MOVIE HIT OR FLOP PREDICTION

Machine Learning

By

Sarvagya Sanjay

Introduction & Objectives

Predicting whether a movie will be a hit or a flop based on several parameters like budget, cast, director, genre, release date, marketing activities, and audience perception is quite challenging. By analyzing features such as **budget, genre, cast popularity, director reputation, marketing spends**, and more, this project aims to develop machine learning models that can forecast the commercial outcome of a movie before its release. Such a system could help producers, investors, and marketing teams make informed decisions to reduce financial risk.

OBJECTIVES

To collect and preprocess movie-related data from reliable sources (IMDb, TMDb, Box Office Mojo, etc.).

To build classification models (Gradient Boosting, SVM, Random Forest) to predict movie success as Hit or Flop.

To analyze key factors that influence a movie's performance using feature importance techniques.

To evaluate the performance of the models using metrics like Accuracy, Precision, Recall, and F1-Score.

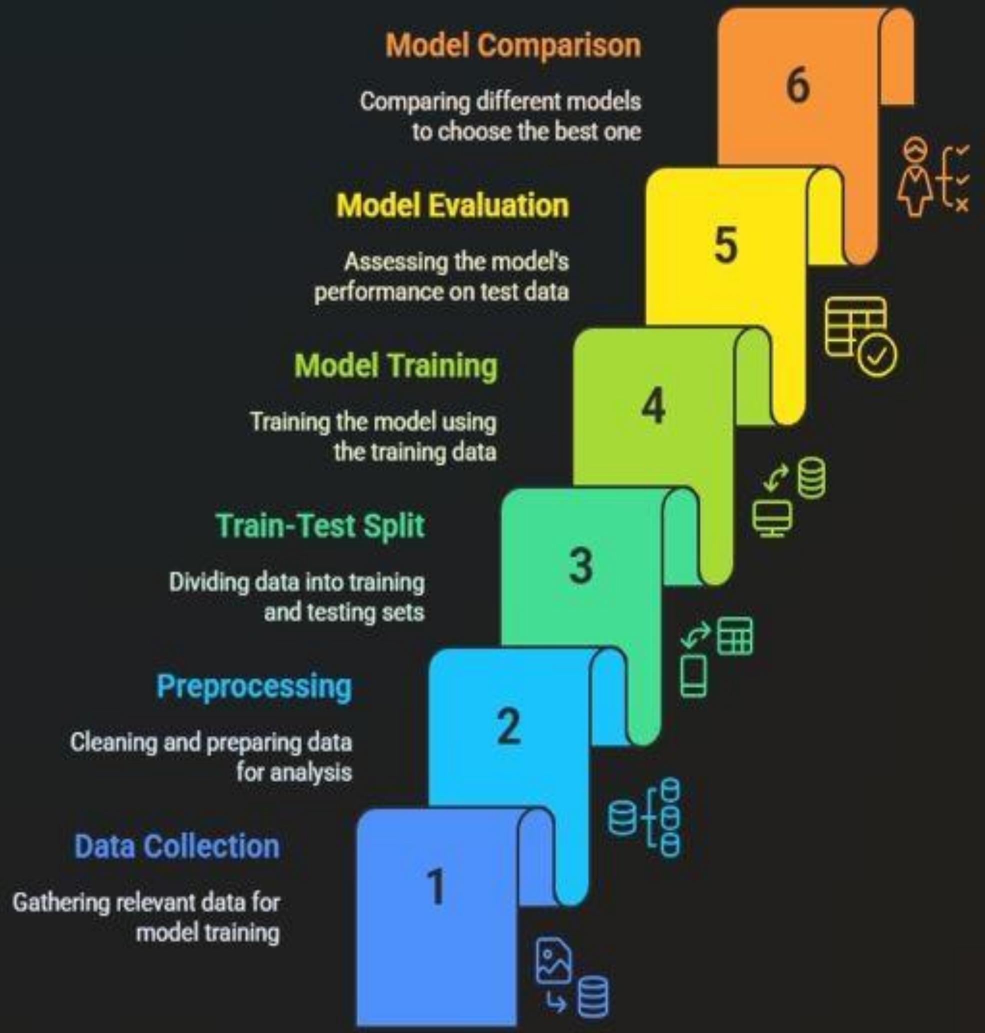
APPLICATIONS

Investors/Producers: Assists in investment decisions based on likely box office outcomes.

Marketing Teams: Allows better allocation of marketing budgets based on predicted movie performance.

Recommendation Engines: Can be integrated with streaming platforms to promote high-potential content.

UNDERSTANDING THE WORKFLOW



NOVELTY followed by METHODOLOGY

"The novelty of our project lies in not just using numerical values like budget and revenue, but also incorporating non-numeric features like cast, director, and genre, and applying One-Hot Encoding to convert them into machine-readable format — allowing the model to learn deeper patterns behind a movie's success."

1. Dataset collection- This is raw material to build the prediction model.

2. Preprocessing-

Handling Missing Values: Fill or drop rows where info like budget or ratings is missing.

Encoding Categorical Features: Convert text categories like genre (Action, Drama) into numbers using Label Encoding or One-Hot Encoding.

Feature Engineering: Normalize/scale numerical features like budget and marketing spend

3. Model Training (*Gradient boosting, SVM, Random Forest)*

three different models:

Gradient boosting: Trees are trained sequentially, focusing on difficult-to-predict samples.
SVM: Classifier that handles complex boundaries.
Random Forest: Tree-based ensemble learning for deep patterns.

Each model learns patterns in the training data that separate Hit and Flop movies.

4. Model Evaluation *(Accuracy & Classification Report)*

- After training, testing the models on the 20% test data.

ML Models

ML MODEL	PURPOSE	HOW IT WORKS	USE CASE
GRADIENT BOOSTING	Predicts whether a movie is a hit or flop by combining many small decision trees that each learn from the mistakes of the previous ones.	It builds decision trees **sequentially**, where each new tree focuses on **correcting the errors** made by the previous ones, boosting overall performance.	Trees are trained **sequentially**
SVM	Finds the best boundary (hyperplane) that separates different classes.	Maximizes the margin between the closest points of the two classes (support vectors).	If the data is separable by a straight line, SVM will find the **best possible margin** between the two classes. If it's more complex (non-linear), SVM can use **kernels**.
RANDOM FOREST	Ensemble method using many decision trees.	Combines results of many decision trees to make a final prediction (majority vote)	Final prediction is the majority of all trees' outputs. If 80 trees say Hit and 20 say Flop → Final output is **Hit**.

IMPLEMENTATION

Taking initial 10000 samples

```python
import pandas as pd
```

```python
df = pd.read_csv("TMDB  IMDB Movies Dataset.csv")
df = df.head(10000)
df
```

0	27205	Inception	8.364	34495	Released	2010-07-15	825532764	148	False	/8ZTVqvKDQ8emSGUEMjsS4yHAwrp.jpg	...	Action, Science Fiction, Adventure	Legendary Pictures, Syncopy, Warner Bros. Pict...	United Kingdom, United States of America	English, French, Japanese, Swahili	rescue, mission, dream, airplane, paris, franc...	Christopher Nolan
1	157336	Interstellar	8.417	32571	Released	2014-11-05	701729206	169	False	/pbrkL804c8yAv3zBZR4QPEafpAR.jpg	...	Adventure, Drama, Science Fiction	Legendary Pictures, Syncopy, Lynda Obst Produc...	United Kingdom, United States of America	English	rescue, future, spacecraft, race against time,...	Christopher Nolan
2	155	The Dark Knight	8.512	30619	Released	2008-07-16	1004558444	152	False	/nMKdUUepR0i5zn0y1T4CsSB5chy.jpg	...	Drama, Action, Crime, Thriller	DC Comics, Legendary Pictures, Syncopy, Isobel...	United Kingdom, United States of America	English, Mandarin	joker, sadism, chaos, secret identity, crime f...	Christopher Nolan
3	19995	Avatar	7.573	29815	Released	2009-12-15	2923706026	162	False	/vL5LR6WdxWPjLPFRLe133jXWsh5.jpg	...	Action, Adventure, Fantasy, Science Fiction	Dune Entertainment, Lightstorm Entertainment, ...	United States of America, United Kingdom	English, Spanish	future, society, culture clash, space travel, ...	James Cameron
4	24428	The Avengers	7.710	29166	Released	2012-04-25	1518815515	143	False	/9BBTo63ANSmhC4e6r62OJFuK2GL.jpg	...	Science Fiction, Action, Adventure	Marvel Studios	United States of America	English, Hindi, Russian	new york city, superhero, shield, based on com...	Joss Whedon

1. Features present in the original dataset
2. Insignificant features

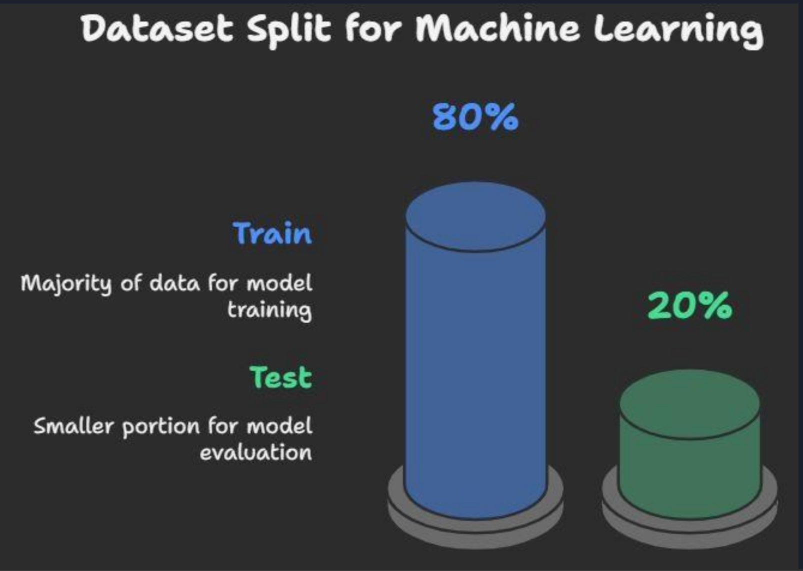
```
[ ] df.columns
```

```
Index(['id', 'title', 'vote_average', 'vote_count', 'status', 'release_date',
       'revenue', 'runtime', 'adult', 'backdrop_path', 'budget', 'homepage',
       'tconst', 'original_language', 'original_title', 'overview',
       'popularity', 'poster_path', 'tagline', 'genres',
       'production_companies', 'production_countries', 'spoken_languages',
       'keywords', 'directors', 'writers', 'averageRating', 'numVotes',
       'cast'],
      dtype='object')
```

Random Forest creates many decision trees, each working on different subsets of the data, and then having a majority vote to make the final prediction. It's very good with messy and complex data but will take longer time to run.

```
columns_to_drop = [
    "id", "tconst", "stat
    "poster_path", "homep                              _companies",
    "production_countries", "spoken_languages", "keywords", "writers",
    "averageRating", "numVotes", "vote_average", "vote_count", "popularity"
]

df = df.drop(columns=columns_to_drop)
# df.shape  # Check the new shape
df.head()  # Verify remaining columns
```

	title	release_date	revenue	runtime	budget	original_language	original_title	genres	directors	cast
0	Inception	2010-07-15	825532764	148	160000000	en	Inception	Action, Science Fiction, Adventure	Christopher Nolan	Leonardo DiCaprio, Joseph Gordon-Levitt, Ken W...
1	Interstellar	2014-11-05	701729206	169	165000000	en	Interstellar	Adventure, Drama, Science Fiction	Christopher Nolan	Matthew McConaughey, Anne Hathaway, Michael Ca...
2	The Dark Knight	2008-07-16	1004558444	152	185000000	en	The Dark Knight	Drama, Action, Crime, Thriller	Christopher Nolan	Christian Bale, Heath Ledger, Aaron Eckhart, M...
3	Avatar	2009-12-15	2923706026	162	237000000	en	Avatar	Action, Adventure, Fantasy, Science Fiction	James Cameron	Sam Worthington, Zoe Saldaña, Sigourney Weaver...
4	The Avengers	2012-04-25	1518815515	143	220000000	en	The Avengers	Science Fiction, Action, Adventure	Joss Whedon	Robert Downey Jr., Chris Evans, Mark Ruffalo...



Preparing Dataset for Training

Condition:

hit_or_flop = revenue >= 2(budget)

Obtaining a new dataset with updated number of features…

"preprocessed_movies.csv"

Splitting the DataSet (80 - 20)

```
from sklearn.model_selection import train_test_split

# Define features (X) and target variable (y)
X = df.drop(columns=["hit_or_flop"])  # Drop the target column
y = df["hit_or_flop"]  # Target variable (Hit = 1, Flop = 0)

# Split dataset into train (80%) and test (20%)
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

```
[ ]  df = pd.read_csv("preprocessed_movies.csv")
```

	genres
0	[Action, Science Fiction, Adventure]
1	[Adventure, Drama, Science Fiction]
2	[Drama, Action, Crime, Thriller]
3	[Action, Adventure, Fantasy, Science Fiction]
4	[Science Fiction, Action, Adventure]

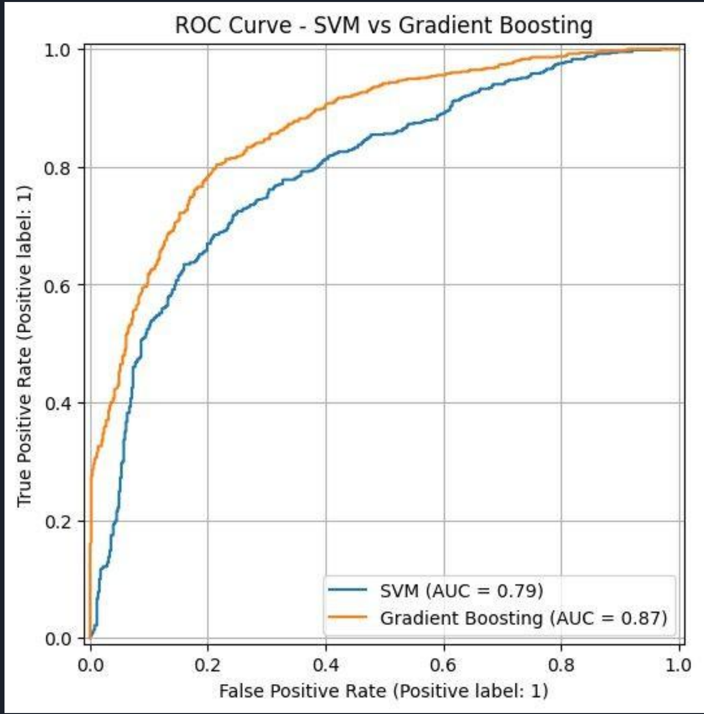
Obtain complete details about a movie

```
[ ] # Search for a movie by title
    movie_name = "Inception"  # Change this to any movie title you want

    # Find the movie in the dataset
    movie_details = df[df["title"].str.contains(movie_name, case=False, na=False)]

    # Display the first matching row
    movie_details.T  # Transpose to view better
```

	0
title	Inception
revenue	825532764
runtime	148
original_language	en
original_title	Inception
genres	[Action, Science Fiction, Adventure]
directors	Christopher Nolan
cast	Leonardo DiCaprio, Joseph Gordon-Levitt, Ken W...
hit_or_flop	1
release_year	2010
release_month	7

MODEL IMPLEMENTATION & RESULTS

Comparison of Machine Learning Model Accuracies



MODEL IMPLEMENTATION & RESULTS

```python
# Load Dataset
df = pd.read_csv("preprocessed_movies.csv")

# Define Features and Target
features = ["genres", "directors", "cast","revenue"]
target = "hit_or_flop"

# Convert Categorical Features into Numerical using One-Hot Encoding
df_encoded = pd.get_dummies(df[features])
```

Features= [“genres” , “directors” ,”cast”,”revenue”]

target= hit_or_flop

MODEL 1 GRADIENT BOOSTING

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, classification_report

# Separate features and target
X = df.drop("hit_or_flop", axis=1)
y = df["hit_or_flop"]

# One-Hot Encode non-numeric (categorical) columns
X_encoded = pd.get_dummies(X, drop_first=True)

# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

# Initialize and train Gradient Boosting model
gb_model = GradientBoostingClassifier(
    n_estimators=200,
    learning_rate=0.1,
    max_depth=4,
    random_state=42
)
gb_model.fit(X_train, y_train)

# Predict on test data
y_pred_gb = gb_model.predict(X_test)

# Evaluate model
print(f"📊 Gradient Boosting Accuracy: {accuracy_score(y_test, y_pred_gb):.4f}")
print("\n📄 Classification Report:")
print(classification_report(y_test, y_pred_gb))
```

```
Gradient Boosting Accuracy: 0.8501
GB - Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.85      0.83       771
           1       0.88      0.85      0.86       970

    accuracy                           0.85      1741
   macro avg       0.85      0.85      0.85      1741
weighted avg       0.85      0.85      0.85      1741
```

Gradient boosting accuracy: 0.85

How does it work?

1. Transforms features (genres, directors, cast) into a high-dimensional space using RBF
2. Finds an optimal hyperplane that separates hit (1) and flop (0) movies.
3. Classifies new movies based on their position relative to the hyperplane.

The **RBF kernel** uses the formula: $K(x, x') = \exp\left(-\gamma \|x - x'\|^2\right)$

- **If two movies are similar (similar cast, director, genre)** → Their **kernel value is high** → More likely to have the same class (hit or flop).
- **If two movies are very different** → Kernel value is low → More likely to be classified differently.

MODEL 2- STATE VECTOR MACHINE

using rbf

```
from sklearn.svm import SVC

# Train SVM model
svm_model = SVC(kernel='rbf', random_state=42)
svm_model.fit(X_train, y_train)

# Predictions
y_pred_svm = svm_model.predict(X_test)

# Accuracy
svm_accuracy = accuracy_score(y_test, y_pred_svm)
print(f" SVM Accuracy: {svm_accuracy:.4f}")

# Classification Report
print("\n SVM - Classification Report:")
print(classification_report(y_test, y_pred_svm))
```

```
SVM Accuracy: 0.7898

SVM - Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.79      0.77       771
           1       0.83      0.79      0.81       970

    accuracy                           0.79      1741
   macro avg       0.79      0.79      0.79      1741
weighted avg       0.79      0.79      0.79      1741
```

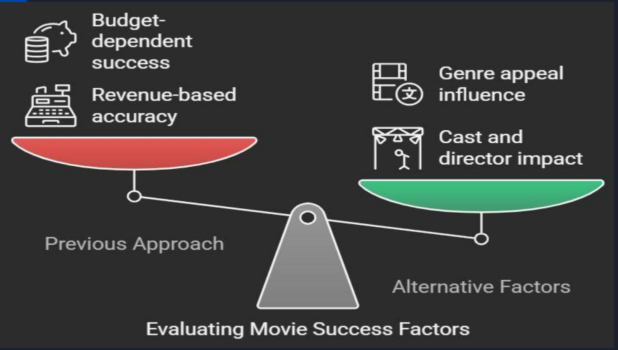
MODEL 3- RANDOM FOREST

```
from sklearn.ensemble import RandomForestClassifier

# Train Random Forest model
rf_model = RandomForestClassifier(n_estimators=500, max_depth=10, random_state=42, class_weight="balanced")
rf_model.fit(X_train, y_train)

# Predictions
y_pred_rf = rf_model.predict(X_test)

# Accuracy
rf_accuracy = accuracy_score(y_test, y_pred_rf)
print(f"📌 Random Forest Accuracy: {rf_accuracy:.4f}")

# Classification Report
print("\n📊 Random Forest - Classification Report:")
print(classification_report(y_test, y_pred_rf))
```

```
📌 Random Forest Accuracy: 0.8820

📊 Random Forest - Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        21
           1       0.88      1.00      0.94       157

    accuracy                           0.88       178
   macro avg       0.44      0.50      0.47       178
weighted avg       0.78      0.88      0.83       178
```

Random Forest Accuracy: 0.8820

MODIFICATIONS?



Previous Approach

Accuracy was primarily based on:

Revenue, Popularity, Budget, IMDB Ratings

Limitation:

- These factors alone might not fully capture what makes a movie a **hit or flop**.
- Some movies gain success due to **strong cast, director reputation, or genre appeal**, even if their budget is low.

New Modifications

incorporated additional key factors:

- **Cast & Director Influence** (e.g., Certain directors/actors have a strong fan following)
- **Genre Impact** (e.g., Horror movies may have lower budgets but still be profitable)
- **Combination of Multiple Factors** (Cast + Genre + Director)

Why This is Better?

- Movies with an **A-list cast & top director** often perform better, even if their budget is low.
- Certain genres (e.g., horror or comedy) **consistently perform well** despite lower production costs.
- **More holistic accuracy calculation** → Instead of just revenue/popularity, we are considering **what actually influences success**.

WHY IS THIS MODEL MORE VALUABLE?

Model using genre, cast, and directors is more valuable for early-stage movie planning, where revenue and ratings are unknown. It helps studios make better casting and genre decisions before production begins

Features like revenue, budget, and average voting are **direct numerical indicators** of a movie's success. They directly correlate with the hit/flop outcome, making it easier for the model to achieve high accuracy.

Even if accuracy is lower, a model using **genre, cast, and director** can still be **more useful in real-world applications** because:

Predicts success before production → Investors want to know a movie's potential *before* budget and revenue data exist. **More explainable insights** → Helps studios decide which actors, directors, or genres are most profitable. **Avoids data leakage** → Revenue and voting data are *post-release* features, which might create **data leakage** (giving unfair advantages in predictions).

References

- A. Anand, "Predicting the Success of a Movie Using Machine Learning Algorithms: An Analysis," *International Journal for Multidisciplinary Research (IJFMR)*, vol. 5, no. 6, pp. 1–5, Nov.–Dec. 2023
- R. Kaur and A. Kumari, "Movie Success Prediction using Machine Learning Algorithms and their Comparison," in *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, Greater Noida, India, Feb. 2021, pp. 1–5.
- M. Agarwal, S. Venugopal, R. Kashyap, and R. Bharathi, "A comprehensive study on various statistical techniques for prediction of movie success," *arXiv preprint arXiv:2112.00395*, Dec. 2021.