**UDEMY COURSE RECOMMENDATION SYSTEM**

**BIG DATA AND ANALYTICS**

**B.E. COMPUTER ENGINEERING**

**Guide:**

**Ms. Anuradha Srinivasaraghavan**

By

MENEZES NITHIN MARK 58

MATHIAS SHERWIN NAVIN 53

LIKHI YOGITA VIKAS 46

MAURYA UAJALA LALSAHAB 54

****

Department of Information Technology

St. Francis Institute of Technology

(Engineering College)

University of Mumbai

**Contents**

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Contents** | **Page No.** |
| **1** | **PROBLEM STATEMENT** | **3** |
| **2** | **OBJECTIVE** | **3** |
| **3** | **EXPERIMENTAL SETUP** | **3** |
| **4** | **IMPLEMENTATION** | **5** |
| **5** | **RESULT AND ANALYSIS** | **13** |
| **6** | **CONCLUSION** | **14** |

Udemy Course Recommendation System

# Problem Statement of the mini project:-

To build a recommendation app, that recommends the courses from Udemy to the person using streamlit for frontend and Udemy Dataset, based on their given interest the app will guide them the courses along with the given Course ID, Course title, URL link, and various other information will be shown for better decision making.

# Objectives

# Our main objective is to build a recommendation app with Streamlit and Udemy dataset.

* The Home page will have a list of random courses one can select and can get an insight of the various courses Udemy provides.
* The left corner will have a search bar where the user can type the courses based on his/her interest.
* The list of courses will be shown specific to their interest along with the number of courses available to be selected
* The columns shown will have the following details for better decision making.
  + Course ID
  + Course Title
  + URL
  + Price of course
  + No. of subscribers
  + No. of reviews
  + No. of lectures
  + Level of course (All, Beginner, Intermediate and Expert)

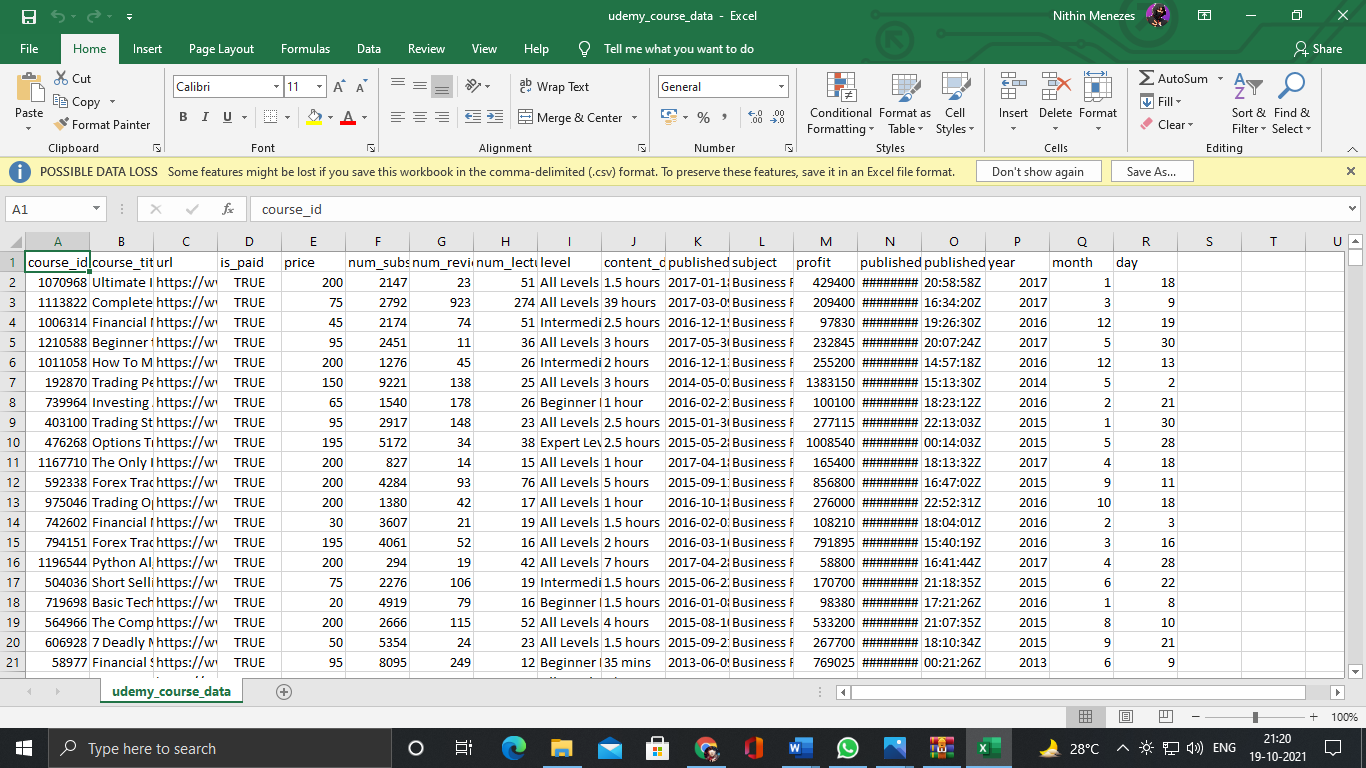
**3. Experimental Setup:**

* Software Requirements-
  + Windows/Linux OS
* Hardware Requirements-
  + Hard Disk- 256 GB or more
  + RAM - 4 GB or above
  + Processor - Core i3 or above
* Technologies Used-
  + Cosine angle for comparison
  + Big Data
  + Streamlit (Anaconda)

# Implementation:-

* The dataset that we used for the mini project is Udemy\_Course\_data.csv.

The following fig. gives a rough idea of what the columns the dataset contains



* The following is the implementation code below.

# -\*- coding: utf-8 -\*-

"""BDA\_Miniproject.ipynb

Automatically generated by Colaboratory.

Original file is located at

    https://colab.research.google.com/drive/1f1iido\_TOKgIoLZR50Rugs3xqgW6asjI

"""

!pip install streamlit

!pip install pyngrok

from google.colab import drive

drive.mount('/content/drive')

%%writefile app.py

# Core Pkg

import streamlit as st

import streamlit.components.v1 as stc

# Load EDA

import pandas as pd

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.metrics.pairwise import cosine\_similarity,linear\_kernel

# Load Our Dataset

def load\_data(data):

  df = pd.read\_csv(data)

  return df

# Fxn

# Vectorize + Cosine Similarity Matrix

def vectorize\_text\_to\_cosine\_mat(data):

  count\_vect = CountVectorizer()

  cv\_mat = count\_vect.fit\_transform(data)

  # Get the cosine

  cosine\_sim\_mat = cosine\_similarity(cv\_mat)

  return cosine\_sim\_mat

# Recommendation Sys

@st.cache

def get\_recommendation(title,cosine\_sim\_mat,df,num\_of\_rec=10):

  # indices of the course

  course\_indices = pd.Series(df.index,index=df['course\_title']).drop\_duplicates()

  # Index of course

  idx = course\_indices[title]

  # Look into the cosine matr for that index

  sim\_scores =list(enumerate(cosine\_sim\_mat[idx]))

  sim\_scores = sorted(sim\_scores,key=lambda x: x[1],reverse=True)

  selected\_course\_indices = [i[0] for i in sim\_scores[1:]]

  selected\_course\_scores = [i[0] for i in sim\_scores[1:]]

  # Get the dataframe & title

  result\_df = df.iloc[selected\_course\_indices]

  result\_df['similarity\_score'] = selected\_course\_scores

  final\_recommended\_courses = result\_df[['course\_title','similarity\_score','url','price','num\_subscribers']]

  return final\_recommended\_courses.head(num\_of\_rec)

RESULT\_TEMP = """

<div style="width:90%;height:100%;margin:1px;padding:5px;position:relative;border-radius:5px;border-bottom-right-radius: 60px;

box-shadow:0 0 15px 5px #ccc; background-image: url('https://www.professionalcontentcreation.com/wp-content/uploads/2016/05/top-10-udemy-courses-for-writers-1.jpg');

  border-left: 5px solid #6c6c6c;">

<h4>{}</h4>

<p style="color:blue;"><span style="color:black;">📈Score::</span>{}</p>

<p style="color:blue;"><span style="color:black;">🔗</span><a href="{}",target="\_blank">Link</a></p>

<p style="color:blue;"><span style="color:black;">💲Price:</span>{}</p>

<p style="color:blue;"><span style="color:black;">🧑‍🎓👨🏽‍🎓 Students:</span>{}</p>

</div>

"""

# Search For Course

@st.cache

def search\_term\_if\_not\_found(term,df):

  result\_df = df[df['course\_title'].str.contains(term)]

  return result\_df

def main():

  st.title("Course Recommendation App")

  menu = ["Home","Recommend","About"]

  choice = st.sidebar.selectbox("Menu",menu)

  df = load\_data('/content/drive/MyDrive/Data/udemy\_course\_data.csv')

  if choice == "Home":

    st.subheader("Home")

    st.dataframe(df.head(20))

  elif choice == "Recommend":

    st.subheader("Recommend Courses")

    cosine\_sim\_mat = vectorize\_text\_to\_cosine\_mat(df['course\_title'])

    search\_term = st.text\_input("Search")

    num\_of\_rec = st.sidebar.number\_input("Number",4,30,7)

    if st.button("Recommend"):

      if search\_term is not None:

        try:

          results = get\_recommendation(search\_term,cosine\_sim\_mat,df,num\_of\_rec)

          with st.expander("Results as JSON"):

            results\_json = results.to\_dict('index')

            st.write(results\_json)

          for row in results.iterrows():

            rec\_title = row[1][0]

            rec\_score = row[1][1]

            rec\_url = row[1][2]

            rec\_price = row[1][3]

            rec\_num\_sub = row[1][4]

            # st.write("Title",rec\_title,)

            stc.html(RESULT\_TEMP.format(rec\_title,rec\_score,rec\_url,rec\_url,rec\_num\_sub),height=350)

        except:

          results= "Course Not Found      OR      If you are using keywords instead of proper course name you won't get recommendations similar to the courses you have joined."

          st.warning(results)

          st.info("Suggested Options Include")

          result\_df = search\_term\_if\_not\_found(search\_term,df)

          st.dataframe(result\_df)

        # How To Maximize Your Profits Options Trading

  else:

    st.subheader("About")

    st.text("Course Recommendation System App basically aims at recommending courses that are")

    st.text(" similar to those that you have completed or joined.")

    st.text("To find recommendations one needs to specify the correct name of their course.")

    st.text("If one uses keywords, only suggestions related to the search term will be displayed.")

    st.text("The user can then copy course name of their choice from there and find")

    st.text("recommendations related to it.")

if \_\_name\_\_ == '\_\_main\_\_':

  main()

!ngrok authtoken 1z4MhgcBk8pUGAce0UYFvW99zyf\_2Ks39WFUmAFNqnqWbi2Pi

from pyngrok import ngrok

!streamlit run --server.port 80 app.py&>/dev/null&

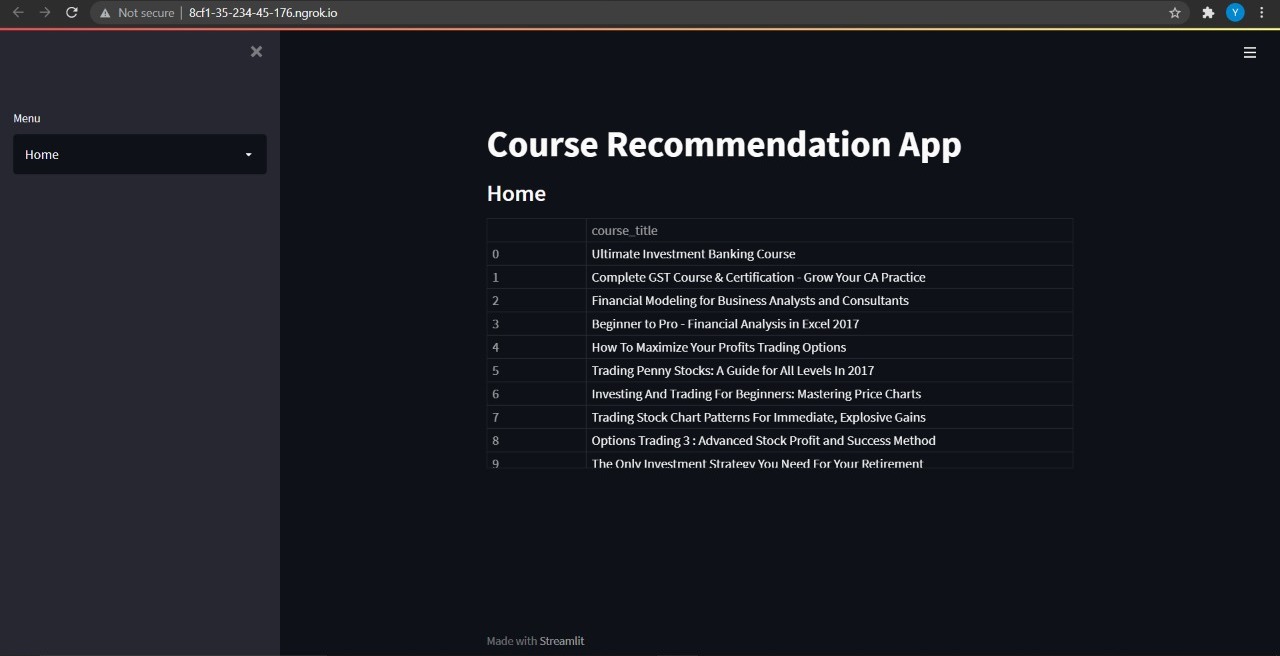
!pgrep streamlit

public\_url = ngrok.connect(port='80')

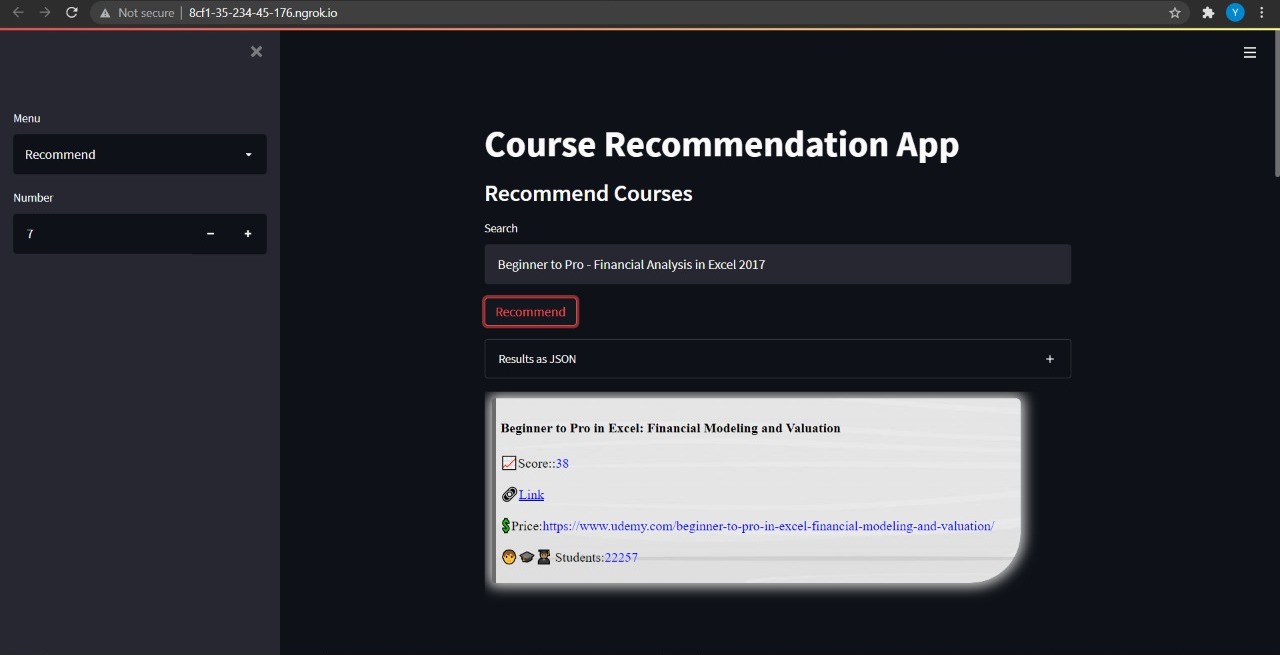
public\_url

# Visualization:-

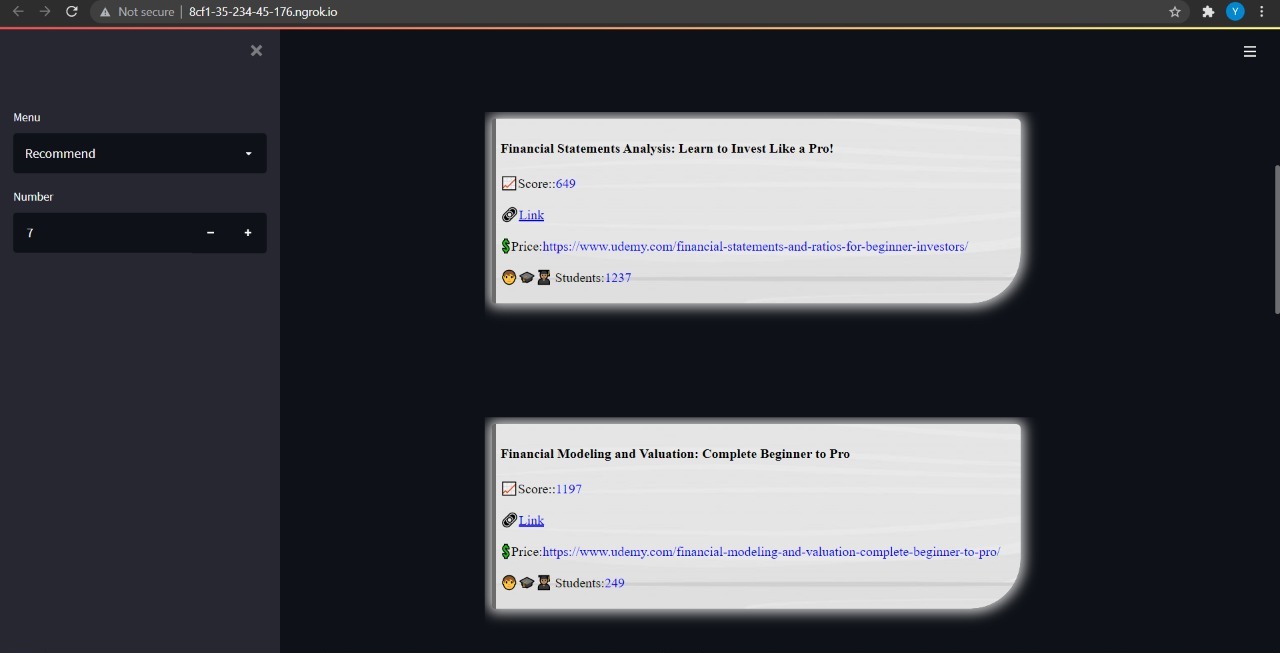
* Home page where one can have a look at the courses and get an insight of the courses available from Udemy.

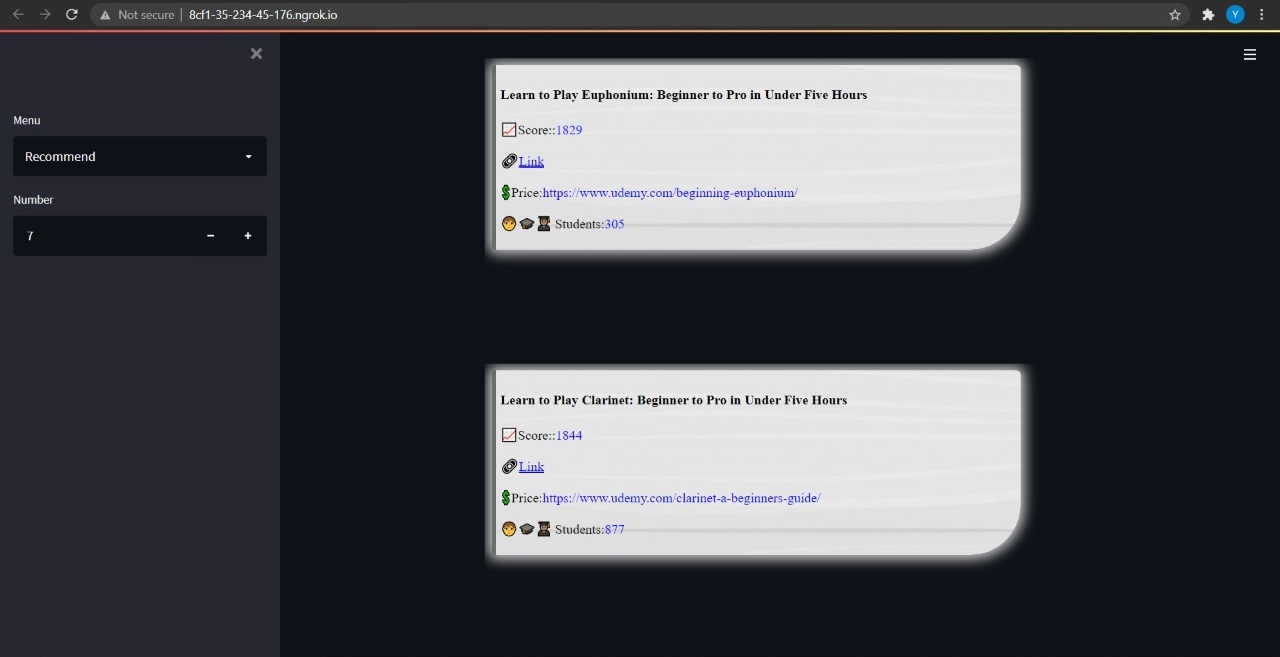


* The user can select the courses from the home page and search in the search engine, the following page will take you to other courses that are similar to the one the user searched.

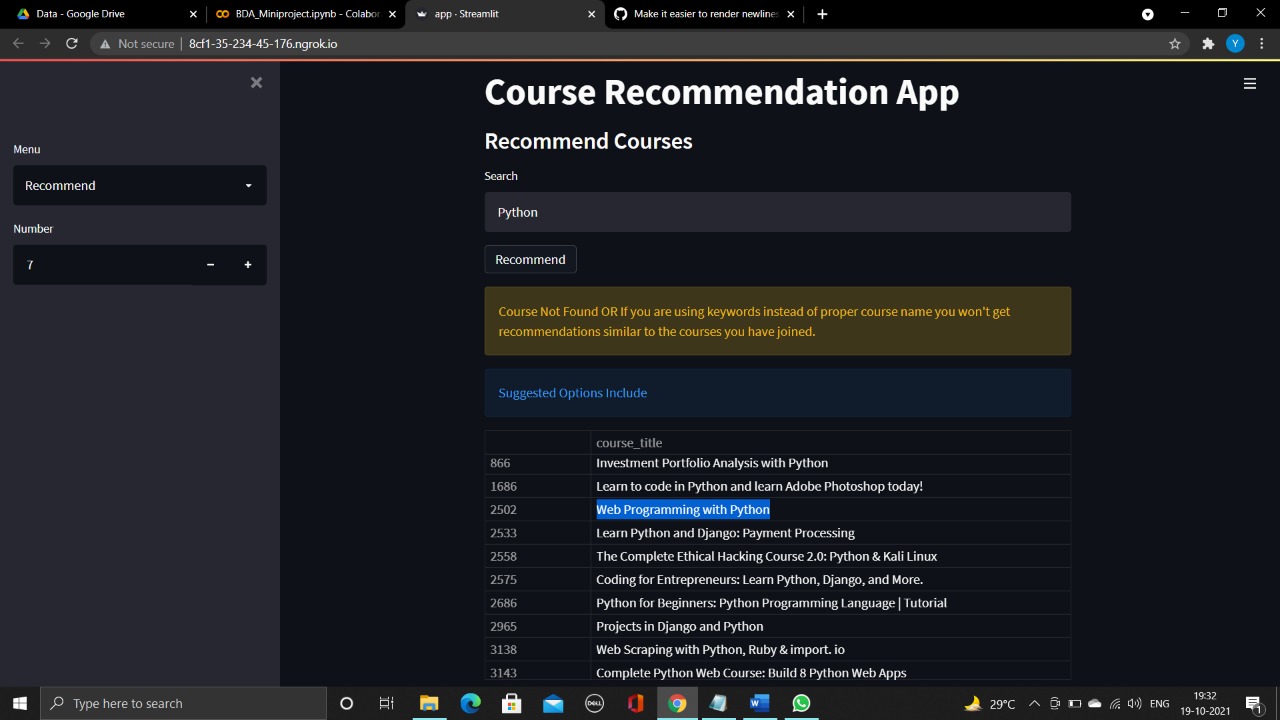


* The number of courses can be seen in the fig i.e7. That means that are 7 similar courses the user searched example- (Related to Beginner to pro- Financial Analysis in Excel 2017.)

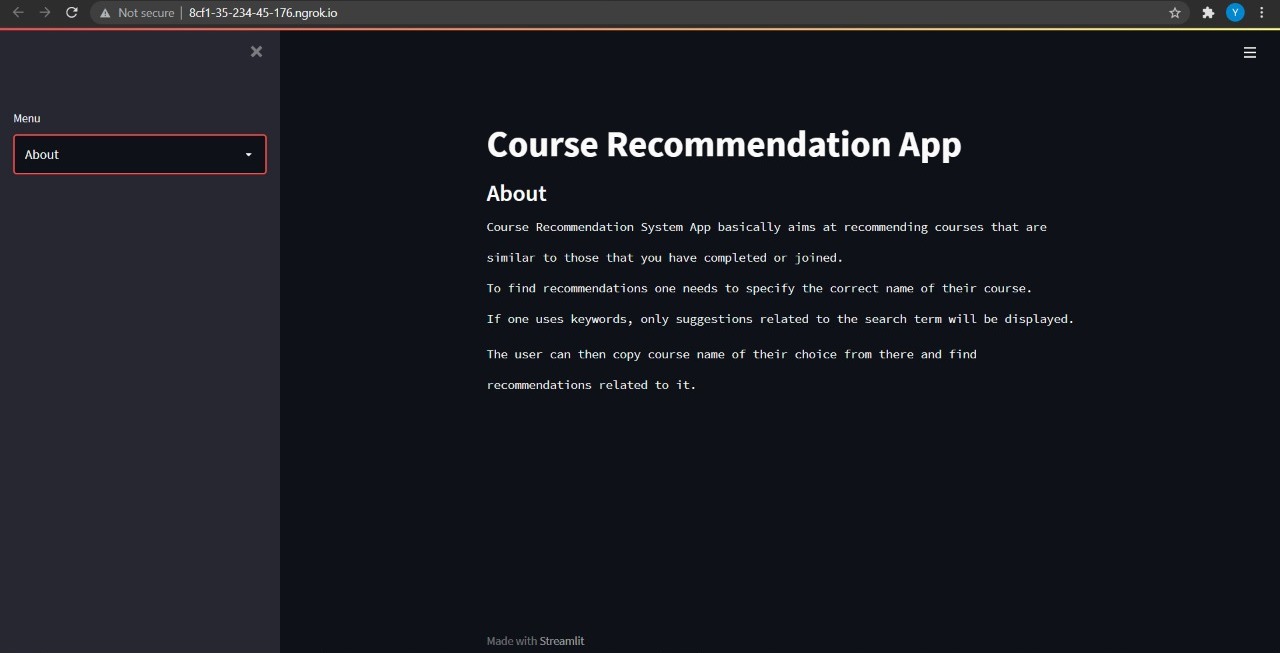




* If the following courses are not available depending on the keyword the app will recommend the courses. The columns shown will have the following details for better decision making.
  + Course ID
  + Course Title
  + URL
  + Price of course
  + No. of subscribers
  + No. of reviews
  + No. of lectures
  + Level of course (All, Beginner, Intermediate and Expert)



* In this way we will be glad that the user won’t make mistakes in selecting the courses and the user will get the courses of their choice.



# Results and Analysis:-

* In such a competitive world today, we need to update our skills day-to-day. The reason we created a recommendation app of courses specially from Udemy is that Udemy is the global e-learning marketplace.
* There are many ways to compute similarity between data, cosine similarity being one. Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It is often used to measure document similarity in text analysis.
* Since we needed to find similarity between the titles of courses we found cosine similarity to be more reliable. By computing cosine similarity for the course titles we are able to recommend similar courses for user's search term.
* In case the user just enters a keyword the app will suggest courses similar to it.

**In this way the user can save time, resources and energy in choosing a course and get the course of their interest and choice .**

Conclusion:-

With the help of Big data analysis we were able to execute queries on a huge dataset like the one we have used on for our course recommendation app. We have used Cosine similarity which is a **metric used to measure how similar two items are**. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The output value ranges from 0–1. 0 means no similarity, where as 1 means that both the items are 100% similar. Therefore we successfully built a recommendation app, that recommends the courses from Udemy to the person using streamlit for frontend and Udemy Dataset, based on their given interest the app will guide them the courses along with the given Course ID, Course title, URL link, and various other information will be shown for better decision making.

# References:-

* <https://medium.com/@jcharistech/how-to-run-streamlit-apps-from-colab-29b969a1bdfc> (How to start with Streamlit)
* <https://www.kaggle.com/andrewmvd/udemy-courses>( For Dataset)
* <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html> (Library
* installation)

<https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html> (Library installation guide)

* <https://docs.streamlit.io/knowledge-base> (Streamlit)
* <http://groups.csail.mit.edu/sls/publications/2010/Dehak_Odyssey.pdf> (Papers

on Cosine Similarity)