Harvard University - CS109B - Data Science II - Spring 2018
Final Project - Milestone 4 - Final Report
Andrew Lund, andrewlund@g.harvard.edu
Nicholas Morgan, nim607@g.harvard.edu
Amay Umradia, amayumradia@gmail.com
Charles Webb, cwebb@college.harvard.edu

Project GitHub repository: https://github.com/Nick-Morgan/109b_final
Canvas Final Group: 11

Movie Genre Multi-label Classification from Plot Descriptions

---

Problem Statement and Motivation

The overarching goal of this project is to build a machine learning pipeline from scratch in order to predict movie genres based on their plot descriptions. To that end, our project has three concrete goals:

1. Build a dataset of 1000 movies by scraping movie information from The Movie Database (TMDB)[1] and the Internet Movie Database (IMDB)[2].
2. Apply bag-of-words with term frequency inverse document frequency (TFIDF), word2vec (w2v), and doc2vec (d2v) transformations to movie plots.
3. Use conventional machine-learning techniques to classify movies as one or more genres using the transformed plots.

Project Question:
"Can we build a robust movie plot and genre dataset through web scraping, then apply bag-of-words and word2vec transformations to plot descriptions to accurately classify movie genres using conventional machine learning multi-label classification techniques, and do longer IMDB or combined IMDB and TMDB plots with potentially richer w2v or d2v representations lead to improved precision and recall metrics over sparser bag-of-words and shorter TMDB plot vector representations?"

Introduction and Description of Data

Data Collection

There are a number of design decisions we made throughout this project. Some are discussed in the "Modeling Approach" section below, and some were made in the data collection described here since one of our project goals was to build a dataset from scratch. For this reason, we will describe our data collection process in this introductory section as well as cover its description.

In order for our team to explore and describe the movie data we used for modeling, we first had to collect it. We scraped our data of 1000 movies from both TMDB and IMDB using the tmdbsimple[3] and IMDb[4] python libraries. One of the first design decisions we made was to only

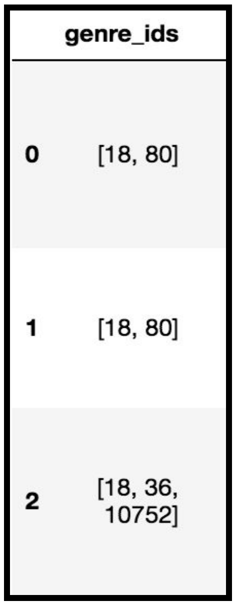
[1] https://www.themoviedb.org/
[2] https://www.imdb.com/
[3] https://pypi.org/project/tmdbsimple/
[4] https://pypi.org/project/IMDb/

collect English-language films, and designate TMDB genres as the overall response variable for this project.

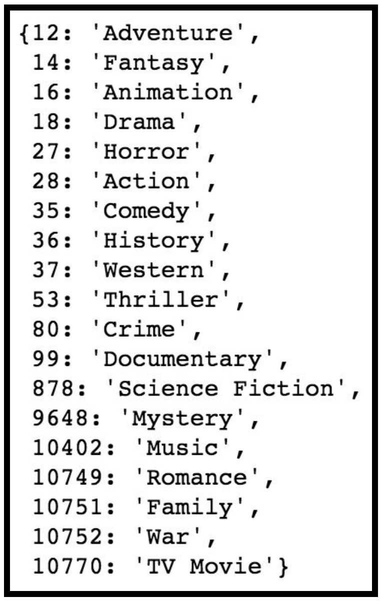
To begin scraping we acquired an API key from TMDB. We found that their API has a 40-requests-per-10-seconds limit, so we had to apply a sleep function to our collection process. One of the convenient movie attributes available with TMDB movies is their accompanying IMDB identification. This made matching movies between the two databases simple, though it did require a separate round of scraping because the IMDB id is not included in the base IMDB movie object.

Since we are primarily interested in predicting genres specifically from plots, another design decision we made was to scrape IMDB plots only, even though IMDB has 58 attributes per movie. Using those other attributes in modeling would be interesting for a follow-on or extension to this project as noted in the conclusions section below. TMDB only includes one plot per film, but IMDB includes a list of plot summaries, so as another design decision we chose to only include the first plot description for each IMDB movie. This will be the top summary listed on the website and should be adequate to compare between the two movie databases when modeling.

To summarize, the data we scraped includes TMDB and IMDB movie IDs, genre labels, titles, plots, popularity metrics (score, vote average, vote count), and release date. The genre labels included in the IMDB movie objects are in the form of numbered lists as seen below We humans know film genres as strings rather than numbers, so we also scraped and saved their associated string representations as the dictionary shown below for modeling analysis.



	genre_ids
0	[18, 80]
1	[18, 80]
2	[18, 36, 10752]

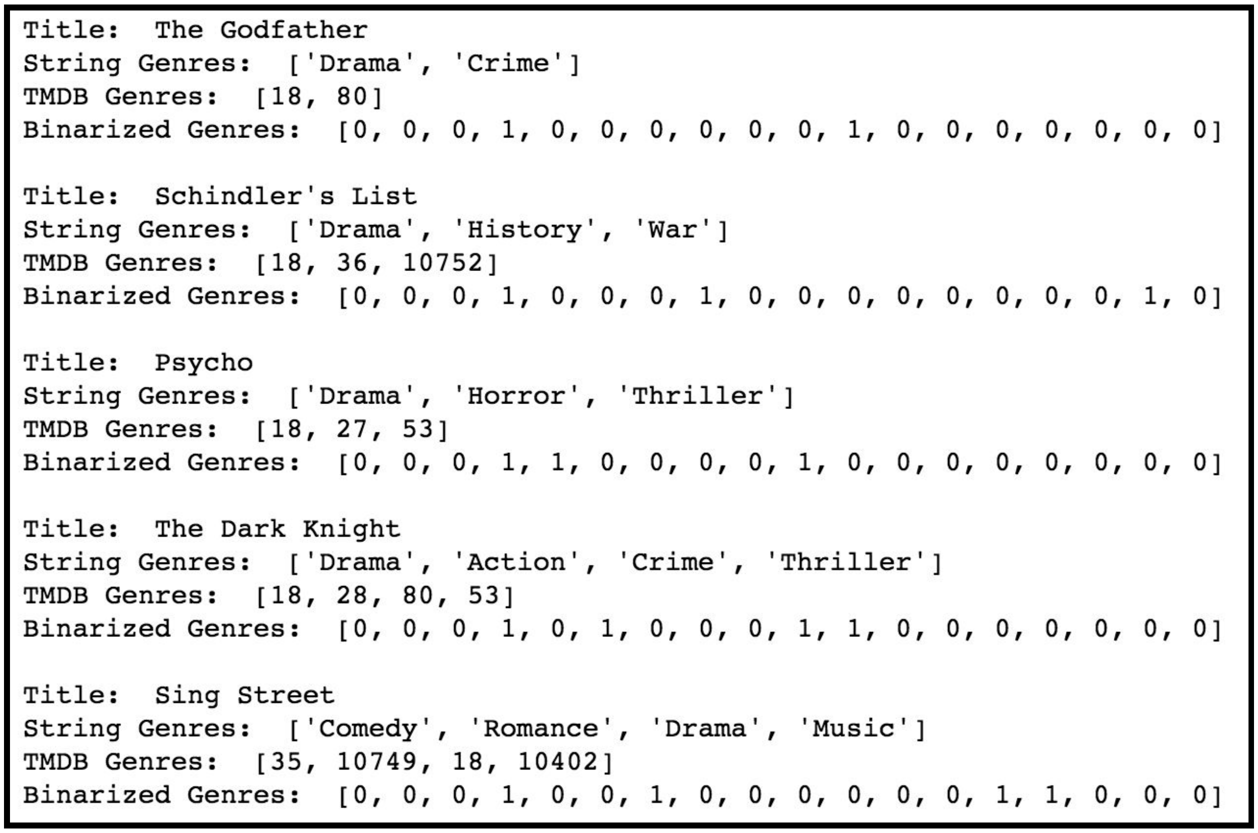


```
{12: 'Adventure',
 14: 'Fantasy',
 16: 'Animation',
 18: 'Drama',
 27: 'Horror',
 28: 'Action',
 35: 'Comedy',
 36: 'History',
 37: 'Western',
 53: 'Thriller',
 80: 'Crime',
 99: 'Documentary',
 878: 'Science Fiction',
 9648: 'Mystery',
 10402: 'Music',
 10749: 'Romance',
 10751: 'Family',
 10752: 'War',
 10770: 'TV Movie'}
```

Our project is most concerned with the content of the plot descriptions, since these are our genre predicting features, but some of the other simple metrics, like popularity and release date, may offer insights into the distributions of our data, as well as answer some inherent questions one might have about the movie data we scraped. These features will be explored in the upcoming EDA section.

Data Cleaning and Predictor Manipulation

In their initially scraped form, both our predictors and response variables are strings. In order to more easily explore the data, as well as apply the multi-label classification modeling and analysis described later, we applied a binary representation to each film's TMDB genre lists. This transformation was done through the use of scikit-learn's MultiLabelBinarizer (MLB).[5] For each list of TMDB genres, the MLB converts it into a numpy array of 0s and 1s. We save those binary genre representations as numpy arrays for future modeling and analysis. An example of the three different genre representation for five films from our dataset is shown below.



```
Title:  The Godfather
String Genres:  ['Drama', 'Crime']
TMDB Genres:  [18, 80]
Binarized Genres:  [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]

Title:  Schindler's List
String Genres:  ['Drama', 'History', 'War']
TMDB Genres:  [18, 36, 10752]
Binarized Genres:  [0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]

Title:  Psycho
String Genres:  ['Drama', 'Horror', 'Thriller']
TMDB Genres:  [18, 27, 53]
Binarized Genres:  [0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]

Title:  The Dark Knight
String Genres:  ['Drama', 'Action', 'Crime', 'Thriller']
TMDB Genres:  [18, 28, 80, 53]
Binarized Genres:  [0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0]

Title:  Sing Street
String Genres:  ['Comedy', 'Romance', 'Drama', 'Music']
TMDB Genres:  [35, 10749, 18, 10402]
Binarized Genres:  [0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0]
```

While the binary genre representation is harder for us to look at compared to the genres strings, it is essential for allowing our models to employ one-vs-rest classification and precision-recall evaluation for each genre for which a movie may be classified. We also cleaned the plots including dropping English stop words, lowering strings, and tokenizing. Finally, we

[5] http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MultiLabelBinarizer.html

transformed the plots into bag-of-words TFIDF vectors, w2v matrices and vectors, and d2v vectors.

Bag-of-words and TFIDF Plot Transformations

The bag-of-words TFIDF vector representations are essentially rarity scores for each word in a movie's plot. In other words, each plot's word is scored based on how important that word is within a given plot compared to the corpus of plots the plot belongs to.[6] The underlying TFIDF algorithm is shown in the image below.

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF
Term x within document y

$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents [7]

For each word in a document, count its frequency within that document, then multiply it by the log of the total number of documents divided by the number of documents containing that word.

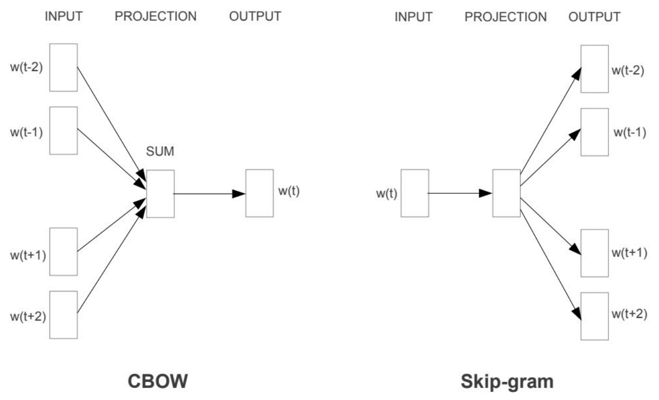
For the sake of our TFIDF transformations, each plot's corpus of plots is the set of 1000 plots for the TMDB, IMDB, and combined plot lists, respectively. We pass the sets of plots through scikit-learn's TfidfVectorizer.[8] This function removed both very common and very rare words through the max/min_df arguments, as well as dropping English stop words. We set max_df and min_df to ignore words specific to our corpus of movie plots that occur both a lot and few times. This reduces the dimensionality of our already sparse TFIDF vector representation of each plot. By dropping some words as previously noted, the resulting vectors are a manageable 1x1171 for each TMDB plot, 1x2442 for each IMDB plot, and 1x3025 for the combined plots. These vectors are inherently sparse. Most of the array values in each plot vector are zero since each movie has an average of about 50 and 100 words for each TMDB and IMDB plot summary, respectively.

Word2vec Plot Transformations

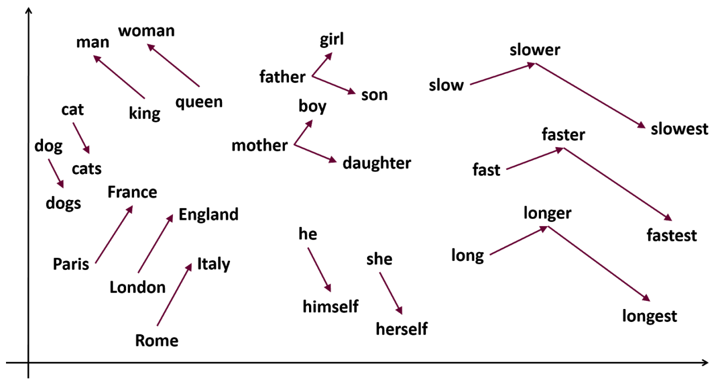
Word2vec models are based upon a feed forward neural net language model (NNLM). Like the NNLM, the word2vec model transforms each word into a vector of values in its first layer. Both models can be trained either to predict a target word based upon the words surrounding it, or predict the words surrounding a target word based solely upon that word. Unlike the NNLM

[6] http://www.tfidf.com/
[7] http://filotechnologia.blogspot.com/2014/01/a-simple-java-class-for-tfidf-scoring.html
[8] http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

model, however, the word2vec model returns the vector of values that it creates in its first layer. Below is an image of the two Word2Vec architectures proposed by Mikolov et. al., continuous bag of words (CBOW) and Skip-gram. One significant result of this model is that the resulting vectors can be manipulated to demonstrate analogy. For example, in the equation X = *vector*('biggest') - *vector*('big') + *vector*('small'), X = *vector*('smallest').[9]



The w2v plot matrices and vectors we made were created using Google's Google News w2v model. This model which was trained on a more than 100-billion-word news corpus. It contains three-million words, each represented by a 300-dimension vector.[10] We decided that, for each plot, if a word in the cleaned plot (lowered, punctuation and stop words dropped, tokenized) is in the w2v model, add it to a running list for that plot, and if not, skip it. We collected those skipped words, and a cursory exploration shows us most skipped words are words like years or numbers and proper nouns (500, 1940s, dunkirk, krueger, fleetwood, etc). The following image does an excellent job of illustrating a toy example in two-dimensions of how word vectors relate to each other in a model like the Google News one we employed in our plot transformations. The thought for our project being that the semantic quality of each word will "point" in the direction of one or more genres when we train our multi-label classifier models.



[11]

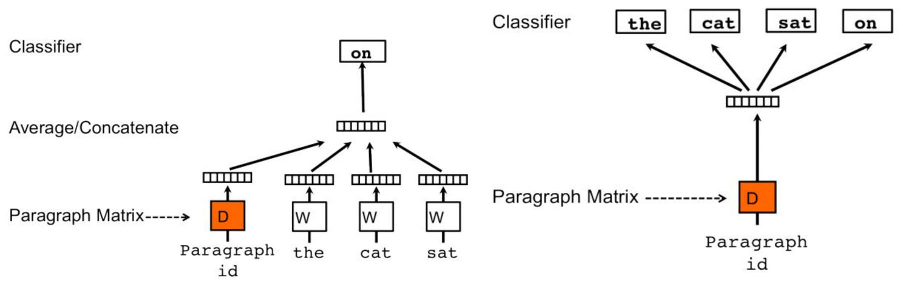
[9] https://arxiv.org/pdf/1301.3781.pdf
[10] https://code.google.com/archive/p/word2vec/
[11] http://www.samyzaf.com/ML/nlp/nlp.html

We saved each plot summary as both a full w2v matrix, where each row represents a 300-dimension word in its plot, as well as the column-wise mean of each matrix saved as a 300-dimension average vector of the plot's words.

We hypothesized that the word2vec representation of each plot might be richer for some movies since each plot is a 300-dimension vector and most TFIDF vectors will be far sparser leading to greater classification precision and recall. Though the w2v vectors are means of plots' words, rather than the TFIDF score of each word in the plot, as represented in our bag-of-words transformation, some genre-specific words may lose their semantic quality through the mean calculation. The assumption we made by taking the mean of each plot is that the resulting 300-dimension vector will still point in the direction of one or more genres, and interestingly we found in our modeling that we achieved nearly identical precision and recall results for the matrix versus vector w2v plot representations.

Doc2vec Plot Transformations

We also made another predictor by creating document vectors using the gensim python package's Doc2Vec function.[12] The d2v model is very similar to the w2v model, except that each plot is treated as a single word like in the w2v model. Below is an image of d2v models proposed by Mikolov et. al. with the distributed memory model on the left and the distributed bag of words model on the right.



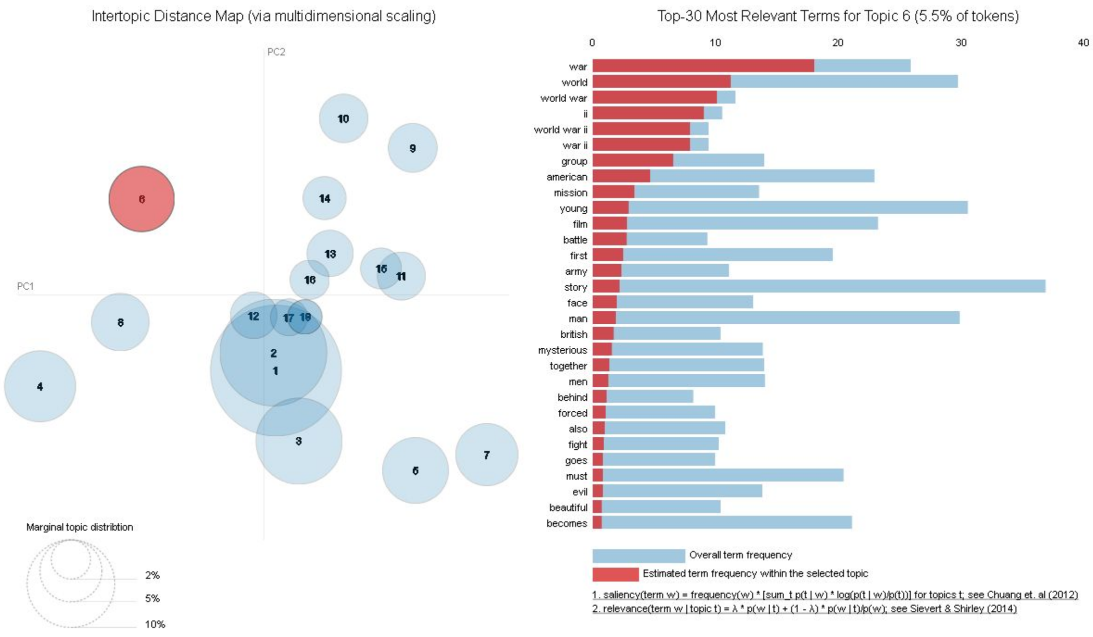
The major difference between the d2v transformations and the w2v transformations is that we were not able to use a pre-trained model like Google's, so we had to create our own models for each set of plots and train them on our dataset. This presented an issue, because our dataset is relatively small compared to other successful, and larger, experiments with d2v transformations, so we had poor d2v model performance as outlined in the results section below.[13]

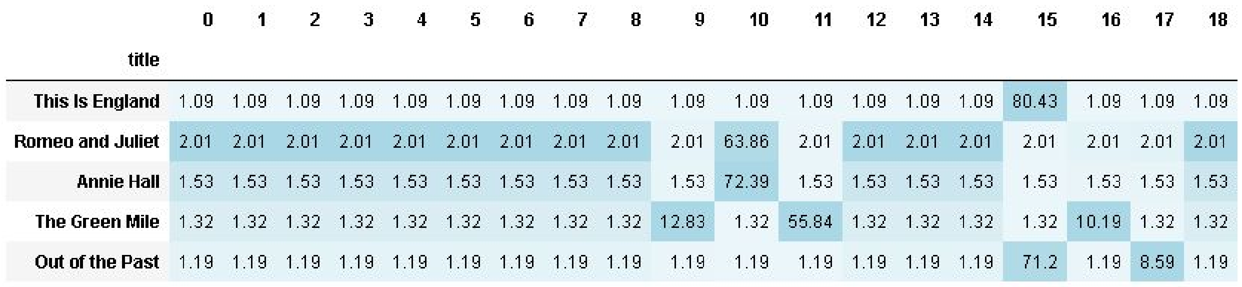
[12] https://radimrehurek.com/gensim/models/doc2vec.html
[13] https://cs.stanford.edu/~quocle/paragraph_vector.pdf (doc2vec proposal)

For our d2v model we chose to use a document vector size of 19, because we were classifying 19 different genres. We hoped that the specifically tailored size of the document vectors would help to make up for the lack of training data.

Latent Dirichlet Allocation (LDA)

We also considered using LDA for our classification models. LDA is a natural language processing technique which attempts to place documents (plots) into groups or clusters (genres). [14] Using scikit-learn's LatentDirichiletAllocation model,[15] we saw interesting results when specifying a discrete number of genres for our dataset, however, a major drawback was that most movies were classified as one genre with high probability and equal probability for all others as seen in the below dataframe. This most likely occurred because many of the 19 genres overlap. Below are some graphics demonstrating these overlapping clusters. The clusters show that a few clusters at the center are very similar, while others are easily separable. For example, it is clear from top 30 words of topic 4 that this topic represents the "War" genre.





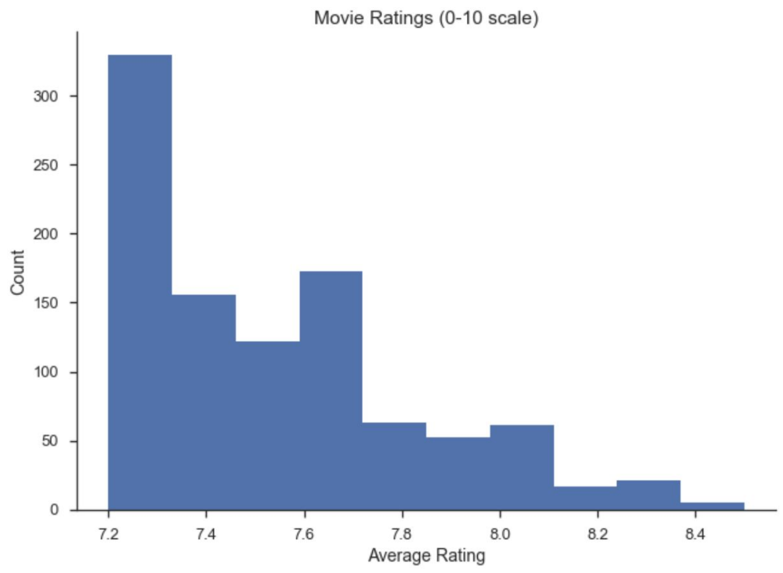
title	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
This Is England	1.09	1.09	1.09	1.09	1.09	1.09	1.09	1.09	1.09	1.09	1.09	1.09	1.09	1.09	1.09	80.43	1.09	1.09	1.09
Romeo and Juliet	2.01	2.01	2.01	2.01	2.01	2.01	2.01	2.01	2.01	2.01	63.86	2.01	2.01	2.01	2.01	2.01	2.01	2.01	2.01
Annie Hall	1.53	1.53	1.53	1.53	1.53	1.53	1.53	1.53	1.53	1.53	72.39	1.53	1.53	1.53	1.53	1.53	1.53	1.53	1.53
The Green Mile	1.32	1.32	1.32	1.32	1.32	1.32	1.32	1.32	1.32	12.83	1.32	55.84	1.32	1.32	1.32	1.32	10.19	1.32	1.32
Out of the Past	1.19	1.19	1.19	1.19	1.19	1.19	1.19	1.19	1.19	1.19	1.19	1.19	1.19	1.19	1.19	71.2	1.19	8.59	1.19

[14] https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation
[15] http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html

The overlap of clusters could demonstrate the value of the semantic data in each plot. If we found the optimal number of topics for our dataset and named each topic that contains several genres as to reflect that (e.g. “drama/mystery”), LDA would be useful for single-class classification. Since our project focuses on multi-label classification, we did not ultimately use LDA in our final modeling and analysis. The visualizations above were made using the pyLDAvis python library.[16] A notebook outlining our LDA implementation is available in our project’s GitHub repository as “future_work_LDA.ipynb.”[17]

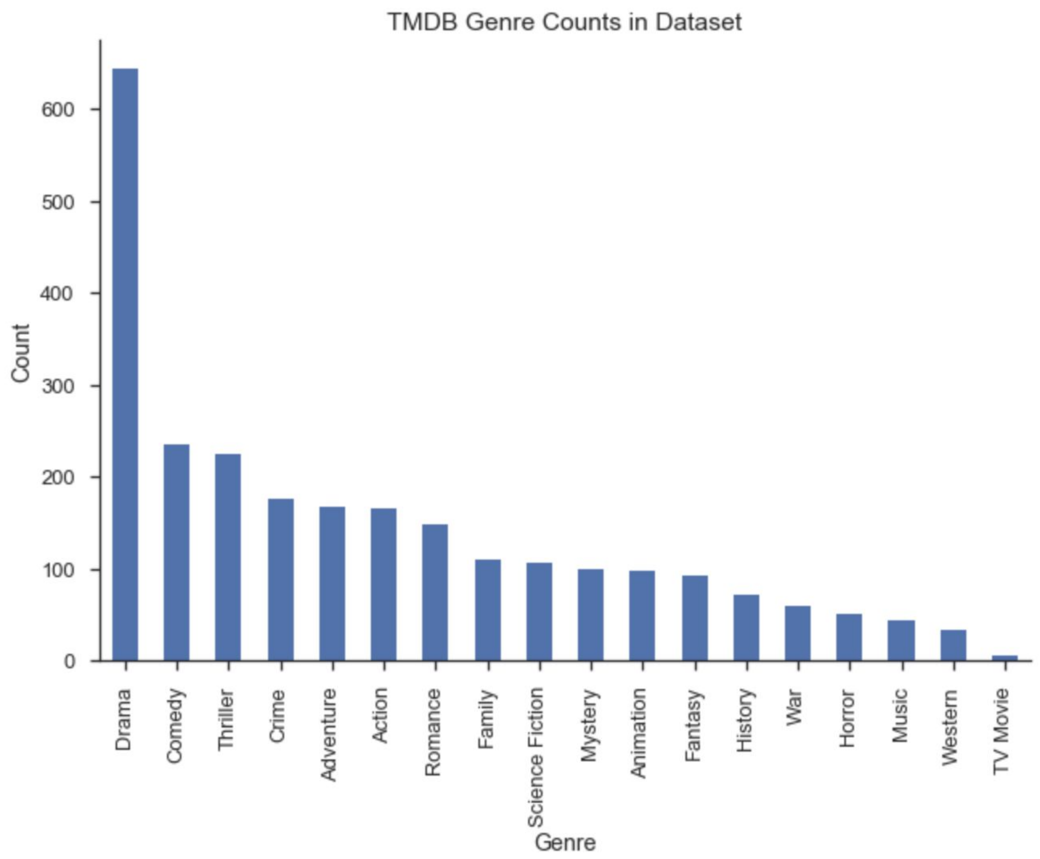
Exploratory Data Analysis

We produced a number of visualizations using the matplotlib[18] and seaborn[19] python visualization libraries to explore our scraped movie dataset. Below we see the distribution of movie ratings on a 1-10 scale. Most of our movies are highly-rated in the 7.2 to 8.4 range. This might lead to more descriptive and rich plots since the movies recieve more attention from both critics and fans.

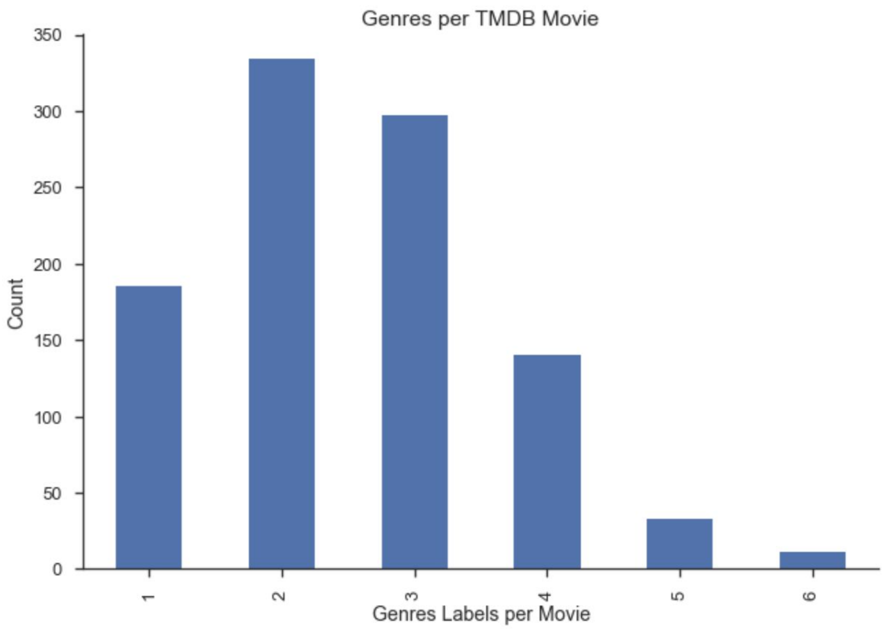


The next plot shows the distribution of TMDB genres for the entire dataset. Of our 1000 movies, more than 600 of them have drama as one of their classifications. This is not surprising, as many movies, be they comedies, thrillers, or action/adventure films, can easily be considered dramas as well, but it ultimately leaves our data skewed.

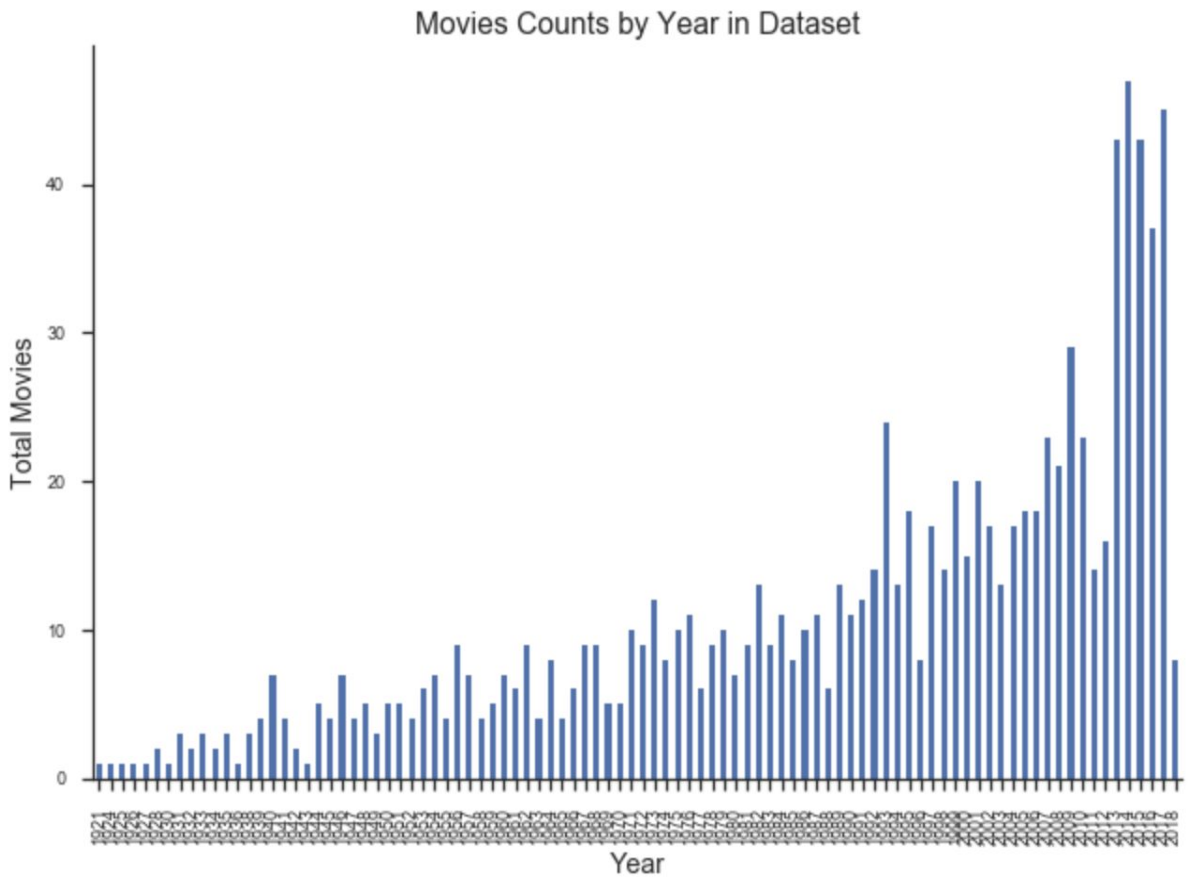
[16] http://pyldavis.readthedocs.io/en/latest/
[17] https://github.com/Nick-Morgan/109b_final
[18] https://matplotlib.org/
[19] https://seaborn.pydata.org/



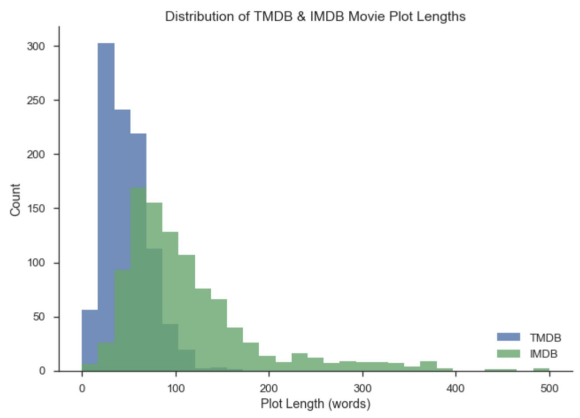
A reasonable follow-up to the overall genre distribution would be, "How many genres is each movie classified as?" Below you can see that distribution with an average of 2.5 genres per film and six at most. This is an important consideration for us to use multi-label classification in our modeling, rather than designing models that classify each film as having only one genre.

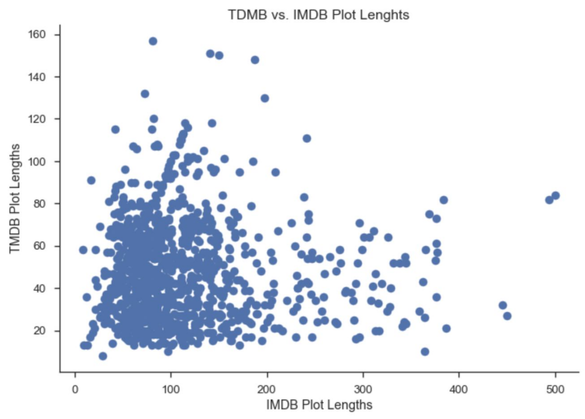


We also looked at the distribution of film release years for our dataset, from 1921 to 2018. It is not surprising to see more younger films since as the film industry expands, so do the numbers of films being produced. This distribution will not play an integral role in any modeling since a plot for a popular older movie should have the same descriptive words as a younger movie regardless of release year, but it is an interesting look at our dataset nonetheless.

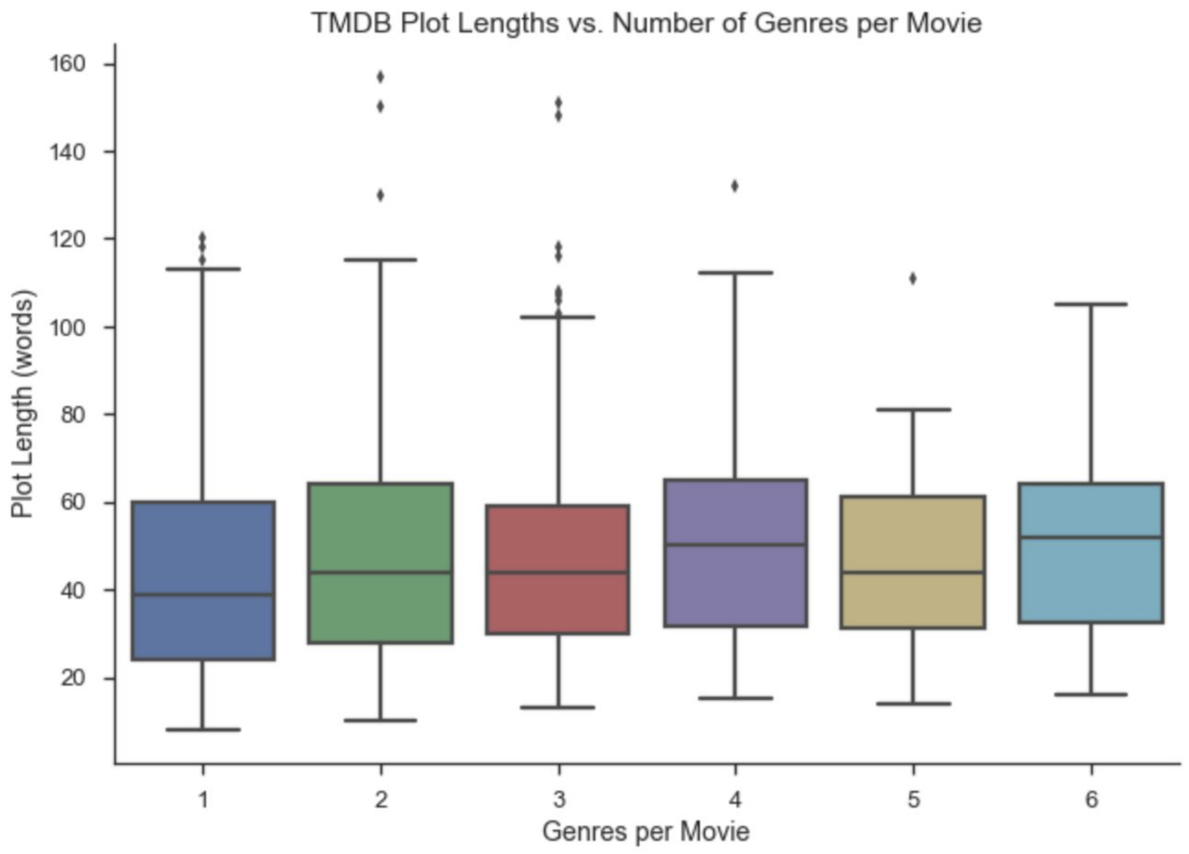


Another interesting look at our scraped data is the word count of their respective TMDB and IMDB plots. As seen in the following two visualizations, TMDB plots are generally much shorter than IMDB plots, with average word counts of about 48 and 112, respectively. The scatter plot to the right shows a one-to-one comparison of films and illustrates this trend - longer IMDB versus TMDB plots for a given movie - nicely. This relationship leads us to hypothesize that IMDB plots may have more descriptors associated with specific genres, and may lead to increased genre classification precision and recall. We further hypothesis that combining the plots will have a net positive effect on precision and recall metrics.





Lastly, we took a different look at the plot word counts distribution. The side-by-side boxplots below show the distributions of plot word counts compared by genre labels per movie. Each genre count boxplot shows similar distributions with means mostly matching and a few longer outliers. This leads us to believe that the descriptiveness of a movie plot is not dependant on the number of genres assigned to it.



Literature Review and Related Work

Aside from the sources found throughout this report, the following are also notable for our overall project understanding, data scraping and preparation, modeling, and analysis.

1. Spandan Madan's 2017 tutorial, "An end to end implementation of a Machine Learning pipeline," was used for inspiration and design ideas for this project: https://spandan-madan.github.io/DeepLearningProject/

2. Mikolov et al propose two word2vec models in the 2013 paper "Efficient Estimation of Word Representations in Vector Space," https://arxiv.org/pdf/1301.3781.pdf

3. Minmin Chen's 2017 doc2vec paper, "Efficient Vector Representation for Documents through Corruption," https://arxiv.org/pdf/1707.02377.pdf

4. Chih-Wei et al's 2016 paper, "A Practical Guide to Support Vector Machines, Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin," https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

5. A good discussion for why we need to scale our w2v and d2v from 0-1, due to Pearson's chi-squared test not being able to handle negative values, https://stackoverflow.com/questions/25792012/feature-selection-using-scikit-learn

6. A Google Groups discussion of the Google News w2v model and associated modeling techniques, https://groups.google.com/forum/#!topic/gensim/_XLEbmoqVCg

---

Modeling Approach

As previously noted, most of the movies in our dataset have more than one genre label. For this reason, we have defined our project as a multi-label classification problem. In other words, we need to employ modeling techniques which allow each film to be classified as a set of genres rather than just one. To this end, we employ one-vs-rest classification, where each class (genre) has a classifier fitted to it, and allows us to easily interpret how each model fits to each genre in our dataset with specific metrics like precision, recall, and F1 score.[20]

[20] http://scikit-learn.org/stable/modules/multiclass.html

As previously noted, there are a number of design choices we made throughout this project from building the dataset, to choice of models. We transformed our dataset by representing the plot of each movie as both a bag-of-words TFIDF vector, w2v matrices and vectors, and d2v vectors. Since we saw almost identical results using the w2v matrices, those have not been included in our final modeling and analysis. Other design choices used in our modeling are described in the following sections.

Because the underlying goal of our project was to create a machine learning pipeline from scratch, we do not have a "baseline" model. Rather, we three three conventional machine learning multi-label classification models on nine different transformations of the movie plots.

Function-based Modeling Pipeline

Using the individual TMDB and IMDB plots, as well as combined plots, we have nine predicting features to train our three multi-label classification models on. This ultimately gives us 27 sets of results to choose the best precision and recall metrics from.

In order to efficiently iterate through each feature and model, we built a function "evaluate_model" that splits the data into an 80% training and 20% test set (we also trained on a 50/50 split, but saw better results with 80/20). We set a random seed in order to ensure our models are training and predicting on the same training and test sets. The inputs to the function are a model object, predictor set, response set, cross-validation argument, and model-specific parameters to tune via cross-validation. The model outputs a dictionary which includes average precision and recall across each class as well as a full classification report for each genre in the dataset.

Before passing our features through the modeling function we had to apply two more modifications of the w2v and d2v arrays. They had to be converted to an array of lists from an array of arrays in order to be compatible with the format of the response variable, and we had to standardize their values between 0 and 1. We used scikit-learn's MinMaxScaler[21] to conduct the scaling.

After the data is prepared, we create a dictionary of models and predictors, then iterate through them saving a dictionary of results for further analysis found in the results section below.

[21] http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

Multi-label Classification Models

The three models we used for each predicting feature were Naive-Bayes with a cross-validated smoothing parameter (alpha), support vector machines (SVM) with cross-validated penalty parameter of the error term (C), and SVM with stochastic gradient descent (SGD) and a cross-validated regularization multiplier (alpha). Each of these models were implemented using scikit-learn's out-of-the-box model classes.[22]

The Naive-Bayes classifier is a probabilistic classifier based on Bayes' theorem which describes the probability of an event given prior knowledge of the conditions which might be related to the event.[23] In the case of our multi-label classifier, it assigns genres to films based on feature values (TFIDF scores or w2v/d2c vectors) where each feature is independent of any other feature values.[24] For example, the movie "The Matrix" might be classified as sci-fi, adventure, and mystery, regardless of the correlations between those genres. By tuning the smoothing parameter through cross-validation it the model the ability to deal with words it may not have seen previously so that posterior probabilities do not drop to zero.[25]

SVMs are a common machine learning classification tool for separating observations into classes by setting a boundary between classes (movie genres) that maximizes their separations.[26] We use a linear kernel for all our classification models and tune the penalty parameter (C) via cross-validation. The penalty parameter tells the SVM how the amount we want to avoid misclassifying genres, making the margin of error larger for small Cs and smaller for large Cs.[27]

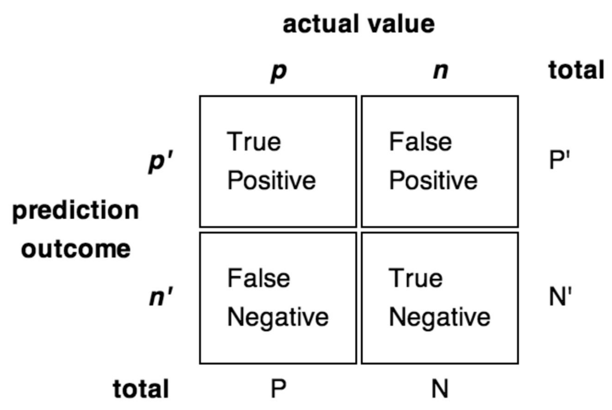
We also trained SVMs using stochastic gradient descent learning. This model fits a linear SVM, then uses the gradient of the loss for each sample as it is fit to update the model. We use the "hinge" loss function which is used for maximum-margin classification,[28] the "L2" penalty function, and tune the regularization multiplier (alpha) through cross-validation.

[22] http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html; http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html; http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
[23] https://en.wikipedia.org/wiki/Bayes%27_theorem
[24] https://en.wikipedia.org/wiki/Naive_Bayes_classifier
[25] https://stats.stackexchange.com/questions/108797/in-naive-bayes-why-bother-with-laplacian-smoothing-when-we-have-unknown-words-i
[26] https://en.wikipedia.org/wiki/Support_vector_machine
[27] https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel
[28] https://en.wikipedia.org/wiki/Hinge_loss

Precision, Recall, F1 Metrics & Classification Reports

We use precision and recall metrics to measure the quality of our fitted models. We calculate these metrics on both the training and test sets, but used the test set results to choose our best classifiers.

Precision and recall are useful when scoring imbalanced datasets like ours where we have one or more classes dominating (dramas).[29] Precision is defined as the number of true positives divided by the sum of true positives and false positives. Recall is the number of true positives divided by the sum of the true positives and false negatives. Precision and recall can also be described through a metric called the F1 score, which is the harmonic mean of precision and recall for a specific genre. The following image and equations outline these three metrics.[30]



		actual value		
		p	**n**	**total**
prediction outcome	**p'**	True Positive	False Positive	P'
	n'	False Negative	True Negative	N'
total		P	N	

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

High precision is a measure of low false positives, and high recall is a measure of low false negatives. A model that has high recall but low precision returns a lot of mostly incorrect labels, while a model with high precision with low recall returns fewer results, but most of them are correct when compared to the their training labels. An ideal model will have both high precision and recall metrics.

We also calculated the average precision and recall across all genres for each model, and use those averages to choose the best overall models before looking more closely at genre-specific classification reports.

[29] http://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html
[30] https://www.safaribooksonline.com/library/view/python-data-analysis/9781785282287/ch10s03.html; https://alliance.seas.upenn.edu/~cis520/dynamic/2017/wiki/index.php?n=Lectures.PrecisionRecall; https://en.wikipedia.org/wiki/Precision_and_recall

Project Trajectory, Results, and Interpretation

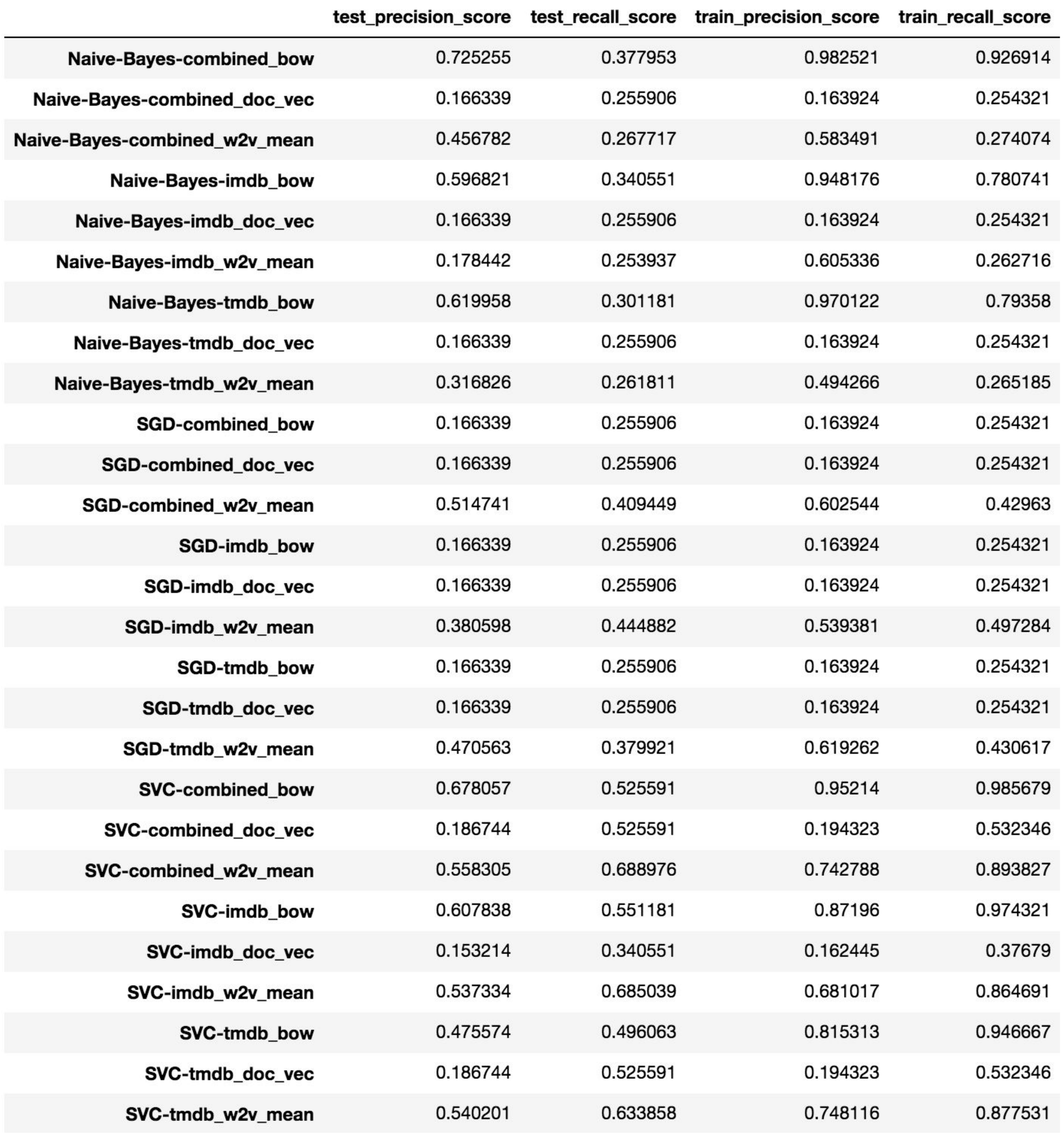
Our overall project goals remained fairly steady throughout the implementation since they were both broad and concrete. We built a dataset from scratch by scraping web-based movie databases, applied bag-of-words TFIDF, word2vec and doc2vec transformations to the plots, and used conventional machine learning multi-label classification methods to predict movie genres based on their plots. One minor change to our modeling pipeline was applying an 80/20 train/test split vice 50/50 due to better precision recall metrics after training the models.

We assumed the w2v matrix representation of each plot, with each word representing a 300-dimension row of a given matrix would result in more semantic quality and predicting power of certain words. We were surprised to see that the matrix and mean-vector w2v plot representations resulting in nearly identical precision and recall scores for TMDB, IMDB, and combined plot representations. Due to their large size and training time, we decided to eliminate them for our modeling pipeline and just use the three different mean-vector representations as predictors. Ultimately, we trained 27 separate models across the three previously described modeling methods. The nine plot features we trained our models on were:

1. Bag-of-words TFIDF plot vectors using the TMDB plot
2. Bag-of-words TFIDF plot vectors using the IMDB plot
3. Bag-of-words TFIDF plot vectors using the concatenated plots from both sources
4. Word2vec mean plot vector using the TMDB plot
5. Word2vec mean plot vector using the IMDB plot
6. Word2vec mean plot vector using the concatenated plots from both sources
7. Doc2vec plot vectors using the TMDB plot
8. Doc2vec plot vectors using the IMDB plot
9. Doc2vec plot vectors using the concatenated plots from both sources

Overall Results

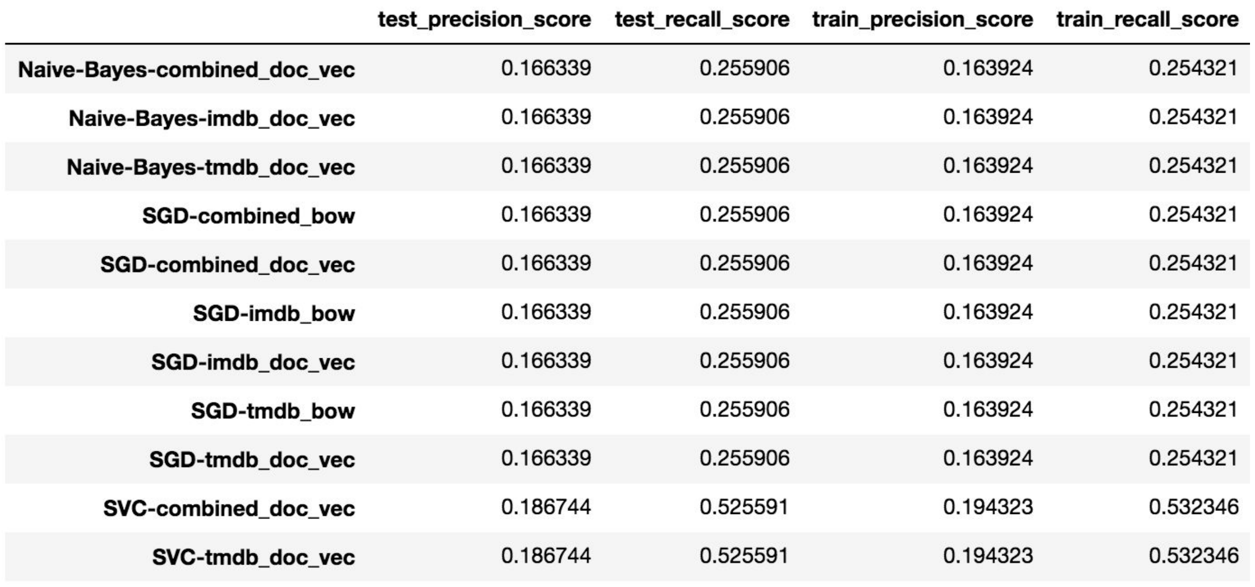
The overall results (average train and test precision and recall) for our 27 models are as follows:



	test_precision_score	test_recall_score	train_precision_score	train_recall_score
Naive-Bayes-combined_bow	0.725255	0.377953	0.982521	0.926914
Naive-Bayes-combined_doc_vec	0.166339	0.255906	0.163924	0.254321
Naive-Bayes-combined_w2v_mean	0.456782	0.267717	0.583491	0.274074
Naive-Bayes-imdb_bow	0.596821	0.340551	0.948176	0.780741
Naive-Bayes-imdb_doc_vec	0.166339	0.255906	0.163924	0.254321
Naive-Bayes-imdb_w2v_mean	0.178442	0.253937	0.605336	0.262716
Naive-Bayes-tmdb_bow	0.619958	0.301181	0.970122	0.79358
Naive-Bayes-tmdb_doc_vec	0.166339	0.255906	0.163924	0.254321
Naive-Bayes-tmdb_w2v_mean	0.316826	0.261811	0.494266	0.265185
SGD-combined_bow	0.166339	0.255906	0.163924	0.254321
SGD-combined_doc_vec	0.166339	0.255906	0.163924	0.254321
SGD-combined_w2v_mean	0.514741	0.409449	0.602544	0.42963
SGD-imdb_bow	0.166339	0.255906	0.163924	0.254321
SGD-imdb_doc_vec	0.166339	0.255906	0.163924	0.254321
SGD-imdb_w2v_mean	0.380598	0.444882	0.539381	0.497284
SGD-tmdb_bow	0.166339	0.255906	0.163924	0.254321
SGD-tmdb_doc_vec	0.166339	0.255906	0.163924	0.254321
SGD-tmdb_w2v_mean	0.470563	0.379921	0.619262	0.430617
SVC-combined_bow	0.678057	0.525591	0.95214	0.985679
SVC-combined_doc_vec	0.186744	0.525591	0.194323	0.532346
SVC-combined_w2v_mean	0.558305	0.688976	0.742788	0.893827
SVC-imdb_bow	0.607838	0.551181	0.87196	0.974321
SVC-imdb_doc_vec	0.153214	0.340551	0.162445	0.37679
SVC-imdb_w2v_mean	0.537334	0.685039	0.681017	0.864691
SVC-tmdb_bow	0.475574	0.496063	0.815313	0.946667
SVC-tmdb_doc_vec	0.186744	0.525591	0.194323	0.532346
SVC-tmdb_w2v_mean	0.540201	0.633858	0.748116	0.877531

Interpreted Results

If you look closely at the results above, there are a 11 models that have matching precision and recall in two groups. These are consolidated below.

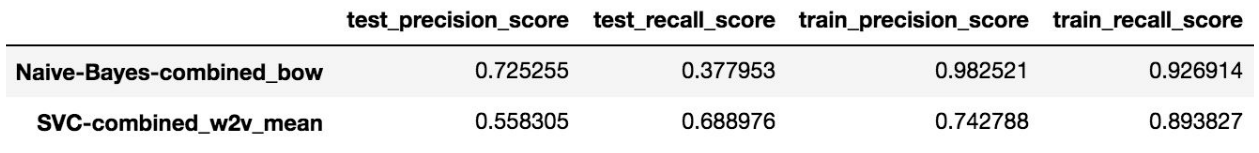


	test_precision_score	test_recall_score	train_precision_score	train_recall_score
Naive-Bayes-combined_doc_vec	0.166339	0.255906	0.163924	0.254321
Naive-Bayes-imdb_doc_vec	0.166339	0.255906	0.163924	0.254321
Naive-Bayes-tmdb_doc_vec	0.166339	0.255906	0.163924	0.254321
SGD-combined_bow	0.166339	0.255906	0.163924	0.254321
SGD-combined_doc_vec	0.166339	0.255906	0.163924	0.254321
SGD-imdb_bow	0.166339	0.255906	0.163924	0.254321
SGD-imdb_doc_vec	0.166339	0.255906	0.163924	0.254321
SGD-tmdb_bow	0.166339	0.255906	0.163924	0.254321
SGD-tmdb_doc_vec	0.166339	0.255906	0.163924	0.254321
SVC-combined_doc_vec	0.186744	0.525591	0.194323	0.532346
SVC-tmdb_doc_vec	0.186744	0.525591	0.194323	0.532346

One group occurs in nine models, and the other occurs in two. As previously noted, our dataset skews toward dramas. Each of these models with matching results are being overfit to the drama genre. For both the train and test sets, each of these models is predicting that 100% of the movies will be dramas. Fine-tuning the model penalization parameters may help with this overfitting, but that exploration is outside the scope of this analysis. We will instead focus on the other models that are not overfit.

Best Results

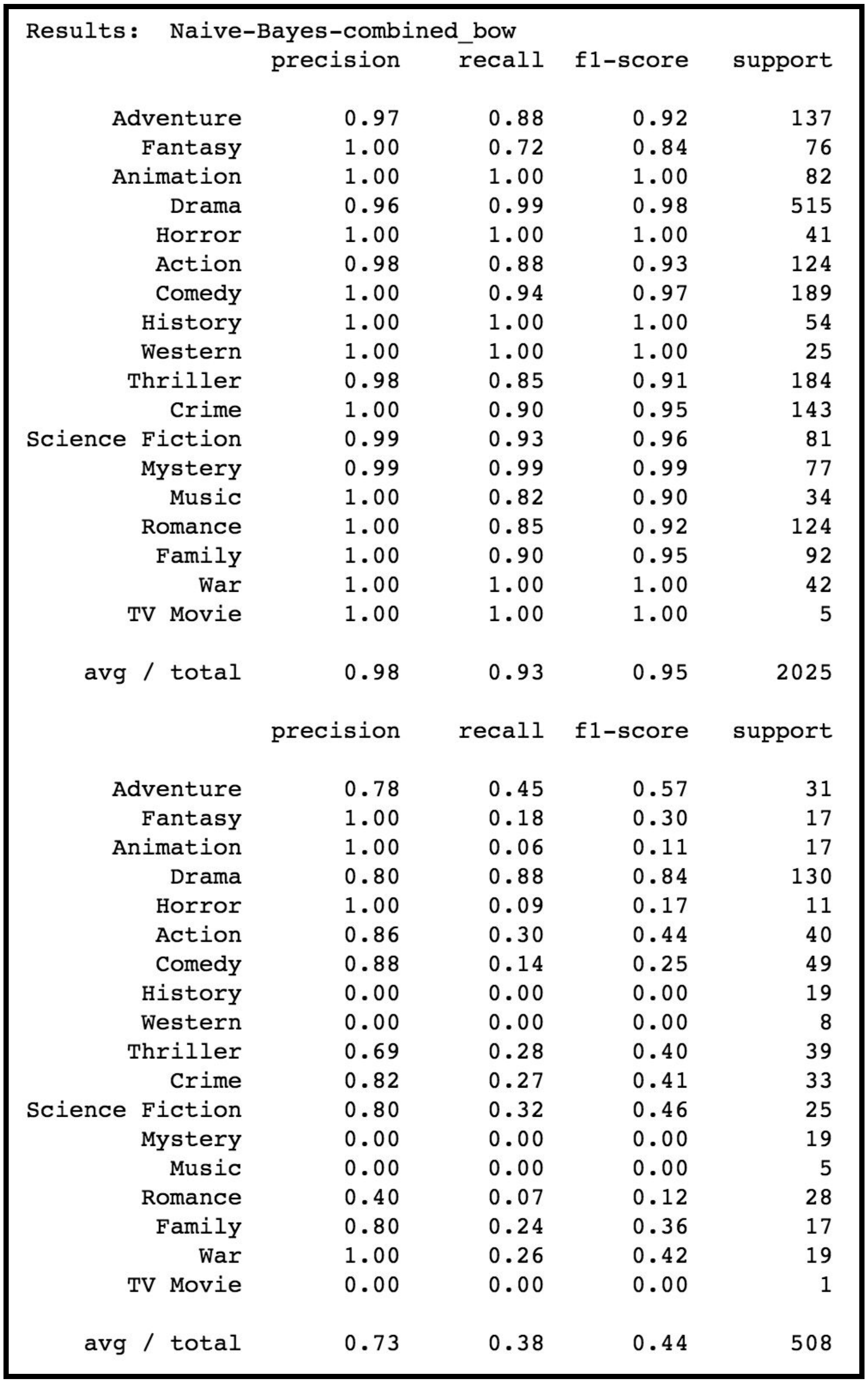
Our project's measure of best results are both the highest mean precision and highest mean recall on the test set. Those two models are shown below.



	test_precision_score	test_recall_score	train_precision_score	train_recall_score
Naive-Bayes-combined_bow	0.725255	0.377953	0.982521	0.926914
SVC-combined_w2v_mean	0.558305	0.688976	0.742788	0.893827

We see that the best precision model is Naive-Bayes with the combined bag-of-words TFIDF plot vectors, and the best accuracy model is SVC with the combined w2v plot mean vectors. You will notice that both of them use the combined plot representations, supporting our hypothesis that the combined (concatenated) plots offer richer and more descriptive representations of the movies. We now take a closer look at the complete classification report for each genre in the best models.

First we look at the train and test classification results (top and bottom, respectively) for the best test precision Naive-Bayes classifier.



```
Results:  Naive-Bayes-combined_bow
                 precision    recall  f1-score   support

      Adventure       0.97      0.88      0.92       137
        Fantasy       1.00      0.72      0.84        76
      Animation       1.00      1.00      1.00        82
          Drama       0.96      0.99      0.98       515
         Horror       1.00      1.00      1.00        41
         Action       0.98      0.88      0.93       124
         Comedy       1.00      0.94      0.97       189
        History       1.00      1.00      1.00        54
        Western       1.00      1.00      1.00        25
       Thriller       0.98      0.85      0.91       184
          Crime       1.00      0.90      0.95       143
Science Fiction       0.99      0.93      0.96        81
        Mystery       0.99      0.99      0.99        77
          Music       1.00      0.82      0.90        34
        Romance       1.00      0.85      0.92       124
         Family       1.00      0.90      0.95        92
            War       1.00      1.00      1.00        42
       TV Movie       1.00      1.00      1.00         5

    avg / total       0.98      0.93      0.95      2025

                 precision    recall  f1-score   support

      Adventure       0.78      0.45      0.57        31
        Fantasy       1.00      0.18      0.30        17
      Animation       1.00      0.06      0.11        17
          Drama       0.80      0.88      0.84       130
         Horror       1.00      0.09      0.17        11
         Action       0.86      0.30      0.44        40
         Comedy       0.88      0.14      0.25        49
        History       0.00      0.00      0.00        19
        Western       0.00      0.00      0.00         8
       Thriller       0.69      0.28      0.40        39
          Crime       0.82      0.27      0.41        33
Science Fiction       0.80      0.32      0.46        25
        Mystery       0.00      0.00      0.00        19
          Music       0.00      0.00      0.00         5
        Romance       0.40      0.07      0.12        28
         Family       0.80      0.24      0.36        17
            War       1.00      0.26      0.42        19
       TV Movie       0.00      0.00      0.00         1

    avg / total       0.73      0.38      0.44       508
```

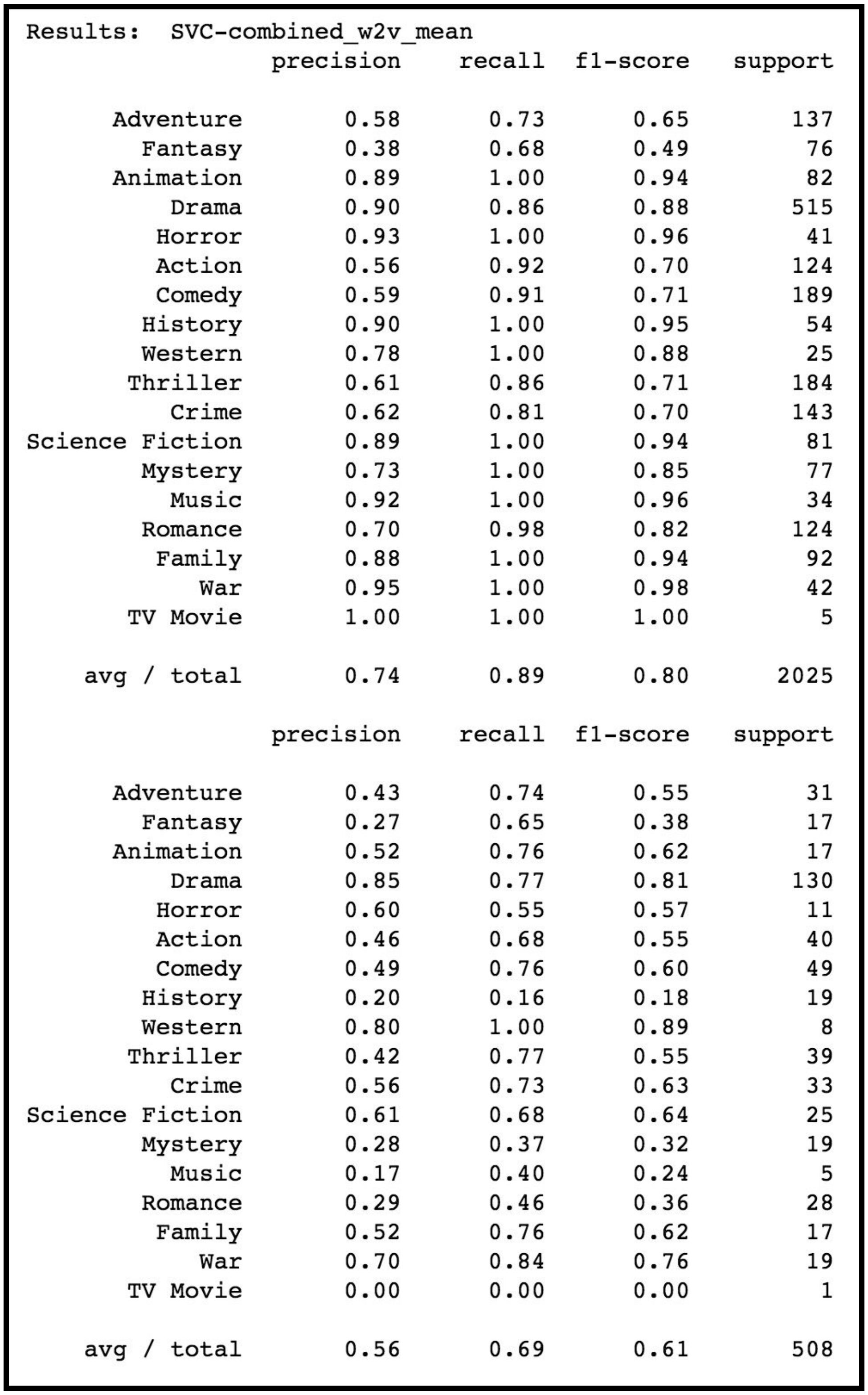
There appears to be some overfitting in this model - the train scores for precision and recall are 0.98 and 0.93, respectively, whereas the test results are much lower. This model has the best precision score out of any of the models, meaning that there are many true-positives and few false-positives. However, the recall score is much lower than many of the other models - indicating that this model has the tendency to result in false-negatives.

Looking at the test classification report, this model was able to identify “Fantasy,” “Animation,” “Horror,” and “War” genres with 100% accuracy. In other words, every time this model predicted one of those genres, it was correct. However, this model was somewhat hesitant to make those predictions, which is what caused the recall score to be so low.

Like most of our models, this model is best at predicting “Drama” movies. Although the precision score is lower than the genres listed above, it is less hesitant to make drama predictions, as evidenced by the higher recall score. This can be partially explained by the skewed dataset, which had far more drama movies than any other genre. This allowed the model to be fit on a more diverse set of words when compared to other genres.

This model was unable to successfully predict any “History,” “Western,” “Mystery,” or “TV Movie” genres. This is likely a result of the model being overfit. This claim is supported by the F1-scores in the training set of these genres, which have scores of [1.00, 1.00, 0.99, and 1.00], respectively.

Next we look at the best recall SVC model train and test classification reports.



```
Results:  SVC-combined_w2v_mean
                 precision    recall  f1-score   support

      Adventure       0.58      0.73      0.65       137
        Fantasy       0.38      0.68      0.49        76
      Animation       0.89      1.00      0.94        82
          Drama       0.90      0.86      0.88       515
         Horror       0.93      1.00      0.96        41
         Action       0.56      0.92      0.70       124
         Comedy       0.59      0.91      0.71       189
        History       0.90      1.00      0.95        54
        Western       0.78      1.00      0.88        25
       Thriller       0.61      0.86      0.71       184
          Crime       0.62      0.81      0.70       143
Science Fiction       0.89      1.00      0.94        81
        Mystery       0.73      1.00      0.85        77
          Music       0.92      1.00      0.96        34
        Romance       0.70      0.98      0.82       124
         Family       0.88      1.00      0.94        92
            War       0.95      1.00      0.98        42
       TV Movie       1.00      1.00      1.00         5

    avg / total       0.74      0.89      0.80      2025

                 precision    recall  f1-score   support

      Adventure       0.43      0.74      0.55        31
        Fantasy       0.27      0.65      0.38        17
      Animation       0.52      0.76      0.62        17
          Drama       0.85      0.77      0.81       130
         Horror       0.60      0.55      0.57        11
         Action       0.46      0.68      0.55        40
         Comedy       0.49      0.76      0.60        49
        History       0.20      0.16      0.18        19
        Western       0.80      1.00      0.89         8
       Thriller       0.42      0.77      0.55        39
          Crime       0.56      0.73      0.63        33
Science Fiction       0.61      0.68      0.64        25
        Mystery       0.28      0.37      0.32        19
          Music       0.17      0.40      0.24         5
        Romance       0.29      0.46      0.36        28
         Family       0.52      0.76      0.62        17
            War       0.70      0.84      0.76        19
       TV Movie       0.00      0.00      0.00         1

    avg / total       0.56      0.69      0.61       508
```

Unlike the Naive-Bayes-combined_bow model, this model does not have any perfect precision scores for the test dataset. This is because this model is less hesitant to make predictions - leading to a larger amount of false positives and thus lowering the precision score. Consequently, because it is less hesitant to make predictions, there are fewer false-negatives, which results in the higher recall score. Compared to the F1 score, this model outperforms the Naive-Bayes model.

Look at genre specific results for this model with regard to precision, this model is most accurate at identifying “Drama,” “Western,” “War,” “Science Fiction,” and “Horror.” Like the Naive-Bayes model, this model also predicts 0 genres as “TV Movie.” This is likely caused by the few occurrences of this genre in the dataset - only occurring 5 times in the training set and once in the test set.

Although Naive-Bayes-combined_bow model has a better precision accuracy metric, it is being overfit - as evidenced by the near perfect F1-scores in the training set in addition to the much lower test results. When this model does make predictions, the predictions tend to be accurate, but this model simply does not make many predictions. This is shown by the low recall score, as well as the four genres that it made zero predictions for.

SVC-combined_w2v_mean has a lower precision score, but has much higher recall scores and F1-scores. It is also evident that there is less of an overfit occuring. We argue that even though the precision for the SVC model is lower than the Naive-Bayes model, it is an overall better classifier, and thus the best model of our project.

Poor Doc2Vec Performance

Doc2vec has proven to perform well on large datasets. Due to our relatively small sample of 1000 movies, the lack of training data we had to transform these into document vectors did not prove helpful for modeling precision and recall. In fact, of the nine predictors that we used, the d2v vectors performed very poorly and were many of those overfit to dramas.

Conclusions & Possible Future Work

Results Summary

The best overall classification model is **SVC with combined word2vec plot mean vectors.**

We can take a look at couple more subset results to analyze the overall performance of our plot transformations across all models. The following table looks at the aggregated min, max, and mean precision and recall metrics for the TMDB, IMDB, and combined plot representations. Again, our hypothesis that the longest and more descriptive combined plots , followed by longer IMDB plots performed best for both precision and recall.

	min_test_precision	max_test_precision	mean_test_precision	min_test_recall	max_test_recall	mean_test_recall
combined	0.166339	0.725255	0.402100	0.255906	0.688976	0.395888
imdb	0.153214	0.607838	0.328140	0.253937	0.685039	0.375984
tmdb	0.166339	0.619958	0.345431	0.255906	0.633858	0.374016

We see that the longer plot descriptions for imdb movies does not lead to higher accuracy metrics with regard to precision or recall. Combining the two plot descriptions does lead to an increase in both of these metrics, although this appears to have more of an effect on the precision score than the recall score. In other words, it seems that combining the plots leads to considerably fewer false-positives, but has a lesser effect in preventing false-negatives.

Next we take a look at the aggregated results for the specific plot transformations we made: bag-of-words TFIDF, word2vec, and doc2vec. Here we see the mean precision and recall are split between bag-of-words and word2vec.

	min_test_precision	max_test_precision	mean_test_precision	min_test_recall	max_test_recall	mean_test_recall
bow	0.166339	0.725255	0.466946	0.255906	0.551181	0.373360
doc_vec	0.153214	0.186744	0.169415	0.255906	0.525591	0.325241
w2v	0.178442	0.558305	0.439310	0.253937	0.688976	0.447288

Bag of words has the best performance with regard to both precision and recall. Word2vec comes shortly there after. Doc2vec has by far the lowest performance of the three, which is expected in considering the difference in granularity. Word2vec analyzes the plots on a word-by-word basis, whereas Doc2vec analyzes the entire plot as a single vector.

Overall we are pleased with these results, and learned a great deal about the basics of building a robust dataset from scratch as well as training conventional multi-label classifiers.

Shortcomings

Our dataset is inherently skewed with more than 600 films out of 1000 having at least one of their genres classified drama. For this reason, and unsurprisingly, many of the models overfit to the drama label. It would be interesting to see how our models would do if there were a more even spread of genres across the dataset. This would not, however, reflect the real distribution of films, as more than half produced dramas as well as other categories. Another approach we could take is dropping the drama category to see how models do predicting on the other genres in the dataset that are more evenly distributed.

Future Work

One of the most exciting results of our project is that successfully created a robust machine learning pipeline, which can be easily modified for future work. Throughout the course of this project we made use of this easy modification attribute and ran the pipeline with different representations of data and model hyperparameters. Below are a few ideas for possible extensions of our work.

We made the assumption early on that we would only use the 18 TMDB genres represented in our movie dataset as the global response variable. The IMDB genres are similar, but not exactly the same as the TMDB labels, and since we can also scrape IMDB genres, it might be interesting to test our models on those to see if the models perform differently. We also decided to only include the first plot summary for each IMDB movie. Some movies have many plot summaries, so another project expansion could be using bag-of-words or w2v on multiple concatenated plots. We would expect this method to result in richer TFIDF and w2v representations similar to our "combined" TMDB and IMDB features which showed good results. We could take the vector of probabilities returned by LDA and use them as vector representations of each plot. This would be interesting to compare with our results from Doc2Vec representation.

Another interesting extension of this project would be to use more of the greater than 60 movie attributes shared between IMDB and TMDB datasets in our models. This project could also move into the deep learning realm and utilized neural nets like CNNs on movie posters, similar to the techniques employed in Spandan Madan's "An end to end implementation of a Machine Learning pipeline" referenced in the literature review section above.