Springboard Data Science Career Track

Classification of Acute Leukemias Using Gene Expression Profiles

Capstone Project I

Caitlin McDonold

April 2019

Table of Contents

I. Introduction

One of the challenges of cancer treatment lies in determining which line of therapy will be most effective in treating the patient's particular type of cancer. A therapy regimen that works strongly against one type of tumor, specifically targeting and eradicating the malignant cells, may have no effect at all on a different tumor. As cancer therapies have become more targeted and specific, the need for accurate tools to classify tumors and discriminate between tumors based on their clinical and biological features has become paramount.

Traditionally, cancers have been classified based on where in the body they originated (i.e. lung cancer), and then further subdivided based on the patient's age, cell type, histology, and sometimes biomarkers such as hormone receptors. However, high-throughput technologies have yielded an unprecedented volume of new data on variations in DNA, RNA, or protein expression in many cancers. The scope and complexity of these new datasets requires the processing power of computer programs and algorithms to organize and analyze the data. By combining the abundance of DNA sequencing and gene expression data with machine learning algorithms, we may be able to discover new tumor subclasses and more accurately predict the therapeutic strategy that is most likely to be successful in achieving remission.

Acute leukemias are an example of morphologically similar tumors that have different clinical pathologies and can respond differently to clinical treatments. Both acute lymphoblastic leukemia (ALL) and acute myeloid leukemia (AML) are cancers of the bone marrow and blood, though they originate in different cell types; ALL develops from immature lymphocytes, while AML develops in early forms of myeloid cells. The distinction between ALL and AML has been well established over years of careful scientific study and characterization, and successful treatment depends on distinguishing between ALL and AML, as they respond best to different chemotherapy agents. However, no single test is yet sufficient to distinguish between ALL and AML; instead, an experienced hematopathologist is needed to interpret results of multiple complex and highly specialized analyses, and misclassifications do sometimes occur. While treating ALL with the ideal chemotherapy regimen for AML can lead to remission (and vice versa), the success rates are greatly diminished.

The goal of this project is to use microarray data of bone marrow samples obtained from acute leukemia patients to train a classifier to distinguish between the two types of acute leukemia. The trained classifier will then be used to classify new acute leukemia samples as either ALL or AML to aid doctors in diagnosis so that they may determine an appropriate chemotherapy regimen.

II. Overview of Dataset

i. Dataset Description

The dataset for this project comes from a proof-of-concept study published by Golub et. al., which demonstrated that new cases of AML and ALL could be classified by gene expression data gathered using DNA microarrays.[1] The initial 'training' dataset comprises gene expression data from 38 bone marrow samples obtained from acute leukemia patients; 27 samples are from ALL patients, and 11 are from AML patients. The samples were analyzed by Affymetrix microarrays containing probes from 6817 human genes. The independent dataset contains microarray data from an additional 34 samples from acute leukemia patients; these samples are much more diverse than those in the training dataset and include samples from peripheral blood rather than bone marrow, from a broader range of patients (both adults and children), and from different laboratories. Both the training and the independent datasets are publicly available on Kaggle for download (https://www.kaggle.com/crawford/gene-expression/home).

The training and independent datasets are stored in separate .csv files: 'data_set_ALL_AML_independent.csv' and 'data_set_ALL_AML_independent.csv'. The files contain columns labeled 'Gene Description' and 'Gene Accession Number,' followed by a column for each tumor sample, labeled by number (i.e. '1', '2', etc.). The entries of the sample columns are the experimental values from the DNA microarrays; larger values indicate higher gene expression levels. Each sample column is followed by a column labeled 'call' – the entries in these columns are either 'P' (present) or 'A' (absent) that indicate whether the specific gene was expressed in that sample. Both files contain data for 7129 hybridization probes; most of

[1] Golub, T.R. et. al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286, 531-537 (1999).

these correspond to individual genes, although a handful serve as controls. The training dataset contains tumor samples 1 through 38, while the independent dataset contains tumor samples 39 through 72.

A third .csv file, 'actual.csv,' contains two columns: 'patient' and 'cancer.' This file provides the labels for both the training and independent datasets; the 'patient' column contains entries from '1' to '72,' which correspond to the sample columns in the previous two .csv files, while the 'cancer' column labels each sample as either 'ALL' or 'AML.'

ii. Data Cleaning and Wrangling

The dataset was fairly clean and therefore easy to shape into useable form. I began by importing the training dataset into a pandas DataFrame, using the 'Gene Accession Number' column as the index. There were no missing or null values in the dataset, though I did need to remove extraneous columns and rows corresponding to the microarray control probes. As I am only interested in the actual gene expression values, the 'call' columns are unnecessary. I generated a list containing these column names (i.e. ['call.1', 'call.2', …]), then passed this list to the `drop()` method to remove the columns.

The rows corresponding to assay controls are also not useful. Conveniently, I discovered that the control probes were at the head of the dataframe and contained the string 'AFFX' in the gene accession number (Affymetrix manufactured the microarray chips used). An additional control was missing the 'AFFX' string but was clearly labeled as a control in the 'Gene Description' column. I created a list containing the index values for these control probes and then removed them from the dataframe. I also dropped the 'Gene Description' column, as it is not useful at this stage. Having done this, I then mapped the gene expression columns as integers and sorted the index so that the patient samples were in order from 1 to 38. Finally, I transposed the dataframe so that the tumor samples (independent observations) formed the rows and the genes (the features) formed the columns.

Another critical step to enable exploratory data analysis was to label each tumor sample with the type of cancer. To do this, I read in the 'actual.csv' file containing the cancer labels for each sample, setting the patient column as the index. I was then able to merge the gene

expression dataframe with the cancer key to label each sample with the appropriate form of cancer in my dataframe. As the cancer key contained the independent samples as well as the training samples, I used an inner join so that only the samples that appeared in both the gene expression dataframe and the cancer key would be retained in the new dataframe.

III. Exploratory Data Analysis

I began my analysis by exploring the dataset for variations and patterns in the gene expression profiles of AML and ALL tumors. I first grouped the data by cancer type and calculated the mean and standard deviation for both the ALL and AML tumors for each gene and saved these calculations as a new dataframe, 'stats'. I was then able to visually explore the mean gene expression level across all genes for both types of cancer. Figure 1 shows a scatterplot of the mean expression level of the ALL samples versus the AML samples for each gene in the dataset. One clear outlier datapoint has large negative mean expression values for both cancer types; this datapoint corresponds to the gene SOX12, which is a transcription factor. Raw

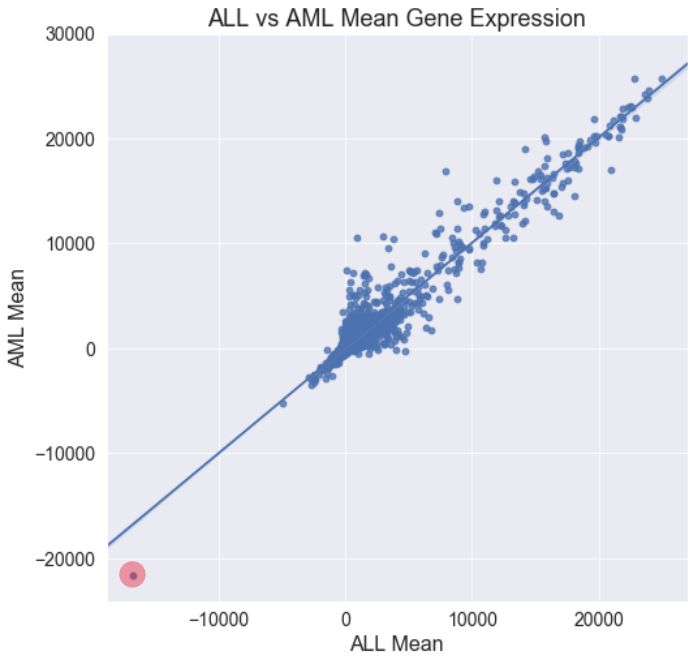

FIGURE 1: Mean gene expression level of ALL samples vs AML samples for each gene. Outlier datapoint for SOX12 gene expression is highlighted in red.

microarray data should be positive, though you can get negative values after data processing

steps such as subtracting out background signal. I was unable to discover what processing and/or transformations were applied to the raw dataset to yield the final values in the published dataset; however, it is likely that the extremely large negative values for SOX12 are either an error or an artifact of data processing that the original researchers used on the raw microarray data; therefore, this gene has been excluded from further analysis.

I next plotted the empirical cumulative distribution functions (ECDF) of the gene expression level for both the ALL and AML samples (Figure 2). As expected, both ECDF plots look similar for the different cancer types; only a small subset of the ~7000 genes tested in the microarrays are likely responsible for determining an ALL cell versus an AML cell. By zooming in on the ECDF plots, we can see that ~25% of the genes have negative expression levels (again, probably an effect of background subtraction on the raw dataset), ~50% of the genes (~3500 genes) have positive expression levels less than 500, ~12.5% have expression levels between 500 and 1000, and ~12.5% have expression levels greater than 1000 (Figure 3). he vast majority of the genes, over 70%, have low to moderate expression levels (between -500 to 500), while about 5% (~350 genes) have very high expression genes (over 4000).

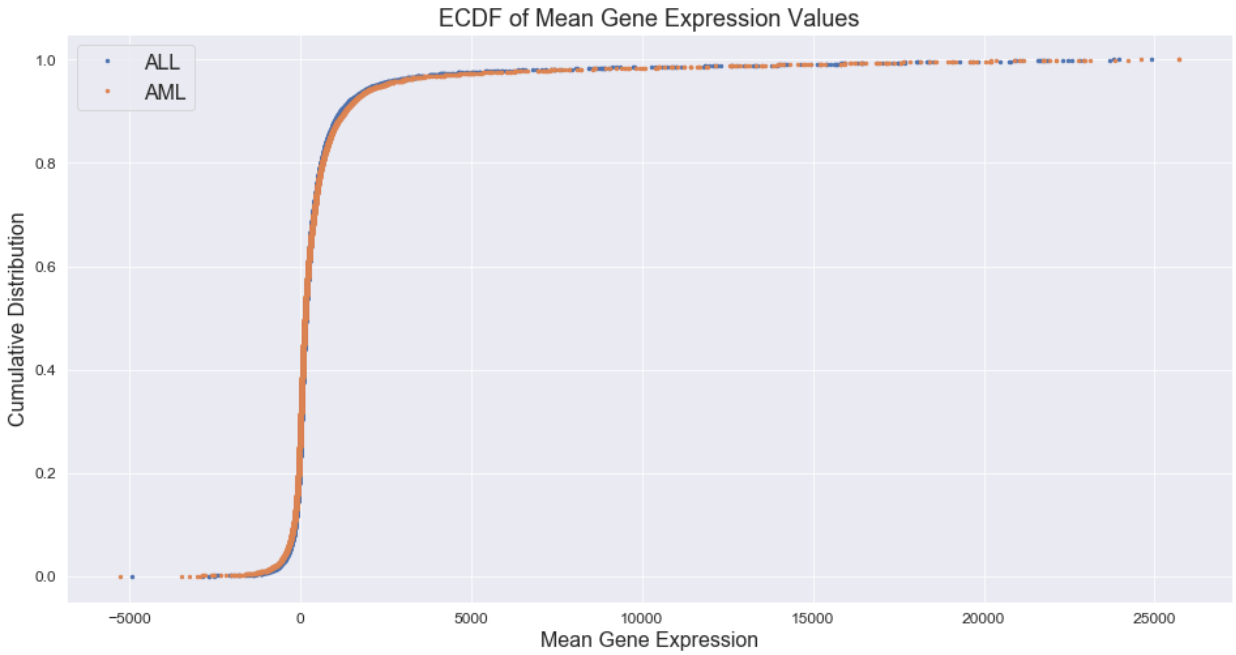

FIGURE 2: Empirical Cumulative Distribution Functions of gene expression levels for the ALL and AML samples.

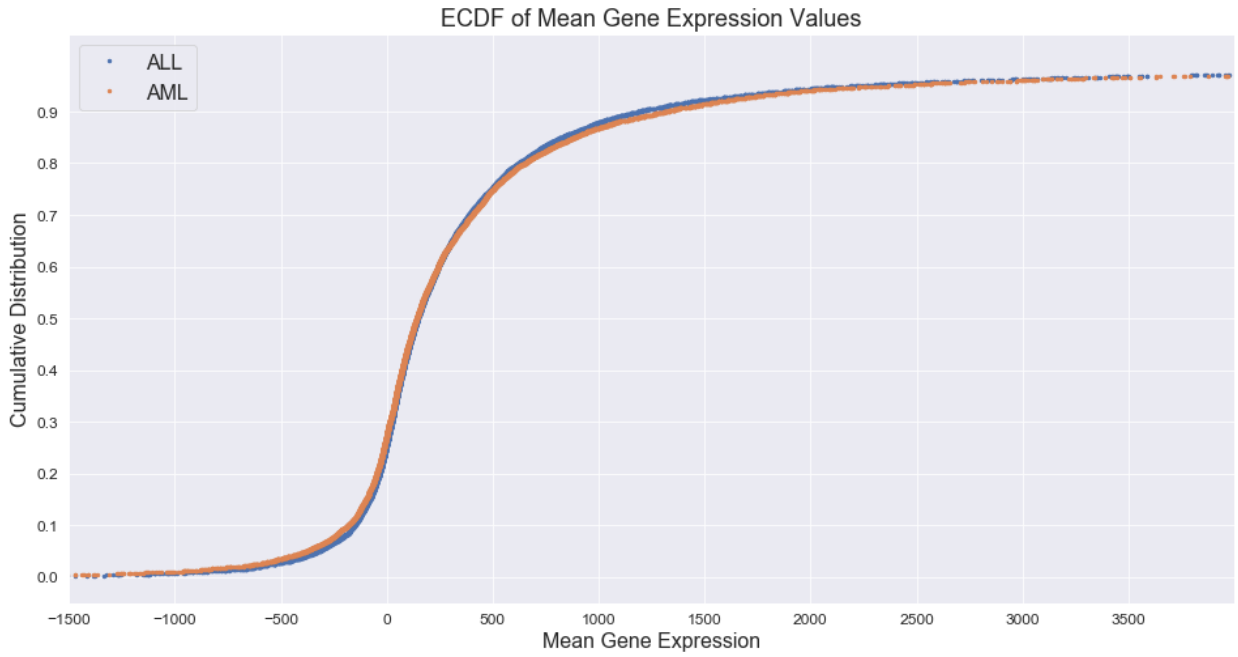

FIGURE 3: Zoom-in view of ECDFs for the ALL samples and the AML samples.

While most of the genes in the dataset have similar expression levels across all samples regardless of cancer type, a small subset of the genes should be more highly expressed in either the ALL or the AML samples. Identifying these differentially expressed genes will allow me to train classifiers using only the genes most highly correlated with the ALL/AML class distinction. The scatterplot of mean gene expression in ALL samples vs AML samples gives a rough picture of the number of genes that are differentially expressed between the two cancer variations; the points that lie above the main regression line are more highly expressed in the AML samples, while those that lie below the line are more highly expressed in the ALL samples. A rough estimate of correlation could be calculated using the ratio of mean gene expression in ALL samples over AML samples; however, the simple ratio of mean expression does not account for the spread of the individual samples for each cancer type. To better identify the genes that are most highly correlated with the ALL and AML samples, I need to account for the both the mean expression level and the standard deviation of the mean for the ALL and AML samples. Therefore, I estimated the correlation between a gene and the cancer type by calculating the difference between the ALL and AML means divided by the sum of the standard deviation of the means:

$$estimated\ correlation = \frac{\mu_{ALL} - \mu_{AML}}{\sigma_{ALL} - \sigma_{AML}}$$

Genes that are more highly expressed in the ALL samples will have positive estimated correlation values, while genes that are more highly expressed in the AML samples will have negative estimated correlation values. A histogram of the correlation estimates for all genes shows that the correlation values are centered around 0 and are roughly normally distributed (Figure 4).

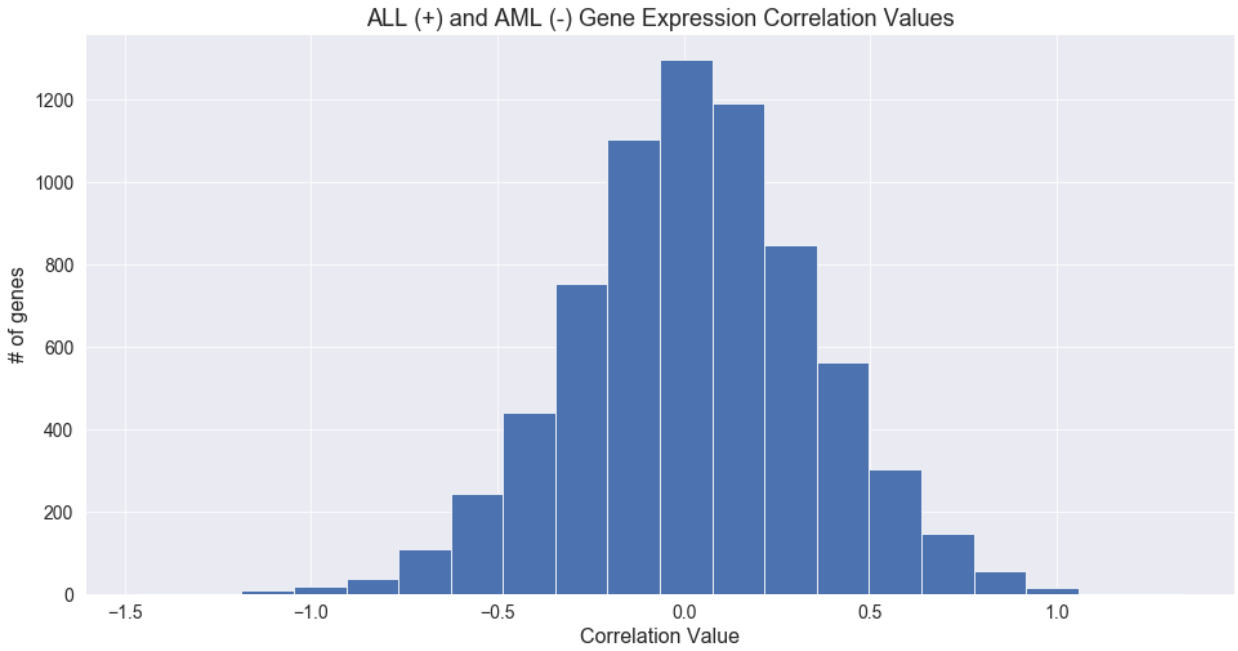

FIGURE 4: Histogram of estimated correlation values for all genes.

After sorting the genes by their correlation estimates, I was then able to pull out the 25 genes most correlated with each cancer type. The location of these genes on the scatterplot of mean gene expression in ALL v AML samples shows that many of genes correlated with cancer type have expression levels between 0 to 5000 (Figure 5). We can also see that the genes lie closer to the regression line at lower expression levels but further away as the expression level increases. Many of genes correlated with the AML samples have higher mean gene expression levels, and 8 of the 25 genes have expression levels over 5000. On the other hand, all of the genes correlated with the ALL samples have expression levels under 5000, and only 6 genes have expression levels over 2000 (Table 1).

I also generated ECDFs of the correlation estimates for both the ALL and the AML samples. As the sign of the correlation value indicates correlation with either ALL or AML, the ECDF of the correlation values for the ALL samples is simply the inverse of the ECDF for the

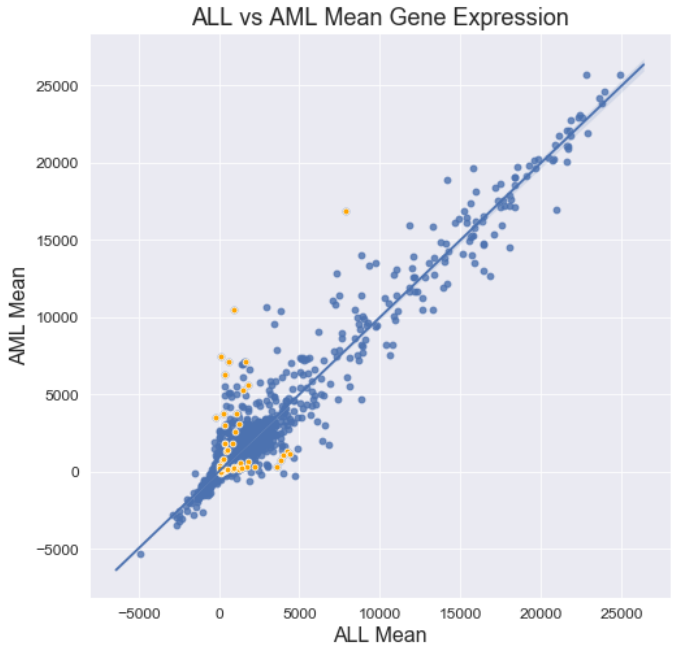

FIGURE 5: Location of the 50 genes with highest correlation estimates in the scatterplot of mean gene expression level of ALL samples vs AML samples.

AML samples. Thus, the inverse of the correlation estimates was used to calculate the ECDF of ALL cancer so that both the ALL and AML ECDFs could be plotted alongside one another (Figure 6). Examination of the ECDF plots for the correlation estimates show that ~100 genes

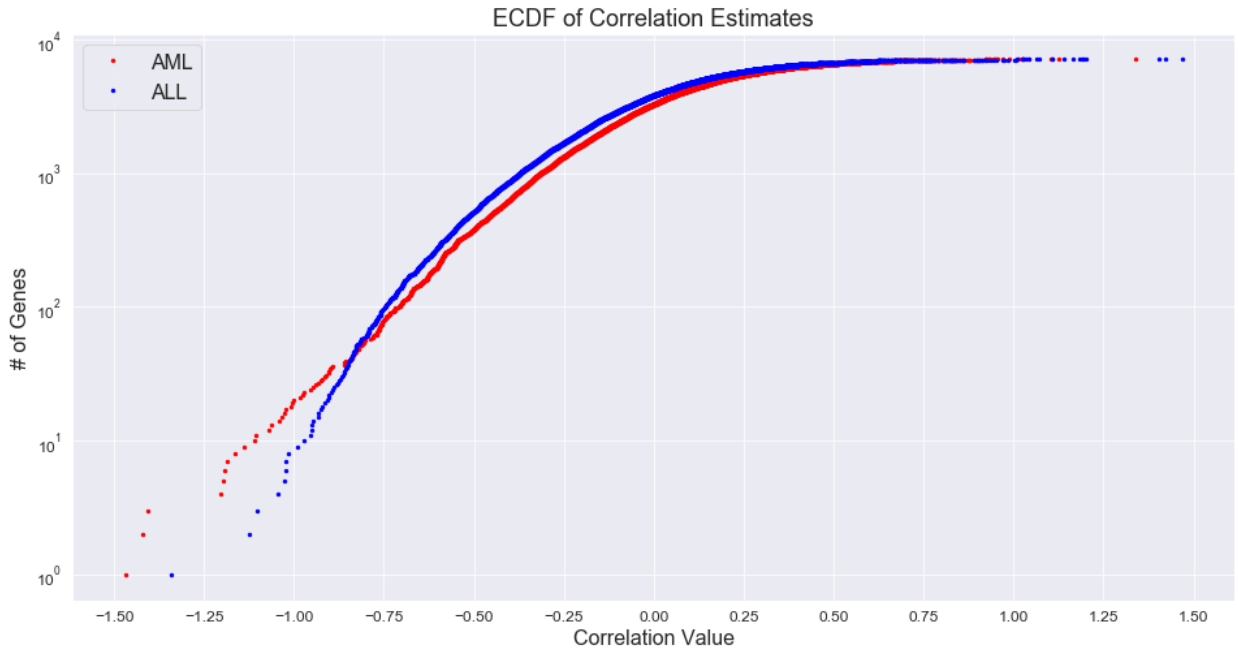

FIGURE 6: Empirical cumulative distribution function of correlation values for ALL and AML samples.

TABLE 1: Statistical summary information for top 50 genes with the highest correlation estimates

	ALL_mean	AML_mean	ALL_stdev	AML_stdev	mean_diff	std_sum	correlation
M55150_at	810.3	1836.27	338.18	360.89	-1025.98	699.06	-1.47
U50136_rna1_at	977.78	2562.18	324.69	789.75	-1584.4	1114.44	-1.42
X95735_at	349.89	3023.64	395.52	1506.46	-2673.75	1901.98	-1.41
M16038_at	375.37	1811.64	240.3	953.69	-1436.27	1193.99	-1.2
M23197_at	175.04	767.27	84.09	411.1	-592.24	495.19	-1.2
M84526_at	-177.41	3483.0	176.45	2896.84	-3660.41	3073.29	-1.19
Y12670_at	411.15	1294.45	212.37	533.21	-883.31	745.57	-1.18
U82759_at	241.37	850.73	239.56	283.87	-609.36	523.43	-1.16
D49950_at	75.37	243.91	54.69	93.18	-168.54	147.87	-1.14
M27891_at	144.44	7423.55	427.02	6135.52	-7279.1	6562.54	-1.11
X17042_at	1642.81	7108.73	1774.33	3167.84	-5465.91	4942.17	-1.11
U12471_cds1_at	152.22	287.18	67.4	58.76	-134.96	126.16	-1.07
U46751_at	1512.15	5291.27	534.44	3017.11	-3779.12	3551.55	-1.06
Y00787_s_at	892.22	10496.91	2052.56	7173.98	-9604.69	9226.54	-1.04
L08246_at	1066.56	3767.45	668.59	1941.52	-2700.9	2610.11	-1.03
M80254_at	33.74	428.64	106.58	277.66	-394.9	384.25	-1.03
M62762_at	1238.59	3070.55	689.8	1101.22	-1831.95	1791.03	-1.02
M81933_at	64.26	248.91	104.27	78.88	-184.65	183.15	-1.01
M96326_rna1_at	571.11	7117.55	527.23	5984.23	-6546.43	6511.46	-1.01
M28130_rna1_s_at	355.15	6298.91	939.0	4999.32	-5943.76	5938.32	-1.0
M63138_at	1779.26	5599.36	795.5	3088.19	-3820.1	3883.69	-0.98
M11147_at	7885.37	16838.09	3956.12	5236.04	-8952.72	9192.16	-0.97
M57710_at	262.85	3760.36	516.34	3084.56	-3497.51	3600.9	-0.97
M81695_s_at	548.19	1439.64	248.77	686.58	-891.45	935.35	-0.95
X85116_rna1_s_at	382.04	1842.73	329.23	1214.66	-1460.69	1543.89	-0.95
J05243_at	867.63	134.0	627.05	197.24	733.63	824.29	0.89
Z69881_at	2196.37	278.45	1883.6	269.31	1917.92	2152.91	0.89
U20998_at	1786.78	690.27	1007.18	216.46	1096.51	1223.64	0.9
X63469_at	315.04	88.0	172.58	79.25	227.04	251.82	0.9
D38073_at	913.41	324.73	473.19	178.77	588.68	651.95	0.9
U29175_at	1202.67	382.55	775.04	128.14	820.12	903.18	0.91
M91432_at	899.89	204.45	664.69	96.58	695.43	761.27	0.91
S50223_at	354.93	51.18	228.41	101.18	303.74	329.59	0.92
AF009426_at	67.85	-49.0	92.59	33.79	116.85	126.38	0.92
X15949_at	294.67	42.91	244.04	26.44	251.76	270.48	0.93
X52142_at	143.3	-70.73	141.72	87.62	214.02	229.35	0.93
Z15115_at	4213.0	1365.27	2518.89	493.94	2847.73	3012.83	0.95
M28170_at	210.11	-21.27	188.93	54.77	231.38	243.7	0.95
L47738_at	1251.74	192.73	897.85	216.42	1059.01	1114.27	0.95
U32944_at	1707.81	328.0	1296.54	148.49	1379.81	1445.02	0.95
M31523_at	1331.7	310.27	906.88	144.82	1021.43	1051.69	0.97
D26156_s_at	1306.0	578.82	520.5	214.69	727.18	735.2	0.99
U09087_s_at	180.89	46.91	106.37	25.72	133.98	132.09	1.01
M31211_s_at	553.48	154.64	287.57	102.4	398.85	389.97	1.02
L13278_at	127.93	0.55	93.87	30.56	127.38	124.43	1.02
X74262_at	1416.96	255.73	971.84	158.08	1161.24	1129.92	1.03
M92287_at	4029.48	1073.64	2352.36	481.48	2955.85	2833.83	1.04
U05259_rna1_at	3562.93	288.18	2726.43	242.03	3274.74	2968.46	1.1
X59417_at	4361.37	1138.18	2291.29	574.69	3223.19	2865.98	1.12
U22376_cds2_s_at	3863.3	702.36	1999.78	360.34	3160.93	2360.12	1.34

have correlation values of 0.75 or higher for both the ALL and AML samples.

Next, I went back to the original dataset to look at the spread of the gene expression data for each of the top 50 genes. For the following plots, I only looked at the gene expression data from the correlated cancer samples (Figures 7, 8). Overall, the genes in both cancer sets show a wide variety of expression levels, from genes with very low expression to those with quite high expression levels. However, as see in the scatterplot earlier, the more genes associated with AML have higher expression levels, and the maximum expression level is over twice as high as the maximum gene expression level in the ALL set. Indeed, if the top 50 genes were plotted

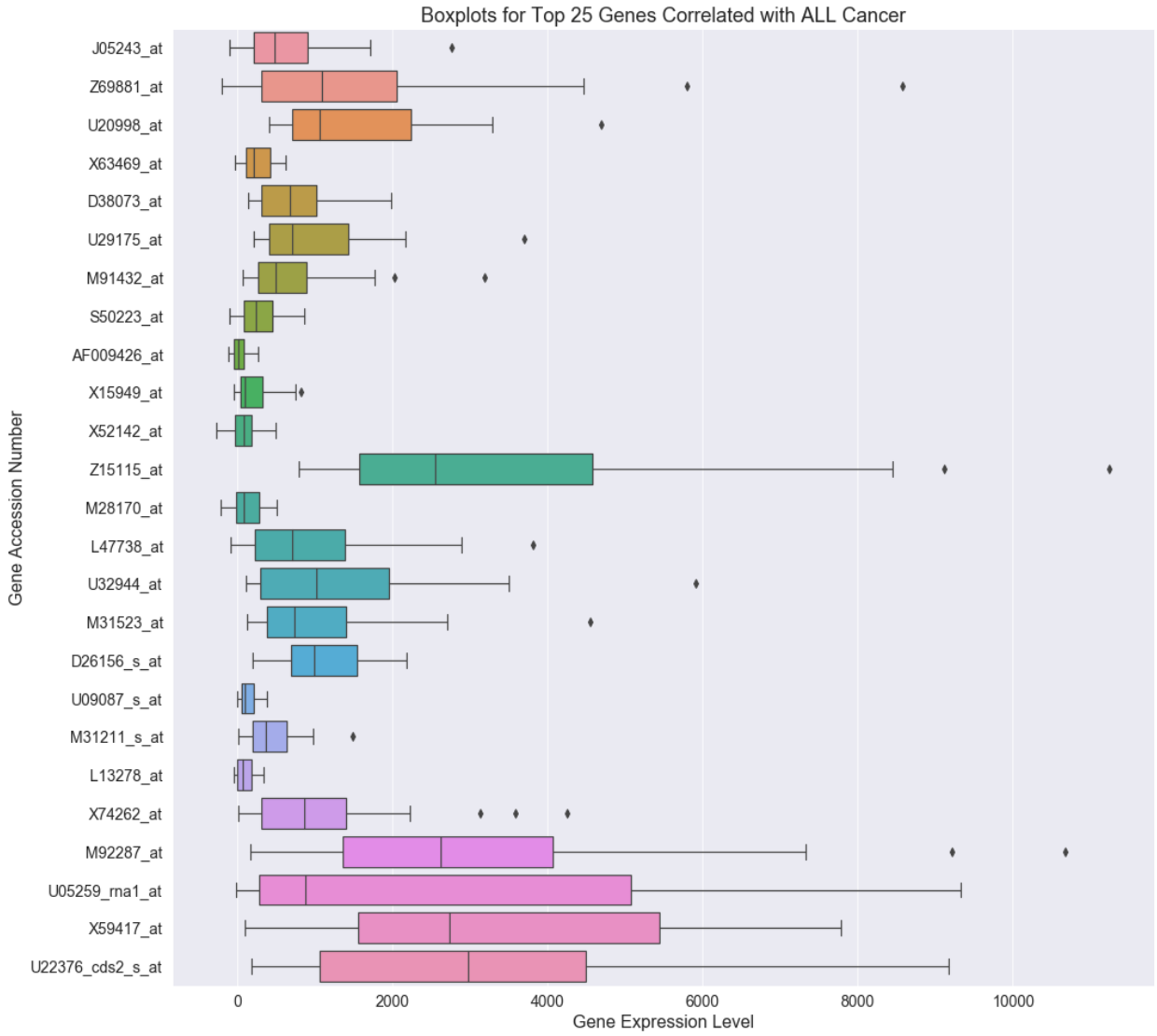

FIGURE 7: Boxplots of the gene expression data from the ALL samples for the top 25 genes correlated with the ALL samples.

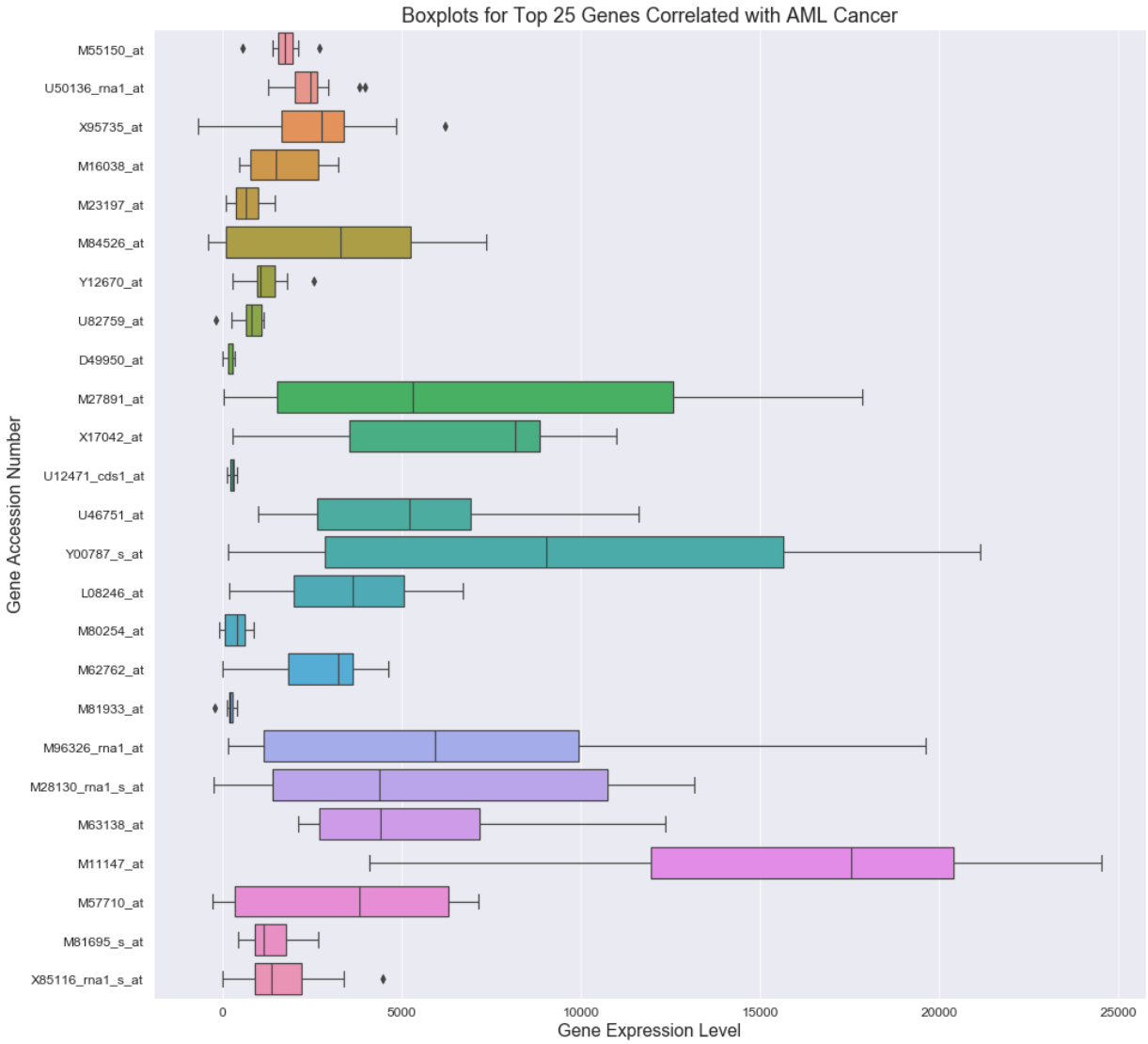

FIGURE 8: Boxplots of the gene expression data from the AML samples for the top 25 genes correlated with the AML samples.

together, the boxplots for all of the ALL-associated genes would fit into the first half of the plot. Another general feature is that the spread of the gene expression data increases as the expression level increases, suggesting that the data is logarithmically distributed and should be log-transformed to normalize data prior to further analysis. Finally, we can also see that about a third of the genes have between 1-3 outlier values. If necessary, these outlier values could be analyzed later on to refine the gene selection (i.e. feature selection) process.

One of the best ways to visualize gene expression data is by using a heatmap, which makes it very easy to see patterns in the gene expression levels across many samples. As the expression levels of the top 50 genes vary widely, each gene was first normalized using the

maximum and minimum expression levels so that patterns across the individual samples could be detected. As you can see in the heatmap below, there is a clear difference in the gene expression pattern between the ALL samples and AML samples (Figure 9). As expected, the top 25 genes are more highly expressed in the ALL samples compared to the AML samples, while the bottom 25 genes are more highly expressed in the AML samples.

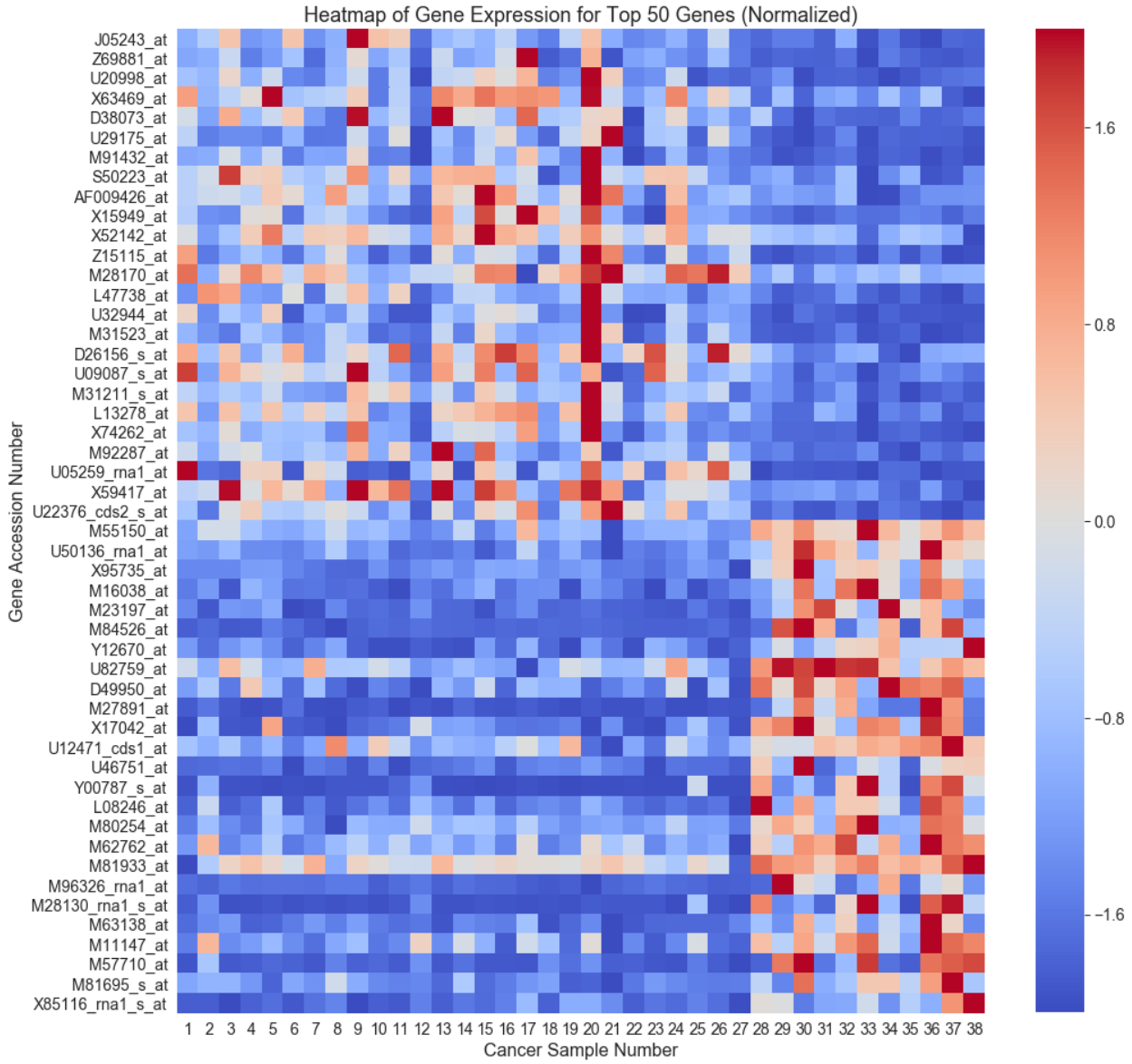

FIGURE 9: Heatmap of normalized gene expression data for the top 50 gene associated with the ALL/AML distinction. Samples 1-27 are from patients with ALL, while samples 28-38 are from patients with AML cancer. Red indicates upregulated (enhanced) gene expression, while blue indicates downregulated (suppressed) gene expression.

Finally, I computed p-values and confidence intervals around the correlation estimates for the top 50 genes. To do this, I tested the null hypothesis that the gene expression levels of the ALL and AML samples have identical probability distributions. The first step was to combine the ALL and AML gene expression data for each gene, generate permutation samples by randomly splitting the gene expression data into ALL and AML samples, and then calculate the correlation values on these permutation samples. Correlation estimates for 10,000 permutation samples were calculated per gene, and the p-values (the probability of the getting the observed correlation estimate if there is no difference between ALL and AML cancer) were then calculated by dividing the number of permutation samples that resulted in correlation estimates at least as extreme as the observed value by the total number of permutations performed. At most, only 1-2 permutations out of 10,000 yielded correlation estimates equal to or greater than the observes correlation estimate. Therefore, the observed p-values less than 0.001 for all of the 50 genes most associated with cancer type (Table 2). Bootstrap analysis was used to calculate confidence intervals for the correlation estimates for each of the top 50 genes. For these calculations, two bootstrap replicates were generated for each gene, one for the ALL samples and one for the AML samples; these bootstrap replicates were then used to calculate the resulting correlation value. For each gene, 10,000 correlation estimates were calculated, and the resulting distribution was used to determine the 95% confidence interval of the correlation estimate for that gene (Table 2).

TABLE 2: Probabilities and 95% confidence intervals for correlation estimates for top 50 genes with the highest correlation estimates.

	correlation	p-value	95% CI
M55150_at	-1.47	0.0	[-2.29 -1.13]
U50136_rna1_at	-1.42	0.0	[-2.23 -1.06]
X95735_at	-1.41	0.0	[-2.24 -1.09]
M16038_at	-1.2	0.0	[-1.96 -0.85]
M23197_at	-1.2	0.0	[-1.87 -0.87]
M84526_at	-1.19	0.0	[-2.21 -0.73]
Y12670_at	-1.18	0.0	[-1.87 -0.92]
U82759_at	-1.16	0.0	[-1.99 -0.75]
D49950_at	-1.14	0.0	[-2.22 -0.7]
M27891_at	-1.11	0.0	[-1.84 -0.77]
X17042_at	-1.11	0.0	[-2.2 -0.66]
U12471_cds1_at	-1.07	0.0	[-1.71 -0.75]
U46751_at	-1.06	0.0	[-1.99 -0.67]
Y00787_s_at	-1.04	0.0	[-1.89 -0.63]
L08246_at	-1.03	0.0	[-1.79 -0.65]
M80254_at	-1.03	0.0	[-1.8 -0.64]
M62762_at	-1.02	0.0	[-1.87 -0.62]
M81933_at	-1.01	0.0	[-1.43 -0.8]
M96326_rna1_at	-1.01	0.0	[-1.72 -0.67]
M28130_rna1_s_at	-1.0	0.0	[-1.77 -0.62]
M63138_at	-0.98	0.0	[-1.58 -0.7]
M11147_at	-0.97	0.0	[-1.69 -0.6]
M57710_at	-0.97	0.0001	[-1.89 -0.54]
M81695_s_at	-0.95	0.0	[-1.55 -0.61]
X85116_rna1_s_at	-0.95	0.0	[-1.5 -0.67]
J05243_at	0.89	0.0	[0.64 1.32]
Z69881_at	0.89	0.0	[0.69 1.3]
U20998_at	0.9	0.0001	[0.64 1.31]
X63469_at	0.9	0.0001	[0.62 1.32]
D38073_at	0.9	0.0	[0.57 1.5]
U29175_at	0.91	0.0	[0.68 1.34]
M91432_at	0.91	0.0	[0.72 1.34]
S50223_at	0.92	0.0001	[0.65 1.35]
AF009426_at	0.92	0.0	[0.67 1.31]
X15949_at	0.93	0.0	[0.69 1.28]
X52142_at	0.93	0.0	[0.71 1.34]
Z15115_at	0.95	0.0	[0.73 1.31]
M28170_at	0.95	0.0001	[0.63 1.45]
L47738_at	0.95	0.0	[0.72 1.34]
U32944_at	0.95	0.0	[0.72 1.38]
M31523_at	0.97	0.0	[0.75 1.48]
D26156_s_at	0.99	0.0001	[0.71 1.41]
U09087_s_at	1.01	0.0	[0.75 1.42]
M31211_s_at	1.02	0.0	[0.77 1.52]
L13278_at	1.02	0.0	[0.73 1.5]
X74262_at	1.03	0.0	[0.82 1.45]
M92287_at	1.04	0.0	[0.83 1.42]
U05259_rna1_at	1.1	0.0	[0.82 1.55]
X59417_at	1.12	0.0001	[0.85 1.56]
U22376_cds2_s_at	1.34	0.0	[1.04 1.87]

IV. In-Depth Analysis and Modeling Results

i. Comparison with Published Results

For the first part of my analysis, I wanted to replicate the analysis performed by the authors of the original paper by using the training set to train a classifier and then testing the classifier using the independent set. In the original paper, the classifier assigned a new sample to one of the two classes (ALL or AML) using a fixed subset of genes that have a high correlation with the class distinction. The classifier then predicted the class of a new sample based on the expression level of each of the chosen genes in the new sample, with each gene casting a weighted vote, which was calculated by multiplying the correlation value for that gene by the difference between the expression level in the new sample and the overall mean expression level for that gene. The weighted votes for the new sample were then summed to determine the final vote and assign each sample to one of the two classes. A measure of the margin of victory, called the prediction strength, was also calculated for each new sample. In the original paper, the authors set a prediction strength (PS) cutoff of < 0.3, and any predictions with PS less that this cutoff were deemed 'uncertain.'

To recreate the classifier used in the original paper as closely as possible, I first wrote a function to fit the classifier which takes the dataset (X) and labels (y) as inputs and returns a dictionary containing the correlation estimate ($corr_{gene}$) and overall mean (μ_{gene}, the mean across both ALL and AML samples) for each gene. A second function then predicts the labels when passed a dataset (X) and the dictionary generated by the fit function (called *weights*) and returns a DataFrame containing the predicted labels and prediction strengths indexed by sample. The predict function uses the following formula to calculate a weighted vote for each of the genes in a given sample: $corr_{gene} * (x_{gene} - \mu_{gene})$. The weighted votes are then summed to give the final vote, and the class is assigned based on the sign of the final vote: samples with positive final votes were assigned to the ALL class, whereas samples with negative final votes were assigned to the AML class. Finally, the prediction strength was calculated for each sample using the following formula: $(|V_{win}| - |V_{lose}|)/(|V_{win}| + |V_{lose}|)$, where $|V_{win}|$ is the absolute value of the sum of the weighted votes for the winning class and $|V_{lose}|$ is the absolute value of the sum of the weighted votes for the losing class.

Having written functions to fit the classifier and predict labels for a new dataset, I then trained my classifier using the training dataset, predicted the labels for the independent training set, and compared my results to that of the original paper. I first assessed my classifier using the 50 genes I had identified as having the highest correlation values as the features (see Table 1) by slicing out the gene expression data for just these 50 genes from both the training and independent datasets and passing these data to my fit and predict functions, respectively. My model correctly predicted 37 of 38 samples when tested on the training dataset (validation), yielding 97.37% accuracy. The prediction strengths for 2 of these samples were < 0.3, which included the incorrectly labeled sample. The original paper found that 36 of the 38 samples were correctly predicted, with two being labeled 'uncertain' (PS < 0.3); however, they do not indicate the accuracy of the uncertain samples. Thus, for the training set, my classifier performed just as well as the original paper results. For the independent training set, my model correctly predicted 33 out of 34 samples, yielding an accuracy of 97.06%. The prediction strength of 4 of these samples was < 0.3; interestingly, however, all these samples were assigned to the correct type of cancer, and the prediction strength of the one incorrectly labeled sample was 0.62, well above the threshold set in the original paper. In contrast, the original paper reported that 29 of the 34 samples had prediction strengths > 0.3, and all of these were correctly labeled – again, the paper does not report whether the 5 samples below the threshold were correctly labeled.

I also assessed the performance of my model when using reduced sets of predictor genes. When my model was trained and assessed using 20 predictor genes, 33 of the 34 independent samples were again predicted correctly, and 4 of the samples had prediction strengths < 0.3. However, while the same sample was incorrectly labeled by the 50-gene model and the 20-gene model, the samples with low prediction strengths varied; the incorrectly labeled sample for the 20-gene model had a prediction strength of 0.22, suggesting that perhaps some of the genes included in the larger predictor set were not well correlated with the class distinction across a wide variety of samples (i.e. the correlation is only seen in the training dataset and is not generalizable to new data). If I further cut the set of predictor genes down to 10, the model again correctly labeled 33 of the 34 samples and identified 4 samples with prediction strengths < 0.3, but these samples were again different from the previous classifiers. The 10-gene model incorrectly labeled a different sample that the 20- or 50-gene models, and the prediction strength for this sample was above the threshold at 0.43. Additionally, the sample that had been

incorrectly predicted by the 20- and 50-gene models had a very low prediction strength of 0.06, although the 10-gene model did label it correctly. In general, however, my classifier was very accurate at assigning the type of cancer to new samples, even when using a very limited set of genes as predictors.

Overall, my model performed well when compared against the results reported in the original paper. My results also lead me to question the usefulness of the prediction strength calculated in the original paper, as the prediction strength excluded correctly predicted samples and did not catch all incorrect predictions in my model. As the original paper neither included tables with the calculated prediction strengths nor specified which samples were categorized as 'uncertain,' I am unable to perform a more detailed analysis of my results against the original paper. However, comparison of the top 50 genes I identified to those used in the original paper reveals a difference of 10 genes (20%) in my dataset that were not found in the original paper. The reason for this difference is unclear, as the original paper does not provide enough detail regarding the calculations used or the results. However, the differences in the genes used as predictors might account for the unreliability of using the prediction strength as measure of prediction reliability in my classifiers. Regardless of these differences, my model, with an accuracy of over 97%, performed very well when taken on its own and when compared to the results of the original paper.

ii. Machine Learning Model Comparisons

As mentioned previously, the dataset I have been working with was split into two files, one that contains the training dataset and one that contains the independent dataset. Up to this point, I have been working exclusively with the training dataset for exploratory data analysis and training of the initial model, in line with what was done in the original paper. Now, rather than using only the training set to train various classifiers, I have combined the training and independent datasets into one dataframe and created my own train/test splits on the data that I have used for hyperparameter tuning, cross-validation and model assessment. The training datasets, X_train and y_train, were used for hyperparameter tuning and cross-validation, while the test sets, X_test and y_test, were reserved for the final performance assessment. I have also used scikit learn to train, tune, and assess several different types of classifiers, rather than writing customized functions to fit and predict data as I did in the previous section.

In the following section, I have assessed the performance of four different machine learning algorithms: k-Nearest Neighbors, Logistic Regression, a Support Vector Machine (SVM), and a Random Forest. In order to train these classifiers, I have again used only the set of 50 highly-correlated genes to reduce the number of features in the dataset. Before the data undergoes any preprocessing or is used to fit the models, it must first be split into train and test sets. Preprocessing is also a critical step in the subsequent analyses, as many of these classifiers use measures of distance between features to inform the models. Gene expression data can vary widely from gene to gene (as indeed is seen in the boxplots in Figures 7 and 8); if gene expression levels are not normalized prior to training a classifier, the genes with much higher expression levels will unduly influence the models. Therefore, the gene expression data were normalized prior to training classifiers or predicting labels to standardize the variance across the gene features. Additionally, it is critical for the train/test split to occur before any preprocessing steps such as normalization or min/max scaling to avoid data leakage; for example, normalizing the entire dataset before the train/test split means that the transformer will have knowledge of the full distribution of the data when calculating the scaling factors. Fortunately, data leakage can be avoided by performing any preprocessing steps within the cross-validation folds.

In order to compare the models against one another, I needed to choose the metrics I would use to evaluate the performance of each model. A common classification metric is the measure of accuracy, i.e. the percentage of correctly classified samples. The `score()` function of all four classifiers tested returns the mean accuracy of the predicted labels. The precision and recall scores are also very useful for assessing classification predictions. Precision is the percentage of samples classified as positive that are not false positives; that is, it measures the ability of the classifier to not mislabel true negative samples as positive samples. Recall is the ability of the classifier to find all the positive samples, i.e. the percentage of true positives identified over the total number of positive samples in the dataset. The precision and recall scores can be obtained using the metrics function `classificationreport()`.

For my initial analysis of the four models, I used `train_test_split()` with the random state set to 42 and the test size set to 0.3 to split my dataset so that 70% of the dataset was used for training and 30% was reserved for testing. However, with this particular split, each of the four models returned exactly the same accuracy, precision, and recall scores (Table 3).

TABLE 3: Accuracy scores and metrics for the four different classifiers with random_state = 42 and test_size = 0.3

Accuracy	Class	Precision	Recall	F1-score
95.45%	ALL	0.93	1.00	0.97
	AML	1.00	0.88	0.93

I then changed the random state to 1582 and the train/test split to 75/25 to see whether a different split would result in performance differences between the two models. The results of these analyses are summarized below:

k-Nearest Neighbors: For the k-nearest neighbors classifier (`KNeighborsClassifier()`), I first performed grid search cross-validation to assess the number of neighbors from 1 to 20 to optimize the n_neighbors hyperparameter. I then fit the model to my training dataset using the optimized parameter (n_neighbors = 2), predicted the labels for the test dataset, and computed the accuracy score and classification report for the model. The resulting model had a fairly high accuracy score of 88.89%, though the ALL precision and AML recall scores were lower than the previous model (Table 4). Additionally, the confusion matrix for the model shows that two AML samples were incorrectly labeled as ALL samples, out of 18 total samples in the test set. The low number of neighbors for this model suggest that the model is overfitting to this particular training set; in comparison, the best n_neighbors parameter for the earlier kNN model (with random state = 42) was 11. However, the differences are still relatively small, and both models display reasonably good metrics scores.

Logistic Regression: The logistic regression classifier (`LogisticRegression()`) performed well out-of-the-box in scikit learn with no hyperparameter tuning for both of the train/test splits. The accuracy, precision and recall scores were roughly the same in the two models, and only one AML sample was misclassified as an ALL sample.

Support Vector Classification: For the support vector classifier (SVC), I again used grid search cross-validation to tune the C (error penalty) and gamma (kernel coefficient) parameters. With the optimized parameters (C = 0.75; gamma = 2), the SVC yielded the same accuracy as the logistic regression classifier, though the precision/recall scores varied slightly, as the SVC model misclassified an ALL sample as an AML sample rather than the other way around.

Random Forest: Finally, I assessed a random forest classifier (`RandomForestClassifier()`). Grid search cross-validation was used to optimize the n_estimators (number of trees) parameter, and the resulting model with n_estimators = 25 was used to assess the test data. The random forest classifier performed better on the second train/test split than on the first, correctly classifying all the test samples. However, the increased accuracy is only due to correcting a single misclassification and should thus not be weighed too heavily.

TABLE 4: Accuracy scores and metrics for four different classifiers with random_state = 1582 and test_size = 0.25

Model	Class	Precision	Recall	F1-score
kNN Accuracy: 88.89%	ALL	0.86	1.00	0.92
	AML	1.00	0.67	0.80
Logistic Regression Accuracy: 94.44%	ALL	0.92	1.00	0.96
	AML	1.00	0.83	0.91
SVC Accuracy: 94.44%	ALL	1.00	0.92	0.96
	AML	0.86	1.00	0.92
Random Forest Accuracy: 100%	ALL	1.00	1.00	1.00
	AML	1.00	1.00	1.00

Overall, all four of the tested models performed very well on both of the train/tests splits, with only two misclassified samples at most, suggesting that the number of features (i.e. genes) used to train the models can likely be reduced. Indeed, even after reducing the number of genes (i.e. features) to only the top two and using a 20/80 train/test split, kNN and logistic regression classifiers still performed with 87.93% and 79.31% accuracy, respectively; the kNN classifier only misclassified 7 out of 58 test samples. From this I can conclude that the class distinction between ALL and AML cancer is quite strong, which could be very useful in a diagnostic setting. Instead of performing microarray analysis on a patient's tumor sample to diagnose the type of cancer, which can be very costly, a much smaller, targeted screen can be performed very quickly and at a lower cost without decreasing the accuracy of diagnosis.

iii. Feature Reduction Using PCA

In my previous models, I used the calculated correlation estimates as a measure for feature reduction, selecting only the genes with high correlation values as features for training the classifiers. However, feature reduction can also be performed using principal component analysis (PCA). PCA projects a high-dimensional dataset (i.e. one with many features) to a lower dimensional space by decomposing the dataset into orthogonal components (the principal components) that explain a maximum amount of the variance. I first used PCA on the reduced set of 50 genes and plotted the PCA components vs. the explained variance to examine the intrinsic dimension of this dataset. The first 3 principal components explain over 80% of the variance, and optimization of the PCA transformation using grid search indicates that 9 is the optimal number of principal components (Figure 10).

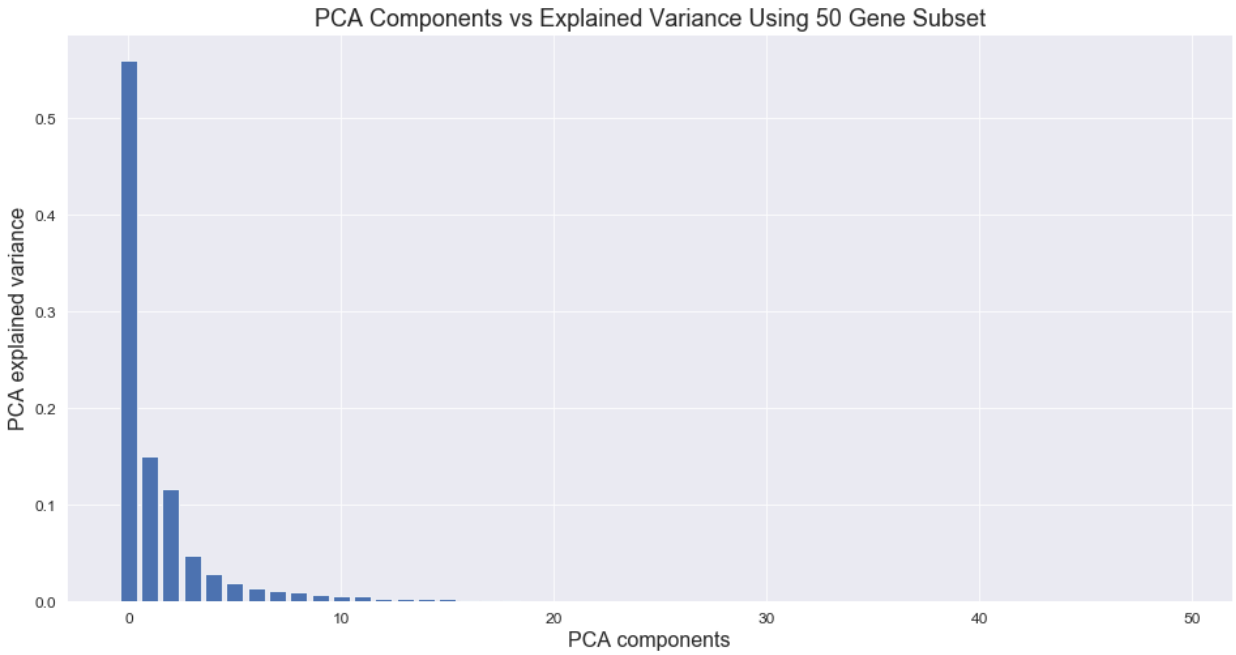

FIGURE 10: PCA components for the reduced feature set containing the 50 most correlated genes.

Next, I performed the same analysis on the full 7070 gene dataset to assess whether PCA alone could be used for dimension reduction without taking the correlation estimates into account. Overall, the individual principal components of the full dataset explain far less of the

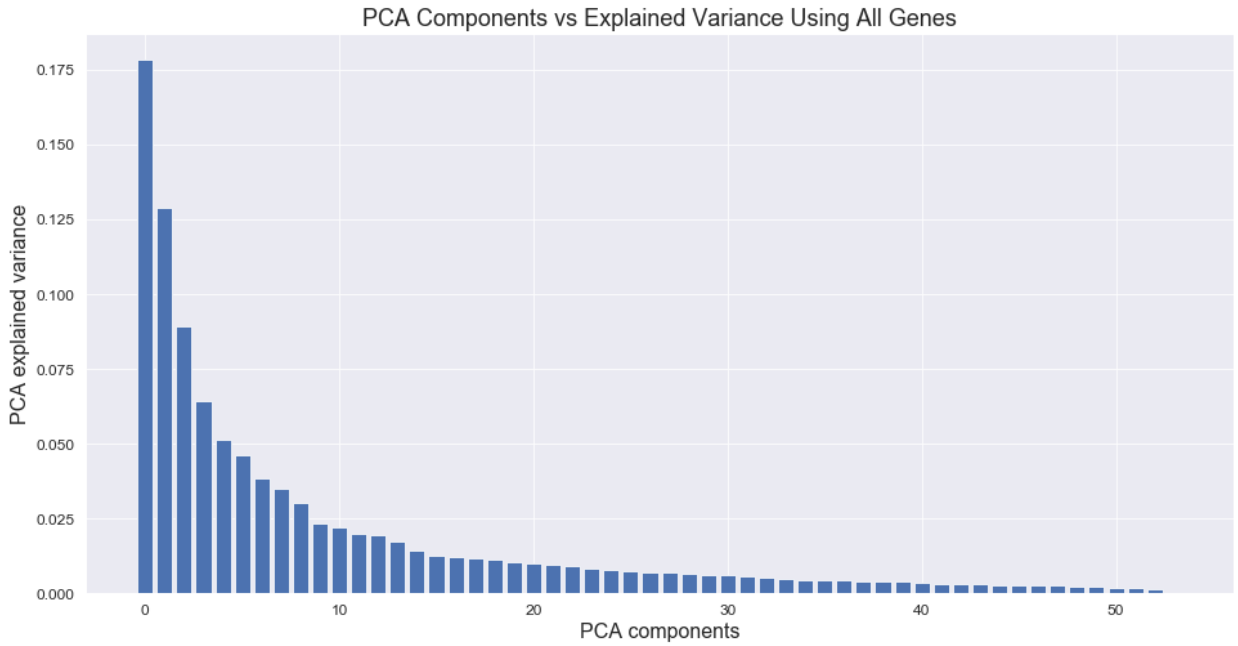

FIGURE 11: PCA components for the full feature set containing all 7070 genes.

variance than the components in the 50 gene set, and more components are needed to explain the same amount of variance (Figure 11).

Finally, I assessed the performance of PCA reduction followed by k-nearest neighbors classification on both the reduced 50 gene dataset and the full 7070 gene dataset. Based on the metrics, reducing the feature set using the correlation estimates produces a better model than simply using PCA reduction on the entire dataset; however, the accuracy improves by only a modest ~5.5% (Table 5). In addition, PCA reduction of the 50 gene dataset followed by kNN classification did not show a performance improvement over kNN classification with PCA reduction. On the other hand, performing PCA reduction on the full dataset prior to fitting the data with a machine learning algorithm is certainly much simpler and faster than calculating the correlation estimates and extracting out a subset of the gene data before fitting the data, and the results are still relatively accurate. It is also possible that PCA reduction of the full dataset would perform better if trained with a larger number of samples; the entire dataset is comprised of only 72 samples, which limits the maximum number of components that can be identified using PCA. As the number of features far exceeds the number of samples, PCA cannot easily extract the principal components from the noise.

TABLE 5: Metrics from PCA reduction & kNN classification of the 50 gene dataset and the full 7070 gene dataset.

Dataset	Parameters	Class	Precision	Recall	F1-Score
50 gene subset Accuracy: 88.89%	n_components: 1	ALL	0.86	1.00	0.92
	n_neighbors: 2	AML	1.00	0.67	0.80
All 7070 genes Accuracy: 83.33%	n_components: 18	ALL	0.80	1.00	0.89
	n_neighbors: 5	AML	1.00	0.50	0.67

V. Conclusion

The original goal of this analysis was to develop a machine learning model capable of distinguishing between the two common forms of leukemia, acute lymphoblastic leukemia (ALL) and acute myeloid leukemia (AML). The results of my analysis suggest that the application of machine learning techniques in the diagnosis of cancers holds a lot of promise, as even this relatively small dataset was sufficient to develop highly accurate models to aid in cancer classification. Analysis of the microarray data allowed for the identification of a subset of genes that are highly correlated with the ALL/AML distinction, suggesting that the diagnostic tests can be simplified without decreasing diagnostic accuracy. In the specific case of diagnosing acute leukemias, the models developed in this report could be used to quickly analyze microarray data to classify a tumor as either ALL or AML. Not only would fewer tests be required to obtain a confident diagnosis, but the tests currently used to generate the microarray datasets could also be simplified to target the gene expression data from only a small subset of informative genes rather than collecting data from over 7000 genes. Taken together, implementation of machine learning algorithms in leukemia diagnosis would decrease the cost, the number of tests, and the time required to accurately diagnose acute leukemias.