# Leukemia Gene Detection

Shubham Joshi

*December 6, 2021*

**Abstract:** Leukemia is a blood-forming tissue cancer that affects the bone marrow and lymphatic system. In this project we will look into Acute Myeloid Leukemia (AML). We'll use supervised machine learning algorithms like KNN, Decision Trees, SVM, and XGBOOST to distinguish AML from hematopoietic cells. Support Vector Machine is a model that can correctly distinguish AML from hematopoietic cells (SVM). The model's accuracy is 92.3 percent. SVM Classifier should be used to categorise AML based on the patient's gene data.

## 1 Introduction

Leukemia is a malignancy of the blood cells as a whole. The kind of leukaemia is determined by the type of cancerous blood cell and how rapidly or slowly it grows. The most prevalent cancer in individuals over 55 is leukaemia, but it is also the most common disease in children under the age of 15.

The kind of leukemia is determined by the type of cancerous blood cell and its rate of growth. The types of leukemia we will be looking at are Acute Myeloid Leukemia (AML) and try to classify it from Normal hematopoietic cells of Bone Marrow, Bone Marrow CD 34, Peripheral Blood (PB), and Peripheral Blood Stem Cells CD34+ cell selection (PBSC CD34) from GSE9476 gene study [2]. CuMiDa is a dataset used for the project, curated from 78 cancer microarray datasets that have been hand-picked [1]. We will examine and classify leukemia AML from other normal hematopoietic cells in this project.

## 2 Case Study

In this project, we will be using a CSV file of CuMiDa dataset for the analysis, which will be loaded into pandas DataFrame via read_csv method. Feature Engineering will be applied to the data, for analyzing any anomalies of missing data and corruption in data. Analyzing different shapes and features of the dataset.

Once the feature engineering is completed, we will select features (genes) that are related to leukemia type classification and help our model in classification based on the Pearson Correlation method. The scaling of the input features will be done, via the Standard

Scalar of sklearn library. Scaling is done so that model generalizes for all the input feature range. Supervised learning models used for the study are K-Nearest Neighbors (KNN), Support Vector Machine Classifier (SVM), Decision Tree Classifier (DT), and gradient boosting model i.e. xgboost. The model will learn from the labels from the existing training dataset and be evaluated on the test dataset. Cross-Validation techniques will be used to know how our model will perform on the unseen data.

KNN and SVM models use the distance-based matrix to classify the dataset for any new query of point. Whereas, decision trees use tree-based classification methods and xgboost uses gradient boosting techniques. Models will be compared based on their accuracy of the prediction on the test dataset, confusion matrix, and F1-score. The model with the best accuracy and the F1 score could be used in the future for classifying the Leukemia gene. The null hypothesis undertaken for the study is all the datasets for each label are evenly distributed. In general, the imbalanced dataset on multi-class classification causes the model to give poor prediction results of the F1-score for that particular class.

# 3 Data Analysis

To begin my analysis on the CuMiDa dataset, I loaded the CSV data in pandas DataFrame (df) and checked for the shape of the dataset. The dataset has 64 rows and 16384 columns. On printing, the head of the df we were able to see the columns like sample, type, "1007-s_at". The sample is the id of the observation whereas the "type" is the classification label of the cell. "1007-s_at" is the gene in the cell that is captured in our row. We will be dropping the sample from our df as they don't affect our label. In our dataset, we have a total of 16383 features that are trying to affect the output label "type". We also have 5 unique labels in type ('AML', 'Bone_Marrow', 'Bone_Marrow_CD34', 'PB', 'PBSC_CD34'). It's a "big-p, little-n" dataset with a lot of features but limited data for each of the classification labels. By checking the dtypes for the DataFrame, we know 16382 features are of float64 except the label "type" which is an object.

On doing the null analysis on the df using isna() method, we observed the dataset does not have any null values and it is clean. Label encoding is applied to the type, resulting in numerical classes that our machine learning algorithms can understand instead of text. The classes label encoding is listed as " {'AML': 0, 'Bone_Marrow': 1, 'Bone_Marrow_CD34': 2, 'PB': 3, 'PBSC_CD34': 4 }". We see that the std deviation of features ranges in a wide range when we use describe() on the df, thus we need to scale the data.

## 3.1 Correlation

Correlation is one of the statistical techniques, used to find the relation between the variables. It is being used as technique for Feature Selection in our project analysis. The Pearson Correlation between the output label "type" and all the features (genes) of the dataset is found using corr() method on the df. Threshold of 67% is applied on the correlation, so that we find the features which are highly correlated to the label. The genes which were which related to the label "type" are '201065_s_at', '201313_at', '202178_at', '202227_s_at', '203194_s_at', '203195_s_at', '204269_at', '204749_at', '204754_at', '205159_at', '205413_at', '205467_at', '205607_s_at', '207643_s_at', '207992_s_at', '208773_s_at', '209252_at', '209586_s_at', '209930_s_at',

'210891_s_at', '212148_at', '212151_at', '212534_at', '212549_at', '213241_at', '213261_at', '213506_at'

On plotting these genes on the heatmap with seaborn library Figure1. We could see the genes 212148_at and 204749_at are less correlated to the output label. We will be dropping these genes in our further study. Finally, we have a list of 25 genes that are related to the "type" label, used in our analysis as below. Figure2 "'201065_s_at', '201313_at', '202178_at', '202227_s_at', '203194_s_at', '203195_s_at', '204269_at', '204754_at', '205159_at', '205413_at', '205467_at', '205607_s_at', '207643_s_at', '207992_s_at', '208773_s_at', '209252_at', '209586_s_at', '209930_s_at', '210891_s_at', '212151_at', '212534_at', '212549_at', '213241_at', '213261_at','213506_at' "

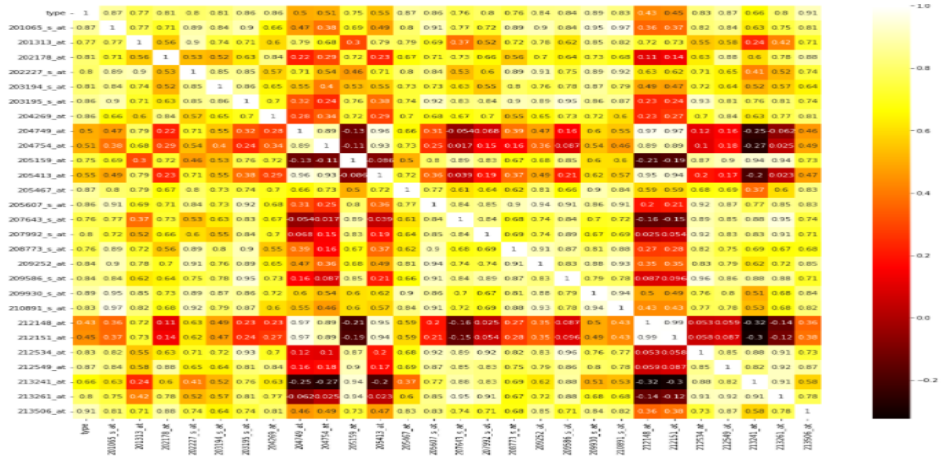

Figure 1: "Correlation in Genes (27)"

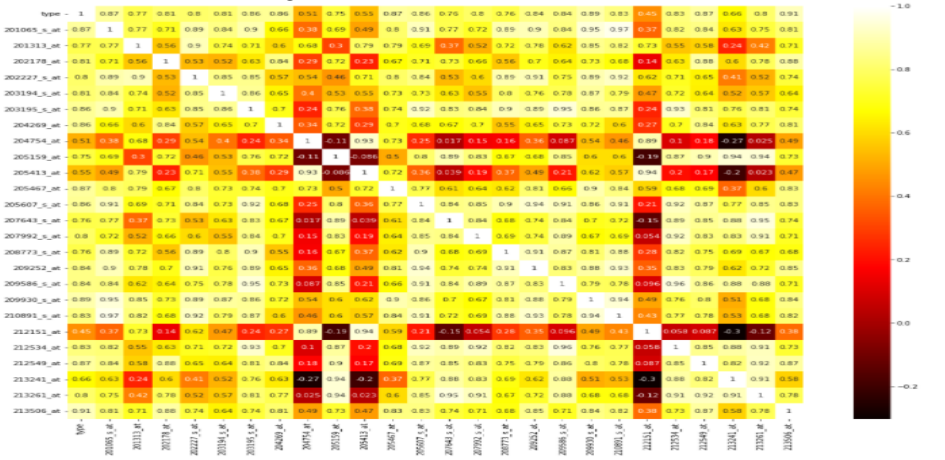

Figure 2: "Correlation in Genes (25) "

## 3.2 Scaling

Standardization of the data is required so that our dataset falls in the range of mean 0 and standard deviation 1. This is done for the model generalization and to improve the performance of the model on the test or unseen data. We will be using StandardScalar from the sklearn library for scaling our data. Firstly, an instance of StandardScalar is created. Once it is done, we can use fit_transform() function on our dataset to transform them Figure 3

| | 201065_s_at | 201313_at | 202178_at | 202227_s_at | 203194_s_at | 203195_s_at | 204269_at | 204754_at | 205159_at | 205413_at | ... | 209252_at | 209586_s_at | 209930_s_at | 210891_s_at | 212151_at | 212534_at | 212549_at | 213241_at | 213261_at | 213506_at |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.159235 | -0.635047 | -0.378241 | 0.027417 | 0.763035 | -0.228541 | 1.828490 | -0.468203 | 0.213512 | 0.580328 | ... | -0.638146 | -0.461822 | -0.457539 | 0.075045 | 1.366905 | -0.843746 | -0.095182 | 0.043539 | -0.131115 | -0.186527 |
| 1 | -1.046527 | 0.885957 | -0.207285 | 0.742327 | 0.808672 | -0.158455 | 0.093184 | 0.445545 | -0.119596 | 0.549387 | ... | 0.104401 | 0.458748 | 0.383237 | -0.476319 | 0.982238 | -0.368011 | -0.225432 | -0.246991 | -0.088749 | 0.571237 |
| 2 | 0.123155 | 1.076344 | -0.478255 | 1.083989 | 1.358835 | 0.927087 | -0.075388 | 0.711761 | -0.253898 | 0.480481 | ... | 0.413358 | 0.980138 | 0.740300 | -0.002950 | 0.953976 | 0.618730 | 0.049236 | -0.052022 | -0.020139 | 0.838996 |
| 3 | 0.155661 | 0.132925 | -0.132760 | 0.945758 | 0.829331 | 0.406224 | 0.145854 | -0.232630 | 0.302784 | 0.356769 | ... | -0.157134 | 0.739959 | 0.817793 | -0.088229 | 1.047327 | 0.286027 | 0.863876 | 0.354785 | -0.260059 | 0.827600 |
| 4 | 0.387177 | 0.385412 | 0.027408 | 0.530850 | 0.294336 | 0.213458 | 0.719803 | 0.906222 | 0.119448 | 0.624159 | ... | -0.349727 | -0.163636 | -0.336206 | 0.225855 | 0.821397 | 0.225638 | 0.132521 | 0.169751 | -0.595549 | -0.051035 |

5 rows × 25 columns

Figure 3: StandardScalar

## 3.3 Train Test Split

Model performance could be determined only after testing it in unseen data, to determine the model is not overfitting. We will split our dataset into the training and test data. The label "type" from the df is separated as y, whereas all the features are categorized as X in DataFrame. Sklearn library train_test_split() is used to split our data in X_train, X_test, y_train, and y_test with the test_size of 0.2. The dataset in our project is imbalanced as we could observe from Figure 4. All the classes do not have the same amount of data. This rejects our null hypothesis that the datasets are distributed evenly for each label and the alternate hypothesis is true of having an imbalanced dataset. We will be using stratified sampling on the output label y while splitting the data to have data distributed evenly after the split by using the stratify=y at the time of train_test_split

Figure 5 demonstrates the data split in training and test dataset with stratified sampling. Class 1 has increased number of test data after stratified sampling. Class 3 data was reduced from 0.23 to 0.15 after we applied stratified sampling. Class 2 has as little data for training the model. The overall data for Class 2 is 0.12 which is very low. Hence, even after stratified sampling, we are having a very low volume of data for training.

```
PORTION OF TYPE IN ORIGINAL DATA
0    0.40625
1    0.15625
3    0.15625
4    0.15625
2    0.12500
Name: type, dtype: float64
```

```
PORTION OF TYPE IN TRAIN DATA
0    0.411765
1    0.176471
4    0.156863
3    0.137255
2    0.117647
Name: type, dtype: float64
```

```
PORTION OF TYPE IN TEST DATA
0    0.384615
3    0.230769
4    0.153846
2    0.153846
1    0.076923
Name: type, dtype: float64
```

Figure 4: Train Test Split

```
PORTION OF TYPE IN ORIGINAL DATA
0    0.40625
1    0.15625
3    0.15625
4    0.15625
2    0.12500
Name: type, dtype: float64
```

```
PORTION OF TYPE IN TRAIN DATA
0    0.411765
1    0.156863
3    0.156863
4    0.156863
2    0.117647
Name: type, dtype: float64
```

```
PORTION OF TYPE IN TEST DATA
0    0.384615
4    0.153846
3    0.153846
1    0.153846
2    0.153846
Name: type, dtype: float64
```

Figure 5: Stratified Splitting

# 4 Models

There are 5 classes in our dataset. It is a multi-class classification problem. We will be using a Supervised Machine Learning algorithm for classifying these classes. The supervised learning model will learn from the labels in the dataset, which is y for us. The models that could be used are based on the distance matrix, to classify the class based on the proximity of the nearby classes. The decision tree model also helps classify the classes in a multi-class dataset.

Models selected for our analysis are Decision Tree Classifier, K-Nearest Neighbor, SVM Classifier, XGBOOST Classifier.

## 4.1 Decision Tree Classifier

It is a supervised machine learning algorithm where the data is split continuously based on the features with the largest information gain (IG). Splitting is done on each child node until we have a pure leaf i.e. sample of each leaf node belonging to the same class.

We will be using this model for our Leukemia gene classification as results are better as the features are split into the trees like the graph for decision making and it takes less time for computing. The hyperparameter used are 'max_depth':[2,4,6,8,10,12], 'min_samples_split': [2,3,4] and 'min_samples_leaf': [1,2] for pre-pruning. These params are fit into the GridSearchCV which gives us the best_estimator_ after tuning it. The model is then fitted with the hyperparameters from GridSearchCV. The prediction is made from the X_test from the dataset. Model accuracy achieved is 77%.

The Classification Report with F1-score for all the classes and confusion matrix is plotted for the decision tree classifier model Figure

Accuracy Score: 76.92307692307693
Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.71 | 1.00 | 0.83 | 5 |
| 1 | 0.00 | 0.00 | 0.00 | 2 |
| 2 | 0.50 | 0.50 | 0.50 | 2 |
| 3 | 1.00 | 1.00 | 1.00 | 2 |
| 4 | 1.00 | 1.00 | 1.00 | 2 |
| accuracy | | | 0.77 | 13 |
| macro avg | 0.64 | 0.70 | 0.67 | 13 |
| weighted avg | 0.66 | 0.77 | 0.71 | 13 |

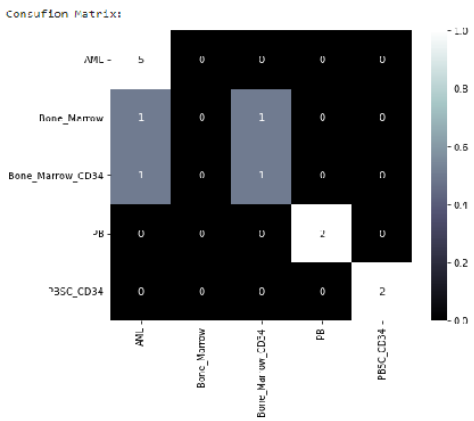


Figure 6: Decision Tree Classifier

## 4.2 K-Nearest Neighbour (KNN)

It is a supervised machine learning algorithm based on the distance matrix between the point for the classification. The class of a query point is determined by the majority vote of the classes from the k nearest point. Euclidean distance is used to measure the distance between the two points.

KNN is used for project analysis on Leukemia gene detection as it uses the similarity measures of the nearby classes (genes) to classify the new cell for us. The n_neighbors parameters from the range 1 to 25 are used with the GridSerchCV to find best the k for our classification. KFold cross-validation with shuffles True and random_state 11 is also used for our model generalization [3]. The best_params_ from the GridSerchCV is 1 with the best_score_ of 92%. The KNN Classifier is fitted with the hyperparameters obtained from the GridSerchCV. The cross-validation mean of KNN is calculated as 92.17% by taking the mean of the scores. Model accuracy achieved is 84.61%.

The Classification Report with F1-score for all the classes and confusion matrix is plotted for the KNN classifier model Figure 7. F1 score of AML, Bone_Marrow, Bone_Marrow_CD34, PB, PBSC_CD34 is 0.89, 0.67, 0.67, 1.0 and 1.0 respectively. We could observe that the KNN model has performed better than the Decision Tree Classifier as it was able to predict the class Bone_Marrow with a 0.67 f1-score. Prediction for PB has also increased from 0.50 to 0.67 in KNN. We could use this model for our analysis in the future.

Accuracy Score: 84.61538461538461
Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.80 | 0.89 | 5 |
| 1 | 0.50 | 1.00 | 0.67 | 2 |
| 2 | 1.00 | 0.50 | 0.67 | 2 |
| 3 | 1.00 | 1.00 | 1.00 | 2 |
| 4 | 1.00 | 1.00 | 1.00 | 2 |
| accuracy | | | 0.85 | 13 |
| macro avg | 0.90 | 0.86 | 0.84 | 13 |
| weighted avg | 0.92 | 0.85 | 0.85 | 13 |

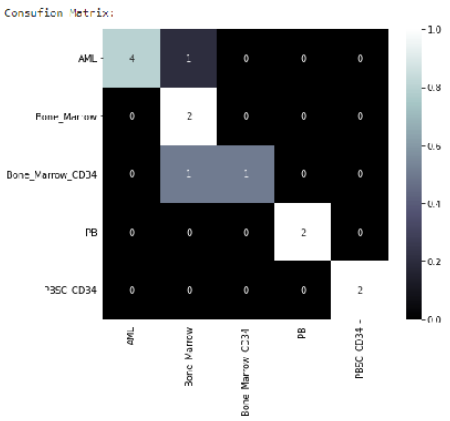


Figure 7: K-Nearest Neighbours

## 4.3 SVM Classifier

It is a supervised machine learning algorithm to find the hyperplane of maximum margin that distinctly classifies the classes and all the data points in it. These are also called decision boundaries.

Support Vector Machine Classifier is used for project analysis, as it will help in creating the decision boundaries between the data points, so we can classify the genes in their respective classes. SVM implementation from the Sklearn library is imported. The hyperparameters used are Radial Basis Function (RBF) kernel, penalty parameter C =1, and gamma as 0 (default) to obtain the accuracy of 92.3%. If we changed the kernel to linear and the gamma is increased further the accuracy of the model was decreasing. The final hyperparameters were kernel='rbf' and C=1.

The Classification Report with F1-score for all the classes and confusion matrix is plotted for the SVM classifier model Figure 8 F1 score of AML, Bone_Marrow, Bone_Marrow_CD34, PB, PBSC_CD34 is 0.91, 1.0, .67, 1.0 and 1.0 respectively. We could observe that the SVM Classifier model has performed better than the KNN. The f1-score for AML class is increased from 0.89 to 0.91 and we also have a significant increase in the f1-score of Bone_Marrow to 1.0 from 0.67 in KNN which means it can identify all the cells for Bone_Marrow. We should use SVM Classifier model for Leukemia Gene detection as the model has performed better than others.

Accuracy Score: 92.3076923076923
Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.83 | 1.00 | 0.91 | 5 |
| 1 | 1.00 | 1.00 | 1.00 | 2 |
| 2 | 1.00 | 0.50 | 0.67 | 2 |
| 3 | 1.00 | 1.00 | 1.00 | 2 |
| 4 | 1.00 | 1.00 | 1.00 | 2 |
| accuracy | | | 0.92 | 13 |
| macro avg | 0.97 | 0.90 | 0.92 | 13 |
| weighted avg | 0.94 | 0.92 | 0.91 | 13 |

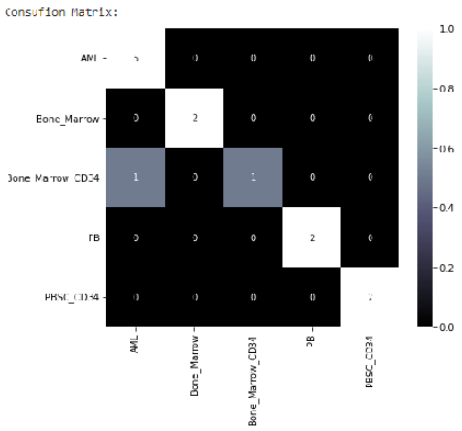


Figure 8: SVM Classifier

## 4.4 XGBOOST Classifier

It is a supervised machine learning algorithm that uses the ensemble technique of the gradient boosted decision trees for better performance and speed of the tree-based model. It is a sequential ensemble technique where the new model is trained to predict the errors of the previous model.

XGBOOST Classifier is used for project analysis, as an improvement of the Decision Tree Classifier model for tree-based learning. We need to install the xgboost library from the pip command. XGBClassifier is imported from xgboost.sklearn. The hyperparameters used to train the XGBClassifier are num_class=5, learning_rate=0.1, num_iterations=1000, max_depth=10, feature_fraction=0.7, scale_pos_weight=1.5, boosting='gbdt', metric='multiclass', eval_metric='mlogloss'. The learning rate specifies at what rate we want our model to learn and it is multiclass classification with a total no of classes as 5. The evaluation metric is logloss by which the decision tree will be learning. The accuracy score achieved with xgboost classifier is 92%

The Classification Report with F1-score for all the classes and confusion matrix is plotted for the SVM classifier model Figure 9. F1 score of AML, Bone_Marrow, Bone_Marrow_CD34,PB, PBSC_CD34 is 0.91, 1.0, .67, 1.0 and 1.0 respectively. We could observe that the XGBOOST Classifier model has performed similarly to the SVM Classifier model. The f1-score for AML class is 0.91 and the f1-score of Bone_Marrow is also 1.0. We could use any one of the models of SVM Classifier or XGBOOST Classifier for our project.

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.83 | 1.00 | 0.91 | 5 |
| 1 | 1.00 | 1.00 | 1.00 | 2 |
| 2 | 1.00 | 0.50 | 0.67 | 2 |
| 3 | 1.00 | 1.00 | 1.00 | 2 |
| 4 | 1.00 | 1.00 | 1.00 | 2 |
| accuracy | | | 0.92 | 13 |
| macro avg | 0.97 | 0.90 | 0.92 | 13 |
| weighted avg | 0.94 | 0.92 | 0.91 | 13 |

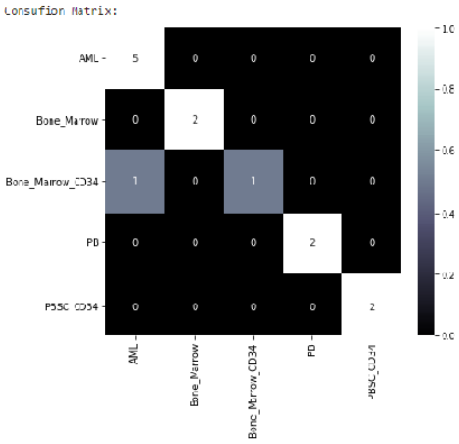


Figure 9: XGBoost Classifier

# 5 Results

The model accuracy and f1-score of all 4 models in our gene analysis are shown in Table 1. The decision tree classifier model performed the worst with 77% accuracy. Accuracy was further improved in KNN with hyperparameters to 84%. The models which performed the best are SVM Classifier and XGBoost Classifier with 92.3% and 92% accuracy respectively. The decision tree was not able to predict Bone Marrow class with an f1-score of 0, we should not use this model. AML is correctly classified by the SVM Classifier and XGBoost Classifier with a 91% f1-score in both of them. Bone Marrow class f1-score improved from 0 to 67% in the SVM Classifier and XGBoost Classifier models. The Bone Marrow CD 34 has the lowest F1 Score of 67%, as seen in the table. In Peripheral Blood and Peripheral Blood Stem Cells CD34+, we could achieve 100% classification in KNN, SVM, and XGBoost classifier models.

For the CuMiDa dataset of Leukemia, we may use the SVM Classifier to classify AML from the normal hematopoietic stem cells of BoneMarrow, Bone Marrow CD34, Peripheral Blood (PB), and Peripheral Blood Stem Cells CD34+.

| Models | Accuracy (%) | F1 Score AML(%) | F1 Score Bone Marrow(%) | F1 Score Bone Marrow CD34 (%) | F1 Score PB (%) | F1 Score PBSC CD34(%) |
|---|---|---|---|---|---|---|
| Decision Tree | 69 | 83 | 0 | 50 | 67 | 80 |
| KNN | 84 | 89 | 67 | 67 | 100 | 100 |
| SVM Classifier | 92.30 | 91 | 100 | 67 | 100 | 100 |
| XGBOOST Classifier | 92 | 91 | 100 | 67 | 100 | 100 |

Table 1: Models

# 6 Conclusion and Future Scope

Our analysis on the Leukemia Gene Detection shows SVM Classifier as the best model to classify the AML cancer cell from other hematopoietic cells i.e. Marrow, Bone Marrow CD 34, Peripheral Blood (PB), and Peripheral Blood Stem Cells CD34+ with an accuracy of 92.3%. Hence, we could use this model for the classification of these cells.

On observing Figure 8 Bone Marrow class has the maximum recall of 50%, indicating that the model is ineffective on the new dataset; however, this might be avoided by training the model with more data. We can see that the training data contains just 11.7 percent Bone Marrow, therefore the dataset is imbalanced. We could use SMOTE technique to increase the minority class and then used it for the SVM classifier using a pipeline.

In the Decision Tree Classifier model, we are just using hyperparameters with a pre-pruning technique. Our model could be overfitting with the dataset. We could use the post pruning technique to generalize our model and get a better result by fitting our X-train and y_train in cost_complexity_pruning_path()

Another thing to note is that we used a Pearson Correlation approach with a 67 percent criterion to choose only 25 features out of a total of 16k+ features. We need to see the features are linearly related to the type or non-linear data by plotting them separately.

# References


[1] FELTES, B. C., CHANDELIER, E. B., GRISCI, B. I., AND DORN, M. Cumida: an extensively curated microarray database for benchmarking and testing of machine learning approaches in cancer research. *Journal of Computational Biology 26*, 4 (2019), 376–386.

[2] ISLAM, Z., HORIKAWA, A., INUI, T., AND ISHIBASHI, O. -dependent interleukin-4-responsive genes in the mouse macrophage cell line RAW264. *Data Brief 23* (Apr 2019), 103669.

[3] JAIN, D. Cross-validation using knn, Dec 2020.