Blood based prediction of immune mediated disorders from bulk RNA-seq data

by

Shivanshu Kukrety

Under the supervision of Dr Debarka Sengupta

Indraprastha Institute of Information Technology Delhi

June, 2022

Blood based prediction of immune mediated disorders from bulk RNA-seq data

by
Shivanshu Kukrety

Submitted
in partial fulfillment of the requirements for the degree of
Master of Technology

to
Indraprastha Institute of Information Technology Delhi
June, 2022

Certificate

This is to certify that the thesis titled "Blood based prediction of immune mediated disorders from bulk RNA-seq data" being submitted by Shivanshu Kukrety to the Indraprastha Institute of Information Technology Delhi, for the award of the Master of Technology, is an original research work carried out by him under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this project have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

June, 2022

Dr Debarka Sengupta
Department of Computational Biology
Indraprastha Institute of Information Technology Delhi
New Delhi, 110020

Acknowledgement

I would like to express my deepest gratitude to my guide and supervisor Dr Debarka Sengupta for guiding me throughout my thesis journey. I would like to thank him for sharing his invaluable knowledge and experience, mentoring me at every stage without which I would not be able to get such fruitful results. I also give my thanks to Department of Computational Biology, faculty and staff of IIITD for making my MTech journey so enriching and enjoyable. I am grateful to National Bioscience Database Center for providing the dataset which was used throughout the project. I specially thank Jeremie Poschmann for providing his inputs, feedback and acting as a co-guide for me.

I want to thank my friends and college mates for walking alongside me on this journey. Eminently I thank my family as none of this would have been possible without their support.

Abstract

Immune-mediated disorders (IMDs) include a wide spectrum of pathologies ranging from autoimmunity to autoinflammation, and they impact a substantial number of individuals worldwide. Although dysfunctional inflammatory cytokine behaviour in IMDs implies abnormal immune cellular activity, not much is known about the underlying responsible genes and sometimes even crucial cell types. Moreover the proportions of gene expression variance explained by the clinical diagnosis is quite small, which makes it difficult to analyze the underlying condition. Recent breakthroughs in artificial intelligence have resulted in widespread industrial and academic use, with machine learning systems outperforming traditional schemes in a wide array of applications. Our project aims to make use of this predictive power to build classification models using gene expression data for prediction of immune mediated diseases. The various models built are tested using nested cross validation on a wide variety of metrics to analyze the generalizability of our classifiers. The best result was achieved by support vector machines with an accuracy of 92.29% and a MCC value of 91.59%. In our project we have also carried out a differential expression analysis in order to obtain a comparison of gene expression patterns between a healthy individual and a patient infected with an immune mediated disease, which enable us to identify genes which may be participating in specific functions such as protein synthesis, hormone delivery and pathological pathways. In addition to the ones stated we also present a deconvolution operation performed on cibersort to obtain the relative cell proportions of 28 immune cell types in a specific disease class. The codebase is available on github and can be accessed using the following link: https://github.com/tom8861/thesis.

Table of Contents

List of figures

List of tables

Chapter 1: Introduction

The immune system is a network of biological mechanisms that guards an organism against illness. It recognizes and responds to a wide range of pathogens, including viruses and parasitic worms, as well as cancer cells, separating them from the organism's own healthy tissue. The immune system is divided into two primary subsystems in many animals. The innate immune system has a pre-programmed reaction to a wide range of events and stimuli. The adaptive immune system responds to each stimulus by learning to distinguish substances that it has previously encountered.

Immune-mediated disorders are a category of ailments characterized by abnormal immune cell activity, such as overreacting or attacking the body's own cells, showing an exaggerated inflammatory response, or losing the capacity to detect and destroy tumour cells. Despite various investigations on IMDs, little is known about the genes and pathological mechanisms that underpin the disease situations. Moreover the proportions of gene expression variance explained by the clinical diagnosis is quite small, which makes it difficult to analyze the underlying condition.

In our project we aim to analyze the gene expression profiles using differential expression analysis to unfold the genes which may be taking part in protein synthesis and/or a pathway leading to an immune mediated disease condition. Our analysis also focuses on finding immune cell type proportions for a disease class to capture the relevant cell types underlying immune mediated diseases. Further we have worked in the direction to try to improve the accuracy of clinical diagnosis by pairing it with trained models using machine and deep learning techniques.

Chapter 2: Background and Related Work

The primary paper used for this project performed an expression quantitative trait loci (eQTL) analysis revealing dynamic fluctuations in the context of immunological circumstances and context-dependent eQTLs were shown to be significantly enriched in immunological disease-associated genetic variations[1]. Literature survey shows many research investigations carried out on the dataset in various analysis domains as discussed below.

An article discussing clinical and immunological indicators that might be used to identify and track disease activity in System Lupus Erythematosus (SLE) patients with and without organ-specific damage[2]. Research areas focusing on the role of monocytes and macrophages in Bencet's disease[3], correlating contribution of oxidative phosphorylation to B-cell function and organ damage to SLE using multiomics analysis[4]. The domain of analysis is also expanded to finding precision medicines for immune-mediated diseases[5].

Chapter 3: Methodology

3.1 Data collection

Data collection is defined as the systematic acquisition, storage and analysis of data to gain insights into a problem domain. The process of collecting data is a crucial step as it ensures well-founded, credible and convincing data driven decisions. Data used in this project is compiled by National Bioscience Database Center (NBDC), and can be found at the link : https://humandbs.biosciencedbc.jp/en/hum0214-v3#E-GEAD-397.

To make data extraction and download more user-friendly , a python script is written, which retrieves the dataset and the necessary metadata files from the mentioned site using the wget library. The downloaded data is then extracted and renamed using the same python script.

3.2 Data Pre-Processing

Data pre-processing is one of the crucial step in data mining and analysis. The sole purpose of this step is to clean and restructure data in a way that it retains it's original meaning while also reducing the complexity and noise, which is part of any kind of data, the same is true for data which is generated and used in the medical domain.

"Garbage in, garbage out" is a popular phrase used in machine learning to signify, inputting poor quality data leads to poor quality results. Thus, for making effective decisions and gaining good results it becomes essential to supply the analysis pipelines quality data.

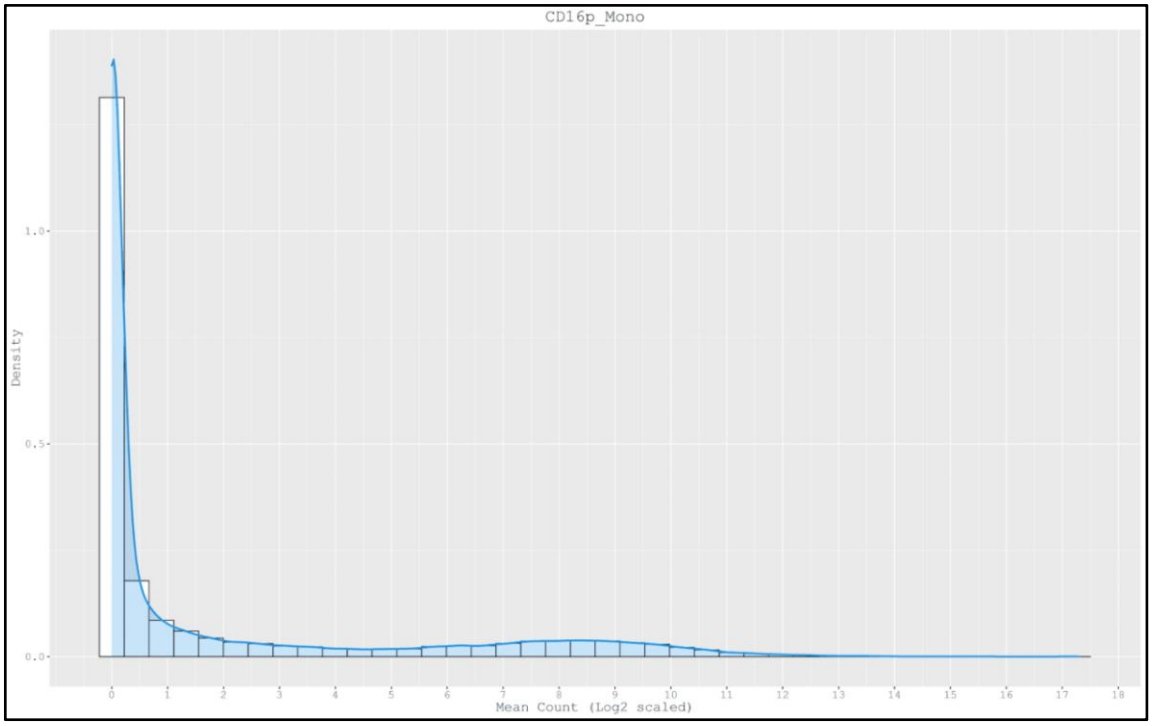
Typical steps involved in data preprocessing are:

1. Data cleaning: Owing to the fact that real world data is noisy, this becomes one of the most important step in the data preprocessing stage as it involves removing noisy data as well as filling out any missing values in the input dataset. Our dataset was cleaned by removing rows of genes which had majority of count data as zero. This was done to remove genes which were not being expressed in majority of patients.

2. Data transformation: This step is needed when feeding the data to a machine learning pipeline. The transformation is carried out by changing the values of numeric columns in the dataset to use a common scale thereby helping in faster training and convergence of algorithms to optimal values. In our case $\log_2$ transformation was applied on the data before training models.

3. Data reduction: Since data analysis needs to be carried out on large amounts of data it becomes necessary to reduce the dimension of data in order to increase the efficiency of pipelines as well restrict the investigation to a particular domain, scope or region. For the purpose of training and testing models, Principal Component Analysis was applied to the dataset. This helped in reducing time for training as well as led to output of better results. In addition to the one stated, the analysis was restricted to only protein coding genes.

Figure 1 Histogram of noisy data before preprocess step



As it can be seen in Figure 1, the dataset contains entries where the density of read counts is skewed towards genes which are not being expressed in a lot of patients. These genes act as noise in our data and hence are cleaned to balance the histogram-density plot as shown in Figure 3.

Figure 2 Steps taken to preprocess data before feeding to other pipelines

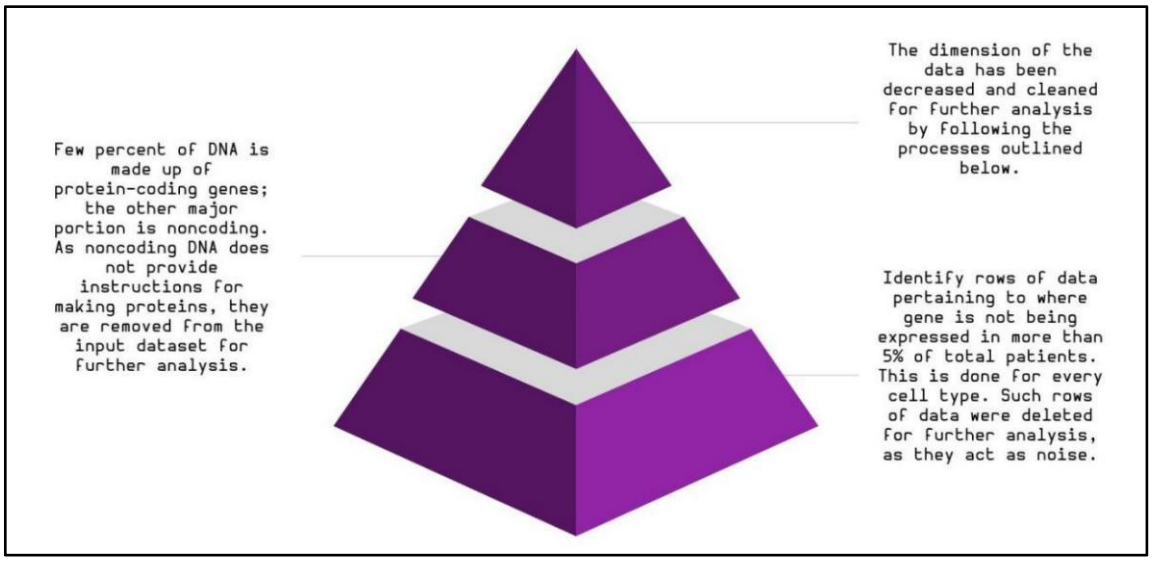
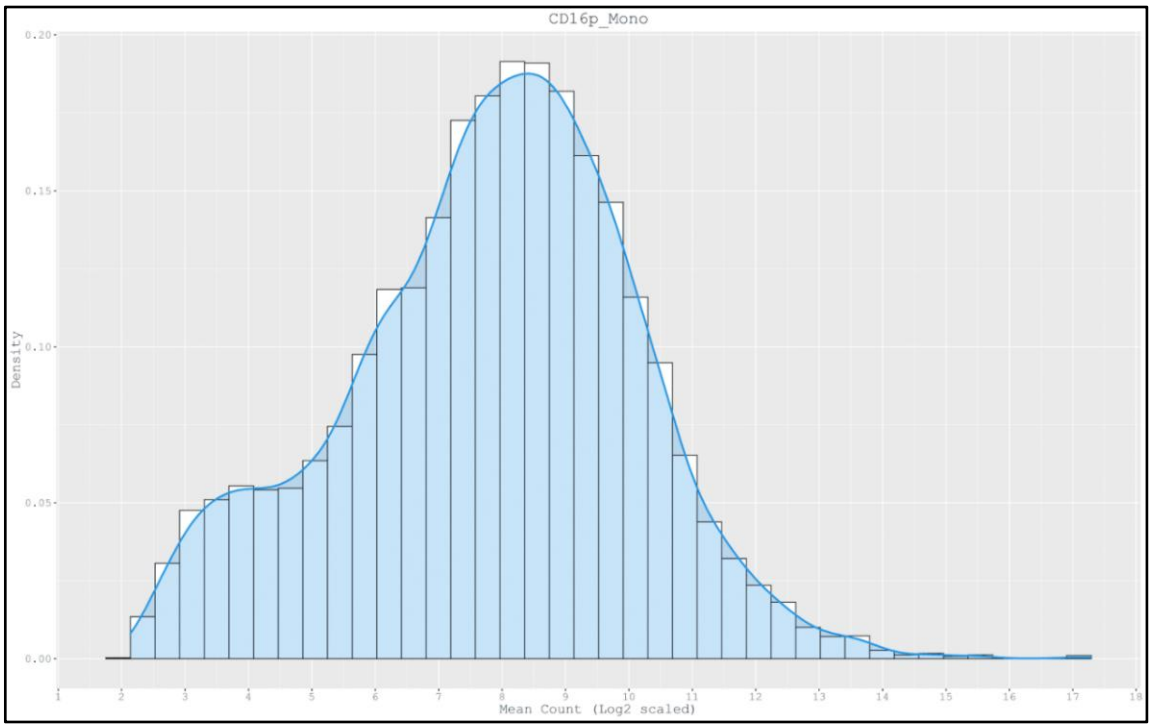


Figure 2 outlines the necessary steps which are carried out to preprocess the data and reorient it for performing better analysis and gaining refined results in future pipelines than what would have been possible without the preprocessing. Figure 3 represents the cleaner form of the noisy data where the low expressed genes are removed from further processing.

Figure 3 Histogram of clean data after preprocess step



3.3 Deconvolution

Deconvolution, the inverse of a convolution operation, is a technique which is used to restore an original image or signal after a convolution filter has been applied to it. Deconvolution is this context is used to find the percentage cell composition of a complex sample from it's gene expression profile. This process can be understood by the following equation.

$$M = f_1 \times Mem\_CD4 + f_2 \times Mem\_CD8 + f_3 \times Naive\_B + \ldots$$

where M denotes the mixture and f_i denotes the fraction of each cell type in the mixture.

CIBERSORTx is used in this scenario in order to deconvolute the given gene expression data fragmented across 28 cell types, shown in Figure 5-B, each containing varied combination of patient data for 11 disease classes as shown in Figure 5-A. The aim is to find the proportions of each cell type present for a particular case. The dataset for each disease class given to cibersort for deconvolution are prepared using steps as shown in Figure 6. Software used for the deconvolution can be found on the link: https://cibersortx.stanford.edu [6]. The deconvolution results are shown in Chapter 4: Results.

LM22 is the default signature matrix used for deconvolution and is already available on CIBERSORT containing 22 cell types. Since our dataset has 28 cell types, a custom signature matrix was prepared using the above mentioned tool by following steps mentioned in Figure 4.

Figure 4 Steps which are carried out to a make a reference matrix for creation of a custom signature matrix

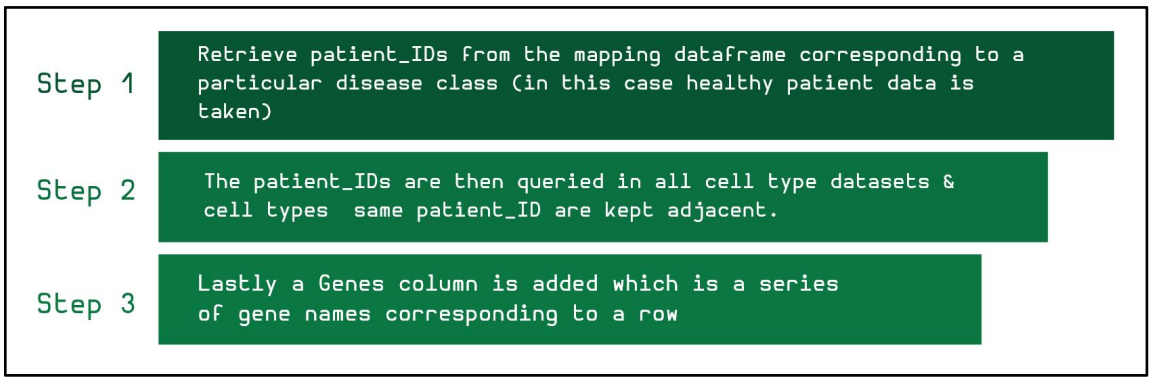
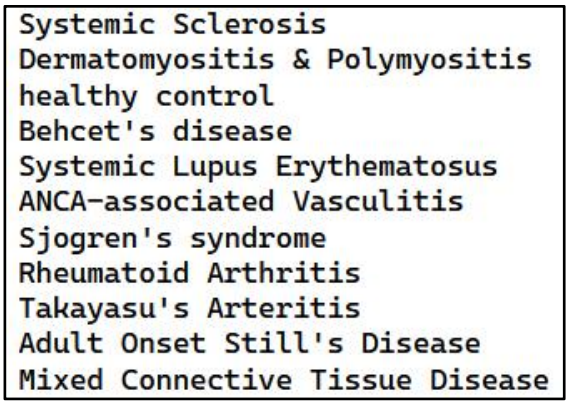
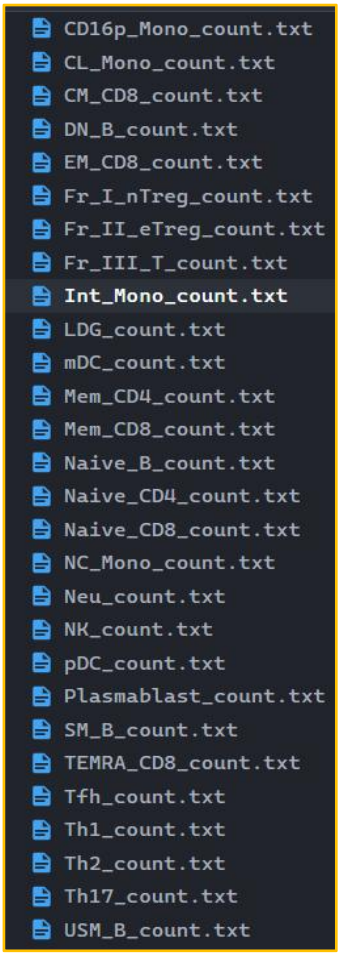


Figure 5 (A) Represents the 11 disease classes present in the dataset (B) Represents 28 cell types across which the read count data is present

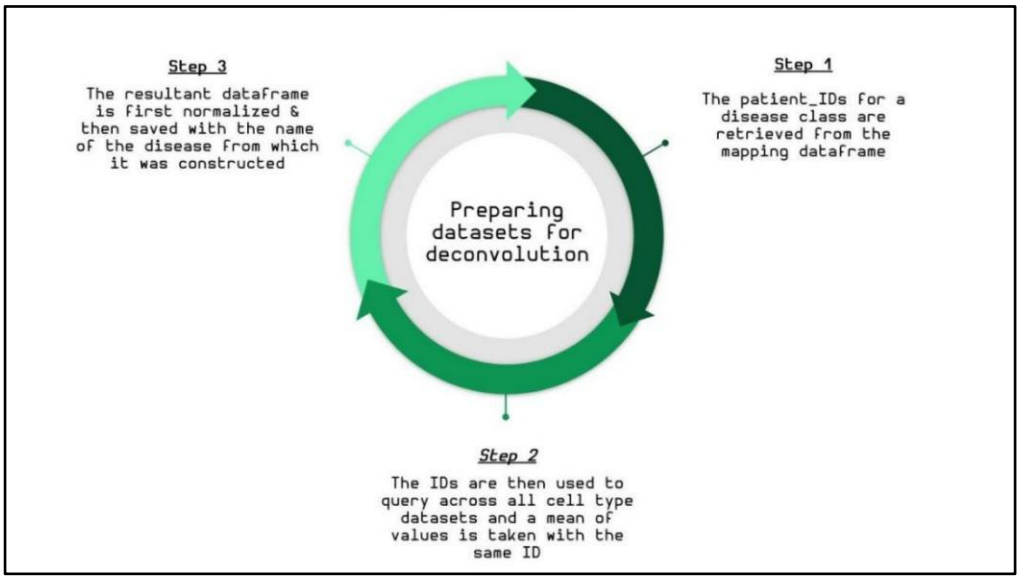


(A)



(B)

Figure 6 Steps taken to prepare datasets which were provided to CIBERSORT for deconvolution



3.4 Differential Expression Analysis

The objective of differential expression analysis is to find out which genes are expressed at various levels in different situations. These genes can provide biological insight into the processes that are influenced by the conditions of interest. This analysis is performed using DESeq2 library in R[7].

The number of sequence reads originating from a certain gene is represented by the count data, which is utilized for differential expression analysis. The greater the number of counts, the more reads linked with that gene, and the presumption that that gene was expressed at a higher level in the sample.

The dataset used for this project is scrambled across 28 cell types as shown in Figure 5-B, with each cell type data having patient ids as columns. The mapping for the patients to their respective disease type is provided in a mapping file. For the purpose of differential expression analysis the data is pooled to a single file and a comparative analysis is performed between healthy patients and a diseased state by following the below steps:

1. Every patient id in the mapping table is retrieved and queried across the cell datasets one by one. The ids which are present in the more than one cell type data are kept adjacent and a mean is taken for them.

2. The read count data & the mapping file is filtered to get patients of each disease class with the healthy patient data eg healthy vs systemic sclerosis, healthy vs mixed connective tissue disease etc.

3. A second mapping file, specifically made for this purpose, is provided to the algorithm to perform analysis between healthy patients and all other diseases under a single label (taken as infected in this case)

Upon passing the necessary data and meta file, the algorithm goes through the following phases for each iteration of analysis:

1. Estimation of size factors
2. Estimation of dispersion

3. Negative Binomial GLM fitting and Wald statistic.

The statistical results shown in Figure 7 performed on every combination is generated by the results function using the deseq2 object. The alpha value, also known as the false discovery rate, used for the results function is `0.01`.

Figure 7 Represents the statistical results after performing differential expression analysis

	baseMean	log2FoldChange	lfcSE	stat	pvalue	padj
ENSG00000000419.12	514.049953	0.062701	0.030696	2.042673	4.108478e-02	0.098896
ENSG00000000457.13	368.893702	-0.041397	0.017641	-2.346693	1.894087e-02	0.052453
ENSG00000000460.16	97.191630	-0.010819	0.031021	-0.348751	7.272759e-01	0.835472
ENSG00000000938.12	3729.767800	0.107293	0.028700	3.738437	1.851677e-04	0.001089
ENSG00000000971.15	85.238781	0.345355	0.091307	3.782352	1.553536e-04	0.000939
ENSG00000001036.13	431.833040	0.078879	0.020203	3.904396	9.446100e-05	0.000616
ENSG00000001084.10	229.194611	0.254566	0.051157	4.976162	6.485731e-07	0.000008
ENSG00000001167.14	421.132386	-0.003115	0.020709	-0.150436	8.804208e-01	0.936020
ENSG00000001460.17	87.888392	-0.116193	0.029392	-3.953244	7.709883e-05	0.000516
ENSG00000001461.16	911.344689	-0.077177	0.027772	-2.778960	5.453320e-03	0.018870

Visualizing the results given by the differential expression analysis we are able to represent the genes which are up or down regulated (genes having expression level higher or lower than normal) by looking at the log2FoldChange column. A positive value indicates the gene is up regulated while a negative value indicates a down regulated gene. MA and volcano are two such plots which are used to the see the statistical results visually.

3.4.1 MA Plots

MA plots are scatter plots which depict the differences in measurements recorded between two conditions (eg healthy vs diseased) by translating the data onto M (log fold change) on Y-axis and A (base mean) on the X-axis.

Genes having similar expression levels group around M=0, implying no notable variations in their levels across both conditions, while points away from M=0 indicate significant differences in the gene expressions. Genes on the positive end reflect up regulation while genes on the negative end show down regulation. One of the MA plot is shown in Figure 8

Since MA plot does not take into account statistical measurements (p values or modified p values), we cannot discern whether genes have statistically significant differences between normal and treated.

Figure 8 Represents a MA plot for comparison between healthy patient data and adult onset still's disease patient data

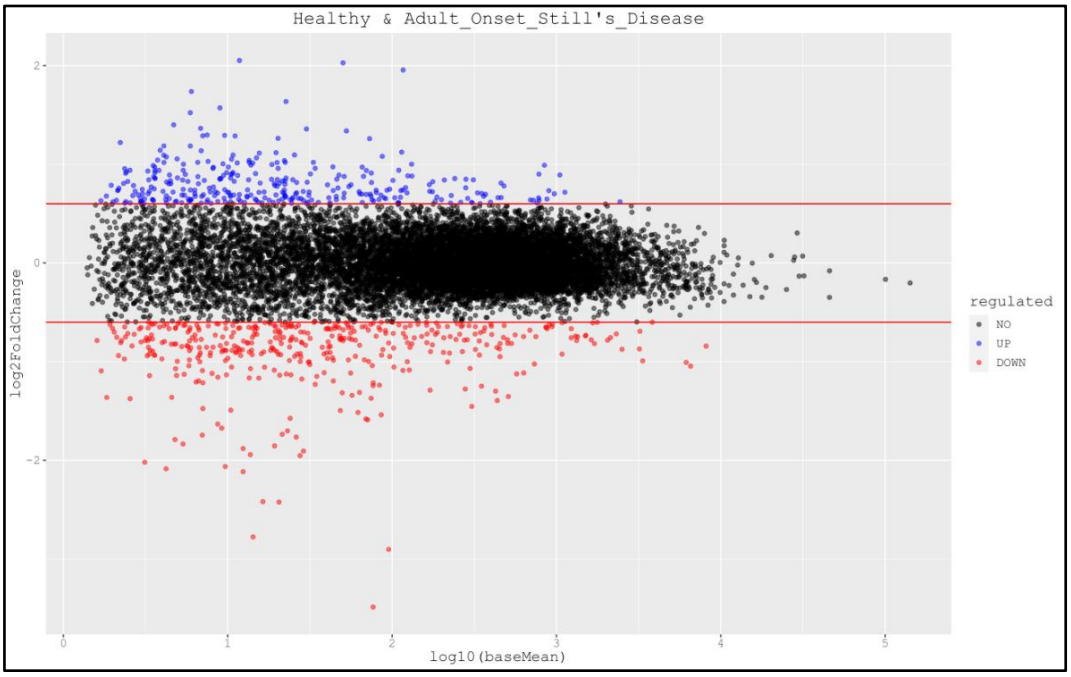


Figure 9 Represents a MA plot for comparison between healthy patient data and anca-associated vasculitis patient data

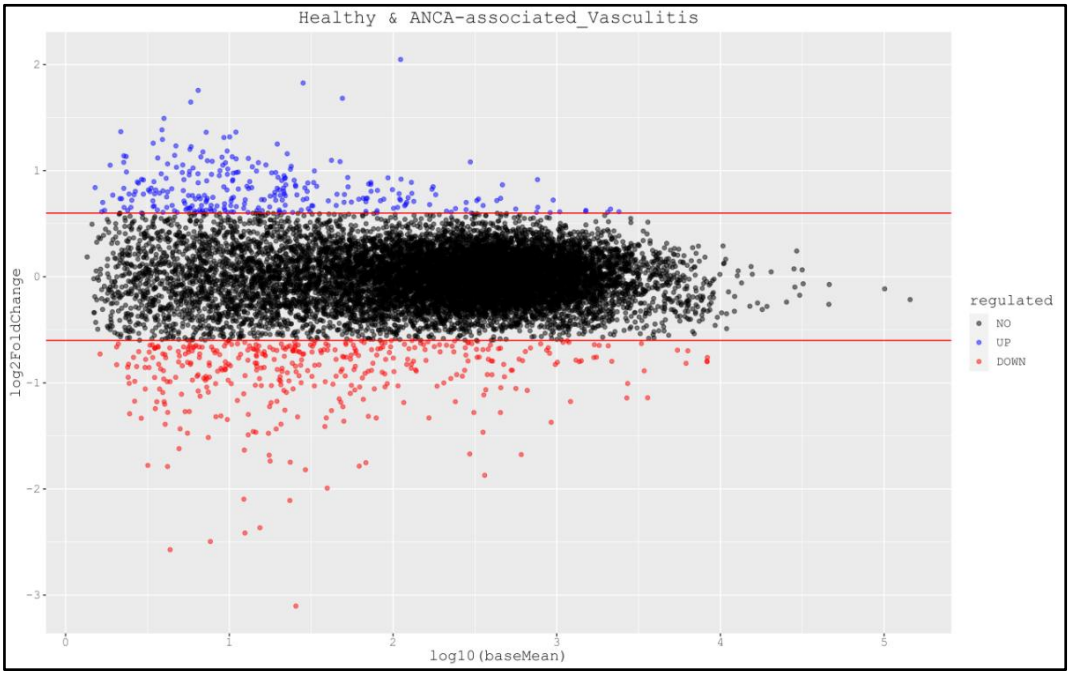


Figure 10 Represents a MA plot for comparison between healthy patient data and behcet's disease patient data

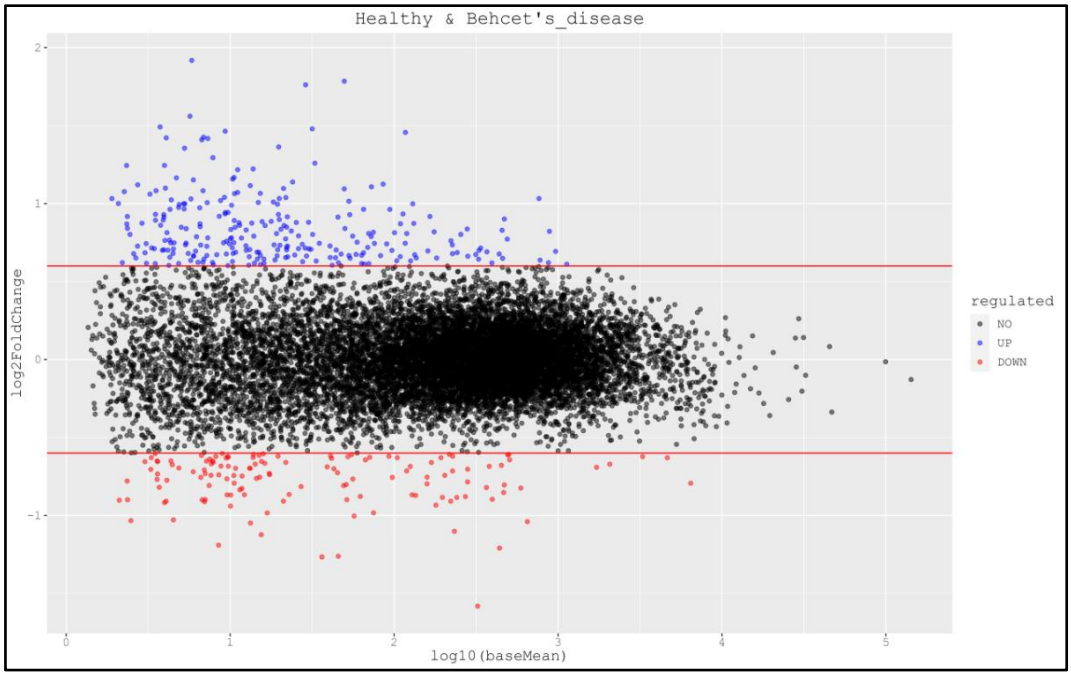


Figure 11 Represents a MA plot for comparison between healthy patient data and dermatomyositis & polymyositis patient data

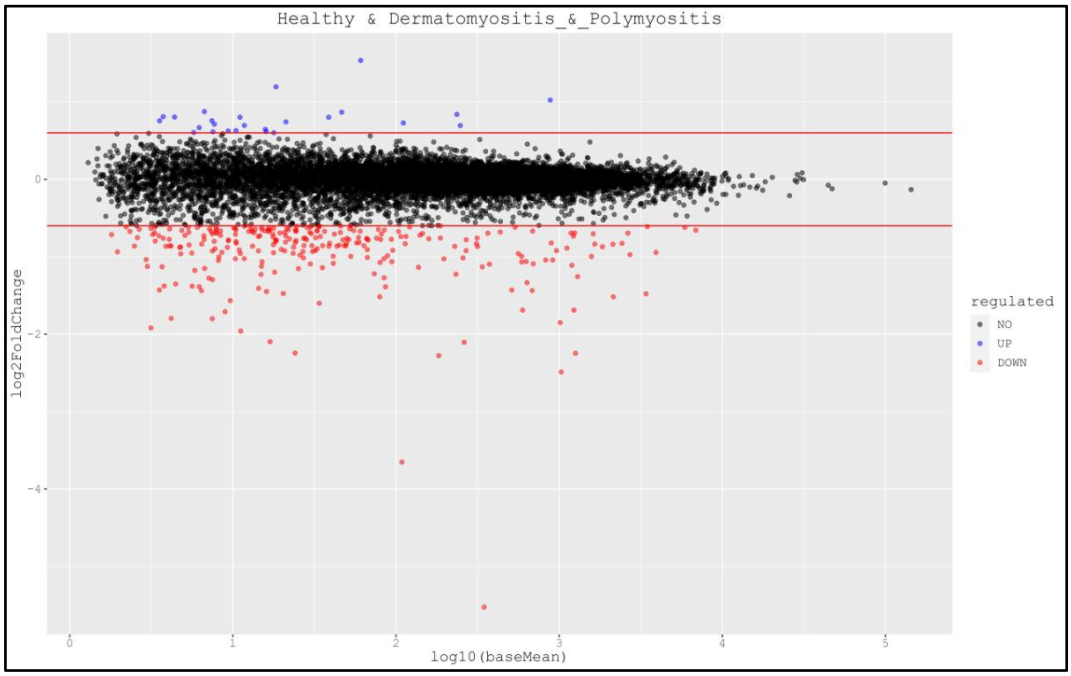


Figure 12 Represents a MA plot for comparison between healthy patient data and mixed connective tissue disease patient data

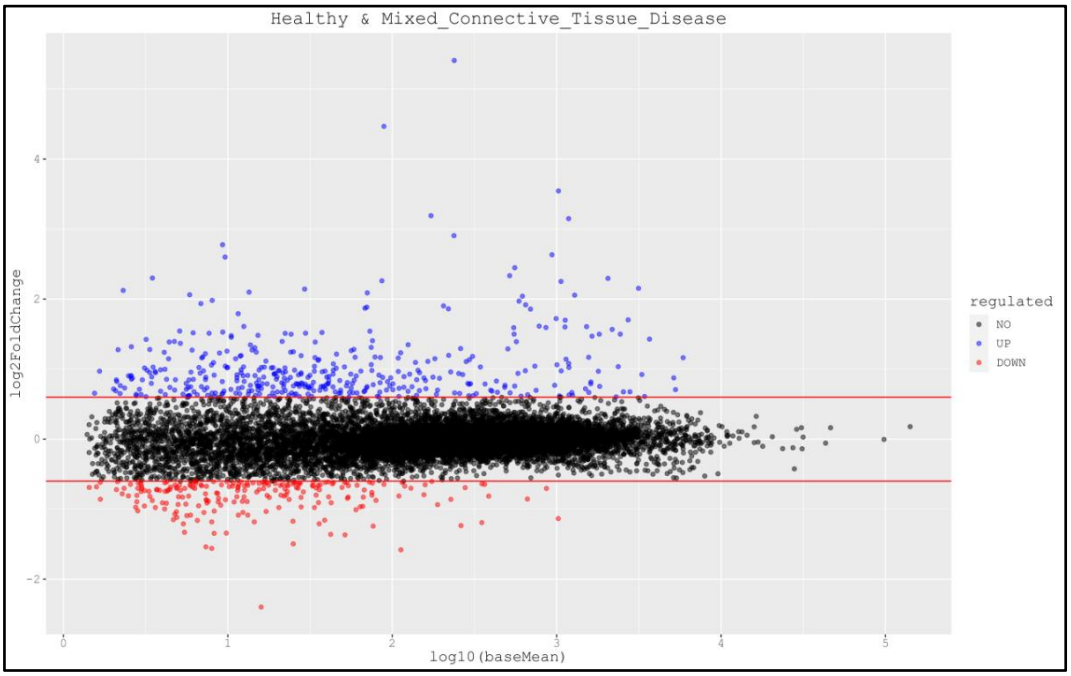


Figure 13 Represents a MA plot for comparison between healthy patient data and infected patient data

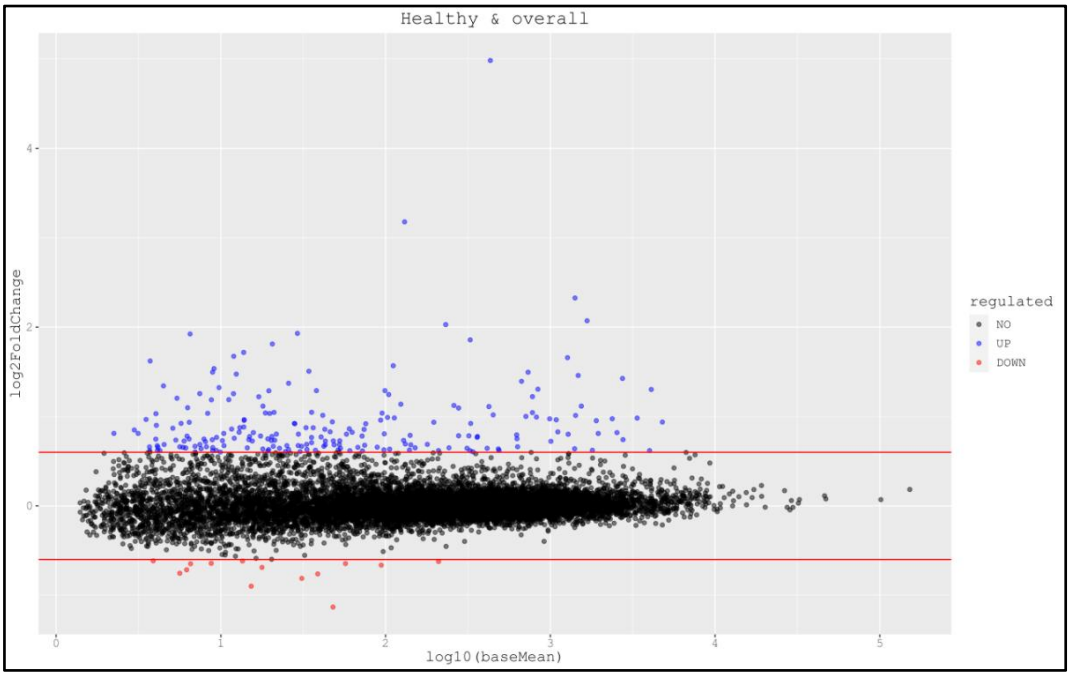


Figure 14 Represents a MA plot for comparison between healthy patient data and rheumatoid arthritis patient data

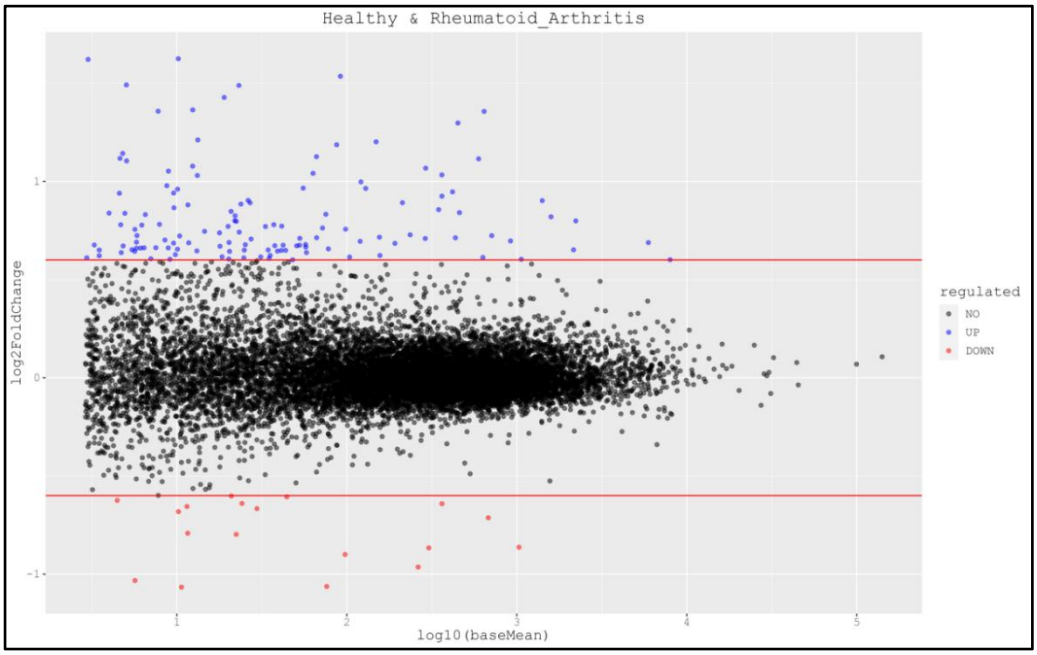


Figure 15 Represents a MA plot for comparison between healthy patient data and sjogren's syndrome patient data

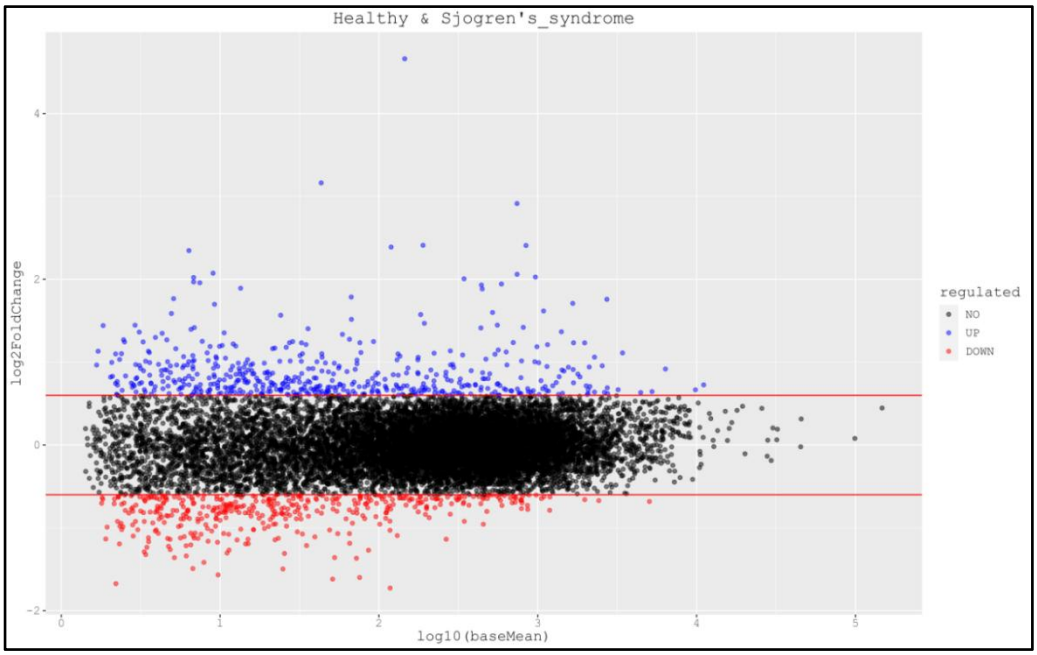


Figure 16 Represents a MA plot for comparison between healthy patient data and systemic lupus erythematosus patient data

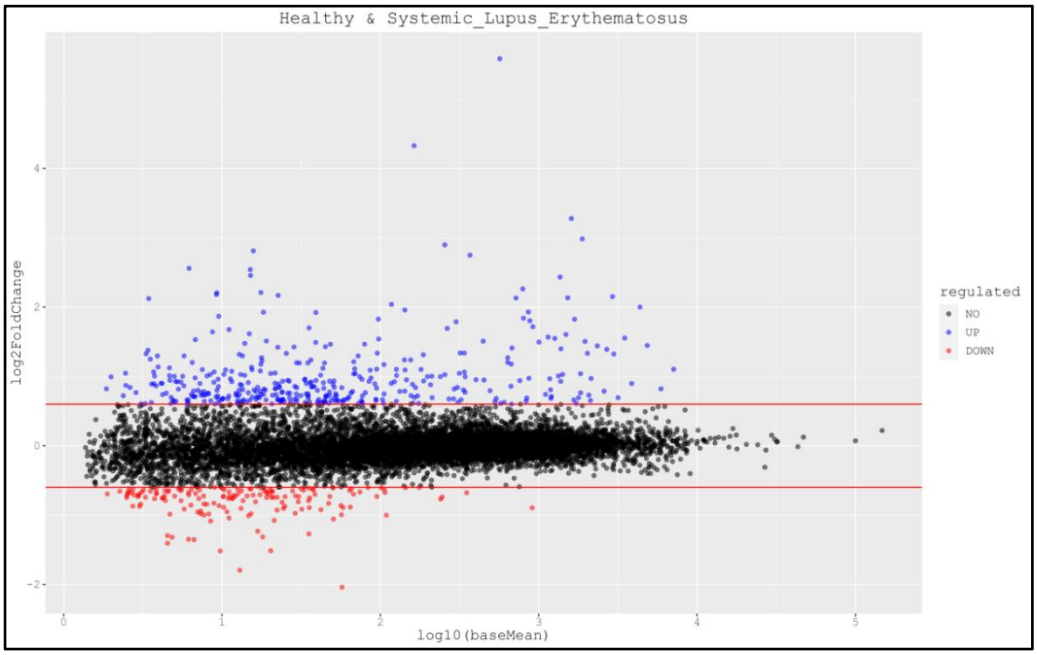


Figure 17 Represents a MA plot for comparison between healthy patient data and systemic sclerosis patient data

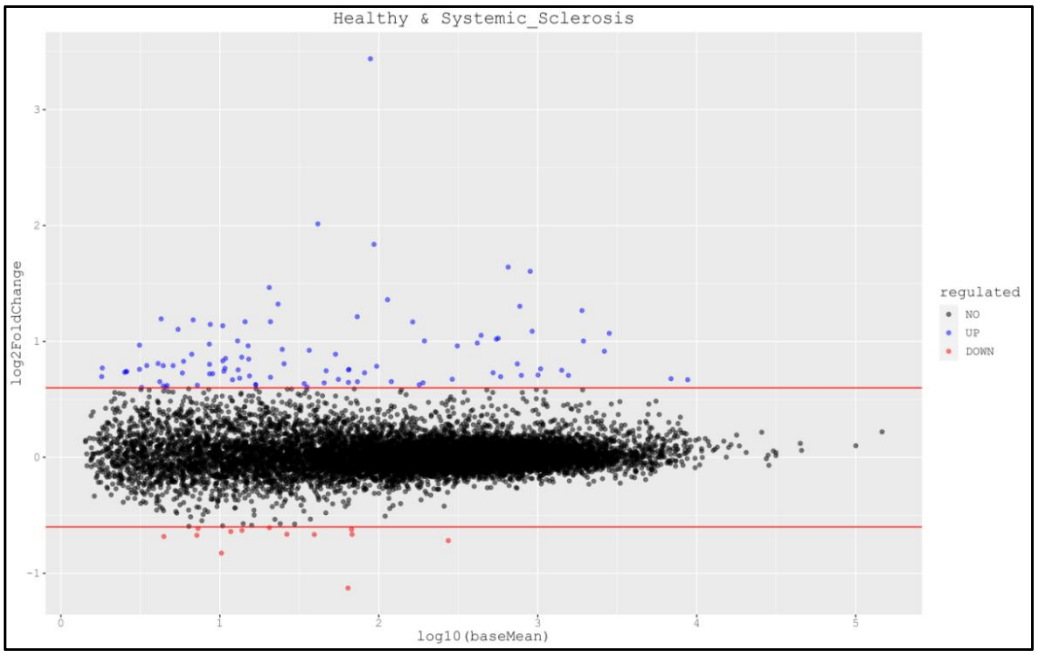
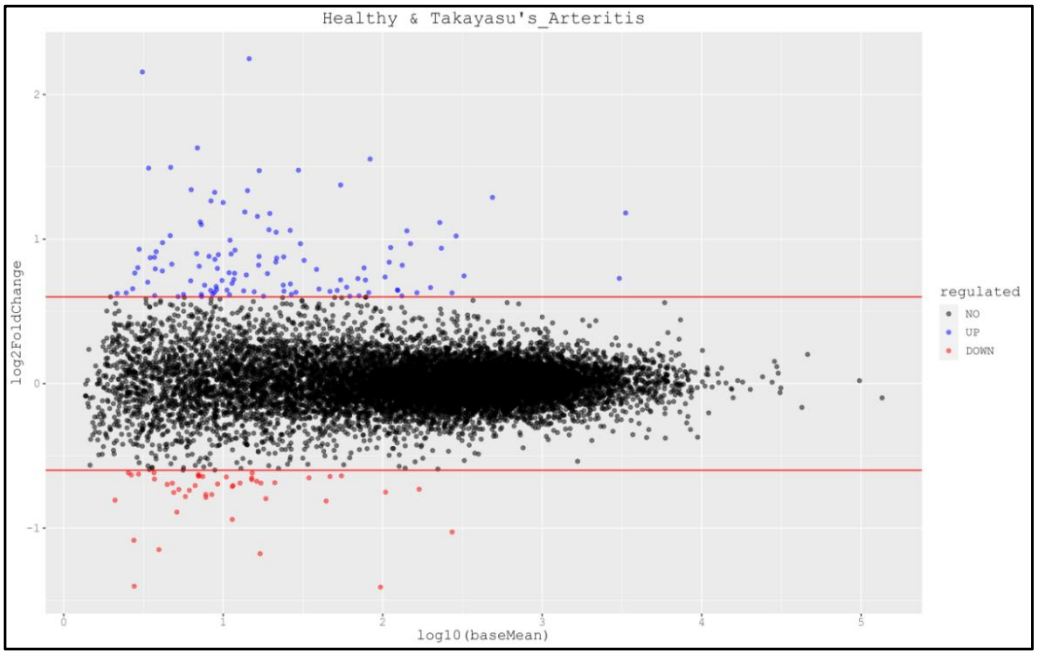


Figure 18 Represents a MA plot for comparison between healthy patient data and takayasu's arthritis patient data



3.4.2 Volcano Plots

Volcano plot combines statistical significance with the amount of the change, allowing for easy visual identification of data-points (genes in this case) that exhibit substantial magnitude changes that are also statistically significant.

The plotting is done by keeping log fold change on X-axis; as this makes changes in both directions to appear equidistant from the center and negative log10 of p value on Y-axis. Data points plotted this way resembles an erupting volcano giving the plot it's name. One of the volcano plot is shown in Figure 19.

The higher the point of y-axis, the lower is the p value and therefore the statistical significance of that gene is more. Moreover when the data points in a volcano plot are spread far away the difference in gene expressions between the two conditions is greater.

Figure 19 Represents a volcano plot for comparison between healthy patient data and adult onset still's disease patient data

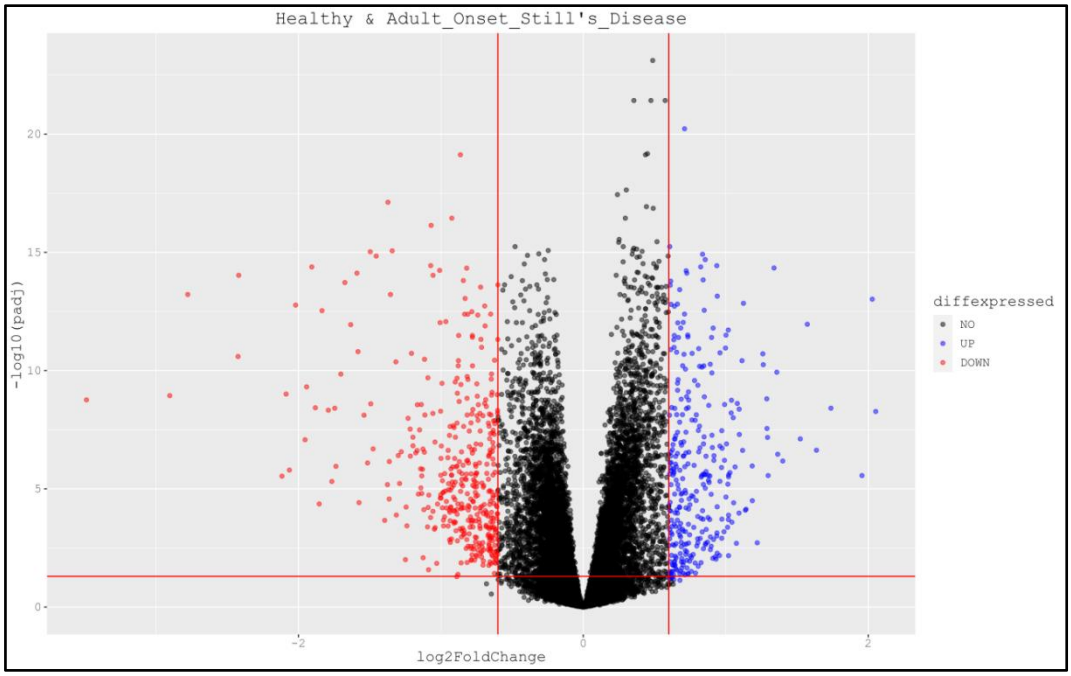


Figure 20 Represents a volcano plot for comparison between healthy patient data and anca-associated vasculitis patient data

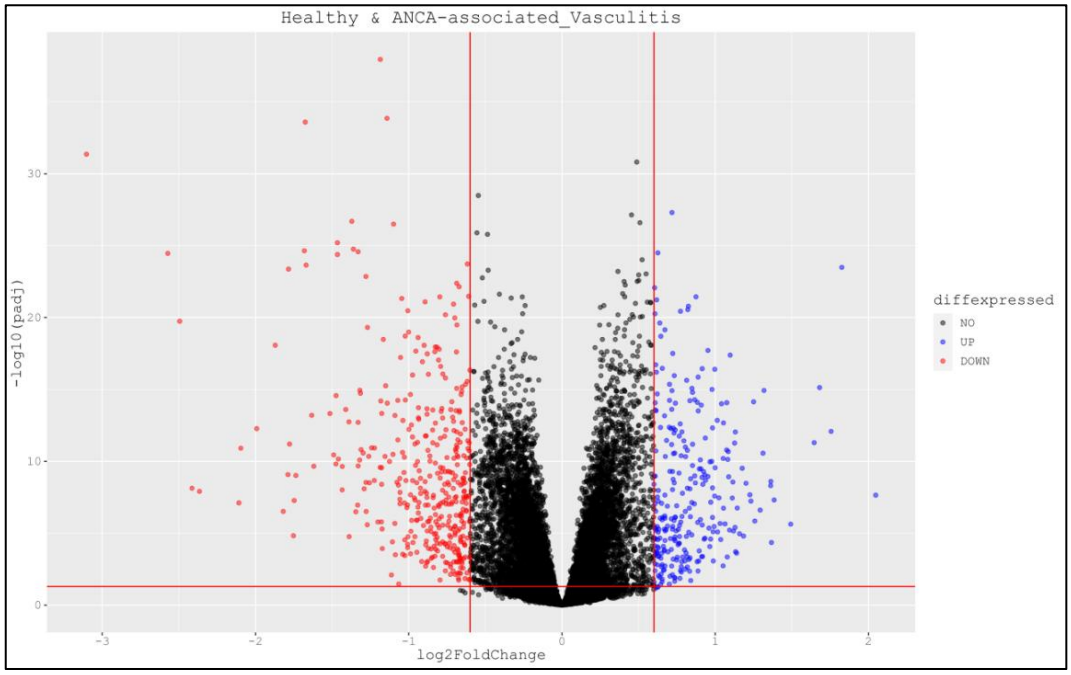


Figure 21 Represents a volcano plot for comparison between healthy patient data and behcet's disease patient data

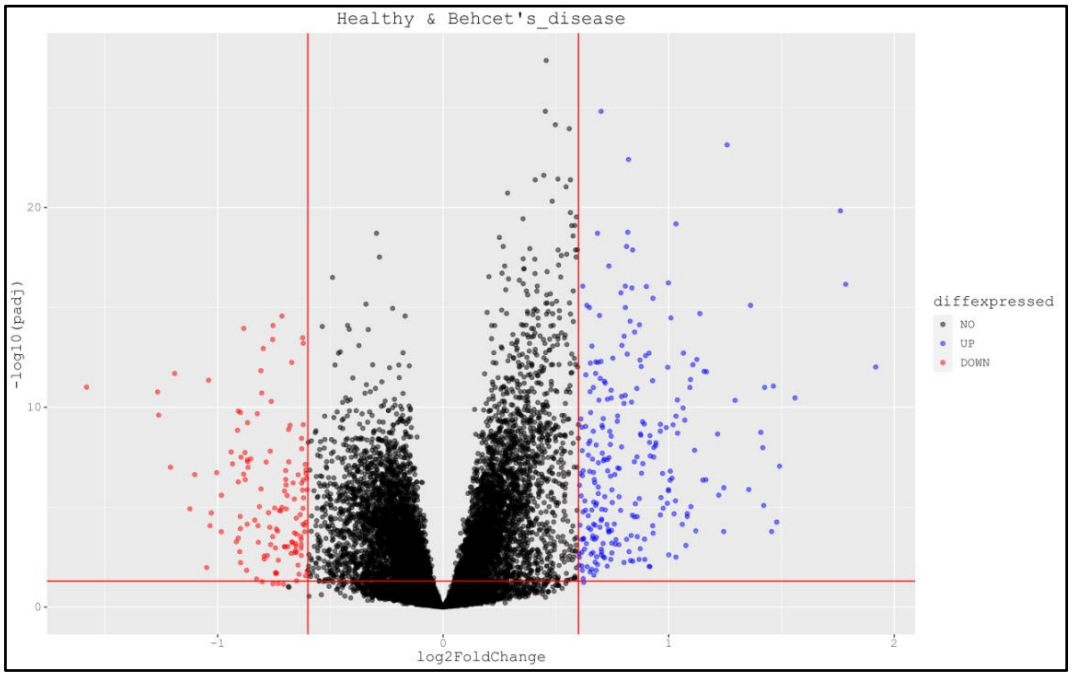


Figure 22 Represents a volcano plot for comparison between healthy patient data and dermatomyositis & polymyositis patient data

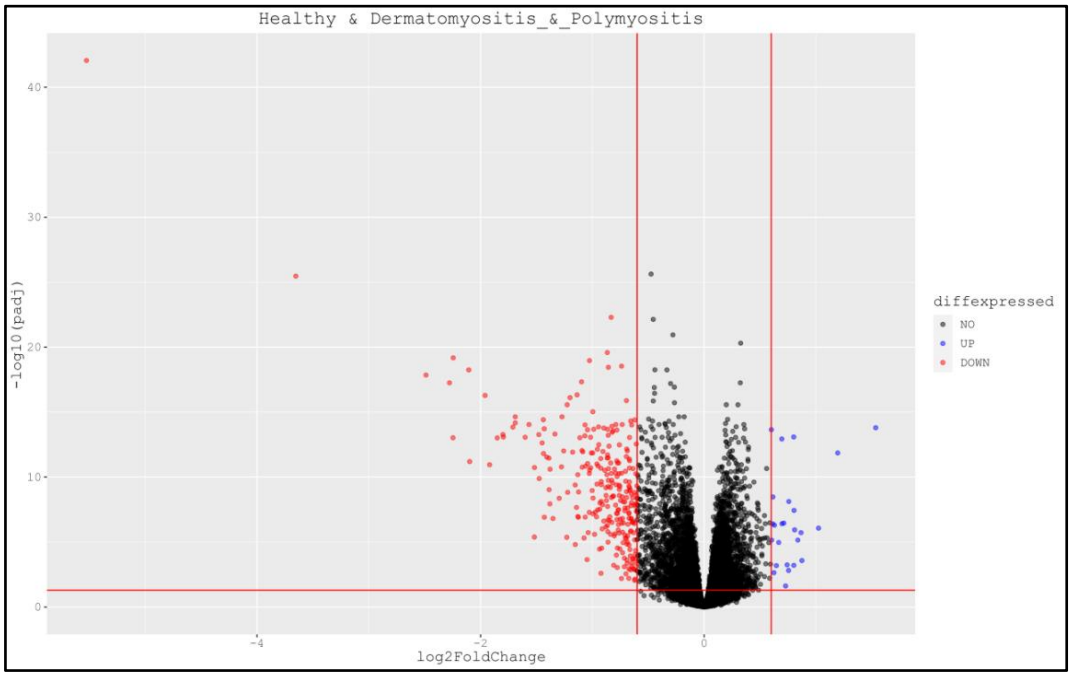


Figure 23 Represents a volcano plot for comparison between healthy patient data and mixed connective tissue disease patient data

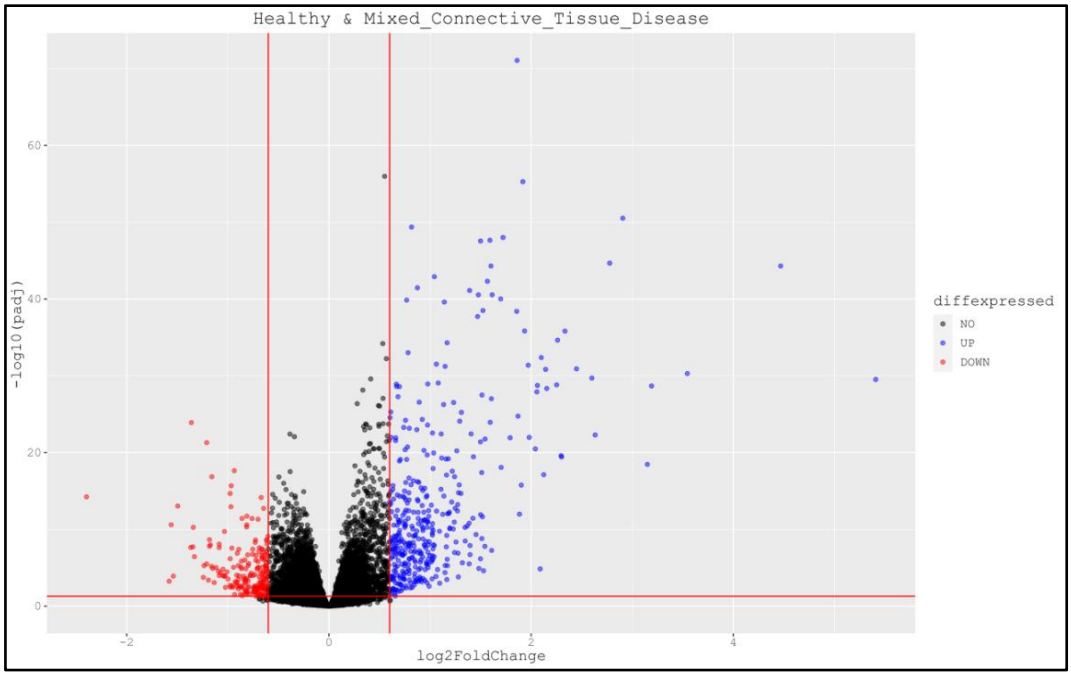


Figure 24 Represents a volcano plot for comparison between healthy patient data and infected patient data

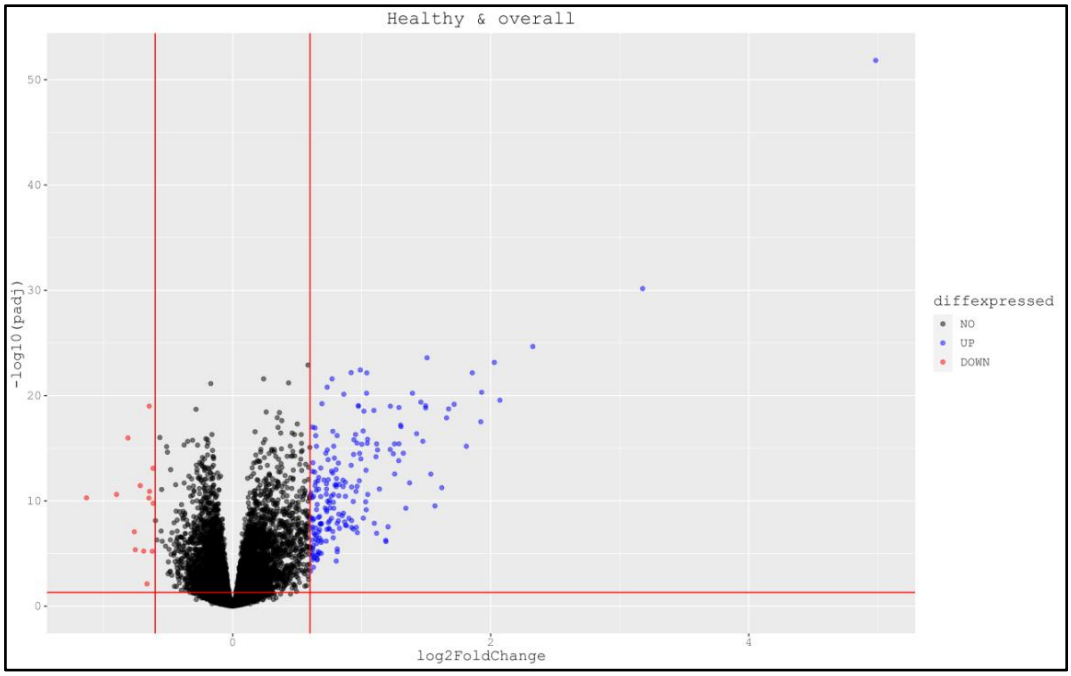


Figure 25 Represents a volcano plot for comparison between healthy patient data and rheumatoid arthritis patient data

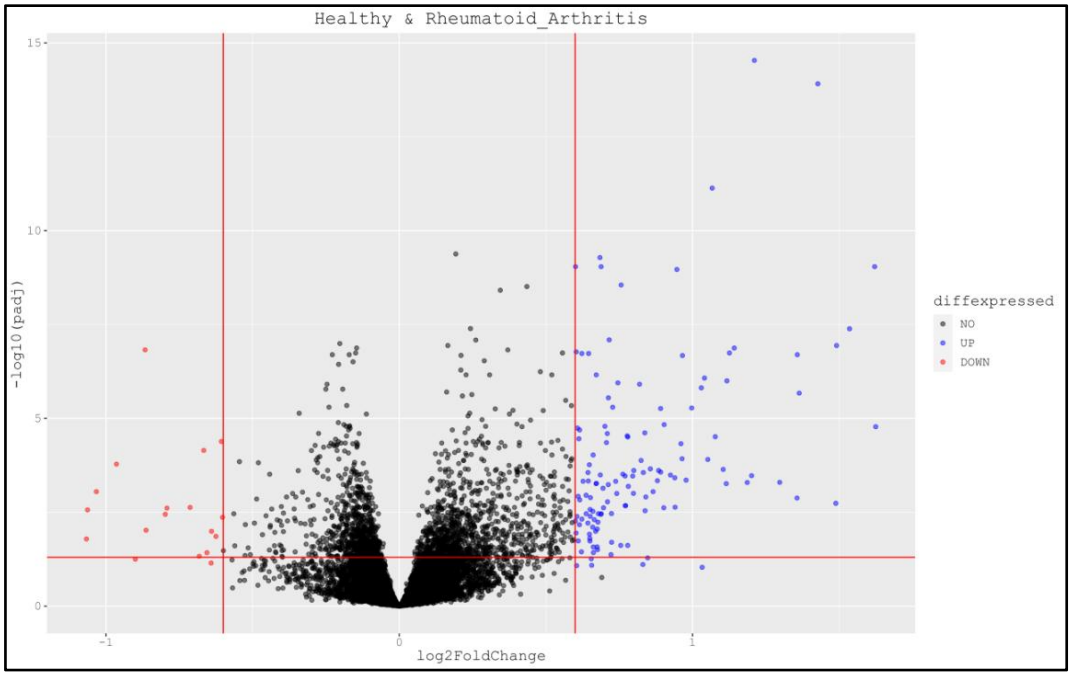


Figure 26 Represents a volcano plot for comparison between healthy patient data and sjogren's syndrome patient data

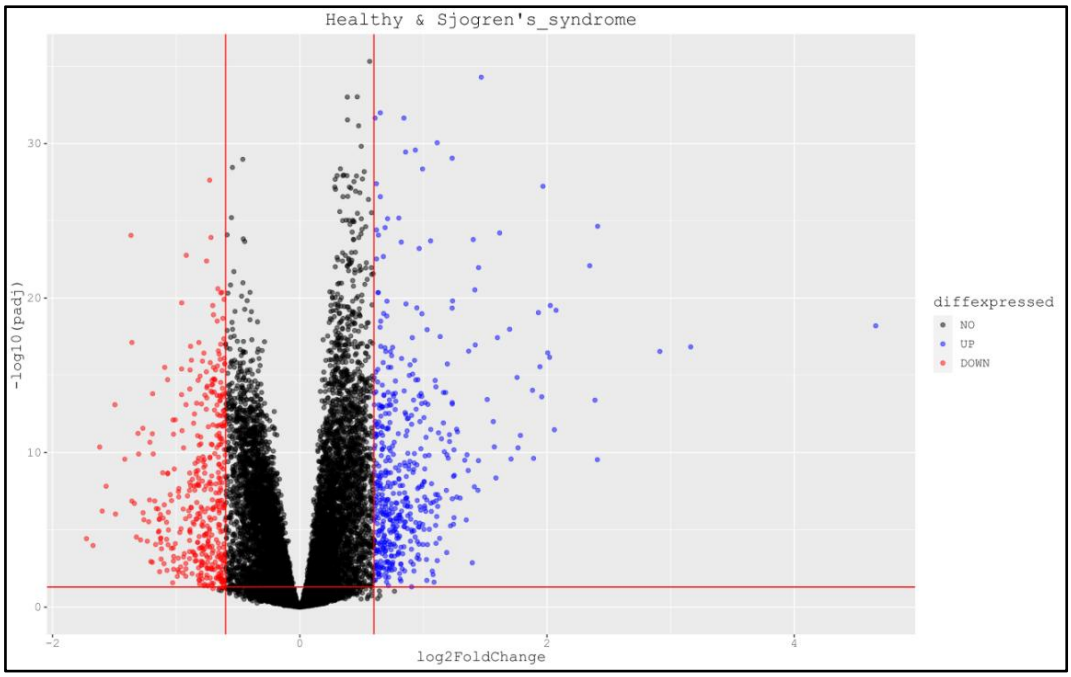


Figure 27 Represents a volcano plot for comparison between healthy patient data and systemic lupus erythematosus patient data

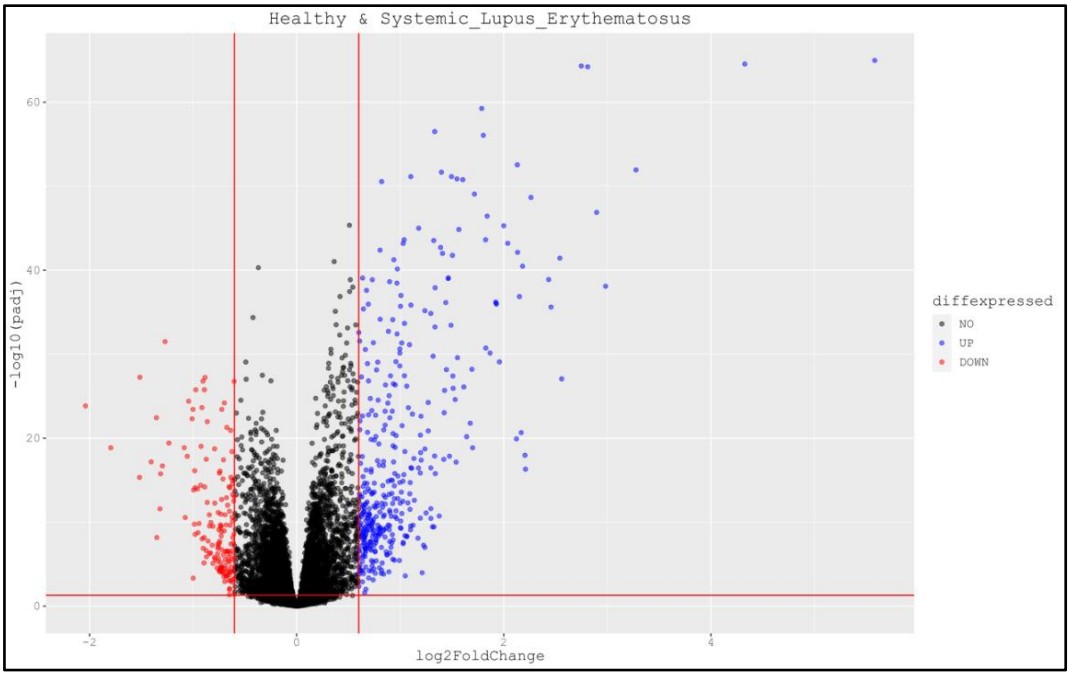


Figure 28 Represents a volcano plot for comparison between healthy patient data and systemic sclerosis patient data

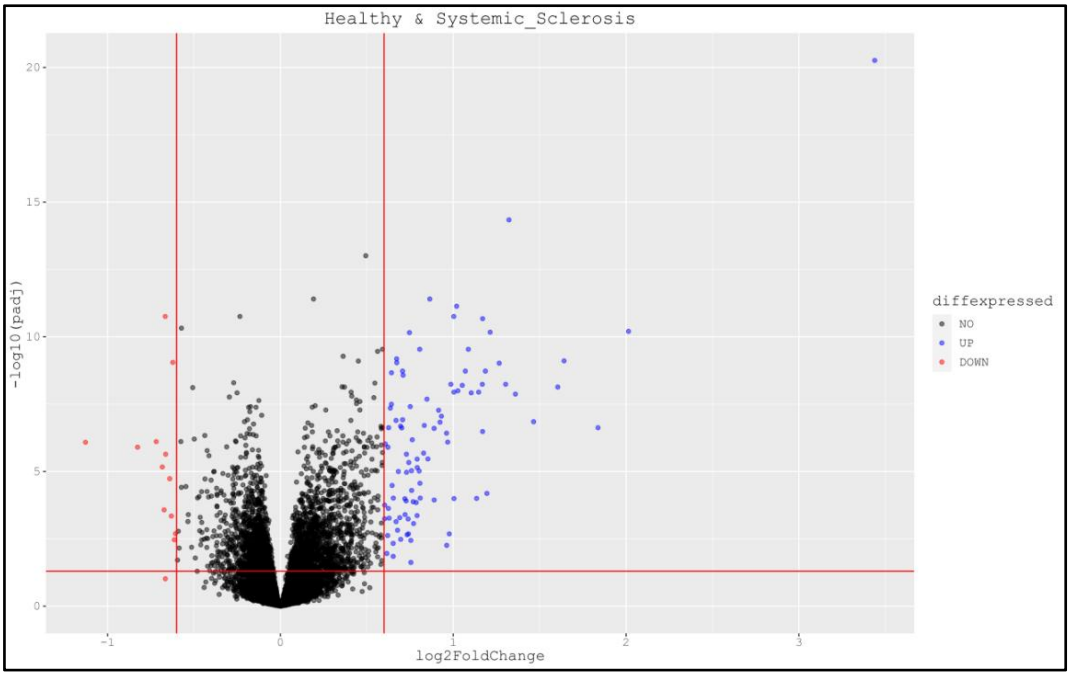
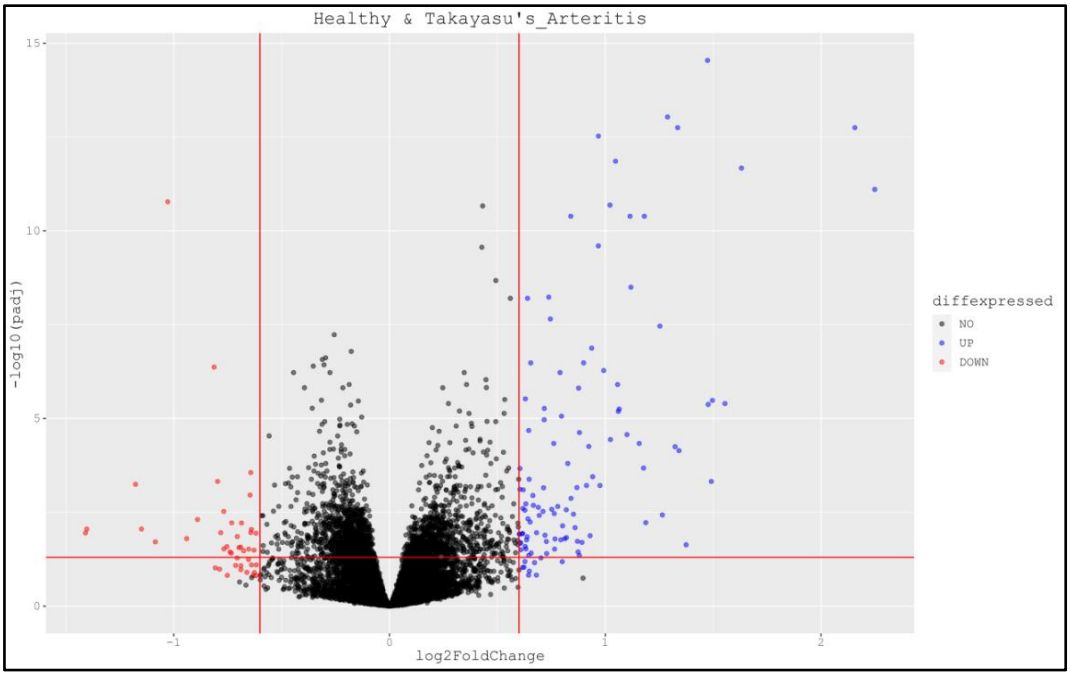


Figure 29 Represents a volcano plot for comparison between healthy patient data and takayasu's arthritis patient data



3.5 Classification

With the proliferation of data generated over the internet, it has become vital to analyze and extract usable information in order to acquire insights and answer various research questions. Big data, however, owing to its volume, variety, velocity and veracity can be hard to deal with using conventional investigative approaches. This problem is overcome by making use of machine learning approaches which are branched into various mechanisms.

The classification domain is concerned with identifying a model that can best describe the input by redistributing and grouping the data into various classes with similar properties. Classification is a supervised learning technique, meaning, the output expected from a sample in the dataset is known beforehand, helping the model learn and later make predictions on unseen data based on the wisdom it gained. The approach taken to perform classification on the data is shown in Figure 30.

3.5.1 Balancing minority classes

The data used in our project contains 11 disease classes out of which some are highly imbalanced. An imbalanced classification problem is an example of a classification problem where the distribution of examples across the known classes is biased or skewed. Imbalanced classifications present a difficulty for predictive modelling since most algorithms for classification are created with the premise of an equal number of samples for each class in mind. As a result, models with poor predictive performance, particularly for minorities, are produced.

One way to solve this problem is to duplicate data in the minority class before fitting the model. This however does not provide any additional information to the model. In our project we utilized a widely adopted technique known as Synthetic Minority Oversampling Technique[8] for oversampling.

SMOTE was implemented using imblearn library, link for which can be found here: https://imbalanced-learn.org/stable/

3.5.2 Principal Component Analysis

A dataset with k numeric attributes/features can be represented graphically as a cloud of dots in a k-dimensional space. In healthcare datasets, the number of attributes (genes in our case) for a specific individual are large. With a high number of dimensions in the feature space, the volume of that space becomes quite enormous, and the points in that space reflect a tiny and non-representative sample.

High dimensionality can have a significant influence on the performance of machine learning algorithms when fitted to data, which is referred to as the curse of dimensionality. As a result, it is beneficial to limit the amount of input features with a technique known as dimensionality reduction. Principal Component Analysis (PCA) is a popular approach which reduces the dimensionality by projecting the data to a lower-dimensional subspace without losing the data's essence. Applying PCA before model fitting helps in reducing the dimensionality, decreases chances of overfitting, and leads to faster training time for algorithm.

PCA was implemented using sklearn library, link for which can be found below:
https://scikitlearn.org/stable/modules/generated/sklearn.decomposition.PCA.html

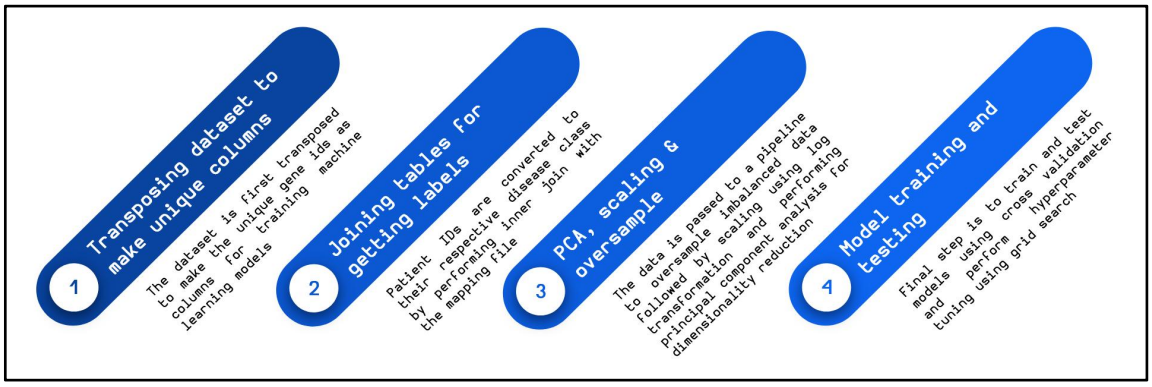
3.5.3 Hyper-parameter tuning

The metrics for a model trained and tested can be improved, sometimes to a significant extent, by making use of the hyper-parameters of a model. Contrary to the parameters of the model (which are learned and tuned by the model algorithm), hyper-parameters are variables which are specified by the user during the configuration of model.

Since it is a difficult task to find the best hyper-parameters of model for a given dataset, techniques like grid search and randomized search are used to optimize a particular cost function. Accuracy as a metric is widely used in hyper-parameter grid search but a recent study has shown matthews correlation coefficient is a more reliable metric for optimization[9].

It is also important to note that when more algorithm hyper-parameters are to be tuned, the procedure becomes slower. As a result, it is preferable to search for a subset of hyper-parameters. In this project, grid search was used to optimize the matthews correlation coefficient for every model. Grid search was implemented using sklearn library, link for which can be found below:
https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Figure 30 Steps taken to prepare data for training and testing multiclass classification using various models



3.5.4 Models

3.5.4.1 Support Vector Machines

Support Vector Machines is an extremely popular supervised learning algorithm which is used for classification as well as regression tasks. Classification in SVM is accomplished by the use of a hyperplane which acts a decision boundary which distinguishes between classes. The most important aspect of SVM algorithms is determining how to draw or establish the decision boundary while also maximizing the distance to the support vectors (the points which are the closest and help in the creation of the hyperplane).

The data points however are not always linearly separable. In these circumstances, SVM employs the kernel trick, which assesses the similarity (or proximity) of data points in a higher dimensional space in order to make them linearly separable. The most used kernels in SVM are polynomial, linear and radial basis function (rbf) kernel.

SVM does not natively support multi-class classification. One way to perform multi-class classification for such classifiers is to split the dataset into various binary classification datasets and fit a binary classifier to each one of them. The One-vs-Rest and One-vs-One tactics are two instances of this strategy. Our project makes use of One-vs-One technique.

One-vs-One strategy uses a total of K(K - 1)/2 binary discriminant functions, one for each conceivable pair of classes; for example a multi-class classification problem with 4 classes A, B, C, D is divided into 6 binary classification problems namely A vs B, A vs C, A vs D, B vs C, B vs D, C vs D. Each point is then categorized based on a majority vote among the discriminant functions.

Support Vector Machines classifier model was implemented using the sklearn library, the link for which can be found below:
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

3.5.4.2 K-Nearest Neighbors

K-Nearest neighbors aka KNN is a supervised learning technique which is mainly used for classification purpose. KNN creates predictions straight from the training dataset. For each new instance, predictions are formed by searching the whole training set for the K most similar examples (the neighbors) and summarizing the output variable for those K instances. This might be the modal (or most common) class value in classification.

A distance metric is used to identify which of the K examples in the training dataset are most similar to a new input. Distance measures include euclidean, hamming, minkowski, manhattan, jaccard, and others. Aside from the distance measure, another hyper-parameter that may be modified is the value of k. (the number of neighbors). In this project, the value of k is set to 11 in order to classify 11 illness kinds.

Since KNN is used widely in various fields, a number of different names have been given to it. Some of the names are mentioned below:

1. Lazy Learning: No model learning is required, and all work is completed when a prediction is sought. As a result, KNN is also known as a lazy learning algorithm.

2. Non-parametric: Owing to the fact that KNN makes no assumption about the underlying distribution of the data, it is also referred to as a non parametric learning algorithm.

3. Instance-Based Learning: Predictions are made using raw training instances. As a result, KNN is frequently referred to as instance-based learning or case-based learning, where each training instance is a case from the problem domain.

Following parameters needs to be considered before using KNN to get good prediction results:

i. Lower Dimensionality: Since KNN is an instance based algorithm, it benefits from lower dimensionality of data which not only reduces the input feature space but also helps the model in preparing better generalizations.

ii. Data Rescale: KNN works substantially better when all of the data is of the same scale. Hence, before fitting the model, it is a good idea to standardize the data.

iii. Addressing Missing Data: If there is missing data, the distance between samples cannot be determined. These samples might be eliminated, or the missing values could be substituted.

K-nearest neighbors classifier model was implemented using the sklearn library, the link for which can be found below:
https://scikitlearn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

3.5.4.3 Gaussian Naive Bayes

Gaussian Naive Bayes classifier is part of a family of supervised learning algorithms which are based on a primary assumption that a feature in the dataset is independent of the values contained in another feature for a given class. This is a very strong assumption that is highly unlikely to be true in real-world data. Despite this, the technique works very well on data where this assumption is not true.

In addition to the basic notion for all naive bayes classifiers, gaussian naive bayes assumes the data distribution to be normal/gaussian. The data distribution can be worked out by finding the mean and standard deviation from the training data.

The classifier makes use of bayes theorem to find the conditional probability (represented by Figure 31) of a class given an input vector using the prior probabilities of the classes and conditional probabilities of each input value given each class value. A predicted class for an input vector is found by taking the class having maximum value out of all calculated posterior probabilities.

Figure 31 Finding conditional probability using bayes theorem

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

where,
$p(C_k \mid x)$: represents the probability of a class 'k' given the input vector 'x', also known as the posterior probability
$p(x|C_k)$: represents the probability of data 'x' given a class 'k'
$p(C_k)$: represents the probability of class 'k'
$p(x)$: represents probability of vector 'x'

Gaussian Naive Bayes model was implemented using the sklearn library, the link for which can be found below:
https://scikitlearn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

3.5.4.4 Multi Layer Perceptron

The field of deep learning is represented using atomic units called neurons, which act as building blocks for more complex architectures. A neuron in this context is a fundamental unit which is used to mimic the neurons inside the brain to solve complex tasks and develop robust approaches to model difficult problems which may not be feasible using traditional approaches.

Some important terms required for understanding neural networks are listed below:

1. Neuron weights: Neuron weights are values associated with each synapse (the output line of a neuron) which are used to transform input data inside a neural network. A bias is a special kind of weight used in each neuron to offset the output value to a positive or negative side.

2. Activation function: An activation function, also known as a transfer function, is the weighted sum of input to neuron output. It is referred to as an activation function since it controls the threshold at which the neuron is triggered as well as the intensity of the output signal.

3. Input Layer: This is the first layer of the network, also known as the visible layer (since it the visible or exposed part of the network), which takes input from the dataset. The number of neurons in this layer is generally taken as the number of features present in the data.

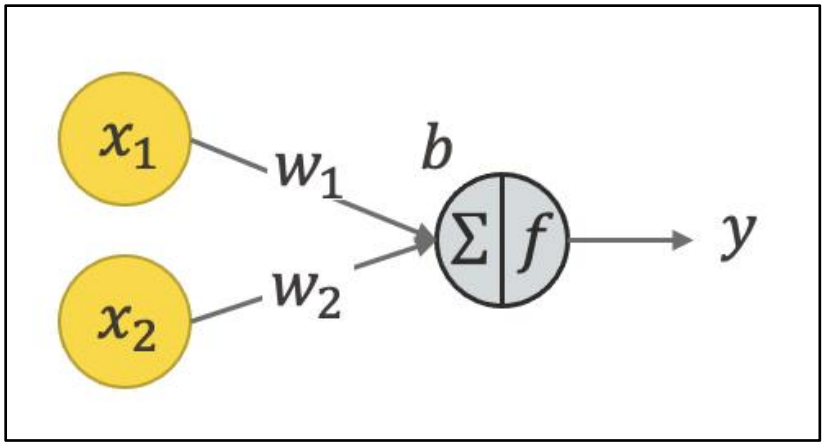
4. Hidden Layer: This layer is sandwiched between the input and the output layer and contains one or more combination of layers of neurons. It is known as the hidden layer since the true value of neurons in this particular layer is not part of the given data.

5. Output Layer: This is the last layer of the network that produces an output for a given task. The number of neurons in this layer for a classification task is the same as the number of classes to be represented.

6. Back Propagation: Back propagation is an algorithm used in artificial neural networks to optimize the error function employing the gradient descent approach. The gradient of weights of a neural network are calculated from last layer to the first layer giving algorithm it's name. The partial computations of gradients of one layer are reused for calculations in other layers making the approach very efficient.

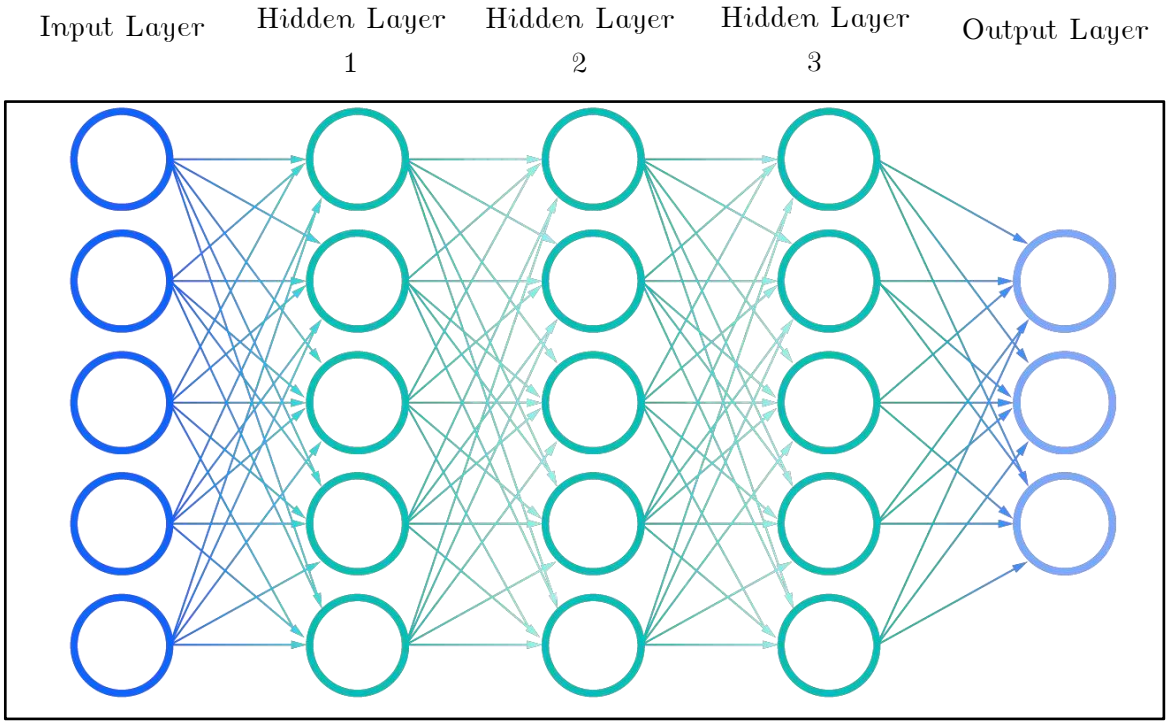
Figure 33 represents a single neuron where x1 and x2 are input to the neuron with weights w1 and w2; b represents the bias, Σ represents the weighted sum of inputs and ƒ represents the activation function used. The weighted sum is passed through an activation function to generate an output for a particular neuron.

Figure 32 The diagram represents a single neuron



Multiple such neurons are used to form complex architectures. The neural network works by optimization of the neural weights by minimization of an error function. The tuning of weights is done by using back propagation.

Figure 33 The figure below represents a multi layer perceptron with 3 hidden layers



Multi Layer Perceptron model was implemented using the sklearn library, the link for which can be found below:
https://scikitlearn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

3.5.4.5 Convolution neural network

Deep Learning has shown to be an extremely useful technique over the last several decades due to its capacity to manage massive volumes of data. Use of hidden layers have eclipsed traditional approaches in popularity, particularly in pattern and image recognition. Convolution Neural Networks are one of the most often used deep neural networks.

Convolution Neural network is a type of deep neural network which is mainly employed in classification tasks involving image or image like data. The primary difference between a standard deep neural network and a

CNN is the convolution layer; which contains tuned kernels/weights that extract characteristics, used to identify differentiating features.

Some important terms related to convolution neural networks are:

i. Stride: The amount of pixels shifted across the input matrix is referred to as the stride. When the stride is 1, the filters are moved one pixel at a time. When the stride is 2, the filters are moved two pixels at a time, and so on.

ii. Pooling: The Pooling layer is in charge of shrinking the spatial size of the convolved matrix. By lowering the size, the computer power required to process the data is reduced. Spatial pooling can be of the following types:

    i. Max pooling: Max Pooling takes the maximum value of a pixel from a portion of the image covered by the kernel.

    ii. Average Pooling: Average Pooling returns the average of all the values from the portion of the image covered by the kernel.

    iii. Sum pooling: Sum Pooling takes the sum of all elements from a portion of the image covered by the kernel.

iii. Padding: The pixel in the corner will only be covered once, whereas the pixel in the centre will be covered several times. To overcome this problem and loss of information at the corners, padding is added to the matrix data

iv. Convolution Layer: The convolution layer performs an element-wise dot product with a unique kernel sweeping across the whole matrix and the output of the same is passed to a series of one or more pooling, normalization and flatten layers (depends on the type of architecture used). Thus there are as many different intermediate outcomes as there are distinct kernels.

Figure 34 Below figure shows the tensorflow representation of the CNN architecture used

```
# CNN architecture
model = Sequential()

model.add(
    Conv2D(32, kernel_size=(5, 5), strides=(1, 1), input_shape=input_shape)
)
model.add(Activation("relu"))
model.add(MaxPooling2D(2, 2))

model.add(Flatten())
model.add(Dense(128, activation="relu"))
model.add(Dense(num_classes, activation="softmax"))

model.compile(
    loss="categorical_crossentropy",
    optimizer="adam",
    metrics=["categorical_accuracy"],
)
```

CNN model was implemented using tensorflow and the architecture used for CNN as shown in Figure 34 was taken from the paper[10].

Chapter 4: Results

Figure 35 Deconvoluted cell proportions of patients belonging to adult onset still's disease using standard matrix LM22

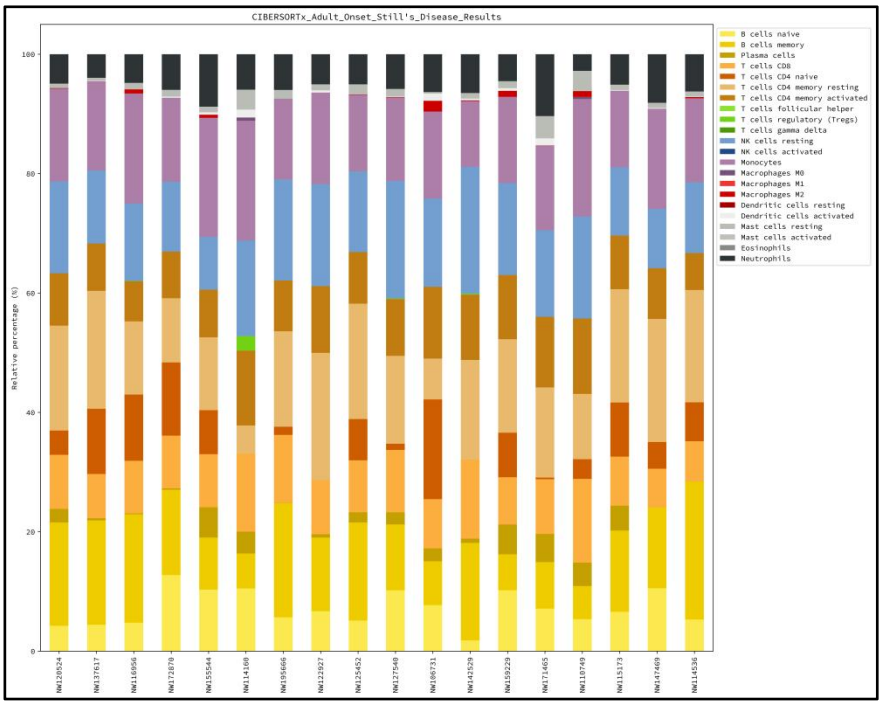


Figure 36 Deconvoluted cell proportions of patients belonging to anca-associated vasculitis using standard matrix LM22

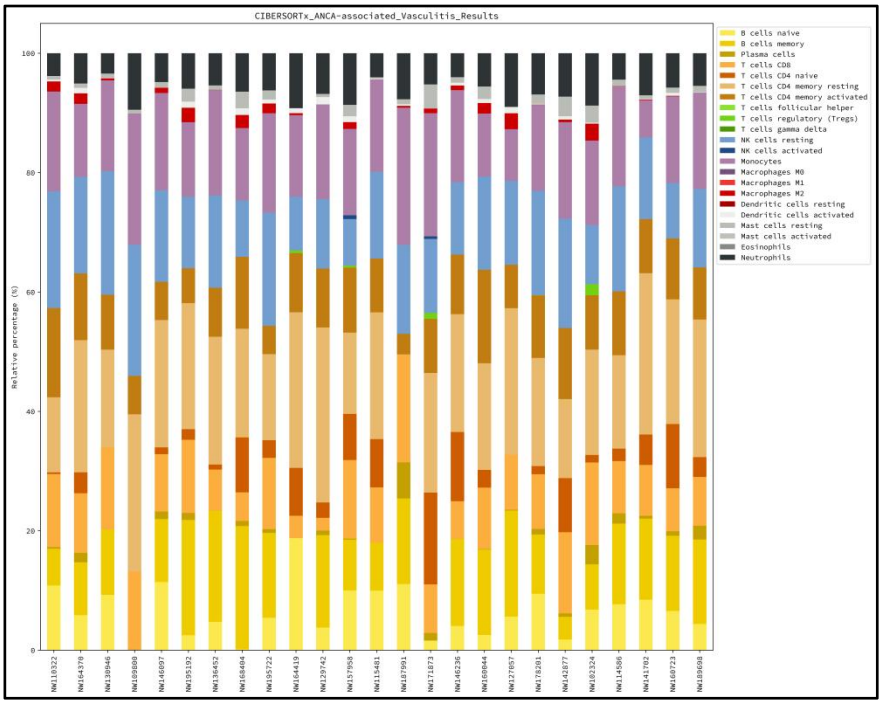


Figure 37 Deconvoluted cell proportions of patients belonging to behcet's disease using standard matrix LM22

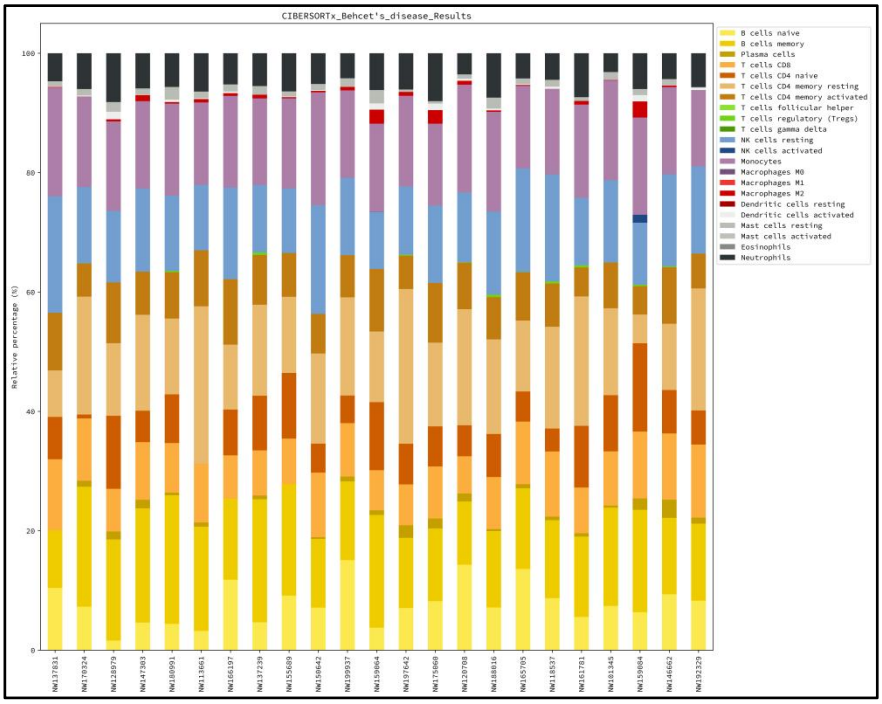


Figure 38 Deconvoluted cell proportions of patients belonging to dermatomyositis & polymyositis using standard matrix LM22

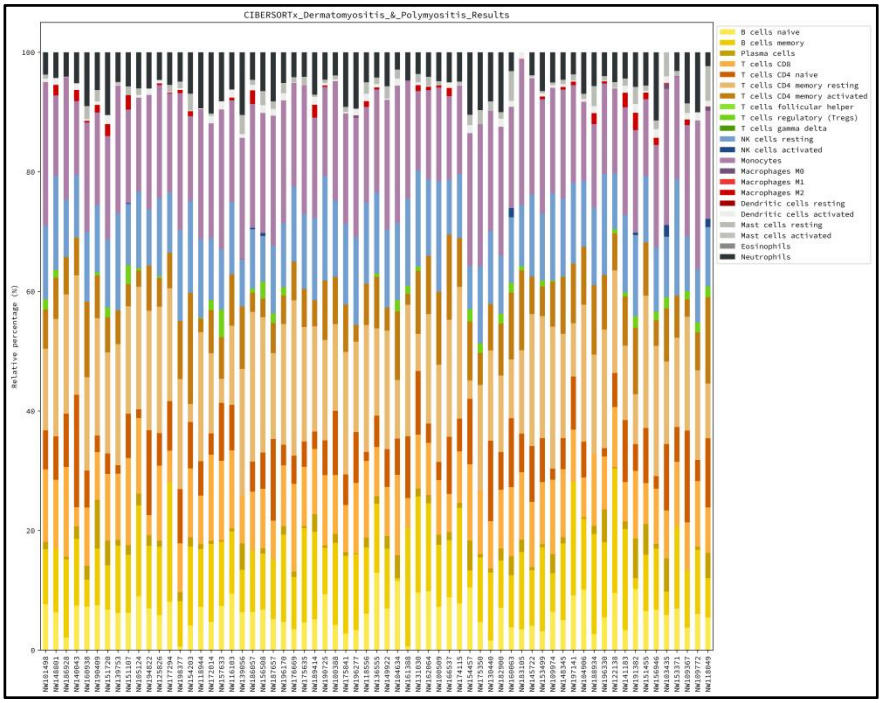


Figure 39 Deconvoluted cell proportions of patients belonging to diseases & healthy using standard matrix LM22

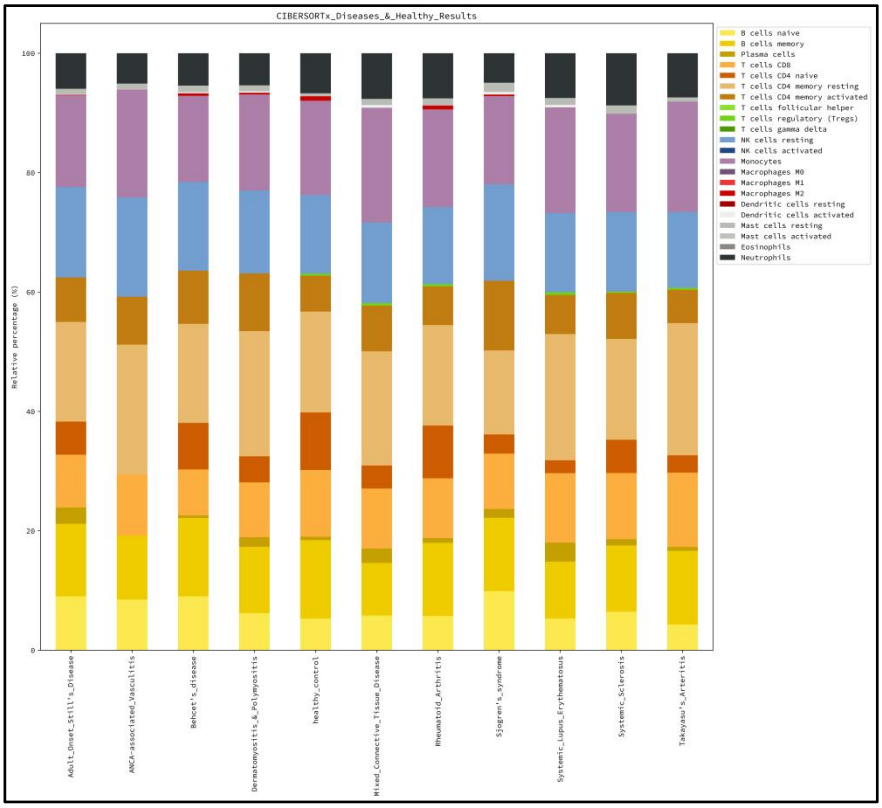


Figure 40 Deconvoluted cell proportions of patients belonging to healthy control using standard matrix LM22

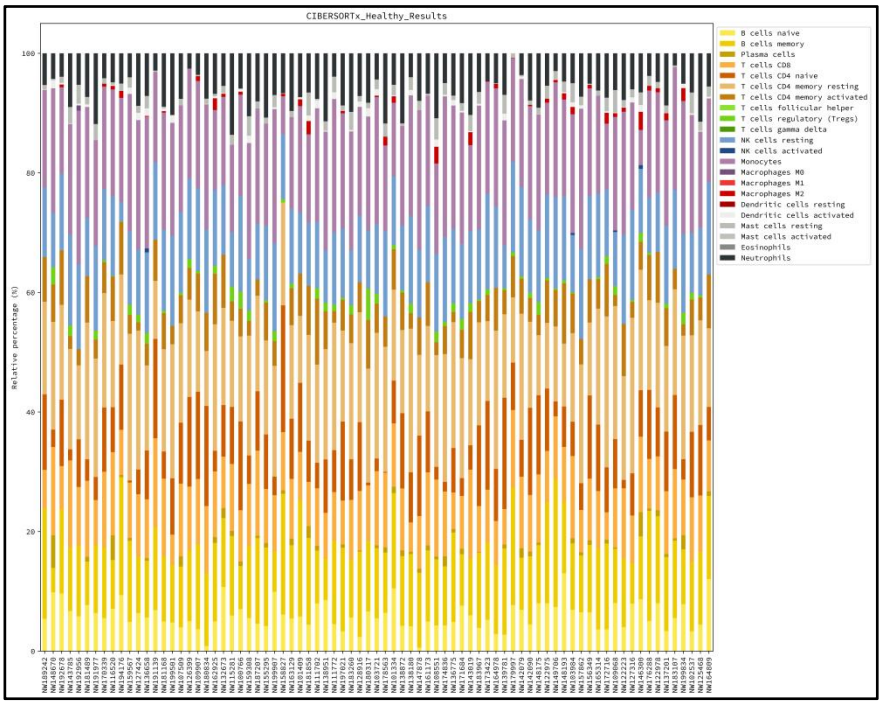


Figure 41 Deconvoluted cell proportions of patients belonging to mixed connective tissue disease using standard matrix LM22

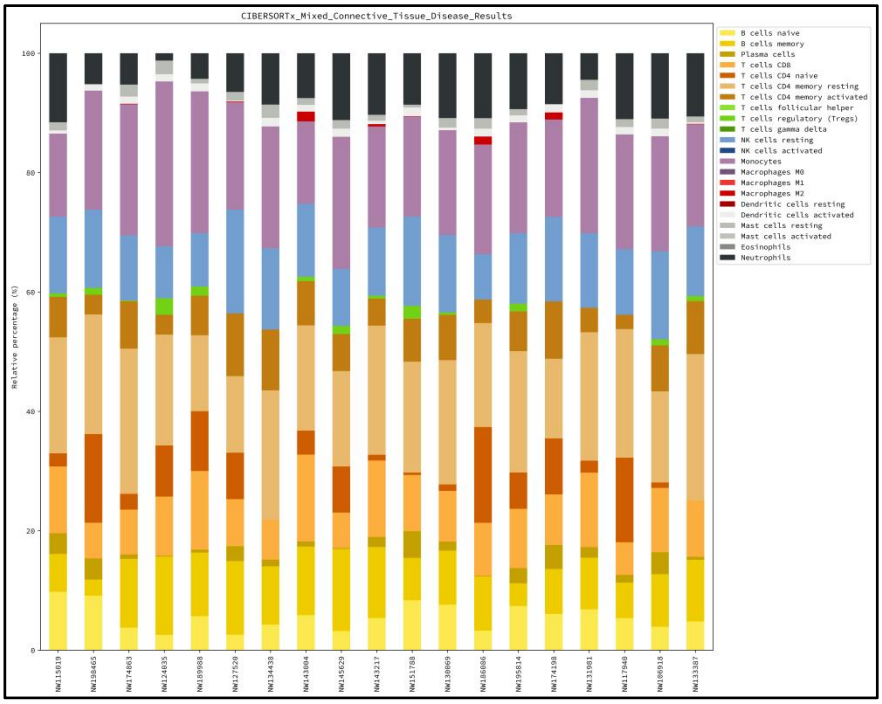


Figure 42 Deconvoluted cell proportions of patients belonging to rheumatoid arthritis using standard matrix LM22

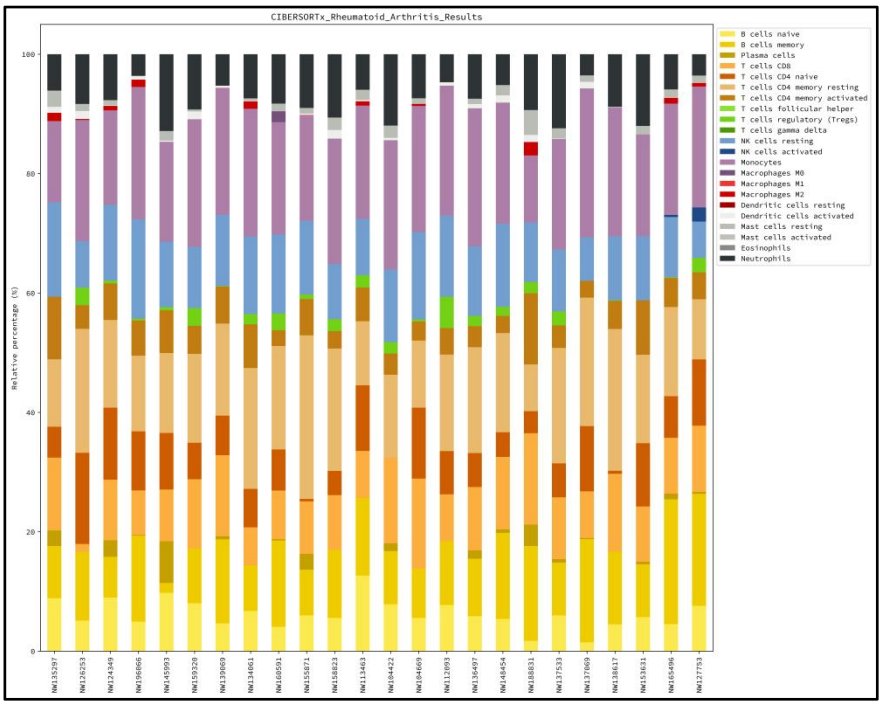


Figure 43 Deconvoluted cell proportions of patients belonging to sjogren's syndrome using standard matrix LM22

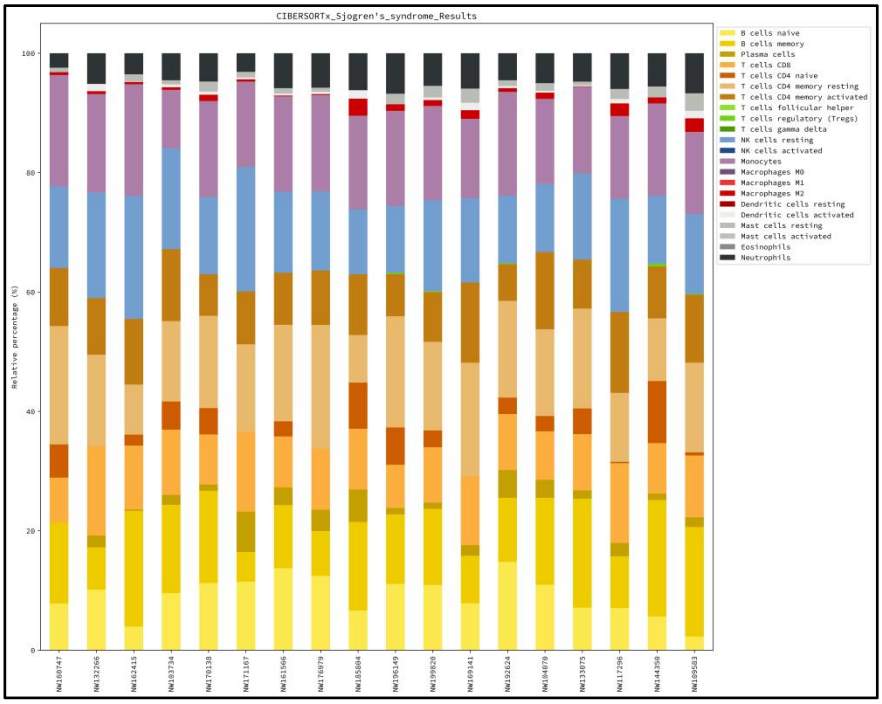


Figure 44 Deconvoluted cell proportions of patients belonging to systemic lupus erythematosus using standard matrix LM22

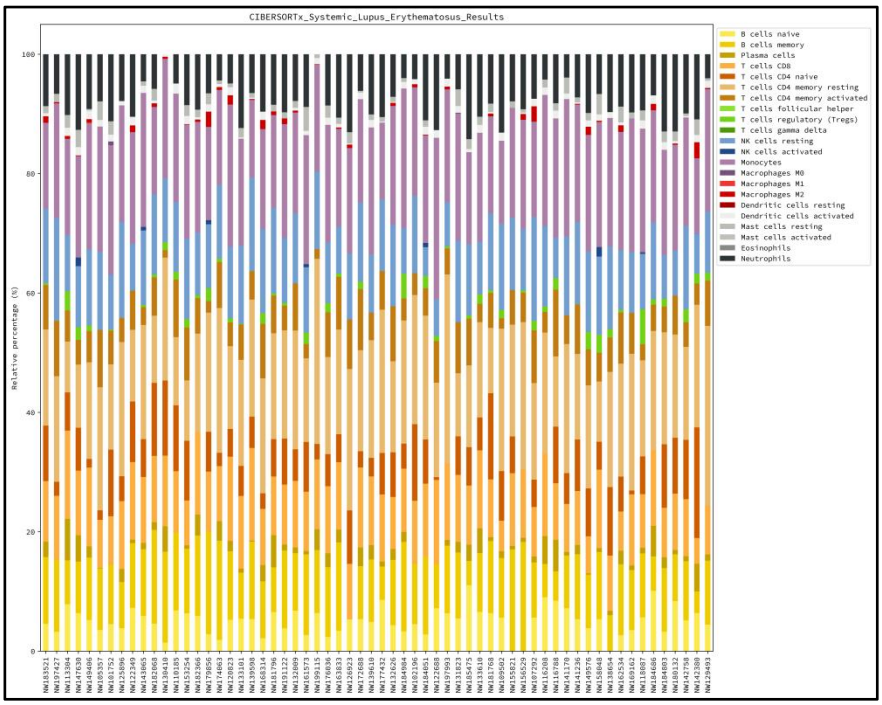


Figure 45 Deconvoluted cell proportions of patients belonging to systemic sclerosis using standard matrix LM22

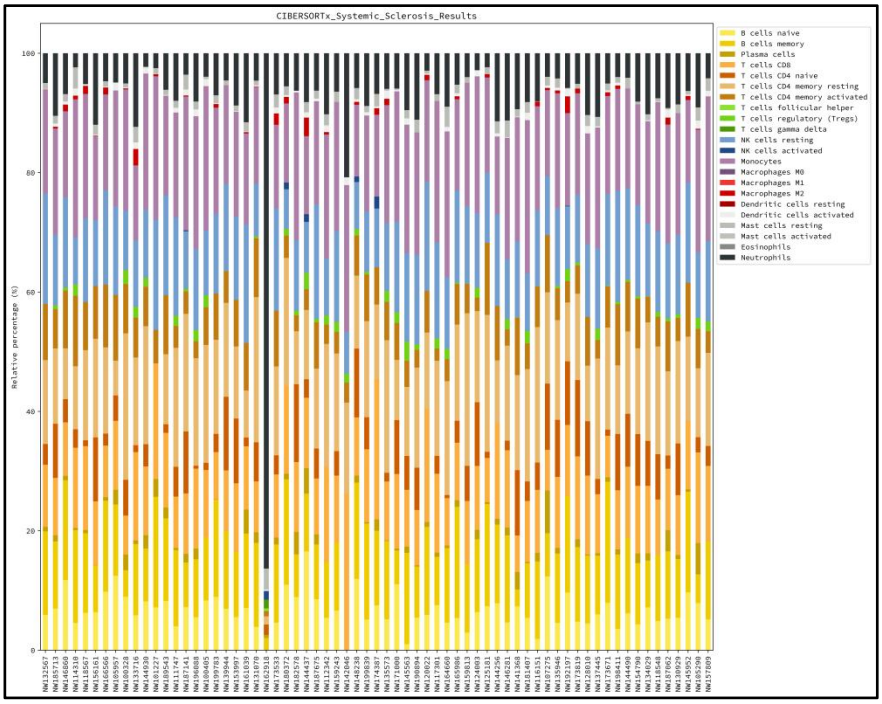


Figure 46 Deconvoluted cell proportions of patients belonging to takayasu's arteritis using standard matrix LM22

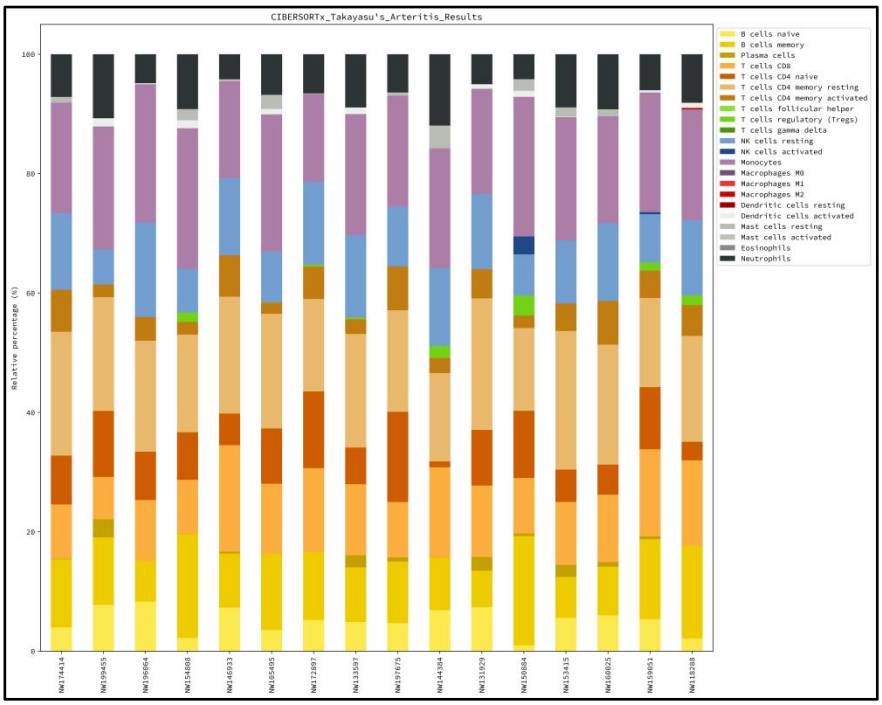


Figure 47 Deconvoluted cell proportions of patients belonging to adult onset still's disease using custom matrix SM28

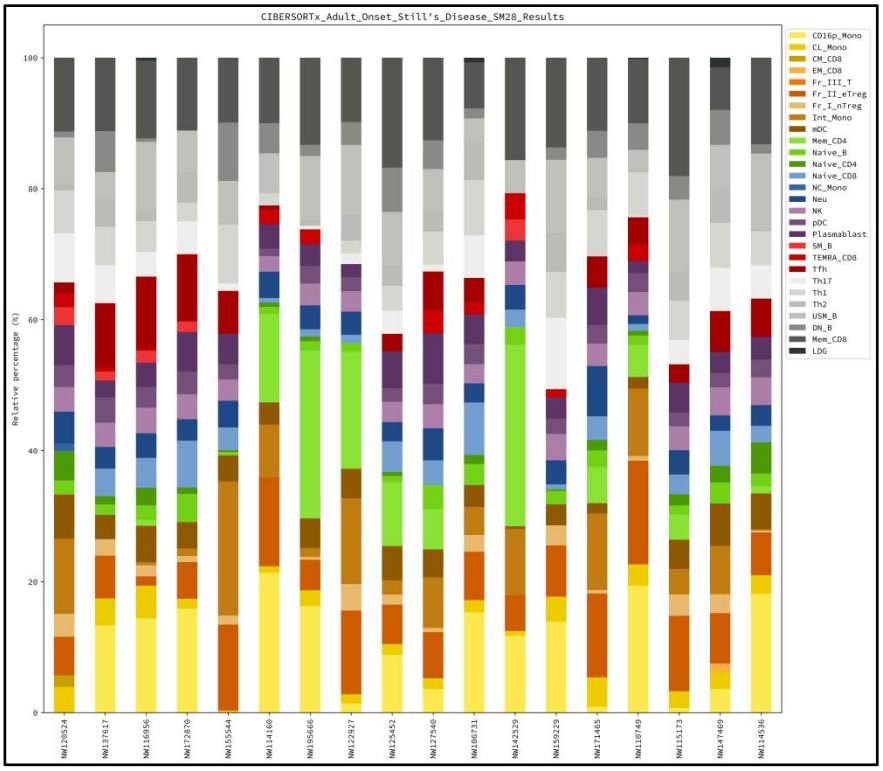


Figure 48 Deconvoluted cell proportions of patients belonging to anca-associated vasculitis using custom matrix SM28

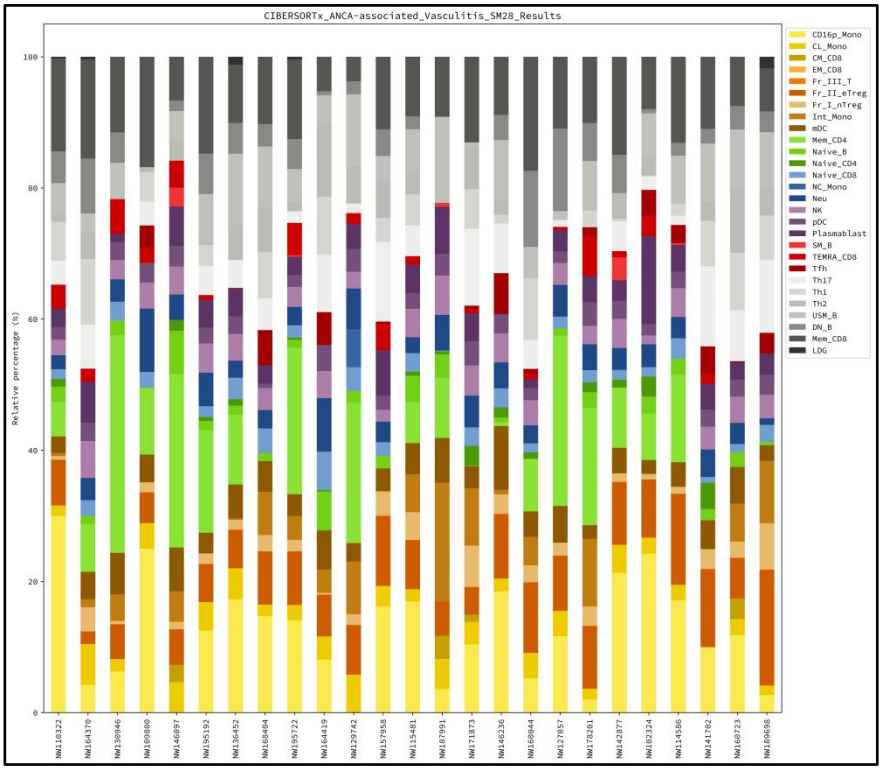


Figure 49 Deconvoluted cell proportions of patients belonging to behcet's disease using custom matrix SM28

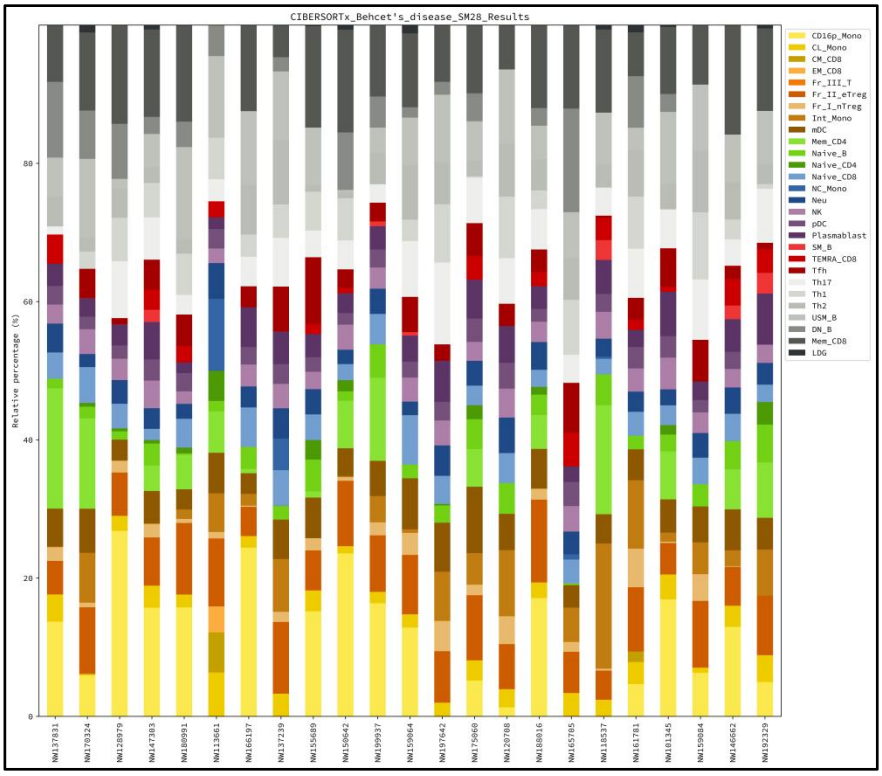


Figure 50 Deconvoluted cell proportions of patients belonging to dermatomyositis & polymyositis using custom matrix SM28

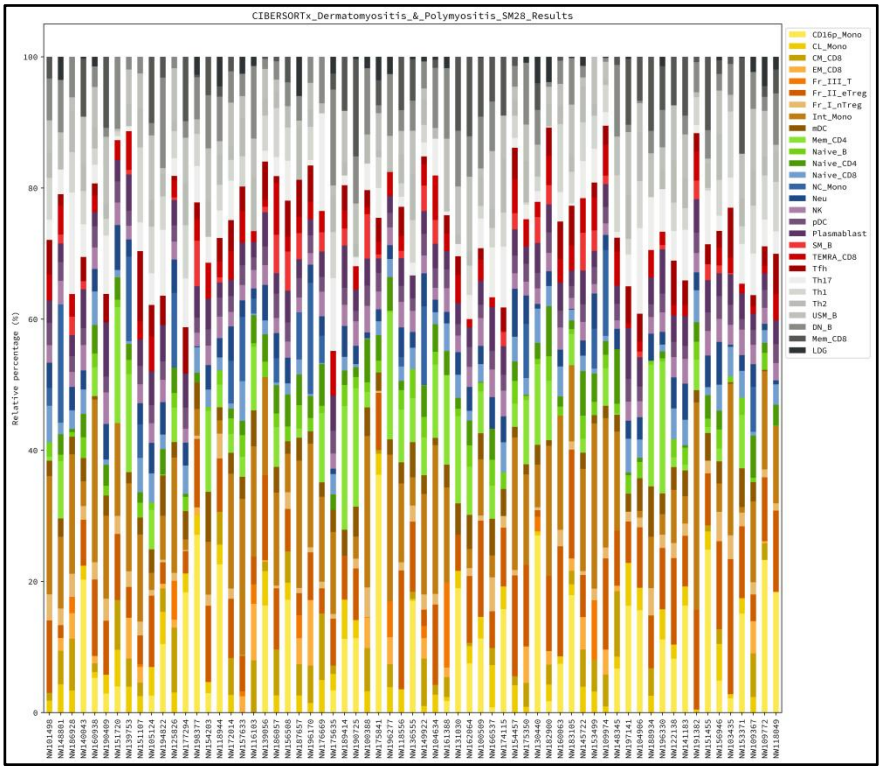


Figure 51 Deconvoluted cell proportions of patients belonging to diseases & healthy using custom matrix SM28

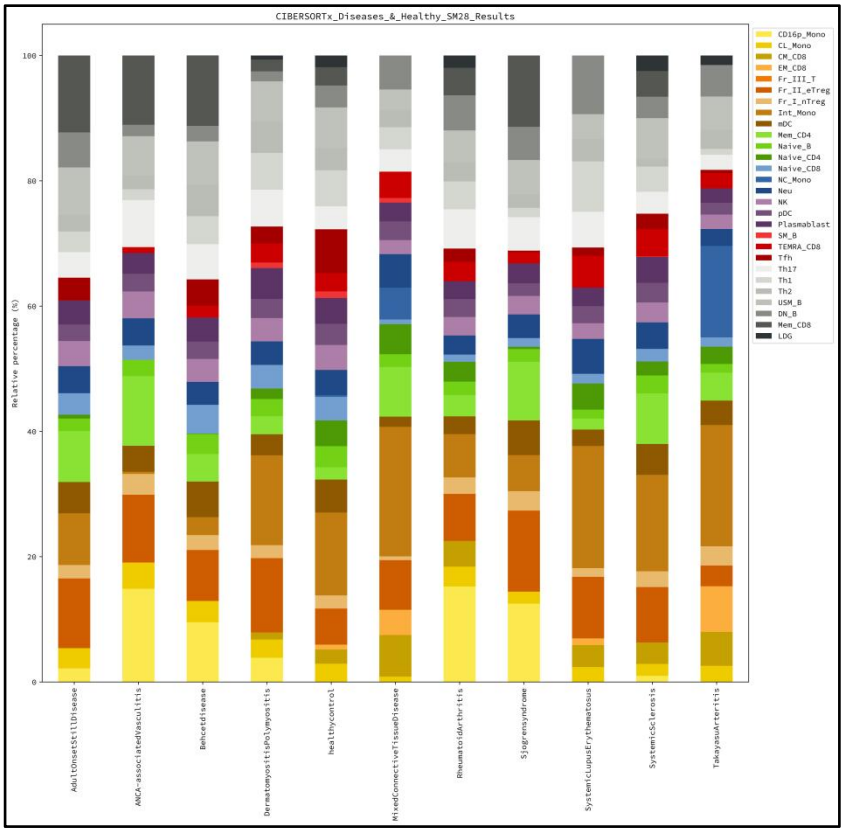


Figure 52 Deconvoluted cell proportions of patients belonging to healthy control using custom matrix SM28

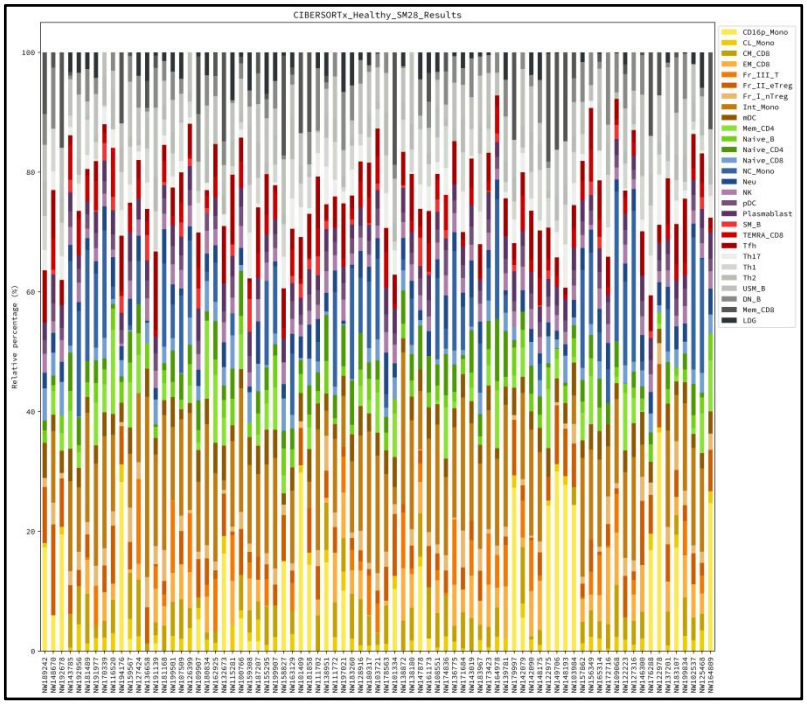


Figure 53 Deconvoluted cell proportions of patients belonging to mixed connective tissue disease using custom matrix SM28

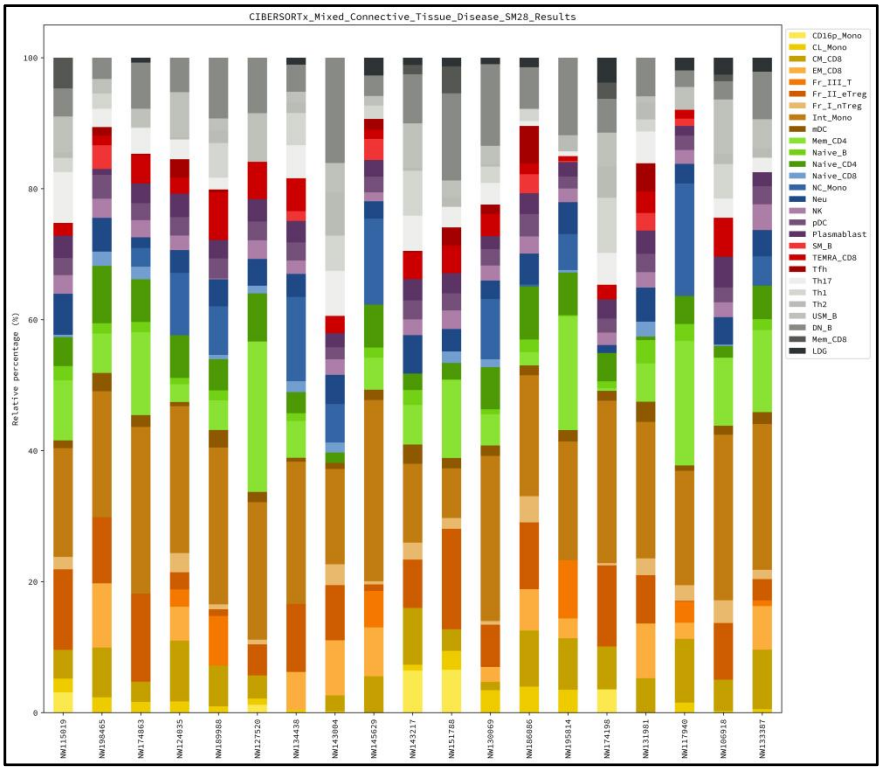


Figure 54 Deconvoluted cell proportions of patients belonging to rheumatoid arthritis using custom matrix SM28

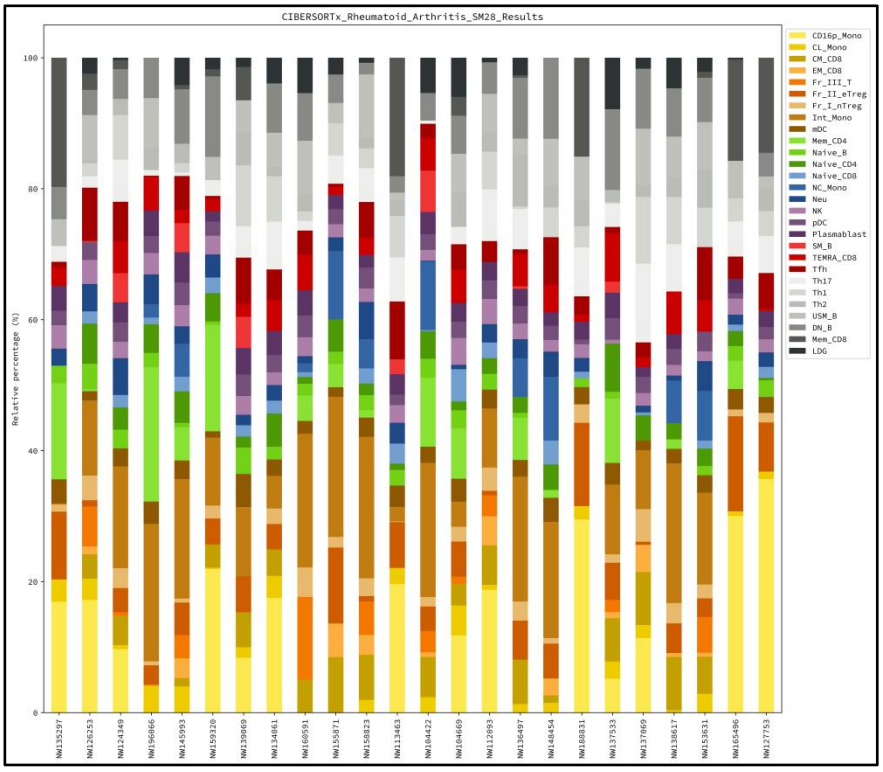


Figure 55 Deconvoluted cell proportions of patients belonging to sjogren's syndrome using custom matrix SM28

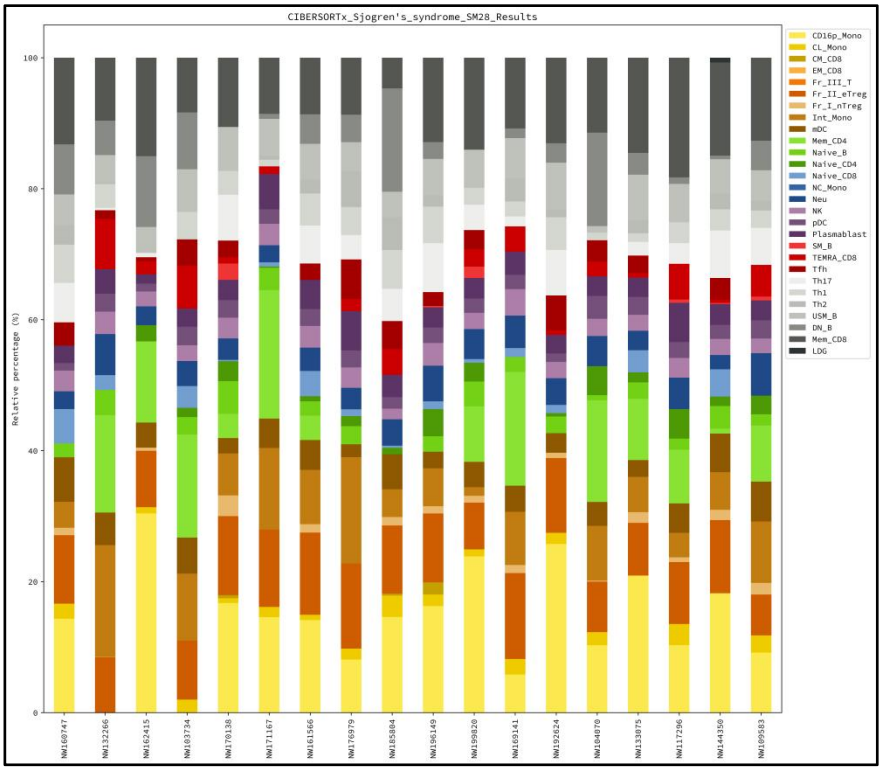


Figure 56 Deconvoluted cell proportions of patients belonging to systemic lupus erythematosus using custom matrix SM28

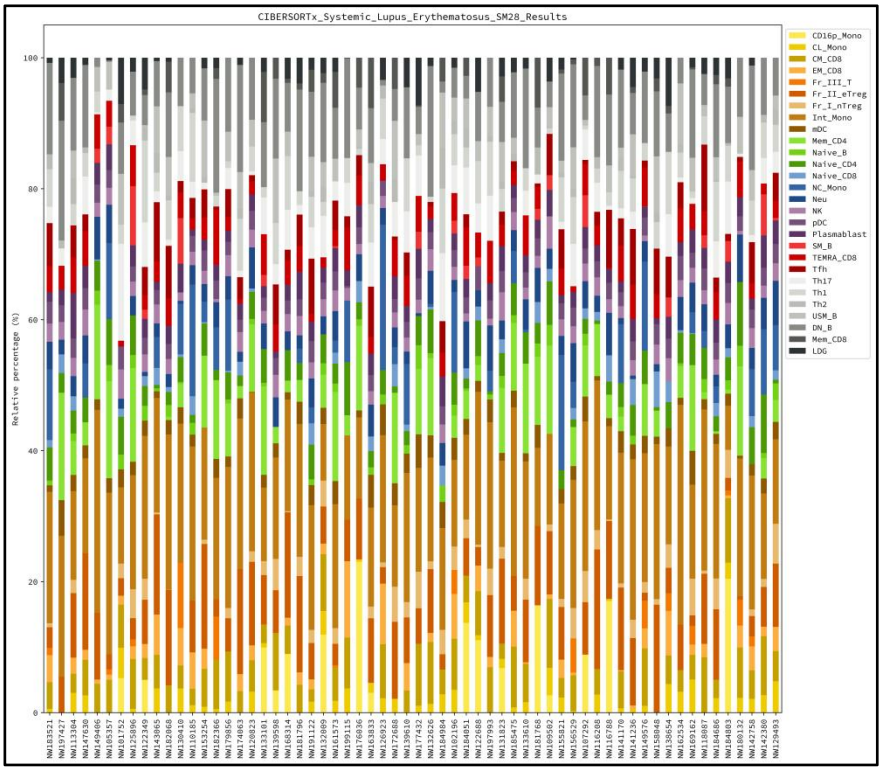


Figure 57 Deconvoluted cell proportions of patients belonging to systemic sclerosis using custom matrix SM28

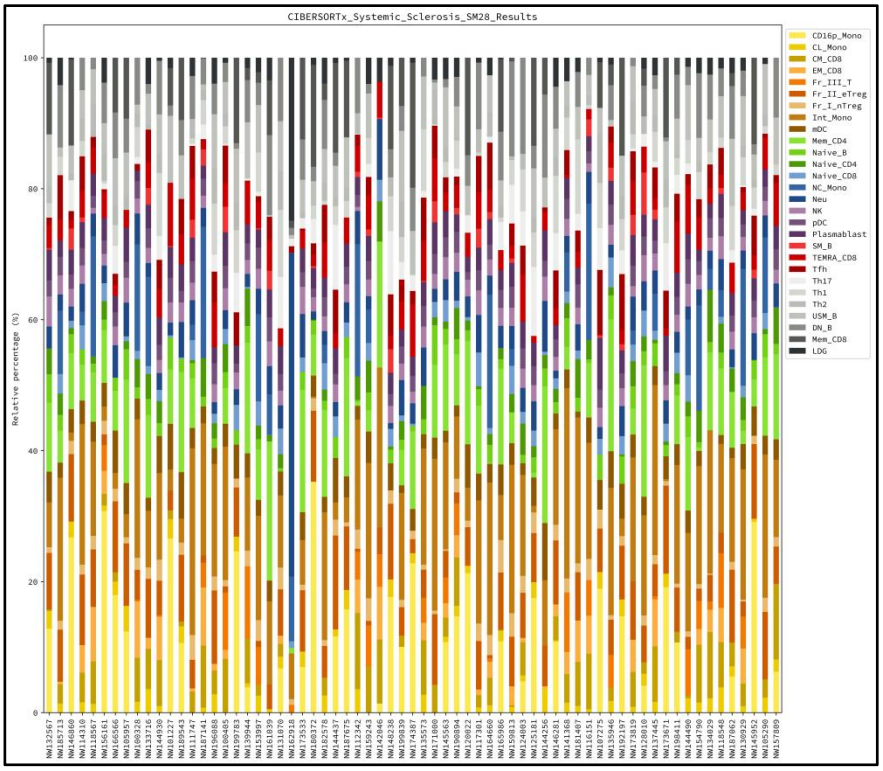


Figure 58 Deconvoluted cell proportions of patients belonging to takayasu's arteritis using custom matrix SM28

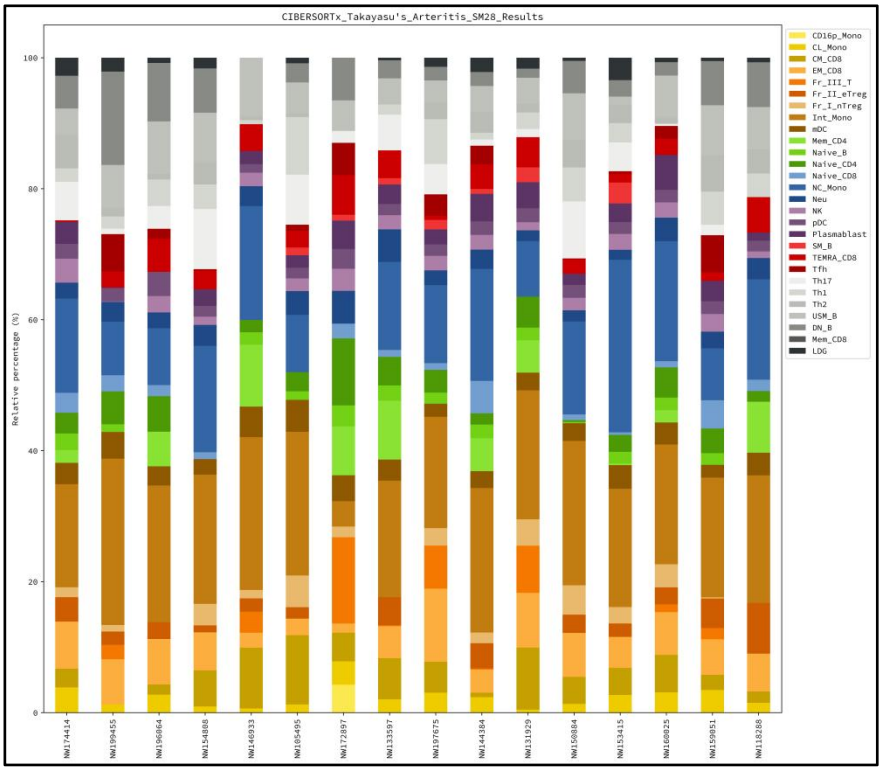


Figure 59 Heatmap representation of confusion matrix on multiclass classification using hyper-parameter tuned support vector machine

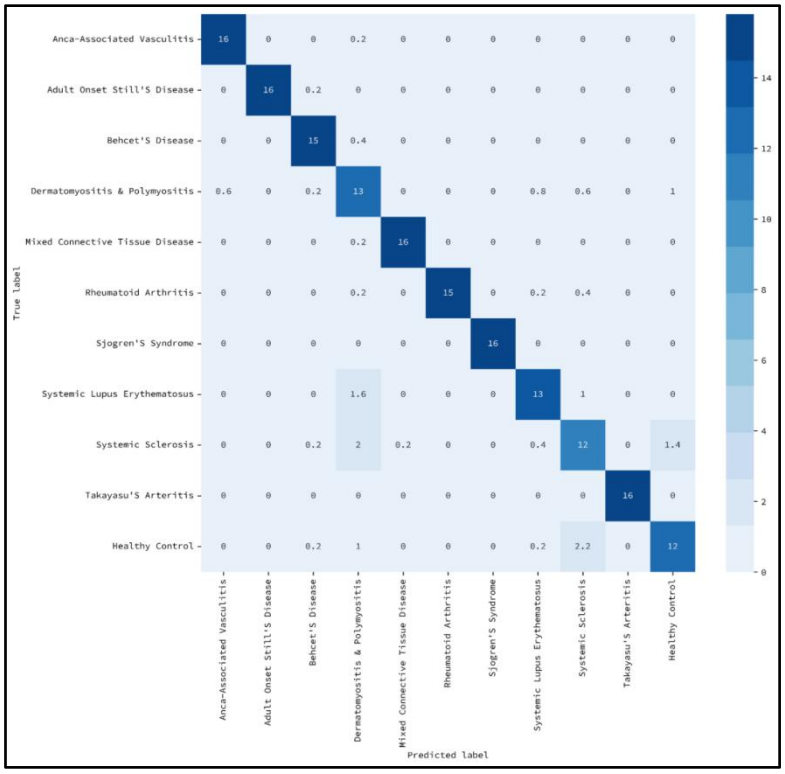


Figure 60 Heatmap representation of confusion matrix on multiclass classification using CNN architecture

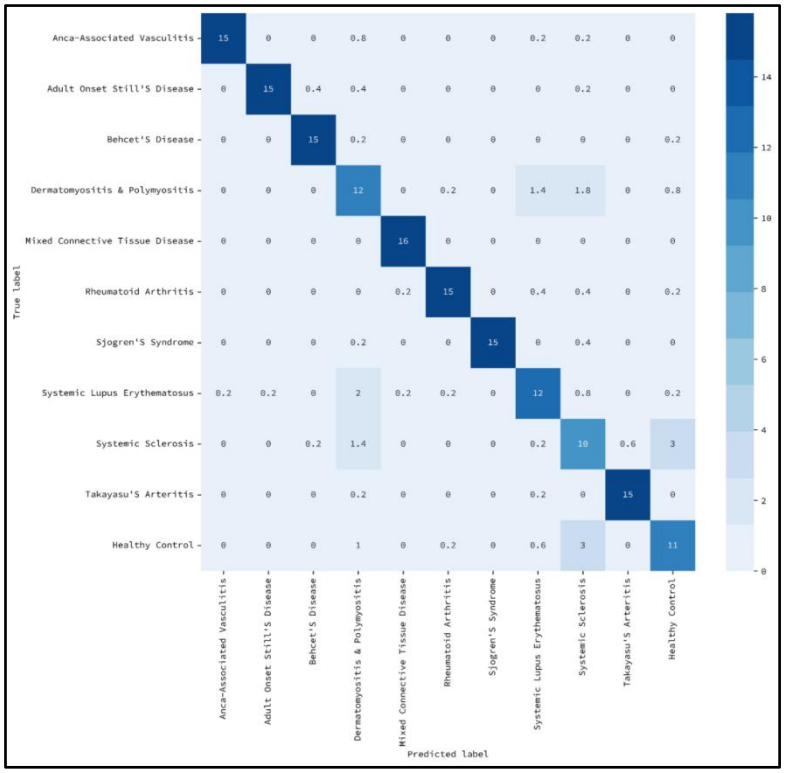


Table 1 The table contains cross validated metrics using various models to perform multi-class classification.

Model type	Precision	Recall	F1 Score	MCC	Accuracy	Kappa Score	F-beta Score
Support Vector Machine	0.9267	0.9231	0.9218	0.9159	0.9229	0.9152	0.9239
Gaussian Naive Bayes	0.8300	0.8054	0.8027	0.7892	0.8055	0.7861	0.8133
Multi Layer Perceptron	0.9134	0.9067	0.9047	0.8985	0.9068	0.8974	0.9085
K-Nearest Neighbours	0.8331	0.8248	0.8048	0.8117	0.8251	0.8076	0.8105
Convolution Neural Network	0.8770	0.8684	0.8692	0.8566	0.8688	0.8557	0.8730

Table 2 The table contains cross validated metrics using KNN model to perform binary classification; disease classes vs healthy control data

Disease Class	MCC	Precision	Recall	F1 Score	Accuracy	Kappa Score	FBeta Score
Dermatomyositis & Polymyositis	0.6765	0.8487	0.8286	0.8261	0.8292	0.6574	0.8361
Behcet's disease	0.8755	0.9446	0.9312	0.9295	0.9308	0.8617	0.9363
Adult Onset Still's Disease	0.8755	0.9446	0.9312	0.9295	0.9308	0.8617	0.9363
ANCA-associated Vasculitis	0.9521	0.9771	0.975	0.9745	0.9746	0.9492	0.9757
Mixed Connective Tissue Disease	0.9882	0.9944	0.9938	0.9937	0.9938	0.9875	0.9941
Rheumatoid Arthritis	0.9401	0.9722	0.9679	0.9681	0.9683	0.9365	0.9701
Systemic Lupus Erythematosus	0.9382	0.9694	0.9688	0.9687	0.9688	0.9375	0.9691
Systemic Sclerosis	0.5571	0.7874	0.7705	0.7671	0.7725	0.5432	0.7757
Infected	0.8564	0.9337	0.9229	0.9223	0.9228	0.8457	0.9276
Sjogren's syndrome	0.9764	0.9889	0.9875	0.9875	0.9875	0.975	0.9881

Takayasu's Arteritis	0.9882	0.9944	0.9938	0.9937	0.9938	0.9875	0.9941

Table 3 The table contains cross validated metrics using SVM model to perform binary classification; disease classes vs healthy control data

Disease Class	MCC	Precision	Recall	F1 Score	Accuracy	Kappa Score	FBeta Score
Dermatomyositis & Polymyositis	0.8285	0.917	0.9116	0.9113	0.9117	0.8232	0.914
Behcet's disease	0.9278	0.9653	0.9625	0.9624	0.9625	0.925	0.9638
Adult Onset Still's Disease	0.9875	0.9938	0.9938	0.9933	0.9933	0.9867	0.9935
ANCA-associated Vasculitis	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Mixed Connective Tissue Disease	0.9757	0.9882	0.9875	0.9871	0.9871	0.9742	0.9875
Rheumatoid Arthritis	0.9882	0.9944	0.9938	0.9937	0.9938	0.9875	0.9941
Systemic Lupus Erythematosus	0.8901	0.9463	0.9438	0.9432	0.9433	0.8867	0.9446
Systemic Sclerosis	0.6854	0.851	0.8348	0.8341	0.8362	0.6717	0.8419
Infected	0.9626	0.9817	0.9808	0.9807	0.9807	0.9615	0.9812
Sjogren's syndrome	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Takayasu's Arteritis	0.9764	0.9889	0.9875	0.9875	0.9875	0.975	0.9881

Table 4 The table contains cross validated metrics using Gaussian Naive Bayes model to perform binary classification; disease class vs healthy control data

Disease Class	MCC	Precision	Recall	F1 Score	Accuracy	Kappa Score	FBeta Score
Dermatomyositis & Polymyositis	0.7452	0.8784	0.867	0.8663	0.8675	0.7345	0.8719
Behcet's disease	0.9302	0.9678	0.9625	0.9622	0.9625	0.925	0.9648
Adult Onset Still's Disease	0.9646	0.9833	0.9812	0.9812	0.9812	0.9625	0.9822
ANCA-associated Vasculitis	0.9282	0.9667	0.9616	0.9618	0.9621	0.924	0.9642
Mixed Connective Tissue Disease	0.9757	0.9882	0.9875	0.9871	0.9871	0.9742	0.9875
Rheumatoid Arthritis	0.9146	0.9592	0.9554	0.9547	0.955	0.9099	0.9568
Systemic Lupus Erythematosus	0.8806	0.9433	0.9375	0.936	0.9367	0.8735	0.9391

Systemic Sclerosis	0.5319	0.7818	0.7536	0.7441	0.7533	0.5061	0.7571
Infected	0.9257	0.9642	0.9615	0.9614	0.9615	0.9231	0.9627
Sjogren's syndrome	0.9757	0.9882	0.9875	0.9871	0.9871	0.9742	0.9875
Takayasu's Arteritis	0.9639	0.9826	0.9812	0.9808	0.9808	0.9617	0.9816

Chapter 5: Conclusion and Future Work

The model building and tuning performed in our project shows good results towards the predictability and classification of immune mediated diseases with the help of gene expression data. The models can be used further to extract and analyze the genes which are contributing towards the outcome in decision making of the algorithm by making use of Explainable Artificial Intelligence - The term explainable AI refers to the ability to define an AI model, its projected impact, and any biases. It contributes to the definition of model correctness, fairness, transparency.

By taking it a step further, Gene Set Enrichment Analysis (GSEA) - a technique that employs statistical methodologies to discover gene groupings which are highly enriched or deficient, can also be applied on the genes to help analyze the models beyond the usual metrics used for testing AI techniques, in turn helping us discover the role of genes in various biological pathways. GSEA can be performed on a set of genes by using popular tools like DAVID[11] and Enrichr[12].

References

[1] M. Ota, Y. Nagafuchi, H. Hatano, K. Ishigaki, C. Terao, Y. Takeshima, H. Yanaoka, S. Kobayashi, M. Okubo, H. Shirai, Y. Sugimori, J. Maeda, M. Nakano, S. Yamada, R. Yoshida, H. Tsuchiya, Y. Tsuchida, S. Akizuki, H. Yoshifuji, K. Ohmura, T. Mimori, K. Yoshida, D. Kurosaka, M. Okada, K. Setoguchi, H. Kaneko, N. Ban, N. Yabuki, K. Matsuki, H. Mutoh, S. Oyama, M. Okazaki, H. Tsunoda, Y. Iwasaki, S. Sumitomo, H. Shoda, Y. Kochi, Y. Okada, K. Yamamoto, T. Okamura, and K. Fujio, "Dynamic landscape of immune cell-specific gene regulation in immune-mediated diseases," Cell, vol. 184, no. 11, pp. 3006–3021.e17, May 2021. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0092867421004293

[2] H. Yu, Y. Nagafuchi, and K. Fujio, "Clinical and Immunological Biomarkers for Systemic Lupus Erythematosus," Biomolecules, vol. 11, no. 7, p. 928, 2021, publisher: Multidisciplinary Digital Publishing Institute.

[3] L. Hirahara, K. Takase-Minegishi, Y. Kirino, Y. Iizuka-Iribe, Y. Soejima, R. Yoshimi, and H. Nakajima, "The Roles of Monocytes and Macrophages in Behçet's Disease With Focus on M1 and M2 Polarization," Frontiers in immunology, p. 1036, 2022, publisher: Frontiers.

[4] Y. Takeshima, Y. Iwasaki, M. Nakano, Y. Narushima, M. Ota, Y. Nagafuchi, S. Sumitomo, T. Okamura, K. Elkon, and K. Ishigaki, "Immune cell multiomics analysis reveals contribution of oxidative phosphorylation to B-cell functions and organ damage of lupus," Annals of the Rheumatic Diseases, 2022, publisher: BMJ Publishing Group Ltd.

[5] M. Ota and K. Fujio, "Multi-omics approach to precision medicine for immune-mediated diseases," Inflammation and regeneration, vol. 41, no. 1, pp. 1–6, 2021, publisher: Springer.

[6] A. M. Newman, C. L. Liu, M. R. Green, A. J. Gentles, W. Feng, Y Xu, C. D. Hoang, M. Diehn, and A. A. Alizadeh, "Robust enumeration of cell subsets from tissue expression profiles," Nature Methods, vol. 12, no. 5, pp. 453–457, May 2015. [Online]. Available: http://www.nature.com/articles/nmeth.3337

[7] M. I. Love, W. Huber, and S. Anders, "Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2," Genome Biology, vol. 15, no. 12, p. 550, Dec. 2014. [Online]. Available: https://doi.org/10.1186/s13059-014-0550-8

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, Jun. 2002, arXiv: 1106.1813. [Online]. Available: http://arxiv.org/abs/1106.1813

[9] D. Chicco, N. Tötsch, and G. Jurman, "The Matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation," BioData Mining, vol. 14, no. 1, p. 13, Feb. 2021. [Online]. Available: https://doi.org/10.1186/s13040-021-00244-z

[10] M. Mostavi, Y.-C. Chiu, Y. Huang, and Y. Chen, "Convolutional neural network models for cancer type prediction based on gene expression," BMC Medical Genomics, vol. 13, no. 5, p. 44, Apr. 2020. [Online]. Available: https://doi.org/10.1186/s12920-020-0677-2

[11] G. Dennis, B. T. Sherman, D. A. Hosack, J. Yang, W. Gao, H. C. Lane, and R. A. Lempicki, "DAVID: Database for Annotation, Visualization, and Integrated Discovery," Genome Biology, vol. 4, no. 9, p. R60, Aug. 2003. [Online]. Available: https://doi.org/10.1186/gb-2003-4-9-r60

[12] E. Y. Chen, C. M. Tan, Y. Kou, Q. Duan, Z. Wang, G. V. Meirelles, N. R. Clark, and A. Ma'ayan, "Enrichr: interactive and collaborative HTML5 gene list enrichment analysis tool," BMC Bioinformatics, vol. 14, no. 1, p.128, Apr. 2013. [Online]. Available: https://doi.org/10.1186/1471-2105-14-128