SAKI SS 2021 Homework 1

Author: Nicolas Webersinke

Program code: https://git.io/JsTTJ

Summary

The problem at hand is a classification task for various current account (checking account) transactions. The data set given consists of 209 transactions with 10 features and labels. Besides technical details, the features comprise booking text, purpose and recipient, as well as the amount. The dataset is annotated with 6 different labels, one of which is assigned to each transaction. Table 1 shows the label names and their respective counts.

Label name	Count
leisure	65
standardOfLiving	47
finance	33
living	26
private	21
income	17

Table 1: Label names and their respective counts.

The data set is obviously imbalanced, which must be taken into account for the fitting and evaluating of the classifier. For example, the number of transactions with the label "private" is only 21 and for those with the label "income" only 17. While I would expect that due to the homogeneity of income transactions the classifier will perform well despite a small number of samples for these, for transactions labeled "private" the classifier is more likely to perform poorly. This could be due to the heterogeneity of transactions expected for a generic term like “private”.

The classification is performed using a Naive Bayes classifier. Before fitting the Naive Bayes classifier to the data, I perform several preprocessing steps to improve accuracy and generalizability. First, all features except for “Buchungstext”, “Verwendungszweck” and “Beguenstigter/Zahlungspflichtiger” are removed. The reason for this is that differentiability and generalizability are not expected to be high for the other features, which primarily consist of technical details needed for processing those transactions. Second, the individual strings of the remaining textual features are merged into one string. Finally, unnecessary punctuation and duplicate white spaces are removed.

After these steps, the task becomes a Natural Language Processing (NLP) problem, since only one (long) string is left as a feature for each transaction. These strings are essentially lists of keywords and must be transformed into a numerical representation for classification with Naive Bayes. For this purpose, simple methods exist that, for example, count the words and create a

count vector for each string (bag-of-words approach), as well as sophisticated methods that take the context of the words into account and allow for a distributed representation. Since we are only dealing with keywords here, a representation by count vectors should already lead to good results. Before transforming the strings to count vectors, I split the data into 70% training and 30% test data, which is a common split in the literature. It is important to do that before fitting Scikit's CountVectorizer because otherwise the test data would no longer provide realistic evaluation results. Consequently, the CountVectorizer is fitted to the training data only and then the training data and the test data are transformed. Before tokenization all characters are converted to lowercase and accents get removed. Moreover, unigrams and bigrams are extracted to improve accuracy. It would also be possible to determine the vectors based on term frequency-inverse document frequency (tf-idf). However, this approach should not lead to better results because only a few, mostly unique keywords are available per transaction.

Subsequently, the Naive Bayes classifier is fitted to the training sample with no prior probabilities of the classes specified. Since the input data are discrete numbers, I apply the multinomial Naive Bayes classifier. The Gaussian Naive Bayes classifier would be inappropriate because the input data is obviously not normally distributed.[1]

Evaluation

The fitted classifier is now used to classify the transactions in the test data set. To evaluate the predicted labels, I calculate different evaluation metrics appropriate for multi-class problems. In particular, these are precision, recall, F1-score, accuracy and various averages suitable for imbalanced data sets. Noteworthy here is the accuracy, which measures the percentage of transactions with correctly assigned labels out of all transactions of the test set.

The screenshot on the next page shows results for the mentioned metrics and the corresponding confusion matrix. Due to the uneven distribution of labels, it is important to also include support for interpretation. Overall, the accuracy of 90% can be considered high. Nevertheless, the results for some labels, such as the "private" label, should be treated with caution, since only a few transactions were included in the test sample. Precision, recall, and F1-score illustrate that the classifier performs better than random assignment. The results shown are from the final tuned algorithm. I tried various options discussed in the summary, such as using tf-idf vectors, but as expected they did not lead to better results. Different values for the hyperparameters, such as for the alpha of the Naive Bayes classifier, were also tried and set to minimize the error or maximize the accuracy. For robustness, I perform a 5-fold cross-validation, too. This seems reasonable due to the small sample size. The results are slightly lower in terms of accuracy, but still good.

While this approach certainly has potential for improvement, the metrics show that it is quite suitable as a solid baseline model. Possible optimization approaches would be, for example, the identification of companies by performing Named Entity Recognition and the provision of a knowledge base in which companies are assigned to specific labels.

[1] See https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes.

Screenshot

	precision	recall	f1-score	support
finance	1.00	0.88	0.93	8
income	1.00	1.00	1.00	8
leisure	0.86	0.96	0.91	25
living	1.00	0.75	0.86	8
private	0.80	1.00	0.89	4
standardOfLiving	0.89	0.80	0.84	10
accuracy			0.90	63
macro avg	0.92	0.90	0.90	63
weighted avg	0.91	0.90	0.90	63