Movie Rating Prediction Using Convolutional Neural Networks

Christopher Pyles

March 20, 2020

Abstract

1 Introduction

2 Data

The data used in this project is from The Movie Database's (TMDb) API. The data queried covers movies released in 2018 in English, spanning many genres:

- Action ($n = 242$)
- Adventure ($n = 134$)
- Animation ($n = 116$)
- Comedy ($n = 566$)
- Crime ($n = 154$)
- Documentary ($n = 432$)
- Drama ($n = 746$)
- Family ($n = 147$)
- Fantasy ($n = 143$)
- History ($n = 66$)

Column	Description
`id`	**primary key**, a unique ID number for each movie
`adult`	whether or not the movie is R+ rated
`genre_ids`	list of genre ID numbers corresponding to `genres` table
`original_language`	the language that the movie was originally released in
`original_title`	the title of the movie
`overview`	movie synopsis
`release_date`	the date of first release
`vote_average`	average of rating votes on a 10-point scale
`vote_count`	the number of votes
`poster_path`	URL path to movie poster

Table 1: Column descriptions for TMDb API data (the `movies` table).

- Horror ($n = 438$)
- Music ($n = 119$)
- Mystery ($n = 143$)
- Romance ($n = 222$)
- Science Fiction ($n = 188$)
- TV Movie ($n = 194$)
- Thriller ($n = 506$)
- War ($n = 29$)
- Western ($n = 31$)

The data, after being queried, were joined into a single table and written to a CSV file. The columns of interest are described in Table 1. After dropping rows with missing values in the columns of interest, the data contained $n = 2700$ rows.

2.1 Data Cleaning

Cleaning the data for this project, after dropping missing values, involved rounding the `vote_average` column, cleaning the synopsis strings, joining the `genres` and movies tables, and converting the movie posters into 140 × 92 × 3 arrays of RGB values.

The vote_average column ($\mu = 6.628$, $\sigma = 1.939$) describes the variable of interest, representing the average rating across votes for a single movie. In this analysis, this column will be transformed from a continuous variable into an ordinal variable with possible values 1 to 10 by rounding the vote average to a whole number. After rounding, the distribution of ratings is provided in Figure 1.

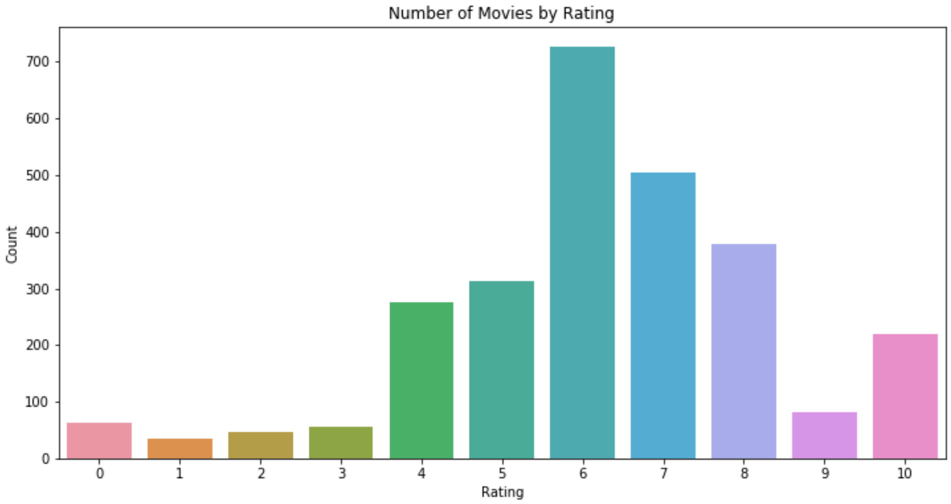


Figure 1: Number of movies for each value of ratings, rounded from vote_average.

To clean the movie synopses, all letters were transformed to lowercas, and any characters not matching the regex `[A-Za-z0-9 ]` were replaced with spaces.

The genre_ids column is a list of genre IDs encoded as a string, so to start any empty lists, "[]", were replaced with NaN. Then, the string were split into Python lists of IDs, and were joined with the genres table, so that movies had a column genres where each value is a list of genres for that movie. Figure 2 shows the breakdown of movies by genre.

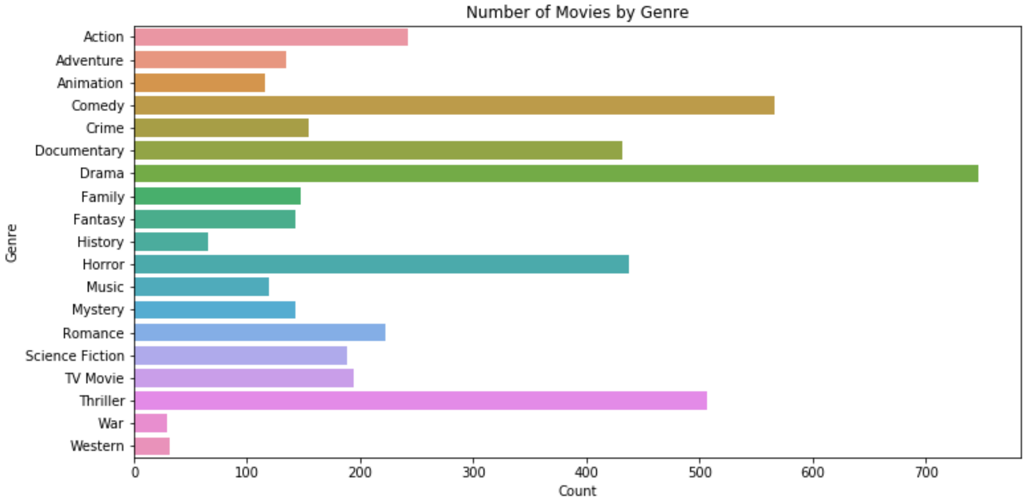


Figure 2: Number of movies in each genre.

Lastly, each movie poster was downloaded from TMDb and stored as a 140 × 92 × 3 array in NumPy, stored as a .mat file. The structure of this file is analagous to a dictionary, where each key is a movie ID (as a string) and each value is the 3-D array of RGB values describing the poster.

2.2 Exploratory Data Analysis

Before modeling, a cursory analysis of the data yielded the interesting relationships:

- Figure 3 shows that vote count tends to increase logarithmically with rating until 8, after which it drops significantly. This demonstrates that having more votes tends to ”bring down the curve.”

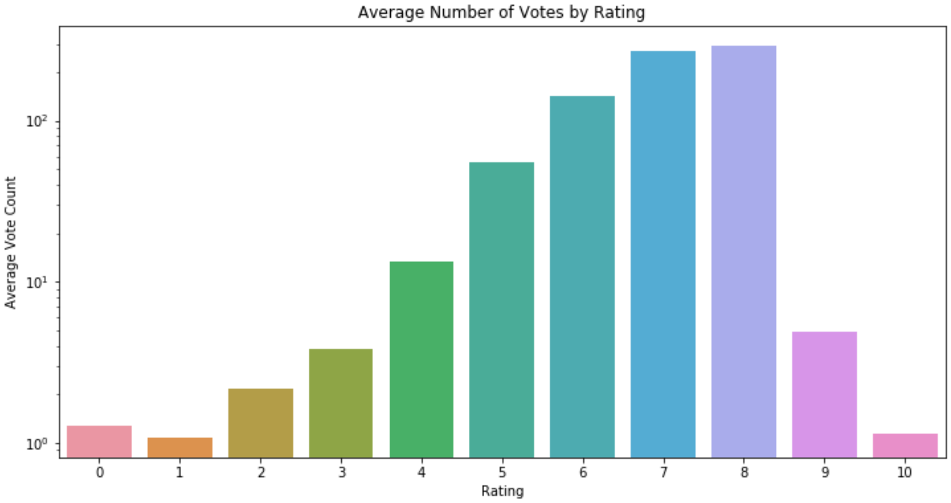


Figure 3: Average vote count by rating.

- There are significantly more non-adult movies than adult movies, as demonstrated in Figure 4. It is interesting to note that adult movies have a double-peak distribution with far less spread than do the non-adult movies.

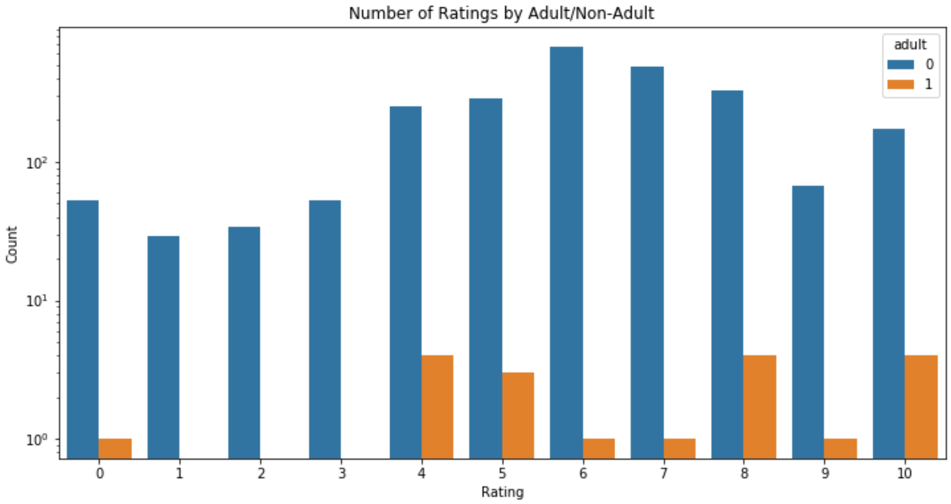
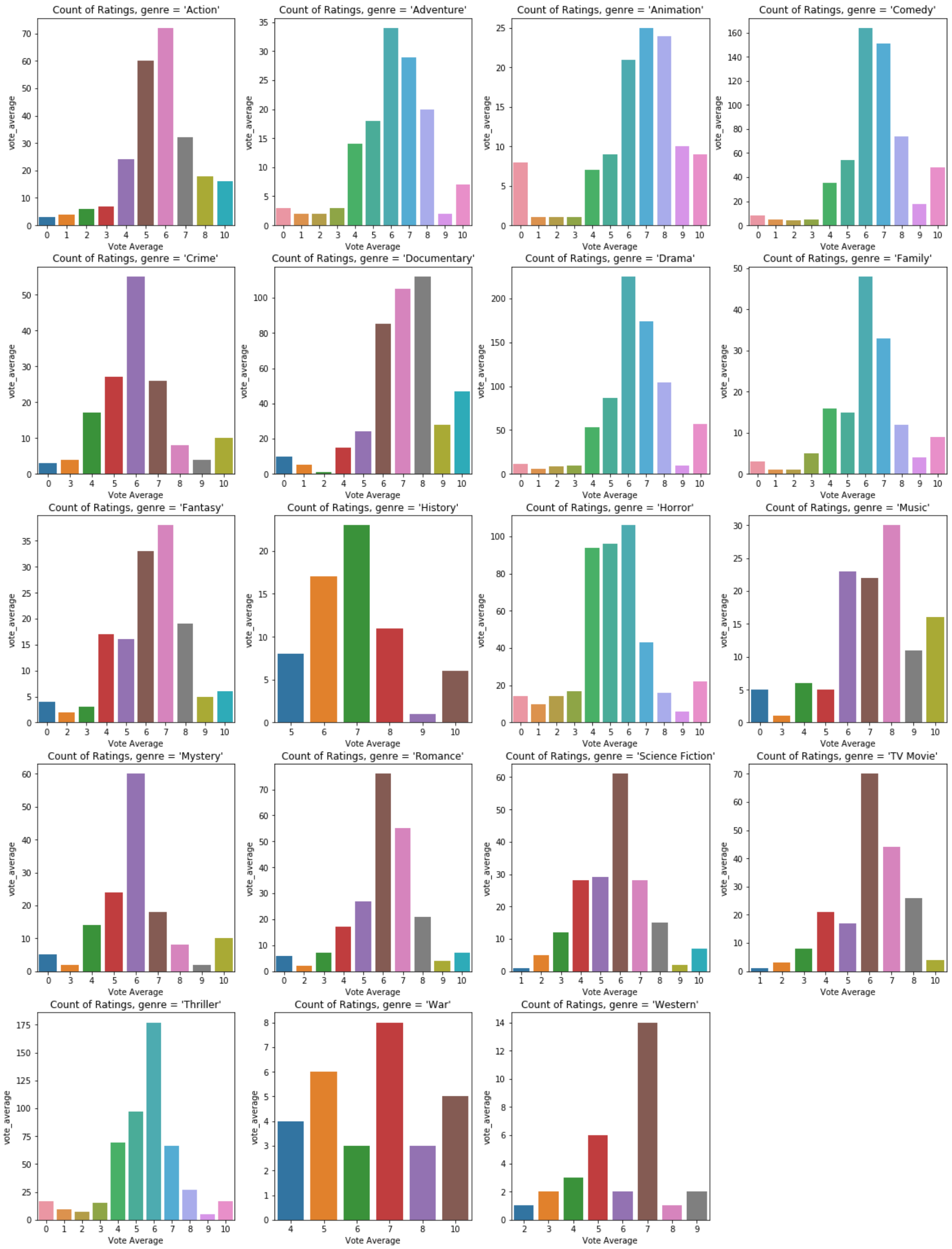


Figure 4: Number of ratings by adult/non-adult.

- Figure 5 shows the distribution of ratings for each genre.


Count of Ratings, genre = 'Action'

Count of Ratings, genre = 'Adventure'

Count of Ratings, genre = 'Animation'

Count of Ratings, genre = 'Comedy'

Count of Ratings, genre = 'Crime'

Count of Ratings, genre = 'Documentary'

Count of Ratings, genre = 'Drama'

Count of Ratings, genre = 'Family'

Count of Ratings, genre = 'Fantasy'

Count of Ratings, genre = 'History'

Count of Ratings, genre = 'Horror'

Count of Ratings, genre = 'Music'

Count of Ratings, genre = 'Mystery'

Count of Ratings, genre = 'Romance'

Count of Ratings, genre = 'Science Fiction'

Count of Ratings, genre = 'TV Movie'

Count of Ratings, genre = 'Thriller'

Count of Ratings, genre = 'War'

Count of Ratings, genre = 'Western'

Figure 5: Distribution of ratings by genre.

- No discernible relationship is seen between the length of the overview and the movie's rating.

- The distributions of ratings by release date day of week appear to be different in skew (Figure 6), indicating that this may be an important feature. No discernible relationship is seen between rating and day of month or month of release.

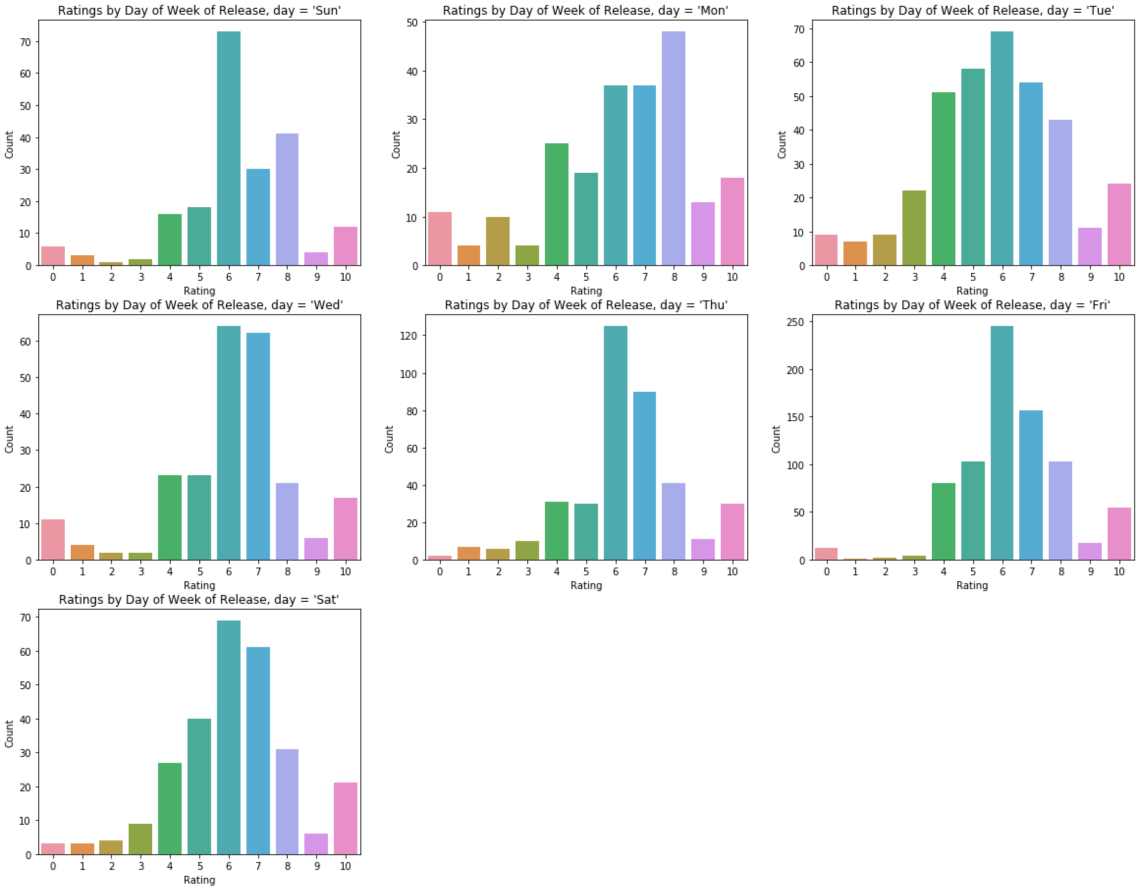


Figure 6: Distribution of ratings by day of week of release.

3 Analysis

3.1 Words in Synopses

As the first part of this analysis, the statistical significance of the presence of different words in synopses is studied using hypothesis testing. The null hypothesis for this test is that the presence of a word does not affect the average rating, and the alternative is that the presence of the word results in an increase in rating. The test statistic used here is the aboslute difference between the mean rating of movies where the word is present in the synopsis and the movies where it is not. Under the null hypothesis, the truth value of a word being present in a synopsis is shuffled for each observation, so that same number of "present"s is obtained. Then, the test statistic is computed and the p-value is calculated as the proportion of the test statistic values that are greater than or equal to the observed value for that word.

3.2 Classification without Posters

The next step in the analysis is to attempt classification using the features in `movies` alone (i.e. without the posters). In this section, several different classification methods are tested, including a ridge classifier, a decision tree classifier, a support vector classifier (SVC), and a k-nearest neighbors (kNN) classifier. Each of these models is trained on training data from `movies` and evaluated using different error metrics on testing data from `movies`. Three different error metrics are used to evaluate each classifier: classification accuracy, root-mean-squared prediction error (RMSE), and mean absolute error (MAE). The last two, normally regression metrics, are used because the ratings are derived from continuous variables, and a classifier that classifies a movie closer to its rating than another is more accurate, and hit-miss accuracy does not account for this.

In order to select features, multiple methods are used and compared. The k best features based on chi-squared statistics, ANOVA F-value, and mutual information are each selected and compared using tuned hyperparameters for each model. k is also tuned using hyperparameter selection.

5-fold cross-validation (CV) is used to tune hyperparameters. The hyperparameters being tuned for each model are:

- ridge classifier: α, the regularization strength
- SVC: C, the regularization parameter (analagous to α^{-1} from the ridge classifier)

- kNN classifier: the number of neighbors

Hyperparameters are tuned individually for sets of features chosen by the different methods of feature selection described above. The model with the best CV error is chosen and trained on the training set using the corresponding features and evaluated using testing error.

3.3 Classifying Movie Posters

In this portion, movie posters, stored as 3-D arrays, are used as inputs to conlutional neural networks (CNNs) in order to classify the movie by rating. This section does not involve any features in `movies`. CNNs are trained on the 4-D array of all images and the classes encoded as dummy variables. The networks are compiled using categorical cross-entropy loss and the Adam optimization algorithm. They are evaluated as in §3.2, using classification accuracy, RMSE, and MAE with 5-fold CV to determine the best network structure. The model with the best metrics is trained on the entire training set and evaluated on the test set.

3.4 Classification with All Data

In the final portion of this analysis, all available data is brought together to build a final classifier. Combinations of models from §3.2 and §3.3 are combined to classify ratings, including by using poster classifier predictions as features in a final model incorporating data from `movies`. As before, models here are evaluated using accuracy, RMSE, and MAE and compared with 5-fold CV, before training and testing a final model on the test set.

4 Results

4.1 Words in Synopses

This analysis found that there were 110 words that were statistically significant, summarized in Table 2 with their p-values. Each of these words was added as a feature to `movies` as dummy variables with 0-1 values indicating the presence of that word in the synopsis, resulting in the data points in `movies` becoming 141-dimensional.

Word	p-value	Word	p-value	Word	p-value
this	0.001	through	0.005	rough	0.004
after	0.006	take	0.012	cross	0.002
known	0.003	group	0.0	hard	0.001
side	0.0	film	0.004	brother	0.037
killer	0.001	gain	0.041	childhood	0.001
business	0.019	great	0.0	small	0.018
seem	0.009	trip	0.02	director	0.02
follows	0.028	care	0.009	feat	0.0
become	0.03	couple	0.049	danger	0.005
house	0.003	even	0.042	dang	0.002
meets	0.0	party	0.027	between	0.003
aunt	0.017	cove	0.003	mean	0.003
each	0.012	down	0.004	christmas	0.016
making	0.001	dire	0.0	year	0.018
place	0.002	story	0.01	press	0.046
only	0.0	father	0.04	ross	0.005
foot	0.009	disc	0.009	dark	0.001
perform	0.0	under	0.047	follow	0.0
lore	0.036	ller	0.003	document	0.002
give	0.038	busi	0.008	first	0.0
found	0.031	direct	0.007	brea	0.009
front	0.047	light	0.002	anger	0.001
women	0.024	does	0.018	strange	0.005
ours	0.035	dangerous	0.002	hunt	0.002
behind	0.001	break	0.007	secret	0.0
kill	0.004	evil	0.005	takes	0.004
attempt	0.01	host	0.002	fall	0.014
disco	0.003	survive	0.001	music	0.0
mysterious	0.004	meet	0.037	which	0.029
king	0.015	realize	0.03	soon	0.002
less	0.013	murder	0.0	know	0.017
turn	0.049	college	0.014	them	0.0
begin	0.001	documentary	0.005	wing	0.032
years	0.002	self	0.02	comedy	0.0
range	0.001	super	0.042	form	0.001
discover	0.011	across	0.006	test	0.02
mall	0.04	cover	0.026		

Table 2: Words with statistically significant presences and corresponding p-values.

Classifier →	**Ridge**		**SVC**		**Decision Tree**	**kNN**	
Feature Selection Scheme ↓	α	**MAE**	C	**MAE**	**MAE**	**neighbors**	**MAE**
No Selection	0	1.19740	100	0	0	1	0
Chi-Squared Statistics	0.1	1.40022	10	0	0	1	0
ANOVA F-value	0	1.40130	100	1.15998	0.27983	1	0.32484
Mutual Information	0	1.41811	10	0	0	1	0

Table 3: Results of feature selection and hyperparameter tuning without posters.

Classifier →	**Ridge**		**SVC**		**Decision Tree**		**kNN**	
Feature Selection Scheme ↓	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
No Selection	1.46016	2.14438	0.95772	1.80289	0.95610	1.94309	1.07805	1.94518
Chi-Squared Statistics	1.36260	1.98449	0.90569	1.76460	1.01789	2.01377	1.07967	1.96638
ANOVA F-value	1.38537	2.03386	1.43577	2.15836	1.14472	2.15271	1.18699	2.20421
Mutual Information	1.46992	2.04661	0.87154	1.72500	1.03089	1.96886	1.08618	1.96804

Table 4: Classifier testing errors on classification without posters.

4.2 Classification without Posters

After the first round of hyperparameter search, the ridge classifier performed best with $\alpha = 0$, the support vector classifier with $C = 100$, and the kNN classifier with 1 neighbor. The decision tree classifier had no hyperparameters of interest. The results of feature selection and tuning are provided in Table 3. Table 4 details the testing MAE and RMSE for each model and feature selection scheme. The hyperparameters used in these models are:

- ridge: $\alpha = 0$
- SVC: $C = 100$
- kNN: 1 neighbor

The results show that a support vector classifier with $C = 100$ and feature selection using mutual information is the best classifier, with an MAE of 0.87154 and an RMSE of 1.72500. The predictions of this classifier are depicted in Figure 7. This feature selection scheme chose 10 features: `popularity`, `vote_count`, `Horror`, `Music`, `Thriller`, `overview_len`, `day`, `take`, `direct`, and `evil`. The last three features correspond to the presence of words as described in §4.1.

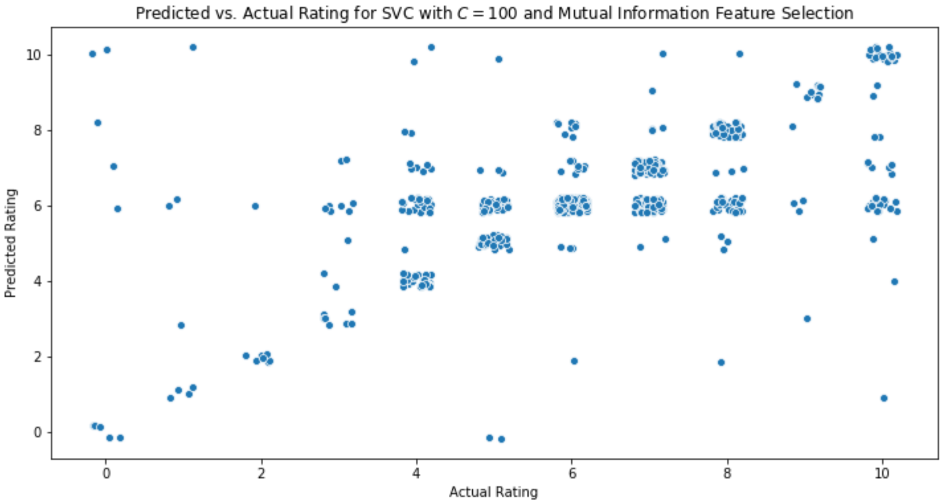


Figure 7: Results of classification without posters with $U(-0.2, 0.2)$ jittering on both axes.

4.3 Classifying Movie Posters

This section created multiple neural networks, both convolutional and otherwise. The non-convolutional neural networks were trained on the flattened image arrays and contained different numbers dense hidden layers. Both softmax and sigmoid activation functions for the final layer were considered. The convolutional NNs were trained on the 3-dimensional image arrays, which were then flattened and run through non-convolutional dense layers. All NNs in this section use the Adam optimizer and categorical cross-entropy loss.

Several neural network models analyzes in this section predicted all fours or all zeros, including one of the convolutional NNs, defined in `keras` as below:

```
model = Sequential()
model.add(Conv2D(32, (2, 2), input_shape=input_shape))
model.add(Activation(’relu’))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (2, 2)))
model.add(Activation(’relu’))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (2, 2)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(dummy_y.shape[1]))
model.add(Activation('sigmoid'))

model.compile(loss="categorical_crossentropy", optimizer="adam",
metrics=["mae", "mse"])
```

The testing prediction plot for this CNN is provided in Figure 8.

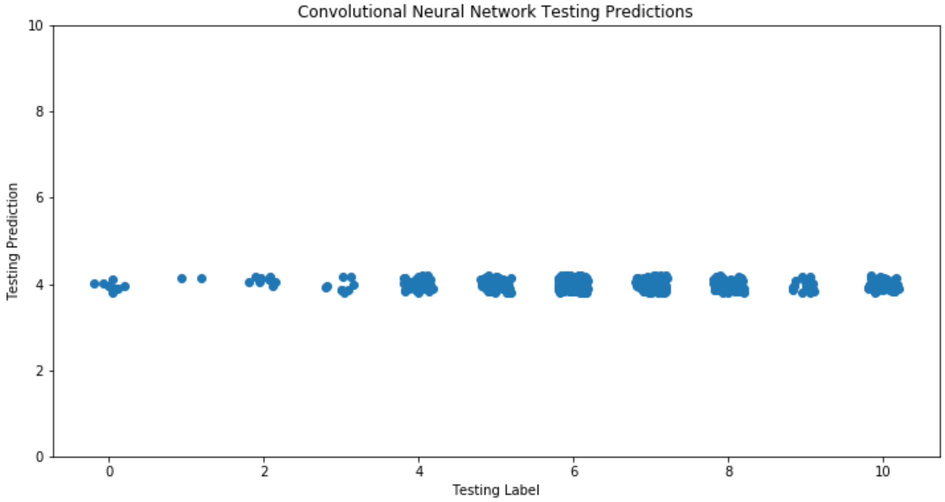


Figure 8: Results of poster CNN with $U(-0.2, 0.2)$ jittering on both axes.

Another non-convolutional network that was tested, consisting of a dense input layer with 8 nodes, a dropout layer with a rate of 0.25, and a dense output layer with a sigmoid activation function predicted almost all zeros, as shown in Figure 9.

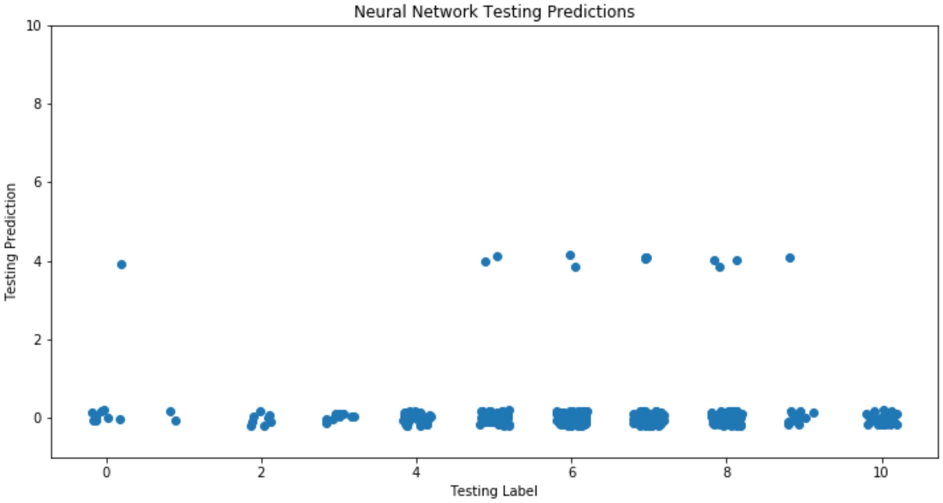


Figure 9: Results of poster NN with $U(-0.2, 0.2)$ jittering on both axes.