Linear Regression Random Forest Logistic Regression

Film and TV Analysis

What contributes to a TV/Films success?

WATCH NEXT +

Project Overview

- FilmTV movie dataset ((link: https://www.kaggle.com/datasets/stefanoleone992/filmtv-movies-dataset))
- Dataset contains over 40k film and tv series from FilmTV.it website
- What variables/factors contribute to a TV/Film success?

TV/Film Dataset

- Title
- Year of release
- Genre
- Duration of tv/film
- Country of origin
- Directors
- Actors
- Votes
  - Average
  - Critic
  - Public
  - Total
- Description of tv/film
- Notes
- Humor
- Rhythm
- Effort
- Tension
- Erotism

Linear Regression Model

Initially with my dataset, I wanted to see if a linear regression model might be able to show a correlation between the genre of a film and the average score the audience rated the film. Naturally, I ran the machine learning model and received a little to no correlation between average vote and genre.

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Encode the genre column
le = LabelEncoder()
df['genre'] = le.fit_transform(df['genre'])

# Choose your features (predictors) and target variables
X = df[['genre']]
y = df['avg_vote']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Fit the linear regression model
reg = LinearRegression().fit(X_train, y_train)

# Make predictions on the test data
y_pred = reg.predict(X_test)

# Evaluate the model's performance
r2 = reg.score(X_test, y_test)
print("R2:", r2)
```

```
R2: 0.0004714900064652916
```

Random Forest

X values:

- year (The year the film was released)
- duration (the length of the film)
- country_grouped (the top 5 countries that pump out films each year)
- genre_grouped (the top 5 genres)
- has_top_actors (the top 100 actors)

Y values:

- popular here indicates where the average vote is greater than 7 out of 10

Model 1:

- My y value in the model is popular. After running the model I got an r-squared of 78%.

Model 2:

- Shows the features tension, rhythm, erotism, effort, and humor compared to the previous y value: popular. This model offered an r-squared of 81% which is a bit higher than our previous model.

Model 3:

- My final model essentially groups all of the previous variables as the X and compares them all to our typical y value: popular. This model offered the highest r-squared with an 82%.

Summary: The conclusion we have is that a film's rating, no matter the genre, depends upon various descriptive elements in order to receive a higher average score.

```
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier

X = df[['year', 'duration', 'country_grouped', 'genre_grouped', 'has_top_actor']]
y = df['popular']

# creates one-hot encoding for desired columns
X = pd.get_dummies(X, columns=['genre_grouped', 'country_grouped'])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

reg = RandomForestClassifier().fit(X_train, y_train)

r2 = reg.score(X_test, y_test)
print(r2)
```

```
0.7815407517677707
```

```
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier

X = df[['effort', 'humor', 'rhythm', 'tension', 'erotism']]#, 'year', 'duration', 'country_grouped', 'genre_grouped', 'has_top_ac
y = df['popular']

# creates one-hot encoding for desired columns
# X = pd.get_dummies(X, columns=['genre_grouped', 'country_grouped'])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

reg = RandomForestClassifier().fit(X_train, y_train)

r2 = reg.score(X_test, y_test)
print(r2)
```

```
0.8123061654881528
```

```
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier

X = df[['effort', 'humor', 'rhythm', 'tension', 'erotism', 'year', 'duration', 'country_grouped', 'genre_grouped', 'has_top_actor
y = df['popular']

# creates one-hot encoding for desired columns
X = pd.get_dummies(X, columns=['genre_grouped', 'country_grouped'])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

reg = RandomForestClassifier().fit(X_train, y_train)

r2 = reg.score(X_test, y_test)
print(r2)
```

```
0.8175164371666046
```

Logistic Regression: Random Oversampling

```
# Calculated the balanced accuracy score
from sklearn.metrics import balanced_accuracy_score
y_pred = model.predict(X_test)
balanced_accuracy_score(y_test, y_pred)
```

```
0.5682432204098257
```

Model 1: Duration

```
# Calculated the balanced accuracy score
from sklearn.metrics import balanced_accuracy_score
y_pred = model.predict(X_test)
balanced_accuracy_score(y_test, y_pred)
```

```
0.6584068341043577
```

Model 2: Tension

```
# Calculated the balanced accuracy score
from sklearn.metrics import balanced_accuracy_score
y_pred = model.predict(X_test)
balanced_accuracy_score(y_test, y_pred)
```

```
0.7012227006664427
```

Model 3: Humor, Rhythm, Effort, Tension, Erotism

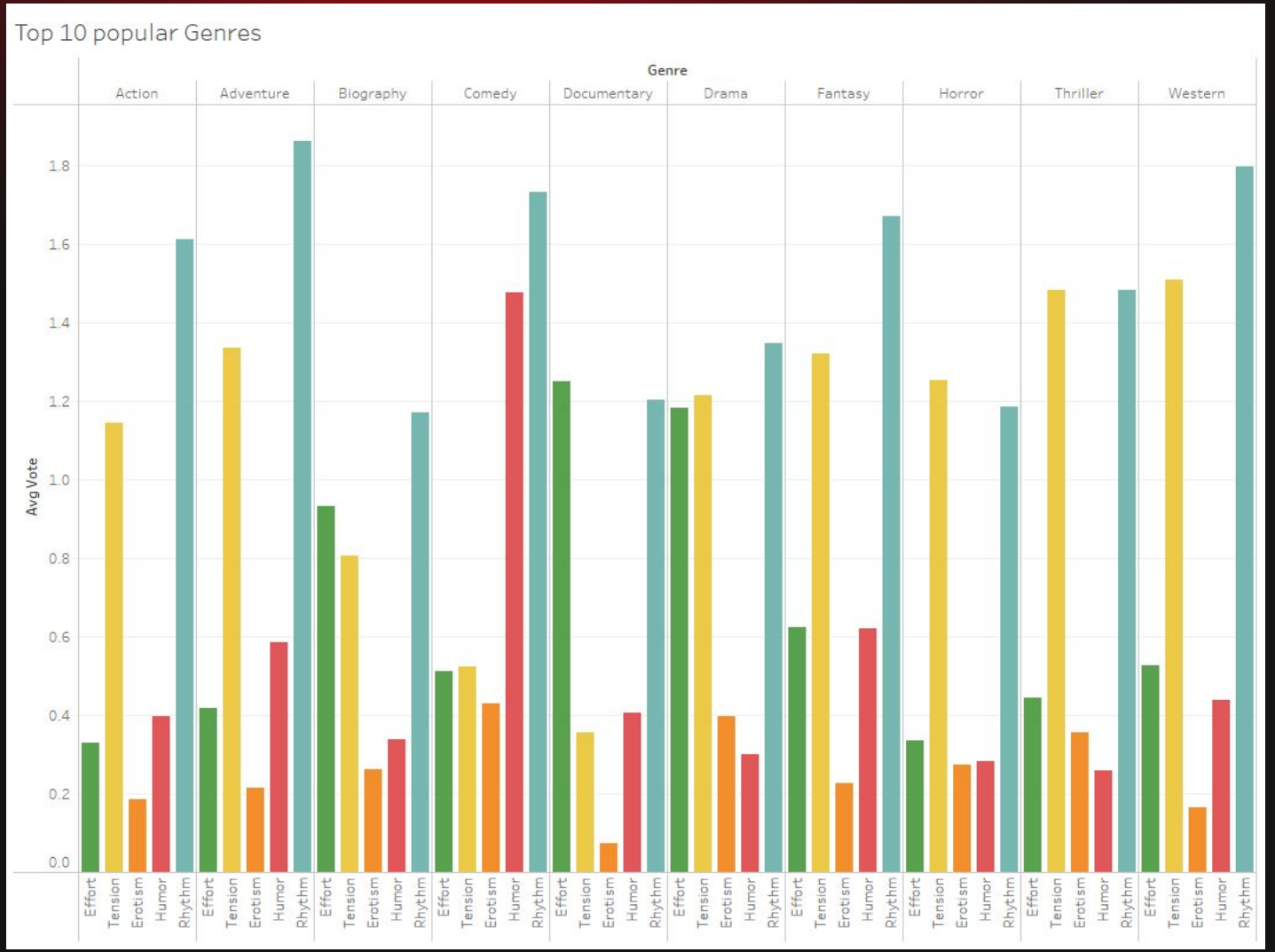
```
# Calculated the balanced accuracy score
from sklearn.metrics import balanced_accuracy_score
y_pred = model.predict(X_test)
balanced_accuracy_score(y_test, y_pred)
```

```
0.7337084407164924
```

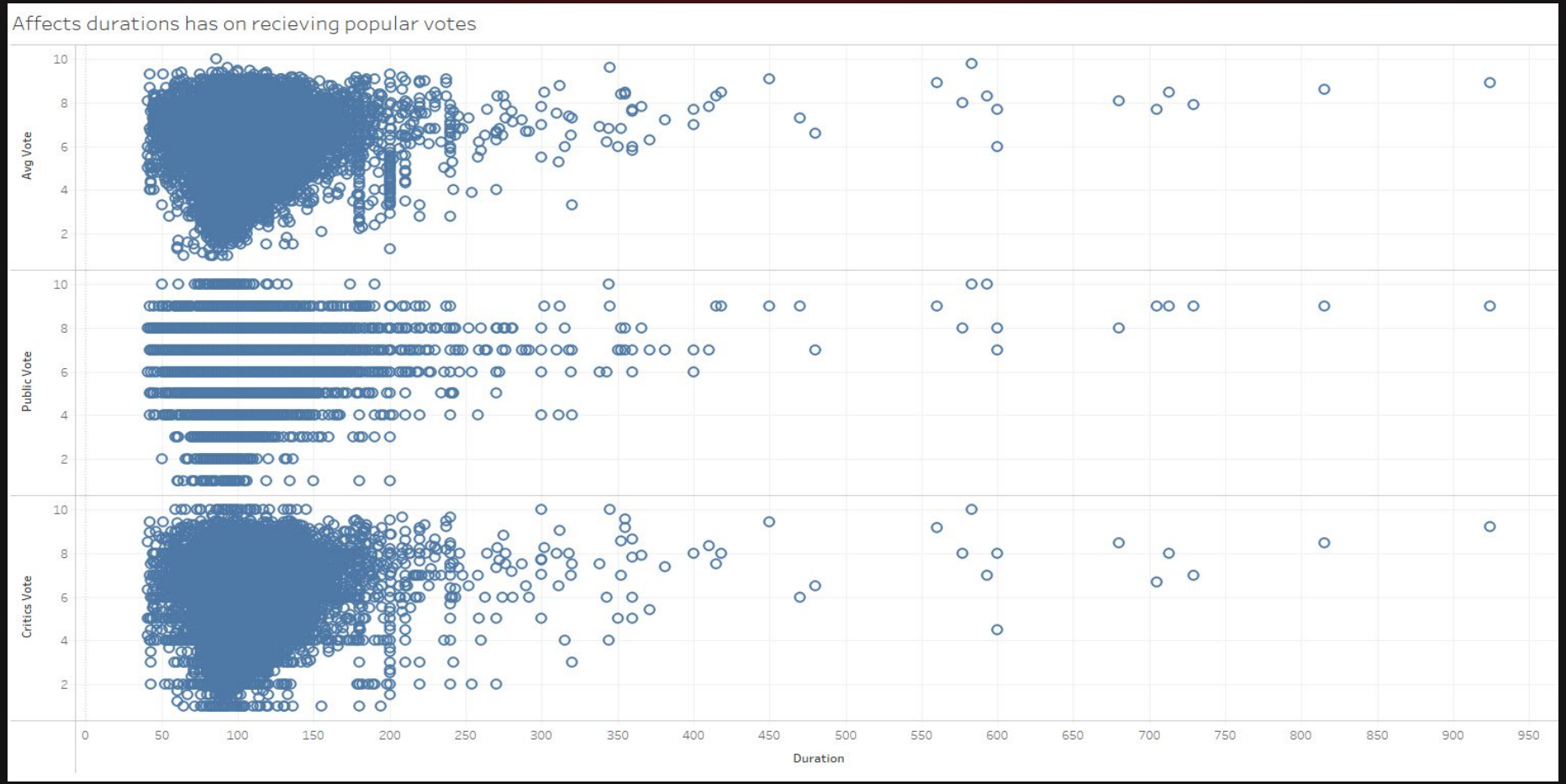
Model 4: Humor, Rhythm, Effort, Tension, Erotism, Duration, Year, Country Grouped, Genre Grouped, Having Top Actor

Graphs

- Break down of the sub category voting done in each genre



Relation that a movie's duration can have on voting



Tools

- Jupyter Notebook
- Python
- Tableau
- Github
- Kaggel

Recommendations/Improvements

- Creating a recommendation model
- Finding more data