May 2019

Machine Learning Prediction of Primary Tissue Origin of Cancer from Gene Expression Read Counts

Lohitha Chintham Reddy
chinthamlohitha@gmail.com

MACHINE LEARNING PREDICTION OF PRIMARY TISSUE ORIGIN OF CANCER FROM GENE EXPRESSION READ COUNTS

By

Lohitha Chintham Reddy

Bachelor of Technology, Computer Science
Jawaharlal Nehru Technological University, Anantapur
2017

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science in Computer Science

Department of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
May 2019

UNLV | GRADUATE COLLEGE

Thesis Approval

The Graduate College
The University of Nevada, Las Vegas

April 16, 2019

This thesis prepared by

Lohitha Chintham Reddy

entitled

Machine Learning Prediction of Primary Tissue Origin of Cancer from Gene Expression Read Counts

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
Department of Computer Science

Fatma Nasoz, Ph.D.
Examination Committee Chair

Kathryn Hausbeck Korgan, Ph.D.
Graduate College Dean

Ajoy Datta, Ph.D.
Examination Committee Member

Kazem Taghva, Ph.D.
Examination Committee Member

Mira Han, Ph.D.
Graduate College Faculty Representative

Abstract

Cancer is a group of diseases characterized by the uncontrolled growth and spread of abnormal cells. Generally, manufacturing of proteins by cells is controlled by genes. Each gene must have the correct instructions for making its protein, so that it allows the protein to perform the correct function for the cell. When one or more genes in a cell mutate and create an abnormal protein, that is when cancer begins. An abnormal protein provides different information compared to a normal protein. This can cause cells to multiply uncontrollably and cause cancer.

RNA sequencing (RNA-seq) can be used to figure out the functional and structural changes occur in genes. RNA-seq can profile the abundance and composition of entire transcriptome. It is a highly sensitive and accurate tool for measuring expression across the transcriptome. Since it can reveal the changes affecting genes, RNA seq can be valuable for diagnosing, characterizing tumors.

For building the models to classify the given transcriptome data into a particular tissue, data from The Cancer Genome Atlas (TCGA) was used. Using this data, we implemented and compared various machine learning classification algorithms including decision trees, support vector machines, random forest classifiers, K nearest neighbors, stochastic gradient descent classifier to classify gene expression data to a particular tissue. Among all the classifiers used, Stochastic gradient descent classifier with squared hinge loss function had the best performance based on the traditional machine learning metrics.

RNA sequencing, which is an easy to perform test, is done in most labs, therefore there is a great variability on the quality of data. To test if the model built using TCGA transcriptome data generalizes well to classify the genome data, a test was performed on Genotype - Tissue Expression (GTEX) data (which is completely independent from TCGA data). The results were not as high as training data results, but the most evident misclassification was in between Esophagus and Stomach. When considered only TCGA data, the classification test accuracy achieved was 95.46 percent whereas when considered GTEX data too, the training accuracy was 91 percent and test accuracy was 58 percent.

Acknowledgements

I would like to express my sincere gratitude to my advisor Dr. Fatma Nasoz, for her motivation, guidance, and constant support throughout the research. She continuously steered me in the right direction in this research as well as my Master's program.

I would also like to thank Dr. Mira Han for being so patient in describing the dataset and for spending so much time to explain the terminology, give domain knowledge as much as required for the thesis as I was new to the domain. I am really grateful for her constant support in this research. I would also like to thank Travis Mize, Richard Van, Xiaogang Wu, Nikolay Stoynov for all their suggestions and constant support.

I would like to extend my thanks to Dr. Ajoy Datta and Dr. Kazem Taghva for being a part of my thesis committee. I am really grateful for all the support from Dr. Ajoy Datta who was always available whenever I needed his guidance.

My deep sense of gratitude to my parents Bhushan Chintham Reddy, Reddi Lakshmi Chintham and my brother Lalithaditya Chintham who are my moral strength and motivation. I would like to thank Sravani Gannavarapu Surya Naga, Anusha Challa for their constant support and encouragement throughout my Master's program.

Finally, I would like to thank all my friends, seniors and juniors who made my time here at UNLV very memorable.

Lohitha Chintham Reddy

University of Nevada, Las Vegas

May 2019

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Throughout our lives, if cells in our body divide and replace themselves in a controlled fashion then functioning of the cells is said to be healthy. Cancer starts when a cell is somehow altered so that it multiplies uncontrollably. A cluster of all these abnormal cells forms a tumor, which will be one of two types, benign or malignant. Benign tumors do not cause any harm like spreading to the other parts of the body. Whereas malignant tumors grow and spread in the body through a process called metastasis, which eventually creates tumors in other parts of the body too.

Since there are hundreds of types of cancer, the treatment to be given will be dependent on the type of cancer and how advanced the situation is. As cancer could be metastatic, to provide the correct treatment it is important to know the origin of cancer. To do any computer analysis regarding finding the origin of cancer, we need to have past data and then using that analysis we can find the origin of the disease to provide appropriate treatment. For that, collected data from The Cancer Genome Atlas (TCGA), a landmark of cancer genomics program, molecularly characterized over 20,000 primary cancer and matching normal samples spanning 33 cancer types. This joint effort between the National Cancer Institute and the National Human Genome Research Institute began in 2006, bringing together research from diverse disciplines and multiple institutions. TCGA generated over 2.5 petabytes of genomic, epigenomic, transcriptome, and proteomic data[Can19]. For this project, we used transcriptome data. Human genome is made up of DNA, which contains the instructions needed to build and maintain cells. These instructions are spelled out in the form of base pairs organized into 20,000 to 25,000 genes. For the instructions to be carried out, DNA must be "read" and transcribed into RNA. These gene readouts are called transcripts, and a transcriptome is a collection of all the gene readouts present in a cell.

An RNA sequence mirrors the sequence of the DNA from which it was transcribed. Conse-

quently, by analyzing the entire collection of RNA sequences in a cell, researches can determine when and where each gene is turned on or off in the cells and tissues of an organism. Depending on the technique used, it is often possible to count the number of transcripts to determine the amount of gene activity, also called gene expression in a certain cell or tissue type. Using those gene expressions of samples, with the help of already collected data samples and their tissue types, we have trained machine learning models and tested on the previously unseen data.

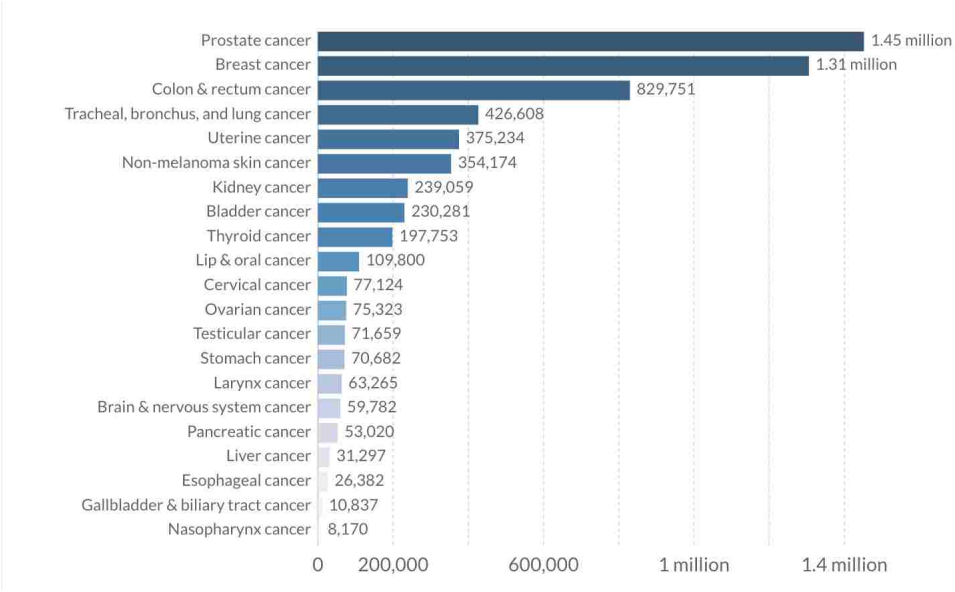

Figure 1.1: Cancer Type

The motivation to work on this project is that according to The Institute of Health Metrics and Evaluation, cancer is the second leading cause of death. According this source, 8.75 to 9.1 million people are estimated to have died from various forms of cancer [MH17]. The graph presented below are visualized regarding cancer prevalence by age. Overall we can observe that majority of cancers occurred in those aged 50 and over. Figure 1.1 and 1.2 shows the statistics of cancer details in the world. The source of these figures is [MH17].

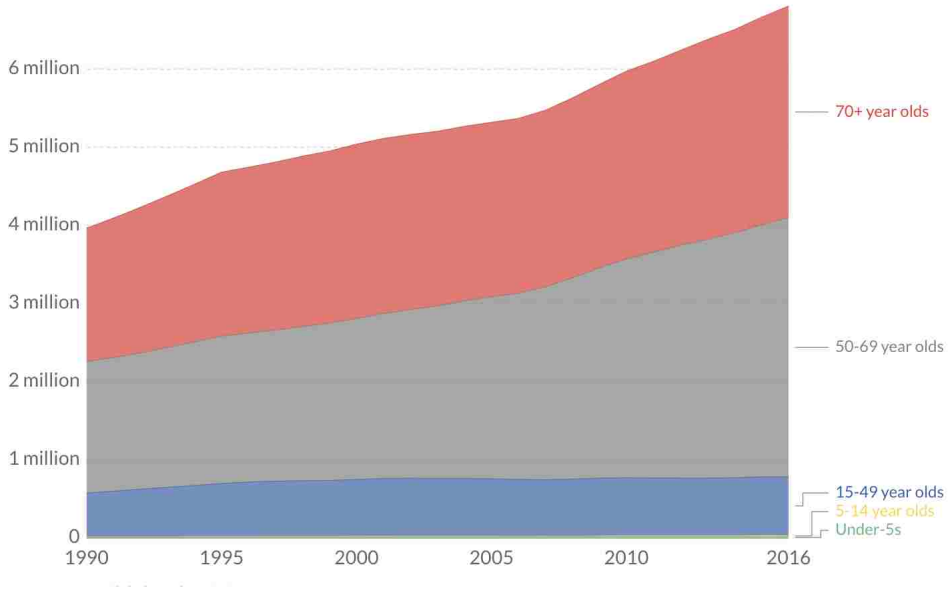

Figure 1.2: Cancer Statistics

By looking at all the information provided, we have to figure out the origin of cancer from which it has started and then spreaded. With that important information, we can provide proper treatment for the patient and as well we can take preventive measures or introduce new policies to prevent that type of cancer to save lives.

1.1 Objective

The main focus of this thesis is to create predictive models to classify the given transcriptome data of a sample to a particular tissue based on its gene expression counts. To do this, machine learning algorithms including decision trees, support vector machines, random forest classifier, stochastic gradient descent classifier were used and trained on TCGA data. Before applying algorithms, different sampling techniques like synthetic minority over-sampling technique (SMOTE), TOMEK link removal were implemented to handle the imbalance nature of the labels distribution in the

dataset. Trained models were used to classify the samples of the test data. The trained models were evaluated and then compared using classic metrics like precision, recall, F-1 score and accuracy to determine which models were best in prediction and the results were also reported.

1.2 Outline

In Chapter 1, the brief reason why cancer is caused, the data which can used to detect the cancer tissue origin, details about the data and the statistics of cancer details are mentioned, as well as the proposed approach to classify the samples.

In Chapter 2, the related work on RNA-seq data analysis will be discussed, the machine learning approaches used in the thesis will be explained, and all the background information required to understand the work will also be provided.

In Chapter 3, we will discuss the methodology used for the analysis including the steps involved in data preparation process.

In Chapter 4, we will report results of the experiments performed on the datasets and then compare the results of different models by also reporting the best performing model for classification.

In Chapter 5, we will summarize the findings of the analysis along with some insights about future work.

Chapter 2

Background and Preliminaries

2.1 Related Work

RNA sequencing (RNA-seq) is a powerful technique that uses the capabilities of next-generation sequencing technologies for the gene-expression profiling of organisms. Developing gene-expression based classification algorithms is an emerging powerful method for diagnosis, disease classification and monitoring at molecular level, as well as providing potential markers of diseases. Since RNA sequencing is the most available technique in many laboratories and the data is easily available, many researches have done experiments in classifying the transcriptome data. In this section, we will discuss about the past research studies done in this field.

In a recent study, Gokmez Zararsiz focused on a comprehensive simulation study on classification of RNA-Seq data. From their study, most of statistical methods proposed for classification of gene-expression data are based on a continuous scale or require a normal distribution assumption, so these cannot be directly applied to RNA seq data since it violates both the data structure and distributional assumptions. However, it is possible to so this with appropriate modifications to RNA-Seq data. One way is to develop count-based classifiers, such as Poisson linear discriminant analysis and negative binomial linear discriminant analysis. Another way is to bring the data closer to microarrays and apply microarray-based classifiers. In their study, they compared several classifiers including probabilistic linear discriminant analysis (PLDA) with and without power transformation, negative binomial linear discriminant analysis (NBLDA), single support vector machines (SVM), bagging SVM (bagSVM), classification and regression trees (CART), and random forests (RF). They also examined the effect of several parameters such as over dispersion, sample size, number of genes, number of classes, differential-expression rate, and the transforma-

tion method on model performances. The results revealed that increasing the sample size and the differential-expression rate and decreasing the dispersion parameter and the number of groups lead to an increase in classification accuracy. Similarly with differential-expression studies, the classification of RNA-Seq data requires careful attention when handling data over dispersion. We conclude that, as a count-based classifier, the power transformed PLDA and, as a microarray-based classifier, vst or rlog transformed RF and SVM classifiers may be a good choice for classification [ZGK+17].

Another study researched Machine Learning Interface to classify RNA-Seq data. MLSeq is a comprehensive package for application of machine learning algorithms in classification of next-generation RNA-Sequencing data. Researchers have appealed to MLSeq for various purposes, which include prediction of disease outcomes, identification of best subset of features (genes, transcripts, other isoforms), and sorting the features based on their predictive importance. For pre processing that data, they used deseq median ratio and trimmed mean of M means (TMM) normalization methods, as well as the logarithm of counts per million reads (log-cpm), variance stabilizing transformation (vst), regularized logarithmic transformation (rlog) and variance modeling at observational level (voom) transformation approaches. Currently, MLSeq package contains 90 microarray-based classifiers including the recently developed voom-based discriminant analysis classifiers. Besides these classifiers, MLSeq package also includes discrete-based classifiers, such as probobilistic linear discriminant analysis (PLDA) and negative binomial linear discriminant analysis (NBLDA) [GZK19].

In another study, researches worked on deep learning to analyze RNA-Seq gene expression data. In this work, a first approximation on the use of deep learning for the analysis of RNA-Seq gene expression profiles data is provided. Three public cancer-related databases are analyzed using a regularized linear model (standard LASSO) as a baseline model, and two deep learning models that differ on the feature selection technique used prior to the application of a deep neural net model. The results indicate that a straightforward application of deep learning implementations available in public scientific tools and under the conditions described within this work is not enough to outperform simpler models like LASSO. In general, it can be seen that the predictive performance in terms of area under curve (AUC) is relatively poor independently of the model used. On two out of three databases, breast (BRCA) and colon (COAD), the performance measured by the AUC is not over 0.65. The results obtained across the three databases confirmed us that the straightforward use of existing implementations of a multi-layer feed-forward artificial neural network (such as the R

package h2o) will very rarely increase the predictive performance compared to a simple regularized linear model [UTF+17].

2.2 Preliminaries

2.2.1 Machine Learning Concepts

Machine learning is the science of getting computers to act without being explicitly programmed. To explain in more formal way with Tom Mitchells well-posed definition [Mit97]:

A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.
Example: Predict traffic patterns
E: The data about the past traffic patterns
T: Predicting traffic patterns at a busy intersection
P: Predicting future traffic patterns

In general, machine learning mainly focuses on the development of computer programs that analyze the data and use it to learn for themselves. The process of learning begins with looking for patterns in the data which are used to make future decisions based on that previous data. As per the definition, machine learning tries to learn patterns from the past data, remembers it as an experience and uses it for future predictions. The primary aim is to allow the computers learn automatically without any human interventions and adjust actions accordingly. The whole idea of the machine learning approach is trying to simulate the working of human brain, just as its performance gets better at a specific task by repeated learning and previous experiences, a computer program also learns based on the hidden patterns in the data based on its previous experiences. As the data becomes huge, it will become difficult for a human to understand whereas using machine learning will perform well due to using computer resources.

Typically, machine learning tasks are classified into broad categories like supervised learning, unsupervised learning and reinforcement learning. In this thesis, all the methods included for analysis comes under supervised learning, which means the algorithm builds a model from a set of data that contains both the input features and desired output labels. Briefly unsupervised learning builds the model with a set of data, which has only input features but not output labels, it is mostly used where the data can be segmented into clusters based on the patterns observed like in case of customer segmentation based on customer data or purchasing history or behavior of customers.

Whereas in case of reinforcement learning, algorithms are given feedback in the form of positive or negative reinforcement in a dynamic environment and are used in autonomous vehicles or in learning to play a game against a human opponent.

2.2.2 Classification

Classification is the process of predicting the class of given data points. Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y). For example, in this project, as we have different types of labels, based on the given data (gene expression counts) the model has to classify the sample to a particular label. The general approach followed is demonstrated in the figure 2.1 and is extracted from source [Pol17]. Generally classification is of two types, binary classification where there are two output labels and multi-class classification where there are three or more output labels. There are two types of learners in classification as lazy learners and eager learners. Lazy learners simply store the training data and wait until a testing data appear, these models use less time for training and more for predicting. Eager learners construct a classification model based on the given training data before receiving data for classification, this takes long time for training and less time to predict [HMS01].

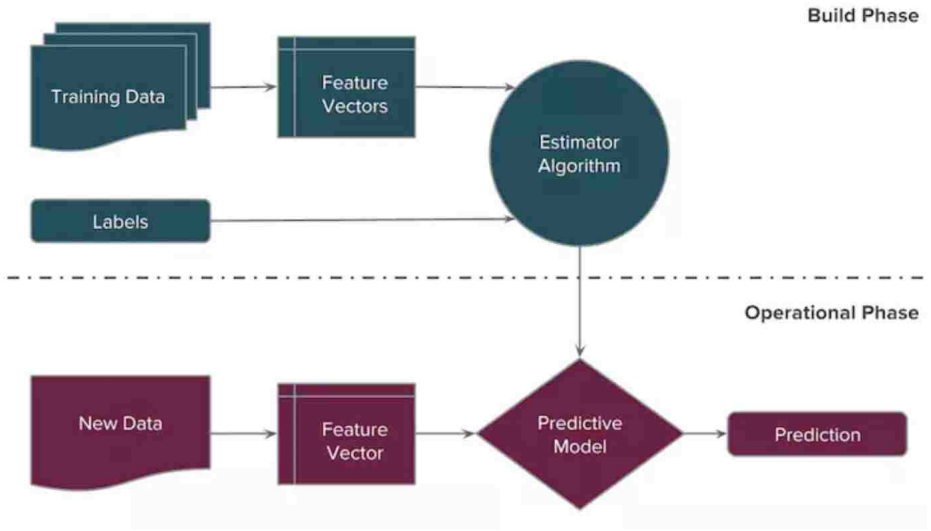

Figure 2.1: Machine Learning Approach

2.2.3 Handling class imbalance problem

Class imbalance problem is as we have multiple output labels, if the samples assigned to each output label are not equal then it is considered as class imbalance problem. In that situation, most of the machine learning algorithms perform poorly, to handle this situation the ways which can be used are resampling approach, ensemble-based approach, and cost-sensitive learning approach. In this thesis, we have used resampling approach. When learning from highly imbalanced data, most of the classifiers are influenced by majority class leading to an increase in the false negative rate.

2.2.4 Resampling Approach

Resampling is a series of methods used to reconstruct the datasets. As the models perform poorly on unbalanced dataset, data pre-processing has to be done before we provide the dataset as an input to any model. Basically, there are three sampling methods: undersampling, oversampling and hybrid methods. The figure 2.2 shows that our dataset have imbalance problem.

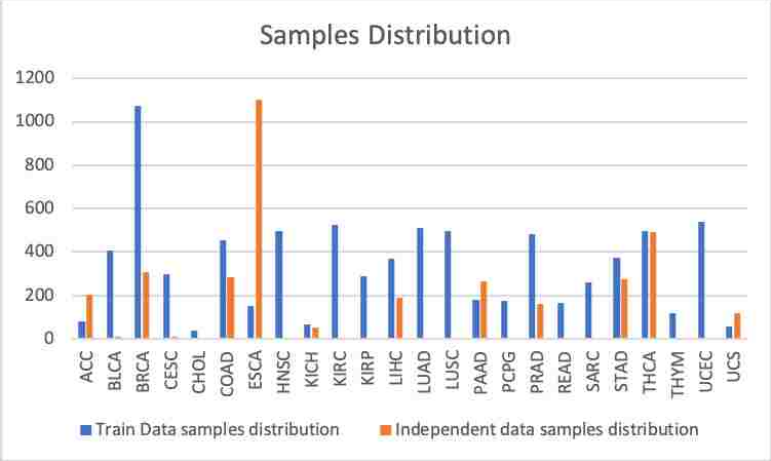

Figure 2.2: Distribution of samples against labels

2.2.4.1 Random Undersampling

Random undersampling is nothing but randomly eliminating some of the samples from majority class to make the dataset balanced. Elimination is being done so that the size of samples of that class can become equal to the size of samples of minority class. The procedure followed during this

process is selecting n random samples from majority class where n is the number of samples for the minority class. By doing this, class imbalance issue will be solved and sensitivity of the model will be increased. Figure 2.3 shows how the distribution have been changed after under sampling is done and the source of the figure is [Faw16]. One problem could be while selecting only random n samples from majority class we may miss the important information to consider which will affect the prediction as well as results.

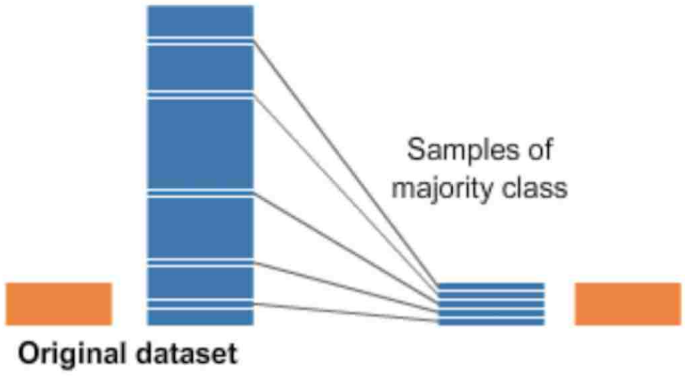

Figure 2.3: Under Sampling

2.2.4.2 Tomek Link Removal

Tomek link removal is an undersampling technique. Let x be an instance of class A and y an instance of class B. Let d(x, y) be the distance between x and y. (x, y) is a T-Link, if for any instance z, d(x, y) ¡ d(x, z) or d(x, y) ¡ d(y, z). If any two examples are T-Link then one of these examples is a noise or both examples are located on the boundary of the classes. T-Link method can be used as a method of guided undersampling where the observations from the majority class are removed [EAMAS16]. The advantage of this method is eliminating some of the samples could significantly reduce the size of the data and this method is good to use only if the size of data is big enough. This also reduces the run time cost especially in case of big data.

2.2.4.3 Random Oversampling

Random oversampling balances the class distribution in a dataset by duplicating samples of minority class. As we are duplicating the information, there will not be any information loss but there is a chance of overfitting because to make the dataset balanced we are replicating the samples. Figure 2.4 shows how the distribution of the dataset becomes balanced after oversampling has been done and the figure is used from [Faw16].

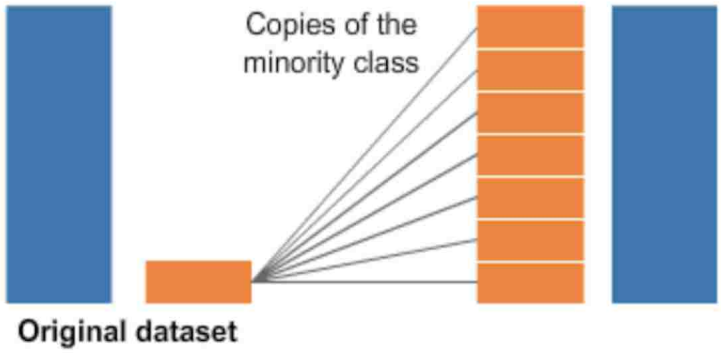

Figure 2.4: Over Sampling

2.2.4.4 SMOTE

Synthetic minority oversampling technique(SMOTE) is an oversampling approach, which synthetically generates instances by randomly selecting instances from the minority class and using interpolation method to generate instances between selected point and its nearby instances. In this process, every minority class instance is considered and new minority class instances are generated along the line segment joining its k-nearest neighbors. The number of synthetic instances is generated based on the percentage of oversampling required [CBHP02] [KG18]. Figure 2.5 shows how the oversampling will be performed and the source is [HL13].

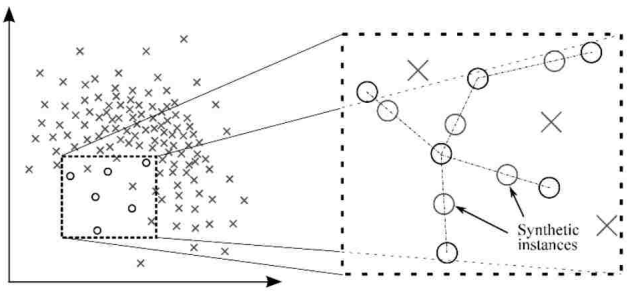

Figure 2.5: Over Sampling

2.2.4.5 Combination of SMOTE and Tomek Link Removal

SMOTE is a powerful approach to balance the class distributions. However, while creating new synthetic minority examples, the minority class cluster might invade the majority class space. Providing such data to the model can lead to over fitting. To mitigate such a situation, both SMOTE and Tomek link removal approach can be applied for balancing the class distribution. In this process, the original training dataset is first oversampled using SMOTE, and then Tomek link removal is applied to the resulting dataset producing a balanced dataset.

2.2.5 Selected Models

Based on input data given to the model, machine learning algorithms are classified into two main groups: supervised and unsupervised learning algorithms. As the data in our case have output labels that are specific to a particular tissue, we use supervised machine learning. As the outputs in our data are categorical, our problem is classification. In this section, we discuss various classification algorithms that were used in this thesis to classify the samples to a particular tissue. The models selected were decision trees, support vector machines, random forest classifier and stochastic gradient descent classifier.

2.2.5.1 Support Vector Machines

Support vector machines are supervised learning models that analyze data used for both classification and regression analysis. It is commonly called as large margin classifier and most often used for classification problems. In case of binary classification, given a set of training examples, each sample with all the features is marked as a dot in the space, SVM builds a model that assigns new

examples to one category or the other category.

The objective of the support vector machine algorithm is to find a hyperplane in an n-dimensional space that distinctly classifies the data points. A hyperplane is a line that splits the input variable space. As visualized in figure 2.6, the input variables (x) are represented in the space and classifier separates the points with a line and the source is [MPGV09].

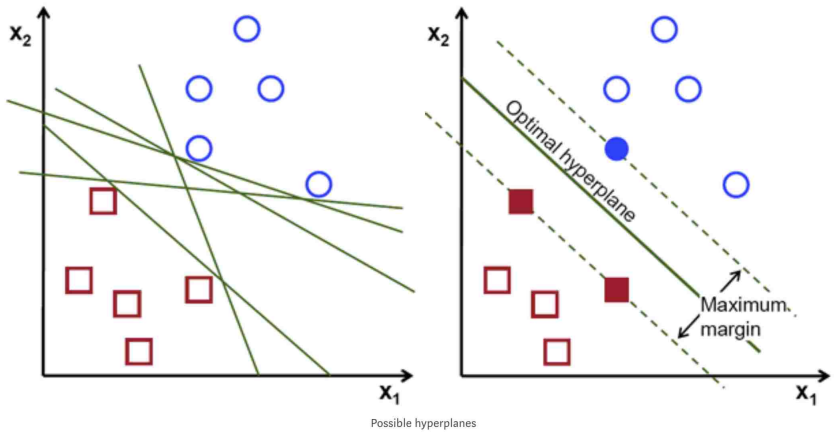

Figure 2.6: Hyper plane selection

To separate two classes of data points, there will be many possible hyper planes that could be chosen. Our objective is to find a plane that has the maximum margin which is the perpendicular distance between the line and the closest data point. We have to maintain the margin as large as possible, because for example if we have a point closest to the hyperplane then it will be difficult to classify it. These closest points play an important role in finalizing the split line and constructing the classifier model, so these are called as support vectors. The best line that separates the data points is the one with large margin, hence called as large margin classifier. After the desired margin is obtained, then new inputs are fed into the model to classify them [HTF09].

Usually support vector machines were designed for binary classification. But several methods have been proposed where we construct a multi-class classifier by combining several binary classifiers. Some methods consider all classes at once. It is computationally expensive to solve multi-class problems especially for methods solving multi-class SVM in one step, a much larger optimization problem is required [WW98]. For optimization, SVM model tunes the parameters C and kernel for

best classification. Based on the chosen C value, SVM model optimizes the hyperplane required. With large values of C, hyperplane margin would be small and if the value is small, then the margin would be large.

Kernel trick is used by SVM to create hyperplanes that separate non-linear data. While using this trick, SVM uses functions called kernels that transform the low dimensional non-separable input space into a higher dimensional separable space. The most commonly used kernels are polynomial kernels, linear kernels, and radial kernels.

2.2.5.2 Random Forest Classifiers

Random forest classifiers is another supervised machine learning algorithm that is used for both regression and classification tasks. This approach is an ensemble learning method that operates by constructing a strong learner with the help of multitude of weak learners. Ensembles are a divide-and-conquer approach used to improve performance. Weak learner is defined to be a classifier that is only slightly correlated with the true classification, in contrast, a strong learner is a classifier that is arbitrarily well correlated with the true classification.

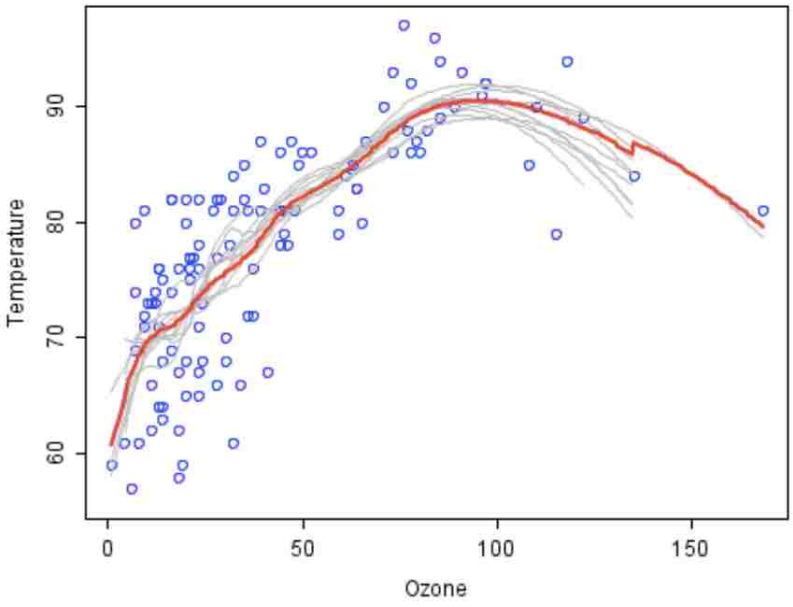

Figure 2.7: A strong classifier from multiple weak classifiers

Figure 2.7 provides an example of random forest classifier. Blue circles are the data to be modelled, each gray curve in the graph is a weak learner, and red curve the ensemble strong learner can be seen as the better approximation to the underlying data [PAWV18].

More specifically random forest starts with a standard machine learning technique called decision trees which correspond to weak learner. When datasets are trained using decision trees, they perform well on training datasets, but might perform poorly on test datasets or any new datasets. This happens because the decision trees try to learn deeply about the patterns in the training datasets which results in overfitting. But in case of random forests, which is an average of multiple decision trees, the final decision will be made depending on the result obtained from all the decision trees.

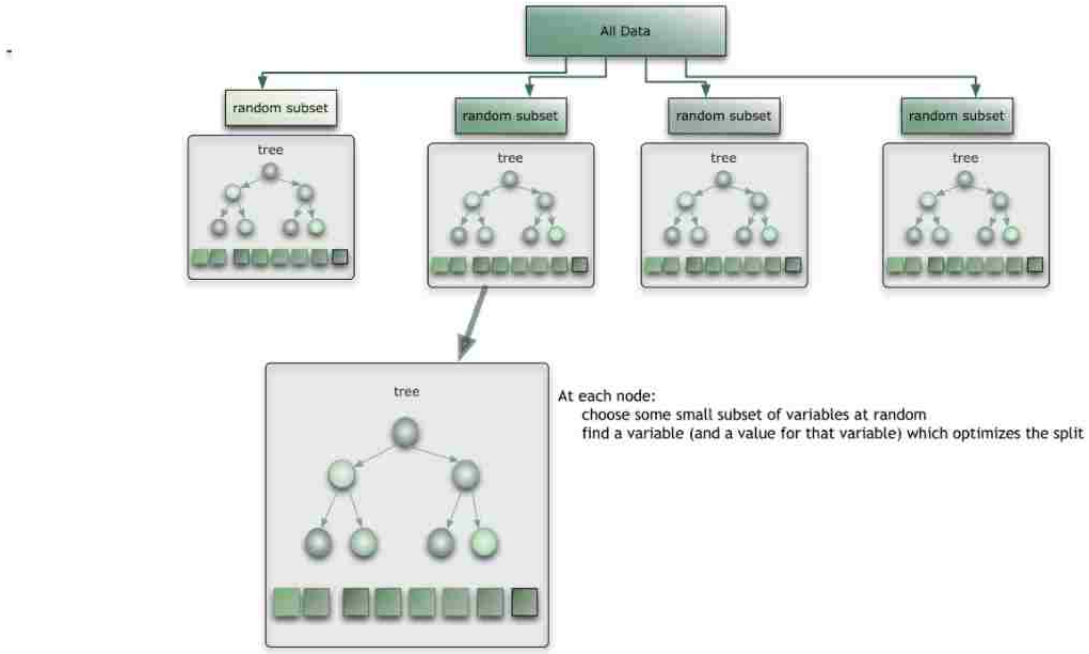

Figure 2.8: Procedure followed in Random Forest Classifier

As discussed above, in decision trees input is fed to the root of the tree and the input gets divided into subsets and gets diverged into different branches where the decisions are made. Whereas in case of random forests, initially the dataset is divided into multiple subsets and each subset is fed to a decision tree. By doing this, we will have multiple structures, and when a new input is given

for prediction, a decision will be made based on the outcome of all structures[SS16]. The whole procedure followed is shown in Figure 2.8. Figures mentioned in this subsection are extracted from [SLSH17].

2.2.5.3 Decision Trees

Decision trees is an another supervised machine learning algorithm that is used for classification tasks. Basically decision trees uses a tree representation to make predictions on the given dataset. It starts at the root, and at each branch a choice is made between number of alternatives of an attribute in the internal nodes leading to a final decision in the leaf node.

The general approach of a decision trees is splitting the data at each branch to subsets based on specifications. A basic example is given in Figure 2.9 that uses titanic data set for predicting whether a passenger will survive or not. In the example given, at each branch decision has been made and finally output has been produced at the leaf node. The source of Figure is [Qui86].

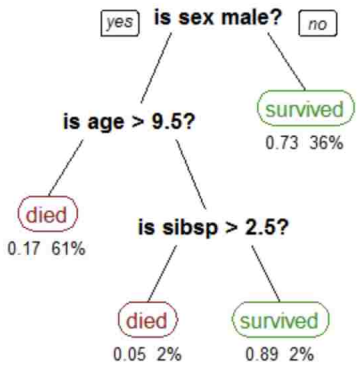

Figure 2.9: Decision Tree Example

The main goal of decision trees is to learn the importance of each feature, so that relations can be noticed easily. Since the number of features will be much higher in a real dataset, it is important to decide which features to use and when to stop the growth of the tree.

As real-life datasets have large number of features, it would result in having large number of splits in the tree, which increases the depth of the tree. If we have more splits, that would result in overfitting because of which pruning is very important. To avoid this problem of overfitting, we

should have an idea of when to stop splitting and growing the tree. To do this, we have to use possibly minimum training samples on each leaf. Another way is to set maximum depth of the model. Maximum depth refers to the the length of the longest path from a root to a leaf [Gup17].

By using pruning, we can improve the performance of a tree. This method removes the branches that involve the features, which have low importance in the classification. By doing this, we can reduce the depth of the tree and the number of decisions to be made. As a result, predictive power can be increased by reducing overfitting. We can start this procedure from either the root or the leaves [Gup17].

2.2.5.4 Stochastic Gradient Descent Classifier

Generally, when building a model for prediction purposes, the model should learn the parameters. The model learns a function f such that f(X) maps x to y. To put differently, the model learns how to model X in order to predict y. One of the ways to learn the parameters is through the cost function. Cost functions are used to estimate how well the models are performing. A cost function is a measure of how wrong the model is in terms of its ability to estimate the relationship between X and y. This typically expresses the distance between the predicted value and the actual value. The cost functions runs the model in an iterative fashion to compare the estimated predictions against the actual values. To obtain the parameters that result in minimizing the cost function, we use gradient descent optimization algorithm.

Gradient descent is an optimization algorithm which is used to find the parameter values of a function f that minimizes the cost function. To illustrate through an example, if we take a bowl initially we consider the parameters at random point. The goal is to continue to try different values and evaluate their cost and find the new values which slightly lower the cost. By doing this repetitively for enough times we end up at the bottom of the bowl where the parameter values result in the minimum cost. The consideration of new parameter values to minimize the cost function is usually done as shown in Figure 2.10 and the source is [SASK12]. This can be slow on large data sets because one iteration of the gradient descent algorithm requires a prediction for each instance in the training set, in such situations we use stochastic gradient descent [Rud16].

In stochastic gradient descent, rather than updating the parameter values at the end of computing the cost for a batch of instances, the update made for each training instance. First, the

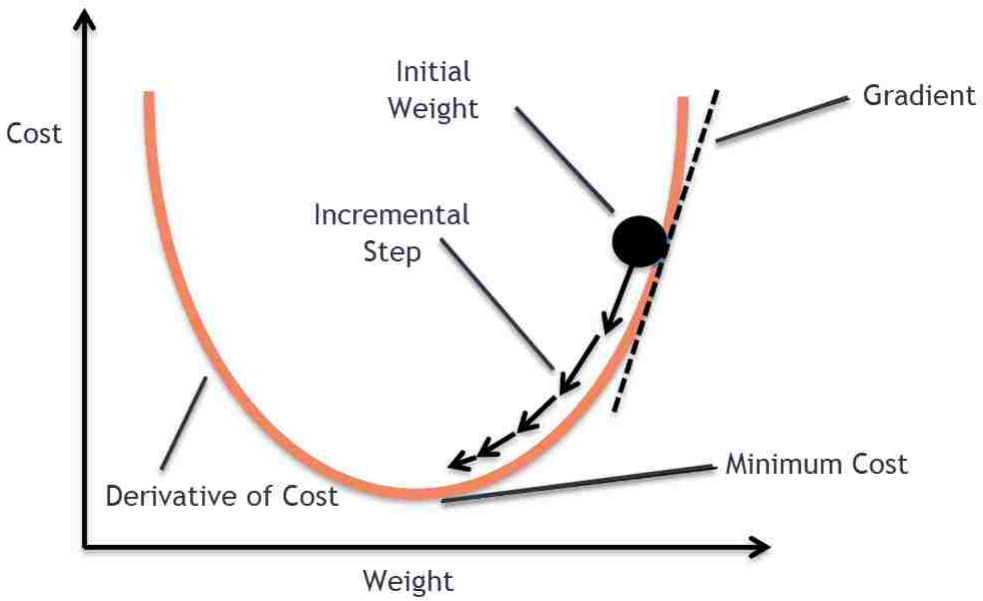

Figure 2.10: Gradient Descent Learning Curve

training data set is randomized, because as coefficients are updated after every training instance, the update will be noisy jumping all over and so will be the corresponding cost function. But by mixing up the order, it harnesses the random walk and avoids it getting distracted or stuck [Rud16].

Stochastic gradient descent classifier is a linear classifier that uses SGD for training. The model it fits can be controlled by loss parameter. By default it fits a linear support vector machines which uses hinge-loss. There are other loss functions that can be used such as modified huber, squared hinge, log and, perceptron.

2.2.6 Evaluation Methods

After building the machine learning models, we need to measure the performance of the model. The performance of the model can be measured using evaluation metrics that we will discuss in this section.

Evaluation metrics are used to measure the performance of the models that predict the output values for the previously unseen data. With the help of evaluation metrics, we can understand if the model really learnt from the training data or remembered the patterns of the training data. The power of the model can be measured appropriately only when model has been tested with previously unseen data, so usually data is initially split into training and testing data where training data is

used for creating and evaluating the model and test data is used for testing. Depending on the type of the problem whether it is classification or regression problem, evaluation metrics also vary. Since our project is a classification problem, specific metrics will be used to measure the performance of the model.

2.2.6.1 Confusion Matrix

Confusion matrix is a generally visual representation of performance of the machine learning model which can also be called as error matrix. It is also helpful in calculating other metrics too. Basically, in binary classification, using actual and predicted values, true positives, false positive, true negatives, and false negatives are calculated. All these four values are composed in confusion matrix. By using these values, we can define the performance of the model by calculating the number of times the model predicted right and number of times it failed to do so.

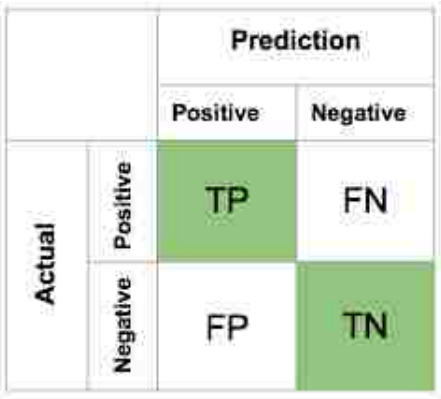

Figure 2.11: Binary Classification Confusion Matrix

True Positive: The case where the actual value was positive and the predicted value was also positive.

True Negative: The case where the actual value was negative and the model also predicted it as negative.

False Positive: The case where the actual value was negative and the model predicted it as positive.

False Negative: The case where the actual value was positive and the model predicted it as negative.

All these values are positioned in the matrix as represented in Figure 2.11

Multi-class classification case is similar with an kxk confusion matrix, where k is the number of classes.

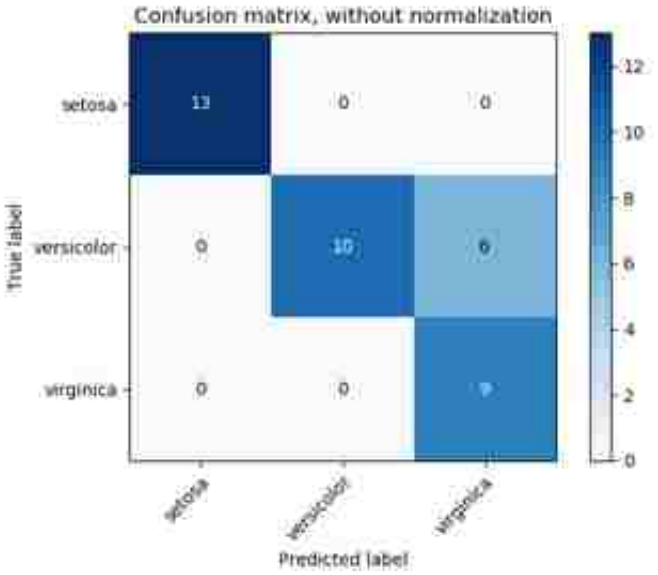

Figure 2.12: Multi-class Classification Confusion Matrix

2.2.6.2 Classification Accuracy

Classification accuracy is the proportion of the number of correct predictions to the total predictions of test data set made by the model. This is the most common evaluation metric used for classification problems. In case of binary classification, accuracy is calculated as the ratio of sum of true positives and true negatives to the total sum of the predicted population.

$$ClassificationAccuracy = \frac{TruePositives + TrueNegatives}{Sizeofpredictedpopulation} \tag{2.1}$$

But in case of multi-class classification, accuracy is calculated as the ratio of sum of all diagonal elements in the matrix to the sum of all the values in the matrix.

$$ClassificationAccuracy = \frac{sum(diagonal(ConfusionMatrix))}{sum(ConfusionMatrix)} \tag{2.2}$$

But only accuracy is not enough to evaluate the performance of the model, so we calculated other metrics too as described below.

2.2.6.3 Precision

Precision defines what proportion of positive identifications were actually accurate. It is calculated as the ratio of true positives to the number of positive predictions (true positives + false positives). It can be calculated using confusion matrix as the ratio of true positives to the sum of true positives and false positives.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{2.3}$$

In case of multi-class classification, precision is calculated as the average of precision values of all labels.

2.2.6.4 Recall

Recall defines what proportion of actual positives were identified correctly. It is calculated as the ratio of true positives to the predicted results (true positives + false negatives). It is calculated from confusion matrix as the ratio of true positives to the sum of true positives and false negatives.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{2.4}$$

In case of multi-class classification, recall is calculated as the average of recall values of all the labels.

2.2.6.5 F1 Score

F1 score is calculated by taking the harmonic mean of the precision and recall of the model's predictions. It is also known as F-measure or F-score.

$$F1Score = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{2.5}$$

F1 score can also be calculated from the confusion matrix.

2.2.6.6 Cross Validation

Generally, validation is done by splitting the training set into two parts using random sampling. The disadvantage with splitting randomly and performing validation is that it might not be fair and a specific combination of input data might always end up in the validation data which the model will not get chance to train. In that case, prediction results will be poor and the model will not perform well.

To deal with such scenarios, we can apply cross validation. In cross validation, the machine learning models are trained on n-1 subsets and validated on remaining 1 subset where n is the number of subsets formed from the training data. There are various strategies to do cross validation, in this thesis we have used k-fold cross validation as it is the most frequently used strategy.

In k-fold cross validation, the data is divided into k randomly sampled subsets. The model is fitted k different times with one of the k subsets as validation data set and the other k-1 subsets are put together as the training data set. In that case, each data point will be in validation set exactly once and gets to be in a training set k-1 times. By doing this, the model will learn the patterns from every data point so that there will not be any chance of leaving any combination of input data. The performance of the model now will definitely be better compared to just using validation data. The typical value of k is 10 as suggested by many researchers. In that case we will have 10 subsets among which, 9 subsets will be put together as the training data which is 90 % of the original data and remaining 1 subset as validation data which is 10 % of the training data [RTL08].

Chapter 3

Methodology

3.1 Data Collection

The data used in this thesis was collected from TCGA, which is a collaboration between the National Cancer Institute (NCI) and the National Human Genome Research Institute (NHGRI). To have an idea of how the models created using TCGA data will perform on a completely independent data, we collected and used a completely independent dataset from GTEX. The following sections describe more about the data and the methods used on the data to train and test the models.

3.2 Data Description

The transcriptome data collected from TCGA was exported to a .tsv file which is to be used for the models. There was also a file that gave details regarding barcode of each sample and its tissue name. Using those details, we created a final file with labels attached and in the format required to be used with the machine learning models.

Gene ID	Read Counts
ENSG00000242440.1	980
ENSG00000255170.2	0
ENSG00000234722.3	400
ENSG00000258181.1	70
ENSG00000225431.1	30

Table 3.1: Sample of Gene Ids with read counts

The raw data consists of 8077 samples when considered only 01A samples (Cancer Samples) and after including 11A samples (normal samples) the number of samples increased to 8783. Each sample represents a patient, each sample has information of 60485 gene expressions. Table 3.1 has sample of gene expressions from the data.

Using both files, we generated an independent file as required to be used in machine learning models. The range of values in the data is so wide such that the value may be any positive value including 0. As shown in Figure 3.1, to have a clear visual look of how the values vary in the datasets, we plotted graphs on both the TCGA and the GTEX datasets, which show the range of values for each sample in both datasets.

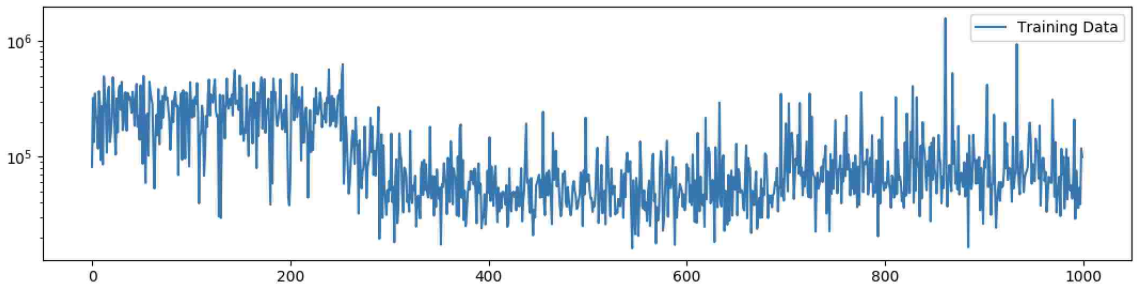

Figure 3.1: Distribution of ranges of values in Training data

3.3 Data Preparation

In this section, we will discuss the steps followed to standardize the datasets in multiple ways and explain those methods. By doing this, we are making sure that the datasets used to train the machine learning models are clean.

3.3.1 Upper Quantile Normalization

Since we collected training and test datasets from two different sources, there was huge difference in between the range of values in those two datasets. So as a part of data cleaning and pre-processing, we have done upper quantile normalization on both datasets, so that all the samples are in the same range. To do so, we got the upper quantile value for non-zero features for all samples and then divided the data with those upper quantile values.

3.3.2 Standard Scalar

Standardization is the process of re-scaling the features of the dataset such that they will have the properties of normal distribution. We used standard scalar package from sklearn in Python to perform standardization. It is important to standardize the dataset before we apply any machine learning model on it, otherwise it might effect the performance of the model. Because some of the models perform based on the variance, if the data is not scaled this distribution will be incorrect and model performance will be affected.

$$Z = \frac{X_i - \mu}{\sigma} \tag{3.1}$$

3.4 Data Resampling

As already mentioned in above sections, the dataset is highly unbalanced. Some of the labels have large number of samples whereas other labels have very few samples. In that case training of the model will not be efficient enough and the performance of the model on the test data will also be poor. To overcome this problem, we used some random sampling techniques like random under sampling, random oversampling, SMOTE, Tomek link removal and combination of SMOTE and Tomek Link Removal. We applied these techniques to the training data to make it balanced, so that training of the model will be efficient.

3.5 Data Pre-processing

Pre-processing is the most important step to be done before we apply any machine learning algorithms on the data. In this, we convert the raw data to a clean dataset. Since datasets will be from different sources we need to transform the raw data into required format. In this particular dataset, there were no missing values since the values are no counts, then the value will be 0. Since we used two different datasets, range of values should also be considered and based on that the datasets should be cleaned. Therefore, the steps required to convert the data into a clean required format will be discussed in the sections below.

3.5.1 Dimensionality Reduction

Dimensionality reduction is the process of reducing the number of features in the dataset. This is done in two ways, feature selection and feature extraction. Feature selection technique finds a

subset of the original features, which contribute maximum percentage in classifying or in making predictions. Whereas in feature extraction or feature projection technique transforms the data in the high-dimensional space to a space of fewer dimensions.

3.5.1.1 ExtraTreesClassifier

Extra trees classifier is a type of feature selection technique where it selects a subset of required features from the original set. The dataset was splitted into two parts as X (which has all features i.e., gene ids) and Y (which has the output label). The dataset X is inputted into ExtraTreesClassifier package in sklearn in Python. This model assigns score to each feature in the inputted dataset based on its contribution to predict the correct tissue label. The total of sum of scores generated by the model for each feature will add up to 1. Depending on the dataset, these scores will be distributed among the features. In our dataset, most of the features had 0 as their score. To get the final set of features to be used, we sorted the scores and just considered the features which added up to top 90 percent of the score. Then from the original dataset, we took the subset of that with only final features and then this new dataset has been used in the machine learning models.

3.5.1.2 Principal Component Analysis

Principal component analysis is the main linear technique for dimensionality reduction. It performs linear mapping of the data to a lower dimensional space in such a way that the variance of the data in the low dimensional representation is maximized. In general, covariance matrix of the data is constructed and the eigen vectors are computed. The eigen vectors that correspond to the largest eigenvalues can now be used to reconstruct a large fraction of the variance of the original data. The original space has been reduced to lower space spanned by few eigen vectors. We used PCA package from sklearn in Python by giving required parameter to number of components argument.

Another thing to consider is, if we would like to perform PCA on two different datasets, which are being used for the same purpose then they have to be reduced into lower space using the same eigen vectors. This can be possible only if we use the model used for fitting one dataset. Using that model, we can transform the another dataset into same space using same eigen vectors.

3.6 HyperParameter Tuning

In machine learning, there will be two types of parameters. Model parameters are the ones that are internal to the model and are estimated from the data. For example, support vectors in SVM, weights in neural networks, coefficients in logistic regression. The other one is hyperparameters, which are external to the model and cannot be estimated from the model. These values are to be decided by the person building the model. Every machine learning model will have various parameters to be decided, so before applying the model we need to tune the hyperparameters to get the most efficient model. This can be done by using different values for all the parameters of the model in every possible combination. We used GridSearchCV, which is a specific package in Python.

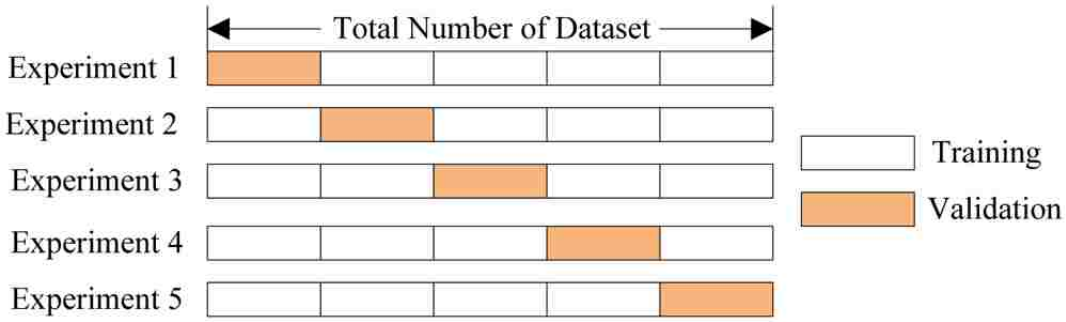

Figure 3.2: Binary Classification Confusion Matrix

GridSearchCV tries every combination given as parameter values using cross-validation. After training and validating the model with different combinations of hyperparameter values, GridSearchCV gives the best combination hyperparameter values of that model which we can use it for the testing dataset. Figure 3.2 represents cross-validation procedure.

While using GridSearchCV, it is important to give different and efficient values for all the hyperparameters of that model. This can be done only after understanding the importance of each parameter in that model. The results obtained will be considered as the best parameters for that model and for that dataset only.

3.7 Testing the selection models

In hyperparameter tuning, instead of GridSearchCV it is generally called as pipeline. After running the pipeline with the best parameters obtained, we performed predictive analysis on the test data for every model. Based on the results obtained, we determined the best model for that dataset. For measuring the performance of the model, we used metrics like precision, recall, F-1 score, and accuracy.

Chapter 4

Experiment and Results

In this section, we will discuss how the pre-processed datasets were split into training and test datasets and the reason behind it. Different machine learning models were trained on the training dataset and then hyperparameters were tuned on the validation set. With the best parameters obtained, models were fit on the entire training set and then these models were evaluated based on the predictions made on the test dataset. The results of all the models were also discussed in this section.

4.1 Data Details

The data collected from TCGA has 90 % cancerous tissue samples and 10 % normal tissue samples whereas in GTEX independent data all samples were of normal tissue samples.

The dataset collected from TCGA was used as the training and validation datasets and the one from GTEX was used as the test dataset. There were 8077 samples in the training and validation dataset and 3458 samples in the test dataset. Once the models were trained and validated, they were used to predict the labels of the samples of the test dataset. The reason behind using the GTEX dataset as the test data is to see how the model will perform on a completely independent dataset collected from different source. After applying various models on the training data, the GTEX data was used to evaluate and compare the performances of the trained models. Even though the test datset has only normal tissue samples, we wanted to see if we can classify the tissue correctly reagrdless of them being cancerous or not.

Label	Precision	Recall	F1-score	Support
ACC	0.92	1.00	0.96	11
BLCA	0.86	0.91	0.89	89
BRCA	1.00	0.97	0.98	250
CESC	0.87	0.78	0.82	68
CHOL	0.50	0.78	0.61	9
COAD	0.82	0.78	0.80	94
ESCA	0.86	0.75	0.80	32
HNSC	0.75	0.79	0.77	92
KICH	0.65	0.81	0.72	16
KIRC	0.96	0.91	0.94	121
KIRP	0.91	0.94	0.92	63
LIHC	0.98	0.94	0.96	101
LUAD	0.86	0.91	0.88	112
LUSC	0.79	0.78	0.78	107
PAAD	0.86	0.97	0.92	39
PCPG	1.00	1.00	1.00	30
PRAD	1.00	1.00	1.00	98
READ	0.41	0.46	0.43	28
SARC	0.95	0.90	0.92	58
STAD	0.90	0.94	0.92	81
THCA	0.98	0.98	0.98	111
THYM	1.00	0.97	0.98	29
UCEC	0.94	0.92	0.93	110
UCS	0.67	0.75	0.71	8
avg	0.90	0.90	0.90	1757

Table 4.1: Classification metrics of test data with both 01A and 11A samples with SVM

4.2 Experiments on Models

The work has been done in two ways regarding the splitting of datasets into train and test datasets. One way was splitting the TCGA data into both training and test datasets. Another way was considering entire TCGA dataset as the training dataset and GTEX dataset as the test dataset.

4.2.1 On TCGA data only

In this way of data usage, we have done experiments considering only cancer samples (01A samples) as one set and considering both cancer and normal samples (01A and 11A samples) as the other set.

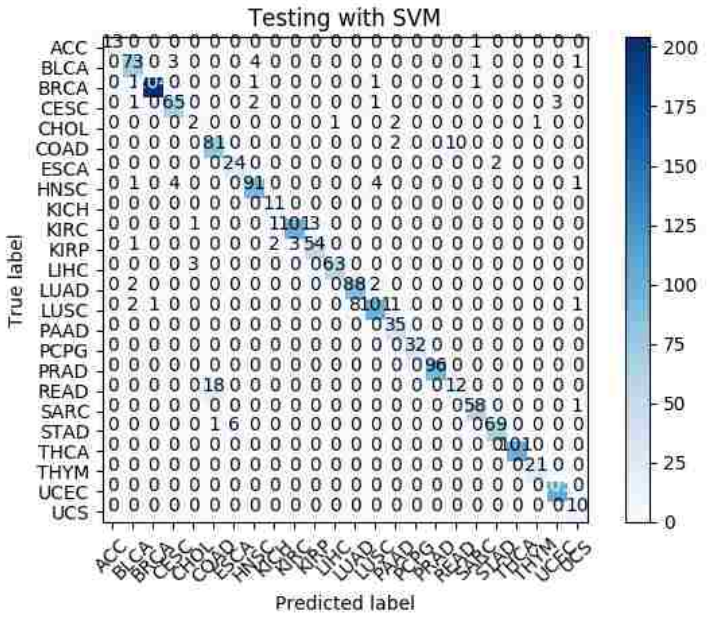

Figure 4.1: Confusion matrix on test data with SVM with only 01A samples

4.2.1.1 Support Vector Machines

The support vector machine (SVM) model used in the thesis was from the sklearn package in Python and the model was trained and validated using training data with the usage of k-fold cross-validation. Hyperparameter tuning was done using various parameters for all arguments and obtained the best parameters as average of all folds metrics.

The performance of the SVM built using training data can be measured based on the predictions made on the test data, which is previously unseen. Using those predictions on test data, the confusion matrix was generated with only 01A samples as shown in Figure 4.1. The other metrics of the dataset with both 01A and 11A samples are listed in Table 4.1.

4.2.1.2 Decision Trees

The decision tree model used in the thesis was from the sklearn package in Python, since this model has various arguments in it, to tune the hyperparameters that suits the data best. We first splitted the TCGA data into 80 % as training data and 20 % as test data. The training data was fitted to the model using GridSearchCV, which is sklearn package in Python.

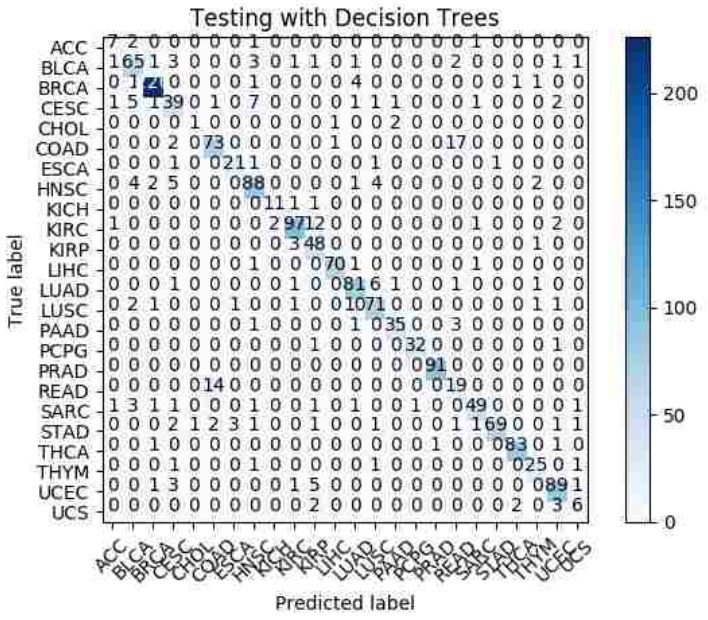

Figure 4.2: Confusion matrix on test data with decision trees with only 01A samples

GridSearchCV takes in the training data performs cross validation with all the parameters to be considered. After testing the performance of the model with all the parameters, it gives the best hyperparameters with which the model obtained the highest performance metrics.

Using the model fitted on the training data, predictions on the test data were made. Using the predictions of the test data, the confusion matrix was generated with only 01A samples in the data and is shown in Figure 4.2.

Table 4.2 shows the classification metrics calculated based on the predictions of the test data with both 01A and 11A samples.

4.2.1.3 Random Forest Classifiers

The random forest classifier model used was from the sklearn package in Python. The model was fitted using the training data. Then the trained model was used to make predictions on the validation data. This is done using GridSearchCV that is a sklearn package in Python, which gives the best parameters for the model on that data. Using the predictions on validation data, the confusion matrix was generated and is shown in Figure 4.3.

Label	Precision	Recall	F1-score	Support
ACC	0.75	0.82	0.78	11
BLCA	0.78	0.79	0.78	89
BRCA	0.96	0.96	0.96	250
CESC	0.79	0.79	0.79	68
CHOL	0.60	0.33	0.43	9
COAD	0.83	0.76	0.79	94
ESCA	0.72	0.81	0.76	32
HNSC	0.82	0.85	0.83	92
KICH	0.50	0.75	0.60	16
KIRC	0.92	0.84	0.88	121
KIRP	0.76	0.81	0.78	63
LIHC	0.93	0.94	0.94	101
LUAD	0.84	0.84	0.84	112
LUSC	0.77	0.78	0.77	107
PAAD	0.78	0.74	0.76	39
PCPG	0.88	1.00	0.94	30
PRAD	1.00	0.95	0.97	98
READ	0.44	0.50	0.47	28
SARC	0.86	0.74	0.80	58
STAD	0.88	0.85	0.87	81
THCA	0.98	0.99	0.99	111
THYM	0.85	0.97	0.90	29
UCEC	0.88	0.87	0.88	110
UCS	0.18	0.25	0.21	8
avg	0.86	0.86	0.86	1757

Table 4.2: Classification metrics of test data with both 01A and 11A samples with decision trees

The trained random forest model was used to perform predictions on the test datasets that was created from the original dataset. To avoid overfitting, we used 10-fold cross validation. Using the predictions on the test data, the confusion matrix was generated.

4.2.1.4 Stochastic Gradient Descent Classifier

The stochastic gradient descent classifier with the modified huber loss function was used was from the sklearn package in Python and the model was fitted using the training dataset. Then the trained model was used to make predictions on the validation data. To avoid overfitting, we used cross-validation so that every part of the training data acts as validation in each fit.

The trained SGD model was used to perform predictions on the test dataset. The confusion matrix generated from the predictions made on the test data is shown in Figure 4.4.

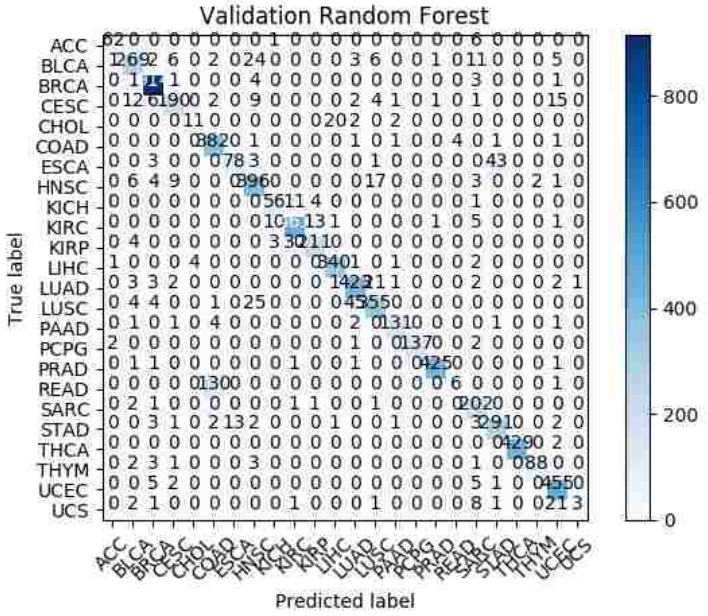

Figure 4.3: Confusion matrix on validation data with random forest with both 01A and 11A samples

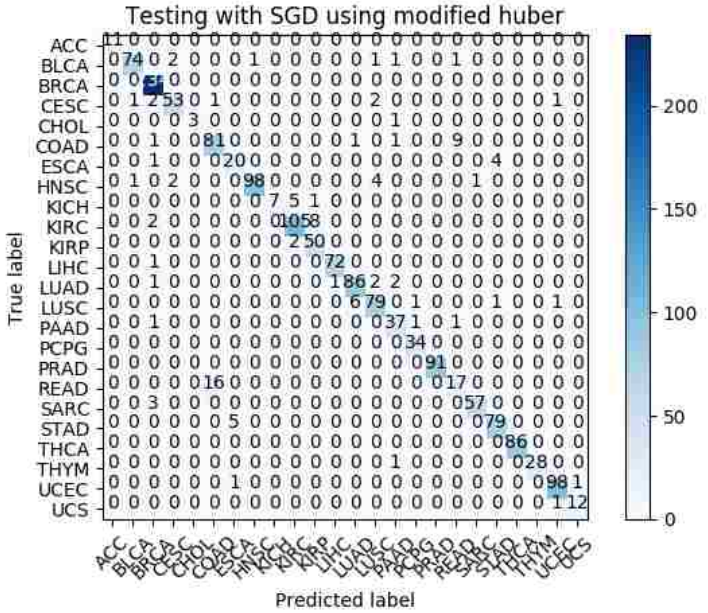

Figure 4.4: Confusion matrix on test data with SGD with only 01A samples

Label	Precision	Recall	F1-score	Support
ACC	0.92	1.00	0.96	11
BLCA	0.92	0.85	0.88	89
BRCA	0.98	0.98	0.98	250
CESC	0.82	0.79	0.81	68
CHOL	0.55	0.67	0.60	9
COAD	0.73	0.94	0.82	94
ESCA	0.80	0.50	0.62	32
HNSC	0.71	0.78	0.74	92
KICH	0.73	0.50	0.59	16
KIRC	0.85	0.85	0.85	121
KIRP	0.74	0.94	0.83	63
LIHC	0.97	0.96	0.97	101
LUAD	0.85	0.86	0.85	112
LUSC	0.79	0.74	0.76	107
PAAD	0.66	0.97	0.78	39
PCPG	0.97	0.97	0.97	30
PRAD	0.89	0.99	0.94	98
READ	0.00	0.00	0.00	28
SARC	0.94	0.52	0.67	58
STAD	0.88	0.90	0.89	81
THCA	0.99	0.99	0.99	111
THYM	0.82	0.97	0.89	29
UCEC	0.98	0.85	0.91	110
UCS	0.50	0.75	0.60	8
avg	0.86	0.86	0.85	1757

Table 4.3: Classification metrics of test data with both 01A and 11A samples with SGD

4.2.2 Testing on the GTEX Data

When working with just TCGA dataset, both the training dataset and the test dataset came from the TCGA dataset. In this case, we considered TCGA dataset as the training data and the GTEX dataset as the test data. These datasets are from different sources, and because of this there are some differences in data quality, methods used, etc. GTEX dataset is completely independent from the TCGA data, which means it is considered as previously unseen data. In subsections below, there are two types of TCGA data used, one with only 01A samples and other with both 01A and 11A samples. So included the results of the both variations of the datasets.

4.2.2.1 Support Vector Machines

The Support vector machine (SVM) used was from the sklearn package in Python. The model was fitted using the training data. Then the trained model was used to make predictions on the validation data. This is done using GridSearchCV that is a sklearn package in Python, which gives the best hyperparameters for the model on that data.

Label	Precision	Recall	F1-score	Support
ACC	0.76	0.68	0.72	205
BLCA	0.03	0.64	0.06	11
BRCA	0.99	0.77	0.87	306
CESC	0.02	0.09	0.04	11
CHOL	0.00	0.00	0.00	0
COAD	0.96	0.36	0.52	285
ESCA	0.47	0.06	0.11	1101
HNSC	0.00	0.00	0.00	0
KICH	0.00	0.00	0.00	50
KIRC	0.00	0.00	0.00	0
KIRP	0.00	0.00	0.00	0
LIHC	0.98	0.81	0.89	188
LUAD	0.00	0.00	0.00	0
LUSC	0.00	0.00	0.00	0
PAAD	0.64	0.98	0.78	264
PCPG	0.00	0.00	0.00	0
PRAD	0.77	0.64	0.70	158
READ	0.00	0.00	0.00	0
SARC	0.00	0.00	0.00	0
STAD	0.28	0.48	0.36	272
THCA	1.00	0.89	0.94	490
THYM	0.00	0.00	0.00	0
UCEC	0.00	0.00	0.00	0
UCS	0.50	0.01	0.02	117
avg / total	0.68	0.47	0.50	3458

Table 4.4: Classification metrics of test data with both 01A and 11A samples with SVM

As GTEX data was the previously unseen data by the model, it is perfect to use it as the test data. Using the trained SVM model, predictions were made on the GTEX dataset. The confusion matrix was generated from the predictions made on the test data is shown in Figure 4.5 is of data with both 01A and 11A samples.

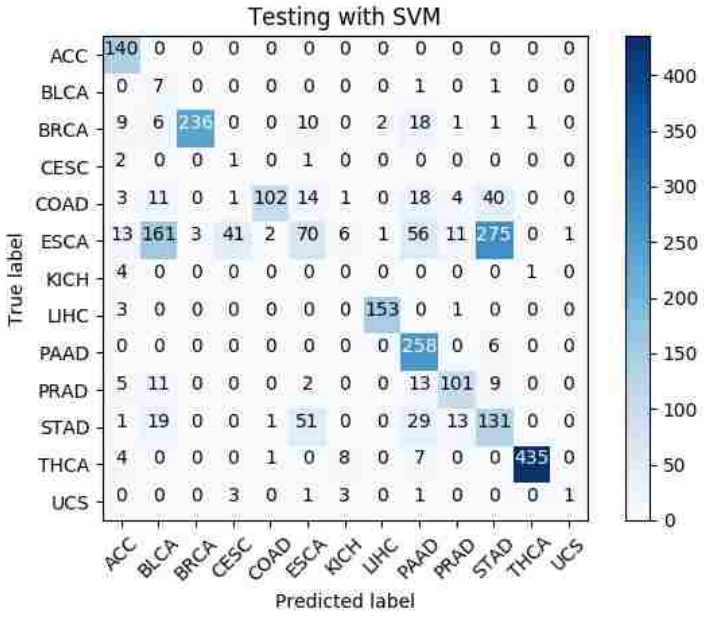

Figure 4.5: Confusion matrix on test data with SVM with both 01A and 11A samples

Table 4.4 shows other classification metrics calculated from the predictions on the test data with both 01A and 11A samples.

4.2.2.2 Decision Trees

The decision tree model used was from the sklearn package in Python. The model was fitted using the training data. Then the trained model was used to make predictions on the validation data. This is done using GridSearchCV that is a sklearn package in Python, which gives the best hyperparameters for the model on that data.

As GTEX data was the previously unseen data by the model, it is perfect to use it as the test data. Using the trained model, the predictions were made on the test dataset. Using these predictions made on the test data, the confusion matrix was generated with both 01A and 11A samples in the data and Figure obtained is 4.6.

Table 4.5 shows other classification metrics calculated from the predictions on the test data with both 01A and 11A samples.

Label	Precision	Recall	F1-score	Support
ACC	0.50	0.08	0.14	205
BLCA	0.02	0.82	0.04	11
BRCA	0.35	0.63	0.45	306
CESC	0.00	0.00	0.00	11
CHOL	0.00	0.00	0.00	0
COAD	1.00	0.54	0.70	285
ESCA	0.31	0.00	0.01	1101
HNSC	0.00	0.00	0.00	0
KICH	0.52	0.22	0.31	50
KIRC	0.00	0.00	0.00	0
KIRP	0.00	0.00	0.00	0
LIHC	0.99	0.82	0.90	188
LUAD	0.00	0.00	0.00	0
LUSC	0.00	0.00	0.00	0
PAAD	0.41	0.91	0.57	264
PRAD	0.94	0.73	0.82	158
READ	0.00	0.00	0.00	0
SARC	0.00	0.00	0.00	0
STAD	0.05	0.00	0.01	272
THCA	0.93	0.90	0.92	490
UCEC	0.00	0.00	0.00	0
UCS	0.00	0.00	0.00	117
avg / total	0.51	0.39	0.37	3458

Table 4.5: Classification metrics of test data with both 01A and 11A samples with decision trees

4.2.2.3 Random Forest Classifiers

The random forest classifier used was from the sklearn package in Python. The model was fitted using the training data. Then the trained model was used to make predictions on the validation data. This is done using GridSearchCV that is a sklearn package in Python, which gives the best hyperparameters for the model on that data.

As GTEX data was the previously unseen data by the model, it is perfect to use it as the test data. Using the trained model, the predictions were made on the test dataset. Using the predictions on the test data, the confusion matrix was generated and is shown in Figure 4.7 is of data with both 01A and 11A samples.

Table 4.6 shows other classification metrics calculated from the predictions on the test data

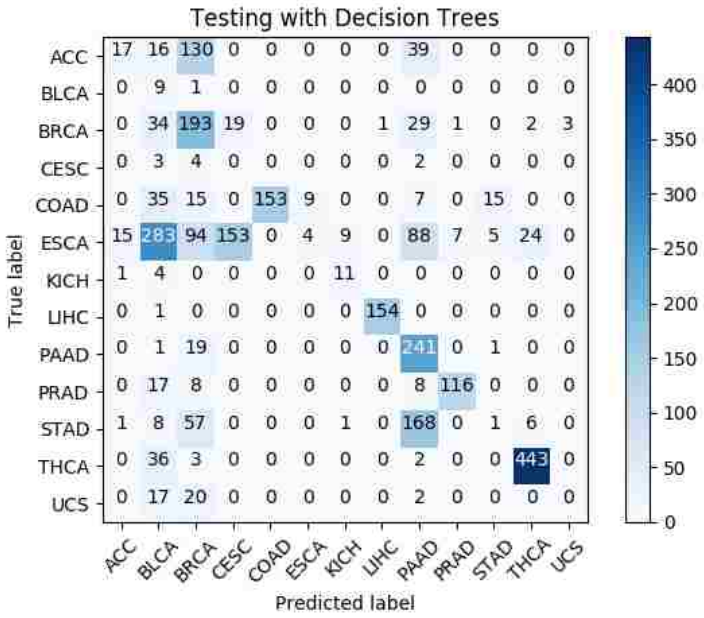

Figure 4.6: Confusion matrix on test data with both 01A and 11A samples

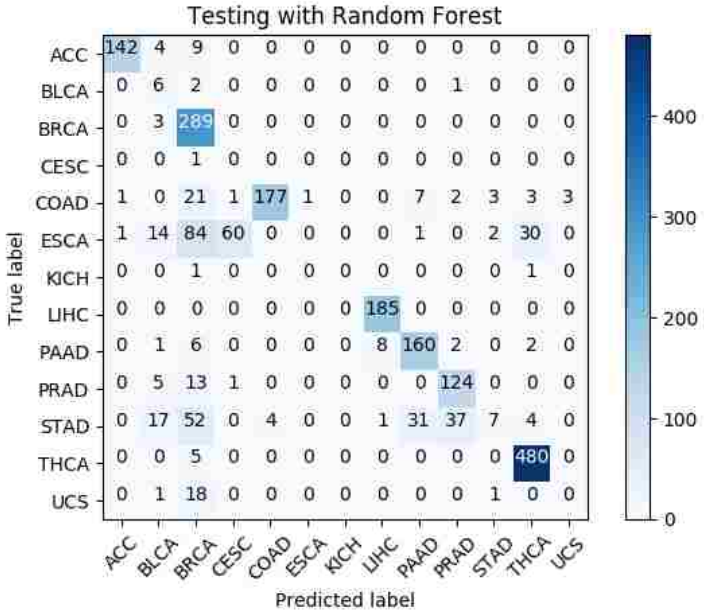

Figure 4.7: Confusion matrix on test data with both 01A and 11A samples

with both 01A and 11A samples.

Label	Precision	Recall	F1-score	Support
ACC	0.99	0.69	0.81	205
BLCA	0.12	0.55	0.19	11
BRCA	0.58	0.94	0.72	306
CESC	0.00	0.00	0.00	11
CHOL	0.00	0.00	0.00	0
COAD	0.98	0.62	0.76	285
ESCA	0.00	0.00	0.00	1101
HNSC	0.00	0.00	0.00	0
KICH	0.00	0.00	0.00	50
KIRC	0.00	0.00	0.00	0
KIRP	0.00	0.00	0.00	0
LIHC	0.95	0.98	0.97	188
LUAD	0.00	0.00	0.00	0
LUSC	0.00	0.00	0.00	0
PAAD	0.80	0.61	0.69	264
PCPG	0.00	0.00	0.00	0
PRAD	0.75	0.78	0.77	158
READ	0.00	0.00	0.00	0
SARC	0.00	0.00	0.00	0
STAD	0.54	0.03	0.05	272
THCA	0.92	0.98	0.95	490
UCEC	0.00	0.00	0.00	0
UCS	0.00	0.00	0.00	117
avg / total	0.51	0.45	0.45	3458

Table 4.6: Classification metrics of test data with both 01A and 11A samples with random forests

4.2.2.4 Stochastic Gradient Descent Classifier

The stochastic gradient descent model with modified huber loss function used was from the sklearn package in Python. The model was fitted using the training data. Then the trained model was used to make predictions on the validation data. This is done using GridSearchCV that is a sklearn package in Python, which gives the best hyperparameters for the model on that data.

As GTEX data was the previously unseen data by the model, it is perfect to use it as the test data. Using the trained SGD model, the predictions were made on test dataset. The confusion matrix was generated from those predictions made on the test data is shown in Figure 4.8 is of data with both 01A and 11A samples.

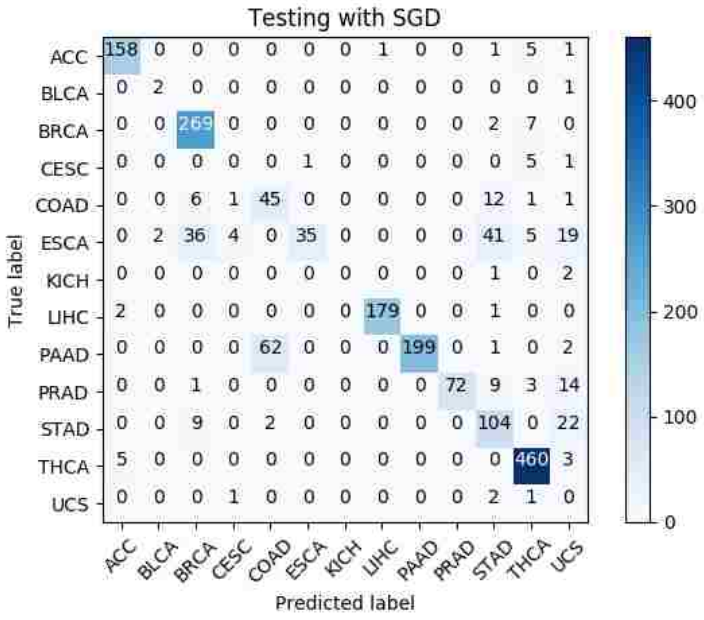

Figure 4.8: Confusion matrix on test data with both 01A and 11A samples

4.3 Comparison of Models

In this section, we will compare the results of support vector machines, decision trees, random forest classifiers, stochastic gradient descent. Each model is tested on the data which is pre-processed in multiple ways. We have also used two types of datasets, one with only 01A samples and other with both 01A and 11A samples.

Model Used	Accuracy
SVM with data preprocessed with StandardScalar, Smote, Tomek	93.97
SVM with data preprocessed and considered only ¿100	94.17
Decision Trees with standard scalar, Smote, Tomek	95.46
Decision Trees	93.13
K Nearest Neighbors	77.4
Naive Bayes Classifier	87.4
Support Vector Machines	90.4

Table 4.7: Results of various models with TCGA dataset only

4.3.1 Results along with GTEX data comparison

All the results mentioned in this section were obtained from the predictions of test data using trained models. We had trained the machine learning models with TCGA data considering it as training data and used GTEX data as test data. Table 4.7 has the results obtained based on predictions of test data using only the TCGA dataset.

We considered GTEX data as the test dataset which is completely independent from the TCGA dataset, we had implemented the machine learning models with various strategies. The table 4.8 has the results of the test data.

Model Used	Validation Accuracy	Test Accuracy
raw data with only 01A samples		
Support Vector Machines	91.23	47.28
Random Forest Classifiers	89.76	45.40
Decision Trees	87.34	38.80
Stochastic Gradient Descent	88.54	44.04
Divide by sum strategy on only 01A samples		
Support Vector Machines	14.47	8.84
Random Forest Classifiers	88.9	38.98
Decision Trees	82.8	42.56
Stochastic Gradient Descent	64.3	7.57
Upper quantile normalization strategy on only 01A samples		
Support Vector Machines	89.1	47.59
Random Forest Classifiers	90.34	38.98
Decision Trees	83.35	37.56
Stochastic Gradient Descent	91.42	58.41
Upper quantile data with both 01A and 11A samples		
Support Vector Machines	88.04	8.27
Random Forest Classifiers	89.4	10.93
Decision Trees	82.4	5.81
Stochastic Gradient Descent	89.6	51.21
Raw data with both 01A and 11A samples		
Support Vector Machines	87.42	47.28
Random Forest Classifiers	88.72	45.40
Decision Trees	83.16	38.84
Stochastic Gradient Descent	89.26	44.04

Table 4.8: Results of various models with TCGA and GTEX datasets

Chapter 5

Conclusion and Future Work

In this thesis, we used machine learning models to predict the tissue of a given sample. For this, we collected data from two major sources: TCGA and GTEX, which has read counts of various gene expressions for various samples. We performed data pre-processing to clean the data and then applied dimensionality reduction to choose the best features among 60K features.

The pre-processed data was used to train the machine learning models support vector machines, decision trees, random forest classifier, stochastic gradient descent classifier. Since every machine learning model has various hyperparameters which are to be set when building a model using training data, to tune those parameters used GridSearchCV, which tries every combination of given input with model. Using those best parameters, we built and trained models, and those models were evaluated and compared using the classification metrics based on the predictions of the test data. Among all models, stochastic gradient descent with modified huber loss function and SVM performed better in predicting the labels of the test data.

Since we have collected datasets from two different sources, there might be differences in terms of data quality, count length considered, methods used, etc. With all these differences in training and testing model, the trained model was able to predict only some of the test labels correctly.

As a part of future work, we can work on choosing appropriate dimensionality reduction methods which to be use with biological data. We can also work on making two different datasets from different sources similar to each other so that a machine learning model can perform well on a dataset when it was trained using the other dataset.

Bibliography

[Can19] https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga. 2019.

[CBHP02] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and Kegelmeyer W Philip. Synthetic minority over sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[EAMAS16] Tusneem A.M Elhassan, Futwan A Al-Mohanna, Mahmoud Aljurf, and Mohamed M Shoukri. Classification of Imbalance Data using Tomek Link (T-Link) Combined with Random Under-sampling (RUS) as a Data Reduction Method. *Journal of Informatics and Data Mining*, 1(2):12–14, 2016.

[Faw16] Tom Fawcett. Learning from Imbalanced Classes. *https://www.svds.com/learning-imbalanced-classes/*, 2016.

[Gup17] Prashant Gupta. https://towardsdatascience.com/decision-trees-in-machine-learning. 2017.

[GZK19] Dincer Goksuluk, Gokmen Zararsiz, and Selcuk Korkmaz. Machine Learning Interface to RNASeq Data. *Computer Methods and Programs in Biomedicine*, 175:223–231, 2019.

[HL13] Feng Hu and Hang Li. A Novel Boundary Oversampling Algorithm Based on Neighborhood Rough Set Model. *Mathematical Problems in Engineering*, 2013:1–10, 2013.

[HMS01] David J Hand, Heikki Mannila, and Padhraic Smyth. Principles of Data Mining. *MIT Press*, 10:546, 2001.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The Elements of Statistical Learning. *Springer Series in Statistics*, 2:9–39, 2009.

[KG18] Prabhjot Kaur and Anjana Gosain. Comparing the Behavior of Oversampling and Undersampling Approach of Class Imbalance Learning by Combining Class Imbalance Problem with Noise. *ICT Based Innovations*, pages 23–30, 2018.

[MH17] Roser Max and Ritchie Hannah. Cancer. *ourworldindata.org/cancer*, 2017.

[Mit97] Thomas M Mitchell. Machine Learning. *McGraw Hill Inc., New York, NY, USA, 1 edition*, 1997.

[MPGV09] Jordi Munoz, Antonio J Plaza, Anthony Gualtieri, and Camps J Valls. Parallel implementations of SVM for earth observation. *Advances in Parallel Computing*, 17:14–22, 2009.

[PAWV18] Dragutin Petkovic, Russ Altman, Mike Wong, and Arthur Vigil. Improving the explainability of Random Forest classifier - user centered approach. *Biocomputing 2018*, pages 204–215, 2018.

[Pol17] Saimadhu Polamuri. http://dataaspirant.com/2017/06/26/random-forest-classifier-python-scikit-learn/. 2017.

[Qui86] John Ross Quinlan. Induction of Decision Trees . *Journals of Biosystems Engineering*, pages 81–106, 1986.

[RTL08] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross Validation. *Encyclopedia of Database Systems*, 10:532–538, 2008.

[Rud16] Sebastian Ruder. An overview of gradient descent optimization algorithms. *Insight Centre for Data Analytics*, 1:3–7, 2016.

[SASK12] Chafik Samir, Pierre Antoine Absil, Anuj Srivastava, and Eric Klassen. A Gradient-Descent Method for Curve Fitting on Riemannian Manifolds. *Foundations of Computational Mathematics*, 12(1):49–73, 2012.

[SLSH17] Xiaofeng Sun, Xiangguo Lin, Shushan Shen, and Zhanyi Hu. High-Resolution Remote Sensing Data Classification over Urban Areas Using Random Forest Ensemble and Fully Connected Conditional Random Field. *ISPRS International Journal of Geo-Information*, 6(8):245, 2017.

[SS16] Asif Salekin and John Stankovic. Detection of Chronic Kidney Disease and Selecting Important Predictive Attributes. *IEEE International Conference on Healthcare Informatics (ICHI)*, 10:262–270, 2016.

[UTF+17] Daniel Urda, Julio Montes Torres, Leonardo Franco, Francesc Moreno, and Jose Jerez. Deep Learning to Analyze RNA-Seq Gene Expression Data. *Springer International Publishing AG*, pages 50–59, 2017.

[WW98] Jason Weston and Chris Watkins. Multi-class support vector machines. *European Symposium on Artificial Neural Networks*, pages 3–8, 1998.

[ZGK+17] Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Godze Erturk Zararsiz, Izzet Parug Duru, and Ahmet Ozturk. A comprehensive simulation study on classification of RNA-Seq data. *PLOS ONE*, 12(8):1–5, 2017.

Curriculum Vitae

Graduate College
University of Nevada, Las Vegas

Lohitha Chintham Reddy
Email: chinthamlohitha@gmail.com

Degrees:

Bachelor of Technology in Computer Science 2017
Jawaharlal Nehru Technological University, Anantapur, India

Master of Science in Computer Science, 2019
University of Nevada, Las Vegas, USA

Thesis Title: Machine Learning Prediction of Primary Tissue Origin of Cancer from Gene Expression Read Counts

Thesis Examination Committee:

Chairperson, Dr. Fatma Nasoz, Ph.D.
Committee Member, Dr. Ajoy Datta, Ph.D.
Committee Member, Dr. Kazem Taghva, Ph.D.
Graduate Faculty Representative, Dr. Mira Han, Ph.D.