# Movie Recommendation System **Project Documentation**

## **1. Importing Libraries**

import pandas as pd

import numpy as np

**Explanation:**

* **Pandas (‘pd’):** Used for data manipulation and analysis. It provides DataFrame structures for efficient handling of structured data.
* **NumPy (‘np’):** Used for numerical operations. It provides support for large, multi-dimensional arrays and matrices.

## **2. Reading and Displaying DataFrame**

file\_path = "C:\\Users\\surus\\Downloads\\archive (9) (1)\\TeluguMovies\_dataset.csv"

df = pd.read\_csv(file\_path)

df.head()

**Explanation:**

* This line displays the first few rows of the DataFrame (‘**df’**). It allows quick inspection of the dataset's structure and contents.

## **3. Setting Display Options**

pd.set\_option('display.max\_columns', None)

pd.set\_option('display.max\_rows', None)

**Explanation:**

* These lines set Pandas display options to show all columns (‘**max\_columns’**) and all rows (‘**max\_rows’**) in the DataFrame. Useful for comprehensive data exploration.

## **4. Importing Regular Expression Module**

import re

**Explanation:**

* The ‘**re’** module is imported to facilitate text cleaning using regular expressions. It will be used in the ‘**clean\_title’** function.

## **5. Text Cleaning Function Definition**

def clean\_title(Movie):

return re.sub("[^a-zA-Z0-9]", "", Movie)

**Explanation:**

* Defines a function, ‘**clean\_title’**, that takes a movie title (‘**Movie’**) and removes non-alphanumeric characters using a regular expression.

## **6. Applying Text Cleaning to DataFrame**

df["clean\_title"] = df["Movie"].apply(clean\_title)

**Explanation:**

Applies the ‘**clean\_title’** function to the 'Movie' column and creates a new column, 'clean\_title', in the DataFrame to store the cleaned movie titles.

## **7. Importing TF-IDF Vectorizer**

from sklearn.feature\_extraction.text import TfidfVectorizer

**Explanation:**

* Imports the TF-IDF vectorizer from scikit-learn. This will be used to convert movie titles into numerical vectors for analysis.

## **8. TF-IDF Vectorization**

# vectorizer = TfidfVectorizer(ngram\_range=(1, 2))

# t = vectorizer.fit\_transform(df["clean\_title"])

**Explanation:**

* Creates a TF-IDF vectorizer with unigrams and bigrams (‘**ngram\_range=(1, 2)’**) and transforms the cleaned movie titles into TF-IDF features (‘**t’**).

## **9. Importing Cosine Similarity and NumPy**

from sklearn.metrics.pairwise import cosine\_similarity

import numpy as np

**Explanation:**

* Imports the cosine\_similarity function for calculating similarity and NumPy for numerical operations.

## **10. User's Movie Preference**

Movie = "War"

movie = clean\_title(Movie)

query = vectorizer.transform([movie])

**Explanation:**

* Sets a user's movie preference to "War," cleans the title, and transforms it into TF-IDF features for comparison.

## **11. Cosine Similarity Calculation and Movie Recommendation**

similarity = cosine\_similarity(query, t).flatten()

indices = np.argpartition(similarity, -5)[-5:]

results = df.iloc[indices]

**Explanation:**

* Calculates cosine similarity between the user's preference and all movies in the dataset. Recommends the top 5 most similar movies based on similarity scores.

## **12. Displaying Indices of Recommended Movies**

Indices

**Explanation:**

* Displays the indices of the recommended movies based on cosine similarity scores.

## **13. Displaying Details of Recommended Movies**

results

**Explanation:**

* Displays details of the recommended movies, including columns such as 'Movie,' 'Year,' 'Genre,' 'Plot,' 'Runtime,' 'Rating,' and 'clean\_title.'

This concludes the documentation for your movie recommendation system project. If you have specific questions or if there's anything else you'd like to add, feel free to let me know!