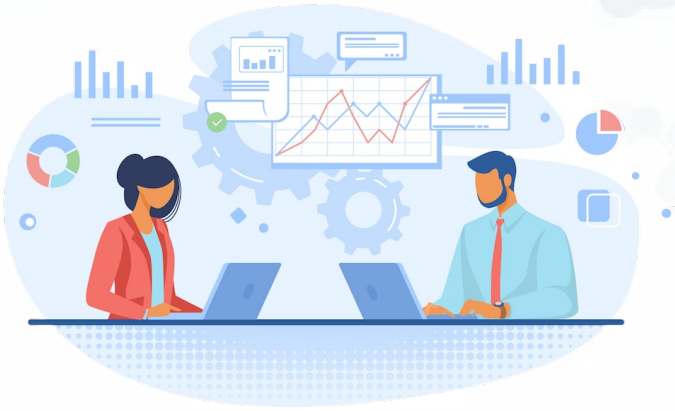


Computational Molecular Medicine
Spring 2023

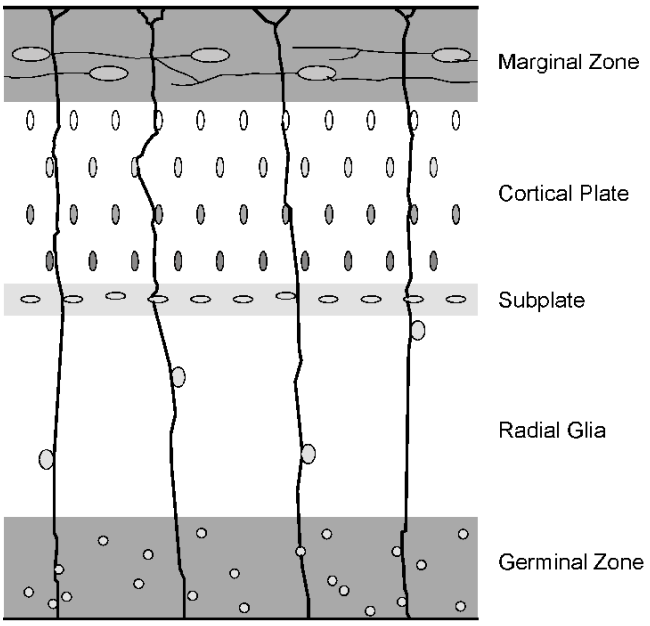
A Comparison of Tree-Based Methods and Generative Modeling for Analyzing Single-Cell RNA Sequencing Data

Debsurya De
dde4@jh.edu, Ph.D. in Applied Mathematics and Statistics
Long Yuan
lyuan13@jhmi.edu, Ph.D. in Immunology
Joseph Lee
jlee733@jhu.edu, M.S.E. in Biomedical Engineering

Institution: The Johns Hopkins University, Baltimore, Maryland
Date: April 22, 2023

1 Introduction

The human cortex is comprised of billions of cells, believed to consist of hundreds or thousands of distinct cell types, each with its own unique functions. In recent years, groundbreaking work using single-cell transcriptomics in mice has demonstrated its ability to elucidate the complexity and heterogeneity of cell types in the brain. With access to high-quality tissue and advances in single-cell transcriptomic technologies, it is now possible to comprehensively and impartially catalog the cell-type diversity of the human cortex. Despite significant progress in characterizing early cortical development, the molecular mechanisms underlying the generation, differentiation, and development of diverse cell types remain largely unknown. The use of molecular taxonomies of cortical cell types from developing human brains offers an opportunity to understand the mechanisms of neurogenesis and how the remarkable cellular diversity found in the human cortex is achieved. While several studies have taken the first step in this direction by analyzing several hundred or a few thousand cells from developing human brains, technological advancements now allow for the analysis of an order of magnitude more cells to provide a more in-depth picture of human cortical development and its perturbation in disease.

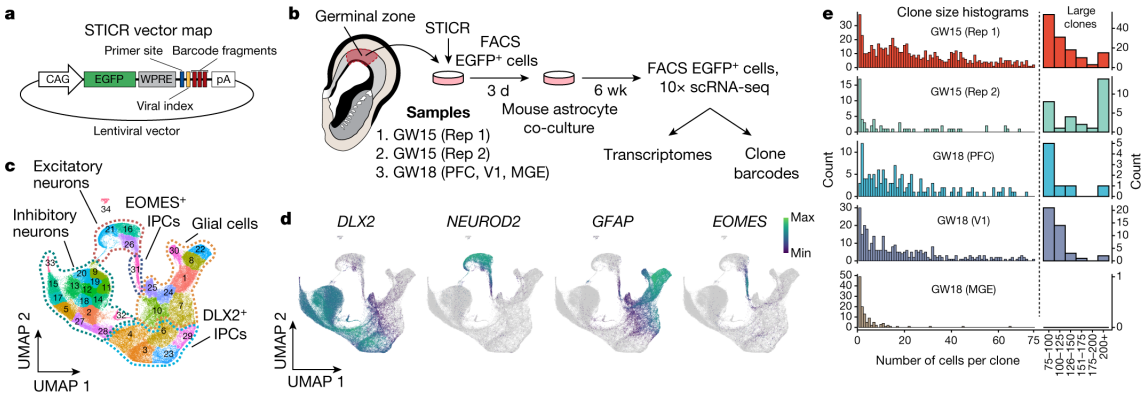


In the context of mid-gestation, dorsal pallial neurogenesis is responsible for producing the majority of excitatory neurons in the neocortex. The mature neocortex in mammals is composed of six layers of post-mitotic neurons, with each new layer being produced sequentially. Neurons are generated from radial glia and intermediate progenitor cells through both symmetric and asymmetric cell division, resulting in an "inside-out" arrangement of neurons by birthdate, with deep layers being born first and superficial layers being born last. The primate lineage has greatly expanded the number and diversity of neural stem and progenitor cells in the cortex, particularly through the appearance of the outer subventricular zone (OSVZ), which underlies the development of the supragranular layers of the cortex. This includes a greater proliferative capacity of neural stem cells and the expansion of outer radial

glia (oRG or basal radial glia, bRG), which are rare in rodents but numerous in primates and especially in humans. As newborn neurons arise contemporaneously from progenitor cells in the germinal zones and migrate radially along processes of the radial glia, outward toward the surface of the cortex to the cortical plate (CP), they come to rest together at their characteristic position in the developing cortex. Neurons arising at the same time and place, and therefore from precursors with common molecular features, share unique morphological and functional characteristics. Our interest lies in the general process by which neural precursor cells yield post-mitotic neurons, as well as the nuanced dynamics by which specific properties are imparted to neurons that are born together and come to rest in the same cellular microenvironment, i.e., cortical region and layer.

The context at hand pertains to the intricate biological processes of dorsal pallial neurogenesis during mid-gestation, a critical stage in which the vast majority of excitatory neurons in the neocortex come into being. In mammals, the mature neocortex comprises six layers of post-mitotic neurons that are created sequentially from radial glia and intermediate progenitor cells through symmetric and asymmetric cell division. The result of this systematic neurogenesis is an inside-out configuration of neurons according to their birthdate, with the deeper layers being formed first, and the outermost layers emerging last.

During the course of primate evolution, the number and diversity of neural stem and progenitor cells in the cortex have undergone a significant expansion. Specifically, the appearance of the outer subventricular zone (OSVZ) seems to be integral to the development of the supragranular layers of the cortex, which consist of neurons in layers II and III. These layers are responsible for extensive cortico-cortico connections, facilitating higher cognition. The prevalence of outer radial glia (oRG), also referred to as basal radial glia (bRG), is strikingly abundant in primates, and particularly so in humans, but they are scarce in rodents.



Noteworthy is the fact that neurons born simultaneously and in the same location, and hence from precursors sharing common molecular characteristics, display unique morphological and functional traits. Consequently, understanding the intricate dynamics through which specific properties are conferred onto neurons is a crucial objective in comprehending the process of neurogenesis in the developing cortex. Analyzing gene expression through classification presents a promising avenue for gaining insight into the molecular basis of neurogenesis, informing our knowledge of human embryonic neurogenesis, and providing greater understanding of the mechanisms involved in neurodevelopmental disorders.

1.1 About the Dataset

Studying how the human brain develops before birth is really complicated and difficult because it involves both technical and ethical issues. Researchers have to collect tiny amounts of tissue from developing human embryos to study the cells, which can be risky and raise ethical concerns. This makes it hard for scientists to use data-driven methods to understand how the brain develops.

However, there is a dataset called the Geschwind dataset, which provides a rare opportunity for scientists to study the brain's development using computers. This dataset includes information about 15,060 individual cells and 35,543 different molecules. This is a huge amount of data that researchers can use to learn more about how the brain develops.

The main goal of using the Geschwind dataset is to develop computer programs that can tell the difference between cells that will become part of the brain and those that won't. This is a crucial step in understanding how the brain develops and could help researchers learn more about conditions that affect the brain.

1.2 Methods and Directions

The $p >> n$ problem, where the number of variables is much greater than the number of observations, is a significant challenge in modern data science. This problem arises in a wide range of applications, such as genomics, neuroscience, finance, and image analysis, where the availability of vast amounts of data poses a significant challenge to traditional statistical methods.

One of the primary issues with this problem is the difficulty in selecting relevant variables that have a significant impact on the response variable of interest. In this context, variable selection plays a crucial role in building accurate predictive models. The selection process aims to identify the most critical variables that have a strong influence on the response while removing less relevant ones that add noise to the model.

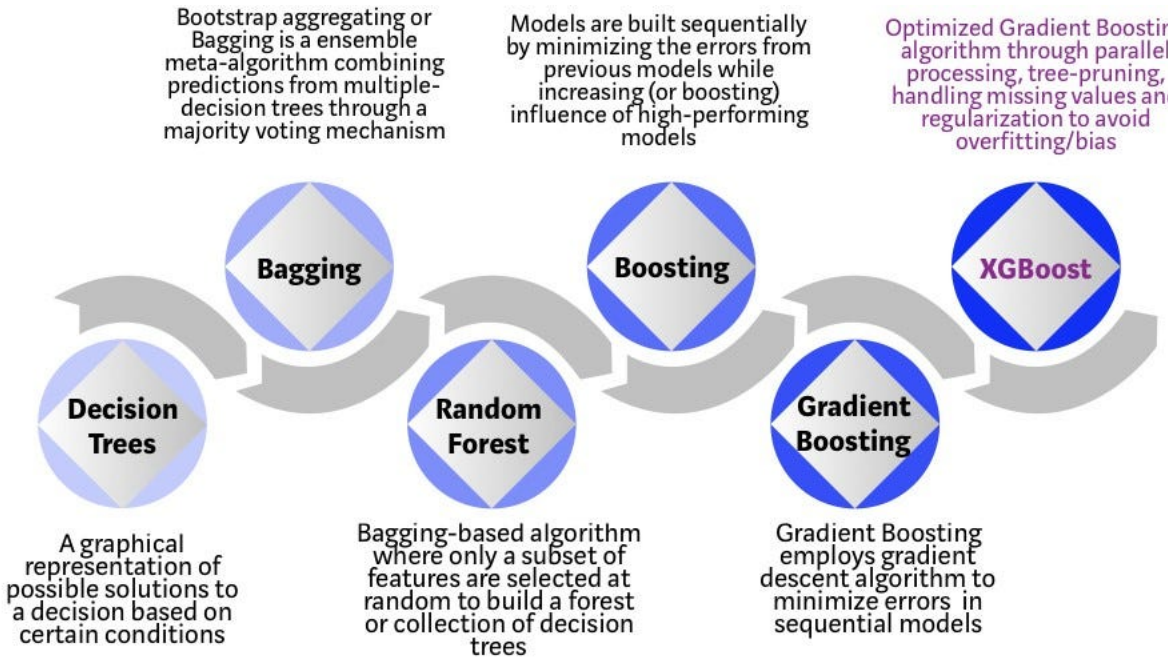
To address this challenge, researchers have developed various techniques for variable selection in machine learning. One popular approach is regularization, which involves adding a penalty term to the loss function of the model to shrink the coefficients of less important variables to zero. This process helps to identify the most significant variables and eliminates the less relevant ones, thereby reducing the complexity of the model and improving its predictive accuracy.

Logistic LASSO is one such regularization technique that is commonly used for variable selection in binary classification problems. It is a form of linear regression that adds an L_1 regularization term to the loss function, thereby forcing some of the coefficients to be exactly zero. This process results in a sparse model that contains only the most critical variables.

Once the variables have been selected using Logistic LASSO, the next step is to classify each cell as belonging to the germinal zone or cortical plate. For this task, we have chosen to use a combination of Logistic Regression and Tree-based methods, namely Classification Trees, Random Forest, and Gradient Boosting.

Logistic Regression is a popular method for binary classification that models the relationship between the predictor variables and the binary response variable. It is a simple yet powerful algorithm that is widely used in various applications, including medical diagnosis, fraud detection, and customer segmentation.

On the other hand, Tree-based methods are a class of algorithms that use decision trees to classify data. These algorithms recursively partition the data into subsets based on the values of the predictor variables until the subsets are sufficiently homogeneous. Random Forest and Gradient Boosting, on the other hand, are ensemble methods that combine multiple trees to improve the accuracy and robustness of the classification model.



Variational autoencoder (VAE) was utilized for generative modeling. A classifier was trained on encoded data obtained from the VAE's latent space to perform the classification task. Performance evaluation was done using standard metrics. Although the VAE is not explicitly trained for classification, its latent space can still be useful. Fine-tuning is possible by adding a classification layer on top of the VAE decoder and training the entire model end-to-end.

In section 3, we will compare and analyze the results obtained from these methods, drawing insights and observations that shed light on the relative performance of these methods in our specific context. We will evaluate the sensitivity, specificity and the ROC curve of the models, as well as their ability to predict correctly. We will perform a comparison of the performance of these models.

In section 4, we will engage in a comprehensive discussion of the results, drawing insights from our analysis and providing possible explanations for the observed trends. We will examine the strengths and weaknesses of each model and suggest possible areas for improvement. We will also discuss the implications of our results for the wider field of machine learning and its applications.

Finally, in section 5, we will provide the relevant codes that we used in our experiments. These codes are provided to facilitate the ease of replication and further exploration by interested individuals.

2 A Review of Methods

2.1 Variable Selection

Logistic LASSO is a widely used regularization technique in machine learning and statistics for variable selection in binary classification problems. This method is a form of linear regression that incorporates an L_1 regularization term into the loss function, thereby forcing some of the coefficients to be exactly zero. This process results in a sparse model that contains only the most critical variables.

Let us begin with a brief overview of the logistic regression model, which is the basis for Logistic LASSO.

2.1.1 Logistic Regression

Logistic Regression is a statistical method used for binary classification, where the response variable takes two possible values, typically 0 or 1. In this method, the probability of an observation belonging to the positive class is modeled as a function of the predictor variables using the logistic function. The logistic function, also known as the sigmoid function, is defined as follows:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

where z is a linear combination of the predictor variables and their corresponding coefficients. Mathematically, we can express the logistic regression model as:

$$\mathbb{P}(y = 1 \mid \boldsymbol{x}) = \sigma(\beta^T \boldsymbol{x})$$

In logistic regression, the model parameters are estimated by maximizing the likelihood function, which is the product of the probabilities of the observations belonging to their respective classes. This process involves minimizing the negative log-likelihood function, which is given by:

$$\mathcal{L}(\beta) = -\sum_{i=1}^{n} [y_i \log(\sigma(\beta^T \boldsymbol{x}_i)) + (1 - y_i) \log(1 - \sigma(\beta^T \boldsymbol{x}_i))]$$

2.1.2 Logistic LASSO

Logistic LASSO, or L_1-penalized logistic regression, is a popular machine learning technique used for variable selection in binary classification problems.

Variable selection is a crucial step in building predictive models in data science and machine learning. It involves selecting the most critical variables that have the most influence on the response variable and discarding the less important ones. Variable selection is particularly essential when the number of predictor variables is much larger than the number of observations, as is often the case in high-dimensional data.

Traditional methods for variable selection, such as stepwise regression and backward elimination, have several limitations. These methods assume that all variables are equally important and that the coefficients of the less important variables are small or zero. In reality, some variables may have a more significant impact on the response variable, and the coefficients of the less important variables may be large due to multicollinearity.

Logistic LASSO is a regularization technique that addresses these limitations by adding a penalty term to the negative log-likelihood function of logistic regression. This penalty term encourages the coefficients of less important variables to be exactly zero, resulting in a sparse model that contains only the most critical variables.

Logistic LASSO is a variable selection technique that can identify the most critical variables that have the most influence on the response variable. It achieves this by adding an L_1 penalty term to the negative log-likelihood function of logistic regression. The objective function for Logistic LASSO is as follows:

$$\underset{\beta}{\operatorname{argmin}} \left\{ -\frac{1}{n} \sum_{i=1}^{n} [y_i \log(\sigma(\beta^T \boldsymbol{x}_i)) + (1 - y_i) \log(1 - \sigma(\beta^T \boldsymbol{x}_i))] + \lambda \sum_{i=j}^{p} |\beta_j| \right\}$$

The L_1 penalty term $\lambda \sum_{j=1}^{p} |\beta_j|$ encourages the coefficients of less important variables to be exactly zero, resulting in a sparse model that contains only the most critical variables.

The tuning parameter λ controls the strength of the penalty term and determines the number of variables that are included in the model. As λ increases, more coefficients become zero, resulting in a simpler model with fewer variables.

2.2 Tree-Based Methods

2.2.1 Classsification Trees

Classification trees are a type of decision tree that are commonly used in machine learning for solving classification problems. Decision trees are models that use a tree-like structure to represent a decision-making process. Each node in the tree represents a test on one of the input features, and each branch represents the possible outcomes of that test. The leaves of the tree represent the predicted class labels for the input data. In classification trees, the predicted class labels are discrete, and the goal is to partition the feature space into regions that are associated with different class labels.

Let $\mathcal{D}$ be a dataset consisting of n samples, each with m features, and let $\mathcal{Y} = 1, 2, \ldots, K$ be the set of possible class labels. The goal of classification tree learning is to partition the feature space into a set of disjoint regions, such that each region is associated with a unique

class label. The learning algorithm builds the tree recursively, starting from the root node, by selecting a feature to split on at each node. The splitting feature is chosen based on some impurity measure that quantifies the purity of the data at that node. The most commonly used impurity measures are the Gini impurity and the cross-entropy.

The Gini impurity is defined as follows:

$$\text{Gini}(p) = \sum_{i=1}^{K} p_i(1 - p_i) \tag{1}$$

where p_i is the proportion of samples in the node that belong to class i, and K is the total number of classes. The Gini impurity ranges from 0 to 0.5, where a value of 0 indicates that all samples in the node belong to the same class, and a value of 0.5 indicates that the samples are evenly distributed among all classes.

The cross-entropy is defined as follows:

$$\text{CE}(p) = -\sum_{i=1}^{K} p_i \log_2 p_i \tag{2}$$

where p_i is the same as in the Gini impurity. The cross-entropy ranges from 0 to $\log_2(K)$, where a value of 0 indicates perfect purity and a value of $\log_2(K)$ indicates maximum impurity.

Once the impurity measure is selected, the algorithm searches over all possible feature splits and selects the split that results in the largest decrease in impurity. The decrease in impurity is calculated as the difference between the impurity of the parent node and the weighted average of the impurities of the child nodes. The weight for each child node is proportional to the number of samples that fall into that node. Once a split is selected, the algorithm recursively applies the same process to each child node until a stopping criterion is met, such as a maximum tree depth or a minimum number of samples in a leaf node.

To classify a new data point, we start at the root node and follow the decision rules defined by the tree until we reach a leaf node. The predicted class label is then the class associated with the leaf node. Classification trees have several advantages, including interpretability, robustness to outliers, and the ability to handle non-linear decision boundaries. However, they can suffer from overfitting and instability due to their sensitivity to small changes in the data.

2.2.2 Random Forest

Random Forest classification is an ensemble learning method that combines multiple decision trees to improve the accuracy and robustness of classification. The idea behind Random Forest is to create a large number of decision trees, each trained on a randomly selected subset of the predictor variables and a bootstrap sample of the training data.

Let $X_1, X_2, ..., X_p$ be the p predictor variables and Y be the binary response variable that we want to predict using the Random Forest method. The algorithm first randomly selects

a subset of m predictor variables from the p total predictors, where m is usually set to the square root of p for classification problems. Then, it draws a bootstrap sample of size n from the training data, where n is the total number of observations in the training set.

For each of the B decision trees that will make up the Random Forest, the algorithm constructs the tree using the same process as in the Classification Tree algorithm. The tree is grown using the binary split algorithm, where the best split at each internal node is chosen based on the Gini index. However, at each split, only the randomly selected subset of m predictor variables are considered for splitting.

After all the trees have been constructed, a new observation is classified by passing it through each of the B decision trees and taking the majority vote of the predicted class across all trees.

The Random Forest method has several advantages over the Classification Tree method. By randomly selecting subsets of predictor variables and bootstrap samples from the training data, the algorithm is able to reduce overfitting and improve the generalization performance of the model. Additionally, the ensemble of decision trees can capture more complex relationships between the predictors and the response than a single decision tree.

The Random Forest method also provides a measure of variable importance, which indicates how much each predictor variable contributes to the accuracy of the model. Variable importance is calculated by permuting the values of each predictor variable and measuring the decrease in accuracy of the model, which provides a measure of the variable's importance in predicting the response.

2.2.3 Gradient Boosting

Gradient Boosting classification is a machine learning technique that uses an ensemble of weak classifiers, such as decision trees, to create a strong classifier. It works by iteratively adding new decision trees to the ensemble, where each new tree is trained on the residuals of the previous tree.

Let $X_1, X_2, ..., X_p$ be the p predictor variables and Y be the binary response variable that we want to predict using the Gradient Boosting method. The algorithm starts with an initial model, such as a single decision tree or a constant value, and then iteratively improves the model by adding new decision trees to the ensemble.

At each iteration, the algorithm calculates the negative gradient of the loss function with respect to the current model's predictions, which serves as the target variable for the new decision tree. The new tree is trained to predict the negative gradient, which is added to the current model's predictions to obtain a new, improved model.

The learning rate parameter controls the contribution of each new tree to the final ensemble, with lower values resulting in a more conservative update of the model. The number of iterations and the depth of the decision trees are other important hyperparameters that can affect the performance of the model.

The Gradient Boosting algorithm is particularly well-suited for handling high-dimensional

datasets with complex relationships between the predictors and the response. It can also handle missing data and categorical variables by using appropriate splitting criteria.

One disadvantage of the Gradient Boosting algorithm is that it can be prone to overfitting, particularly if the hyperparameters are not tuned correctly. Regularization techniques, such as shrinkage, subsampling, and early stopping, can help to reduce overfitting and improve the generalization performance of the model.

2.3 Variational Autoencoder

A Variational Autoencoder (VAE) is a type of generative neural network that can learn the underlying distribution of a set of data and generate new data points from that distribution. It is composed of an encoder network that maps input data to a low-dimensional latent space, and a decoder network that maps the latent variables back to the original input space.

The objective of a VAE is to maximize the evidence lower bound (ELBO) loss, which is a lower bound on the log-likelihood of the data. The ELBO is defined as:

$$\text{ELBO} = \mathbb{E}q_\phi(z|x)[\log p_\theta(x|z)] - D_{\text{KL}}(q_\phi(z|x)||p(z))$$

where ϕ and θ are the parameters of the encoder and decoder networks respectively, x is the input data, z is the latent variable, $p(z)$ is the prior distribution over the latent space (usually a standard Gaussian), and $q_\phi(z|x)$ is the variational distribution that approximates the true posterior distribution over the latent space given the input data x.

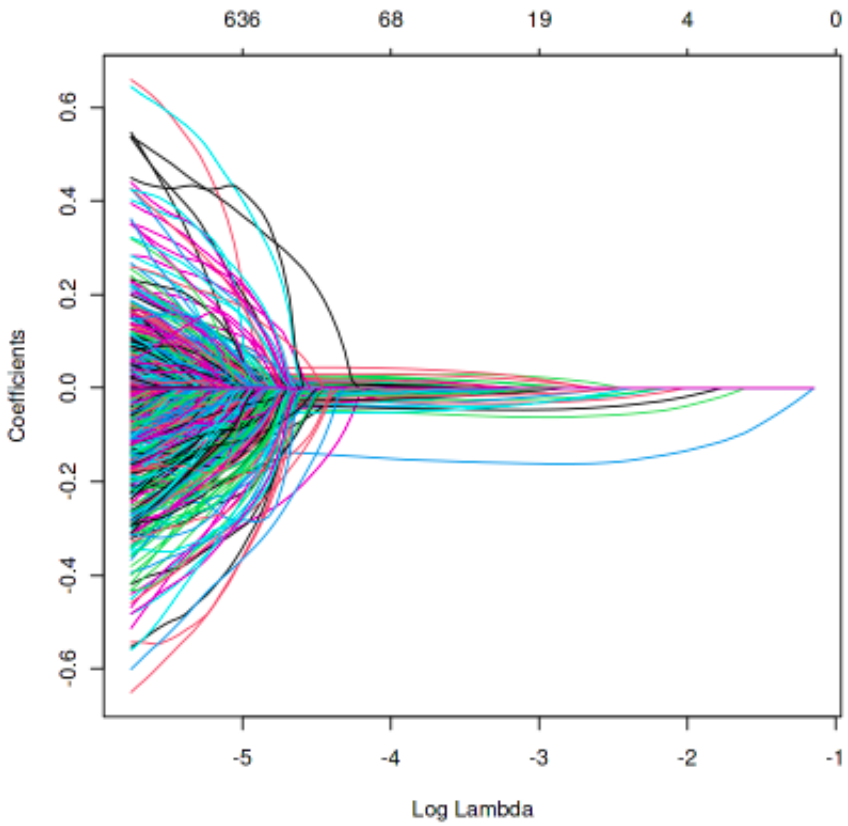
The ELBO loss consists of two terms: the reconstruction term $\mathbb{E}q\phi(z|x)[\log p_\theta(x|z)]$, which measures how well the decoder can reconstruct the input data from the latent variable, and the regularization term $D_{KL}(q_\phi(z|x)||p(z))$, which encourages the variational distribution to be close to the prior distribution over the latent space.

By maximizing the ELBO loss, the VAE learns a compressed representation of the input data in the latent space, which can be used to generate new data points by sampling from the variational distribution.

3 Analysis

3.1 Variable Selection

In our specific case, we applied the Logistic LASSO method to identify the variables impacting the response, i.e., the germinal zone or cortical plate classification of the cells in our dataset. By running the method over a range of λ values, we obtained a plot of the β_j coefficients versus λ.



As the plot revealed, several coefficients rapidly dwindled down to zero for a modest value of λ at $1/e^5$. This finding indicated that the corresponding variables were not significantly impacting the response.

Based on this observation, we chose a threshold value of $\lambda = 0.012$ ($\log\ \lambda = -4.42$), at which we discarded all variables with zero coefficients and retained the remaining ones. This process resulted in the selection of 131 variables.

Subsequently, we will use Random Forest Classification and Gradient Boosting to analyze the impact of the selected variables on the response in further sections. Remarkably, we discovered that even among the selected 131 variables, only a small subset of approximately 15 variables played a significant role in determining the response. This finding underscored the efficacy of the Logistic LASSO method in selecting only the relevant variables for constructing parsimonious models. Overall, our experience underscores the importance of variable selection in dealing with high-dimensional datasets and demonstrates the value of Logistic LASSO as an effective tool for this purpose.

3.2 Prediction via Tree Based Methods

In order to achieve a robust evaluation of the benchmark models, it is essential to segregate a portion of the data for testing purposes. This serves as a means of validating the model's predictive capabilities and its ability to generalize well to unseen data. Therefore, we allocate

10% of the available data to the test set, which is then kept separate from the training set during the entire model building process.

It is noteworthy that the performance of a model on the training set alone can be highly misleading, as it may lead to overfitting and result in a model that is not able to generalize well to new data. By allocating a test set, we can assess the true performance of the model on unseen data and ensure that the model has not only memorized the training set but also learned generalizable patterns that can be applied to new data.

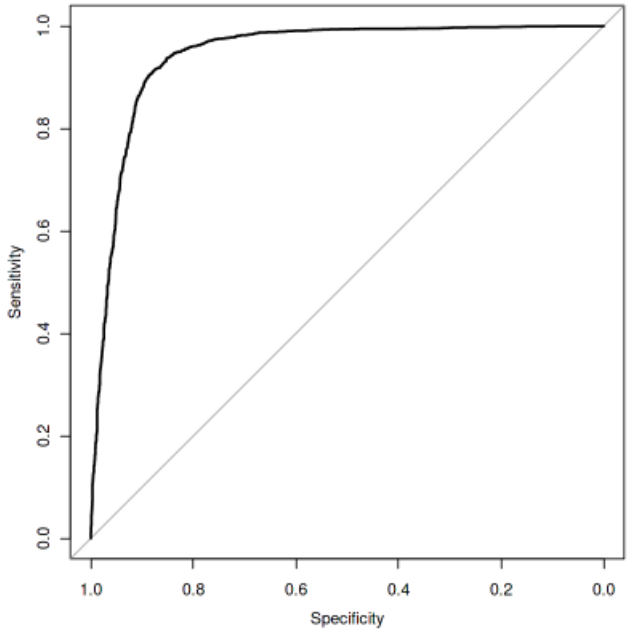
Hence, the test set serves as a vital component in model evaluation and enables us to draw meaningful conclusions about the model's performance and its ability to make accurate predictions on unseen data.

3.2.1 Logistic Regression

After the variable selection process, we utilized the benchmark Logistic Regression to evaluate the selected 131 variables against the Tree based methods. The results for Logistic Regression are as follows:

	Value
Specificity	91.52%
Sensitivity	87.46%
Accuracy	89.54%

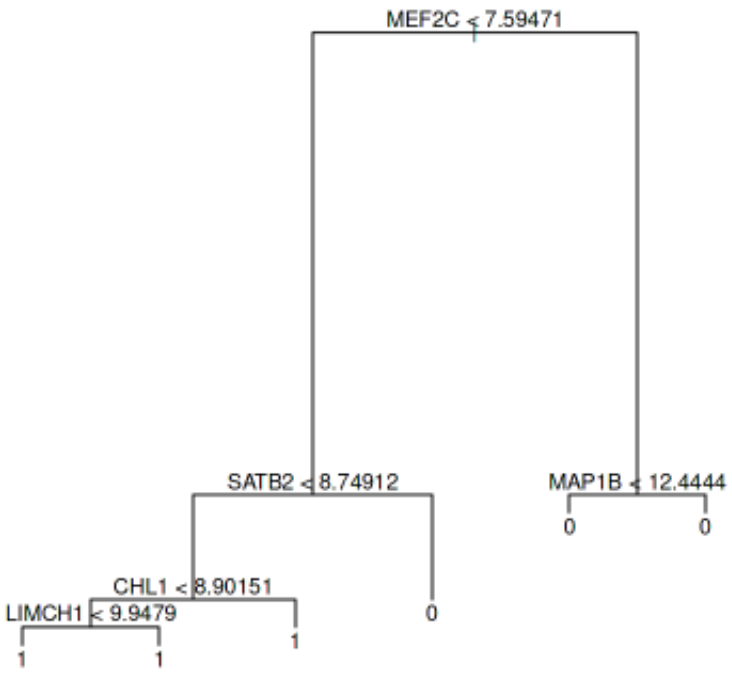
The Receiver Operating Characteristic (ROC) curve for the Logistic Regression is presented below:



These results demonstrate that the Logistic Regression method performs reasonably well in predicting the classification of cells into germinal zone or cortical plate based on the selected variables. The high specificity and sensitivity scores indicate the accuracy of the predictions. The ROC curve provides a graphical representation of the trade-off between the true positive rate and false positive rate for different classification thresholds. It can be seen that the Logistic Regression method produces a curve that is relatively close to the top left corner, which is indicative of a good classifier.

3.2.2 Classification Trees

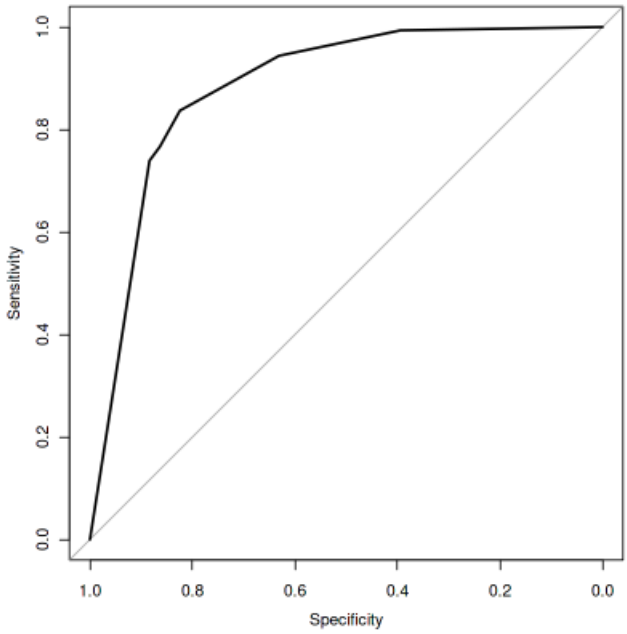
In this section, we present our results for constructing a single classification tree and its corresponding prediction benchmarks. The tree structure is visually depicted in the following figure:



We evaluate the performance of the tree based on the metrics of specificity, sensitivity, and accuracy, which are presented in the following table:

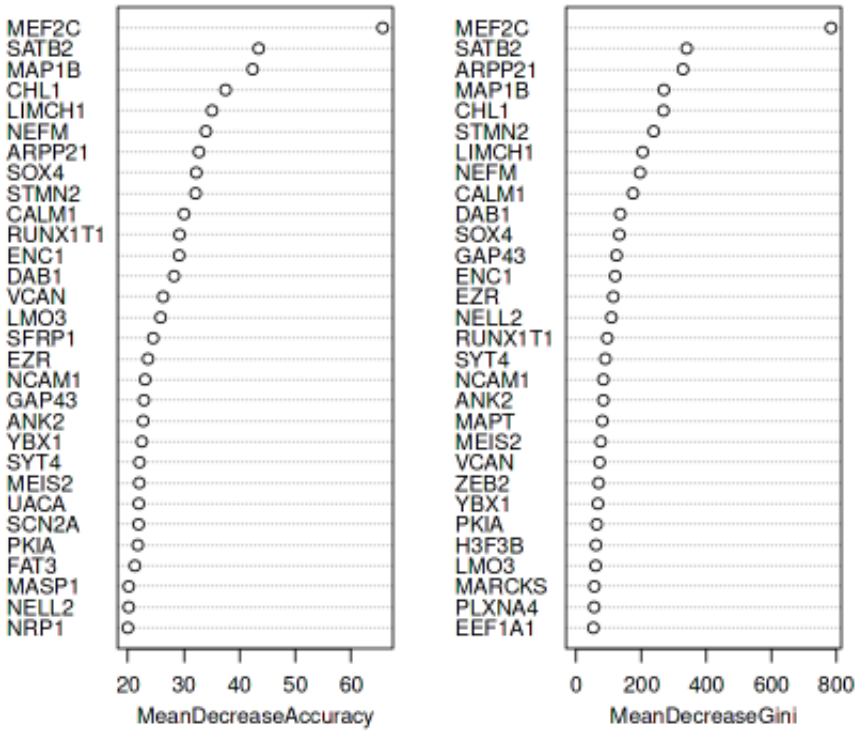
	Value
Specificity	83.70%
Sensitivity	82.43%
Accuracy	82.42%

Additionally, we depict the ROC curve for the Classification Tree in the following figure:



3.2.3 Random Forest

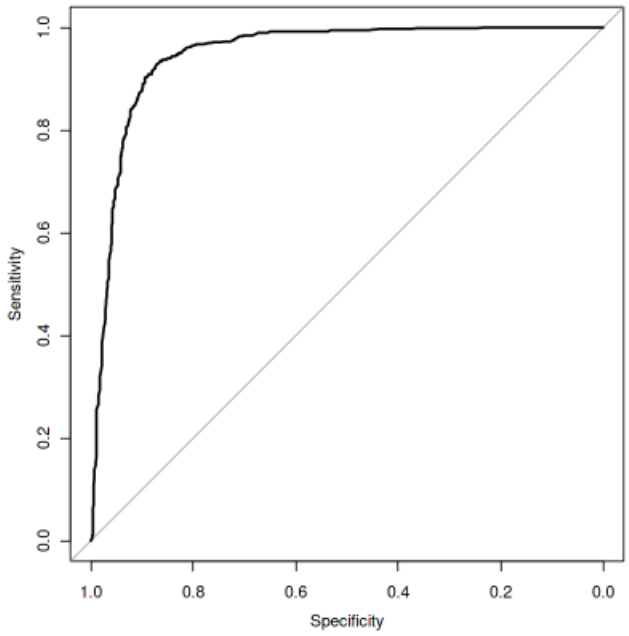
In this section, we highlight the significance of our variable selection method by constructing a plot for Variable Influence on the response based on Mean Decrease Accuracy and Mean Decrease Gini based off of Random Forest Classification.



he plot reveals that only a few variables exert significant influence on the response, including MEF2C, SATB2, MAP1B, CHL1, etc. Moreover, the influence of the less influential variables decreases exponentially, further attesting to the effectiveness of our variable selection.

	Value
Specificity	91.53%
Sensitivity	87.97%
Accuracy	89.78%

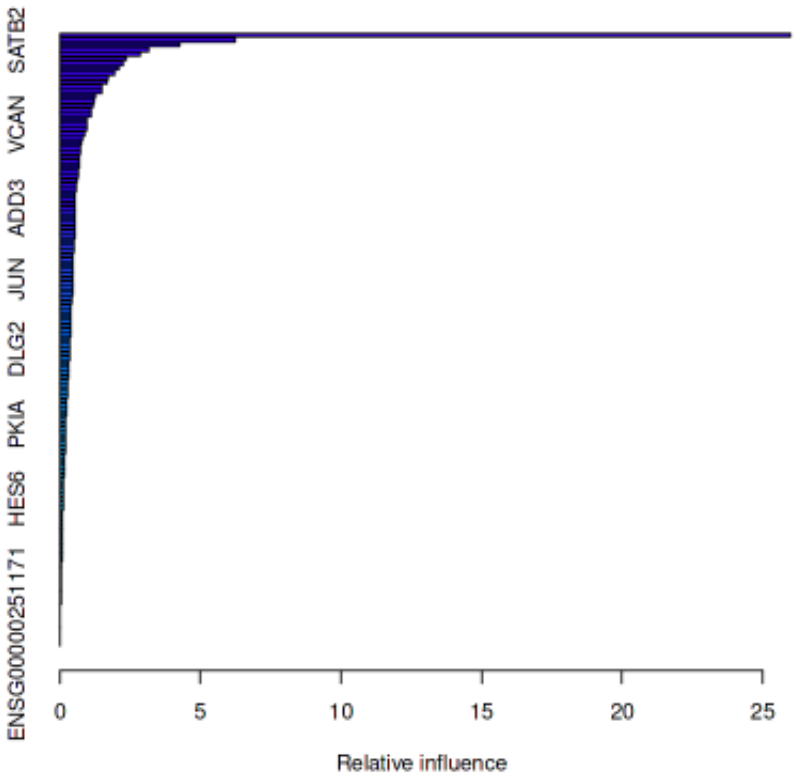
Furthermore, after testing our trained model on the test data, we observed that the selected variables can predict the response with good accuracy, as evident from the prediction benchmarks based on a probability cut-off of 0.5. The model achieved a specificity score of 91.53%, a sensitivity score of 87.97%, and an accuracy score of 89.78%.



The ROC curve for the Random Forest Classification, which depicts the trade-off between sensitivity and specificity, is also presented for a visual representation of the model's performance. Overall, the results further corroborate the effectiveness of our variable selection method and demonstrate the reliability of the Random Forest Classification model in predicting the response.

3.2.4 Gradient Boosting

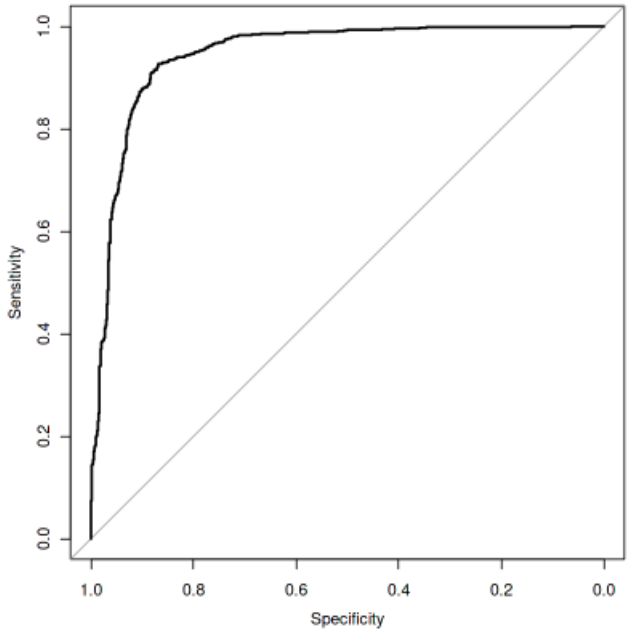
In this section, we delve into the Gradient Boosting method and construct a variable influence plot. The plot is intended to highlight the variables that have a significant impact on the response.



Subsequently, we present the prediction benchmarks of the Gradient Boosting method, which are based off a probability cut-off value of 0.5. Notably, the method performed reasonably well, achieving an accuracy score of 89.65%. The Specificity and Sensitivity values are also recorded as 83.43% and 82.43%, respectively.

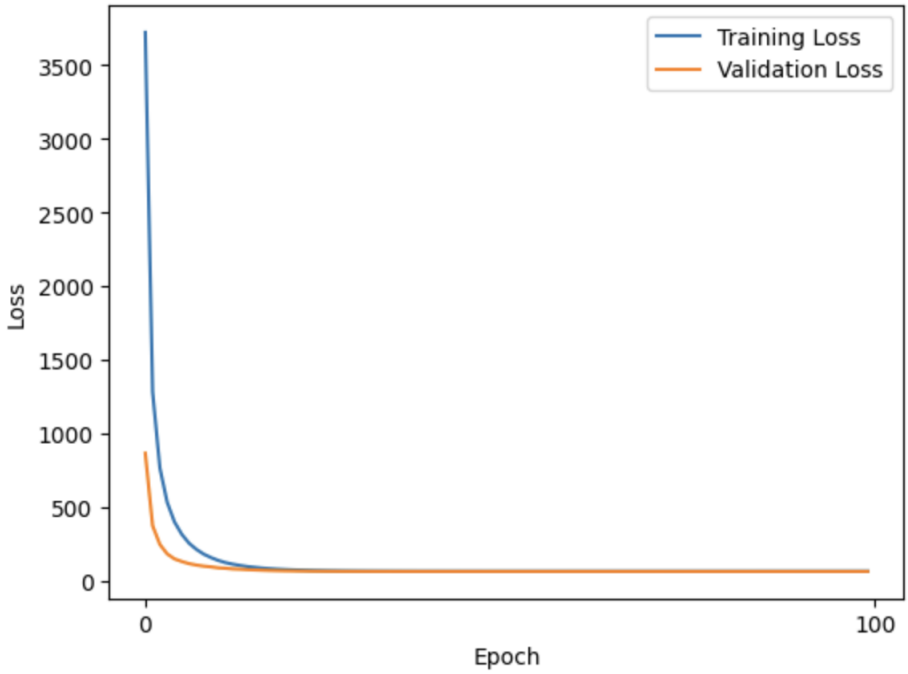
	Value
Specificity	83.43%
Sensitivity	82.43%
Accuracy	89.65%

Furthermore, the ROC curve, which plots the trade-off between Sensitivity and Specificity values, is provided below.

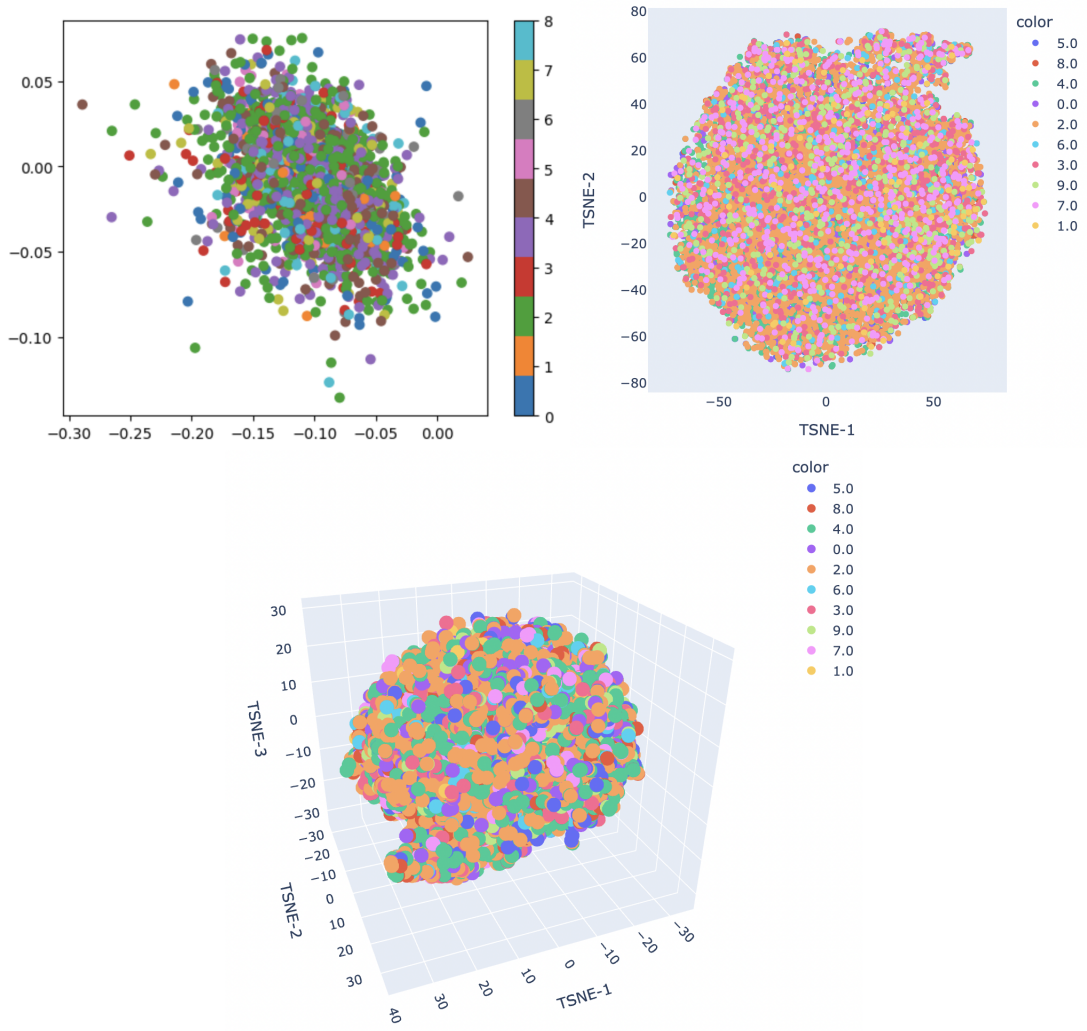


3.3 Prediction via Variational Autoencoder

After initializing the latent space dimension to 30, our model underwent 100 epochs of training, resulting in a loss curve that exhibited a satisfying convergence. The training and validation sets' loss curves are depicted below.

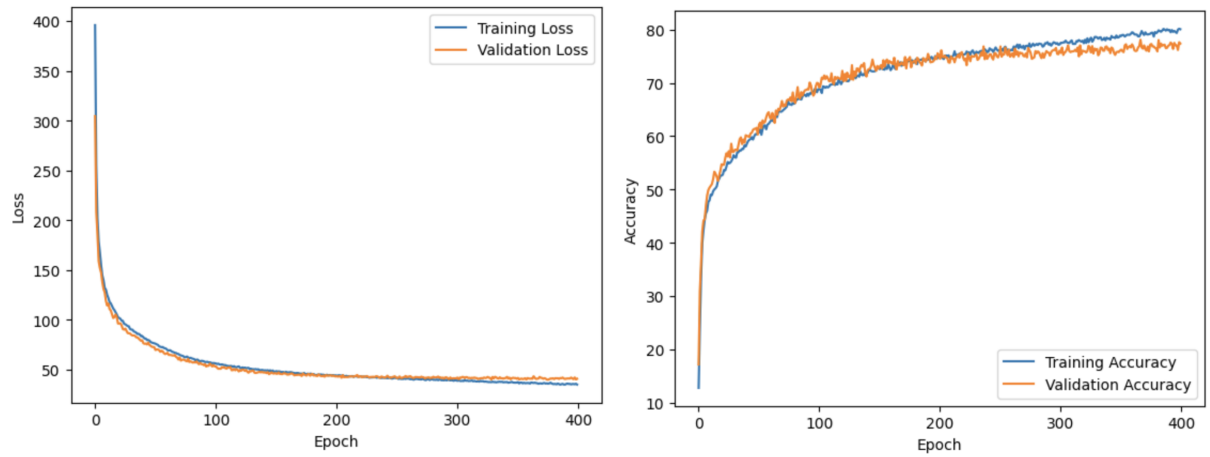


Subsequently, we visualized the latent space by utilizing a scatter plot and t-Distributed Stochastic Neighbor Embedding (t-SNE) on the first two dimensions. Figure below shows that the model learned an impressive low-dimensional representation for various classes in the dataset.

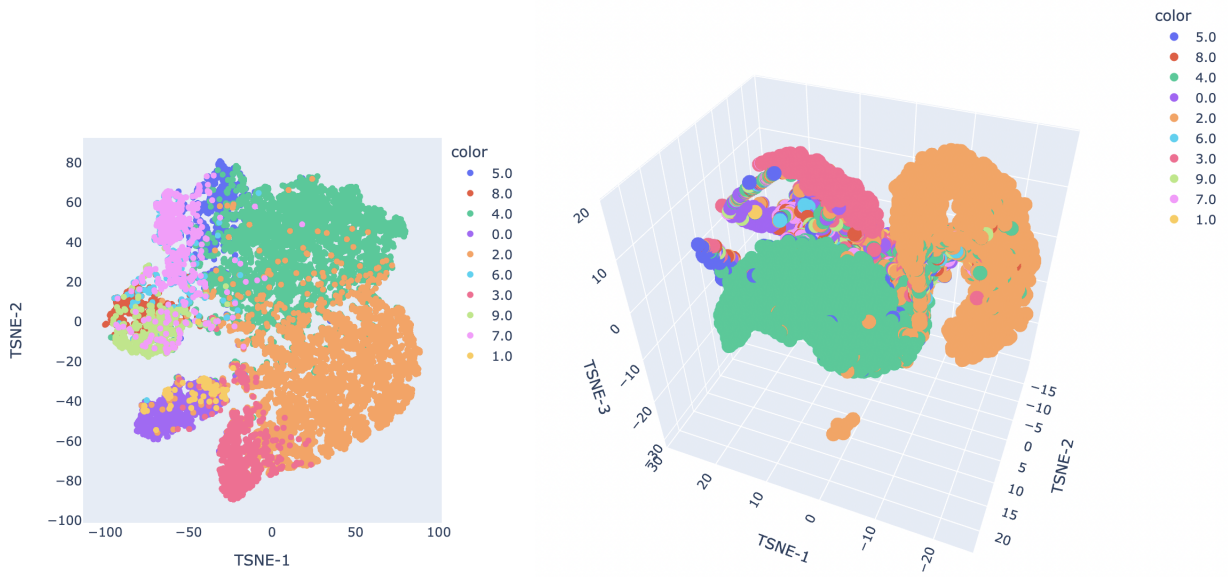


However, the clusters were densely packed and lacked clear boundaries, leading us to develop a classifier on the latent space.

Our classifier, a multilayer perceptron (MLP), was trained for only 100 epochs due to limited GPU resources, with loss and accuracy serving as performance evaluation metrics. The loss and accuracy curves are presented below.



Although the MLP was limited in size and depth, the model achieved decent classification accuracy of 77.51% for different cell types in the dataset. The classifier on the latent space helped the model learn more semantic understanding of features (i.e., genes) in the dataset. Specifically, visualizing the latent space in 2D and 3D with t-SNE revealed that the classifier facilitated better separations between different clusters, leading to a more accurate representation. This improved representation is presented below.



4 Discussion

Upon evaluating the performance of various classification methods on the Geschwind dataset, it can be inferred that the Random Forest Classification algorithm emerges as the most efficient and effective technique, surpassing all other methods in terms of classification accuracy and other evaluation metrics. Close on its heels is the Logistic Regression method, exhibiting a noteworthy performance that is almost at par with that of Random Forest. However, the

Generative Modeling approach has proved to be quite inadequate in terms of classification scores, being incapable of achieving the benchmark set by even a single Classification Tree. Therefore, it can be surmised that the Random Forest Classification algorithm is the most promising technique for accurate classification of the Geschwind dataset among the various evaluated methods.

5 Relevant Codes

We hereby present the codes used for the purpose of variable selection via Logistic LASSO and for prediction through employment of the Tree based methods.

Preprocessing the Data

```
df <- read.csv('GW2_data.csv')
samples <- sample(1:nrow(df), nrow(df) * 0.66)
df.train <- as.matrix(df[samples, -1])
df.test <- as.matrix(df[-samples, -1])
y <- read.csv('GW2_meta.csv')
y.train <- as.factor(as.numeric(y$Layer == 'GZ'))[samples]
y.test <- as.factor(as.numeric(y$Layer == 'GZ'))[-samples]
```

Perform Logistic LASSO for Variable Selection

```
library(MASS)
library(glmnet)

fit.lasso <- glmnet(df.train, y.train, family="binomial", alpha=1)
plot(fit.lasso, xvar= "lambda")
features <- as.numeric(coef(fit.lasso, 0.012)) != 0

x.train <- df.train[,features[-1]]
x.test  <- df.test[,features[-1]]

df.train <- data.frame(x.train)
df.train$response <- y.train

df.test <- data.frame(x.test)
df.test$response <- y.test
```

Logistic Regression

```
library(pROC)
library(caret)

model <- glm(response ~ ., family=binomial(link='logit'), data=df.train)
mean(as.numeric(predict(model, data.frame(x.test), type="response") >=
    0.5) == y.test) #Accuracy
conf_matrix<-table(as.numeric(predict(model, data.frame(x.test), type="
    response") >= 0.5)
                   ,y.test) #Confusion Matrix

sensitivity(conf_matrix) #sensitivity
specificity(conf_matrix) #specificity
```

```
dfp <- data.frame(y.test)
dfp$prob <- as.numeric(predict(model, data.frame(x.test), type="response"))
g <- roc(y.test ~ predict(model, data.frame(rbind(x.test), type="response"), data = dfp))
plot(g) #ROC Curve
```

Classification Tree

```

df <- df[features]
df$response <- as.factor(as.numeric(y$Layer == 'GZ'))
tree.df <- tree(response ~ . , df)

plot(tree.df)
text(tree.df , pretty = 0)
train <- sample(1:nrow(df), nrow(df) * 0.9)

x.test <- df[-train , -ncol(df)]
y.test <- df[-train , ncol(df)]
tree.df <- tree(response ~ . , df ,subset = train)
tree.pred <- predict(tree.df , x.test, type = "class")
conf_matrix <- table(tree.pred , y.test) #Confusion Matrix

sensitivity(conf_matrix) #sensitivity
specificity(conf_matrix) #specificity

dfp <- data.frame(y.test)
dfp$prob <- as.numeric(predict(tree.df , x.test, type = "vector")[,2])
g <- roc(y.test ~ prob, data = dfp)
plot(g) #ROC Curve
```

Random Forest

```
library(randomForest)

rf.df <- randomForest(response ~ ., data = df, subset = train, importance = TRUE)
yhat.rf <- predict(rf.df, newdata = x.test)
sum(yhat.rf == y.test) / length(y.test) #Accuracy

importance(rf.df)
varImpPlot(rf.df)

conf_matrix <- table(yhat.rf, y.test) #Confusion Matrix
sensitivity(conf_matrix) #sensitivity
specificity(conf_matrix) #specificity

dfp <- data.frame(y.test)
dfp$prob <-  predict(rf.df, newdata = x.test, type = "prob")[,2]
g <- roc(y.test ~ prob, data = dfp)
plot(g) #ROC Curve
```

Gradient Boosting

```
library(gbm)

boost.df <- gbm(response ~ ., data = df[train , ], n.trees = 5000,
    distribution = "multinomial")
summary(boost.df)

yhat.boost <- predict(boost.df, newdata = x.test, type = "response")
mean(as.numeric(yhat.boost <= 0.5)[1:1507] == y.test) #Accuracy
conf_matrix <- table(tree.pred , y.test) #Confusion Matrix

sensitivity(conf_matrix) #sensitivity
specificity(conf_matrix) #specificity

dfp <- data.frame(y.test)
dfp$prob <- 1 - as.numeric(yhat.boost)[1:1507]
g <- roc(y.test ~ prob, data = dfp)
plot(g) #ROC Curve
```

References

1. D. Polioudakis et al (2019). A single-cell transcriptomics atlas of human neocortical development during mid-gestation *Neuron*.

2. Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

3. Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

4. Hosmer Jr, D. W., and Lemeshow, S. (2000). *Applied Logistic Regression*. Wiley.

5. Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and Regression Trees*. Wadsworth.

6. Chen, T., and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

7. Meinshausen, N. (2006). Quantile regression forests. *Journal of Machine Learning Research*, 7, 983-999.

8. Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, 29(5), 1189-1232.

9. Chen, T., and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

10. Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

11. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T. Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems (NIPS)*.

12. Mason, L., Baxter, J., Bartlett, P. L., and Frean, M. (1999). Boosting Algorithms as Gradient Descent. *Advances in Neural Information Processing Systems (NIPS)*.