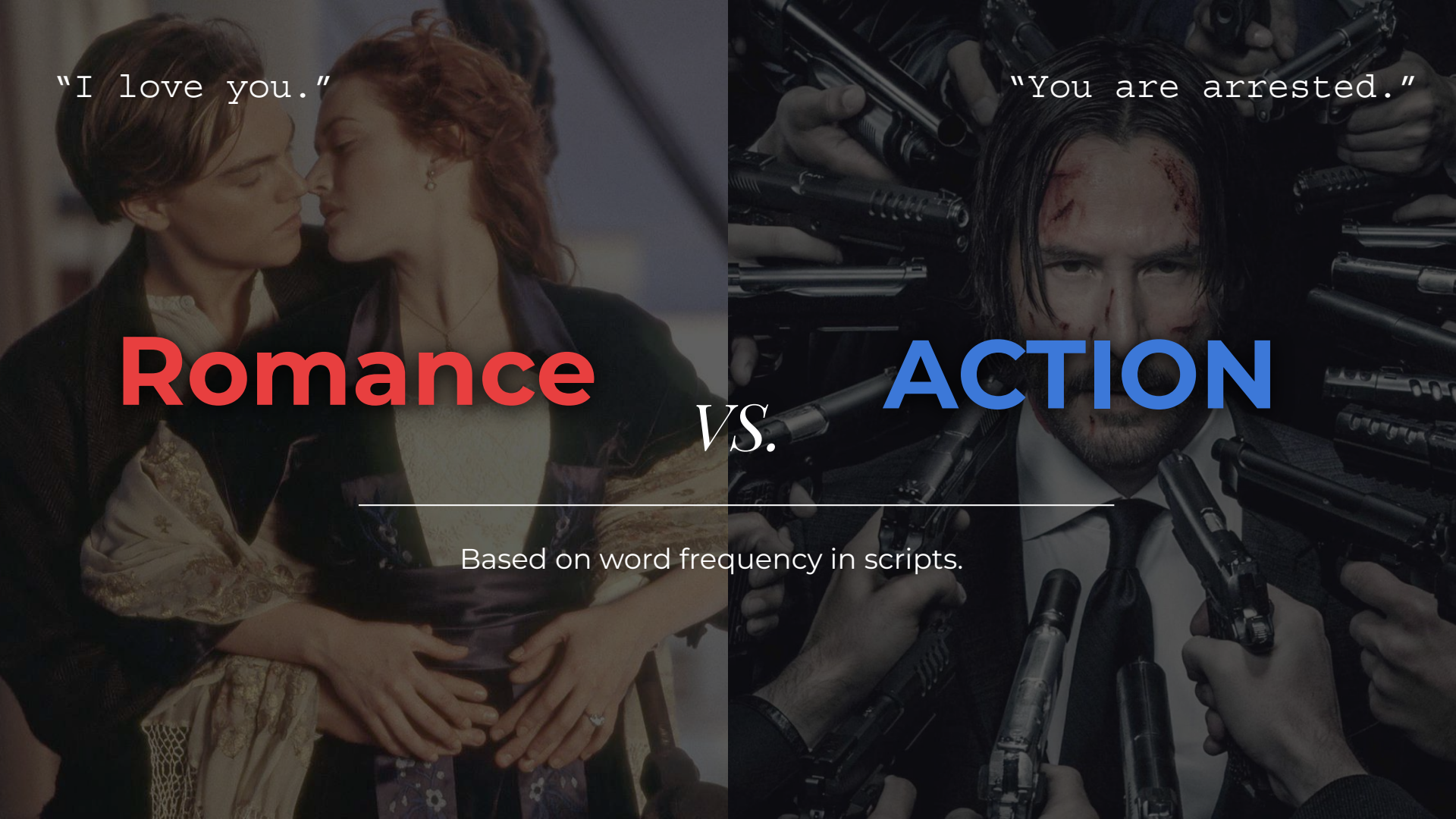
MOVIE CLASSIFIER

Shivali Chainani


"I love you."
"You are arrested."
Romance
VS.
ACTION
Based on word frequency in scripts.

Data Exploration and Feature Selection

- Import Packages
  - pandas, seaborn, numpy, matplotlib
- Load each dataset
- Data Overview
  - Use `.info()` for data frame summaries
  - Display first few rows with `.head()`

	Title	Genre	Year	Rating	# Votes	# Words	i	the	to	a	...	foster	pub
0	the terminator	action	1984	8.1	183538	1849	0.040022	0.043807	0.025419	0.024878	...	0.0	0.0
1	batman	action	1989	7.6	112731	2836	0.051481	0.033850	0.023977	0.028209	...	0.0	0.0
2	tomorrow never dies	action	1997	6.4	47198	4215	0.028707	0.054330	0.030368	0.021827	...	0.0	0.0
3	batman forever	action	1995	5.4	77223	3032	0.036609	0.042216	0.020449	0.031003	...	0.0	0.0
4	supergirl	action	1984	4.1	6576	3842	0.041905	0.032275	0.028891	0.026288	...	0.0	0.0

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 242 entries, 0 to 241
Columns: 5006 entries, Title to fling
dtypes: float64(4952), int64(52), object(2)
memory usage: 9.2+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40053 entries, 0 to 40052
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Stem    40053 non-null  object
 1   Word    40053 non-null  object
dtypes: object(2)
memory usage: 626.0+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4999 entries, 0 to 4998
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   i       4999 non-null   object
dtypes: object(1)
memory usage: 39.2+ KB
```

Continued.

- Data Preparation
  - Remove non-essential columns (Title, Genre, etc.)
  - Retain numeric word frequency data
- Filter Common Words
  - Exclude ~130 non-informative words (prepositions, appositives)
  - Create and use a filter list from CSV
- Genre Segregation
  - Separate data into action and romance genres
- Frequency Analysis
  - Calculate and compare mean word frequencies by genre

```
# Filter out non-numeric columns (like Title and Genre)
numeric_data = movies_df.drop(columns=['Title', 'Genre', 'Year', 'Rating', '# Votes', '# Words'])
```

```
# Load the CSV file that includes all unnecessary prepositions and appositives
unneeded_words_df = pd.read_csv('list_prep_appositives.csv')
unneeded_words = unneeded_words_df.squeeze().tolist()

# Filter out columns in numeric_data that are in the unneeded_words list
numeric_data = numeric_data.drop(columns=[word for word in unneeded_words if word in numeric_data.
```

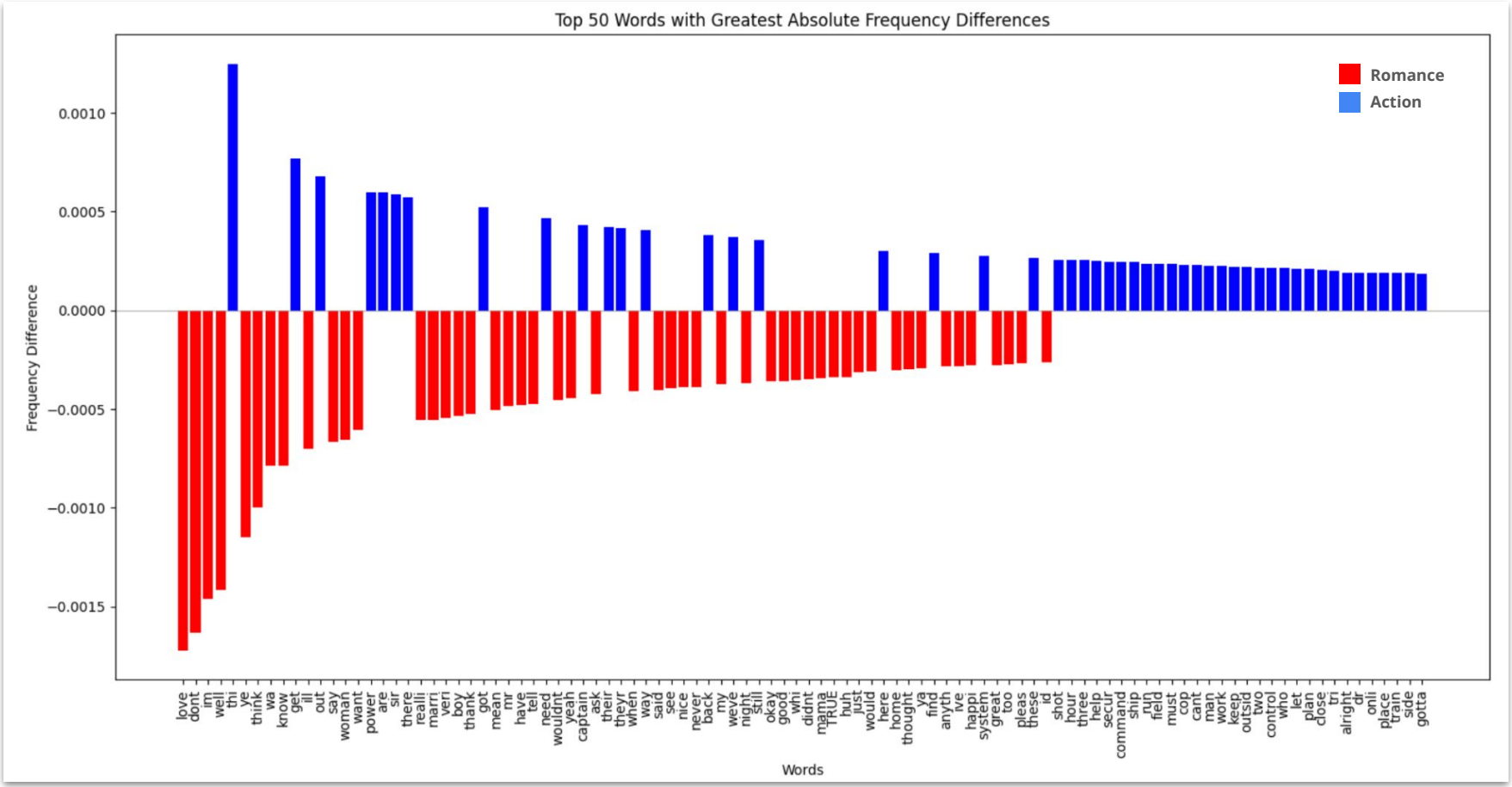
```
# Separate the dataset into action and romance movies
action_movies = numeric_data[movies_df['Genre'] == 'action']
romance_movies = numeric_data[movies_df['Genre'] == 'romance']
```

```
# Compute the mean frequency of each word for both genres
action_mean = action_movies.mean()
romance_mean = romance_movies.mean()
```

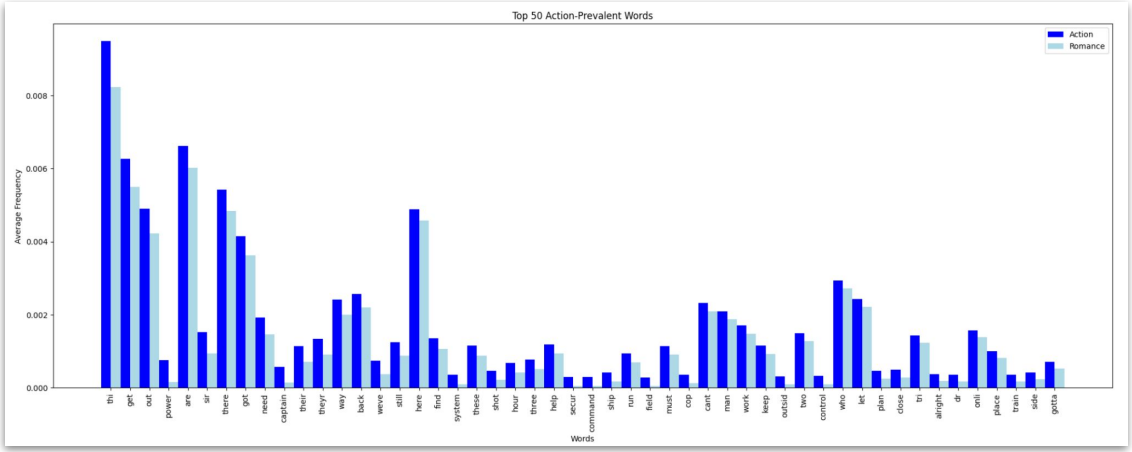
```
# Calculate the difference in means between the two genres
mean_diff = action_mean - romance_mean

positive_diffs = mean_diff[mean_diff > 0].sort_values(ascending=False)[:50]
negative_diffs = mean_diff[mean_diff < 0].sort_values()[:50]

combined_top_50_words = pd.concat([positive_diffs, negative_diffs]).sort_values(ascending=False)
combined_top_50_indices = combined_top_50_words.index
```

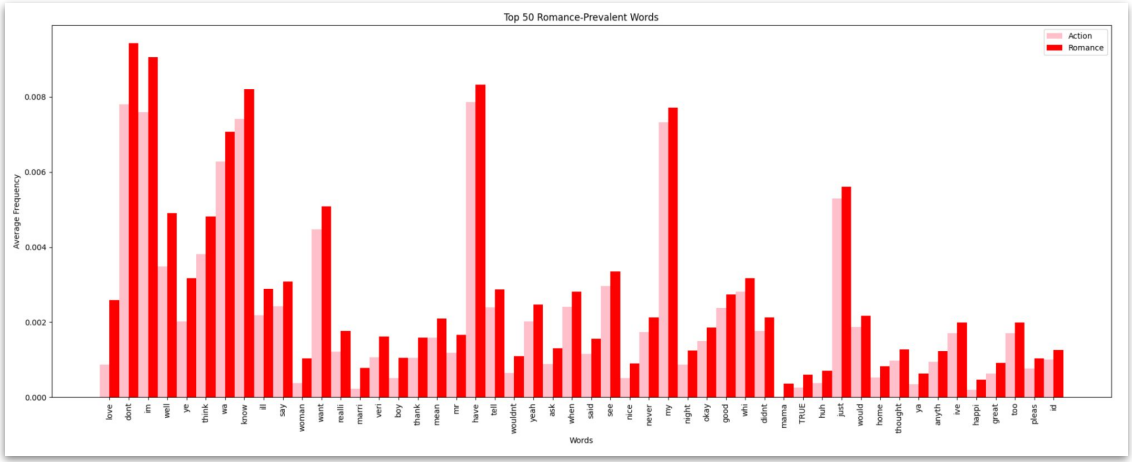
```
# For visualization, you can combine these two series and sort them for a comprehensive view
combined_top_50_words_abs_sorted = combined_top_50_words.abs().sort_values(ascending=False)
combined_top_50_indices_abs_sorted = combined_top_50_words_abs_sorted.index
```

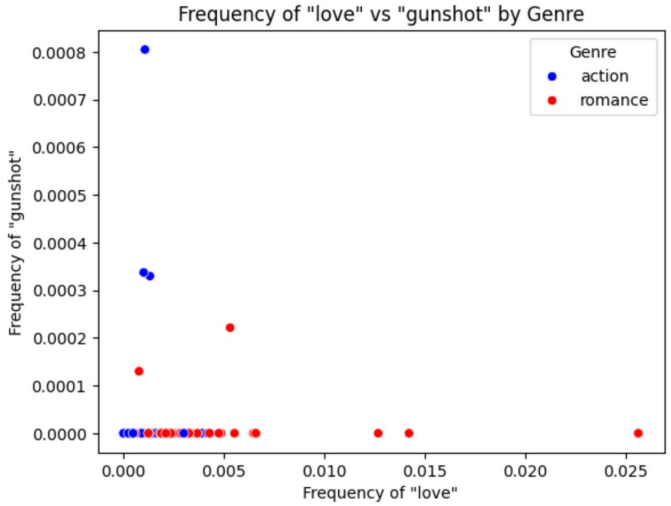
Bar Graph:

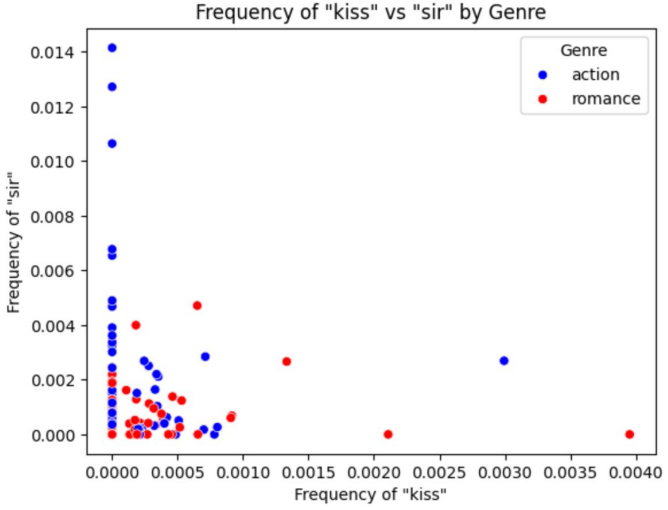

Top 50 Words with Greatest Absolute Frequency Differences
Romance
Action
Frequency Difference
0.0010
0.0005
0.0000
−0.0005
−0.0010
−0.0015
Words
love dont im well thi ye think wa know get ill out say woman want power are sir there reall marri veri boy thank got mean mr have tell need wouldnt yeah captain ask their theyr when way said see nice never back my were night still okay good whi didnt mama TRUE huh just would here home thought ya find anyth ive happi system great too pleas these id shot hour three help secur command ship run field must cop cant man work keep outsid two control who let plan close tri alright dr onli place train side gotta

Frequency Charts + Dot Plots:









Development of Classifier

- Top Words Extraction
  - Top 50 words per genre identified
- Model Selection
  - Decision Tree Classifier
  - Imported: `DecisionTreeClassifier`, splitting data, and evaluation tools

```
features = np.concatenate((chosen_action_words, chosen_romance_words))
feature_data = movies_df[features]

#4.2
#Model Selection: Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score

# Initialize the classifier
classifier = DecisionTreeClassifier()

#4.3
#Training: Split your data into training and testing sets. Use the training set to train your classifier.
# Prepare the target variable
y = movies_df['Genre'].apply(lambda x: 1 if x == 'action' else 0)  # 1 for action, 0 for romance
# Split the data
X_train, X_test, y_train, y_test = train_test_split(feature_data, y, test_size=0.2, random_state=42)
# Train the classifier
classifier.fit(X_train, y_train)
```

```
#3 Feature Selection
# Based on analysis, select top 50 words that are good indicators of a movie's genre
# Good indicators = words with significantly different frequencies in each genre
# *make sure not too correlated with each other to avoid multicollinearity

# Selecting the top 50 words for both categories and converting them to arrays
chosen_action_words = positive_diffs.index.to_numpy()
chosen_romance_words = negative_diffs.index.to_numpy()

chosen_action_words, chosen_romance_words
```

```
(array(['the', 'we', 'of', 'us', 'thi', 'is', 'our', 'them', 'they', 'on',
        'were', 'get', 'out', 'power', 'are', 'sir', 'there', 'got', 'one',
        'need', 'from', 'captain', 'their', 'theyr', 'way', 'where', 'up',
        'back', 'weve', 'still', 'as', 'here', 'find', 'him', 'some', 'he',
        'system', 'these', 'shot', 'hour', 'three', 'help', 'secur',
        'command', 'into', 'ship', 'in', 'off', 'run', 'field'],
       dtype=object),
 array(['i', 'she', 'that', 'love', 'oh', 'dont', 'no', 'im', 'well', 'me',
        'it', 'a', 'ye', 'and', 'her', 'like', 'think', 'wa', 'know',
        'ill', 'do', 'say', 'woman', 'did', 'want', 'with', 'realli',
        'marri', 'veri', 'boy', 'thank', 'mean', 'mr', 'have', 'tell',
        'at', 'but', 'wouldnt', 'yeah', 'about', 'ask', 'when', 'said',
        'see', 'nice', 'never', 'my', 'night', 'okay', 'good'],
       dtype=object))
```

- Data Preparation and Training
  - Genres encoded: action (1), romance (0)
  - Data split: 80% training, 20% testing
  - Model trained with `fit` method
- Model Implementation
  - Classifier predicts genres from word frequencies
  - Testing ensures unbiased evaluation

Assessment of the Classifier

Import accuracy and precision built-in calculators

- Accuracy_score from sklearn.metrics
- Precision_score from sklearn.metrics

```
#4.2
#Model Selection: Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
```

Compute accuracy and precision based on the testing data

- Accuracy range at ~75%
- Precision range at ~72%

```
# Calculate the accuracy
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy: ', accuracy)

# Calculate the precision
precision = precision_score(y_test, y_pred)
print('Precision:', precision)
```

```
Accuracy:  0.7551020408163265
Precision: 0.7272727272727273
```

Comparison to knn classifier

```
#kNN comparison
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
```

Import KNN classifier from sklearn.neighbors

- Reuse accuracy_score and precision_score
  - This will later be used to evaluate the knn in comparison to our decision tree

Compute accuracy and precision

- This showed to range very similarly to our own classifier
- Relatively low for knn but average in terms of over or underfitting
  - Could be due to word choice in the data set themselves

Define the n_neighbors value as 50

Fit our data sets to the model

```
# Instantiate the k-NN classifier with k=3
knn = KNeighborsClassifier(n_neighbors=50)

# Train the classifier on the training data
knn.fit(X_train, y_train)

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

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Calculate precision
precision = precision_score(y_test, y_pred)
print('Precision:', precision)
```

```
Accuracy: 0.7142857142857143
Precision: 0.6333333333333333
```

Human Context

- Enhance user experience and marketing effectiveness for streaming platforms

- Boost economic benefits by improving recommendation algorithms, which enhance subscriber satisfaction and retention, leading to increased user engagement and higher subscription rates.

- Provide valuable insights during film production and distribution, helping producers forecast market appeal

Thank you for listening!