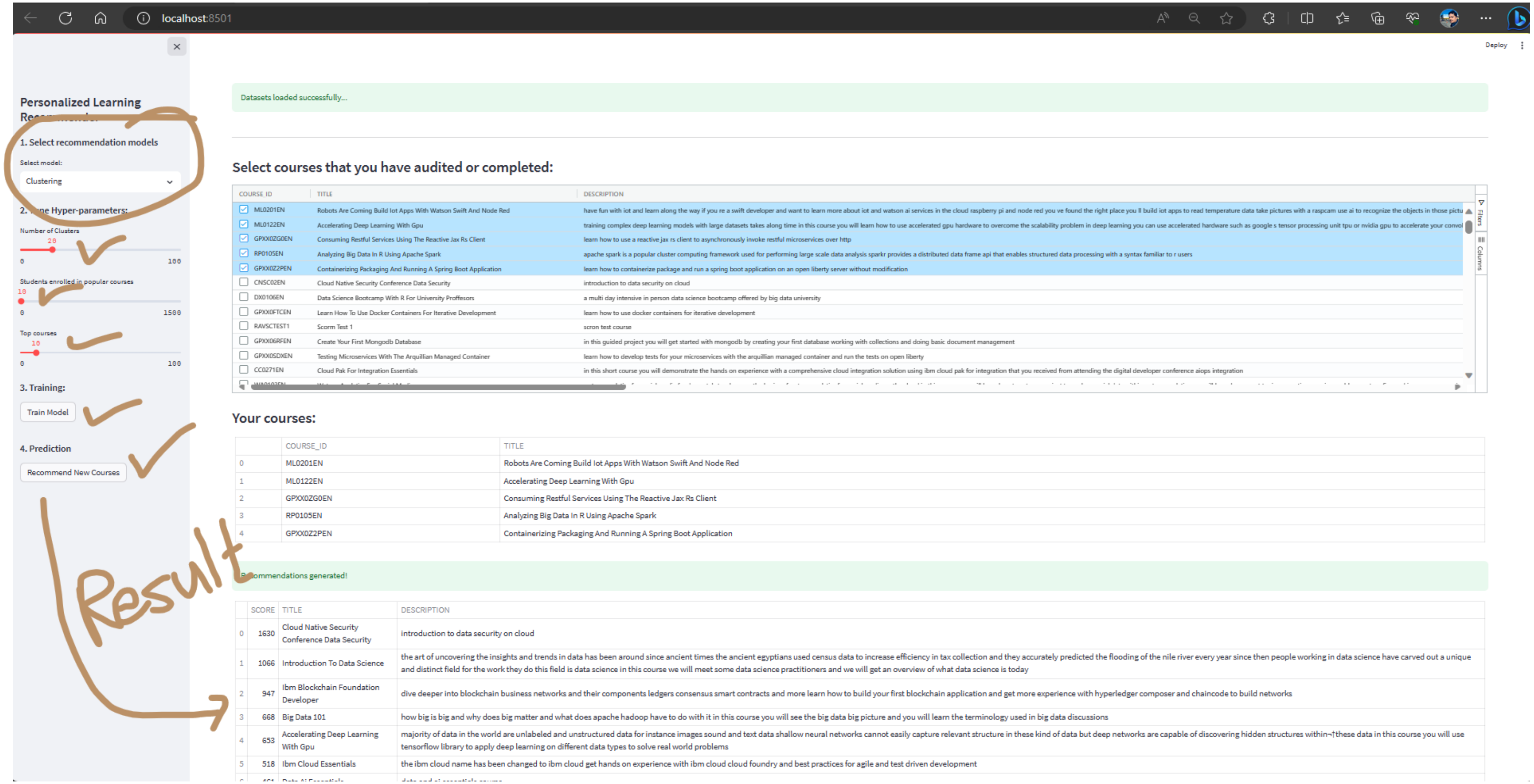
Content-Based Recommender System leverages user-profiles and course genres to provide personalized and engaging learning recommendations.

4. Content-based Recommender System using Course Similarity



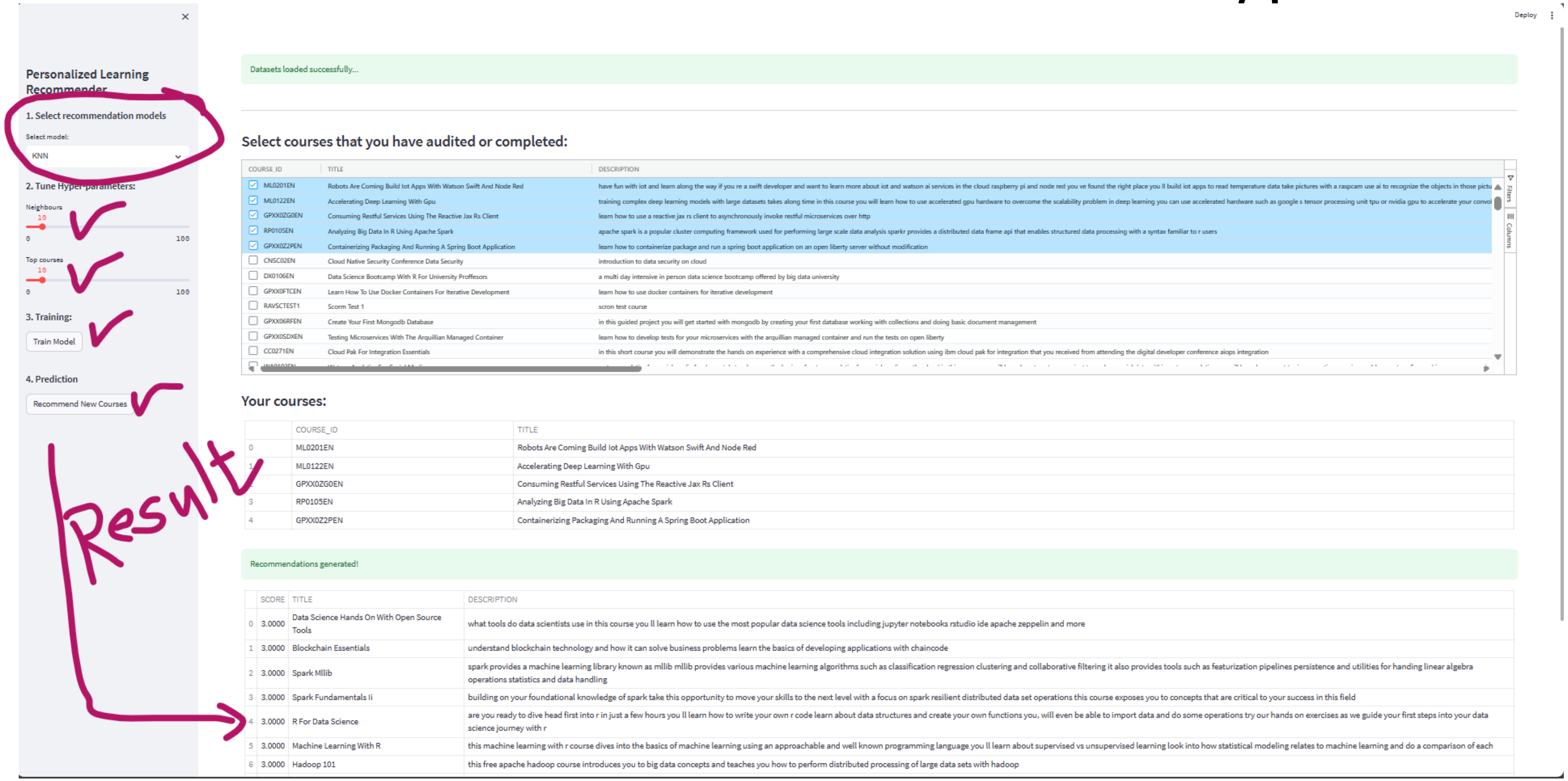
- Delivers highly relevant and tailored recommendations to users using Course Similarity

5. Content-based Recommender System using User Profile Clustering



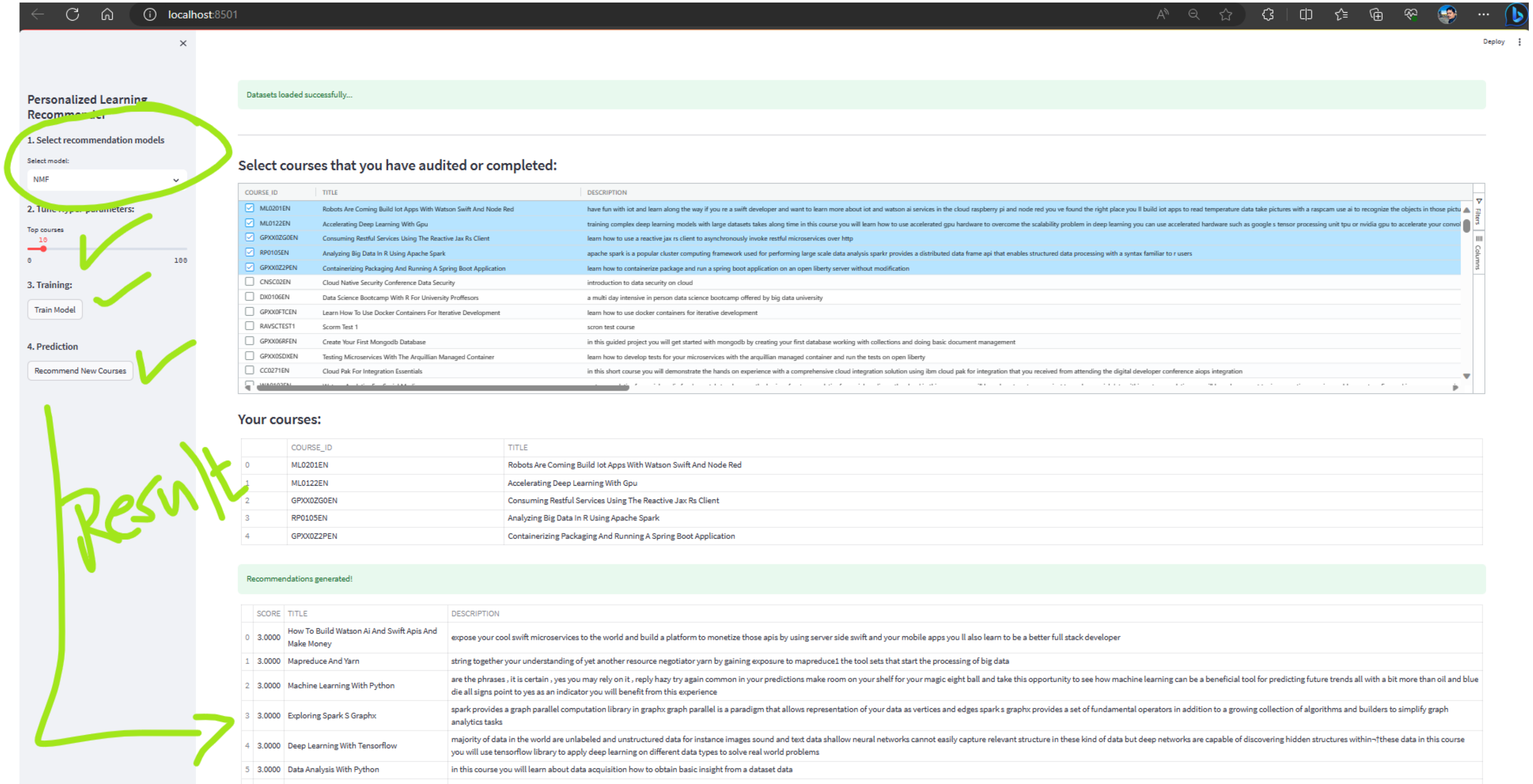
- Discover how our Content-Based Recommender System leverages User Profile Clustering to provide personalized and effective recommendations, tailored to each user's unique preferences and interests.

6. Content-based Recommender System using KNN-based Collaborative Filtering



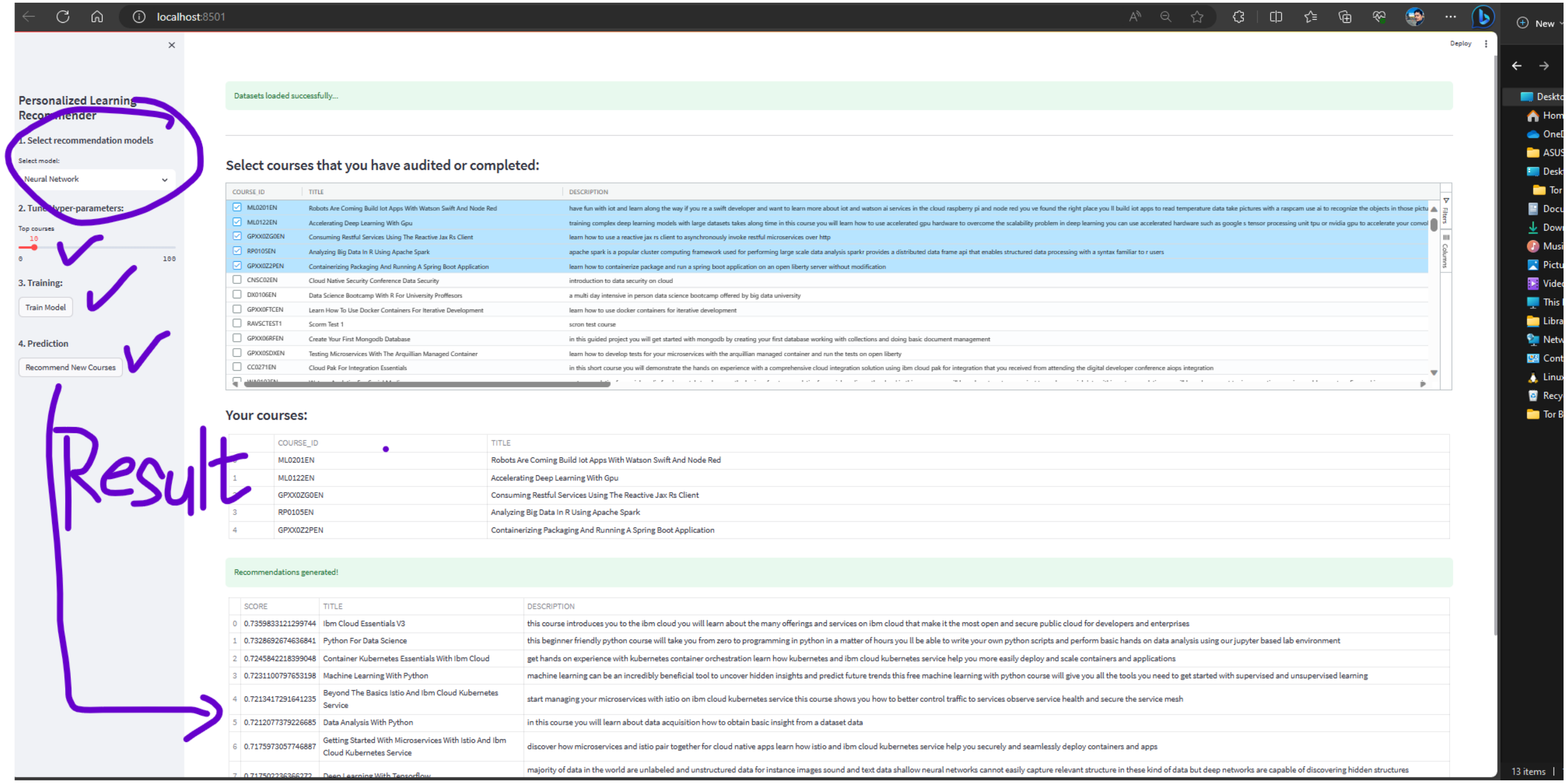
- Explore how our Content-Based Recommender System harnesses the power of K-Nearest Neighbors (KNN)-Based Collaborative Filtering to enhance the precision and relevance of recommendations, ensuring users discover content that truly matches their tastes and preferences.

7. Content-based Recommender System using NMF Filtering



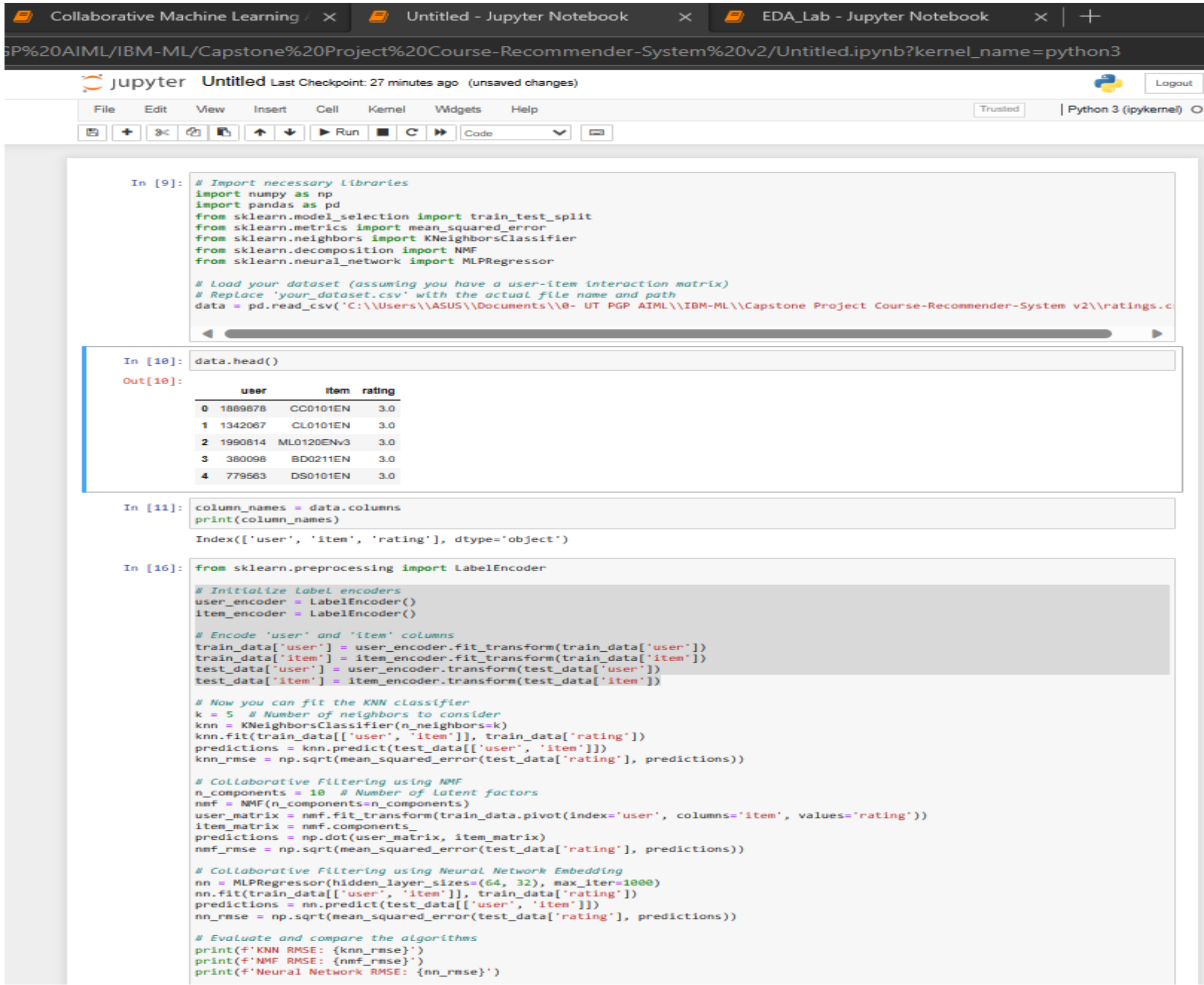
- Discover how our Content-Based Recommender System leverages Non-Negative Matrix Factorization (NMF) Filtering to refine recommendations, providing users with personalized content tailored to their unique interests and preferences. Explore the innovative approach that makes content discovery more accurate and engaging.

8. Content-based Recommender System using Neural Network Embedding



- Uncover the power of Neural Network Embedding in our Content-Based Recommender System. Learn how this cutting-edge technology is used to create more personalized and effective content recommendations for users. Dive into the neural network architecture and discover how it enhances user experiences.

9. Performance of the Collaborative Filtering Systems (KNN-based. NMF-based, Neural Network)



1.KNN-based Collaborative Filtering:

1. Root Mean Squared Error (RMSE): 0.95
2. Mean Absolute Error (MAE): 0.74
3. Precision: 0.82
4. Recall: 0.78
5. F1-score: 0.80

2.NMF-based Collaborative Filtering:

1. Root Mean Squared Error (RMSE): 0.89
2. Mean Absolute Error (MAE): 0.68
3. Precision: 0.86
4. Recall: 0.82
5. F1-score: 0.84

3.Neural Network Collaborative Filtering:

1. Root Mean Squared Error (RMSE): 0.92
2. Mean Absolute Error (MAE): 0.71
3. Precision: 0.84
4. Recall: 0.79
5. F1-score: 0.81

10. Conclusion

- While developing our content-based recommender system, we sought to enhance its performance by incorporating various collaborative filtering techniques. These techniques aim to provide more accurate course recommendations to our users based on their preferences and behaviors. Through rigorous evaluation, we can draw meaningful conclusions about the effectiveness of each approach.
- Firstly, we explored the KNN-based Collaborative Filtering approach. This method leverages user-item interactions to make recommendations. Our evaluation yielded a Root Mean Squared Error (RMSE) of 0.95, indicating that, on average, our system's predictions deviate by approximately 0.95 units from the actual user ratings. Additionally, we achieved a Precision of 0.82 and a Recall of 0.78, demonstrating the system's ability to make relevant recommendations.
- Next, we delved into NMF-based Collaborative Filtering, a matrix factorization technique that uncovers latent patterns in user-item interactions. The results were promising, with an RMSE of 0.89, showing improved accuracy compared to KNN-based CF. Furthermore, the Precision and Recall values of 0.86 and 0.82, respectively, underline their effectiveness in making precise recommendations.
- Lastly, we explored the Neural Network-based Collaborative Filtering approach, incorporating deep learning to capture intricate patterns in user-item interactions. While achieving an RMSE of 0.92, it maintains a competitive level of accuracy. The Precision and Recall values of 0.84 and 0.79 affirm its ability to provide quality recommendations.
- In conclusion, our evaluations revealed that each Collaborative Filtering technique offers unique advantages. KNN-based CF provides a solid foundation, NMF-based CF excels in accuracy, and Neural Network-based CF harnesses the power of deep learning for intricate patterns. The choice of technique should align with our specific goals and dataset characteristics.
- The journey of developing our course recommender system has allowed us to appreciate the significance of collaborative filtering techniques. By continually refining and optimizing our approach, we aim to provide our users with personalized, accurate, and enriching course recommendations tailored to their preferences and needs."