Predicting TV Shows Grades Based on Several Simple Attributes

Nienke Wessel - s4598350

January 2017

Abstract

TV shows are popular, but producing them costs a lot of money. Therefore, it is interesting to figure out what has influence on whether a tv show becomes popular or not. In this research, I use some simple attributes (such as broadcasting network, average number of episodes per season, etc.) to predict the grade a show will get. I predict this by constructing regression trees and performing nearest neighbor classification. The highest score obtained in this research project was about 0.28, which means there is a lot to be gained in predicting the success of tv shows.

1 Introduction

A lot of money is pushed around each year in the television show business. This makes it very interesting to see what makes a good tv show that is likely to produce a lot of money and what makes less of a great show. This research project tries to find whether some very simple attributes of a tv show have influence on the score that the tv show gets. This is done by constructing regression trees and performing nearest neighbor regression on the data set and calculate the score a tv show is likely to get. In this report, I will discuss the way this was done in more detail and share the results. Lastly, I will propose some suggestions for further research in this field.

2 Related works

While it is potentionally interesting what factors correlate to the success of a tv show, surprisingly little research has been done concerning this. Most related works are about movies and not tv shows. For example, [1] tries to predict movie success by analysing fora where the movies are discussed. [2] tries to figure out whether online reviews really play a part in the success of a movie. [3] looks at the influence of the director on the success of the movie.

There are some works that concern tv shows, but they are more focused on giving recommendations based on previous behaviour of a customer, and not really on predicting the succes of a tv show (see [5] for example).

3 Methodology

3.1 General overview

I collected data from the dutch tv show website mijnserie.nl. This site contains information about all kinds of tv shows from countries all over the world. Users can keep track of their favorite show by adding them to their profiles. They can also keep track of which episodes they have watched of which tv shows. But most importantly, they can give a grade to a tv show. In this research, I have collected different attributes of different tv shows and tried to find whether certain attributes contribute to a higher or lower grade.

3.2 The attributes

In this section, I will describe the attributes I used and why I thought they were relevant to include in the research.

The first attribute I collected was the **name** of the tv show. This was simply for bookkeeping purposes.

The second attribute is the **grade** the tv show got from the users of the website. This is the attribute that I will try to predict for each tv show.

The third attribute is the **total numbers of votes** given to a tv show by the users. The guess is that popular tv shows (tv shows with a lot of viewers, and thus a lot of votes) will also have a higher grade.

I also looked at the **duration** of an episode. mijnserie.nl keeps track of the average length of an episode for each series. This is used to figure out whether tv shows with longer episodes are more popular or not. In general the length of an episode falls in one of the following three classes (of course there are some exceptions): short (20-25 min), medium (40-45 min) and long (50-55 min). The first category is usually comedy, the second drama shows from commercial networks (CBS, ABC) and the last drama shows from cable networks or other paid services (HBO, netflix).

The next attribute is which **network** owns the show. One should think of BBC, HBO, CBS, Netflix, NPO etc. People tend to say that shows from for example HBO are better than shows from CBS. I want to investigate whether this is true.

Another attribute I looked at was **starting year**. Are people nostalgic or do they think that newer shows are better?

Average number of episodes per season. I have often heard that tv shows with too much episodes per season get dragged out too much. If this is true, this would probably result in a lower grade.

Number of votes given to actors. This is the total amount of votes given to actors. The logic behind this is the same as that of the general number of votes for a tv show: popular shows will most likely get more votes.

Percentage of votes for most popular actor. This is to take a look at whether shows with one very popular actor are more popular than shows with multiple popular actors of about the same popularity.

3.3 Data collection method

For data collection, I built my own data scraper. In advance, I obtained permission from the site owner to scrape data from the site, but with the condition that I would use the English version of the site, myseries.tv, which, according to him, is less populated and on another server. However, both the Dutch site and the English site are linked to the same database, so the information on them is the same. I scraped the site on january 18th and 19th.

In the first round of scraping, I used the recommendations of a tv show to find new tv shows, and then used their recommendations, etc. The scraper started at the page of a series called Band of Brothers, found at myseries.tv/band_of_brothers/. The choice for this tv show was arbitrary. This turned out to give little tv shows (around 300), as each show recommends shows that are popular and you quickly collect all popular shows, but not the less popular shows. So I tried the same thing by starting at several other shows, but this did not improve the amount of shows much.

So in the second round of scraping, the scraper was adjusted to collect data from the page that first lists all countries and then per country all networks. Then I collected the first 20 shows of each network (more than 20 would have been hard, since the website uses an infinite scroller). This resulted in the data set of

3363 data points.

For the scraping, I used Python with several libraries. Most of the scraping was simple html parsing, which was done with the `requests` library and `html` from the `lxml` library.

3.3.1 The attributes

Most of the attributes are pretty self-explanatory and could be retrieved straight from the page. The networks were collected as a string for each show, but were changed to binary values for each network. A 1 indicating the show belongs to this network, a 0 indicating it does not. Something similar was done for the genre-attribute. The average number of episodes per season were calculated as the average from the last three seasons, as those are the ones that are retrievable straight from the home page of a tv show. The total votes on actors were retrieved as the votes on the five most popular actors, as they are mentioned on the main page of a tv show. This total is also used in the calculation of the percentage of votes the most popular actor got.

3.4 The algorithms

I chose tree regression and nearest neighbor regression, because they are simple to use and suited for the problem. I used the classes `TreeRegressor` and `NearestNeighborRegressor` from the `sklearn` library, because they are easy to use and the limited time given for this project makes it hard to do a lot of research into other methods or write my own method. All the standard settings were used, unless stated otherwise in the results section. These can be found in the documentation of `sklearn`.

For building the regression tree and the nearest neighbor classifier, total number of votes and total number of actor votes were not taken into account. These values are usually not known beforehand, so it is more interesting to leave them out. The attributes that were used are: duration of episode, starting year, nr episodes per season, percentage of votes most popular actor got, genres, networks.

Table 1: Pearson Coefficients

Attribute	Pearson	p-value
nrVotes	0.16204	3.2004e-21
duration	0.055721	0.0012266
year	-0.024532	0.15492
nrEpSeas	0.069220	5.8821e-05
totAcVotes	0.16995	3.2676e-23
percTopAc	0.42660	8.0318e-149

3.5 Evaluating the Predictions

The built-in `score()` functions were used for evaluating the predictions. The details about the calculation of the score can be found in the `sklearn` documentation. The maximum score is 1. A score can be negative if the regressor performs worse than guessing the mean for every input.

The scores were calculated using 10-fold cross validation. The module `KFold` was used from `sklearn.model_selection`. 10-fold cross validation is not too computationally expensive, but provides a reasonably good generalisation error (see also [4] for a discussion about KFold vs. hold-out).

4 Results

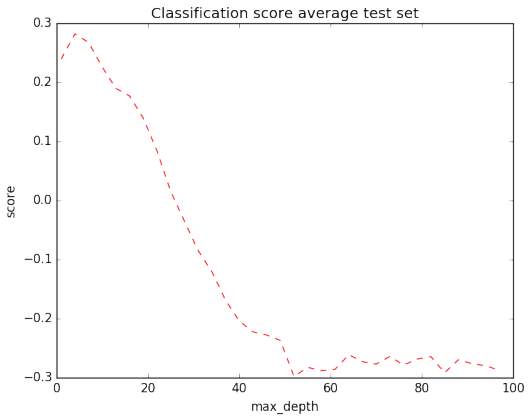
4.1 Pearson Coefficient

The pearson coefficient was only calculated for the numerical values, not for the booleans. That means that 'genres" and "broadcasting network" were not taken into account. See table 1. It is worth noting that the percentage of top actors votes has such a high pearson coefficient. This is potentially interesting.

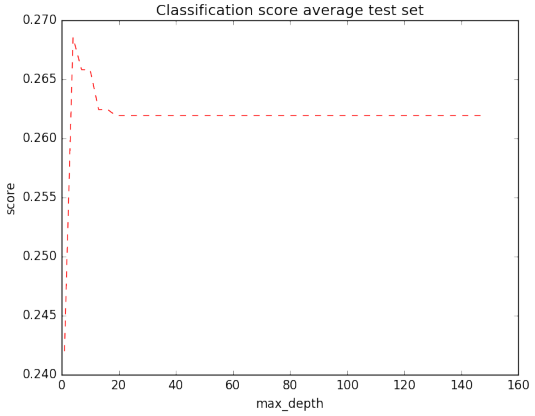
However, it should be noted that very unpopular shows where the score is 0, because no one voted, usually have 0% as percentage top actor. This would account for a part of the high correlation. However the same holds for number of votes for the score and number of votes on the actors in total, and these have significantly lower pearson coefficients. So, even though part of the high pearson coefficient might be misleading, it might still be interesting to investigate this further.
Other than this, there appears to be a slight preference for longer episodes and more episodes per season. The negative correlation for the year is not really significant.

4.2 Regression Tree

As explained in the Methods chapter, the trees were made with the RegressionTree class from the sklearn module in the Python programming language. In order to optimise the trees, several values were tried in the settings of the tree. The scores were calculated with 10-fold cross validation. In figure 1 the average scores are shown as a function of the depth. The depth ranges from 1 to 100. In 1a the standard settings were used that come with the Python module. In 1b the optima (see below) for the other settings were used. The optimal maximum depth appears to be somewhere below 10. After that, the score significantly drops, indicating that overfitting is occuring. In figure 2 the average scores are shown as a function of the min_samples_leaf. This value ranged from 1 to 100. In 2a the standard settings were used that come with the Python module. In 2b the optima (see above and below) for the other settings were used. The optimum for this value appears to be somewhere after 10. After that, the score does not improve a lot. With a value lower than 10, the scores become significantly lower. This is most likely due to overfitting. In figure 3 the average scores are shown as a

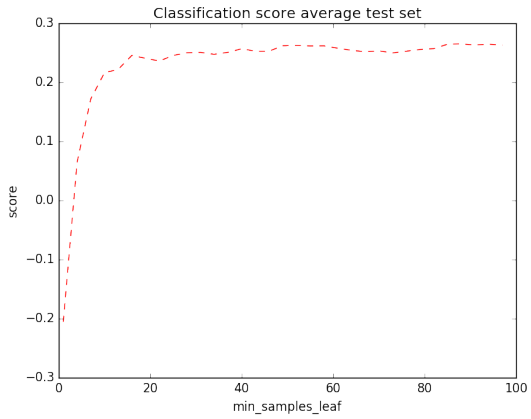


(a) min_samples_split=2, min_samples_leaf=1

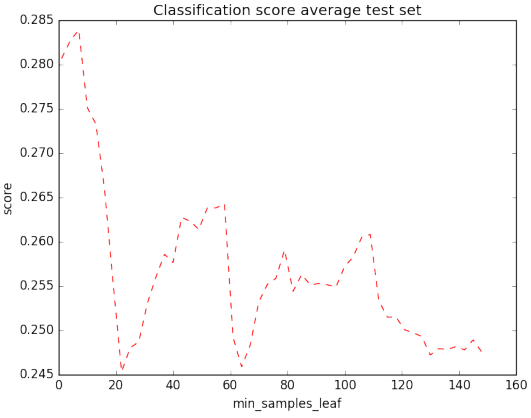


(b) min_samples_split=130, min_samples_leaf=20

Figure 1: Classification score based on maximum depth

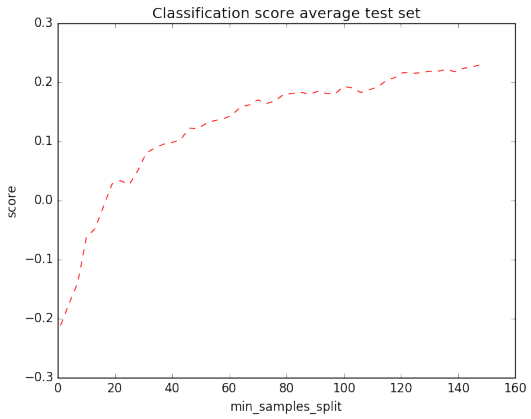


(a) min_samples_split=2, max_depth=None

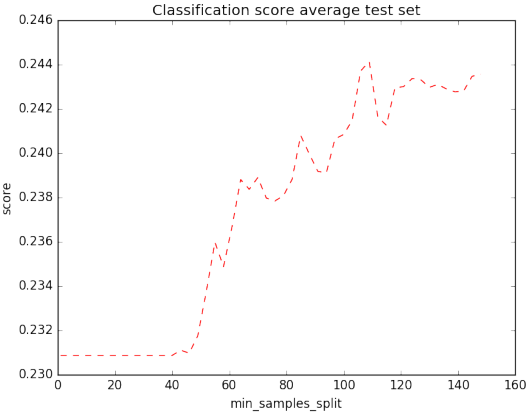


(b) min_samples_split=130, max_depth=10

Figure 2: Classification score based on min samples leaf



(a) min_samples_leaf=1, max_depth=None



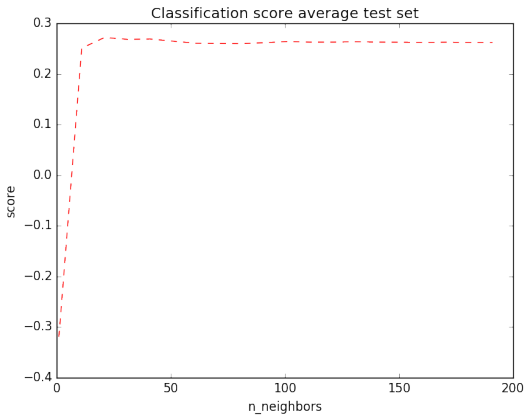
(b) min_samples_leaf=20, max_depth=10

Figure 3: Classification score based on min samples split

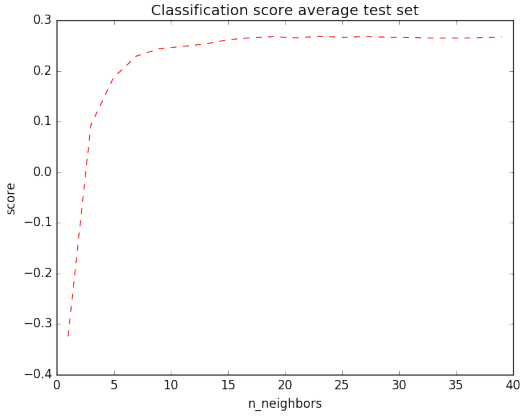
function of the min_samples_split. This value ranged from 1 to 150 (because the score did not seem to stabilize after 100, the value was extended to 150 to give a better idea of its behaviour). In 3a the standard settings were used that come with the Python module. In 3b the optima (see above) for the other settings were used. The optimum for this value appears to be somewhere after 120. After that, the score does not improve a lot. With a value lower than 50, the scores become significantly lower. This is most likely due to overfitting.
The graphs show that there is a lot to gain with more tweaking of the different settings in the regression tree algorithm. The highest score obtained was little below 0.285 (see figure 2).

4.3 Nearest Neighbor Regression

As explained in the methods chapter, the NearestNeighborRegression class from sklearn was used in the Python programming language. The scores are calculated using the standard score function of that class using 10-fold cross validation. Two different distance measures where used to see if one produces better (better meaning giving a higher score) trees than the other. In figure 4 the average scores are shown using the cityblock measure. In 4a the number of neighbors ranges from 1 to 140. In 4b that value ranges from 1 to 40. This is done to show more detail in the graph. In figure 5 the average scores are shown using the cityblock measure. In 5a the number of neighbors ranges from 1 to 140. In 5b that value ranges from 1 to 40. This is done to show more detail in the graph.
From the graphs, it becomes apparent that the optimum for the cityblock measure appears to be around 10, after that there is not much improvement. The optimum of minkowski is slightly later; after about 25 there does not appear to be much improvement. What also is interesting is that the cityblock measure gives

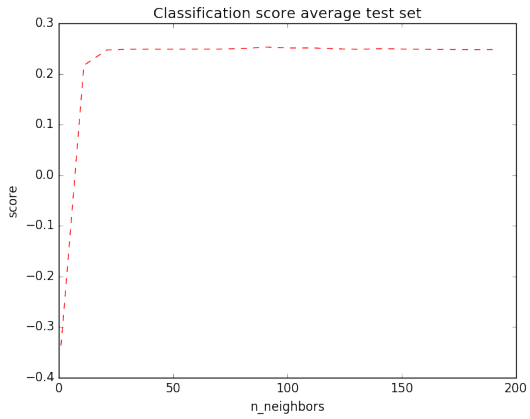


(a) With 1 to 150 neighbors

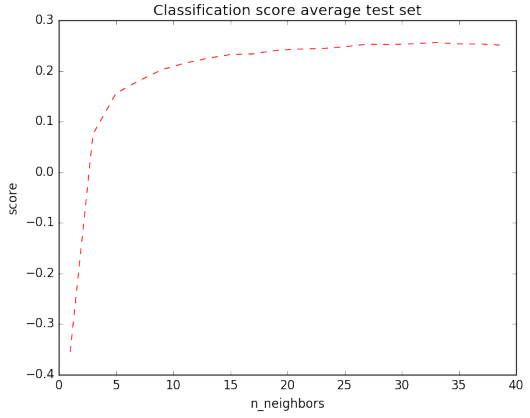


(b) With 1 to 40 neighbors

Figure 4: Classification score with cityblock distance



(a) With 1 to 150 neighbors



(b) With 1 to 40 neighbors

Figure 5: Classification score with minkowski distance

slightly better scores at and after the optimum. The difference is about 0.01-0.03 in score. A score higher than 0.28 does neither seem to achieve.

5 Conclusion

It appears to be really hard to classify tv shows correctly. Scores higher than 0.3 are very hard to obtain. And moreover, the achieved scores might be partly because there are a lot of unpopular shows in the data set that have a show score of 0. These are easy to classify (for example by using the 0% of votes the most popular actor got), so probably improve the scores of the classifiers.
The regression trees appear to do slightly better than the nearest neighbor regressor, but only after tweaking the settings. It is likely that we can improve both measures by tweaking the settings more.

6 Suggestions for further research

Even though the results are not very promising, there appears to be a slight correlation between some of the variables used in this research and the score of a tv show. It could be interesting to investigate this further. By removing the 0 values from the data set¸ it becomes clearer whether this correlation is useful in real life applications. Also, it could be interesting to investigate the generated trees to see which attributes they use (that was not possible within the time frame of this small research project). More tweaking of the variables would also be interesting. This research shows that using settings different from the standard settings can generate significant improvements in the scores. The tree variables (max_depth, min_samples_leaf and min_sample_split) could be optimized together, instead of separately

and then combined (which was done in this research). Also, other variables could be investigated and optimized. Lastly, there are other attributes of tv shows (for example the country in which the show was produced) that were not included in this research, but that could be investigated.

References

[1] Kraus, J. S., Simon, D., Fischbach, K. and Gloor, P. Predicting Movie Success and Academy Awards through Sentiment and Social Network Analysis. *16th European Conference on Information Systems*, June 2008.

[2] Duan, W., Gu, B. and Whinston, A. B. Do online reviews mattter? - An empirical investigation of panel data. *Decision Support System*, volume 45, issue 4, p. 1007-1016

[3] Delmestri, G., Montanari, F. and Usai, A. Reputation and Strength of Ties in Predicting Commercial Succes and Artistic Merit of Independents in the Italian Feature Film Industry. *Journal of Management Studies*, volume 42, issue 4, p. 975-1002

[4] Blum, A., Kalai, A., and Langford, J. Beating the hold-out: Bounds for k-fold and progressive cross-validation. *Proceedings of the twelfth annual conference on Computational learning theory p. 203-208. ACM.*

[5] Xu, Mengxi, et al. Catch-up tv recommendations: show old favourites and find new ones. Proceedings of the 7th ACM conference on Recommender systems. ACM, 2013.

Appendices

A Used programs

This appendix contains a list of the programs I used

- readInternetz2.py; this program collects the data used in this research from the internet(z).
- clean.py; used to make boolean values of the genres and networks. This made this data ready for processing
- pearsonCoefficient.py; used to calculate the pearson coefficients of different attributes. This program also contains code used to remove all 'nan' (not a number) values from the generated data set. These nan values were the result of calculating the average of an empty list. They were replaced by 0s.
- regressionTree.py; this program contains three functions. The first one gets the data from the json file and puts the attributes in one large two dimensional array, so that it is ready for processing. The second makes a regression tree and produces a graph with scores in some interval for some variable. The third does a similar thing for nearest neighbor regression.