Biomarkers of AML Utilizing Machine Learning Feature Extraction

ANNE FRENCH, University of Washington Tacoma

Acute Myelogenous Leukemia (AML) is currently the leading cause of leukemic childhood death but, due to molecular diversity of patients and the disease, there is not currently an adequate understanding of pediatric AML nor its distinctive features. This paper discusses expansion of experiments conducted during the NCBI February 2019 Hackathon in which automated feature selection was explored for determination of low or not low risk of disease progression in AML patients. In addition, we evaluate the accuracy of classification using these extracted features.

CCS Concepts: • **Computing methodologies** → **Ensemble methods**; **Feature selection**; • **Applied computing** → **Bioinformatics**;

Additional Key Words and Phrases: bioinformatics, ensemble methods, machine learning, feature selection

ACM Reference format:
Anne French. 2019. Biomarkers of AML Utilizing Machine Learning Feature Extraction. 1, 1, Article 1 (March 2019), 7 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Acute Myelogenous Leukemia (AML) is currently the leading cause of leukemic childhood death, making up 20% of perdiatric acute leukemias and being the most common form of childhood cancer [1]. AML comprises a molecularly diverse group of diseases and affects people of any age, however pediatric targeted therapies are not used to fight the disease and there is currently not an adequate understanding of the biology of pediatric AML [1]. To date, there has been no thorough description of the distinctive features of pediatric AML [1]. Because of this, the problem of determining pediatric AML risk based on gene expression became the target of the National Center for Biotechnology Information (NCBI) February 2019 hackathon.

During the hackathon, teams used machine learning algorithms to select features from clinical and assay data from pediatric cancer patients from the Therapeutically Applicable Research to Generate Effective Treatments (TARGET) initiative [4], a program that analyzes molecular changes that drive pediatric cancer [6]. Teams explored the utilization of data science and machine learning for automated feature selection [4]. While teams are still working on extending and polishing their methods and papers, this study will extend upon methods already existing in the hackathon data in GitHub.

2 DATASET

Researchers in the hackathon extracted TARGET cancer assay and clinical data on February 4, 2019 [4]. The TARGET dataset extracted consists of blood and bone marrow samples from 156 patients [4]. Researchers used the included variable which yields the patient's risk of disease progression to prepare the data for binary classification to determine if a person was at low risk (0) or not low risk (1) for AML progression, with the latter category comprising both standard and high risk patients [4]. All samples with unknown disease risk were removed from the dataset. The dataset contains data for 21,404 genes.

3 METHODS

First, we'll discuss classifiers that were used in our experiments as standalone classifiers, as well as ensembles (and sometimes ensembles of ensembles). After, we'll go over the feature section and classification processes. A number of machine learning classifiers were built using available libraries, most from Scikit-learn [7]. Those classifiers are described in brief detail below.

3.1 Previous Classifiers

Three ensemble classifiers (all tree boosting ensembles) were used in the initial project done during the NCBI Hackathon in February of 2019: Random Forest, XGBoost, and GradientBoost [4].

3.1.1 Random Forest. In random forests, each tree is built from a bootstrapped sample from a training set [7]. Here, trees are built by splitting nodes based on the best split among a randomly selected feature subset, causing a slight increase in bias which is compensated for by averaging and a decrease in variance [7]. A number of these decision trees are used to build this random forest, taking the overall average of their probablistic predictions to find the prediction of the ensemble [7]. For our random forest, we use 300 trees in each experiment, and set the maximum depth for each tree to three.

3.1.2 GradientBoost. Gradient boosting is a boosted tree ensemble method, but its data subset selection method is not random. Gradient boosting focuses on data and features that are difficult to correctly predict. After each round of training, the gradient–or partial derivative–of the loss function is calculated, and the goal of gradient boosting is to minimize this gradient. However, since this is a boosting algorithm, the predictions of many models are used to make predictions and the algorithms keeps a running tab of gradients during the training process. According to Scikit-learn, gradient boosting is unable to be parallelized (but we will see in a minute that there is a modified algorithm that found a way) [7]. In our algorithm, we set the maximum depth of each GradientBoost tree to four and use 280 estimators.

3.1.3 XGBoost. Like Random Forest and GradientBoost, XGBoost is also a tree boosting classifier system. XGBoost is one of the newest, and also most frequently used algorithms in data science [2]. The algorithm's systems and optimizations, along with the fact that it enables parallel and distributed computing, makes it highly scalable, as well as fast and efficient [2]. XGBoost uses second-order gradients of the loss function, unlike the first-order gradients that regular gradient boosting utilizes, as well as advanced regularization parameters [2]. The algorithm also utilizes computer resources and enables out-of-core computation by dividing data into blocks and storing compressed blocks on disk until use and using independent data block fetching threads [2]. This block division and sharding, along with collecting and comparing statistics for the many blocks, is what enables parallelization. In our algorithm, we limit XGBoost tree depth to 3 and number of trees in each round to 700.

3.2 New Classifiers Added

A number of new classifiers were added to the code already available on GitHub, some simply to be used to build ensembles, while others were added to be tested as feature selection methods. New classifiers added are described below.

3.2.1 Support Vector Machine (SVM). Like many other classification methods, Support Vector Machines are relatively easy to understand. In the model, features are represented as feature vectors in a t-dimensional space. The idea is to separate the vectors belonging to two categories, c_a and c_b, by inserting a decision plane (which is simply a hyperplane) in between the differing categories, dividing them. The decision plane should be inserted so that it maximizes the distance of the closest feature vectors from two neighboring categories. These closest feature vectors which form the delimiting hyperplane are called the *support vectors*.

In the case of binary classification, a new data point is represented as vector and classified using the decision function, which labels the new point as positive or negative, depending on which side of the hyperplane it lies. If the sign of a new data point is positive, then the new data point is assigned to class c_a. If the function has a negative result, the new point is assigned to the other class, c_b [3]. There are methods for applying SVMs to multi-class categorization, such as simply applying the SVM using a one-class-versus-all-others method.

In our algorithm, we used a polynomial kernel (decision) function, which allows the model to be built using combinations of features, as well as individual features [7].

3.2.2 k-Nearest Neighbors (kNN). The kNN algorithm is quite possibly the laziest of all machine learning algorithms. Like with SVM, kNN represents all data points as vectors. However, all computation occurs at classification time in this case. The majority vote of the nearest k neighbors to a new data point is calculated, and this vote is what determines the new data point's classification. kNN can be used for both binary and multi-class classification. There are, however, different variations of this algorithm. For instance, in some instances weights can be applied to existing decision points, with nearer points to a new data point holding more weight.

Our algorithm uses the 3 nearest neighbors for classification.

3.2.3 Extreme Learning Machine (ELM). In order to understand what an Extreme Learning Machine is, one has to first understand the basics behind neural networks. A neural network is just a simplified graph representation of connected brain neurons [3]. Neural networks are structured so that they form layers of neurons, with connections passing between the layers. At any point, a neuron's state is defined by the combined effect of all input received from preceding layers. A neural network is comprised of an input layer that serves as an entry point for the input data, one or more hidden layers that apply a function and weights to the input data in order to make computations and also transform the input data into a suitable form for the output layer, and an output layer that contains the result of all computations performed.

There are many different types of neural networks–recurrent neural networks, Long/Short Term Memory (LTSM), Markov Chain, Deep Convolutional Network, etc. An Extreme Learning Machine is simply a feedforward neural network (the connections between nodes do not form a cycle) with a single hidden layer that doesn't require parameter tuning of hidden layer nodes [5].

In our algorithm we used the sigmoid function as the kernel (activation) function in the hidden layer nodes of ELM, and used 80 nodes in the hidden layer.

3.2.4 AdaBoost. AdaBoost, as its name implies, is a boosting classifier. It builds ensembles of weak classifiers to build a single strong classifier. AdaBoost is short for "adaptive boost", and this classifier adapts by tweaking new classification experiments to focus on previously misclassed instances. The difference between AdaBoost and GradientBoost is that GradientBoost focuses on loss, with each subsequent learner predicting the loss after the previous step, while AdaBoost calculates the weight of each learner based on its ability to correctly classify and continues until it converges to minimization of loss.

The beauty of AdaBoost is that it can utilize any base learner that accepts weights, but works best with weak classifiers (classifiers that do only slightly better than random guessing) [7]. We used the Stagewise Additive Modeling using a Multi-class Exponential loss function (SAMME) algorithm as the base learner in our case, which is a multi-class extension of the original two-class AdaBoost model and uses the multi-class Bayes rule [8].

3.2.5 LogitBoost. To put it simply, LogitBoost is similar to AdaBoost and GradientBoost but the difference lies in the fact that LogitBoost is driven by logistic loss minimization. Also, LogitBoost uses logistic regression as its base learner [5]. Just as we did with AdaBoost and Random Forest, we limit the number of estimators in our LogitBoost experiments to 300.

Table 1. 1,000 features with most and least variance were selected from both low-risk and not-low-risk groups and used to fit and test a number of different machine learning algorithms and ensembles. The log loss results were compared with three experiments which randomly chose 2,000 features.

Log Loss of High Variance, Low Variance, and Randomly Chosen Features											
Feature Selection Method	GB	RF	XGB	Orig. Ens.	SVM	KNN	ELM	AB	LB	New Ens.	Tot. Ens.
High variance features	0.3621	0.3262	0.2218	0.2877	0.2372	0.1767	0.4044	0.3563	2.8756	0.2208	0.2431
Low variance features	0.4926	0.4822	0.3429	0.4237	0.3218	2.2346	0.7105	0.4666	3.4020	0.3956	0.4033
Random feature selection 1	0.4594	0.4180	0.2528	0.3560	0.2605	2.0875	0.4020	0.4330	1.8771	0.2916	0.3116
Random feature selection 2	0.4428	0.4412	0.2972	0.3815	0.2421	2.0547	0.6791	0.4893	7.2623	0.3969	0.3890
Random feature selection 3	0.4319	0.4107	0.3202	0.3500	0.2297	1.1477	0.7202	0.5468	4.1281	0.3142	0.3258

3.3 Process

In our experiments, we followed the same process as [4]. First, we analyzed the performance of all classifiers with respect to various simple feature selection methods. Of course, when speaking of feature selection, we are talking about selecting the best genes to determine whether a patient is at low or not low risk of AML progression. Second, we used the most accurate classifiers in our first round of experiments to distinguish the most important genes used in their classification of low/not low AML progression risk. While the initial Hackathon ended with analysis of features selected [4], we went one step further. We also built ensemble classifiers out of the classifiers used in the initial Hackathon experiments, new classifiers added, and original and new classifiers together. Lastly, we used the various sets of genes selected in the second step and fed their data through the ensemble methods constructed to compare and test accuracy and loss of these feature selection methods and classifiers.

3.3.1 Feature Selection. As in the initial Hackathon experiments, we begin by extracting the 1,000 features with the most variance from both low-risk and not-low-risk groups, as well as the 1,000 features with the least variance from the two groups. We also build three sets of randomly extracted features. We use GradientBoost (GB), Random Forest (RF), XGBoost (XGB), ensemble of original classifiers (Orig. Ens.), SVM, kNN, ELM, AdaBoost (AB), LogitBoost (LB), ensemble of new classifiers (New Ens.), and ensemble of all classifiers (Tot. Ens.) to test both the accuracy of each classifier, as well as determine how well using features with low and high variance compares with randomly selected features.

Additionally, the highest performing ensemble classifiers (Random Forest and XGBoost) were used as feature selection methods over all possible features. We also wanted to add one of our ensemble methods (either AdaBoost or LogitBoost) as a feature selection method to compare with the original methods. Log loss was used to evaluate the classifier predictions across the three feature selection methods. In all experiments, 33% of the data was reserved for testing.

As shown in Table 1, AdaBoost performed relatively well, doing better than Random Forest on some occasions and performing consistently better than GradientBoost. LogitBoost performed the worst out of all classifiers, with log loss so high it likely never converged. Therefore, we selected AdaBoost as a feature selection method to compare its features along with feature sets selected by methods in the original experiment.

With regard to classifier evaluation according to the three feature selection methods, what was expected to happen did. All classifiers performed poorest (the most loss was observed) when attempting to classify patients' AML progression risk as low or not low using features with low variance. We wouldn't expect classifiers to perform very well using data points that closely resemble each other because there isn't as much to learn between the points as there would be with data points with greater variance or spread.

Table 2. Used 174 genes extracted from Lasso convergence, 332 most important genes in XGBoost classification, 28 most important genes in AdaBoost classification, and intersection of combinations of the three. Used features to determine log loss of classification of low risk and not low risk AML patients using ensemble methods. Original ensemble methods include GradientBoost, XGBoost, and Random Forest. New ensemble methods include SVM, KNN, Extreme Learning Machine, AdaBoost, and LogitBoost. 33% of the data, or 37 samples, were reserved for training in each case.

Log Loss of Ensemble Classification using Features extracted from Lasso, XGBoost, Adaboost, and Combinations of the Three			
Feature Selection Method	Ensemble All Methods	Ensemble of New Methods	Ensemble of Original Methods
Lasso	0.2877	0.2648	0.3466
XGBoost	0.2373	0.2106	0.3007
AdaBoost	0.2584	0.2585	0.2720
Lasso/XGBoost	0.3567	0.3601	0.3718
Lasso/AdaBoost	0.3965	0.3935	0.4288
XGBoost/AdaBoost	0.2584	0.2606	0.2638
Lasso/XGBoost/AdaBoost	0.4101	0.4042	0.4385

Of note, the original experimenters noted that the Random Forest chose 1,147 features as most important in classification in the original experiments. While the researchers mentioned this, they did not include this feature subset in the final evaluation and findings, likely because this is too large of a number to argue for the case for making use of gene epression-based heterogeneity in AML [4]. Also, in addition to using the sets of features found most useful by XGBoost, a logistic regression classifier with Least Absolute Shrinkage and Selection Operator (LASSO) was fit on all features and left to converge in the original experiment [4]. The researchers used these genes found in LASSO logistic regression, along with the genes selected by XGBoost, in their final findings [4].

Therefore, feature selection methods in our experiment will include the most important genes found when all features are evaluated using converged logistic regression with LASSO, XGBoost, and AdaBoost. After the important features are extracted, these classifiers, as well at the three ensemble classifiers will be rebuilt using the three (and combinations of the three) sets of features in order to evaluate whether or not these feature sets are able to correctly classify AML patients as being at low or not low risk of disease progression.

4 RESULTS

Just as with the original experiment, XGBoost found that there were 332 important genes in its classification, and logistic regression with LASSO regularization deemed 174 genes as most important [4]. Additionally, when building an AdaBoost classifier using all features, there were only 28 genes found most important and only 7 of these genes were not genes also deemed important by XGBoost (the two feature selection method subsets shared 21 genes). AdaBoost and logistic regression with LASSO only shared 6 important genes, and XGBoost and logistic regression with LASSO only shared 14 genes. There were only 5 genes shared in all three feature selection methods.

We used the set of genes found by logistic regression with LASSO, set of genes found by XGBoost, set of genes found by AdaBoost, the intersection of combinations of the sets, and the intersection of the combination of all three sets and used the three ensemble classifiers we built using the original classifiers in [4], our new classifiers, and all classifiers to determine if any of these features alone were capable of determining whether an AML patient was at low or not low risk of disease progression. We used the log loss that was in the original Hackathon experiment (see Table 2) [4], and we also determined accuracy after 5-fold cross-validation of the classifiers (see Table 3).

When using the five genes in the intersection of the combination of all three sets, log loss was only slightly better than when using random feature selection. Log loss was lowest when using the 332 genes extracted from

Table 3. Used 174 genes extracted from Lasso convergence, 332 most important genes in XGBoost classification, 28 most important genes in AdaBoost classification, and intersection of combinations of the three. Used features to determine mean accuracy of classification of low risk and not low risk AML patients using ensemble methods using 5-fold cross-validation. Original ensemble methods include GradientBoost, XGBoost, and Random Forest. New ensemble methods include SVM, KNN, Extreme Learning Machine, AdaBoost, and LogitBoost. There were an average of 22 samples in each cross validation fold.

Mean Accuracy of Ensembles using Features extracted from Lasso, XGBoost, Adaboost, and Combinations of the Three			
Feature Selection Method	Ensemble All Methods	Ensemble of New Methods	Ensemble of Original Methods
Lasso	0.8987	0.8896	0.8896
XGBoost	0.9632	0.9632	0.9541
AdaBoost	0.9632	0.9632	0.9632
Lasso/XGBoost	0.8983	0.8623	0.8896
Lasso/AdaBoost	0.8706	0.8797	0.8706
XGBoost/AdaBoost	0.9632	0.9632	0.9632
Lasso/XGBoost/AdaBoost	0.8706	0.8801	0.8615

the XGBoost classifier. Most notably, the 21 genes that formed the intersection of the XGBoost and AdaBoost sets and 28 genes that formed the AdaBoost set gave very similar log loss performance and did only slightly worse than the 332 genes selected by XGBoost.

When looking at accuracy, the 28 genes extracted by AdaBoost and the 21 genes that formed the intersection of the XGBoost and AdaBoost sets all had the same accuracy of 96.32% across all three ensemble methods. The 332 XGBoost genes gave 96.32% classification accuracy, except when the ensemble of original methods was used as the classifier.

5 DISCUSSION

Our results indicate there is a strong potential for automated selection of features to characterize cancer and subsets of cancer information (as we did here with determining whether or not a patient was at low or not low risk of AML progression) that could previously not be characterized due to molecular heterogeneity. With close to 100% accuracy in our experiments and narrowing 21,404 genes down to less than 30 genes, it's worth at least delving into the 21 genes in the intersection of the AdaBoost and XGBoost extracted features for further analysis.

6 NEW CONTRIBUTIONS TO GITHUB MATERIALS

To reiterate our contributions to the existing GitHub material, we have added SVM, kNN, ELM, AdaBoost, and LogitBoost classifiers to the existing classifiers, and have built ensemble classifiers out of the existing and new classifiers. Additionally, we added the ability to calculate accuracy of ensemble methods and 5-fold cross-validation. We added the utilization of AdaBoost for important feature selection. One thing that was not previsouly mentioned is the addition of printing all important sets of features to a .csv file, which comes in handy for visualization of results. Also, we added code that prints classifier performance information to a log file.

ACKNOWLEDGMENTS

To my kids and husband, for all the time pursuance of my graduate degree has taken away from the three of you. It's finally over.

REFERENCES

[1] Hamid Bolouri, Jason Farrar, Timothy Triche, Rhonda Ries, Emilia Lim, Todd Alonzo, Yussanne Ma, Richard Moore, Andrew Mungall, Marco Marra, Jinghui Zhang, Xiaotu Ma, Yu Liu, Yanling Liu, Jaime Guidry Auvil, Tanja Davidsen, Patee Gesuwan, Leandro Hermida, Bodour Salhia, Stephen Capone, Giridharan Ramsingh, Christian Michel Zwaan, Sanne Noort, Stephen Piccolo, Edward Anders Kolb, Alan Gamis, Malcolm Smith, Daniela Gerhard, and Soheil Meshinchi. 2017. The molecular landscape of pediatric acute myeloid leukemia reveals recurrent structural alterations and age-specific mutational interactions. *Nature Medicine* 24 (2017), 103–112.

[2] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 785–794.

[3] W. Bruce Croft, Donald Metzler, and Trevor Strohman. 2015. *Search Engines Information Retrieval in Practice.* Pearson Education, Inc.

[4] Sean Maden, David Lee, Jenny Smith, Ronald Buie, Vikas Peddu, and Ryan Shean. Detecting Cancer Biomarkers from RNA-seq Using Machine Learning. (????). Retrieved March 10, 2019 from https://github.com/NCBI-Hackathons/ConsensusML

[5] Abbas Mikhchi, Mahmood Honarvar, Nasser Emam Jomeh Kashan, and Mendhi Aminafshar. 2016. Assessing and comparison of different machine learning methods in parent-offspring trios for genotype imputation. *Journal of Theoretical Biology* 399 (June 2016), 148–158. https://doi.org/10.1016/j.jtbi.2016.03.035

[6] Office of Cancer Genomics. TARGET: Therapeutically Applicable Research To Generate Effective Treatments. (????). Retrieved March 13, 2019 from https://ocg.cancer.gov/programs/target/

[7] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (Feb. 2011), 2825–2830. https://arxiv.org/pdf/1201.0490.pdf

[8] Ji Zhu, Hui Zou, Saharon Rosset, and Trevor J. Hastie. 2005. Multi-class AdaBoost.