Movie Genre Classification using IMDB Data

SENG474 - Fall 2019
Department of Computer Science, University of Victoria

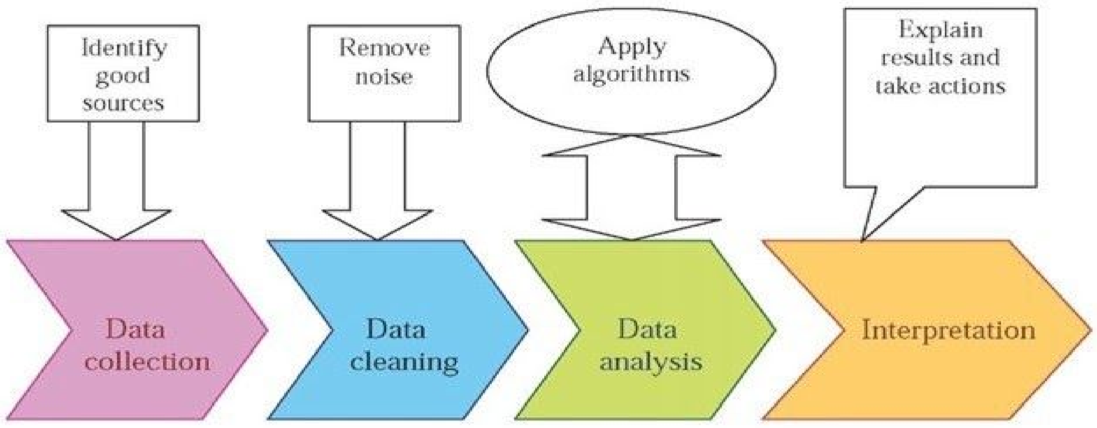
For: Professor Alex Thomo

By: Gursewak Singh (V00875106)
Jaskaren Saini (V00877182)
Amrit Rajpal (V00882331)

Table of Contents

Prelude

Whenever movie maniacs stream films on streaming services like Netflix or Disney Plus, they always like to jump to movie genres they absolutely love. Hence the providers are always keen to filter out the movies on the basis of genres so as to make it easier for users to browse through. Tagging of movies is a complex procedure that often involves the manual process of assigning movies to different genres, sometimes more than once, on the basis of customer feedback and suggestions. Automating this process of labeling the movies based upon the genres will save enormous human efforts and will be quick at the same time. Nevertheless, the process has a huge advantage of accuracy over an individual.



1. Introduction

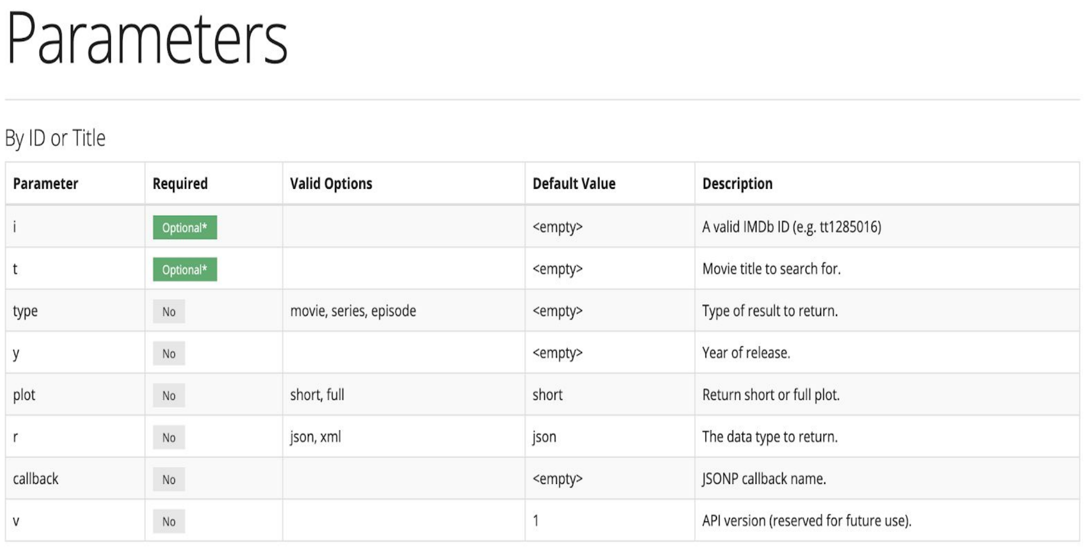
Movie genre prediction is a problem of multilabel classification where a case can arise when a movie needs to be labeled under two or more genres. By the end of this project, we will have used various data mining and machine learning procedures to predict if a movie belongs to some particular genre or not. The data to be used will be collected from an API available over the internet, and then a data set will be compiled which will be primarily based on the IMDB data. Our procedure will simply rely on the text analysis of movie plots or summaries from the collected movie data and will use analysis techniques to train the classifier.

2. Related Work

“Movie Genre Classifier using Neural Network” is a paper published by Sanjay K. Jain to detect movie genres through the application of the neural network. The classifier in the above paper is designed through feed forward neural network and demonstrates its effectiveness for classification into action, comedy, horror, music and drama genres. The dataset we have used has been created by us ourselves, and we have used a web API to fetch it from the internet. We used some ideas from this paper as the methods used in this paper are neural networks, logistic regression and back propagation learning algorithm.

3. Data Collection

In order to collect and build a data set for this project, we used the OMDB (Open Movie Database) web API. OMDB is a RESTful web service that helps seek information on movies and series. The API returns all the data about a movie or a series in JSON/XML format, given the IMDb id or the name of the movie/series. For the scope of this project, we restricted our dataset to strictly movies. In order to fetch data from the OMDB API, we created a python program, which retrieves data from the API in JSON and exports the relevant information regarding each movie to a CSV file. For this task, we started with a title ID and sequentially incremented the title ID, until a considerable dataset was achieved.



Parameters

By ID or Title

Parameter	Required	Valid Options	Default Value	Description
i	Optional*		<empty>	A valid IMDb ID (e.g. tt1285016)
t	Optional*		<empty>	Movie title to search for.
type	No	movie, series, episode	<empty>	Type of result to return.
y	No		<empty>	Year of release.
plot	No	short, full	short	Return short or full plot.
r	No	json, xml	json	The data type to return.
callback	No		<empty>	JSONP callback name.
v	No		1	API version (reserved for future use).

3.1 Features Selected

The OMDB, API return a large number of fields/parameters for each movie, however, for the scope of this project we will restrict our outlook to just the following parameters per movie :

- Movie ID: (IMDB ID)
- Plot
- Genre: (Up to three)

In the achieved data set, there are movies that have more than one genre associated with them. The genres could be as much as five genres per movie. However, we filter our results to three genres per movie and select the first main genre for our classification.

3.2 Pruning

For the purpose of filtering and pruning data, we created a filter which only added movies to our dataset with the following characteristics:

- Dataset type = ‘Movie’
- The length of the plot was greater than 50 characters to yield better results.
- The movie should have at least one genre. (Excluding cases with ‘N/A’ as a genre)

4. Data Processing

The programming language of choice for collecting and analysing data was python. The primary reason for choosing python was its ease of use and the abundance of various different libraries such as numpy and pandas which make data manipulation convenient. After filtering the movie data through the above mentioned pruning process we obtained 3504 movie titles across 19 different genres.
After the data was collected, we removed numbers, special characters and punctuation using regex in python.

Comedy	Horror
Action	Fantasy
Drama	Mystery
Animation	Romance
Documentary	Sci-Fi
Adventure	Family
Biography	Thriller
Western	Short
Crime	War
Musical	

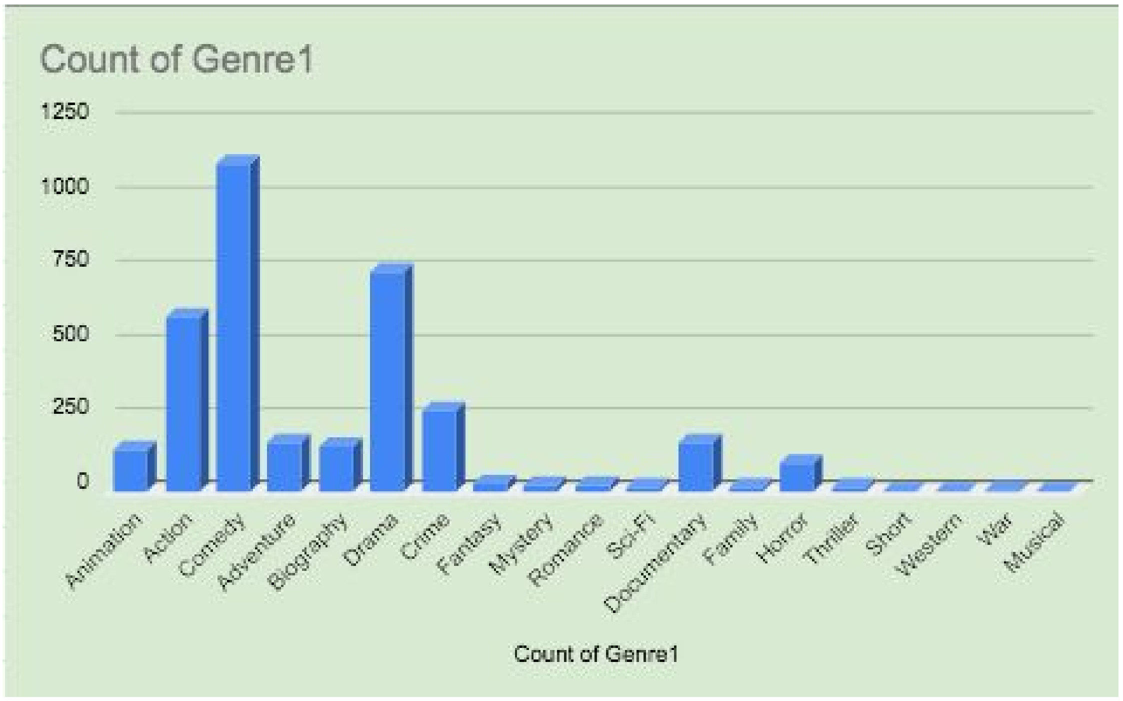
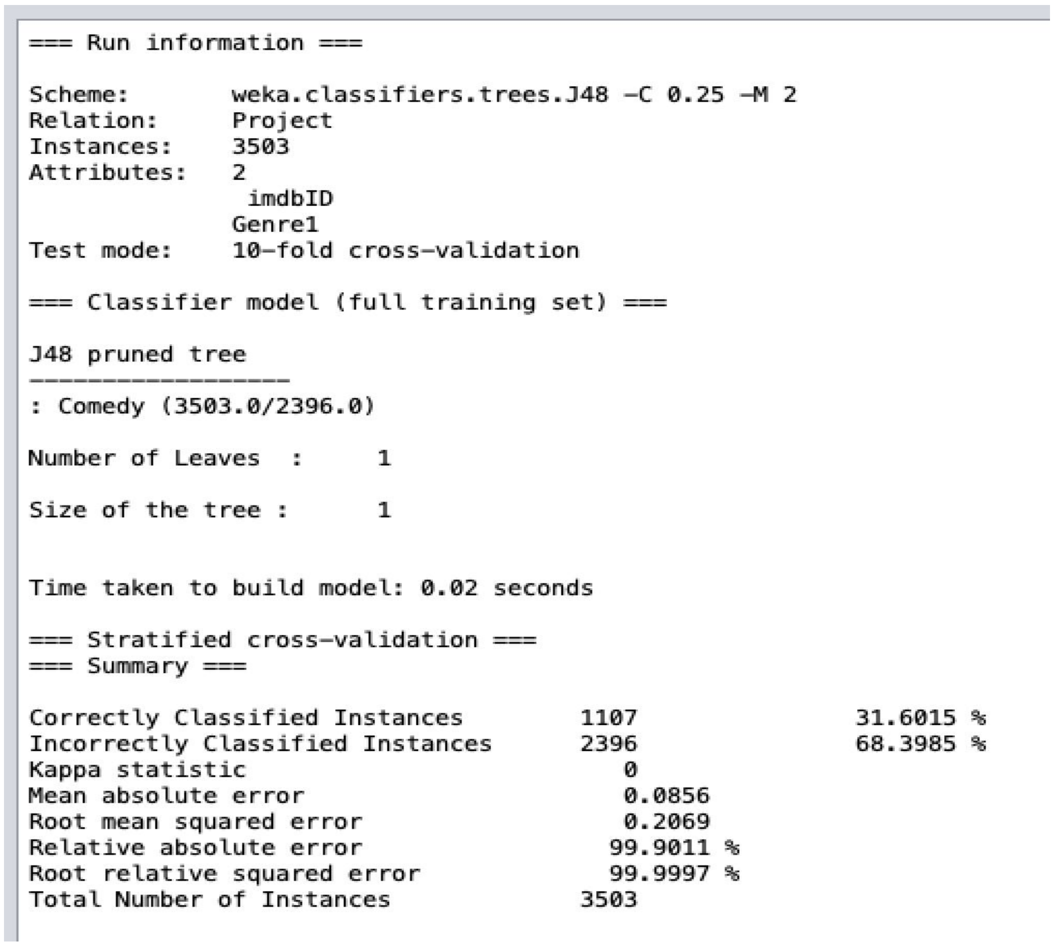


Figure 1: Distribution of genres

Figure 1 shows the distribution of movies in different genres. We consider the comedy genre as our main in our one-vs-rest approach as it is clear that the number of comedy genres is large in comparison to others. Comedy movies constitute about 30% of movies in total, so having a naive classifier that classifies all of the movies to be in the comedy type, we establish an accuracy of 30 percent out of total genres. In order to produce useful trends, there is a need to perform better than the current results.

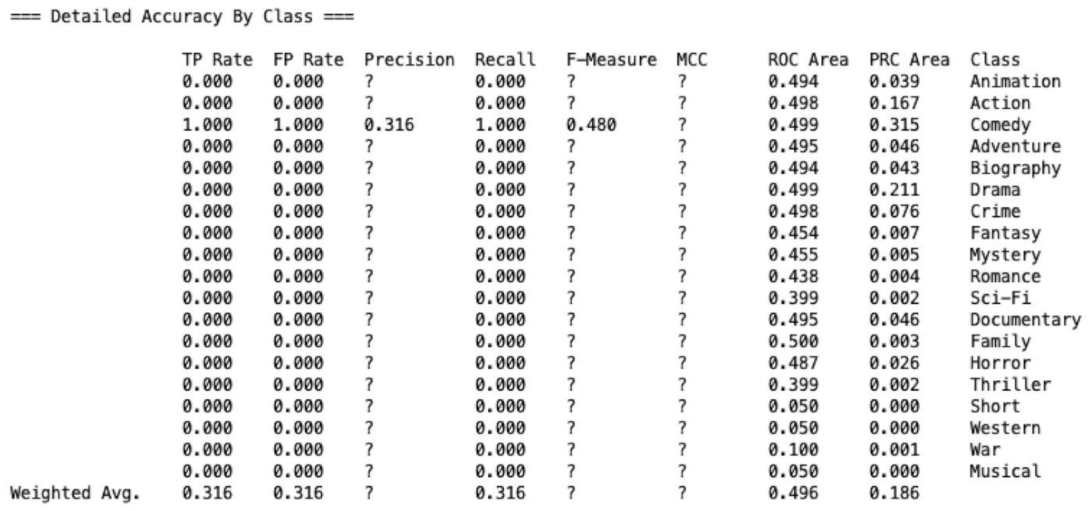
5. Data Mining

We split our data into two subsets: training data and testing data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. We had the test dataset in order to test our model's prediction on this subset.



```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     Project
Instances:    3503
Attributes:   2
               imdbID
              Genre1
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------
: Comedy (3503.0/2396.0)

Number of Leaves  :     1

Size of the tree :      1


Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1107               31.6015 %
Incorrectly Classified Instances      2396               68.3985 %
Kappa statistic                          0
Mean absolute error                      0.0856
Root mean squared error                  0.2069
Relative absolute error                 99.9011 %
Root relative squared error             99.9997 %
Total Number of Instances             3503
```

Figure 2: Weka classifier model



```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.000    0.000    ?          0.000    ?          ?        0.494     0.039     Animation
                 0.000    0.000    ?          0.000    ?          ?        0.498     0.167     Action
                 1.000    1.000    0.316      1.000    0.480      ?        0.499     0.315     Comedy
                 0.000    0.000    ?          0.000    ?          ?        0.495     0.046     Adventure
                 0.000    0.000    ?          0.000    ?          ?        0.494     0.043     Biography
                 0.000    0.000    ?          0.000    ?          ?        0.499     0.211     Drama
                 0.000    0.000    ?          0.000    ?          ?        0.498     0.076     Crime
                 0.000    0.000    ?          0.000    ?          ?        0.454     0.007     Fantasy
                 0.000    0.000    ?          0.000    ?          ?        0.455     0.005     Mystery
                 0.000    0.000    ?          0.000    ?          ?        0.438     0.004     Romance
                 0.000    0.000    ?          0.000    ?          ?        0.399     0.002     Sci-Fi
                 0.000    0.000    ?          0.000    ?          ?        0.495     0.046     Documentary
                 0.000    0.000    ?          0.000    ?          ?        0.500     0.003     Family
                 0.000    0.000    ?          0.000    ?          ?        0.487     0.026     Horror
                 0.000    0.000    ?          0.000    ?          ?        0.399     0.002     Thriller
                 0.000    0.000    ?          0.000    ?          ?        0.050     0.000     Short
                 0.000    0.000    ?          0.000    ?          ?        0.050     0.000     Western
                 0.000    0.000    ?          0.000    ?          ?        0.100     0.001     War
                 0.000    0.000    ?          0.000    ?          ?        0.050     0.000     Musical
Weighted Avg.    0.316    0.316    ?          0.316    ?          ?        0.496     0.186
```

Figure 3: Detailed accuracy

We are using a one-vs-rest approach where we will test comedy records against all the other records. Python's scikit-learning library was used for performing the classification of the dataset using the comedy vs rest approach.The methods used for the classification are as follows:

5.1 Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression is estimating the parameters of a logistic model. It is a type of regression model where the dependent variable has only two values.

Accuracy	0.65
Precision	0.46
Recall	0.46
F Measure	0.47

5.2 SVM

Support-vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, the SVM training algorithm the algorithm outputs an optimal hyperplane that categorizes new examples. The SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. The results we got from SVM are shown in the table below:

Accuracy	0.68
Precision	0.50
Recall	0.49
F Measure	0.95

5.3 K-Nearest Neighbours

The k-nearest neighbors algorithm is a non-parametric method used for classification and regression. K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure. It relies on labeled input data to learn a function that produces an appropriate output when given new unlabeled data. It is commonly used for its ease of interpretation and low calculation time.

Accuracy	0.64
Precision	0.44
Recall	0.35
F Measure	0.38

5.4 SVM with TFIDF

TFIDF, short for term frequency-inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection. We introduced this method in order to get better accuracy for the data provided. It considers the frequency of each word in the vocabulary and the weight associated with each word in the vocabulary to further improve the results. The formula given for TFIDF is:

$$\mathrm{tfidf}_{i,j} = \mathrm{tf}_{i,j} \times \log\left(\frac{N}{\mathrm{df}_i}\right)$$

$\mathrm{tf}_{i,j}$ = total number of occurences of i in j

df_i = total number of documents (speeches) containing i

N = total number of documents (speeches)

The application of TFIDF increased the efficiency and the accuracy of the algorithm significantly. There was also an improvement in precision, recall, and f-measure values.

Accuracy	0.77
Precision	0.70
Recall	0.47
F Measure	0.56

5.5 Logistic Regression with TFIDF

We applied logistic regression with TFIDF which increased the performance. There was an increase in the performance for logistic regression with TFIDF.

Accuracy	0.70
Precision	0.59
Recall	0.49
F Measure	0.52

5.6 K-Nearest Neighbors with TFIDF

KNN is a very popular algorithm for classification. Algorithm objective is to classify objects into one of the predefined classes of a sample group that was created by machine learning. The algorithm does not require the use of training data to perform classification, training data can be used during the testing phase. It performed the same with no improvements in the results.

Accuracy	0.65
Precision	0.43
Recall	0.34
F Measure	0.37

6. Conclusion

We have used a certain number of data mining techniques for classification of the data. The best accuracy was achieved through the use of SVM with TFIDF with an accuracy of 77 percent.

Method	**Accuracy**	**Precision**	**Recall**	**F Measure**
Logistic Regression	0.65	0.46	0.46	0.47
SVM	0.68	0.50	0.49	0.95
KNN	0.64	0.44	0.35	0.38
SVM with TFIDF	0.77	0.70	0.47	0.56
Logistic Regression with TFIDF	0.70	0.59	0.49	0.52
KNN with TFIDF	0.65	0.43	0.34	0.37

7. Future Work

We have put our best efforts in the project but there still exist some limitations which are as follows:

- A primary limitation exists as we have just created a prototype for the required approach and hence our data set is not huge. This constraint will easily be removed when the data set is increased to acquire more variations in the genres, and this will significantly improve the end results.

- Further implementation of data preprocessing steps like data stemming as well as word repetitions will increase the accuracy of our model.
- Multiple genres associated with a single movie will be considered as an enhancement to the current methods which only use the main genre. We can also expand our process to include more genres in the future as opposed to the current methods.

8. References

[1] Imdb alternative interfaces. http://www. imdb.com/interfaces. Accessed: 2019-11- 02.
[2] Omdb api. http://www.omdbapi.com/. Accessed: 2019-11-05.
[3] Movies genres classifier using neural networks. https://ieeexplore.ieee.org/abstract/document/5291884. Accessed: 2019-11-26.
[4] Skit Learn ml algorithms https://scikit-learn.org/stable/. Accessed: 2019-11-10
[5] Train/Test Split and Cross Validation in Python https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6
[6] KNN with TF-IDF based Framework for Text Categorization. https://www.sciencedirect.com/science/article/pii/S1877705814003750