Machine learning workflows to estimate class probabilities for precision cancer diagnostics on DNA methylation microarray data

Máté E. Maros[1,9], David Capper[6,10], David T. W. Jones[4,14], Volker Hovestadt[3,12,13], Andreas von Deimling[6,7], Stefan M. Pfister[4,5,8], Axel Benner[2], Manuela Zucknick[11] and Martin Sill[2,4,5]

1 Institute of Medical Biometry and Informatics (IMBI), University of Heidelberg, Heidelberg, Germany

2 Division of Biostatistics, German Cancer Research Center (DKFZ), Heidelberg, Germany

3 Division of Molecular Genetics, German Cancer Research Center (DKFZ), Heidelberg, Germany

4 Hopp Children's Cancer Center Heidelberg (KiTZ), Heidelberg

5 Division of Pediatric Neurooncology, German Cancer Consortium (DKTK), German CancerResearch Center (DKFZ), Heidelberg, Germany

6 German Cancer Consortium (DKTK), German Cancer Research Center (DKFZ), partner site Berlin, Germany

7 Department of Neuropathology, University Hospital Heidelberg, Heidelberg, Germany

8 Department of Pediatric Oncology, Hematology and Immunology, University Hospital Heidelberg, Heidelberg, Germany

9 Department of Neuroradiology, University Medical Center, Medical Faculty Mannheim of Heidelberg University, Mannheim, Germany

10 Charité — Universitätsmedizin Berlin, corporate member of Freie Universität Berlin, Humboldt-Universität zu Berlin, and Berlin Institute of Health, Department of Neuropathology, Berlin, Germany

11 Oslo Centre for Biostatistics and Epidemiology, Department of Biostatistics, Institute of Basic Medical Sciences, University of Oslo, Oslo, Norway

12 Department of Pathology and Center for Cancer Research, Massachusetts General Hospital and Harvard Medical School, Boston, Massachusetts, USA

13 Broad Institute of MIT and Harvard, Cambridge, Massachusetts, USA

14 Pediatric Glioma Research Group, German Cancer Consortium (DKTK) and German Cancer Research Center (DKFZ), Heidelberg, Germany

Authors' Email addresses and ORCID iDs:

M.E.M., maros@uni-heidelberg.de ; ORCID 0000-0002-1589-8699

D.C., david.capper@charite.de ; ORCID 0000-0003-1945-497X

D.T.W.J., david.jones@kitz-heidelberg.de ; ORCID 0000-0002-2036-5141

V.H., hovestad@broadinstitute.org ;

S.M.P., s.pfister@kitz-heidelberg.de ;

A.v.D., Andreas.vonDeimling@med.uni-heidelberg.de ;

A.B., benner@dkfz-heidelberg.de ; ORCID 0000-0002-7238-6956

M.Z., manuela.zucknick@medisin.uio.no ;

M.S., m.sill@kitz-heidelberg.de ; ORCID 0000-0001-7616-7665

To whom correspondence should be addressed:

Dr. Martin Sill

Im Neuenheimer Feld 580

69120 Heidelberg, Germany

Tel.: +49 6221 42 2719 ; Fax.: +49 6221 42 4639

m.sill@kitz-heidelberg.de

www.kitz-heidelberg.de

ORCID iD 0000-0001-7616-7665

KEYWORDS machine learning, classifier calibration, post-processing, multiclass probability estimates, DNA methylation, precision cancer diagnostics, random forest, elastic net, support vector machines, boosted trees

EDITORIAL SUMMARY This work compares several machine learning (ML) and calibration algorithms for classifying tumour DNA methylation profiles. The resulting protocol provides workflows for selecting, training and calibrating ML algorithms to generate well-calibrated multiclass probability estimates.

TWEET Comparison of machine learning (ML) and calibration algorithms for tumour classification from DNA methylation data and protocols for selecting, training, and calibrating ML workflows for well-calibrated multiclass probability estimates. @dtwjones @hovestadt @ManuelaZucknick @mwsill

COVER TEASER Machine Learning algorithms for highly multiclass tumour classification from DNA methylation data

Related links

Key reference using this protocol

Capper, D et al. Nature 555, 469–474 (2018) https://doi.org/10.1038/nature26000

Other publication focused on one particular method/workflow:

Capper, D. et al. (2018). 136(2), 181–210 (2018) https://doi.org/10.1007/s00401-018-1879-y

Sharma, T. et al. Acta Neuropathologica, 137, 1-18 (2019) https://doi.org/10.1007/s00401-019-02020-0

Abstract

DNA methylation data-based precision cancer diagnostics is emerging as the state-of-the-art for molecular tumor classification. Standards for choosing statistical methods with regard to well-calibrated probability estimates for these typically highly multiclass classification tasks are still lacking. To support this choice, we evaluated well-established machine learning (ML) classifiers including random forests (RF), elastic net (ELNET), support vector machines (SVM) and boosted trees in combination with post-processing algorithms and developed ML-workflows that allow for unbiased class probability estimation. Calibrators included ridge penalized multinomial logistic regression (MR) and Platt scaling by fitting logistic regression (LR) and Firth's penalized LR. We compared these workflows on a recently published brain tumor 450k DNA methylation cohort of 2801 samples with 91 diagnostic categories using a 5 × 5-fold nested cross-validation scheme and demonstrated their generalizability on external data from The Cancer Genome Atlas.

ELNET was the top stand-alone classifier with the best calibration profiles. The best overall two-stage workflow was MR-calibrated SVM with linear kernels closely followed by ridge-calibrated tuned RF. For calibration, MR was the most effective regardless of the primary classifier.

The protocols developed as a result of these comparisons provide valuable guidance on choosing ML-workflows and their tuning to generate well-calibrated class probability estimates for precision diagnostics using DNA methylation data. Computation times vary depending on the ML-algorithm from <15mins to 5d using multi-core desktop PCs. Detailed scripts in the open-source R language are freely available on GitHub, targeting users with intermediate experience in bioinformatics and statistics and using R with Bioconductor extensions.

Introduction

Supervised analysis of DNA methylation data for precision cancer diagnostics

Methylation data-based cancer diagnostics are currently emerging as state-of-the-art in oncology and molecular pathology[1-6]. However, there are no guidelines for choosing statistical methods to analyze high-throughput DNA methylation data with regard to well-calibrated probability estimates for highly multiclass and unbalanced classification problems [7-11]. Since the cancer methylome is a combination of both somatically acquired DNA methylation changes and characteristics reflecting the cell-of-origin[12], it is especially suitable for molecular classification of tumors and thus for stratifying cancer patients[4,13]. Therefore, DNA methylation has proven to be particularly suitable for individualized cancer diagnostics, prognosis, and treatment prediction[1,4,13,14].

Illumina Infinium Human Methylation BeadChip arrays are a popular tool to measure genome-wide single-nucleotide CpG site methylation levels[1]. Their readout is tens or hundreds of thousands of beta values that are the ratio of methylated- (m) to the sum of un- (u) and methylated probe intensities (m/m+u) [15,16]. Hence beta values are continuous parameters with values between 0 and 1 (representing the completely un- and fully methylated states of a CpG locus, respectively).

For stratified medicine it is paramount to correctly estimate the class probability (CP) of a case of interest with regard to a specific diagnosis[17-19]. Because CP serves as a confidence measure for the predicted disease status, it supports the treating physician to translate classifier outputs into robust diagnoses and optimal treatment modalities[11]. So, instead of just labelling the patient with a certain diagnosis, it is more appropriate to provide a numerical measure (probability) for that particular diagnosis [20]. Thus, the physician and the patient can make a more concise decision about treatment selection including its risks and benefits [20].

However, in high-dimensional (p >> n) settings of high-throughput genomic technologies, where the number of features (p) vastly outnumbers, the sample size (n), there is no hope for even asymptotic convergence of the CP estimates to true class probabilities[9,20]. Therefore, a more reasonable requirement is that the estimated class probability function (CPF) provides well calibrated predictions[20]. This means that if the CPF (developed on the basis of an arbitrary and limited training set) estimates the probability of future cases for class C of about P_C, then on average it should actually be about $100P_C$% of such cases in class C and this should be true for all P class estimates [20]. Well-calibrated CP estimates can be achieved through various model-updating/post-processing strategies like (re-)calibration [18,21]. In the machine learning (ML) sense re-calibration describes the mapping of a raw predictor output to the probability domain of [0; 1] so that all the probabilities sum to one or alternatively, re-calibration rescales the output of the predictor within the [0; 1] range[22-24]. A popular calibration algorithm is Platt scaling, which passes the raw classifier output through the sigmoid of logistic regression [24].

Although binary classification problems are extensively studied in the literature, multiclass problems are more complex and investigated to a substantially lesser extent[9,17,23,25-27]. Multiclass classification problems occur when the aim is to predict many different outcome categories, which is a typical situation in personalized diagnostics, where the number of diagnostic classes in which patients are stratified is very high. In the medical domain classification problems are often limited to two classes or to a selected few entities of interest [9,17,23,26]. Although there is no well-accepted threshold from which a task is regarded as highly multiclass in medicine, we consider comprehensive diagnostics systems that incorporate 50 or even more than 100 diagnoses as such [1,28]. A further potential statistical difficulty is presented by the fact that the number of cases in different diagnostic classes are highly variable according to common and rare entities [1,9,18]. Thus, while certain classes are well-represented others are sparsely populated resulting in unbalanced classification problems.

Random forests (RF), elastic net penalized multinomial logistic regression (ELNET) and support vector machines (SVM) are powerful classification algorithms for DNA methylation data[1,9,29-34]. Recently, boosting methods are having their renaissance in the ML community since multiple Kaggle competitions (https://www.kaggle.com ; https://en.wikipedia.org/wiki/Kaggle) have been won solely (or as part of the final model) by boosted trees (https://www.kaggle.com/c/otto-group-product-classification-challenge/discussion/14335 ; http://blog.kaggle.com/2017/02/27/allstate-claims-severity-competition-2nd-place-winners-interview-alexey-noskov/) [35-38]. There are multiple works regarding probabilistic calibration profiles of the aforementioned classifiers – mostly performed on binary classification problems[17,23,26,39-44]. However, there is a lack of studies on adaptation to highly multiclass and unbalanced data, and these adaptations need to be addressed when interrogating methylation data with the intent to diagnostically stratify human tumors.

The purpose of our study was to perform a benchmark analysis to support the choice for optimal DNA methylation microarray data analysis through extensive comparisons of well-established ML-classifiers such as RF, ELNET, SVM and boosted ensemble trees (**”Machine Learning Algorithms”**) and their combination with post-processing algorithms. The investigated post-processing algorithms (**”Calibration methods”**) were i) Platt scaling, implemented by fitting logistic regression (LR) or Firth’s penalized likelihood LR and ii) ridge penalized multinomial LR (MR) [9,24,45]. To provide valid performance estimates and practical guidance for hyperparameter settings all methods were implemented within a 5 × 5-fold nested cross-validation scheme using a primary brain tumor 450k DNA methylation data set published in Capper et al. 2018a [1]. This data set is uniquely large, with a total sample size of n=2,801 comprising k=91 diagnostic categories. Model fits were assessed with a comprehensive panel of performance metrics (**“Experimental design > Performance evaluation”**). A glossary of key terms is listed in **Table 1** [46].

The ML-workflows described here are generally applicable to any high-dimensional and multiclass data in biology and medicine[47] or any fields of science and technology, where well-calibrated probability forecasts for individualized diagnostics are of major interest. Such applications might include, but are not limited to, cancer diagnosis using gene expression analysis [9,48], radiological imaging-based tumor profiling (i.e. radiomics) [49] and particle physics [37,50] as well as materials sciences [51] or finance and marketing (e.g. credit risk assessment or ad clicks)[38]. If these presented ML-workflows are applied to data sets with features that are measured on various (and potentially vastly different) scales, special care should be taken for scaling and centering of the features, particularly for ELNET and SVM. On the other hand, tree-based algorithms like RF and XGBoost can cope automatically with scale differences. Further, this is less of a concern for methylation microarray data as beta values are confined to the [0, 1] range.

Analyses Undertaken

In **Fig. 1** we present the sequence of steps needed to develop and assess ML-workflows for high precision diagnostics from unprocessed Illumina DNA methylation array data (IDAT) to data preparation and pre-processing to downstream analyses (i.e. classifier development) including internal validation and calibration.

All methods were applied to 450K DNA methylation microarray data of the unique primary central nervous system (CNS) tumor reference cohort published in [1,2]. Hereinafter, we refer to this reference cohort as brain tumor methylation data (BTMD). BTMD consists of 2,801 biologically independent samples belonging to 91 (82 tumor and 9 non-tumor) methylation classes with concurrent extreme class imbalances (n_{min}=8 [0.3%], n_{max}=143 [5.1%]) [1]. Unprocessed IDAT files including the reference cohort (**Fig. 1**) are downloadable from the Gene Expression Omnibus under accession number GSE109381 [1]. Briefly, BTMD is based on genome-wide quantitative measurements of DNA methylation at 485,577 CpG sites using Infinium Human Methylation450 BeadChip technologies (Illumina, San Diego, US). The 450k BeadChip provides >98% coverage of reference sequence (RefSeq) genes and 96% of CpG islands [1,14] through the combination of two assay technologies (Infinium I and II). Beta value readouts by Illumina iScan array (IDAT) were obtained and preprocessed using the minfi Bioconductor R package[15] with additional filtering, i.e. removal of probes: targeting the X and Y chromosomes (n=11,551), containing single nucleotide polymorphisms (dbSNP132Common; n =7,998), probes not mapping uniquely to human reference genome 19 (hg19) allowing for one mismatch (n=3,965), and probes not included on the Illumina EPIC (850k) array (n=32,260), leaving 428,799 CpG probes for further analyses (**Fig. 1**, part 1; step 1) [1,52] – for details see the corresponding GitHub repository (https://github.com/mwsill/mnp_training).

Each sample was individually normalized by performing a background correction (**Fig. 1**, part 1; box I) as described in the reference article[1]. We performed the comparative analyses both without any batch effect correction (**Fig. 1**, part 1; box II, path of solid arrows) and with adjustments for batch effects caused by the type of tissue material (**Fig. 1**, part 1; box II, path of dashed arrows) – as published in [1,2]. Nonetheless, we report and focus on results solely from the non-batch adjusted analyses. However, if desired, batch effects caused by the type of tissue material from which tumor samples originated (formalin-fixed paraffin-embedded (FFPE) or freshly frozen) can be corrected for by fitting univariate linear models to the $\log_2$-transformed methylated and unmethylated intensity values using the limma package v3.24.15 [53] (**Fig. 1**, step 3).

To make analyses computationally tractable on multi-core CPUs, we performed feature selection using unsupervised variance filtering of the 10,000 most variably methylated probes [19,33,54]. Still, each supervised classifier had to fit over 2.5 billion (10^4 × 2801 × 91) data points. To prevent information leakage, this variance filtering was performed on the respective training set for each outer- and innerfold (altogether n=30 folds with foldIDs 1.0 – 5.5) while the corresponding test- or calibration sets were subset accordingly (**Fig. 1**, part 2; step 5, light blue rectangles). These variance filtered training-test set pairs were saved in separate .RData files and provided the foundation for all later comparative analyses. They can be generated through scripts on GitHub (https://github.com/mematt/ml4calibrated450k) or are readily available to directly download (~5.3Gb) from our Dropbox (http://bit.ly/2vBg8yc).

In order to robustly assess classifier and calibrator performance all algorithms were implemented within a 5 × 5-fold nested CV scheme (**Fig. 1**, part 2) similar to [1,17]. Due to class imbalances, fold assignments were performed in a stratified manner making sure that all classes are present in all (sub)folds (n_{min}=4-6). Nested CV was chosen because i) one can separate the classifier learning problem from the class probability calibration task [17], and ii) it more accurately estimates the external test error of the given algorithm on unseen datasets by averaging its performance metrics across folds [7,9,18]. We followed suggestions by [9,55,56] for choosing K=5 for K-fold nested CV as a good overall compromise between bias-variance trade-off to estimate prediction error and to limit computational burden. Furthermore, this setup allows for large enough calibration sets to limit overfitting and to yield robust tuning of post-processing algorithms [40]. In detail, classifier optimization during nested CV (**Fig. 1**, part 2; internal validation) is as follows: train (i.e. tune) the base ML-classifier on the respective outer or innerfold training set. This hyperparameter tuning involves an extra nested 3- or 5-fold CV loop to perform a grid search of suitable parameter settings, which is not shown on

Fig. 1 but is indicated in the description of the given ML-classifier (for details see tRF, ELNET, SVM and XGBoost in **'Machine Learning Algorithms'**). Then the tuned model object is used to predict the corresponding innerfold (calibration-) and/or the outer fold test sets (raw scores; **Fig. 1**, red rectangles and boxes). During post-processing the calibration models are trained on the raw classifier output scores on all combined calibration data sets (**Fig. 1**, inner CV loop, red rectangles S1.1-1.5), that is, the sum of inner fold test sets (**Fig. 1**, outer CV fold, blue boxes 1-4), which by virtue of nested CV design add up to the corresponding outerfold training set (**Fig. 1**, outer CV loop, blue boxes 1-4). Then the trained calibration model (**Fig. 1**, inner CV loop, green rectangle) is used to predict on the raw scores of the outerfold test set and produce calibrated probability estimates (**Fig. 1**, outer CV loop, green arrow & boxes P1.0-1.5) [1,2,17]. Thus, the calibration model only "sees" (predicts) the outerfold test set once, based on which final performance metrics are generated as an average over the predictions on the 5-folds.

The primary goal of this comparative study was to identify the best combination of ML- and calibrator algorithms by focusing particularly on downstream analyses of ML-workflow development and evaluation of their overall calibration profiles.

Application of the protocol to TCGA data. To show the general usability of our protocol beyond the BTMD, we applied the workflow presented in [1], to a 450K DNA methylation microarray data set from The Cancer Genome Atlas (TCGA) NCI GDC Legacy Archive (https://gdc-portal.nci.nih.gov/legacy-archive). First, we downloaded raw 450K data of 7,147 malignant tumor samples from 30 different TCGA projects (Supplementary Data 1). The data was normalized and preprocessed in the same way as described here and in [1] without applying any batch adjustment method (i.e. **Fig. 1**, part 1, solid arrow path). For BTMD the class label for each sample was provided by neuropathologists and other medical experts in cancer genomics. In contrast, the TCGA data set is (besides the project abbreviations for example BRCA for the breast cancer project) unlabeled. To generate methylation class labels for the TCGA data set, we performed a t-distributed stochastic neighbor embedding (t-SNE)[57,58] dimension reduction followed by DBSCAN clustering[59]. In brief, the data was reduced to the 32k CpG probes with highest standard deviation across samples, followed by a principal component analysis (PCA)[9]. The first 100 PCs were then used as input data for the t-SNE (https://github.com/mwsill/mnp_training/blob/master/tsne.R). On the resulting 2-dimensional t-SNE coordinates we applied the DBSCAN algorithm to identify clusters that we then used as methylation class labels[59]. DBSCAN is a density-based clustering algorithm that tries to estimate the number of clusters and to label samples that do not fit in any cluster as outliers. With the DBSCAN we identified 46 clusters and 344 samples were identified as outliers and removed from further analysis. The clusters identified by DBSCAN showed good overlap with the different TCGA projects (**Fig. 2a, b**) and some of the clusters might be potential candidates for new molecular subtypes, for example cluster 40 and 41 appear to be two distinct subtypes of uveal melanomas. However, the biological interpretation of these purely data-driven methylation classes is out of the scope of this work. We used the 46 clusters to label our data and then train a classifier following the workflow presented in [1] . Overall, the 5 × 5-fold nested CV estimated a misclassification error of 6.7% and a BS of 0.099 indicating a good prediction performance only slightly worse than for the BTMD. Thus, we conclude that the all other ML-workflows presented here will show a comparable performance on the TCGA data. The ten best performing workflows w.r.t. to a minimal BS highlighted in Table 2 are therefore all good candidates, to train a diagnostic classifier on methylation data for highly multiclass classification problems, as typically encountered in personalized medicine.

The presented procedures can be used to either tune and evaluate the performance of a single ML-classifier or calibrated workflow, or to compare all the presented ML- workflows and choose the best performing one for the respective scenario case of the user. We also show performance improvement compared to the method presented in Ref. 1.

Machine Learning Algorithms

Random forests

Random forests (RF) are an ensemble method of bootstrap aggregated (bagged) binary classification trees [29]. RF grows binary classification trees based on bootstrapped samples of the training data while using only a random subset of available features at each node to find the optimal splitting rule [9,29,31,60]. Through repeating these processes RF can generate thousands of decorrelated decision trees (i.e. the ensemble) that can provide more robust committee-type decisions. For each case passed through the classifier, the majority vote over all trees generates the final class label. RF for classification tasks tend to be deep as they are grown to purity – leaving only one case in the terminal nodes[29].

We used the randomForest package in R [61] with default settings for classification: `ntree` = 500, `mtry` = $\sqrt{p}$ (i.e. optimal random subset of p features during node splitting) and minimal size of terminal nodes = 1. All RF implementations were based on two runs of RF: first, CpG probes were ranked according to their importance using permutation-based mean decrease in accuracy that measures the average difference over all trees of the out-of-bag sample error before and after permuting predictor variables [29,61]; second, a RF with the most important p_{varsel} (p) CpG probes was used for prediction [1,2,6]. Hereinafter, we refer to the default RF implementation with p = 200 as 'vanilla' RF (vRF).

For tuned RF (tRF) we optimized parameters including `ntree`, `mtry`, and terminal `nodesize`. Optimal settings were found using 5-fold CV on a custom grid within the framework of the caret package [62]. Although `ntree` = [500; 1000; 1500; 2000] is often not considered a real tuning parameter, it has to be big enough (~500) for error estimates to stabilize [9,18,31].

Because of bootstrapping, we are not prone to overfitting if `ntree` is set to be large (>1000), but computation times might increase considerably[29,31]. For the primary tuning parameter `mtry` , we chose values of 1%, 5% and in 10% steps up to 120% $\sqrt{p}$ [61,62]. Initial testing showed, however, that the search grid could be narrowed down to $\sqrt{p}$ ±(10%; 20%). Terminal node sizes including the default for classification (1), regression (5), and 1% and 10% of n were tested [23,26,29,61,63,64]. For tRF the parameter p_{varsel} = [100; 200; 500; 1000; 2000; 5000; 7500; 10000] (i.e the number of most important CpG probes used for model fitting or prediction) was also tuned with regard to ME, BS, and LL (see **'Performance evaluation'** in **'Experimental design'**). The importance ranking of CpG probes was based on the type 1 variable importance measure, that is, the mean decrease in accuracy computed on the out-of-bag data by RF[61]. It is of note, that the type 1 variable importance is proved to be biased and dependent on `mtry` [65] so that correlations between predictors can distort variable importance rankings[65].

In all RF implementations we downsampled to the minority class (training subfolds n_{min}=4-6) to counteract severe class imbalances [66]. The two-stage workflow of RF in combination with ridge-penalized multinomial logistic regression represents the same methodology published in [1,2], although, here we switched from the 3 x 3 to a more robust 5 x 5-fold nested CV scheme as we performed only internal validation [55,56].

Elastic net penalized multinomial logistic regression

The glmnet package was used to fit and tune elastic net penalized multinomial logistic regression (ELNET) as a stand-alone method [67]. Using ELNET without post-processing was justified by its well-known applicability to high-dimensional genomic microarray data and based on findings that the sigmoid of Platt scaling is less effective on logistic regression [9,33,40,41,68]. Optimal mixing parameter (α) and penalty strength (λ) of the L1 (lasso) and L2 (ridge) terms were found by concurrent CV of their two-dimensional parameter space [34,67-70]. The cv.glmnet function utilizes a warm start (through an additional first run) that sets up the λ-space for coordinate descent more precisely than other grid search-based methods[67,69]. To exploit this favorable property, we implemented a custom function – as suggested by the glmnet package authors [67,69]. We used fixed fold assignments of balanced stratified 5x CV reassuring that the results of different α were comparable with each other. First, a grid of α = [0; 0.1; … ; 0.9; 1] was tested, which was then fine-tuned in the [0; 0.025; 0.05; 0.075; 0.1] range – as exclusively α=0 (ridge) type settings were selected on the first grid. We used MSE as the loss during CV. The number of λ and ratio of λ_{min} were left at default at 100 and 10^{-6} respectively [67,69]. Probability estimates were generated at λ_{1SE} to improve their robustness [60]. An important technical note is that `standardize` was left at default (`TRUE`), thus each (sub)fold was scaled and centered, although this might not be necessary as CpG methylation beta values are confined to the [0,1] range [67].

Support vector machines

Support vector machines (SVM) were implemented using linear- and radial basis function kernels (RBF)[30]. Linear kernels (LK) have a single tuning parameter C that is the cost parameter of the error term, while RBFs have an additional hyperparameter that defines the variance of the Gaussian, i.e. how far a single training example's radius of influence reaches [9,30].

We investigated multiple R packages with SVM running on both GPUs and/or on CPUs. Due to performance discrepancies we ended up fully evaluating the e1071 [71,72] and LiblineaR [73,74] packages on CPU; and the GPU-accelerated Rgtsvm package[75]. It is of note that both e1071 and LiblineaR use the 1-vs-1 extension to generate probability estimates for multiclass tasks[42,72,73]. In contrast, Rgtsvm uses the 1-vs-all method proposed by Crammer and Singer to obtain class labels for multiclass tasks [76] while it uses another framework with 1-vs-all coupling strategy along with a global, log loss-optimized softmax [77] to calculate probabilities. RBF were tuned on $C = 2^{-3:3}$ and $= 2^{-5:5}$ grid using 5-fold CV on the prototyping subfold 1.1. LK can provide similar accuracy to RBF but at lower computational costs, especially for $p > n$ tasks when mapping data points to a higher dimensional space becomes unnecessary[9,78,79]. For LK we explored the parameter space of $C = 10^{-3:3}$ with 5x CV using both e1071 and LiblineaR packages. The LiblineaR package has the advantage that it provides 8 types of linear and logistic regression and support vector classification (SVC) models[73] . Hence, one can concurrently optimize both model type and C. Both the training set and its respective test set of a given (sub)fold were scaled and centered using the corresponding training attributes before tuning or prediction [72,79]. We investigated the effect of weighting with inverse class frequencies to compensate for class imbalances [72].

Boosted decision trees

Boosted decision trees are fundamentally different from bagged trees (like RF): i) boosting grows shallow trees, ii) boosted trees are dependent on previous steps, whereas RF trees are identically distributed, iii) boosted trees are prone to overfitting if `nrounds/ntree` is large (>500), whereas bagged trees are not [9,31]. Additionally, the type 1 variable importance measure used in RF for feature selection is prone to be biased because of correlations between predictors. Such correlations can distort variable importance rankings so that otherwise irrelevant features with high correlations to informative predictors get disproportionately large importance values [18,65] or conversely, if there are many truly relevant predictors but they are highly correlated, their importance measures get diluted and will be decreased[18]. In theory boosting is more resistant to dilution than RF as boosted trees additively learn from previous steps and focus their learning on areas, which has not been well modeled up to that point [9,18,31,65]

For the implementation we chose the currently popular extreme gradient boosting (XGBoost) package in R [80] . Softmax was the objective function for multiclass classification. The evaluation metric was misclassification error to ensure more direct comparability with other methods, although we also tried multiclass log loss, but it yielded worse results during training. We used tree-based boosters that translated to an ensemble of trees [80]. In the XGBoost formulation the main difference between an RF and boosted trees is how are they trained [81] . To our knowledge, there are no recommendations in the literature on how to optimally initiate hyperparameters for XGBoost when fitting high-throughput methylation array data. Therefore, we performed an extensive optimization of XGBoost's tuning parameters (see also **Table 3**) including: `nrounds`, the number of iterations (that is equivalent to ntree of RF); `max_depth`, maximum depth of a tree; `eta`, learning rate; `gamma`, minimum loss reduction required to make a further tree partition; `colsample_bytree`, subsample ratio of columns when constructing each tree; `min_child_weight`, minimum sum of instance weight (hessian) needed in a child; and `subsample`, subsample ratio of the training instance (0.632 is the special case to mimic RF's bootstrapping) [80,81]. Hyperparameters of the best performing top 4 models on the prototyping subfold 1.1 (**Table 4**) were used to fit the complete data set with `nrounds` (=100), `min_child_weight` (=1), `subsample` (=1) left at default. For this, an extra nested 3-fold CV grid search within each training loop was performed using the framework of the *caret* package [62] to find optimal hyperparameters (**Table 5**). Then, the `xgb.train()` function was refitted on the training data with those tuned settings to exploit its `watchlist` functionality and to find the optimal number of `nrounds` iterations[80,81]. Finally, these settings were applied to the respective test/calibration set to generate raw probability estimates (i.e. raw scores).

Calibration methods

Platt scaling

Platt suggested this parametric method originally for binary classification problems when mapping SVM raw scores to posterior probabilities[24,40,44]. Nonetheless, any classifier's output can be post-processed with Platt scaling. Briefly, Platt's main idea was to pass raw SVM estimates through a sigmoid function[24]:

$$P(y = 1|f) = \frac{1}{1+\exp(Af+B)} \qquad (1)$$

, where f(x) denotes the output of the predictor algorithm, and the parameters A and B are fit using maximum likelihood estimation from a training set (f_i , y_i) , where $y_i \in \{-1 , +1\}$ for mutually exclusive binary classes [24]. Platt then defined a new training set (f_i , t_i), where $t_i = \frac{y_i + 1}{2}$, which is generated ideally using CV. In this new space A and B are found so that they minimize the negative log likelihood (for details see [24,40]). To avoid overfitting on the training set Platt added some regularization by changing t_i from $\{0 , 1\}$ to their maximum *a posteriori* estimates using Bayes' rule [24]. Thus, the log loss cannot be 0 even if we had completely correct probability estimates of 0 and 1 for all examples [40]. This regularization becomes increasingly relevant when there are separable classes with few samples. To apply Platt's method, multiclass problems need to be reduced to a series of binary calibration tasks and then recombined to obtain multiclass probabilities [40,44]. Two well-known binary reduction approaches are the 1-vs-1 and 1-vs-all [24,27,40,42,44]. We used the latter approach and simply normalized all obtained probabilities such that they sum up to 1 [9,18,42,69]. Notably this might occasionally lead to changing of the most probable classes (see **Table 2**).

Logistic regression (LR). A fairly straightforward implementation of Platt scaling is post-processing with logistic regression using the `glm` function of base R [82,83] (see also http://danielnee.com/tag/platt-scaling/). LR maps the raw predictor to a probability for the true binary outcome [18]. We iterate through each k class and combine the results. We should point out, however, that this approach does not explicitly incorporate Platt's regularization step of t_i [24]. Due to class imbalances complete or quasi-complete separation of datapoints (i.e. classes) can occur[24,82] thereby motivating the switch to methods that replace the maximum likelihood estimate (MLE) with penalized estimates [45].

Firth's penalized logistic regression (FLR). Firth proposed a solution [45] for the problem of separation in regular models. He aimed to remove the first order bias term for which he proposed a penalized likelihood function using the Jeffreys invariant prior [45]. Thus, Firth's penalized likelihood is second-order unbiased and the resulting estimates and standard errors are always finite [84,85]. Although this method removes bias at the coefficient level, it concurrently results in biased event probabilities [86]. Unfortunately, Firth-estimates (similar to MLEs) are not unique for p >> n [84-86]. We implemented Firth regression using the `brglm` function of the identically named R package [87]. The number of maximum iterations were varied from the default 500 up to 10,000.

Ridge penalized multinomial logistic regression

Ridge regression is perhaps the most widely used shrinkage method [31]. It is particularly suitable for p >> n problems and in case of multicollinearity to stabilize the estimates of LR [9,34,67,68,88]. We used the glmnet R package [67], which readily offers the multinomial extension of the binomial ridge (L2) penalized LR for multiclass outcome variables [67,69]. For ridge calibration the cv.glmnet function was fit with mixing parameter $\alpha = 0$ [67,69]. We applied the default function settings of 10-fold CV for finding λ_{min} on the inner fold test sets (calibration set). The large size of BTMD allowed a sufficiently large calibration set (n > 2,000) to stabilize tuning parameters and estimates [17,40,44]. Ridge regression can also be interpreted from the Bayesian perspective, as it represents an increased prior belief that beta coefficients are close to 0 by imposing a normal prior [9,31]. Ridging shrinks the estimates towards each other and 0, thus introducing biased MLEs of $\hat{\beta}$ without performing any feature selection [9]. These properties make ridge regression, intuitively, suitable for being a probability calibrator as it does not dismiss any features during post-processing [7,9,67].

Overview and variations of the proposed procedure

We present a pipeline and affiliated scripts to perform each analysis step in **Fig. 1**. The pipeline is highly modular and enables an entry at any step. However, as our main focus is on describing the internal validation process and comparing the results of the investigated ML-classifiers and calibrators, we recommend starting from step 6 when trying to replicate these ML-workflows using the referenced data set (BTMD) or trying these algorithms on their own data sets (**Fig. 1**, part 2). The provided framework allows the user to flexibly plug-in (steps 8-10) and explore any other predictor algorithm suitable for high-dimensional data analyses such as nearest shrunken centroids, k-nearest neighbours (k-NN), and various neural networks architectures like multi-layer perceptrons (MLP) [9] or other methods based on simple sign averaging [89] or eigenvalue shrinkage [90].

Similarly, other post-processing algorithms can be inserted in steps 11-14 including isotonic regression, naïve Bayes estimates or variations of local error frequencies [17,22,41]. Alternative methods for calibrating RF were proposed by Boström [22] and more recently by Dankowski and Ziegler[21]. Although both of these methods were developed explicitly on RF, they can be applied to any other classifier as well[21,22].

Although the focus of this pipeline is on 450K methylation array data-based tumor classification it can be applied to the newest generation of Illumina EPIC (850K) array or any other high-dimensional, multiclass data set. However, in the latter case additional care should be taken to scale and center the features (especially if they are measured on different scales). Such preprocessing is standardization that centers the respective feature around 0 using its mean and scales it with respect to its standard deviation. This is less of a concern for methylation data as beta values are measured on the same scale and confined to the [0, 1] range.

To account for massive class imbalances in the cohort, we used down-sampling to the minority class (n_k=8, for all 91 classes) similar to the reference paper [1], however, only for RF implementations. For SVM the effect of weighting with the inverse class frequency ($1/n_k$) was investigated [9,30,72,79]. The remaining classifiers were fitted directly on the nested CV data, which was generated by stratified sampling to ensure that all classes are present in each CV (sub)fold. A possible extension of this protocol would be to test whether more complex resampling strategies for imbalanced datasets, such as random under- or over-sampling, would improve classifier or workflow performance. Unfortunately, there are no ready-to-use implementations for multiclass problems[91], as most available resampling packages primarily offer strategies for binary classification tasks only [49,62,92-95].

Comparison with other methods

In this protocol there are some important changes compared to the approach described in [1]. In this comparative analysis, we use a benchmarking data set that was normalized but not adjusted for possible batch effects between samples stemming from FFPE or freshly frozen material. In [1] the batch effect adjustment was included into the 3 x 3 cross-validation scheme to estimate prediction performance when the batch adjustment of the test data is based on batch effects that were estimated on the training data. However, as the primary goal of this study is to compare the performance of different ML-workflows, we performed all analyses on a normalized but not batch adjusted feature space. Additionally, our preliminary studies showed that the influence of batch adjustment on the overall performance of the ML-workflows is negligible and it affects all workflows in the same way.

Second, for feature selection in step 5 (**Fig. 1**), we use an unsupervised variance-based pre-filtering to select the 10k most variable CpGs [33,54]. Other feature selection and dimensionality reduction methods that limit the feature space either independently or in combination with a supervised ML-algorithm [96] would be also suitable. These might include: basic filtering, controlling for false positive selections, correlation filters, wrapper methods (e.g. greedy forward selection), embedded methods (e.g. the investigated ML-algorithms that provide feature importance like RF[1,2], boosting, regularized LR or SVM), the combination of embedded and wrapper methods (e.g. SVM with recursive feature elimination[9,97,98]), feature construction methods (e.g. sample or feature clustering, principal component analysis (PCA) projections) – for details see [96] and references therein. To note, that the applied unsupervised pre-filtering to the relatively large number of 10k most variable CpGs is quite uninformative[33,54]. Therefore, when there is an interest to use one of the alternative methods mentioned in this paragraph for more stringent pre-filtering, then the feature selection should be included in the validation via additional internal validation loops[96]. On top of that, some of these dimensionality reduction methods can be computationally very expensive especially on a feature space as large as 450k or 850k CpGs produced by methylation microarrays. Hence, we did not study the impact of the number of selected features and the choice of pre-filtering methods on prediction performance in this manuscript; for comparison studies focusing on these aspects please refer to [9,33,54,96].

Third, we performed no additional threshold analysis, that is, for all classifiers or calibrated workflows the final predicted class for each case was the one with the highest assigned score or probability estimate among the 91 diagnoses. Finding an optimal, common threshold for all classes at a desired sensitivity and specificity depends on the practical application of the resulting classifiers but is not important for the benchmarking of ML-workflows, as here threshold independent metrics like the AUC, Brier-score and log loss should be used for evaluation. For example, if the resulting calibrated classifier were to be deployed as a diagnostic tool in a clinical setting, thresholds that result in high specificities, that is low type one error rates, are preferred. However, when the classifier is applied as a research tool a different threshold might be more suitable.

Finally, for pre-processing and normalization of the raw data we apply functions and data classes available in the Bioconductor[99] package minfi [15]. Alternative software packages that provide comparable functionalities are the Bioconductor packages RnBeads [100], ChAMP [101] and wateRmelon [102]. All mentioned packages provide different data classes to store the data, but often share implementations of the same popular normalization functions developed for Illumina methylation array data.

Experimental design

In order to train a classifier, a suitable training data set needs to be established. As for any ML-project this involves expert knowledge to define classes and assign class labels to samples. For example, to generate the training data set presented in [1] we used several preceding studies that defined new methylation-based tumor classes by performing unsupervised methylation data analysis and describing the molecular and clinical differences of these classes [5,14,52]. Statistical considerations when planning to train a classifier for diagnostics using genomic data include the high-dimensionality of the data, the large number of classes as well as the heavily imbalanced class sample sizes.

To reduce dimensionality and computation time, we performed an unsupervised variance filtering that was implemented into the nested CV (**Fig. 1**, step 5) to prevent information leakage. Furthermore, like the RF presented in [1] all ML-workflows shown in this study are based on methods developed for high-dimensional settings and often include a feature selection step or perform automated feature selection, like the lasso penalized regression model, to further reduce dimensionality.

The minimal class sample size for BTMD is eight, which is critical and a minimal class sample size of ten or even more might be desired[67,69]. Otherwise class sizes might become so small during the 5 x 5 nested CV that the calculations for ML-classifier training cannot be carried out or their estimates become highly unstable. However, we did not observe any substantial misclassifications with the smaller than recommended class size and the minimal inclusion sample size of eight allowed us to include several rare tumor classes of special interest for neuropathologists that otherwise would have been excluded[1,2].

Moreover, the class sample size will be further reduced in the training folds of the CV. This is an additional consideration when planning the CV and setting the number of folds. For example, with a higher number of CV folds, the computational burden increases but also the minimal sample size in the training folds is larger [9,17,18,55,56,92]. In addition, when generating CV folds, the sampling needs to be done in a stratified manner to guarantee balanced minimal class sample sizes in all training folds [18,62,92].

Tree-based algorithms like the RF are known to suffer from heavily imbalanced class sample sizes, which are inherently present in BTMD. To deal with this problem several possible strategies have been proposed [66]. In [1] and also in this study we deal with this problem by downsampling all classes to the minority class for each tree in the RF[18,61]. Therefore, each tree is trained on a relatively small balanced bootstrap data set that has the additional advantage to greatly improve the computation time to fit a single tree. Additionally, we assessed inversed class frequency weighting for SVM, that is, classes are weighted inversely proportional to their distributions [72]. However, this can result in artificially distorted class weights, as most data sets represent a convenience cohort that do not necessarily resemble the true underlying population distributions[20]. The algorithms ELNET and XGboost do not benefit from additional data balancing strategies other than the previously mentioned stratified resampling.

Confounding factors. A typical confounding factor present in Illumina methylation array data is the source material the sample DNA originates from. For example, DNA can be extracted from FFPE as well as from freshly frozen material, and different extraction protocols are applied for each sample type. The type of source material can be easily determined by reading out the restoration control probes available for each sample on the array. In [1] the methylation data were adjusted for the differences between FFPE and freshly frozen material by applying a linear model approach available in the Bioconductor package limma [53] that was performed independently within each CV-loop. Another possible confounding factor that might affect methylation data is the age of the patient, as it is known that methylation is associated with aging and that the age of human tissue may even be predicted by methylation data [103]. However, as the brain tumors in BTMD belong to adult as well as pediatric brain tumor classes, age is expected to be strongly associated with these classes and thus we did not adjust for this possible confounder. DNA-samples measured by methylation arrays are usually bulk samples comprising a mixture of tumor cells, infiltrating immune cells and other stromal components[104]. Thus the proportion of tumor cells in the sample, which is known as the tumor purity may also influence the DNA methylation and as already shown in [1] methylation data can be used to measure tumor purity. Like age, however, the tumor classes defined in the BTMD data set are associated with tumor purity, with some tumors known to have much higher tumor purity compared to others, and thus we did not adjust for purity. Finally, after adjusting for the FFPE/fresh frozen batch effect in the reference paper, BTMD was tested with the sva algorithm [105-107] for additional confounding batch effects and we were not able to detect any significant surrogate variables.

Software setup. All computation steps in **Fig. 1** can be carried out in the open source R statistical programming environment (R Foundation for Statistical Computing, Vienna, Austria, https://www.r-project.org/) within the recommended RStudio integrated development environment (RStudio, Inc., Boston, MA, http://www.rstudio.com/) [83]. R has a wide variety of packages including a dedicated bioinformatics analysis suite Bioconductor for orchestrating high-throughput genomic analyses [108,109]. Besides the optional GPU-accelerated variant of support vector machines (Rgtsvm) and xgboost, all presented ML-classifier (steps 8-10), post-processing algorithms (steps 11-14) can be run solely on multi-core CPUs without the need to install software outside the scope of R [75,80]. To speed up computations, we used base R's (>v2.14.0) built-in high-performance parallel computing package parallel, which incorporates multiple other packages like multicore, snow and foreach – the latter of which is used by glmnet to speed-up hyperparameter tuning ('**Machine Learning Algorithms'**). We prefer using a single software platform, which provides the advantage of a simplified workflow and maintenance [108]. Some users might prefer the general-purpose programming language Python over R – a popular language for software development, prototyping and scientific computing – that also provides a wide variety of tools for ML via the scikit-learn library [110]. However, Python requires interface packages to provide some Bioconductor functionalities. Nonetheless, downstream analyses from step 5 could easily be implemented in Python, if desired.

In the PROCEDURE section we present the steps needed to perform hyperparameter tuning for the RF classifier including its calibration with MR (i.e. tRF_{BS} + MR and tRF_{ME} + MR and tRF_{LL} + MR) and its final performance evaluation. Because the respective R package for each investigated ML-classifier algorithm has different built-in functionalities our R scripts follow a 3-layered approach to carry out the internal validation process (steps 8-10): i) subfunctions are invoked to extract optimal hyperparameter settings from the output object of the respective predictor algorithm (e.g. vRF, ELNET, SVM) or from the ML-framework of the caret package (e.g. tRF) or both (e.g. XGBoost); ii) the train function (e.g. `trainRF_caret_custom_tuner`) performs hyperparameter tuning using the corresponding subfunctions; iii) finally, the train function is implemented within the nested cross-validation scheme (e.g. `run_nestedcv_tunedRF`) and consequently dedicated calibrator (steps 11-14) and performance evaluator functions (step 15) might be applied separately to its output.

Performance evaluation. We combined a comprehensive panel of numerical performance metrics to assess model fits.

Misclassification error (ME) is defined as the proportion of incorrectly classified cases over all classes divided by the total number of cases. In medical applications the lowest achievable misclassification error is preferable. For each classifier the provided ME is the average of the errors measured in each of the 5-fold CV (outer) test sets.

While the ME is measured when using the maximum classification score as threshold to determine the predicted class, the area under the receiver operating characteristics curve (AUC) provides a way to compare the separability of a classification rule for all possible thresholds [111]. Hand and Till proposed a generalization of the AUC for multiclass classification problems by extending its probabilistic form using a 1-vs-1 approach over all k classes[111]. Their multiclass AUC measure is available in the R package HandTill2001[112].

Brier score (BS) is a proper scoring rule that measures the accuracy of probabilistic predictions of mutually exclusive classes [113-116]. Originally, Brier proposed this method in 1950 to assess weather forecasts in terms of probability[114]. His formulation (2) is applicable to multiclass forecasts and is defined as the quadratic difference between the assigned probability and the value (1, 0) for the class [92,114]:

(2)

$$BS = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K}(f_{i,k} - o_{i,k})^2$$

, where f_i is the predicted probability and o_i is the actual outcome of binary coded *i* (0 if happened, 1 if not). In our benchmarking dataset *n* denotes the number of samples (2,801) and *K* the number of diagnostic classes (91). For binary *K* Equation (2) gives back the mean squared error of the prediction, while for multiclass problems it is the sum over all 1-vs-all comparisons [92]. Our target is to minimize BS thereby indicating better calibrated predictions. Cross-entropy loss or log loss (LL) is extensively used to assess probability estimates of predictor models instead of just focusing on their discrete label assignments [18,113,115]. The advantage of the logarithmic scoring rule over BS is that it is local strictly proper [113]. We used a multiclass extension of log loss (LL):

$$log\ loss = -\log Pr(Y|P) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} y_{i,k}\log(p_{i,k}) \tag{3}$$

, here n and *K* denote the number of samples and classes, respectively; log is the natural logarithm, $y_{i,k}$ is the binary true outcome 1 if sample *i* is in class *k* and 0 otherwise, and $p_{i,k}$ is the predicted probability that observation *i* belongs to class *k*. Log loss does not explicitly require that predicted probabilities add up to one, however a simple division by the row sum is recommended. Initial predictor model outputs (raw scores) can theoretically lie anywhere $[-\infty, +\infty]$. Thus, before calculating LL, we constrained extremely marginal predicted raw scores or calibrated probabilities (close to 0 or 1) to max (min $(p\,, 1 - 10^{-15}),\ 10^{-15})$. Similarly to BS, the objective is to minimize LL. Both BS and LL were used as loss functions for optimizing feature space (p_{varsel}) during ML-algorithm tuning (e.g. tRF; **Table 2**) as well as evaluation metrics of overall calibration.

BS encourages predicted and true probabilities to lie close to each other, whereas LL does not [115]. Extensive empirical testing, however, stressed LL's favorable local property that it will always assign a higher score to a higher probability estimate for the correct class. In contrast, BS can perform poorly in this regard [113]. R scripts are provided to calculate all performance metrics in the GitHub repository.

Expertise needed to implement the protocol

The presented scripts in the open-source R language target users with intermediate experience in bioinformatics and statistics, especially using ML algorithms and using R with Bioconductor extensions. Valuable introductory material for ML can be find in the following book [117] and free online course taught by two (of the book's co-authors, who are) world-renowned//highly distinguished Stanford University Professors of the field (https://lagunita.stanford.edu/courses/HumanitiesSciences/StatLearning/Winter2016/about).

Sample preparation

DNA methylation data of the CNS tumour reference cohort used in this pipeline were generated from formalin-fixed paraffin-embedded (FFPE) or freshly frozen tissue samples at the Genomics and Proteomics Core Facility of the DKFZ (Heidelberg, Germany) using Illumina Infinium HumanMethylation 450K Bead Chip (450k) arrays according to the manufacturer's instructions (Illumina, Inc.). For further details please see [1,2].

Limitations

Possible limitations of the proposed protocol to train a DNA methylation-based classifier for tumor class predictions are, as already mentioned previously, the highly imbalanced class sizes and low minimal class sample size in BTMD. Even though we did not observe any abnormalities like biased predictions towards tumor classes with larger sample size, these limitations can have an impact on the classification performance and for future training sets it might be worthwhile to increase the minimal class sample size as well as to try balancing class sample sizes. All presented classifiers use an intersection set of CpG probes available on both the 450k and the EPIC human methylation array and are thus able to classify samples derived by both array technologies. However, as BTMD is solely comprised of 450k methylation samples, these classifiers might still be biased towards the 450k, i.e. by generating higher raw scores and calibrated probabilities for 450k array samples. To deal with this problem in the future, new training data sets should also incorporate EPIC methylation array samples, to allow assessing possible batch effects between these two array technologies as well as to estimate the impact of incorporating two different arrays on the classifier performance by cross-validation. This bias toward a specific platform is also an important consideration when deploying a classifier for practical applications[1,2]. For example, when using the classifier presented in the reference paper that is freely available on the website www.molecularneuropathology.org, users may upload EPIC and 450k samples, generated in different labs under different conditions with varying sample quality and thus we cannot expect the same prediction performance that we observed in our cross-validations. Due to the highly multiclass nature and relatively large size of BTMD associated with extended run-times, we did not systematically compare the performance of the investigated ML-classifiers and calibrated workflows with respect to the number of available CpG features. Although, RF (p_{varsel}) and XGboost (`colsample_bytree`) were tuned with respect to the number of CpG probes and the performance of ELNET was also assessed on the 1000 most variable probes, SVM were applied to the 10k feature set only. This arbitrary but deliberate reduction of the feature space (balancing computational effort and feature space size) might limit and more negatively influence some classifiers than others. SVM were reported to be sensitive to the size of the feature space in p>>n multiclass gene expression studies[9,48]. As the number of genes is reduced, SVM classifiers respond by degrading accuracies or even suddenly collapse with poor overall performance [9,48] . Our results show that the investigated ML-classifiers performed in a similar range on the 10k feature space: tree-based algorithms (tRF$_{BS | LL}$ and boosting) chose comparably sized smaller feature subsets of 100-500 CpGs while tRF$_{ME}$ and ELNET selected larger (1000-10,000 CpG) or consequently the full 10k CpG subsets.

Computational effort on CPUs varied largely between classifier algorithms, which particularly for XGBoost restricted the fine-tuning of its hyperparameter space and consequently its overall performance. In the meantime, GPU-accelerated versions of the XGboost package[118] have become available that might alleviate this restriction.

MATERIALS

EQUIPMENT

Starting data

- *Optional*: unprocessed IDAT files containing complete methylation values for the reference set and validation set as published in Ref. 1, available for download from the NCBI Gene Expression Omnibus (GEO) under accession number GSE109381.
- *Optional*: If the user desires to replicate the analysis pipeline presented in the main reference paper (Ref. 1), R scripts for data pre-processing including basic filtering and normalization, and also 3 x 3-fold nested CV and calibration (see the corresponding GitHub page https://github.com/mwsill/mnp_training).
- The benchmarking data, an unsupervised variance-filtered subset of the reference set using the 10,000 (10k) most variable CpG probes and
- R scripts necessary to run each ML-workflow within the 5 x 5-fold nested CV scheme and their evaluations, provided in the GitHub repository (https://github.com/mematt/ml4calibrated450k/tree/master/data).
- *Optional*: 450K DNA methylation tumor samples from The Cancer Genome Atlas (TCGA). For details on how to prepare the TCGA external validation cohort please see the GitHub repository (https://github.com/mwsill/mnp_training/blob/master/tsne.R) and for download details the NCI GDC Legacy Archive (https://gdc-portal.nci.nih.gov/legacy-archive).

Software

- Operating system: Linux$^+$ (e.g. Ubuntu 16.04.5 LTS, 18.04.2 LTS) and Macintosh (OSX El Capitan 10.11.6 or newer) were tested; $^+$for GPU (NVIDIA CUDA) accelerated SVM (Rgtsvm) we suggest using Linux.

- R: A language and environment for statistical programming v3.3.3 or newer: https://www.R-project.org/.
- R Studio IDE, a free and open-source integrated development environment for R v1.0.136 or newer: https://www.rstudio.com/products/RStudio/.
- R and RStudio running in Docker containers (rocker) ensuring clear and dedicated software environments. We tested our scripts in rocker containers with R v3.5.2 and RStudio v1.1.463. For details see https://www.rocker-project.org or https://github.com/rocker-org.

R packages for data preparation and pre-processing

CRITICAL The R packages listed below are under development and regularly updated therefore we recommend using the most recent stable version. These are downloadable from the Comprehensive R Archive Network (CRAN is a network of servers that store up-to-date versions of packages and documentations for R (https://cran.r-project.org/)) or from Bioconductor (an open source software for bioinformatics that uses R (https://www.bioconductor.org/)). Please select the checkbox "Install dependencies" in RStudio or explicitly set the dependencies argument to TRUE (`install.packages("foo", dependencies=T)`). Installing the devtools package (`install_github`) permits to directly install packages from GitHub.

- conumee Bioconductor package v1.3.0 for copy-number variation analysis
- minfi Bioconductor package v1.14.0 for obtaining raw signal intensities from IDAT files and normalization
- rhdf5 Bioconductor package v2.26.2 to provide an interface between HDF5 and R to store and access very large and/or complex datasets with metadata[119].

- limma package v3.24.15 (`removeBatchEffect` function) to fit univariate linear models to correct for the type of tissue material (FFPE or frozen)
- Rtsne package v0.15 to apply t-distributed stochastic neighbor embedding (t-SNE)[57,58]
- dbscan package v1.1-3 for density-based clustering of applications with noise (DBSCAN) and related algorithms[59]
- RSpectra package v0.14-0 containing solvers for large-scale eigenvalue and SVD problems[120]

General machine learning frameworks:

- Classification and Regression Training

  caret v6.0-81 (http://topepo.github.io/caret/index.html)
- *Optional*: Machine learning in R

  mlr v2.13 (https://mlr.mlr-org.com/ or https://github.com/mlr-org/mlr)

Machine learning algorithms(classifiers):

- randomForest v4.6-12 or newer (most recent 4.6-14): https://cran.r-project.org/web/packages/randomForest/index.html
- glmnet v2.0-10 or newer (most recent v2.0-16): https://cran.r-project.org/web/packages/glmnet/index.html
- We tested multiple SVM implementations including:

  e1071 v1.7-0 (https://cran.r-project.org/web/packages/e1071/index.html),

  LiblineaR v2.10-8 (https://www.csie.ntu.edu.tw/~cjlin/liblinear/)

  kernlab v0.9-25 or newer (most recent v0.9-27) also available through the caret package (https://github.com/cran/kernlab)

  #Rgtsvm v0.5 (https://github.com/Danko-Lab/Rgtsvm)

- xgboost v0.82.1 (https://xgboost.readthedocs.io/en/latest/R-package/index.html) more recently GPU support became available, see https://xgboost.readthedocs.io/en/latest/build.html (note: in our scripts we use the CPU version only)

  > *CRITICAL: Please note that on Mac OSX only a single-threaded version of xgboost will be installed when using the install.packages("xgboost") command. This is because the default Apple Clang compiler does not support OpenMP. To enable multi-threading on Mac OSX please consult the xgboost installation guide (https://xgboost.readthedocs.io/en/latest/build.html#osx-multithread).*

Calibration (i.e. post-processing) algorithms:

- base R for logistic regression (`glm` function)
- brglm v0.6.1 and brglm2 v.0.5.1 (https://github.com/ikosmidis/brglm2) for Firth's penalized logistic regression
- glmnet v2.0-10 or newer (most recent v2.0-16) https://cran.r-project.org/web/packages/glmnet/index.html for ridge/L2 penalized multinomial logistic regression

Performance evaluation

- HandTill2001 v0.2-12 https://cran.r-project.org/web/packages/HandTill2001/index.html for multiclass AUC

Optional extra libraries and R packages:

CRITICAL: For GPU accelerated Rgtsvm and builds of xgboost NVIDIA CUDA-capable graphic cards are required with additional software setup of the CUDA toolkit library

- CUDA library: https://developer.nvidia.com/cuda-toolkit-archive and installation guide for Linux https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html *CRITICAL: Compiling Rgtsvm using CUDA 9.0 prohibits the architecture (sm_20) of early GeForce series. Furthermore, a distinct architecture type is needed for TitanX (sm_50) or P100 (sm_60) cards with additional manual configuration. Please consult the GitHub page for Rgtsvm* (https://github.com/Danko-Lab/Rgtsvm).
- Boost library: http://www.boost.org/users/download/ (is required only for Rgtsvm)
- Extra R packages (required for Rgtsvm, https://github.com/Danko-Lab/Rgtsvm):

  bit64 v0.9-7, snow v0.4-3, SparseM v1.77, Matrix v1.2-15

EQUIPMENT SETUP

Required data All comparative analyses described in this study were performed using 450k Illumina Methylation Beadchips data of the CNS tumour reference cohort (Ref. 1) . To limit computational burden, we performed an unsupervised variance filtering selecting the 10 000 most variable CpG probes based on training data of each (sub)fold while subsetting the CpG features of the corresponding test or calibration set accordingly. These variance filtered train-test matrix pairs (n=30; `betas.1.0.RData – betas.5.5.RData`) provided the basis of all ML-workflow comparisons. For instance, the `betas.1.1.RData` file contains a `betas.train` matrix object with 1720 rows (cases) and 10,000 columns (i.e. most variable CpG probes for those 1720 cases) and a `betas.test` matrix object with 484 rows (cases) and the same 10,000 most variable CpGs as selected on the `betas.train` matrix (**Fig. 1**, part 2, step 5). The `betas.K.k.RData` files can be readily downloaded from our Dropbox folder (`betas.train.test.10k.filtered`; http://bit.ly/2vBg8yc) or generated using scripts (PROCEDURE step 5). The true label vector (`y`) for each of the 2,801 cases from 91 possible classes is provided within the `y.RData` file (3kB) at https://github.com/mematt/ml4calibrated450k/tree/master/data. The exact same fold assignment `nfolds.RData` (84kB) for distributing cases (using stratified sampling) into more robust 5 x 5-fold nested CV scheme is also provided on the GitHub page above (https://github.com/mematt/ml4calibrated450k/tree/master/data).

Optional data A collection of scripts used for pre-processing (normalization and batch adjustment of the IDAT data), and to train (3 x 3-nested CV) and validate the RF classifier presented in the main reference article is provided at https://github.com/mwsill/mnp_training.

Required hardware Multi-core laptops, preferably high-end desktop or workstation level CPUs and at least 8Gb RAM (32Gb or more for certain highly-parallelized implementations under Linux) are suggested. A list of CUDA-capable NVIDIA graphic cards can be found at https://www.geforce.com/hardware/technology/cuda/supported-gpus.

To provide an impression, run-times (**Table 2**) are based on various PCs and/or laptops equipped with Intel Core i7 (7700k @ 4.2 GHz, 4 cores/8 threads; 6850k @ 3.6GHz, 6 cores/ 12threads) or Core i9 (i9-8950HK @ 2.9GHz 6 cores/12threads; 7960X @ 2.8 GHz, 16 cores/32 threads) CPUs and 32 - 128Gb RAM, and on NVIDIA Geforce GTX 1080Ti cards using CUDA 8.0 or on high-performance computing optimized Amazon Elastic Compute Cloud (EC2) C5n instances (c5n.18xlarge with 72 vCPU on Intel Xeon Platinum 8000 and 192Gb RAM).

Downloading and installing software Please follow the instructions in the installation links listed in Equipment or in the R scripts on the respective GitHub pages (https://github.com/mematt/ml4calibrated450k and https://github.com/mwsill/mnp_training).

CRITICAL Most commands should be executed within R, however, for certain installations the Unix shell prompt using a terminal window or within RStudio (available in more recent versions) might be necessary. To perform all analyses, we recommend creating a separate directory and downloading all data and scripts there.

Downloading and organizing the data The size of `betas.K.k.RData` is around 215 MB/(sub)fold ~5.3Gb in total (n=30 folds; http://bit.ly/2vBg8yc). All other files are provided in the respective GitHub repositories. We suggest using a common path for all data objects either within the working directory (e.g. "./data/") or outside ("/home/rstudio/data/").

PROCEDURE

Download and extract the data TIMING ~ 1.5h (50 MB/s)

1. Download and unzip the raw data archive of the GEO series GSE90496 (22.7 Gb) from NCBI GEO:

CRITICAL STEP: These two commands (wget and tar) should be run in a UNIX Terminal.

```
# Note: these commands should be run in a UNIX Terminal
wget https://www.ncbi.nlm.nih.gov/geo/download/?acc=GSE90496&format=file
tar GSE90496_RAW.tar
```

2. Download the corresponding annotation data from GEO using the Bioconductor package GEOquery. Execute the commands below in the R or RStudio console:

```
library(GEOquery)
gse <- getGEO(“GSE90496”, GSEMatrix = TRUE, getGPL = FALSE)
anno <- pData(gse$GSE90496_series_matrix.txt.gz)
```

Pre-processing TIMING ~ 1h

3. After extracting the raw data archive, the methylation data can be read into the R workspace using the R or RStudio console:

```
filepath <- file.path("GSE90496_RAW/",
                      gsub("_Grn.*", "",
                           gsub(".*suppl/","",anno$supplementary_file)
                           )
                      )
RGset <- read.metharray(filepath, verbose = TUE)
```

CRITICAL STEP: The pre-processing steps (**Fig. 1,** part 1, step 2) of normalization and basic filtering presented in https://github.com/mwsill/mnp_training/blob/master/preprocessing.R need to be performed to generate the matrix of beta methylation values. All comparisons and presented results in

Table 2 were obtained on normalized but not batch adjusted beta values (**Fig. 1**, part 1, step 2 & 4, path of solid arrows).

CRITICAL STEP: If desired, batch effects originating from FFPE or fresh frozen samples (**Fig. 1,** part 1, step 3, path of dashed arrows) can also be adjusted for using a linear model similar to [1]. To note, however, that we found no major confounding effect associated with FFPE or frozen sample types.

Load prerequisite data objects TIMING < 1 min

CRITICAL: We assume that the GitHub repository (https://github.com/mematt/ml4calibrated450k) is downloaded with all R scripts into a folder, which serves as the current working directory for R (see EQUIPMENT SETUP). We suggest using a separate data folder to hold the required `.RData` files and objects within the working directory (“./data/”) or at another common path e.g. “/home/rstudio/data/”.

4. Load the true labels vector `y` (containing true class labels for all 2801 cases) to perform internal 5 x 5-fold nested cross-validation (CV) and load the list nfolds (`nfolds.RData`) to reproduce our fold assignments. The list `nfolds` was generated by the `makefolds.R` script and function `makefolds(y, cv.fold = 5)` that generates stratified samples from each class of `y` (overall $n_{min} = 8$; within subfolds $n_{min} = 4 - 6$).

```r
load("./data/y.RData") # or “/home/rstudio/data/y.RData”
load("./data/nfolds.RData") # or “/home/rstudio/data/nfolds.RData”
```

?TROUBLESHOOTING For details see **Table 6**.

Download the pre-filtered benchmarking data set TIMING ~ 18 min (5MB/s)

5. Download the readily prepared benchmarking data sets (`betas.1.0.RData –` `betas.5.5.RData` ; 5.3Gb) that contain the variance filtered (10,000 CpG probes) training-test set (`betas.train` ; `betas.test`) pair for each K.k (sub)fold from the linked Dropbox folder (*betas.train.test.10k.filtered*) at http://bit.ly/2vBg8yc.

```
# We suggest a common data folder, for instance:
“/home/rstudio/data/”
“./data/” # or within the working directory
```

CRITICAL STEP: All implementations of the investigated ML-algorithms require this same type of input data structure to function correctly. This data structure can be generated from any input data using the subfunction_load_subset_filter_match_betasKk() function in the identically named .R script (https://github.com/mematt/ml4calibrated450k/blob/master/data/).

PAUSEPOINT: After data preparation, the protocol can be halted and the ML-workflows can be applied at any later time point.

Internal validation example using tuned random forests (tRF) classifier TIMING: ~13h

6. Setup and import required R packages (**Fig. 1**, step 6).

```
# Parallel backend
library(doParallel)
library(doMC)
# Random Forests classifier
library(randomForest)
# Caret framework for tuning randomForest hyperparameters
library(caret)
```

?TROUBLESHOOTING For details see **Table 6**.

7. Setup the parallel backend for the parallelized version of RF (`rfp.R`) and the caret package for variable tuning.

```
# Number of cores (threads available).
# It is usually 2x (physical core count) on hyperthreaded Intel or AMD CPUs
cores <- detectCores() # assigns all available threads

# Consider leaving 1 thread for the operating system.
cores <- detectCores()-1
registerDoMC(cores) # registerDoParralel(cores)
```

CRITICAL STEP: it is recommended to leave one thread free for the operating system.

?TROUBLESHOOTING For details see **Table 6**.

8. For step 9-10 (**Fig. 1**) source the R.script (`subfunctions_tunedRF.R`) that contains:

- the `rfp()` function that provides a parallelized wrapper for the `randomForest()` function of the identically named R package. This function is also provided in a separate R sciprt (`rfp.R`) that can be used to fit vanilla RF (vRF).
- the `customRF` function for the caret package to enable tuning RF hyperparameters including `ntree`, `mtry` and `nodesize`
- the function `subfunc_rf_caret_tuner_customRF()` to perform grid search using an extra nested n-fold CV with the *caret* package

CRITICAL STEP: `rfp` *and* `customRF` *functions can be adjusted depending on the tuning grid size and the available thread count of the hardware, directly with the argument* `mc = 4L` *or with the* `cores = 4L` *argument of the* `trainRF_caret_custom_tuner` *function. For example, on a 72 vCPU (c5n.18xlarge AWS instance) for a hyperparameter grid of 8* `mc` *can be set up to* `9` *to fully utilize CPU resources.*

```
# Subfunctions to define and perform custom grid search using
the caret package
source("subfunctions_tunedRF.R")
```

?TROUBLESHOOTING For details see **Table 6**.

9. Next, source the R script (`train_tunedRF.R`) that contains a custom function for the whole tuning process of RF hyperparameters including `mtry`, `ntree` and `nodesize` as well as p_{varsel}.

This script contains the function `trainRF_caret_custom_tuner()` that performs the following tasks:

- An extra, nested hyperparameter tuning using 5-fold CV fitted only on the training set.
- The function performs two runs of the RF algorithm in order to select the most important features:

  During the 1. run: variable selection of `p` most important CpG probes (i.e. p_{varsel}) based on the importance measure of mean decrease in accuracy for this the `rfp` function is applied in parallel by default on 4 cores (`mc = 4L`).

  In the 2. run: the `randomForest()` function is fit again on the whole training data using only the selected `p` probes from the 1. run (single core), then the performance metrics *ME*, *BS* and *LL* are evaluated to provide the optimal number of `p` features with respect to each tuning metric.

```r
# Training & Hyperparameter tuning & Variable selection performed here
source("train_tunedRF.R")
```

?TROUBLESHOOTING For details see **Table 6**.

10. Finally, source the R script (`nestedCV_tunedRF.R`) that contains the full implementation of tuning RF within the 5 x 5-fold nested CV scheme and the R script containing the required evaluation metrics (`evaluation_metrics.R`).

    The computation time for tuning a hyperparamter grid of size 16 with extra nested 5-fold CV within each train set (`nodesize = 1; ntree = [500, 1000, 1500,` `2000]; mtry = [80, 90, 100, 110]`) while tuning p_{varsel} `= [100, 200, 500, 1000, 2000, 5000, 7500, 10000]` for BS, ME, and LL concurrently using 72 vCPU c5n.18xlarge AWS instances amounted to ~16 – 25min/(sub)fold; and a total of 12 – 13 h for the full 5 x 5-fold nested CV.

The first script contains the function `run_nestedcv_tunedRF()` that performs the following tasks:

- It creates an output folder *tRF* and exports resulting variables and objects into a `CVfold.1.0.RData` file for each (sub)fold, respectively (see *comment #(1)* in the code segment below).
- If `mtry.min` and `mtry.max` arguments are left at default (`NULL`), the function equally divides `floor(sqrt(ncol(betas))*0.5)` and `floor(sqrt(ncol(betas)))` to `length.mtry` parts (see *comment #(2)* in the code below).
- *Sourcing the* `evaluation_metrics.R` *script is required by the* `run_nestedcv_tunedRF()` *function to be able perform* p_{varsel} *tuning with respect to BS, ME, and LL.*
- *If the argument* `use.default.nodesize.1 = TRUE` *then only the default value of* `nodesize = 1` *for classification is tested and all values provided for the argument* `nodesize.proc` *are ignored. Otherwise the percentage values provided for* `nodesize.proc` *are complemented with the defaults for classification (*`= 1`*) and regression (*`= 5`*), respectively; and this extended* `.nodesize = floor(c(1, 5, value_nodesizes))` *vector is used in the* `subfunc_rf_caret_tuner_customRF()` to expand the tuning grid (see *comment #(3)* in the code below).

```
# Source scripts
source("nestedcv_tunedRF.R")
# Source evaluation metrics (BS, ME, LL) for p_varsel tuning
source("evaluation_metrics.R")
```

```r
# Run the function that performs the task
run_nestedcv_tunedRF(y.. = NULL,
                  betas.. = NULL,
                  nfolds..= NULL,
                  # y, nfolds are fetched from the .GlobalEnv
                  path.betas.var.filtered ="./data/betas.train.test.10k.filtered/",
                  fname.betas.p.varfilt = "betas",
                  subset.CpGs.1k = F, # subset to 1k CpGs
                  n.cv.folds = 5,
                  K.start = 1, k.start = 0,
                  K.stop = NULL, k.stop = NULL,
                  n.cv = 5, n.rep = 1, # caret # extra nested tuning
                  mtry.min = 80, mtry.max = 110, length.mtry = 4, # (2)
                  ntrees.min = 500, ntrees.max = 2000, ntree.by = 500,
                  use.default.nodesize.1.only = T, # (3)
                  nodesize.proc = c(0.01, 0.05, 0.1),
                  p.n.pred.var = c(100, 200, 500, 1000,
                                   2000, 5000, 7500, 10000),
                  cores = cores,
                  seed = 1234,
                  out.path = "tRF", # (1)
                  out.fname = "CVfold")
```

The output file `CVfold.1.0.RData` is comprised of the following objects:

- predicted scores by the tuned RF using p_{varsel} (`p.n.pred.var`) features based on the lowest BS, ME and LL values: `scores.pred.rf.tuned.brier`, `scores.pred.rf.tuned.miscerr`, `scores.pred.rf.tuned.mlogl`,
- `rfcv.tuned`: the output object of the `trainRF_caret_custom_tuner()` function
- `fold`: the corresponding (sub)fold with training and test sets

CRITICAL STEP: the output .RData file can be large, as it contains multiple copies of large matrices (2,801 x 10,000 approx. 215 MB each) adding up to 1 - 1.5 Gb. Hence, the complete nested CV scheme might require 40-50Gb free space on the respective drive.

CRITICAL STEP: Alternatively, all tRF functions can be substituted for similarly named functions of other investigated ML-classifiers at each Step 8 – 10. At each respective step, the corresponding `subfunctions` and (if available) the `train` function (e.g. `trainRF`, `trainGLMNET`, `train_SVM_LiblineaR`, `train_SVM_e1071_LK`, `trainXGBOOST_caret_tuner`) and finally the integrated `run` within the nested CV scheme (e.g. `run_nestedcv_vRF`, `run_nestedcv_GLMNET, run_nestedcv_SVM_e1071, run_nestedcv_SVM_LiblineaR, run_nestedcv_SVM_Rgtsvm, run_nestedcv_XGBOOST`) should be used.

PAUSEPOINT: Each `run_nestedcv_`…`` function can be halted using the "break/stop" button in Rstudio at any time point or (sub)fold. It can be restarted at a later time point from the K.k-fold and using `K.start`, `k.start` arguments.

However, when stopping/breaking the function it might require some time (up to several minutes) for RStudio to exit and recover from highly parallel implementations. During this phase, RStudio can prompt you in a window to terminate or close the RStudio session, which can be cancelled. Nevertheless, in certain cases, the R session might still eventually terminate or collapse with complete data loss of the .GlobalEnv.

?TROUBLESHOOTING For details see **Table 6**.

Calibration TIMING on a single thread for LR ~30s; FLR ~9 min; MR ~20min; multi-threaded (n=11) MR ~8min

Calibration can be performed at any later time point after all 5 x 5-fold CV (n=30; 1.0 – 5.5) base ML-classifier output scores are generated and saved.

To replicate the presented results in (**Table** 2) the user can perform calibration with all investigated post-processing algorithms in steps 11-14 on the (raw) score output of any ML-algorithm. A worked example for tRF is presented below.

11. Source the R script (`calibration_Platt_LR.R`) that performs post-processing (**Fig. 1**, step 11) using Platt scaling with logistic regression (LR) on the predicted (raw) scores of tuned RF.

```
# Source the script
source(“calibration_Platt_LR.R”)
```

The script contains the function `calibrate_LR()` that outputs either `probsCVfold.LR.K.k.RData` (if `save.metric.name.into.output.file = F`) or `probsCVfold.<brier|miscerr|mlogl>.LR.K.k.RData` (if `save.metric.name.into.output.file = T`) files into a subdirectory that is comprised of the matrices of raw tRF `scores` and their LR calibrated probabilities (`probs`) for each (sub)fold.

?TROUBLESHOOTING For details see **Table 6**.

12. Likewise, to perform Platt scaling with Firth’s penalized LR source the R script (`calibration_Platt_FLR.R`) (**Fig. 1**, step 12).

```
# Source the script
source(“calibration_Platt_FLR.R”)
```

This script contains the function `calibrate_FLR()` with maximum iteration (`br.maxit`) = `10000`. Analogous to `calibrate_LR()`, this function outputs either `probsCVfold.FLR.K.k.RData` or `probsCVfold.<brier|miscerr|mlogl>.FLR.K.k.RData` files into the predefined subdirectory.

?TROUBLESHOOTING For details see **Table 6**.

13. Source the R script (`calibration_MR.R`) to perform post-processing (**Fig. 1**, step 13) using ridge penalized multinomial logistic regression (MR) on the raw scores of tRF.

```
# Source the script
source(“calibration_MR.R”)
```

Similarly to the previously described calibrator functions, this script contains the `calibrate_MR()` function that outputs either `probsCVfold.MR.K.k.RData` or `probsCVfold.<brier|miscerr|mlogl>.MR.K.k.RData` files into the `out.path` directory.

?TROUBLESHOOTING For details see **Table 6**.

14. We also provide a wrapper function `calibrator_integrated_wrapper()` around `calibrate_LR()`, `calibrate_FLR()`, and `calibrate_MR()`. Use this to apply all post-processing algorithms (LR, FLR and MR) to all ML-classifiers and $\text{tRF}_{\text{BS | ME | LL}}$ with a single function call.

```
# Source the script
source(“calibrator_integrated_wrapper_LR_FLR_MR.R”)

calibrator_integrated_wrapper(out.path ="./tRF-ME-calibrator-integrated/",
                              load.path.w.name = "./tRF/CVfold.",
                              which.optimized.metric.or.algorithm = "miscerr",
                 # c("brier", "miscerr", "mlogl", "vanilla", "svm", "xgboost")
                              which.calibrator = "all",
                 # c("Platt-LR", "Platt-FLR", "ridge-MR", "all")
                              verbose.messages = F,
                              brglm.ctrl.max.iter = 10000, # for FLR
                              save.metric.name.into.output.file = T,
                              parallel.cv.glmnet = T, # for MR
                              setseed = 1234)
```

The argument `which.optimized.metric.or.algorithm` accepts the following values: for tRF (`"brier", "miscerr", "mlogl"`), for vRF (`"vanilla"`), for SVM (e1071; `"svm"`), and for XGBoost (`"xgboost"`). The argument `which.calibrator` can be used to specify what type of post-processing (`”Platt-LR”`, `“Platt-FLR”`, `”ridge-MR”`, `”all”`) should be applied.

PAUSEPOINT: Performance evaluation can be performed at any later time point on raw ML-classifier outputs (raw scores) or probability outputs of calibrated workflows. To carry out the evaluation only the .RData object of the outerfold test sets (1.0 – 5.0) are required. ?TROUBLESHOOTING For details see **Table 6**.

Performance evaluation TIMING < 30 s / ML-algorithm

15. Source the R script `performance_evaluator.R` that invokes subfunctions for each performance metrics including (ME, AUC, BS and LL) by sourcing the `evaluation_metrics.R` script that carries out the performance evaluation on tRF’s (or any other ML-algorithms’) raw/uncalibrated `scores` and post-processed probabilities (`probs`).

```r
# Source the script for complete performance evaluation of tRF
source("performance_evaluator.R")

# Source the required evaluation metrics
source("evaluation_metrics.R")
```

Use the `performance_evaluator()` function (**Fig. 1,** step 14) that returns a list that includes `misc.err`, `auc.HandTill`, `brier` and `mlogloss` values for the respective `scores` or `probs` matrix objects.

```r
# Folder structure:
# “./tRf/CVfold.1.0.RData … CVfold.5.5.RData # raw scores tRF_BS|ME|LL
# “./tRF-BS-calibrator-integrated/probsCVfold.brier.LR.1.0. … 5.0.RData
# “./tRF-BS-calibrator-integrated/probsCVfold.brier.FLR.1.0 … 5.0.Rdata
# “./tRF-BS-calibrator-integrated/probsCVfold.brier.MR.1.0 … 5.0.RData
# “./tRF-ME-calibrator-integrated/probsCVfold.miscerr.<LR|FLR|MR>.1.0 … 5.0.RData
# …
# “./tRF-LL-calibrator-integrated/probsCVfold.mlogl.<LR|FLR|MR>.1.0 … 5.0.RData
# …


# Create concurrent lists of tRF.folder.path & tRF filename stumps to replicate
# the above folder & file path structure
tRF.opt.metrics.folder.name <- c("BS", "ME", "LL")
tRF.folder.path <- as.list(rep(paste("tRF",
                                     tRF.opt.metrics.folder.name,
                                     "calibrator-integrated", sep = "-"),
                               each = 3))
tRF.opt.metrics.RData <- rep(c("brier", "miscerr", "mlogl"), each = 3)
calibrator.name <- c("LR", "FLR", "MR")
tRF.fname.stump <- as.list(paste("probsCVfold",
                                 tRF.opt.metrics.RData,
                                 rep(calibrator.name, 3),
                                 sep = "."))

# The actual function call of `performance_evaluator` using mapply()
tRF.l.perfevals.all <- mapply(FUN = performance_evaluator,
                              load.path.folder = tRF.folder.path,
                              load.fname.stump = tRF.fname.stump)
# Add column names
colnames(tRF.l.perfevals.all) <- tRF.fname.stump
tRF.l.perfevals.all  # matrix
```

The code snippet above generates (in < 4 min) the complete performance evaluation of all tRF algorithms (3 [$\text{tRF}_{\text{BS | ME | LL}}$] x 3 [$\text{calibrator}_{\text{LR | FLR | MR}}$] x 4 [$\text{metrics}_{\text{ME | AUC | BS | LL}}$]), see also **Table 2**. The `performance_evaluator()` function can be used to evaluate the performance of other ML-classifiers and workflows after specifying the `load.path.folder`, `load.fname.stump`, and `name.of.obj.to.load` arguments. By default, the cases are reordered and scaled and all metrics (ME, AUC, BS, LL) are calculated.

CRITICAL STEP: If raw scores or probabilities of SVM generated by the e1071 package are evaluated, it is essential to set the argument `reorder.columns.svm.e1071 = TRUE` *because* *the 1-vs-1 coupling approach of e1071 changes the order of columns (class labels) for each K.k (sub)fold differently. Without re-matching the columns to the levels of* y*, the resulting performance metrics are extremely poor.*

?TROUBLESHOOTING For details see **Table 6**.

TIMING

Detailed timing information about each ML-classifier is provided in the Run-time column of **Table 2**. The most time-consuming part of the PROCEDURE section is Step 10, the tuning of the respective ML-classifier within the 5 x 5-fold nested CV scheme (**Fig. 1**, steps 7-10), which can vary from ~5 h up to 4-5 days depending on the respective algorithm and given hardware.

Steps 1-2, NCBI GEO download: ~1 h

Steps 3, pre-processing: ~1 h

Steps 4 and 5, prerequisite data download: ~20 min

Steps 6-10, internal validation of tuned RF (on 72 threads, AWS EC2 c5n.18xlarge): ~12-13h

Step 11-14, calibration: - single thread: A) LR ~30 s ; B) FLR ~9 min ; C) MR ~20 min;
- multi-threaded (@ 11 threads): C) MR ~8 min

Step 15, performance evaluation: <30 s / ML-classifier

?TROUBLESHOOTING

Troubleshooting advice is provided in **Table 6**.

ANTICIPATED RESULTS

In this section, the benchmarking results using the investigated ML-workflows are described and interpreted in detail corresponding to their order in **Table 2**.

Random forests

Vanilla RF (vRF) with default settings represented the computationally least expensive baseline method (<40 min). Nonetheless vRF achieved a ME of 4.8%, an AUC of 99.9% with corresponding BS and LL of 0.32 and 0.78 respectively (**Table 2**). BS was similar to SVM-LK but high compared with ELNET or boosting. Raw scores of vRF ranged between 0 – 0.948. Platt scaling with LR and Firth-LR improved BS and LL by factors of 2-4, the latter yielded somewhat better numerical results. MR slightly outperformed both Platt variants and achieved remarkably low 10th and 9th overall BS (0.073) and LL (0.155) metrics, respectively. Notably, row sum scaling to 1 could alone improve BS by up to 20% while concurrently lowering LL by 0-10%.

Tuned RF variants selected almost always `ntree` =1000-2000) trees, except for a few occasions (`ntree` =500) on nested subfolds. RF tuned for ME ($\mathrm{tRF_{ME}}$) showed the 10th lowest error rate (3.5%) overall with 4th highest AUC (99.9%), while it had relatively high BS (0.35) and LL (0.86) similar to vRF (**Table 2**). Interestingly, BS- ($\mathrm{tRF_{BS}}$) and LL-tuned RF ($\mathrm{tRF_{LL}}$) both had substantially (57%) higher error rates of ~5.5%. Both $\mathrm{tRF_{BS}}$ and $\mathrm{tRF_{LL}}$ chose similarly small models consisting of ca. 100-500 CpG probes, whereas $\mathrm{tRF_{ME}}$ models were inflated upwards to 1,000-10,000 CpG probes. The range of raw score outputs was closer to 1 for $\mathrm{tRF_{BS}}$ [0, 0.973] and $\mathrm{tRF_{LL}}$ [0, 0.986] while $\mathrm{tRF_{ME}}$ had more confined [0, 0.881] raw score outputs. Terminal node size tuning revealed that the default setting for classification (=1) was always superior to regression (=5) or to 1% and 10% of CpGs. Firth regression was marginally better than simple LR. Among these calibrated models $\mathrm{tRF_{ME}}$

showed the lowest error rates (3.7-4.2%) and highest AUCs (99.8-99.9%). Likewise, BS (0.062-0.086) and LL (0.15-0.156) metrics of tRF_{ME} benefited the most from calibration. The respective metrics of tRF_{BS} (BS 0.086; LL 0.194-0.266) and tRF_{LL} (BS 0.089; LL 0.205-0.291) were markedly higher. After calibration, ME and AUC metrics often got slightly worse compared to the corresponding raw tRF model. This happened to all three tRF models, noticeably when using Platt scaling with LR or FLR. Worsening of ME and AUC metrics was typical of tRF_{ME}, which already had lower values of ME and AUC. Although tRF_{BS} and tRF_{LL} models had higher baseline ME they responded only marginally to calibration (if at all) in terms of ME improvement. On the other hand, BS and LL metrics were the major beneficiaries of post-processing with improvements by factors of ~2-6x independent of how the raw model was tuned during feature selection (**Table 2**). Calibration with MR resulted in the largest performance improvement for nearly all versions of tRF. Additionally, this was the only calibrator that could improve all metrics compared to the respective raw tRF model (except for ME of tRF_{LL} and AUC of tRF_{BS}). MR-calibrated tRF_{ME} (tRF_{ME}+MR) showed the fourth lowest ME (2.7%), and BS (0.046), while it achieved the 2nd best overall LL (0.095), and AUC (99.9%).

Elastic net

ELNET as a stand-alone method was tested in two scenarios. First, we fitted only the 1,000 most variable CpG probes, which nevertheless resulted in the 8th lowest ME and 5th highest AUC overall, plus it produced lower BS and LL measures by a large margin compared to all other base classifiers trained on all 10,000 CpGs (**Table 2**). Almost exclusively $\alpha = 0$ was used for fitting except for a single subfold (1.3) for which $\alpha = 0.025$ with $\lambda_{1SE} = [0.0010 – 0.0036]$ settings were used. Probability scores on outerfold test sets were in the range of $[6.76\times10^{-13}, 0.99996]$.

In the second scenario ELNET was applied to the full scope of 10,000 probes with outerfold test set probabilities in the range of [4.61×10^{-13}, 0.99997]. It outperformed all other tuned-, but uncalibrated classifier algorithms across all metrics (**Table 2**). Despite the larger 10,000 feature space $\alpha = 0$ (ridge) settings were selected for all outer folds. Optimal λ_{1SE} ranged between 0.012 – 0.038 depending on the given (sub)fold. The limiting solution of an $\alpha = 0$ ELNET can be thought of as a SVM with all 10,000 CpG probes selected [9,78]. In contrast, SVM implementations used 1,300-1,600 support vectors.

Overall ELNET (10k) achieved the 3rd–5th best performance profile behind two-stage workflows of calibrated SVMs and tRF_{ME}+MR showing marginally higher ME (2.7%), BS (0.048), LL (0.109) and negligibly lower AUC (99.9%).

Support vector machines

RBF kernel SVM were fitted on the 100 most variable CpGs running 5-repeats of 10-fold CV grid search on $C = 2^{-5:8}$; $\sigma \mid \gamma = 2^{-10:4}$. Their CPU implementations showed optimal settings at $C = 16$; $\sigma = 2^{-7}$ with ME = 9.1 – 9.3% (caret-ksvm) and $C = 0.1$; $\gamma = 0.5$ with ME = 14.3 – 15.1% (e1071). In contrast, GPU-accelerated RBFs fitted on all 10,000 CpGs showed extremely poor 5x CV ME ranging between 65.23 – 98.43% over the tuning grid of $C = 10^{-5:3} \mid 2^{-5:-1}$ and $\gamma = 10^{-5:0} \mid 2^{-5:5}$.

Therefore, we switched to linear kernel SVM (SVM-LK). Tuning all 8 available LK models in LiblineaR on the prototype subfold (1.1) using $C = 10^{-3:3}$ grid and 5-fold CV showed that the accuracies of all models laid within a close range (92-96%). Individual model accuracies varied by less than 5% (except for L1-regularized LR) given that C was sufficiently large ($C \geq 0.01$) supporting the fact that SVM-LK solvers are indeed not very sensitive to C. Among these models, SVC by Crammer and Singer (CS)[73,74,76,121] with C = 1000 performed the best (ME = 2.7%). Closely followed by L2-regularized L2-loss SVC and L2-regularized LR (ME

= 2.9%) models. Hence the CS model was implemented into the nested-CV scheme and showed the 6th lowest ME (2.8%). It is of note, however, that CS only allows for class but not probability outputs. We also implemented SVM-LK using the e1071 package and tested whether weighting with inverse class frequency improved results as suggested in the literature[71,72,79]. To the contrary, it increased test errors by ~10% from 4.5% to 4.9%. SVM-LK of e1071 behaved similarly to LiblineaR's CS method (**Table 2**), achieving an ME = 3.2% (7th lowest overall) while it had substantially worse BS (0.37) and LL (0.98; worst overall) calibration profiles than ELNET or XGboost comparable to vRF and tRF models. Raw probability estimates of SVM-LK ranged between [5.08×10^{-5}, 0.9657]. Platt scaling with Firth regression was more effective for improving ME = 2.1% (lowest overall) while simple LR could more effectively improve BS (2nd) and LL (4th) by factors of 8x-9x respectively. Post-processing with LR and FLR spread out probabilities to the identical range of [2.22×10^{-16}, 1]. The most comprehensive improvement for all metrics was achieved by MR (SVM-LK+MR). It reduced BS by a factor of 9.5 and LL by 11.5 resulting in the 2nd lowest ME (2.1%) and AUC (99.9%), lowest BS (0.039) and lowest LL (0.085). Correspondingly, MR-calibrated probabilities extended over the range [7.1×10^{-25} - 0.99999]. GPU-accelerated SVM-LK internally applied a 1-vs-all coupling framework with LL-optimized global softmax to calculate multiclass probabilities. This yielded somewhat higher ME (3.3%; 9th overall) with 2nd best BS (0.056) and LL (0.144) among base classifiers behind ELNET. Nonetheless, training SVM-LKs on the GPU had the major advantage that it was 10-15x faster than on the CPU. Both e1071 and Rgtsvm identified C = 0.001 | 0.01 as optimal settings with the number of support vectors varying between 1,300-1,600 depending on the (sub)fold.

Boosted trees

Using XGboost's default parameter settings of boosted decision trees (**Table 3**) achieved a dismal ME of ~16%. To our knowledge, there is no information available in the literature on how to initialize hyperparameters for gradient boosted trees when fitting high-throughput methylation array data. Thus, we conducted an extensive parameter search of XGBoost investigating the combination of all settings shown in **Table 3** on the same prototyping subfold as before. Boosted models that used ME as an evaluation metric outperformed those using LL. XGBoost converged fast in `nrounds` < 200. Error rates of best performing parameter settings are presented in **Table 4**. The combination of all hyperparameters that showed the lowest MEs (0.045-0.066) on the prototype subfold (1.1) were implemented within an extra nested 3-fold CV scheme to find the optimal settings (**Table 5**). XGboost performed similarly to raw vRF and $\text{tRF}_{\text{LL}\,|\,\text{BS}}$ (**Table 2**) by showing an overall ME of 5.1%, AUC of 99.9% with 2nd lowest BS (0.15) and LL (0.43) among the investigated base ML-classifiers. Raw probability scores of tuned XGBoost models spanned the [8.75×10^{-6} - 0.9987] range. Also, among all the ML classifiers XGBoost responded the most to calibration with MR (XGBoost+MR), which revealed 4.6% ME while BS improved ~60% and LL approximately halved, while AUC slightly decreased. Platt-type post-processors improved BS slightly more efficiently than MR by a factor of 2 but LL only by 10-20%, while they concurrently worsened ME and AUC.

Calibration algorithms

Multinomial ridge regression demonstrated to be the best overall calibration method for all classifiers. It consistently outperformed Platt scaling variants for most evaluation metrics. Tuned RF and SVM-LK were most improved by calibration with MR. Their BS and LL were reduced by a factor of ~7.6-9 and ~9.5-11.5 respectively. Boosted trees benefited less markedly from ridge-calibration.

It is of note that calibration changed ME and AUC of raw classifier algorithms on multiple occasions. Although such changes were generally small, involving only 1-2 cases (0.3-0.7%) this could result in quite substantial proportional alteration of these performance metrics as even such minor changes represented 8-20% of the baseline ME (2.5-5.1%) of raw models including boosting, vRF and tRF. For binary classification, post-processing does not change the most probable class of a sample [22], but this is not necessarily true in the multiclass setting [27,44]. In part this is caused by the way in which we applied the binomial calibrator functions in a 1-vs-all manner [44]. Hence, they retained the ordering of scores within the class (i.e. over all samples) but not within rows (i.e. for each sample across all diagnostic classes). Thus, calibration can occasionally decrease the probability of the true class and result in a switch of the most probable class to another column or vice versa [22]. Additionally, complete or quasi-complete separation of classes in the p>>n feature space can lead to infinite calibrator model estimates, which can further complicate the above scenario of switching classes.

External validation on DNA methylation data from The Cancer Genome Atlas

Performance evaluation of vRF in a 5-fold nested CV setup on the external validation cohort based on the combination of thirty 450k DNA methylation microarray studies from TCGA with n=7,142 cases belonging to 46 classes using a feature space that was limited to the 32k most variable CpG probes across samples returned an ME of 0.135, AUC of 0.997, BS of 0.44, and LL of 1.07. Post-processing vRF scores with MR showed (**Fig. 2a-d**) similar improvements to the vRF + MR workflow that was fitted on BTMD by substantially improving ME (0.067), BS (0.100), and LL (0.217) by factors of ~2x, 4.4x and 5x respectively while slightly improving AUC (0.998). These results indicate that the vRF + MR workflow and likewise all other presented workflows are robust, generalize well, and can be easily applied to other methylation data sets to train well performing classifiers.

Summary

We performed extensive comparative analyses of four well-established classifier algorithms including RF, ELNET, SVM and boosted ensemble trees in combination with Platt scaling and multinomial ridge regression to support the choice for optimal high-throughput DNA methylation data analysis with regard to well-calibrated probability estimates in highly multiclass settings.

Tuned ELNET proved to be the best stand-alone algorithm. ELNET has the advantages that the lasso term can select a subset of CpG probes suitable for later biomarker identification development and that it is the most straightforwardly interpretable among the tested algorithms.

The best overall two-stage workflow was MR-calibrated linear kernel SVM and it generated the best overall BS, LL and AUC metrics. The second-best workflow was MR-calibrated tRF while also being computationally the fastest CPU workflow (and corresponding to the method published in Ref. 1 and Ref. 2).

Notably, linear kernel SVM and RF had worse BS and LL metrics than other classifiers, but they benefited the most from calibration. Although limited by the need to perform extensive parameter tuning, boosted trees achieved ME similar to uncalibrated vanilla and tuned RF but with better (second-best among base classifiers) BS and LL profiles.

For calibration, multinomial ridge penalized regression was the most effective regardless of the primary classifier, and hence should be the method of choice. Platt scaling variants suffered from separation of the classes and were (as originally designed) most suited for SVM.

Although all methods presented here were developed on a unique brain tumor methylation reference cohort and then applied to external TCGA data, the provided blueprint and insights are not limited to analyzing high-throughput biomedical data but can also be applied to any high-dimensional highly multiclass classification problem in other scientific fields.

We suggest hyperparameter values for the investigated ML-workflows to limit the tuning grid and the resulting computational burden while maximizing potential yield in model performance when fitting multiclass DNA methylation data sets like BTMD. For ELNET ridge ($\alpha = 0$) or ridgelike ($\alpha = 0.025$) settings with $\lambda_{1SE} = [0.0010 – 0.0036]$ can be good starting values. To limit the tuning of linear kernel SVM the range of C = $10^{-3:-2}$ proved to be sufficiently large enough for the ME estimates to stabilize. Tuned RF variants selected almost always at least `ntree` =1000(-2000) trees, while `mtry` values varied in the ±10% vicinity of the default value $\sqrt{p}$. The default setting for classification of terminal `nodesize` (=1) was always superior. Tuning RF for ME is more effective than BS or LL but it results in larger models (feature space of p_{varsel} =1000-10000 CpG probes). XGBoost requires the most extensive tuning, however, it converges fast (`nrounds`<100) while using `max_depth`= 6, `eta`= 0.1, `gamma`= 0 or 0.01, `colsample_bytree`= 200 or 500 yielded the best performance on BTMD.

Acknowledgements

The authors gratefully acknowledge funding from the DKFZ-Heidelberg Center for Personalized Oncology (DKFZ-HIPO) through HIPO-036 and from the German Childhood Cancer Foundation ("Neuropath 2.0 - Increasing diagnostic accuracy in paediatric neurooncology" (DKS 2015.01)). M.E.M. gratefully acknowledges funding from the German Federal Ministry for Economic Affairs and Energy within the scope of Zentrales Innovationsprogramm Mittelstand (ZF 4514602TS8).

The results shown in the external validation section and **Fig. 2** are entirely based on data generated by the TCGA Research Network: https://www.cancer.gov/tcga.

Author contributions

M.E.M. and M.S. conceptualized and developed machine learning workflows, performed the comparative analyses and wrote the manuscript. M.S., V.H. performed data preparation; D.C., D.T.W.J, S.M.P., A.v.D. composed the reference cohort and defined methylation classes; A.B. and M.Z. supervised the statistical aspects and data analysis, M.S. supervised the work and wrote the manuscript. All authors critically reviewed the manuscript and approved the final version.

Competing interests

A patent for a "DNA-methylation based method for classifying tumor species of the brain" has been applied for by the Deutsches Krebsforschungszentrum Stiftung des öffentlichen Rechts and Ruprecht-Karls-Universität Heidelberg (EP 3067432 A1) with V.H., D.C., D.T.W.J., S.M.P., A.v.D., and M.S. as inventors. The other authors have no competing interests to declare.

Data availability Data and code availability

The described collection of R scripts and the associated data files provided in GitHub repositories (https://github.com/mwsill/mnp_training and https://github.com/mematt/ml4calibrated450k) are free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation version 2. All analyses were performed within either local (https://www.R-project.org/) or docker containerized (https://www.docker.com) versions (rocker; https://www.rocker-project.org or https://github.com/rocker-org.) of the R: A language and environment for statistical programming v3.3.3 - 3.5.2 using the R Studio IDE, a free and open-source integrated development environment for R (v1.0.136 or v1.1.463; https://www.rstudio.com/products/RStudio/). Unprocessed IDAT files containing complete methylation values for the reference set and validation set as published in Ref. 1, available for download from the NCBI Gene Expression Omnibus (GEO) under accession number GSE109381 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE109381). The variance filtered outer- (fold IDs: 1.0, 2.0, …, 5.0) and innerfold (fold IDs: 1.1, 1.2, …, 1.5; 2.1, 2.2, …, 2.5; … 5.1, 5.2, … 5.5) training-test set pairs (altogether n=30; i.e. 1.0 – 5.5; **Fig 1**, part 2, outer & inner CV loops) of .RData files can be generated through scripts on Github (https://github.com/mematt/ml4calibrated450k/blob/master/data/subfunction_load_subset_filter_match_betasKk.R) or are directly downloadable (~5.3Gb) from our Dropbox (http://bit.ly/2vBg8yc). For details on how to prepare the 450K DNA methylation tumor samples from The Cancer Genome Atlas, please see the GitHub repository (https://github.com/mwsill/mnp_training/blob/master/tsne.R) and to download the source data visit the NCI GDC Legacy Archive (https://gdc-portal.nci.nih.gov/legacy-archive). The the combined TCGA cohort with vRF+MR predictions is available as .xlsx file (Supplementary dataset).

Table 1 | Glossary

Term	Abbreviation	Definition
1-vs-1		Coupling approach to tackle multiclass problems. Classes are compared in a 1-against-1 fashion, particularly for support vector machines in the LiblineaR and e1071 packages.
1-vs-all		Coupling approach that uses 1-against-the-rest comparison, which is applied by the GPU-accelerated support vector machine implementation in Rgtsvm package.
AUC	AUC	Multiclass extension of the area under the Receiver Operating Characteristics curve (AUC) measure as defined by Hand and Till (2001)
Batch effect		A source variation that occurs because measurements are influenced by laboratory or technical conditions. This can be crucial and result in biased conclusions when batch effects are confounded with one or more outcomes.
Batch adjustment		If not corrected by suitable numerical algorithms, batch effects may distort subsequent analyses by increasing variability and covering or confounding the real biological signal[122,123]. There are various computational solutions to counteract batch effects such as ComBat, Surrogate Variable Analysis, and Functional normalization[105,122-124].
Benchmark		Is the method to compare the performance of computer programs or calculations using various settings or on different hardware.
Brier score	BS	BS is a proper score function that was proposed by Glenn Brier in 1950 – initially to assess weather forecasts in terms of probability[114].
Calibrator / Post-processing algorithm		An algorithm that is applied to the output of a classifier to achieve better calibrated probability estimates such as Platt scaling using logistic regression, Firth’s penalized regression or multinomial ridge penalized regression.
Calibrated probabilities		Probability estimates of machine learning classifiers that were post-processed (i.e. calibrated) using an additional algorithm such as Platt scaling.
Classifier		Any collection of prediction rules learned by an ML-algorithm that provide class estimates. For the sake of simplicity, we call all investigated ML-algorithm and statistical models (RF, ELNET, SVM and XGBoost) a classifier.
Cross-Validation	CV	During CV the data set is randomly partitioned into complementary subsets then the analysis is performed on one subset (training set) and validated on the remaining subset (called the validation- or test set). To estimate variability of the model's predictive performance, usually multiple rounds of CV are carried out and the validation results are averaged over these rounds.
DNA methylation		DNA methylation is one of the principal mechanisms of epigenetic modifications that enzymatically adds methyl groups to the DNA[4].
Elastic net	ELNET	To consider the effect of many strongly correlated predictor variables Zou and Hastie [34] proposed the elastic net penalty to generalized linear models by combining the lasso (L1) and ridge (L2) penalties. Thus, ELNET can concurrently perform variable selection and shrink coefficients of correlated variables.
Feature selection		Is method of machine learning algorithms to select the subset of relevant predictors/features to create robust classifiers.
Firth’s penalized regression	FLR	Firth proposed a solution for the problem of separation in regular models by removing the first order bias term and replacing it with a penalized likelihood function using the Jeffreys invariant prior[45].
Graphical processing unit	GPU	A type of hardware within the computer specifically designed to accelerate the creation of images.
Generalized linear model via penalized maximum likelihood	GLMNET	Glmnet is a package that fits a generalized linear model via penalized maximum likelihood[67]. The regularization path is computed for the lasso or elastic net penalty at a grid of values for the regularization parameter lambda (λ).
Information leakage		Is when information outside the training set is provided (“leaked”) to the model. Hence, the model can learn or know some additional information. This can result in overly optimistic or invalid predictive models.
Logarithmic loss	LL	Multiclass formulation of the logarithmic loss metric.
Logistic regression	LR	Is a widely used statistical model, which is used to model relationships between the binary response variable and multiple explanatory variables[82].
Machine learning	ML	ML is a way to meaningfully process data using algorithms and statistical models to perform specific (un- or supervised) tasks without hardcoding explicit instructions.
Misclassification error	ME	Is defined by the proportion of incorrectly classified cases over all classes divided by the total number of cases.
Multinomial ridge-penalized regression	MR	This is the multinomial special case of the generalized elastic net family with mixing parameter $\alpha = 0$ [9,67,69].
Nested cross-validation		Nested CV is an extension of CV so that each initial partition of the data set (outerfold) containing the training set is further partitioned into a nested (innerfold) training- and validation set (**Fig** 1. part 2). It is particularly suitable to train a model in which hyperparameters also need to be optimized or to train a secondary model for e.g. calibration. The combination of the innerfold validation (i.e. calibration) sets is used for for training the post-processing algorithm.
Overfitting		Is the production of an analysis that corresponds too closely to a particular set of data and fails to fit or predict future data sets reliably. The available data points have been overly used to optimize the decision boundary and to extract information from noise. Hence, an overfitted classifier does not generalise well to other data sets.
Platt scaling		Is a post-processing approach to transform raw classifier scores into a probability distribution over classes. The method was proposed by John Platt originally for support vector machines but can be applied to any classifier output [24].
Random forests	RF	Are an ensemble method of bootstrap aggregated (bagged) binary classification trees, which were proposed by Leo Breiman in 2001[29].
Raw scores		Uncalibrated (“raw”) score outputs of machine learning classifiers.
Shrinkage methods		Is a subset of selection methods that fit models containing all p predictors and apply techniques that constrain or regularize the coefficient estimates by shrinking them towards zero[117].
Support vector machines	SVM	Is a supervised learning algorithm that was first introduced for binary classification by Cortes and Vapnik in 1995. SVM try to find the optimal separating hyperplane with the largest margin between two classes [30]. They are very popular for $p >> n$ problems, especially for genomic data analysis[9,31,32].
The Cancer Genome Atlas	TCGA	A landmark cancer genomics program that molecularly characterized over 20,000 primary cancer and matched normal samples spanning 33 cancer types as a joint initiative between the National Cancer Institute and the National Human Genome Research Institute.
Workflow		We refer to the combination of a machine learning classifier and the consecutive post-processing with a calibrator algorithm as a workflow.
(E)xtreme Gradient Boosting	XGBoost	It is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable [80,81].

Table 2 | Summary table of performance measures of tested classifier algorithms and calibrated workflows.

Workflow	Top 10 BS	Classifier	Run-time 5x5 CV (/fold)	# of CPU threads [hardware]	R package	Calibrator	Optimized Metric	Hyperparameters	ME	AUC	BS	LL
vRF		RF	38 min	1 [2]	randomForest	raw	ME	ntree =500, mtry=100	0.048	0.999	0.320	0.780
vRF + LR		RF	+30 s [LR]	1 [2]	randomForest	Platt LR (us)	ME	$p_{\text{varsel}} = 200$	0.052	-	0.106	0.289
vRF + LR		RF	"	"	randomForest	Platt LR	ME	"	0.052	0.994	0.081	0.262
vRF + FLR		RF	+8-9 min [FLR]	1 [2]	randomForest	Platt Firth (us)	ME	"	0.048	-	0.105	0.193
vRF + FLR		RF	"	"	randomForest	Platt Firth	ME	"	0.048	0.999	0.081	0.193
vRF + MR	10	RF	+7-8 min [MR]	11 [2]	randomForest	MR	ME	"	0.043	0.999	0.073	0.155
tRF_{BS}		RF	12-13 h (16-25 min/fold)	72 [5]	randomForest	raw	BS	ntree = [500, 1000, 1500, 2000]	0.055	0.999	0.272	0.673
tRF_{ME}		RF			randomForest	raw	ME	mtry = [80, 90, 100, 110]	0.035	0.999	0.351	0.855
tRF_{LL}		RF			randomForest	raw	LL	p_{varsel} = [100, 200, 500, 1000,	0.055	0.999	0.273	0.672
tRF_{BS}+LR		RF	+30 s [LR]	1 [2]	randomForest	Platt LR	BS	, 2000, 5000, 7500, 10000]	0.056	0.997	0.086	0.266
tRF_{ME}+LR	9	RF	"	"	randomForest	Platt LR	ME	nodesize = 1	0.042	0.998	0.062	0.156
tRF_{LL} +LR		RF	"	"	randomForest	Platt LR	LL	"	0.058	0.995	0.089	0.291
tRF_{BS} +FLR		RF	+8-9 min [FLR]	1 [2]	randomForest	Platt Firth	BS	"	0.054	0.997	0.086	0.194
tRF_{ME} +FLR	8	RF	"	"	randomForest	Platt Firth	ME	"	0.037	0.999	0.062	0.150
tRF_{LL} +FLR		RF	"	"	randomForest	Platt Firth	LL	"	0.056	0.999	0.089	0.205
tRF_{BS} +MR		RF	+7-8 min [MR]	11 [2]	randomForest	MR	BS	"	0.051	0.997	0.082	0.176
tRF_{ME} +MR	4	RF	"	"	randomForest	MR	ME	"	0.027	0.999	0.046	0.095
tRF_{LL} +MR		RF	"	"	randomForest	MR	LL	"	0.055	0.999	0.086	0.188
ELNET (1k)	7	ELNET	~7.5 h (12-15 min/fold)	31 [4]	glmnet	raw	ME	$\alpha = 0 \mid 0.025 ; \lambda = [0.0010-0.0036]$	0.032	0.999	0.059	0.131
ELNET (10k)	5	ELNET	~72 h (2-2.25 h/fold)	31 [4]	glmnet	raw	ME	$\alpha = 0 ; \lambda = [0.012-0.038]$	0.027	0.999	0.048	0.109
SVM-LK		SVM	~28 h (50-70 min/fold)	11 [3]	e1071	raw	ME	$C = 0.001 \mid 0.01$	0.032	0.999	0.372	0.978
SVM-LK+LR	2	SVM	+30 s [LR]	1 [2]	e1071	Platt LR	ME	"	0.025	0.999	0.043	0.112
SVM-LK+FLR	3	SVM	+8-9 min [FLR]	1 [2]	e1071	Platt Firth	ME	"	0.021	0.999	0.044	0.135
SVM-LK+MR	1	SVM	+7-8 min [MR]	11 [2]	e1071	MR	ME	"	0.021	0.999	0.039	0.085
SVM-LK (GPU)	6	SVM	~5 h	1080Ti	Rgtsvm-GPU	global softmax	ME	$C = 0.01 \mid 0.001$; n.SV= 1300-1600	0.033	0.998	0.056	0.144
SVM-CS[6]		SVM	~6 h (13-15 min/fold)	7 [1]	LiblineaR	-	ME	$C \geq 0.001$	0.028	-	-	-
XGBoost		BT	~65-70 h (110-130 min/fold)	72 [5]	xgboost	raw	ME	Table 3 & 4.	0.051	0.999	0.150	0.430
XGBoost+LR		BT	+30 s [LR]	1 [2]	xgboost	Platt LR	ME	"	0.055	0.991	0.087	0.452
XGBoost+FLR		BT	+8-9 min [FLR]	1 [2]	xgboost	Platt Firth	ME	"	0.053	0.993	0.089	0.384
XGBoost+MR		BT	+7-8 min [MR]	11 [2]	xgboost	MR	ME	"	0.046	0.999	0.092	0.247

AUC: multiclass area under the ROC after Hand and Till (that can only be calculated if probabilities are scaled to 1), us: unscaled, rowsum $\neq 1$, BS: Brier score, ME: misclassification error, LL: multiclass log loss, vRF and tRF: vanilla- and tuned random forests, LR: logistic regression, FLR: Firth-penalized LR with 10 000 iterations; MR: multinomial ridge penalized LR; ELNET: elastic net penalized multinomial logistic regression, 1k and 10k: 1000 or 10000 most variable CpG probes, SVM: support vector machines, LK: linear kernel SVM, CS: Crammer and Singer (type 4) without probability output, n.SV: number of support vectors; XGBoost: extreme gradient boosting using trees as base learners, BT: boosted trees. Used [hardware]: i) CPU: [1] 8 threads on i7 7700k @ 4.2GHz ; [2] 12 threads on MacBook Pro 15" i9-8950HK @ 2.9 GHz or [3] i7-6850k @ 3.6 Ghz; [4] 32 threads on i9-7960X @ 2.8 Ghz; [5] 72 threads on AWS EC2 c5n.18xlarge @ 3.5Ghz; ii) GPU: NVIDIA GTX 1080Ti.

Table 3 | Combination of investigated XGBoost hyperparameter settings

Booster parameter	Parameter description	Default	Tested combination of settings
`nrounds`	number of iterations (equivalent to ntree of RF)	100	100, 150, 200
`max_depth`	maximum depth of a tree	6	2, 3, 4, 5, 6, 8, 10
`eta`	learning rate	0.3	0.05, 0.1, 0.3
`gamma`	minimum loss reduction required to make a further tree partition	0	0, 0.001, 0.01, 0.05, 0.1
`colsample_bytree`	subsample ratio of columns when constructing each tree	1	0.01, 0.02, 0.05, 0.1, 0.2, 0.4, 0.5, 0.6, 0.8, 1
`min_child_weight`	minimum sum of instance weight (hessian) needed in a child	1	1, 2
`subsample`	subsample ratio of the training instance	1	(0.632), 1

Parameter settings were compared on subfold 1.1 (n = 1720 cases). `subsample` = 0.632 is the special case to mimic RF's bootstrapping. For further details of the listed booster parameters see the methods section or consult the XGboost help.

Table 4 | Characteristics of best performing XGboost models on prototyping subfold 1.1 (n=1720)

Error rate	$n_{\text{best\_iter}}$	`max_depth`	`eta`	`gamma`	`colsample_bytree`
0.045	62	6	0.1	0	0.01 (100)
0.052	90	6	0.1	0.01	0.02 (200)
0.054	73	6	0.1	0	0.05 (500)
0.066	107	6	0.1	0	0.2 (2000)

`min_child_weight` and `subsample` were left at default = 1. $n_{\text{best\_iter}}$ is the optimal `nrounds/ntrees` achieving the highest accuracy found within the default `nrounds=100` range using the built-in `watchlist` functionality of xgboost.

Table 5 | XGboost hyperparameter combinations chosen in extra nested 3-fold CV on outerfolds 1.0 -5.0

Outerfold	Error rate	$n_{\text{best\_iter}}$	`max_depth`	`eta`	`gamma`	`colsample_bytree`
-	-	-	6	0.1	0	0.01 (100)
2.0	0.071	41	6	0.1	0	0.02 (200)
3.0	0.051	66	6	0.1	0	0.02 (200)
5.0	0.029	95	6	0.1	0	0.02 (200)
1.0	0.050	90	6	0.1	0.01	0.02 (200)
4.0	0.052	73	6	0.1	0	0.05 (500)
-	-	-	6	0.1	0	0.2 (2000)

`min_child_weight` and `subsample` were left at default = 1. $n_{\text{best\_iter}}$ is the optimal `nrounds/ntrees` achieving the highest accuracy found within the default `nrounds=100` range using the built-in `watchlist` functionality of xgboost.

Table 6 | Troubleshooting table

Step	Problem	Possible reason	Solution
4	Objects (`y`, `nfolds`) are not found.	Incorrect path or missing files.	Check path; Download from GitHub (https://github.com/mematt/ml4calibrated450k/tree/master/data).
6-9	Error messages during package installation: `package 'foo' is not available (for R version x.y.z)`	Typo in code; R or Bioconductor is out of date.	Check spelling, Check `?setRepositories`, Update R and/or Bioconductor.
10, 15	Error messages displayed; the program stops	Missing performance metrics (typical for tRF)	Load the script containing the evaluation metrics: `source("evaluation_metrics.R")`. Check in the R- or RStudio console whether the package `HandTill2001` is installed and loaded into the global environment (`.GlobalEnv`).
10	Error messages displayed; the program stops	Any ML-classifier (`run_nestedcv_<”ML-algorithm”>`): a custom matrix object was provided for `betas.. = NULL`	`betas..=` argument requires the pre-variance filtered *.RData* files with `betas.train-betas.test` matrix pairs (Step 5). This data structure can be generated using the `subfunction_load_subset_filter_match_betasKk()` function in the identically named .R script (https://github.com/mematt/ml4calibrated450k/blob/master/data/).
11-14	`Error in gzfile(file, “wb”): cannot open the connection.` … `cannot open compressed file` `` `./…/.RData` `` …	Incorrect loading path or file name combination. `` `No such file or directory` ``	Check the folder path argument `load.path.w.name = "./tRF/CVfold."`. Please note that the “dot” at the end of `CVfold.` is required for creating correct file.path.

References

1 Capper, D. *et al.* DNA methylation-based classification of central nervous system tumours. *Nature* **555**, 469, doi:10.1038/nature26000 https://www.nature.com/articles/nature26000#supplementary-information (2018).

2 Capper, D. *et al.* Practical implementation of DNA methylation and copy-number-based CNS tumor diagnostics: the Heidelberg experience. *Acta neuropathologica* **136**, 181-210, doi:https://doi.org/10.1007/s00401-018-1879-y (2018).

3 Heyn, H. & Esteller, M. DNA methylation profiling in the clinic: applications and challenges. *Nature Reviews Genetics* **13**, 679, doi:10.1038/nrg3270 (2012).

4 Rodríguez-Paredes, M. & Esteller, M. Cancer epigenetics reaches mainstream oncology. *Nature Medicine* **17**, 330-339, doi:10.1038/nm.2305 (2011).

5 Sturm, D. *et al.* New brain tumor entities emerge from molecular classification of CNS-PNETs. *Cell* **164**, 1060-1072 (2016).

6 Sharma, T. *et al.* Second-generation molecular subgrouping of medulloblastoma: an international meta-analysis of Group 3 and Group 4 subtypes. *Acta Neuropathologica*, doi:10.1007/s00401-019-02020-0 (2019).

7 Baek, S., Tsai, C.-A. & Chen, J. J. Development of biomarker classifiers from high-dimensional data. *Briefings in bioinformatics* **10**, 537-546 (2009).

8 Dupuy, A. & Simon, R. M. Critical review of published microarray studies for cancer outcome and guidelines on statistical analysis and reporting. *Journal of the National Cancer Institute* **99**, 147-157 (2007).

9 Hastie, T., Tibshirani, R. & Friedman, J. *The elements of statistical learning: Data Mining, Inference and Prediction*. 2. edn, (Springer, New York, NY, 2009).

10 Lee, J. W., Lee, J. B., Park, M. & Song, S. H. An extensive comparison of recent classification tools applied to microarray data. *Computational Statistics & Data Analysis* **48**, 869-885, doi:https://doi.org/10.1016/j.csda.2004.03.017 (2005).

11 Simon, R. Roadmap for developing and validating therapeutically relevant genomic classifiers. *Journal of Clinical Oncology* **23**, 7332-7341 (2005).

12 Hoadley, K. A. *et al.* Cell-of-origin patterns dominate the molecular classification of 10,000 tumors from 33 types of cancer. *Cell* **173**, 291-304 (2018).

13 Fernandez, A. F. *et al.* A DNA methylation fingerprint of 1628 human samples. *Genome research* **22**, 407-419 (2012).

14 Wiestler, B. *et al.* Assessing CpG island methylator phenotype, 1p/19q codeletion, and MGMT promoter methylation from epigenome-wide data in the biomarker cohort of the NOA-04 trial. *Neuro-oncology* **16**, 1630-1638 (2014).

15 Aryee, M. J. *et al.* Minfi: a flexible and comprehensive Bioconductor package for the analysis of Infinium DNA methylation microarrays. *Bioinformatics* **30**, 1363-1369 (2014).

16 Weinhold, L., Wahl, S., Pechlivanis, S., Hoffmann, P. & Schmid, M. A statistical model for the analysis of beta values in DNA methylation studies. *BMC bioinformatics* **17**, 480 (2016).

17 Appel, I. J., Gronwald, W. & Spang, R. Estimating classification probabilities in high-dimensional diagnostic studies. *Bioinformatics* **27**, 2563-2570 (2011).

18 Kuhn, M. & Johnson, K. *Applied Predictive Modeling*. (Springer Science+Business Media, New York, 2013).

19 Simon, R. Development and validation of biomarker classifiers for treatment selection. *Journal of statistical planning and inference* **138**, 308-320 (2008).

20 Simon, R. Class probability estimation for medical studies. *Biometrical Journal* **56**, 597-600, doi: https://doi.org/10.1002/bimj.201300296 (2014).

21 Dankowski, T. & Ziegler, A. Calibrating random forests for probability estimation. *Statistics in medicine* (2016).

22 Boström, H. Calibrating random forests. in *Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on.* 121-126 (IEEE, 2008).

23 Kruppa, J. *et al.* Probability estimation with machine learning methods for dichotomous and multicategory outcome: theory. *Biometrical Journal* **56**, 534-563 (2014).

24 Platt, J. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers* **10**, 61-74 (1999).

25 Hastie, T. & Tibshirani, R. Classification by pairwise coupling. in *Advances in neural information processing systems.* Vol. **10** 507-513 (MIT Press).

26 Kruppa, J. *et al.* Probability estimation with machine learning methods for dichotomous and multicategory outcome: Applications. *Biometrical Journal* **56**, 564-583 (2014).

27 Wu, T.-F., Lin, C.-J. & Weng, R. C. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research* **5**, 975-1005 (2004).

28 Gurovich, Y. *et al.* Identifying facial phenotypes of genetic disorders using deep learning. *Nat Med* **25**, 60-64, doi:10.1038/s41591-018-0279-0 (2019).

29 Breiman, L. Random forests. *Machine learning* **45**, 5-32 (2001).

30 Cortes, C. & Vapnik, V. Support-vector networks. *Machine learning* **20**, 273-297 (1995).

31 Efron, B. & Hastie, T. *Computer Age Statistical Inference*. Vol. **5** (Cambridge University Press, 2016).

32 Wang, X., Xing, E. P. & Schaid, D. J. Kernel methods for large-scale genomic data analysis. *Briefings in bioinformatics* **16**, 183-192 (2014).

33 Zhuang, J., Widschwendter, M. & Teschendorff, A. E. A comparison of feature selection and classification methods in DNA methylation studies using the Illumina Infinium platform. *BMC bioinformatics* **13**, 59 (2012).

34 Zou, H. & Hastie, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **67**, 301-320 (2005).

35 Freund, Y. & Schapire, R. E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences* **55**, 119-139, doi:https://doi.org/10.1006/jcss.1997.1504 (1997).

36 Schapire, R. E. Using output codes to boost multiclass learning problems. in *ICML.* Vol. **97** 313-321.

37 Chen, T. & He, T. Higgs boson discovery with boosted trees. in *NIPS 2014 Workshop on High-energy Physics and Machine Learning.* 69-80.

38 He, X. *et al.* Practical lessons from predicting clicks on ads at facebook. in *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising (ADKDD'14).* 1-9 (ACM).

39 Caruana, R. & Niculescu-Mizil, A. An empirical comparison of supervised learning algorithms. in *Proceedings of the 23rd international conference on Machine learning.* 161-168.

40 Niculescu-Mizil, A. & Caruana, R. Predicting good probabilities with supervised learning. in *Proceedings of the 22nd international conference on Machine learning.* 625-632.

41 Niculescu-Mizil, A. & Caruana, R. Obtaining Calibrated Probabilities from Boosting. in *UAI.* 413.

42 Van Calster, B. *et al.* Comparing Methods for Multi-class Probabilities in Medical Decision Making Using LS-SVMs and Kernel Logistic Regression. in *Artificial Neural Networks -- ICANN 2007.* (eds Joaquim Marques de S\'a, Lu 'i s A. Alexandre, W. l odzis l aw Duch, & Danilo Mandic) 139-148 (Springer Berlin Heidelberg).

43 Zadrozny, B. & Elkan, C. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. in *ICML.* Vol. **1** 609-616.

44 Zadrozny, B. & Elkan, C. Transforming classifier scores into accurate multiclass probability estimates. in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining.* 694-699.

45 Firth, D. Bias reduction of maximum likelihood estimates. *Biometrika* **80**, 27-38 (1993).

46 Lafzi, A., Moutinho, C., Picelli, S. & Heyn, H. Tutorial: guidelines for the experimental design of single-cell RNA sequencing studies. *Nat Protoc* **13**, 2742-2757, doi:10.1038/s41596-018-0073-y (2018).

47 Rajkomar, A., Dean, J. & Kohane, I. Machine Learning in Medicine. *N Engl J Med* **380**, 1347-1358, doi:10.1056/NEJMra1814259 (2019).

48 Ramaswamy, S. *et al.* Multiclass cancer diagnosis using tumor gene expression signatures. *Proceedings of the National Academy of Sciences* **98**, 15149-15154 (2001).

49 Kickingereder, P. *et al.* Radiogenomics of glioblastoma: machine learning–based classification of molecular characteristics by using multiparametric and multiregional MR imaging features. *Radiology* **281**, 907-918 (2016).

50 Radovic, A. *et al.* Machine learning at the energy and intensity frontiers of particle physics. *Nature* **560**, 41-48, doi:10.1038/s41586-018-0361-2 (2018).

51 Butler, K. T., Davies, D. W., Cartwright, H., Isayev, O. & Walsh, A. Machine learning for molecular and materials science. *Nature* **559**, 547-555, doi:10.1038/s41586-018-0337-2 (2018).

52 Wiestler, B. *et al.* Integrated DNA methylation and copy-number profiling identify three clinically and biologically relevant groups of anaplastic glioma. *Acta neuropathologica* **128**, 561-571 (2014).

53 Ritchie, M. E. *et al.* limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic acids research* **43**, e47-e47 (2015).

54 Bourgon, R., Gentleman, R. & Huber, W. Independent filtering increases detection power for high-throughput experiments. *Proceedings of the National Academy of Sciences* **107**, 9546-9551 (2010).

55 Breiman, L. & Spector, P. Submodel selection and evaluation in regression. The X-random case. *International statistical review/revue internationale de Statistique*, 291-319 (1992).

56 Kohavi, R. & others. A study of cross-validation and bootstrap for accuracy estimation and model selection. in *Ijcai.* Vol. **14** 1137-1145.

57 Krijthe, J. H. Rtsne: T-distributed stochastic neighbor embedding using Barnes-Hut implementation. *R package version 0.15* (2015).

58 Maaten, L. v. d. & Hinton, G. Visualizing data using t-SNE. *Journal of machine learning research* **9**, 2579-2605 (2008).

59 Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. in *Kdd.* Vol. **96** 226-231.

60 Breiman, L., Friedman, J., Stone, C. & Olshen, R. Classification and regression trees. Florida (Chapman and Hall/CRC press, 1984).

61 Liaw, A. & Wiener, M. Classification and regression by randomForest. *R news* **2**, 18-22 (2002).

62 Kuhn, M. Caret package. *Journal of Statistical Software* **28**, 1-26 (2008).

63 Kruppa, J., Schwarz, A., Arminger, G. & Ziegler, A. Consumer credit risk: Individual probability estimates using machine learning. *Expert Systems with Applications* **40**, 5125-5131 (2013).

64 Malley, J. D., Kruppa, J., Dasgupta, A., Malley, K. G. & Ziegler, A. Probability machines: consistent probability estimation using nonparametric learning machines. *Methods of Information in Medicine* **51**, 74 (2012).

65 Strobl, C., Boulesteix, A.-L., Zeileis, A. & Hothorn, T. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC bioinformatics* **8**, 1 (2007).

66 Chen, C., Liaw, A. & Breiman, L. Using random forest to learn imbalanced data. *University of California, Berkeley* **110** (2004).

67 Friedman, J., Hastie, T. & Tibshirani, R. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software* **33**, 1, doi:10.18637/jss.v033.i01 (2010).

68 Zou, H. & Hastie, T. Regression shrinkage and selection via the elastic net, with applications to microarrays. *JR Stat Soc Ser B* **67**, 301-320 (2003).

69 Hastie, T. & Qian, J. Glmnet vignette. (2016).

70 Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267-288 (1996).

71 Chang, C.-C. & Lin, C.-J. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* **2**, 27:21-27:27 (2011).

72 e1071: Misc Functions of the Department of Statistics, Probability
Theory Group (Formerly: E1071), TU Wien v. R package version 1.7-1 (The Comprehensive R Archive Network, 2019).

73 Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R. & Lin, C.-J. LIBLINEAR: A library for large linear classification. *Journal of machine learning research* **9**, 1871-1874 (2008).

74 LiblineaR: Linear predictive models based on the LIBLINEAR C/C++ Library v. R package version 2.10-8 (2017).

75 Wang, Z., Chu, T., Choate, L. A. & Danko, C. G. Rgtsvm: Support Vector Machines on a GPU in R. *arXiv preprint arXiv:1706.05544* (2017).

76 Crammer, K. & Singer, Y. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research* **2**, 265-292 (2001).

77 Milgram, J., Cheriet, M. & Sabourin, R. Estimating accurate multi-class probabilities with support vector machines. in *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on.* Vol. **3** 1906-1911.

78 Hastie, T., Rosset, S., Tibshirani, R. & Zhu, J. The entire regularization path for the support vector machine. *Journal of Machine Learning Research* **5**, 1391-1415 (2004).

79 Hsu, C.-W., Chang, C.-C. & Lin, C.-J. A practical guide to support vector machines. *Department of Computer Science \& Information Engineering, National Taiwan University* (2003).

80 Chen, T. & He, T. Xgboost: extreme gradient boosting. *R package version 0.4-2*, doi: http://dx.doi.org/10.1145/2939672.2939785 (2016).

81 Chen, T., He, T., Benesty, M., Khotilovich, V. & Tang, Y. XGBoost - Introduction to Boosted Trees (Online XGBoost - Read the docs, 2017).

82 Dobson, A. J. & Barnett, A. *An introduction to generalized linear models*. (CRC press, 2008).

83 R: A Language and Environment for Statistical Computing (R Foundation for Statistical Computing, Vienna, Austria, 2017).

84 Geroldinger, A., Dunkler, D. & Puhr, R. Accurate Prediction of Rare Events with Firth's Penalized Likelihood Approach. (2015).

85 Puhr, R., Heinze, G., Nold, M., Lusa, L. & Geroldinger, A. Firth's logistic regression with rare events: accurate effect estimates and predictions? *Statistics in Medicine* (2017).

86 Heinze, G. & Schemper, M. A solution to the problem of separation in logistic regression. *Statistics in medicine* **21**, 2409-2419 (2002).

87 Kosmidis, I. brglm: Bias reduction in generalized linear models. in *The R User Conference, useR! 2011 August 16-18 2011 University of Warwick, Coventry, UK.* 111.

88 Shen, J. & Gao, S. A solution to separation and multicollinearity in multiple logistic regression. *Journal of data science: JDS* **6**, 515 (2008).

89 Zhao, S. D., Parmigiani, G., Huttenhower, C. & Waldron, L. Más-o-menos: a simple sign averaging method for discrimination in genomic data analysis. *Bioinformatics* **30**, 3062-3069 (2014).

90 Donoho, D. L. & Ghorbani, B. Optimal Covariance Estimation for Condition Number Loss in the Spiked Model. *arXiv preprint arXiv:1810.07403* (2018).

91 Agrawal, A., Viktor, H. L. & Paquet, E. SCUT: Multi-class imbalanced data classification using SMOTE and cluster-based undersampling. in *Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K), 2015 7th International Joint Conference on.* Vol. **1** 226-234 (IEEE).

92 Bischl, B. *et al.* mlr: Machine Learning in R. *Journal of Machine Learning Research* **17**, 1-5 (2016).

93 Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* **16**, 321-357 (2002).

94 Lunardon, N., Menardi, G. & Torelli, N. ROSE: A Package for Binary Imbalanced Learning. *R journal* **6** (2014).

95 Menardi, G. & Torelli, N. Training and assessing classification rules with imbalanced data. *Data Mining and Knowledge Discovery* **28**, 92-122 (2014).

96 Hauskrecht, M., Pelikan, R., Valko, M. & Lyons-Weiler, J. Feature Selection and Dimensionality Reduction in Genomics and Proteomics in *Fundamentals of Data Mining in Genomics and Proteomics.* (eds W. Dubitzky, M. Granzow, & D. Berrar) 149-172 (Springer US, 2007).

97 Guyon, I., Weston, J., Barnhill, S. & Vapnik, V. Gene Selection for Cancer Classification using Support Vector Machines. *Machine Learning* **46**, 389-422, doi:10.1023/a:1012487302797 (2002).

98 Hastie, T., Tibshirani, R. & Friedman, J. High-Dimensional Problems: p N in *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 649-698 (Springer New York, 2009).

99 Huber, W. *et al.* Orchestrating high-throughput genomic analysis with Bioconductor. *Nature Methods* **12**, 115, doi:10.1038/nmeth.3252 (2015).

100 Assenov, Y. *et al.* Comprehensive analysis of DNA methylation data with RnBeads. *Nature Methods* **11**, 1138, doi:10.1038/nmeth.3115 https://www.nature.com/articles/nmeth.3115#supplementary-information (2014).

101 Morris, T. J. *et al.* ChAMP: 450k Chip Analysis Methylation Pipeline. *Bioinformatics* **30**, 428-430, doi:10.1093/bioinformatics/btt684 (2013).

102 Pidsley, R. *et al.* A data-driven approach to preprocessing Illumina 450K methylation array data. *BMC genomics* **14**, 293-293, doi:10.1186/1471-2164-14-293 (2013).

103 Horvath, S. DNA methylation age of human tissues and cell types. *J Genome Biology* **14**, 3156, doi:10.1186/gb-2013-14-10-r115 (2013).

104 Johann, P. D., Jäger, N., Pfister, S. M. & Sill, M. RF_Purify: a novel tool for comprehensive analysis of tumor-purity in methylation array data based on random forest regression. *BMC Bioinformatics* **20**, 428, doi:10.1186/s12859-019-3014-z (2019).

105 Leek, J., Johnson, W., Parker, H., Jaffe, A. & Storey, J. sva: Surrogate Variable Analysis R package version 3.10. 0 (2014).

106 Leek, J. T. & Storey, J. D. Capturing heterogeneity in gene expression studies by surrogate variable analysis. *PLoS genetics* **3**, e161 (2007).

107 Leek, J. T. & Storey, J. D. A general framework for multiple testing dependence. *Proceedings of the National Academy of Sciences* **105**, 18718-18723 (2008).

108 Anders, S. *et al.* Count-based differential expression analysis of RNA sequencing data using R and Bioconductor. *Nature protocols* **8**, 1765 (2013).

109 Huber, W. *et al.* Orchestrating high-throughput genomic analysis with Bioconductor. *Nature Methods* **12**, 115, doi:10.1038/nmeth.3252 (2015).

110 Pedregosa, F. *et al.* Scikit-learn: Machine learning in Python. *Journal of machine learning research* **12**, 2825-2830 (2011).

111 Hand, D. J. & Till, R. J. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine learning* **45**, 171-186 (2001).

112 Cullmann, A. D. HandTill2001: Multiple Class Area under ROC Curve. *R package* (2016).

113 Bickel, J. E. Some comparisons among quadratic, spherical, and logarithmic scoring rules. *Decision Analysis* **4**, 49-65 (2007).

114 Brier, G. W. Verification of forecasts expressed in terms of probability. *Monthly weather review* **78**, 1-3 (1950).

115 Friedman, D. An effective scoring rule for probability distributions. *UCLA Economics Working Papers* **164** (1979).

116 Gneiting, T. & Raftery, A. E. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association* **102**, 359-378 (2007).

117 James, G., Witten, D., Hastie, T. & Tibshirani, R. *An Introduction to Statistical Learning with Applications in R.* 1 edn, (Springer-Verlag New York, 2013).

118 Mitchell, R. & Frank, E. Accelerating the XGBoost algorithm using GPU computing. *PeerJ Computer Science* **3**, e127, doi:10.7717/peerj-cs.127 (2017).

119 Fischer, B., Pau, G. & Smith, M. rhdf5: HDF5 interface to R. *R# Package Version; RCoreTeam: Vienna, Austria*, doi:10.18129/B9.bioc.rhdf5 (2019).

120 Qiu, Y., Mei, J., Guennebaud, G. & Niesen, J. RSpectra: Solvers for Large Scale Eigenvalue and SVD Problems. *R package version 0.12-0* **405** (2016).

121 Crammer, K. & Singer, Y. On the learnability and design of output codes for multiclass problems. *Machine learning* **47**, 201-233 (2002).

122 Akulenko, R., Merl, M. & Helms, V. BEclear: Batch Effect Detection and Adjustment in DNA Methylation Data. *PLOS ONE* **11**, e0159921, doi:10.1371/journal.pone.0159921 (2016).

123 Price, E. M. & Robinson, W. P. Adjusting for Batch Effects in DNA Methylation Microarray Data, a Lesson Learned. *Front Genet* **9**, 83-83, doi:10.3389/fgene.2018.00083 (2018).

124 Leek, J. T. *et al.* Tackling the widespread and critical impact of batch effects in high-throughput data. *Nature Reviews Genetics* **11**, 733, doi:10.1038/nrg2825

https://www.nature.com/articles/nrg2825#supplementary-information (2010).

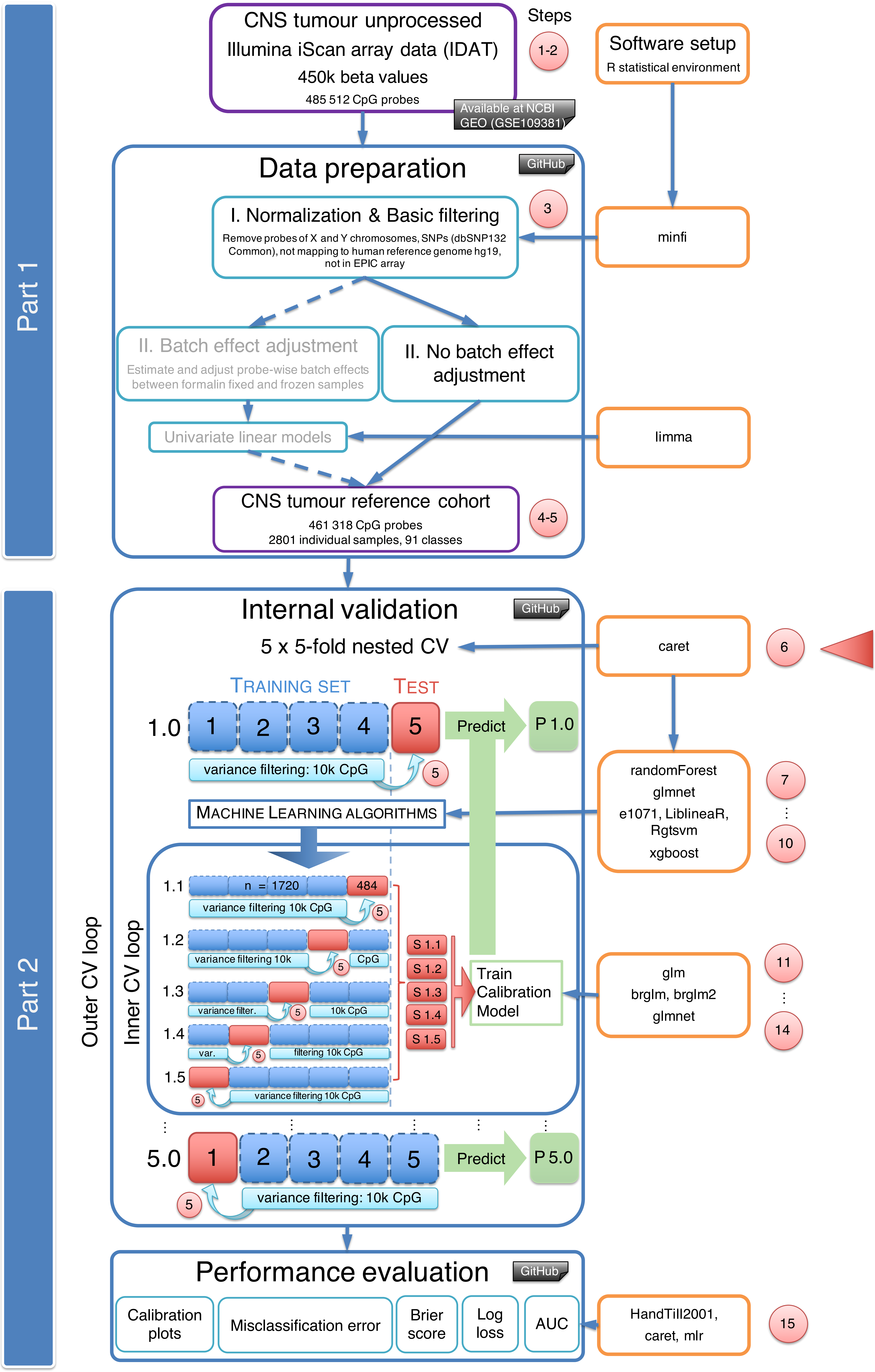
Figure legends

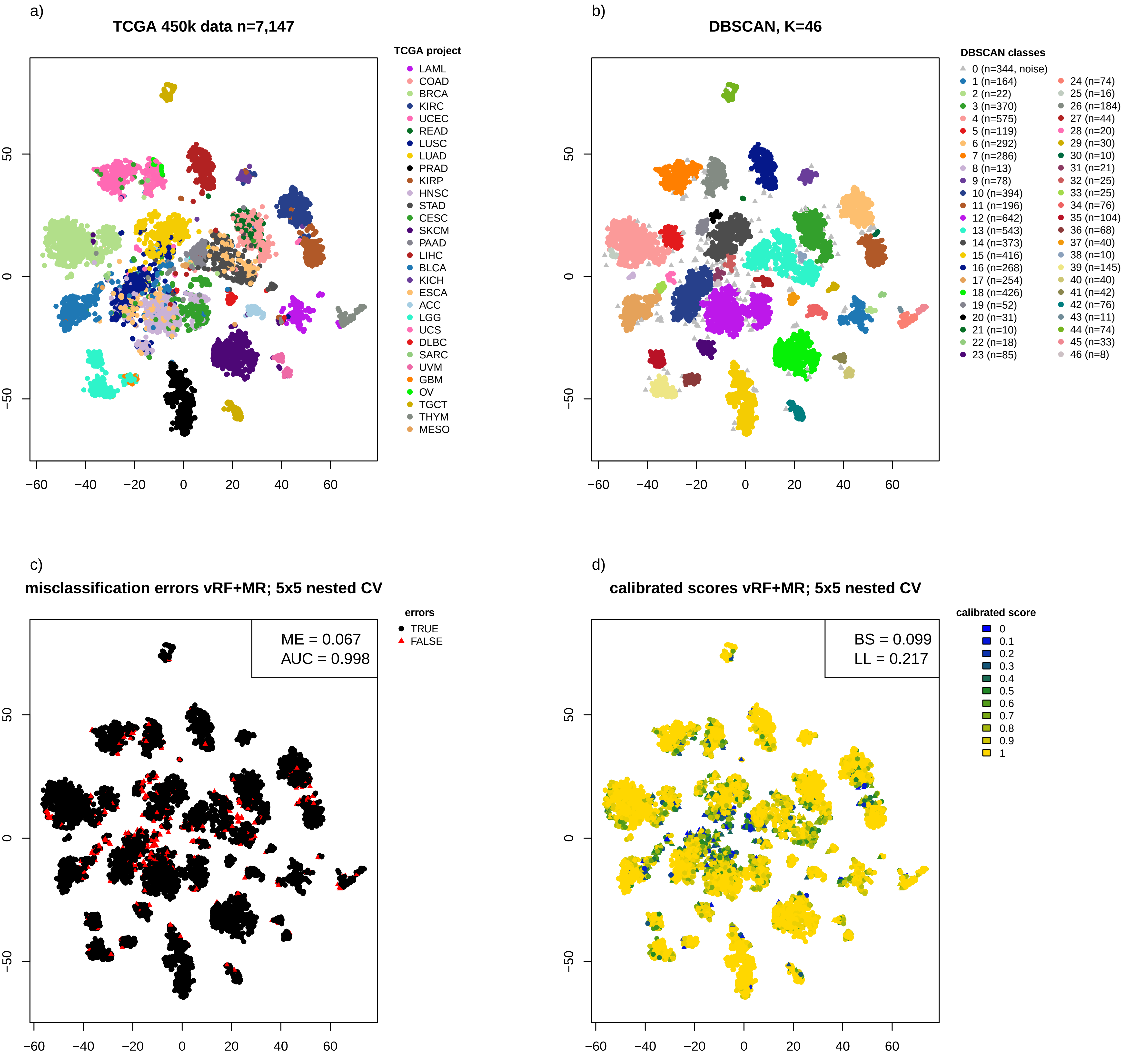
Figure-1 | Pipeline of methylation microarray data-based machine learning workflow development and comparison for well-calibrated personalized cancer diagnostics. Data preparation and pre-processing steps are described (Part 1) but our primary focus is on providing a blueprint for internal validation of algorithms using a nested resampling scheme and calibration (Part 2). The protocol is modular and provides flexible entry points at any step (red circles), however, the recommended entry is marked with a red triangle (step 6). Required R statistical programming packages are indicated in orange-framed boxes. Purple-framed rectangles indicate data objects like Illumina methylation microarray outputs that are used in the 5 x 5 nested cross-validation (CV) scheme to be fitted by machine learning algorithms. Rectangles with light blue frames represent processes: either data preprocessing methods such as normalization or batch effect adjustment, or performance evaluation metrics. Dark blue and red boxes in the outer CV loop represent the outerfold (1.0; 2.0; … ; 5.0) training and test sets respectively. The numbers "1, 2, …, 5" within these boxes indicate the fold identification numbers of the 5-fold CV. Similarly, in the inner CV loop (1.1; 1.2; … ; 1.5) , blue and red rectangles indicate the nested training and calibration sets. Light blue rectangles represent the unsupervised variance filtering (step 5) of the 10,000 most variable CpG probes that was performed on the respective training set of each 5 x 5-fold (n=30) ;the corresponding test/calibration set was subset (light blue arrows) accordingly in order to prevent information leakage. The collection of red rectangles (S 1.1 – 1.5) represent the combined calibration set of raw probability outputs ("raw scores") that is used for training (i.e. tuning) the calibrator algorithms (green-framed rectangle). Green arrows represent post-processing, that is the fitting of the tuned calibrator algorithm on the outerfold "raw scores", to generate calibrated probabilities (green boxes; P 1.0 – 5.0). Dark grey rectangles indicate data- or code available in NCBI GEO or in our GitHub repositories.

Figure-2 | External validation on various tumor types from the The Cancer Genome Atlas (TCGA). We generated the external validation cohort (n=7,147) by extracting and combining 450k DNA methylation microarray data from 30 different TCGA projects. First, an unsupervised variance filtering was performed to reduce the feature space to the 32k CpG probes with highest standard deviation across samples, followed by a principal component analysis (PCA). Then t-distributed stochastic neighbor embedding (t-SNE)[57,58] was applied to the first hundred PCs. Finally, the density-based algorithm for discovering clusters (DBSCAN) was applied to generate data-driven ("artificial") methylation tumour classes. The (**a**) image shows the clustering of these projects based on t-distributed stochastic neighbor embedding (t-SNE) [58]. The legend shows the color-coding and abbreviations of the 30 TCGA projects that were used to create the external validation cohort. (**b**) shows the results of DBSCAN that identified k=46 distinct clusters (i.e. classes) embedded within the combined TCGA validation cohort. Distribution of cases over the k=46 classes is presented in the legend on the right. Cases that could not be associated with a cluster by DBSCAN were defined as outliers or noise (light grey triangles, n=344). (**c**) shows the cases (red triangles) that were misclassified during 5-fold nested CV on the t-SNE clustered TCGA cases by the multinomial ridge calibrated vRF (vRF + MR). The overall metrics of this workflow were similar to those seen on BTMD (Table 2). (**d**) shows the distribution of vRF+MR-calibrated probabilities for each case on the same t-SNE clustered image on a diverging color-coded scale [0 – blue; 0.5 – green; 1 – yellow] with steps by 0.1 alongside the numerical performance metrics of BS and LL in the upper right corner.

Supplementary material legends

Supplementary dataset | Provides the compiled external validation DNA methylation data of the 30 combined TCGA projects comprised of altogether n=7,148 cases. The included variables (9 columns) are as follows: the TCGA sample IDs (TCGA_ID), the sentrix IDs (Sentrix_ID) from illumina sentrix arrays containing positional information that might be useful, the coordinates derived by the t-SNE algorithm (tsne1; tsne2), type of tissue sample (material), abbreviation of the respective TCGA project (TCGA_Project), the “artificial” target labels (i.e. cluster) created by the DBSCAN algorithm (DBSCAN_cluster), the maximal probability estimates (max_calibrated) and the corresponding class labels (predicted) generated by multinomial ridge calibrated vRF (vRF + MR), which was the fastest CPU workflow among the investigated ML-workflows.


Part 1
Part 2
Steps
CNS tumour unprocessed
Illumina iScan array data (IDAT)
450k beta values
485 512 CpG probes
Available at NCBI GEO (GSE109381)
1-2
Software setup
R statistical environment
Data preparation
GitHub
I. Normalization & Basic filtering
Remove probes of X and Y chromosomes, SNPs (dbSNP132 Common), not mapping to human reference genome hg19, not in EPIC array
3
minfi
II. Batch effect adjustment
Estimate and adjust probe-wise batch effects between formalin fixed and frozen samples
II. No batch effect adjustment
Univariate linear models
limma
CNS tumour reference cohort
461 318 CpG probes
2801 individual samples, 91 classes
4-5
Internal validation
GitHub
5 x 5-fold nested CV
caret
6
TRAINING SET
TEST
1.0
1
2
3
4
5
Predict
P 1.0
variance filtering: 10k CpG
5
MACHINE LEARNING ALGORITHMS
randomForest
glmnet
e1071, LiblineaR,
Rgtsvm
xgboost
7
⋮
10
Outer CV loop
Inner CV loop
1.1
n = 1720
484
variance filtering 10k CpG
5
1.2
variance filtering 10k
CpG
5
1.3
variance filter.
10k CpG
5
1.4
var.
filtering 10k CpG
5
1.5
5
variance filtering 10k CpG
S 1.1
S 1.2
S 1.3
S 1.4
S 1.5
Train
Calibration
Model
glm
brglm, brglm2
glmnet
11
⋮
14
⋮
5.0
1
2
3
4
5
Predict
P 5.0
5
variance filtering: 10k CpG
Performance evaluation
GitHub
Calibration plots
Misclassification error
Brier score
Log loss
AUC
HandTill2001,
caret, mlr
15


a)
TCGA 450k data n=7,147
TCGA project
LAML
COAD
BRCA
KIRC
UCEC
READ
LUSC
LUAD
PRAD
KIRP
HNSC
STAD
CESC
SKCM
PAAD
LIHC
BLCA
KICH
ESCA
ACC
LGG
UCS
DLBC
SARC
UVM
GBM
OV
TGCT
THYM
MESO
b)
DBSCAN, K=46
DBSCAN classes
0 (n=344, noise)
1 (n=164)
2 (n=22)
3 (n=370)
4 (n=575)
5 (n=119)
6 (n=292)
7 (n=286)
8 (n=13)
9 (n=78)
10 (n=394)
11 (n=196)
12 (n=642)
13 (n=543)
14 (n=373)
15 (n=416)
16 (n=268)
17 (n=254)
18 (n=426)
19 (n=52)
20 (n=31)
21 (n=10)
22 (n=18)
23 (n=85)
24 (n=74)
25 (n=16)
26 (n=184)
27 (n=44)
28 (n=20)
29 (n=30)
30 (n=10)
31 (n=21)
32 (n=25)
33 (n=25)
34 (n=76)
35 (n=104)
36 (n=68)
37 (n=40)
38 (n=10)
39 (n=145)
40 (n=40)
41 (n=42)
42 (n=76)
43 (n=11)
44 (n=74)
45 (n=33)
46 (n=8)
c)
misclassification errors vRF+MR; 5x5 nested CV
ME = 0.067
AUC = 0.998
errors
TRUE
FALSE
d)
calibrated scores vRF+MR; 5x5 nested CV
BS = 0.099
LL = 0.217
calibrated score
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
50
0
−50
−60
−40
−20
0
20
40
60