

**NAME : SACHIN SINGH**

**ROLL NUMBER : 2023BCS0064**

## **Artificial Intelligence Project Report**

### **Movie Genre Prediction using ANN**

#### **1. Abstract**

This project focuses on predicting the genres of a movie using Artificial Neural Networks (ANN) based on metadata such as title, overview, keywords, and cast. Since movies often belong to multiple genres, the task is modeled as a multi-label classification problem.

The system processes raw movie information, converts it into numerical features using TF-IDF and standardized numeric attributes, and trains a deep learning model to identify the most relevant genres. The model achieves strong performance metrics and demonstrates the practical application of ANN in text-based classification tasks.

#### **2. Introduction**

Movie genre classification plays a central role in recommendation systems, content organization, and user preference modeling. Modern datasets contain rich textual descriptions such as summaries, cast lists, and keywords, which can be used to infer a film's genre automatically.

This project utilizes **The Movies Dataset** from Kaggle, combining metadata, cast information, and descriptive keywords to build a robust ANN-based genre predictor. The model learns patterns in text and metadata to assign one or more genres to each movie.

A multi-label deep learning framework is adopted because movies frequently belong to multiple genres (e.g., Action + Thriller, Drama + Romance).

### **3. Objectives**

- Predict multiple genres for a movie using deep learning.
- Build and train an ANN model using TensorFlow/Keras.
- Perform comprehensive preprocessing of metadata, cast, keywords, and text.
- Use TF-IDF to convert movie descriptions into numerical vectors.
- Handle class imbalance through sample weighting.
- Evaluate performance using micro/macro F1-scores.
- Provide a prediction function that accepts new text descriptions.

### **4. Dataset Description**

The project uses The Movies Dataset (Kaggle: [rounakbanik/the-movies-dataset](https://www.kaggle.com/rounakbanik/the-movies-dataset)), which contains detailed metadata from The Movie Database (TMDb).

Key files used:

- **`movies_metadata.csv`** – titles, overviews, genres, runtime, popularity
- **`credits.csv`** – cast and crew information
- **`keywords.csv`** – descriptive keyword tags

Important columns extracted:

- Title
- Overview
- Genres (target labels)
- Keywords
- Cast (top 5 actors)
- Popularity
- Vote count
- Vote average
- Runtime

These files were merged using a cleaned and standardized **id** field.

## 5. Technologies Used

- Python
- TensorFlow / Keras
- Scikit-learn (TF-IDF, preprocessing, metrics)
- Pandas, NumPy
- Google Colab (execution environment)
- Kaggle API (dataset download)

## **6. Model Architecture**

The ANN was designed to handle high-dimensional TF-IDF input along with numeric metadata. The architecture includes:

- **Input Layer:** TF-IDF features + scaled numeric features
- **Hidden Layers:**
  - Dense(512, ReLU) + BatchNormalization + Dropout(0.5)
  - Dense(256, ReLU) + BatchNormalization + Dropout(0.35)
  - Dense(128, ReLU) + Dropout(0.25)
- **Output Layer:**
  - Dense(number\_of\_genres, activation='sigmoid') for multi-label output

**Loss Function:** Binary Crossentropy

**Optimizer:** Adam

The model outputs independent probabilities for each genre.

## **7. Data Preprocessing**

Steps performed:

### **1. Cleaning and merging:**

- Converted all **id** fields to numeric
- Removed invalid IDs

- Merged metadata, credits, and keywords using a left join

## 2. Parsing JSON-like fields:

- Extracted lists from **genres, keywords and cast**.
- Limited cast lists to top 5 names

## 3. Text construction:

- Combined title, overview, keywords, and cast into one text field

## 4. TF-IDF vectorization:

- Max 5000 features
- English stopwords removed

## 5. Numeric attributes:

- Popularity, vote\_average, vote\_count, runtime normalized using StandardScaler

## 6. Multi-label encoding:

- Used MultiLabelBinarizer to encode genres into binary vectors

## **8. Handling Class Imbalance**

Genres in the dataset are unevenly distributed.

To handle this:

- Computed class frequencies
- Derived inverse-frequency class weights

- Converted class weights into **sample weights**
- Used sample weights during model training

This approach improves performance on rare genres such as War, Western, and Music.

## **9. Training Process**

- Used **EarlyStopping** to halt training on no improvement
- **ReduceLROnPlateau** to lower learning rate dynamically
- Trained for up to 40 epochs with batch size 256
- Monitored validation loss throughout training

All training was conducted in Google Colab.

## **10. Results & Model Performance**

After applying genre-specific threshold tuning, the model achieved:

- **Micro F1-score:** ~0.62
- **Macro F1-score:** ~0.55
- **Micro Precision:** ~0.57
- **Micro Recall:** ~0.67

These results indicate strong generalization for a multi-label text classification problem.

## **11. Prediction Workflow**

The prediction function:

1. Converts new text using the trained TF-IDF vectorizer
2. Appends placeholder numeric features
3. Uses the ANN model to compute genre probabilities
4. Applies optimized thresholds per genre
5. Returns top-k genres and binary classification mask

Several test descriptions were evaluated, consistently producing sensible multi-genre predictions.

## **12. Instructions to Run the Project**

1. Upload your Kaggle API details or configure environment variables
2. Download the dataset using the Kaggle API
3. Run the preprocessing cells to merge and clean data, then train the model.
4. Evaluate using the provided metric functions
5. Use the prediction function to classify new movie summaries

The full implementation is provided in the **.ipynb** notebook.

## **13. Conclusion**

This project demonstrates the application of Artificial Neural Networks in a real-world multi-label classification problem. By combining textual metadata, TF-IDF features, and numeric attributes, the model achieves reliable movie genre predictions.

The approach is scalable and can be extended to recommendation systems or automated content tagging.

## **14. Future Enhancements**

- Incorporating transformer-based embeddings (e.g., BERT, RoBERTa)
- Improving rare-genre performance with advanced sampling techniques
- Adding explainability tools such as LIME or SHAP
- Deploying the model using a web interface or API
- Integrating user reviews or full plot summaries for richer context