TOWARDS IMPROVED CANCER DIAGNOSIS BASED ON MACHINE LEARNING TECHNIQUES

A Project progress Report submitted in the partial fulfilment of the requirements for the award of degree

of

Bachelor of Technology

In

Computer Science and Engineering

by

Aditya Anand
DE/16/CS/06

Suresh Gurung
DE/16/CS/03

Under the Guidance of

Mr. Aswini Kumar Patra

Assistant Professor

Department of Computer Science and Engineering,
North Eastern Regional Institute of Science and Technology
Deemed to be University(MHRD),Govt. of India
Nirjuli-791109,Arunachal Pradesh,India

CERTIFICATE OF APPROVAL

This is to certify that the project work entitled "**TOWARDS IMPROVED CANCER DIAGNOSIS BASED ON MACHINE LEARNING TECHNIQUES**" is hereby approved a bona fide work of study as an engineering subject, carried out jointly by:

ADITYA ANAND (DE/16/CS/06)

SURESH GURUNG(DE/16/CS/03)

And presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is hereby understood by this approval that the undersigned do not endorse or approve any statement made, opinion expressed or conclusions drawn therein, but approves the report only for the purpose for which it has been submitted.

Mr. Aswini Kumar Patra
(Project Guide)

Mr. Pradeep Khamboj
(Project Coordinator)

Mrs. Margaret Kathing
(Head of Department)
Computer Science and Engineering
NERIST

CANDIDATE’S DECLARATION

We hereby declare that the project work entitled “**TOWARDS IMPROVED CANCER DIAGNOSIS BASED ON MACHINE LEARNING TECHNIQUES**” submitted to North Eastern Regional Institute of Science and Technology, Arunachal Pradesh, is a record of an original work done by us under the guidance of **Mr. Aswini Kumar Patra**, Assistant Professor, Department of computer Science and Engineering, North Eastern Regional Institute of Science and Technology, and this project work is submitted in the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Aditya Anand	Suresh Gurung
DE/16/CS/06	DE/16/CS/03

ABSTRACT

The diagnosis of cancer has been a necessity in the field of cancer research. The detection of cancer from gene expression data is still a challenge due to its high dimensionality and complexity. The goal of the project is to improve cancer diagnosis based on Machine Learning techniques. Mostly, the medical experts carry out tests on the cells' proteins, DNA, RNA to diagnose cancer. These cells contain countless number of genes to be examined for cancer diagnosis. It involves a lot of human effort. Thus, machine based algorithms are designed to get activities of genes of numerous cells and hence classification is performed. A variety of techniques like Artificial Neural Networks(ANNs), Support Vector Machines(SVMs),Random Forest have been used to build predictive cancer models. The main approach is to deal with cancer data and develop more generalized versions of cancer classifiers. In this work, we present the working of Deep Learning Classifiers, Support Vector Machine Classifiers, Adaboost Classifiers and Random Forest Classifiers with PCA. We have tried out three different cancer datasets, namely, Diffuse Large B-Cell Lymphoma dataset, Prostate Cancer dataset and Breast Cancer dataset.

TABLE OF CONTENTS

1. Introduction

Dealing with algorithms related to improvement of cancer diagnosis has been one of the most challenging works till date. Cancer is nearly always diagnosed by an expert who has looked at cell or tissue samples under a microscope. In some cases, tests done on the cells' proteins, DNA, and RNA can help tell doctors if there's cancer. These test results are very important when choosing the best treatment options. These cell or tissue samples are actually composed of activities of number of genes inside the particular sample of cell or tissue. Since, there are almost countless genes, inside a cell, to be examined. Thus, the examination of cells requires a lot of human effort. The accurate prediction of the disease poses a challenge to the doctor. To reduce this human effort and accurately predict the disease , machine based algorithms are designed to get activities of genes of numerous cells and hence classification is performed. The machine based algorithms requires a set of gene expression data of a cell to be examined as input and gives the output class of cancer to which the examinee cell belongs.
A tissue contains many cells and each cell contains many genes. These genes produce RNA from DNA by the process of transcription. The amount of RNA produced from transcription of a gene gives the level of activity of genes. This level of activity of gene is known as Gene Expression. The gene expressions of a cell are stored in the form of a micro-array. Each element of the micro-array works as a feature for a cell. For example, if there are ‘n’ number of genes, rather say there are ‘n’ elements in the micro-array data, of each cell, then we get ‘n’ number of attributes for classification of each cell.
The main approach is to deal with cancer data and develop more generalized versions of cancer classifiers.
Over the past years, there has been a continuous research in the field of cancer .Various machine learning models have been used for diagnosis and prognosis of cancer. Moreover, new strategies have been developed for the early prediction of cancer treatment. The analysis of gene expression has the potential to lead to significant biological discoveries. The machine based algorithms uses a set of gene expression data of a cell to be examined as input and it gives the output class of cancer to which the examinee cell belongs to. The high dimensionality and the noise associated with these data presents a challenge.

Our main approach is to deal with cancer datasets and develop more generalized versions of cancer classifiers. For this, we explored various Feature Selection Methods and this includes Univariate Feature Selection, Variance Threshold Feature Selection, Linear SVC, Tree Based Feature Selection and Pipelined Feature Selection. Four kinds of classifiers namely Deep Learning Model (DLM) Classifier, Support Vector Machine (SVM) Classifier, Random Forest Classifier and AdaBoost Classifier are used to classify the cancer cells. The cancer datasets used for the classification are Diffuse Large B-Cell Lymphoma dataset, Prostate Cancer dataset and Breast Cancer dataset. Feature Selection, in our case, was not much useful for any of the three datasets with any of the four classifiers. Various training and testing methods are used with different multiple combinations. Still, no satisfactory result was obtained. We, then, moved on to PCA

Dimensionality Reduction, from where we collected some Principal Components. On classification of the three datasets by using the four classifiers separately with PCA, we observed that SVM with PCA outperformed for Diffuse Large B-Cell Lymphoma dataset with a cross-validation score of nearly 96% with (+/–) 16% standard deviation. Also, the cross validation score of AdaBoost Classifier with PCA for the same dataset was nearly 92% with (+/–) 16% standard deviation.

2. Literature Survey

1. Rasool Fakoor et. al. proposed an algorithm "*Using deep learning to enhance cancer diagnosis and classification*". ICML, June-2013.

   Using automated computer tools and in particular machine learning to facilitate and enhance medical analysis and diagnosis is a promising and important area. This paper shows that how unsupervised feature learning can be used for cancer detection and cancer type analysis from gene expression data. The main advantage of the proposed method over previous cancer detection approaches is the possibility of applying data from various types of cancer to automatically form features which help to enhance the detection and diagnosis of a specific one. The technique is here applied to the detection and classification of cancer types based on gene expression data. This domain shows that the performance of this method is better than that of previous methods, therefore promising a more comprehensive and generic approach for cancer detection and diagnosis.

2. Runxuan Zhang et. al. suggested an algorithm for "*Multicategory Classification Using an Extreme Learning Machine for Microarray Gene Expression Cancer Diagnosis*." IEEE/ACM Transactions on Computational Biology and Bioinformatics, Vol. 4, No. 3, July-September 2007.

   In this paper, the recently developed Extreme Learning Machine (ELM) is used for directing multicategory classification problems in the cancer diagnosis area. ELM avoids problems like local minima, improper learning rate and overfitting commonly faced by iterative learning methods and completes the training very fast. The approach was to evaluate the multicategory classification performance of ELM on three benchmark microarray data sets for cancer diagnosis, namely, the GCM data set, the Lung data set, and the Lymphoma data set. The results indicate that ELM produces comparable or better classification accuracies with reduced training time and implementation complexity compared to artificial neural networks methods like conventional back-propagation ANN, Linder's SANN, and Support Vector Machine methods like SVM-OVO and Ramaswamy's SVM-OVA. ELM also achieves better accuracies for classification of individual categories.

3. Lijuan Zhong, Data Science M.S. Program, UMN from Institute for Health Informatics, UMN. Lecture Notes: "*Deep Learning for Micro-Array Based Cancer Classification.*"

   In this paper, the main objective is to apply deep learning methods on gene expression microarray datasets spanning various cancer domains and prediction tasks, and also to compare performance of deep learning with previous “best-in-class” learners: Is Deep Learning a viable or superior alternative to current classifiers?. Prior study evaluated combinations of classification, gene selection algorithms and cross-validation designs across many array-based cancers. Results suggest multicategory support vector machines (MC-SVMs) are a dominant method for accurate cancer diagnosis and prognosis from gene expression data.

4. Padideh Danaee et. al. proposed “*A Deep Learning Approach For Cancer Detection and Relevant Gene Identification*”. Pac Symp Biocomput - December 2016.

   This paper presents a deep learning approach to cancer detection, and to the identification of genes critical for the diagnosis of breast cancer. First, a Stacked Autoencoder (SAE) is used to deeply extract functional features from high dimensional gene expression profiles. Next, the performance of the extracted representation is evaluated through supervised classification models to verify the usefulness of the new features in cancer detection. Lastly, a set of highly interactive genes are identified by analysing the SDAE connectivity matrices. Results and analysis illustrate that these highly interactive genes could be useful cancer biomarkers for the detection of breast cancer that deserve further studies.

5. Konstantina Kourou et. al. proposed *“Machine Learning application in cancer prediction and prognosis”*. Computational and Structural Bio technology Journal 13(2015) 8–17.

   Cancer has been characterized as a heterogeneous disease consisting of many different subtypes. The early diagnosis and prognosis of cancer type have become a necessity in cancer research, as it can facilitate the subsequent clinical management of patients. The work here presents a review of recent ML approaches employed in the modelling of cancer progression. The predictive model discussed here are based on various supervised ML techniques as well as different input features and data samples, given the growing trend and the application of ML methods in cancer research.

6. Mathew J. Garnettet. al. proposed “*Systematic identification of genomic markers of drug sensitivity in cancer cells*”. Nature. PMC 2012 September 29.

   To uncover new biomarkers of sensitivity and resistance to cancer therapeutics, we screened a panel of several hundred cancer cell lines, which represent much of the tissue-type and genetic diversity of human cancers, with 130 drugs under clinical and preclinical investigation. In aggregate, we found mutated cancer genes were associated with cellular response to most currently available cancer drugs. Classic oncogene addiction paradigms were modified by additional tissue-specific or expression biomarkers, and some frequently mutated genes were associated with sensitivity to a broad range of therapeutic agents. Unexpected relationships were revealed, including the marked sensitivity of Ewing's sarcoma cells harbouring the EWS-FLI1 gene translocation to PARP inhibitors. By linking drug activity to the functional complexity of cancer genomes, systematic pharmacogenomics profiling in cancer cell lines provides a powerful biomarker discovery platform to guide rational cancer therapeutic strategies.

7. Michael P. Menden et. al. proposed "*Machine Learning Prediction of Cancer Cell Sensitivity to Drugs Based on Genomic and Chemical Properties*". PLos One, April 2013 | Volume 8 | Issue 4 | e61318.

   Various computational approaches have been proposed to predict sensitivity based on genomic features, while others have used the chemical properties of the drugs to ascertain their effect. In an effort to integrate these complementary approaches, this paper develops machine learning models to predict the response of cancer cell lines to drug treatment, quantified through IC50 values, based on both the genomic features of the cell lines and the chemical properties of the considered drugs. Models predicted IC50 values in a 8-fold cross-validation and an independent blind test with coefficient of determination R2 of 0.72 and 0.64 respectively. Furthermore, models were able to predict with comparable accuracy (R2 of 0.61) IC50s of cell lines from a tissue not used in the training stage. Our in silico models can be used to optimise the experimental design of drug-cell screenings by estimating a large proportion of missing IC50 values rather than experimentally measuring them. The implications of our results go beyond virtual drug screening design: potentially thousands of drugs could be probed in silico to systematically test their potential efficacy as anti-tumour agents based on their structure, thus providing a computational framework to identify new drug repositioning opportunities as well as ultimately be useful for personalized medicine by linking the genomic traits of patients to drug sensitivity.

8. Zhihua Cai et. al. proposed *"Classification of lung cancer using ensemble-based feature selection and machine learning methods".* Molecular BioSystems.

   Lung cancer is one of the leading causes of death worldwide. There are three major types of lung cancers, non-small cell lung cancer (NSCLC), small cell lung cancer (SCLC) and carcinoid. NSCLC is further classified into Lung adenocarcinoma (LADC), squamous cell lung cancer (SQCLC) as well as large cell lung cancer. Many previous works demonstrated that DNA methylation has emerged as potential lung cancer-specific biomarkers. However, whether there exists a set of DNA methylation markers simultaneously distinguishing such three types of lung cancers remains elusive. In the present study, ROC (Receiving operating Curve) ,RFs(random forests) and mRMR (Maximum Relevancy and Minimum Redundancy) were proposed to capture the unbiased, informative as well as compact molecular signatures followed by machine learning methods to classify LADC,SQCLC and SCLC. As a result, a panel of 16 DNA methylation markers exhibits an ideal classification power with an accuracy of 86.54%, 84.6% and a recall 84.37%, 85.5% in the leave-one-out cross-validation (LOOCV) and independent data set test experiments, respectively. Besides, comparison results indicate that ensemble-based feature selection methods outperform individual ones when combined with incremental feature selection (IFS) strategy in terms of the informative and compact property of features. Taken together, results obtained suggest the effectiveness of ensemble based feature selection approach and the possible existence of a common panel of DNA methylation markers among such three types of lung cancer tissue, which would facilitate clinical diagnosis and treatment.

9. Daniele Ravi et. at. Suggested a method for "*Deep Learning in health informatics"*. IEEE Journal of Biomedical and Health Informatics, Vol. 21, No. 1, January 2017.

   With a massive influx of multimodality data, the role of data analytics in health informatics has grown rapidly in the last decade. This has also prompted increasing interests in the generation of analytical, data driven models based on machine learning in health informatics. Deep learning, a technique with its foundation in artificial neural networks, is emerging in recent years as a powerful tool for machine learning, promising to reshape the future of artificial intelligence. Rapid improvements in computational power, fast data storage, and parallelization have also contributed to the rapid uptake of the technology in addition to its predictive power and ability to generate automatically optimized high-level features and semantic interpretation from the input data. This article presents a comprehensive up-to-date review of research employing deep learning in health informatics, providing a critical analysis of the relative merit, and potential pitfalls of the technique as well as its future outlook. The paper mainly focuses on key applications of deep learning in the fields of translational bioinformatics, medical imaging, pervasive sensing, medical informatics, and public health.

10. Herve Abdi et. al. described about *"Princiapal Component Analysis"* . 2010 John Wiley & Sons, Inc. WIREs Comp Stat 2010 2 433–459.

    Principal component analysis (PCA) is a multivariate technique that analyzes a data table in which observations are described by several inter-correlated quantitative dependent variables. Its goal is to extract the important information from the table, to represent it as a set of new orthogonal variables called principal components, and to display the pattern of similarity of the observations and of the variables as points in maps. The quality of the PCA model can be evaluated using cross-validation techniques such as the bootstrap and the jackknife. PCA can be generalized as correspondence analysis (CA) in order to handle qualitative variables and as multiple factor analysis (MFA) in order to handle heterogeneous sets of variables. Mathematically, PCA depends upon the eigen-decomposition of positive semidefinite matrices and upon the singular value decomposition (SVD) of rectangular matrices.

11. Stephen Marsland’s *Machine Learning –An Algorithmic Perspective 2nd Edition.*

12. Jiawei Han and Micheline Kamber’s *Data Mining Concepts and Techniques 2nd Edition.*

3. Background and Preliminaries

3.1 MACHINE LEARNING

Machine learning is about making computers modify or adapt their actions (whether these actions are making predictions, or controlling a robot) so that these actions get more accurate, where accuracy is measured by how well the chosen actions reflect the correct ones. It is only over the past decade or so that the inherent multi-disciplinary of machine learning has been recognised. It merges ideas from neuroscience and biology, statistics, mathematics, and physics, to make computers learn. There is a fantastic existence proof that learning is possible. Another thing that has driven the change indirection of machine learning research is data mining, which looks at the extraction of useful information from massive datasets (by men with computers and pocket protectors rather than pickaxes and hard hats), and which requires efficient algorithms, putting more of the emphasis back onto computer science. The computational complexity of the machine learning methods will also be of interest to us since what we are producing is algorithms. It is particularly important because we might want to use some of the methods on very large datasets, so algorithms that have high degree polynomial complexity in the size of the dataset (or worse) will be a problem. The complexity is often broken into two parts: the complexity of training, and the complexity of applying the trained algorithm. Training does not happen very often, and is not usually time critical, so it can take longer. However, we often want a decision about a test point quickly, and there are potentially lots of test points when an algorithm is in use, so this needs to have low computational cost.

There are various types of machine learning:

- **Supervised learning**: A training set of examples with the correct responses (targets) is provided and, based on this training set, the algorithm generalises to respond correctly to all possible inputs. This is also called learning from exemplars.
- **Unsupervised learning:** Correct responses are not provided, but instead the algorithm tries to identify similarities between the inputs so that inputs that have something in common are categorised together. The statistical approach to unsupervised learning is known as density estimation.
- **Reinforcement learning:** This is somewhere between supervised and unsupervised learning. The algorithm gets told when the answer is wrong, but does not get told how to correct it. It has to explore and try out different possibilities until it works out how to get the answer right. Reinforcement learning is sometime called learning with a critic because of this monitor that scores the answer, but does not suggest improvements.
- **Evolutionary learning:** Biological evolution can be seen as a learning

process: biological organisms adapt to improve their survival rates and chance of having offsprings in their environment. We'll look at how we can model this in a computer, using an idea of fitness, which corresponds to a score for how good the current solution is.

3.2 CLASSIFICATION

The classification problem consists of taking input vectors and deciding which of N classes they belong to, based on training from exemplars of each class. The most important point about the classification problem is that it is discrete each example belongs to precisely one class, and the set of classes covers the whole possible output space. These two constraints are not necessarily realistic; sometimes examples might belong partially to two different classes. In addition, there are many places where we might not be able to categorise every possible input. For example, consider a vending machine, where we use a neural network to learn to recognise all the different coins. We train the classifier to recognise all New Zealand coins, but what if a British coin is put into the machine? In that case, the classifier will identify it as the New Zealand coin that is closest to it in appearance, but this is not really what is wanted: rather, the classifier should identify that it is not one of the coins it was trained on. This is called **novelty detection.** Let's consider how to set up a coin classifier. When the coin is pushed into the slot, the machine takes a few measurements of it. These could include the diameter, the weight, and possibly the shape, and are the features that will generate our input vector. In this case, our input vector will have three elements, each of which will be a number showing the measurement of that feature (choosing a number to represent the shape would involve an encoding, for example that 1=circle, 2=hexagon, etc.). Of course, there are many other features that we could measure. If our vending machine included an atomic absorption spectroscope, then we could estimate the density of the material and its composition, or if it had a camera, we could take a photograph of the coin and feed that image into the classifier. The question of which features to choose is not always an easy one. We don't want to use too many inputs, because that will make the training of the classifier take longer, but we need to make sure that we can reliably separate the classes based on those features. For example, if we tried to separate coins based only on colour, we wouldn't get very far, because the 20¢ and 50¢ coins are both silver and the $1 and $2 coins both bronze. However, if we use colour and diameter, we can do a pretty good job of the coin classification problem for NZ coins. There are some features that are entirely useless. For example, knowing that the coin is circular doesn't tell us anything about NZ coins, which are all circular (see Figure 3.1).

Fig. 3.1: The New Zealand Coins

- **Binary Classification:** Binary classification is used to predict one of two possible outcomes. A two class problem (binary problem) has possibly only two outcomes: “yes or no”/ “success” or “failure”, and is much more known as a Bernoulli trial.

  Given a collection of objects let us say we have the task to classify the objects into two groups based on some feature(s). For example, let us say given some pens and pencils of different types and makes, we can easily separate them into two classes, namely pens and pencils. This seemingly trivial task, if we take a moment would seem to involve a lot of underlying computation. The question to ask would be what is our basis for classification? And how perhaps might we incorporate the principle to allow for efficient classification by *'intelligent algorithms'*.

  Binary Classification would generally fall into the domain of Supervised Learning since the training dataset is labelled. And as the name suggests it is simply a special case in which there are only two classes. Some typical examples include: Credit Card Fraudulent Transaction detection, Medical Diagnosis, Spam Detection, etc..

Now there are some paradigms that are used for learning binary classifiers which include:

- **Deep Learning Model (DLM) Classifier:** The idea is fairly simple: we train a single auto-encoder, and then use the hidden layer of that network as the input to another one. This second network learns a higher-order representation of the initial inputs that are based on the activations of the hidden nodes of the original network. Progressively more auto-encoders can be trained and stacked on top of one another, and if the purpose is to do classification or regression, then a Perceptron can be added at the top, to take the activations of the final set of hidden nodes and perform supervised learning on them. The simplicity of the scheme is also the problem with it. Each autoencoder is trained in a pretty much unsupervised way; it is normal back-propagation learning, but the input and the desired output are the same, so there is no real information about what the hidden layer should be learning. There is some real supervised learning in the Perceptron at the final layer of the network, but this network has to deal with what it gets as the inputs, there is no error signal that informs the weights in all of the lower-down autoencoders and enables them to be trained further to produce more useful representations of the inputs. There is some similarity between autoencoders and RBMs; they perform the same job, but the autoencoder is directional in that the weights run from input to hidden node to output, while the RBM weights are symmetrical. This means that we can run the RBM backwards: we can take samples of the hidden nodes and infer the values of the visible nodes that gave rise to them. This is a generative model of the type of inputs that the network sees. And this means that we can get information from the top of a stack of RBMs and push it back down through the RBMs all the way back to the input visible units, which gives us a chance of actually changing the weights of these RBMs, and so the representations that they find. Deep learning is a very popular area for research at the moment, and various companies like Google obviously believe it is important, since they are employing a large number of deep learning researchers, and buying up companies that have been successful in developing applications based upon these ideas.

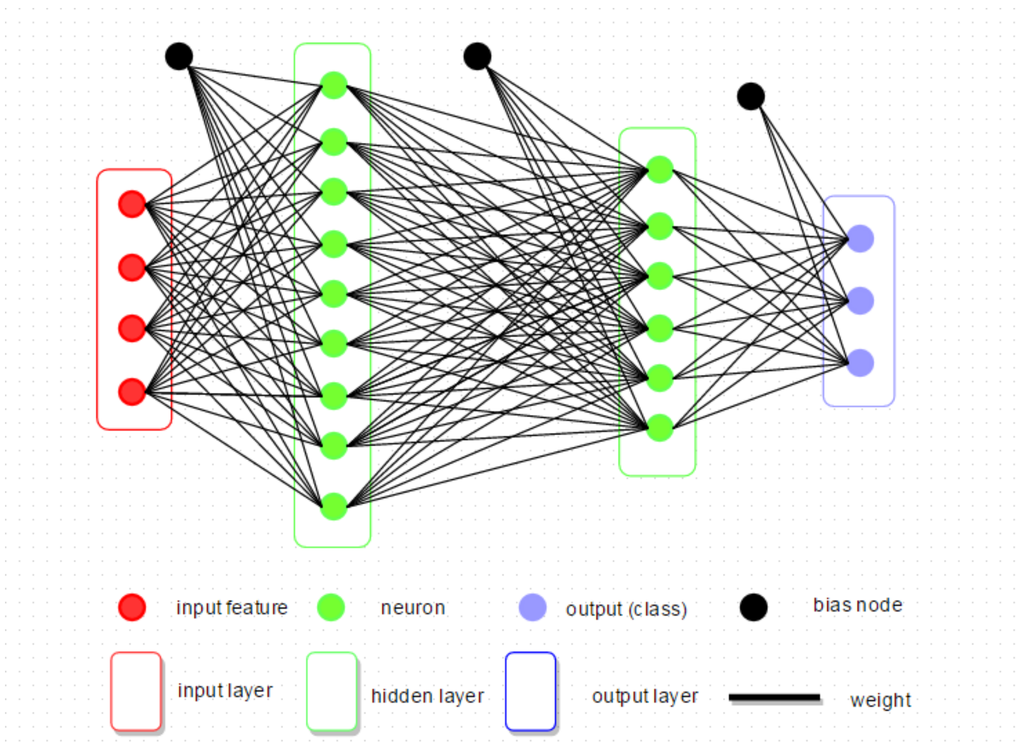


Fig 3.2: Basic structure of Deep Learning Modelled Classifier

Some activation functions that are used here are given below:

- **ReLU:** The Rectifier Linear Unit activation function is defined as
$$f(x) = \max(0, x)$$
where *x* is the input to a neuron. This is also known as a ramp function and is analogous to half-wave rectification.

- **Sigmoid:** A sigmoid function is a mathematical function having a characteristic "S"-shaped curve or sigmoid curve. Often, *sigmoid function* refers to the special case of the logistic function shown in the first figure and defined by the formula $f(x) = \frac{1}{1+e^{-x}}$.

- **Support Vector Machine (SVM) Classifier:** SVM is a supervised machine learning algorithm which can be used for both classification and regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well (see Figure 3.3).

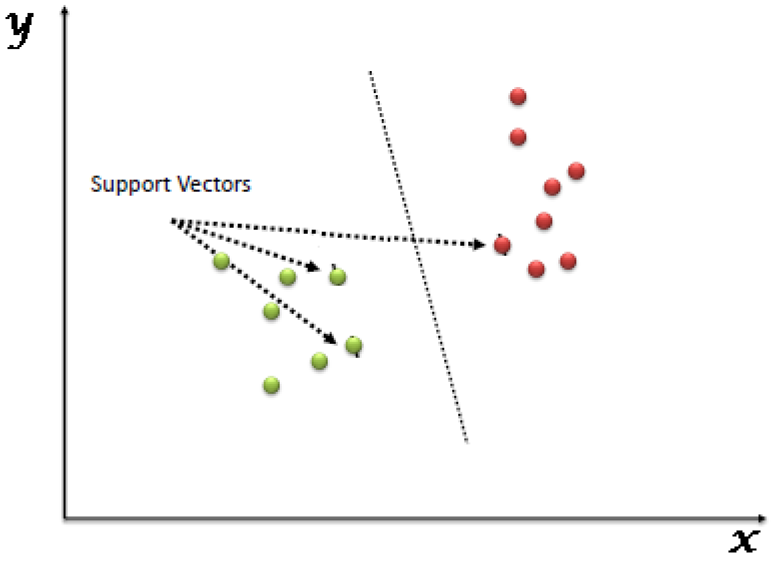


Fig. 3.3: Classification using SVM.

Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, the SVM falls weaker.

- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).

- **Random Forest (RF) Classifier:** Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

  Random forest algorithm is a supervised classification algorithm. As the name suggest, this algorithm creates the forest with a number of trees.

  In general, the **more trees in the forest** the more robust the forest looks like. In the same way in the random forest classifier, the **higher the number** of trees in the forest gives **the high accuracy** results.

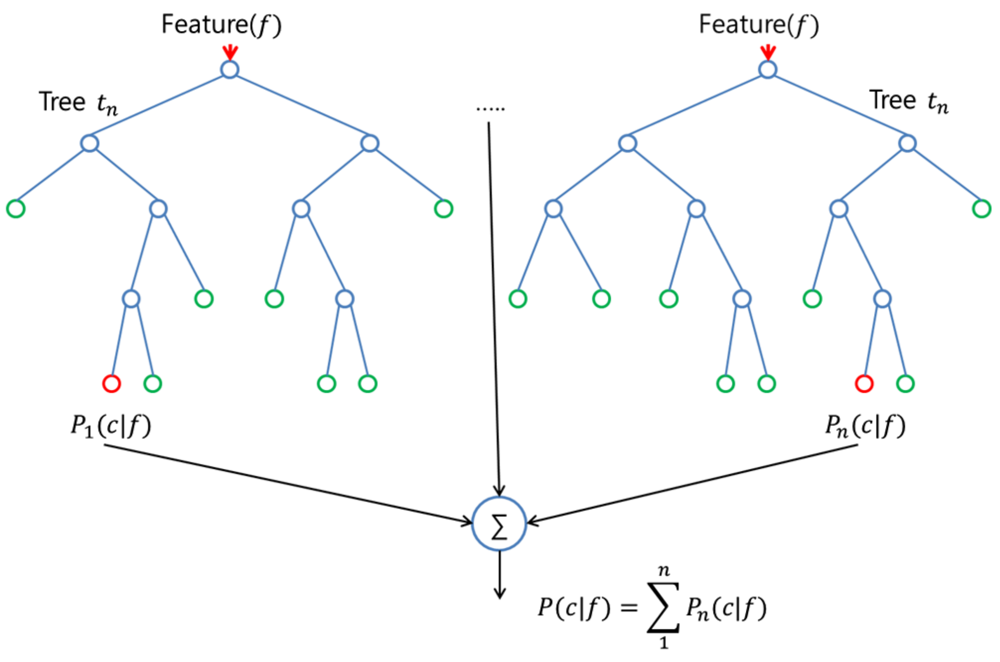


Fig. 3.4: Basic structure of Random Forest Classifier.

To perform prediction using the trained random forest algorithm uses the below pseudocode.

Step 1: Takes the test features and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome (target).

Step 2: Calculate the votes for each predicted target.

Step 3: Consider the high voted predicted target as the final prediction from the random forest algorithm.

3.3 CROSS VALIDATION

Cross-validation is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. In a prediction problem, a model is usually given a dataset of known data on which training is run (training dataset), and a dataset of unknown data (or first seen data) against which the model is tested (called the validation dataset or testing set). The goal of cross validation is to define a dataset to "test" the model in the training phase (i.e., the validation set), in order to limit problems like overfitting, give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem), etc. One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). To reduce variability, multiple rounds of cross-validation are performed using different partitions and the validation results are combined (e.g. averaged) over the rounds to estimate a final predictive model.

One of the main reasons for using cross-validation instead of using the conventional validation (e.g. partitioning the data set into two sets of 70% for training and 30% for test) is that there is not enough data available to partition it into separate training and test sets without losing significant modelling or testing capability. In these cases, a fair way to properly estimate model prediction performance is to use cross-validation as a powerful general technique.

In summary, cross-validation combines (averages) measures of fit (prediction error) to derive a more accurate estimate of model prediction performance.

- **K-Fold Cross Validation:** In k-fold method, you have to divide the data into k segments, k-1 of them are used for training, while one is left out and used for testing. It is done k times, first time, the first segment is used for testing, and remaining are used for training, then the second segment is used for testing, and remaining are used for training, and so on.

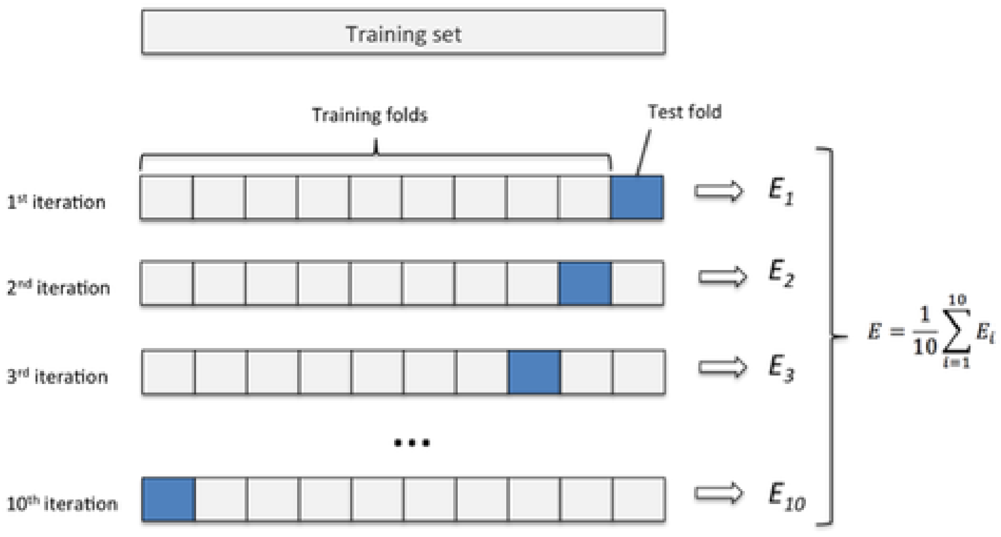


Fig. 3.5: 10 Fold cross validation(K=10).

3.4 FEATURE SELECTION

Feature selection refers to the process of reducing the inputs for processing and analysis. The data contains many features that are irrelevant and can be removed without having much loss of information. Irrelevant features in the data can decrease the accuracy of many models. It returns a subset of the features. It reduces the dimensionality of data by selecting only a subset of measured features to create a model. Selection criteria, usually involves the minimization of a specific measure.

Feature selection techniques are used for the following reasons:

- Simplification of models: Feature selection model improves the quality of the model and makes the process of modelling more efficient. Simplified models can be easier to interpret.
- Smaller data set: Most data mining algorithms require a much larger training set if the data set is of high-dimension. Feature selection reduces the dimensionality and thus removes the need of larger training data set.
- Shorter training and testing time: With selection of needed features, less time is required for the computation.
- It reduces over fitting: Less irrelevant data results in less opportunity to make decisions based on noise.

➢ **Univariate Feature Selection:** Univariate feature selection is used to have a better understanding of data, its structure and characteristics. Univariate feature selection examines each feature individually to determine the strength of the relationship between the feature and the response variable. It works by selecting the best features based on univariate statistical tests. The statistical tests can be used to select those features that have the strongest relationship with the output variable. Univariate Selection is a pre-processing step to an estimator. The Scikit-learn library provides “Select K Best” to select a specific number of features by removing all except the K highest scoring features.

➢ **Variance Threshold:** It is a baseline feature selection method. It removes all the features whose variance doesn’t meet some threshold. By default, it removes all non zero-variance features, i.e., features that have the same values in all the samples.

➢ **Linear SVC:** It is used for classification. Linear models penalized with the L1 norm have sparse solutions: many of their estimated coefficients are zero. The objective of a Linear SVC (Support Vector Classifier) is to fit to the data we provide, returning a best fit hyperplane that divides, or categorizes, our data. From there, after getting the hyperplane, we can then feed some features to your classifier to see what the predicted class is.

- **Tree Based Feature Selection:** Tree-based estimators can be used to compute feature importances, which in turn can be used to discard irrelevant features. The sklearn.tree module includes decision tree-based models for classification and regression.

- **Pipelined feature Selection:** Feature Selection is usually used as a pre-processing step before doing the actual learning. The recommended way to this pipelined feature selection. Pipelines work by allowing for a linear sequence of data transforms to be chained together culminating in a modelling process that can be evaluated.

3.5 PRINCIPAL COMPONENT ANALYSIS

(PCA) is a dimension-reduction tool that can be used to reduce a large set of variables to a small set that still contains most of the information in the large set.

- Principal component analysis (PCA) is a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components.
- The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible
- Traditionally, principal component analysis is performed on a square symmetric matrix.

3.6 GENE EXPRESSION

Gene expression data measures the level of activity of genes within a given tissue and thus provides information regarding the complex activities within the corresponding cells. This data is generally obtained by measuring the amount of messenger ribonucleic acid (mRNA) produced during transcription which, in turn, is a measure of how active or functional the corresponding gene is. As cancer is associated with multiple genetic and regulatory aberrations in the cell, these should reflect in the gene expression data. To capture these abnormalities, microarrays, which permit the simultaneous measurement of expression levels of tens of thousands of genes, have been increasingly utilized to characterize the global gene-expression profiles of tumour cells and matched normal cells of the same origin. Specifically, **microarrays** are used to identify the deferential expression of genes between two data instances, typically test vs. control, and to identify similarly expressed genes over multiple data instances. The processing pipeline of microarray data involves the raw data pre-processing to obtain a gene expression matrix, and then analysing the matrix for differences and/or similarities of expression. The gene expression matrix, GEM, contains preprocessed expression values with genes in the rows, and data instances in the columns. Thus, each column corresponds to an array (or gene-chip) experiment, and could contain multiple data instances if there were replicates. Each row in the matrix represents a gene expression profile.

Gene-chips can hold probes for tens of thousands of genes, whereas the number of data instances, limited by resources like time and money, is much smaller, at most in the hundreds. Thus, the gene expression matrix is typically very narrow (i.e. number of genes, n, is significantly larger than the number of data instances, m). This is known as the **curse of dimensionality** and it is a serious problem in gene network inference.

3.7 PACKAGES REQUIRED

The following are some of the required packaged to be imported in python:

- **Keras:** Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation.
- **SKLearn:** Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

4. Proposed Work

Before directly going to the algorithm, we performed the following actions:

- Finding and Downloading Dataset: We needed a few varieties of datasets for implementation of the proposed algorithm. Hence, we chose three datasets, namely, *Breast cancer dataset, Prostate cancer Dataset and Diffuse Large B-Cell Lymphoma Dataset*.
  - Breast Cancer: 128 samples and 47293 genes.
  - Prostate Cancer: 102 samples and 2135 genes.
  - Diffuse Large B-Cell Lymphoma: 77 samples and 7129 genes.

  All the above mentioned datasets were downloaded from Interdisciplinary Computing and Complex Bio-Systems (ICOS) group.
  (URL: http://ico2s.org/datasets/microarray.html).

- Data Pre-processing: After loading the dataset from the file (to the programming environment), the data must be processed first as per the input requirements of the classifier or any other methods that we will be using. The pre-processing, in our case, involves:
  - Conversion of string values to equivalent float or int values.
  - Conversion of data in list to Numpy array format.
  - Interchanging the rows and columns of the matrix as per function input requirement.

  (*Refer Appn* for data pre-processing method).

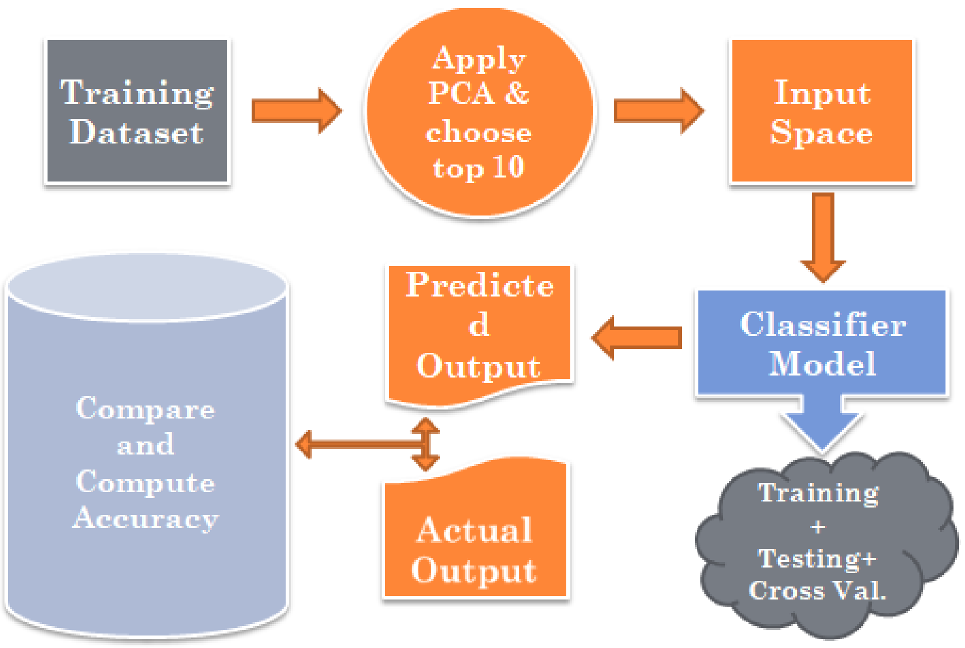

Training Dataset
Apply PCA & choose top 10
Input Space
Classifier Model
Predicte d Output
Training + Testing+ Cross Val.
Compare and Compute Accuracy
Actual Output

Fig. 4.1: Diagrammatic representation of the proposed work.

Our main algorithm performs the following actions:

Step 1: Once the data is processed, we take the complete data and send it to PCA(*refer Appn*).We choose 10 principal components for input space. The input space now consists of **a matrix of size *N x 10***, where N denotes the total number of samples*(rows)* with 10 attributes. The values in these attributes are the values of the Eigen-vectors generated by PCA.

Step 2: After getting features in the input space we send them for to the classifiers. The classifiers performs training, testing and cross validation (*refer B&P*) and outputs a set of predicted values. We have used four different model of classifiers for comparative study. The four models are Deep Learning Model, Support Vector Machine Model, Adaboost and Random Forest Model (*refer B&P*).

In DLM, we have used a learning rate of 0.1, 20 epochs and 10-Fold cross validation. The structure of DLM has 1 input layer with N number of neurons, 3 hidden layers with N/1.5, N/2, N/3.4 number of neurons and 1 output layer with 1 neuron. It used **Rectifier Linear Unit** (ReLU) for the first 4 layers and **sigmoid** for the output layer as activation functions.

In SVM, we have used the **SVM with Linear kernel** with all default parameters.

In ADABOOST, we have used only Adaboost with all default parameters.

In RF, we have used the entropy concept of **decision trees**, each tree having a max-depth of 4 and all other default parameters (*refer Appn* for codes).

Step 3:Once a set of predicted output is obtained, we compare them with the actual output and compute the accuracy of the particular model evaluating it.

5. Results and Observations

As per the results recorded in Table 1, we have made the following observations:

- None of the feature selection methods proved to be useful with the three datasets

But, it can be easily seen from the table that **SVM with PCA** outperforms all other models of classifier.

As per the results recorded in *LOGP, GP1*, we can see that the data in the two datasets, namely Breast Cancer Dataset and Prostate Cancer Dataset, are highly condensed in a region which makes classification difficult.

Moreover, after plotting the hyper-plane*(in LOGP, GP2)* using Linear SVM, it can be seen that a Maximum Margin Separating Hyperplane, that completely separates two distinct classes, was only generated in DLBC Lymphoma Dataset.

Also, it is observed that, for Diffuse Large B-Cell Lymphoma dataset the Cross-validation score of SVM and AdaBoost Classifiers with the inputs obtained from PCA(*refer LOT, Table 1*) is outperforming, giving a cross-validation score of nearly 96% with standard deviation of (+/-) 16% , as compared to the algorithm proposed in *Ref. 6* (*refer LOT, Table 2*).

6. Conclusion

In this work, we discussed some Machine Leaning techniques like feature selection *(which did not prove to be much useful)*, Dimensionality Reduction (PCA) and classification, while we outlined their application in cancer prediction and prognosis based on Binary Classification. Many methods are being tried using different Machine Learning Techniques for cancer prediction and prognosis. From our work till now, it is evident that Linear SVM with PCA is better for DLBC Lymphoma Dataset and more work related to this will be done.

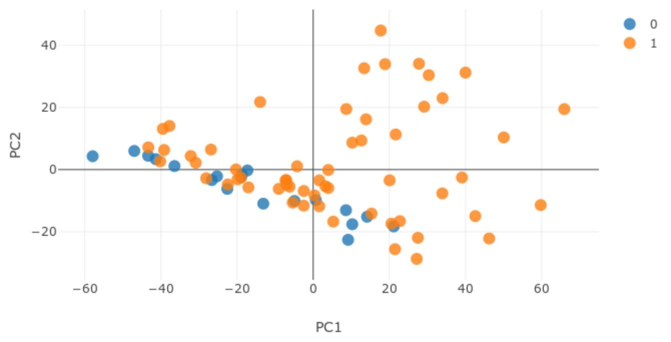
7. Future Work

DLBC Lymphoma Datasets from various experiments will be tested by the proposed algorithm. Moreover, classification accuracy of other cancer datasets could be improved by exploring various machine learning and Dimensionality Reduction techniques.

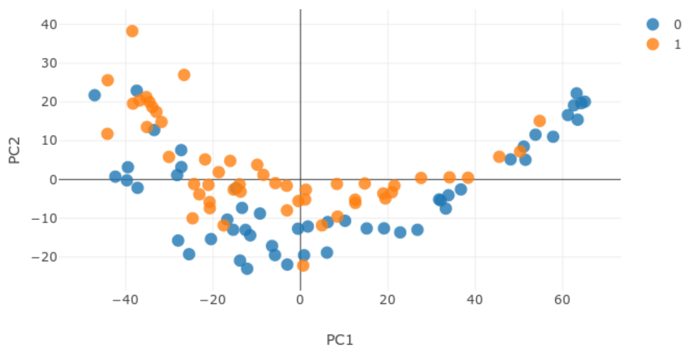
LIST OF GRAPHICAL PLOTS

<u>GP 1:</u> A plot showing the distribution of 2 components in a 2D plane by PCA Algorithm for

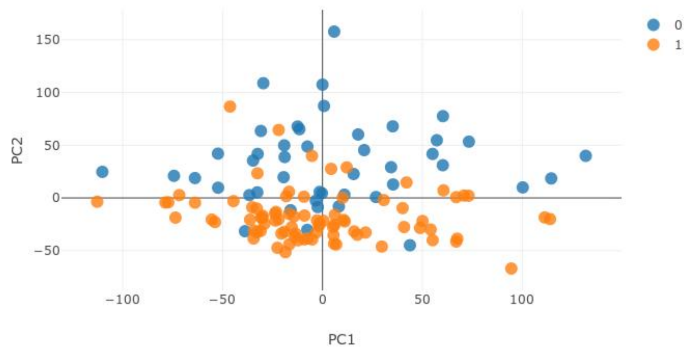
a) DLBC Lymphoma Dataset



b) Prostate Cancer Dataset

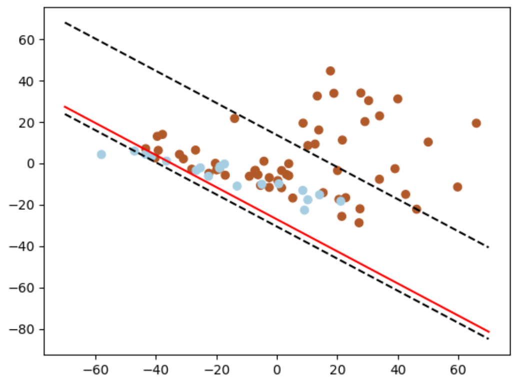


c) Breast Cancer Dataset

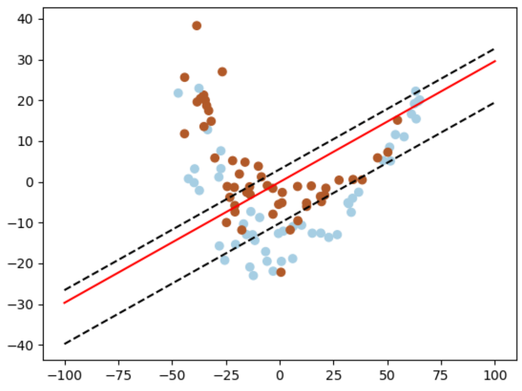


GP 2: A plot showing Maximum Hyper-plane Margin created by SVM using PCA features for

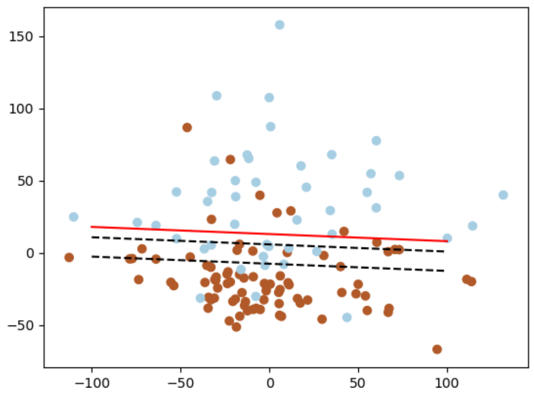
a) DLBC Lymphoma Dataset



b) Prostate Cancer Dataset



c) Breast Cancer Dataset



LIST OF TABLES

	Breast Cancer	Prostate	Lymphoma
DLM **CV**	65% (+/- 0.04)	50% (+/- 0.00)	70% (+/- 0.31)
DLM **Test**	55.26%	50%	78.26%
SVM **CV**	85% (+/- 0.09)	89% (+/- 0.11)	**96% (+/- 0.16)**
SVM **Test**	89.06%	92.15%	100%
RF **CV**	81% (+/- 0.15)	84% (+/- 0.16)	86% (+/- 0.19)
RF **Test**	94.53%	93.13%	100%
AB **CV**	76% (+/- 0.20)	79% (+/- 0.25)	92% (+/- 0.17)
AB **Test**	100%	100%	100%

Table 2: Comparison of SVM and RF classifiers with CFS, PLSS and RFS feature selection methods using error function as accuracy for 3 datasets (*Ref. 6*).

Dataset	Feature Selection	Classification	readAccuracy (%)
Prostate Cancer	CFS	SVM	90
	CFS	RF	92
	PLSS	SVM	90
	PLSS	RF	92
	RFS	SVM	88
	RFS	RF	93
Diffuse Large B-cell lymphoma	CFS	SVM	87
	CFS	RF	87
	PLSS	SVM	91
	PLSS	RF	87
	RFS	SVM	91
	RFS	RF	89
Breast Cancer	CFS	SVM	86
	CFS	RF	86
	PLSS	SVM	84
	PLSS	RF	89
	RFS	SVM	80
	RFS	RF	89

ABBREVIATIONS & NOMENCLATURE

- Appn – Refer to Appendix
- B&P – Refer to Background and Preliminaries
- LOT – Refer to List of Tables
- LOGP – Refer to List of Graphical Plots
- Ref. X– Refer to reference number X in the References.

REFERENCES

1. Daniele Ravi et. at.. *Deep Learning in health informatics.* IEEE Journal of Biomedical and Health Informatics, Vol. 21, No. 1, January 2017.

2. Jiawei Han and Micheline Kamber's *Data Mining Concepts and Techniques 2nd Edition.*

3. Konstantina Kourou et. al.. *Machine Learning application in cancer prediction and prognosis*. Computational and StructuralBiotechnologyJournal 13(2015) 8–17.

4. Lijuan Zhong, Data Science M.S. Program, UMN from Institute for Health Informatics, UMN. Lecture Notes : *Deep Learning for Micro-Array Based Cancer Classification.*

5. Mathew J. Garnettet. al.. *Systematic identification of genomic markers of drug sensitivity in cancer cell*. Nature. PMC 2012 September 29.

6. Michael P. Menden et. al.. *Machine Learning Prediction of Cancer Cell Sensitivity to Drugs Based on Genomic and Chemical Properties*. PLos One,April 2013 | Volume 8 | Issue 4 | e61332.

7. Padideh Danaee et. al.. *A Deep Learning Approach For Cancer Detection and Relevant Gene Identification*. Pac Symp Biocomput - December 2016.

8. Rasool Fakoor et. al.. *Using deep learning to improve cancer detection.* ICML, June-2013.

9. Stephen Marsland's *Machine Learning –An Algorithmic Perspective 2nd Edition.*

10. Runxuan Zhang et. al.. *Multicategory Classification Using an Extreme Learning Machine for Microarray Gene Expression Cancer Diagnosis*. IEEE/ACM Transactions on Computational Biology and Bioinformatics, Vol. 4, No. 3, July-September 2007.

11. Zhihua Caiet. al.. *Classification of lung cancer using ensemble-based feature selection and machine learning methods.*Molecular BioSystems.

12. Herve Abdi et. al. *Princiapal Component Analysis* . 2010 John Wiley & Sons, Inc. WIREs Comp Stat 2010 2 433–459

APPENDICES

Appendix 1: The complete python codes for our proposed algorithm.

```
#importing all required packaged
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA as sklearnPCA
import numpy as np
import random
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.feature_selection import chi2
import plotly
plotly.offline.init_notebook_mode(connected=True)
import plotly.offline as py
from plotly.graph_objs import *
import plotly.tools as tls
import matplotlib.pyplot as plt

#below are some user defined functions
def processData(inputList,fname):
    mainList=[]
    nameList=[]
    output=[]
    desiredOutput=[]
    for i in range(0,len(inputList)-1):
        mainListPart=[]
        for j in range(0,len(inputList[i])):
            if(j==0):
                nameList.append(inputList[i][j])
            else:
                mainListPart.append(float(inputList[i][j]))
        mainList.append(mainListPart)

    mainList=[list(x) for x in zip(*mainList)]
    mainList=np.asarray(mainList, dtype=np.float64)
    output=inputList[-1]
    if(fname=="breast_preprocessed.txt"):
        for i in range(1,len(output)):
            #for prostate use if(output[i]=="tumor")
            #not required for lymphoma dataset... output already in 0/1
            if(output[i]=="luminal"):
                desiredOutput.append(1)
            else:
                desiredOutput.append(0)
    if(fname=="prostate_preprocessed.txt"):
        for i in range(1,len(output)):
            #for prostate use if(output[i]=="tumor")
            #not required for lymphoma dataset... output already in 0/1
            if(output[i]=="tumor"):
                desiredOutput.append(1)
            else:
                desiredOutput.append(0)
    if(fname=="lymphoma_preprocessed.txt"):
        for i in range(1,len(output)):
            #for prostate use if(output[i]=="tumor")
            #not required for lymphoma dataset... output already in 0/1
            if(output[i]=="1"):
                desiredOutput.append(1)
            else:
                desiredOutput.append(0)

    desiredOutput=np.asarray(desiredOutput, dtype=np.int64)
    return mainList,nameList,desiredOutput




def create_model():
    #the deep learning model structure
    model = Sequential()
    model.add(Dense(len(trainOut), input_dim=len(trainIn[0]), activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(int(len(trainOut)/1.5), activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(int(len(trainOut)/2), activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(int(len(trainOut)/3.5), activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(1, activation='sigmoid'))

    #learning rate=0.1 and rest set to default
    rms=keras.optimizers.RMSprop(lr=0.1, rho=0.9, epsilon=1e-08, decay=0.0)
    model.compile(loss='mean_squared_error',optimizer=rms,metrics=['accuracy'])
    return model

def deepClassifier(x,y,t,o):
    seed = len(x) - 1
    np.random.seed(seed)
    #enclosing model in a classifier
    model = KerasClassifier(build_fn=create_model, epochs=20, batch_size=len(y),
verbose=0)
    model.fit(x,y,epochs=20,batch_size=len(x))
    #evaluate using 10-fold cross validation
    kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=seed)
    scores = cross_val_score(model, x, y, cv=kfold)
    print("DLM CV: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))

    obtainedOutput=model.predict(t)

    count=0
    for i in range(0,len(o)):
        if(o[i]==obtainedOutput[i][0]):
            count=count+1

    count=float(count)
    accuracy=count/len(o)*100
    print("DLM Accuracy:",accuracy)




def svmClassifier(x,y,t,o):
    clf =
svm.SVC(kernel="linear",C=1,decision_function_shape="ovo",cache_size=1000)
    seed = len(x) - 1
    np.random.seed(seed)
    kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=seed)
    scores = cross_val_score(clf, x, y, cv=kfold)
    print("SVM CV: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
    clf.fit(x, y)
    X=x
    Y=y
    # get the separating hyperplane
    w = clf.coef_[0]
    a = -w[0] / w[1]
    xx = np.linspace(-100, 100)
    yy = a * xx - (clf.intercept_[0]) / w[1]

    # plot the parallels to the separating hyperplane that pass through the
    # support vectors
    b = clf.support_vectors_[0]
    yy_down = a * xx + (b[1] - a * b[0])
    b = clf.support_vectors_[-1]
    yy_up = a * xx + (b[1] - a * b[0])

    # plot the line, the points, and the nearest vectors to the plane
    plt.plot(xx, yy, 'r-')
    plt.plot(xx, yy_down, 'k--')
    plt.plot(xx, yy_up, 'k--')

    plt.scatter(clf.support_vectors_[:, 0], clf.support_vectors_[:, 1],s=80,
facecolors='none')
    plt.scatter(X[:, 0], X[:, 1], c=Y, cmap=plt.cm.Paired)

    plt.axis('tight')
    plt.show()

    obtainedOutput=[]
    obtainedOutput=clf.predict(t)
    count=0
    for i in range(0,len(obtainedOutput)):
                                if(obtainedOutput[i]==o[i]):
                                          count=count+1
    total=len(obtainedOutput)
    count=float(count)
    total=float(total)
    accuracy=count/total*100
    print("SVM Accuracy:",accuracy)



def rfClassifier(x,y,t,o):
              clf = RandomForestClassifier(max_depth=4)
              seed = len(x) - 1
```

```
        np.random.seed(seed)
        kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=seed)
        scores = cross_val_score(clf, x, y, cv=kfold)
        print("RF CV: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
        clf.fit(x, y)
        obtainedOutput=[]
        obtainedOutput=clf.predict(t)
    count=0
        for i in range(0,len(obtainedOutput)):
         if(obtainedOutput[i]==o[i]):
               count=count+1
        total=len(obtainedOutput)
        count=float(count)
        total=float(total)
        accuracy=count/total*100
        print("RF Accuracy:",accuracy)

def adaboost(x,y,t,o):
        clf = AdaBoostClassifier()
        seed = len(x) - 1
        np.random.seed(seed)
        kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=seed)
        scores = cross_val_score(clf, x, y, cv=kfold)
        print("AB CV: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
        clf.fit(x, y)
        obtainedOutput=[]
        obtainedOutput=clf.predict(t)
        count=0
        for i in range(0,len(obtainedOutput)):
         if(obtainedOutput[i]==o[i]):
               count=count+1
        total=len(obtainedOutput)
        count=float(count)
        total=float(total)
        accuracy=count/total*100
        print("AB Accuracy:",accuracy)


def partitionData(mainList,desiredOutput):
      n=int(0.7*len(mainList))

      trainingInput=[]
      trainingInput=mainList[0:n]
      trainingInput=np.asarray(trainingInput, dtype=np.float64)
```

```
        trainingOutput=[]
        trainingOutput=desiredOutput[0:n]
        trainingOutput=np.asarray(trainingOutput, dtype=np.int64)

        testingInput=[]
        testingInput=mainList[n+1:]
        testingInput=np.asarray(testingInput, dtype=np.float64)

        testingOutput=[]
        testingOutput=desiredOutput[n+1:]
        testingOutput=np.asarray(testingOutput, dtype=np.int64)
        return trainingInput,trainingOutput,testingInput,testingOutput



def  pcaFeatures(X,y):
        X_std = StandardScaler().fit_transform(X)
        sklearn_pca = sklearnPCA(n_components=10)
        Y_sklearn = sklearn_pca.fit_transform(X_std)


        traces = []

        for name in (0, 1):

            trace = Scatter(
                    x=Y_sklearn[y==name,0],
                    y=Y_sklearn[y==name,1],
                    mode='markers',
                    name=name,
                    marker=Marker(
                            size=12,
                            line=Line(
                                    color='rgba(217, 217, 217, 0.14)',
                                    width=0.5),
                            opacity=0.8))
            traces.append(trace)


        data = Data(traces)


        layout = Layout(xaxis=XAxis(title='PC1', showline=True),
                                        yaxis=YAxis(title='PC2', showline=True))
        fig = Figure(data=data, layout=layout)
        py.plot(fig)
        return Y_sklearn







#main begins here
#"prostate_preprocessed.txt" for prostate dataset
#"lymphoma_preprocessed.txt" for lymphoma dataset
file_name="prostate_preprocessed.txt"

f = open(file_name,"r")
d = f.readlines()
inputList=[x.split(" ") for x in d]

#preprocessing dataset
mainList,nameList,desiredOutput=processData(inputList,file_name)
mainList=np.asarray(mainList, dtype=np.float64)
desiredOutput=np.asarray(desiredOutput, dtype=np.int64)




mainList=pcaFeatures(mainList,desiredOutput)

#trainIn,trainOut,testIn,testOut=partitionData(mainList,desiredOutput)

trainIn=mainList
testIn=mainList
trainOut=desiredOutput
testOut=desiredOutput


#implementing different classifiers
print("\n\n
####################################################################
######### \n")
deepClassifier(trainIn,trainOut,testIn,testOut)
```

```
svmClassifier(trainIn,trainOut,testIn,testOut)
rfClassifier(trainIn,trainOut,testIn,testOut)
adaboost(trainIn,trainOut,testIn,testOut)
print("\n\n
####################################################################
######### \n")

#endMain
```