MOVIE RECOMMENDATION SYSTEM

***AIML Project Report Submitted in partial fulfilment of the***

***Requirements for the award of the Degree of* BACHELOR OF ENGINEERING** IN

**INFORMATION TECHNOLOGY**

*BY*

**<RIDHI SUNKARA><1602-22-737-187>**

**<A.ASHRITHA><1602-22-737-136>**

Under the guidance of Mrs. Satyadevi



DepartmentofInformationTechnology Vasavi Colegeof Engineering(Autonomous)

(AffiliatedtoOsmaniaUniversity) Ibrahimbagh, Hyderabad-31 2020

**DECLARATION BY THE CANDIDATE**

We, **RIDHI SUNKARA**and **ASHRITHA.A**, bearing hall ticket numbers, **1602- 22- 737-187** and **1602-22-737-136**, hereby declare that the project report entitled **” MOVIE RECOMMENDATION SYSTEM”** Department of Information Technology, Vasavi College of Engineering, Hyderabad, is submitted in partial fulfilment of the requirement for the award of the degree of Bachelor of Engineering in Information Technology

This is a record of bonafide work carried out by me and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

**RIDHI SUNKARA<1602-22-737-187>**

**ASHRITHA.A<1602-22-737-136>**

**(Faculty In-Charge) (Head,DeptOfIt)**

# Introduction:

# A Movie Recommendation System is a type of content filtering system designed to suggest movies to users based on their preferences and interactions. By leveraging Natural Language Processing (NLP) and Vectorization models, these systems can analyze textual data, such as movie descriptions, reviews, or user preferences, to provide personalized recommendations.

# NLP Techniques:

# Text Preprocessing: Cleaning movie metadata (e.g., removing stop words, punctuation, and stemming/lemmatizing).

# Feature Extraction: Extracting meaningful features from unstructured text, such as movie titles, genres, or synopses.

# Vectorization Models:

# TF-IDF (Term Frequency-Inverse Document Frequency): A statistical measure to evaluate how important a word is in a document relative to a collection of documents.

# Word Embeddings: Techniques like Bag Of Words convert words or sentences into dense vectors that capture semantic meaning.

# Sentence Embeddings: Models like Sentence-BERT or Universal Sentence Encoder represent whole sentences as vectors.

# Workflow:

# Data Collection: Gather data on movies, including titles, genres, descriptions, and reviews.

# Preprocessing: Apply NLP techniques to process text, converting raw data into a clean and structured format.

# Vectorization:

# Convert the preprocessed text into numerical vectors using TF-IDF or embeddings.

# Ensure vectors capture contextual relationships between words, genres, or reviews.

# Similarity Computation:

# Use cosine similarity or other distance metrics to compute the similarity between movie vectors.

# Higher similarity values indicate closely related movies.

# Recommendation Generation:

# Based on user preferences or viewed movies, recommend movies with the highest similarity scores.

# Advantages of NLP and Vectorization:

# Contextual Understanding: Captures subtle relationships in movie metadata.

# Scalability: Efficiently handles large datasets with textual descriptions.

# Personalization: Adapts recommendations based on user-specific input or behavior.

# By combining the power of NLP and vectorization models, movie recommendation systems can go beyond simple metadata filtering and offer nuanced, highly personalized suggestions.

# Data Preprocessing Steps

# Load the Datasets:

# Import the datasets (tmdb\_5000\_movies.csv and tmdb\_5000\_credits.csv) into pandas DataFrames.

# Inspect the Data:

# Display the first few rows of each dataset using head() to understand their structure and key columns.

# Use info() and describe() to check for missing values, data types, and summary statistics.

# Merge Datasets:

# Combine movies and credits datasets on a common key (e.g., id) for a unified structure.

# Handle Missing Values:

# Identify columns with missing values using isnull().sum().

# Decide whether to drop rows/columns or fill missing values (e.g., using mean, mode, or placeholder text like "Unknown").

# Text Cleaning:

# For textual fields (e.g., movie overviews, titles):

# Convert to lowercase.

# Remove punctuation, special characters, and numerical digits.

# Remove stop words (e.g., "the," "and").

# Apply stemming or lemmatization to reduce words to their base forms.

# Feature Engineering:

# Create additional columns if necessary, such as combining genres, keywords, or cast names into a single feature.

# Tokenize text into words or phrases if working on individual components.

# Vectorization:

# Apply vectorization methods like TF-IDF or Count Vectorizer to convert text-based columns into numerical representations.

# Use word embeddings if needed for more semantic-rich representations.

# Dimensionality Reduction:

# Apply techniques like Principal Component Analysis (PCA) if the feature space becomes too large.

# Export Cleaned Data:

# Save the cleaned DataFrame for use in further modeling.

**Results and Discussion**

1. **Data Insights**:
   * The datasets (tmdb\_5000\_movies and tmdb\_5000\_credits) were successfully loaded, containing essential movie information like titles, genres, overviews, and credits (cast and crew details).
   * After preprocessing, missing values were handled, and relevant features like genres, keywords, and overviews were combined into a unified textual feature for recommendation.
2. **Textual Features**:
   * The processed text data was vectorized using **TF-IDF** (or other embeddings, as per implementation). This transformation captured the semantic relationships between movies based on their descriptions.
3. **Similarity Scores**:
   * Cosine similarity was computed between movie vectors to identify movies with similar content. For example:
     + Input movie: *Inception* → Recommended movies: *Interstellar*, *The Matrix*, *Shutter Island*.
4. **Recommendation Engine**:
   * The recommendation system was able to provide:
     + **Content-based recommendations**: Based on movie overviews, genres, and keywords.
     + **Dynamic recommendations**: Users can input their favorite movies to get personalized suggestions.
5. **Performance Metrics**:
   * The model demonstrated efficient processing for the dataset size, generating recommendations with minimal computational overhead.
   * The qualitative accuracy was validated by manual review, showing logical suggestions.

# CODE:

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# REFERENCES:

# -Nltk documentation

# -Scikit-learn Documentation