Team Members

Student Name	Section	Code
Mohamed Shehab Eldeen Khalil	2	50
Mohamed Shady Aish	2	49
Mohamed Nasser Mohamed	باقى تالتة	...
Saleh Saber Ebrahem	1	23
Mahmoud Mostafa Mohamed Khayal	2	55
Mostafa Ezzat Abd ElNaeem	2	58
Moataz Hamdy Ali	2	59
Yossef Mohamed Helmy	3	71
Yossef Essam Fawzy	3	68

Movie Audience Score Prediction

1. Project Overview

This project aims to **predict the Audience Score (0–10)** of a movie based on a combination of:

- **Textual features** extracted from the movie overview
- **Numerical metadata** such as budget, revenue, popularity, etc.

The system consists of:

1. A **Jupyter Notebook** for data analysis, feature engineering, model training, and evaluation.
2. A **Streamlit web application** that loads the trained artifacts and provides real-time predictions.

---

2. Dataset & Target Variable

- **Target Variable:** audience_score
- **Problem Type:** Regression

---

3. Data Preprocessing

3.1 Text Cleaning

The movie overview text is processed using:

- Lowercasing
- Removal of digits
- Removal of punctuation
- Stopwords removal (NLTK)
- Lemmatization using WordNetLemmatizer

clean_text(text)

An additional feature is created:

- **overview_length** → number of words after cleaning

3.2 Numerical Features

- The following numerical features are used:

Feature
Budget
Revenue
Runtime
Popularity
Vote Count
Release Year
Release Month
Number of Genres
Number of Keywords
Number of Main Cast
Overview Length

- All numerical features are **scaled using StandardScaler**.

---

4. Feature Engineering

4.1 Text Vectorization

- **TF-IDF Vectorizer** is applied to the cleaned overview text.
- Produces a sparse high-dimensional feature matrix.

4.2 Feature Combination

Final input matrix is created by:

[TF-IDF Text Features] + [Scaled Numerical Features]

using scipy.sparse.hstack.

---

5. Models Trained

Multiple regression models were experimented with. The most important ones are:

- Random Forest Regressor (Best model – saved)
- XGBoost Regressor

The **best performing model** was selected based on test set metrics and saved for deployment.

---

6. Model Evaluation Metrics

The following regression metrics are used:

- **MAE (Mean Absolute Error)**
- MSE(**Mean Squared Error**)
- **RMSE (Root Mean Squared Error)**
- **R^2 Score**

---

7. Model Performance Summary

7.1 XGBoost Regressor Performance

Metric	Train	Test
MAE	0.299	0.565
RMSE	0.382	0.784
R^2	0.901	0.475

Observation:
High training performance but noticeable drop on test data → some overfitting.

7.2 Random Forest Regressor (Best Model)

Metric	Train	Test
MAE	0.203	0.526
RMSE	0.281	0.750
R^2	0.946	0.519

Selected Model: RandomForestRegressor

- Better generalization
- Lower MAE and RMSE on test data

- Higher R^2 compared to XGBoost

---

8. Model Artifacts Saved

The following artifacts are persisted using joblib:

- tfidf_vectorizer.pkl
- scaler.pkl
- best_rf_model.pkl

---

9. Streamlit Application (app.py)

9.1 Purpose

Provides a **user-friendly UI** to predict audience score using the trained model.

9.2 Application Flow

1. Load artifacts using cached function
2. Accept user inputs:
    - Movie overview (text)
    - Numerical movie details
3. Apply:
    - Text cleaning
    - TF-IDF transformation
    - Numerical scaling
4. Merge features
5. Predict using Random Forest model
6. Display score (0–10)

9.3 Output

Predicted Audience Score: X.XX / 10

10. Final Conclusion

- Combining **NLP features + structured data** significantly improves prediction quality.
- Random Forest provided the best balance between bias and variance.
- The pipeline is production-ready and successfully deployed via Streamlit.

---

11. Deep Learning Experiment – GRU Model

11.1 GRU Model Overview

In addition to traditional machine learning models, a **GRU (Gated Recurrent Unit)** deep learning model was experimented with to capture sequential patterns in the movie overview text.

The GRU model was designed to:

- Process the movie overview as a sequence of tokens
- Learn contextual and semantic relationships in text
- Combine text-based representations with numerical movie features

11.2 Training Challenges

Although the GRU model was successfully implemented and tested, it presented several practical challenges:

- **Very long training time** compared to classical ML models
- High computational cost
- Slower experimentation and tuning cycles

Due to these limitations, the GRU training code was **commented out** in the notebook to keep the workflow efficient and reproducible.

11.3 Performance Comparison

After evaluation, the GRU model **did not outperform** the Random Forest Regressor in terms of key regression metrics.

Model	MAE (Test)	RMSE (Test)	R^2 (Test)
GRU	Lower performance than RF	Lower performance than RF	Lower performance than RF
Random Forest	**0.526**	**0.750**	**0.519**

Key Insight:
Despite GRU's ability to model sequential text data, the **Random Forest model achieved better overall metrics**, especially on the test set, while being significantly faster to train and easier to deploy.

11.4 Final Model Selection Decision

The GRU model was **not used in the final deployment** for the following reasons:

- Inferior evaluation metrics compared to Random Forest
- Excessive training time
- Higher system complexity with no performance gain

As a result, **Random Forest Regressor** was selected as the final production model due to:

- Better generalization
- Higher evaluation metrics
- Faster inference
- Simpler and more robust deployment

11.5 Professional Takeaway

- This experiment demonstrates a critical machine learning principle:
- **More complex models do not always guarantee better performance.** Model selection should be driven by empirical evaluation, not model complexity.

GitHub_Repo for the project: https://github.com/moshehab99/Movie-Audience-Score-Predictor