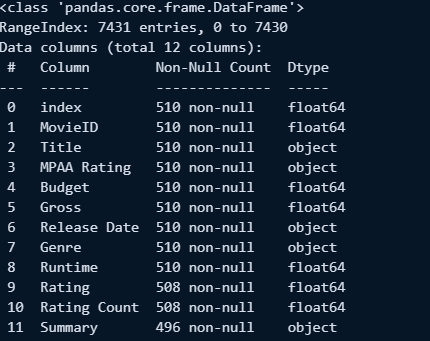
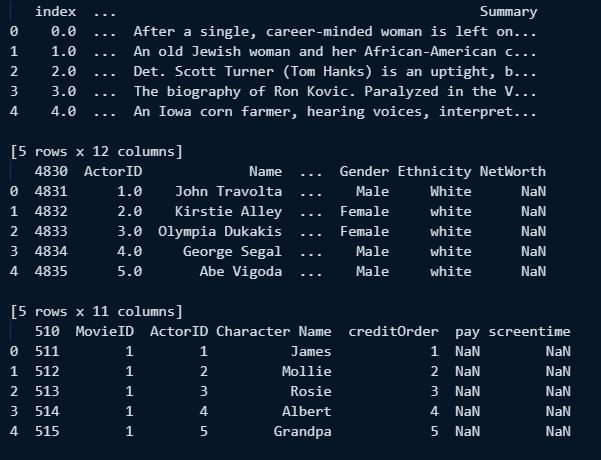
Movie Performance Predictor Documentation

# Web url: <https://movieperformancepredictorapp.streamlit.app/>

GitHub url: https://github.com/HasanFarajov/Movie\_Performance\_Predictor\_APP

# We have movies, actors and movieactors dataset:





# First, we developed a model that predicts movie revenue based only on movie attributes — we call this the Basic Model. Later, we improved the model by adding actor-related information such as the number of actors, the ratio of male to female actors, and their average height — this is known as the Advanced Model. Both models were trained, saved using joblib, and connected to an interactive web application built with Streamlit, a Python library for building data-driven apps.

# 1. Basic Model Training (model.py)

The Basic Model is trained using core movie-level attributes such as:  
- Budget  
- MPAA Rating  
- Genre  
- Runtime  
- IMDb Rating  
- Number of IMDb Ratings  
  
We applied preprocessing pipelines to handle missing values and to encode categorical data using OneHotEncoding.  
A RandomForestRegressor was used for modeling, and the performance was evaluated using Mean Absolute Error (MAE).  
The trained pipeline was saved as ‘movie\_gross\_predictor.pkl’.

1. Load the Dataset   
df\_new = pd.read\_csv(os.path.join("movies", "movies1.csv"))

Loads the movie dataset containing details like budget, genre, rating, runtime, and gross revenue. Only entries with a valid target (‘Gross’) are kept.

2. Data Cleaning   
df\_model = df\_new.dropna(subset=['Gross'])  
Removes rows where the target value (Gross revenue) is missing.

3. Feature Selection   
X = df\_model[['Budget', 'MPAA Rating', 'Genre', 'Runtime', 'Rating', 'Rating Count']]  
y = df\_model['Gross']  
  
Selects the relevant input features (‘X’) and defines the target variable (‘y’).

4. Train/Test Split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
Splits the dataset into training and testing sets (80/20 split).

5. Data Preprocessing  
Handles missing data and categorical encoding using pipelines.

6. Model Definition and Training  
pipeline = Pipeline([...])  
pipeline.fit(X\_train, y\_train)  
  
Trains a RandomForestRegressor using the processed data.

7. Evaluation  
mae = mean\_absolute\_error(y\_test, y\_pred)

Calculates the Mean Absolute Error (MAE).

8. Save the Model  
joblib.dump(pipeline, "models/movie\_gross\_predictor.pkl")

Saves the trained model for use in the Streamlit app.

# 2. Advanced Model Training (model2.py)

The Advanced Model improves prediction accuracy by including actor-based features.  
To achieve this, we merged actor data with movie-actor linking data, and then calculated:  
- Actor Count (per movie)  
- Average Height (of actors)  
- Male Ratio (proportion of male actors)  
  
These features were then combined with the same movie-level features from the Basic Model.  
After preprocessing and training with RandomForestRegressor, the model was saved as ‘movie\_gross\_predictor\_v2.pkl’.

1. Load Datasets  
Loads movie, actor, and linking datasets using pd.read\_csv.

2. Merge Actor Data  
merged\_df = movies\_actors\_df.merge(actors\_df, on="ActorID", how="left")  
Connects actor data with movies.

3. Extract Actor Features  
Calculates actor count, average height, and male ratio per movie.

4. Merge Actor Features with Movie Data  
df = movies\_df.merge(actor\_features, on="MovieID", how="left")  
  
Combines actor metrics with movie-level data.

5. Preprocess and Train Model  
Builds preprocessing and RandomForest pipeline and fits it.

6. Save Model  
joblib.dump(pipeline, "models/movie\_gross\_predictor\_v2.pkl")  
  
Model is saved for Advanced Mode in the Streamlit app.

# 3. Streamlit Web App (app.py)

We created a user-friendly web interface using Streamlit that allows users to interact with both models.  
  
The app includes:  
- A radio selector to choose between Basic and Advanced prediction modes.  
- Interactive input fields for movie information.  
- A sidebar for filtering and selecting actors (in Advanced mode).  
- API integration with Wikipedia to fetch actor profiles and display:  
 - Biography  
 - Photo  
 - Birthplace  
 - Net Worth  
 - Other public information  
  
The app also includes dynamic visualizations such as:  
- A gender distribution pie chart of selected actors  
- A histogram of actor height  
  
The app uses session state to retain selected actors and ‘functools.lru\_cache’ to optimize Wikipedia API calls.

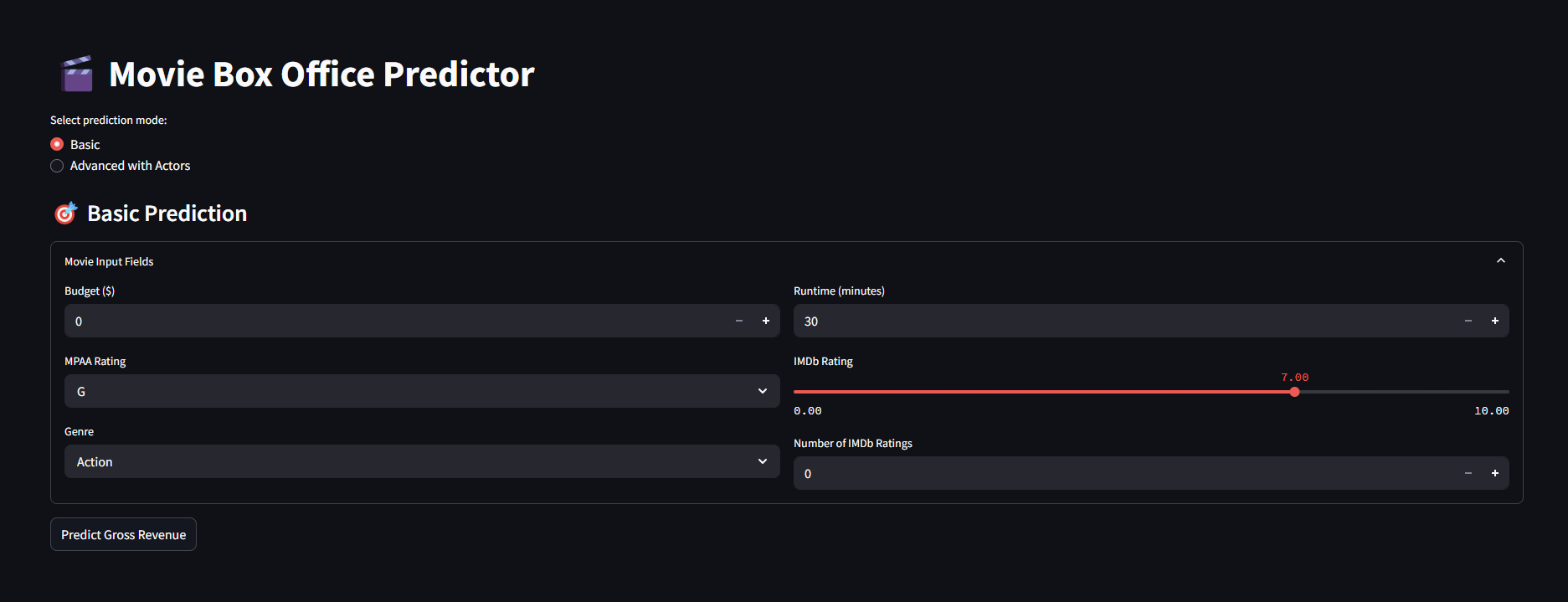
1. Load Models and Data  
basic\_model = joblib.load(...)  
advanced\_model = joblib.load(...)  
  
Models and datasets are loaded at the start.

2. UI Setup  
st.set\_page\_config(layout="wide")  
st.radio(...)  
  
Defines layout and mode selection.

3. Basic Mode  
- Users input budget, runtime, genre, rating, etc.  
- Model predicts gross revenue using ‘basic\_model’

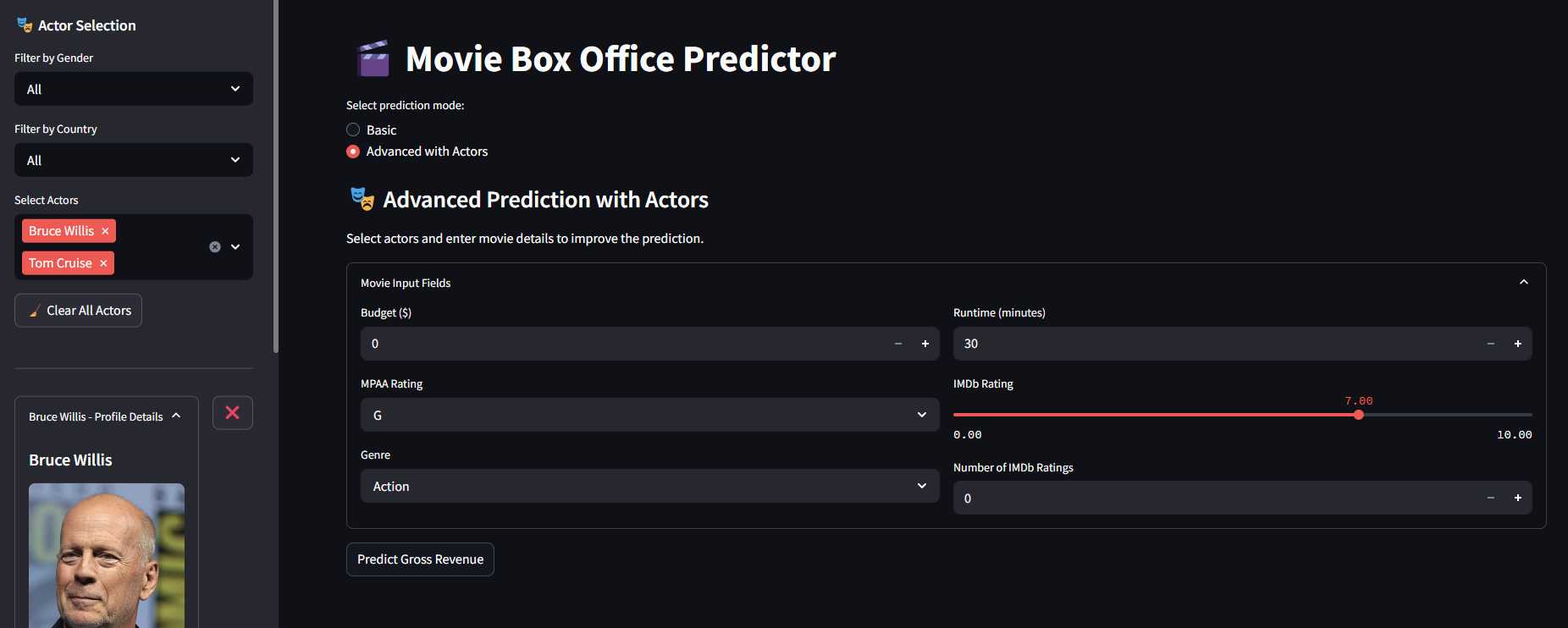
4. Advanced Mode  
- Users filter and select actors  
- Calculates ‘Actor Count’, ‘Avg Height’, ‘Male Ratio’  
- Uses Wikipedia API to retrieve actor details  
- Displays profile and visual statistics  
- Predicts with ‘advanced\_model’

5. Caching and Session State  
- API results are cached for performance  
- Actor selections stored in session for continuity



A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A blue circle and a blue circle with a bar chart

AI-generated content may be incorrect.

After making a prediction in the **Advanced Model**, the app shows a summary of the selected actors:

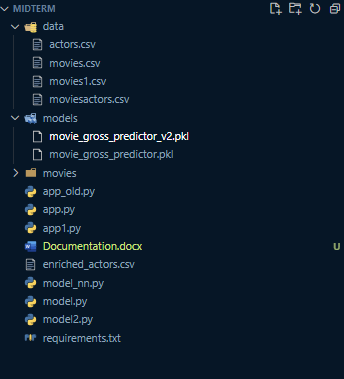
* **Left chart:** A pie chart showing the gender distribution. In this case, all selected actors are male (100%).
* **Right chart:** A bar chart showing actor height. It shows how tall the selected actors are.

These charts help the user better understand the group of actors used in the prediction.

**4. Summary**

This project combines data preprocessing, machine learning, and interactive visualization to create an effective tool for predicting movie box office revenue. It shows how using features like actor demographics can improve prediction accuracy when applied correctly.

ADDITIONAL:



A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program code

AI-generated content may be incorrect.