CS 4780 Final Project Report
The Least Square Movie: Film Rating Prediction using Regression

1. Introduction

In our project, we tackle the problem of predicting movie ratings on The Internet Movie Database, IMDb (IMDb.com, 2014). Movie ratings are something so elemental to a film that they often influence who views the film, how much revenue a film grosses and how a movie's fame is recorded in history.

Our approach was to first collect and format all the data related to films which had IMDb ratings and split it into different data sets. To predict ratings, we trained a number of regression models, including linear, lasso, and support vector regression, used kernels to explore different feature mappings, and used feature selection to determine the most important film attributes.

Our project is similar to other projects such as the Netflix Prize (Netflix, 2009), but instead of predicting movie ratings for individual users, we tried to solve the more general problem of predicting average movie ratings on IMDb for all users. In addition, we chose to predict movie ratings rather than box office revenue, because we were primarily interested in the quality of a film.

Our results indicate that ridge regression using a polynomial kernel performed the best for predicting movies. One reason this may be is that this method incorporates feature selection and interaction features that were relevant to our data. Using kernelized ridge regression, we are able to show with 95% confidence that our true RMSE will fall within the range of 0.84–0.95. Through these results we were able to analyze the effects of individual attributes and discover interesting properties of movie rating distributions.

2. Problem Definition and Methods

2.1. Task Definition

The purpose of our system is to predict a movie's IMDb rating based on known attributes, which include director, cast, budget, release date, etc. This is especially useful for gauging the quality of unreleased movies from the perspective of both viewers and filmmakers.

For filmmakers, creating a new movie is a colossal investment of resources and carries a great deal risk in terms of the film maker's reputation and future prospects. It would be incredibly beneficial if they could determine which attributes would make their movie more popular so they can best allocate their resources. Our project provides a solution to this by allowing a director to predict a rating to a movie which based purely on features.

Likewise, for viewers setting out to see a movie an investment of time and money that could otherwise be used for other purposes. Paying to see a movie is a way to show appreciation for a movie, however it doesn't make sense to show appreciation for something that does not deserve it. Our project allows a user to predict whether or not a movie deserves his appreciation.

From these predictions, we can also elucidate certain trends in viewer preferences with respect to the overall quality of a film, such as actors and actresses, and the best months to release movies. Our data indicates which movie attribute are most positive and negative correlated with rating, demonstrating which attributes affect audiences the most and reflecting popular culture.

This project answers the following key questions:

i. Is there a correlation between movie attributes and IMDb viewer ratings?
ii. Which movie attributes have the greatest influence on ratings (perhaps the release time is just as important as its cast)?

2.2. Dataset Formation

2.2.1. Movie Features

To train our regression models, we considered the followig movie attributes: Director, Cast, Budget, Production Company, Production Country, Runtime, Genres, Decade of release, Month of release, Writers, MPAA rating, and Languages. We represented each movie as a vector of attributes. However, since each of these features can take on many different values, we needed a way to quantify each feature to be used in our prediction models described in Section 2.3. We

decided to represent the majority of the features as binary attributes that take on a value of 1 if that particular feature value is present in a movie, and 0 otherwise. For example, the Director feature is split into its values, such that if Woody Allen directed a movie, the vector would look like: "Woody Allen: 1, Clint Eastwood: 0, Tim Burton: 0...".

2.2.2. Feature Selection Methods

After enumerating all features with binary attributes in this way, we found that we had 12,259 attributes in total. Having such a large space is potentially detrimental, since it could slow down our prediction algorithm considerably and overfit the training data. To reduce the number of features and mitigate these effects, we explored several different feature selection methods.

First, we chose to implement a custom scoring method for attributes with an exceedingly large number of feature values, which were Director (1166 values), Cast (5229 values), Writers (4181 values), and Production Companies (1884 values). To do this, we computed a score for each of those attributes, by taking the average score of the movies that each attribute value appeared in. For example, if the Director was "Christopher Nolan", we would look at all of the movies directed by him and average those scores. If the value was unknown, meaning it did not appear in our training set, then we would assign the average director score. Finally, we normalized the scores based on the minimum and maximum director scores. Sometimes, a feature value would only appear in a small number of movies (an actor might only appear in one movie). To ensure that these scores were not disproportionate weighted, we simply assigned a feature the average score if their frequency was below a certain threshold.

In addition to scoring, we employing several other forms of features selection, including different regression techniques and recursive feature elimination (RFE) detailed in Section 2.3.

2.2.3. Data collection

We formed our raw movie data-set by compiling data from multiple sources and combining them into a single data-set. We first scraped Wikipedia for a list of every movie title made in the past century, from 1915 to 2015. With the comprehensive list of movie titles, we queried three online movie database APIs: Rotten Tomatoes, The Open Movie Database (OMDb), and The Movie Database (TMDb), which each gave us information on different relevant attributes. To ensure that the ratings associated with movies in our data-set were representative of real movie quality, we chose to ignore movies with less than 500 IMDb rating votes. If the number of votes is too low, the rating assigned to the movie may not be an accurate reflection of the movie's quality.

2.3. Algorithms and Methods

We made use of seven different regression techniques to predict movie ratings: ordinary least squares, lasso regression, ridge regression, support vector regression, logistic regression and KNN regression. Additionally, we ran ridge regression using four different kernels, and explored the use of feature selection to reduce the number of attributes in the feature space. We explored multi-class regression techniques (logistic) because we want to see see how they compared to normal regression methods.

2.3.1. Ordinary Least Squares

Our first and most basic regression approach was to run ordinary least squares on our data. The objective function of ordinary least squares is

$$\min_w ||Xw - y||_2^2$$

One issue with ordinary least squares is that coefficient estimates rely on the independence of the features, which is likely not the case for, say, actors and genres.

2.3.2. Ridge Regression

Because our feature space of movie attributes is so large, running ordinary least squares risks over-fitting to our data. To account for this possibility, we also tried using ridge regression to predict movie scores. The objective function of ridge regression is

$$\min_w ||Xw - y||_2^2 + \alpha||w||_2^2$$

By introducing the shrinkage parameter α, ridge regression imposes a penalty on large coefficients, such that most coefficients are close to zero. The higher the value of α, the higher the penalty. This way, some of the problems with ordinary least squares are reduced. A value of $\alpha = 0$ corresponds to ordinary least squares. In order to find a suitable value of α to predict movies well on the test set, we used leave-one-out cross-validation on the training set.

2.3.3. Lasso Regression

A similar regression technique we used to minimize the weights of our predictor is lasso regression. Lasso

regression's objective function is

$$\min_w ||Xw - y||_2^2 + \alpha||w||_1$$

The biggest difference between ridge and lasso regression is that instead of penalizing large values of the ℓ_2-norm of the weight vector, it uses the ℓ_1-norm. This produces a sparser weight vector than ridge regression with more entries equal to or close to zero. Again, to find a good value of α, we used leave-one-out cross-validation on the training set.

2.3.4. Support Vector Regression

Another regression technique we explored was Support Vector Regression (SVR), which attemps to fit a function on the feature space to the training data.

The dual objective function of SVR using kernels is as follows:

$$max \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i*)(\alpha_j - \alpha_j*)k(x_i, x_j) \\ -\epsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i*) + \sum_{i=1}^{l}(\alpha_i + \alpha_i*) \end{cases}$$

subject to

$$\sum_{i=1}^{l}(\alpha_i + \alpha_i*) = 0 \text{ and } \alpha_i, \alpha_i* \in [0, C]$$

2.3.5. Logistic Regression

We also wanted to compare our regression methods to a multi-class classifier. For this, we used logistic regression, which makes predictions based on probability estimates derived from the training data.

Logistic regression with an L2 regularized penalty has the following objective function:

$$\min_{w,c} \frac{1}{2}w^T w + C\sum_{i=1}^{n} log(exp(-y_i(X_i^T w + c)) + 1)$$

2.3.6. Kernel Ridge Regression

However, the data might not has a linear trend. Therefore, we considered projecting into a higher dimensional feature space and using a kernel to run regression on that space. Since this projection introduces many new features, ridge regression is a promising choice to focusing on the most important features and get a good result.

Ridge regression in the new higher dimension space can be expressed as

$$\min_w \sum_i (\Phi(x_i) * w - y_i)^2 + \alpha^*||w||_2^2$$

(where α^* is the ridge coefficient), which can be transformed into

$$\max_\alpha(-\frac{1}{2}\sum_i(\alpha_i)^2 - \frac{1}{2\lambda}\sum_i\sum_j \alpha_i\alpha_j\Phi(x_i)\Phi(x_j) - \sum_i \alpha_i y_i)$$

where $\Phi(x_i)$, $\Phi(x_j)$ are vectors from the higher dimensional feature space,and $\alpha_i, \alpha_j$are dual variables. As the above equation suggests, we only need $\Phi(x_i)\Phi(x_j) = K(x_i, x_j)$.

We used and evaluated multiple kernels for our kernel ridge regression,including polynomial kernel, Gaussian Kernel,Sigmoid Kernel and Radial Bias Function Kernel. Among them, polynomial performs the best.

2.3.7. KNN Regression

In addition to the above linear regression methods, we also apply K-Nearest Neighbor Regression to predict rating based on a similarity measure of movie attributes. In our implementation, Euclidean distance is used to measure the similarity between two feature vectors. Once we have calculated the distances between every pair of features vectors, we can predict the rating based on two methods: the first one is to calculate the average of the numerical rating of the K-nearest neighbors and predict using that rating. A second approach is to use an inverse distance weighted average of the K-nearest neighbors. KNN regression uses the same distance functions as KNN classification.

Since the performance of KNN depends on the value of k, the number of nearest neighbors, we used 4-fold cross validation to get the performance of each k (by measuring RMSE) to select the best k and the better weight method.

3. Experimental Evaluations

3.1. Methodology

The collected dataset was split into a training set, a validation set, and a test set. We trained our algorithms on the training set and tuned parameters according to their performance on the validation set. In order to evaluate the performance of our algorithms, we measured the mean absolute error and the root-mean-square error of each algorithm on the test set. In order to establish statistical significance, we compared each algorithm's performance to a baseline predictor which always predicts the average rating of all movies in the training set. We also compared the rating distributions for our algorithms and analyzed the performance of ridge, lasso, and KNN regression with various parameters. Finally, we considered the per-

formance of multiple types of kernels alongside ridge regression.

3.2. Results and Discussion

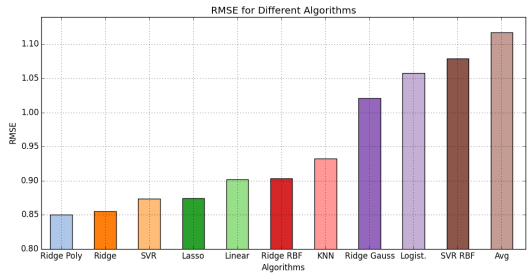

Figure 1. Comparison of Algorithms using RMSE.

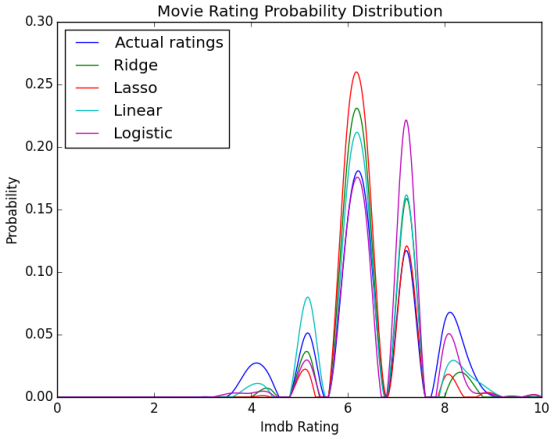

Figure 2. Probability distribution of IMDb movie ratings on our test set.

Our results are summarized in Figure 1 and indicate that ridge regression with a polynomial kernel had the best prediction performance. Next came ridge, followed by support vector regression. Since ridge regression imposed L2 penalty on the weight vector, it essentially provided feature selection, which prevented overfitting of the training set. Additionally, the polynomial kernel incorporated interaction features, which improved results. This was expected, since interactions between attributes such as Director and Production Company are likely to affect film ratings. For a visual indication of how accurate our predictive methods were, we plotted the distribution of ratings on our test set, along with the distribution of predicted ratings for Ridge, Lasso, Linear, and Logistics regression. As shown in Figure 2, the distribution of ratings clusters in the midrange, especially near integer values like 6 and 7. We find that our algorithms do a good job of conforming to the distribution of ratings even on an unknown set of movies. Specifically, lasso regression produces predictions the most frequently in the 6 range, while logistic predicts most frequently in the 7 range.

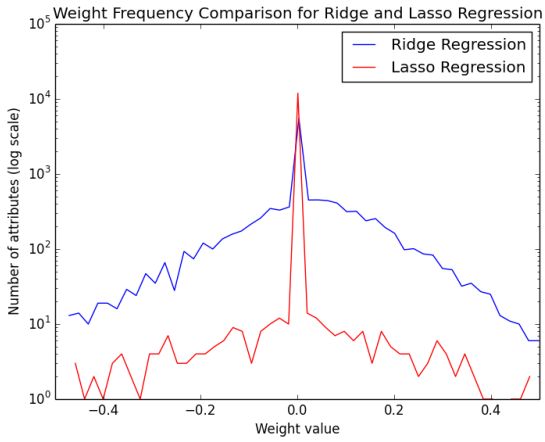

Figure 3. Coefficient frequencies for lasso and ridge regression trained on the training set. Almost all 12,259 features have a weight equal to or close to zero. Lasso regression penalizes high weight values more harshly.

Figure 3 shows the coefficient frequencies for lasso and ridge regression after training them on the training set. The results confirm that lasso regression imposes a harsher, sharper penalty on the weights of the attributes, yielding a sparse weight vector wherein nearly all attributes have a weight of zero. This indicates that the "built-in" feature selection of these algorithms were effective in eliminating many features by assigning them zero weight. The results in Figure 4 confirm the expected effect of the value of alpha on the error of lasso regression on the test set. For $\alpha = 0$, lasso regression is equivalent to linear regression, so the predictor is over-fitting the training set leading to high errors on the test set. For an optimal value of α, the error on the test set is minimized, and for large values of α, too many attributes have zero or low weights so the predictor under-fits the data. In this way, Figure 4 represents the classical over-fitting/under-fitting curve.

For kernelized ridge regression, changing the ridge coefficient α did not really influence the performance of the algorithm. However, changing kernels yielded quite different results. Among them, polynomial performs the best, whereas Sigmoid performs the worst,

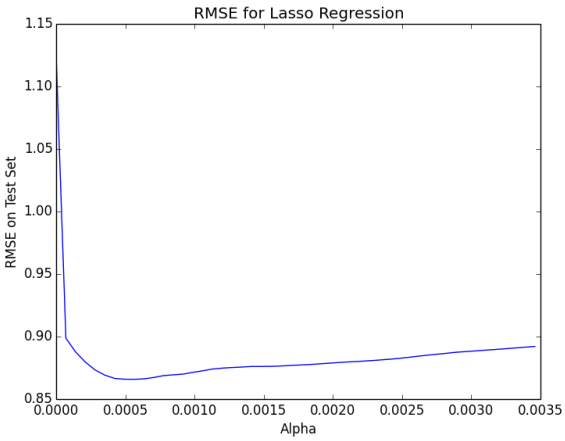

Figure 4. Root mean square error of lasso regression on the test set for different values of alpha. The value of α found using leave-one-out cross-validation on the training set was 0.001, which is quite close to the optimal value of $\alpha = 0.0005$.

which a root mean square error even higher than our baseline. Intuitively, this suggests that there are dependencies in our features since polynomial kernels take into account not only the given features, but also the interaction between those features. And although a polynomial kernel is equivalent to that of polynomial regression, its implicit feature space avoids adding a large number of interactive features to our already large feature space.

The results in Figures 6 and 5 show the attributes with the highest and lowest weights for each algorithm. Runtime, the production company Bad Robot, director Tim Burton, and the Documentary and Animation genres are examples of attributes which were very positively-weighted in both ridge and lasso regression, indicating that movies with these attributes are more likely to receive higher ratings. On the other hand, the attributes Uwe Boll and Greg Robbins had large negative weights, indicating that movies with these attributes were likely to receive lower ratings. This makes sense, because director Uwe Boll actually has a reputation for having made some absolutely terrible movies.

For both algorithms, runtime is the most highly-weighted feature. In order to explore why this was, we plotted the test set ratings for this feature in Figure 7 alongside the regression line for ridge regression and the best-fit (least-squares) line for the test set ratings. The plot shows that the data points from the test set (blue) were indeed distributed in a positive linear fashion, indicating that higher runtime is associated with a higher rating. We also plotted the prediction line that our algorithm has computed from the training data, which appears to closely match the trend line of the test data. This again demonstrates that our prediction algorithm does a good job of modeling real movie data.

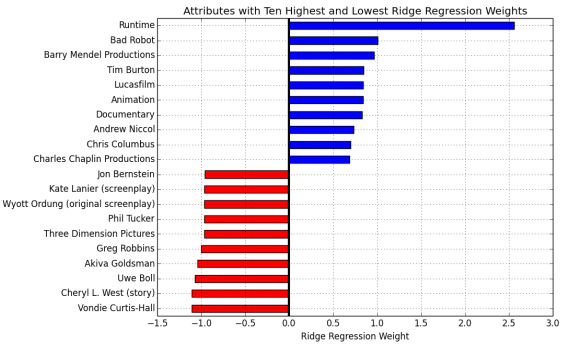

Figure 5. Attributes with the highest and lowest weights for ridge regression.

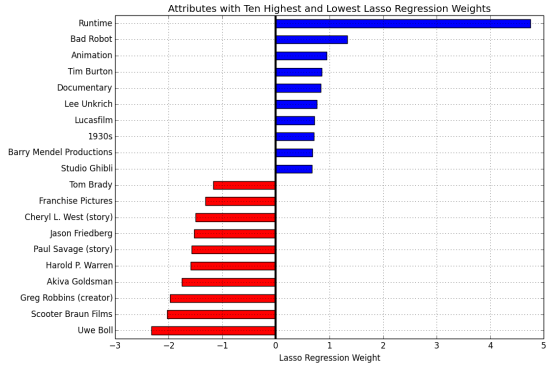

Figure 6. Attributes with the highest and lowest weights for lasso regression.

From these weights we can discern which attributes are likely to boost, or hurt, the rating of a movie. To create the optimal movie, we can select attributes with the highest weights in each category. For example, a movie directed by Tim Burton, produced by Bad Robot, with a long runtime, and has the genre Animation and/or Documentary is highly likely to be well received by viewers who use IMDb.

For KNN regression, using inverse distance weight to predict ratings had an overall better performance than using uniform weighting, which is expected since it

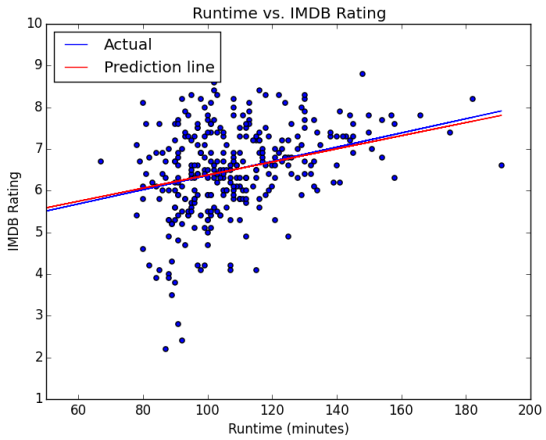

Figure 7. Runtime vs IMDb rating with Ridge regression. There is clearly a positive linear correlation between runtime and IMDb rating on our test set, as indicated by our results. Our predictions matched the actual ratings well.

uses more similarity information. Changing the number of nearest neighbors also influences the performance of the algorithm on the test set (as Figure 8 shows) with the best k being around 10. The graph also shows the classic overfitting curve, as the RMSE on the test set first quickly drops and then starts to increase for $k > 10$.

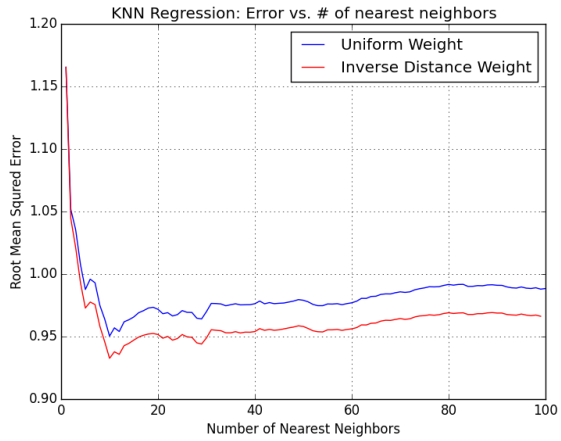

Figure 8. Root Mean Square Error vs number of nearest neighbors in KNN

The performance of our regression models doesn't improve much with recursive feature elimination. However, feature selection does offer insight regarding the influence of each movie feature on the movie rating. The top 5 important features according to feature selection are:

i. Language is English
ii. Total runtime
iii. Genre is Drama
iv. Made in United States of America
v. Made in 2000s

4. Related Work

The Netflix Prize (Netflix, 2009) also centers around predicting movie ratings. Whereas we tried to predict the overall rating for a movie, the Netflix Prize focuses on predicting ratings for individual users. Therefore, their data also contains information about users' past ratings. Participatants in the Netflix competition typically use neighborhood similarity to predict ratings based on the ratings of users with similar preferences (Lab, 2012), whereas our machine learning approaches focus on the movie attributes themselves, using regression models to find a correlation between film attributes and their ratings. Taking film attributes into account to determine how they could influence a particular user's rating, similar to our approach, could improve the performance of algorithms on the Netflix Challenge, and it is indeed an approach which was adpoted by the winning team.

While the Netflix problem is useful for user recommendentions, ours provides more insight into the movie itself and could be easily adapted and refined to predict box office revenue, critic ratings, or awards winnings, thus assisting investors and production companies in making better production decisions.

5. Future work

For future research into the area, we believe several avenues are open for exploration. The first and simplest is to use another set of data. Throughout this project we used IMDb ratings, but our actual data set also includes scores from Rotten Tomatoes (Rotten Tomatoes, 2014) which we left unused due to scope and time constraints. It would be interesting to see the results on these sets of rankings and to see if similar trends exist and how classification methods may differ on their effectiveness.

Additionally, although we experimented with kernels, our range was limited as we used very few kernels. With this, the fact that our lowest error was achieved while using a kernel points to the power of kernels. Leaving kernels further unexplored in the future would seem as a huge mistake as they could greatly reduce

our error. However, further exploration would likely need the help of extreme learning machines to create valid kernels. The idea with these machines is to generate valid kernels and quickly test to see if error reduction is achieved (Bin Li & Li, 2014).

Finally, one area we left untouched was neural networks. Neural networks are known to be useful for both classification and regression analysis for these reasons would seem an obvious choice for our problem of assigning movie ratings.

6. Conclusion

The results on our dataset suggest that using a polynomial kernel alongside ridge regression is a good approach to predicting movie ratings, providing predictions with a mean average error of 0.63 and a root mean square error (RMSE) of 0.85. With 95% confidence , our true RMSE will fall within the range of 0.84–0.95. This is reasonable because polynomial kernels not only consider the given features of the input samples to determine their similarity, but also combinations of these. For films, combinations of attributes are important in determining the quality of a movie. For example, a good drama actor appearing in a drama film will likely increase the movie's rating, whereas a comedy actor may not.

In general, using our custom scored attributes decreased the performance of our algorithms. One reason this could be the case is that scored attributes provide less information than the full binary attributes for cast, directors, production companies, or writers.

The runtime of the film was the second most important feature according to recursive feature elimination on ordinary least squares regression, and the highest-weighted feature in both ridge and lasso regression. This suggests to filmmakers that producing movies with a long runtime may be the key to a good rating. Interestingly, the budget of a film was not one of the most critical factors in determining the rating of a movie, although it was in the top 30 features according to recursive feature elimination.

References

Bin Li, Xuewen Rong and Li, Yibin. An Improved Kernel Based Extreme Learning Machine for Robot Execution Failures. http://www.hindawi.com/journals/tswj/2014/906546/, 2014. Accessed: 2014-12-10.

Domke, Justin. Kernel Methods and SVMs. http://users.cecs.anu.edu.au/~jdomke/courses/sml2010/06kernels.pdf, 2010. Accessed: 2014-12-10.

Glowing Python, The. Linear regression with Numpy. http://glowingpython.blogspot.com/2012/03/linear-regression-with-numpy.html, 2012. Accessed: 2014-11-05.

IMDb.com, Inc. The Internet Movie Database. http://www.imdb.com/, 2014. Accessed: 2014-12-10.

Lab, Princeton Edge. How does Netflix recommend movies? http://scenic.princeton.edu/network20q/wiki/index.php?title=Q4:_How_does_Netflix_recommend_movies%3F, 2012. Accessed: 2014-12-10.

Netflix, Inc. Netflix Prize. http://www.netflixprize.com/, 2009. Accessed: 2014-12-10.

OMDb. OMDb API - The Open Movie Database. http://omdbapi.com/, 2014. Accessed: 2014-11-05.

Rotten Tomatoes. Rotten Tomatoes API. http://developer.rottentomatoes.com/, 2014. Accessed: 2014-11-05.

Rotten Tomatoes, Inc. Rotten Tomatoes. http://www.rottentomatoes.com/, 2014. Accessed: 2014-12-10.

Sayad, Dr.Saed. KNN Regression. http://www.saedsayad.com/k_nearest_neighbors_reg.htm, 2012. Accessed: 2014-12-10.

ScikitLearn. ScikitLearn Feature Selection. http://scikit-learn.org/stable/modules/feature_selection.html, 2014. Accessed: 2014-12-04.

Scrapy. A fast and powerful web crawling framework. http://scrapy.org, 2014. Accessed: 2014-11-05.

TMDb. The Movie Database API. http://docs.themoviedb.apiary.io/, 2014. Accessed: 2014-11-05.

Wikipedia. Lists of films. http://en.wikipedia.org/wiki/Lists_of_films, 2014. Accessed: 2014-11-05.