**Content-Based Movie Recommendation System Using AI and Machine Learning**

**Submitted to:** Dr Anu Bajaj

**Team Members:**

Karan Bindra 102283042

Harsh Arora 1021

Paras Prashar 1021

Kajol Kabiraj 1021

**Project Description:**

This project aims to develop a content-based movie recommendation system using AI and machine learning techniques. The system analyses the content and metadata of movies (such as genre, cast, director, and plot summary) to suggest movies similar to the user's preferred movie. This project aims to provide users with more personalised and accurate movie recommendations, ultimately enhancing their movie-watching experience.

**Introduction:**

Movie recommendation systems have become popular in recent years due to the growth of online streaming platforms. Traditional recommendation systems rely on collaborative filtering techniques based on user behaviour and preferences. However, these techniques often result in inaccurate recommendations due to limited user data and the "cold start" problem. Content-based recommendation systems, on the other hand, analyse the content and metadata of movies to suggest movies that are similar to the user's preferred movie. In this project, we aim to develop a content-based movie recommendation system using AI and machine learning techniques.

**Libraries Imported:**

pandas: Pandas provide powerful tools for manipulating and reshaping tabular data. It offers functions for filtering, sorting, merging, and aggregating data, making working with large and complex datasets easier.

ast: In Python, the **last** module provides a way to parse and manipulate Python code’s abstract syntax tree (AST). The AST is a structured representation of the code, similar to a parse tree, but with nodes representing higher-level language constructs such as statements, expressions, and functions.

nltk: ‘**nltk’** stands for Natural Language Toolkit, a popular Python library for natural language processing (NLP). It provides tools and resources for text processing and analysis, including tokenisation, stemming, tagging, parsing, and semantic analysis.

CountVectorizer: ‘**CountVectorizer’** is a class in the **sklearn.feature\_extraction.text** module of the sci-kit-learn library in Python, which is used to convert a collection of text documents into a matrix of token counts. It is a commonly used method for text feature extraction in machine learning.

cosine\_similarity: ‘**cosine\_similarity’** is a function in the **sklearn. Metrics. The pairwise** module of the sci-kit-learn library in Python is used to compute the cosine similarity between pairs of vectors. Cosine similarity is a widely used metric for measuring the similarity between two vectors in a high-dimensional space.

Pickle: ‘**pickle’** is a Python module that can serialise and deserialise objects in Python. Serialisation converts a Python object into a stream of bytes that can be stored in a file or sent over a network. Deserialisation is the process of reconstructing a Python object from its serialised form.

**Dataset:**

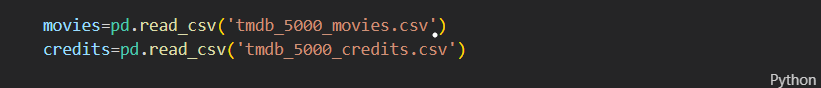
1.tmdb\_5000\_movies.csv

2.tmdb\_5000\_credits.csv

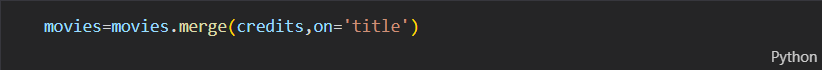
(<https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata>)

**Algorithm:**

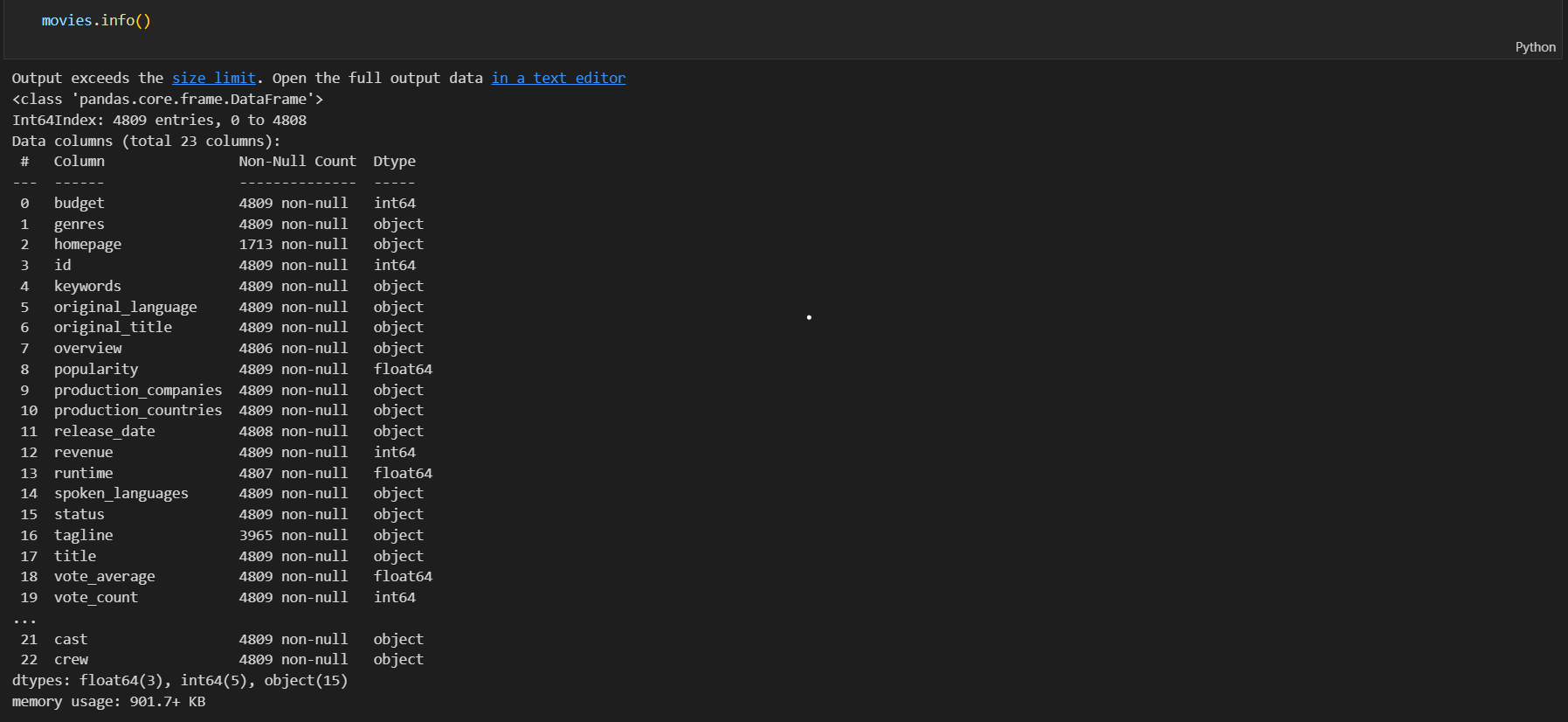
Firstly, we will put our movie and credit datasets into two sets.

****

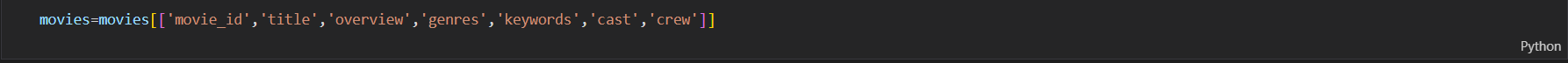
We will merge credits with the movie dataset on the title.



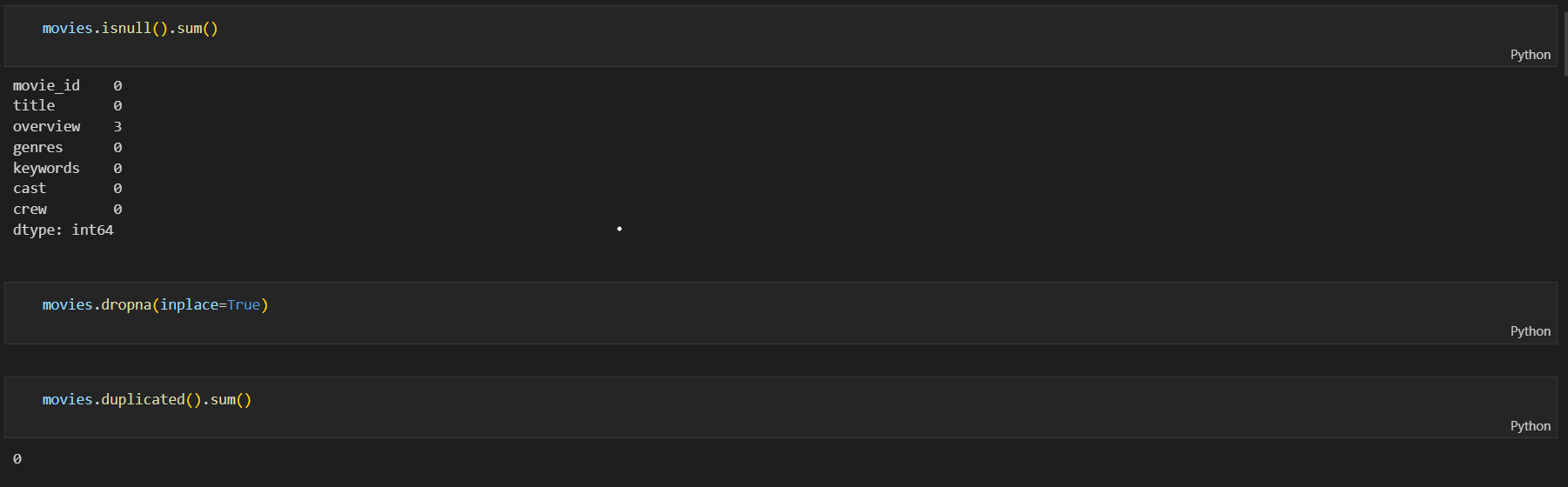
We then get the info on all the datasets we have prepared so that we can choose the crucial data we require for our work.



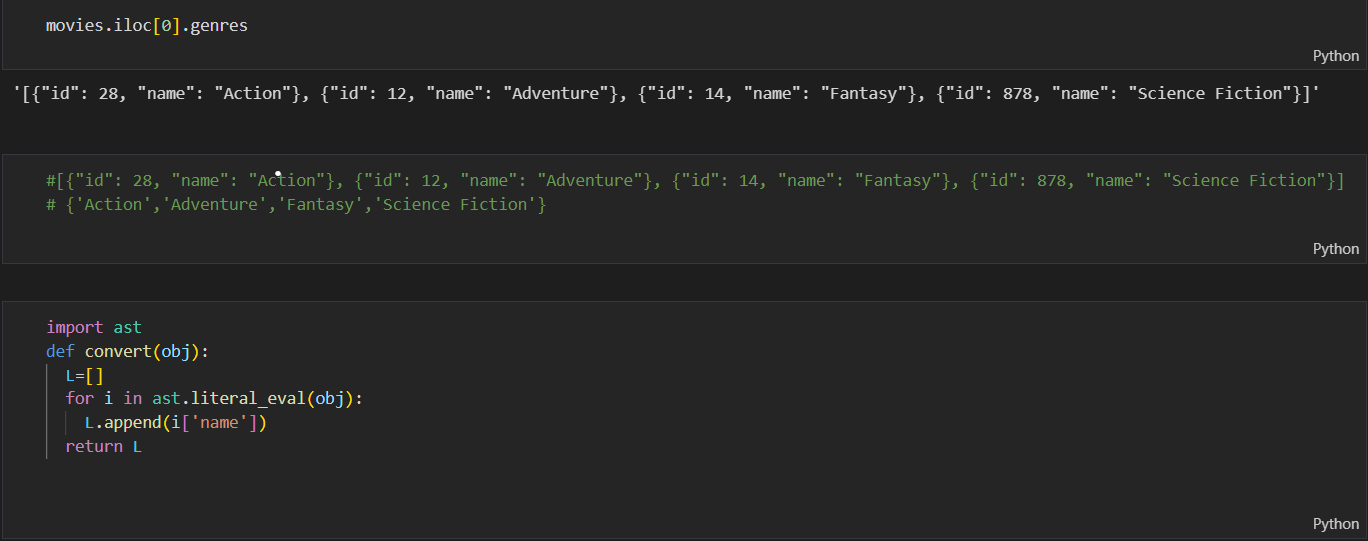
Then we only keep the essential columns we require and remove the others.



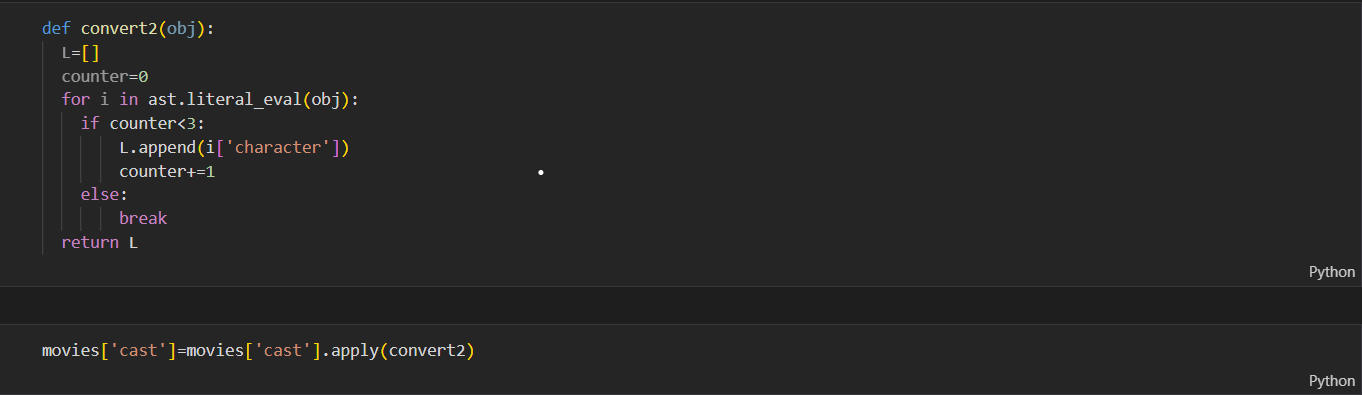
We also checked for null and duplicate values in these columns and dropped all that rows.



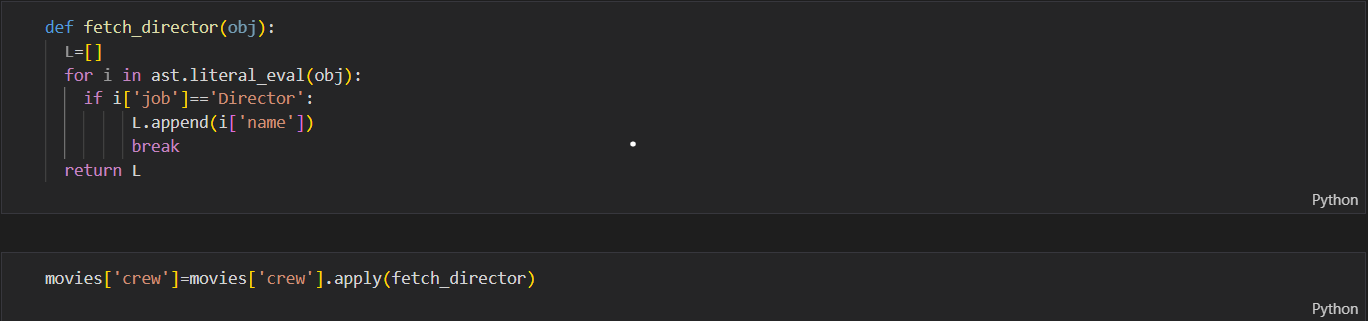
We then created a list of all the genres and keywords related to every movie using ast to use that for our purpose.



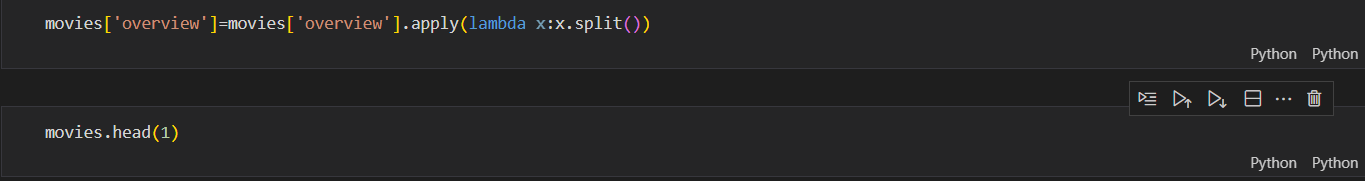
We will use the type above algorithm to find the top three actors in every movie.



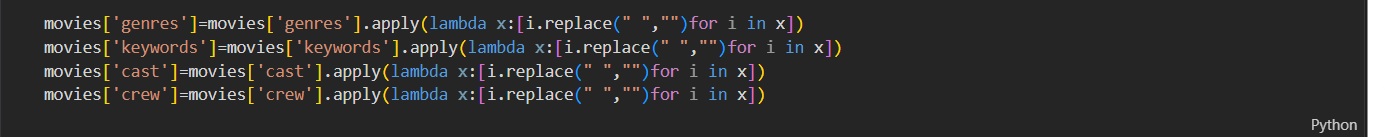
Then we will get the director’s name from our crew dataset.



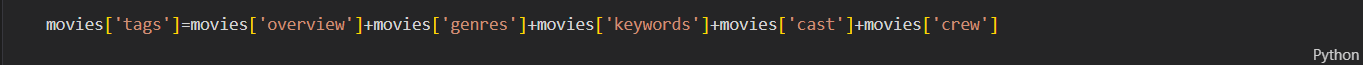
We also used the overview of the movie and split every word of the outline to find the most appropriate film.



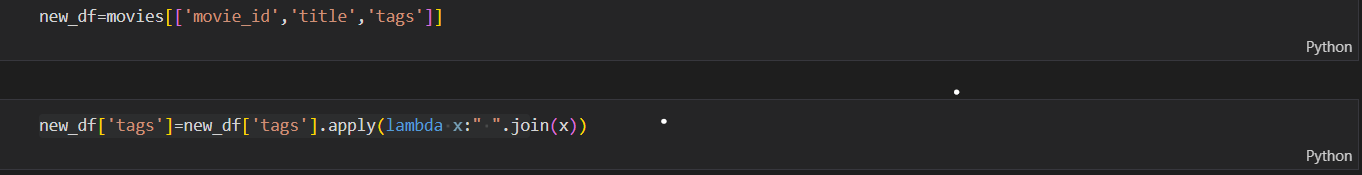
We then removed all the blank spaces from the datasets we prepared so that they could not affect the working of our system.



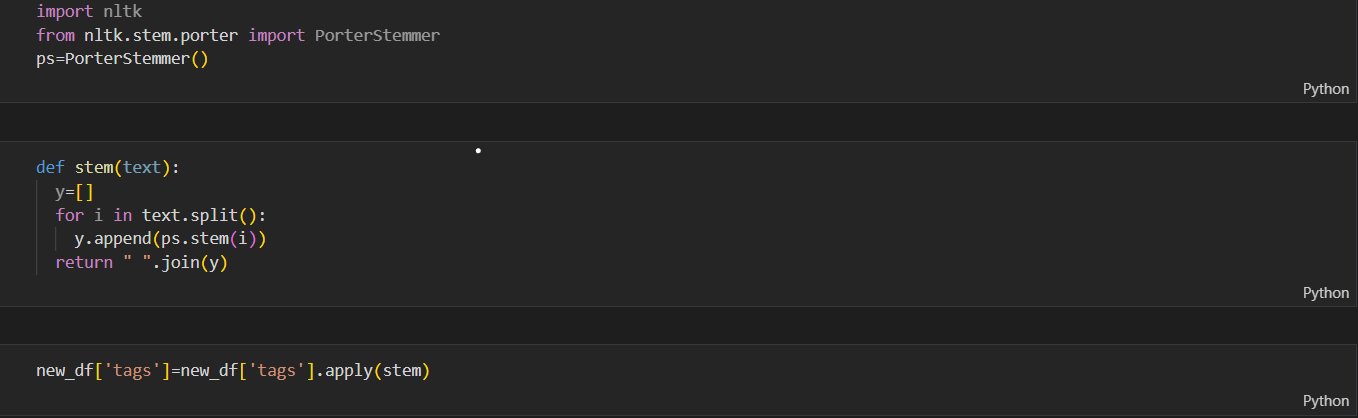
We made a tags column in movies that contains an overview, genres, keywords, cast, crew



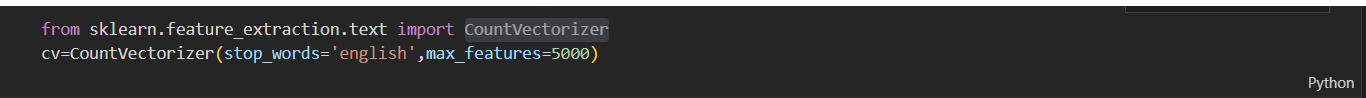
Then we make a new data frame that contains the movie id, title and tags of that movie. We also converted the tags of movies into a single paragraph.



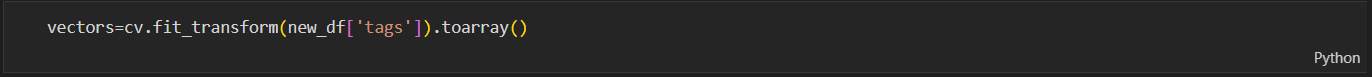
We apply stemming to tags to remove similar types of words.



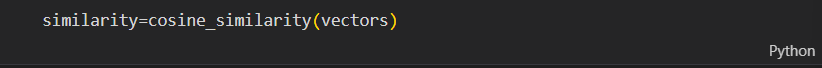
We performed count vectorization on the tags, created a vector of 5000x5000 for every movie and ignored stop words.



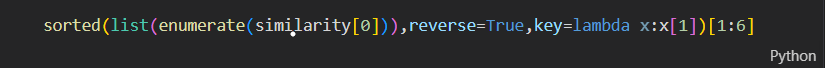
After this, we will convert the cv, i.e. sparse matrix, to a NumPy array.



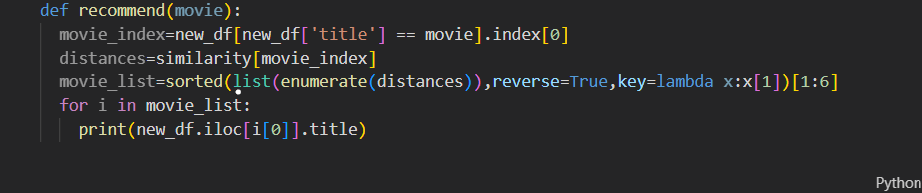
Then we use cosine\_similarity to find the similarity between each movie.



After this, we sorted the similarity list based on resemblance to the current list based on similarity.



Then using the sorted list, we recommend the top 5 movies to the user based on their input.



**Result:**

