University of California, Irvine

The Paul Merage School of Business

Data & Programming Final Project:
Are Oscar Award Winner Movies really that “good?”

Group Members: Ryan Charles Donaghy, Lei Ye, Chirag Madhukar, Flavio Wang, Haixin Zhao

BANA 212: Data & Programming Analytics

Professor Nian

4 Dec. 2021

Are Oscar Award Winner Movies really that “good?”

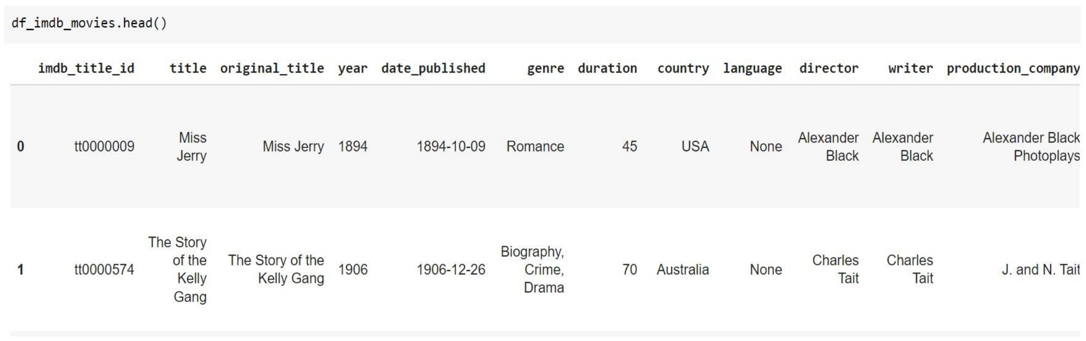
Introduction

For this project, our group used data mining/analysis to determine whether the Oscar Award winning movie in the best picture category each year accurately represented the public's favorite movie that year. Our interest in this topic was fueled by recent news headlines reporting the substantial decline in viewership of the awards ceremony. Although the award is still regarded as the highest honor in the film industry and is watched by millions each year, we found the results of our inquiry shocking. According to Nielsen, viewership for the 2021 awards dropped 59% compared to 2020, from 23.6 million to 9.85 million. This drastic decline combined with negative online sentiment, whereby Oscars judges are critiqued for only highly rating the movies that fit their specific tastes and not those that are perceived to be the best by the public.

To assist us in answering this question, we acquired datasets from both Kaggle and iMDB. Both of these datasets will be covered in greater detail in the following section.

About the Data

From Kaggle



```
df_imdb_movies.head()
```

	imdb_title_id	title	original_title	year	date_published	genre	duration	country	language	director	writer	production_company
0	tt0000009	Miss Jerry	Miss Jerry	1894	1894-10-09	Romance	45	USA	None	Alexander Black	Alexander Black	Alexander Black Photoplays
1	tt0000574	The Story of the Kelly Gang	The Story of the Kelly Gang	1906	1906-12-26	Biography, Crime, Drama	70	Australia	None	Charles Tait	Charles Tait	J. and N. Tait

(Figure 1: The screenshot of the dataset we used)

We found a dataset from Kaggle, *IMDB movies.csv*. This file is a web scrape of the IMDB Database with 22 columns, consisting of the movies name, date of release, genre, production company, actor names, average rating (See figure 1). We conducted some exploratory analysis using this dataset.

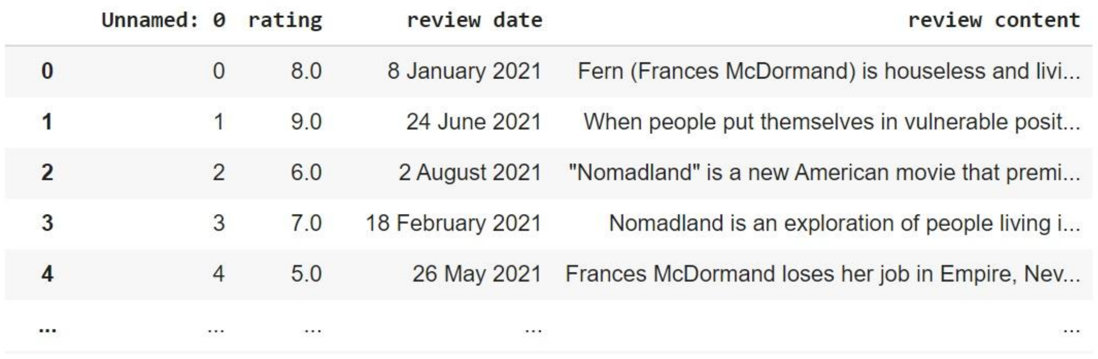
Web scraping

The other dataset we used consists of user's ratings and reviews for each Oscar winning movie from 1990-2020. To collect this data, we utilized a web scraping tool on the IMDB website.

The scraping code was created with Python and BeautifulSoup packages and it was run on every oscar winning movie's webpage.

This web scraping tool enabled us to collect users' reviews and ratings (there were reviews that didn't have a rating, in such cases we placed "NA" in the cells) and posting date of every review. Since there was a load more button and each button had an ajax url attached to it on the html, we used BeautifulSoup to find the button and open that url with requests, take the data and save it in its respective list and then load the new page until there weren't anymore load more buttons.

We compiled the lists into a dataframe and wrote it to a csv file. In the end we had one csv file for every Oscar winning movie. Sample of the scraped data is pasted below.



	Unnamed: 0	rating	review date	review content
0	0	8.0	8 January 2021	Fern (Frances McDormand) is houseless and livi...
1	1	9.0	24 June 2021	When people put themselves in vulnerable posit...
2	2	6.0	2 August 2021	"Nomadland" is a new American movie that premi...
3	3	7.0	18 February 2021	Nomadland is an exploration of people living i...
4	4	5.0	26 May 2021	Frances McDormand loses her job in Empire, Nev...
...	...	...	...	...

Machine learning: Sentiment Analysis on iMDB Reviews

In this section we'll analyze the machine learning model we have created. In order to see if the reviews were biased in some way content-wise, we wanted to see how a machine would rate the reviews.

Text Pre-processing, Tokenization, and Vectorization

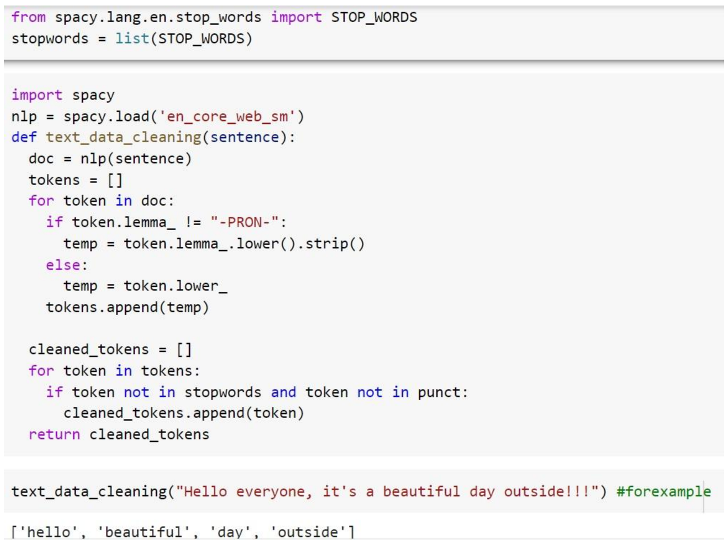
Tokenization is essentially a form of text pre-processing. It is a process of converting text into tokens before transforming it into vectors. For example, a document into paragraphs or sentences to words. In this case we are tokenizing reviews into words.

These words are later converted into vectors used to build different machine learning models.

We did some text preprocessing before running it through a machine learning model by performing the following steps:

- By removing stop words like ‘the’, 'if', 'but', 'we', 'he', 'she', and 'they’, etc.
- By removing punctuations, extra blank spaces
- By converting everything into lower case

Below is the code for text-preprocessing and an example that shows a preprocessed text:



```
from spacy.lang.en.stop_words import STOP_WORDS
stopwords = list(STOP_WORDS)
```

```
import spacy
nlp = spacy.load('en_core_web_sm')
def text_data_cleaning(sentence):
  doc = nlp(sentence)
  tokens = []
  for token in doc:
    if token.lemma_ != "-PRON-":
      temp = token.lemma_.lower().strip()
    else:
      temp = token.lower_
    tokens.append(temp)

  cleaned_tokens = []
  for token in tokens:
    if token not in stopwords and token not in punct:
      cleaned_tokens.append(token)
  return cleaned_tokens
```

```
text_data_cleaning("Hello everyone, it's a beautiful day outside!!!") #forexample
```

```
['hello', 'beautiful', 'day', 'outside']
```

We used two types of vectorization methods to convert the tokens into vectors. They are TF-IDF vectorization and Count vectorization. TF-IDF vectorization was used for Linear SVC and Random Forest Classification while Count vectorization was used for Naive Bayes Classification.

To make computers understand texts, we convert text into numbers or vectors. We do this by using two vectorizers.

- **Count Vectorization:** Count vectorizer is a way to convert a given set of strings into a frequency representation. Count Vectors can be helpful in understanding

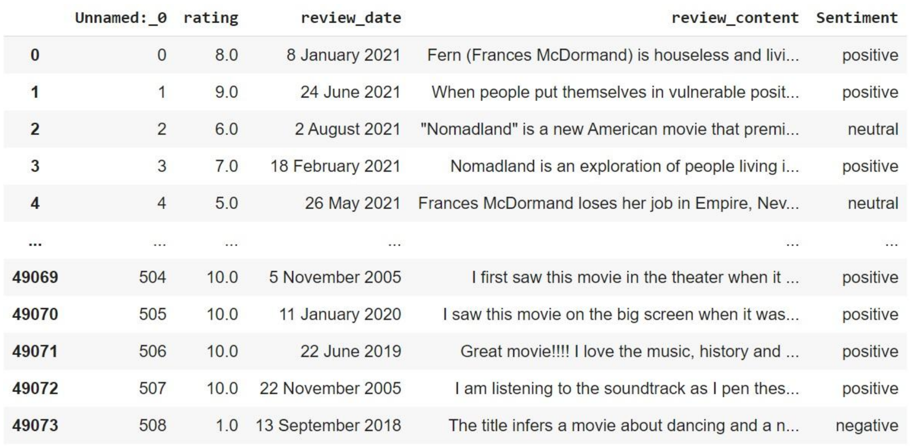
the type of text by the frequency of words in it. But some of its major disadvantages are:
(i) It's inability in identifying more important and less important words for analysis.
(ii) It only considers words that are abundant in a corpus as the most statistically significant word.
(iii) It also doesn't identify the relationships between words such as linguistic similarity between words.

- **TF-IDF Vectorization:** TF-IDF means Term Frequency - Inverse Document Frequency. This is a statistic that is based on the frequency of a word in the corpus but it also provides a numerical representation of how important a word is for statistical analysis.

  TF-IDF works better than Count Vectorizers because not only does it focus on the frequency of words present in the corpus but also provides the importance of the words.

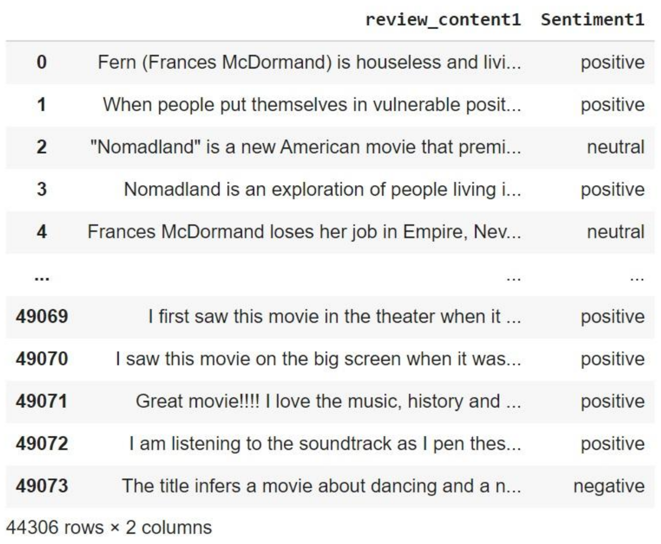
Machine Learning Models:

We assigned sentiments for each review based on the user ratings. If a user reviewed and rated the movie 7 points and above, we assigned a "positive" rating sentiment, if a user reviewed and rated the movie 4.9 and below, we assigned a "negative" sentiment rating, and the ratings which had any ratings in between 4.9 and 7 we assigned a neutral "sentiment" rating.



	Unnamed:_0	rating	review_date	review_content	Sentiment
0	0	8.0	8 January 2021	Fern (Frances McDormand) is houseless and livi...	positive
1	1	9.0	24 June 2021	When people put themselves in vulnerable posit...	positive
2	2	6.0	2 August 2021	"Nomadland" is a new American movie that premi...	neutral
3	3	7.0	18 February 2021	Nomadland is an exploration of people living i...	positive
4	4	5.0	26 May 2021	Frances McDormand loses her job in Empire, Nev...	neutral
...	...	...	...	...	...
49069	504	10.0	5 November 2005	I first saw this movie in the theater when it ...	positive
49070	505	10.0	11 January 2020	I saw this movie on the big screen when it was...	positive
49071	506	10.0	22 June 2019	Great movie!!!! I love the music, history and ...	positive
49072	507	10.0	22 November 2005	I am listening to the soundtrack as I pen thes...	positive
49073	508	1.0	13 September 2018	The title infers a movie about dancing and a n...	negative

This was the data finally used for running different Machine Learning algorithms.



	review_content1	Sentiment1
0	Fern (Frances McDormand) is houseless and livi...	positive
1	When people put themselves in vulnerable posit...	positive
2	"Nomadland" is a new American movie that premi...	neutral
3	Nomadland is an exploration of people living i...	positive
4	Frances McDormand loses her job in Empire, Nev...	neutral
...	...	...
49069	I first saw this movie in the theater when it ...	positive
49070	I saw this movie on the big screen when it was...	positive
49071	Great movie!!!! I love the music, history and ...	positive
49072	I am listening to the soundtrack as I pen thes...	positive
49073	The title infers a movie about dancing and a n...	negative

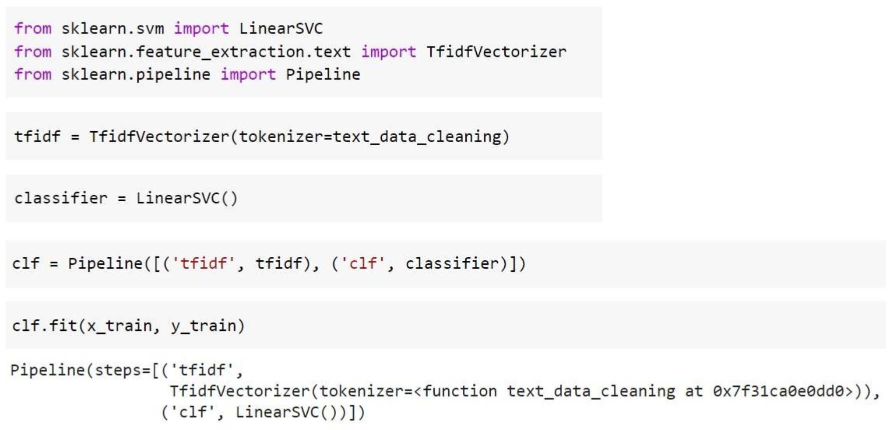
44306 rows × 2 columns

X is the "review_content" column
Y is the "Sentiment" column

1. **Support Vector Classification**
"Support Vector Machine" (SVM) is a supervised machine learning algorithm that can be used for both classification and regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates these classes very well. In our case, "Positive", "Negative", or "Neutral".

So once the text was preprocessed and converted into tokens, they were then converted into vectors using TF-IDF vectorization.

The data is split into 80% and 20% training and testing sets and Linear SVC classifier is run on the model.
Code:



```
from sklearn.svm import LinearSVC
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline
```

```
tfidf = TfidfVectorizer(tokenizer=text_data_cleaning)
```

```
classifier = LinearSVC()
```

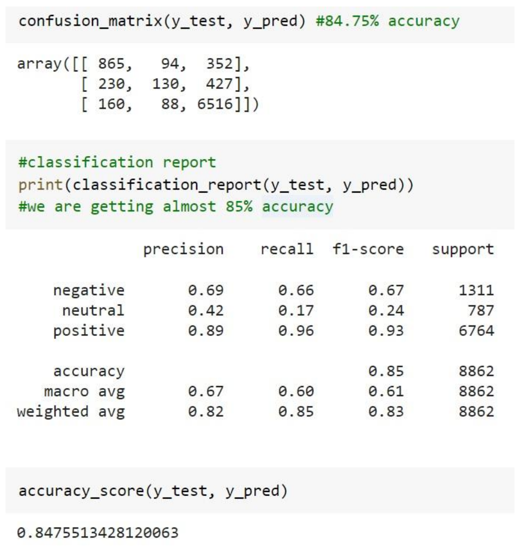
```
clf = Pipeline([('tfidf', tfidf), ('clf', classifier)])
```

```
clf.fit(x_train, y_train)
```

```
Pipeline(steps=[('tfidf',
                 TfidfVectorizer(tokenizer=<function text_data_cleaning at 0x7f31ca0e0dd0>)),
                ('clf', LinearSVC())])
```

<u>Analysis Summary:</u>

We find that the accuracy of model is 84.75%



```
confusion_matrix(y_test, y_pred) #84.75% accuracy
```

```
array([[ 865,   94,  352],
       [ 230,  130,  427],
       [ 160,   88, 6516]])
```

```
#classification report
print(classification_report(y_test, y_pred))
#we are getting almost 85% accuracy
```

```
              precision    recall  f1-score   support

    negative       0.69      0.66      0.67      1311
     neutral       0.42      0.17      0.24       787
    positive       0.89      0.96      0.93      6764

    accuracy                           0.85      8862
   macro avg       0.67      0.60      0.61      8862
weighted avg       0.82      0.85      0.83      8862
```

```
accuracy_score(y_test, y_pred)
```

```
0.8475513428120063
```

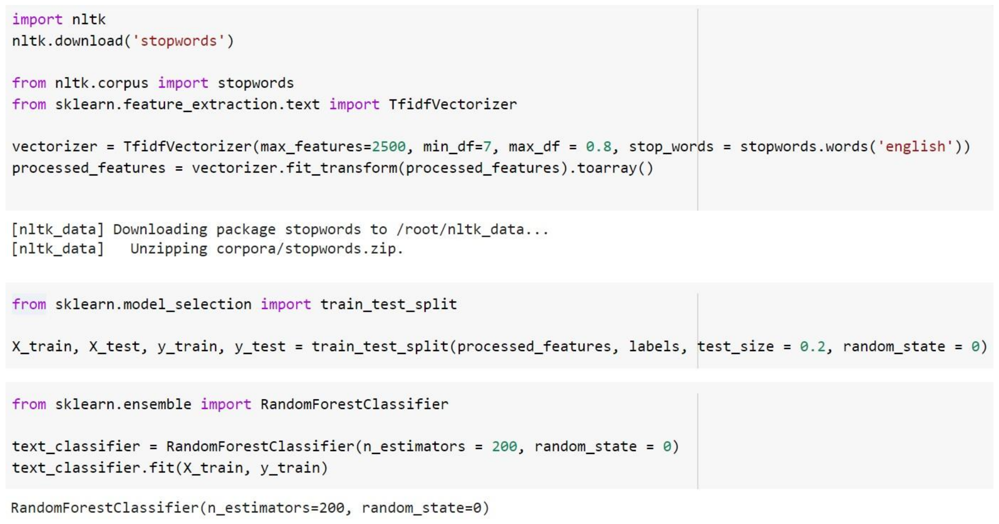
2. **Random Forest Classification**

   The random forest is a **classification algorithm** consisting of many decision trees. It uses bagging and feature randomness when building each individual tree to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree. In our case, in this Architecture, there is a large dataset consisting of certain rows and columns and since we are using Random Forest means that there will be multiple models of individual decision trees using the concept of Bagging, which means using multiple machine learning algorithms.

Once the text was preprocessed and converted into tokens, they were converted into vectors using TF-IDF vectorization.

The data is split into 80% and 20% training and testing sets and Random Forest Classifier is run on the model with n_estimators equal to 200 (which is the number of trees in the forest.

<u>Code:</u>



```
import nltk
nltk.download('stopwords')

from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(max_features=2500, min_df=7, max_df = 0.8, stop_words = stopwords.words('english'))
processed_features = vectorizer.fit_transform(processed_features).toarray()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(processed_features, labels, test_size = 0.2, random_state = 0)
```

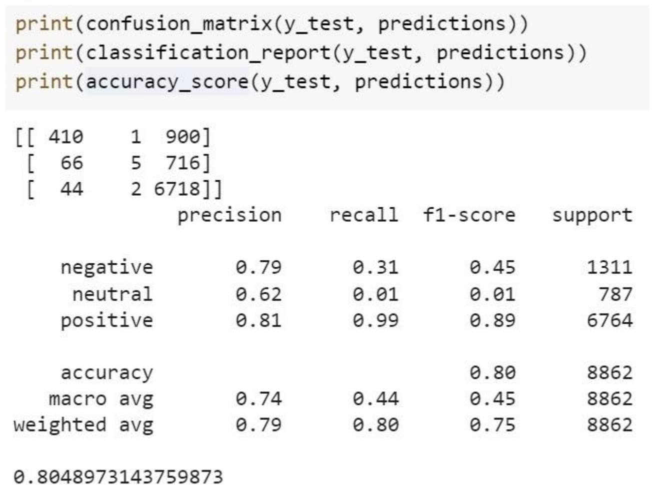
```
from sklearn.ensemble import RandomForestClassifier

text_classifier = RandomForestClassifier(n_estimators = 200, random_state = 0)
text_classifier.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=200, random_state=0)
```

<u>Analysis Summary:</u>

We find that the accuracy of the model is 80.49%.



```
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
```

```
[[ 410    1  900]
 [  66    5  716]
 [  44    2 6718]]
              precision    recall  f1-score   support

    negative       0.79      0.31      0.45      1311
     neutral       0.62      0.01      0.01       787
    positive       0.81      0.99      0.89      6764

    accuracy                           0.80      8862
   macro avg       0.74      0.44      0.45      8862
weighted avg       0.79      0.80      0.75      8862

0.8048973143759873
```

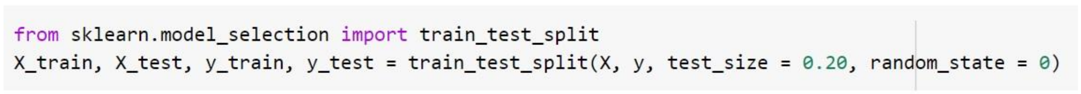
3. **Naive Bayes Classification**

   Gaussian Naive Bayes is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data. Naive Bayes are a group of supervised machine learning classification algorithms based on the Bayes theorem. It is a simple classification technique, but has high functionality. They find use when the dimensionality of the inputs is high. Complex classification problems can also be implemented by using Naive Bayes Classifier.

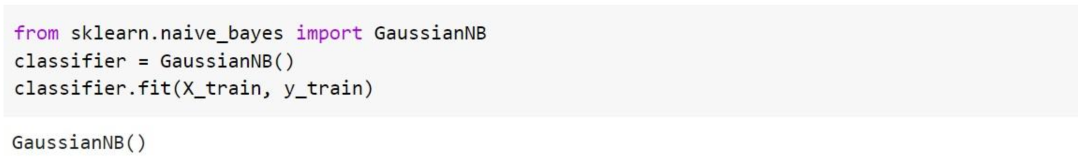
   So once the text was preprocessed and converted into tokens, they were converted into vectors using Count Vectorization.

   The data is split into 80% and 20% training and testing sets and Gaussian Naive Bayes is run on the model.

   Code:



```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
```

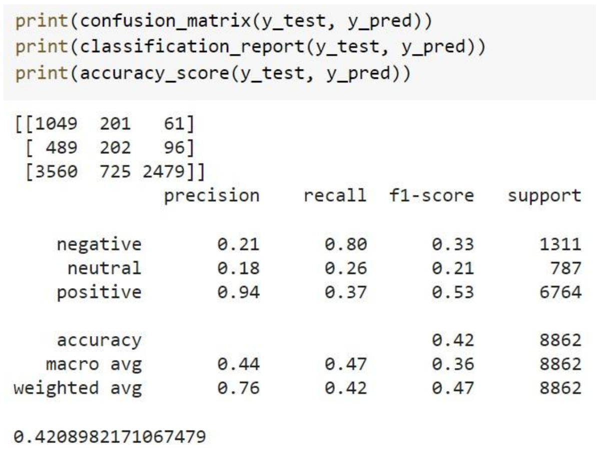


```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

```
GaussianNB()
```

   Analysis Summary:

   We find that the accuracy of model is 42.09%



```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
```

```
[[1049  201   61]
 [ 489  202   96]
 [3560  725 2479]]
              precision    recall  f1-score   support

    negative       0.21      0.80      0.33      1311
     neutral       0.18      0.26      0.21       787
    positive       0.94      0.37      0.53      6764

    accuracy                           0.42      8862
   macro avg       0.44      0.47      0.36      8862
weighted avg       0.76      0.42      0.47      8862

0.4208982171067479
```

Models results

MODEL	ACCURACY	CONFUSION MATRIX

Simple Vector Classification	84.76%	`[[ 865   94  352]`
		` [ 230  130  427]`
		` [ 160   88 6516]]`

Random Forest Classification	80.49%	`[[ 410    1  900]`
		` [  66    5  716]`
		` [  44    2 6718]]`

Naive Bayes Classification	42.09%	`[[1049  201   61]`
		` [ 489  202   96]`
		` [3560  725 2479]]`

From the above results we can see that the Simple Vector Classification (84.76%) performed slightly better than the Random Forest (80.49%) and drastically better than the Naive Bayes Classification (42.09%).

Stratified accuracies for SVC Model is as follows:
For “Negative” = 65.98%
For “Neutral” = 16.52%
For “Positive” = 96.33%

The results also indicate that even our best model has difficulty assessing whether a reviews’ content is actually negative or neutral. Therefore, people might rate lower than how they actually view the movie on a scale of 1-10. They might believe it was a decent movie but rate it badly, for example a 5 because it’s the middle score, however a 5 is a low score in a rating.

Text Analysis

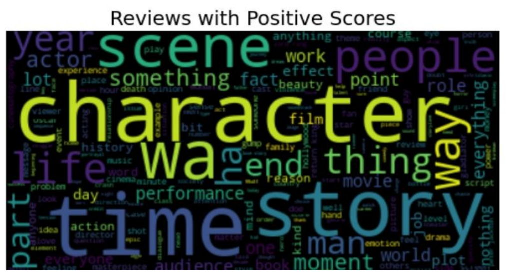
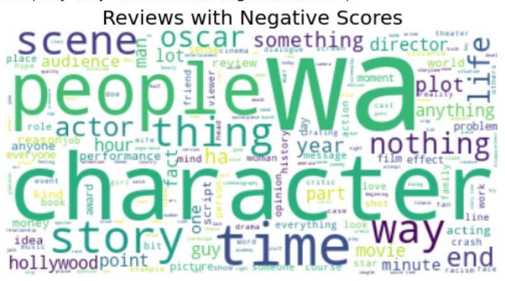
Text Blob

TextBlob python library for NLP. It has a pre-defined dictionary for negative and positive words. It will assign scores to every words and the final results will be calculated through some pooling operation. It will return two scores: polarity and subjectivity. Polarity lies between [-1,1], -1 defines a negative sentiment and 1 defines a positive sentiment. Negation words reverse the polarity. TextBlob has semantic labels that help with fine-grained analysis. For example — emoticons, exclamation marks, emojis, etc. Subjectivity lies between [0,1]. Subjectivity quantifies the amount of personal opinion and factual information contained in the text. The higher subjectivity means that the text contains more personal opinion rather than factual information.

Word Cloud:

A word cloud was built to find the most frequently occurring words in the positive and negative reviews.

Negative Reviews	**Positive Reviews**

`pos_rev_words = nltk.word_tokenize(positive_reviews_str)`
`pos_rev_words = nltk.pos_tag(pos_rev_words)`
`pos_words = ''`
`for tp in pos_rev_words:`
`  if tp[0] == "movie" or tp[0] == "film":`
`    continue`
`  if tp[1][0] == 'N':`
`    pos_words += tp[0] + ' '`
`wordcloud_positive = WordCloud(background_color='black').generate(pos_words)`
`fig = plt.figure(figsize=(10,10))`
`ax2 = fig.add_subplot(212)`
`ax2.imshow(wordcloud_positive,interpolation='bilinear')`
`ax2.axis("off")`
`ax2.set_title('Reviews with Positive Scores',fontsize=20)`
`plt.show()`

	`wordcloud_negative = WordCloud(background_color='white').generate(neg_words)`
	`# Plot`
	`fig = plt.figure(figsize=(10,10))`
	`ax1 = fig.add_subplot(211)`
	`ax1.imshow(wordcloud_negative,interpolation='bilinear')`
	`ax1.axis("off")`
	`ax1.set_title('Reviews with Negative Scores',fontsize=20)`
	`Text(0.5, 1.0, 'Reviews with Negative Scores')`
	

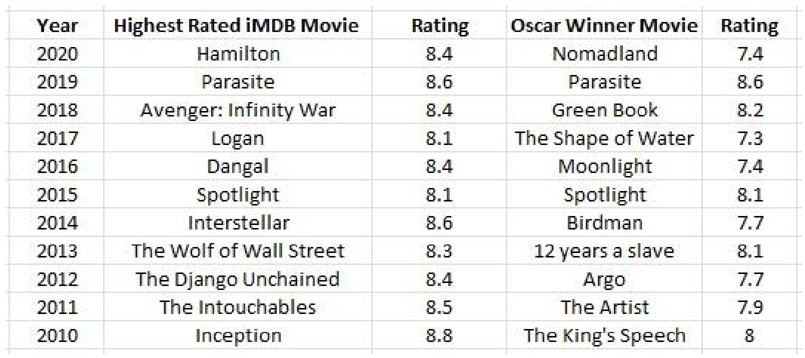
We can see from the two word clouds that there are no differences between the most used words in positive and negative reviews. This could mean that the reason why in many years Oscar winning movies were different from the highest rated movie on IMDB that year is due to just a personal preference and that probably the genre of the movie didn’t fit everyone.

Our findings through visualization

In the following paragraphs we will present our findings with regards to the relationship between the two datasets using graphs.

- **iMDB vs. Oscar**

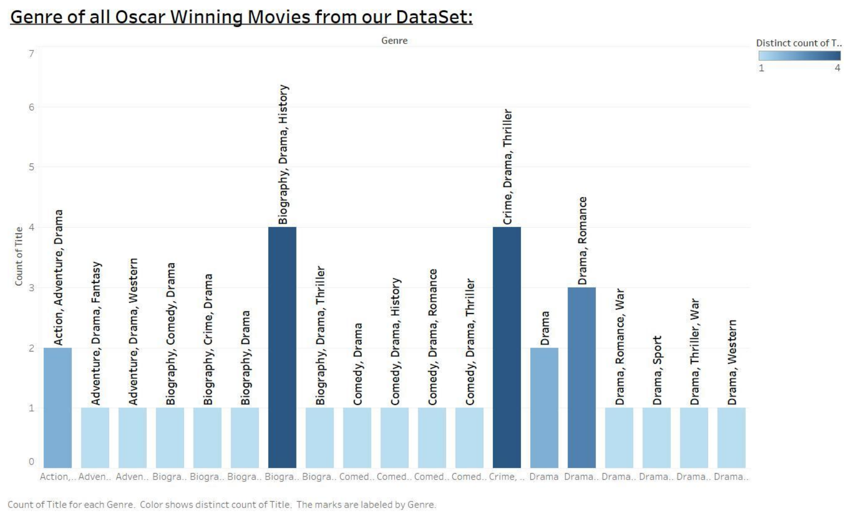
  Here we compared the ratings of the highest rated iMDB movie with the Oscar winner movie in best picture of the same year. We observed that the average ratings of the Oscar winner movies is lower than that of the iMDB movies.
  Interpretation: Based on the public’s reviews posted on the iMDB website, just because a movie wins the Oscar does not inherently mean that it will be the highest rated movie on iMDB that year.



Year	Highest Rated iMDB Movie	Rating	Oscar Winner Movie	Rating
2020	Hamilton	8.4	Nomadland	7.4
2019	Parasite	8.6	Parasite	8.6
2018	Avenger: Infinity War	8.4	Green Book	8.2
2017	Logan	8.1	The Shape of Water	7.3
2016	Dangal	8.4	Moonlight	7.4
2015	Spotlight	8.1	Spotlight	8.1
2014	Interstellar	8.6	Birdman	7.7
2013	The Wolf of Wall Street	8.3	12 years a slave	8.1
2012	The Django Unchained	8.4	Argo	7.7
2011	The Intouchables	8.5	The Artist	7.9
2010	Inception	8.8	The King's Speech	8

- **Genre of Oscar Winning Movies**

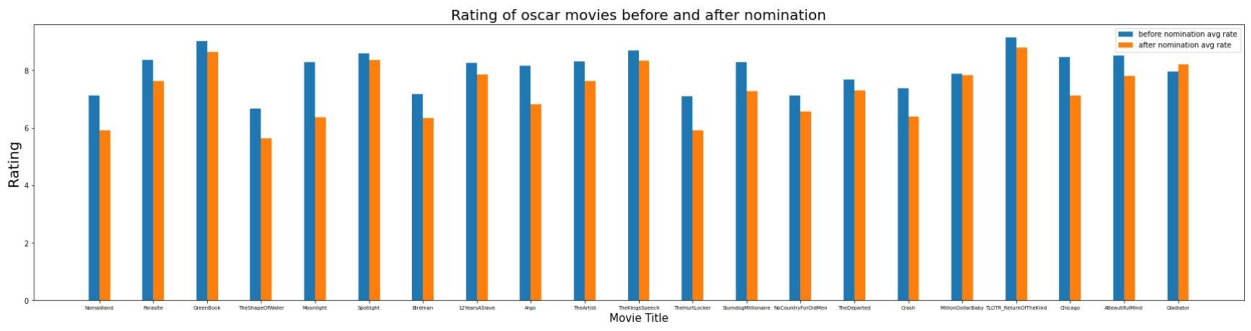
  We also wanted to see which genre of movie most frequently won the Oscar Awards. We analyzed the genres from the dataset after filtering the winners out. We found that movies in the Biography and Drama genre won most of the Oscars.
  Therefore, if we were to make recommendations to a movie production company competing for an Oscar, we would advise them to focus on making movies in these two categories to have the best chance of winning.



- **Rating of Oscar Movies Before and After Oscar Nomination**

  The graph below compares the ratings of Oscar winning movies from the iMDB website before and after receiving the nomination. The blue bar represents the ratings before receiving the nomination, and the orange bar represents the ratings after the nomination. An unusual trend we observed here is the ratings tended to decrease after receiving the nomination.

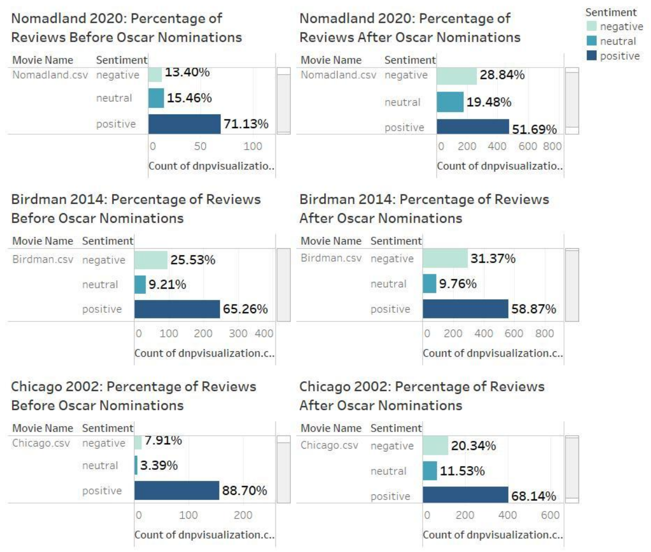
  Interpretation: We believe the ratings decline upon receiving an Oscar nomination can be largely attributed to the increased expectations viewers have. Once the movie is nominated for the Oscar award, people become more critical toward the movie and its shortcomings thus giving a lower rating.



- **Comparison of Sentiment Before and After Oscar Nominations**

  We wanted to make a comparison of the review sentiment before and after the movie's Oscar nominations. We used three movies as examples: Nomadland, Birdman, and Chicago. All of these exhibited an increase in the percentage of “negative” reviews after the nomination per our sentiment analysis .

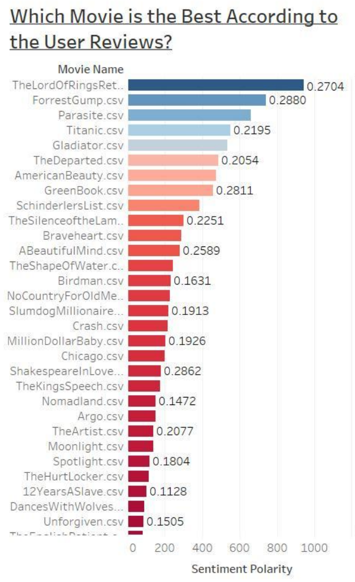
Therefore, we can infer that expectations increase after a movie wins the Oscar Awards for Best Picture. And if the movie does not meet these increased expectations, viewers are more likely to write negative reviews and comments.

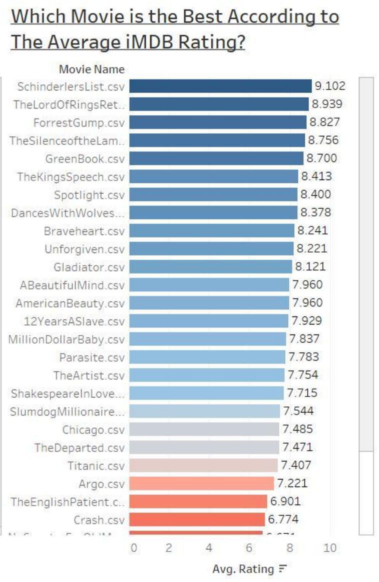


- **Difference Regarding the "Best" Movies in People's Hearts**

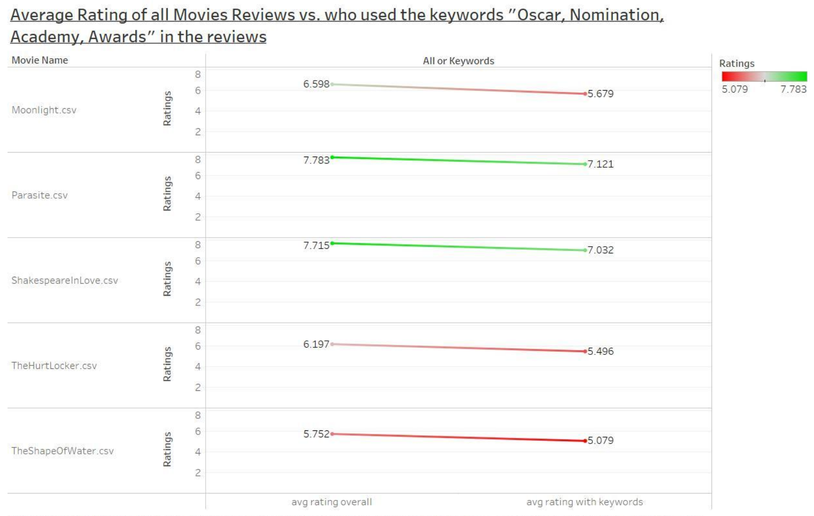
  The left graph shows the ranking of the best movies based on the polarity score we got from TextBlob on the users' reviews. The right graph shows the ranking of the best movies based on iMDB ratings. For example, according to iMDB ratings, the best movie is Schindler's List. However, based on the polarity, the same movie ranks 9th place.

  <u>Interpretation:</u> Through the sentiment analysis we have done, we can see that our results are different from the IMDB rating. According to the polarity score, the highest rated movie on IMDB is ranked 6th on our ranking. Our analysis based on the users' reviews got a different result than the users' ratings. It could be that people don't know how to numerically evaluate how good the movie was to them.





- **Overall Rating vs. Rating of Reviews Nominating Oscar**
  From the graph below we can see the difference of the average users' rating between all the reviews and just the one that nominates the Oscar award for 5 movies. The pattern is similar across all the movies, albeit to a different degree. We can see that the reviews nominating the Oscar have an overall score below the average of all the reviews. From this visualization it's clear that the Oscar nomination is one of the causes of the rating drop.



The trend of sum of Ratings for All or Keywords broken down by Movie Name. Color shows sum of Ratings. The marks are labeled by sum of Ratings. The view is filtered on Movie Name, which keeps Moonlight.csv, Parasite.csv, ShakespeareInLove.csv, TheHurtLocker.csv and TheShapeOfWater.csv.

Takeaways & Conclusion

According to IMDB users, most Oscar winning movies are not the best films of that year.

There could be several reasons for that. In our analysis, we have found prevalently two of them, based mostly on people’s bias in rating:

1. Selection committee has a genre bias toward drama and biography. We have seen that most, if not all, the movies are biographies and dramas. But these two genres might not be popular to all. In fact, from our word clouds and text analysis, we have found out that the negative points of the reviews are the same for the positive ones. Therefore, it is highly probable that it is a genre-mismatch.

2. Users’ expectations after an Oscar nomination increases, therefore ratings after the award nomination are lower. According to our sentiment analysis after Oscar nominations there are, in fact, more negative reviews. Seeing that the movie won an Oscar, the person might think that the movie is really good just because of the award and go watch it, even though the genre maybe doesn’t fit him or the story doesn’t appeal to him.

Bibliography

Stefano L. (2020, January). IMDb movies extensive dataset, Version 2, Retrieved December 4, 2021 from https://www.kaggle.com/stefanoleone992/imdb-extensive-dataset?select=IMDb+movies.csv

Web scraping sources:
https://www.imdb.com/title/tt0099348/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt0102926/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt0105695/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt0108052/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt0109830/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt0112573/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt0116209/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt0120338/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt0138097/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt0169547/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt0172495/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt0268978/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt0299658/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt0167260/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt0405159/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt0375679/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt0407887/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt0477348/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt1010048/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt0887912/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt1504320/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt1655442/reviews/?ref_=tt_ql_urv
https://www.imdb.com/title/tt1024648/reviews/?ref_=tt_ql_urv

https://www.imdb.com/title/tt2024544/reviews/?ref_=tt_ql_urv

https://www.imdb.com/title/tt2562232/reviews/?ref_=tt_ql_urv

https://www.imdb.com/title/tt1895587/reviews/?ref_=tt_ql_urv

https://www.imdb.com/title/tt4975722/reviews/?ref_=tt_ql_urv

https://www.imdb.com/title/tt5580390/reviews/?ref_=tt_ql_urv

https://www.imdb.com/title/tt6966692/reviews/?ref_=tt_ql_urv

https://www.imdb.com/title/tt6751668/reviews/?ref_=tt_ql_urv

https://www.imdb.com/title/tt9770150/reviews/?ref_=tt_ql_urv