# Movie Box Office Revenue Prediction Project Overview

## Introduction

This project builds a regression model to predict the worldwide box office revenue of movies based on metadata such as release year, genre, MPAA rating, user vote count, original language, and production countries. Using a Random Forest regressor wrapped in a scikit-learn pipeline, we transform raw inputs into features, train on historical data, and evaluate predictive performance.

## Dataset

- **Source**: A CSV file (box_office_data.csv) containing the top 1000 (or more) movies by worldwide gross.
- **Columns**:
    - Rank – box office ranking
    - Release Group – movie title
    - $Worldwide, $Domestic, $Foreign – revenues in USD
    - Domestic %, Foreign % – revenue splits
    - Year – release year
    - Genres – comma-separated list (e.g., “Action, Drama”)
    - Rating – MPAA rating (G, PG-13, R, etc.)
    - Vote_Count – IMDb vote count
    - Original_Language – ISO code (e.g., “en”)
    - Production_Countries – comma-separated list

## Preprocessing & Methodology

1. **Missing Data Handling**
    - Drop any rows missing the target ($Worldwide).
    - Drop or impute remaining missing values.

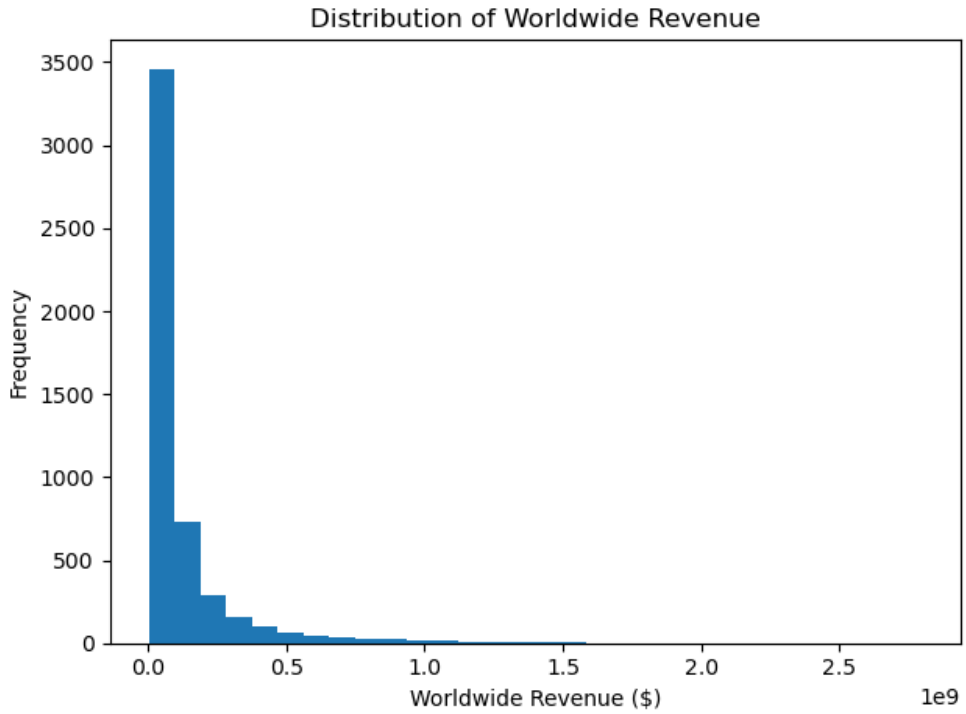
2. **Feature / Target Split**
    - **Target**: y = $Worldwide
    - **Features**: X = [Year, Genres, Rating, Vote_Count, Original_Language, Production_Countries]
3. **Pipeline Construction**
    - **Numerical Transformer**:
        - StandardScaler on Year and Vote_Count
    - **Categorical Transformer**:
        - OneHotEncoder (ignore unknowns) on Genres, Rating, Original_Language, Production_Countries
    - **Model**:
        - RandomForestRegressor(n_estimators=100, random_state=42)
4. **Train/Test Split**
    - 80% train / 20% test, random_state=42 for reproducibility
5. **Training & Prediction**
    - Fit pipeline on training data
    - Predict on test set

# Visualizations

1. **Actual vs. Predicted Scatter Plot**

- A scatter of each test-set actual revenue (y_test) against its predicted revenue (y_pred), overlaid with a 45° dashed line.

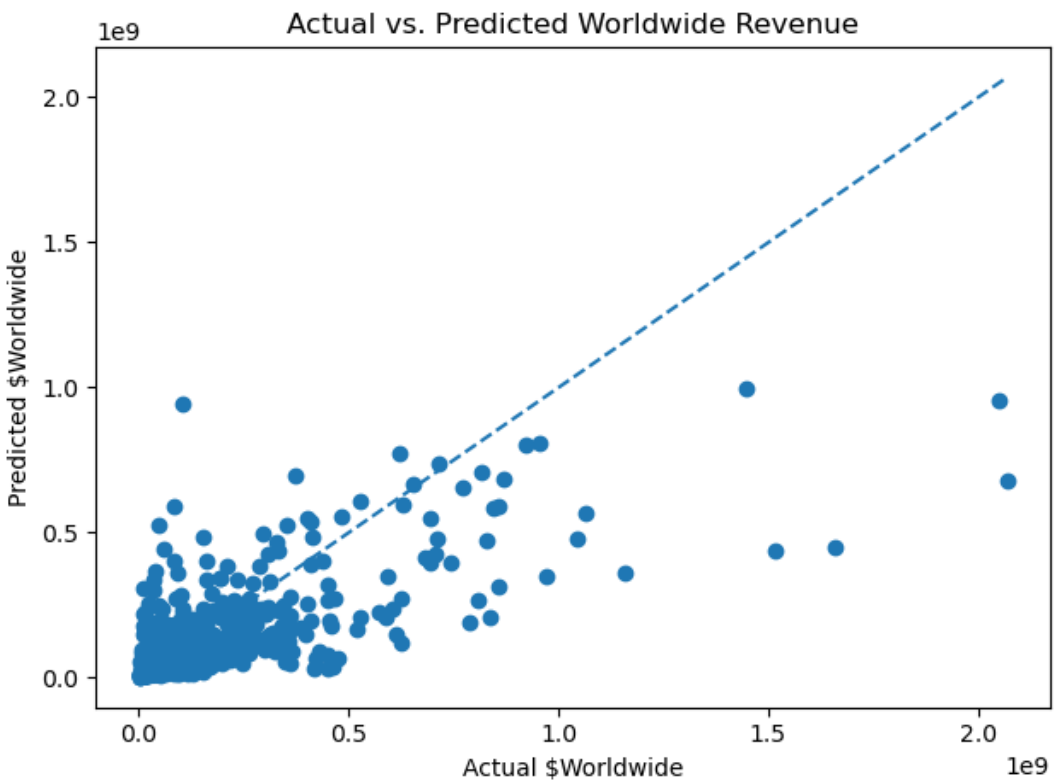
- **Shows:** how closely the model's predictions track the true box-office figures and highlights any outlier points where the model over- or under-predicts.




2. **Residuals Distribution Histogram:**

   A histogram of the residuals (y_test – y_pred) across the test set.

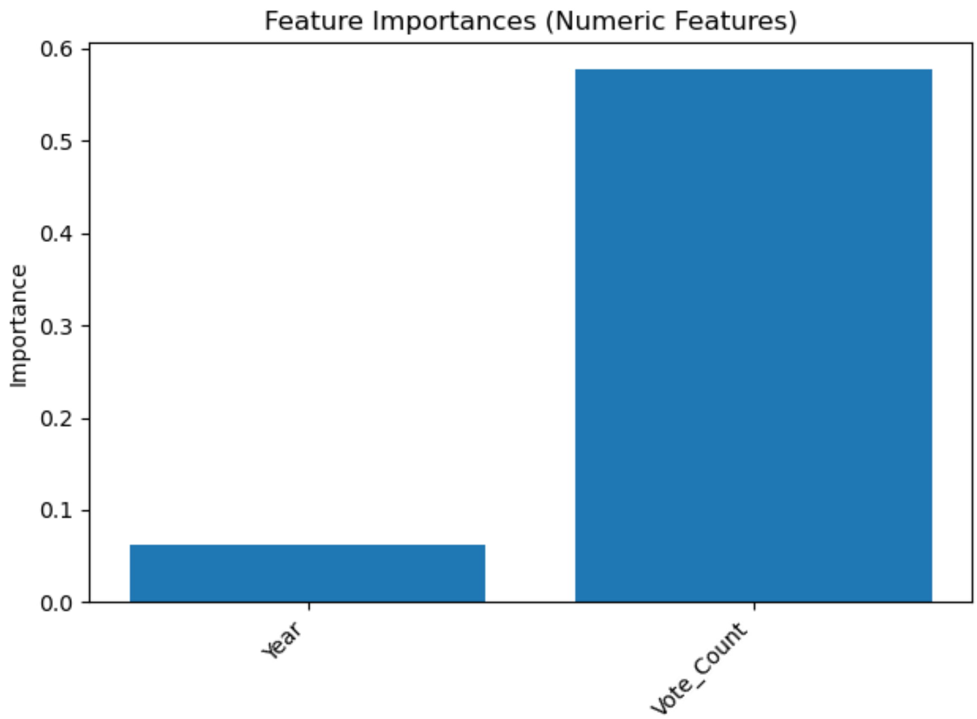
   **Shows:** the model's bias (the mean of the residuals—whether it tends to over- or under-estimate) and the variance (the width of the distribution—how consistent its errors are).



3. **Feature Importance Bar Chart:**

A bar chart of the top 10 most important features from the Random Forest (e.g., Vote_Count, Year, Domestic_US, Genre_Action, Genre_Drama, etc.).

**Shows:** which inputs drive the model's predictions most strongly—both numerical (like vote count or release year) and categorical indicators (e.g., "Genre=Action").




# Results & Output

- **Mean Squared Error (MSE)**: $1.886 \times 10^{16}$
- **$R^2$ Score**: 0.528
  - Explains ~52.8% of variance in worldwide gross.
- **Example Predictions**:
  - Input: 2005, Comedy, G, 230, en, India → Predicted Worldwide: $25,806,250.49
- **Interpretation**:
  - Moderate predictive power—additional features or model tuning could improve performance.
  - Large MSE due to high variance in box office figures (hundreds of millions).

## Limitations

- **Feature Granularity**: Genres and production countries are one-hot encoded without grouping rare categories.
- **Model Simplicity**: Does not account for marketing budgets, star power, franchise effects, or seasonal release windows.
- **Imbalanced Distribution**: Extreme outliers (blockbusters) can skew error metrics.

## Future Work

1. **Add Metadata**: Incorporate director, cast popularity, budget, release month.
2. **Hyperparameter Tuning**: Grid search over tree depth, number of estimators, min_samples_leaf.
3. **Alternative Models**: Gradient Boosting (XGBoost, LightGBM) or neural networks.
4. **Error Analysis**: Analyze large residuals to find systematic biases.
5. **Cross-Validation**: Use k-fold CV for more robust performance estimates.

## References


- **Scikit-Learn Documentation**: Pipeline, ColumnTransformer, RandomForestRegressor
- **IMDb Datasets**: [https://www.imdb.com/interfaces/](https://www.imdb.com/interfaces/)