NETFLIX MOVIES DATA MINING

Laura Soto
University of Miami
lms338@miami.edu

ABSTRACT

This project aims to apply different data mining techniques to a Netflix movies dataset. The first part of the project tries to predict the IMDb rating of a movie given its general characteristics. The second phase of the project proposes a recommendation algorithm based on the movie description. The data was built by combining two different datasets retrieved from Kaggle. In order to predict the IMDb rating, this project proposes three different approaches, a decision tree classifier algorithm, a pruned decision tree classifier algorithm, and a K-Neighbors algorithm. The recommendation algorithm uses text similarities to identify the top 10 movies with the highest similarities in their descriptions. In order to build these models, the high-level language used was Python and its libraries. The classification models obtained are not successful, with an accuracy of 35%, 44%, and 46%, while the recommendation algorithm is successful.

1. INTRODUCTION

The motivation of my project is to explore some applications that data mining algorithms can have on a simple Netflix movie dataset. In fact, Netflix is considered to be the most valued media company in the world, with a company evaluation of over $164 billion and the one characteristic that has to distinguish Netflix from its competition is the fact that they have been able to make more successful TV shows and movies by identifying what their audience wants. The key to their success is the fact that Netflix collects data from its 151 million subscribers and implements data analytics models to discover customer behavior, buying patterns, and in general to make smart decisions. Therefore, Netflix's success highlights the value of data analytics. By using a simple dataset that only includes the general information of a movie, like a title, cast, director, runtime, IMDb score, and others, this project aims to analyze how much can be predicted from it. Additionally, it mimics a very simple recommendation algorithm, which only uses text similarities.

For the first part of the project, the IMDb rating is considered to be the class label, different classification algorithms are applied in order to try to create a model that would be able to predict the rating, just by knowing its general characteristics. Therefore, to be able to study if there is a clear relationship between the general characteristics of a movie, that is known before the movie is released, and the rating, so how liked would be by the audience. The IMDb rating is an online database that contains information related to films, television programs, and more. Its users can rate any film on a scale of 1 to 10 and the total is converted into a weighted mean rating. IMDb user ratings are reliable indicators of what a general movie audience or film critics thought about a movie. The classification learning models that are going to be used in order to achieve this first goal are a decision tree classifier, a pruned decision tree classifier, and k-nearest neighbors. The models would be tested by their accuracy score when classifying unlabeled instances.

The second phase of the project consists of creating a simple recommendation algorithm, that would look for similarities between the given movie description, and the other movies in the dataset. The text similarities in the description would be ranked, and finally, the ten titles of the movies with the most similarities would be displayed.

2. DATA

A new dataset was created for this project by merging two public Kaggle datasets. The new Netflix movies dataset contains 2689 instances, 14 attributes, and one class attribute. The attributes are both nominal and numeric and include: *Title, Year, Director, Actor1, Actor2, Actor3, Country, Language, Age, Genre1, Genre2, Genre3, Description,* and *Runtime.* The class attribute contains float values that represent the IMDb rating.

3. DATA PRE-PROCESSING

In order to build the data mining models, the first step in the process is to do some data preprocessing. Data preprocessing is the step where data gets transformed or encoded. Moreover, represents the transformation of raw data into an understandable format. During this project, we first need to create the new dataset, then clean the data, then check for missing values, encode nominal values into numeric, and last make sure that all the data can be used for training and testing the model.

3.1 Merging the data

In order to create our dataset, two Kaggle datasets were used, one contains information about Netflix TV shows and movies, and the second one information about movies on Netflix, Hulu, Prime, and Disney+. Both datasets are used and merged in order to create a more consistent and complete dataset. The first dataset contained 6234 instances and 7 attributes, while the second dataset contained 16744 instances and 9 attributes. Through a loop, we identify similar titles, and we store the values of the attributes that we would like to use on the new dataset. Finally, after identifying common instances the new dataset is created.

	Title	Year	Director	Cast	Country	Language	Age	Description	Genres	Runtime	IMDb
0	Inception	2010	Christopher Nolan	Leonardo DiCaprio, Joseph Gordon-Levitt, Ellen...	United States,United Kingdom	English,Japanese,French	13	In this mind-bending sci-fi thriller, a man ru...	Action,Adventure,Sci-Fi,Thriller	148.0	8.8
1	The Matrix	1999	Lana Wachowski,Lilly Wachowski	Keanu Reeves, Laurence Fishburne, Carrie-Anne ...	United States	English	18	A computer hacker learns that what most people...	Action,Sci-Fi	136.0	8.7

Fig. 1. Visualization of the head of the Netflix movies dataset

3.2 Data Cleaning

After the new data set is created the next step is to clean the data. During this step, different tasks are performed. The first tasks consist of cleaning the attributes that contain multiple values per instance, like the director or the cast attribute. During this step, only an established number of values are stored per attribute, reducing the number of the cast to only the main three actors, or the language to only one. For the description attribute, a loop was used in order to identify and only store the keywords.

	Title	Year	Director	Actor1	Actor2	Country	Language	Age	Genre1	Genre2	Genre3	Description	Runtime	IMDb
0	Inception	2010	Christopher Nolan	Leonardo DiCaprio	Joseph Gordon-Levitt	United States	English	13	Action	Adventure	Sci-Fi	mind subconscious thoughts mold bending sci ma...	148.0	8.8
1	The Matrix	1999	Lana Wachowski	Keanu Reeves	Laurence Fishburne	United States	English	18	Action	Sci-Fi	None	break free actually rebellion joins people per...	136.0	8.7

Fig. 1. Visualization of the head of the Netflix movies dataset after the data cleaning

3.3 Missing values

In order to deal with the missing values in the dataset, different approaches are used. First, because the class label is the IMDb score, the instances that contain missing values for this class label are removed from the dataset. Second, for the numeric attributes like *Runtime*, the missing values are filled in with the average of the non-missing values. Finally, for the nominal attributes that contain missing values, like *Actor2* and *Genre3* the missing values is considered to be a valid value.

Title	0
Year	0
Director	0
Cast1	0
Cast2	571
Country	0
Language	0
Age	1387
Genre1	0
Genre2	884
Genre3	1648
Description	0
Runtime	133

Fig. 3. Number of missing values per attribute

3.4 Nominal to numeric

In order to build a classification learning model, the Python Skirt-learn library is going to be used. On the other hand, the decision tree and k-neighbors algorithm implemented using this library uses only continuous numerical values. Therefore, the transformation of our data from nominal values to numerical values is needed. To achieve this transformation, we are going to apply the Hot Encoding technique, which creates a vector size with numerical values for each nominal value, excluding the *Description* attribute. By encoding our dataset each nominal attribute value is transformed into numeric. Finally, after performing this the data is ready to be used for creating the model.

	Title	Year	Director	Actor1	Actor2	Country	Language	Age	Genre1	Genre2	Genre3	Description	Runtime	IMDb
0	983	2010	374	1015	872	62	11	0	0	1	16	mind subconscious thoughts mold bending sci ma...	148	8.8
1	2231	1999	1106	929	1003	62	11	3	0	17	21	break free actually rebellion joins people per...	136	8.7

Fig. 4. Visualization of the head of the encoded dataset

The description attribute contains value phrases. Therefore, each description is transformed into a vector using the python sklearn feature extraction text function. A numeric vector is created for the description of each movie. In order to find the vector co-ordinance, the cosine similarity is used.

3.5 Class Label

The class label is composed of float values from 1.0 to 10. For the purpose of this project, these values were classified on a grading scale of A, B, C, D, and F. The first step is to normalize the data using the diving by maximum value method. The following step is to establish the limits for each grading and through a loop converting all the float values into the grading scale. Different grading distributions were analyzed, and the one that had the highest overall performance was selected. The selected distribution for the class label is represented as followed:

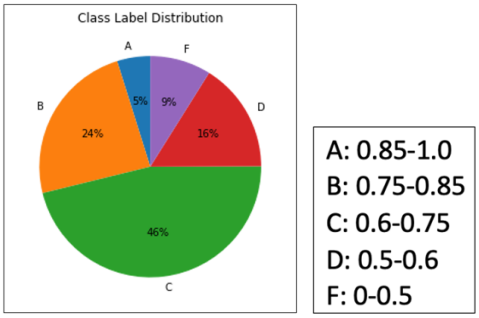


Fig. 5. Visualization of the class label data distribution in

4. CLASSIFICATION ALGORITHMS

4.1 Plot Analysis

The scope of the use of classification algorithms is to be able to classify the performance of a movie, evaluated in the IMDb rating grading scale, by analyzing the movie's general characteristics. The first step was to perform some plots analysis in order to identify the general characteristics of the highest-ranked movies:

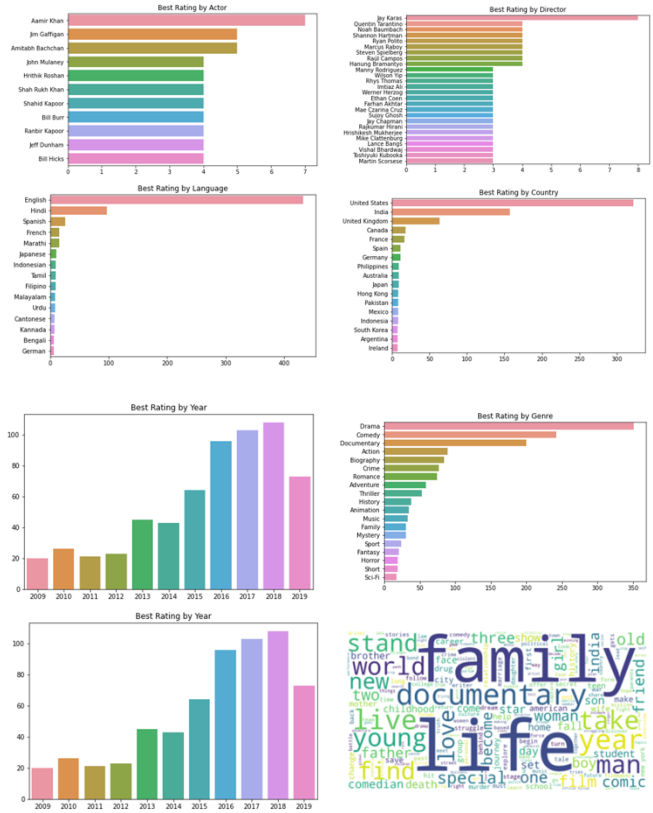


Fig. 6. Best ranking by attribute plots

As we can observe, we could conclude by looking at these plots that some characteristics have a higher probability of having a higher rating than another value. For example, looking at the 'Best Rating by Genre' plot, we could say that a drama movie has a higher probability of having a higher IMDb ranking than a Sci-Fi movie. So, from these plot analyses, we could conclude that there are some values per attribute that would most likely have a better ranking, than others. Now, is time to see if these probabilities that can be observed on the plots are reflected on the classification algorithms.

4.1 Decision Tree Classifier Model:

Decision tree learning is a predictive model used in data mining. This approach uses a decision tree to go from the observations/conditions to the conclusions. In order to build a decision tree classifier, three basic steps are followed; first, the dataset has to be split into testing and training, second, the model has to be trained using the training set, and third, the model has to be tested on the testing set.
The data is split it 80% for training and 20% for testing. The data is randomly distributed, and the classification decision three is created by fitting the training set into the model. This model uses a rule-based approach, which tries to identify the relationship between the conditions, and the conclusions present on the given data. That is why the conclusion of this process, is a decision tree model that has the rules/relationships founded from the training data as branches and the class label as the leaves. The results of our model are:

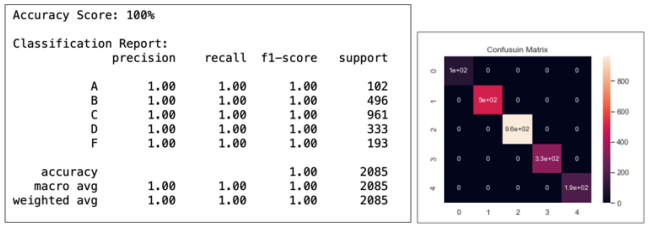


Fig. 6. Results for the training set for the Decision Tree Model

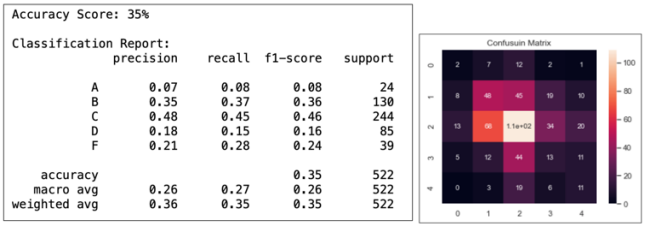


Fig. 7. Results for testing for the Decision Tree Model

As we can observe, the accuracy obtained when predicting the training dataset is 100%, while the accuracy when predicting the testing dataset is 35%, which is very

low and therefore we can consider this model to fail, and to be overfitted because it only performs well on known data. In fact, as we can observe by the tree created on this model, it is very specific, and it is not viable to predict unknown instances.

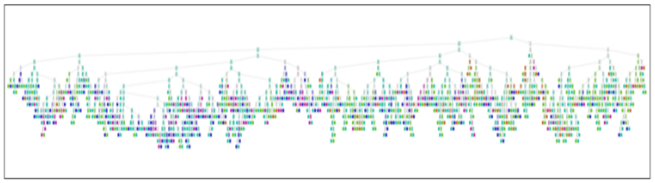


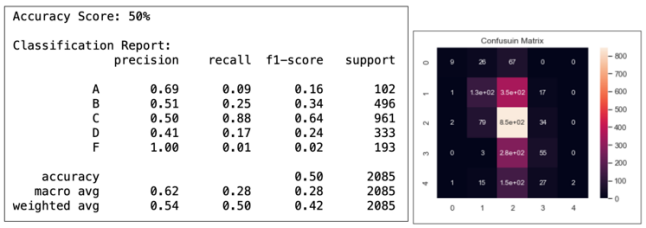
Fig. 7. Tree created from the first model

4.2 Pruned Decision Tree Classifier Model:

A pruned decision tree classification model approach is used, where a model is created, following the same steps used for the first model, but the main difference is that the model performs pre pruning with a depth equals to 5. Therefore, the remaining tree looks like the following:

Fig. 7. Tree created from the pruned model

And we can evaluate its performance on the training and the testing set as follows:

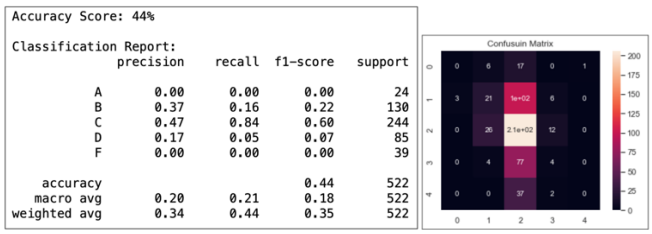


Accuracy Score: 50%

Classification Report:

	precision	recall	f1-score	support
A	0.69	0.09	0.16	102
B	0.51	0.25	0.34	496
C	0.50	0.88	0.64	961
D	0.41	0.17	0.24	333
F	1.00	0.01	0.02	193
accuracy			0.50	2085
macro avg	0.62	0.28	0.28	2085
weighted avg	0.54	0.50	0.42	2085

Fig. 8. Results for the training set for the Pruned Model



Accuracy Score: 44%

Classification Report:

	precision	recall	f1-score	support
A	0.00	0.00	0.00	24
B	0.37	0.16	0.22	130
C	0.47	0.84	0.60	244
D	0.17	0.05	0.07	85
F	0.00	0.00	0.00	39
accuracy			0.44	522
macro avg	0.20	0.21	0.18	522
weighted avg	0.34	0.44	0.35	522

Fig. 9. Results for the testing set for the Pruned Model

On this pruning tree model, the accuracy obtained when classifying unknown instances is equal to 44%, and it's greater than the one obtained on the first model. On the other hand, the accuracy obtained on the training performed worse, with an accuracy of only 50%. Therefore, even though this model performs better than the first one, and it is not overfitted, it is still considered to fail when predicting the rating grade of an unknown movie.

4.3 K-Nearest Neighbors

The k-nearest neighbors' algorithm is a type of instance-based learning used for classification. Given an unclassified instance, it looks at the k closest examples in the training set and outputs the most common class among its k nearest neighbors. To determine how many neighbors to consider, we evaluate the accuracy obtained by different k values varying from 1 to 100, and we selected the value that obtains the highest accuracy.

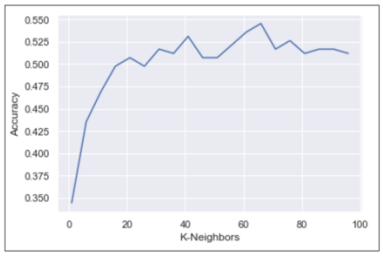
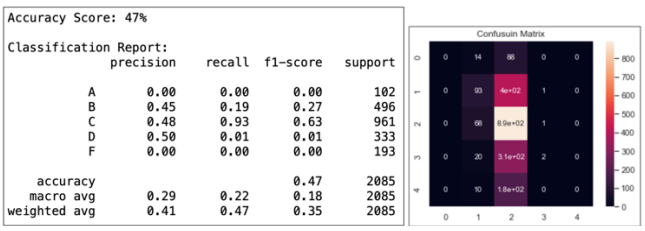


Fig. 10. Accuracy obtain for different neighbors' values

After performing this analysis, we found that the best value for k is equal to 66. Therefore, for this project, we created a k-nearest neighbors' model that considers the 66 nearest neighbors. The data was split into training and testing with a distribution of 80% and 20%. After the data is split, the model is trained using the training set, and then tested on both the training and the testing set. Its performance can be observed as follows:

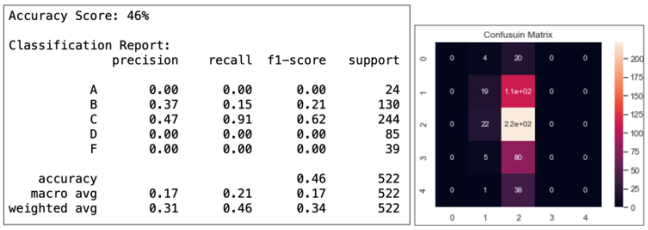


Accuracy Score: 47%

Classification Report:

	precision	recall	f1-score	support
A	0.00	0.00	0.00	102
B	0.45	0.19	0.27	496
C	0.48	0.93	0.63	961
D	0.50	0.01	0.01	333
F	0.00	0.00	0.00	193
accuracy			0.47	2085
macro avg	0.29	0.22	0.18	2085
weighted avg	0.41	0.47	0.35	2085

Fig. 11. Results for the training set for the KNN Model



Accuracy Score: 46%

Classification Report:

	precision	recall	f1-score	support
A	0.00	0.00	0.00	24
B	0.37	0.15	0.21	130
C	0.47	0.91	0.62	244
D	0.00	0.00	0.00	85
F	0.00	0.00	0.00	39
accuracy			0.46	522
macro avg	0.17	0.21	0.17	522
weighted avg	0.31	0.46	0.34	522

Fig. 12. Results for the testing set for the KNN Model

The accuracy obtained for the training set is 47% while the accuracy for the testing set is equal to 46%. Even if this model performs better than the previous ones when classifying unknown instances, it still has a very low accuracy and therefore it is also considered to fail.

5. RECOMMENDATION ALGORITHM

In order to create a simple recommendation algorithm, different approaches were considered. Finally, the selected approach for this project consists of recommending the top 10 movies that have the most similarities in their descriptions.

The first step to achieve this algorithm is to assign a score to each description based on its keywords. These scores are evaluated using cosine similarity. After the scores are stored for each movie, the algorithm is created. The algorithm consists of a function that takes a movie title as the parameter, which has to be included in the dataset, then it creates a series of movie indexes that have a similar cosine score, and inputs them in descending order. Only the index of the 10th most similar scores is stored, and the titles of the movies are printed.

Therefore, by imputing a movie title, 10 recommended movies would be displayed, which are the most similar movies based on their description. Additionally, a word cloud graph is included, in order to show all the common words that these movies share in their description and observed why they were recommended.

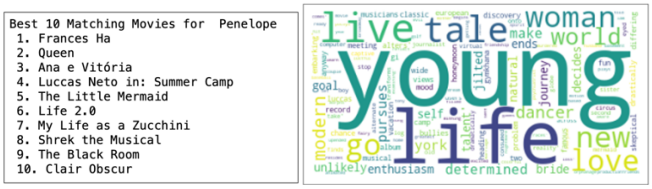


Fig. 13. Example of the output for the movie 'Patria'

6. CONCLUSION

In conclusion, the project was successful to apply different data mining techniques to a Netflix movies dataset. On the other hand, the results obtained for the classification part of the project were not ideal and can let us conclude that it is not possible to predict the IMDb rating giving very general information about a movie. In fact, what we can analyze from this is that how a movie is perceived differs on each user experience with the movie, as well as each user's background. Therefore, to create an accurate and effective rating predictor more of the human user data would be needed. Moreover, we can conclude that the text similarities recommendation algorithm was successful in identifying similar titles by its descriptions. Finally, what we can learn from this project, is that the success of a movie goes further of its general characteristics and therefore, in order to create a more successful model or a more accurate recommendation algorithm, knowing the user is key, and this is why Netflix data analytics is focused on understanding each user.

7. REFERENCES

Bansal, S. (2020, January 20). Netflix movies and TV shows. Kaggle. https://www.kaggle.com/shivamb/netflix-shows?select=netflix_titles.csv

Bhatia, R. (2020, May 22). *Movies on Netflix, Prime Video, Hulu and Disney+*. Kaggle. https://www.kaggle.com/ruchi798/movies-on-netflix-prime-video-hulu-and-disney

Dixon, M. (2020, July 3). *How Netflix used big data and analytics to generate billions*. Selerity. https://seleritysas.com/blog/2019/04/05/how-netflix-used-big-data-and-analytics-to-generate-billions/

Thakur, P. S. (2020, June 18). *How does IMDb work? IMDb Ratings Algorithm Decoded*. The Subsequent. https://www.thesubsequent.com/entertainment/how-does-imdb-work/