# Binary text classification of a manually annotated dataset of news articles according to food-relevant content applying machine learning models (Random Forest and SVM)

Lisa Teichmann

September 29, 2022

**Abstract**

This is a report on the document classification phase of LaDiRec recits_faim English corpus and includes a summary of the workflow, tested models, feature selection for optimization, and fine-tuning which was done from July until October 2021. The respective rscripts and data are available here: https://doi.org/10.5683/SP3/8KLAXJ.

## 1 General Purpose

Machine learning algorithms such as Random Forest classify documents in relevant and non-relevant based on decision trees based on training data representing manually classified documents. The result is a binary classification into relevant and non-relevant documents. We have a corpus of 10963 English newspaper articles of which we only want to keep the articles with food-related content and therefore apply and test different ML classification algorithms for curating our corpus of analysis.

## 2 Workflow

1. Manual text annotation by two research assistants of a total of 2225 articles extracted from Eureka.
2. Pre-processing and first training tests
3. Fine-tuning of the random forest model
4. Feature selection
5. Compiling performance results
6. Cross-validation

## 3 Manual Classification

A total of 2225 articles have been manually classified by three annotators according to a classification guide (see project documentation). Articles that fulfill the criteria set forth in the annotation guide and are categorized as relevant for the topic of food and nutrition have been labeled "2keep" or "2rmv" if not relevant. The classification resulted in an unbalanced dataset of the two categories.

| Class | Number of Articles |
|---|---|
| 2keep | 1454 |
| 2rmv | 737 |

Table 1: Number of articles per class (first training)

# 4 Pre-processing

Text has been converted to lower-case, removed punctuation and stopwords and transformed into a TF-IDF matrix in RStudio where each column represents a term and each row a document. The dataset then was split into a testing and training set with a ratio of 80/20. This resulted in a training set of 1754 (590 2rmv and 1164 2keep), and a testing set of 437 articles with 6240 features (unique terms).[1]

# 5 Training

In the initial training phase we tested four different machine-learning model to identify the model with the best results: Support-vector models (SVM), k-nearest neighbours (KNN), and Classification And Regression Trees (CART) in order to compare with the randomforest model. Based on the literature these models have been successfully used for binary text classification (add sources).

# 6 Results

The SVM model reached the highest accuracy (0.817), based on which we decided to focus on further fine-tuning and fitting it to our data, followed by CART (0.794) and KNN last (0.676).

These are the results without applying any feature selection.

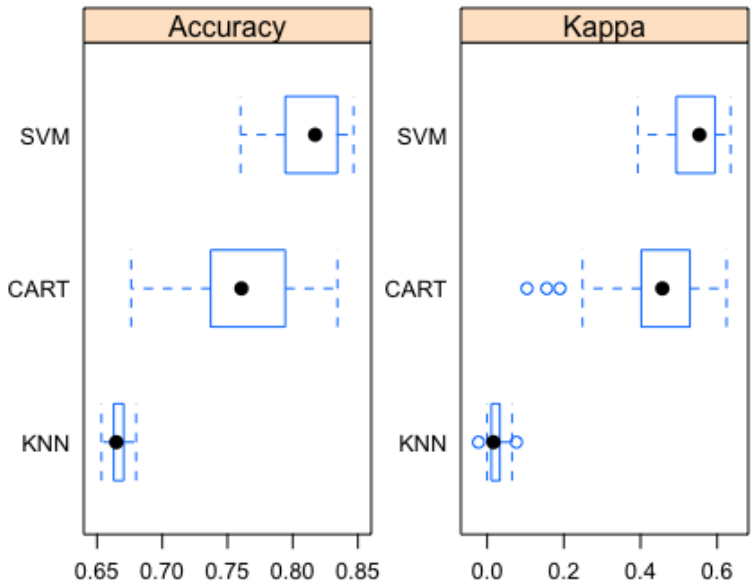


Figure 1: Accuracy of different machine-learning models. Random forest achieved the highest accuracy compared to Support-vector models (SVM), k-nearest neighbours (KNN)), and Classification And Regression Trees (CART) for 1633 articles (first round of training) after feature selection with Gini-coefficient

## 6.1 Random Forest

Before starting the training process, it was necessary to find the right mtry parameter (the number of variables available for splitting at each tree node based on the OOB error rate). This may increase the fitness of the model slightly. For our dataset, an mtry of 225 has the lowest OBB error rate (18%) and we therefore used it to train our model.

[1] We only kept documents above a threshold of 15 terms per document and a minimum 5 documents per term.

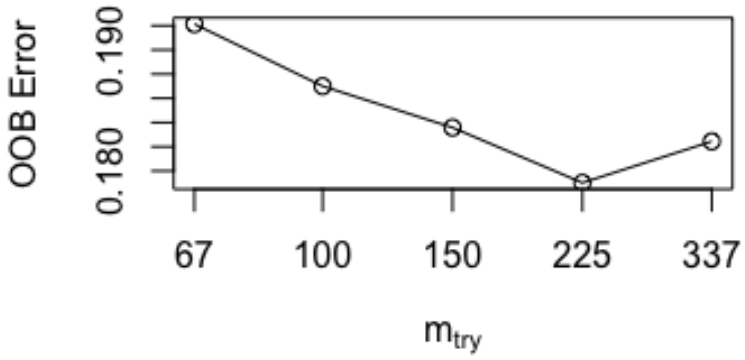


Figure 2: Mtry OOB error rates.

Only by finetuning the mtry parameter a simple randomforest model achieved an accuracy of 0.8284 and unbalanced sensitivity (0.8673) to specificity (0.8171) results. The randomforest model therefore reached a higher accuracy than the other models. As these results and table 2 shows, the model did far better at false positives than false negatives.

| class | 0 | 1 |
|---|---|---|
| 0 | 85 | 62 |
| 1 | 13 | 277 |

Table 2: Sum of false positives and false negatives versus true positives and true negatives. 0 stands for the articles to keep and 1 for the articles not to keep.

# 7 Optimization

We tried three different methods for optimization, two of which were feature selection models (mean Gini-coefficient and Boruta feature selection) as well as upsampling. As table 3 illustrates, all of the feature selection methods applied to the data significantly increased the accuracy of our classification model by succeeding 82% achieved with running a randomforest model on all features of the TFIDF. Upsampling increased accuracy comparatively more than any other feature selection method with 87% unbalanced accuracy scores.

| Measure | RF | Gini | Boruta | Upsampling | Upsampling Boruta | Upsampling Gini |
|---|---|---|---|---|---|---|
| Accuracy | 0.8284 | 0.8467 | 0.8444 | 0.8764 | 0.8513 | 0.8513 |
| Kappa | 0.5812 | 0.639 | 0.6381 | 0.7175 | 0.6617 | 0.6594 |
| Sensitivity | 0.8673 | 0.8390 | 0.8160 | 0.8444 | 0.7971 | 0.8060 |
| Specificity | 0.8171 | 0.8495 | 0.8558 | 0.8907 | 0.8763 | 0.8713 |

Table 3: Results for random forest model with and without feature selection methods.

## 7.1 Feature selection with mean-decrease Gini-coefficient

The mean decrease Gini-coefficient weights each feature by its equality in distribution (it is most commonly a measure of inequality),[HGY] so we could reduce our feature space from 6265 to 257 representing the core set of unique terms relevant for our classification by applying a mean decrease of less than 0.5. We then subsetted our trainingset by these terms and after finetuning (mtry=67) trained

the randomforest model. While this resulted in a slightly higher accuracy, the differences with the initial randomforest model are especially visible in ration between specificity (0.8171) and sensitivity (0.8673), meaning that feature selection with Gini especially increased balanced accuracy. This could be partially related to our unbalanced dataset, where non-relevant articles are underrepresented. With an accuracy of 0.8284 applying the Gini-coefficient for feature selection did result in a better model than the initial randomforest.

## 7.2 Upsampling

Since our dataset is unbalanced with relevant articles in the majority class(2keep), we had to increase the amount of non-relevant articles (2rmv) to the sum of relevant articles. This was done by duplication of features for non-relevant articles which left us with a balanced dataset of 1164 of each class in the trainingset. Upsampling the original dataset without feature selection significantly increased accuracy. With a fine-tuned model (mtry=79) the model achieved an accuracy of 0.8764. Interestingly, upsampling also contributed to increase specificity (0.8907) more than sensitivity (0.8444). Likewise, upsampling combined with Boruta also yielded an increased accuracy (0.8513), but not as much as Boruta feature selection without upsampling. Compared to the previously attained results upsampling and Boruta displays the widest gap between sensitivity (0.7971) and specificity (0.8763). Upsampling after feature-selection with Gini-coeffeicient achieved the same accuracy as Boruta and upsampling (0.7712) as well as a similar imbalance (sensitivity 0.8060, specificity : 0.8711).

As these results illustrate increasing the amount of non-relevant articles in the dataset by upsampling did not increase the model's efficiency to detect this class.

## 7.3 Feature selection with Boruta

Boruta is a feature-selection algorithm which was specifically designed to use as a wrapper for Random Forest which is why it was of interest to us. It firstly creates shuffled copies of all features (shadow features). By training a random forest classifier and applying a feature importance measure (e.g. Mean Decrease Accuracy) it calculates an importance measure based by comparing each feature's importance to its shadow feature. It then iterates over all features while shadows are re-created in each iteration until all features are confirmed or when it reaches maxRuns (in which case additional features are classified as "Tentative").[KR10]

Boruta is a time-intensive method of feature selection and when running it on the TFIDF it performed 99 iterations in 2.169424 hours (on a MacBook Pro with an M1 chip). It confirmed 39 attributes important, amongst others: albums, assault, bbq, burgers, and busy and 6171 attributes unimportant (e.g. 'function', 'repeat', à, aaron, ab).[2] It determined 30 tentative attributes (e.g. aid, alleged, andy, baked, boxes). All in all it reduced our initial list of features to 59 relevant ones. After re-training the Random Forest model only with those features - we subset the TFIDF table by the relevant Boruta features - fine-tuned to an mtry of 10, our model achieved an accuracy of 0.7506. While it achieved a lower accuracy than the initial model, it appears to be balanced for sensitivity (0.7568) and specificity (0.8713).

## 7.4 Cross-validation of misclassified articles by the RandomForest model

Cross-validation is a necessary step to establish which exact content the model was struggling with classifying. Additionally from the common topics or keywords linking articles that the model has misclassified (for instance when it predicted an article to be non-relevant which was in fact relevant according to our manual classification), the features for additional filters can be extracted.

We first extracted all articles from the final list of predicted documents by the Random Forest for which the predicted class differed from the initial class. A total of 71 articles could be identified where the predicted value did not match the initial class. Common topics or phrases which these articles have in common were amongst others mentioning animal food, food services, food supply, street food, food trucks and stands, filtered or potable water, phrases that only referred to a location (in a food court or at a restaurant), and metaphors (eye candy). Based on these common themes a list of regular expressions was assembled as an additional filter after the machine learning step. Since this method

[2] You can find the full list in our Github repo $20220218_LTclass_public_corpen_subr d_boruta_confirmed_attributes.csv$

involves the close reading of articles it can be time-intensive if the list of misclassified articles exceeds 100. Otherwise it allows to further narrow down the classification process.

### 7.5 Classification of Corpus

Using the classifier trained on the upsampled trainingset predicted 1769 as non-relevant(0) and 9194 as relevant(1).

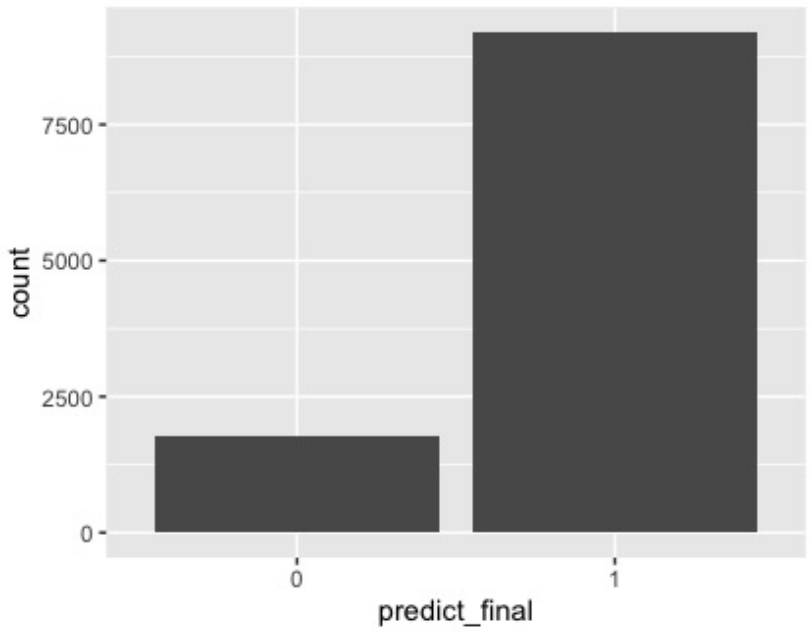


Figure 3: Randomforest predicted classification of food-related articles (n=10963 articles

### 7.6 Manual cross-validation of classified articles by the RandomForest model

For the final predicted classification we extracted 100 articles in had one of the initial annotators manually classify. Out of 100 articles classified by the randomforest model, 5 articles were arbitrary, one of which was in French. For 12 articles, the model's classification did not match with the manual annotation, two of which were classified as relevant while the model predicted them to be relevant, meaning that the model appears to be overconfident in predicting the relevant class. These articles include topics such as gardening, general health concerns, food-related phrases such as "lunch hour" without a direct mention of food or nutrition, a wrestler chewing decks of cards in half amongst other articles without any mention of food or nutrition. Amongst the articles misclassified as non-relevant by the model was one article mentioning food banks. Further work is required to avoid misclassification for these topics.

## 8 Annexe

### 8.1 Documentation of first round of results with 1633 documents

Since manual classification was done in two separate phases, initial tests of the model were performed on a smaller dataset. The first round of thests were performed on a training set of 1307, and a testing set of 326 articles with 5297 features (unique terms). As the results below show, the increased number of classified documents did not result in a significant increase of model performance.

| Class | Number of Articles |
|---|---|
| 2keep | 1037 |
| 2rmv | 596 |

Table 4: Number of articles per class (first training)

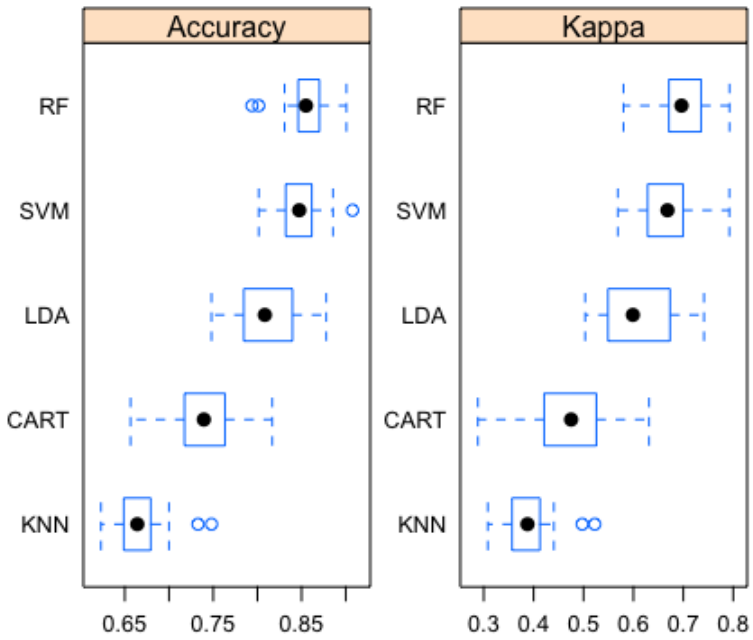


Figure 4: Accuracy of different machine-learning models. Random forest achieved the highest accuracy compared to Support-vector models (SVM), k-nearest neighbours (KNN), Latent Dirichlet allocation (LDA), and Classification And Regression Trees (CART) for 1633 articles (first round of training) with Gini-feature selection

A simple Random forest model with mtry=22 and feature selection with Gini-coefficient resulting in 150 terms (features), provided the following results:

| | |
|---|---|
| Accuracy | 0.8067 |
| Kappa | 0.57 |
| Sensitivity | 0.75 |
| Specificity | 0.83 |

Table 5: Results for random forest model with reduced features by gini-coefficient (first training with 1633 articles).

## References

[HGY] Hong Han, Xiaoling Guo, and Hua Yu. Variable selection using mean decrease accuracy and mean decrease gini based on random forest | IEEE conference publication | IEEE xplore.

[KR10] Miron B. Kursa and Witold R. Rudnicki. Feature selection with the Boruta package. *Journal of Statistical Software*, 36(11):1–13, 2010.