Encoding categorical predictors

A summary based on *Feature Engineering and Selection: A Practical Approach* by Kuhn and Johnson

Ruben Agazzi, 844736

Hellem Carrasco, 805840

Davide Dell'Orto, 828873

Contents

Abstract

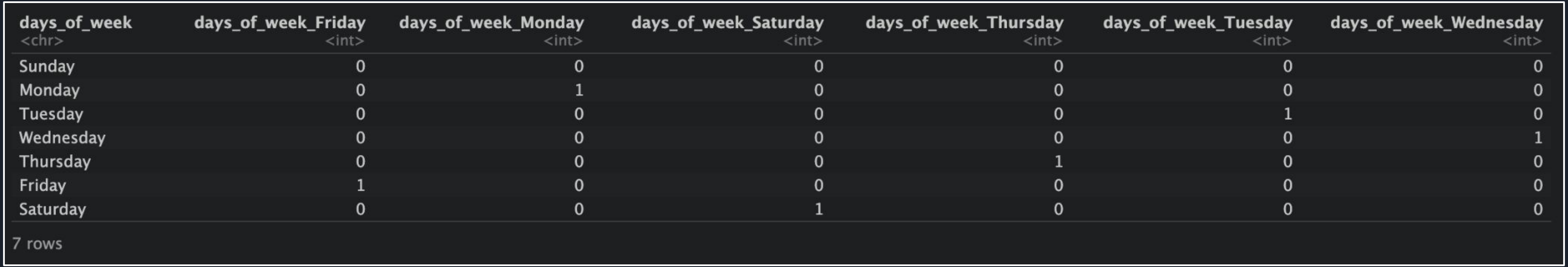
This is a comprehensive analysis on how categorial, and more generally non numerical data can be encoded into a numerical form, in order to be used by all types of models.

In particular will be addressed how to manage unordered and ordered categorical data, and also how to extract features from textual data.

5.1
Encoding of unordered categorical data

Dummy variables

The most common form used for handling unordered categorical data consists in creating binary dummy for categorical predictors. Here below, an example where the variable “days_of_week” assumes 7 values:



days_of_week <chr>	days_of_week_Friday <int>	days_of_week_Monday <int>	days_of_week_Saturday <int>	days_of_week_Thursday <int>	days_of_week_Tuesday <int>	days_of_week_Wednesday <int>
Sunday	0	0	0	0	0	0
Monday	0	1	0	0	0	0
Tuesday	0	0	0	0	1	0
Wednesday	0	0	0	0	0	1
Thursday	0	0	0	1	0	0
Friday	1	0	0	0	0	0
Saturday	0	0	1	0	0	0

7 rows

5.2
Encoding with many categories

Hashing functions

Since a categorical predictor can assume a large number of different values, if we use dummy variables we will end up with too many of them.

We can use an *hashing functions* to map the values into fewer values, and then use these to create dummy variables. The process consist of:

1- Calculate the hash from the feature, which is an integer value;

2- Calculate the number of dummy variables. If we want 16 different values we do (hashValue mod 16) + 1

Signed hashes

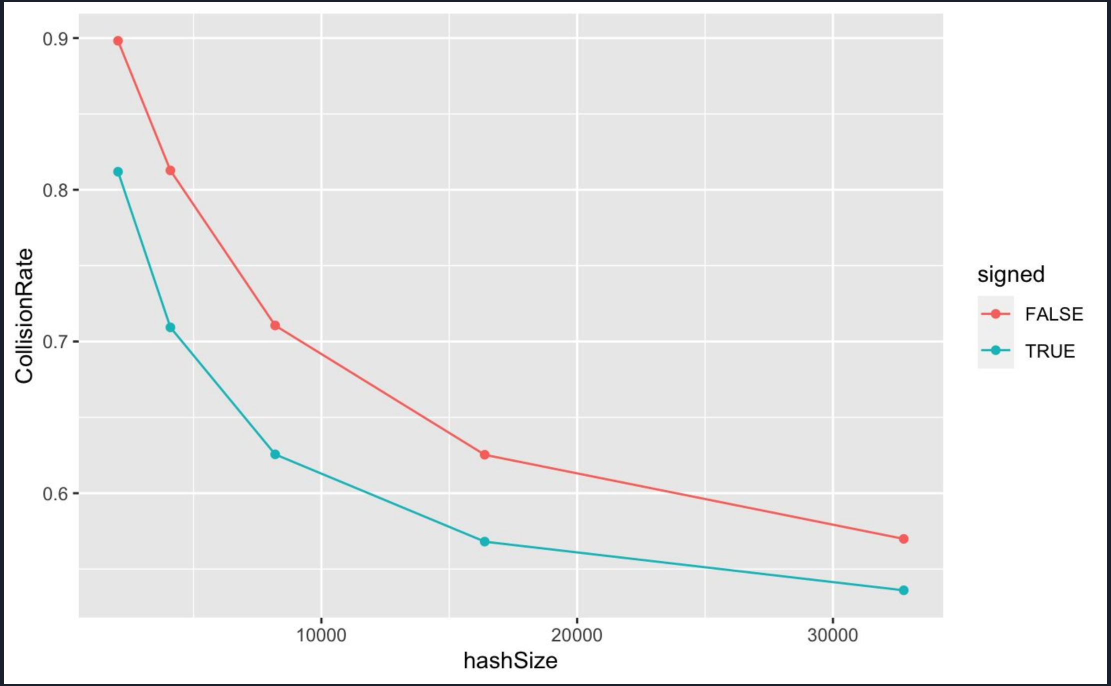
Hashes have the problem of **collision**: this occurs when two different categories are mapped to the same hash value.

A possible way to solve this problem is using “signed hashes”: signed hashes consists in hashes that can have negative values; in this case, using the same hash length, we can represent a larger number of different hashes, and this can avoid collisions.

To test this method we made a simulation, using a test predictor with 10k different categories, and the we tested different hash sizes, both signed and unsigned.

Simulation of signed feature hashing

As we can see from the chart, using signed hashes led to a reduction of collision rates, for every hash size.



Plot of collision rates with raw and signed hashes

Problems of feature hashing

In the end feature hashing can solve the problem of encoding but has a few disadvantages:

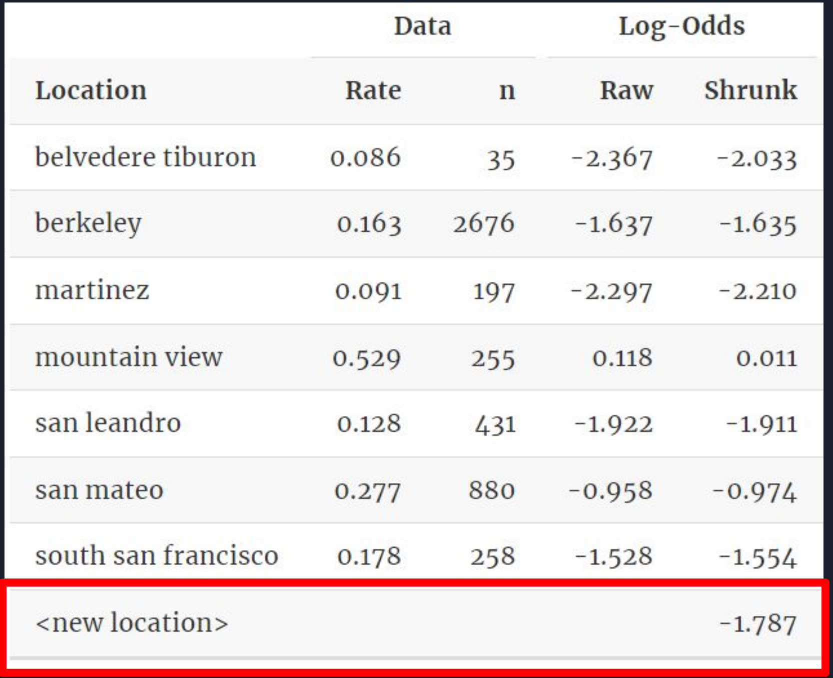
1. Collisions: some categorical values can have the same values, even though they have different values;
2. Meaning collision: if two values collide it is not guaranteed that their value is similar, even semantically;
3. Probability collision: using hashing function to encode predictors can cause collisions, and rarer categories could collide. This causes a loss of information about rarer categories, that could be, for example, incorporated into more frequent categories, making them even more prevalent.

5.3
The problem of novel categories

The problem and how to solve it

We can come across a particular problem: the models could have been trained on categorical data that don't assume all possible values.

A simple way to solve this problem consists in adding an "other" category to the dataset: this permits to adapt the model to unseen categories, just by putting the unseen categories into the "other" category.



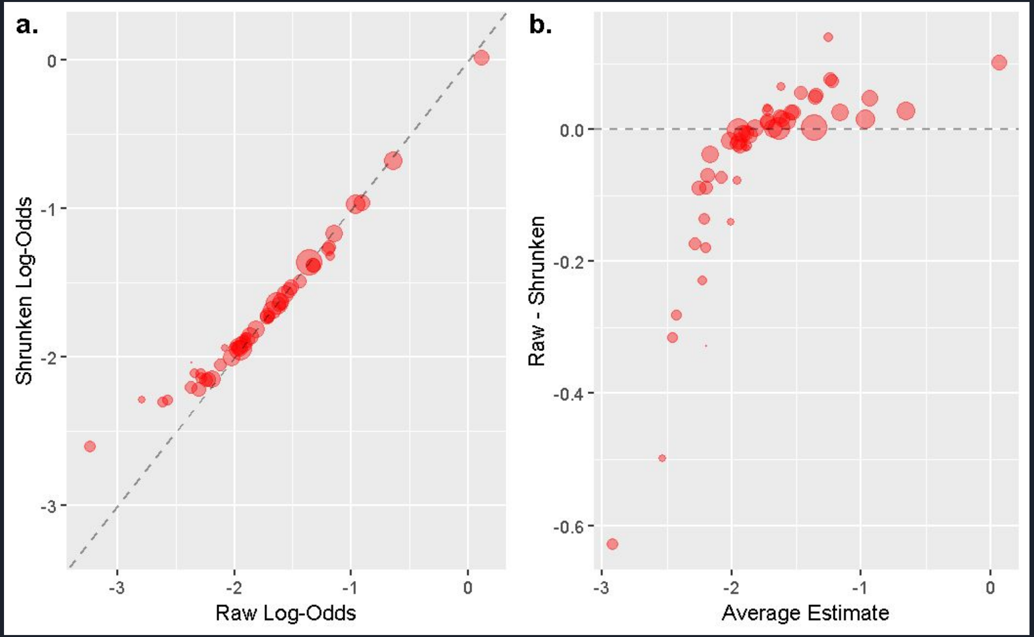
	Data		Log-Odds	
Location	Rate	n	Raw	Shrunk
belvedere tiburon	0.086	35	-2.367	-2.033
berkeley	0.163	2676	-1.637	-1.635
martinez	0.091	197	-2.297	-2.210
mountain view	0.529	255	0.118	0.011
san leandro	0.128	431	-1.922	-1.911
san mateo	0.277	880	-0.958	-0.974
south san francisco	0.178	258	-1.528	-1.554
<new location>				-1.787

5.4
Supervised encoding methods

Likelihood encoding

We can measure the effect of a categorical predictor on the outcome by calculating the odds and using this to represent the factor levels in the model.

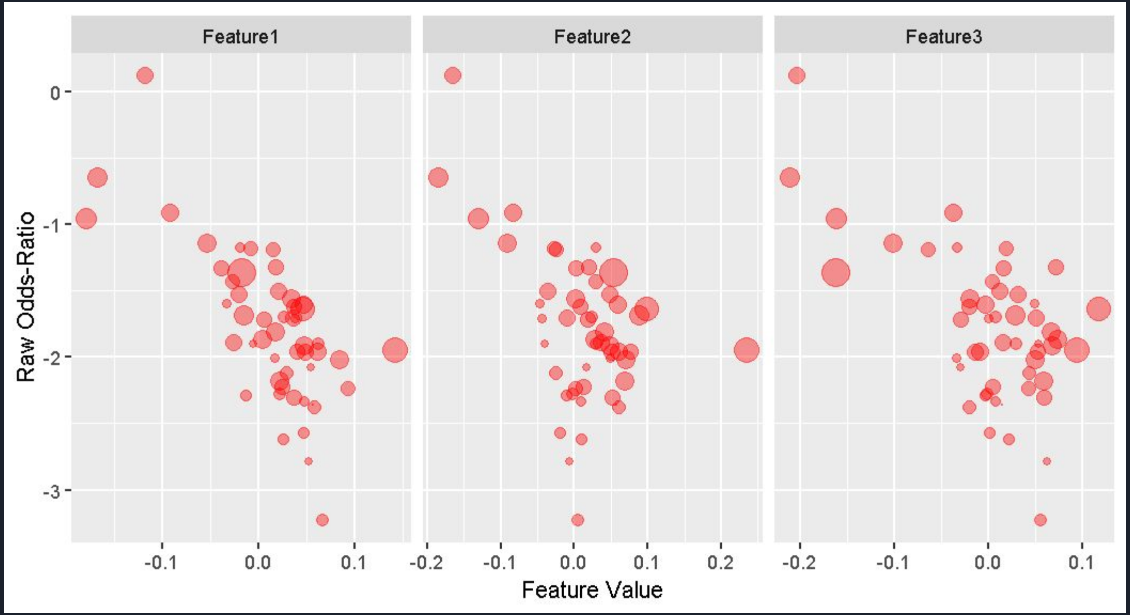
Using the log-odds representation leads, in some cases, to inaccurate values. It is possible to overcome this with shrunken log-odds obtained with **bayesian likelihood encoding**.



Word embedding

With word embedding the idea is to represent categorical predictors as vectors, using different techniques in order to avoid sparsity.

A neural network can be used where the encoding happens in the so called *hidden layers.*



5.5
Encoding of ordered data

How to encode ordered data

Ordered categorical data can have a different type of relationship between the values: for example the values can have a linear or a quadratic relationship. The encoding used to maintain this type of relationship is called *Polynomial Contrast*.

Drawbacks

1- The complexity of the contrast can be, at most, equal to the number of categories of the predictor minus one;
2- It may not relate directly the predictor to the response, for example when two levels of the categorical ordered predictor are very close;

Alternatives to polynomial contrast

1. Treat the predictors as unordered factors. This can be useful if the pattern of the categorical data is not polynomial;

2. Manually translate the categories into a set of numeric scores, using domain-specific knowledge of the data.

5.6
Creating features from text data

Preprocessing

In order to extract features from textual data we need to do some preprocessing:

- Stop-words removal
- Useless elements removal, like html tags left in the text, or like string symbols like \n
- Text normalization, like making all the text lowercase
- Removal of words with low frequency in texts
- Stemming-lemmatization

Keywords

One potential way of extracting features from text can consist in creating a predictor that indicates the presence of a certain word.

As an example, we can make a predictor indicating if an url is present or not in the profile description: with the OkCupid dataset STEM profiles with link were 1.65 times higher than the non-STEM profiles, and making a confidence interval with 95% confidence, we saw that the lower bound is 1.42, which makes this prevalence significant.

Other text features

We can obtain other types of text features, like predictors that indicates the number of exclamation points, hashtags, commas, etc. We can also make sentiment analysis in order to extract features, like creating a predictor that indicates if the text is written in first or third person or number of sentences.

Analysis of text features

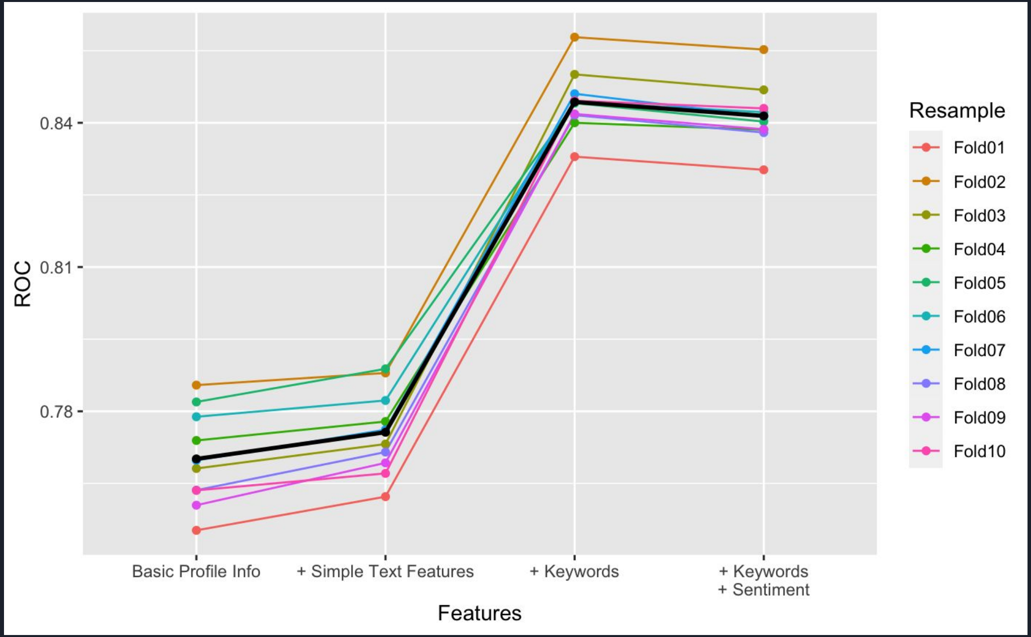
To test the effectiveness of these text features we made **4 different tests** with **4 different feature sets**:

- Basic OkCupid dataset, with dummy variables for categorical predictors;
- Basic OkCupid dataset and text features like the number of occurrences of urls, commas, exclamation points, etc.;
- Basic OkCupid dataset, text features and keyword features, like the presence of the “software” term in the description;
- Basic OkCupid dataset, text features, keyword features and sentiment features, like the number of sentences, number of sentences written in first or third person, etc;

Training

In order to do the experiment we performed a 10-folds cross validation using a GLM model.

The results showed that adding text features, greatly increases the area under the ROC curve. Only the fourth feature set lowered a bit the area, but using sentiment analysis with a different combination of features could achieve higher performances.



Alternatives

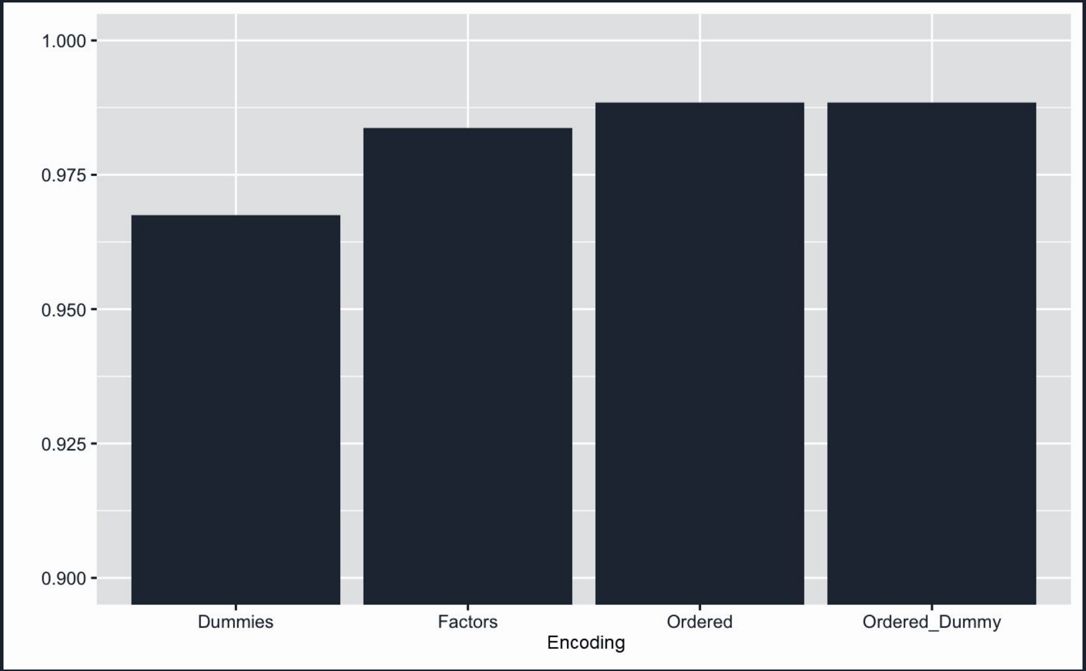
An alternative to extract features from text can consist into using the **tf-idf** representation to train the classifier: this representation takes advantage of the words frequencies into the text and with respect to all the texts in the dataset, and can also use **n-grams** and not only single words.

5.7 Factors versus dummy variables

Accuracy evaluation

The plot of the accuracy of the Boosted C5.0 Rules Models shows:

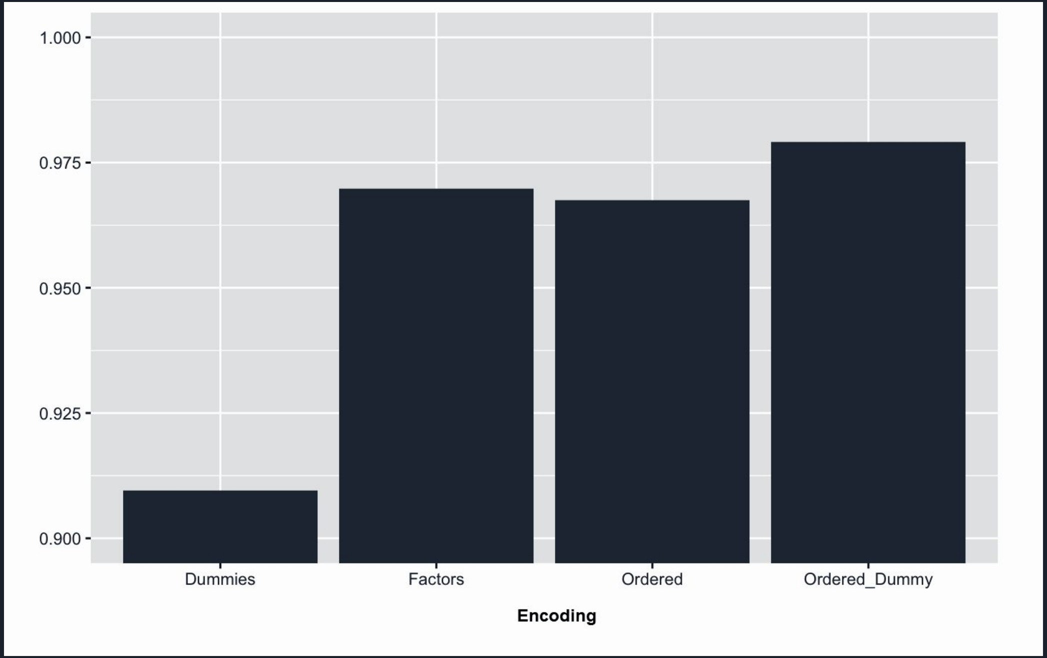
- The factor encodings are better than unordered dummy variables: this may be due to the fact that the dataset has 4 classes and all the predictors are categorical;
- No difference in the case of ordered dummy variables.



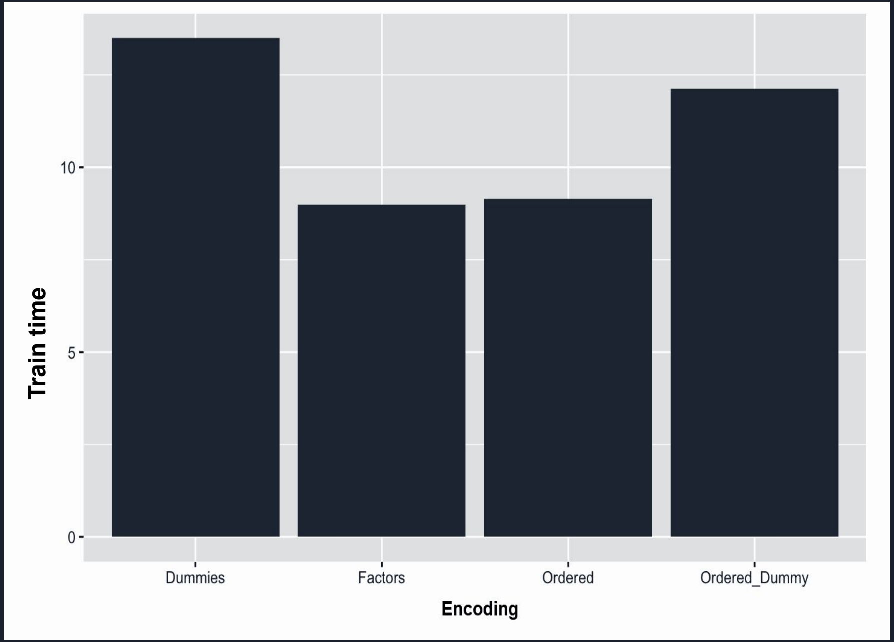
Accuracy evaluation

The plot of the accuracy of the Random Forest Model shows:

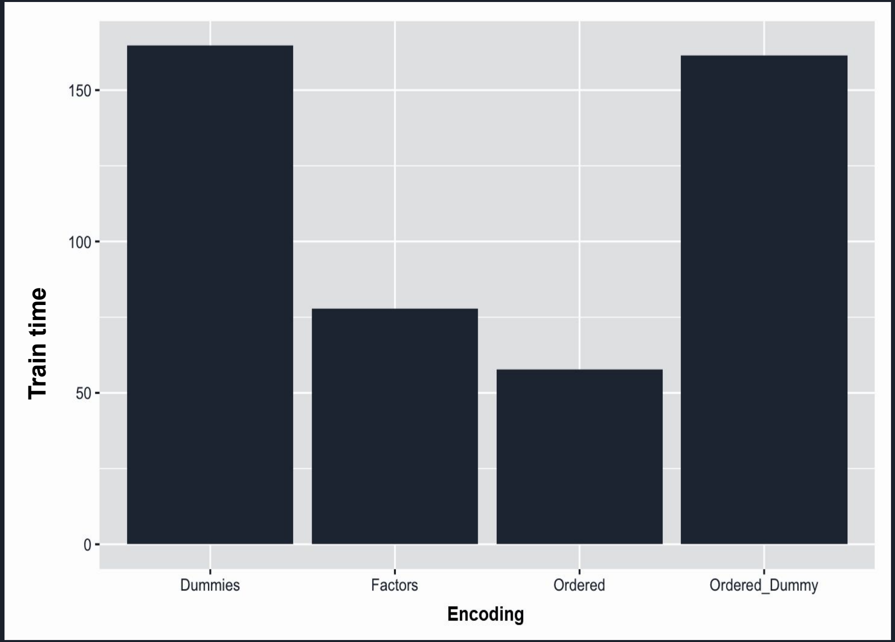
- A strong difference when using unordered dummies and factor, where the factor encoded predictor have a higher values of accuracy;
- If the predictor is ordered, the dummy encoding performs a little better than the ordered factors predictor.



Time evaluation



Training time of the Boosted C5.0 model



Training time of the Random Forest model

Thank you for your attention!