CS 484 Report

Movie Multi Label Classification

By:
Naassom Rocha **01039313**
Aster Bodden Pineda **00964499**

Table of Contents

Introduction

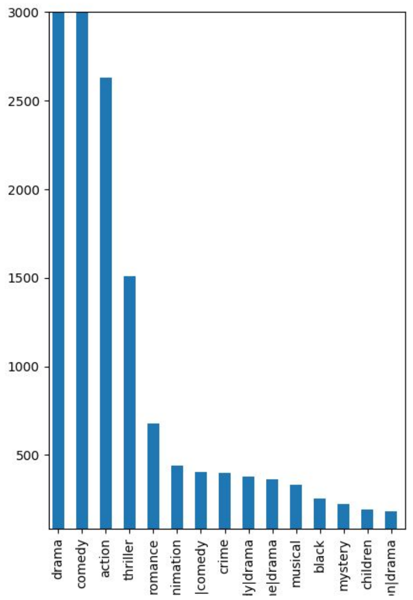
When my partner and I first started looking for a data set to use we encountered a wide variety of data set themes and challenges. Many of the data sets involved medical, public affairs, and consumer data, while these topics were of interest we wanted to have some knowledge of the data we were going to use. As we continued we narrowed our search to classifications problems that involved the entertainment industry. Finally, in the Kaggle website, we found a data set on information scraped from a movie's wiki page. The features in the data had the release year, director, actors, origin and the most prominent feature the movie plot. Besides the movie plot, all other features are categorical string values or unknown. Each movie in the data set has zero, one or more labeled genres that come from the movies corresponding wiki page. The data consisted of a single table of 35,000 movies with unknown values in some features. Our goal for this project is to build a model using a subset of the movies provided to determine the genres of movies the model has not seen before. This report will show the process of choosing and testing different classifiers to create the former model.

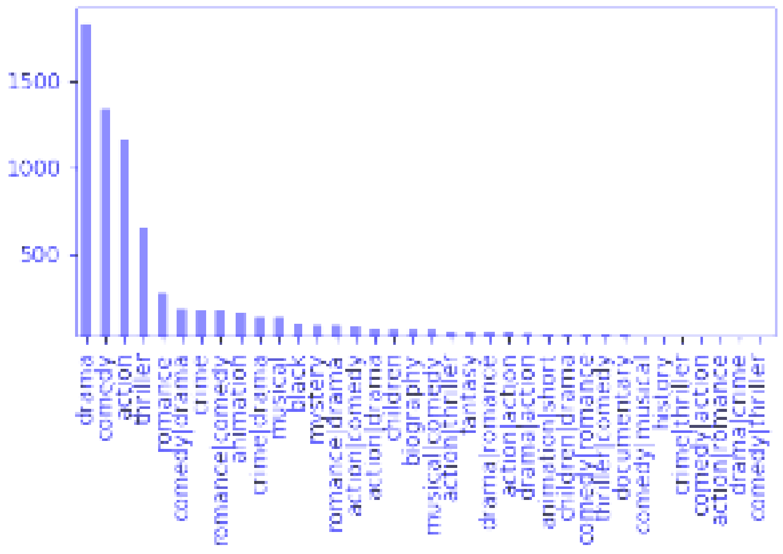
Constructing Train and Test data:

The original Wikipedia data set consisted of only one table we had to split into a training subset and a testing subset. Using scikit learn we were able to split the data into 70%-30% subsets and using data frames we removed and saved the test labels into a separate table. Both subsets were shuffled before splitting to ensure that the data was well distributed between the two subsets. The label (Genres) attributes were removed since we won't be able to test the model's performance on those values.

Data Set Visualization:

Train v. Test Distribution





Problem Definition

The model we are trying to build requires a classification model that can determine the genre of movies from the data set features. This task differs from a regular classifier since a movie can be described as having several genres. This fact alone limits the type of classifiers that can be used since most classifiers do not support multilabel classification. Also as we will further discuss it is harder to evaluate the model's performance since for a given prediction the model could predict correctly half of the genres or all of them. The data set also further adds problems to the classification because it contains around 2,000 different genres of which many are compound or have very few instances in both the training and test data set. As we have experimented in other class assignments classifiers have a harder time learning the data when they have to fit imbalanced data or a large number of labels to classify. The above is something we have to think when implementing the model and evaluating its performance. The following paragraphs will describe how we tried to overcome these problems both in the pre-processing stage as well as the classifiers we chose to use.

Motivation

Our main motivation is to make a model that can predict multiple labels for a single instance of test data. My partner and I wanted to learn how such classifiers would work when predicting movie genres based on the information obtained in their wiki page. This sort of classification problems is of interest because it reduces the time for a movie to be classified and the users who watch a movie know what movie they are watching based on the contents of the plot. This also is an interesting classification problem that could be applied to different circumstances where the objects being classified belong to several classifiable groups.

Background Information

After reading several research papers regarding document multi-label classification we found that most papers had the same view as in Pieters, Wiering (2018) where they describe the problem as " Multi-label classification is considered to be significantly more difficult, due to the vast amount of possible label combinations"(pg.2). In their paper, Pieter and Wiering were able to show that by combining their baseline Naive Bayes classifier with other classifiers by using a basic neural network they were able to improve their results substantially. In their conclusion, they demonstrate that with text classifications the type of problem will determine how effective the classifying model will work. The paper recommends using simple classifiers since they might often outperform complex algorithms and their combined predictions may considerably improve the performance of the classifier. By reading papers such as Read, Jesse, et al (2011) we discovered how labels can be encoded into binary values and each label prediction can be independent of the other. In their paper, Jesse, et al describe that even though binary relevance algorithms do not take into consideration the label correlations their scalability and flexibility make them still useful to use. In other words, they describe that complex classifier method for multi-label classification "restricts their usefulness as many multi-label contexts involve large numbers of examples and labels"(pg. 255) making them nearly impossible to run on large data sets. The former paper also introduced an ensemble of classifiers that predict on the object's label and the popular vote is determined in the final prediction. The paper describes this method as:

"These predictions are summed by label so that each label receives a number of votes. A threshold is used to select the most popular labels which form the final predicted multi-label set"(pg. 258)

With this knowledge on how multi-label classifiers work we could start doing the preprocessing and model building for our voting classifier model.

Preprocessing and Feature Reduction

The data set we obtained from Kaggle consisted of 4 string categorical feature (Title, Origin, Director and Genre) 1 numerical value(release year) and in paragraph form the movie plot. Starting with the movie plots we had to use several feature reduction techniques before fitting the data in the classifier. All of the movie plots where stemmed to their root words by using NLTK Porter Stemmer and at the same time reduce the word to lowercase and remove all the special characters such as (-, ' and HTML links"). The movie plot data was later transformed to a document vector and its dimensions were reduced to around 250 features using Truncated SVC. On the other hand, the string categorical features had to be encoded into numerical values for them to be processed by some classifiers that only work on numbers. We found that one of the reasons we had a large number of genres was because some sub-genres were too specific and could be grouped to their general genres. One Kaggle user Amine Jallouli solved this by "harmonizing" the data with regular expression statements that substituted sub-genres for general genres. Using regular expressions all of the movies that had several genres were harmonized to use the same separator. Since we are implementing a binary relevance voting classifier we encoded them using multi-label binarizer in scikit learn. Finally, we tried reducing the number of genres further by filtering the genres that had low frequency in both the train and test data, but the result was not as good as expected.

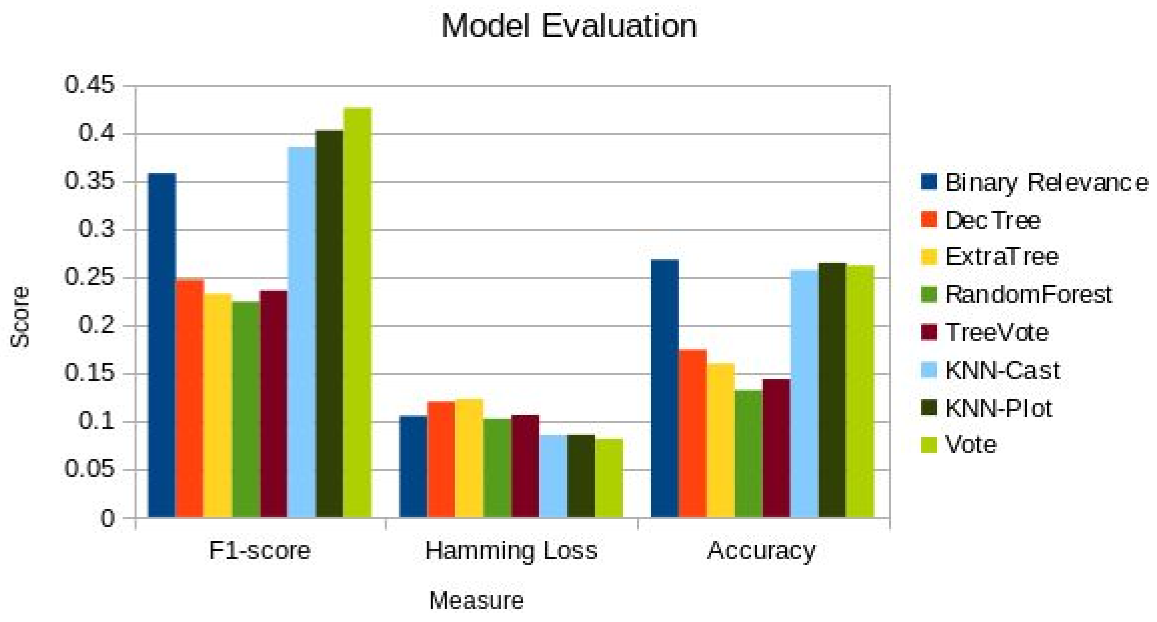
Model Building

For classification, we experiment with Decision Trees and KNN. We wanted to also experiment with Naive Bayes, but from our research, we could not find a Naive Bayes Classifier that supports Multi Label classification. We needed to find which features are useful. Some we could easily discard like Wiki Page. Some made no sense to keep such as Title because no two movies would have the same title. To get our results we created our own voting function that took the predictions from both classifiers weighted them and only kept the predictions that were a majority. We made two KNN classifiers, one for cast and one for plot. It made sense to us to test each feature separately and not let them influence each other. We ran into some problems with KNN. First of all with a KNN > 1, the averaging of the closest points frequently returned a label with no genre or only one genre instead of multiple genres. We figured this was because of some averaging within the library's classifier. So we tried to get KNN = 1 to have the most labels. This was obviously not great for accuracy or F1 but did not have a bad hamming loss. Surprisingly our KNN on cast worked much better than on plot. This could because the TFIDF vectors had too many features in the plot. Aster worked Binary Relevance, which I do not understand as much but it seems to have performed well in comparison to KNN. The big challenge of this project was working with multiple labels. It makes it hard to evaluate which makes it harder to improve the model. We will talk more about this in the Model Evaluation section. Decision Trees performed below average, so I tried to use different versions of the decision trees and vote among them. We joined together votes from sklearn's DecisionTreeClassifier, RandomForest, and ExtraTreesClassifier, weighed them equally and got a resulting vote from Decision Trees. We also had to use sklearn's Label Encoder to make strings into ints for classification.

Evaluation

We have three evaluation functions Accuracy, F1-Measure, and Hamming Loss. We are using accuracy to see how many of our predictions are exactly correct. F1-Measure is the score we are trying to improve. However, Hamming Loss is the best measure to use for Multi-Label Classification. We had low Hamming Loss, ranged from 5% to 10%. However there is a problem, because the labels are similar to a sparse matrix with only a couple labels out of the total number of genres being set to 1 for being that genre. We do not know the exact percentage, but guessing no genre for all of the movies would give a low score, probably low 20s. Therefore

Hamming Loss is not a great measure either. With more time, we would experiment with Micro Averaging and Macro Averaging which we found in a towards data science article. However, a workaround was to decrease even more the number of genres. We found that by removing genres with less than 100 movies with that genre, we still have 23000 movies. The resulting number of genres is 27. That increases the error from measuring Hamming Loss dramatically because we decreased the number of genres from 2265->1198->27. After reducing the number of labels we were able to increase our K-value in KNN. Also as a result, our Hamming Loss ranged from 9%-15%. Below are some of our results for tests.



Conclusion

Overall, we still believe there is much more work to be done. We need to research Multi Label Classification more deeply because although our Hamming Loss is small < 10% error, Our F1-score is only 42%, and accuracy is 25%. The results aren't bad for Multi Label but can be improved. We were able to increase our F1-Score about 7% from 35% to 42%. Decision Trees performed the worst, KNN was much better. Plot was most surprising in our conclusions because we believed it would have the best results by far because they have the most relevant information. Surprisingly, cast information was just as good at indicating genre. The results for plot mirror my results for HW1, more preprocessing needs to be done, and more research on improving Multi Label score (accuracy, F1-score). We are satisfied with our results because of the small Hamming loss error, which indicates our predictions are 89% correct.

Citation:

Pieters, Mathijs, and Marco Wiering. “Comparison of Machine Learning Techniques for Multi-Label Genre Classification.” Communications in Computer and Information Science Artificial Intelligence, 2018, pp. 131–144., doi:10.1007/978-3-319-76892-2_10.

Read, Jesse, et al. “Classifier Chains for Multi-Label Classification.” Machine Learning, vol. 85, no. 3, 2011, pp. 333–359., doi:10.1007/s10994-011-5256-5.

“1.12. Multiclass and Multilabel Algorithms” Scikit, scikit-learn.org/stable/modules/multiclass.html.

Reducing the number of labels using regular expressions::

Jallouli, Amine. “Genre Classification Based on Wiki Movies Plots.” Kaggle, www.kaggle.com/aminejallouli/genre-classification-based-on-wiki-movies-plots.