|  |
| --- |
| Using a Crowdsourcing Approach to Build Predictive Models for Treatment Response in Acute Myeloid Leukemia  Michael Theisen1,2,3,5,\*, Trevor Tomlin1,4,\* and Ye Li1,2,7,\*  1University of Washington Tacoma, Computer Science and Systems, Undergraduate Level, 2Determined suitable data subset. 3Contributed R data and typed paper. 4Contributed Python data, 5Contributed 114 hours to project. 6Contributed 10 hours to project. 7Contributed 6 hours to project.  \* [yl424@uw.edu](mailto:yl424@uw.edu), [mtheis@uw.edu](mailto:mtheis@uw.edu) <https://orcid.org/0000-0002-6177-1007>, [ttomlin@uw.edu](mailto:ttomlin@uw.edu)  Published: March 14, 2023  Abstract  **Motivation:** Acute Myeloid Leukemia (AML) is a cancer represented by the unwanted growth of cells that accumulate in the bone marrow and blood. This can lead to an interference of normal blood cell production. Once present in the body, it can progress rapidly and can be fatal within weeks or months if left untreated. There are a few methods of treating AML from chemotherapy to stem cell transplants, but there may be yet unknown methods of treatment that may be found through transcriptomic analysis. If we can predict its presence, we can find new ways to treat or cure it. Here, we attempt to do so by using a crowdsourcing approach using the GSE122517 dataset.  **Results:** We decided to use multiple groups of students using different iterations of machine learning algorithms to see if it was possible to detect the presence of AML within the datasets. Our group in particular was able to detect with a level of 97% certainty using ensemble models of the presence of AML within certain subsets of genes that could be used as identifying markers of AML presence. These subsets may be used in the future to check for presence in other patients that have yet to be diagnosed with AML or to detect genes that may be useful in new treatment techniques. |

# Introduction

The goal of precision medicine is to find a way to personalize treatment based on a patient’s individual characteristics. Here, we apply a precision medicine approach to treatment for cancer patients. Acute Myeloid Leukemia (AML) is a type of cancer represented by a rapid growth of cells that accumulate in the bone marrow and blood which leads to an interference of normal blood cell production. It progresses rapidly and can be fatal within weeks if not treated. While there have been various methods of which to treat AML, such as chemotherapy or stem cell transplants, there may be yet unknown methods of treatment that may be found by predicting its presence with a crowdsourcing approach using the GSE122517 dataset from the scientific paper, “Scalable Prediction of Acute Myeloid Leukemia Using High-Dimensional Machine Learning and Blood Transcriptomics''. Warnat-Herresthal et al. iScience 2020.

The genetic characteristics of patients diagnosed with AML can be used to see if they may be responsive to known or novel types of cancer treatments. If we can find ways to predict AML through transcriptomics, we may find new ways to prevent or cure it in its early stages before symptoms are even noticeable by other means. Treatment recommendations at the moment of diagnosis may even be able to improve patient outcomes.

We used R programing language within a Jupyter notebook as well as the Python programming language run locally to build the algorithms. After importing the datasets, we ran correlations and t-tests on the data to find a specific feature selection. This Dataset included 2,500 samples assessed by HG-U133A microarray, but we chose “AML” cases and “non-AML” control tests as our particular feature. From here, we ran a variety of algorithms to find a model that could yield highly accurate results.

# Methods

When using both R and Python, we imported the data from Dataset 1, labeled “GSE122505”, of the 3 datasets from the research paper after they were provided by the author. Using Bioconductor and GEOquery yielded mixed results among different groups within the class, and it was decided that downloading the data directly was faster and easier. The author of the paper was kind enough to send us an R data file that contained all of the datasets from the experiment. We uploaded this file titled “AML\_datasets.RData”, and were then able to use two files contained within labeled “data.1” for the gene expression data, and “info.1” for the phenotype data.

Our first foray into the data was utilizing R within the Jupyter Notebook. After importing both “data.1” and “info.1” into R, we quickly realized that choosing to run correlations and t-tests on such a massive dataset would not only take a long time, but it would cause the R kernel to fail, and the Jupyter Notebook would crash. We eventually found a few different avenues to pursue before ultimately finding a way to get good p-values from the t-test. We would eventually merge the two datasets, separate via CASE (AML) and CONTROL (non AML), apply the t-test, and then list a set of genes with lowest p-values.

We continued on by constructing a logistic regression model on the data set. In summary, a forward stepwise logistic regression model yields better results than that of the backwards stepwise function, which yielded only the null model. The forward stepwise logistic regression model yielded the genes CREG1, ERLIN1, MAPKAPK3, CD22, CD79A, KIT, POU2AF1, CD79B, CFD, and CD19 with an AIC of 22 and listing True Positives of only 1751. Notably, these are the exact same genes acquired from the above t-test.

A second method of overcoming the hurdles in Jupyter with R was in searching for an even smaller set of data within GSE122505 to see if we could find a concise set that was still relevant to our feature selection. After many hours of parsing through the data, we came across an additional dataset labeled “GSE9476” that was a subset of GSE122505. This subset was easy to separate from the superset but yielded similarly less useful results.

After having little accomplishment with the dataset in the Jupyter Notebook, we made attempts to take the data and use Python to reduce the size of the data. In doing so, we ran the same CASE (AML) and CONTROL (non AML) data in Python and received more promising and detailed results.

# Results

Our attempts at trying to gain a good dataset within the R language of the Jupyter Notebook were successful, but we were unable to run substantial enough tests on the data due to many mitigating factors of the Jupyter Notebook and its consistent kernel crashes. We then moved on to Python to employ new ways of interpreting the data and reducing its size before finding and running more algorithms that would yield better and more accurate models.

## Using Jupyter Notebook and R programming on the first subset GSE122505 (Dataset 1)

We did manage to construct a significant enough attempt toward using Forward Stepwise Logistic Regression of the dataset to find a certain subset of genes that yielded the lowest p-values. They were CREG1, ERLIN1, MAPKAPK3, CD22, CD79A, KIT, POU2AF1, CD79B, CFD, and CD19 seen if Figure 1. The Forward Stepwise Logistic Regression algorithm when run yielded details seen in Figure 2. The data was ran with training data and testing data as a 70% split with 30%. The data when ran through a correlation function yielded a histogram with two distinct peaks, clearly representative of two distinct sets of CASE (AML) and CONTROL (non AML) data as seen in Figure 3.

Table

Description automatically generated with medium confidence**Figure 1.**Lowest p-value genes, Forward Stepwise Logistic Regression

Table

Description automatically generated**Figure 2.**Forward Stepwise Logistic Regression.

Chart, histogram

Description automatically generated**Figure 3.**Correlation histogram of CASE (AML) and CONTROL (non AML) data

## Using Python on the first subset GSE122505 (Dataset 1)

We later used Python running on local hardware to run the same CASE (AML) and CONTROL (non AML) data on a variety of different algorithmic models. In particular, we used Correlation, Principal Component Analysis (PCA) and t-test as different ways to get good sets of 10 different best genes. The accuracy is listed in Figure 4. Using scikit-learn, pyreadr, matlabplot, pandas, itertools, and nubpy we ran five different models with an 80% to 20% split on testing and training data. With the genes gained from correlation, we ran Logistic Regression and got 90% accuracy. With Support Vector Machines (SVM) we received 91% accuracy. With K- Nearest Neighbor (KNN) we received 95%. With Decision Trees, we received 95%, and with Ensemble models, we received 93%. The same methodology was used with PCA as well as t-test. By far the highest accuracy received was with 97% as an Ensemble model with PCA. The Ensemble method used in Python is essentially a ‘majority rule’ implementation based on the other models presently being used to analyze the dataset.

**Figure 4.**Correlation, PCA, and t-test Algorithm tests.

Text

Description automatically generated

A good way to see the correlation data would be through a histogram such as that in Figure 5.

**Figure 5.**Correlation Histogram of data

Chart, histogram

Description automatically generated

**Figure 6.**Decision Tree representation of top 10 correlated genes.

Diagram

Description automatically generated with medium confidence

Finally, we see the Decision Tree that was generated from the correlation data. Everywhere there was a split in the data, the rule at the top of the node tells why the data was split there. If the rule is true, move to the left, and if the rule is false, move to the right. The gini impurity decides the likelihood of the new, random data being misclassified if given a random class label according to the distribution of the class in the dataset. The split in the data was chosen to maximize node purity. The sample size was set as 2000 samples, and below that, the number of two different classes in the node can be seen as well as the majority class in that node after the split. Here, the blue nodes labeled as “class = Case” are representations of the top 10 correlated genes.

# Discussion

Our attempts at trying to gain a good dataset within the R language were successful, but only so far as running a few algorithms. The use of Python on a local machine proved to be significantly more valuable in getting accurate models. The consistency of Jupyter Notebooks having kernel failure roughly 120 to 150 times throughout the process of running data was a significant limitation to the study. This can be understandable though since the usage of Python on a local machine required 10 gigabytes of memory usage in order to get it to run smoothly, and that still took on average around 200 seconds for the program to run. The Jupyter Notebook running R would timeout after about 10 seconds.

The data that was used specifically for this trial was Dataset 1 GSE122505 from the superset GSE122517. Acquiring this data was a significant hurdle insofar as the usage of Bioconductor and GEOquery ended up not be useful because the data was unable to be reached. In the end, it was necessary to reach out directly to the author to acquire the data, download it locally, and upload it into Jupyter Notebooks that way.

Results from other teams yielded similar sets of results to that of our R data. However, they had better success in running multiple algorithms on their data using R. Group A, consisting of Anthony Owens, Jeffrey Stockman, Kenneth Copeland, Richard Le, and Xiaojie Li seem to have the most thorough set of detailed results with their set of 20 genes: KCTD3, BZW2, BMP1, SCML2, IGF2BP2, SCCPDH, UBE2E1, TEC, RUNX1, RCN1, CDK6, UCK2, CCNB1IP1, NREP, MYB, FADS1, PAICS, MAPK12, STMN1, FLT3. When we set out to make our algorithms, we used similar avenues, but gained different sets of genes. From what I can gather, this was due to the fact that we used different training data. It is important to note though, that our particular Python models did yield higher percentages of accuracy.

Group F, composed of Cynthia Pang, Lang Wang, Zhengyang Wang, Mary Yang, and Ruby Zhao also received entirely different sets of genes, those being NPR3, SHTN1, HOXA9, LILRA1, HPGDS, MS4A6A, KYNU, HOXA7, PROM1. This group also chose to focus on AML versus non-AML data before moving on to a smaller subset of M3-AML versus non-M3-AML. It seems that our group in particular was not the only one subject to the limitations of Jupyter Notebook’s consistent kernel crashes. Also, because they chose such a smaller subset of the data and only ran a couple of algorithms on their data, they reached a surprising accuracy of 99.6% with both KNN and SVM.

If our group was to go back and do this process again, we may wish to allocate more memory to the Jupyter Notebooks server before setting out on running algorithms on such a broad set of data. Otherwise, it might be in our interest to use the same Python implementation, but with additional subsets in the data to see if we can train more accurate models.

Declaration of Interests

There are no competing or conflicts of interest.

References

Warnat-Herresthal S, Perrakis K, Taschler B, Becker M et al. Scalable Prediction of Acute Myeloid Leukemia Using High-Dimensional Machine Learning and Blood Transcriptomics. iScience 2020 Jan 24;23(1):100780. PMID: 31918046

Warnat-Herresthal S, Perrakis K, Taschler B, Becker M et al. Scalable Prediction of Acute Myeloid Leukemia Using High-Dimensional Machine Learning and Blood Transcriptomics. iScience 2020 Jan 24;23(1):100780. PMID: 31918046

Stirewalt DL, Meshinchi S, Kopecky KJ, Fan W et al. Identification of genes with abnormal expression changes in acute myeloid leukemia. Genes Chromosomes Cancer 2008 Jan;47(1):8-20. PMID: 17910043