The Credits project

Contents

[Overview 1](#_Toc5272368)

[Data cleaning 2](#_Toc5272369)

[Data exploration 3](#_Toc5272370)

[The modeling approach 3](#_Toc5272371)

[Results 4](#_Toc5272372)

[Conclusions 5](#_Toc5272373)

1. **Overview**

The project is based on the study of the dataset “German credit data”, it has been downloaded from the UCI machine learning repository at <https://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29> It’s consist of a data text file containing 1k records and twenty attributes and has been converted in a dataframe named “credits\_df”. At a first sight there are factors and numbers, all the factors are coded data as “Axxx” format whose meanings are clearly explained in a description text file “german.doc” in located at the same repository, from the the same have been extracted the metadata and promptly converted in variables names.

This project rely on a typical binary classification problem: there is only a label “credit” whose possible values are “good” and “bad” which refer to a bank customer who asked for a credit; the one who payed back are “good” , the one didn’t, “back”.

The bank board would wish to know in advance who would be good an who would be bad, on the basis of customer attributes. The target is to study and evalute best approaches to predict with high reliability the customer category on the base of attributes/factors after data manipulation/trasformation and a set of analysis tasks.

1. **Data Cleaning**

The first operation that have to be done on data is to check the presence of the inconsistent values that could affect the correctness of analysis. Generally, if we don’t take in count incoherent values, mean and sd could be affected by bias, all the further analisys would be affected by some kind of error, and prediction would fail. First of all, the presence of NA’s must be checked. Fortunately, the data frame does not contain NA’s value as shown in the following R code

As shown in the table, there are not NA’S or empty values, so all the metrics are calculated on true data.

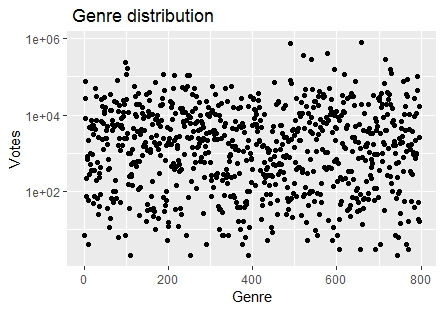
1. **Data Exploration**

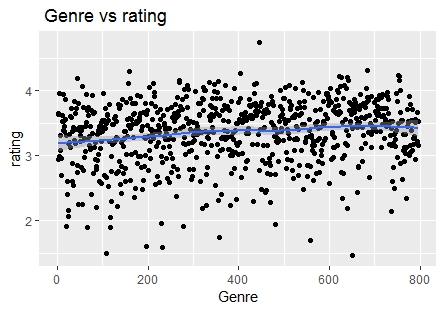
Whatever ML algorithm we want to apply to our data, before it’s necessary to explore data prior to some *preprocessing* activities on the possible candidate predictors, that is: standardization, log transform, removing some that are highly correlated with others, and removing that with very few non-unique values or close to zero variation.

In the movielens data there are six, but only five of them are relevant as predictors candidate:

* 1. **The genres**

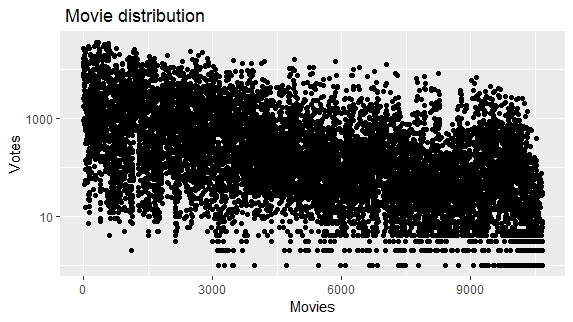
There are 797 genres, but in the most of cases, they are combination of different genres. As shown in the first plot, there is great variability across genres, in the sense that blockbusters have much more votes than cult movies. Generally, with this kind of variability, regularization should be considered: items with few votes have lower weight compared to that most voted. The second plot suggests that the average rating has some variability across genres and this fact could be useful in improving prediction.

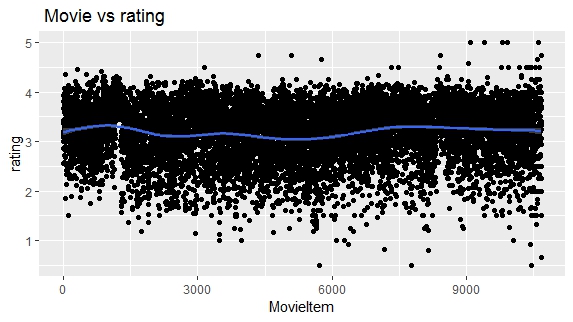




* 1. **The movieId**

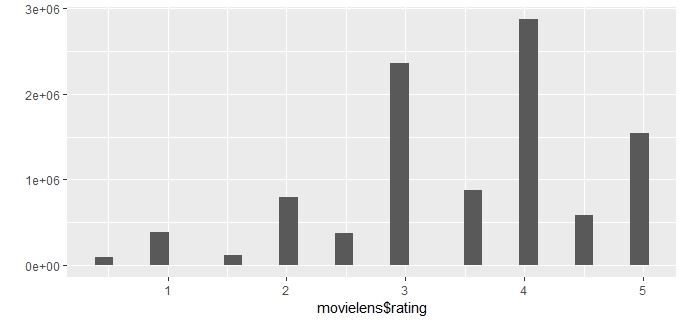
As shown in the next 2 plots, and as intuition suggests, the movieId is a central feature that must be used as predictor. On the y axis of the first plot the scale is logarithmic to make the graphics more readable. There is great variability, among movies and votes. The second plot shows movie versus average rating, the smoothing gives fair evidence of variation, but also some apparent outliers that should be treated with regularization approach.





* 1. **The rating**

The rating is the outcome feature, the one to be predicted. Is a numeric continuous type and, in this particular project, will be predicted using ML regression algorithms, so the RMSE metrics will be computed. On the other hand, the feature could be transformed in categorical so it would be a classification problem and the target metric would be accuracy, sensitivity and sensibility. But this is not the case.



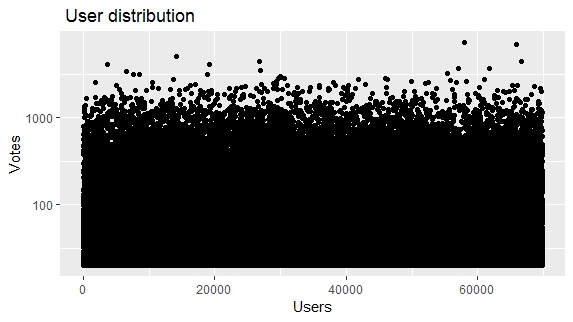
* 1. **The userId**

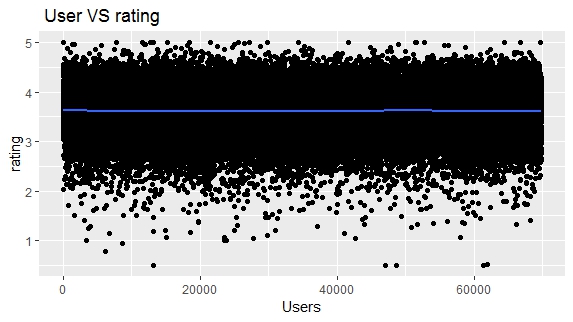
The following two plots show users vs votes and ratings; the first one shows in y axis - in the log mode scale – the amount of preferences among the users, the graphics appears quite uniform, but there is a wide range of preferences, between the most active giving thousands of ratings to the laziest who give few ratings, as the following command explains:

> range(user\_domain$votes)

[1] 20 7359

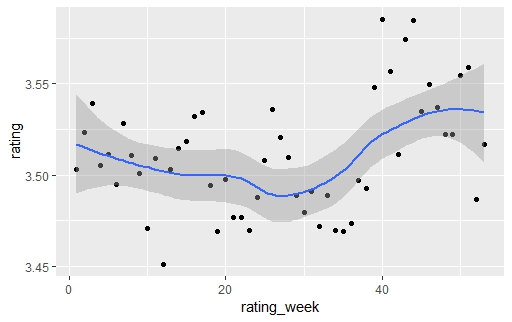
The second plot shows Users vs rating, the smoothing function does not give a very useful information, but such a wide range of votes should be taken in account when valuing RMSE.

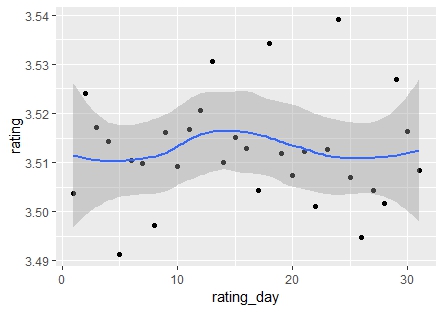




* 1. **Timestamp**

The next graphics shows the relationship between the two variables. The time base chosen for the analysis is the week; with the support of the smoothing function, it’s quite clear that the timestamp has some effect on rating, but not so strong to be considered as a predictor factor.





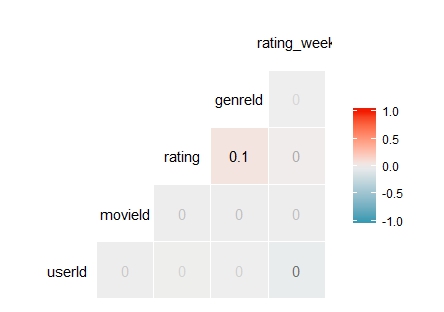
* 1. **Title**

The title has the same information of movieId (the difference is only the data type)

and could be adopted only for some visualization.

* 1. **Correlation**

At the end of data exploration, it’s important to inspect the relationship between variables, if two or more of them are correlated, they would give quite same information about prediction, so we will choose only the ones with no relationship among them. Using the R function ggcorr() from the GGally library, we can see at one glance the correlation between the numeric movielens variables, apart from not numeric variables that we early decided to keep out. Furthermore, because of the low prediction effect, rating\_week factor will not be considered. The *genreId* is related to the string variable *genres* which have been converted into numeric to be evaluated in the correlation computation.



1. **The modeling approach**

To apply the machine learning concepts the data frame must be split in two data set:

* The *training set* which will be used to train the ML algorithm
* The *test set* on which the ML algorithm trained on *training set* will be implemented to make the predictions

In this project the **movielens** df is split in the **edx** data frame as training set and the **validation** data frame as test set.

The *rating* is the variable to predict, it’s a numeric real type so the model to be used is “regression” and the metric for the evaluation is the RMSE

* 1. **Regularization and the user+movie effect approach**

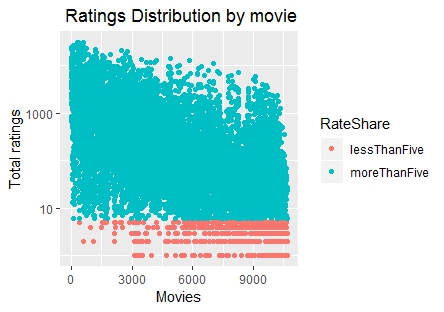
This approach is based on the processing of the following formula:

Yu,i=μ+bi+bu+εu,i

Where:

* Yu,i = The rating based on user+movie effect
* μ= The average rating
* bi= movie effect
* bu= user effect
* εu,i= error

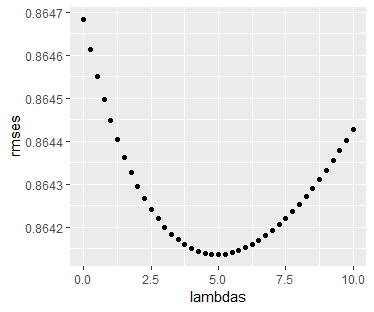
Although this is a valid approach, the rating distribution through movies, as hinted in the above paragraph, tells that some movies are rated more than others. Next graphics shows movies versus ratings. The graphic is split in two parts: the upper in blue referring to the movies with more than five ratings, the lower in red to that movies with five or lower number of ratings. The red section, just as an example, show that movies with few users ratings is a minor but a significant part of data. This are noisy data, that should be processed by regularization.



To overcome the problem of this kind of variability it must be considered a mathematical model that minimizes the effect of low number of ratings and emphasize the weight of high rated movies. The central element of this model is a parameter is λ which must be chosen after tuning operations, the formulas are:

The tuning processing consists of a CROSS VALIDATION procedure, so we must further split the edx data frame in two data frames:

The *edx\_train* that is used to calculate and and the *edx\_test* that is used to calculated RMSEs. Then, we can plot the CROSS VALIDATION results.



The lambda that minimize RMSE on the edx\_test set:

>best\_lambda <- lambdas[which.min(rmses)]

>best\_lambda

> [1] 5.00

Next, let's apply the best\_lambda on the target edx and validation sets and finally calculate predicted ratings. The RMSE is

**>rmse\_ui\_reg <- RMSE(predicted\_ratings, validation$rating)**

**> rmse\_ui\_reg**

**> [1] 0.864817**

# Customize your TOC

The space between an entry and its page number in a TOC is known as a tab leader. By default, Word makes the tab leader a row of dots (dot leader), but you can easily switch to something else, like an underline. You don’t need to start over—you don’t even need to select the TOC. Word knows where it is. Just use the Custom TOC option to make this type of change, and Word will do its thing.

Try It: Change the dot leader to an underline.

1. On the **References** tab, click **Table of Contents**, and then near the bottom, click **Custom Table of Contents**.
2. From the list of **Tab Leader** options, select **Line** (last choice in the list), and click **OK**.



1. When you’re prompted to replace the TOC, click **Yes**.



And just like that, Word found your TOC and changed the tab leader from dots to an underline.

# Remove a TOC

You can’t delete a TOC like you can a picture or other things in a doc. Well, you can, but if you do it too many times, your TOC can get out of whack. Remember the heavy lifting Word does for you? All the scaffolding needs to be removed too. Tell Word to remove the TOC, and Word will clean up after itself.

Try It: On the **References** tab, click **Table of Contents**, and then near the bottom, click **Remove Table of Contents**.

Poof! The TOC, and the stuff to make it work, are gone from the document. But you can add your TOC back any time, in any location. Word will remember everything you did—even your change to the tab leader.

**Under the hood:** The stuff Word removes is a collection of hidden bookmarks that keep track of the heading text and page number shown in the TOC.

# Explore more

If you want to customize your TOC even more, give these a try. (If you didn’t add your TOC back, do that now. You can add it above this section if you’d like. Or, if removing it is the last thing you did, press Ctrl+Z to undo.)

## Change text formatting of the TOC entries

Try it: In your TOC, select an entire Level 1 entry and make a formatting change. For example, change the font color to blue. (Make sure you select only one TOC entry, including the tab leader and page number. Notice that even though the whole TOC may look like it’s selected, the one entry you select will have darker highlighting.)



Like the rest of the TOC magic, all the TOC Level 1 entries changed too.

**Under the hood:** Okay, it’s not really magic. The TOC entries are assigned to a style (TOC 1, TOC 2, and so on), and those styles are set to update automatically whenever you make a formatting change.

## Change the number of TOC levels

Try it: Include only Heading 1 headings in your TOC, no subheadings.

1. On the **References** tab, click **Table of Contents**, and then near the bottom, click **Custom Table of Contents**.
2. Change **Show levels** to **1** and click **OK**.



1. When you’re prompted to replace the TOC, click **Yes**.
2. Verify your TOC no longer includes subheadings, such as Add a Level 2 TOC entry.

# Get help in Word

The **Tell me** search box takes you straight to commands and Help in Word.



Try it: Go to **Tell me what you want to do** near the top of the window, and then type what you want to do.

For example, type:

* **table of contents** to quickly get to the Table of Contents options and other TOC help topics
* **styles** if you want to know more about using styles in Word
* **help** to go to Word help
* **training** to see the list of Word training courses

# Let us know what you think

Please [give us feedback on this learning guide](https://go.microsoft.com/fwlink/?linkid=2027721), so we can provide content that’s truly useful and helpful. Thanks!

