Appendix to: *Machine Learning for Occupation Coding - A Comparison Study*

Malte Schierholz
Institute for Employment Research, Nuremberg, Germany

Matthias Schonlau
University of Waterloo, Waterloo, Canada

Contents

1 Part A: Study Descriptions and Data

Data set A.
The survey '*Working and learning in a changing world*' was conducted by the German Institute for Employment Research (IAB) to study the pathways how informal competencies and knowledge support professional careers. A clustered sample of persons living in Germany and born between 1956 and 1988 was surveyed in 2007/2008 with computer-assisted telephone interviews (Antoni et al. 2010). Among other themes, the questionnaire contained questions about the employment biography, i.e., about all the jobs that each person has held during her lifetime. Two questions were asked to collect detailed information about each job. The first question was about the 'berufliche Tätigkeit' (occupational activity) and the second question asked for a more precise activity description. For the first question, a total of 32,931 verbal answers from 9,230 persons is available. For the second question, many respondents were less motivated and their answers are either identical to their first answer or not meaningful for coding (e.g., "don't know"). After replacing such non-informative answers with empty strings, we are left with 19,330 answers for the second question. For both text fields, the inputs were restricted to have at most 50 characters and additional text beyond this threshold was discarded.

Drasch et al. (2012) described the subsequent coding process: Instead of using the KldB 2010 for coding, answers were assigned to the *Dokumentationskennziffer*, an internal list of job titles that is continuously updated by the German Federal Employment Agency. Transition tables are available to convert the assigned job titles into classification codes. To select appropriate job titles from the *Dokumentationskennziffer*, a three-step process was developed, involving (1) automatic coding, (2) manual coding by specially trained coders and (3) manual coding by supervisors. For automatic coding, the respondents' verbal answers were compared with the job titles from the *Dokumentationskennziffer* and with another database of search words. A job title was selected automatically if an entry was identical with the verbal answer, applicable for 39% of the answers. Human coders were involved in this automatic process only seldom, that is, only when the automatic text matching suggested more than a single job title and a human decision was required. The residual answers (61%) were coded by human coders and their supervisors who were trained to follow a set of rules. Their task was to enter the verbal answers from respondents in a web mask and to choose an appropriate job title from a list of suggestions. Additional answers from respondents about each job were available to coders and, as a fundamental rule, coders were required to select only job titles, if the job titles' usual educational requirement level is appropriate when compared to the differentiated occupational status as reported by the respondents. Final quality assurance showed that the inter-coder reliability of manual coding is 50% for job titles and improves when these job titles are grouped together into higher-level units of occupational classifications (65% inter-coder reliability for the 4-digit KldB 2010).

Data set B.
The '*BIBB/BAuA Employment Survey 2012*' was conducted by the German Federal Institute for Vocational Education and Training (BIBB) in cooperation with the Ger-

man Federal Institute for Occupational Safety and Health (BAuA) to collect data about employees and workplaces, focusing on working conditions and qualifications. The population consisted of persons aged 15 or older in paid employment for at least ten hours per week. Random digit dialing was used to select respondents and, if a short screening interview proved their eligibility, their data were collected in computer-assisted telephone interviews. 20.036 persons participated in the survey (Rohrbach-Schmidt and Hall 2013, Hall et al. 2015). A single open-ended question asked for a 'Tätigkeitsbezeichnung' (occupational activity title). Implementing an automatic filtering mechanism, the answer was then compared with a list of general job titles to decide if asking for further specification with a second open-ended question was necessary. Respondents did not only answer a question about their current occupation (20.031 verbal answers), but also about the first occupation they had in their life (17.379 answers) and about the occupation of a parent when they were 15 years old (16.130 answers about fathers and 2.404 answers about mothers), yielding a total of 55.944 answers that we pool for our analysis.

Hartmann et al. (2012) described the rules and procedures applied for coding the verbal answers into the 2010 GCO. If possible, answers were coded automatically using a coding index or other hand-crafted rules, which are based on information from other survey variables. Answers that were not codable automatically were coded manually by professionals, who followed a set of general and specific rules. Quality measures of the coding process are not published.

In addition to the categories from the 2010 GCO, 22 special categories were used for coding. 18 of those special codes (e.g., 'Technician', 'Engineer', 'Kaufmann (=Merchant / Business Administration)', ...) comprising 1.346 verbal answers are not directly translatable to any of the special codes that we use for our analysis. In order to still obtain comparable coding schemes among our data sets, we aggregate those codes and move the answers into the special category 'Job description not precise enough / Uncodeable'. This recoding procedure partly explains the high proportion of uncodables in this data set.

Data set C.

The '*German Health Update 2014/2015*', harmonized with the '*European Health Interview Survey*' (GEDA 2014/2015-EHIS), was carried out by the German Robert Koch Institute (RKI) to obtain representative statistics related to public health. The population consisted of all persons aged 15 or older with permanent residence in Germany. Based on a two-stage stratified cluster sampling approach using official population registries, selected residents were invited per mail to fill out self-administered questionnaires, either online or paper-based. In total, 24.824 individuals responded, for a response rate (AAPOR RR 1) of 27.6% (Lange et al. 2017). Persons who were employed at the time of the interview were asked in detail about their current occupation.

A related study, the '*German Health Interview and Examination Survey for Children and Adolescents*' (KiGGS) was designed to monitor public health in the next generation. We use data from wave two, conducted between September 2014 and June 2017 by the Robert Koch Institute (RKI), which has both a cross-sectional and a longitudinal component. The population from the cross-sectional component consists of all children aged 0 to 17 with permanent residence in Germany, sampled with a two-stage

stratified cluster approach. 15.023 children participated, for a response rate of 40.1% (AAPOR RR 2) (Hoffmann et al. 2018). The longitudinal component is a follow-up of a similar study that took place between 2003 and 2006. In wave two, grown-up children of age 10 to 31 were recontacted and invited to participate again. 10.853 children participated (61.5% of the baseline study). Data for both components were collected in temporary medical examination centers and with self-administered questionnaires. As the unit of interest is a child, whole families were actually involved if children were younger than the legal age (18). Most families and grown-up children received a printed version of the questionnaire, but a few children aged 18 and older were also invited to participate in a web survey (Lange et al. 2018). For children aged 17 and younger, both parents were asked about their respective current occupations in a background questionnaire. Children aged 18 and older were asked about their own current occupation (Mauz et al. 2017).

Both studies used (mostly) self-administered paper questionnaires with question wordings that are nearly identical regarding the occupational questions. All currently employed persons were asked for a precise job title in a first open-ended question. A second open-ended questions was then asked to obtain further details about the occupational activity. From the German Health Update (GEDA) we obtain verbal answers from 14.143 currently employed respondents and from the KiGGS we obtain current occupations from 15.425 fathers, 15.507 occupations of mothers, and 3.919 occupations of children aged 18 to 31. Since a single coding unit at the Robert Koch Institute coded all answers from both studies and the data are also in other relevant aspects quite similar, we pool verbal job descriptions from both studies, yielding a total of 48.994 answers.

Albrecht et al. (2017) described the coding process and its evaluation. The first step necessary for coding was to capture the input texts from paper questionnaires in a digital format. In a second step, answers were coded according to the 2010 GCO. This was done automatically using a coding index whenever possible (approx. 20% of all answers). Remaining cases were coded manually by trained coders. A computer-assisted coding software was developed to assist coders and supervisors. Regular meetings between coders and supervisors took place to discuss difficult decisions and to improve the process. To evaluate the quality of coding, approx. 20% of the answers from the German Health Update (GEDA) were double-coded by a second independent coder and, subsequently, an expert decided which of the two codes was more appropriate. For 77.7% of the cases both codes were equally correct (either being identical or according to expert judgment), for 6.7% of the cases only the first code was considered correct, for 3.4% the first code was better, but the second code was considered possible, for 4.1% the second code was better, but the first code was considered possible, and for 8.1% only the second code was considered correct, implying that the decision from the first coder was wrong.

Data set D.

The study '*Labour Market and Social Security*' is an ongoing panel study at the German Institute for Employment Research (IAB), designed to analyze long-term unemployment and accompanying welfare benefits in Germany (Trappmann et al. 2013). Computer-assisted personal interviews and telephone interviews are employed for data

collection. For our study, we use all job descriptions that were collected during the 10th wave of the panel, carried out in 2016 (Berg et al. 2017). Since the occupation is often already known from previous waves, not everyone was asked about it. 4309 respondents, some of them living together in the same households, replied to occupational questions. The extensive questionnaire contained filter questions to ensure that respondents were asked only questions relevant to them. Occupational questions about the current and all recent occupations since 2014 (2411 answers and additional 64 answers from elderly people), about current minijobs (1365 answers), about the respondents' first and last job in their lifetime (938/449 answers) and about the occupations of mothers and fathers when the respondent was 15 years old (846/1566 answers) were asked, yielding a total of 7639 occupational descriptions that we pool for our analysis. The question wording was about the 'berufliche Tätigkeit' (occupational activity). Interviewers were instructed to probe for a job title when the first answer was not precise enough. Unlike other studies used here, the verbal answers were recorded in a single text field.

A team of trained coders coded the verbal answers into the KldB 2010. Coders had access to answers from additional questions providing background information about the respondents' jobs. For assistance, code suggestions were sometimes available from an automatic textual comparison with a coding index. Two coders worked independently on each answer. The second coder did not know the code from the first coder, but an automatic indicator of agreement was available, such that the second coder could reconsider and correct his decision. The agreement rates between both coders range from 67.2% to 80.2%, depending on the specific persons involved. If no agreement was reached, the final decision was made by a third person or, for the most difficult cases, in a group discussion. Only the final codes from this process are used in our analysis.

Data set E.

The survey '*Selectivity effects in address handling*' was conducted in 2014 by the Institute for Employment Research (IAB) to analyze a number of methodological issues. Individuals were drawn at random from a German federal database used in the social security administration. Valid computer-assisted telephone interviews were conducted with 1.208 persons (Sakshaug et al. 2016). Respondents were asked about their current occupation or—for the unemployed—about their most recent occupation and, as a result, verbal answers on occupation were collected from 1.064 individuals.

A central element of the study was testing a new instrument for occupation coding, which applied supervised learning techniques to predict possible job titles at the time of the interview. The most probable job titles were suggested to the respondents who in turn could choose the most appropriate occupation. To evaluate the new instrument and assess its quality, all answers were coded into the KldB 2010 by two independent coding professionals. They were given access to several occupation-related background variables. Only codes from one of the two coders are used in our subsequent analysis. To obtain those codes, the professional coder used automatic coding as a first step, that is, the verbal answers were compared textually with a coding index. The correctness of the so-found codes was checked manually. In a second step, leftover cases were coded manually, hereby following a few priority rules concerning how to interpret answers

with conflicting information and how to code ambiguous cases. A separate indicator variable is part of the data, stating for three observations that the "level of qualification [is] unknown, lowest is coded" and for another 19 observations that "multiple codes [are] possible, decision made". 1042 observations have no such cautionary note in the data. The quality from this coder is as follows: The second professional coder, who worked independently from the first and applied different coding procedures and rules, obtains for 60.7% of the answers identical 5-digit KldB codes. Moreover, the quality was assessed by two student assistants, who were given access to the complete information and all available codes. Their respective findings are that 89.6%/93.2% of the assigned codes are "acceptable" (different codes were allowed to be acceptable for the same individual), 6.4%/1.1% are "uncertain" and 4.0%/5.7% of the codes are considered "wrong". Schierholz et al. (2018) described the study and the complete evaluation in more detail.

1.1 Question Wording

The general recommendation to collect occupational data in Germany is to ask three open-ended questions, the first asking for a 'berufliche Tätigkeit' (occupational activity), the second question should ask for a more precise occupational activity description, and the third question should ask if this occupation has yet another job title (Statistisches Bundesamt 2016). No study that we use fully implemented this time-demanding protocol requiring three questions, but all studies ask for either a job title, or an occupational activity, or both. Some questionnaires explicitly ask respondents to provide a precise answer, whereas others mention this in the instructions given to the interviewers.

The following surveys have published their questionnaires online:

- Working and learning in a changing world (data set A): http://doku.iab.de/fdz/reporte/2010/DR_02-10.pdf
- BIBB/BAuA Employment Survey 2012 (data set B): https://metadaten.bibb.de/download/732
- Labour Market and Social Security (data set D)[1]: https://fdz.iab.de/de/FDZ_Individual_Data/PASS/PASS-SUF0617.aspx
- Selectivity effects in address handling (data set E): https://rss.onlinelibrary.wiley.com/action/downloadSupplement?doi=10.1111%2Frssa.12297&attachmentId=2205795635

The questionnaires of the '*German Health Update 2014/2015*' and of the '*German Health Interview and Examination Survey for Children and Adolescents*' (data set C) are not available on-line. We report the question wording of the two open-ended questions in the following. Additional job-related variables were available for coding.

[1] Answers from both open-ended questions are written down in a single text field. The second question is only asked if the first question is not precise enough.

What is your current occupation?
Please specify the precise job title, not the vocational degree or the rank.
For example:

- Flower seller (not seller)
- Bricklayer (not construction worker)
- Management consultant (not business administration graduate)

To facilitate the classification of your job, please add further explanations in keywords.
For example:

- Customer service, sales, packaging of plants (as a flower seller)
- Customs investigation, operational planning, public relations (as a customs officer)
- Maintenance, repair, equipment of vehicles, workshop management (as an automotive mechanic)

If you have management responsibilities, please mention this.

Question Wording in the '*German Health Update 2014/2015*' and in the '*German Health Interview and Examination Survey for Children and Adolescents*'

2 Part B: Evaluation Metrics

To describe the metrics, we first clarify our notation. $y_{n_f} = (y_{n_f 1}, ..., y_{n_f K})$ is a 0-1-vector (the element at the k-th position equals 1 and all other elements are 0), indicating the 'true' category of individual $n_f = 1, ..., N_f$ from the test data. If an algorithm outputs probabilities, the probabilities predicted for individual n_f are denoted by $\hat{p}_{n_f} = (\hat{p}_{n_f 1}, ..., \hat{p}_{n_f K})$. The predicted probabilities can be transformed into category suggestions $\hat{y}_{n_f}^{(m)} = (\hat{y}_{n_f 1}, ..., \hat{y}_{n_f K})$ by setting the m highest-ranked elements from $\hat{p}_{n_f}$ to 1. In the main paper m always equals 1, but higher integers might be used as well, allowing to evaluate the top-m-suggestions. Using observed categories y_{n_f} and suggested categories $\hat{y}_{n_f}^{(m)}$, we create a scalar indicator of agreement. $a_{n_f}^{(m)} := \hat{y}_{n_f}^{(m)} y_{n_f}^T$ is 1 for individual n_f only iff the 'true' category is among the m highest-ranked categories. The predicted probability that the 'true' category is among the top-m-suggestions, $\hat{p}_{n_f}^{(m)}$, is calculated as a sum over the m most probable categories, $\hat{p}_{n_f}^{(m)} := \hat{y}_{n_f}^{(m)} \hat{p}_{n_f}^T$. In the main text we use $a_{n_f}^{(1)}$ and $\hat{p}_{n_f}^{(1)}$ to calculate agreement rates among top 1.

2.1 True Predictions and False Predictions, Production Rates and Agreement Rates

Production rates and agreement rates are described in the main text. For completeness we provide the mathematical definitions here. m and t are two values a user needs to select based on their particular application. For example, we would use $m = 1$ and $t = 0$ to obtain the agreement rate at 100% production rate.

The number of individuals that will be coded automatically (where $\hat{p}_{n_f}^{(m)}$ exceeds the threshold t) and in agreement with the test data is called the

$$\textit{Number of true predictions}_t^{(m)} = \sum_{n_f : \hat{p}_{n_f}^{(m)} > t} a_{n_f}^{(m)}$$

The number of individuals that will be coded automatically but not in agreement with the test data is called the

$$\textit{Number of false predictions}_t^{(m)} = \sum_{n_f : \hat{p}_{n_f}^{(m)} > t} 1 - a_{n_f}^{(m)}$$

The proportion of cases that go into automatic coding is called the

$$\textit{Production rate}_t^{(m)} = \frac{\text{Number of true predictions}_t^{(m)} + \text{Number of false predictions}_t^{(m)}}{N_f}$$

The proportion of successes among all automatic codings is called the

$$\textit{Agreement rate}_t^{(m)} = \frac{\text{Number of true predictions}_t^{(m)}}{\text{Number of true predictions}_t^{(m)} + \text{Number of false predictions}_t^{(m)}}$$

Agreement rate$_0^{(1)}$ is typically called the *accuracy*.

Our use of production rates and agreement rates is inspired by Chen et al. (1993), who used similar definitions to calculate separate thresholds for each category.

Agreement rates and production rates are meaningful and easily interpretable terms in our context. Yet, we caution that the agreement rate is an estimator having high variance at low production rates, making it prone to over-interpretation. In addition, since the agreement rate averages over easier-to-code and harder-to-code individuals, this number obscures for the harder-to-code individuals how successful an automated coding procedure is and if it is useful at all.

For this reason we prefer for our own inspection sometimes a different type of presentation, plotting the number of true predictions versus the number of false predictions as the threshold to use automatic coding increases. Formally, one can calculate the Number of true predictions$_t^{(m)}$ and the Number of false predictions$_t^{(m)}$ if the Production rate$_t^{(m)}$ and the Agreement rate$_t^{(m)}$ are known (a bijective transformation exists between both diagrams).

$$\text{Number of true predictions}_t^{(m)} = N_f \cdot \text{Production rate}_t^{(m)} \cdot \text{Agreement rate}_t^{(m)}$$
$$\text{Number of false predictions}_t^{(m)} = N_f \cdot \text{Production rate}_t^{(m)} \cdot (1 - \text{Agreement rate}_t^{(m)})$$

Thus, both diagrams are equivalent in a mathematical sense. Yet, both diagrams have their role and we exemplify the diagram plotting true predictions against false predictions in Figure A7 so that readers get an impression which type of diagram is more useful for their own purposes.

2.2 Probabilistic Forecast Evaluation: Calibration, Sharpness, and Scoring Rules

Motivated through probability theory, penalized multinomial logistic regression (algorithm 5) and tree boosting (algorithms 6 and 7) minimize the binomial log-likelihood in the training data. We hence may expect that predicted probabilities from these algorithms and true probabilities of some data-generating process are identical if the estimated function correctly models the data-generating process. With this background, it makes sense to look at some criteria that evaluate predicted probabilities directly instead of only looking at the predicted categories. In the following we describe three measures developed in the field of probabilistic forecast verification (see, e.g., Jollife and Stephenson 2012, Bröcker 2012, for an overview) that we will report on in Section 4 (Appendix Part D: Model Tuning). The log loss in the test data is our main criterion for model selection.

We evaluate probabilistic predictions with respect to three criteria: (1) calibration (= reliability), (2) sharpness, and (3) log loss. Calibration refers to the desired attribute that observed relative frequencies should realize as often as the predicted probabilities suggest. Sharpness refers to the degree of certainty in predictions about future values. Log loss is an overall score, incorporating both calibration and sharpness, that has been shown to be ideal under certain reasonable conditions.

We assess *calibration* using reliability diagrams that compare predicted probabilities with their observed counterparts (Hsu and Murphy 1986, Bröcker and Smith 2007). Usually, reliability diagrams are used for binary outcomes. Since our application has K outcome categories, we extend this as follows. The probabilities to be accurate among top m (all results will be reported with $m = 1$ only), $\hat{p}_{n_f}^{(m)}$, are partitioned into ten equal-length segments $[0, 0.1), ..., [0.9, 1]$. We then average the probabilities within each segment and plot them against their observed averaged accuracies $a_{n_f}^{(m)}$.

Sharpness is another desired property of good predictions. The idea is that with little background information the forecast about a future event should be rather vague, i.e., the forecast should allow for many different outcomes, but as more background information becomes available (e.g., more information about a respondent's job), predictions should become sharper, i.e., the certainty about the outcome should increase. As a result, one can obtain better and worse forecasts, all being calibrated, which means that calibration alone is not sufficient to evaluate probabilistic forecasts. Gneiting et al. (2007) argued in the context of forecasting continuous variables that the optimal forecast should maximize sharpness subject to calibration. Ideally, we would like that probabilistic predictions are certain, i.e., they should predict an outcome with probability 1.

Following Murphy and Epstein (1967) and Potts (2012), we use the information-theoretic measure of entropy, H, and average over it to determine sharpness $\bar{H}$,

$$\bar{H} = \frac{1}{N_f} \sum_{n_f=1}^{N_f} H_{n_f} = \frac{1}{N_f} \sum_{n_f=1}^{N_f} (-\sum_{k=1}^{K} \hat{p}_{n_f k} \log_2 \hat{p}_{n_f k}) \tag{1}$$

If $\hat{p}_{n_f k} = 0$, the term $\hat{p}_{n_f k} \log_2 \hat{p}_{n_f k}$ is not defined and, as a workaround, we will set it to zero.

The minimal value is zero (optimal sharpness) if all forecasts predict a single outcome with probability 1. The maximum (no sharpness) occurs if all categories have equal probability, $\hat{p}_{n_f k} = \frac{1}{K}$ for all individuals and all k. Note that sharpness is a property of the predictive distribution only and is not related to values that will eventually realize. Each component H_{n_f} may be interpreted as the average information content, i.e., the expected minimal number of bits that would need to be transferred between two persons if both knew $\hat{p}_{n_f}$ and one person wants to inform the other about an expected realization drawn from this distribution. To acknowledge sample variation in the test data, we calculate standard errors using the formula $\sigma(\bar{H}) = \sqrt{\frac{1}{N_f} \hat{\mathbb{V}ar}(H_{n_f})} = \sqrt{\frac{1}{N_f(N_f-1)} \sum_{n_f=1}^{N_f} (H_{n_f} - \bar{H})^2}$

A general score to evaluate probabilistic predictions is the *log loss* (or ignorance score). It is the average negative $\log$ probability of the categories that will actually realize in the test data,

$$\log_2 \text{loss} = \frac{1}{N_f} \sum_{n_f=1}^{N_f} \log_2 \text{loss}_{n_f} = \frac{1}{N_f} \sum_{n_f=1}^{N_f} \sum_{k=1}^{K} -y_{n_f k} \log_2 \hat{p}_{n_f k} \tag{2}$$

The optimal $\log_2$ loss equals 0 and occurs if all observed categories $y_{n_f k}$ have been predicted correctly and with probability one. The worst case is ∞, which happens

if at least one category realizes although it was predicted with probability zero. To acknowledge sample variation in the test data, we calculate standard errors using the formula $\sigma(\log_2 \text{loss}) = \sqrt{\frac{1}{N_f(N_f-1)} \sum_{n_f=1}^{N_f} (\log_2 \text{loss}_{n_f} - \log_2 \text{loss})^2}$.

Scoring rules like the logarithmic loss have been developed in the context to evaluate an expert's probabilistic judgment. If his prediction is in line with the reality as observed later, the expert should be rewarded, otherwise he should get punished. A good scoring rule should ensure that the expert thinks carefully about his judgment and answers as best as he can. In our case, we regard the algorithms' predicted probabilities $\hat{p}$ as expert judgments. The scoring rule $\log \hat{p}_{n_f k}$ is, apart from linear transformations, the *unique* scoring rule that is proper, symmetric and impartial (O'Hagan and Forster 2004, 55ff.), three conditions that are desirable for our application. A scoring rule is called proper, if it encourages the expert to report his actual beliefs. It is symmetric if the rule induces no preference for a specific outcome category that might realize. It is impartial if calculations for the final reward are only based on the expert's prediction of the category that actually realizes and on no other categories. We use the linear transformation $-\log_2 \hat{p}_{n_f k} = -\log \hat{p}_{n_f k} / \log 2$ because in this way $-\log_2 \hat{p}_{n_f k}$ has an information-theoretic interpretation: it is the minimal number of bits that need to be transferred between two persons if both knew $\hat{p}_{n_f k}$ and one person wants to inform the other about a realized outcome y_{n_f} (Roulston and Smith 2002).

3 Part C: Algorithms

3.1 Algorithms Without Training Data

3.1.1 Coding Index (Exact Matching)

Provided that a coding index containing entries like "Media-Designer" $\rightarrow$ 23224 exists, its application is straightforward. If the textual input is exactly identical with some entry from a coding index, the corresponding code is assigned. Lyberg and Andersson (1983) provided an early account of experiences when statistical agencies were just starting to use this algorithm. Bekkerman and Gavish (2011) argued that the development of a coding index was a highly efficient way to classify millions of users at LinkedIn.

The coding index used throughout this paper is, in principle, the alphabetical dictionary that is part of the GCO. Yet, most entries in the alphabetical dictionary have a gender-neutral formatting (e.g., "Media-Designer/in"), unsuited for exact matching with verbal answers. We use a related dictionary, continuously updated by the Federal Employment Agency (2019) to include new jobs, because this file separates male and female job titles. To simplify exact matching further, we remove all text written in parentheses (e.g., "Receptionistin (Hotel)" $\rightarrow$ "Receptionistin"). The resulting coding index contains a male and a female entry for each title, totaling in more than 50,000 entries.

Despite this size, not every possible input text can be included in the coding index and exact matching will often fail.

3.1.2 Coding Index (Similarity Matching)

Even if the textual input and an entry from the coding index are not exactly identical, they may still be similar enough to be considered a match.

Some examples demonstrate how coding indexes and similarity matching have been used. Klingemann and Schönbach (1984) iterate through all words in the verbal answer. The first word being identical with an entry from the coding index is scored highest and used for automated coding. Ossiander and Milham (2006) preprocess the textual input (i.e., correct spelling errors, remove punctuation) before they search the coding index for similar entries. Their similarity score is the number of words that appear in both the textual input and in job titles from the coding index. The job title with the highest score is used for automated coding. Likewise, Russ et al. (2014) calculate a score for automated coding, but they divide the above score by the total number of different words from both texts (known as the Jaccard index), standardizing the score between zero and one. To be robust against misspellings, Damerau-Levenshtein distances between individual words are also taken into account. Yet another example is from Munz et al. (2016), who calculate the number of characters that would need to change before the textual input and the index entry are identical (generalized Levenshtein distance). As this is implemented in a computer-assisted coding application, coders see the most similar job titles to choose from.

Statistical agencies were early adopters of related approaches, which can be quite sophisticated. Algorithms developed at the US Census Bureau and at Statistics Canada

had routines to standardize the textual input (e.g., "Private Family Babysitter" $\rightarrow$ "PRIV FAMIL BABYSIT", an example taken from Appel and Hellerman (1983)) to eliminate and replace misspellings, abbreviations, trivial words, and other undesired characters, only keeping those terms that are useful for coding (Wenzowski 1988). Rather complex scoring algorithms were developed to weigh the relative importance of various terms against each other and to calculate an overall score for each category (Knaus 1987). Speizer and Buckley (1998) review this American line of research in more detail. Riviere (1997) collected descriptions about developments in a number of European statistical agencies, which used various similarity measures. These were often not based on splitting the text into words, but instead the text was split into short sequences of two or three characters.

One prominent and widely used computer program that is based on similarity calculations is the 'computer-assisted structural coding tool' (CASCOT). Index editors can fine-tune the tool to achieve better performance by downgrading less important words, defining equivalent word endings, defining rules to replace abbreviations and other terms, among many other available options. This makes the tool very flexible, although the authors write that fine-tuning is a "resource-demanding, time-consuming" task. The tool has been developed to be used with different languages for a variety of classifications (Elias et al. 2014).

For our evaluation, we load the GCO and the same coding index as before into CASCOT, but make no further adjustments. Results obtained this way are certainly suboptimal and one might try an infinite number of ways to improve them. We intentionally keep it simple to see how out-of-the-box solutions for occupation coding perform. CASCOT has a mode for automated coding, which outputs the highest similarity score along with the predicted category. To evaluate computer-assisted coding with CASCOT, we captured the data from the computer screen with optical character recognition. The analysis will require that we rank different textual inputs by their likelihood that one of the top five categories will be selected. CASCOT outputs for every suggested category a score between 1 and 100. Summing these scores as we do for other algorithms would be inadequate here, because a single suggested category with a score of 100 (aggregate score = $1 \cdot 100$ and very likely to be selected) would be lower-ranked than 5 suggested categories each having a score of 30 (aggregate score = $5 \cdot 30$, but unlikely to be selected). Instead of a sum, we use the mean among the top five categories.

3.2 Algorithms Based on Training Data

The algorithms above were designed to utilize coding indexes, imitating their usage by professional coders. If no such coding index exists, it can be created from previously coded answers, either by careful manual inspection of the raw material or automatically by using algorithms that have been developed for such a purpose (O'Reagon 1972, Thompson et al. 2014). However, a coding index is just one way to encode what has been learned from previously coded answers. If the index went through an editorial process during development and, for example, spelling errors have been corrected, this makes its style different from natural-occurring verbal answers, complicating the use of automated matching routines. As an alternative, we next look at algorithms that were

designed to learn prediction rules directly from previously coded answers, avoiding the need to construct a coding index.

Some basic preprocessing of text is needed for all algorithms that follow in this section. The idea is to reduce the dimensionality of text and to bring it in a numeric format, making it more suitable for analysis. Common steps include lower-casing all letters, removing punctuation marks, or substituting special characters (e.g., '€' $\rightarrow$ 'euro'). Stemming (e.g., 'designer' $\rightarrow$ 'design') and the removal of stop words (e.g., 'and', 'the') are also often used to make similar answers identical, reducing the dimensionality, at the risk that decisive information for coding is lost. A document-term matrix consists of one row per document and one column per word. It is created from the standardized texts by counting how often each word/term appears in each document. This disregards the word order. As a resort, one may add additional features to the document-term matrix, for example counting the frequency that all two-word sequences appear in each document. Additional information may be included in the predictor matrix as well, such as the number of words per document or other variables from a survey. The exact choices how predictor matrices are calculated vary widely as different authors try what works best for them, depending on the data and the algorithms they wish to use.

3.2.1 Memory-based Reasoning

Creecy et al. (1992) proposed a k-nearest neighbor algorithm, which aims to code answers in the same way as similar answers were coded previously. Their document-term matrix does not only include columns for every word in the training data, but also for all two-word co-occurances in the training data. Two metrics to calculate similarity between different answers were proposed. Both metrics take into account that words like "the" provide little insight about an appropriate occupational category, whereas words like "weaver" are more meaningful and should have higher weights. To predict a category, one searches the training data for k nearest neighbors having highest similarity. Among these, the similarity scores are added per category and the category with the highest summed score is selected. If one wishes to use this algorithm to separate easy-to-code from hard-to-code answers, the authors recommended calculating confidence scores and defining thresholds category-wise. However, since Creecy et al. (1992) determined thresholds using the same test data that was used again to evaluate the performance, they risked that their performance metrics are biased upwards. We do not implement this problematic procedure in our software, but related approaches still might help to improve the performance at low production rates.

This algorithm contrasts with previous research at the US Census Bureau, which was based on coding indices and similarity matching. Creecy et al. (1992) argued that their new algorithm outperformed the previous system by wide margins. Moreover, they developed the new system in just "four person-months while the [previous] expert system required 192 person-months." With impressive results like this, confirmed by Gillman and Appel (1994) in a comparison with other supervised learning algorithms, it is worth including the memory-based reasoning algorithm in our comparison.

3.2.2 Adapted Nearest Neighbor

Gweon et al. (2017) proposed an *adapted nearest neighbor* algorithm for occupation coding. To classify a new response x, one searches in the training data for all responses v that have largest similarity. The cosine similarity

$$s(x, v) = \frac{\sum x_j v_j}{\sqrt{\sum x_j^2}\sqrt{\sum v_j^2}}$$

between vectors from the document-term matrix is used (the index j runs across columns/words in the document-term matrix). It ranges between 0 if both answers have no words in common and 1 if both texts (i.e., rows from the document-term matrix) are identical. Let $K(x)$ be the number of responses from the training data that are most similar with the new response x and let $\hat{p}_{nn}(l|x)$ be the relative frequency of category l among those most similar responses. The category which appears most often ($\arg\max_l \hat{p}_{nn}(l|x)$) is predicted along with a score to express the certainty of correctness. This score is calculated as a product of the relative frequency, the similarity, and another multiplier,

$$\gamma(l|x) = \hat{p}_{nn}(l|x)s(x)\frac{K(x)}{K(x) + 0.1}$$

The motivation is that relative frequencies alone are poorly suited as an expression of certainty. Therefore $s(x)$ shrinks the relative frequency towards 0 depending on how similar the most similar training cases are. Likewise, $\frac{K(x)}{K(x)+0.1}$ shrinks the relative frequencies towards zero if the prediction is based on very few (e.g., $K(x) = 1$) training observations.

Gweon et al. (2017) developed the adapted nearest neighbor algorithm specifically for occupation coding and compared it with a number of other algorithms. It outperformed support vector machines with linear kernel, a duplicate algorithm, a hybrid combination of support vector machines with the duplicate algorithm, and other combinations of algorithms that exploit the hierarchical structure of occupational classifications. This superior performance leads us to include the adapted Nearest Neighbor algorithm in our comparison.

3.2.3 Penalized Multinomial Logistic Regression

We now turn to algorithms that are popular not only for occupation coding but for a wide range of supervised learning applications. The first approach is *multinomial logistic regression with elastic net regularization* as implemented in the R-package `glmnet` (Friedman et al. 2010, Hastie et al. 2015). Let $Y = (0, ..., 0, 1, 0, ..., 0)$ be a coordinate vector of length K, with a 1 at the lth position indicating that the lth category has been selected. Given a p-dimensional feature vector x, the probability for category l, $l = 1, ..., K$, is modeled as $\mathbb{P}(Y_l = 1|x) = \frac{\exp f_l^{\text{reg}}(x)}{\sum_{k=1}^K \exp f_k^{\text{reg}}(x)}$ with $f_l^{\text{reg}}(x) = \beta_{0l} + x^T\beta_l$. The parameter matrix $\beta \in \mathbb{R}^{K\times(p+1)}$ contains several million parameters in our application. To deal with the high dimensionality we employ elastic net regularization, $P_\alpha(\beta_l) = \sum_{j=1}^p (1-\alpha)\frac{1}{2}\beta_{jl}^2 + \alpha|\beta_{jl}|$. Since α is typically chosen in the model tuning

phase, the elastic net is more flexible than ridge regularization ($\alpha = 0$) and lasso regularization ($\alpha = 1$). Parameters are estimated using a pathwise coordinate descent algorithm that maximizes the regularized multinomial log-likelihood

$$\max_{\beta} \frac{1}{N} \sum_{i=1}^{N} (\sum_{l=1}^{K} y_{il} \cdot f_l^{\text{reg}}(x_i) - \log(\sum_{l=1}^{K} \exp f_l^{\text{reg}}(x_i))) - \lambda \sum_{l=1}^{K} P_\alpha(\beta_l)$$

In practice, `glmnet` does not support very rare categories. We were forced to remove between 61 and 178 observations, depending on the training data used, whose categories appear less than two times in the training data. Rare categories will never be predicted. We set their predicted probabilities to zero, which is needed for some of our subsequent evaluations.

Various authors have used logistic regression (Measure 2014), support vector machines with linear kernels (Takahashi et al. 2005, 2014, Westermark et al. 2015, Gweon et al. 2017), or maximum entropy classifiers (Jung et al. 2008, Russ et al. 2016) for automated occupation coding. These algorithms share the common feature that they search the function space of linear functions (e.g., $f_l^{\text{reg}}(x)$) to maximize some objective. Theoretical results summarized by Hastie et al. (2009) prove a high similarity between regularized logistic regression and support vector machines with linear kernels. This makes us believe that the different algorithms will achieve similar performance, confirmed by an empirical result from Measure (2014). We therefore include only one of these algorithms in our comparison. We prefer regularized multinomial logistic regression because, unlike support vector machines, it returns predictions on a probability scale and it avoids estimating a large number of binary classifiers.

3.2.4 Tree Boosting

Gradient tree boosting (Friedman 2001) is another well-known supervised learning technique. We use the speed-optimized implementation from the R-package `XGBoost`, which enhances Friedman's proposal with some additional tweaks (Chen and Guestrin 2016). Like in logistic regression, the goal is to obtain functions $f_l(x)$ that minimize the negative multinomial log-likelihood; only the regularization term is different. However, logistic regression is rather restrictive and inflexible as it forces the functions $f_l^{\text{reg}}(x) = \beta_{0l} + x^T \beta_l$ to be linear in their parameters. In contrast, gradient boosting can learn more flexible functions $f_l^{\text{boost}}(x) = \sum_{t=1}^{T} f_l^{(t)}(x)$. Thus, one learns an ensemble of base learners $f_l^{(t)}$ and the flexibility of the algorithm is determined by the class of functions that $f_l^{(t)}$ comes from. As is common, we choose decision trees as base learners. Decision trees learn step functions in a high-dimensional input space and allow for high-order interactions. The base learners are trained iteratively in T rounds, with each iteration focusing on examples that the previous iterations got wrong. The higher flexibility comes at a cost. There exist a dozen tuning parameters in `XGBoost`, some of them relating to the overall gradient boosting algorithm, others relating to the algorithm that grows the individual trees, making a careful tuning phase mandatory. This can be time-consuming. Developers should have a sufficient number of CPUs in their computing environment and a tuning strategy to achieve good-enough results within a limited number of iterations.

We include `XGBoost` in our comparison because the algorithm has an impressive track-record in winning machine learning competitions (Chen and Guestrin 2016), showing that it is state-of-the-art when excellent predictions are desired. Moreover, tree-based approaches appear very promising for occupation coding, because trees are very fast in detecting relevant features and interactions and because split point selection is obvious for binary features (i.e., the (non-)occurrence of words in our situation). Tree boosting could potentially be used to build a prediction function that resembles a coding index, because the trained model, a sum of trees, can be aggregated to form a single decision tree, which in turn can be transformed into a list containing thousands of rules (as in a coding index). Unlike regression, `XGBoost` outputs positive predicted probabilities for all categories, even for those categories not observed in the training data.

Two algorithms in our comparison rely on tree boosting (`XGBoost`). While the first one uses survey data only, the second algorithm pools survey data and job titles from the coding index and uses both in the training phase. To obtain good results with the larger training data, some tuning parameters are changed as well (see on-line appendix), but otherwise both algorithms identical.

3.2.5 Other Algorithms

We believe the algorithms mentioned above are most promising for occupation coding. There exist, of course, additional studies that have explored other directions. For example, Ikudo et al. (2018) trained a random forest and Nahoomi (2018) tried whether convolutional neural networks could outperform classic approaches, albeit both studies do not use survey data. Besides, the large number of categories in occupational classifications led Javed et al. (2015), Gweon et al. (2017), and Nahoomi (2018) to consider hierarchical approaches, which exploit that classifications aggregate the bottom-level categories into a smaller number of medium- or top-level categories. The empirical evidence from this literature gives no reason to expect large gains in performance by using any of these algorithms. Therefore, they are not part of our comparison.

Several authors have combined various algorithms in a number of ways. Jung et al. (2008) and Westermark et al. (2015) followed a sequential approach, in which matching with a coding index is tried first and only if this is unsuccessful a support vector machine (or a maximum entropy model) is used for prediction. Likewise, Takahashi et al. (2005, 2014) used exact matching first (in fact, their matching process is more complicated due to Japanese grammar), but its agreement with human coders was unsatisfactory. Thus, the resulting category is not coded directly, but it is added as a feature to a document-term matrix and used in a support vector machine. This strategy relates to 'stacking'-based ensemble classifiers, implemented for occupation coding by Russ et al. (2016) and Schierholz et al. (2018). The idea of 'stacking' is to predict categories using various classification algorithms and to use the predictions as features in a final adjudication algorithm. The verbal answers enter the adjudication algorithm not directly, but only through the predictions from the other algorithms. Thompson et al. (2014) described another related algorithm. They automatically built a coding index from the data, a first-level classifier, but its predictions were adjudicated with a logistic regression model that takes several other variables into account. Taken together, sev-

eral independent research projects have combined different algorithms in a number of ways but there is no consensus about best practices. This paper focuses on isolated algorithms, which can be used as a stand-alone module or combined with one another to form an integrated system.

4 Part D: Model Tuning

The results from any algorithm depend on the choice of tuning parameters. This appendix demonstrates the performance of different parameter configurations, explores the importance of the parameters. For each algorithm, we need to select a final parameter configuration that we use to report the results in the main text. For this purpose, a grid search was performed (except for algorithm 6 and 7) and the best configuration among the ones tested was chosen. The goal was to find for each algorithm a parameter configuration that optimizes performance. The final parameter configurations after tuning as used in the main paper are shown in Table A1.

The document-term matrix should be tailored towards the statistical learning algorithm. For example, treating stop words like any other word in the adapted nearest neighbor algorithm (algorithm 4) does not bode well, but if an algorithm does variable selection (like algorithms 5-7), including a few more terms in the document-term matrix should not matter much and might even help if some stop words are in fact predictive of any category. To give another example, adding extra columns to the document-term matrix that indicate co-occurring words is an essential part of algorithm 3 and we wanted to follow its original description, but it can be computationally demanding with other algorithms.

We did not expect that the type of preprocessing (simple or extended) would matter much and this was confirmed in our test with algorithm 4.

In general, we will see in this appendix that the performances of different parameter configurations are often very similar to another. This suggests that our chosen parameters are close to optimal and a more profound grid search over additional values will not improve the predictions in a practically meaningful way.

4.1 Penalized Multinomial Logistic Regression

The following parameters are varied:

- whether to exclude stop words or not (while creating the document-term matrix);
- whether to use stemming or not (while creating the document-term matrix);
- whether to add an additional column to the predictor matrix counting the number of words in a document;
- choice of the regularization parameter λ (see below);
- choice of the elastic-net regularization parameter α on a grid $(0, 0.05, 0.2)$ (see below).

The default in the `glmnet`-package is to estimate a sequence of models for 100 values of λ using a highly efficient cyclical coordinate descent algorithm. The largest value of λ is chosen such that all coefficients in the linear term are shrunk to zero. The smallest value of λ is 0.0001 times the largest value. A regular grid (equidistant on a logarithmic scale of λ) is spanned in between and we report predictions for values of λ that are at the 50-th, 70-th, 80-th, and 100-th position in this grid.

Table A1: Model Tuning Results: Final Choice of Parameter Values†

	Text Processing While Preparing the Document-Term Matrix				
	Preprocessing	Stop Words	Stemming	Word Count	Word co-occurrences
3. Mem-based reasoning	Extended	Excluded*	Yes*	No	Yes
4. Adapt. nearest neighbor	Simple*	Excluded*	Yes*	No	No
5. Multinomial regression	Extended	Not excluded*	Yes*	No*	No
6. Tree boosting (XGBoost)	Extended	Not excluded	Yes	Yes	No
7. Tree boosting (\w cod index)	Extended	Not excluded	Yes	Yes	No

	Method-specific parameters
3. Mem-based reasoning	metric = ERROR*, $k = 7$*
4. Adapt. nearest neighbor	$mult = 0.1$*
5. Multinomial regression	$\alpha = 0.05$*, λ**
6. Tree boosting (XGBoost)	max_rounds = 40, $\eta = 0.5$*, max_delta_step = 1*, max_depth = 20*, $\gamma = 0.6$*, $\lambda = 1e^{-4}$*, $\alpha = 0$*, min_child_weight = 0*, subsample = 0.75*, colsample_by_tree = 1*, colsample_by_level = 1*
7. Tree boosting (\w Cod Index)	max_rounds = 40, $\eta = 0.75$*, max_delta_step = 1*, max_depth = 40*, γ**, $\lambda = 1e^{-4}$*, $\alpha = 0$*, min_child_weight = 0*, subsample = 0.75*, colsample_by_tree = 1*, colsample_by_level = 1*

†A cell is marked with a single star (*), if the value results from model tuning. A cell is marked with double stars (**), if the value results from model tuning but depends on the respective training data set. If no stars are shown, the value was selected without testing alternatives.

Simple preprocessing refers to lowercasing all letters and removing punctuation marks. *Extended preprocessing* does the same and, in addition, substitutes German umlauts and other non-standard characters, removes superfluous empty space characters, removes the text 'dipl.' and 'diplom', and substitutes the abbreviation 'ing.' with 'ingenieur' (engineer).

German *stop words* are taken from the R-package `tm` (Feinerer and Hornik 2018). For *stemming* we use the Porter stemmer from package `SnowballC` in German language (Bouchet-Valat 2018).

An additional column counting the number of words is added to the document term matrix if *Word Count* = *Yes*. Further columns are added if *Word co-occurrences* = *Yes*, one column for any two words that co-occur in an answer.

In our experience with occupational data, the algorithm in `glmnet` often does not converge during a limited number of iterations if values of α are substantially larger than zero. To counter this problem, we increase the maximum number of iterations `maxit` to 10^6 (the default is 10^5) and reduce the convergence threshold `thresh` to 10^{-4} (the default is 10^{-7}). If the threshold is still not reached within the maximum number of iterations, we report the performance for the smallest value of λ that is available. Default values are used for all other parameters in the `glmnet`-package. The low threshold can potentially have a negative impact on model performance. To rule out this possibility, the $\texttt{thresh} = 10^{-4}$ is only used in this appendix to select the best-performing parameter configuration. The threshold 10^{-7} was used to estimate all models in the main paper.

What are good tuning parameters to use? Table A2 and Table A3 show the predictive performance for various combinations of α and λ in data set D (small) and data set C (larger). Both tables indicate that the choice $\alpha = 0.05$ is close to optimal with respect to the criteria agreement rate, logloss and sharpness. In Tables A4 and A5 we show more results for various parameter configurations from both data sets when α is fixed at 0.05. We observe that the agreement rate is almost constant, sharpness improves as λ decreases, and logloss is optimal at mean values of λ. No meaningful differences are found whether or not stop words and counting of words are used, a result that also holds for other choices of α (not shown). The performance appears to improve marginally if the German Porter stemmer is used.

Based on these results, almost all models in the main paper are estimated with $\alpha = 0.05$, $maxit = 10^6$, $thresh = 10^{-7}$, with stemming, without excluding stop words and without counting words. We look at λ-values at the 50-th, 70-th, 80-th, and 100-th position in the provided grid and find in all data sets that the λ-value at the 70-th position is close to optimal, minimizing logloss. An exception is the model that we trained on the complete data set (A,B,C,D). Due to the large numbers of outcome categories and predictor variables, computational limitations forced us to build a grid of 60 values of λ and the algorithm stopped with an error (convergence not reached) at the 26-th position. The 25-th λ-value is thus reported, although smaller values of λ, if calculable, would certainly improve the results.

The role of λ is better understood in the context of a reliability diagram as shown in Figure A1. If λ is too large, categories are predicted for many observations with a probability that is too low. Thus, the predicted categories will realize more often than expected (underfitting with $\lambda = 0.02499$). Alternatively, if λ is too small, for many observations a single category is predicted with overly high probability (overfitting with $\lambda = 0.00024$). Overfitting still improves sharpness, which would be optimal if all probabilities are either exactly one or zero. However, one would like that forecasted probabilities reflect the observed relative frequencies in the test data. This requires a value for λ that is neither too large nor too small.

Table A2: Performance measures of penalized multinomial logistic regression in data set D without stop words, without stemming and without counting words. Selected combinations of α and λ are shown.†

α	λ	$agreem.rate$	$logloss$	$sharpness$
0.00	1.7540	0.04 ± 0.01	7.12 ± 0.06	7.65 ± 0.00
0.00	0.2729	0.43 ± 0.02	5.17 ± 0.09	7.43 ± 0.02
0.00	0.1076	0.56 ± 0.02	4.07 ± 0.11	6.52 ± 0.04
0.00	0.0167	0.59 ± 0.02	3.30 ± 0.13	4.14 ± 0.08
0.05	0.0351	0.58 ± 0.02	3.55 ± 0.12	5.27 ± 0.07
0.05	0.0055	0.59 ± 0.02	3.26 ± 0.13	3.37 ± 0.09
0.05	0.0022	0.59 ± 0.02	3.29 ± 0.14	2.84 ± 0.09
0.05	0.0003	0.59 ± 0.02	3.54 ± 0.16	2.19 ± 0.09
0.20††	0.0244	0.03 ± 0.00	147.91 ± 0.99	0.00 ± 0.00

†Data set D was split into 6575 training observations and 1064 test observations. Logloss = ∞ in the complete test set since the realized category of some cases was predicted with probability 0. Excluding those cases, we report logloss on a subset of 991 test observations.
††With $\alpha = 0.20$ results for $\lambda < 0.0244$ are not available since the algorithm did not converge after 10^6 iterations.

Table A3: Performance measures of penalized multinomial logistic regression in data set C without stop words, without stemming and with counting words. Selected combinations of α and λ are shown.†

α	λ	$agreem.rate$	$logloss$	$sharpness$
0.00	1.2495	0.11 ± 0.01	7.86 ± 0.07	8.45 ± 0.00
0.00	0.1944	0.51 ± 0.02	5.61 ± 0.10	8.05 ± 0.03
0.00	0.0767	0.58 ± 0.02	4.41 ± 0.11	6.76 ± 0.05
0.00	0.0119	0.60 ± 0.02	3.57 ± 0.13	3.95 ± 0.07
0.05	0.0250	0.59 ± 0.02	3.95 ± 0.12	5.53 ± 0.07
0.05	0.0039	0.59 ± 0.02	3.50 ± 0.14	3.23 ± 0.08
0.05	0.0015	0.59 ± 0.02	3.56 ± 0.15	2.61 ± 0.07
0.05	0.0002	0.59 ± 0.02	3.92 ± 0.17	1.92 ± 0.07
0.20††	0.0132	0.58 ± 0.02	4.03 ± 0.13	5.47 ± 0.07

†Data set C was split into 47,930 training observations and 1064 test observations. Logloss = ∞ in the complete test set since the realized category of some cases was predicted with probability 0. Excluding those cases, we report logloss on a subset of 1061 test observations.
††With $\alpha = 0.20$ results for $\lambda < 0.0132$ are not available since the algorithm did not converge after 10^6 iterations.

Table A4: Performance measures of penalized multinomial logistic regression in data set D with $\alpha = 0.05$. Selected λ and all combinations of stop words, stemming and word counting are shown.†

λ	*stem*	*stop*	*count*	*agreem.rate*	*logloss*	*sharpness*
With threshold $= 10^{-4}$*:*						
0.0055	NO	YES	YES	0.59 ± 0.02	3.26 ± 0.13	3.37 ± 0.09
0.0055	NO	NO	YES	0.59 ± 0.02	3.26 ± 0.13	3.38 ± 0.09
0.0055	NO	YES	NO	0.59 ± 0.02	3.26 ± 0.13	3.37 ± 0.09
0.0055	NO	NO	NO	0.59 ± 0.02	3.26 ± 0.13	3.37 ± 0.09
0.0055	YES	YES	YES	0.60 ± 0.02	3.15 ± 0.13	3.23 ± 0.09
0.0055	YES	NO	YES	0.60 ± 0.01	3.15 ± 0.13	3.23 ± 0.09
0.0055	YES	YES	NO	0.60 ± 0.01	3.15 ± 0.13	3.22 ± 0.09
0.0055	YES	NO	NO	0.60 ± 0.01	3.15 ± 0.13	3.23 ± 0.09
0.0022	NO	YES	YES	0.59 ± 0.02	3.30 ± 0.14	2.84 ± 0.09
0.0022	NO	NO	YES	0.59 ± 0.02	3.29 ± 0.14	2.85 ± 0.09
0.0022	NO	YES	NO	0.59 ± 0.02	3.30 ± 0.14	2.83 ± 0.09
0.0022	NO	NO	NO	0.59 ± 0.02	3.29 ± 0.14	2.84 ± 0.09
0.0022	YES	YES	YES	0.61 ± 0.01	3.21 ± 0.14	2.66 ± 0.09
0.0022	YES	NO	YES	0.60 ± 0.01	3.19 ± 0.14	2.67 ± 0.09
0.0022	YES	YES	NO	0.61 ± 0.01	3.21 ± 0.14	2.66 ± 0.09
0.0022	YES	NO	NO	0.61 ± 0.01	3.20 ± 0.14	2.67 ± 0.09
0.0003	NO	YES	YES	0.59 ± 0.02	3.55 ± 0.16	2.19 ± 0.09
0.0003	NO	NO	YES	0.59 ± 0.02	3.54 ± 0.16	2.20 ± 0.09
0.0003	NO	YES	NO	0.59 ± 0.02	3.55 ± 0.16	2.18 ± 0.09
0.0003	NO	NO	NO	0.59 ± 0.02	3.54 ± 0.16	2.19 ± 0.09
0.0003	YES	YES	YES	0.60 ± 0.02	3.51 ± 0.16	1.98 ± 0.08
0.0003	YES	NO	YES	0.60 ± 0.02	3.48 ± 0.16	2.00 ± 0.09
0.0003	YES	YES	NO	0.60 ± 0.02	3.51 ± 0.16	1.97 ± 0.08
0.0003	YES	NO	NO	0.60 ± 0.02	3.48 ± 0.16	1.99 ± 0.08
With threshold $= 10^{-7}$ *(Improvements expected):*						
0.0055	YES	NO	NO	0.60 ± 0.01	3.15 ± 0.13	3.19 ± 0.09
0.0022	YES	NO	NO	0.61 ± 0.01	3.21 ± 0.14	2.64 ± 0.09
0.0003	YES	NO	NO	0.60 ± 0.02	3.50 ± 0.16	1.99 ± 0.09

†Data set D was split into 6575 training observations and 1064 test observations. Logloss = ∞ in the complete test set since the realized category of some cases was predicted with probability 0. Excluding those cases, we report logloss on a subset of 991 test observations.

Table A5: Performance measures of penalized multinomial logistic regression in data set C with $\alpha = 0.05$. Selected λ and all combinations of stop words, stemming and word counting are shown.†

λ	*stem*	*stop*	*count*	*agreem.rate*	*logloss*	*sharpness*
0.0250	NO	YES	YES	0.59 ± 0.02	3.95 ± 0.12	5.53 ± 0.07
0.0250	NO	NO	YES	0.59 ± 0.02	3.95 ± 0.12	5.53 ± 0.07
0.0250	NO	YES	NO	0.59 ± 0.02	3.95 ± 0.12	5.53 ± 0.07
0.0250	NO	NO	NO	0.59 ± 0.02	3.95 ± 0.12	5.53 ± 0.07
0.0250	YES	YES	YES	0.58 ± 0.02	3.96 ± 0.12	5.58 ± 0.07
0.0250	YES	NO	YES	0.58 ± 0.02	3.96 ± 0.12	5.57 ± 0.07
0.0250	YES	YES	NO	0.58 ± 0.02	3.96 ± 0.12	5.58 ± 0.07
0.0250	YES	NO	NO	0.58 ± 0.02	3.96 ± 0.12	5.57 ± 0.07
0.0039	NO	YES	YES	0.59 ± 0.02	3.50 ± 0.14	3.24 ± 0.08
0.0039	NO	NO	YES	0.59 ± 0.02	3.50 ± 0.14	3.23 ± 0.08
0.0039	NO	YES	NO	0.59 ± 0.02	3.50 ± 0.14	3.24 ± 0.08
0.0039	NO	NO	NO	0.59 ± 0.02	3.50 ± 0.14	3.23 ± 0.08
0.0039	YES	YES	YES	0.59 ± 0.02	3.45 ± 0.13	3.27 ± 0.08
0.0039	YES	NO	YES	0.59 ± 0.02	3.45 ± 0.13	3.27 ± 0.08
0.0039	YES	YES	NO	0.59 ± 0.02	3.45 ± 0.13	3.27 ± 0.08
0.0039	YES	NO	NO	0.59 ± 0.02	3.45 ± 0.13	3.27 ± 0.08
0.0015	NO	YES	YES	0.59 ± 0.02	3.56 ± 0.15	2.61 ± 0.07
0.0015	NO	NO	YES	0.59 ± 0.02	3.56 ± 0.15	2.61 ± 0.07
0.0015	NO	YES	NO	0.59 ± 0.02	3.56 ± 0.15	2.61 ± 0.07
0.0015	NO	NO	NO	0.59 ± 0.02	3.56 ± 0.15	2.61 ± 0.07
0.0015	YES	YES	YES	0.59 ± 0.02	3.49 ± 0.14	2.63 ± 0.07
0.0015	YES	NO	YES	0.59 ± 0.02	3.49 ± 0.14	2.63 ± 0.07
0.0015	YES	YES	NO	0.59 ± 0.02	3.49 ± 0.14	2.63 ± 0.07
0.0015	YES	NO	NO	0.59 ± 0.02	3.49 ± 0.14	2.63 ± 0.07
0.0002	NO	YES	YES	0.59 ± 0.02	3.92 ± 0.17	1.93 ± 0.07
0.0002	NO	NO	YES	0.59 ± 0.02	3.92 ± 0.17	1.92 ± 0.07
0.0002	NO	YES	NO	0.59 ± 0.02	3.93 ± 0.17	1.93 ± 0.07
0.0002	NO	NO	NO	0.59 ± 0.02	3.93 ± 0.17	1.92 ± 0.07
0.0002	YES	YES	YES	0.59 ± 0.02	3.83 ± 0.17	1.93 ± 0.07
0.0002	YES	NO	YES	0.59 ± 0.02	3.84 ± 0.17	1.92 ± 0.07
0.0002	YES	YES	NO	0.59 ± 0.02	3.84 ± 0.17	1.93 ± 0.07
0.0002	YES	NO	NO	0.59 ± 0.02	3.84 ± 0.17	1.92 ± 0.07

†Data set C was split into 47,930 training observations and 1064 test observations. Logloss = ∞ in the complete test set since the realized category of some cases was predicted with probability 0. Excluding those cases, we report logloss on a subset of 1061 test observations.

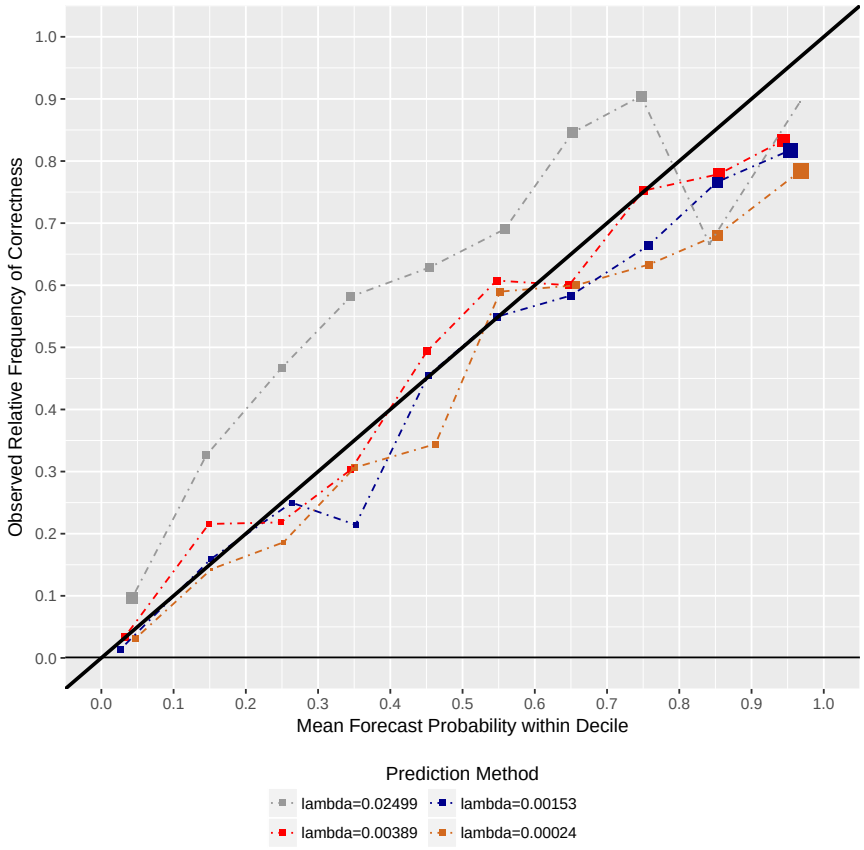

Figure A1: Reliability diagram of most probable category (k=1); Ideal probabilistic predictions should match the observed relative frequencies, the diagonal; The horizontal line at $\frac{1}{1291}$ represents random guessing among all 1291 categories; Point size is proportional to the number of observations within each bin. Data set C was split into 47,930 training observations and 1064 test observations and a regularized logistic regression with parameters $\alpha = 0.05$, $maxit = 10^6$, $thresh = 10^{-7}$, with stemming, without excluding stop words and without counting words was estimated.

4.2 Adapted Nearest Neighbor (Gweon et al. 2017)

The following parameters are varied:

- whether to exclude stop words or not (while creating the document-term matrix);
- whether to use stemming or not (while creating the document-term matrix);
- whether to do extended preprocessing (YES) or just removal of punctuation marks (NO);
- choice of multiplier $mult$.

We can judge the performance of different parameter configurations with respect to the agreement rate or with respect to N, the number of observations in the test data that had the true category predicted with non-zero relevance score. The logloss itself is less suited for an overall comparison because it is evaluated on subsets of test data that have different size N. Also, the logloss is derived from probabilistic theory, but it is doubtful if a probabilistic framework is appropriate to judge the output from this algorithm.

The different parameter configurations are evaluated on data set D (Table A6) and data set C (Table A7). The highest agreement rates and the largest N are achieved if stop words are removed. Wether stemming or the type of preprocessing have a positive impact is more speculative and additional analyses would be needed. The exact choice of the multiplier $mult$ should not be judged based on this result since it was developed for a different purpose. We still see that it has only a small effect on the evaluation metrics provided here.

This analysis confirms that Gweon et al. (2017) made reasonable choices. We follow them and exclude stop words, use stemming, remove punctuation marks, and set the multiplier to 0.1 for all analyses in the main paper.

Table A6: Performance measures of adapted nearest neighbor in data set D under different configurations.†

stop	*stem*	*prep*	*mult*	*agreem.rate*	*logloss* (N)	*sharpness*
NO	NO	NO	0.05	0.50 ± 0.02	2.26 ± 0.14 (722)	1.67 ± 0.10
NO	NO	NO	0.10	0.50 ± 0.02	2.28 ± 0.14 (722)	1.68 ± 0.10
NO	NO	NO	0.20	0.50 ± 0.02	2.33 ± 0.14 (722)	1.70 ± 0.10
NO	NO	YES	0.05	0.51 ± 0.02	2.09 ± 0.14 (718)	1.55 ± 0.09
NO	NO	YES	0.10	0.51 ± 0.02	2.12 ± 0.14 (718)	1.56 ± 0.09
NO	NO	YES	0.20	0.51 ± 0.02	2.17 ± 0.14 (718)	1.58 ± 0.09
NO	YES	NO	0.05	0.51 ± 0.02	2.15 ± 0.14 (722)	1.60 ± 0.09
NO	YES	NO	0.10	0.51 ± 0.02	2.18 ± 0.14 (722)	1.62 ± 0.09
NO	YES	NO	0.20	0.51 ± 0.02	2.22 ± 0.14 (722)	1.64 ± 0.09
NO	YES	YES	0.05	0.52 ± 0.02	2.01 ± 0.13 (719)	1.50 ± 0.09
NO	YES	YES	0.10	0.52 ± 0.02	2.03 ± 0.13 (719)	1.51 ± 0.09
NO	YES	YES	0.20	0.52 ± 0.02	2.08 ± 0.13 (719)	1.53 ± 0.09
YES	NO	NO	0.05	0.56 ± 0.02	2.26 ± 0.14 (783)	1.74 ± 0.10
YES	NO	NO	0.10	0.56 ± 0.02	2.28 ± 0.14 (783)	1.75 ± 0.10
YES	NO	NO	0.20	0.56 ± 0.02	2.32 ± 0.14 (783)	1.77 ± 0.10
YES	NO	YES	0.05	0.56 ± 0.02	2.08 ± 0.13 (768)	1.59 ± 0.10
YES	NO	YES	0.10	0.56 ± 0.02	2.10 ± 0.13 (768)	1.60 ± 0.10
YES	NO	YES	0.20	0.56 ± 0.02	2.15 ± 0.13 (768)	1.62 ± 0.10
YES	YES	NO	0.05	0.58 ± 0.02	2.09 ± 0.13 (784)	1.63 ± 0.10
YES	YES	NO	0.10	0.58 ± 0.02	2.12 ± 0.13 (784)	1.64 ± 0.10
YES	YES	NO	0.20	0.58 ± 0.02	2.16 ± 0.13 (784)	1.66 ± 0.10
YES	YES	YES	0.05	0.58 ± 0.02	1.95 ± 0.13 (772)	1.51 ± 0.09
YES	YES	YES	0.10	0.58 ± 0.02	1.98 ± 0.13 (772)	1.52 ± 0.09
YES	YES	YES	0.20	0.58 ± 0.02	2.02 ± 0.13 (772)	1.54 ± 0.09

†Data set D was split into 6575 training observations and 1064 test observations. Logloss = ∞ in the complete test set since the realized category of some cases was predicted with probability 0. Excluding those cases, we report logloss on different, non-comparable subsets of size N.

Table A7: Performance measures of adapted nearest neighbor in data set C under different configurations.†

stop	*stem*	*prep*	*mult*	*agreem.rate*	*logloss* (N)	*sharpness*
NO	NO	NO	0.05	0.55 ± 0.02	1.98 ± 0.11 (820)	1.70 ± 0.08
NO	NO	NO	0.10	0.55 ± 0.02	1.99 ± 0.11 (820)	1.71 ± 0.08
NO	NO	NO	0.20	0.55 ± 0.02	2.01 ± 0.11 (820)	1.72 ± 0.08
NO	NO	YES	0.05	0.56 ± 0.02	1.78 ± 0.10 (818)	1.53 ± 0.07
NO	NO	YES	0.10	0.56 ± 0.02	1.79 ± 0.10 (818)	1.53 ± 0.07
NO	NO	YES	0.20	0.56 ± 0.02	1.81 ± 0.10 (818)	1.55 ± 0.07
NO	YES	NO	0.05	0.55 ± 0.02	1.92 ± 0.11 (821)	1.66 ± 0.08
NO	YES	NO	0.10	0.55 ± 0.02	1.93 ± 0.11 (821)	1.67 ± 0.08
NO	YES	NO	0.20	0.55 ± 0.02	1.95 ± 0.11 (821)	1.69 ± 0.08
NO	YES	YES	0.05	0.56 ± 0.02	1.75 ± 0.10 (819)	1.52 ± 0.07
NO	YES	YES	0.10	0.56 ± 0.02	1.76 ± 0.10 (819)	1.53 ± 0.07
NO	YES	YES	0.20	0.56 ± 0.02	1.78 ± 0.10 (819)	1.54 ± 0.07
YES	NO	NO	0.05	0.56 ± 0.02	1.99 ± 0.11 (837)	1.74 ± 0.08
YES	NO	NO	0.10	0.56 ± 0.02	2.00 ± 0.11 (837)	1.75 ± 0.08
YES	NO	NO	0.20	0.56 ± 0.02	2.02 ± 0.11 (837)	1.77 ± 0.08
YES	NO	YES	0.05	0.58 ± 0.02	1.77 ± 0.10 (832)	1.56 ± 0.07
YES	NO	YES	0.10	0.58 ± 0.02	1.78 ± 0.10 (832)	1.56 ± 0.07
YES	NO	YES	0.20	0.58 ± 0.02	1.80 ± 0.10 (832)	1.58 ± 0.07
YES	YES	NO	0.05	0.57 ± 0.02	1.92 ± 0.11 (839)	1.69 ± 0.08
YES	YES	NO	0.10	0.57 ± 0.02	1.93 ± 0.11 (839)	1.70 ± 0.08
YES	YES	NO	0.20	0.57 ± 0.02	1.95 ± 0.11 (839)	1.72 ± 0.08
YES	YES	YES	0.05	0.58 ± 0.02	1.74 ± 0.10 (835)	1.54 ± 0.07
YES	YES	YES	0.10	0.58 ± 0.02	1.75 ± 0.10 (835)	1.55 ± 0.07
YES	YES	YES	0.20	0.58 ± 0.02	1.77 ± 0.10 (835)	1.56 ± 0.07

†Data set C was split into 47,930 training observations and 1064 test observations. Logloss = ∞ in the complete test set since the realized category of some cases was predicted with probability 0. Excluding those cases, we report logloss on different, non-comparable subsets of size N.

4.3 Memory-based Reasoning (Creecy et al. 1992)

The following parameters are varied:

- whether to exclude stop words or not;
- whether to use stemming or not;
- choice of similarity metric to calculate 'nearness' between observations;
- choice of k, the number of nearest matches to consider.

In all trials we use extended preprocessing and not just removal of punctuation marks. The reason is that preprocessing standardizes the text better by replacing German umlauts and other special characters as well. The exact type of preprocessing should, however, not lead to very different results (and we did see an empirical proof of this when tuning the adapted nearest neighbor algorithm).

The different parameter configurations are evaluated on data set D (Table A8) and data set C (Table A9). The agreement rates are very similar in all situations unless k is chosen too small. This suggests that the exact parameter configuration is of minor relevance, mirroring a finding by Creecy et al. (1992). Note, however, that the metric used for tuning, the agreement rate, relates to 100% production rate. We did not try if the agreement rate among top 1 could be improved at lower production rates using a different parameter configuration.

For all of our analyses reported in the main text we use extended preprocessing, exclude stop words, apply the German Porter stemmer, and use the ERROR-metric with $k = 7$.

4.4 Tree Boosting (`XGBoost`)

Tuning an XGBoost-model is a demanding task because many different tuning parameters are available and interactions between them need to be considered. Due to the special characteristics of occupational data, we use a parameter configuration that is different from standard recommendations.

Parameter tuning of logistic regression (see above) suggested that the German Porter stemmer can improve performance. The exclusion of stop words and counting of words did not have a large impact on the results. For simplicity we do not analyze this again, but use stemming and do not remove stop words with XGBoost. A variable counting the number of words is included in the predictor matrix to allow for possible interactions with it.

Tree boosting is an iterative process, where the r-th tree aims to find a signal that the $r-1$ previous iterations have not fully detected. To save time, we aim to achieve close-to-optimal predictions after approximately 20 rounds and fix the maximal number of iterations at 40. η (and other parameters) control the learning rate and larger values of η will make sure that we obtain reasonable results by then. However, if η is too large or if we run too many iterations, overfitting becomes an issue and the r-th tree will pick up noise in the training data that will lead to poor predictions in the test data. In such

Table A8: Agreement rate of memory-based reasoning in data set D under different configurations.†

$metric$	k	$stem$	$stop$	$agreem.rate$
SUM	2	NO	YES	0.508 ± 0.015
SUM	2	NO	NO	0.502 ± 0.015
SUM	2	YES	YES	0.497 ± 0.015
SUM	2	YES	NO	0.486 ± 0.015
SUM	7	NO	YES	0.567 ± 0.015
SUM	7	NO	NO	0.563 ± 0.015
SUM	7	YES	YES	0.568 ± 0.015
SUM	7	YES	NO	0.569 ± 0.015
SUM	12	NO	YES	0.573 ± 0.015
SUM	12	NO	NO	0.572 ± 0.015
SUM	12	YES	YES	0.571 ± 0.015
SUM	12	YES	NO	0.570 ± 0.015
SUM	17	NO	YES	0.573 ± 0.015
SUM	17	NO	NO	0.574 ± 0.015
SUM	17	YES	YES	0.576 ± 0.015
SUM	17	YES	NO	0.577 ± 0.015
ERROR	2	NO	YES	0.549 ± 0.015
ERROR	2	NO	NO	0.535 ± 0.015
ERROR	2	YES	YES	0.557 ± 0.015
ERROR	2	YES	NO	0.542 ± 0.015
ERROR	7	NO	YES	0.568 ± 0.015
ERROR	7	NO	NO	0.564 ± 0.015
ERROR	7	YES	YES	0.576 ± 0.015
ERROR	7	YES	NO	0.570 ± 0.015
ERROR	12	NO	YES	0.564 ± 0.015
ERROR	12	NO	NO	0.555 ± 0.015
ERROR	12	YES	YES	0.569 ± 0.015
ERROR	12	YES	NO	0.566 ± 0.015
ERROR	17	NO	YES	0.567 ± 0.015
ERROR	17	NO	NO	0.554 ± 0.015
ERROR	17	YES	YES	0.569 ± 0.015
ERROR	17	YES	NO	0.559 ± 0.015

†Data set D was split into 6575 training observations and 1064 test observations.

Table A9: Agreement rate of memory-based reasoning in data set C under different configurations.†

$metric$	k	$stem$	$stop$	$agreem.rate$
SUM	2	NO	YES	0.486 ± 0.015
SUM	2	NO	NO	0.479 ± 0.015
SUM	2	YES	YES	0.482 ± 0.015
SUM	2	YES	NO	0.475 ± 0.015
SUM	7	NO	YES	0.575 ± 0.015
SUM	7	NO	NO	0.571 ± 0.015
SUM	7	YES	YES	0.569 ± 0.015
SUM	7	YES	NO	0.565 ± 0.015
SUM	12	NO	YES	0.573 ± 0.015
SUM	12	NO	NO	0.570 ± 0.015
SUM	12	YES	YES	0.566 ± 0.015
SUM	12	YES	NO	0.563 ± 0.015
SUM	17	NO	YES	0.576 ± 0.015
SUM	17	NO	NO	0.571 ± 0.015
SUM	17	YES	YES	0.576 ± 0.015
SUM	17	YES	NO	0.571 ± 0.015
ERROR	2	NO	YES	0.573 ± 0.015
ERROR	2	NO	NO	0.567 ± 0.015
ERROR	2	YES	YES	0.570 ± 0.015
ERROR	2	YES	NO	0.564 ± 0.015
ERROR	7	NO	YES	0.586 ± 0.015
ERROR	7	NO	NO	0.582 ± 0.015
ERROR	7	YES	YES	0.582 ± 0.015
ERROR	7	YES	NO	0.581 ± 0.015
ERROR	12	NO	YES	0.571 ± 0.015
ERROR	12	NO	NO	0.568 ± 0.015
ERROR	12	YES	YES	0.571 ± 0.015
ERROR	12	YES	NO	0.569 ± 0.015
ERROR	17	NO	YES	0.568 ± 0.015
ERROR	17	NO	NO	0.564 ± 0.015
ERROR	17	YES	YES	0.564 ± 0.015
ERROR	17	YES	NO	0.56 ± 0.015

†Data set C was split into 47,930 training observations and 1064 test observations.

cases we use early stopping and report results from earlier iterations before overfitting occurred.

In multiclass classification with K classes, XGBoost learns in each iteration K trees with binary outcome. The parameters max_depth, min_child_weight, γ, and $tree\_method$ control the tree growing process.

- max_depth determines the maximal number of leaves in a tree, 2^{max_depth}. For example a tree of depth 2 can have a leaf for "chicken" (a first partition) and another leaf for "egg" (a second partition). These leaves should output high probabilities for chicken farming occupations. Trees with depth 2 also allow for a leaf that may predict a probability if the word from the first partition ("chicken") co-occurs with some other word. A final leaf will cover all other verbal answers that neither contain "egg" nor "chicken". More than four leaves are not possible with $max\_depth = 2$. Thus, if max_depth is chosen too small, a single tree cannot detect all words (and possibly co-occurrences of words) that are predictive of a given category. A large number of iterations would then be needed before the boosting algorithm can make good predictions.

- min_child_weight controls that the tree building process is stopped whenever the sum of hessians in a leaf $\sum_{i \in \text{leaf}} \partial^2_{\hat{y}_i^{(r-1)}} l(y_i, \hat{y_i}^{(r-1)}) = \sum_{i \in \text{leaf}} p_i(1-p_i)$ is smaller than min_child_weight. We anticipate that tree leaves can be very pure (a leaf for the job title "beekeeper" perfectly predicts a single category, $p_i = 1$), which suggests that min_child_weight should be set to zero.

- γ determines the minimum loss reduction needed to split a leaf further. If it is too small, many words that are not job-related (e.g., "and", "am", ...) will be used to make predictions about job categories, which does not seem reasonable. If it is too large, the algorithm will ignore important words.

- $tree\_method$ controls the tree construction algorithm, which mainly differ in how they enumerate possible splitting points. We use the exact greedy algorithm ($tree\_method =' exact'$), reasoning that our predictor variables are mostly binary (either a word occurs in a verbal answer or not) and faster but more approximate algorithms for continuous predictors should not be necessary.

λ (L2 norm), α (L1 norm), and max_delta_step are shrinkage parameters that control by how much the weights from each leaf are shrunk towards zero. max_delta_step is recommended in situations as ours with high class imbalance and we find it to be one of the most important parameters. It defines a maximal weight that can be returned from a single leaf, making sure that the leaf does not overfit by too much. Because the returned value is then multiplied by η, the choice of max_delta_step has a direct impact on the learning rate.

The parameters $subsample$, $colsample\_bytree$, and $colsample\_bylevel$ let us specify if we do not want to train every tree with the complete data. $subsample$ draws a sample of training instances and uses only the selected cases to learn a tree. The $colsample$ parameters let us randomly sample a set of variables to be used for tree learning. The XGBoost default implementation fixes all three parameters at a proportion 1 (=use complete data), but lowering the proportions can speed up computation

of each iteration (although we will need more iterations) and may prevent overfitting. In our application we are skeptical whether drawing a subsample of variables that will be used to construct the complete tree ($colsample\_bytree$) is reasonable. If two words interact, but the subsample contains only one of the two words, these interactions could not be detected during tree learning.

An important property of tree boosting should be noted. All categories are predicted with probability close to zero in the first few iterations and the probabilities will usually increase in each iteration if this improves the objective function. Consequently, the sum of probabilities of the most probable category ($\sum_{n_f=1}^{N_{test}} \hat{p}_{n_f}^{(1)}$ in the notation from Section 2) increases with the number of iterations. However, it should not overshoot the observed number of cases correctly predicted, $\sum_{n_f=1}^{N_{test}} a_{n_f}^{(1)}$, which would be hard to reverse and is a sign of overfitting. In our experience, increasing γ or λ helps to prevent overshooting behavior.

Our main objective is to minimize logloss. We thus stop iterating when logloss stays approximately constant. If we were to continue iterating for too long, overfitting would occur and logloss would deteriorate. However, we do not always iterate until logloss remains constant and report non-optimal logloss for some parameter configurations. Two reasons for this exist that must be kept in mind when interpreting the results.

- The first reason is that we run at most 40 iterations. If we had time for more iterations, logloss might improve further.
- The second reason is that we require calibrated probabilities. $\bar{\hat{p}}_{n_f}^{(1)} = \frac{1}{N_{test}} \sum_{n_f=1}^{N_{test}} \hat{p}_{n_f}^{(1)}$ increases with the number of iterations and it may overshoot the observed agreement rate. We thus stop early if $\bar{\hat{p}}_{n_f}^{(1)}$ is by 0.01 points larger than the agreement rate. When this is the reason for stopping, logloss is often satisfactory, but further improvements might be possible with additional fine-tuning of parameters.

Many different parameter configurations were tested on data set D until we finally reached a default parameter configuration that is sufficient for our purpose. Results are shown in Table A10. To demonstrate the performance under different parameter configurations, all parameters except the one indicated are kept constant, which allows to compare deviating configurations with the default configuration. The optimal parameter configuration would minimize logloss and sharpness, the latter is only useful under the condition that probabilities are calibrated. High agreement rates are also desirable, but note that most differences in the agreement rate are rather small and may be due to chance. In addition, we would like that the mean most probable probability over all test cases, $\bar{\hat{p}}_{n_f}^{(1)} = \frac{1}{N_{test}} \sum_{n_f=1}^{N_{test}} \hat{p}_{n_f}^{(1)}$, is close to the agreement rate $agree = \frac{1}{N_{test}} \sum_{n_f=1}^{N_{test}} a_{n_f}^{(1)}$, an indicator of calibration.

The results in Table A10 confirm our theoretical reasoning. max_delta_step and η determine the rate of learning. If they are too small, results might still improve but the calculations will take too long. If they are too large, we obtain poor results within few iterations. The exact choice of max_depth makes little difference in Table A10, although we will find later that similar results are obtainable in fewer iterations if max_depth is set to larger values (Table A12). With small γ we observe overshooting

behavior and the algorithm stops early. Fixing $\lambda = 0$ leads to poor learning behavior. However, if either γ or λ are set to overly large values, $\bar{\hat{p}}_{nf}^{(1)}$ will underestimate the agreement rate. Increasing α leads to additional iterations and makes the results worse. Similarly, the results deteriorate drastically if $min\_child\_weight > 0$. Furthermore, no improvements can be observed if $subsample$ is varied or if $colsample\_bytree$ is reduced, but sampling smaller proportions will increase the number of iterations needed. Changing $colsample\_bylevel$ to 0.6 leads to minimal improvements in logloss, but sharpness and the agreement rate are worse, a behavior that could be explored further. All in all, one could certainly try additional parameter configurations and/or run more iterations to achieve better results, though this time would be misspend. Results of most parameter configurations are already quite similar to another, suggesting that our default parameter configuration is not too far away from some hypothetical optimum.

Since the default parameter configuration was optimized in data set D, another question is if the results can be generalized to other occupational data. Table A11 shows performance measures in data set C using identical parameter configurations as before. The most salient patterns described above are confirmed (referring to max_delta_step, η, max_depth (partly confirmed), γ, λ, min_child_weight, $colsample\_bytree$) except that changes in α may actually improve performance and lowering $subsample$ has in this larger data set no effect on the number of iterations needed. Among the reported parameter configurations, the logloss is minimal with $\lambda = 1$. With the default parameter configuration logloss is 0.083 bits away from this observed optimum, still a reasonable choice that will be used in the main paper for a comparison of XGBoost with other methods. Yet, if XGBoost predictions of data set C were used for practical applications, we would recommend trying a few more parameter configurations, maybe increasing γ and/or λ in a first step. This would prevent the omnipresent overshooting behavior in this table and reduce logloss further.

Table A10: Performance measures of XGBoost in data set D. All parameters except one are kept constant at the default configuration.†

Parameter configuration	$iter$††	$agree$	$\bar{\hat{p}}_{n_f}^{(1)}$	$logloss$	$sharp$
$default$	22 [1]	0.591	0.588	3.826	3.788
$max\_delta\_step = 0.33$	40 [3]	0.583	0.543	$\leq$3.878	4.210
$max\_delta\_step = 3$	15 [1]	0.590	0.591	3.862	3.849
$\eta = 0.25$	40 [3]	0.590	0.574	$\leq$3.810	4.019
$\eta = 1$	9 [2]	0.572	0.576	$\leq$3.915	3.944
$max\_depth = 10$	23 [1]	0.583	0.580	3.885	3.690
$max\_depth = 30$	22 [1]	0.582	0.589	3.818	3.806
$\gamma = 0.3$	16 [2]	0.583	0.592	$\leq$4.007	4.034
$\gamma = 0.9$	22 [1]	0.585	0.554	3.826	4.079
$\lambda = 0$	11 [2]	0.039	0.049	$\leq$8.128	8.228
$\lambda = 1$	26 [1]	0.576	0.525	3.931	4.505
$\alpha = 0.2$	26 [1]	0.583	0.545	3.865	4.346
$min\_child\_weight = 0.01$	22 [1]	0.555	0.543	4.122	4.213
$subsample = 0.5$	28 [1]	0.579	0.588	3.861	3.689
$subsample = 1$	20 [1]	0.583	0.589	3.857	3.863
$colsample\_bytree = 0.6$	28 [1]	0.580	0.533	3.858	4.072
$colsample\_bylevel = 0.6$	22 [1]	0.575	0.570	3.808	3.952

†Data set D was split into 6575 training observations and 1064 test observations. The complete test set was used to calculate logloss. Default parameter configuration: Stop words = NO, Stemming = YES, extended preprocessing = YES, word counts = YES, max. rounds = 40, early stopping if performance does not improve for 1 round, $\eta = 0.5$, max_delta_step = 1, max_depth = 20, $\gamma = 0.6$, $\lambda = 1e^{-4}$, $\alpha = 0$, min_child_weight = 0, subsample = 0.75, colsample_by_tree = 1, colsample_by_level = 1.

††Reason for stopping at r-th iteration: [1] = Logloss minimal or approximately constant; [2] = $\bar{\hat{p}}_{n_f}^{(1)} - agree > 0.01$ in subsequent iteration; [3] = maximal number of iterations reached. Reasons [2] and [3] entail the possibility to reduce logloss further (indicated by $\leq$).

Table A11: Performance measures of XGBoost in data set C. All parameters except one are kept constant at the default configuration.†

Parameter configuration	$iter$††	$agree$	$\bar{\hat{p}}_{n_f}^{(1)}$	$logloss$	$sharp$
$default$	19 [2]	0.597	0.605	$\leq$3.361	3.321
$max\_delta\_step = 0.33$	39 [2]	0.590	0.597	$\leq$3.327	3.265
$max\_delta\_step = 3$	10 [2]	0.582	0.586	$\leq$3.503	3.809
$\eta = 0.25$	36 [2]	0.589	0.594	$\leq$3.371	3.498
$\eta = 1$	7 [2]	0.578	0.564	$\leq$3.539	3.929
$max\_depth = 10$	30 [2]	0.586	0.594	$\leq$3.374	3.291
$max\_depth = 30$	17 [2]	0.600	0.608	$\leq$3.385	3.377
$\gamma = 0.3$	16 [2]	0.587	0.595	$\leq$3.482	3.533
$\gamma = 0.9$	24 [2]	0.597	0.602	$\leq$3.300	3.290
$\lambda = 0$	15 [1]	0.044	0.031	8.489	8.568
$\lambda = 1$	24 [1]	0.594	0.602	3.278	3.305
$\alpha = 0.2$	24 [2]	0.594	0.602	$\leq$3.312	3.295
$min\_child\_weight = 0.01$	20 [2]	0.562	0.569	$\leq$3.608	3.731
$subsample = 0.5$	19 [2]	0.597	0.592	$\leq$3.437	3.412
$subsample = 1$	20 [2]	0.592	0.599	$\leq$3.358	3.451
$colsample\_bytree = 0.6$	27 [1]	0.598	0.575	3.347	3.549
$colsample\_bylevel = 0.6$	17 [2]	0.582	0.588	$\leq$3.408	3.571

†Data set C was split into 47.930 training observations and 1064 test observations. The complete test set was used to calculate logloss. Default parameter configuration: Stop words = NO, Stemming = YES, extended preprocessing = YES, word counts = YES, max. rounds = 40, early stopping if performance does not improve for 1 round, $\eta = 0.5$, max_delta_step = 1, max_depth = 20, $\gamma = 0.6$, $\lambda = 1e^{-4}$, $\alpha = 0$, min_child_weight = 0, subsample = 0.75, colsample_by_tree = 1, colsample_by_level = 1.

††Reason for stopping at r-th iteration: [1] = Logloss minimal or approximately constant; [2] = $\bar{\hat{p}}_{n_f}^{(1)} - agree > 0.01$ in subsequent iteration; [3] = maximal number of iterations reached. Reasons [2] and [3] entail the possibility to reduce logloss further (indicated by $\leq$).

4.4.1 Tree Boosting (`XGBoost` \w Coding Index)

This algorithm (algorithm 7) is the same as before (algorithm 6), except that training observations (used within algorithm 6) and job titles from the coding index (used within algorithms 1 and 2) are pooled. This increases the number of observations in the training data substantially. Moreover, we suspect differences between both parts of the training data regarding how often the assigned categories are identical with what human coders would pick. To account for such differences, an additional predictor indicating the origin of each observation, survey data or coding index, is created. When predicting new test observations, the indicator is set to a value as if test observations originate from survey data (what they do).

With the expanded data set, training took more time and the results were not optimal when trying the same tuning parameters as before (algorithm 6). To speed up the training phase, the learning rate η is changed to 0.75 and the tree depth is doubled to 40. All other tuning parameters remain as before. Table A12 shows that results in data set D would not improve by much if a different parameter configuration would have been used. Minor improvements still seem possible by reducing the values for max_depth, $subsample$ or $colsample\_bytree$. Since this requires additional time-demanding iterations during training, we did not explore this direction further.

The same parameter configuration we used with data set D was also tested with data set C (see Table A13). In general, the algorithm stops early due to overshooting behavior, $\bar{\hat{p}}_{nf}^{(1)} - agree > 0.01$, and further improvements appear possible. To stop the overshooting behavior, we set γ to 1.5. For each data set we tried both configurations, $\gamma = 0.6$ and $\gamma = 1.5$, and choose for our final model the value for γ which gives superior performance (see Table A14).

Table A12: Performance measures of XGBoost (with coding index) in data set D. All parameters except one are kept constant at the default configuration.†

Parameter configuration	$iter$††	$agree$	$\bar{\hat{p}}_{n_f}^{(1)}$	$logloss$	$sharp$
$default$	22 [1]	0.633	0.630	3.196	3.376
$max\_delta\_step = 0.33$	33 [1]	0.619	0.584	3.288	3.662
$max\_delta\_step = 3$	16 [1]	0.647	0.646	3.264	3.294
$\eta = 0.5$	35 [1]	0.635	0.617	3.232	3.530
$\eta = 1$	16 [1]	0.634	0.634	3.252	3.281
$max\_depth = 20$	35 [1]	0.644	0.634	3.114	3.226
$max\_depth = 60$	23 [1]	0.639	0.631	3.224	3.398
$\gamma = 0.3$	11 [2]	0.600	0.604	$\leq$3.516	3.789
$\gamma = 0.9$	22 [1]	0.622	0.600	3.280	3.669
$\gamma = 1.5$	20 [1]	0.630	0.546	3.329	4.248
$\lambda = 0$	9 [1]	0.004	0.009	9.217	9.614
$\lambda = 1$	18 [1]	0.625	0.584	3.290	3.824
$\alpha = 0.2$	18 [1]	0.629	0.590	3.293	3.828
$min\_child\_weight = 0.01$	18 [1]	0.600	0.597	3.606	3.813
$subsample = 0.5$	26 [1]	0.635	0.641	3.145	3.148
$subsample = 1$	19 [1]	0.625	0.615	3.389	3.611
$colsample\_bytree = 0.6$	33 [1]	0.648	0.616	3.020	3.385
$colsample\_bylevel = 0.6$	19 [1]	0.628	0.615	3.212	3.514

†Data set D was split into 6575 training observations and 1064 test observations. The coding index was appended to the training data. The complete test set was used to calculate logloss. Default parameter configuration: Stop words = NO, Stemming = YES, extended preprocessing = YES, word counts = YES, max. rounds = 40, early stopping if performance does not improve for 1 round, $\eta = 0.75$ *(different from above)*, max_delta_step = 1, max_depth = 40 *(different from above)*, $\gamma = 0.6$, $\lambda = 1e^{-4}$, $\alpha = 0$, min_child_weight = 0, subsample = 0.75, colsample_by_tree = 1, colsample_by_level = 1.

††Reason for stopping at r-th iteration: [1] = Logloss minimal or approximately constant; [2] = $\bar{\hat{p}}_{n_f}^{(1)} - agree > 0.01$ in subsequent iteration; [3] = maximal number of iterations reached. Reasons [2] and [3] entail the possibility to reduce logloss further (indicated by $\leq$).

Table A13: Performance measures of XGBoost (with coding index) in data set C. All parameters except one are kept constant at the default configuration.†

Parameter configuration	$iter$††	$agree$	$\bar{\hat{p}}_{n_f}^{(1)}$	$logloss$	$sharp$
$default$	10 [2]	0.596	0.594	$\leq$3.277	3.609
$max\_delta\_step = 0.33$	23 [2]	0.602	0.608	$\leq$3.184	3.207
$max\_delta\_step = 3$	6 [2]	0.610	0.606	$\leq$3.299	3.551
$\eta = 0.5$	16 [2]	0.600	0.605	$\leq$3.236	3.491
$\eta = 1$	7 [2]	0.588	0.590	$\leq$3.342	3.672
$max\_depth = 20$	17 [2]	0.604	0.613	$\leq$3.178	3.191
$max\_depth = 60$	10 [2]	0.604	0.614	$\leq$3.235	3.411
$\gamma = 0.3$	10 [2]	0.602	0.608	$\leq$3.306	3.472
$\gamma = 0.9$	12 [2]	0.602	0.611	$\leq$3.218	3.329
$\gamma = 1.5$	19 [1]	0.606	0.599	3.128	3.377
$\lambda = 0$	12 [1]	0.044	0.017	8.846	9.406
$\lambda = 1$	12 [2]	0.605	0.610	$\leq$3.195	3.295
$\alpha = 0.2$	12 [2]	0.607	0.611	$\leq$3.157	3.306
$min\_child\_weight = 0.01$	11 [2]	0.573	0.581	$\leq$3.516	3.712
$subsample = 0.5$	11 [2]	0.591	0.595	$\leq$3.277	3.519
$subsample = 1$	11 [2]	0.602	0.612	$\leq$3.229	3.403
$colsample\_bytree = 0.6$	21 [2]	0.612	0.615	$\leq$3.081	3.073
$colsample\_bylevel = 0.6$	10 [2]	0.602	0.592	$\leq$3.268	3.617

†Data set C was split into 47.930 training observations and 1064 test observations. The coding index was appended to the training data. The complete test set was used to calculate logloss. Default parameter configuration: Stop words = NO, Stemming = YES, extended preprocessing = YES, word counts = YES, max. rounds = 40, early stopping if performance does not improve for 1 round, $\eta = 0.75$ *(different from above)*, max_delta_step = 1, max_depth = 40 *(different from above)*, $\gamma = 0.6$, $\lambda = 1e^{-4}$, $\alpha = 0$, min_child_weight = 0, subsample = 0.75, colsample_by_tree = 1, colsample_by_level = 1.

††Reason for stopping at r-th iteration: [1] = Logloss minimal or approximately constant; [2] = $\bar{\hat{p}}_{n_f}^{(1)} - agree > 0.01$ in subsequent iteration; [3] = maximal number of iterations reached. Reasons [2] and [3] entail the possibility to reduce logloss further (indicated by $\leq$).

Table A14: Performance measures for algorithm 7 (XGBoost with coding index) depending on γ. All other tuning parameters are set as before. Minimal logloss is highlighted for each data set, because this determines the value of γ used in the final model.†

Training data	γ	$agree$	$logloss$	$sharp$
(A)	0.6	0.676	2.804	2.765
(A)	1.5	0.681	**2.746**	3.189
(B)	0.6	0.773	**1.861**	1.943
(B)	1.5	0.771	1.922	2.519
(C)	0.6	0.602	3.298	3.599
(C)	1.5	0.612	**3.086**	3.340
(D)	0.6	0.633	**3.253**	3.462
(D)	1.5	0.624	3.359	4.375
(A,B,C,D)	0.6	0.678	2.524	2.579
(A,B,C,D)	1.5	0.691	**2.399**	2.659

†Test data are from the same data set as the training data.

5 Part E: Detailed Results

5.1 Agreement Rate vs. Production Rate (m=1)

The following diagrams show agreement rates of the most probable category at various production rates, supplementing Table 4 in the main paper.

Technical note: The curves from algorithms 2, 3, and 4 end at production rates below 100% for the following reason. If some novel answer is mentioned in the test data, for which a similar answer is not available in the training data, similarity-based algorithms cannot predict a reasonable category (and algorithms 5, 6, and 7 just output the majority category from the training data while ignoring the textual input). For example, if a curve ends at 90% production rate, the algorithm makes no prediction for the remaining 10%. To still obtain agreement rates at 100% production rate (as used in Table 4), uncodeable answers are thought of as being coded but never being in agreement with the expert-coded category.

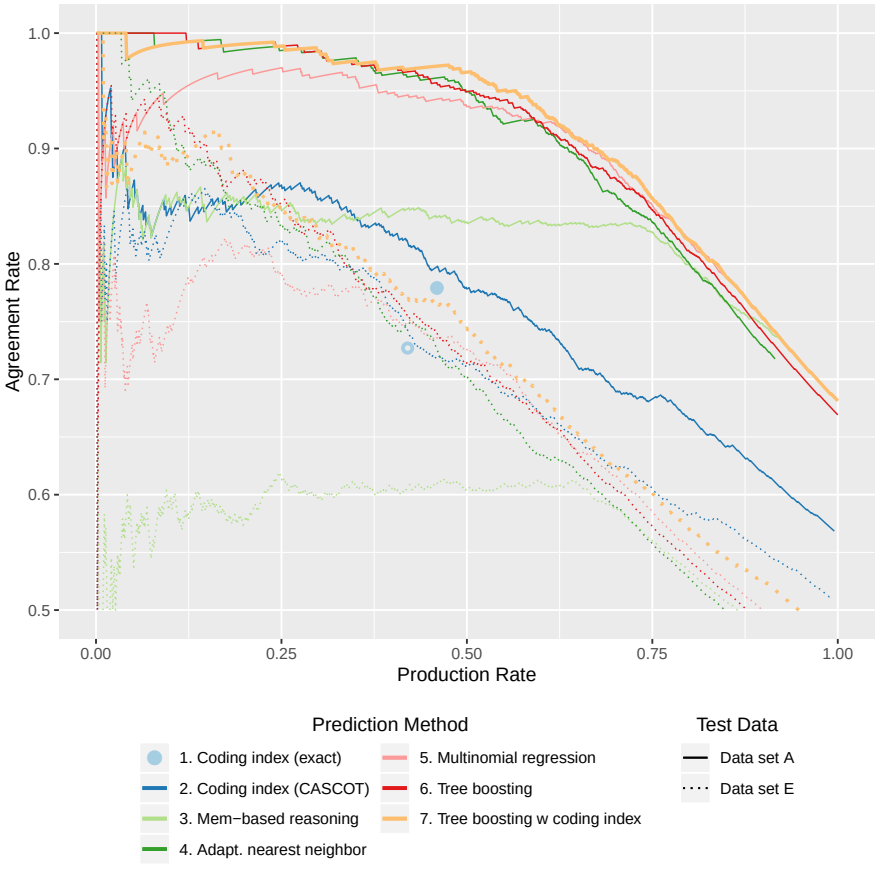

Figure A2: Agreement rates of the most probable category at various production rates; Training data: *Data set A* ($N = 31,867$)

.

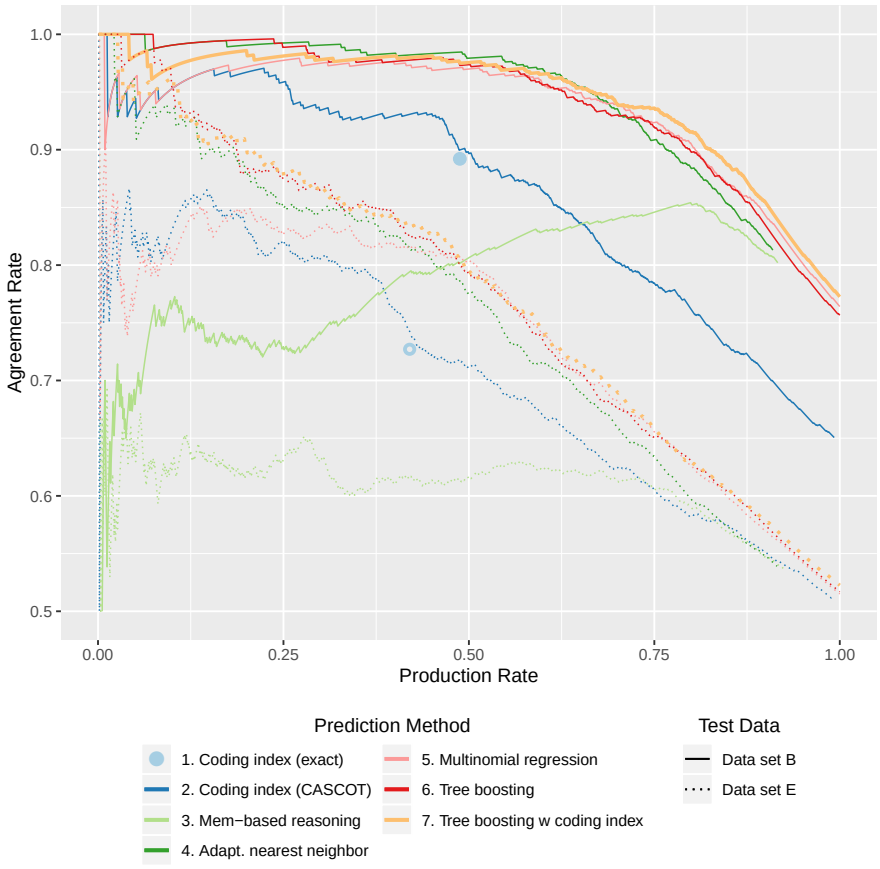

Figure A3: Agreement rates of the most probable category at various production rates; Training data: *Data set B* ($N = 54,880$)

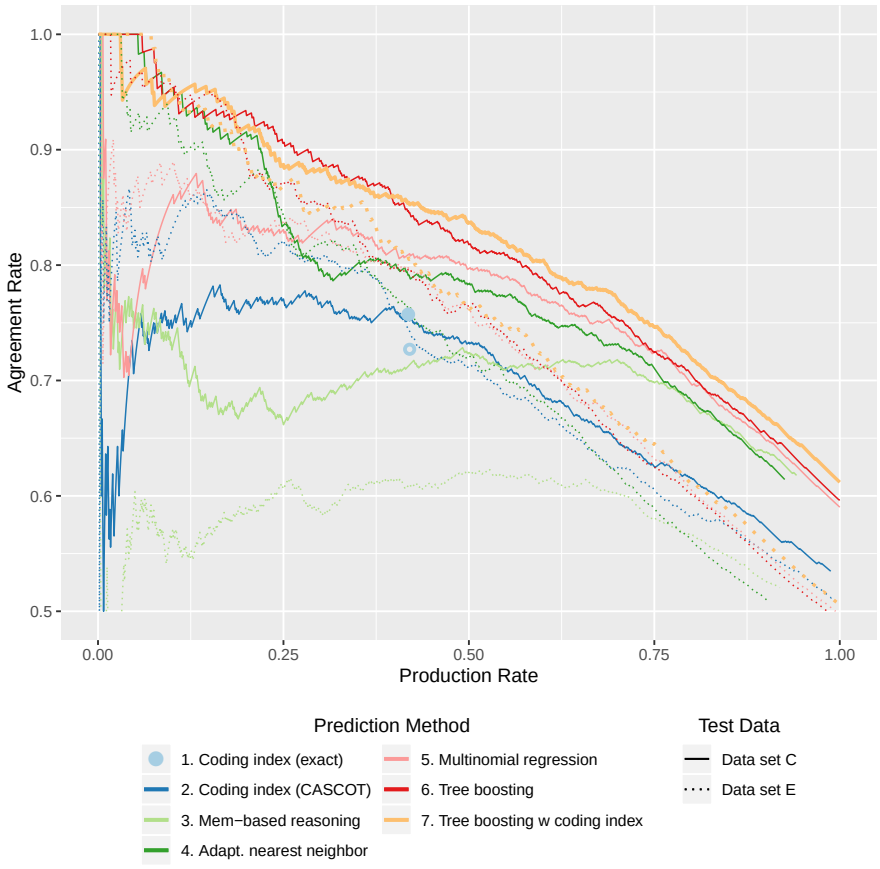

Figure A4: Agreement rates of the most probable category at various production rates; Training data: *Data set C* ($N = 47,930$)

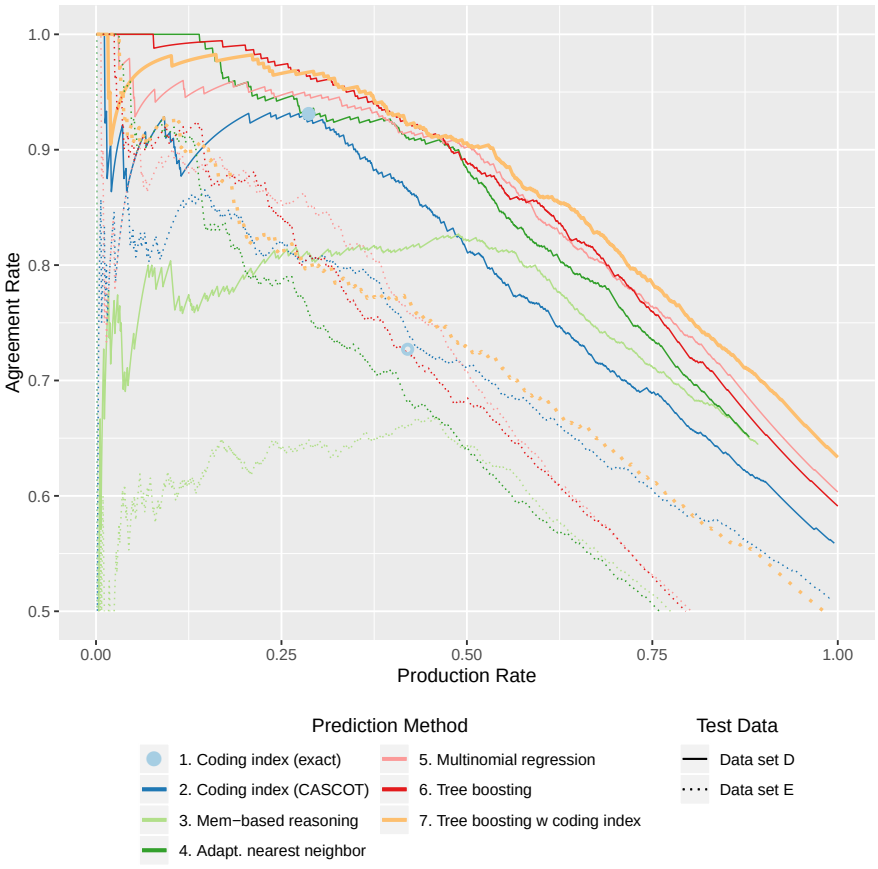

Figure A5: Agreement rates of the most probable category at various production rates; Training data: *Data set D* ($N = 6,575$)

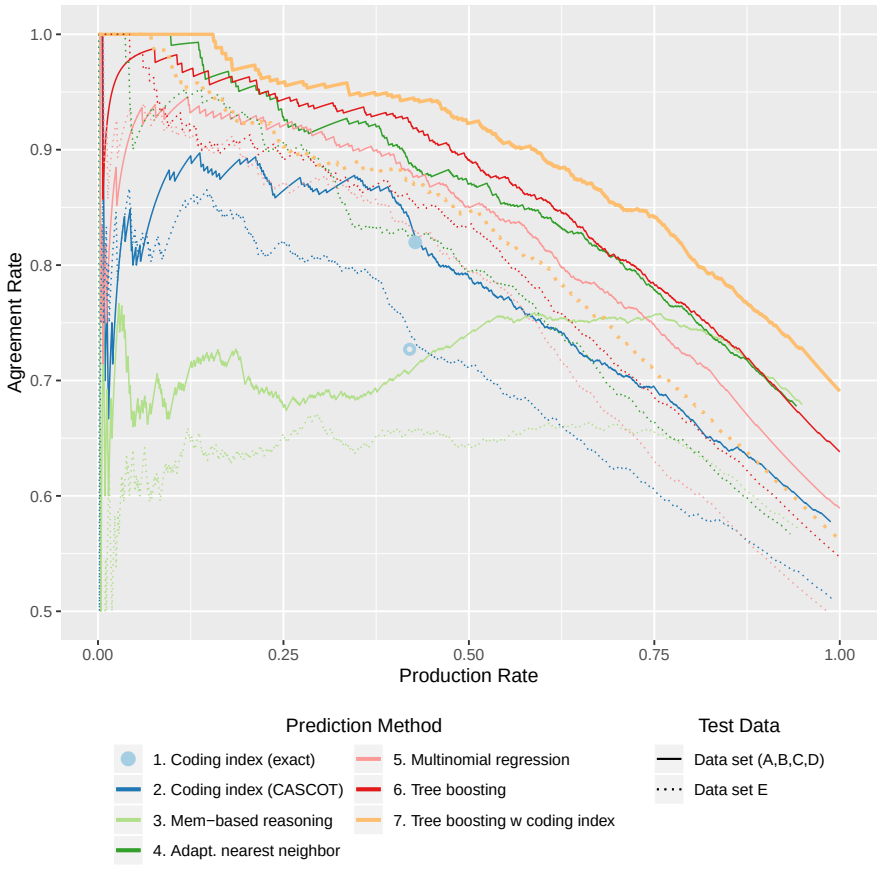

Figure A6: Agreement rates of the most probable category at various production rates; Training data: *all data sets combined* ($N = 144.444$)

5.2 True Predictions vs. False Predictions (m = 1)

Diagrams of the following type may prove helpful to choose appropriate thresholds. It makes essentially the same comparison as Figure A2, but one sees immediately that, for example, the performance of algorithm 7 in data set A levels off after around 600 true predictions (approximately at a 600 TP + 50 FP / 1064 = 61% production rate).

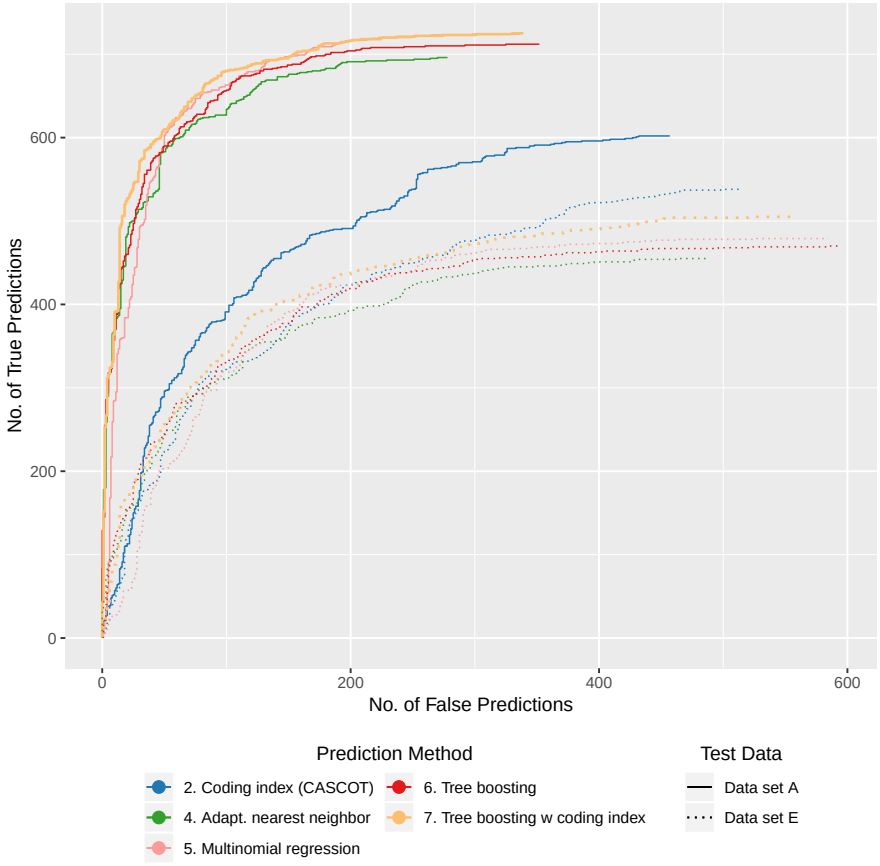

Figure A7: Number of True Predictions vs. Number of False Predictions (using only the most probable category); Training data: *Data set A* ($N = 31,867$)

5.3 Reliability Diagrams (m = 1)

Reliability diagram: Ideal probabilistic predictions should match the observed relative frequencies, the diagonal; Point size is proportional to the number of observations within each bin.

The diagrams show that predictions are better calibrated if training data and test data come from the same data set, but they are not calibrated anymore if data set E is used for testing.

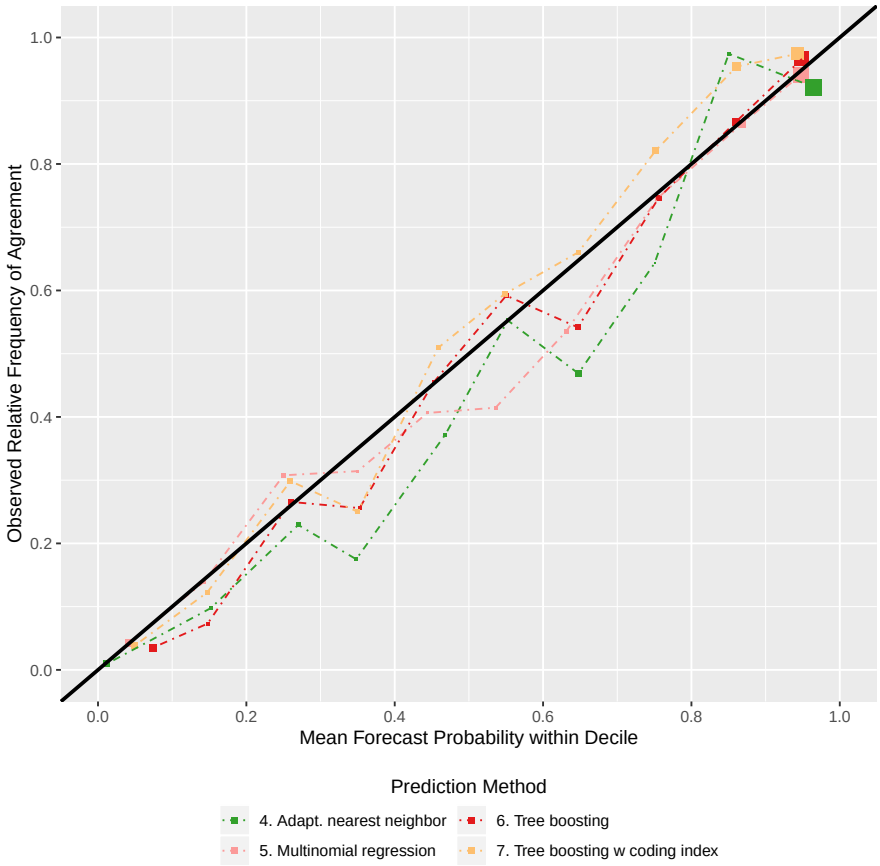

Figure A8: Reliability diagram of most probable category (m=1); Training data: *Data set A* ($N = 31,867$), Test data: *Data set A* ($N = 1.064$)

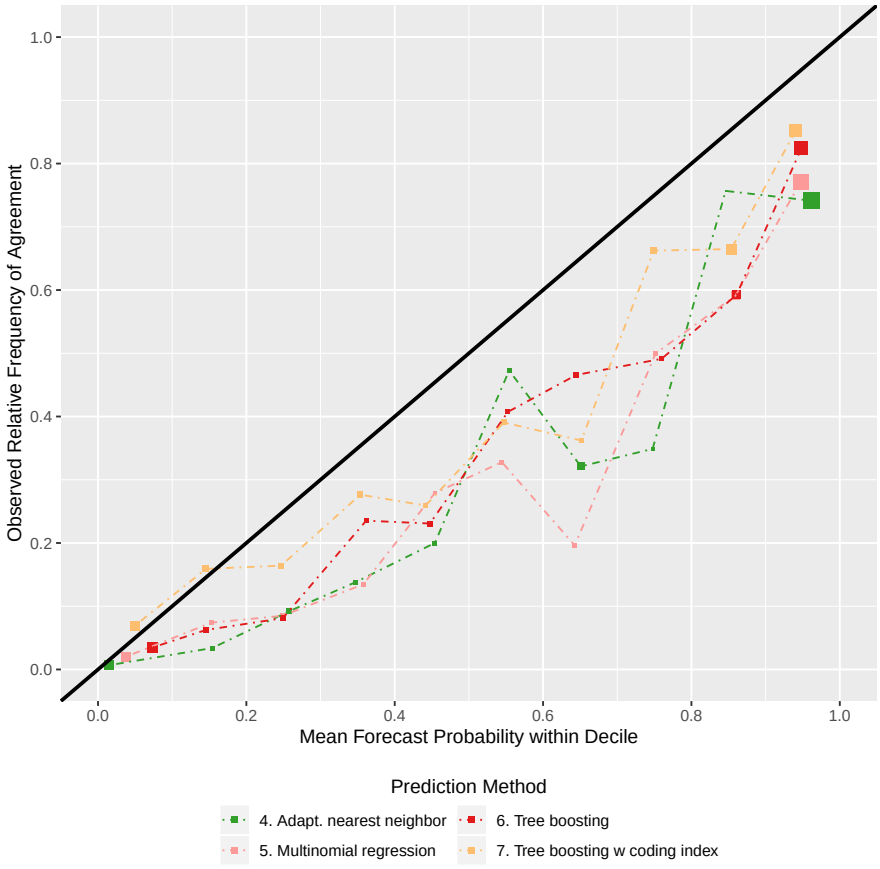

Figure A9: Reliability diagram of most probable category (m=1); Training data: *Data set A* ($N = 31,867$), Test data: *Data set E* ($N = 1.064$)

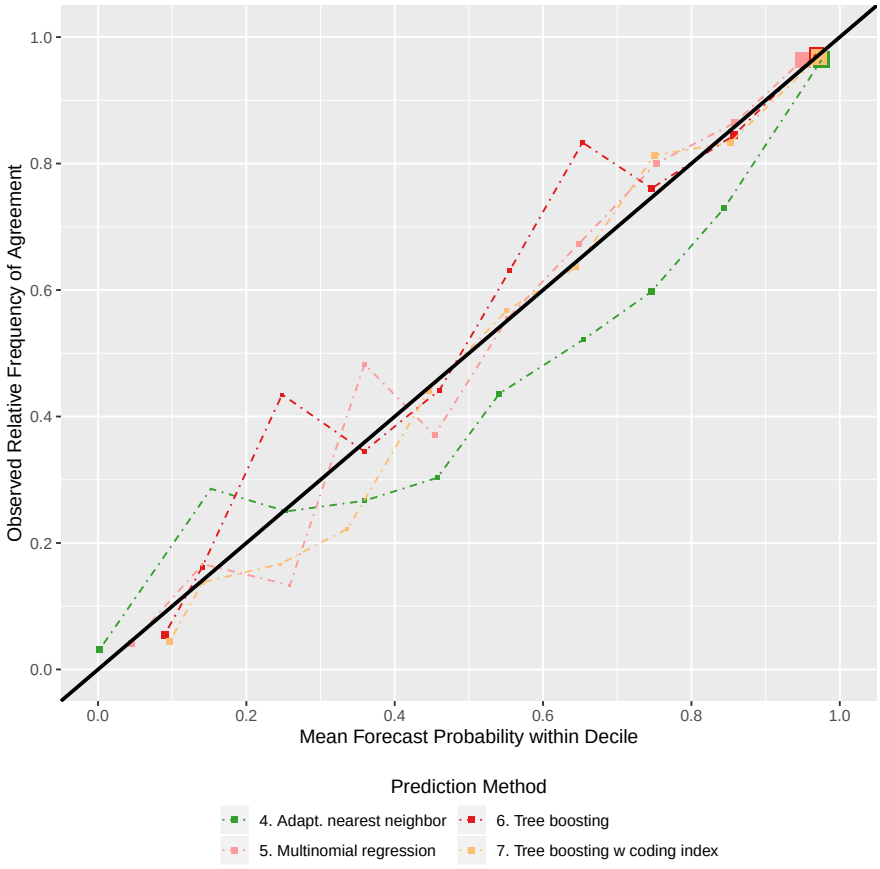

Figure A10: Reliability diagram of most probable category (m=1); Training data: *Data set B* ($N = 54,880$), Test data: *Data set B* ($N = 1.064$)

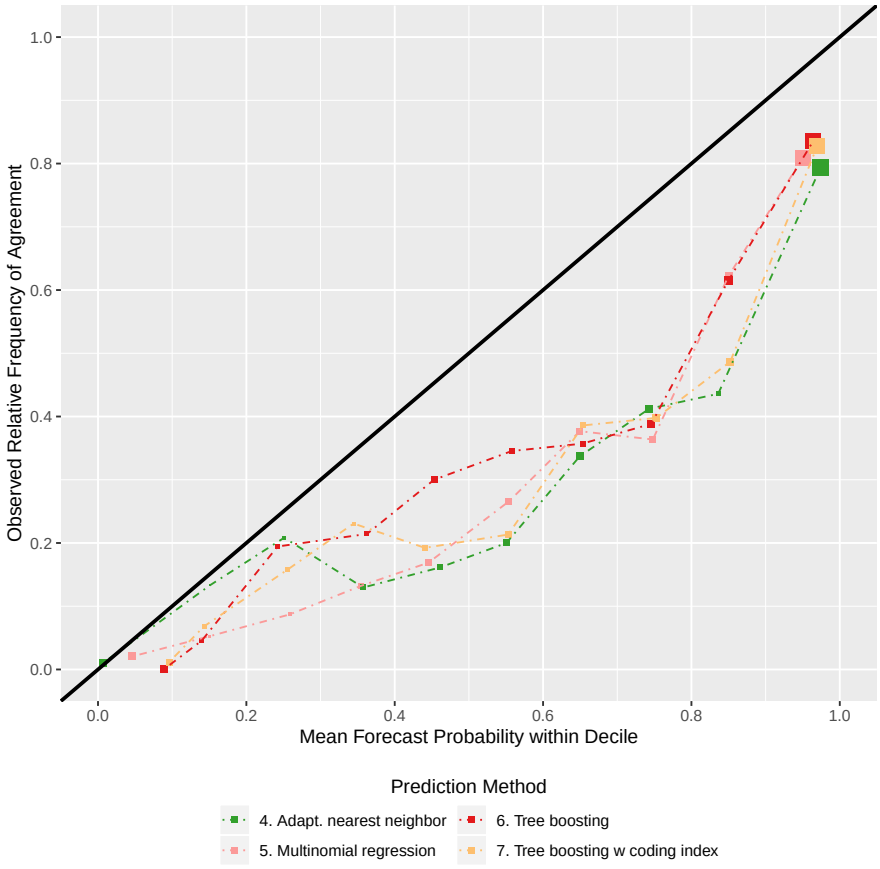

Figure A11: Reliability diagram of most probable category (m=1); Training data: *Data set B* ($N = 54,880$), Test data: *Data set E* ($N = 1.064$)

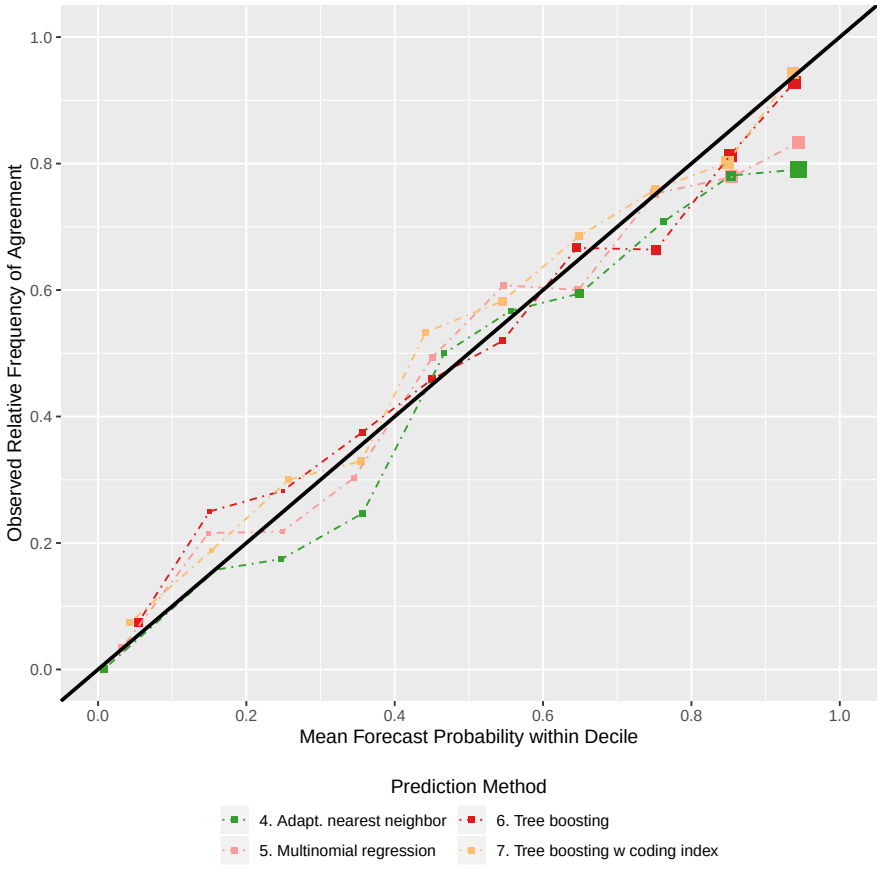

Figure A12: Reliability diagram of most probable category (m=1); Training data: *Data set C* ($N = 47,930$), Test data: *Data set C* ($N = 1.064$)

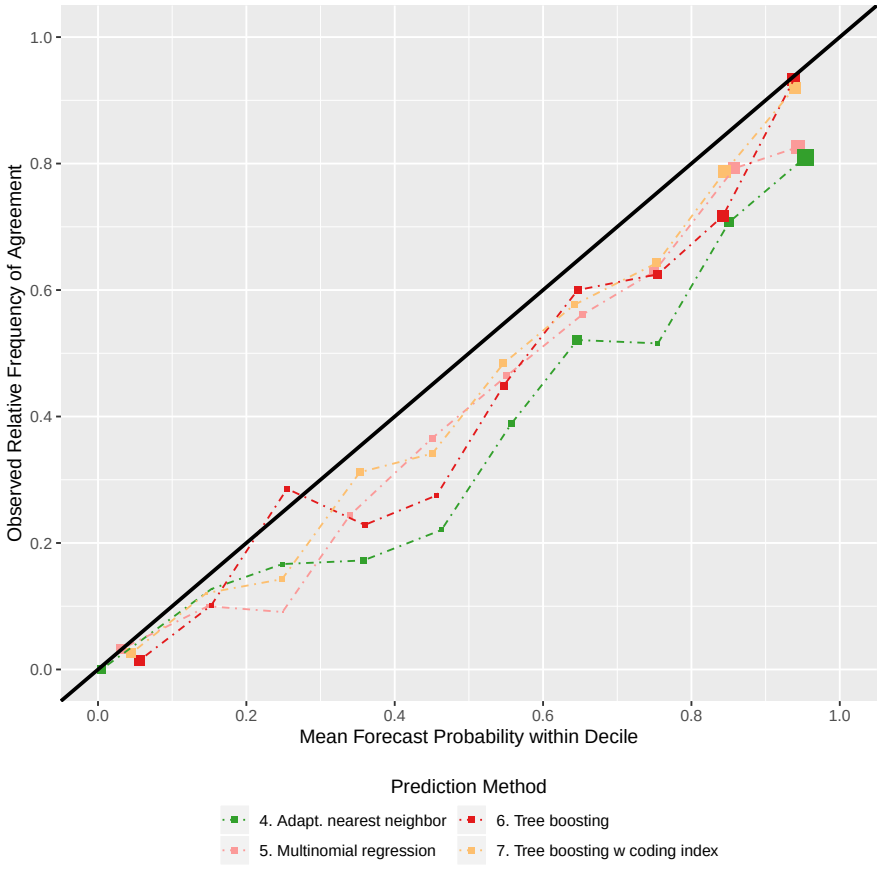

Figure A13: Reliability diagram of most probable category (m=1); Training data: *Data set C* ($N = 47,930$), Test data: *Data set E* ($N = 1.064$)

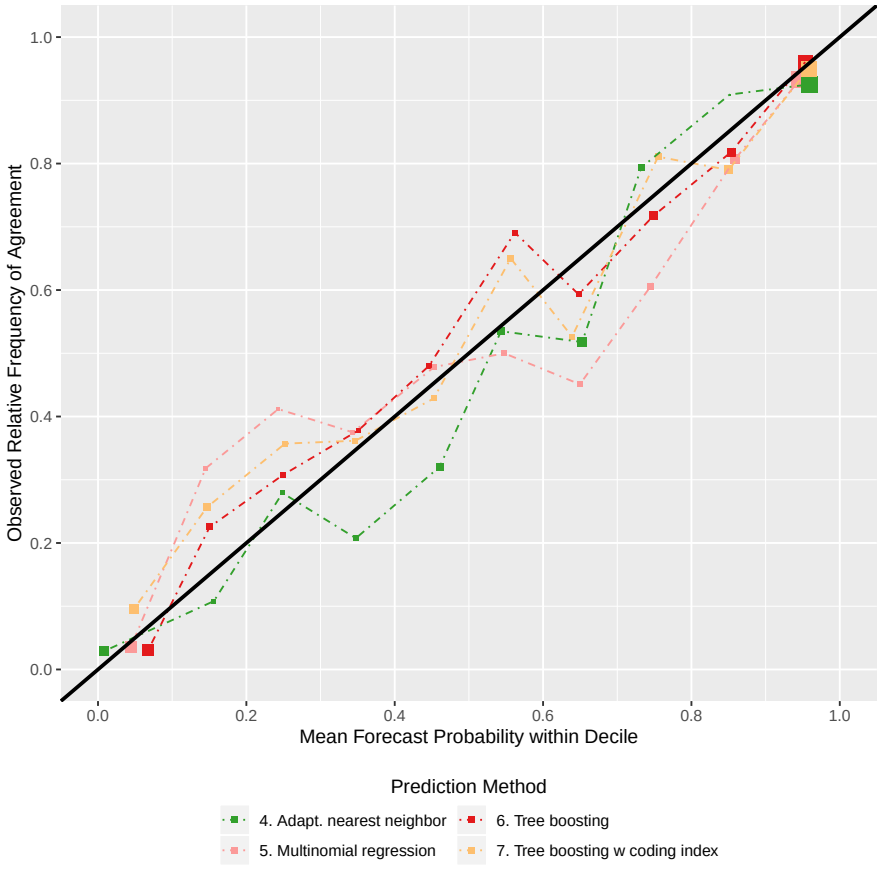

Figure A14: Reliability diagram of most probable category (m=1); Training data: *Data set D* ($N = 6,575$), Test data: *Data set D* ($N = 1.064$)

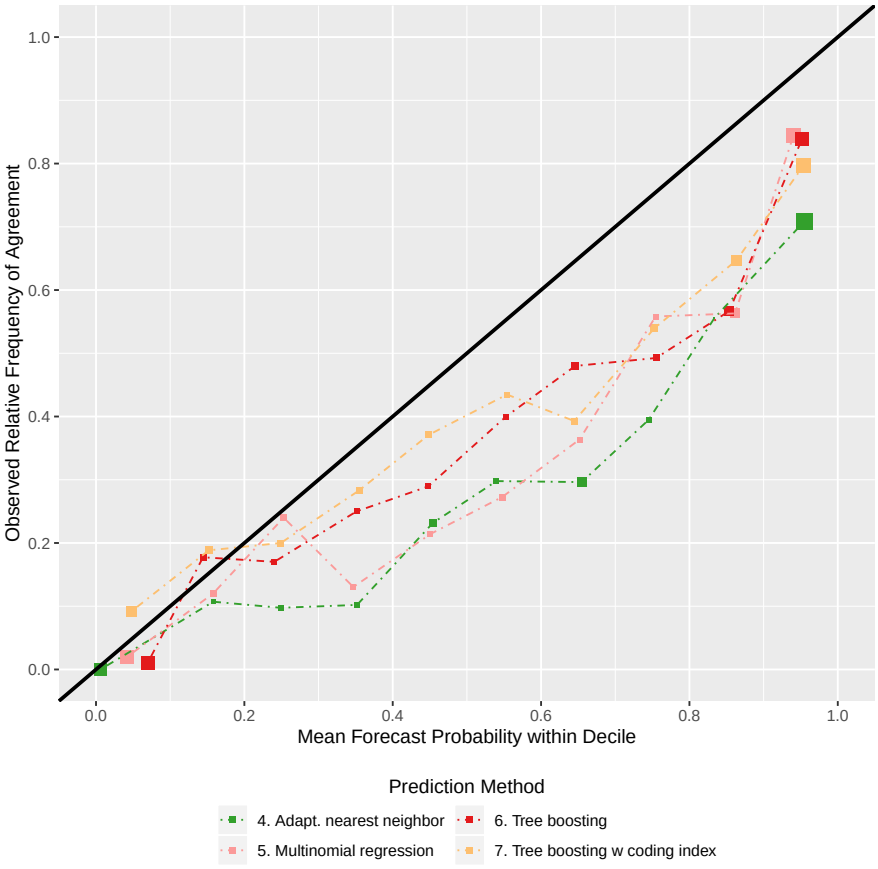

Figure A15: Reliability diagram of most probable category (m=1); Training data: *Data set D* ($N = 6,575$), Test data: *Data set E* ($N = 1.064$)

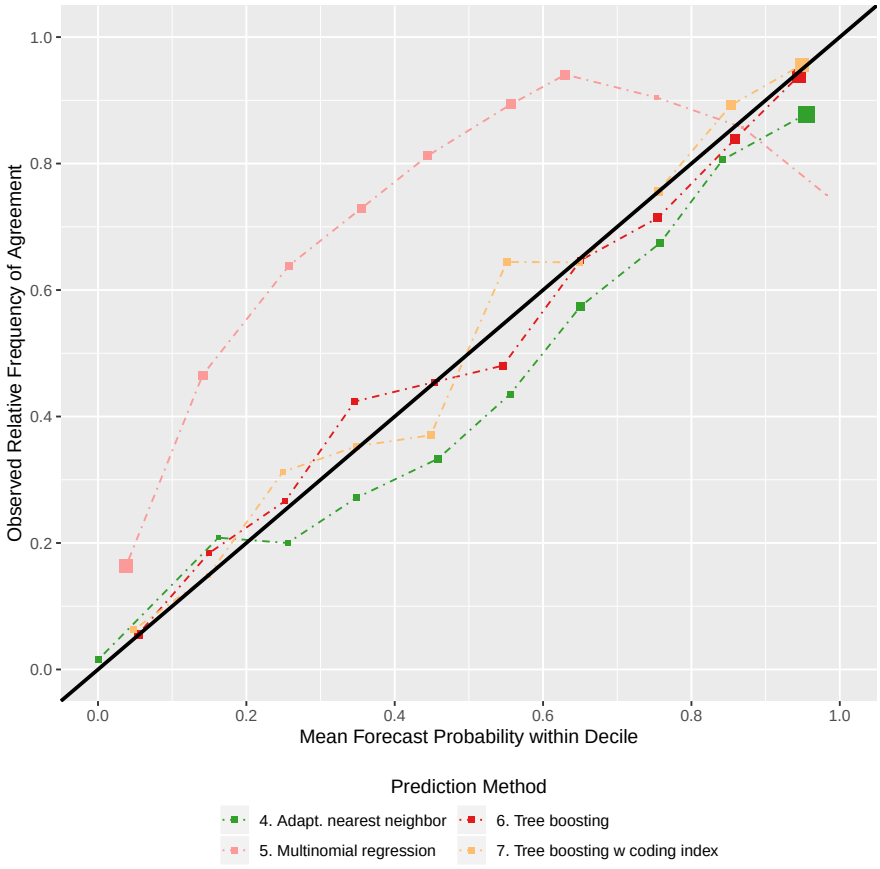

Figure A16: Reliability diagram of most probable category (m=1); Training data: *all data sets combined* ($N = 144.444$), Test data: *all data sets combined* ($N = 1.064$)

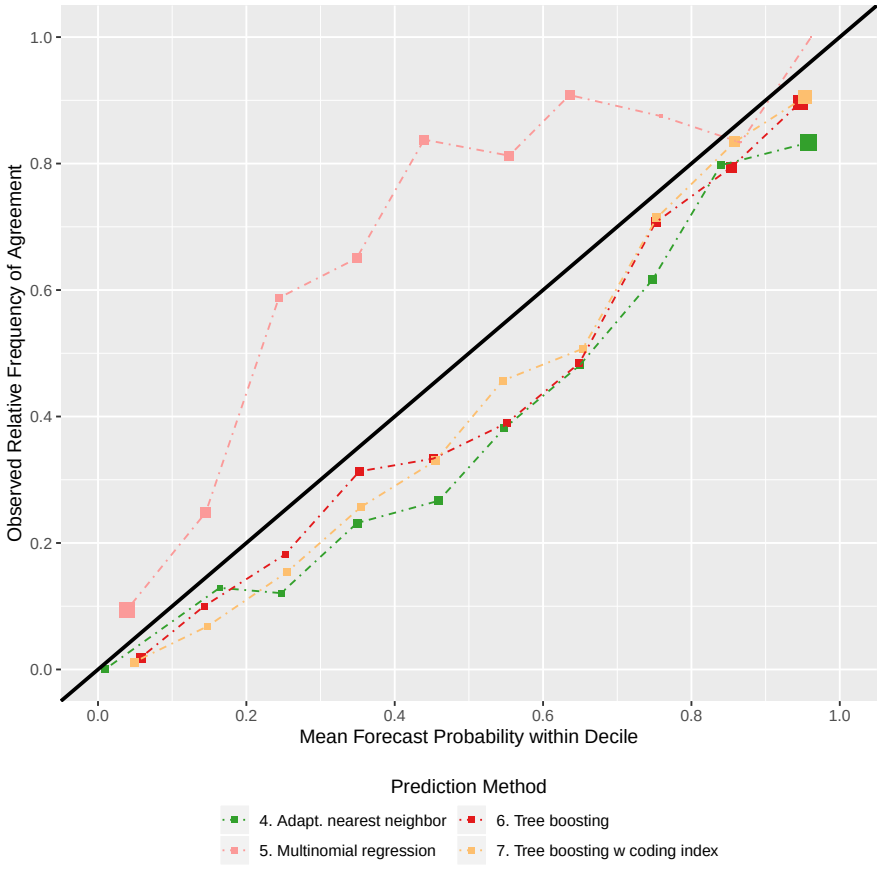

Figure A17: Reliability diagram of most probable category (m=1); Training data: *all data sets combined* ($N = 144.444$), Test data: *Data set E* ($N = 1.064$)

5.4 Sharpness

Tree boosting (\w cod index) (algorithm 7) has in most data sets greater sharpness than tree boosting (XGBoost) (algorithm 6) (see Table A15). Sharpness is only relevant if the predicted probabilities are calibrated (which is not the case if data set E is used for evaluation).

Table A15: Sharpness $\pm$ standard errors [avg. bits]†

	Training data				
	(A)	*(B)*	*(C)*	*(D)*	*(A,B,C,D)*
	Test data are from the same data set as the training data.				
4. Adapt. nearest neighbor	1.58 ± 0.09	1.35 ± 0.09	1.70 ± 0.08	1.64 ± 0.10	1.43 ± 0.07
5. Multinomial regression	2.63 ± 0.08	2.23 ± 0.08	3.17 ± 0.08	3.19 ± 0.09	6.64 ± 0.06
6. Tree boosting (XGBoost)	2.78 ± 0.08	2.25 ± 0.08	3.32 ± 0.08	3.79 ± 0.09	2.98 ± 0.08
7. Tree boosting (\w cod index)	3.19 ± 0.08	1.94 ± 0.08	3.34 ± 0.07	3.46 ± 0.09	2.66 ± 0.07
	Test data are from data set E.				
4. Adapt. nearest neighbor	2.01 ± 0.10	1.43 ± 0.09	1.96 ± 0.09	2.41 ± 0.12	1.67 ± 0.08
5. Multinomial regression	3.07 ± 0.09	2.53 ± 0.08	3.55 ± 0.08	4.03 ± 0.09	6.78 ± 0.06
6. Tree boosting (XGBoost)	3.51 ± 0.09	2.64 ± 0.08	3.71 ± 0.08	4.45 ± 0.10	3.16 ± 0.08
7. Tree boosting (\w cod index)	3.79 ± 0.09	2.33 ± 0.08	3.72 ± 0.08	3.63 ± 0.10	2.99 ± 0.08

†Adapted nearest neighbor (algorithm 4) and regularized multinomial regression (algorithm 5) frequently predict $\hat{p}_{n_f k} = 0$ for several categories k. $\hat{p}_{n_f k} \log_2 \hat{p}_{n_f k}$ is not defined in this case. In our implementation it is then set to 0, which explains the excellent sharpness of algorithms 4. and 5.

5.5 Log_2 loss

Tree boosting (\w Cod Index) (algorithm 7) achieves smaller Log_2 loss than tree boosting (XGBoost) (algorithm 6) (see Table A16). For other algorithms, usually, Log_2loss $= \infty$, because the predicted probabilities are zero for categories that realize in the test data.

Table A16: Log_2 loss $\pm$ standard errors [avg. bits] †

	Training data				
	(A)	*(B)*	*(C)*	*(D)*	*(A,B,C,D)*
	Test data are from the same data set as the training data.				
4. Adapt. nearest neighbor	∞	∞	∞	∞	∞
5. Multinomial regression	∞	∞	∞	∞	4.36 ± 0.12
6. Tree boosting (XGBoost)	2.92 ± 0.13	2.10 ± 0.11	3.36 ± 0.13	3.83 ± 0.14	2.89 ± 0.12
7. Tree boosting (\w cod index)	2.75 ± 0.12	1.86 ± 0.11	3.09 ± 0.12	3.25 ± 0.13	2.40 ± 0.11
	Test data are from data set E.				
4. Adapt. nearest neighbor	∞	∞	∞	∞	∞
5. Multinomial regression	∞	∞	∞	∞	5.17 ± 0.13
6. Tree boosting (XGBoost)	5.64 ± 0.17	4.85 ± 0.17	4.65 ± 0.15	6.16 ± 0.16	4.01 ± 0.14
7. Tree boosting (\w cod index)	4.90 ± 0.15	4.68 ± 0.17	4.34 ± 0.14	5.18 ± 0.16	3.78 ± 0.14

†Adapted nearest neighbor (algorithm 4) and regularized multinomial regression (algorithm 5) predict sometimes $\hat{p}_{nfk} = 0$ although category k was selected in the validation data ($y_{nfk} = 1$). Then, $\log_2$ loss $= \infty$

6 References

References

Albrecht, S., Schmich, P. and Varga, M. (2017). Occupation coding in the german health update (geda-study 2014/15). 7th Conference of the European Survey Research Association.
URL: *https://www.europeansurveyresearch.org/conference/programme2017?sess=4#630*

Antoni, M., Drasch, K., Kleinert, C., Matthes, B., Ruland, M. and Trahms, A. (2010). Arbeiten und Lernen im Wandel * Teil 1: Überblick über die Studie, *FDZ-Methodenreport 05/2010*, Forschungsdatenzentrum der Bundesagentur für Arbeit im Institut für Arbeitsmarkt- und Berufsforschung, Nuremberg.

Appel, M. V. and Hellerman, E. (1983). Census Bureau Experience with Automated Industry and Occupation Coding, *Proceedings of the Survey Research Methods Section: American Statistical Association*, pp. 32–40.

Bekkerman, R. and Gavish, M. (2011). High-Precision Phrase-Based Document Classification on a Modern Scale, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, ACM, New York, NY, USA, pp. 231–239.
URL: *http://doi.acm.org/10.1145/2020408.2020449*

Berg, M., Cramer, R., Dickmann, C., Gilberg, R., Jesske, B., Kleudgen, M., Beste, J., Dummert, S., Frodermann, C., Fuchs, B., Schwarz, S., Trappmann, M. and Trenkle, S. (2017). Codebuch und Dokumentation des Panel 'Arbeitsmarkt und soziale Sicherung' (PASS) * Band I: Datenreport Welle 10, *FDZ-Datenreport 07/2017*, Forschungsdatenzentrum der Bundesagentur für Arbeit im Institut für Arbeitsmarkt- und Berufsforschung, Nuremberg.

Bouchet-Valat, M. (2018). *SnowballC: Snowball Stemmers Based on the C 'libstemmer' UTF-8 Library*.
URL: *https://CRAN.R-project.org/package=SnowballC*

Bröcker, J. (2012). Probability forecasts, *in* I. T. Jollife and D. B. Stephenson (eds), *Forecast Verification: A Practitioner's Guide to Atmospheric Science, 2nd Edtion*, Wiley, Chichester, pp. 119–139.
URL: *http://dx.doi.org/10.1002/9781119960003.ch7*

Bröcker, J. and Smith, L. A. (2007). Increasing the reliability of reliability diagrams, *Weather and Forecasting* **22**(3): 651–661.
URL: *https://doi.org/10.1175/WAF993.1*

Chen, B.-C., Creecy, R. H. and Appel, M. V. (1993). Error control of automated industry and occupation coding, *Journal of Official Statistics* **9**(4): 729–745.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, ACM, New York, NY, USA, pp. 785–794.
URL: *http://doi.acm.org/10.1145/2939672.2939785*

Creecy, R. H., Masand, B. M., Smith, S. J. and Waltz, D. L. (1992). Trading mips and memory for knowledge engineering, *Commun. ACM* **35**(8): 48–64.
URL: *http://doi.acm.org/10.1145/135226.135228*

Drasch, K., Matthes, B., Munz, M., Paulus, W. and Valentin, M.-A. (2012). Arbeiten und Lernen im Wandel * Teil V: Die Codierung der offenen Angaben zur beruflichen Tätigkeit, Ausbildung und Branche, *FDZ-Methodenreport 04/2012*, Forschungsdatenzentrum der Bundesagentur für Arbeit im Institut für Arbeitsmarkt- und Berufsforschung, Nuremberg.

Elias, P., Birch, M. and Ellison, R. (2014). CASCOT International version 5, *User Guide*, Institute for Employment Research, University of Warwick, Coventry. (Available from http://www2.warwick.ac.uk/fac/soc/ier/software/cascot/internat/ .).

Federal Employment Agency (2019). Gesamtberufsliste der Bundesagentur für Arbeit (Stand: 03.01.2019), *Gesamtberufsliste_der_BA.xlsx*, Bundesagentur für Arbeit, Nuremberg. (Available from http://download-portal.arbeitsagentur.de/files/.).

Feinerer, I. and Hornik, K. (2018). *tm: Text Mining Package*.
URL: *https://CRAN.R-project.org/package=tm*

Friedman, J. (2001). Greedy function approximation: A gradient boosting machine, *Ann. Statist.* **29**(5): 1189–1232.
URL: *https://doi.org/10.1214/aos/1013203451*

Friedman, J., Hastie, T. and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent, *Journal of Statistical Software* **33**(1): 1–22.

Gillman, D. W. and Appel, M. V. (1994). Automated Coding Research at the Census Bureau, *Statistical Research Report Series 94/04*, United States Census Bureau, Suitland.
URL: *https://www.census.gov/srd/papers/pdf/rr94-4.pdf*

Gneiting, T., Balabdaoui, F. and Raftery, A. E. (2007). Probabilistic forecasts, calibration and sharpness, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **69**(2): 243–268.
URL: *http://dx.doi.org/10.1111/j.1467-9868.2007.00587.x*

Gweon, H., Schonlau, M., Kaczmirek, L., Blohm, M. and Steiner, S. (2017). Three Methods for Occupation Coding Based on Statistical Learning, *Journal of Official Statistics* **33**(1): 101–122.

Hall, A., Siefer, A. and Tiemann, M. (2015). BIBB/BAuA Employment Survey of the Working Population on Qualification and Working Conditions in Germany 2012. vt_1.0, sv_2.0, *Research Data Center at BIBB (ed., data access)*, Federal Institute for Vocational Education and Training, Bonn.
URL: *https://doi.org/10.7803/501.12.1.4.10*

Hartmann, J., Tschersich, N. and Schütz, G. (2012). Die Vercodung der offenen Angaben zur beruflichen Tätigkeit nach der Klassifikation der Berufe 2010 (KldB 2010) und nach der International Standard Classification of Occupations 2008 (ISCO08), *Technical report*, TNS Infratest Sozialforschung, Munich.
URL: *https://metadaten.bibb.de/download/684*

Hastie, T., Tibshirani, R. and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer Series in Statistics, 2 edn, Springer.

Hastie, T., Tibshirani, R. and Wainwright, M. (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations*, Chapman and Hall/CRC Press, Boca Raton.

Hoffmann, R., Lange, M., Butschalowsky, H., Houben, R., Schmich, P., Allen, J., Kuhnert, R., Schaffrath Rosario, A. and Gößwald, A. (2018). KiGGS Wave 2 cross-sectional study – participant acquisition, response rates and representativeness, *Journal of Health Monitoring* **3**(1): 78–91.

Hsu, W.-r. and Murphy, A. H. (1986). The attributes diagram: A geometrical framework for assessing the quality of probability forecasts, *International Journal of Forecasting* **2**(3): 285 – 293.

Ikudo, A., Lane, J., Staudt, J. and Weinberg, B. (2018). Occupational Classifications: A Machine Learning Approach, *Working Paper 24951*, National Bureau of Economic Research, Cambridge, MA.

Javed, F., Luo, Q., McNair, M., Jacob, F., Zhao, M. and Kang, T. S. (2015). Carotene: A Job Title Classification System for the Online Recruitment Domain, *Proceedings of the IEEE International Conference on Big Data*, Institute of Electrical and Electronics Engineers (IEEE), Redwood City, pp. pp. 286–293.

Jollife, I. T. and Stephenson, D. B. (2012). *Forecast Verification: A Practitioner's Guide to Atmospheric Science, 2nd Edtion*, Wiley, Chichester.
URL: *http://dx.doi.org/10.1002/9781119960003*

Jung, Y., Yoo, J., Myaeng, S.-H. and Han, D.-C. (2008). A Web-Based Automated System for Industry and Occupation Coding, *in* J. Bailey, D. Maier, K.-D. Schewe, B. Thalheim and X. Wang (eds), *Web Information Systems Engineering - WISE 2008*, Vol. 5175 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 443–457.
URL: *http://dx.doi.org/10.1007/978-3-540-85481-4_33*

Klingemann, H.-D. and Schönbach, K. (1984). Computerunterstützte Inhaltsanalyse als Instrument zur Vercodung offener Fragen in der Umfrageforschung, *in* H.-D. Klingemann (ed.), *Computerunterstüzte Inhaltsanalyse in der empirischen Sozialforschung*, Campus Verlag, Frankfurt/Main, pp. 227–278.

Knaus, R. (1987). Methods and Problems in Coding Natural Language Survey Data, *Journal of Official Statistics* **3**(1): 45–67.

Lange, C., Finger, J., Allen, J., Born, S., Hoebel, J., Kuhnert, R., Müters, S., Thelen, J., Schmich, P., Varga, M., von der Lippe, E., Wetzstein, M. and Ziese, T. (2017). Implementation of the European health interview survey (EHIS) into the German health update (GEDA), *Archives of Public Health* **75**(1): 40.
URL: *https://doi.org/10.1186/s13690-017-0208-6*

Lange, M., Hoffmann, R., Mauz, E., Houben, R., Gößwald, A., Schaffrath Rosario, A. and Kurth, B.-M. (2018). KiGGS Wave 2 longitudinal component – data collection design and developments in the numbers of participants in the KiGGS cohort, *Journal of Health Monitoring* **3**(1): 92–107.

Lyberg, L. and Andersson, R. (1983). Automated Coding at Statistics Sweden, *Proceedings of the Survey Research Methods Section: American Statistical Association*, pp. 41–50.

Mauz, E., Gößwald, A., Kamtsiuris, P., Hoffmann, R., Lange, M., Schenck, U. v., Allen, J., Butschalowsky, H., Frank, L., Hölling, H., Houben, R., Krause, L., Kuhnert, R., Lange, C., Stephan, M., Neuhauser, H., Christina, P.-M., Richter, A., Schaffrath Rosario, A., Schaarschmidt, J., Schlack, R., Schlaud, M., Schmich, P., Gina, S., Wetzstein, M., Ziese, T. and Kurth, B.-M. (2017). New data for action. Data collection for KiGGS Wave 2 has been completed, *Journal of Health Monitoring* **2**(S3): 2–27.

Measure, A. (2014). Automated coding of worker injury narratives, *Proceedings of the Government Statistics Section: American Statistical Association*, pp. 2124–2133.

Munz, M., Wenzig, K. and Bela, D. (2016). String coding in a generic framework, *in* H.-P. Blossfeld, J. von Maurice, M. Bayer and J. Skopek (eds), *Methodological Issues of Longitudinal Surveys: The Example of the National Educational Panel Study*, Springer Fachmedien Wiesbaden, Wiesbaden, pp. 709–726.
URL: *https://doi.org/10.1007/978-3-658-11994-2_39*

Murphy, A. H. and Epstein, E. S. (1967). Verification of probabilistic predictions: A brief review, *Journal of Applied Meteorology* **6**(5): 748–755.
URL: *https://doi.org/10.1175/1520-0450(1967)006¡0748:VOPPAB¿2.0.CO;2*

Nahoomi, N. (2018). *Automatically Coding Occupation Titles to a Standard Occupation Classification*, Master's thesis, University of Guelph, Guelph.
URL: *http://hdl.handle.net/10214/14251*

O'Hagan, A. and Forster, J. (2004). *Kendall' Advanced Theory of Statistics, Volume 2B: Bayesian Inference, 2nd Edition*, Arnold, London.

O'Reagon, R. T. (1972). Computer-assigned codes from verbal responses, *Commun. ACM* **15**(6): 455–459.

Ossiander, E. M. and Milham, S. (2006). A computer system for coding occupation, *American Journal of Industrial Medicine* **49**(10): 854–857.
URL: *https://onlinelibrary.wiley.com/doi/abs/10.1002/ajim.20355*

Potts, J. M. (2012). Basic concepts, *in* I. T. Jollife and D. B. Stephenson (eds), *Forecast Verification: A Practitioner's Guide to Atmospheric Science, 2nd Edtion*, Wiley, Chichester, pp. 11–29.
URL: *http://dx.doi.org/10.1002/9781119960003.ch2*

Riviere, P. (1997). Automated Coding - Foreword, *in* United Nations Statistical Commission and Economic Commission for Europe (ed.), *Statistical Data Editing Volume No. 2*, United Nations, New York.

Rohrbach-Schmidt, D. and Hall, A. (2013). BIBB/BAuA Employment Survey 2012, *BIBB-FDZ Data and Methodological Reports Nr. 1/2013. Version 4.1*, Federal Institute for Vocational Education and Training, Bonn.

Roulston, M. S. and Smith, L. A. (2002). Evaluating probabilistic forecasts using information theory, *Monthly Weather Review* **130**(6): 1653–1660.

Russ, D. E., Ho, K.-Y., Colt, J. S., Armenti, K. R., Baris, D., Chow, W.-H., Davis, F., Johnson, A., Purdue, M. P., Karagas, M. R., Schwartz, K., Schwenn, M., Silverman, D. T., Johnson, C. A. and Friesen, M. C. (2016). Computer-based coding of free-text job descriptions to efficiently identify occupations in epidemiological studies, *Occupational and Environmental Medicine* **73**(6): 417–424.
URL: *https://oem.bmj.com/content/73/6/417*

Russ, D. E., Ho, K.-Y., Johnson, C. A. and Friesen, M. C. (2014). Computer-based coding of occupation codes for epidemiological analyses, *2014 IEEE 27th International Symposium on Computer-Based Medical Systems*, pp. 347–350.

Sakshaug, J. W., Schmucker, A., Kreuter, F., Couper, M. P. and Singer, E. (2016). Evaluating active (opt-in) and passive (opt-out) consent bias in the transfer of federal contact data to a third-party survey agency, *Journal of Survey Statistics and Methodology* **4**(3): 382–416.
URL: *http://jssam.oxfordjournals.org/content/4/3/382.abstract*

Schierholz, M., Gensicke, M., Tschersich, N. and Kreuter, F. (2018). Occupation coding during the interview, *Journal of the Royal Statistical Society: Series A* **181**(2): 379–407.

Speizer, H. and Buckley, P. (1998). Automated coding of survey data, *in* M. P. Couper, R. P. Baker, J. Bethlehem, C. Z. F. Clark, J. Martin, W. L. Nicholls II and J. M. O'Reilly (eds), *Computer Assisted Survey Information Collection*, Wiley, New York, pp. pp. 223–243.

Statistisches Bundesamt (2016). *Demographische Standards*, Statistisches Bundesamt, Wiesbaden.

Takahashi, K., Takamura, H. and Okumura, M. (2005). Automatic Occupation Coding with Combination of Machine Learning and Hand-Crafted Rules, *in* T. B. Ho, D. Cheung and H. Liu (eds), *PAKDD 2005. LNCS, vol. 3518*, Springer, Berlin, pp. 269–279.

Takahashi, K., Taki, H., Tanabe, S. and Li, W. (2014). An Automatic Coding System with a Three-Grade Confidence Level Corresponding to the National/International Occupation and Industry Standard, *Proceedings of the International Conference on Knowledge Engineering and Ontology Development - Volume 1: KEOD, (IC3K 2014)*, INSTICC, SciTePress, pp. 369–375.

Thompson, M., Kornbau, M. E. and Vesely, J. (2014). Creating an automated industry and occupation coding process for the American Community Survey, *Background Material for a meeting of the Federal Economic Statistics Advisory Committee*, U.S. Census Bureau, Suitland. (Available from http://www.census.gov/about/adrm/fesac/meetings/june-13-2014-meeting.html.).

Trappmann, M., Beste, J., Bethmann, A. and Müller, G. (2013). The pass panel survey after six waves, *Journal for Labour Market Research* **46**(4): 275–281.
URL: *https://doi.org/10.1007/s12651-013-0150-1*

Wenzowski, M. (1988). Actr - a generalized automated coding system, *Survey Methodology* **14**(2): 299–307.

Westermark, M., Franzen, M. and Kraft, K. (2015). Automatic Coding of Occupation Using Spell Checking and Machine Learning, *30th JOS Anniversary Conference*.