Abstract

We describe the basic steps for predicting the type of chemotherapy based on prior decisions by medical experts. The steps taken can be summarised as follows

- data cleaning; removal of Affimetrix marker genomes, removal of samples from adult patients (> 17 years)
- batch normalisation using the L/S method with standard normalisation with scaling by standard deviation
- probeset grouping per genome, mean value
- noise addition, 5%
- feature reduction using False Discovery Rate method with ANOVA for the distribution comparison, following the Benjamin-Hochberg procedure, with a maximum p-value of 0.05
- Repeated model generation; using several tree models, probabilistic models, neural networks and linear models

  weighted mean of the models based on model-wise accuracy to increase overall accuracy and robustness

  extraction of feature weights and importances from the models to identify genomic drivers

The target values of our predictor are based on treatment decisions by medical experts, hence we are predicting the decision that medical experts would make. This decision itself is likely based on a mixed set of predictors/features like, the chance of survival, the clinical situation of the patient and perhaps even the family wishes, which of course is not present in the data. Also, in this specific case the number of samples that actually contain labels is small.

Another approach is to use the overall survival as a proxy for the chemotherapy intensity. A reason to choose this approach would be that we have more samples available. The downside of this approach is that we are no longer predicting what the most suitable therapy is, but rather which patient, regardless of therapy will not survive. Also, the overall survival of a patient does not necessarily relate to the aggressiveness of the cancer but could be related to spurious clinical events, that may or may not be related to the chemotherapy.

Hence we have two training strategies to generate predictive models with a slightly different interpretation

- $\mathcal{I}$ train on all available chemotherapy labels and use cross-validation to get an estimation for the accuracy, test using cross-validation (with 10 folds)
- $\mathcal{II}$ train on the overall survival of Cohort 1, Cohort 2, IA, and JB patients and test on the $ALL - 10$ patients, i.e. survival/death is a proxy for low/high intensity treatment

Finally, apply the models to predict the HR/non-HR classification for the cohort 1 patients. The work flow is schematised in figure 1 and applies to both training scenarios.

1 Clinical data

(Age, gender, whitebloodcellcount) v. OS, and OS v. HR.

LR regressor for (age, gender, whitebloodcell count, pathways) vs OS and OS v. HR

The clinical data presents only a small portion of the available features per sample.

2 Pre-processing

2.1 Cohort-bias removal

For the cohort-bias removal we apply a genome-wise Location and scale (L/S) adjustment per cohort. Using a normalisation per cohort guarantees that the features have the same bounds over the cohorts and that the means are similar. The caveat of this approach is that we asume that the genome expression measurements are independent and we have no outliers. The standard normalisation transforms the genome expression values $\mathbf{x}$ per genome as follows

$$\mathbf{x}^* = \frac{\mathbf{x} - \overline{\mathbf{x}}}{\sigma}, \tag{1}$$

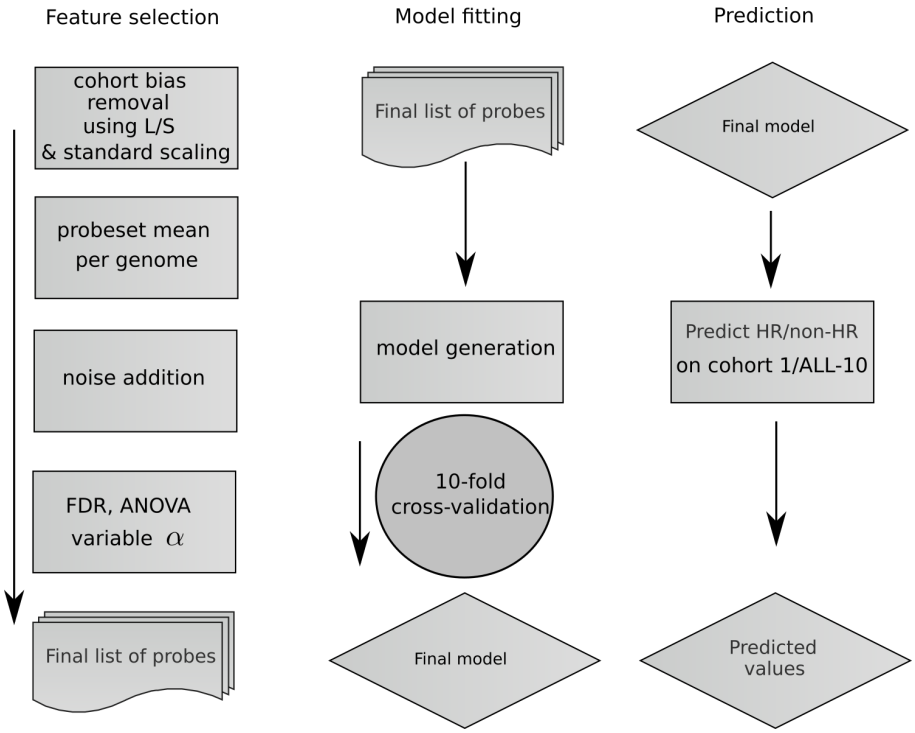


Figure 1: Workflow for the model generation and prediction

where $\mathbf{x}$ is the genome expression vector for some genome over all samples. This centers the mean and normalises the expression values with the standard deviation. To limit the influence of outliers we can center the median and use the interquantile range (IQR) for the scaling, i.e.

$$\mathbf{x}^* = \frac{\mathbf{x} - median\,(\mathbf{x})}{IQR}, \tag{2}$$

To demonstrate the effect of these transformations with regard to cohort bias we take two genomes, one with high and one with low variance over the classifications. There are various more elaborate methods

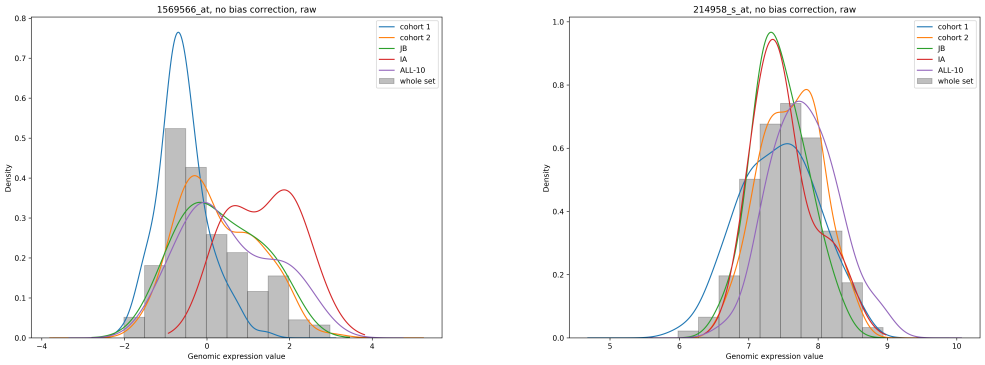


Figure 2: Two sets of distributions prior the bias correct, for, (left) a strong predictor and (right) a weak predictor

to remove bias such as the SVD-based method from Alter et al.[2], the PCA-based bias removal methods EIGENSTRAT by Price et al.[18], MANCIE by Zang et al.[24], the distance weighted discrimination (DWD) approach from Benito et al.[3] or the ComBat method by Johnson et al.[12] who apply an empirical Bayes approach. A comparison of bias removal methods is out of scope for this work, for more

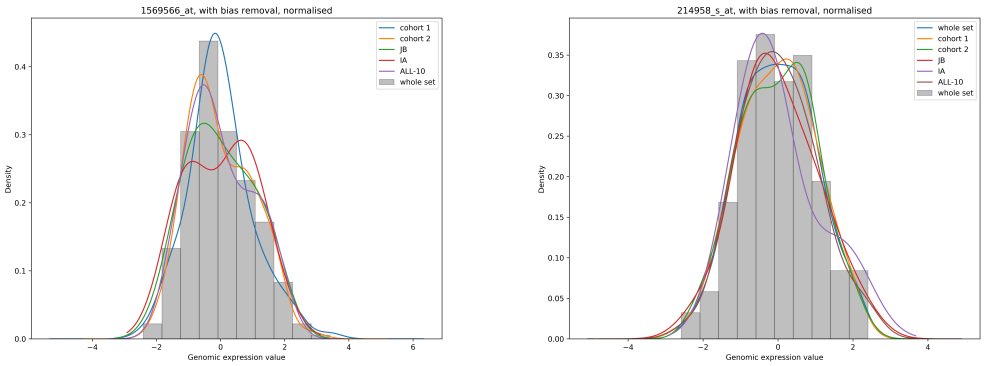


Figure 3: Two sets of distributions with L/S cohort correct, for, (left) a strong predictor and (right) a weak predictor

details we refer the reader to Johnson et al[12]. The basic underlying assumption for all methods is that the samples are stratified over the cohorts, i.e. that in terms of patients each cohort represents a random selection from the total set of patients. Also, it is assumed that the distribution has only one mode. In figures 2 and 3 we show examples of distributions for two different genomes without and with bias removal respectively.

2.2 Dimension reduction

We considered several dimension reduction techniques such as Principal Component Analysis (PCA, see e.g. Shlens[21]), Linear Discriminant Analysis (LDA) and the False discovery rate (FDR, see Yoav and Hochberg[4]).
PCA is basically a transformation of the feature space based on the eigenvectors of the covariance matrix and can be applied to the entire dataset, including the test set. Using the eigenvectors of the covariance matrix as the basis for the features ensures maximal variance perpendicular to the coordinate axes. This is a coarse of saying that we maximize the information content per dimension. The downside is that we obfuscate the biological meaning of the features: any value in the feature set of the transformed matrix is now a linear combination of N genome expression values, where N is the number of dimensions.
In LDA we try to find a linear transformation that maximizes the seperation **between** classes with respect to the seperation **within** classes, requiring two covariance matrices. LDA requires availability of the classification label for fitting, hence the transformation is biased to the training set, also the features are obfuscated similar to PCA. For both LDA and PCA we need to select the number of dimensions a priori. Furthermore we are restricted to a minimum number of dimensions equal to the number of samples. Particularly suited for the dimension reduction of problems with more dimensions than samples is Partial Least Squares Discriminant Analysis (PLS-DA). (TO-DO literature reference or explanation)
The FDR method is basically a feature selection based on a minimum statistical seperarability of the distributions over the different classes. This minimum seperability is in this case the p-value for the rejection of the null hypothesis that the samples are drawn from the same distribution, this p-value is usually denoted by α.

Because the Covariance or linear-discrimination based transformation obfuscates the biological meaning of the feature vectors we choose the FDR method as the most suitable method to reduce the number of dimensions. Also, the FDR method is commonly applied in genomic research (TO-DO add literature).
We apply the FDR method with the Benjamin-Hochberg approach and the ANOVA model to compare the distributions with a maximum p-value set at 0.05.
Arguably, a shortcoming of the FDR method is that, as for LDA, it has a bias towards the training set because it dismisses features solely on the basis of variance across the different classifications which are obviously not available for the test set. Another shortcoming is that it ignores feature interdependency, i.e. we may accidentally dismiss feature combinations as predictors because we have removed their con-

stitutive parts, see e.g. Sun et al.[22].
For this reason, the use of a generic dimension reduction technique such as PCA or Autoencoding is advised to improve the robustness of the model in terms of classification accuracy for larger datasets. For smallers data sets with the number of samples smaller than the number of dimensions we advise PLS-DA.

Alternatively one can apply any model generator (to a subset of the data) that produces importance values for the features, e.g. Random Forest (RF), linear Support Vector Machines (lSVM), Logistic Regression (LR).

3 Classification

We will shortly describe the methods used for the predictions and the determination of genome importances. We will not go in detail on the selection of the method parameters, we refer the reader to the appendix for the parameter selection.

3.1 Tree based

Single decision trees are known to be sensitive to changes in the input data. These ensemble methods help to decrease the variance without increasing the bias, i.e. increasing the ability to be generalised. For small data sets however they are sensitive to overfitting. We employ several tree-ensemble methods: Random Forest (RF) by Breiman[5], ExtraTrees (ET) by Geurts et al.[11] XGBoost (XGB) by Chen and Guestrin[7] and (Light)GBM (LGBM) by Ke et al.[13]. The RF and ET methods are ensemble methods that combine an arbitrary number of decision trees, using bootstrapped samples, random feature selection and a majority vote classification. The XGB and LGBM methods are ensemble methods that apply a technique called gradient boosting by Breiman[6].

3.2 Neural networks

We use two types of neural networks, a Deep Neural Network (DNN) [15] and a Convolutional Neural Network (CNN) [16]. The main advantage of neural networks is that they can learn non-linear relationships between features. Despite the small sample size, it is interesting to apply neural networks in this context due to the high dimensionality of the data. Neural networks with multiple layers are proficient in discerning more subtle patterns in the data compared to other approaches. Shallow neural networks have been succesfully applied to similar sets of genetic expression data in the past, such as in [14]. A DNN, as shown in figure 4, uses several fully connected layers of nodes as a network architecture. A high level explanation of the difference between DNNs and CNNs is that a DNN looks at the entire dataset in each node (layers are fully connected). While a CNN contains operations that allow it to focus on smaller subsets of the data (convolutional layers) and operations that allow it to filter out irrelevant data (pooling layers). An example of a CNN architecture for image classification is shown in figure 5.

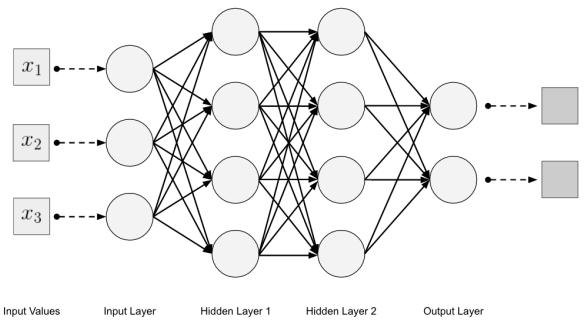


Figure 4: A generic architecture for a deep feedforward neural network.

Using Local Interpretable Model-Agnostic Explanations (LIME) developed by Ribeiro et al.[19] we can get an idea of the so-called local decision boundary (in feature space) for individual samples which

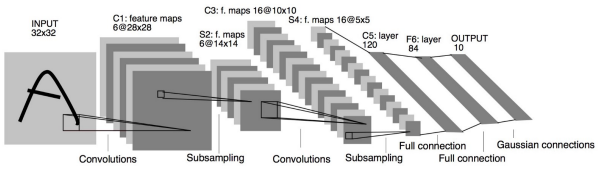


Figure 5: A typical convolutional neural network architecture for image classification from [16].

may assist in understanding the model decision in that particular case. However it is not tractable to discern which features are important for the final overall model.

3.3 Linear methods

The prior discussed methods are general learners that account for non-linearity. These methods cannot give a general first-order direction of influence for perturbations in the features. I.e. we do not get a sense of whether a positively valued change of some feature leads to a positive or negative change in the probability of the target variable.

To do this we need to apply linear methods, specifically we applied Logistic Regression (LR), linear Support Vector Machines (lSVM) and Linear Discriminant Analysis (LDA).

Logistic Regression is a probabilistic seperator, whereby per dimension a logistic regression curve is fit to the dimension/classification values. The seperating plane is given by the feature values giving a probability of 0.5 per dimension. In our specific case we have between 200 and 400 features with a minimum of 50 samples, so we fulfill the so-called *rule of ten*; Supposedly LR requires about 10 data points per feature to give stable predictions, the so-called *rule-of-ten*, and it assumes that the features are independent.

With linear Support Vector Machines (lSVM) we try to find a hyperdimensional plane that maximally seperates the data points for the different classifications. Because this plane, although hyperdimensional, is multi-linear, we basically have a resultant slope in each dimension.

With Linear Discriminant Analysis (LDA) we base the seperation on a probability threshold, whereby the distributions per feature are assumed to be uniform, independent Gaussians and with homogeneity of variance.

The hyperplanes that are created by lSVM, LDA and LR should be very similar, also see [17], [23],[1].

3.4 Bayesian methods

Naive Bayes (NB) classification revolves around the chaining of conditional probabilities under the naive assumption that features are conditionally independent. Naive Bayes has been applied in genomic classification, see e.g. Sandberg et al.[20], Ferrari and Aitken[9], DePristo et al.[10]. The method is naive because it assumes that the features are independent in determining the probability of a classification. Also, the model is zeroeth order dependent on the prior probability of the class, hence, it assumes this prior probability also when applied outside the training data. As the genomic expression values are continuous we have to apply Gaussian NB (GNB) specifically; with Gaussian NB we assign a Gaussian distribution to each feature per classification.

Gaussian Processes Classification (GPC) uses Gaussian Processes to model the prior probabilities and tries to find the conditional class probabilities comparable to Naive Bayes, see e.g. Chu et al.[8] who applied it to genome sequences. The GPC method requires a sampling method such as Markov-Chain-Monte-Carlo which scales quadratically with the dimensionality, hence it is limited to low-dimensional problems. The downside of Bayesian methods, the dependency on prior probabilities (i.e. bias), is also their strength, as the model is much less prone to overfitting.

3.5 Stacking

For the final predictions we stack the models, combining the predictions of the individual models in one final prediction. To do this we use the uncertainty estimation from the model per sample plus the overall accuracy from cross-validation as weights

$$y_i = \frac{\sum_i 2(|\hat{y} - 1/2|\beta\hat{y})_i}{\sum_i 2(|\hat{y} - 1/2|\beta)_i}$$

4 Results

4.1 Model Evaluation

Model evaluation for training strategy $\mathcal{I}$ is done using the standard k-fold cross-validation approach. The data is split into k random folds. In each iteration, a model is trained on $k-1$ folds and tested on the remaining fold. After this procedure we obtain predictions for every sample from a model that is trained on a different part of the data. This gives us a clear idea about how stable the model is and how well it will perform on new data.

4.2 Training strategy $\mathcal{I}$

method	DNN, CNN	RF,ET,XGB,LGBM	lSVM,LR,LDA	GNB, GPC
RAW, $d = 54613$	$71, 71$	$71, 68, 71, 66$	$72, 62, X$	$62, 56$
FDR $\alpha = 0.01$, $d \approx 12$	$96, X$	$86, 84, 80, 81$	$82, 87, \mathbf{88}$	$\mathbf{88}, 86$
FDR $\alpha = 0.02$, $d \approx 15$	$95, X$	$86, 85, 82, 83$	$79, 84, 86$	$86, 84$
FDR $\alpha = 0.03$, $d \approx 57$	$99, \mathbf{92}$	$81, 82, 75, 84$	$89, 89, 82$	$86, 88$
FDR $\alpha = 0.05$, $d \approx 170$	$99, \mathbf{93}$	$79, 78, 75, 79$	$87, 87, 83$	$82, 81$
FDR $\alpha = 0.1$, $d \approx 412$	$100, \mathbf{93}$	$79, 79, 75, 79$	$86, 86, 83$	$81, 81$
PLS-DA, $d = 100$	$99, 81$	$79, 66, 83, 75$	$100, 100, 100$	$\mathbf{86}, 56$
PLS-DA, $d = 200$	$99, 82$	$74, 60, 82, 75$	$100, 100, 100$	$\mathbf{86}, 56$
PLS-DA, $d = 400$	$100, 78$	$75, 64, 82, 76$	$100, 100, 100$	$\mathbf{86}, 56$

Table 1: Training strategy $\mathcal{I}$. Mean accuracies in % over 10 runs with 5% added random noise per run, with 10 folds for the cross-validation. d denotes the number of features

The DNN model is likely overfitted, despite the added 5% noise. If we use PLS we see overfitting for lSVM, LDA and LR, again despite the noise. The model overfitting can be a result of the small number of samples with respect to the number of dimensions. Overall, the accuracy does not increase with the number of features. This may be due to the noise generated by non-important genomes leading to a high model variance. The linear methods perform well which may be due to the low effective dimensionality where a few genomic expressions dominate *and* have a monotonous correlation with the classification probability.

To further increase the accuracy of our predictions (without increasing the bias) we can apply a so-called ensemble method that combines the predictors into one predictor. For the ensemble predictor we use DNN, RF, lSVM and GNB.

Apparantly, using the full feature space is problematic in terms of accuracy, most likely due to the large amount of noise given that only about 500, i.e. 1% plays a significant role in driving the predictors.

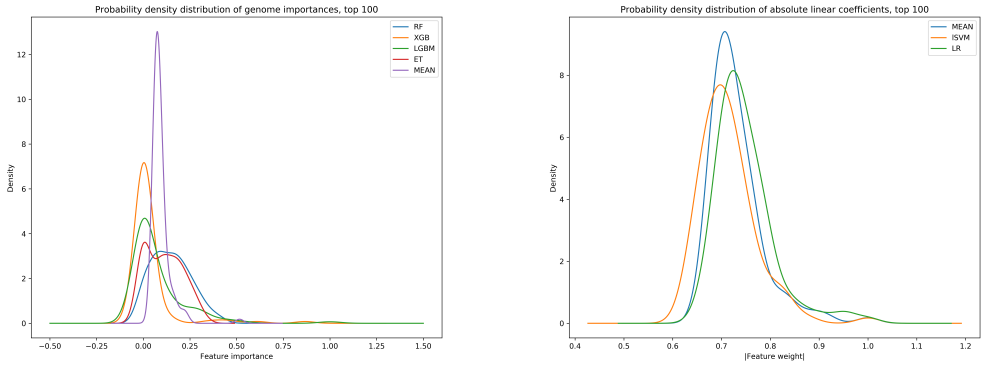
4.3 Training strategy $\mathcal{II}$

5 Post-processing

In figure 7 we present the density distributions of the feature weights/importances that are part of the models generated using all available features (i.e. no feature selection or dimension reduction),

method	soft mean	majority vote
RAW, $d = 54613$	63	64
FDR $\alpha = 0.01$, $d \approx 12$	88	86
FDR $\alpha = 0.02$, $d \approx 42$	90	87
FDR $\alpha = 0.03$, $d \approx 71$	90	85
FDR $\alpha = 0.05$, $d \approx 220$	80	79
FDR $\alpha = 0.1$, $d \approx 469$	80	81

Table 2: Training strategy $\mathcal{I}$. Mean accuracies in % over 10 runs with 5% added random noise per run, with 10 folds for the cross-validation. d denotes the number of features



(a) (b)

Figure 6: Distribution of (a) weights and (b) importances associated with genomic expression values with regard to their influence on several models for training strategy $\mathcal{I}$

clearly only few genomic expression values have a significant individual contribution. The distribution of importances that are produced by the tree methods are dominated by the large number of insignificant features. If we apply the FDR method with $\alpha = 0.05$ we reduce the dimensionality to about 200 features. We then train the models on the reduced data set so we get a smoother distribution of importances, see figure 7, and the same for the weights obtained from the lSVM and LR methods, see figure 8.

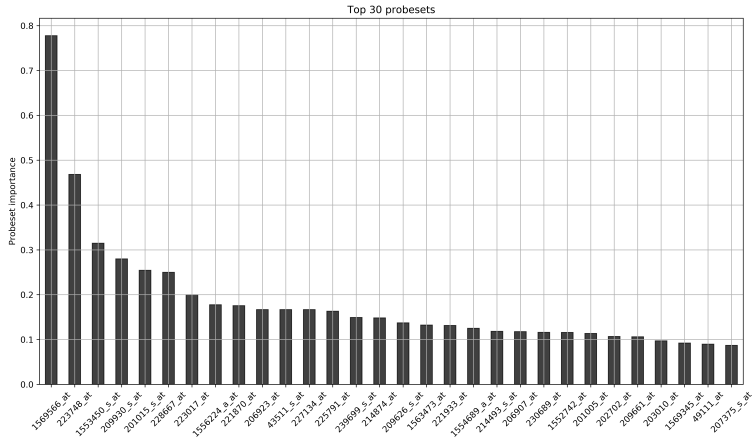


Figure 7: Top 30 probesets in terms of median absolute importances (weights) for the RF, ET, XGB and LGBM models

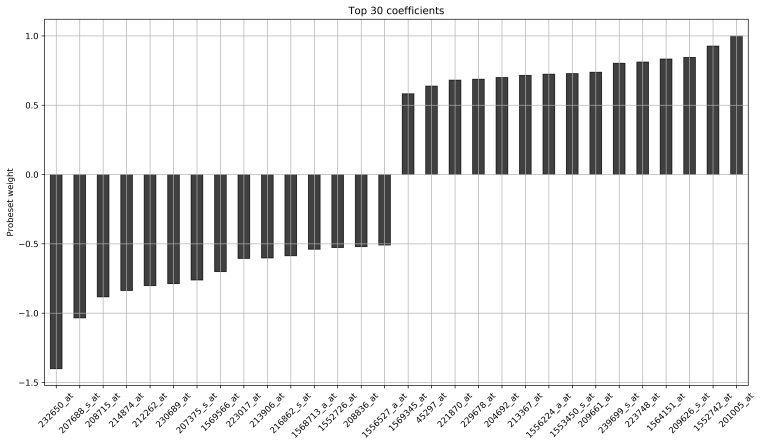


Figure 8: Top 30 probesets in terms of median absolute weights for the lSVM and LR models

6 Summarising

For the cohort-bias removal we apply a genome-wise Location and scale (L/S) adjustment per cohort. This guarantees similar bounds and means over the different cohorts. To remove the multiplicity of probesets per patients we simply take the mean of the measurements per patient. Having obtained a normalised dataset we add 5% of noise (with respect the maximum value) to increase the robustness of the final model. To further increase robustness, and improve interpretability we reduce the number of features by applying the FDR method with ANOVA for the distribution comparison, following the Benjamin-Hochberg procedure, using a maximum p-value of 0.05.
We use the data from adult patients with a known high/low intensity diagnosis (where medium intensity

is considered equal to the low intensity diagnosis) to train a model to predict the diagnosis for the other non-labelled patients.
For the model creation we employ several *tree-ensemble methods*: Random Forest (RF) by Breiman[5], ExtraTrees (ET) by Geurts et al.[11] XGBoost (XGB) by Chen and Guestrin[7] and (Light)GBM (LGBM) by Ke et al.[13]. We use three types of linear methods, Linear Discrimination Analysis, Logistic Regression and linear SVM. We use two nonlinear methods, both *neural networks*, a Deep Neural Network (DNN) [15] and a Convolutional Neural Network (CNN) [16]. Finally, we use two Bayesian methods, Naive Bayes and Gaussian Processes Classication (GPC).
Finally we combine them in a so-called stacked method, either by a majority vote of the classifications or by weighted averaging based on their prediction probabilities.
Having obtained the models we can extract their relative importance from the tree-based models, and the weights from the linear models. In our case we could reduce the more than 50.000 expression values to about 200 genomes without losing accuracy. The best reported accuracies are close to 90% with 10 to 500 features.

7 Discussion

- when choosing PCA, LDA, check for inflection point in eigenvalue magnitude to 'smartly' select the number of components
- successively apply standard scaling and maxabs scaling to center cohort data?
- improve bias removal method L/S by ignoring outliers during normalisation
- instead of applying FDR once on the dataset, apply it several times on the same dataset with added noise to remove spurious contributors
- we can combine the different models in one meta-model. This bagging of models increases the accuracy, removes method-specific biases and at the same time its helps reduce overfitting
- the model generators that we applied, primarily used default parameter settings, we can likely improve the model quality by performing parameter optimisation
- we have not, but perhaps should have, applied methods to detect overfitting, i.e. parameter-change sensitivity, accuracy variability with super-imposed noise.

Appendix

Model parameters

If any parameter is missing, please consult the documentation of the sklearn library (v. 0.19.1), the lightGBM library (v. 2.1.0), the XGboost library (v. 0.71) or Keras (v. 2.0.8).

```
            "GPC":{'optimizer': 'fmin_l_bfgs_b', 'n_restarts_optimizer': 0,
                   'max_iter_predict': 100, 'warm_start': False,
                   'copy_X_train': True, 'random_state': self.SEED,
                   'multi_class': 'one_vs_rest', 'n_jobs': 1},
            "LDA":{'shrinkage': 'auto', 'solver': 'lsqr', 'priors': None},
            "LGBM": {'boosting_type':'gbdt' ,'learning_rate': 0.75,
                   'max_depth': 4, 'num_leaves': 100, 'n_jobs': -1,
                   'n_estimators':100, 'random_state': self.SEED},
            "XGB": {'seed': self.SEED, 'n_estimators': 100, 'max_depth': 3,
                   'learning_rate': 0.1, 'objective': 'reg:linear', 'nthread': -1},
            "ET": {'n_estimators': 50, 'max_depth': 15, 'n_jobs': -1,
 'min_samples_split': 10, 'min_samples_leaf': 5},
            "RF": {'n_estimators': 200, 'max_depth': 10, 'n_jobs': -1,
 'min_samples_split': 10, 'min_samples_leaf': 5},
            "SVM":{'kernel': 'linear', 'gamma': 'auto', 'tol': 0.0005, 'C':  0.9,
```

```
'probability' : True, 'max_iter': 3000},
          "LR": {'penalty':'l2', 'dual': False, 'tol':0.0001, 'C':0.9, 'max_iter': 100,
'fit_intercept': True, 'intercept_scaling': 1},
```

References

[1]

[2] Orly Alter, Patrick O. Brown, and David Botstein. Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Sciences*, 97(18):10101–10106, 2000.

[3] Monica Benito, Joel Parker, Quan Du, Junyuan Wu, Dong Xiang, Charles M. Perou, and J. S. Marron. Adjustment of systematic microarray data biases. *Bioinformatics*, 20(1):105–114, 2004.

[4] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the royal statistical society. Series B (Methodological)*, pages 289–300, 1995.

[5] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[6] Leo Breiman. Arcing the edge. Technical Report Technical Report 486, Statistics Department University of California, 08 1997.

[7] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[8] W. Chu, Z. Ghahramani, F. Falciani, and D. L.Wild. Biomarker discovery in microarray gene expression data with gaussian processes. *Bioinformatics*, 21(16):3385–3393, 2005.

[9] L. De Ferrari and S. Aitken. Mining housekeeping genes with a naive bayes classifier. *BMC Genomics*, 7(1):277–291, 2006.

[10] M.A. DePristo, E. Banks, R. Poplin, K.V. Garimella, J.R Maguire, C. Hartl, A.A. Philippakis, G. del Angel, M.A. Rivas, M. Hanna, A. McKenna, T.J. Fennell, A.M. Sivachenko, K. Cibulskis, S.B. Gabriel, D. Altshuler, and M.J. Daly. A framework for variation discovery and genotyping using next-generation dna sequencing data. *Nature Genetics*, 43(491):491–498, 2011.

[11] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[12] W. Evan Johnson, Cheng Li, and Ariel Rabinovic. Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics*, 8(1):118–127, 2007.

[13] G. Ke, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T-Y Liu. Lightgbm: a highly efficient gradient boosting decision tree. In *31st conference on Neural Information Processing Systems*. NIPS, 2017.

[14] Javed Khan, Jun S Wei, Markus Ringner, Lao H Saal, Marc Ladanyi, Frank Westermann, Frank Berthold, Manfred Schwab, Cristina R Antonescu, Carsten Peterson, et al. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature medicine*, 7(6):673, 2001.

[15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[16] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[17] M. Pohar, M. Blas, and S. Turk. *Metodoloski zvezki*, 1(1):143–161, 2004.

[18] A.L. Price, Patterson N.J, R.M. Plenge, M.E. Weinblatt, N.A. Shadick, and D. Reich. Principal components analysis corrects for stratification in genome-wide association studies. *Nature Genetics*, 38:904–909, 2006.

[19] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[20] R. Sandberg, G. Winberg, Bränden C-I, A. Kaske, I. Ernberg, and J. Cöster. Capturing whole-genome characteristics in short sequences using a naïve bayesian classifier. *Genome Research*, 11:1404–1409, 2001.

[21] Jonathon Shlens. A tutorial on principal component analysis. *CoRR*, abs/1404.1100, 2014.

[22] G-W Sun, T.L. Shook, and G.L. Kay. Inappropriate use of bivariable analysis to screen risk factors for use in multivariable analysis. *Journal of Clinical Epidemiology*, 49(8):907–916, 1996.

[23] Xinyi Yong and Carlo Menon. Eeg classification of different imaginary movements within the same limb. 10:e0121896, 04 2015.

[24] C. Zang, T. Wang, K. Deng, B. Li, T. Xiao, S. Zhang, C.A. Meyer, H.H. He, M. Brown, J.S. Liu, Y. Xie, and X.S. Liu. High-dimensional genomic data bias correction and data integration using mancie. *Nature Communications*, 7, 2016.