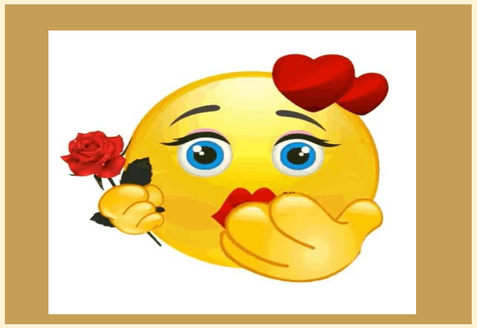
Group 4

✨Predicting Movie Success

Meet the team

Predicting Movies Success- Group #4

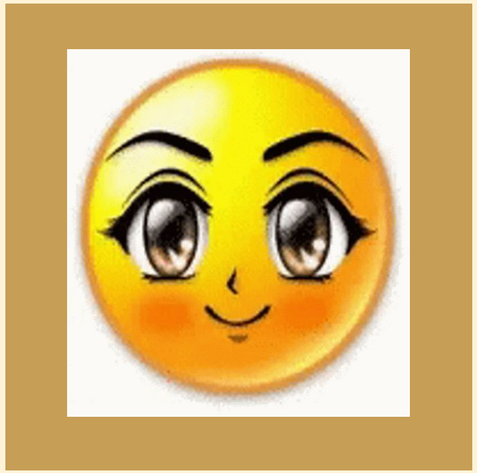
Name
Mai Houa Hang



Name
Susan Thao Vang

Name
Sam Carty

Name
Brendan Smith



Name
Mee Thao

Project Goals & Summary

- **Goal:** Use OMDB Movie Dataset to attempt to predict IMDB Ratings of movies.
- **Method:** Extract data from OMDB via API, transform the data for preprocessing, and run different Machine Learning models to see if meaningful prediction can occur given the existing data (75% Accuracy or better)
- **Definition of "Movie Success":** One standard deviation above the average IMDB rating, which is a rating of 7.4 out of 10. This puts a "good movie" by our definition in the top 15% of IMDB ratings.

Project Scope

Considerations

Movie Type Only

No TV shows

United States in ‘Country Field’

Movies released 2010 to 2025

Parameters

Release Date

Genre

Directors

IMDB Rating

Box Office

Content Rating

RunTime

Other

No “Fundamental” Data Missing

Adjusted Box Office for Inflation (2024)

ETL Process

- **API Extraction:**
  - Used IMDB TSV file to get IDs to query via OMDB API
  - Create loop to do API call for 129k records (5+ hours)
  - Originally tried normalizing data in API call loop- had to abandon due to increase in the time of each iteration.

- **Transformation:**
  - All data was in string format- enforced types.
  - Removed records missing fundamental data.
  - Used Python CPI library to adjust box office values for inflation.
  - Unfortunately, Rotten Tomatoes data and Box Office data was only available for about a third of records.

ETL Code Examples

Handmade “Get Dummies” loop for denormalized data.

```
final = main.copy().reset_index(drop=True)
final['Genre'] = final['Genre'].fillna('')
for row in tqdm(final.index):
    genres = final.loc[row,'Genre'].split(",")
    for g in genres:
        final.loc[row, g.strip().upper()] = 1
    time.sleep(.001)
```

```
100%|| 18966/18966
```

Looped through DataFrame to adjust BoxOffice via CPI library function.

```
for row in tqdm(final.index):
    boxoffice = final.loc[row,'BoxOffice']
    year = final.loc[row,'R_Year']
    if pd.isna(year) or pd.isna(boxoffice):
        continue
    if year < 2024:
        final.loc[row,'BoxOffice_Inf_Adj'] = cpi.inflate(boxoffice, int(year), to=2024)
        time.sleep(.001)
    else:
        continue
```

Used tqdm library to have loading screen for longer operations.

Data Set - CSV Output

Title	Rated	Released	Runtime	Genre	Director	Actors	Language	Country	Awards	Metascore	imdbRatir	imdbVote	imdbID	BoxOffice	Rotten_To	R_Year	R_Month	R_Day	Rounded I	Good_Mo
The Other	R	11/2/2018	122	Drama	Orson We	John Hust	English, G	France, Ir	9 wins & 8	79	6.7	8,188	tt0069049		82	2018	11	2	7	FALSE
Grizzly II: I	Not Rated	1/8/2021	74	Horror, Mu	AndrÃ© Sz	George Cle	English	United States		7	2.7	1,801	tt0093119		9	2021	1	8	3	FALSE
Mariette in	PG-13	########	101	Drama	John Baile	Geraldine	English	United States			7.4	72	tt0116991			2019	11	13	7	TRUE
Cooper an	Unrated	9/27/2013	120	Document	John Mulh	Sam Wate	English	United States		54	7.3	100	tt0159369			2013	9	27	7	FALSE
Heaven &	TV-MA	11/6/2018	104	Drama	Stuart Pau	Cheryl M. I	English	United States			3.1	151	tt0192528			2018	11	6	3	FALSE
Foodfight!	PG	2/12/2013	91	Animatior	Lawrence	Hilary Duf	English	United States, South Korea, Uni			1.3	12,157	tt0249516			2013	2	12	1	FALSE
Return to I	Not Rated	8/11/2013	75	Biography	Alex Mont	Jennifer Ti	English	United States			6	224	tt0255820			2013	8	11	6	FALSE
Bachelor '	R	5/22/2012	88	Comedy	Tony Vital	Darren Ge	English	United States			3.6	281	tt0285252			2012	5	22	4	FALSE
Dark Bloo	TV-PG	4/26/2014	86	Thriller	George Slu	River Phoe	English	United States, United Kingdom,			6.3	1,446	tt0293069		80	2014	4	26	6	FALSE
Mortal Koi	R	4/23/2021	110	Action, Ad	Simon Mc	Lewis Tan	English, Ja	United Sta	4 wins & 9	44	6	194,432	tt0293429	42326031	55	2021	4	23	6	FALSE
In My Slee	PG-13	4/23/2010	104	Drama, M	Allen Wolf	Philip Wir	English	United Sta	6 wins	33	5.5	2,150	tt0326965	30158	6	2010	4	23	6	FALSE
The Killer'	R	9/13/2024	104	Action, Co	J.J. Perry	Dave Baut	English	Spain, United States,		37	5.7	9,475	tt0327785	5404378	46	2024	9	13	6	FALSE
Bunyan ar	PG	1/12/2017	84	Animatior	Louis Ros	John Gooc	English	United States, India			4.8	459	tt0331314			2017	1	12	5	FALSE
On the Ro	R	5/23/2012	124	Adventure	Walter Sal	Sam Riley,	English, Fi	France, Ur	2 wins & 4	56	6	43,662	tt0337692	744296	46	2012	5	23	6	FALSE
The Evil W	Not Rated	8/30/2017	98	Horror	Andrew Gi	Sean Patri	English	United Sta	3 wins & 1	nominatic	5.6	4,090	tt0339736		100	2017	8	30	6	FALSE
The Secret	PG	########	114	Adventure	Ben Stiller	Ben Stiller	English, Sp	United Sta	5 wins & 1	54	7.3	351,568	tt0359950	58236838	52	2013	12	25	7	FALSE
Fahrenhei	TV-14	5/20/2018	100	Drama, Sc	Ramin Bal	Michael B	English	United Sta	Nominated for 5 Prin		5	23,762	tt0360556		31	2018	5	20	5	FALSE
Nappily Ev	TV-MA	9/21/2018	98	Comedy, D	Haifaa Al-	Sanaa Lat	English	United Sta	1 win & 5 r	63	6.4	9,996	tt0365545		71	2018	9	21	6	FALSE
A Walk An	R	9/19/2014	114	Action, Cri	Scott Fran	Liam Nees	English, Sp	United Sta	1 nominat	57	6.5	129,214	tt0365907	26307600	68	2014	9	19	6	FALSE
Jurassic V	PG-13	6/12/2015	124	Action, Ad	Colin Trev	Chris Prat	English	United Sta	15 wins &	59	6.9	693,345	tt0369610	6.53E+08	72	2015	6	12	7	FALSE
Spy	Not Rated	8/9/2011	110	Action, Cri	Alexander	Vincent Pa	English	United States			6.6	101	tt0372538			2011	8	9	7	FALSE
Remedy	R	3/29/2005	82	Crime, Dra	Christian I	Arthur J. N	English	United States			4.1	251	tt0375008			2005	3	29	4	FALSE
The Rum D	R	########	119	Comedy, D	Bruce Rob	Johnny De	English, Sp	United Kir	2 wins & 4	56	6.1	109,040	tt0376136	13109815	51	2011	10	28	6	FALSE
American	R	########	108	Crime, Dra	Ewan McG	Ewan McG	English, H	Hong Kon	1 win & 3 r	43	6.1	18,477	tt0376479	544098	24	2016	10	21	6	FALSE
Gnomeo &	G	2/11/2011	84	Animatior	Kelly Asbu	James Mc	English, Ja	United Kir	1 win & 12	53	5.9	60,898	tt0377981	99967670	55	2011	2	11	6	FALSE
The Three	PG	4/13/2012	92	Comedy, F	Bobby Far	Sean Haye	English	United Sta	4 nominat	56	5.1	33,620	tt0383010	44338224	51	2012	4	13	5	FALSE
Motherles	R	11/1/2019	144	Crime, Dra	Edward N	Edward N	English, Fi	United Sta	2 wins & 1	60	6.8	64,071	tt0385887	9277736	65	2019	11	1	7	FALSE
Anderson'	R	5/20/2010	98	Comedy, D	Jerome Els	Michael W	English	United States			5.8	163	tt0393049			2010	5	20	6	FALSE
Tangled	PG	########	100	Animatior	Nathan Gr	Mandy Mc	English, G	United Sta	Nominate	71	7.7	513,682	tt0398286	2.01E+08	89	2010	11	24	8	TRUE

Website for data set: https://www.omdbapi.com

Tableau Data Overview

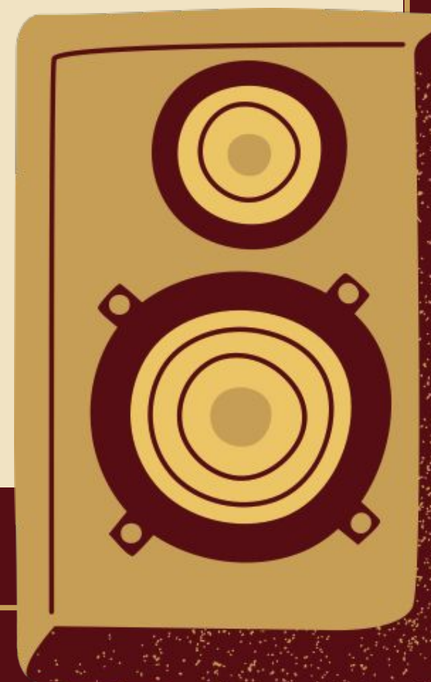
- We’ve prepared some visualizations to give a quick overview of the dataset.
- As a quick preview they cover metrics such as Director box office earnings, as well as Genres with the highest average ratings.
- Finally, there will be a quick showcase of some covid analytics.

Logistic Regression

Analysis

- **Parameters:**
  - Used IMDB TSV file to get IDs to query via OMDB API
  - Create loop to do API call for 129k records (5+ hours)
  - Originally tried normalizing data in API call loop- had to abandon due to increase in the time of each iteration.

- **Preprocessing Steps:**
  - All data was in string format- enforced types.
  - Removed records missing fundamental data.
  - Used Python CPI library to adjust box office values for inflation.
  - Unfortunately, Rotten Tomatoes data and Box Office data was only available for about a third of records.



Preprocessing- Code

Examples

Use logic to divide Inflation adjusted box office into quartiles.

```
#Creating a BoxOffice Category
q1, q2, q3 = start_data['BoxOffice_Inf_Adj'].quantile([0.25, 0.50, 0.75])


conditions = [
    start_data['BoxOffice_Inf_Adj'].isna(),
    start_data['BoxOffice_Inf_Adj'] < q1,
    (start_data['BoxOffice_Inf_Adj'] >= q1) & (start_data['BoxOffice_Inf_Adj'] < q2),
    (start_data['BoxOffice_Inf_Adj'] >= q2) & (start_data['BoxOffice_Inf_Adj'] < q3),
    start_data['BoxOffice_Inf_Adj'] >= q3
]

choices = [
    'Unrated',
    'Quart_1',
    'Quart_2',
    'Quart_3',
    'Quart_4'
]

start_data['BoxOfficeCategory'] = np.select(conditions, choices)
start_data.groupby('BoxOfficeCategory').count()
```

Used get dummies for large dimension - “Director” Column which was over 12k fields.

```
model_data = pd.get_dummies(model_data, columns=['Rated','R_Year','BoxOfficeCategory','R_Month','R_Day','Director'])
model_data
```

	imdbVotes	Runtime	Good_Movie	DRAMA	HORROR	MUSIC	THRILLER	DOCUMENTARY	ANIMATION	ACTION	...	Director
0	8188	122	False	1	0	0	0	0	0	0	...	
1	1801	74	False	0	1	1	1	0	0	0	...	
2	72	101	True	1	0	0	0	0	0	0	...	
3	100	120	False	0	0	0	0	1	0	0	...	
4	151	104	False	1	0	0	0	0	0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
17973	9831	92	False	0	1	0	0	0	0	1	...	
17974	893	101	False	0	0	0	0	0	0	0	...	
17975	921	108	False	0	0	0	0	0	1	0	...	
17976	14780	88	False	0	0	0	0	0	0	1	...	
17977	19650	111	False	0	1	0	0	0	0	0	...	

17976 rows × 12458 columns

Logistic Regression- Results

Overall Accuracy
87.6%

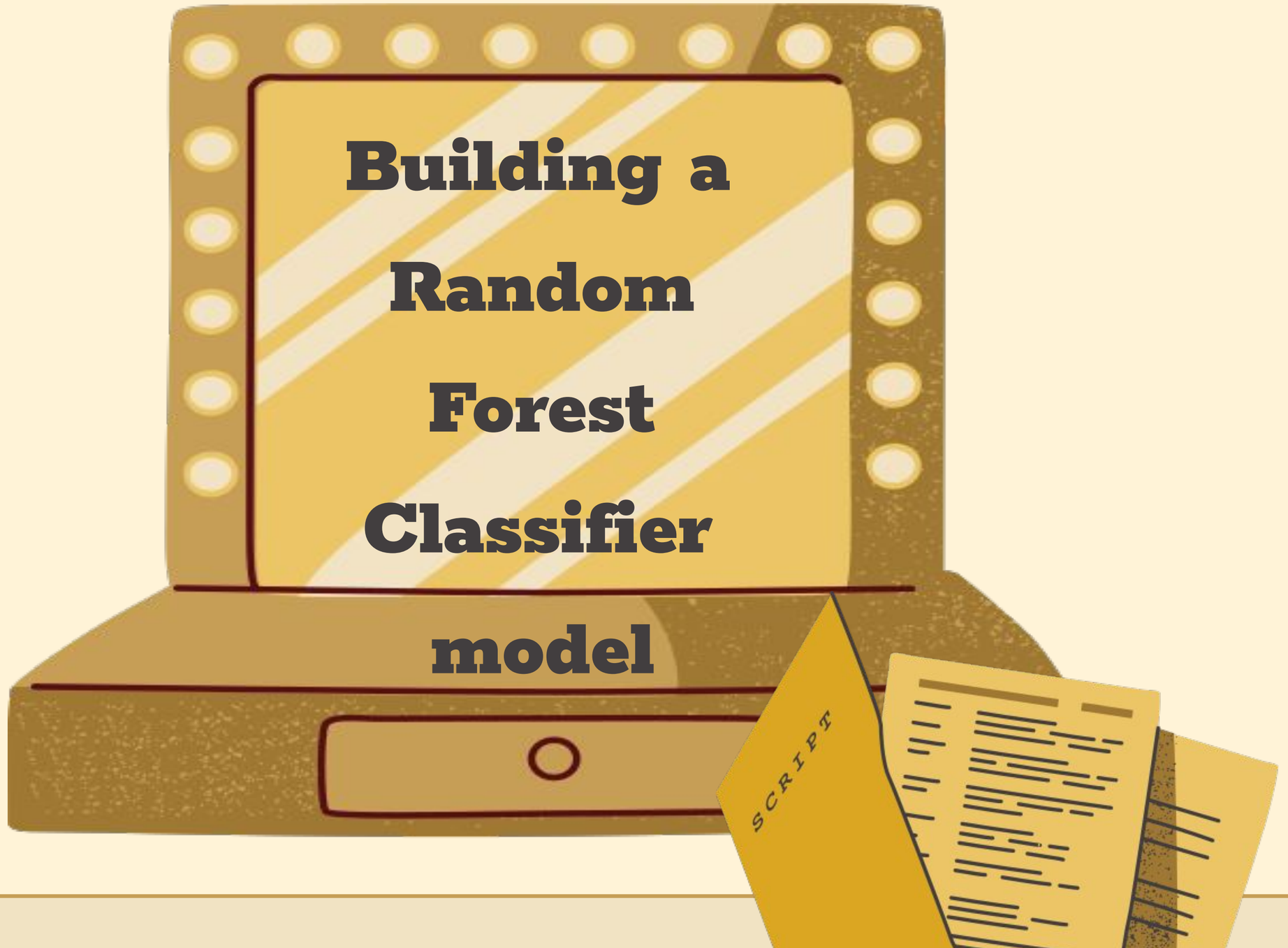
Precision
60%

Recall
23%

- **Results Summary:**
  - Our model can predict if an IMDB Rating is 7.4 or higher with 87% accuracy.
  - Accuracy is slightly misleading because there are far more movies rated under 7.4 in the training and test data than movies over that margin.
  - 2000 iterations used in training.

True Negatives	False Positives
3802	92
False Negatives	**True Positives**
464	136

Building a Random Forest Classifier model



Coding For the Random

[18]:
```
# Check for missing values in each column
missing_values = df_updated.isnull().sum()

# Display the columns with missing values
missing_values[missing_values > 0]
```

[18]:
```
Metascore            11460
imdbRating             727
imdbVotes              173
BoxOffice            13210
Rotten_Tomatoes      10030
Rounded Rating         727
BoxOffice_Inf_Adj    13453
dtype: int64
```

[19]:
```
# Fill missing values in the 'Rotten_Tomatoes' column with the mean
df_updated['Rotten_Tomatoes'] = df_updated['Rotten_Tomatoes'].fillna(df_updated['Rotten_Tomatoes'].mean())
```

[21]:
```
# Fill missing values in all other columns with 0
df_updated = df_updated.fillna(0)
```

[22]:
```
# Check again for missing values
missing_values_after = df_updated.isnull().sum()
missing_values_after[missing_values_after > 0]
```

[22]:
```
Series([], dtype: int64)
```

[23]:
```
# Define success based on Rotten Tomatoes score
threshold = 70
df_updated['Success'] = (df_updated['Rotten_Tomatoes'] >= threshold).astype(int)
```

[24]:
```
# Count the total number of movies using the 'Title' column
total_movies_count = df['Title'].count()
total_movies_count
```

[24]:
```
17978
```

[25]:
```
# Count the total number of movies with a Rotten Tomatoes score of 70 or more
successful_movies_count = df[df['Rotten_Tomatoes'] >= 70]['Rotten_Tomatoes'].count()
successful_movies_count
```

[25]:
```
3796
```

[26]:
```
df_updated = df_updated.drop(columns=['Rotten_Tomatoes'])
df_updated.columns
```

[32]:
```
# Save dataframe to a csv file
test_data.to_csv('test_data.csv', index=False)
```

[33]:
```
# Define features and target
X_train = train_data.drop(columns=['Success', 'Title'])
y_train = train_data['Success']
X_test = test_data.drop(columns=['Success', 'Title'])
y_test = test_data['Success']
```

[34]:
```
# Standardize numeric features
numeric_features = [
    'Runtime', 'Metascore', 'imdbRating', 'imdbVotes', 'BoxOffice',
    'Rounded Rating', 'Good_Movie', 'BoxOffice_Inf_Adj', 'Year'
]
scaler = StandardScaler()
X_train[numeric_features] = scaler.fit_transform(X_train[numeric_features])
X_test[numeric_features] = scaler.transform(X_test[numeric_features])
```

[35]:
```
# Define Random Forest with constraints to prevent overfitting
model = RandomForestClassifier(
    n_estimators=100,
    max_depth=15,
    min_samples_split=5,
    min_samples_leaf=2,
    class_weight='balanced',
    random_state=42
)
```

[36]:
```
# Cross-validation
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cross_val_scores = cross_val_score(model, X_train, y_train, cv=cv, scoring='accuracy')
print(f"Cross-validation Accuracy: {cross_val_scores.mean():.4f} ± {cross_val_scores.std():.4f}")
```

```
Cross-validation Accuracy: 0.9067 ± 0.0049
```

[37]:
```
# Train model
model.fit(X_train, y_train)
```

Results For the Random Forest

Cross-validation Accuracy

Accuracy:

90.67%

Std. Dev: ±

0.0049

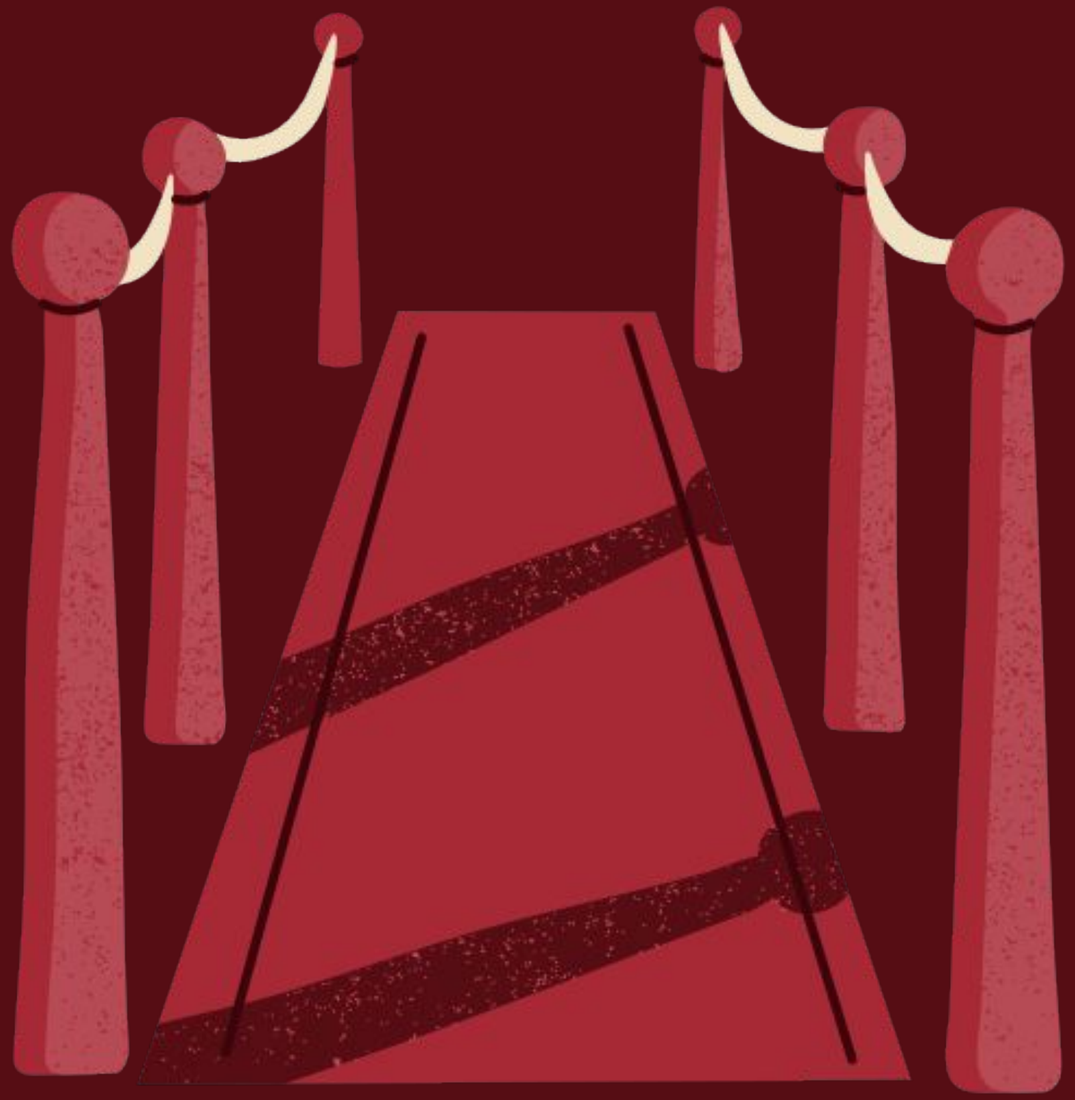
Classification Report

Test Accuracy: 82.26%

	precision	recall	f1-score	support
0	0.91	0.80	0.85	40
1	0.70	0.86	0.78	22
accuracy			0.82	62
macro avg	0.81	0.83	0.81	62
weighted avg	0.84	0.82	0.83	62

Feature Importances

	Feature	Importance
1	Metascore	0.338733
2	imdbRating	0.136921
3	imdbVotes	0.131218
5	Rounded Rating	0.092306
4	BoxOffice	0.064690
7	BoxOffice_Inf_Adj	0.053241
8	Year	0.043925
0	Runtime	0.041011
15	Documentary	0.022029
16	Drama	0.007454



Creating A Decision Tree

```
# Initial imports
import pandas as pd
from pathlib import Path
from sklearn import tree
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
```

```
# Creating StandardScaler instance
scaler = StandardScaler()
```

```
# Fitting Standard Scaller
X_scaler = scaler.fit(X_train)
```

```
# Scaling data
X_train_scaled = X_scaler.transform(X_train)
X_test_scaled = X_scaler.transform(X_test)
```

```
# Creating the decision tree classifier instance
model = tree.DecisionTreeClassifier()
```

```
# Fitting the model
model = model.fit(X_train_scaled, y_train)
```

```
# Making predictions using the testing data
predictions = model.predict(X_test_scaled)
```

```
# Calculating the confusion matrix
cm = confusion_matrix(y_test, predictions)
cm_df = pd.DataFrame(
    cm, index=["Actual 0", "Actual 1"], columns=["Predicted 0", "Predicted 1"]
)

# Calculating the accuracy score
acc_score = accuracy_score(y_test, predictions)
```

Decision Tree

Overall Accuracy

85%

Precision

43%

Recall

37%

- **Results Summary:**
  - The decision tree model performs well on predicting False with a high precision of 90% and a recall of 91%.
  - Performance for True isn't doing as well, having a low precision 43% and recall of 37% .
  - Overall accuracy is 85% but the macro and weighted average suggest that the model's performance is unbalanced favoring class False.

Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	3600	294
Actual 1	378	222

Accuracy Score : 0.8504672897196262

Classification Report

	precision	recall	f1-score	support
False	0.90	0.92	0.91	3894
True	0.43	0.37	0.40	600
accuracy			0.85	4494
macro avg	0.67	0.65	0.66	4494
weighted avg	0.84	0.85	0.85	4494

Creating A KNN

K-Nearest Neighbors

KNN DATA

```
# Create the StandardScaler instance
scaler = StandardScaler()
# Fit the Standard Scaler with the training data
X_scaler = scaler.fit(X_train)
# Scale the training data
X_train_scaled = X_scaler.transform(X_train)
X_test_scaled = X_scaler.transform(X_test)
```

```
# Instantiate the model with k = 6 neighbors
model = KNeighborsClassifier(n_neighbors=6)
```

```
# Train the model
model.fit(X_train_scaled, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=6)
```

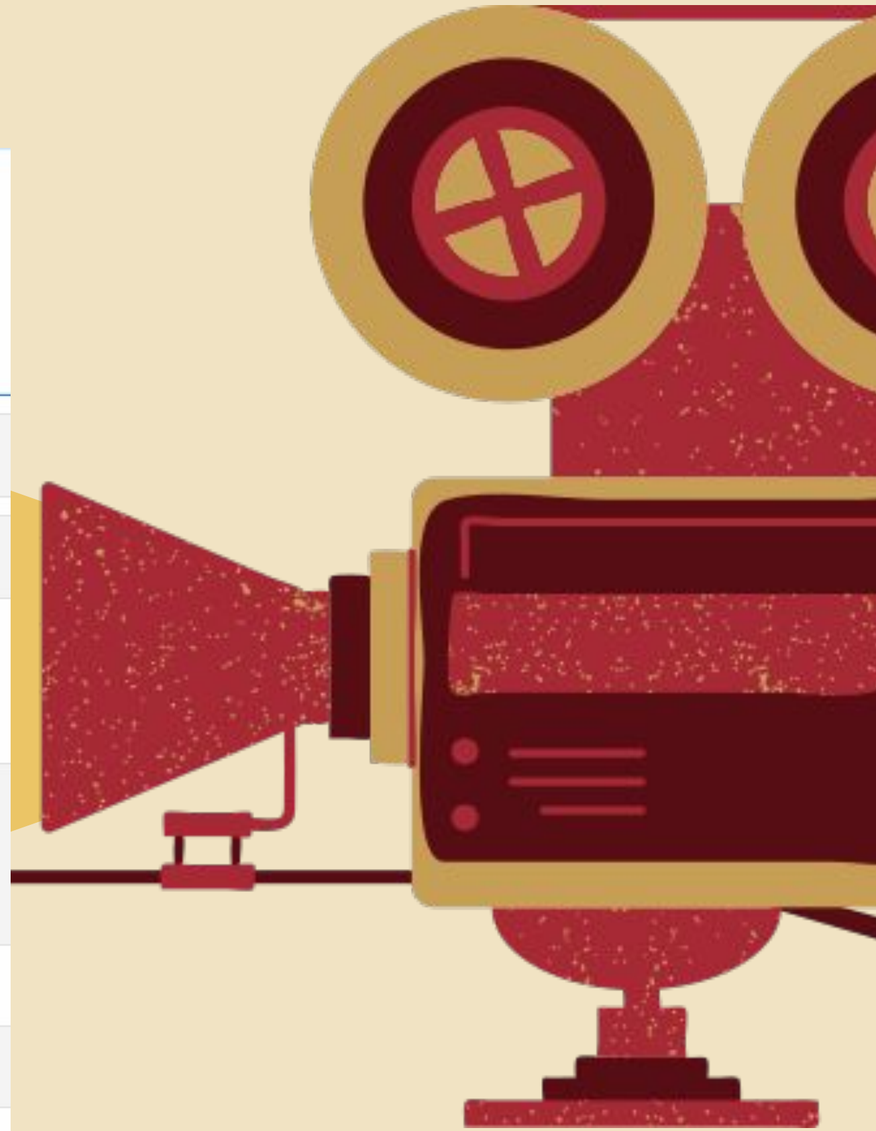
```
# Create predictions
y_pred = model.predict(X_test_scaled)

# Review the predictions
y_pred
```

```
array([False,  True,  True, ..., False, False,  True])
```

```
# Print confusion matrix
confusion_matrix(y_pred,y_test)
```

```
array([[3188,  430],
       [ 690,  186]], dtype=int64)
```

```
# Print classification report
print(classification_report(y_pred,y_test))
```



KNN

OVERALL ACCURACY

75%

PRECISION

30%

RECALL

21%

	precision	recall	f1-score	support
False	0.82	0.88	0.85	3618
True	0.30	0.21	0.25	876
accuracy			0.75	4494
macro avg	0.56	0.55	0.55	4494
weighted avg	0.72	0.75	0.73	4494

Model Results

K- Nearest Neighbors

Overall accuracy is 75%

Random Forest

The Overall accuracy: 82.26%

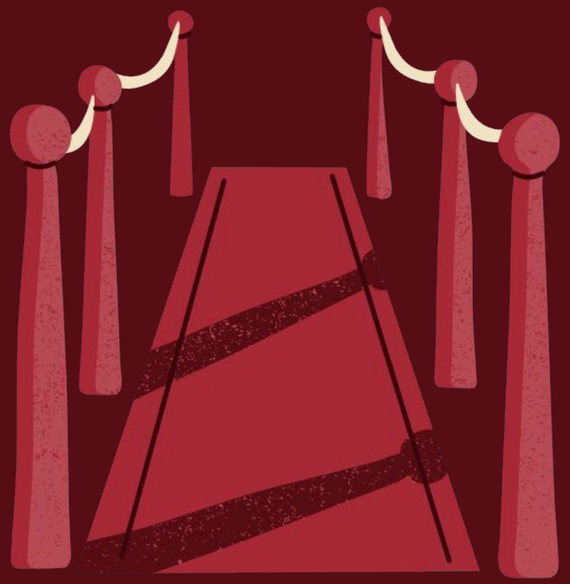
Decision Tree

Overall accuracy is 85%

Logistic Regression

Overall accuracy is 87.6%

Project Results Summary

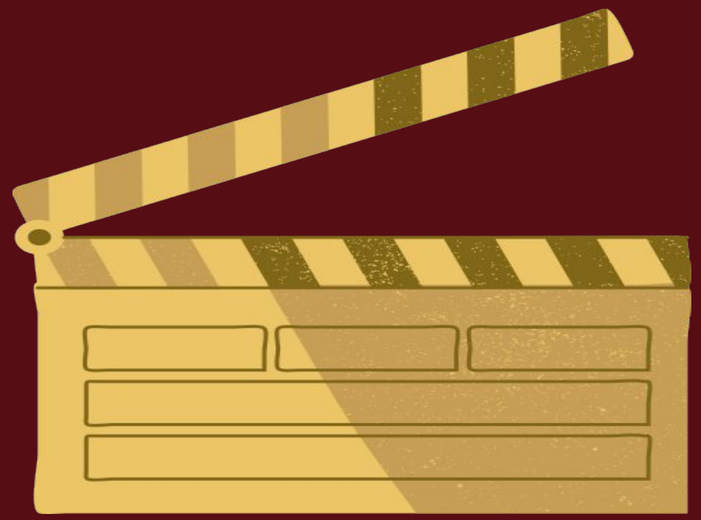


- **Model Success**
  - While the model met the 75% accuracy metric we were hoping for, results were skewed by the ratio of Good movies to movies below 7.4 IMDB ratings.
- **Data Quality**
  - Our models would have been more successful if the data correlated more closely with some of the major factors that go into IMDB Rating.
- **If We had more time...**
  - Parse Award Field into Oscars, Nominations, etc.
  - Incorporate Actor Data
  - Join other data sets

Resource Page

OMDb API. *OMDb API*. OMDb API, www.omdbapi.com/. Accessed 17 Mar. 2025. Accessed 18 Mar. 2025. Accessed 20 Mar. 2025. Accessed 22 Mar. 2025

Questions?




Thank you!