Movie Rating Prediction

Project Report

presented by
Team 3

submitted to the
Data and Web Science Group
Prof. Dr. Paulheim
University of Mannheim

November 2019

Contents

1 Application area and goals

Producing a movie is a time and resource expensive venture, whose success highly depends on whether or not the audience likes it. It is also not only interesting whether a movie is generally liked but especially how the target audience thinks about it. Knowing this before even producing it could save production companies a lot! Therefore it is highly interesting to predict the rating based on data available before its release.

One way to evaluate whether an audience enjoys a movie is to have a look at the average rating users award in online rating websites and communities like MovieLens (see [MovieLens, 2019]). Being able to predict this means to predict how it appeals to the audience and therefore its success. To do this, regression can be used with the rating as target variable. The task can also be adapted to classification when ratings are used as labels based on the range their in. To find the target audience of a movie, viewers can be clustered based on the genres of movies they have watched. It is then possible to find the cluster and therefore the target audience a new movie belongs to based on its genres.

Based on this, our goals are to: (1) cluster users and movies to find the target audience a movie belongs to, (2) predict the average rating a new movie will receive and (3) predict the rating with the target audience as additional feature to see whether the target audience has an influence on the rating.

2 Structure and size of dataset

In this project, the *movies dataset* from Kaggle [Banik, 2017] is used. It includes 26 million ratings by more than 270,000 users on 45,000 movies and its attributes are well described. Each movie has 29 metadata attributes, which can be grouped into four groups as can be seen in table 1. The ”ratings” table contains multiple ratings in a range from 0.5 to 5.0 per user for different movies. The ”credits” table contains additional movie information about the cast and crew involved in its production. Some attributes have missing values, one example is the ”homepage”. Since not all movies have a website, the attribute can sometimes be empty. Another reason for missing information is that some movies where produced a long time ago and not all information could be collected anymore. Also, there are some movies that have not been released yet and have missing values in attributes that are not known before a release, like e.g. ”revenue”. Overall the quality of the data is good. The attributes in the dataset have many different data types. For example the genre or cast of a movie are stored in a JSON format. The objects in the JSON data contain different information such as name, id, description or department of a person.

Numerical attributes	budget*, id*, imdb_id*, popularity, release_date, revenue, runtime*, vote_average, vote_count, userid, movieid*, rating**,timestamp
boolean attributes	adult**, video
JSON format attributes	belongs_to_collection**, genres**, production_companies, production_countries, spoken_languages**, cast**, crew**
String Attributes	homepage**, original_language, original_title, overview, poster_path, status, title

* attribute is selected, ** attribute is selected and transformed

Table 1: Attribute overview

There are no duplicates in the data. Since the movie rating is especially interesting for our business case, the distribution of ratings is also examined. The grouped movie ratings after pre-processing show that the ratings are generally distributed following a normal distribution as can be seen in figure 1. A more detailed data exploration can be found in the *01 Data Exploration* Jupyter Notebook.

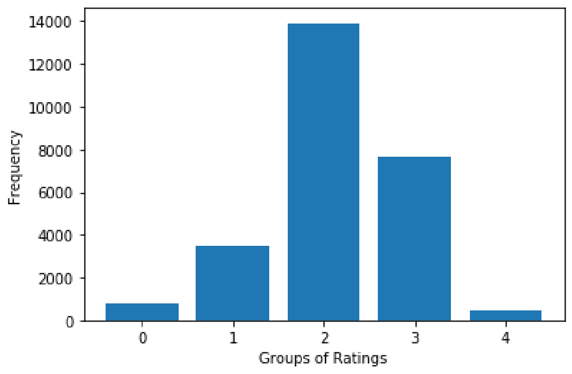


Figure 1: Distribution of grouped movie ratings

3 Preprocessing

The three main datasets described in the previous section needed to be pre-processed and combined. As there were many different features available, the pre-processing was not a one time task but was repeated several times until the best outcome could be achieved. We started by analysing the "movies_metadata" dataset which had 45,463 entries. The different genres needed to be hot encoded as they are an important feature for clustering and prediciton alike. Additionally, we encoded the attributes “part of collection”, “adult” and “hasHompage” into binary values for the rating prediction through regression and classification. The three attributes “productionCompanies”, “productionCountries” and “spokenLanguage” were encoded in a JSON format as well, therefore we needed to extract the information using regular expressions. In order to obtain the genres of the movies a user rated, the user ratings have to be combined once more with the hot encoded movie metadata. The joint data set can then be grouped by user to get the absolute numbers of movies a user rated in each genre, which can then be used to compute the percentage (as decimal in [0,1]) of movies a user rated per genre. Since the data contains more dimensions than can be plotted, we use a t-sne and pca transformation (see [Maaten and Hinton, 2008] and [Jolliffe, 2003]) to be able to at least elementary visually verify the results of the clustering. The clustering itself is performed on the original data. The last file to analyse was the “credits” dataset with 45,476 entries of actors and directors for every movie. As the file was saved as JSON we extracted the data as well using regular expressions. Additionally, the attributes “original languages”, “spoken languages”, “production companies” and “production countries” were hot encoded. As there were too many production companies and actors, we filtered on only those who occurred more than 100 times in the dataset to reduce the number of created features and increase the performance. The result were the 22 most common companies and 11 most common actors. In the last step, outliers in the two attributes “budget” and “runtime” were handled by normalization and all values which had a “zero” were removed.

In total we created four preprocessed datasets to be used in regression and classification. The first file consists of 43,872 entries from which we created a second file filtering out all rows including a zero value. The third file with a total of 118,856 entries additionally contains the cluster ID created in the clustering step. Here we also created a fourth file through filtering out all rows including a zero value leaving 34,243 entries in total.

4 Data Mining

4.1 Clustering

Users are clustered based on the genres of the movies they watched (rated) in percent. We applied different clusterin techniques, namely *KMeans*, *KMedoids*, *agglomerative clustering* and *DBSCAN*. The clustering model that yielded the best results was then also applied to the movies to find the cluster and the average rating of their target audience. The complete cluster experiments can be found in the Jupyter Notebook *04 Cluster Experiments*.

4.1.1 KMeans

To find the optimal KMeans model for the data, we calculated the squared standard error (SSE) for various k values as described in [Natakarnkitkul, 2019]. As can be seen in figure 2, there is no significant ”elbow” in the curve, but in combination with the scatter plots for different k values it was possible to set k = 9 as optimum. KMeans generally has a good performance, especially compared to the other clusterers. It yielded good results with the chosen k value as can be seen in the final cluster distribution in figure 2 and in the overview in table 2.

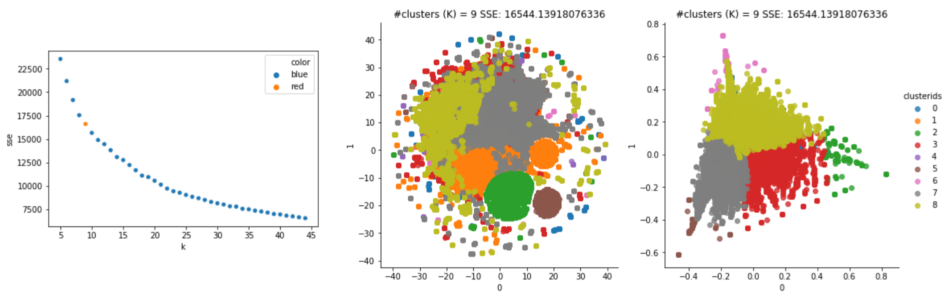


Figure 2: KMeans with k = 9 clusters on the transformed data

Cluster ID	Top 1	Top 2	Top 3	Size
0	Fantasy = 0.598	Documentary =0.205	Comedy = 0.103	7183
1	Drama = 0.156	Comedy = 0.096	Science Fiction = 0.066	62532
2	Documentary = 0.972	Fantasy = 0.014	Comedy = 0.007	22148
3	Documentary = 0.355	Drama = 0.318	War = 0.074	16636
4	Animation = 0.954	Comedy = 0.011	Western = 0.01	2592
5	Drama = 0.992	War = 0.003	Crime = 0.002	8093
6	Comedy = 0.983	Drama = 0.005	Music = 0.004	5226
7	Drama = 0.413	War = 0.082	Comedy = 0.074	33659
8	Comedy = 0.349	Documentary = 0.201	Animation = 0.189	21840

Table 2: KMeans Clusters: Genre Distribution and Size

4.1.2 KMedoids

After we found the optimal KMeans model, we adapted the same method to evaluate the optimal KMedoids model for our data. Due to performance issues, KMedoids could not be applied to the whole dataset. Therefore, a sample of 10,000 records was used.

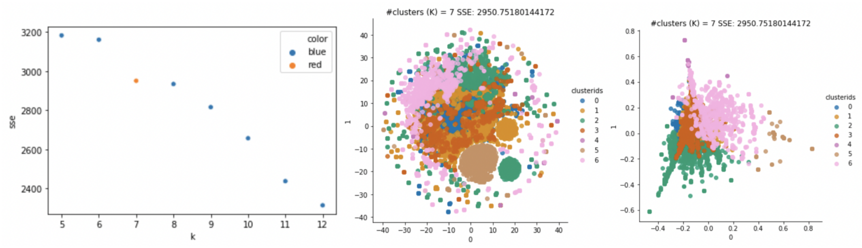


Figure 3: KMedoids with k = 7 clusters on the transformed data

As for KMeans, the SSE for various k values was calculated. As can be seen in figure 3, for KMedoids there is a significant ”elbow” in the curve at k=7. With this data used, the final cluster distribution looks similar to the cluster distribution of KMeans.

4.1.3 Agglomerative Clustering

It was not possible to apply the sklearn agglomerative algorithm to all users at once, therefore we applied it to a sample of 10,000 entities. During the experiments, agglomerative clustering with linkage = single and linkage = ward with k values between 7 and 13 was tried. Linkage = single did not perform well, while k = 12 was the best parameter setting with linkage = ward (see figure 4), but it appeared to be slightly overfitted since it returned some very small specialized clusters. Yet KMeans still returned a better cluster size distribution and ran on all data so we prefer it.

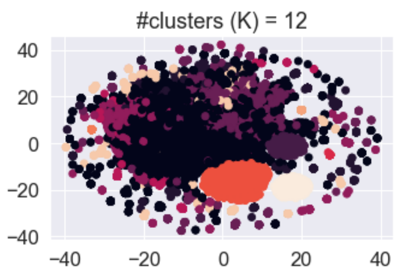
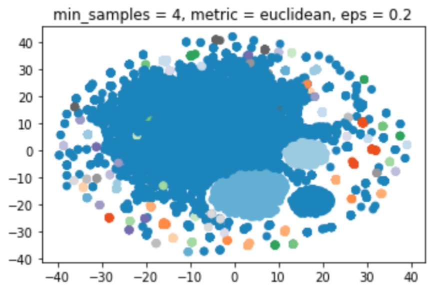


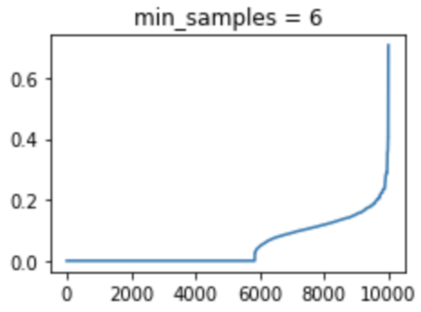
Figure 4: Agglomerative Clustering

4.1.4 DBSCAN

With DBSCAN it was also not possible to apply the algorithm to all users at once. Therefore, a record of 10,000 entities was used. To evaluate the otimal EPS value for DBSCAN, we roteted over different min sample sizes. Figure 5b shows an example of our results. Overall, the results show that EPS is similar for different min samples and EPS = 0.2 was chosen as optimal. As a next step, the cluster distribution was generated with EPS = 0.2. The evaluation of different min sample sizes and distance measures showed, that an ideal configuration of DBSCAN for our dataset is EPS = 0.2, euclidean distance and min sample = 4.



(a) DBSCAN Clustering



(b) Evaluate optimal EPS

Figure 5: DBSCAN Results

But even with this configuration, the clustering of DBSCAN is worse than KMeans (see figure 5a). As a result it can be stated that of the clustering algorithms compared here, KMeans is best suited for our data.

4.1.5 Final Cluster Application

KMeans with k = 9 was the best performing clustering algorithm for clustering users based on the genres of movies they watched. The trained clustering model returned the cluster each user belongs to, the meaning of each final cluster and their size can be seen in table 2. These clusters could then be added to the user ratings based on the userId. The model could also be applied to the movies since their genres have the same format as for the users. With users and movies clustered, it was then possible to evaluate whether the affiliation to a certain target audience (a movies cluster ID) has an influence on its rating.

4.2 Regression

Since the attribute we want to predict is continuos, our use case requires regression algorithms. As the value to predict can only be in between 0.5 and 5.0, we are handling interpolating regression. Our process consisted of four phases. First, we identified the best features. We proceeded only with the selected features and ran multiple regression algorithms using the standard parameters. This way we got a first impression of the different algorithms and continued with the optimization of the results for different algorithms. Finally, in the fourth phase we tested our learned models using the test data we split up at the beginning.

4.2.1 Feature Selection

After our preprocessing we have around 60 features, including our self-created attribute ‘userclustering’. Before training the models, we removed the less important features, which do not have impact on our target variable. Our goal was to remove misleading information and to reduce the learning time. For the feature selection we used the algorithms *F-Regression*, *Select K-Best* and the fast and popular *XG-Boost* using gradient boosting. We combined the results into a merged list of most important features. The algorithms classified some genres as the most relevant attributes. Also included are the self-generated attribute or other meta information such as ”runtime” and ”hasHomepage”.

4.2.2 Initial evaluation of different algorithms

To identify the best algorithms for our regression problem we first applied *Ridge* representing linear regression, Ada Boost Regressor using a *Decision Tree Regressor*, *K-Neighbors Regressor*, *Random Forest Regressor*, *Gradient Boosting Regressor* and *XGB Regressor* to our data. After learning the models on our training data, we applied the model to our test data. To evaluate the differences between predicted and actual values, we calculated the Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and the coefficient of determination (R^2). When using MAE, all differences have the same weight, while MSE and RMSE penalize larger errors. R^2 evaluates the relationship between the independent variable (features) and the dependent variable (target). While MAE, MSE and RMSE should be as small as possible, the R^2 score should be as close to 1 as possible. [Grant and Kenton, 2019, Drakos, 2018] For a better estimate of the performance we applied cross validation on the test data and calculated the mean of R^2 of all folds. The results in table 3 show that no model was working properly. XGB Regressor achieved the best results with the highest average at cross validation, smallest error scores and the highest R^2. Also visible is that the clustering group did not help to create a better model either. It even led to a minimal deterioration. It seems that the dependent variable rating can’t be predicted using the remaining independent variables. This assumption is confirmed by the R^2 scores, which are always next to 0 or even negative, meaning that our features are not suitable and there is no relation between them and the target.

	∅ Score	*MAE*	*MSE*	*RMSE*	R^2
	Without Cluster IDs				
Ridge (linear regression)	0.095	0.512	0.475	0.689	0.098
Ada Boost Regressor	-0.191	0.600	0.619	0.786	-0.175
K-Neighbors Regressor	-0.068	0.563	0.561	0.749	-0.064
Random Forest Regressor	-0.142	0.568	0.573	0.757	0.088
Gradient Boosting Regressor	0.097	0.085	0.509	0.471	0.686
XGB Regressor	0.097	0.509	0.471	0.686	0.105
	With Cluster IDs				
Ridge (linear regression)	0.094	0.581	0.612	0.782	0.095
Ada Boost Regressor	-0.091	0.613	0.714	0.845	-0.056
K-Neighbors Regressor	-0.027	0.604	0.671	0.819	0.007
Random Forest Regressor	-0.076	0.601	0.681	0.825	-0.008
Gradient Boosting Regressor	0.128	0.566	0.587	0.766	0.131
XGB Regressor	0.128	0.567	0.588	0.766	0.130

Table 3: Analysis of the Regression Algorithms: with and without Cluster IDs

4.2.3 Hyper-parameter tuning

In the next step, we took different algorithms and optimized the hyperparameters. As hyperparameters are settings of a model which must be set beforehand, we needed a tool to try out different parameters sequentially. For this task we used Grid Search. We defined different values for each parameter and used R^2 as the evaluation method. In table 6 are the results using the best parameters possible on the dataset with cluster IDs.

	∅ Score	*MAE*	*MSE*	*RMSE*	R^2
Ridge (linear regression)	0.094	0.581	0.612	0.782	0.095
Random Forest Regressor	0.091	0.548	0.578	0.760	0.144
XGB Regressor	0.138	0.549	0.567	0.753	0.161

Table 4: Analysis of Hyper-parameter tuning

Apart from Ridge, which had similar results, Grid Search found hyperparameter which lead to better results. Unfortunately, even an execution of the algorithms with optimal parameters did not lead to a good result. Our original assumption seems to be confirmed.

4.3 Classification

As the Regression approach did not yield a good model, we tried in a next step to adapt the regression problem to a classification problem. For that, we transformed the average rating back to the discrete classes [0, 1, 2, 3, 4, 5]. For example, an average rating between 3,5 and 4,5 got assigned to the label 4. From this approach we hoped to get a better model since we discretize the data. The Approach for the model evaluation is similar to the one of the regression problem. First, we choose a set of different algorithms and let them train on the training data. As algorithms we choose the following ones: (1) KNNClassifier, (2) DecisionTree Classifier, (3) Random Forest Classifier, (4) Gaussian Naïve Bayes Classifier, (5) Support Vector Classifier. We chose those algorithms in order to cover a wide area of different classes of algorithms. The general approach is similair to the one of the regression. We first run all algortihms with the default settings and choose two ones for hyperparameter tuning with a 10-fold cross validation. Afterwards we test the tuned algorithms on the test dataset. The results are shown below. After the first analysis we saw that the algortihm only learns the majority class what led to this results. Therefore we tried additionaly to over- and undersample the training data.

	Initial Acc.	*Initial F1*	*∅ Acc. Train*	*∅ Acc. Test*	*F1*
		Without Cluster IDs			
Random Forest	0.458	0.445	0.542	0534	0.424
Decision Tree	0.445	0.413	-	-	-
KNN	0.446	0.405	0.515	0.516	0.394
Naive Bayes	0.529	0.366	-	-	-
SVC	0.445	0.373	-	-	-
		With Cluster IDs			
Random Forrest	0.502	0.486	0.517	0.517	0.467
Decision Tree	0.458	0.457	-	-	-
KNN	0.507	0.484	0.497	0.507	0.484
Naive Bayes	0.417	0.269	-	-	-
SVC	0.287	0.282	-	-	-

Table 5: Analysis of Classifiers: with and without Cluster ID, initial and tuned values - not sampled Data

The result of the undersampling you can also see in the table below. The oversampling led to very bad results, which are most propably caused by overfitted models. Another indicator for that is that the average accuracy on the training data was about 80% while the result on the test data was 20%.

	Initial Acc.	*Initial F1*	*∅ Acc. Train*	*∅ Acc. Test*	*F1*
			Without Cluster IDs		
Random Forrest	0.246	0.292	0.298	0.244	0.263
Decision Tree	0.238	0.283	-	-	-
KNN	0.213	0.259	0.203	0.216	0.266
Naive Bayes	0.072	0.079	-	-	-
SVC	0.428	0.340	-	-	-
			With Cluster IDs		
Random Forrest	0.282	0.322	0.312	0.284	0.318
Decision Tree	0.268	0.305	-	-	-
KNN	0.219	0.258	0.230	0.218	0.258
Naive Bayes	0.069	0.074	-	-	-
SVC	0.202	0.224	-	-	-

Table 6: Analysis of Classifiers: with and without Cluster ID, initial and tuned values - Undersampled Data

5 Results

Our first goal, clustering users and movie's, was accomplished by using KMeans. The prediction of a movies rating, however, was a more complex task and did not yield sufficient results. Unfortunately, the models always performed poorly in the test phase. It seems that the features in general are not enough or good enough to determine the rating. Changing the problem to a classification task and the additional information of the target audience did not help to achieve a better result either, but the result was at least better than guessing (16.67% based on 6 classes). One potential for improvement is an improved integration of actors and directors. While we have counted their frequency in movies, it could be more effective to take popularity through the help of external sources, i.e. social media accounts, into account. In addition to the actors and directors, keywords describing the content of a film could also be considered. Additionally, it might be helpful to obtain better data. Since we had to delete a lot of datasamples in the preprocessing step, we might have lost crucial information. More data would also help to overcome the challange of unbalanced classes, which had a significant impact on the classification models. Lastly, we come to the conclusion that it might simply not be possible to predict ratings based on metadata of movies, since movies are influenced by a lot more. There is no way to measure the chemistry of actors or the story of a movie, which might be the real influence of a rating of a movie.

References

[Banik, 2017] Banik, R. (2017). The movies dataset. https://www.kaggle.com/rounakbanik/the-movies-dataset. Accessed: 2019-11-06.

[Drakos, 2018] Drakos, G. (2018). How to select the right evaluation metric for machine learning models: Part 1 regression metrics. https://towardsdatascience.com/how-to-select-the-right-evaluation-metric-for-machine-learning-models-part-1-regrression-metrics-3606e25beae0. Online. Accessed: 2019-11-27.

[Grant and Kenton, 2019] Grant, M. and Kenton, W. (2019). Coefficient of determination. https://www.investopedia.com/terms/c/coefficient-of-determination.asp. Online. Accessed: 2019-11-27.

[Jolliffe, 2003] Jolliffe, I. (2003). Principal component analysis. *Technometrics*, 45(3):276.

[Maaten and Hinton, 2008] Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

[MovieLens, 2019] MovieLens (2019). About movielens. https://movielens.org/info/about. Accessed: 2019-11-10.

[Natakarnkitkul, 2019] Natakarnkitkul, S. (2019). Get the optimal k in k-means clustering. https://medium.com/towards-artificial-intelligence/get-the-optimal-k-in-k-means-clustering-d45b5b8a4315. Online. Accessed: 2019-11-20.