Predicting Movie Ratings using IMDb dataset

Akshay Sadarangani and Ashton Vaz
Association for the Advancement of Artificial Intelligence
publications16@aaai.org

Abstract

Filmmakers have been making movies since the 1920s and they have been an important form of recreation since. IMDb (Internet Movie Database), dubbed as the world's most popular source of movie content, provides ratings to movies on a scale of 0.0 – 10.0. Predicting the ratings of recently released movies that haven't been rated yet is of substantial importance in decision making. We plan to use Machine Learning algorithms including Naïve Bayes, SVM, Decision Trees and Random Forests for prediction. We will also explore the impact of social media popularity, and determine its role in the prediction of movie ratings.

Introduction

Movies are a universal source of entertainment, primarily as they can be watched by all age groups combined with the fact that they are made in countries all over the world in different languages. The size of the movie industry is increasing with the passing years, with plenty of new releases every week to choose from. IMDb is an online database of information related to films, television programs and video games, including cast, production crew, fictional characters, biographies, plot summaries, trivia, and reviews[1]. Every movie has a numerical rating ranging from 1.0 to 10.0 which represents how good the movie is.

Movies that have been recently released, however, do not have a rating. Nevertheless, IMDb does provide information about new releases. With so many such movies to choose from, deciding which movie to watch can be a strenuous task. Usually, we look at the actors and actresses starring in the film to decide if it's worth a watch or not. But that isn't a really feasible method of doing so, principally due to the number of "flops", which are essentially the movies that do not quite live up to the hype.

We would like to apply Machine Learning to solve this social dilemma. Machine Learning is the process by which a computer program searches through data to look for patterns, without being explicitly programmed to do so [2]. Our data will consist of the IMDb data set that includes 5000+ movies. We will split the data into training data and testing data in an 80:20 ratio in accordance with the Pareto principle [3]. We will be using Naïve Bayes, Support Vector Machines (SVM), Decision Trees and Random Forests to search the training data for patterns and predict the ratings of the movies in the testing data. These algorithms come under the class of Supervised Machine Learning algorithms.

Naïve Bayes

Given a class variable y and a dependent feature vector x_1 through x_n, Bayes' theorem states the following relationship:

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) P(x_1, \dots x_n \mid y)}{P(x_1, \dots, x_n)}$$

Using the naive independence assumption that

$$P(x_i | y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i | y),$$

for all i, this relationship is simplified to

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

Since $P(x_1, \dots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

Naïve Bayesian model is easy to build and particularly useful for very large data sets. Along with simplicity, Naïve Bayes is known to outperform even highly sophisticated classification methods [4].

Support Vector Machine

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data, the algorithm outputs an optimal hyperplane which categorizes new examples.

For a linearly separable set of 2D-points which belong to one of two classes, find a separating straight line.

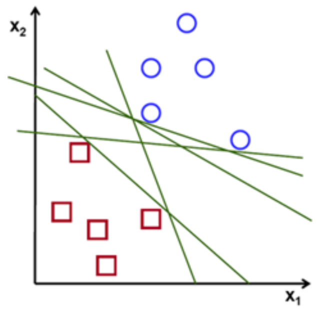

Figure 1: Finding a separating straight line for a linearly separable set of 2D points which belongs to one of two classes

The operation of the SVM algorithm is based on finding the optimal hyperplane, which is the hyperplane that gives the largest minimum distance to the training examples. Therefore, the optimal separating hyperplane maximizes the margin of the training data, as shown[5] :

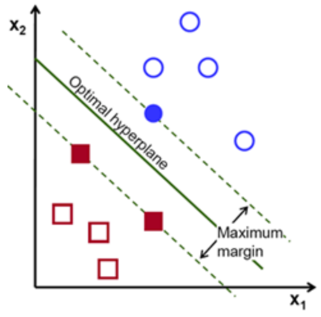

Figure 2: Finding the optimal hyperplane for a linearly separable set of 2D points which belongs to one of two classes

Decision Tree

Decision Trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features [6].

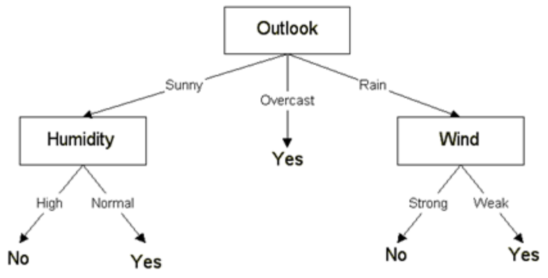

Figure 3: An example of a Decision Tree Classifier

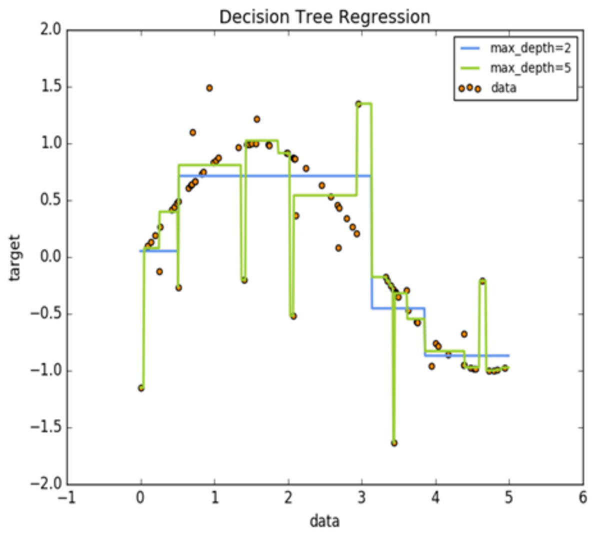

Figure 4: Decision Tree used to fit a sine curve with addition noisy observation, an example of decision tree being used for regression

Decision tree learning uses a decision tree as a predictive model which maps observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modelling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a finite set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.

Random Forest

Random forests are an ensemble learning method for classification, regression, and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set [7].

A random forest is a meta estimator that fits several decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement [8].

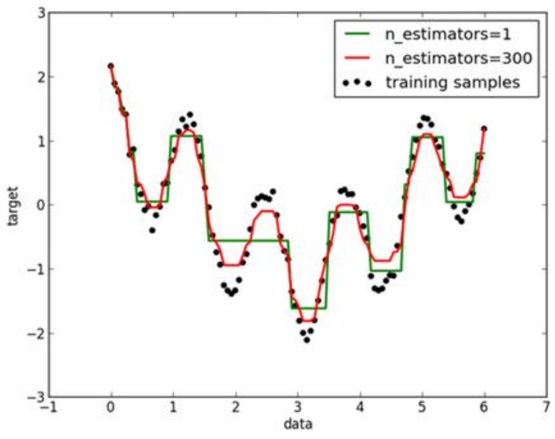

Figure 5: Random forest being used for regression analysis

Random forest is like bootstrapping algorithm with Decision tree (CART) model. Say, we have 1000 observation in the complete population with 10 variables. Random forest tries to build multiple CART model with different sample and different initial variables. For instance, it will take a random sample of 100 observation and 5 randomly chosen initial variables to build a CART model. It will repeat the process (say) 10 times and then make a final prediction on each observation. Final prediction is a function of each prediction. This final prediction can simply be the mean of each prediction [9].

Implementation

We obtained the IMDb dataset, which is a subset of the IMDb database from https://www.kaggle.com/deepmatrix/imdb-5000-movie-dataset. It included a list of 5043 movies, and we used 23 variables as features for our Machine Learning algorithms. We split the movies in a ratio of 80:20 for training data and test data. That totaled to 4033 movies as training data and 1010 movies as test data.

Next, we extract the data from the training dataset which is stored in a .csv format. We read the rows of the training data into a dictionary format in Python, which is a set of key: value pairs where the keys are unique. The keys here represent the variables and the values are the data stored in the respective column.

Now that the data is in readable format, we vectorize the data (except the IMDb scores) into ASCII format so that our algorithms can use them. We vectorize the data by adding the ASCII values of all the characters of the field. This, along with the actual IMDb rating, serves as our input to the algorithms.

We make use of the GaussianNB(), scm.SCV(), RandomForestClassifier() with 100 trees, and DecisionTreeClassifier() which are provided by the sklearn package. For our testing data, we convert it into a dictionary format, like how we processed the training data. We predict the IMDb scores of the movies in the testing data, and we calculate the average error margin.

Empirical Results

We cover a broad range of 23 feature values covering details such as movie name, director, actors, social media following, country, language, etc. Our evaluation function is based on the rating of the movie where we compare the rating predicting by our algorithms versus the actual rating. We have covered over 5000 movies and split them into two with 80% of the content being treated as training data while the other 20% of the content being treated as testing data.

For our success metrics, we conducted a set of tests covering various types of feature values and calculating the error margin for each algorithm. The error margin is calculated as: -

$$\frac{\sum predicted\ score - actual\ score}{n}$$

Passing the datasets through the Naïve Bayes, Decision Tree, and Random Forest algorithm gives us decent predictions, but we observe that SVM takes too much time to process and requires phenomenal computational power with large datasets. This is because the implementation is based on libsvm. The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to datasets with more than a couple of 1000 samples [10].

Feature Values: Using all 23 feature values

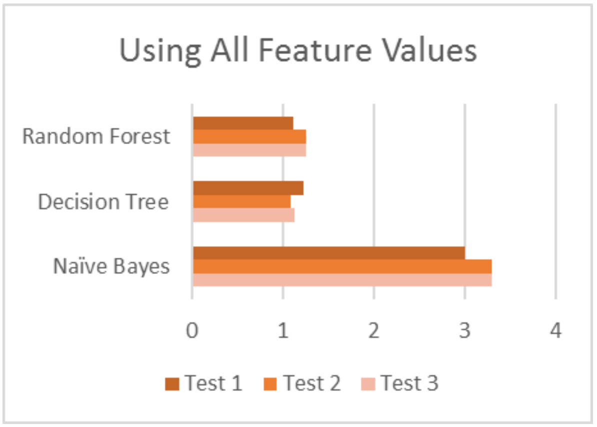

Figure 6: Error margin of algorithms for all feature values

Error Margin of Algorithm	Naïve Bayes	Decision Tree	Random Forest
Test 1	3.0	1.22	1.12
Test 2	3.30	1.09	1.26
Test 3	3.30	1.13	1.25

Table 1: Error margin of algorithms for all feature value

We observe that the error margin for Naïve Bayes is the maximum with an average variance of 3.0. Compared to the error margin of Decision Tree at 1.22 and Random Forest at 1.12, the error margin of Naïve Bayes is much higher. This is due to the behavior of Naïve Bayes and its use of prior probability which is computed even before the test data is observed. Thus, test data which belong to the categories of less likelihood tend to have a high error margin.

SVM couldn't be performed on this dataset, so we made subsets of our datasets (400 training data, 100 testing data) and ran it through the SVM algorithm. It took about 7 minutes to run, and returned an error margin of 2.3.

Feature Value: Without social media

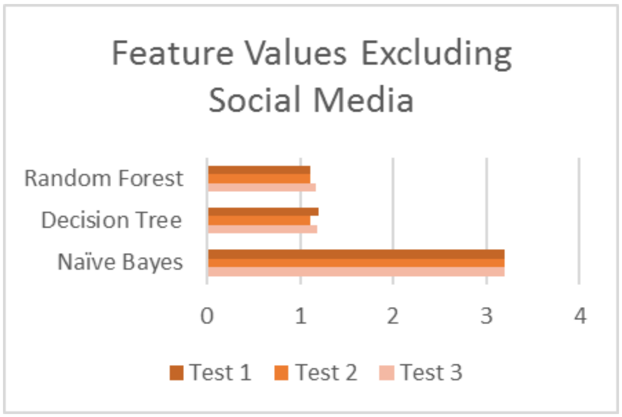

Figure 7: Error margin of algorithms for feature values excluding those related to social media

Error Margin of Algorithm	Naïve Bayes	Decision Tree	Random Forest
Test 1	3.19	1.20	1.11
Test 2	3.19	1.11	1.19
Test 3	3.19	1.11	1.17

Table 2: Error margin of algorithms for feature values excluding those related to social media

Excluding social media gives us an improvement in Decision Tree as well as Random Forest. This is because old movies tend to not have much social media hype even if they were good. An example of such a situation would be the hit movie The Godfather which is considered one of the best movies of all times. The Godfather from 1972 is rated 9.2 on IMDB while the number of IMDB Facebook likes for it is a few thousands. Since our dataset consists of movies starting from 1917 where social media was not even popular, the exclusion of social media is expected to lower error margin.

Feature Values: Names of actors and director

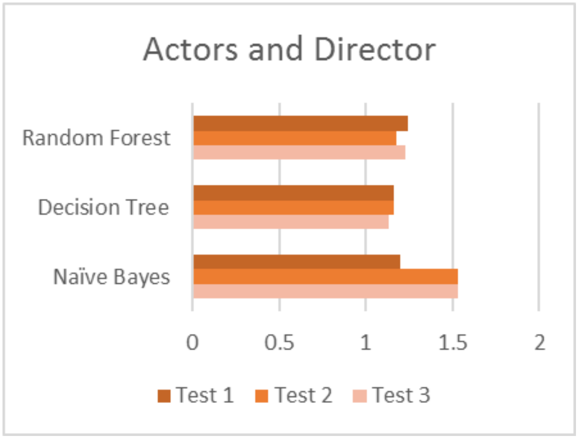

Figure 8: Error margin of algorithms for names of actors, and the director

Error Margin of Algorithm	Naïve Bayes	Decision Tree	Random Forest
Test 1	1.20	1.16	1.24
Test 2	1.53	1.16	1.18
Test 3	1.53	1.13	1.23

Table 3: Error margin of algorithms for names of actors, and the director

The range of error margin from 1.16 to 1.24 shows that the predictions made are quite close to the original score. This also helps us analyze the trend followed in ratings which in this case is that in general, movies with the same director and cast have similar ratings. We also see huge improvement in Naïve Bayes which is probably because of the way it works. In general, the data for director and cast is always available which prevents it from treating blanks as low values e.g. 0.

Feature Value: Movie title

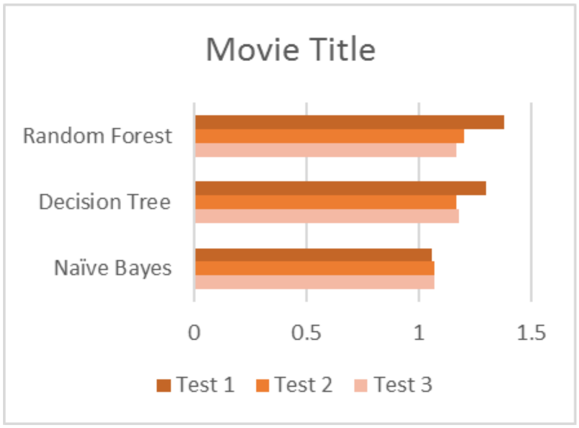

Figure 9: Error margin of algorithms for movie titles

Error Margin of Algorithm	Naïve Bayes	Decision Tree	Random Forest
Test 1	1.06	1.30	1.38
Test 2	1.07	1.17	1.20
Test 3	1.07	1.18	1.17

Table 4: Error margin of algorithms for movie titles

We observe huge improvement in Naïve Bayes again. This makes our assumption of blanks being considered as low values even stronger as all entries in the dataset have a movie title. In other words, there are no blanks in the movie title column.

Prediction based on movie titles alone could relate to movies with multiple sequels since they have related names

Feature Value: Country

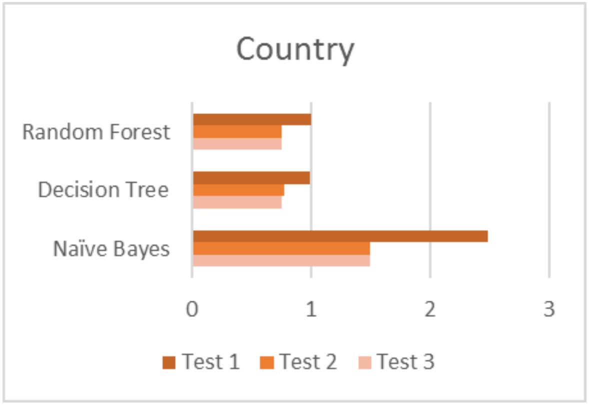

Figure 10: Error margin of algorithms for the country in which the movie is produced

.

Error Margin of Algorithm	Naïve Bayes	Decision Tree	Random Forest
Test 1	2.48	0.99	1.00
Test 2	1.50	0.77	0.75
Test 3	1.50	0.74	0.75

Table 5: Error margin of algorithms for the country in which the movie is produced

Random Forest and Decision Tree perform quite well here suggesting movies from the same country have similar ratings.

Feature Value: Language

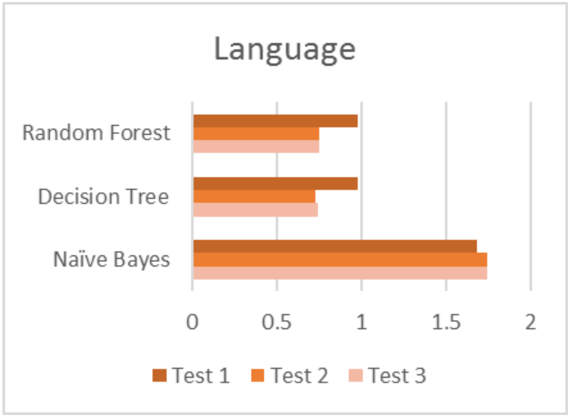

Figure 11: Error margin of algorithms for language of the movie

Error Margin of Algorithm	Naïve Bayes	Decision Tree	Random Forest
Test 1	1.68	0.98	0.98
Test 2	1.74	0.73	0.75
Test 3	1.74	0.74	0.75

Table 6: Error margin of algorithms for language of the movie

Just like the analysis based on country, the results of using only feature value language suggests that movies with the same language have similar ratings.

Conclusion

Predicting the IMDb ratings of a movie is an arduous task, but we can make decent predictions using Decision Trees and Random Forests. SVM is an accurate and widely-used classification algorithm but it isn't particularly useful for large datasets. We can overcome this by implementing Stochastic Gradient Descent (SGD). The gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength schedule (aka learning rate).

Naïve Bayes is known to be one of the best prediction models notwithstanding the fact that it is based on a flawed assumption. However, we observe that Naïve Bayes performs really poorly when there are blank fields, but with features which have minimal such fields with null values, Naïve Bayes' predicted values are not too far off from the actual values.

However, the most accurate predictions are given by Decision Trees and Random Forests. Decision trees implicitly perform feature selection, and require very little effort from users for data preparation [11]. Random forest produces a highly accurate classifier as it has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing [12].

References

[1] Wikipedia. 2016. *Internet Movie Database (IMDB)*. https://en.wikipedia.org/wiki/IMDb

[2] Rouse, M. February 2016. *Machine Learning*. http://whatis.techtarget.com/definition/machine-learning

[3] Investopedia, *Pareto Principle*. http://www.investopedia.com/terms/p/paretoprinciple.asp

[4] scikit-learn developers. 2016. *Naïve Bayes*. http://scikit-learn.org/stable/modules/naive_bayes.html

[5] OpenCV. December 2016. *What is SVM?* http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

[6] scikit-learn developers. 2016. *Decision Trees*. http://scikit-learn.org/stable/modules/tree.html

[7] Wikipedia. 2016. *Random Forest*. https://en.wikipedia.org/wiki/Random_forest

[8] scikit-learn developers. 2016. *Random Forest Classifiers*. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[9] Srivastava, T. June 2014. *Introduction to Random Forest.* https://www.analyticsvidhya.com/blog/2014/06/introduction-random-forest-simplified/

[10] scikit-learn developers. 2016. *C-Support Vector Classification.* http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[11] Deshpande, B. July 2011. *Decision Trees for Predictive Analysis* http://www.simafore.com/blog/bid/62333/4-key-advantages-of-using-decision-trees-for-predictive-analytics

[12] Radenkovic, P. Random Forest home.etf.rs/~vm/os/dmsw/Random%20Forest.pptx