they all do very bad things. It should be called 'Badfellas'

By Crispin Smith on 26 November 2015

There is nothing 'Good' about the 'fellas' in this movie.

Cost me my marriage

16 December 2017

Format: Prime Video

Terrifying, the very thought of toys coming alive when you aren't watching gave me chills. Subsequently I burned all my kids toys which apparently makes me a "bad father" and my wife has left me and taken the kids away from me. Terrible movie, ruined my life

Movie Review Analysis

Five Stars

By colton - March 19, 2015

Amazon Verified Purchase

I never knew what struggles bugs went through.... changed my life

Natural Language Processing, NER, and Genre Classification

Great movie, or GREATEST MOVIE EVER!!!

By J. janousek - March 1, 2007

I'm confused, does 1 star mean good or not??

1 of 35 people found this review helpful

Word Sentiment Analysis

Dataset and Question

- **Movie Reviews Dataset -** a dataset containing 50,000 movie reviews (from which we extracted 20,000 reviews) and corresponding positive or negative sentiment classification

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

- **Question:** Which words are most influential in determining whether a review is positive or negative?

Preprocessing Steps

1) Remove all characters except A-Z and a-z from the reviews
2) **Tokenize** text so that each review is a list of words
3) Remove all **stopwords** (a, I, the, etc.) from the tokenized reviews
4) **Lemmatize** all words (“run,” “running,” “ran” —> “run”)
5) Compile a count of the number of instances of each word in each review

	raw_text	modified_text	word_count	aaron	abandon	abandoned	abc	abilities	ability	able	...	youth	youthful	youtube	youve	zany	zero	zombie	zombies	zone	sentiment
0	One of the other reviewers has mentioned that ...	[One, reviewer, mention, watch, Oz, episode, y...	320	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	positive
1	A wonderful little production The filming tech...	[A, wonderful, little, production, The, film, ...	166	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	positive
2	I thought this was a wonderful way to spend ti...	[I, think, wonderful, way, spend, time, hot, s...	172	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	positive
3	Basically theres a family where a little boy J...	[Basically, there, family, little, boy, Jake, ...	141	0	0	0	0	0	0	0	...	0	0	0	0	0	0	1	0	0	negative
4	Petter Matteis Love in the Time of Money is a ...	[Petter, Matteis, Love, Time, Money, visually,...	236	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	positive

Finding Word Weights

- Logistic Regression with 86.7% test accuracy
- We extract the coefficients the model assigned to each word:

10 Most **Positive** Words:

	word	weight
2214	excellent	0.781233
3823	loved	0.725900
2378	favorite	0.614576
751	brilliant	0.608669
7105	wonderful	0.572822
2351	fantastic	0.552297
218	amazing	0.534505
4623	perfect	0.496655
577	best	0.490964
5110	realistic	0.487599

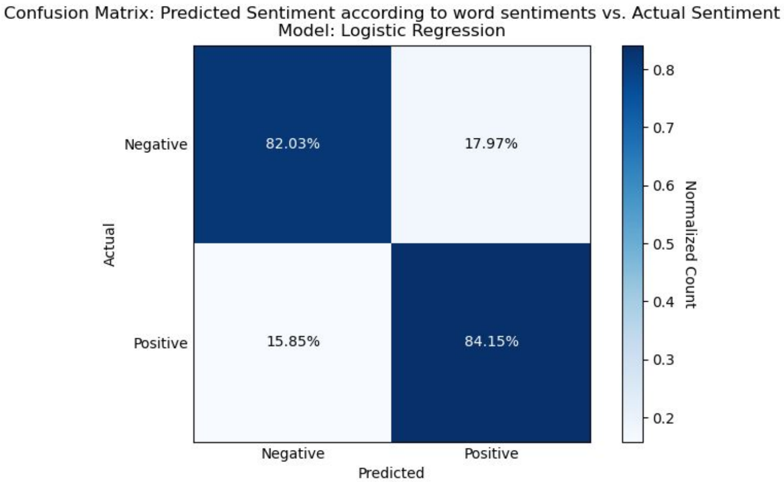
10 Most **Negative** Words:

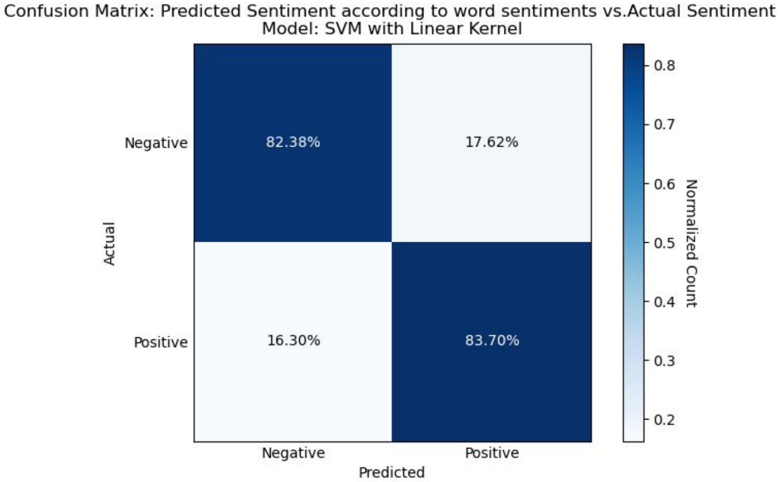
	word	weight
7132	worst	-1.134225
6971	waste	-1.081301
454	awful	-0.915316
687	boring	-0.766668
7130	worse	-0.713850
3082	horrible	-0.669245
440	avoid	-0.656827
6405	terrible	-0.610543
1806	disappointing	-0.604421
4764	poorly	-0.581686

Assigning Scores to Each Review

- We calculate the sum of the word weights of each review
- This sum is assigned as the total score for each review

	raw_text	modified_text	sentiment	word_count	total_score
0	One of the other reviewers has mentioned that ...	[One, reviewer, mention, watch, Oz, episode, y...	positive	320	-0.662223
1	A wonderful little production The filming tech...	[A, wonderful, little, production, The, film, ...	positive	166	0.866091
2	I thought this was a wonderful way to spend ti...	[I, think, wonderful, way, spend, time, hot, s...	positive	172	1.542105
3	Basically theres a family where a little boy J...	[Basically, there, family, little, boy, Jake, ...	negative	141	-0.584063
4	Petter Matteis Love in the Time of Money is a ...	[Petter, Matteis, Love, Time, Money, visually,...	positive	236	1.409738
5	Probably my alltime favorite movie a story of ...	[Probably, alltime, favorite, movie, story, se...	positive	125	1.864355
6	I sure would like to see a resurrection of a u...	[I, sure, would, like, see, resurrection, date...	positive	161	-0.318894
7	This show was an amazing fresh innovative ide...	[This, show, amaze, fresh, innovative, idea, f...	negative	181	-3.649886
8	Encouraged by the positive comments about this...	[Encouraged, positive, comment, film, I, look,...	negative	130	-1.441471
9	If you like original gut wrenching laughter yo...	[If, like, original, gut, wrench, laughter, li...	positive	34	0.106713
10	Phil the Alien is one of those quirky films wh...	[Phil, Alien, one, quirky, film, humour, base,...	negative	101	-0.466289
11	I saw this movie when I was about when it cam...	[I, saw, movie, I, come, I, recall, scary, sce...	negative	184	-0.791748
12	So im not a big fan of Bolls work but then aga...	[So, im, big, fan, Bolls, work, many, I, enjoy...	negative	412	-0.987699
13	The cast played ShakespeareShakespeare lostI a...	[The, cast, play, ShakespeareShakespeare, lost...	negative	122	-0.057414
14	This a fantastic movie of three prisoners who ...	[This, fantastic, movie, three, prisoner, beco...	positive	51	0.584989





Linear SVM:
Accuracy: 0.8302924268932766

	precision	recall	f1-score	support
0	0.84	0.82	0.83	2026
1	0.82	0.84	0.83	1975
accuracy			0.83	4001
macro avg	0.83	0.83	0.83	4001
weighted avg	0.83	0.83	0.83	4001

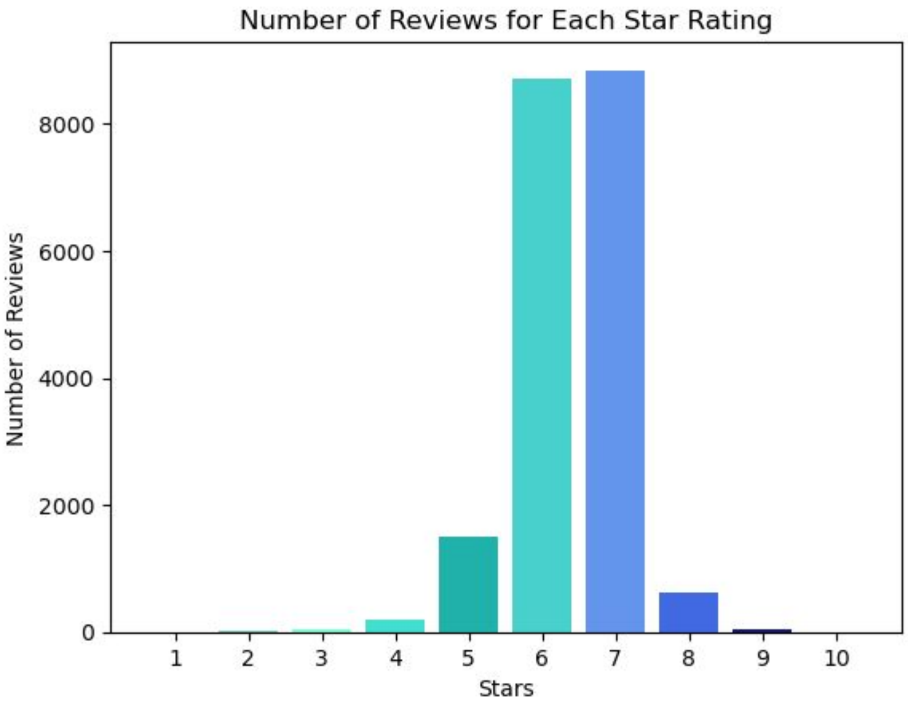
Polynomial SVM:
Accuracy: 0.7983004248937765

	precision	recall	f1-score	support
0	0.74	0.94	0.83	2026
1	0.91	0.65	0.76	1975
accuracy			0.80	4001
macro avg	0.82	0.80	0.79	4001
weighted avg	0.82	0.80	0.79	4001

RBF SVM:
Accuracy: 0.83104223944014

	precision	recall	f1-score	support
0	0.83	0.83	0.83	2026
1	0.83	0.83	0.83	1975
accuracy			0.83	4001
macro avg	0.83	0.83	0.83	4001
weighted avg	0.83	0.83	0.83	4001

Assigning a Star Rating to Each Review



Classifying New Samples

Positive Sample Review

"Oppenheimer is an exceptional film that deserves the highest praise for its thought-provoking narrative, exquisite craftsmanship, and compelling performances. From the very first frame, the movie draws the audience into the fascinating world of J. Robert Oppenheimer, the brilliant physicist whose contributions during World War II profoundly altered the course of history . . ."

```
#Positive Review for "Oppenheimer"
review_sample1 = "Oppenheimer is an exceptional film that deserves the h

print("Movie Sample 1's review score is: ",review_score(review_sample1))
```

```
Movie Sample 1's review score is:  9
```

Negative Sample Review

"Where do I even begin with the Oppenheimer movie? It's a perplexing mess of a film that fails to capture the essence of its subject matter and leaves the audience scratching their heads in confusion . . ."

```
#Negative Review for "Oppenheimer"
review_sample2 = "Where do I even begin with the Oppenheimer movie? Its a

print("Movie Sample 2's review score is: ", review_score(review_sample2))
```

```
Movie Sample 2's review score is:  5
```

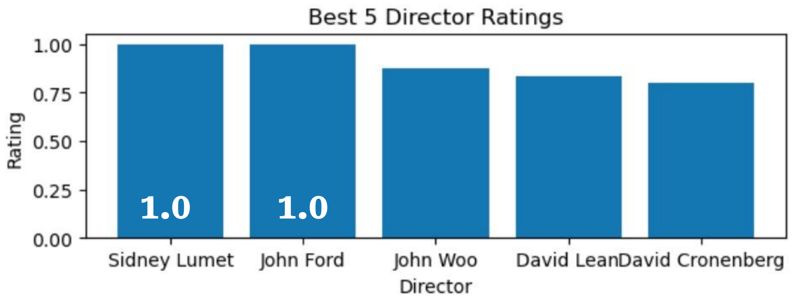
Sentiment Analysis and NER with Mentioned Names

Got the names of **well known directors, actors, actresses, and movie characters** via webscraping

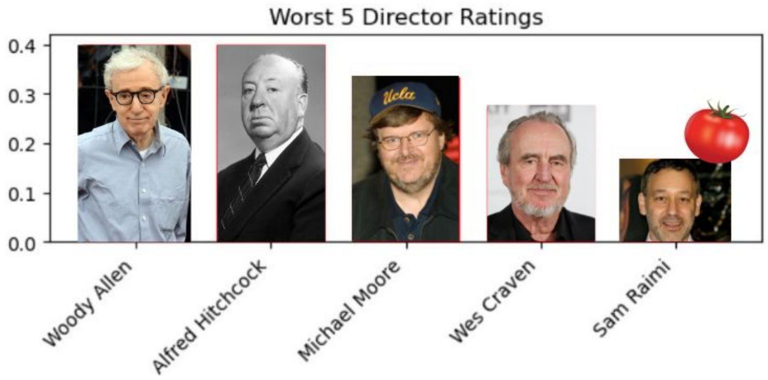
Used Spacy module to extract the name of entities appearing in movie reviews

Analyzed which people / movie characters had the highest or lowest average ratings

Directors

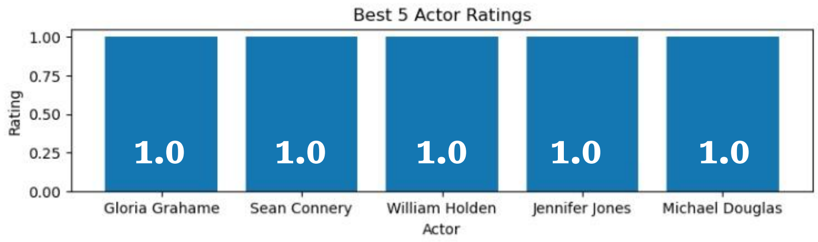


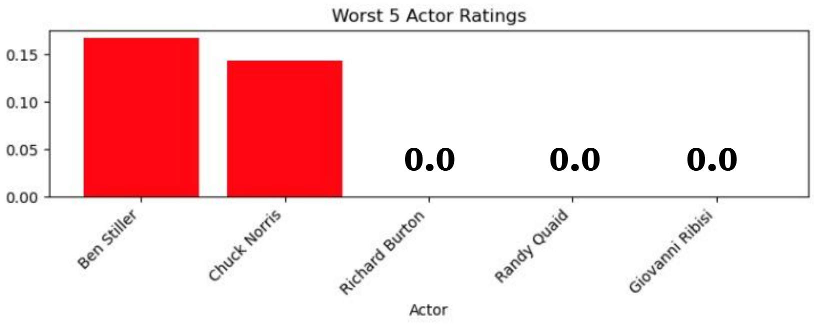
Dataset = 5000 reviews



Actors/Actresses

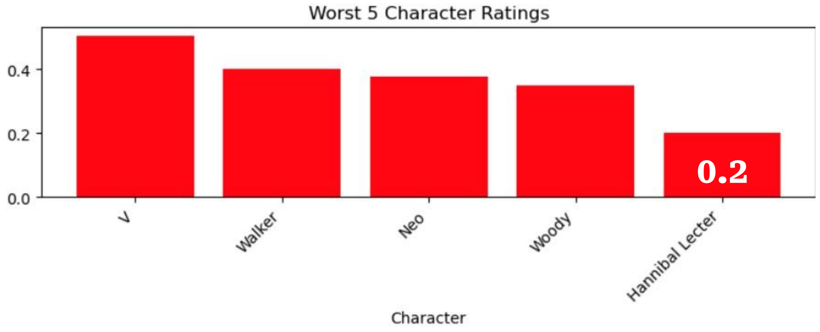
Dataset = 5000 reviews

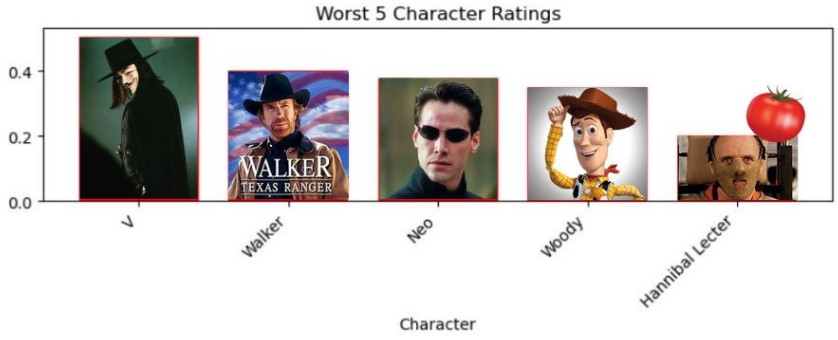




Movie Characters

Dataset = 5000 reviews

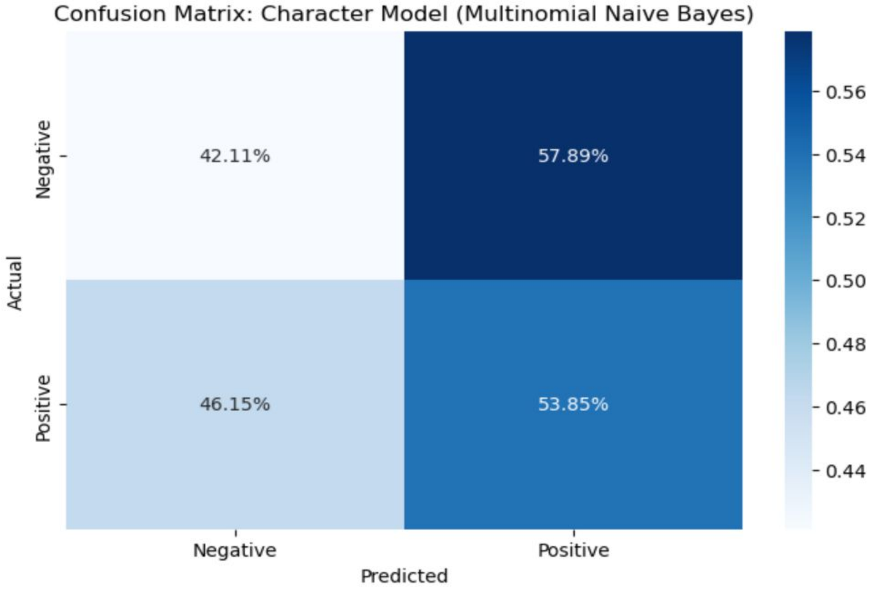




ML Model: Movie Review sentiment from mentioned movie characters

~56%

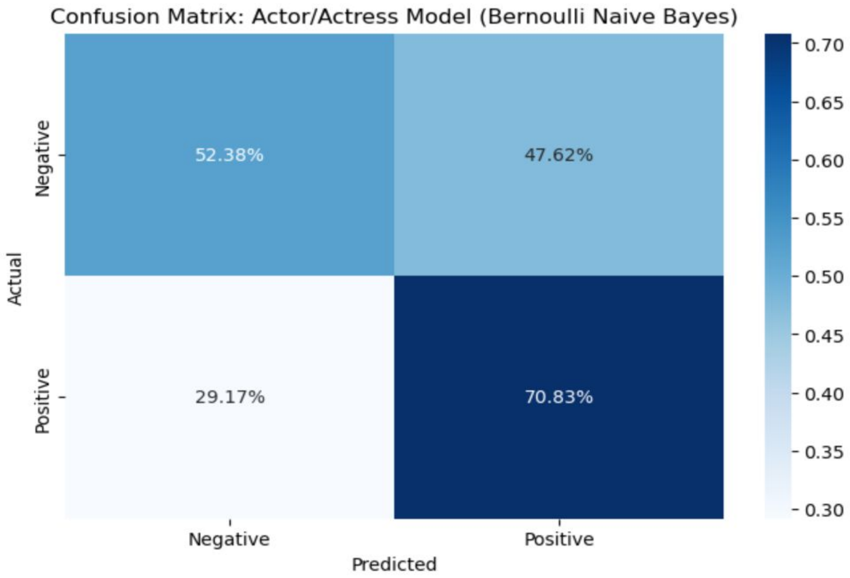
Accuracy



ML Model: Movie Review sentiment from mentioned actors/actresses

~60%

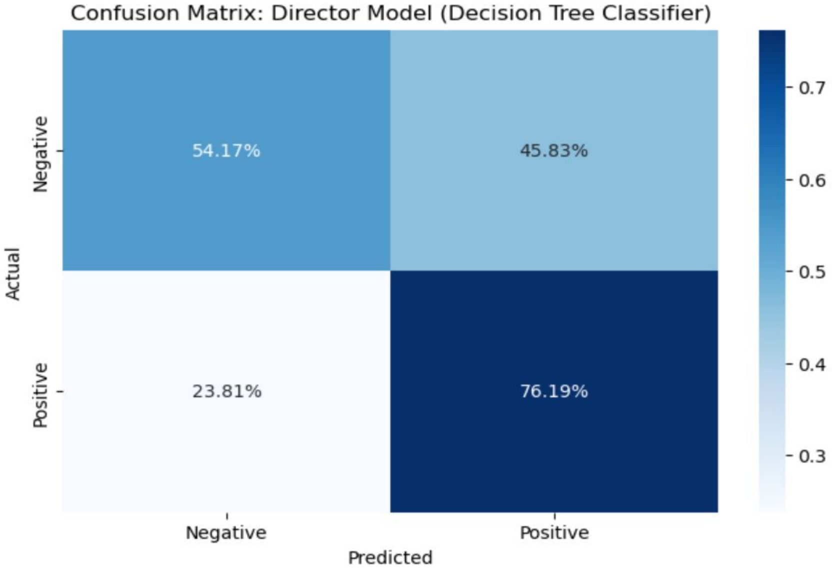
Accuracy



ML Model: Movie Review sentiment from mentioned directors

~63%

Accuracy



ML Model Review

For better accuracy: could’ve used more comprehensive names from webscraping

No strong correlation between mentioned names and movie reviews

Names of directors most indicative of movie sentiments

Movie Genre Prediction

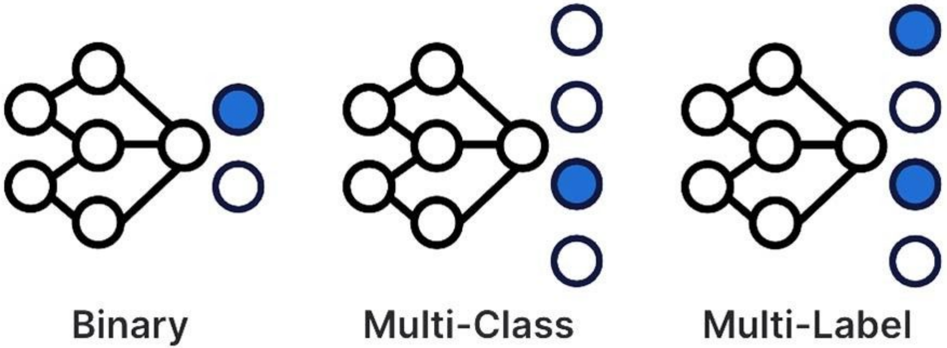
Dataset and Goal

Dataset: IMDb movies dataset from Kaggle

```
In [184]: df.columns

Out[184]: Index(['imdb_title_id', 'title', 'original_title', 'year', 'date_published',
                 'genre', 'duration', 'country', 'language', 'director', 'writer',
                 'production_company', 'actors', 'description', 'avg_vote', 'votes',
                 'budget', 'usa_gross_income', 'worlwide_gross_income', 'metascore',
                 'reviews_from_users', 'reviews_from_critics'],
                dtype='object')
```

```
In [185]: len(df)

Out[185]: 85855
```

Goal: Predict genres using descriptive texts of movie plots



genre
Romance
Biography, Crime, Drama
Drama
Drama, History
Adventure, Drama, Fantasy
Biography, Drama
Biography, Drama, Romance
Drama, History
History, War
Drama
Drama
Crime, Drama
Drama
Crime, Drama
Drama
Drama, War
Crime, Drama, Mystery
Drama
Drama, Fantasy, Horror
Crime, Drama
Adventure, Drama
Drama
Crime, Drama, Horror
Western

Make each genre ***even-sized***: 7 genres of 400 movies

```
# Explore genres with abundant data
# df['genre'].value_counts()[0:7]
df['genre'].value_counts()
```

```
Drama          12105
Comedy          7146
Horror          2241
Thriller        1217
Action           699
Western          588
Romance          415
Family           277
Sci-Fi           255
Adventure        232
Crime            157
Mystery          130
Musical          118
Animation         97
Fantasy           73
War               53
Biography         51
Music             27
History           26
Sport             15
Documentary        1
Name: genre, dtype: int64
```

	genre	description
0	Drama	Two men of high rank are both wooing the beaut...
1	Drama	Richard of Gloucester uses manipulation and mu...
2	Drama	After Dr. Friedrich's wife becomes mentally un...
3	Drama	Single mother is separated from her children d...
4	Drama	Leslie Swayne, an adventurer, in order to obta...
...	...	...
2795	Romance	Sato is 27 years old, lives in the northern pr...
2796	Romance	A family entertainer, the story of Ammammagari...
2797	Romance	Tej, a youngster who's highly attached to his ...
2798	Romance	The film is a rom-com which explores the life ...
2799	Romance	How will 3 sisters save the Shakespeare Chatea...

2800 rows × 2 columns

	genre	description
0	Drama	Two men of high rank are both wooing the beaut...
1	Drama	Richard of Gloucester uses manipulation and mu...
2	Drama	After Dr. Friedrich's wife becomes mentally un...
3	Drama	Single mother is separated from her children d...
4	Drama	Leslie Swayne, an adventurer, in order to obta...
...	...	...
2795	Romance	Sato is 27 years old, lives in the northern pr...
2796	Romance	A family entertainer, the story of Ammammagari...
2797	Romance	Tej, a youngster who's highly attached to his ...
2798	Romance	The film is a rom-com which explores the life ...
2799	Romance	How will 3 sisters save the Shakespeare Chatea...

2800 rows × 2 columns

Things need to be fixed:

1. Both upper and lower cases exist
   → Convert to all lower cases

2. Non-english characters exist (, . ‘’)
   → Replace them by empty string
   → Save spaces at this point! We’ll use them to chop sentences into words soon

3. Same words now have different forms
   E.g. man VS men
   become VS becomes
   separate VS separated
   → Requires lemmatization!

→ ***Similar Preprocessing Steps***

Before training our models…

1. “Word To Vectors” - Tf-idf

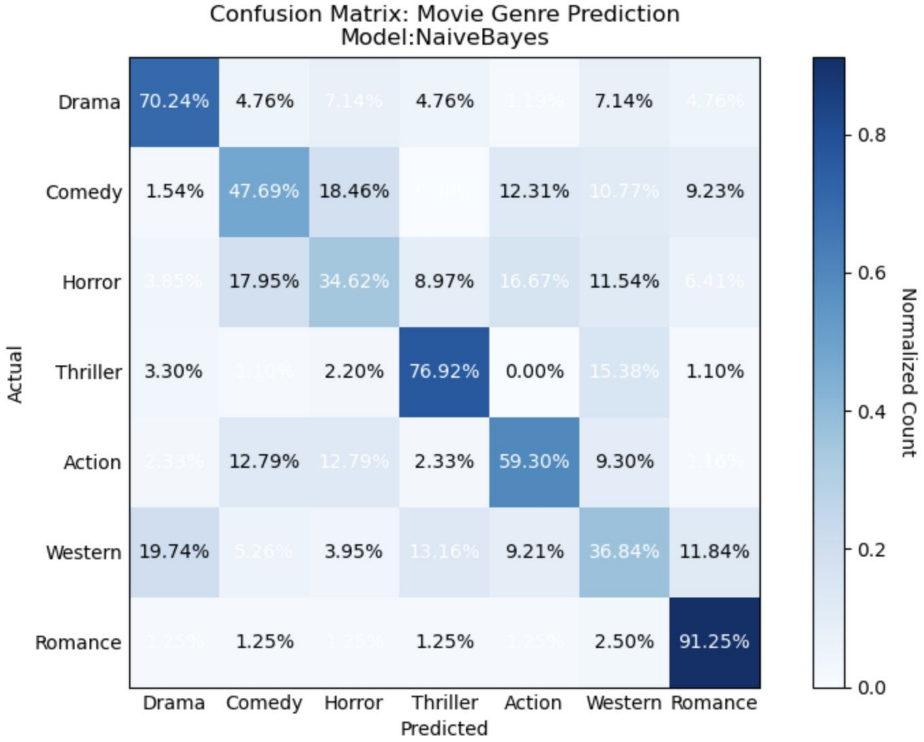
```
In [232]: # Vectorize words using Tf-Idf
          vec = TfidfVectorizer(stop_words = "english")
          tfidf = vec.fit_transform(data['description'])
          tfidf = tfidf.toarray()
          tfidf_df = pd.DataFrame(tfidf, columns = vec.get_feature_names_out())
          tfidf_df
Out[232]:
```

	aadi	aage	abacco	abandon	abandoned	abbey	abbott	abby	abdicate	abduct	...	zenith	zhai	zhang	zhao	zhigalovs	zinochka	zita	zombie	zo
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2795	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2796	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2797	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2798	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2799	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

2800 rows × 9278 columns

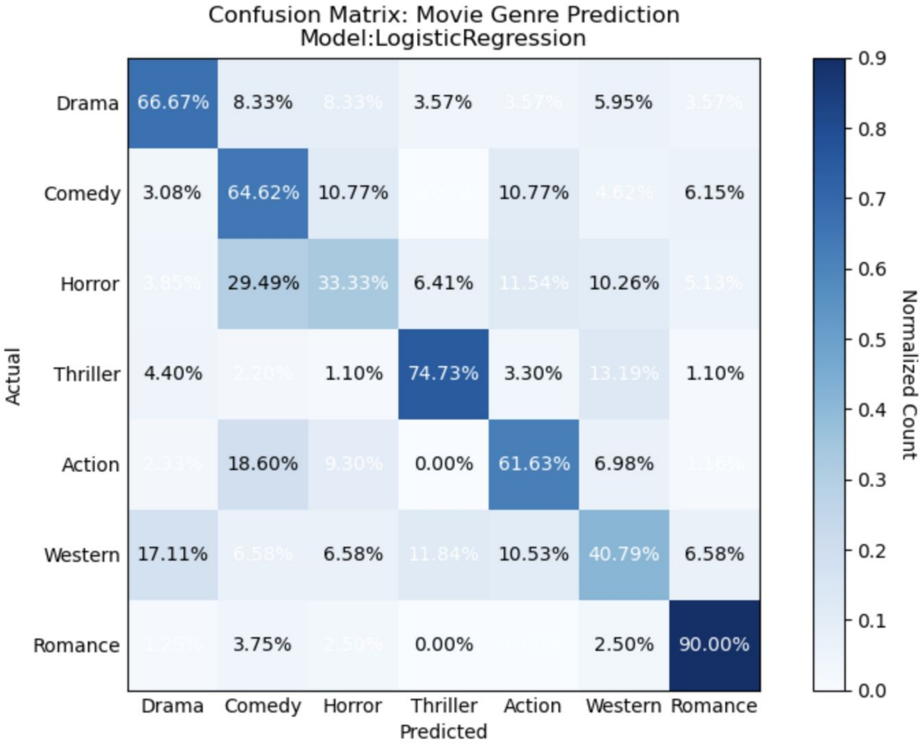
2. Features: word vectors from plot descriptions / Targets: movie genres
3. Train Test Split

Model_1: Multinomial Naive Bayes




```
Multinomial Naive Bayes:
Accuracy: 0.6053571428571428
              precision    recall  f1-score   support

      Action       0.70      0.70      0.70        84
      Comedy       0.47      0.48      0.47        65
       Drama       0.44      0.35      0.39        78
      Horror       0.74      0.77      0.76        91
     Romance       0.63      0.59      0.61        86
    Thriller       0.38      0.37      0.37        76
     Western       0.74      0.91      0.82        80

    accuracy                           0.61       560
   macro avg       0.59      0.60      0.59       560
weighted avg       0.59      0.61      0.60       560
```

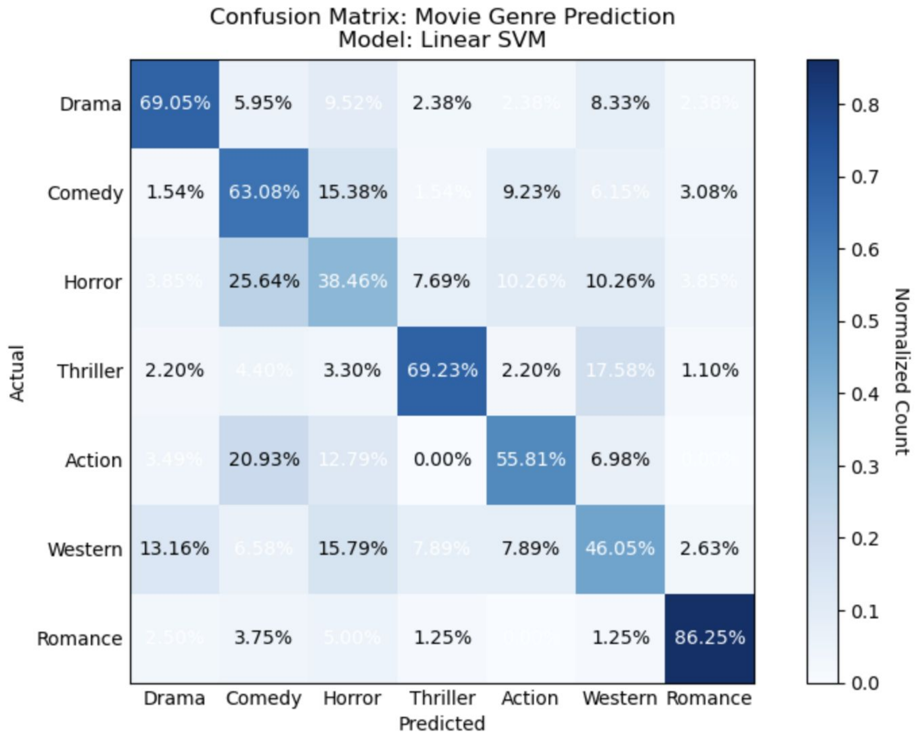
Model_2: Logistic Regression



Logistic Regression:
Accuracy: 0.6214285714285714

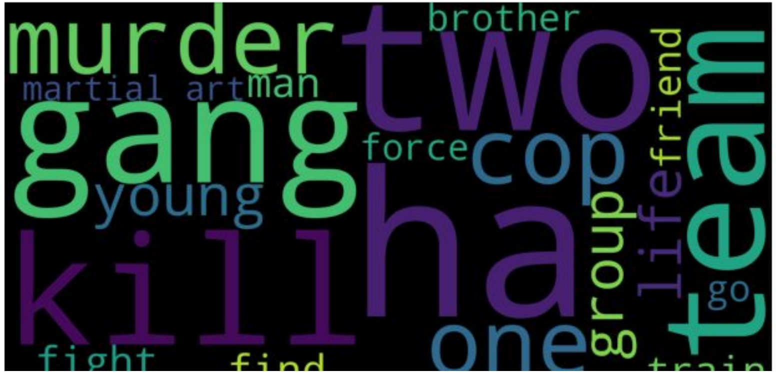
	precision	recall	f1-score	support
Action	0.69	0.67	0.68	84
Comedy	0.43	0.65	0.52	65
Drama	0.46	0.33	0.39	78
Horror	0.80	0.75	0.77	91
Romance	0.64	0.62	0.63	86
Thriller	0.46	0.41	0.43	76
Western	0.80	0.90	0.85	80
accuracy			0.62	560
macro avg	0.61	0.62	0.61	560
weighted avg	0.62	0.62	0.62	560

Model_3: Linear Support Vector Machine

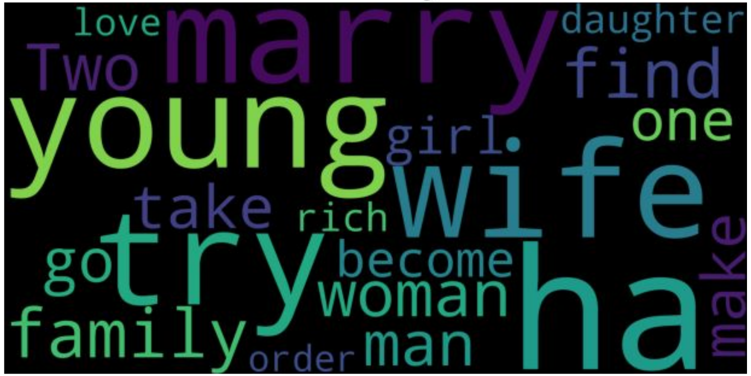



```
Linear SVM:
Accuracy: 0.6142857142857143
              precision    recall  f1-score   support

      Action       0.73      0.69      0.71        84
      Comedy       0.43      0.63      0.51        65
       Drama       0.38      0.38      0.38        78
      Horror       0.80      0.69      0.74        91
     Romance       0.67      0.56      0.61        86
    Thriller       0.45      0.46      0.46        76
     Western       0.87      0.86      0.87        80

    accuracy                           0.61       560
   macro avg       0.62      0.61      0.61       560
weighted avg       0.63      0.61      0.62       560
```

action



comedy



horror

Observations:

1. Frequent words not that meaningful
   → "two" "become" "go" "find"
   → overlaps among genres

2. Not-perfect NLP preprocessing
   → "ha": probably stripping the ending 's' from "has"

Pytorch Neural Network + Bert Transformer

```
# Encoding
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)

# Model selection
model_name = "bert-base-uncased"
tokenizer = BertTokenizer.from_pretrained(model_name)
embedding_model = BertModel.from_pretrained(model_name)

# Device selection
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
embedding_model.to(device)

# Tokenizer & embeddings
max_length = 20
encoded_inputs = tokenizer(list(train_dataset['description']), padding='max_length', truncation=True,
                           max_length=max_length, return_attention_mask=True)

train_dataset = TensorDataset(torch.tensor(encoded_inputs['input_ids']),
                              torch.tensor(encoded_inputs['attention_mask']), torch.tensor(y_train_encoded))
train_loader = DataLoader(train_dataset, batch_size=12, shuffle=True)

encoded_inputs_test = tokenizer(list(test_dataset['description']), padding='max_length', truncation=True,
                                max_length=max_length, return_attention_mask=True)
y_test_encoded = label_encoder.fit_transform(y_test)

test_dataset = TensorDataset(torch.tensor(encoded_inputs_test['input_ids']),
                             torch.tensor(encoded_inputs_test['attention_mask']), torch.tensor(y_test_encoded))
test_loader = DataLoader(test_dataset, batch_size=12, shuffle=True)
```

 # Categorical Genres → Numerical Labels

 # Model / Device Preparation

 # Tokenization + Embeddings

```python
# nn definition
class GenreClassifier(nn.Module):
    def __init__(self, embedding_model, num_classes):
        super(GenreClassifier, self).__init__()
        self.embedding_model = embedding_model
        self.fc = nn.Linear(embedding_model.config.hidden_size, num_classes)

    def forward(self, input_ids, attention_mask):
        embeddings = self.embedding_model(input_ids, attention_mask=attention_mask).last_hidden_state[:, 0]
        logits = self.fc(embeddings)
        return logits

# Create the new model and move to device
num_classes = len(label_encoder.classes_)
model = GenreClassifier(embedding_model, num_classes)
model.to(device)

# Optimizer & loss
optimizer = optim.AdamW(model.parameters(), lr=1e-5)
loss_fn = nn.CrossEntropyLoss()
```

```python
# Train...
model.train()
for epoch in range(10):
    progress_bar = tqdm(train_loader, desc=f"Epoch {epoch + 1}/10", leave=False)
    total_correct = 0
    total_samples = 0

    for batch in progress_bar:
        optimizer.zero_grad()
        input_ids, attention_mask, labels = [item.to(device) for item in batch]
        logits = model(input_ids, attention_mask)
        loss = loss_fn(logits, labels)
        loss.backward()
        optimizer.step()

        # accuracy
        _, predicted = torch.max(logits, 1)
        total_correct += (predicted == labels).sum().item()
        total_samples += labels.size(0)
        accuracy = total_correct / total_samples

        progress_bar.set_postfix({"loss": loss.item(), "accuracy": accuracy})

        # Accuracy * epoch
    print(f'Epoch {epoch + 1} - Accuracy: {accuracy:.4f}')
```

```
Epoch 1 - Accuracy: 0.4147
Epoch 2 - Accuracy: 0.6504
Epoch 3 - Accuracy: 0.7643
Epoch 4 - Accuracy: 0.8661
Epoch 5 - Accuracy: 0.9268
Epoch 6 - Accuracy: 0.9665
Epoch 7 - Accuracy: 0.9830
Epoch 8 - Accuracy: 0.9955
Epoch 9 - Accuracy: 0.9982
Epoch 10 - Accuracy: 0.9951
```

```python
# Evaluate...
model.eval()
total_correct = 0
total_samples = 0
with torch.no_grad():
    progress_bar = tqdm(test_loader, desc="Evaluating", leave=False)
    for batch in progress_bar:
        input_ids, attention_mask, labels = [item.to(device) for item in batch]
        logits = model(input_ids, attention_mask)
        _, predicted = torch.max(logits, 1)
        total_correct += (predicted == labels).sum().item()
        total_samples += labels.size(0)
        progress_bar.set_postfix({"accuracy": total_correct / total_samples})

accuracy = total_correct / total_samples
print(f'Final Accuracy: {accuracy:.4f}')
```

```
Final Accuracy: 0.6714
```

Inspired by the open notebook done by EUGENIO SCHIAVONI on Kaggle
https://www.kaggle.com/code/eugeniokukes/bert-torch-superior-to-tensorflow-22min-run-0-65

TBC on NLP...

- Multi-label Problems of Text Categorization

- Scrutinize on each step: lemmatization, embedding…
- More advanced model to be trained other than traditional ML
- More powerful tools to explore (E.g. KerasNLP)

Thank You!