Jessica Brennan, Kirsten Brown

Spencer Christensen, Zac Gallagher

Jace Sheu, Feargal Walsh

8 December 2016

COMP379 - 001

**Final Project Report - IMDB Score Predictor**

**Introduction:**

Our project implements a number of machine learning algorithms to analyze information about various popular movies released between 1916 and 2016. The objective of our machine learning model is to predict the ratings the movies received by users on the Internet Movie Database (IMDB). The model attempts to predict the movies’ IMDB score within the closest integer value using NumPy’s “round” method.

**Dataset Description:**

The dataset we utilized to train and test our model came from kaggle.com. The dataset includes information about several movies on IMDB, including movie titles, directors, genres, countries of origin, and the facebook popularity of the top three actors featured in the film. This information was provided in a variety of formats, including strings, integers, and floating point data. In order to implement the machine learning algorithms effectively and avoid underutilization of certain aspects of the movies provided in the dataset, the data was converted to numerical values using the Scikit learn preprocessing library to scale the features[[1]](#footnote-1).

The dataset was accompanied by a description by the author that detailed how he obtained the data from IMDB pages using the scrapy library[[2]](#footnote-2). The dataset was also accompanied by 149 kernels[[3]](#footnote-3) (at time of writing), which gave the group some direction in regards to how to approach the dataset with regards to preprocessing and algorithm implementation, for example the decision to use IMDB ratings as the predicted feature was based on previous attempts to do so in R.

**Baseline Approach Description:**

One of the key steps in training a model was first “cleaning” the data so that it could be used effectively by a machine learning model. After loading the data into a python dataframe, the first step in preprocessing the data was dropping all of the null values from each column so that each feature had a value in each row.

The following step represented the first major preprocessing challenge. Approximately half of the values were of either an integer or float type, which would be easy to input into a machine learning model. However, the other half of the values were in a string format, with some having unusual punctuation separating words in the string as well. Although our initial attempt to simplify the data involved dropping features that were presented in non-numerical formats, it was ultimately decided that altering the data into number format would be a more optimal solution as it would not prematurely alter the possible results of the model by limiting the data. The group utilized LabelEncoder via the scikit learn preprocessing library. This is used to normalize numerical data and transform non-numerical data to numerical data[[4]](#footnote-4). This was especially helpful, as a good portion of our data was non-numerical. Utilizing this allowed the group to use every column in the dataset.

In order to gauge the relevance of each feature of the dataset in relation to the others and the IMDB score, a correlation graph was implemented using the Seaborn python library. This graph provided a visual representation of the significance each feature in the data-frame contained in determining the IMDB score[[5]](#footnote-5). (see fig. 1).

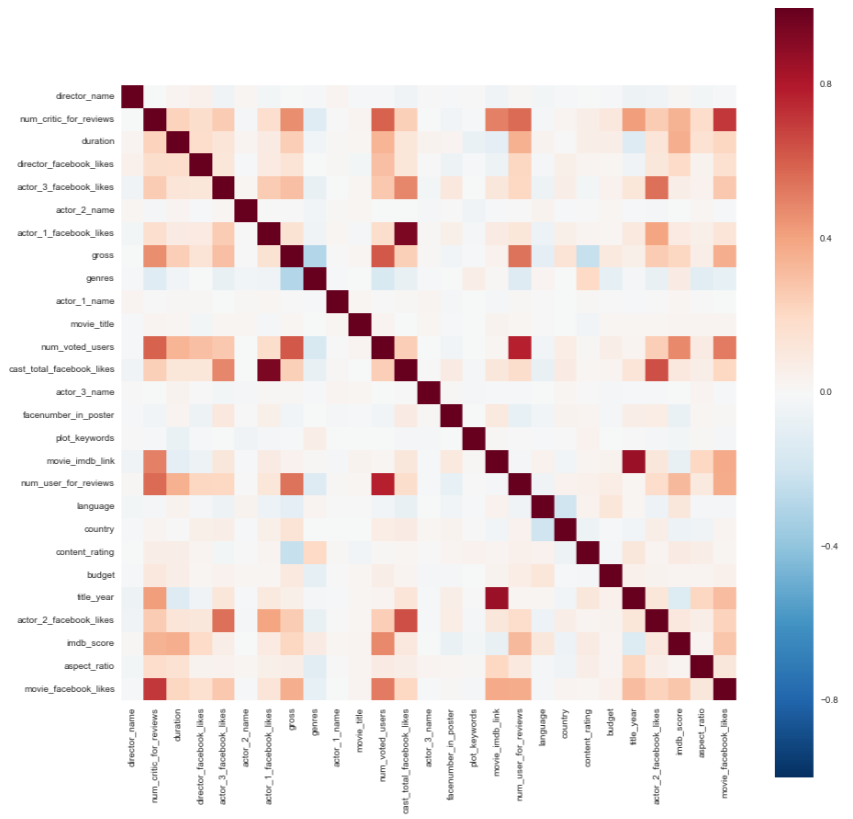


fig. 1: The correlation graph produced with the Seaborn library

**Method Description:**

Prior to implementing any machine learning algorithms on the preprocessed data, the data was divided into two sections using SciKit-Learn’s cross-validation library. All of the example data was split between the training set, which was composed of approximately 70% of the total number of examples, and the test set, which consisted of the remaining 30% of example data. These data sets were then randomized from the original data in 190 iterations by setting the “random\_state” to 190. This seemed to be the optimal number of iterations for the amount of data present in the dataset and the behavior of the various algorithms tested.

In building our model, a variety of different machine learning algorithms were implemented to predict the IMDB score of the various movies in the dataset. The first algorithm utilized was linear regression because it allows for scalar classification (i.e. the IMDB score does not have to be predicted as one of two classes)[[6]](#footnote-6). This was necessary as the IMDB scores of the movies range between the values of 1 and 10. This algorithm was also suitable for predicting the box office earnings of the movies, which was the original goal of the model before adjustments were made.

The second approach to implementing our machine learning model involved logistic regression, which in contrast to linear regression, is a simple binary classifier[[7]](#footnote-7). Due to the simplified nature of the possible predictions made by a model utilizing logistic regression, comparisons of the data in determining the the performance involved grouping the IMDB scores into two distinct categories. We chose to divide the IMDB scores into a category that consisted of scores below five (5) inclusively and a category that consisted of scores above five. This grouping of the scores into two categories allowed for the implementation of logistic regression.

In addition to allowing the model to implement logistic regression, the simplifying of the data to allow binary classification also enabled our group to test other algorithms that are limited to binary outputs. For example, support vector machines (SVM) were utilized in an attempt to maximize the accuracy and performance of the IMDB predictor. The SVM model contains four standard kernel options to control the shape of the decision function that ultimately determines which category a dataset example will be classified as. The four kernels include the linear kernel, polynomial kernel, sigmoid kernel, and the radial basis function (RBF) kernel, which is the default kernel used by SciKit-Learn[[8]](#footnote-8). Our model implemented the RBF SVM kernel, which scored the highest accuracy rating between the four kernels’ tests. The model was evaluated with a variety of C values, which was ultimately set to 0.5 to reduce overfitting and maximize accuracy.

The fourth machine learning algorithm tested in the development of our machine learning model was a decision tree, which uses past data fed into the model to make a list of possible outcomes. This non-parametric algorithm does not require a binary classification system, although it does need to simplify result in order to maintain efficiency[[9]](#footnote-9). This was done by testing the algorithm both with the binary categories outlined in previous paragraphs and rounded IMDB scores. The implementation of this algorithm was ineffective due to the large number of features provided in the dataset and the weak correlation between many of the features and the resulting IMDB score. The algorithm does not have a way to regularize the data to eliminate data feature entries that hold little to no relevance to the classification decision. Possibilities are built outward based on past examples and then a decision is reached by eliminating possibilities using past data.

After the moderately successful implementation of the Decision Tree model, the next algorithm utilized was K-Nearest Neighbors, which is also used as a binary classifier. The algorithm computes the number of nearest neighbors of each point data point and the one with the most, the classifier will select that as the outcome[[10]](#footnote-10). This was useful for the IMDB dataset when we rounded the data, so there is a clearer distinction between each data point. Some changes made to the algorithm from the default values was the p value, which was set to 7, and the nearest neighbors, which was set to 12.

The final algorithm tested was Random Forest. This algorithm uses trees and regression to decide the y value, or example class. The Random Forest classifier takes the various example features, constructs multiple trees, which collectively make up the “forest,” and outputs the label, or the IMDB score, that received the majority of votes[[11]](#footnote-11).It is useful for a moderate amount of possible outcomes, such as our data set, which contains possible integer outcomes between 1 and 10.

Although the classification range of the algorithms tested varied, all of the algorithms used the same features and attempted to predict the same y value: the example movies’ IMDB score. The algorithms were implemented through sklearn and used the same data. Parts of the algorithms were adjusted and altered, such as the independently set parameters of the machine learning models, but generally, the default settings were used for most of them.

**Evaluation:**

The performance metric used to evaluate our machine learning model was accuracy due to the ease it provides in understanding the different results of the various algorithms present and tested in the model. The range of classes the different algorithms were able to use for their predictions also influenced our choice to use accuracy as the performance metric.

The Linear Regression model performed the worst with an accuracy rating of only 35%. The K-Nearest Neighbors algorithm performed marginally better with a accuracy score of 43%. Logistic Regression and Decision Tree both score 50% accuracy rating, while the Random Forest and SVM algorithms performed with the best results. Random Forest achieved 53% accuracy and the RBF-kernel SVM model marginally outperformed all the other tested algorithms with a final accuracy score of 54%. In comparison, the Linear SVM only achieved 51% accuracy.

Due to the easy alteration and manipulation of machine learning model parameters provided by SciKit-Learn, our group was able to test a large number of algorithms’ performances and adjust the parameter accordingly in order to maximize results. The use of the same data-frame when providing the models with example features also led to the wide range of possible testing.

**Discussion:**

Overall, the most efficient algorithm utilized by our model was the SVM model implemented with the RBF kernel. The machine learning model allowed for overfitting consideration and reduction with adjustments to the C score[[12]](#footnote-12). Lower C values make the classifier’s convergence parameters stricter, which may cause the classifier to never finish making conclusions about the examples’ labels. By lowering the C value, classification analysis can be more lenient, which allows the classifier to label certain examples in opposition to the recommended label dictated by the training set. The value we set for our model that achieved the best results was 0.5. Similarly, the gamma value for the model remained at the default “auto” value, which is calculated as 1 divided by the number of examples (1/nexamples).

The reason for many of the other algorithm’s failures may have been due to the low correlation between features, as well as the large number of features each algorithm had to take into consideration; the SVM’s ability to limit the effect of certain irrelevant features on the classification decision was particularly imperative in producing better results than other algorithms due to the seemingly uncorrelated-nature of features present in our dataset. Another factor in determining the success of certain algorithms may be the diverse nature of the classification possibilities. Although a significant amount of effort was placed in ensuring that the classification of the examples was standardized and suited to the various algorithms implemented, including rounding IMDB scores to the nearest integer value and grouping scores into two categories to accommodate binary classifiers, the lack of optimization for each particular algorithm may have led to discrepancies in the algorithms’ ability to properly classify the example labels. For example, although decision trees are capable of classifying examples in multiple categories, it would still perform better with fewer possible classification labels.

**Conclusion:**

The IMDB dataset was a fun and interesting dataset to analyze. The group implemented and tested several algorithms in order to make meaningful predictions. Although the prediction of different y values were discussed, the group ultimately decided to work with and predict the IMDB score. Many of the algorithms achieved only marginal success, with the highest accuracy score being the RBF-kernel SVM’s 54%. The dataset was difficult to analyze with our models due to its large number of feature vectors that were presented in various formats. The data was also not highly correlated, which affected the performance of the algorithms tested.

**Appendix:**

Jessica Brennan - algorithm analysis, testing, and findings

Kirsten Brown - data preprocessing, documentation, and findings

Spencer Christensen - algorithm implementation and testing

Zac Gallagher - data preprocessing and algorithm implementation

Jace Sheu - data preprocessing, algorithm implementation, and presentation materials

Feargal Walsh - infrastructure, preprocessing, algorithm analysis, and findings

Presentation link - https://prezi.com/ernafvcc92cn/imdb-score-predictor/

1. http://scikit-learn.org/stable/modules/preprocessing.html [↑](#footnote-ref-1)
2. https://www.kaggle.com/deepmatrix/imdb-5000-movie-dataset [↑](#footnote-ref-2)
3. https://www.kaggle.com/deepmatrix/imdb-5000-movie-dataset/kernels [↑](#footnote-ref-3)
4. http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html [↑](#footnote-ref-4)
5. http://seaborn.pydata.org/ [↑](#footnote-ref-5)
6. http://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LinearRegression.html [↑](#footnote-ref-6)
7. http://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LogisticRegression.html [↑](#footnote-ref-7)
8. http://scikit-learn.org/stable/modules/svm.html [↑](#footnote-ref-8)
9. http://scikit-learn.org/stable/modules/tree.html [↑](#footnote-ref-9)
10. http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html [↑](#footnote-ref-10)
11. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html [↑](#footnote-ref-11)
12. http://www-bcf.usc.edu/~gareth/ISL/ISLR%20Sixth%20Printing.pdf [↑](#footnote-ref-12)