

*Dissertation Project On*

**“APPLICATION OF DATA MODELLING IN BUILDING  
STRUCTURE-ACTIVITY CLASSIFIERS FROM  
*Plasmodium falciparum* BIOASSAY SCREENS”**

*Submitted in partial fulfillment of the requirements for the award of degree of*

**MASTER OF TECHNOLOGY**

**In**

**BIOINFORMATICS**

**By**

**Ms. ANISHA KATHPALIA**

**(02/B.INF/2K10)**

Under the guidance of

**Dr. Asmita Das**  
**DTU (formerly DCE)**

**Dr. Andrew M. Lynn**  
**JNU, New Delhi**



**DEPARTMENT OF BIOTECHNOLOGY  
DELHI TECHNOLOGICAL UNIVERSITY  
(FORMERLY DELHI COLLEGE OF ENGINEERING)**

**2010-2012**

## CERTIFICATE

This is to certify that the work entitled “*Application of data modelling in building structure-activity classifiers from Plasmodium falciparum bioassay screens*” submitted by *Ms. Anisha Kathpalia* in partial fulfillment for the award of degree of Master of Technology in Bioinformatics from Delhi Technological University, (formerly Delhi College of Engineering), Delhi has been carried out under my supervision.

*Dr. Asmita Das*

*Assistant Professor  
Department of Biotechnology  
Delhi Technological University (formerly DCE)*

## Declaration

I, Anisha Kathpalia, hereby declare that the work entitled “Application Of Data Modelling in Building Structure-Activity Classifiers from *Plasmodium falciparum* Bioassay Screens” has truly been carried out by me under the guidance of Dr. Andrew Lynn, JNU Delhi and Dr. Asmita Das, D.T.U, Delhi.

This project is a part of partial fulfillment of requirement for the degree of M.Tech in Bioinformatics. This is the original work and has not been submitted for any other degree in any other university.

Anisha Kathpalia

(02/B.INF/2K10)

## ACKNOWLEDGEMENT

First and foremost, I bow my head and fold my hands in faith, to the supreme sculptor of our destiny and every existence “**GOD ALMIGHTY**”.

It express my gratitude to **Prof. P. B. Sharma**, Vice Chancellor, DTU, Delhi for according me permission to undertake the project work at DTU, Delhi for partial fulfilment for the degree of Master of Technology. It is my great pleasure to express my thanks to **Prof. S. Maji**, HOD, Dept. of Biotechnology, Delhi Technological University, for providing me with the opportunity to utilize the facilities of the college.

With deep gratitude, I express my reverence to my esteemed teacher **Dr. Asmita Das**, Assistant Professor, Delhi Technological University, Delhi for her constant encouragement and unflinching support throughout my post graduate curriculum. I have been astoundingly fortunate to have an advisor who gave me the freedom to explore on my own, and at the same corrected me when my steps faltered. . I owe my deepest gratitude for his timely guidance, thought provoking discussions, words of encouragement, acceptance and appreciation, for believing in me, for teaching me with patience and understanding.

I express my heartfelt thanks to **Dr. Andrew M. Lynn**, Associate Professor, Jawaharlal Nehru University, New Delhi, for providing me an opportunity to work under his esteemed guidance and providing me with various innovative ideas and suggestions which led to the successful completion of my project. I feel very much privileged to have worked under his guidance.

It is with immense gratitude that I acknowledge the support and help of my teachers **Dr. Yasha Hasija** , **Dr. Navneeta Bharadwaj** ,**Dr. Monica Sharma**, and all the respected teachers for their guidance, invaluable support and professional insight throughout my post-graduation course. Last, but not the least, I thank my beloved family and friends for giving me moral support, encouragement and blessings.

**ANISHA KATHPALIA**

Chapters	PageNo.
<b>1. Abstract.....</b>	<b>1</b>
<b>2.Introduction.....</b>	<b>2</b>
<b>3. Review of iterature.....</b>	<b>7</b>
3.1 Data Mining.....	7
3.1.1 Association rule learning.....	9
3.1.2 Clustering.....	10
3.1.3 Regression.....	10
3.1.4 Classification.....	11
3.2Limitations.....	11
3.3 Cheminformatics.....	12
3.4 Machine Learning: an indispensable tool in Bioinformatics .....	15.
3.4.1 Supervised learning.....	17
3.4.2 Unsupervised learning or clustering.....	17
3.4.3 Semisupervised learning.....	17
3.4.4 Reinforcement learning.....	18
3.4.5 Optimization.....	18
3.4.6 Steps involved in Supervised Learning.....	18
3.5 Predictive Models.....	20

3.5.1 Predictive modeling methods in R.....	20
3.6 Ensemble method.....	21
<b>4. Materials &amp; Methods.....</b>	<b>22</b>
4.1 Materials.....	22
4.2 Methodology.....	24
4.2.1 Partial Least Squares (PLS).....	30
4.2.2 Flexible Discriminant Analysis (FDA) .....	31
4.2.3 Multivariate adaptive regression splines (MARS) .....	33
<b>5. Results.....</b>	<b>37</b>
5.1 Result Analysis.....	46
<b>6. Discussion.....</b>	<b>49</b>
<b>7. Conclusion.....</b>	<b>53</b>
<b>8. Bibliography.....</b>	<b>54</b>
<b>9. Appendix.....</b>	<b>57</b>

### LIST OF GRAPHS

S.NO.	TITLE	PAGE.NO
GRAPH -I	A PLOT OF THE ESTIMATES OF THE ROC VALUES CALCULATED USING THE BOOTSTRAP 632 RULE (PLS)	37
GRAPH -II	ROC CURVE FOR THE TEST SET (PLS)	38
GRAPH -III	A PLOT OF THE ESTIMATES OF THE ROC VALUES CALCULATED USING THE BOOTSTRAP 632 RULE (FDA)	39
GRAPH -IV	ROC CURVE FOR THE TEST SET (FDA)	40
GRAPH - V	A PLOT OF THE ESTIMATES OF THE ROC VALUES CALCULATED USING THE BOOTSTRAP 632 RULE (earth)	41
GRAPH -VI	ROC FOR THE TEST SET (earth)	42

### LIST OF FIGURES

S.NO.	TITLE	PAGE.NO
FIGURE -I	CLASS PROBABILITIES FOR THE TEST SET (earth)	43
FIGURE -II	CLASS PROBABILITIES FOR THE TEST SET (FDA)	43
FIGURE -III	CLASS PROBABILITIES FOR THE TEST SET (PLS)	44

### LIST OF TABLES

SL.NO	TITLE	PAGE NO.
TABLE -I	THE CONFUSION MATRIX FOR THE TEST (PLS) SET	38
TABLE -II	THE CONFUSION MATRIX FOR THE TEST SET (FDA)	40
TABLE -III	THE CONFUSION MATRIX FOR THE TEST SET (earth)	42
TABLE -IV	THE CONFUSION MATRIX (ENSEMBLE)	45
TABLE -IV	THE ROC VALUES AND OVERALL ACCURACY FOR ALL THE MENTIONED METHODS	46



## 1. Abstract

Malaria is responsible for approximately 1 million deaths annually; thus, continued efforts to discover new antimalarials are required. The prevalence of resistance to known antimalarial drugs has resulted in the expansion of antimalarial drug discovery efforts. *P. falciparum* is the deadliest of the species of *Plasmodium* that infect humans, and it accounts for the majority of malaria infections and virtually all of the malaria-related mortality worldwide. So here we are trying to build trained Classifiers which enable virtual screening of compounds with specific activity towards *P.falciparum*.

Due to rapid growth of various databases, business and health industry leaders have turned to computer applications that can increase the efficacy of treatment and services. Data Mining or Knowledge Discovery in Databases is a process of discovering meaningful new correlations, patterns, and trends by digging into large amounts of data stored in warehouses. The purpose of bioinformatics data mining is to discover the relationships and patterns in large databases to provide useful information for biomedical analysis and diagnosis.

Cheminformatics is paving a way for identification and development of novel drugs. In recent years, there has been an explosion in the availability of publicly accessible chemical information, including chemical structures of small molecules, structure-derived properties and associated biological activities in a variety of assays. These data sources present us with an opportunity to develop and apply computational tools to extract and understand the underlying structure-activity relationships.<sup>1</sup>

Here, the various structure activity classifiers for the *Plasmodium falciparum* bioassay screens were made. The data is downloaded from PubChem Bioassay containing the compounds active and inactive for malaria. The molecular descriptors are calculated using Dragon software. The data is then preprocessed using the scripting language 'R'. **The script "Classification.Rnw" distributes the live malaria data in train data & test data (trainClass.Rdata and testClass.Rdata) on which we can further work. The script "Classification-method.Rnw" uses different methods and builds up the predictive models using the 'caret' package.**

# *INTRODUCTION*

## 2. Introduction

The large volumes of data that are now available present an opportunity for the development of better predictive models, for the selection of more promising drug candidates, and for more comprehensive decisions to be made by researchers in chemistry and biology. It is common knowledge that malaria is a serious worldwide health problem due to the emergence of parasites that are resistant to well-established antimalaria drugs. Although continued attempts to develop a vaccine for malaria are ongoing, drugs continue to be the only treatment option. The progress of this approach in drug discovery has also introduced challenges associated with the need to develop better tools to manage the large volumes of data.<sup>2</sup>

**Cheminformatics** deals with gathering and systematic use of chemical information, and the use of those data to predict the behaviour of unknown compounds in silico.[3] This branch of science basically combines the scientific working fields of chemistry with computer science for the purpose of making faster decisions in the area of drug lead identification and optimization. Implementing, handling and searching chemical databases is a crucial aspect of chemoinformatics.<sup>3</sup>

**Cheminformatics** is paving a way for identification and development of novel drugs. There has been an explosion in the availability of publicly accessible chemical information, including chemical structures of small molecules, structure-derived properties and associated biological activities in a variety of assays. Using Computational tools on this large amount of data various structure-activity relationships can be known and understood.<sup>1</sup>

A quantitative structure-activity relationship is a mathematical relationship between a molecule's physical properties and its chemical properties. The primary purpose of QSAR is to make predictions about how an as-yet-unstudied molecule will behave, based on better-known molecules that are structurally similar to it.<sup>4</sup>

The molecular descriptor is the final result of a logical and mathematical procedure which transforms chemical information encoded within a symbolic representation of a molecule into a useful number or the result of some standardized equipment.<sup>5</sup>

**Molecular descriptors are numerical values that characterize properties of molecules.** Biological active substances interact, in most cases, with biomolecules, triggering specific molecular mechanisms like activation of an enzyme cascade or opening of an ion channel, which finally leads to a certain biological response.

Quantitative structure-activity relationships correlate this response with molecular properties of compounds under interest. Because the response depends on the concentration of the active substance at the site of action and on the strength of interaction with the biological macromolecule, both of these aspects must be modeled quantitatively by QSAR.

In the mean time, a variety of descriptors of molecular properties have been developed. Computational approaches to lipophilicity are nearly as diverse as the QSAR methods themselves. Electronic properties in terms of point charges or molecular electrostatic potentials can be evaluated by quantum-chemical ab initio methods for molecules up to 50 atoms. Using semi empirical methods like AM1 or PM3, such properties can be calculated even for larger systems. Steric descriptors reach from molecular surface and volume to connectivity and topological indices and to Verloop parameters.<sup>6</sup>

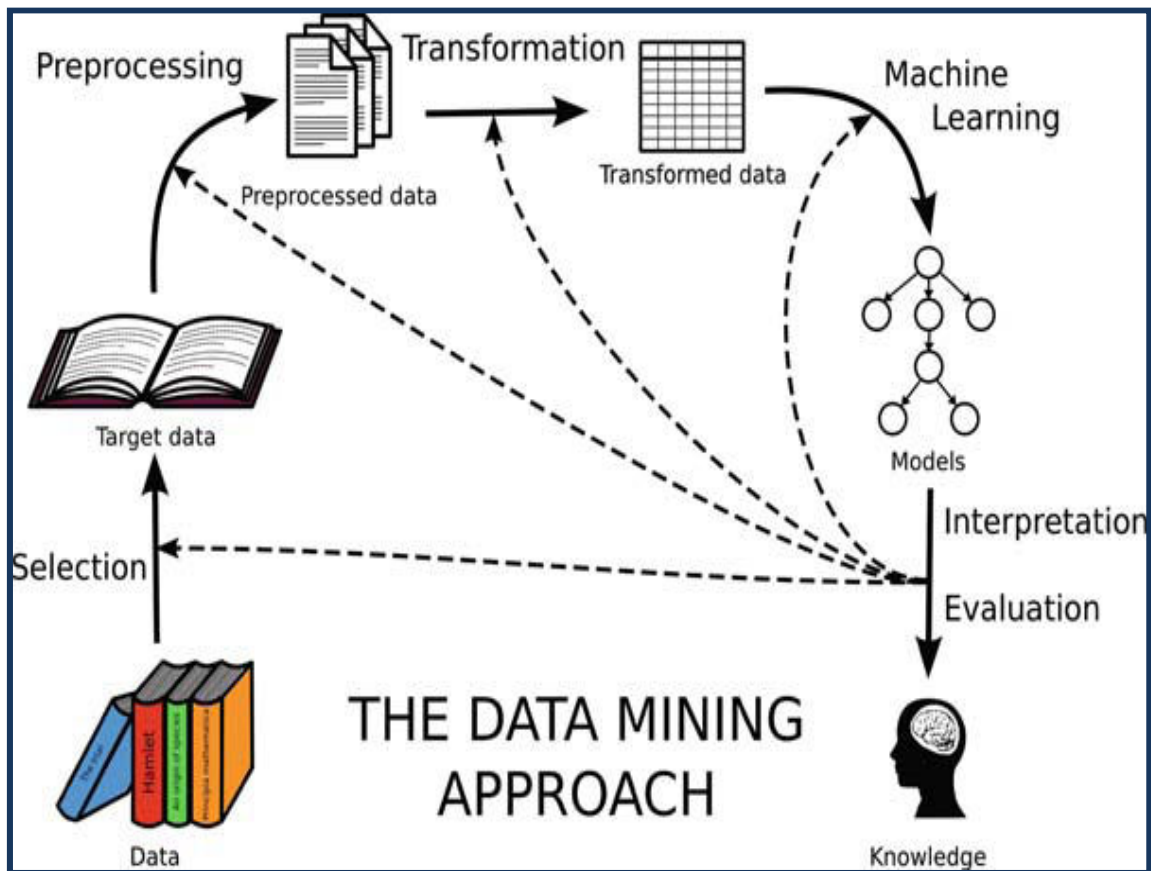
There are number of softwares available for descriptor calculation such as E-DRAGON, the electronic remote version of the well known software DRAGON, which is an application for the calculation of molecular descriptors developed by the Milano Chemo-metrics and QSAR research group.<sup>7</sup>

DRAGON provides more than 4,000 molecular descriptors that are divided into logical blocks. The user can calculate not only the simplest atom type, functional group and fragment counts, but also several topological and geometrical descriptors. These descriptors can be used to evaluate molecular structure-activity or structure-property relationships, as well as for similarity analysis and high throughput screening of molecule databases.<sup>7</sup>

To judge for all these activities of molecules data mining was the only feasible option for discovering knowledge and interpreting them. Data mining is a process of extraction of useful patterns and relationships from data sources such as databases or

web. It uses statistical and pattern matching techniques and borrows from statistics, data visualization, machine learning etc.

Data mining is a step in the KDD (knowledge discovery in databases) process consisting of applying computational techniques that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns. It is emerging as a new active area of research which combines methods and tools from the fields of statistics, machine learning, database management and data visualization .<sup>8</sup>



Data mining is a new discipline lying at the interface of statistics, database technology, pattern recognition, machine learning, and other areas. It is concerned with the secondary analysis of large databases in order to find previously unsuspected relationships which are of interest or value to the database owners.<sup>9</sup>

Hence, we use the branch of artificial intelligence i.e., machine learning which is a scientific discipline concerned with the design and development of algorithms that

allow computers to evolve behavior based on data and generate output which can be easily understood.

Machine learning methods are essentially computer programs that make use of sampled data or past experience information provide solutions to a given problem. A wide spectrum of algorithms commonly based on the artificial intelligence and statistics fields have been proposed by the machine learning community in the last decades.<sup>10</sup>

Supervised learning involves generation of a function that maps inputs to desired output where the class labels are already known whereas unsupervised learning models a set of inputs.

The two "high-level" primary goals of data mining, are prediction and description. The main tasks well suited for data mining, all of which involves mining meaningful new patterns from the data, are:

**Classification:** Classification is learning a function that maps (classifies) a data item into one of several predefined classes.

**Estimation:** Given some input data, coming up with a value for some unknown continuous variable.

**Prediction:** Same as classification & estimation except that the records are classified according to some future behaviour or estimated future value.

**Association rules:** Determining which things go together, also called dependency modeling.

**Clustering:** Segmenting a population into a number of subgroups or clusters.

**Description & visualization:** Representing the data using visualization techniques

**Classification** is the common and simplest task which we are using in this topic to generate known structure in order to apply it on new data. The goal of classification is to accurately predict the target class for each case in the data.<sup>11</sup>

R is an open source programming language and software environment for statistical computing and graphics. It is quite similar to the S language and environment which was developed at Bell Laboratories by John Chambers and colleagues. The **caret** package was developed to create a unified interface for modeling and prediction.

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. R is very much a vehicle for newly developing methods of interactive data analysis. It has developed rapidly and has been extended by a large collection of packages.

The use of complex classification and regression models is becoming more and more commonplace in science, finance and a myriad of other domains. The R language (R Development Core Team 2008) has a rich set of modeling functions for both classification and regression, so many in fact, that it is becoming increasingly more difficult to keep track of the syntactical nuances of each function. **The caret package, short for classification and regression training**, was built to eliminate syntactical differences between many of the functions for building and predicting models.<sup>12</sup>

The caret package (short for classification and regression training) contains functions to stream-line the model training process for complex regression and classification problems.<sup>13</sup>

**Ensemble learning** is the process by which multiple models, such as classifiers or experts, are strategically generated and combined to solve a particular computational intelligence problem. Ensemble learning is primarily used to improve the (classification, prediction, function approximation, etc.) performance of a model, or reduce the likelihood of an unfortunate selection of a poor one<sup>10</sup>

Although the most common approach is to use a single model for class prediction, the *combination of classifiers* with different biases is gaining popularity in the machine learning community

. As each classifier defines its own decision surface to discriminate between problem classes, the combination could construct a more flexible and accurate decision surface and thus increasing the accuracy of the models



*REVIEW OF  
LITERATURE*

### **3. Review of Literature**

#### **3.1 DATA MINING**

It is an analytic process designed to explore data in search of consistent patterns or systemic relationships between variables and then to validate the findings by applying the detected patterns to new subset of data, here the ultimate goal is prediction. It's all about solving problems by analyzing data already present in databases. The analysis tools can include statistical models, algorithms and machine learning methods. These applications use a variety of parameters to examine the data. They include association, sequence or path analysis, classification, clustering and forecasting. Few reasons for using data mining:-

- Human skills are inadequate.
- Volume & dimensionality of data.
- High data growth rate.

Data mining is a new discipline lying at the interface of statistics, database technology, pattern recognition, machine learning, and other areas. It is concerned with the secondary analysis of large databases in order to find previously unsuspected relationships which are of interest or value to the database owners. Seeking knowledge from massive data is one of the most desired attributes of Data Mining. <sup>14</sup>

The development of methods for the analysis of this massive (and constantly increasing) amount of information is one of the key challenges in bioinformatics. This analysis step - also known as computational biology - faces the challenge of extracting biological knowledge from all the in-house and publicly available data. Furthermore, the knowledge should be formulated in a transparent and coherent way if it is to be understood and studied by bio- experts. The term “data mining” in bioinformatics refers to the set of techniques aimed at discovering useful relationships and patterns in biological data that were previously undetected. <sup>15</sup>

As an application compared to other data analysis applications such as structured queries or statistical analysis software, data mining represents a difference of kind. In order to conduct effective data mining, one needs to first examine what kind of features an applied knowledge discovery system is expected to have and what kind of challenges one may face at the development of data mining techniques.

1. Handling of different types of data.
2. Efficiency and scalability of data mining algorithms.
3. Usefulness, certainty, and expressiveness of data mining results.
4. Expression of various kinds of data mining requests and results.
5. Interactive mining knowledge at multiple abstraction levels.
6. Mining information from different sources of data.
7. Protection of privacy and data security.

The process of data mining consists of three stages:

**(1) Initial exploration-**

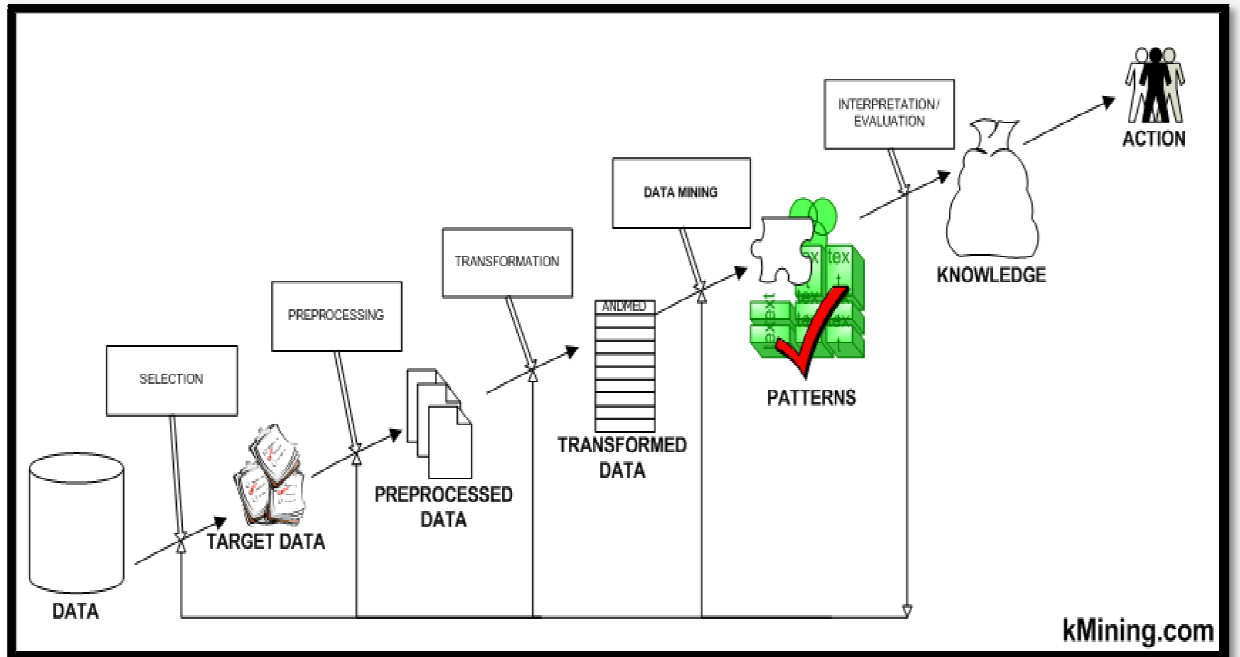
This stage usually starts with data preparation which may involve cleaning data, data transformations, selecting subsets of records and - in case of data sets with large numbers of variables ("fields") - performing some preliminary feature selection operations to bring the number of variables to a manageable range (depending on the statistical methods which are being considered)

**(2) Model building or pattern identification with validation/verification-**

This stage involves considering various models and choosing the best one based on their predictive performance (i.e., explaining the variability in question and producing stable results across samples).

**(3) Deployment (i.e., the application of the model to new data in order to generate predictions)-**

This final stage involves using the model selected as best in the previous stage and applying it to new data in order to generate predictions or estimates of the expected outcome. <sup>16</sup>



## Data mining tasks

Data mining basically involves collecting and managing of data along with analysis and prediction. <sup>14</sup>

Data mining involves four main classes of tasks that are – association rule learning, clustering, classification and regression.

### 3.1.1 Association rule learning

It is a popular and well researched method for discovering interesting relations between variables in large databases. Given set of transactions, the process finds rules that will predict the occurrence of an item based on the occurrences of other. Association rules are if/then statements that help uncover relationships between seemingly unrelated data in a relational database or other information repository.

An association rule has two parts, an antecedent (if) and a consequent (then). An antecedent is an item found in the data. A consequent is an item that is found in combination with the antecedent. Association rules are created by analyzing data for frequent if/then patterns and using the criteria *support* and *confidence* to identify the most

important relationships. *Support* is an indication of how frequently the items appear in the database. *Confidence* indicates the number of times the if/then statements have been found to be true. In data mining, association rules are useful for analyzing and predicting customer behavior. They play an important part in shopping basket data analysis, product clustering, catalog design and store layout.<sup>17</sup>

### **3.1.2 Clustering**

It is a process of partitioning a set of data (or objects) in a set of meaningful sub classes, called clusters. It helps users to understand natural grouping or structure in a dataset. Cluster is actually a collection of data objects that are 'similar' to one another and thus can be treated collectively as one group. Data clustering has immense number of applications in every field of life. One has to cluster a lot of thing on the basis of similarity either consciously or unconsciously. So the history of data clustering is old as the history of mankind. In order to detect many diseases like tumor etc, the scanned pictures or the x-rays are compared with the existing ones and the dissimilarities are recognized. Using this technique and some really precise methods for the pattern matching, diseases like really fine tumor can also be detected.<sup>18</sup>

### **3.1.3 Regression**

It is a data mining function that predicts a number. A regression task begins with a data set in which the target values are known. In the model build (training) process, a regression algorithm estimates the value of the target as a function of the predictors for each case in the build data. These relationships between predictors and target are summarized in a model, which can then be applied to a different data set in which the target values are unknown.

For example, a regression model could be used to predict the value of a house based on location, number of rooms, lot size, and other factors. A regression model that predicts house values could be developed based on observed data for many houses over a period of time. In addition to the value, the data might track the age of the house, square footage,

number of rooms, taxes, school district, proximity to shopping centres, and so on. House value would be the target, the other attributes would be the predictors, and the data for each house would constitute a case.<sup>19</sup>

## **Classification**

It is a data mining function that assigns items in a collection to target categories or classes. In the model build (training) process, a classification algorithm finds relationships between the values of the predictors and the values of the target. Different classification algorithms use different techniques for finding relationships. These relationships are summarized in a model, which can then be applied to a different data set in which the class assignments are unknown.

Different classification algorithms use different techniques for finding relationships. The most common algorithms involved are decision tree learning, nearest neighbor, Bayesian classification, neural networks and support vector machines. Classification models are tested by comparing the predicted values to known target values in a set of test data. The historical data for a classification project is typically divided into two data sets: one for building the model; the other for testing the model. Accuracy refers to the percentage of correct predictions made by the model when compared with the actual classifications in the test data.<sup>18</sup>

### **3.2 Limitations of Data Mining:**

While data mining products can be very powerful tools, they are not self sufficient applications. To be successful, data mining requires skilled technical and analytical specialists who can structure the analysis and interpret the output that is created.

Although data mining can help reveal patterns and relationships, it does not tell the user the value or significance of those patterns. These types of determinations must be made by the user.<sup>20</sup>

### 3.3 Cheminformatics :

The shift in the drug discovery paradigm that has resulted in increased dependence on large volumes of information for decision making has created new challenges and opportunities in cheminformatics research. The field of cheminformatics is multi-disciplinary and is an amalgam of chemistry, mathematics and computer science.

Cheminformatics focuses on the development of methods and tools to address problems in the management and analysis of chemical information. Such information can be of a variety of types including chemical structure (in various formats such as SMILES, SDF, CML and so on) and derived aspects of chemical structure (such as number of atoms and various descriptors of structure).<sup>1</sup>

With the advent of large public repositories of chemical and biological data, there has been an explosion in the type and amount of data that is now available for cheminformatics analysis. Thus, we can now access chemical structures of millions of compounds, as well as the biological activities of many of these structures in a variety of biological assays. Public literature databases now allow one to move from chemical structure to specific documents and in some cases vice versa. The various data mining techniques are used in Cheminformatics including Classical QSAR, regression analysis, clustering, neural networks etc.

A number of cheminformatics tasks are based on mathematical and statistical modeling techniques (such as QSAR). QSAR represents predictive models derived from application of statistical tools correlating biological activity of chemicals with descriptors representative of molecule structure and properties. QSAR model depends on factors like quality of biological data, choice of descriptors and statistical methods. It attempts to find consistent relationship between biological activity and molecular properties so that these rules can be used to evaluate the activity of new compounds.<sup>21</sup>

Molecular Descriptors play an important role in chemistry, pharmaceutical sciences, environmental sciences, health research and quality control being obtained when molecules are transformed into a molecular representation allowing some mathematical treatment. Many molecular descriptors are derived from different theories

and approaches with the aim of predicting biological and physico-chemical properties of molecules.<sup>22</sup>

The descriptors (independent variables) are correlated to the biological activity (dependent variable) by means of statistical methods. Most commonly multivariate linear regression (MLR) is used, but also partial least squares (PLS) or neural networks. In some QSAR approaches genetic algorithms are employed to identify the relevant descriptors: population of models are created and step by step, models with a better "fitness score" (i.e. with better predictivity) are produced by "genetic operations" like cross-over, point mutations or selection generate.

E-DRAGON is the electronic remote version of the well known software DRAGON, which is an application for the calculation of molecular descriptors. To run DRAGON the user needs molecular structure files previously obtained by other specific molecular modeling software. The most common molecular file formats are accepted. In E-DRAGON the accepted molecular structure files SMILES, SDF (MDL) or MOL2 (Sybyl) files. DRAGON requires 3D optimised structures with HYDROGENS.<sup>22</sup>

Though cheminformatics has been in existence since the 1960's, it is only recently that large collections of chemical information have highlighted the need for infrastructures capable of handling them. QSAR models were developed to predict the anti-cancer activities of compounds tested against the 60 NCI cancer cell lines by the Developmental Therapeutics Program. The random forest was employed to predict the anti-cancer activity of a compound based on MACCS keys as structural descriptors. The performance of the models ranged from 75% to 80% correct over the 60 cell lines. [1]

Random forest models were also employed to predict the ability of compounds to inhibit cell proliferation in a variety of human cell lines. These cell lines have been used in high-throughput screening (HTS) assays [Xia2008], which tested approximately 1300 compounds. This data was used to train the individual random forest models, which were then used to predict the cytotoxicity of an external set of compounds.

SVM is widely used in cheminformatics and Quantitative-Structure Activity Relationship (QSAR) modeling. It is used for creation of virtually represented molecules and assessment of their likely suitability for synthesis and viability for use in the



body. The study of drug-likeness and report that SVM predictions were more robust than those from neural networks. Trained Classifiers enable virtual screening for discovering molecules with specific therapeutic target affinities from potentially millions of virtual representations. Finding the bioactive conformations of active molecules is key to understanding their mechanisms of action and thus for improving specificity and selectivity.<sup>1</sup>

Recently different classifiers such as randomforest or support vector machines (SVMs) have also been used to predict cytochrome P450 (CYP) mediated metabolism. Overall, 2439 compounds were prepared from the Pfizer proprietary compound library. These compounds included more than 13 chemical series that were synthesized for multiple drug discovery projects. Among these compounds, 487 test compounds were randomly selected, and the remaining 1952 compounds were used for training.<sup>23</sup>

The training set was used for model building, and the test set was used for validation, with 193 descriptors calculated by MOE 2005.06 (Molecular Operating Environment, Chemical Computing Group Inc., Montreal, Canada). MOE descriptors include various 2D descriptors such as volume, shape, atom and bonds count, Kier-Hall connectivity, adjacency, partial charges, etc. The results using test compounds have demonstrated that all classifiers yielded satisfactory results (accuracy > 0.8, sensitivity > 0.9, specificity > 0.6, and precision > 0.8). Above all, classification by randomforest as well as SVM yielded kappa values of approximately 0.7 in an independent validation set, slightly higher than other classification tools. These results suggest that nonlinear/ensemble-based classification methods might prove useful in the area of *in silico* ADME modeling.<sup>23</sup>

Random Forest Clustering is used for tumour profiling based on tissue microarray data. Random Forest clustering is attractive for tissue microarray and other immunohistochemistry data since it handles highly skewed tumor marker expressions well and weighs the contribution of each marker according to its relatedness with other tumor markers. This is the first tumor class discovery analysis of renal cell carcinoma patients based on protein expression profiles. To explore whether the tissue microarray data can be used to identify fundamental subtypes of renal cell carcinoma patients, we

first carried out random forest clustering of all 366 patients. By analyzing the tumor markers simultaneously, the procedure automatically detected classes that correspond to clear- vs non-clear cell tumors.<sup>24</sup>

SVM is a new learning algorithm for classification being used in SAR(Structure Activity Relationship) analysis, a technique used by pharmaceuticals companies in the drug discovery process. Artificial intelligence techniques have been applied to SAR analysis since the late 1980s, mainly in response to increased accuracy demands. Intelligent classification techniques that have been applied to this problem include neural networks, genetic algorithms and decision trees. Each of the training sets was then used to train an algorithm to produce a classification rule. This classification rule can then be used to predict which of two unseen compounds has the greatest activity. The generalization ability of such a rule, that is, the probability that it will correctly rank two unseen compounds in terms of activity. Each of the trained classifiers is used to classify the corresponding test set. Classifiers typically learn by empirical risk minimization (ERM), that is they search for the hypothesis with the lowest error on the training set. On a simple but real SAR analysis problem the SVM outperforms several of the most frequently used machine learning techniques.<sup>25</sup>

### **3.4 Machine Learning: an indispensable tool in Bioinformatics**

Machine learning methods are essentially computer programs that make use of sampled data or past experience information to provide solutions to a given problem. A major focus of machine learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data; the difficulty lies in the fact that the set of all possible behaviors given all possible inputs is too large to be covered by the set of observed examples (training data). Hence the learner must generalize from the given examples, so as to be able to produce a useful output in new cases.<sup>10</sup>

Machine learning is able to deal with the huge volumes of data generated by novel high throughput devices, in order to extract hidden relationships that exist and that are not noticeable to experts. As new data and novel concept types are generated every day in molecular biology research, it is essential to apply techniques able to fit this fast-

evolving nature - Machine learning can be adapted efficiently to these changing environments.<sup>10</sup>

Machine Learning algorithms have proven to be of practical value for approximating nonlinear separable data, especially for classifying biological target data . Artificial neural networks (ANN) , support vector machines (SVM) , as well as decision trees (DT) have been applied in the past.

Fang *et al.* presented an effective application of SVMs in mining HTS data from a type I methionine aminopeptidases (MetAPs) inhibition study. This method was applied on a compound library of 43,736 organic small molecules and 50% of the active molecules could be recovered by screening just 7% of the test set. According to Plewczynski *et al.* , a SVM was able to achieve classification rates of up to 100% in evaluating the activity of compounds with respect to specific targets. Their overall hit rate, however, was somewhat lower, 80%. Stahura and Bajorath looked at several computational approaches, including SVMs, as a way to complement HTS.<sup>26</sup>

Burton *et al.* applied DTs in combination with a statistical learning method for predicting the CYP1A2 and CYP2D6 inhibition. CYP2D6 datasets provided eleven models with an accuracy of over 80%, while CYP1A2 datasets counted five high-accuracy models for HTS. The application of DTs in drug discovery is discussed by Rusinko *et al.* Their research focuses on a dataset with 1,650 monoamine oxidase inhibitors. Recently, Simmons *et al.* described an ensemble based DT model to virtually screen and prioritize compounds for acquisition. Fogel analyzed a combination of clustering and ANNs pre-screen compounds for HIV inhibition optimizing specificity and potency.<sup>26</sup>

Various machine learning methods have been used to predict diverse inhibitors of protein protein interactions. Collection of structurally diverse inhibitors of Protein Protein Interactions was compared against the FDA drug database and a subset of the ZINC database by machine learning methods which rely on classical QSAR descriptors. Descriptors were calculated by the program DRAGON 5 by Todeschini et al. A decision tree that contains three descriptors was obtained. The divide and-conquer algorithm (J48) was implemented in the software package WEKA 3.4.6 by Witten and Frank to construct

decision trees. Activity is expressed in a binary manner: T (true) for PPI inhibitor and F (false) for non-PPI inhibitor.<sup>27</sup>

Machine Learning techniques are successfully applied to establish quantitative relations between chemical structure and biological activity (QSAR), i.e. classify compounds as active or inactive with respect to a specific target biological system. This paper presents a comparison of Artificial Neural Networks (ANN), Support Vector Machines (SVM), and Decision Trees (DT) in an effort to identify potentiators of metabotropic glutamate receptor 5 (mGluR5), compounds that have potential as novel treatments against schizophrenia. When training and testing each of the three techniques on the same dataset enrichments of 61, 64, and 43 were obtained and an area under the curve (AUC) of 0.77, 0.78, and 0.63 was determined for ANNs, SVMs, and DTs, respectively. For the top percentile of predicted active compounds, the true positives for all three methods were highly similar, while the inactives were diverse offering the potential use of jury approaches to improve prediction accuracy.<sup>26</sup>

Machine learning algorithms have been taxonomized in the following way:

### **3.4.1 Supervised learning:**

Starting from a database of training data that consists of pairs of input cases and desired outputs, its goal is to construct a function(or model) to accurately predict the target output of future cases whose output value is unknown .When the target output is a continuous-value variable, the task is known as regression. Otherwise, when the output (or label) is defined as a finite set of discrete values, the task is known as classification.<sup>10</sup>

### **3.4.2 Unsupervised learning or clustering**

Starting from a database of training data that consists of input cases, its goal is to partition the training samples into subsets (clusters) so that the data in each cluster show a high level of proximity. In contrast to supervised learning, the labels for the data are not used or are not available in clustering.<sup>10</sup>

### **3.4.3 Semisupervised learning**

Starting from a database of training data that combines both labeled and unlabeled examples, the goal is to construct a model able to accurately predict the target output of

future cases for which its output value is unknown. Typically, this database contains a small amount of labeled data together with a large amount of unlabeled data.<sup>10</sup>

### **3.4.4 Reinforcement learning**

These algorithms are aimed at finding a policy that maps states of the world to actions. The actions are chosen among the options that an agent ought to take under those states, with the aim of maximizing some notion of long-term reward. Its main difference regarding the previous types of machine learning techniques is that input–output pairs are not present in a database, and its goal resides in online performance.<sup>10</sup>

### **3.4.5 Optimization**

This can be defined as the task of searching for an optimal solution in a space of multiple possible solutions. As the process of learning from data can be regarded as searching for the model that best fits the data, optimization methods can be considered an ingredient in modeling.<sup>10</sup>

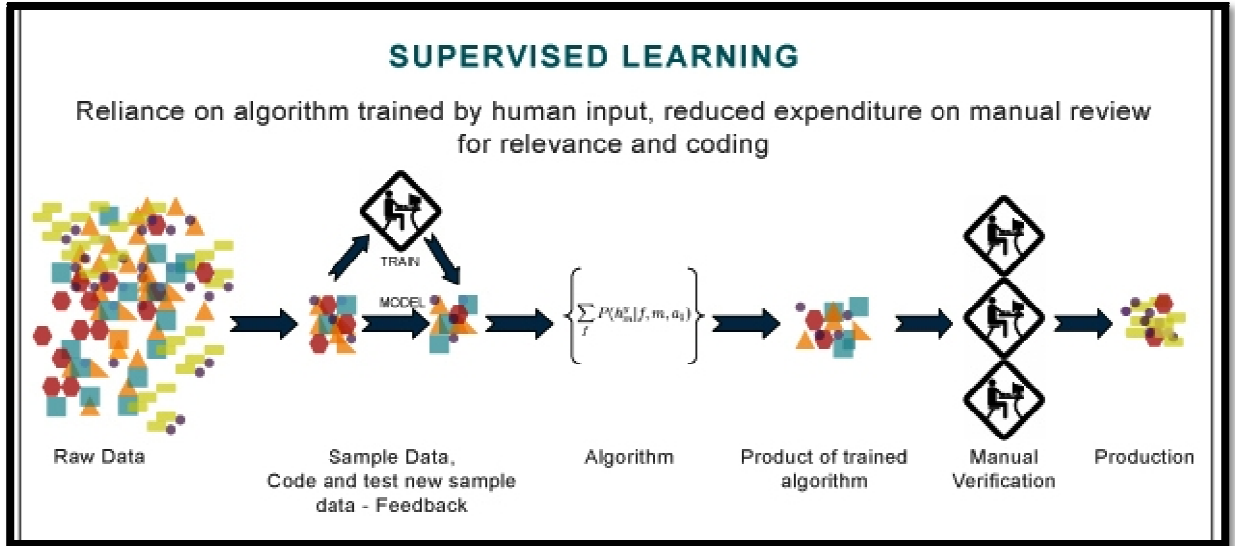
Supervised machine learning is the search for algorithms that reason from externally supplied instances to produce general hypotheses, which then make predictions about future instances. In other words, the goal of supervised learning is to build a concise model of the distribution of class labels in terms of predictor features. The resulting classifier is then used to assign class labels to the testing instances where the values of the predictor features are known, but the value of the class label is unknown. (Eibe Frank, Ian H. Witten & Mark A. Hall)

### **3.4.6 Steps involved in Supervised Learning**

- i. Determine the type of training example and before doing anything else one should decide what kind of data is to be used as an example.
- ii. Gather a training set: The training set needs to be representative of the real world use of the function. Thus a set of input objects is gathered and corresponding outputs are also gathered, either from human experts or from measurements.
- iii. Determine the input feature representation of the learned function. The accuracy of the learned function depends strongly on how the input object is represented.

Typically the input vector is transformed into a feature vector which contains a no. of features that are descriptive of the object. The no. of features should not be too large, because of the curse of dimensionality but should contain enough information to accurately predict the output.

- iv. Determine the structure of the learned function and corresponding learning algorithm. For example – one may choose the use of support vector machines or decision trees.
- v. Complete the design. Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset known as validation set of the training set.
- vi. Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.
- vii. A pair consisting of an object and its associated class is called a labeled example. The set of labeled example provided to learning algorithm is called the training set. Suppose we provide a training set to a learning algorithm and it outputs a classifier then the quality of the classifier can be evaluated by the approach that employs a second set of labeled examples called the test set where one can measure the percentage of test examples classified or misclassified.<sup>28</sup>
- viii. The reason we employ a separate test set is that most learned classifiers will be very accurate on the training examples. And a classifier that simply memorized the training example would be able to classify them perfectly.



### 3.5 Predictive Model

It is a model whose primary purpose is for prediction.

To create a good model that predicts well on future samples, you need to know,

- Your predictors and how they relate to each other.
- The mechanism that generated the data (sampling, technology etc).
- The measurement system and its error.

#### 3.5.1 Predictive modeling methods in R

There is range of predictive models available in R. These include:

1. **Parametric regression models:** ordinary/generalized/robust regression models; neural networks; partial least squares; projection pursuit regression; multivariate adaptive regression splines; principal component regression
2. **Sparse/penalized models:** ridge regression; the lasso; the elastic net; generalized linear models; partial least squares; nearest shrunken centroids; logistic regression
3. **Kernel methods:** support vector machines; relevance vector machines; least squares support vector machine; Gaussian processes
4. **Trees:** CART; C4.5; conditional inference trees; node harvest
5. **Ensembles:** random forest; boosting (trees, linear models, generalized additive models, generalized linear models; bagging (trees, multivariate adaptive regression splines)

6. **Prototype methods:** k nearest neighbors; learned vector quantization
7. **Discriminant analysis:** linear; quadratic; penalized; stabilized; sparse; mixture; regularized
8. **Others:** Naive Bayes; Bayesian multinomial probit models <sup>12</sup>

The predictive models that were built are Flexible discriminant analysis (MARS basis), Multivariate adaptive regression splines (earth), & Partial least squares (PLS)

### 3.6 Ensemble method

These are the learning algorithms that construct a set of classifiers and then classify new data sets by taking a vote of their predictions. The original ensemble method is Bayesian averaging but more recent algorithms include error correcting output coding, bagging and boosting. One key to successful ensemble methods is to construct individual classifiers with error rates below 0.5 whose errors are at least somewhat correlated. <sup>19</sup>

These ensemble methods combine multiple models into one usually more accurate than the best of its components. Two recent developments are,

- Importance sampling: It reveals ensemble methods- bagging, random forests and boosting to be special cases of a single algorithm, thereby showing how to improve their accuracy and speed.
- Rule ensembles: These are linear rule models derived from decision tree ensembles. They are the most interpretable version of ensembles, which is essential to applications such as credit scoring and fault diagnosis.



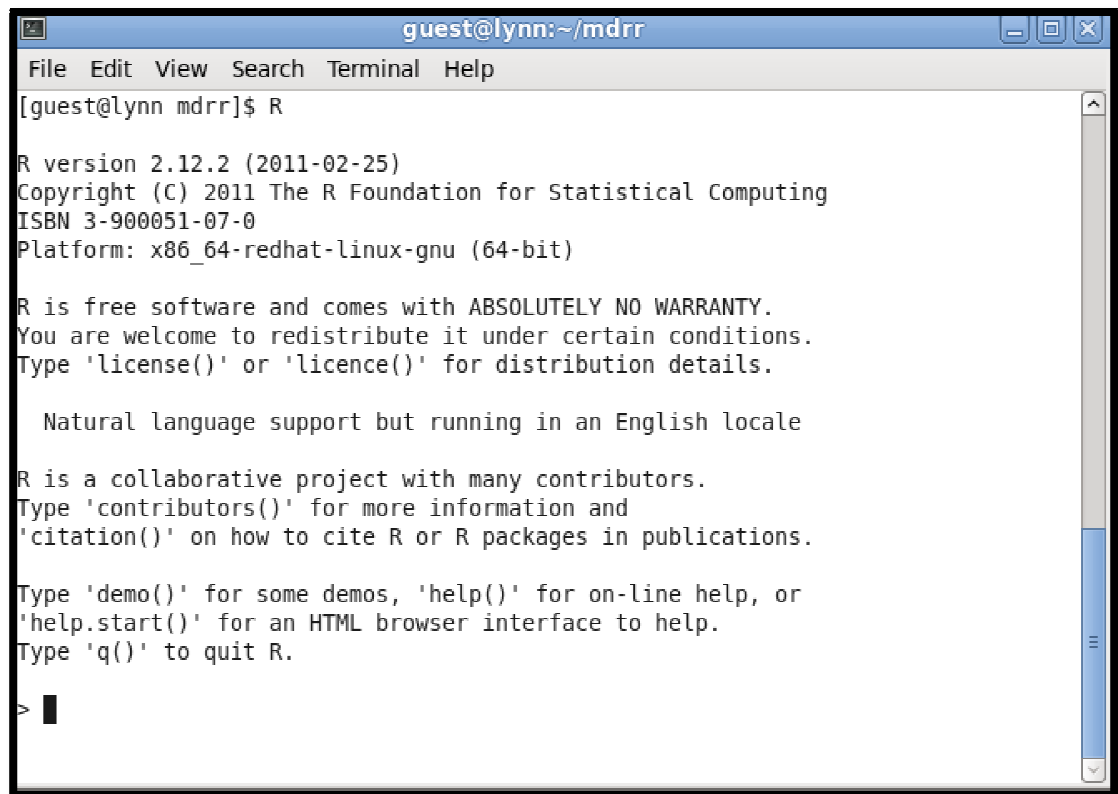
*MATERIALS*  
*&METHODS*

## 4. Materials & Methods

### 4.1 Materials

For the execution of whole of the process the basic requirements are:

- Programming language – R
- Package – CARET (**Classification and Regression Training**)

A screenshot of a terminal window titled "guest@lynn:~/mdrr". The terminal shows the command "R" being executed, which displays the R version 2.12.2 (2011-02-25) startup screen. The screen includes copyright information for The R Foundation for Statistical Computing, the platform "x86\_64-redhat-linux-gnu (64-bit)", and various help messages such as "R is free software and comes with ABSOLUTELY NO WARRANTY.", "Natural language support but running in an English locale", and "R is a collaborative project with many contributors." The terminal ends with a prompt ">".

```
guest@lynn:~/mdrr
File Edit View Search Terminal Help
[guest@lynn mdrr]$ R
R version 2.12.2 (2011-02-25)
Copyright (C) 2011 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-redhat-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

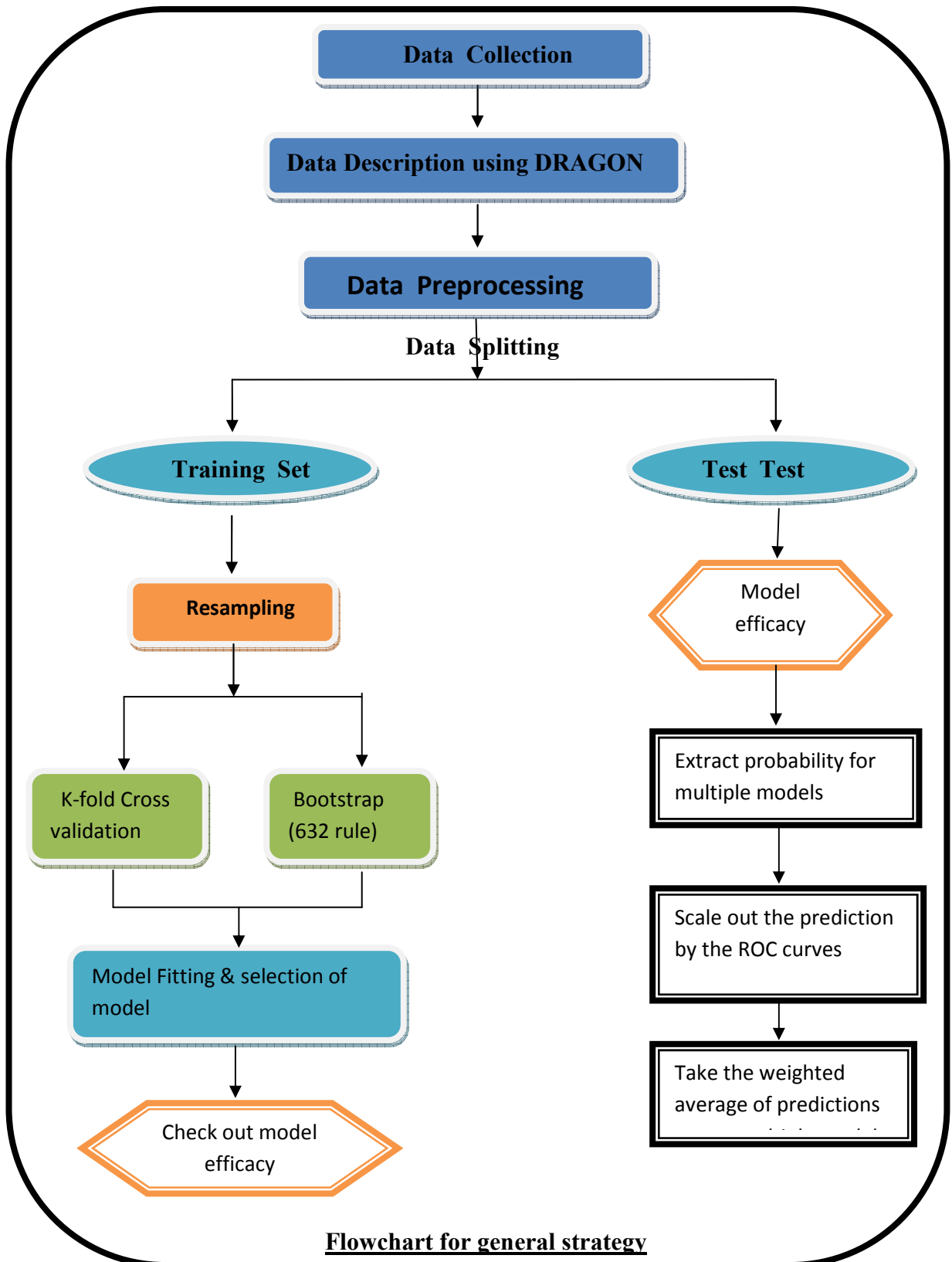
#### Shows the R language screen

- Condor: It is the product of the Condor Research Project at the University of Wisconsin-Madison (UW-Madison). Condor is a specialized workload management system for compute-intensive jobs. Like other full-featured batch systems, Condor provides a job queuing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. The main objective of distributed computing is to provide its users with a transparent, open and scalable

way to connect with its available resources so that it is more fault tolerant and more powerful than any of the possible stand alone systems.

- Dataset –Malaria data set (active and inactive compounds for *Plasmodium falciparum*). Given a list of chemicals from the **PubChem BioAssay**, the molecules in a relevant format (sdf format) were downloaded for the accession Ids which have the maximum number of tested compounds. The Following AIDs were taken: **504582, 504621, 504832, 504834 & 504690**

## 4.2 Methodology



**Flowchart for general strategy**

## 1. Data Collection

### Pubchem bioassay:-

It consists of deposited bioactivity data and descriptions of bioactivity assays used to screen the chemical substances contained in the PubChem Substance database, including descriptions of the conditions and the readouts (bioactivity levels) specific to the screening procedure. PubChem is a scientific showcase of the Molecular Libraries Program (MLP), a US National Institutes of Health (NIH) Roadmap Initiative (<http://mli.nih.gov/mli/>) that aims to enhance chemical biology efforts through high-throughput screening (HTS) so as to identify small molecule probes effective at modulating a given biological process or disease.

The 3D structures in sdf format for active and inactive compounds for the *Plasmodium falciparum* with the Accession Ids (AIDs) with maximum number of tested compounds were downloaded.

## 2. Data Description

**Descriptors calculation using Dragon:** Dragon 6.0 is an application for the calculation of molecular descriptors. These descriptors can be used to evaluate molecular structure-activity or structure-property relationships, as well as for similarity analysis and high-throughput screening of molecule databases.

Here, the script condor dragon trainee.pl was used to calculate the dragon descriptors from a set of sdf files for both actives and inactives.

The files obtained after running Dragon with the extension '.out' were taken. The header from all the '.out' files except one are removed. And then these files are merged to obtain the final data. This is done for both actives and inactives.

For the final file of actives and inactives obtained, a perl script is written to generate R ready input of descriptors. To select a number of inactives of the same size as actives, we use a threshold for the random number generator = **no of actives/no of inactives**.

Here, this ratio was obtained as **0.035795545**. This value was put in the perl script to obtain the data for training containing equivalent sets of actives and inactives.

### **3. Data preprocessing**

The data preprocessing task is subdivided as a set of relevant steps that could improve the quality & success, when applying machine learning modelization techniques. These procedures are considered “engineering” the input data. They refine the data to make it more tractable for machine learning schemes.<sup>10</sup>

In this chunk, the data set is preprocessed in order to discard off the non-zero values, highly correlated values, missing values, outliers, unbalanced distribution, and zero variance predictors. In case of removal of highly correlated values, the cutoff threshold limit is given for finding out correlated values and the values sharing the highest correlation are removed.

The data is first analyzed for 0 and Near Zero values. The values corresponding to this criteria is then removed using the function `nzv`. The correlation of the predictors is calculated. Values which had a correlation above 0.75 were removed.<sup>12</sup>

The preprocessing task was carried out using various functions from `caret`.

### **4. Principal Component Analysis**

This phase distilled out the 342 predictors down 3 variables i.e. principal component in a manner that attempts to maximize the amount of information preserved from the original predictor set. It reduces the dimension of the data set and presents it in the plot in such a way which makes task easy to interpret the class of compounds. Figures contain plots of the components responsible for percent variability in the original predictors (Kuhn, 2012).

### **5. Data splitting**

The script `Classification.Rnw` is initially used as the Sweave template for loading in the original dataset which then preprocesses the data, distills out the predictors down to principal components by reducing their dimensions and splits the data into training and test set. The training set is used for selecting model parameters, its values and model building and test set is used to get an independent assessment of model efficacy by using the command “`createDatapartition`” and according to the already defined variable “`pctTrain`” where the percentage for the split was mentioned.

## **6. Building and tuning models :**

The model building phase is done by running the **Classification-method.Rnw** script as the Sweave template to produce a Tex file. **This script** is used for building and describing classification models. A dummy set is defined in order to set values for variables used in the program to prevent error on parsing with Sweave and the variables “trainClass.RData” and “testClass.RData” is loaded into. In addition the model names are altered as well.

Here initially a dummy set is defined for defining variables for resample statistics such as for the ROC, sensitivity, specificity, kappa statistics, overall accuracy etc (Kuhn, 2012).The training data is resampled by Boot (632 rule) so as to check whether the choices for the parameters’ values are good or bad. It tries to inject variation in the system to approximate the model performance on future samples. <sup>12</sup>

In order to add more models we can even expand or contract the grid size taking the grid parameters of those models into account. This is called setting up of grid (Kuhn, 2012).

## **7. Model Fitting**

This is the model fitting step which is done to check the prediction on the held out data in the training set which was kept aside during the bootstrap resampling method and is saved in the appropriate “modelFit.RData”. Then a parameter tuning result is plotted in a graph taking the resample method .

## **8. Prediction on the test set**

The model efficacy or the performance estimation is done by testing the model fit results on the test set that was initially split. As a result we get ROC graphs and confusion matrix for test set and sensitivity, specificity and overall accuracy of the model where we can compare the observed values and predicted values of the test set.The results are written into a Tex format,which are converted into PDF files using 'pdflatex'.

## 9. Use of Ensemble Method:

An ensemble is itself a supervised learning algorithm, because it can be trained and then used to make predictions. The trained ensemble, therefore, represents a single hypothesis. This hypothesis, however, is not necessarily contained within the hypothesis space of the models from which it is built. Thus, ensembles can be shown to have more flexibility in the functions they can represent. This flexibility can, in theory, enable them to over-fit the training data more than a single model would, but in practice, some ensemble techniques tend to reduce problems related to over-fitting of the training data. <sup>10</sup>

Extract the probability for multiple methods like the flexible discriminant analysis, partial least squares, multiple adaptive regression splines in which the extract function loops through each model, runs the training and test & then extracts out the predictions & probabilities etc. Then we took the weighted average among all the single classifiers and the AUC for the test set was generated.

### Components for estimating performance of a model

**ROC graph:**-It is a graphical plot of the sensitivity or true positive rate versus false positive rate or (1-specificity) for a binary classifier system as its discrimination threshold is varied. Also known as relative operating characteristic curve because it is a comparison of 2 operating characteristics i.e. true positive rate & false positive rate) as the criterion changes. ROC analysis provides tools to select possibly optimal models and to discard suboptimal ones independently from the cost context or the class distribution.

**Sensitivity & specificity** are statistical measures of the performance of a binary classification test. Sensitivity measures the proportion of actual positives which are correctly identified as such and specificity measures the proportion of negatives which are correctly identified.



$$\text{sensitivity} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}}$$

$$\text{specificity} = \frac{\text{number of true negatives}}{\text{number of true negatives} + \text{number of false positives}}$$

**Confusion matrix:**-It is a matrix that gives a clear idea of the true positives, true negatives, false positives, and false negatives as the observed and predicted values in a given sample.

The predictive models that were built are Flexible discriminant analysis (MARS basis), Multivariate adaptive regression splines (earth), & Partial least squares (PLS)

### 4.1.1 Partial least squares (PLS)

Partial least squares is a popular method for soft modeling in industrial applications. It is a recent technique that generalizes and combines features from principal component analysis and multiple regression. It is used for constructing predictive models when the factors are many and highly collinear. It is particularly useful when we need to predict a set of dependent variables from a (very) large set of independent variables (i.e., predictors). It is an extension of the multiple regression model.<sup>29</sup>

In its simplest form, a linear model specifies the relationship between a dependent variable  $Y$  and a set of predictor variables the  $X$ 's, so that;

$$Y = b_0 + b_1 X_1 + b_2 X_2 + \dots + b_p X_p$$

In this equation,  $b_0$  is the regression coefficient for the intercept and the  $b_1$  values are the regression coefficients computed from the data.<sup>30</sup>

#### Goal of PLS

The  $I$  observations described by  $K$  dependent variables are stored in a  $I \times K$  **matrix denoted  $Y$** , the values of  $J$  predictors collected on these  $I$  observations are collected in the  $I \times J$  **matrix  $X$** .<sup>19</sup>

The goal of PLS regression is to predict  $Y$  from  $X$  and to describe their common structure. When  $Y$  is a vector and  $X$  is full rank, this goal could be accomplished using ordinary multiple regression. When the number of predictors is large compared to the number of observations,  $X$  is likely to be singular and the regression approach is no longer feasible (i.e., because of multicollinearity).

PLS regression is probably the least restricted of the various multivariate extensions of the multiple linear regression models. This flexibility allows it to be used in situations where the use of traditional multivariate methods is severely limited, such as when there are fewer observations than predictor variables. Furthermore, PLS regression can be used as an exploratory analysis tool to select suitable predictor variables and to identify outliers before classical linear regression. PLS is not usually appropriate for screening out factors that have a negligible effect on the response.<sup>29</sup>

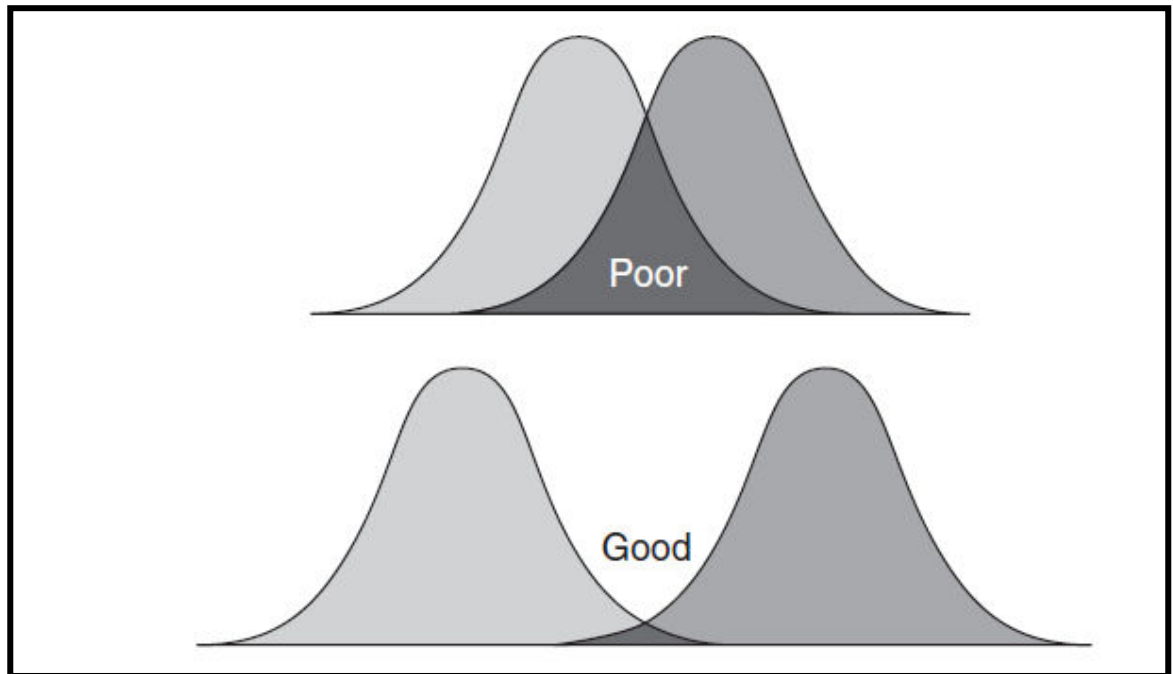
### 4.1.2 Flexible discriminant analysis (FDA)

Discriminant function analysis is used to determine which variables discriminate between two or more naturally occurring groups.

For example, an educational researcher may want to investigate which variables discriminate between high school graduates who decide (1) to go to college, (2) to attend a trade or professional school, or (3) to seek no further training or education. For that purpose the researcher could collect data on numerous variables prior to students' graduation. After graduation, most students will naturally fall into one of the three categories. *Discriminant Analysis* could then be used to determine which variable(s) are the best predictors of students' subsequent educational choice.

Computationally, discriminant function analysis is very similar to analysis of variance (*ANOVA*). Let us consider a simple example. Suppose we measure height in a random sample of 50 males and 50 females. Females are, on the average, not as tall as males, and this difference will be reflected in the difference in means (for the variable *Height*). Therefore, variable height allows us to discriminate between males and females with a better than chance probability: if a person is tall, then he is likely to be a male, if a person is short, then she is likely to be a female.<sup>31</sup>

There are two types of discriminant analysis, i.e., PDA and DDA, with different histories of development. Predictive discriminant analysis (PDA), or "classification" as it is sometimes called, generally includes "a set of predictor variables and one criterion variable, the latter being a grouping variable with two or more levels, that is, there are two or more groups" (Huberty & Barton, 1989, p. 158). **Predictive discriminant analysis** (PDA) is similar to multiple regression analysis except that PDA is used when the criterion variable is categorical and nominally scaled. **Descriptive Discriminant Analysis** (DDA) includes a collection of techniques involving two or more criterion variables and a set of one or more grouping variables, each with two or more levels, whose effects are assessed through MANOVA (Multivariate Analysis of Variance)



### **Good and Poor Discriminant Analysis**

#### **Stepwise Discriminant Analysis**

Probably the most common application of discriminant function analysis is to include many measures in the study, in order to determine the ones that discriminate between groups. For example, an educational researcher interested in predicting high school graduates' choices for further education would probably include as many measures of personality, achievement motivation, academic performance, etc. as possible in order to learn which one(s) offer the best prediction.

**Model:** Here we want to build a "model" of how we can best predict to which group a case belongs. In the following discussion we will use the term "in the model" in order to refer to variables that are included in the prediction of group membership, and we will refer to variables as being "not in the model" if they are not included.

**Forward stepwise analysis:** In stepwise discriminant function analysis, a model of discrimination is built step-by-step. Specifically, at each step all variables are reviewed

and evaluated to determine which one will contribute most to the discrimination between groups. That variable will then be included in the model, and the process starts again.

**Backward stepwise analysis:** One can also step backwards; in that case all variables are included in the model and then, at each step, the variable that contributes least to the prediction of group membership is eliminated. Thus, as the result of a successful discriminant function analysis, one would only keep the "important" variables in the model, that is, those variables that contribute the most to the discrimination between groups.

***F to enter, F to remove:*** The stepwise procedure is "guided" by the respective *F* to enter and *F* to remove values. The *F* value for a variable indicates its statistical significance in the discrimination between groups, that is, it is a measure of the extent to which a variable makes a unique contribution to the prediction of group membership. If you are familiar with stepwise multiple regression procedures, then you may interpret the *F* to enter/remove values in the same way as in stepwise regression.

**Capitalizing on chance:** A common misinterpretation of the results of stepwise discriminant analysis is to take statistical significance levels at face value. By nature, the stepwise procedures will capitalize on chance because they "pick and choose" the variables to be included in the model so as to yield maximum discrimination. Thus, when using the stepwise approach the researcher should be aware that the significance levels do not reflect the true *alpha* error rate, that is, the probability of erroneously rejecting  $H_0$  (the null hypothesis that there is no discrimination between groups).<sup>32</sup>

### **4.1.3 Multivariate adaptive regression splines (MARS)**

It is a non-parametric regression technique and can be seen as an extension of linear models that automatically models non-linearities and interactions between variables. **The earth package is an implementation of Jerome Friedman's Multivariate Adaptive Regression Splines, commonly known as "MARS". The earth source code is licensed under the GPL and runs in an R environment, or can be used as a stand-alone C library.**

MARS is an *adaptive* procedure because the selection of basis functions is data-based and specific to the problem at hand. This algorithm is a nonparametric regression procedure that makes no specific assumption about the underlying functional relationship between the dependent and independent variables.

The MARSplines technique has become particularly popular in the area of **data mining** because it does not assume or impose any particular type or class of relationship (e.g., linear, logistic, etc.) between the predictor variables and the dependent (outcome) variable of interest. Instead, useful models (i.e., models that yield accurate predictions) can be derived even in situations where the relationship between the predictors and the dependent variables is non-monotone and difficult to approximate with parametric models.

Multivariate Adaptive Regression Splines makes no assumption about the underlying functional relationship between the dependent and independent variables. Instead, MARSplines constructs this relation from a set of coefficients and so-called basis functions that are entirely determined from the regression data.

**The MARSplines model:** The basis functions together with the model parameters (estimated via **least squares estimation**) are combined to produce the predictions given the inputs. The general MARSplines model equation is given as:

$$y = f(X) = \beta_0 + \sum_{m=1}^M \beta_m h_m(X)$$

where the summation is over the  $M$  nonconstant terms in the model. In this equation,  $y$  is predicted as a function of the predictor variables  $X$  (and their interactions); this function consists of an intercept parameter ( $\beta_0$ ) and the weighted (by  $\beta_m$ ) sum of one or more basis functions  $h_m(X)$  <sup>33</sup>

## Algorithm

Implementing MARSplines involves a two step procedure that is applied successively until a desired model is found. In the first step, we build the model, i.e. increase its complexity by adding basis functions until a preset (user-defined) maximum level of complexity has been reached. Then we begin a backward procedure to remove the least significant basis functions from the model, i.e. those whose removal will lead to the least reduction in the (least-squares) goodness of fit. This algorithm is implemented as follows:

1. Start with the simplest model involving only the constant basis function.
2. Search the space of basis functions, for each variable and for all possible knots, and add those which maximize a certain measure of goodness of fit (minimize prediction error).
3. Step 2 is recursively applied until a model of pre-determined maximum complexity is derived.
4. Finally, in the last stage, a pruning procedure is applied where those basis functions are removed that contribute least to the overall (**least squares**) goodness of fit.

## Applications

- Multivariate Adaptive Regression Splines have become very popular recently for finding predictive models for "difficult" data mining problems, i.e., when the predictor variables do not exhibit simple and/or monotone relationships to the dependent variable of interest.
- MARSplines selects predictors (basis functions) for the model in a specific manner so it generally works "well" in situations where regression-tree models are also appropriate, i.e., where hierarchically organized successive splits on the predictor variables yield good (accurate) predictions.
- In fact, instead of considering this technique as a generalization of multiple regression, MARSplines may be considered as a generalization of regression trees, where the "hard" binary splits are replaced by "smooth" basis functions.<sup>33</sup>

# *RESULTS*



## 5. Results

### Data Set:

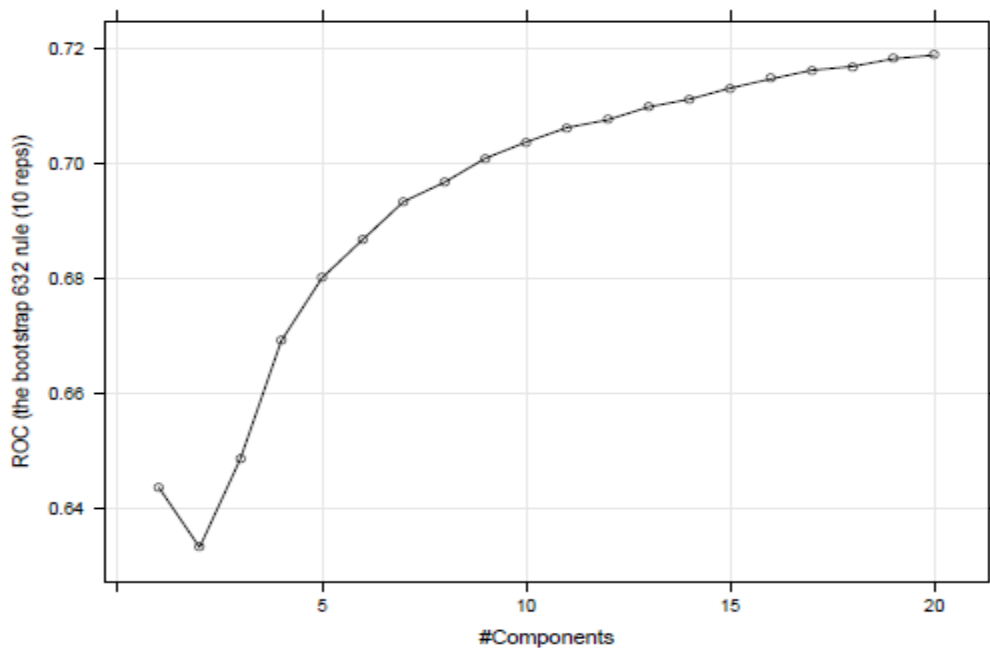
The initial data set consisted of 65367 samples and 799 predictor variables. The breakdown of the outcome data classes were: \active" (n=32712) and \inactive" (n=32655).

### 1. Partial Least Squares (PLS)

#### Model Building:

The parameter tuning values of different models are calculated in such a way that makes the interpretation quite easy to visualize from the graph itself. The ROC values of training sets are taken in the Y axis and the parameter values of a particular model is taken in the X axis.

There is 1 tuning parameter associated with this model: the number of components. To choose an appropriate value of the tuning parameter, the bootstrap 632 rule (10 reps) was used to generated a profile of performance across the 20 candidate values.



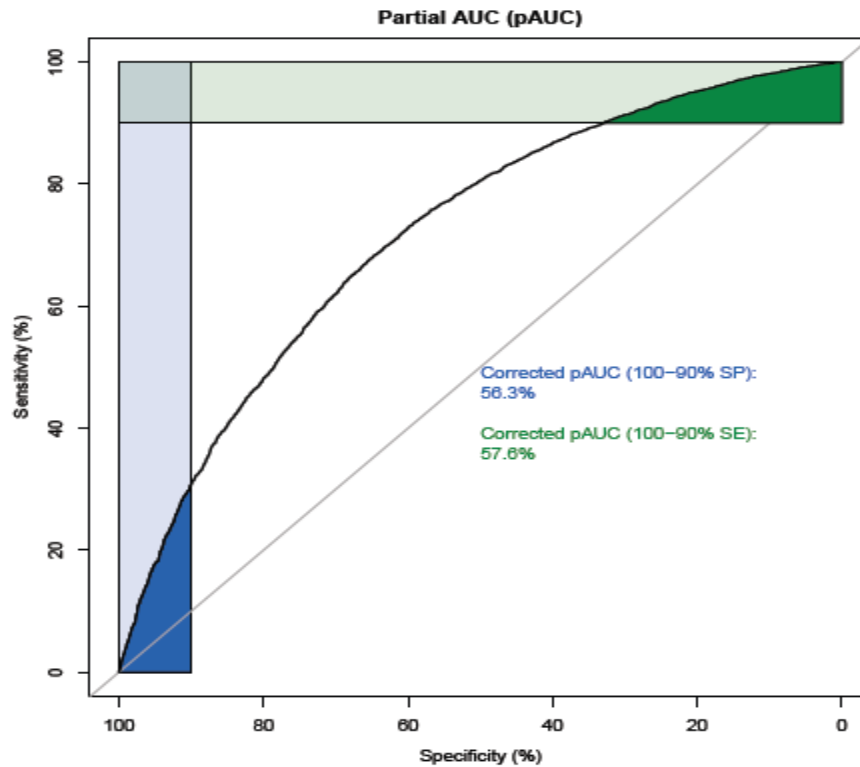
**GRAPH I: A plot of the estimates of the ROC values calculated using the bootstrap 632 rule (PLS).**

### Test Set Results

	OBSERVED VALUES	
	ACTIVE	INACTIVE
ACTIVE	5122	2436
INACTIVE	3055	5727

**TABLE I: The confusion matrix for the test (PLS) set**

Based on the test set of 16340 samples, the overall accuracy was 0.664, the Kappa statistic was 0.328, the p-value that accuracy is greater than the no-information rate was  $< 2.22e-16$ , the p-value of concordance from McNemar's test was  $< 2.22e-16$ , the sensitivity was 0.719, the specificity was 0.626 and the area under the ROC curve was 0.702. Using the bootstrap 632 rule (10 reps), the training set estimates were area under the ROC curve was 0.719, sensitivity was 0.63 and specificity was 0.698.

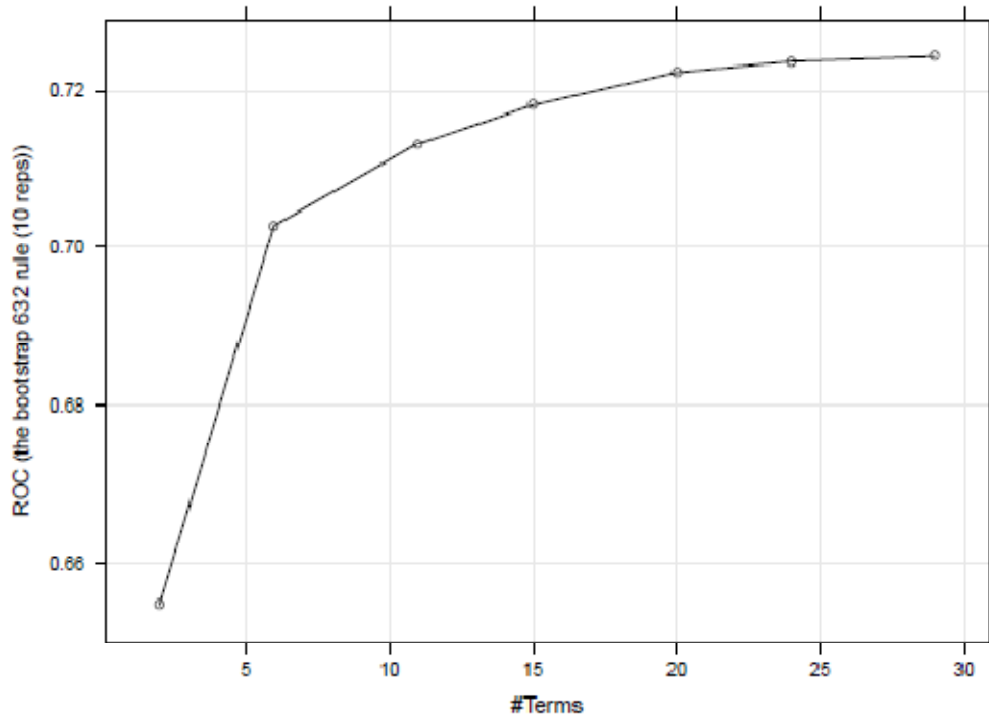


**GRAPH II: ROC Curve for the test set (PLS)**

## 2. Flexible discriminant analysis (FDA)

### Model Building:

There are 2 tuning parameters associated with this model: the number of terms and product degree. To choose appropriate values of the tuning parameters, the bootstrap 632 rule (10 reps) was used to generate a profile of performance across the 7 combinations of the tuning parameters.



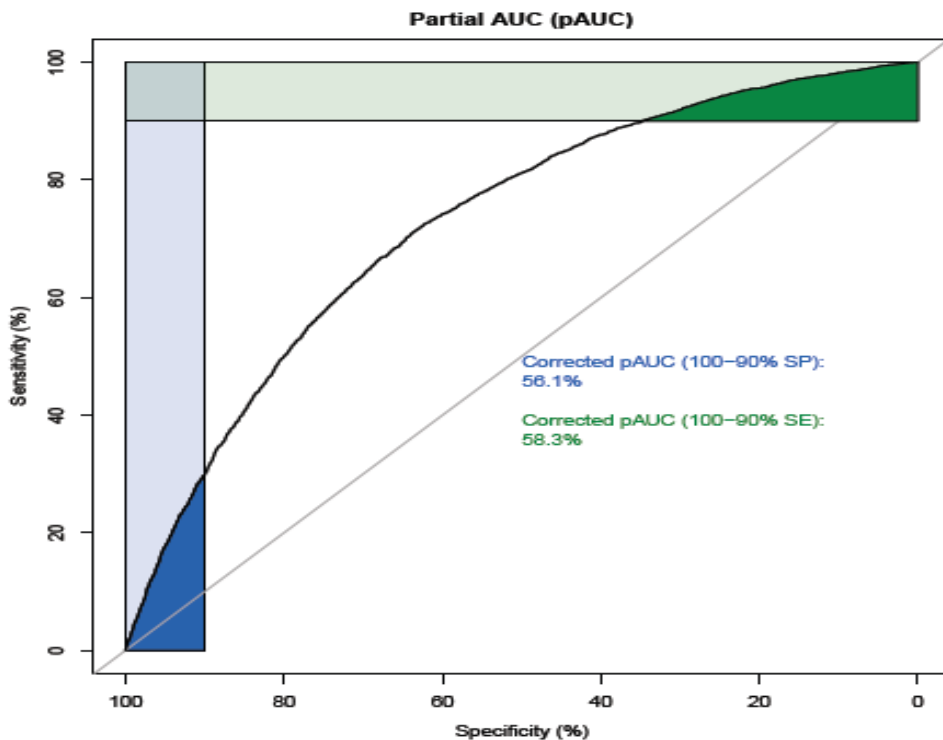
**GRAPH III: A plot of the estimates of the ROC values calculated using the bootstrap 632 rule (FDA)**

## Test Set Results

	OBSERVED VALUES	
	ACTIVE	INACTIVE
ACTIVE	5346	2544
INACTIVE	2831	5619

**TABLE II: The confusion matrix for the test set (FDA)**

Based on the test set of 16340 samples, the overall accuracy was 0.671, the Kappa statistic was 0.342, the p-value that accuracy is greater than the no-information rate was  $<2e-16$ , the p-value of concordance from McNemar's test was  $1e-04$ , the sensitivity was 0.726, the specificity was 0.654 and the area under the ROC curve was 0.688. Using the bootstrap 632 rule (10 reps), the training set estimates were area under the ROC curve was 0.724, sensitivity was 0.651 and specificity was 0.686.

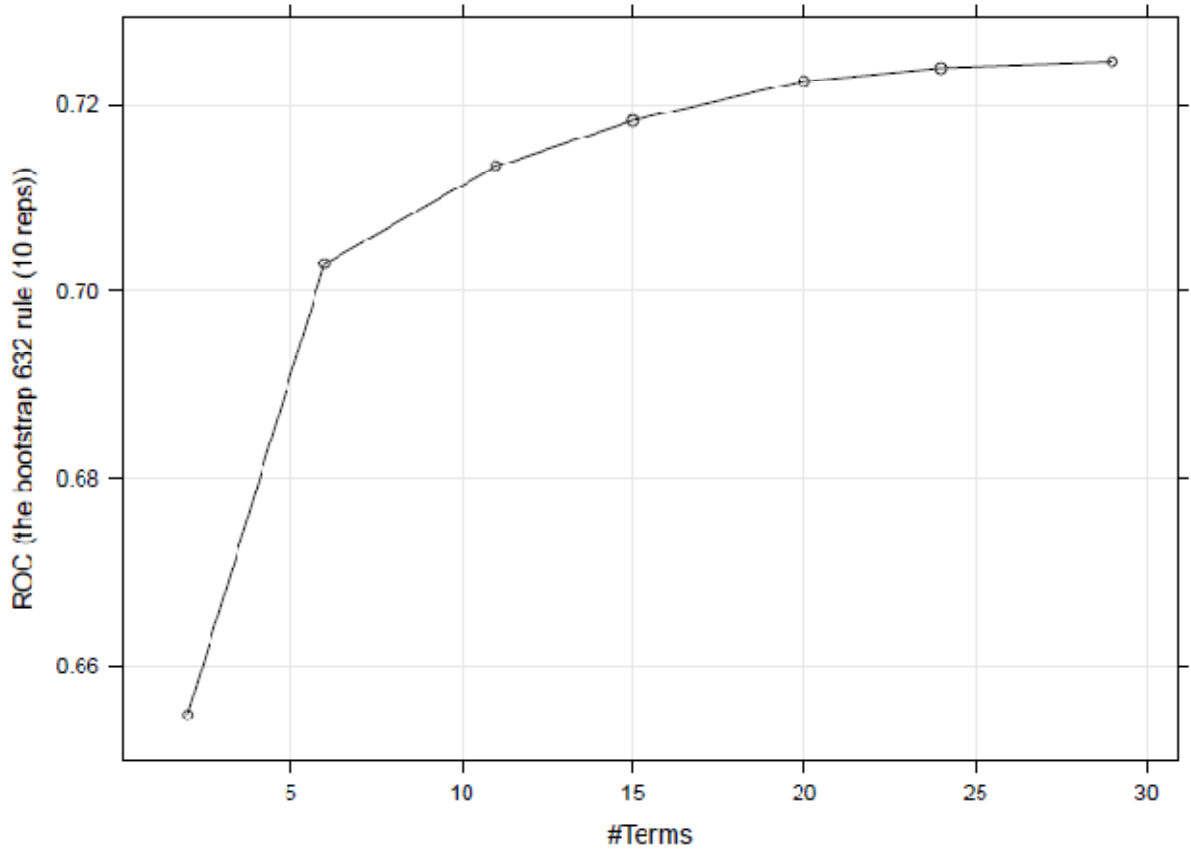


**GRAPH IV: ROC Curve for the test set (FDA)**

### 3. Multivariate adaptive regression splines – model name –earth

#### Model Building :

There are 2 tuning parameters associated with this model: the number of terms and product degree. To choose appropriate values of the tuning parameters, the bootstrap 632 rule (10 reps) was used to generated a profile of performance across the 7 combinations of the tuning parameters.



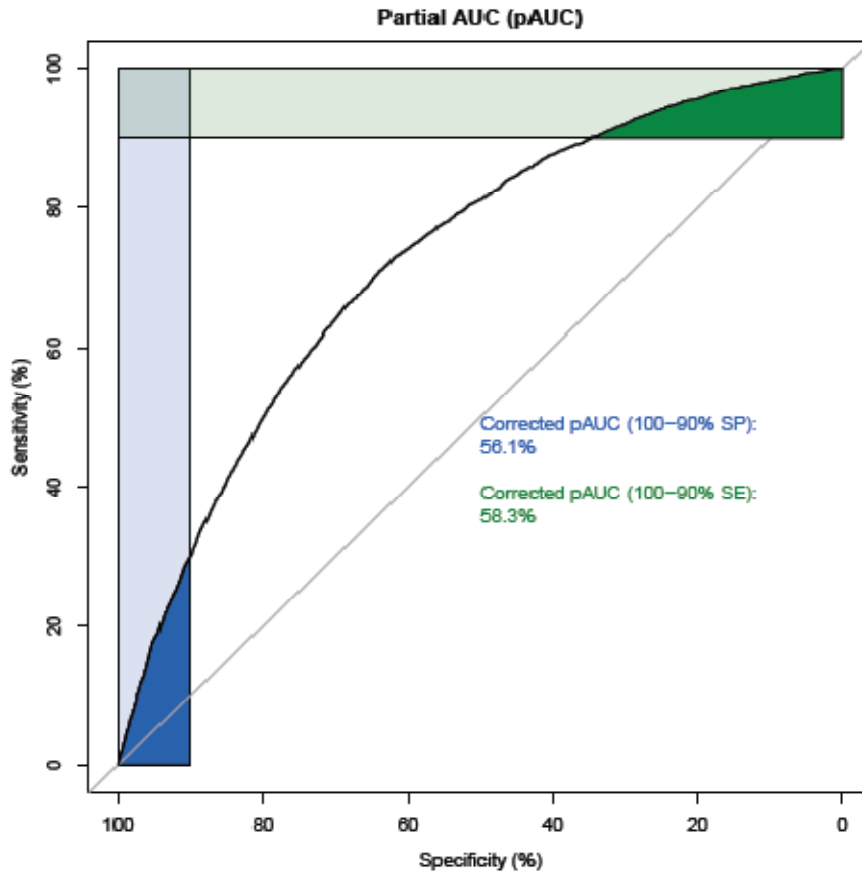
**GRAPH V: A plot of the estimates of the ROC values calculated using the bootstrap 632 rule (earth)**

**Test Set Results:**

	OBSERVED VALUES	
	ACTIVE	INACTIVE
ACTIVE	5359	2552
INACTIVE	2818	5611

**TABLE III: The confusion matrix for the test set (earth)**

Based on the test set of 16340 samples, the overall accuracy was 0.671, the Kappa statistic was 0.343, the p-value that accuracy is greater than the no-information rate was  $<2e-16$ , the p-value of concordance from McNemar's test was  $3e-04$ , the sensitivity was 0.726, the specificity was 0.655 and the area under the ROC curve was 0.687. Using the bootstrap 632 rule (10 reps), the training set estimates were area under the ROC curve was 0.724, sensitivity was 0.653 and specificity was 0.684.



**GRAPH VI: ROC for the test set (earth)**

Class probabilities for the test set. Each panel contains separate classes

FIGURE I: Class probabilities for the test set using (Earth)

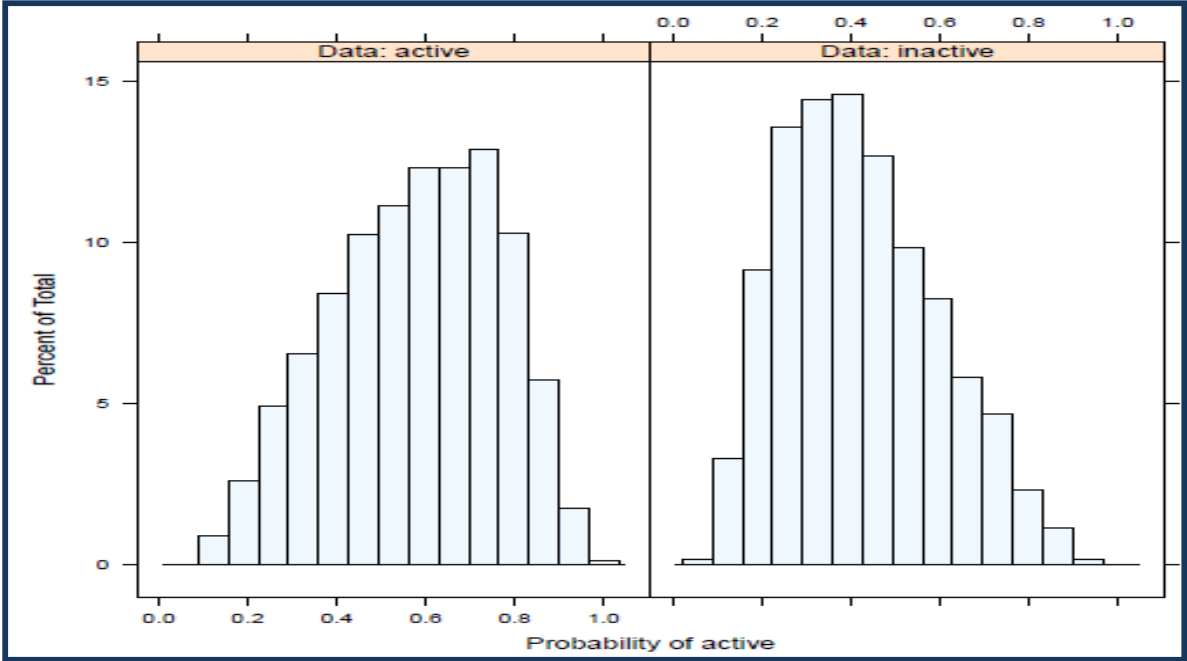
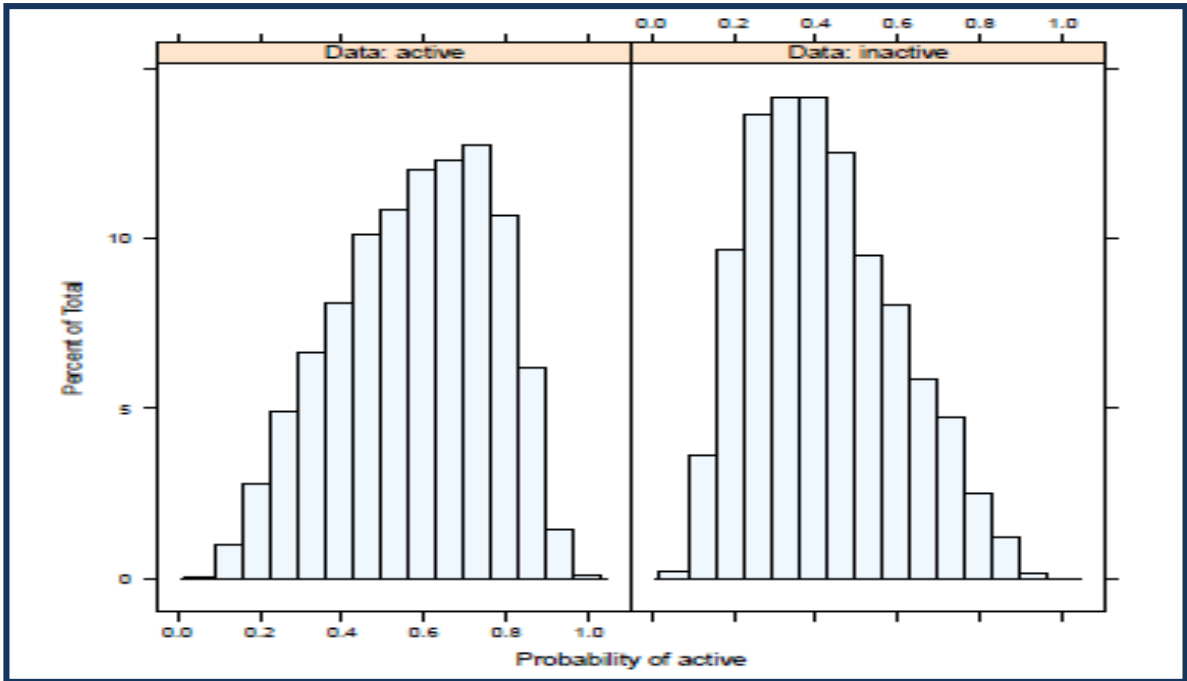
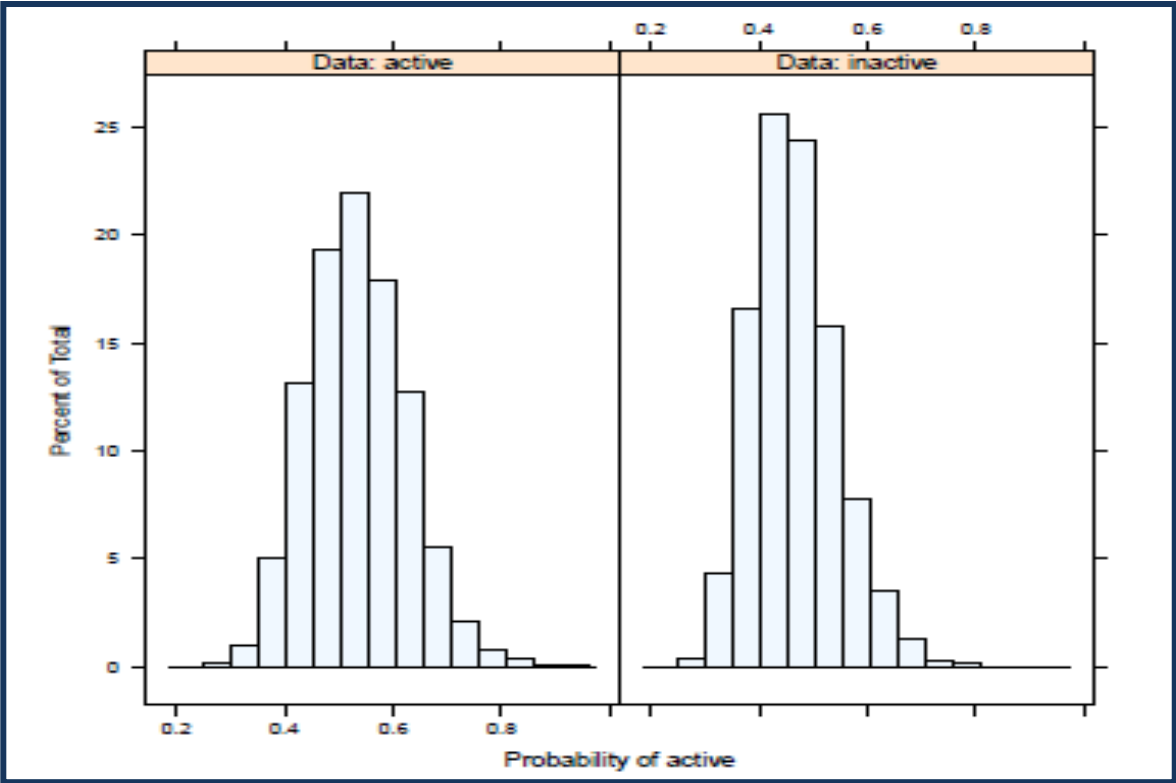


FIGURE II: Class probabilities for the test set (FDA)



**FIGURE III: Class probabilities for the test set( PLS)**





## Ensemble Results

Table IV: Confusion Matrix (Ensemble)

PREDICTION	ACTIVE	INACTIVE
ACTIVE	5305	2476
INACTIVE	2872	5687

### Statistics

Accuracy: 0.6727

95% CI : (0.6654, 0.6799)

No Information Rate : 0.5004

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.3454

Mcnemar's Test P-Value : 6.615e-08

Sensitivity : 0.6488

Specificity : 0.6967

Pos Pred Value : 0.6818

Neg Pred Value : 0.6644

Prevalence : 0.5004

Detection Rate : 0.3247

Detection Prevalence : 0.4762

'Positive' Class: active

### 5.1 Result Analysis:

<b>Model name</b>	<b>Overall Accuracy</b>	<b>Area under ROC (test)</b>	<b>Area under ROC (train)</b>
PLS	0.664	0.702	0.719
FDA	0.671	0.688	0.724
earth	0.671	0.687	0.724

**TABLE IV: The ROC values and overall accuracy for the mentioned methods**

<b>Model name</b>	<b>Sensitivity</b>	<b>Specificity</b>	<b>Kappa Statistic</b>
PLS	0.719	0.626	0.328
FDA	0.726	0.654	0.342
earth	0.726	0.655	0.343

**TABLE V: Performance score of the Predictive Models**

A **confusion matrix** is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class. In this predictive analysis, a **confusion matrix**, is a table with two rows and two columns that reports the number of *false positives*, *false negatives*, *true positives*, and *true negatives*.<sup>10</sup>

A **receiver operating characteristic (ROC)**, or simply **ROC curve**, is a graphical plot which illustrates the performance of a binary classifier system. It is created by plotting the fraction of true positives out of the positives (TPR = true positive rate) vs. the fraction of false positives out of the negatives (FPR = false positive rate), at various threshold settings. TPR is also known as sensitivity, and FPR is one minus the specificity or true negative rate. ROC analysis provides tools to select possibly optimal models and to discard suboptimal ones independently from the class distribution. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold.<sup>19</sup>

The value for the *area under the ROC curve* can be interpreted as follows: an area of 0.664, for example, means that a randomly selected individual from the positive group has a test value larger than that for a randomly chosen individual from the negative group in 66.4% of the time. When the variable under study cannot distinguish between the two groups, i.e. where there is no difference between the two distributions, the area will be equal to 0.5 (the ROC curve will coincide with the diagonal). When there is a perfect separation of the values of the two groups, i.e. there no overlapping of the distributions, the area under the ROC curve equals 1 (the ROC curve will reach the upper left corner of the graph).

McNemar's test is a test of equality of the marginal distributions in a 2 by 2 classification table and, therefore, a test of symmetry. The P-value is the probability that the Area under the ROC curve (like e.g. 0.947) is found when in fact, the true (population) Area under the ROC curve is 0.5 (null hypothesis: Area = 0.5). If P is low (P<0.05) then it can be concluded that the Area under the ROC curve is significantly

different from 0.5 and that therefore there is evidence that the test does have an ability to distinguish between the two groups.<sup>19</sup>

A test with perfect discrimination (no overlap in the two distributions) has a ROC curve that passes through the upper left corner (100% sensitivity, 100% specificity). Therefore the closer the ROC curve is to the upper left corner, the higher the overall accuracy of the test .

The Kappa statistic corrects the simple inter-rater agreement for chance agreement, and thus provides a better estimate in most cases of the reliability of categorical behavioural observation data. The Kappa ratio ( $\kappa$ ) expresses the corrected proportion of agreement between raters as a ratio of the total proportion of agreements corrected for chance responses. If the raters are in complete agreement then  $\kappa = 1$ . If there is no agreement among the raters other than what would be expected by chance.<sup>19</sup>

# *DISCUSSION*

## 6. Discussion

Humans have suffered from the burden of malarial infections for thousands of years, and the disease has greatly influenced human evolution and history. Malaria is a serious worldwide health problem due to the emergence of parasites that are resistant to well-established antimalaria drugs.

Different methods for building predictive models of the relationships between molecular structure and useful properties are becoming increasingly important for drug discovery. Tools for designing libraries and extracting information from molecular data bases and high-throughput screening experiments robustly and quickly enable leads to be discovered more effectively. One of the most effective applications comes from the emerging science and industry of knowledge discovery in databases, also known as **data mining**.<sup>1</sup>

With the need for million compounds chemical libraries, cheminformatics is now playing a key role in many aspects of drug discovery and drug development. Machine Learning techniques are successfully applied to establish quantitative relations between chemical structure and biological activity (QSAR), i.e. classify compounds as active or inactive with respect to a specific target biological system.<sup>10</sup>

So we have modelled various structure activity classifiers for *P.falciparum* using the caret (**Classification and regression training**) package in R. The caret contains functions to stream-line the model training process for complex regression and classification problems. The package focuses on simplifying model training and tuning across a wide variety of modeling techniques. It also includes methods for pre-processing training data, calculating variable importance, and model visualizations. The script used for building models uses ‘**Sweave**’ which is a function in the statistical programming language R that enables integration of R code into LaTeX documents. In this work we have used supervised learning methods as machine learning approaches for predicting active accuracy of compounds.

There are a few data sets included in caret like **Multidrug Resistance Reversal (MDRR) Agent Data**. The original response variable is a ratio measuring the ability of a compound to reverse a leukemia cell's resistance to adriamycin. However, the problem

was treated as a classification problem, and compounds with the ratio  $>4.2$  were considered active, and those with the ratio  $\leq 2.0$  were considered inactive. Compounds with the ratio between these two cutoffs were called moderate and removed from the data for two class classification, leaving a set of **528 compounds (298 actives and 230 inactives)**. When the predictive models were built for this dataset the overall accuracy ranged from 0.8 to 0.9 which is quite accurate.

The malaria dataset used here is a ‘big’ data containing large number of samples (more than 60000) and a large number of predictor variables. The accuracy of the for this large dataset vary and was reduced due to data overlap which reduces the sensitivity and specificity of the classifiers.

The Classification models were built and trained with molecular descriptors calculated by DRAGON. The models that were built are: **Flexible discriminant analysis (FDA), Multivariate adaptive regression splines (earth) & Partial least squares (PLS)**

Partial Least Squares (PLS) is a versatile tool for the analysis of high-dimensional genomic data. For structure-activity correlation, Partial Least Squares (PLS) has many advantages over regression, including the ability to robustly handle more descriptor variables than compounds, nonorthogonal descriptors and multiple biological results, while providing more predictive accuracy and a much lower risk of chance correlation. PLS methods are in general characterised by high computational and statistical efficiency. They also offer great flexibility and versatility in terms of the analysis problems.<sup>35</sup>

Discriminant Analysis may be used when we want to *assess* the adequacy of classification, given the group memberships of the objects under study; or we wish to *assign* objects to one of a number of (known) groups of objects. **Flexible discriminant analysis** recasts Linear Discriminant Analysis as a linear regression problem and substitutes linear regression by a non parametric one which implicitly enlarges the basis of the vector space. This analysis can allow multiple dependent variables with reduced error rates. It allows easier interpretation of Between-group Differences as each discriminant function measures something unique and different. But they assume that the relationships between variables are assumed to be linear in all groups.<sup>19</sup>

**Multivariate adaptive regression splines (MARS)** It is a nonparametric method that estimates complex nonlinear relationships by a series of spline functions of the independent predictors. MARS models are more flexible than linear regression models. It can handle both continuous and categorical data. Building MARS models often requires little or no data preparation. It is suitable for handling fairly large datasets and large number of predictor values and makes quick prediction on data. These models do not give as good fits as boosted trees, but can be built much more quickly and are more interpretable. <sup>19</sup>

**These models showed different overall accuracy. Their respective overall accuracies were calculated: Flexible discriminant analysis (MARS basis)-0.671, Multivariate adaptive regression splines (earth)-0.671, & Partial least squares (PLS)-0.664**

**These results were not very accurate so in order to improve the result, we have employed ensemble method which combines** multiple models to obtain better predictive performance than could be obtained from any of the constituent models. This was done to obtain better accuracy than the single classifier. Ensemble learning is primarily used to improve the (classification, prediction, function approximation, etc.) performance of a model, or reduce the likelihood of an unfortunate selection of a poor one.

The probability for all the three methods was extracted and weighted average among all the single classifiers and the AUC for the test set was generated. The accuracy was slightly improved using this method. The accuracy finally obtained was 0.6727. These results can be further improved by using more methods with better accuracy and different variable.

Prior to any direct application of machine learning algorithms, it is essential to be conscious of the quality of the initial raw data available, the lack of data quality will lead to poor quality in the mined results. As a result, the need to ensure a minimum quality of the data – which might require among other decisions, to discard a part of the original data – is critical, especially in the field. The data pre-processing is the most important



task in data mining. These generally include missing value imputation, data normalization, and discretization.<sup>10</sup>

They refine the data to make it more tractable for machine learning methods. Quality of result strongly depends on the quality of input data, and therefore the preprocessing step is crucial. Therefore choosing the right data mining technique and proper refining of data may further lead to improved models with better accuracy.<sup>10</sup>

*CONCLUSION*

## 7. Conclusion

Thus we can conclude that Data Mining is an analytic process designed to explore data in search of consistent patterns and/or systematic relationships between variables, and then to validate the findings by applying the detected patterns to new subsets of data. **In this case, the dataset used was the active and inactive compounds for Plasmodium falciparum.**

Here an attempt was made to build predictive relationships between the structure of a chemical and some observed endpoint, such as activity against a biological target. Using the structural formula of a compound, chemical descriptors were generated that attempt to capture specific characteristics of the chemical, such as its size, complexity, etc. The whole procedure involves the usage of R programming along with Caret package. Currently, caret works with a large number of existing modeling functions and will continue to add new models as they are developed. Structure Activity Classifiers were built that use these descriptors to predict the outcome of interest. These models can be applied for prediction or classification of new data. Here, these models can further classify the new test set compounds as active or inactive.

Data Mining can be further applied to different fields of Bioinformatics including gene finding, protein function domain detection, function motif detection, protein function inference, disease diagnosis, disease prognosis, disease treatment optimization, protein and gene interaction network reconstruction, data cleansing, and protein sub-cellular localization prediction.

# *BIBLIOGRAPHY*

## 8. Bibliography

1. Guha R, Gilbert K, Fox G, Pierce M, Wild D , Yuan H. Advances in Cheminformatics Methodologies and Infrastructure to Support the Data Mining of Large, Heterogeneous Chemical Datasets. *Current computer-aided drug design*.2010 March;6(1) : 50-67 .
2. Villar HO, Hansen MR. Mining and visualizing the chemical content of large databases. *Curr Opin Drug Discov Devel*. 2009 May;12(3):367-75.
3. Jónsdóttir OS , Jørgensen FX and Brunak S.Prediction methods and databases within chemoinformatics: emphasis on drugs and drug candidates.*Bioinformatics*. 2005 21(10):2145–2160
4. Review1 of Handbook of Chemoinformatics Algorithms by Faulon, Bender, eds. CRC Press, 2010 440 pages, hardcover )
5. Balakin K V .Pharmaceutical Data Mining: Approaches and Applications for Drug Discovery. John Wiley & Sons, 2009:77-8
6. Klein C T, Kaiblinger N, Wolschann P. Internally defined distances in 3D-quantitative structure-activity relationships. *Journal of Computer-Aided Molecular Design*, 2002  
16: 79–93
7. Todeschini R and Consonni V. *Molecular Descriptors for Chemoinformatics*, WILEY-VCH, Weinheim (Germany) 2009, 1257
8. Lianga Z, Xinmingb T, Land WU , Lina LI, Zhongyuana W . DATA MINING AND ITS APPLICATION IN DATABASE CONTENT REFINEMENT
9. Athauda R, Tissera M , Fernando C. Data Mining Applications: Promise and Challenges .*DataMining and Knowledge Discovery in Real Life Applications*,
10. Inza I, Calvo B, Armañanzas R, Bengoetxea E, Larrañaga P, Lozano JA. Machine learning: an indispensable tool in bioinformatics.*Methods Mol Biol*. 2010; 593:25-48.
11. Raza K . APPLICATION OF DATA MINING IN BIOINFORMATICS  
*Indian Journal of Computer Science and Engineering* .1 (2): 114-118

12. Kuhn M. Building Predictive Models in R Using the caret Package. November 2008, 28( 5)
13. Kuhn M . Data Sets and Miscellaneous Functions in the caret Package.March, 2012
14. [http://www.kmining.com/info\\_definitions.html](http://www.kmining.com/info_definitions.html)
15. Brachman, R.J., and Anand, T. The Process Of Knowledge Discovery In Databases: A Human-Centered Approach. In Advances In Knowledge Discovery And Data Mining 1996,37-57.
16. <http://www.statsoft.com/textbook/data-mining-techniques/>
17. From Data Mining to Knowledge Discovery in Databases, Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. AI Magazine . 1996 .17(3): 37-54
18. Raza Ali (425), Usman Ghani (462), Aasim Saeed (464). Data Clustering and Its Applications
19. Witten IH, Frank E,. Hall MA . Data Mining: Practical Machine Learning Tools and Techniques . Elsevier, 30-Jan-2011:364-5
20. Kaur G, Singh L .Data Mining: An Overview. IJCSt , Ju n e 2011 , 2(2)
21. Xu J and Hagler A .Chemoinformatics and Drug Discovery. *Molecules* 2002, 7, 566-600
22. Mauri A, Consonni V, Pavan M, Todeschini R . Dragon software: An easy approach to molecular descriptor calculations.Match Communications In Mathematical And In Computer Chemistry 2006.56(2),: 237-248
23. Sakiyama Y et al . Predicting human liver microsomal stability with machine learning techniques. Journal of molecular graphics modeling ,2008. 26( 6),: 907-915
24. Shi T, Seligson D, Belldegrun AS , Palotie A, Horvath S. Tumor classification by tissue microarray profiling: random forest clustering applied to renal cell carcinoma. Modern Pathology (2005) 18, 547–557
25. Burbidge R , Trotter M, Buxton B, Holden S. Drug design by machine learning: support vector machines for pharmaceutical data analysis. Computers and Chemistry 26 (2001) 5–14

26. Butkiewicz M, Mueller R, Selic D, Dawson E, Meiler J .Application of Machine Learning Approaches on Quantitative Structure Activity Relationships . CIBCB 2009: 255-262
27. Neugebauer A, Hartmann RW, Klein CD.Prediction of protein-protein interaction inhibitors by chemoinformatics and machine learning methods. Journal of Medicinal Chemistry 2007.50(19): 4665-4668
28. <http://hudsonlegalblog.com/e-discovery/predictive-analytics-artificial-intelligence-science-fiction-e-discovery-truth.html>
29. Costa C, Menesatti P and Spinelli R. Performance Modeling in Forest Operations Through Partial Least Square Regression. Silva Fennica 46(2) research articles
30. <http://www.statsoft.com/textbook/partial-least-squares/>
31. Carver, R. P. The case against statistical significance testing. Harvard Educational Review,1978, 48, 378-399.
32. *Whitaker* JS Use of Stepwise Methodology in Discriminant Analysis Texas A&M University, January 1997
33. Lewicki P,Hill T. Statistics:Methods and applications. StatSoft, Inc. (2007):403-9
34. Narashiman ,Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression PNAS 2002 99:6567-6572
35. Boulesteix AL and Strimmer K. Partial least squares: a versatile tool for the analysis of high-dimensional genomic data .Bioinform (2007) 8 (1): 32-44.

# *APPENDIX*



## 9. Appendix

### I. Two scripts are used to calculate descriptors for the data named as :

- **condor\_trainee.pl**

```
#!/usr/bin/perl -w
#
# We assume that all files are in the format Conformers_*.sdf as
# downloaded from Pubchem. You can change this so that they pick all sdf files
#
@allmegafiles=`ls -l Conformers_*.sdf`;
#
#process these files one at a time
#
while (@allmegafiles){
$megafile=shift(@allmegafiles);
#split the larger files to contain only 150 molecules at a time....
# This step uses mayachemtools, and a perlscript that splits SDF files.
#SplitSDFfiles.pl should be on your path!
chomp($megafile);
$megafile=~s/\.sdf//;
system "/nfs/condor/pubchem/mayachemtools/bin/SplitSDFfiles.pl -m Cmpds --numcmpds 150 -r $megafile
-o $megafile.sdf";
# read these files for descriptor calculations
#
@allfiles=`ls -l *Part*.sdf`;
print @allfiles;
## The following loop is a script that was written for
## processing in lots of 8, now replaced with the condor submit script....
#
while(@allfiles){
for ($i=0; $i <= 7; $i++){
    $temp=shift(@allfiles);
    # check if no value
    unless($temp eq ""){push(@files,$temp);}
    }
foreach $file (@files){
chomp($file);
$file=~s/\.sdf//;
print $file, "\n";
# copy file template
system "cp template.drs $file.drs";
system "perl -i -p -e s/filename/$file/ $file.drs";
open(CMD,">$file.cmd");
print CMD 'universe = vanilla',"\n";
#print CMD 'requirements = (Arch=="X86_64" || Arch=="INTEL") && OpSys=="LINUX"',"\n";
##print CMD 'requirements = Arch=="X86_64" && OpSys=="LINUX"',"\n";
##print CMD 'requirements = Arch=="INTEL" && OpSys=="LINUX"',"\n";
#print CMD 'requirements = machine=="dolphin.osdd.jnu.ac.in"',"\n";
print CMD 'requirements = machine=="lynn.osdd.jnu.ac.in" || machine=="dolphin.osdd.jnu.ac.in"',"\n";
print CMD "\nexectable = /usr/share/dragon6/dragon6shell\n";
print CMD "output = $file.stdout\n";
```

```

print CMD "error      = $file.err\n";
print CMD "log        = $file.log\n";
print CMD "arguments  = -s $file.drs\n";
print CMD "queue\n";
close(CMD);
system("condor_submit $file.cmd");
#system "nohup /usr/share/dragon6/dragon6shell -s $file.drs > $file.err &";
}
#print "sleeping...\n\n";
#sleep 80;
@files=();
}
#clean files
#system "rm *Part*sdf";
} #while

```

- **template.drs**

```

<?xml version="1.0"?>
<DRAGON version="6.0.2" script_version="1" generation_date="2010/11/25">
  <OPTIONS>
    <SaveLayout value="true"/>
    <ShowWorksheet value="false"/>
    <Decimal_Separator value="."/>
    <Missing_String value="NaN"/>
    <DefaultMolFormat value="1"/>
    <HelpBrowser value="/usr/bin/xdg-open"/>
    <RejectUnusualValence value="false"/>
    <Add2DHydrogens value="false"/>
    <LogPathWalk value="true"/>
    <LogEdge value="true"/>
    <Weights>
      <weight name="Mass"/>
      <weight name="VdWVolume"/>
      <weight name="Electronegativity"/>
      <weight name="Polarizability"/>
      <weight name="Ionization"/>
      <weight name="I-State"/>
    </Weights>
    <SaveOnlyData value="false"/>
    <SaveLabelsOnSeparateFile value="false"/>
    <SaveFormatBlock value="%b - %n.txt"/>
    <SaveFormatSubBlock value="%b-%s - %n - %m.txt"/>
    <SaveExcludeMisVal value="false"/>
    <SaveExcludeAllMisVal value="false"/>
    <SaveExcludeConst value="false"/>
    <SaveExcludeNearConst value="false"/>
    <SaveExcludeStdDev value="false"/>
    <SaveStdDevThreshold value="0.0001"/>
    <SaveExcludeCorrelated value="false"/>
    <SaveCorrThreshold value="0.95"/>
    <SaveExclusionOptionsToVariables value="false"/>
    <SaveExcludeMisMolecules value="false"/>
    <SaveExcludeRejectedMolecules value="false"/>
  </OPTIONS>
  <DESCRIPTORS>
    <block id="1" SelectAll="true"/>
    <block id="2" SelectAll="true"/>
    <block id="3" SelectAll="true"/>
    <block id="4" SelectAll="true"/>

```

```

<block id="5" SelectAll="true"/>
<block id="6" SelectAll="true"/>
<block id="7" SelectAll="true"/>
<block id="8" SelectAll="true"/>
<block id="9" SelectAll="true"/>
<block id="10" SelectAll="true"/>
<block id="11" SelectAll="true"/>
<block id="12" SelectAll="true"/>
<block id="13" SelectAll="true"/>
<block id="14" SelectAll="true"/>
<block id="15" SelectAll="true"/>
<block id="16" SelectAll="true"/>
<block id="17" SelectAll="true"/>
<block id="18" SelectAll="true"/>
<block id="19" SelectAll="true"/>
<block id="20" SelectAll="true"/>
<block id="21" SelectAll="true"/>
<block id="22" SelectAll="true"/>
<block id="23" SelectAll="true"/>
<block id="24" SelectAll="true"/>
<block id="25" SelectAll="true"/>
<block id="26" SelectAll="true"/>
<block id="27" SelectAll="true"/>
<block id="28" SelectAll="true"/>
<block id="29" SelectAll="true"/>
</DESCRIPTORS>
<MOLFILES>
  <molInput value="file"/>
  <molFile value="filename.sdf"/>
</MOLFILES>
<OUTPUT>
  <SaveStdOut value="true"/>
  <SaveProject value="false"/>
  <SaveFile value="true"/>
  <SaveType value="singlefile"/>
  <SaveFilePath value="filename.out"/>
  <logMode value="none"/>
</OUTPUT>
</DRAGON>

```

## II. Perl script used for merging data and generating output for training

```

#!/usr/bin/perl
open(ACIN,"<$ARGV[0]>");
open(INACIN,"<$ARGV[1]>");
open(OUT,">>output_for_training.out");
@data=<ACIN>;
$header=shift @data;
@line=split(/\t/, $header);
shift @line;
push(@line,'CLASS');
chomp @line;
$line2 = join(" ", @line);
print OUT "$line2\n";
foreach(@data)
{
  @line=split(/\t/, $_);
  shift @line;
  push(@line,'active');
  chomp @line;
  $line2 = join(" ", @line);
  print OUT "$line2\n";
}

```

```

}
open (FHOUT,">>not_in_train.out");
while($line=<INACIN>){
  @line=split(/\t/, $line);
  shift @line;
  push(@line,'inactive');
  chomp @line;
  $line2 = join(" ", @line);
  $rand=rand();
#change the value below to (No of active)/(No of inactive)
  if($rand <= 0.2){
    print OUT "$line2\n";
  }else{
    print FHOUT "$line2\n";
  }
}
}

```

### III. Script for Data Preprocessing

```

library(caret)
osdd <- read.delim("final11.out", na.strings = "NA")
## save the compound name in a separate vector and then
## remove it from the data set
compound <- osdd$NAME
osdd <- osdd[, -1]
## Do the same thing for the class
classes <- osdd$class
osdd$class <- NULL
isNZV <- nearZeroVar(osdd)
osdd2 <- osdd[, -isNZV]
## Determining the percent of missing descriptor values for
## each column
pctMissing <- unlist(lapply(osdd2, function(x) mean(is.na(x))))
## Get a vector of column names that have at least one missing value
missingCols <- names(pctMissing)[pctMissing > 0]
## Remove columns with missing values
osdd3 <- osdd2[, !(names(osdd2) %in% missingCols)]
descrCor <- cor(osdd3)
highCorr <- findCorrelation(descrCor, cutoff = .85, verbose = FALSE)
osdd4 <- osdd3[, -highCorr]
set.seed(1)
inTrain <- createDataPartition(classes, times = 1, p = .75)[[1]]
trainingDescr <- osdd4[inTrain,]
testDescr <- osdd4[-inTrain,]
trainClasses <- classes[inTrain]
testClasses <- classes[-inTrain]
set.seed(2)
plsFit <- train(trainingDescr, trainClasses,
  method = "pls",
  tuneLength = 12,
  metric = "ROC",
  trControl = trainControl(
    method = "cv",
    classProbs = TRUE,
    summaryFunction = twoClassSummary))
plot(plsFit, metric = "ROC")
## Predictor unknowns that do not have corresponding class values
## preds <- extractPrediction(list(pls = plsFit), unkX = osdd4[1:5,])
## predict(ldaFit, newdata = testDescr[1:5,])
## vpredict(ldaFit, newdata = testDescr[1:5,], type = "prob")
plsPred <- predict(plsFit, testDescr)
plsProbs <- predict(plsFit, testDescr, type = "prob")

```

```

testPreds <- plsProbs
testPreds$obs <- testClasses
testPreds$pred <- plsPred
twoClassSummary(testPreds,
                 lev = levels(testPreds$obs))
## for modification:
## sensitivity.table
## Do the resampling manually to make a bunch of ROC curves
## The resampling indices are in plsFit$control$index
## Setup a blank plot
plot(0, 0, type = "n",
     xlim = c(0, 1),
     ylim = c(0, 1),
     ylab = "True Positive Rate",
     xlab = "False Positive Rate")
title("ROC Curves for Held-Out Data Sets")
abline(0, 1, lty = 2, col = "grey")

## work across each resample
for(i in 1:length(plsFit$control$index))
{
  inTrain <- plsFit$control$index[[i]]
  trainData <- trainingDescr[inTrain,]
  trainClass <- trainClasses[inTrain]
  holdoutDescr <- trainingDescr[-inTrain,]
  ## Fit the actual model using the optimal tuning parameters
  ## found by train()
  plsFoldFit <- plsda(trainData, trainClass,
                    ncomp = plsFit$bestTune$.ncomp)
  holdoutPreds <- data.frame(obs = trainClasses[-inTrain],
                           prob = predict(plsFoldFit, holdoutDescr, type = "prob")[,1,1])
  holdoutROC <- roc(holdoutPreds$prob, holdoutPreds$obs)
  points(1 - holdoutROC[, "specificity"],
        holdoutROC[, "sensitivity"],
        type = "l",
        col = i)
}

```

#### IV. The Sweave script “Classification.Rnw” and “Classification-method.Rnw” kindly provided by Dr. Max Kuhn, Director of Non-clinical Statistics, Pfizer.

- **Classification .Rnw**

```

%% Classification Modeling Script
%% Max Kuhn (max.kuhn@pfizer.com, mxkuhn@gmail.com)
%% Version: 1.00
%% Created on: 2010/10/02
%%
%% This is an Sweave template for building and describing
%% classification models. It mixes R and LaTeX code. The document can
%% be processed using R's Sweave function to produce a tex file.
%%
%% The inputs are:
%% - the initial data set in a data frame called 'rawData'
%% - a factor column in the data set called 'class'. this should be the
%%   outcome variable
%% - all other columns in rawData should be predictor variables
%% - the type of model should be in a variable called 'modelName'.
%%
%% The script attempts to make some intelligent choices based on the
%% model being used. For example, if modelName is "pls", the script will

```



```

\pagestyle{fancy}
\lhead{}
%% OPTION Report header name
\chead{Classification Model Script}
\rhead{}
\lfoot{}
\cfoot{}
\rfoot{\thepage\ of \pageref{LastPage}}
\renewcommand{\headrulewidth}{1pt}
\renewcommand{\footrulewidth}{1pt}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% OPTION Report title and modeler name
\title{OSDD4 - first run: Preprocessing and SVMRadial}
\author{Anisha Kathpalia}
\begin{document}
\maketitle
\thispagestyle{empty}
<<startup, results = hide, echo = FALSE>>=
library(Hmisc)
library(caret)
versionTest <- compareVersion(packageDescription("caret")$Version,
                              "4.65")
if(versionTest < 0) stop("caret version 4.65 or later is required")
library(RColorBrewer)
listString <- function(x, period = FALSE, verbose = FALSE)
{
  if(verbose) cat("\n  entering listString\n")
  flush.console()
  if(!is.character(x))
    x <- as.character(x)
  numElements <- length(x)
  out <- if (length(x) > 0) {
    switch(min(numElements, 3), x, paste(x, collapse = " and "),
          {
            x <- paste(x, c(rep(" ", numElements - 2), " and", ""), sep = "")
            paste(x, collapse = " ")
          })
  }
  else ""
  if(period) out <- paste(out, ".", sep = "")
  if(verbose) cat("  leaving listString\n\n")
  flush.console()
  out
}
resampleStats <- function(x, digits = 3)
{
  bestPerf <- x$bestTune
  colnames(bestPerf) <- gsub("^\\.", "", colnames(bestPerf))
  out <- merge(x$results, bestPerf)
  out <- out[, colnames(out) %in% x$perfNames]
  names(out) <- gsub("ROC", "area under the ROC curve", names(out), fixed = TRUE)
  names(out) <- gsub("Sens", "sensitivity", names(out), fixed = TRUE)
  names(out) <- gsub("Spec", "specificity", names(out), fixed = TRUE)
  names(out) <- gsub("Accuracy", "overall accuracy", names(out), fixed = TRUE)
  names(out) <- gsub("Kappa", "Kappa statistics", names(out), fixed = TRUE)
  out <- format(out, digits = digits)
  listString(paste(names(out), "was", out))
}
twoClassNoProbs <- function (data, lev = NULL, model = NULL)
{

```





```

        "missing data. ",
        ifelse(sum(colExclude) > 1,
              " These were excluded. ",
              " This was excluded. "))
    predictorNames <- predictorNames[!colExclude]
    rawData <- rawData[, names(rawData) %in% c("outcome", predictorNames), drop = FALSE]
  }
rowRate <- apply(rawData[, predictorNames, drop = FALSE],
                1, function(x) mean(is.na(x)))
rowExclude <- rowRate > .20
if(any(rowExclude))
{
  missingText <- paste(missingText,
                      ifelse(sum(rowExclude) > 1,
                            " There were ",
                            " There was "),
                      sum(colExclude),
                      ifelse(sum(rowExclude) > 1,
                            " samples ",
                            " sample "),
                      "with an excessive number of ",
                      "missing data. ",
                      ifelse(sum(rowExclude) > 1,
                            " These were excluded. ",
                            " This was excluded. "),
                      "After filtering, ",
                      sum(!rowExclude),
                      " samples remained.")
  rawData <- rawData[!rowExclude, ]
  hasMissing <- apply(rawData[, predictorNames, drop = FALSE],
                    1, function(x) mean(is.na(x)))
} else {
  hasMissing <- apply(rawData[, predictorNames, drop = FALSE],
                    1, function(x) any(is.na(x)))
  missingText <- paste(missingText,
                      ifelse(missingText == "",
                            "There ",
                            "Subsequently, there "),
                      ifelse(sum(hasMissing) == 1,
                            "was ",
                            "were "),
                      ifelse(sum(hasMissing) > 0,
                            sum(hasMissing),
                            "no"),
                      ifelse(sum(hasMissing) == 1,
                            "sample ",
                            "samples "),
                      "with missing values.")
}
@
The initial data set consisted of \Sexpr{numSamples} samples and
\Sexpr{numPredictors} predictor variables. The breakdown of the
outcome data classes were: \Sexpr{classDistString}. \Sexpr{missingText}
<<pca, echo = FALSE, results = hide>>=
predictors <- rawData[, predictorNames, drop = FALSE]
## PCA will fail with predictors having less than 2 unique values
isZeroVar <- apply(predictors, 2,
                  function(x) length(unique(x)) < 2)
if(any(isZeroVar)) predictors <- predictors[, !isZeroVar, drop = FALSE]
## For whatever, only the formula interface to precomp
## handles missing values
pcaForm <- as.formula(

```

```

        paste("~",
              paste(names(predictors), collapse = "+"))
pca <- prcomp(pcaForm,
             data = predictors,
             center = TRUE,
             scale = TRUE,
             na.action = na.omit)
## OPTION: the number of components plotted/discussed can be set
numPCAcomp <- 3
pctVar <- pca$sdev^2/sum(pca$sdev^2)*100
pcaText <- paste(round(pctVar[1:numPCAcomp], 1),
                "\\%",
                sep = "")
pcaText <- listString(pcaText)
@
To get an initial assessment of the separability of the classes,
principal component analysis (PCA) was used to distill the
\Sexpr{numPredictors} predictors down into \Sexpr{numPCAcomp}
surrogate variables (i.e. the principal components) in a manner that
attempts to maximize the amount of information preserved from the
original predictor set. Figure \ref{F:initialPCA} contains plots of
the first \Sexpr{numPCAcomp} components, which accounted for
\Sexpr{pcaText} percent of the variability in the original predictors
(respectively).
%% OPTION: remark on how well (or poorly) the data separated
\setkeys{Gin} {width = 0.8\textwidth}
\begin{figure}[p]
  \begin{center}
<<pcaPlot, echo = FALSE, results = hide, fig = TRUE, width = 8, height = 8>>=
trellis.par.set(caretTheme(), warn = TRUE)
if(numPCAcomp == 2)
{
  axisRange <- extendrange(pca$x[, 1:2])
  print(
    xyplot(PC1 ~ PC2,
           data = as.data.frame(pca$x),
           type = c("p", "g"),
           groups = rawData$outcome,
           auto.key = list(columns = 2),
           xlim = axisRange,
           ylim = axisRange))
} else {
axisRange <- extendrange(pca$x[, 1:numPCAcomp])
print(
  splom(~as.data.frame(pca$x)[, 1:numPCAcomp],
        type = c("p", "g"),
        groups = rawData$outcome,
        auto.key = list(columns = 2),
        as.table = TRUE,
        prepanel.limits = function(x) axisRange
        ))
}
@
\caption[PCA Plot] {A plot of the first \Sexpr{numPCAcomp}
principal components for the original data set.}
\label{F:initialPCA}
\end{center}
\end{figure}
<<initialDataSplit, results = hide, echo = FALSE>>=
## OPTION: in small samples sizes, you may not want to set aside a
## training set and focus on the resampling results.
pctTrain <- .8

```

```

# pctTrain <- 1
if(pctTrain < 1)
{
  ## OPTION: seed number can be changed
  set.seed(1)
  inTrain <- createDataPartition(rawData$outcome,
                                p = pctTrain,
                                list = FALSE)
  trainX <- rawData[ inTrain, predictorNames]
  testX <- rawData[-inTrain, predictorNames]
  trainY <- rawData[ inTrain, "outcome"]
  testY <- rawData[-inTrain, "outcome"]
  splitText <- paste("The original data were split into ",
                    "a training set ($n$=",
                    nrow(trainX),
                    ") and a test set ($n$=",
                    nrow(testX),
                    ") in a manner that preserved the ",
                    "distribution of the classes.",
                    sep = "")
  isZeroVar <- apply(trainX, 2,
                    function(x) length(unique(x)) < 2)
  if(any(isZeroVar))
  {
    trainX <- trainX[, !isZeroVar, drop = FALSE]
    testX <- testX[, !isZeroVar, drop = FALSE]
  }
} else {
  trainX <- rawData[, predictorNames]
  testX <- NULL
  trainY <- rawData[, "outcome"]
  testY <- NULL
  splitText <- "The entire data set was used as the training set."
} trainDist <- table(trainY)
nir <- max(trainDist)/length(trainY)*100
niClass <- names(trainDist)[which.max(trainDist)]
nirText <- paste("The non--information rate is the accuracy that can be ",
                "achieved by predicting all samples using the most ",
                "dominant class. For these data, the rate is ",
                round(nir, 2), "% using the `",
                niClass,
                "` class.",
                sep = "")

@
\Sexpr{splitText}
\Sexpr{nirText}
<<nzv, results = hide, echo = FALSE>>=
## OPTION: other pre-processing steps can be used
ppSteps <- caret::suggestions(modName)
set.seed(2)
if(ppSteps["nzv"])
{
  nzv <- nearZeroVar(trainX)
  if(length(nzv) > 0)
  {
    nzvVars <- names(trainX)[nzv]
    trainX <- trainX[, -nzv]
    nzvText <- paste("There were ",
                    length(nzv),
                    " predictors that were removed due to",
                    " severely unbalanced distributions that",
                    " could negatively affect the model fit",

```

```

        ifelse(length(nzv) > 10,
              ".",
              paste(":",
                    listString(nzvVars),
                    ".",
                    sep = "")),
              sep = "")
      if(pctTrain < 1) testX <- testX[, -nzv]
    } else nzvText <- ""
  } else nzvText <- ""
@
<<corrFilter, results = hide, echo = FALSE>>=
if(ppSteps["corr"])
{
  ## OPTION:
  corrThresh <- .75
  highCorr <- findCorrelation(cor(trainX, use = "pairwise.complete.obs"),
                             corrThresh)
  if(length(highCorr) > 0)
  {
    corrVars <- names(trainX)[highCorr]
    trainX <- trainX[, -highCorr]
    corrText <- paste("There were ",
                     length(highCorr),
                     " predictors that were removed due to",
                     " large between--predictor correlations that",
                     " could negatively affect the model fit",
                     ifelse(length(highCorr) > 10,
                             ".",
                             paste(":",
                                     listString(highCorr),
                                     ".",
                                     sep = "")),
                     " Removing these predictors forced",
                     " all pair--wise correlations to be",
                     " less than ",
                     corrThresh,
                     ".",
                     sep = "")
    if(pctTrain < 1) testX <- testX[, -highCorr]
  } else corrText <- ""
} else corrText <- ""
@
<<preProc, echo = FALSE, results = hide>>=
ppMethods <- NULL
if(ppSteps["center"]) ppMethods <- c(ppMethods, "center")
if(ppSteps["scale"]) ppMethods <- c(ppMethods, "scale")
if(any(hasMissing) > 0) ppMethods <- c(ppMethods, "knnImpute")
##OPTION other methods, such as spatial sign, can be added to this list
if(length(ppMethods) > 0)
{
  ppInfo <- preprocess(trainX, method = ppMethods)
  trainX <- predict(ppInfo, trainX)
  if(pctTrain < 1) testX <- predict(ppInfo, testX)
  ppText <- paste("The following pre--processing methods were",
                 " applied to the training",
                 ifelse(pctTrain < 1, " and test", ""),
                 " data: ",
                 listString(ppMethods),
                 ".",
                 sep = "")
  ppText <- gsub("center", "mean centering", ppText)
}

```

```

ppText <- gsub("scale", "scaling to unit variance", ppText)
ppText <- gsub("knnImpute",
  paste(ppInfo$k, "--nearest neighbor imputation", sep = ""),
  ppText)
ppText <- gsub("spatialSign", "the spatial sign transformation", ppText)
ppText <- gsub("pca", "principal component feature extraction", ppText)
ppText <- gsub("ica", "independent component feature extraction", ppText)
} else {
  ppInfo <- NULL
  ppText <- ""
}

predictorNames <- names(trainX)
if(nzvText != "" | corrText != "" | ppText != "")
{
  varText <- paste("After pre--processing, ",
    ncol(trainX),
    "predictors remained for modeling.")
} else varText <- ""
save(trainX,trainY,file="trainClass.RData")
save(testX,testY,file="testClass.RData")
@
\{Sexpr{nzvText} \{Sexpr{corrText} \{Sexpr{ppText} \{Sexpr{varText}
The pre-processed data : trainX and trainY datasets - are saved as trainClass.RData in the working directory, and can be
reused without preprocessing for model building. The test data is saved as testClass.RData, to be used as a hold out
data set for model validation.
\clearpage
\section*{Model Building}
<<setupWorkers, echo = FALSE, results = hide>>=
numWorkers <- 1
##OPTION: turn up numWorkers to use MPI
if(numWorkers > 1)
{
  mpiCalcs <- function(X, FUN, ...)
  {
    theDots <- list(...)
    parLapply(theDots$cl, X, FUN)
  }
  library(snow)
  cl <- makeCluster(numWorkers, "MPI")
}
@
<<setupResampling, echo = FALSE, results = hide>>=
##OPTION: the resampling options can be changed. See
## ?trainControl for details
resampName <- "boot632"
resampNumber <- 10
numRepeat <- 1
resampP <- .75
modelInfo <- modelLookup(modName)
if(numClasses == 2)
{
  foo <- if(any(modelInfo$probModel)) twoClassSummary else twoClassNoProbs
} else foo <- defaultSummary
set.seed(3)
ctlObj <- trainControl(method = resampName,
  number = resampNumber,
  repeats = numRepeat,
  p = resampP,
  classProbs = any(modelInfo$probModel),
  summaryFunction = foo)
##OPTION select other performance metrics as needed

```

```

optMetric <- if(numClasses == 2 & any(modelInfo$probModel)) "ROC" else "Kappa"
if(numWorkers > 1)
{
  ctlObj$workers <- numWorkers
  ctlObj$computeFunction <- mpiCalcs
  ctlObj$computeArgs <- list(cl = cl)
}
@
<<setupGrid, results = hide, echo = FALSE>>=
##OPTION expand or contract these grids as needed (or
##  add more models
gridSize <- 3
if(modName %in% c("svmPoly", "svmRadial", "svmLinear", "lvq", "ctree2", "ctree")) gridSize <- 5
if(modName %in% c("earth", "fda")) gridSize <- 7
if(modName %in% c("knn", "rocc", "glmboost", "rf", "nodeHarvest")) gridSize <- 10
if(modName %in% c("nb")) gridSize <- 2
if(modName %in% c("pam", "rpart")) gridSize <- 15
if(modName %in% c("pls")) gridSize <- min(20, ncol(trainX))

if(modName == "gbm")
{
  tGrid <- expand.grid(.interaction.depth = -1 + (1:5)*2,
                      .n.trees = (1:10)*20,
                      .shrinkage = .1)
}
if(modName == "nnet")
{
  tGrid <- expand.grid(.size = -1 + (1:5)*2,
                      .decay = c(0, .001, .01, .1))
}
@
<<fitModel, results = hide, echo = FALSE, eval = TRUE>>=
##OPTION alter as needed
set.seed(4)
modelFit <- switch(modName,
  gbm =
  {
    mix <- sample(seq(along = trainY))
    train(
      trainX[mix,], trainY[mix], modName,
      verbose = FALSE,
      bag.fraction = .9,
      metric = optMetric,
      trControl = ctlObj,
      tuneGrid = tGrid)
  },
  multinom =
  {
    train(
      trainX, trainY, modName,
      trace = FALSE,
      metric = optMetric,
      maxiter = 1000,
      MaxNWts = 5000,
      trControl = ctlObj,
      tuneLength = gridSize)
  },
  nnet =
  {
    train(
      trainX, trainY, modName,
      metric = optMetric,

```

```

        linout = FALSE,
        trace = FALSE,
        maxiter = 1000,
        MaxNWts = 5000,
        trControl = ctrlObj,
        tuneGrid = tGrid)
    },
  svmRadial =, svmPoly =, svmLinear =
  {
    train(
      trainX, trainY, modName,
      metric = optMetric,
      scaled = TRUE,
      trControl = ctrlObj,
      tuneLength = gridSize)
  },
  {
    train(trainX, trainY, modName,
          trControl = ctrlObj,
          metric = optMetric,
          tuneLength = gridSize)
  })
})

@
<<modelDescr, echo = FALSE, results = hide>>=
summaryText <- ""
resampleName <- switch(tolower(modelFit$control$method),
  boot = paste("the bootstrap (", length(modelFit$control$index), " reps)", sep = ""),
  boot632 = paste("the bootstrap 632 rule (", length(modelFit$control$index), " reps)", sep = ""),
  cv = paste("cross-validation (", modelFit$control$number, " fold)", sep = ""),
  repeatedcv = paste("cross-validation (", modelFit$control$number, " fold, repeated ",
    modelFit$control$repeats, " times)", sep = ""),
  lgocv = paste("repeated train/test splits (", length(modelFit$control$index), " reps, ",
    round(modelFit$control$p, 2), "$\\%$)", sep = ""))
tuneVars <- latexTranslate(tolower(modelInfo$label))
tuneVars <- gsub("\\#", "the number of ", tuneVars, fixed = TRUE)
if(ncol(modelFit$bestTune) == 1 && colnames(modelFit$bestTune) == ".parameter")
{
  summaryText <- paste(summaryText,
    "\n\n",
    "There are no tuning parameters associated with this model.",
    "To characterize the model performance on the training set,",
    resampleName,
    "was used.",
    "Table \\ref{T:resamps} and Figure \\ref{F:profile}",
    "show summaries of the resampling results. ")
} else {
  summaryText <- paste("There",
    ifelse(nrow(modelInfo) > 1, "are", "is"),
    nrow(modelInfo),
    ifelse(nrow(modelInfo) > 1, "tuning parameters", "tuning parameter"),
    "associated with this model:",
    listString(tuneVars, period = TRUE))
  paramNames <- gsub(".", "", names(modelFit$bestTune), fixed = TRUE)
  for(i in seq(along = paramNames))
  {
    check <- modelInfo$parameter %in% paramNames[i]
    if(any(check))
    {
      paramNames[i] <- modelInfo$label[which(check)]
    }
  }
}
paramNames <- gsub("#", "the number of ", paramNames, fixed = TRUE)

```

```

## Check to see if there was only one combination fit
summaryText <- paste(summaryText,
  "To choose",
  ifelse(nrow(modelInfo) > 1,
    "appropriate values of the tuning parameters,",
    "an appropriate value of the tuning parameter,"),
  resampleName,
  "was used to generated a profile of performance across the",
  nrow(modelFit$results),
  ifelse(nrow(modelInfo) > 1,
    "combinations of the tuning parameters.",
    "candidate values."),
  "Table \\\ref{T:resamps} and Figure \\\ref{F:profile} show",
  "summaries of the resampling profile. ",
  "The
final model fitted to the entire training set was:",
  listString(paste(latexTranslate(tolower(paramNames)), "=", modelFit$bestTune[1,]), period = TRUE))
}
@
\Sexpr{summaryText}
<<resampTable, echo = TRUE, results = tex>>
tableData <- modelFit$results
if(all(modelInfo$parameter == "parameter"))
{
  tableData <- tableData[,-1, drop = FALSE]
  colNums <- c(length(modelFit$perfNames), length(modelFit$perfNames))
  colLabels <- c("Mean", "Standard Deviation")
  constString <- ""
} else {
  isConst <- apply(tableData[, modelInfo$parameter, drop = FALSE],
    2,
    function(x) length(unique(x)) == 1)

  numParamInTable <- sum(!isConst)
  if(any(isConst))
  {
    constParam <- modelInfo$parameter[isConst]
    constValues <- format(tableData[, constParam, drop = FALSE], digits = 4)[1,,drop = FALSE]
    tableData <- tableData[, !(names(tableData) %in% constParam), drop = FALSE]
    constString <- paste("The tuning",
      ifelse(sum(isConst) > 1,
        "parameters",
        "parameter"),
      listString(paste("", names(constValues), "", sep = "")),
      ifelse(sum(isConst) > 1,
        "were",
        "was"),
      "held constant at",
      ifelse(sum(isConst) > 1,
        "a value of",
        "values of"),
      listString(constValues[1,]))
  } else constString <- ""
  cn <- colnames(tableData)
  for(i in seq(along = cn))
  {
    check <- modelInfo$parameter %in% cn[i]
    if(any(check))
    {
      cn[i] <- modelInfo$label[which(check)]
    }
  }
  colnames(tableData) <- cn

```



```

colNums <- c(numParamInTable,
            length(modelFit$perfNames),
            length(modelFit$perfNames),
            length(modelFit$perfNames))
colLabels <- c("", "Mean", "Standard Deviation", "Apparant")
}
#
colnames(tableData) <- gsub("SD$", "", colnames(tableData))
colnames(tableData) <- latexTranslate(colnames(tableData))
rownames(tableData) <- latexTranslate(rownames(tableData))
latex(tableData,
      rowname = NULL,
      file = "",
      cgroup = colLabels,
      n.cgroup = colNums,
      where = "h!",
      digits = 4,
      longtable = nrow(tableData) > 30,
      caption = paste(resampleName, "results from the model fit.", constString),
      label = "T:resamps")
@
\setkeys{Gin}{ width = 0.9\textwidth}
\begin{figure}[b]
\begin{center}
<<profilePlot, echo = FALSE, fig = TRUE, width = 8, height = 6>>=
trellis.par.set(caretTheme(), warn = TRUE)
if(all(modelInfo$parameter == "parameter") | all(isConst) | modName == "nb")
{
  resultsPlot <- resampleHist(modelFit)
  plotCaption <- paste("Distributions of model performance from the ",
                    "training set estimated using ",
                    resampleName)
} else {
  if(modName %in% c("svmPoly", "svmRadial", "svmLinear"))
  {
    resultsPlot <- plot(modelFit,
                      metric = optMetric,
                      xTrans = function(x) log10(x))
    resultsPlot <- update(resultsPlot,
                      type = c("g", "p", "l"),
                      ylab = paste(optMetric, " (", resampleName, ")", sep = ""))
  } else {
    resultsPlot <- plot(modelFit,
                      metric = optMetric)
    resultsPlot <- update(resultsPlot,
                      type = c("g", "p", "l"),
                      ylab = paste(optMetric, " (", resampleName, ")", sep = ""))
  }
  plotCaption <- paste("A plot of the estimates of the",
                    optMetric,
                    "values calculated using",
                    resampleName)
}
print(resultsPlot)
@
\caption[Performance Plot]{\Sexpr{plotCaption}.}
\label{F:profile}
\end{center}
\end{figure}
<<stopWorkers, echo = FALSE, results = hide>>=
if(numWorkers > 1) stopCluster(cl)
@

```

```

<<testPred, results = tex, echo = FALSE>>=
if(pctTrain < 1)
{R sum
cat("\clearpage\n\\section*{Test Set Results}\n\n")
classPred <- predict(modelFit, testX)
cm <- confusionMatrix(classPred, testY)
values <- cm$overall[c("Accuracy", "Kappa", "AccuracyPValue", "McnemarPValue")]
values <- values[!is.na(values) & !is.nan(values)]
values <- c(format(values[1:2], digits = 3),
            format.pval(values[-(1:2)], digits = 5))
nms <- c("the overall accuracy", "the Kappa statistic",
        "the  $\$p$ --value that accuracy is greater than the no--information rate",
        "the  $\$p$ --value of concordance from McNemar's test")
nms <- nms[seq(along = values)]
names(values) <- nms
if(any(modelInfo$probModel))
{
classProbs <- extractProb(list(fit = modelFit,
                             testX = testX,
                             testY = testY)
classProbs <- subset(classProbs, dataType == "Test")
if(numClasses == 2)
{
tmp <- twoClassSummary(classProbs, lev = levels(classProbs$obs))
tmp <- c(format(tmp, digits = 3))
names(tmp) <- c("the sensitivity", "the specificity",
              "the area under the ROC curve")
values <- c(values, tmp)
}
probPlot <- plotClassProbs(classProbs)
}
testString <- paste("Based on the test set of",
                  nrow(testX),
                  "samples,",
                  listString(paste(names(values), "was", values), period = TRUE),
                  "The confusion matrix for the test set is shown in Table",
                  "\\\ref{T:cm}.")
testString <- paste(testString,
                  " Using ", resampleName,
                  ", the training set estimates were ",
                  resampleStats(modelFit),
                  ". ",
                  sep = "")
if(any(modelInfo$probModel)) testString <- paste(testString,
                                                "Histograms of the class probabilities",
                                                "for the test set samples are shown in",
                                                "Figure \\\ref{F:probs}",
                                                ifelse(numClasses == 2,
                                                    " and the test set ROC curve is in Figure \\\ref{F:roc}.",
                                                    ".")
)
latex(cm$stable,
      title = "",
      file = "",
      where = "h",
      cgroup = "Observed Values",
      n.cgroup = numClasses,
      caption = "The confusion matrix for the test set",
      label = "T:cm")
} else testString <- ""
@
\Sexpr {testString}
<<classProbsTex, results = tex, echo = FALSE>>=

```

```

if(any(modelInfo$probModel))
{
  cat(
    paste("\begin{figure}[p]\n",
          "\begin{center}\n",
          "\includegraphics{classProbs}",
          "\caption[PCA Plot]{Class probabilities",
          "for the test set. Each panel contains ",
          "separate classes}\n",
          "\label{F:probs}\n",
          "\end{center}\n",
          "\end{figure}")
  )
}
if(any(modelInfo$probModel) & numClasses == 2)
{
  cat(
    paste("\begin{figure}[p]\n",
          "\begin{center}\n",
          "\includegraphics[clip, width = .8\textwidth]{roc}",
          "\caption[ROC Plot]{ROC Curve",
          "for the test set.}\n",
          "\label{F:roc}\n",
          "\end{center}\n",
          "\end{figure}")
  )
}
@
<<classProbsTex, results = hide, echo = FALSE>>=
if(any(modelInfo$probModel))
{
  pdf("classProbs.pdf", height = 7, width = 7)
  trellis.par.set(caretTheme(), warn = FALSE)
  print(probPlot)
  dev.off()
}

if(any(modelInfo$probModel) & numClasses == 2)
{
  rocPoints <- as.data.frame(
    roc(classProbs[, levels(trainY)[1]],
        testY,
        positive = levels(trainY)[1]))
  pdf("roc.pdf", height = 8, width = 8)
  plot(1 - rocPoints$specificity, rocPoints$sensitivity,
       ylab = "Sensitivity", xlab = "1 - Specificity",
       type = "n",
       main = paste("AUC:", round(aucRoc(rocPoints), 3)))
  abline(0, 1, lty = 2, col = "darkgrey")
  points(1 - rocPoints$specificity, rocPoints$sensitivity,
        type = "S")
  dev.off()
}
@
\section*{Versions}
<<versions, echo = FALSE, results = tex>>=
toLatex(sessionInfo())
@
\end{document}

```

- **Classification-method.Rnw**

```

%% Classification Modeling Script
%% Max Kuhn (max.kuhn@pfizer.com, mxkuhn@gmail.com)
%% Version: 1.00
%% Created on: 2010/10/02
%%
%% This is an Sweave template for building and describing
%% classification models. It mixes R and LaTeX code. The document can
%% be processing using R's Sweave function to produce a tex file.
%%
%% The inputs are:
%% - the initial data set in a data frame called 'rawData'
%% - a factor column in the data set called 'class'. this should be the
%% outcome variable
%% - all other columns in rawData should be predictor variables
%% - the type of model should be in a variable called 'modelName'.
%%
%% The script attempts to make some intelligent choices based on the
%% model being used. For example, if modelName is "pls", the script will
%% automatically center and scale the predictor data. There are
%% situations where these choices can (and should be) changed.
%%
%% There are other options that may make sense to change. For example,
%% the user may want to adjust the type of resampling. To find these
%% parts of the script, search on the string 'OPTION'. These parts of
%% the code will document the options.

\documentclass[12pt]{report}
\usepackage{amsmath}
\usepackage[pdftex]{graphicx}
\usepackage{color}
\usepackage{ctable}
\usepackage{xspace}
\usepackage{fancyvrb}
\usepackage{fancyhdr}
\usepackage{lastpage}
\usepackage{longtable}
\usepackage{algorithm2e}
\usepackage[
  colorlinks=true,
  linkcolor=blue,
  citecolor=blue,
  urlcolor=blue]
  {hyperref}
\usepackage{lscap}
\usepackage{Sweave}
\SweaveOpts{keep.source = TRUE}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% define new colors for use
\definecolor{darkgreen}{rgb}{0,0.6,0}
\definecolor{darkred}{rgb}{0.6,0,0}
\definecolor{lightbrown}{rgb}{1,0.9,0.8}
\definecolor{brown}{rgb}{0.6,0.3,0.3}
\definecolor{darkblue}{rgb}{0,0,0.8}
\definecolor{darkmagenta}{rgb}{0.5,0,0.5}

```

%%%%%%%%  
%%%%%%%%

```
\newcommand{\bld}[1]{\mbox{\boldmath $#1$}}
\newcommand{\shell}[1]{\mbox{ $#1$}}
\renewcommand{\vec}[1]{\mbox{\bf {#1}}}
\newcommand{\ReallySmallSpacing}{\renewcommand{\baselinestretch}{.6}\Large\normalsize}
\newcommand{\SmallSpacing}{\renewcommand{\baselinestretch}{1.1}\Large\normalsize}
\newcommand{\halfs}{\frac{1}{2}}
\setlength{\oddsidemargin}{-.25truein}
\setlength{\evensidemargin}{0truein}
\setlength{\topmargin}{-0.2truein}
\setlength{\textwidth}{7truein}
\setlength{\textheight}{8.5truein}
\setlength{\parindent}{0.20truein}
\setlength{\parskip}{0.10truein}
\setcounter{LTchunksize}{50}
```

%%%%%%%%  
%%%%%%%%

```
\pagestyle{fancy}
\lhead{}
%% OPTION Report header name
\chead{Classification Model Script}
\rhead{}
\lfoot{}
\cfoot{}
\rfoot{\thepage\ of \pageref{LastPage}}
\renewcommand{\headrulewidth}{1pt}
\renewcommand{\footrulewidth}{1pt}
```

%%%%%%%%  
%%%%%%%%

```
%% OPTION Report title and modeler name
\title{Classification Model Script - Model building only using PLS}
\author{Anisha Kathpalia}
\begin{document}
\maketitle
\thispagestyle{empty}
<<dummy, eval=TRUE, echo=FALSE, results=hide>>=
# sets values for variables used later in the program to prevent the \Sexpr error on parsing with Sweave
numSamples="
classDistString="
missingText="
numPredictors="
numPCAcomp="
pcaText="
nzvText="
corrText="
ppText="
varText="
splitText="Dummy Text"
nirText="Dummy Text"
# pctTrain is a variable that is initialised in Data splitting, and reused later in testPred
pctTrain=0.8
@
<<startup, eval= TRUE, results = hide, echo = FALSE>>=
library(Hmisc)
library(caret)
versionTest <- compareVersion(packageDescription("caret")$Version,
"4.65")
if(versionTest < 0) stop("caret version 4.65 or later is required")
```

```

library(RColorBrewer)
listString <- function(x, period = FALSE, verbose = FALSE)
{
  if(verbose) cat("\n  entering listString\n")
  flush.console()
  if(!is.character(x))
    x <- as.character(x)
  numElements <- length(x)
  out <- if(length(x) > 0) {
    switch(min(numElements, 3), x, paste(x, collapse = " and "),
          {
            x <- paste(x, c(rep(",", numElements - 2), " and", ""), sep = "")
            paste(x, collapse = "")
          })
  }
  else ""
  if(period) out <- paste(out, ".", sep = "")
  if(verbose) cat("  leaving listString\n\n")
  flush.console()
  out
}
resampleStats <- function(x, digits = 3)
{
  bestPerf <- x$bestTune
  colnames(bestPerf) <- gsub("^\\.\\.", "", colnames(bestPerf))
  out <- merge(x$results, bestPerf)
  out <- out[, colnames(out) %in% x$perfNames]
  names(out) <- gsub("ROC", "area under the ROC curve", names(out), fixed = TRUE)
  names(out) <- gsub("Sens", "sensitivity", names(out), fixed = TRUE)
  names(out) <- gsub("Spec", "specificity", names(out), fixed = TRUE)
  names(out) <- gsub("Accuracy", "overall accuracy", names(out), fixed = TRUE)
  names(out) <- gsub("Kappa", "Kappa statistics", names(out), fixed = TRUE)
  out <- format(out, digits = digits)
  listString(paste(names(out), "was", out))
}
twoClassNoProbs <- function(data, lev = NULL, model = NULL)
{
  out <- c(sensitivity(data[, "pred"], data[, "obs"], lev[1]),
          specificity(data[, "pred"], data[, "obs"], lev[2]),
          confusionMatrix(data[, "pred"], data[, "obs"])$overall["Kappa"])
  names(out) <- c("Sens", "Spec", "Kappa")
  out
}
##OPTION: model name: see ?train for more values/models
modName <- "pls"
#data(mdr)
#rawData <- mdr$Descr
#rawData$outcome <- mdr$class
##Commenting out the default lines above to use our data saved as rawData.Rdata
#load("rawData.Rdata")
#rawData[1:20, 2] <- NA
#rawData[1:120, 30:32] <- NA
## No longer dealing with raw data - this contains the preprocessed training dataset
load("/nfs/condor/malaria/final/first-Classification/trainClass.RData")
load("/nfs/condor/malaria/final/first-Classification/testClass.RData")
rawData <- trainX
rawData$outcome <- trainY
#rawData <- iris
#names(rawData)[5] <- "outcome"
@
%% This is a test line - to find a way to comment text outside the code chunk, which has inserted values from R
variables. We are using the method: \Sexpr{modName}.

```

```

\section*{Data Sets}\label{S:data}
%% OPTION: provide some background on the problem, the experimental
%% data, how the compounds were selected etc
<<getDataInfo, echo = FALSE, results = hide>>=
if(!any(names(rawData) == "outcome")) stop("a variable called outcome should be in the data set")
if(!is.factor(rawData$outcome)) stop("the outcome should be a factor vector")
## OPTION: when there are only two classes, the first level of the
## factor is used as the "positive" or "event" for calculating
## sensitivity and specificity. Adjust the outcome factor accordingly.
numClasses <- length(levels(rawData$outcome))
numSamples <- nrow(rawData)
numPredictors <- ncol(rawData) - 1
predictorNames <- names(rawData)[names(rawData) != "outcome"]
isNum <- apply(rawData[,predictorNames, drop = FALSE], 2, is.numeric)
if(any(!isNum)) stop("all predictors in rawData should be numeric")
classTextCheck <- all.equal(levels(rawData$outcome), make.names(levels(rawData$outcome)))
if(!classTextCheck) warning("the class levels are not valid R variable names; this may cause errors")
## Get the class distribution
classDist <- table(rawData$outcome)
classDistString <- paste("'",
                        names(classDist),
                        "' ($n$=",
                        classDist,
                        ")",
                        sep = "'")
classDistString <- listString(classDistString)
@
<<missingFilter, eval=FALSE, echo = FALSE, results = hide>>=
colRate <- apply(rawData[, predictorNames, drop = FALSE],
                2, function(x) mean(is.na(x)))
##OPTION thresholds can be changed
colExclude <- colRate > .20
missingText <- ""
if(any(colExclude))
{
  missingText <- paste(missingText,
                      ifelse(sum(colExclude) > 1,
                            " There were ",
                            " There was "),
                      sum(colExclude),
                      ifelse(sum(colExclude) > 1,
                            " predictors ",
                            " predictor "),
                      "with an excessive number of ",
                      "missing data. ",
                      ifelse(sum(colExclude) > 1,
                            " These were excluded. ",
                            " This was excluded. "))
  predictorNames <- predictorNames[!colExclude]
  rawData <- rawData[, names(rawData) %in% c("outcome", predictorNames), drop = FALSE]
}
rowRate <- apply(rawData[, predictorNames, drop = FALSE],
                1, function(x) mean(is.na(x)))
rowExclude <- rowRate > .20
if(any(rowExclude)) {
  missingText <- paste(missingText,
                      ifelse(sum(rowExclude) > 1,
                            " There were ",
                            " There was "),
                      sum(colExclude),
                      ifelse(sum(rowExclude) > 1,
                            " samples ",

```

```

        " sample "),
        "with an excessive number of ",
        "missing data. ",
        ifelse(sum(rowExclude) > 1,
              " These were excluded. ",
              " This was excluded. "),
        "After filtering, ",
        sum(!rowExclude),
        " samples remained.")
rawData <- rawData[!rowExclude, ]
hasMissing <- apply(rawData[, predictorNames, drop = FALSE],
                   1, function(x) mean(is.na(x)))
} else {
  hasMissing <- apply(rawData[, predictorNames, drop = FALSE],
                    1, function(x) any(is.na(x)))
  missingText <- paste(missingText,
                      ifelse(missingText == "",
                              "There ",
                              "Subsequently, there "),
                      ifelse(sum(hasMissing) == 1,
                              "was ",
                              "were "),
                      ifelse(sum(hasMissing) > 0,
                              sum(hasMissing),
                              "no"),
                      ifelse(sum(hasMissing) == 1,
                              "sample ",
                              "samples "),
                      "with missing values.")
}

```

@

The initial data set consisted of \Sexpr{numSamples} samples and \Sexpr{numPredictors} predictor variables. The breakdown of the outcome data classes were: \Sexpr{classDistString}.

%% \Sexpr{missingText}

```

<<pca, eval=FALSE, echo = FALSE, results = hide>>=
predictors <- rawData[, predictorNames, drop = FALSE]
## PCA will fail with predictors having less than 2 unique values
isZeroVar <- apply(predictors, 2,
                  function(x) length(unique(x)) < 2)
if(any(isZeroVar)) predictors <- predictors[ !isZeroVar, drop = FALSE]
## For whatever, only the formula interface to prcomp
## handles missing values
pcaForm <- as.formula(
  paste("~",
        paste(names(predictors), collapse = "+")))
pca <- prcomp(pcaForm,
             data = predictors,
             center = TRUE,
             scale. = TRUE,
             na.action = na.omit)
## OPTION: the number of components plotted/discussed can be set
numPCAcomp <- 3
pctVar <- pca$sdev^2/sum(pca$sdev^2)*100
pcaText <- paste(round(pctVar[1:numPCAcomp], 1),
                "\\|\\|%",
                sep = "")
pcaText <- listString(pcaText)

```

@

%% To get an initial assessment of the separability of the classes,



```

%% principal component analysis (PCA) was used to distill the
%% \Sexpr{numPredictors} predictors down into \Sexpr{numPCAcomp}
%% surrogate variables (i.e. the principal components) in a manner that
%% attempts to maximize the amount of information preserved from the
%% original predictor set. Figure \ref{F:initialPCA} contains plots of
%% the first \Sexpr{numPCAcomp} components, which accounted for
%% \Sexpr{pcaText} percent of the variability in the original predictors
%% (respectively).
%% OPTION: remark on how well (or poorly) the data separated
%% \setkeys{Gin}{width = 0.8\textwidth}
%% \begin{figure}[p]
%% \begin{center}
<<pcaPlot, eval = FALSE, echo = FALSE, results = hide, fig = TRUE, width = 8, height = 8>>=
trellis.par.set(caretTheme(), warn = TRUE)
if(numPCAcomp == 2)
{
  axisRange <- extendrange(pca$x[, 1:2])
  print(
    xyplot(PC1 ~ PC2,
      data = as.data.frame(pca$x),
      type = c("p", "g"),
      groups = rawData$outcome,
      auto.key = list(columns = 2),
      xlim = axisRange,
      ylim = axisRange))
} else {
  axisRange <- extendrange(pca$x[, 1:numPCAcomp])
  print(
    splom(~as.data.frame(pca$x)[, 1:numPCAcomp],
      type = c("p", "g"),
      groups = rawData$outcome,
      auto.key = list(columns = 2),
      as.table = TRUE,
      prepanel.limits = function(x) axisRange
    ))
}
@
%% \caption[PCA Plot]{A plot of the first \Sexpr{numPCAcomp}
%% principal components for the original data set.}
%% \label{F:initialPCA}
%% \end{center}
%% \end{figure}
<<initialDataSplit, eval = FALSE, results = hide, echo = FALSE>>=
## OPTION: in small samples sizes, you may not want to set aside a
## training set and focus on the resampling results.
pctTrain <- 1
if(pctTrain < 1)
{
  ## OPTION: seed number can be changed
  set.seed(1)
  inTrain <- createDataPartition(rawData$outcome,
    p = pctTrain,
    list = FALSE)
  trainX <- rawData[ inTrain, predictorNames]
  testX <- rawData[-inTrain, predictorNames]
  trainY <- rawData[ inTrain, "outcome"]
  testY <- rawData[-inTrain, "outcome"]
  splitText <- paste("The original data were split into ",
    "a training set (\$n\$=",
    nrow(trainX),
    ") and a test set (\$n\$=",
    nrow(testX),

```

```

        ") in a manner that preserved the ",
        "distribution of the classes.",
        sep = "")
isZeroVar <- apply(trainX, 2,
  function(x) length(unique(x)) < 2)
if(any(isZeroVar))
  {
    trainX <- trainX[, !isZeroVar, drop = FALSE]
    testX <- testX[, !isZeroVar, drop = FALSE]
  }
} else {
  trainX <- rawData[, predictorNames]
  testX <- NULL
  trainY <- rawData[, "outcome"]
  testY <- NULL
  splitText <- "The entire data set was used as the training set."
}
trainDist <- table(trainY)
nir <- max(trainDist)/length(trainY)*100
niClass <- names(trainDist)[which.max(trainDist)]
nirText <- paste("The non--information rate is the accuracy that can be ",
  "achieved by predicting all samples using the most ",
  "dominant class. For these data, the rate is ",
  round(nir, 2), "% using the `",
  niClass,
  "` class.",
  sep = "")

@
%% \Sexpr{splitText}
%% \Sexpr{nirText}
<<nzv, eval=FALSE, results = hide, echo = FALSE>>=
## OPTION: other pre-processing steps can be used
ppSteps <- caret::suggestions(modName)
set.seed(2)
if(ppSteps["nzv"])
  {
    nzv <- nearZeroVar(trainX)
    if(length(nzv) > 0)
      {
        nzvVars <- names(trainX)[nzv]
        trainX <- trainX[, -nzv]
        nzvText <- paste("There were ",
          length(nzv),
          " predictors that were removed due to",
          " severely unbalanced distributions that",
          " could negatively affect the model fit",
          ifelse(length(nzv) > 10,
            ".",
            ""),
          paste(":",
            listString(nzvVars),
            ".",
            sep = "")),
          sep = "")
        if(pctTrain < 1) testX <- testX[, -nzv]
      } else nzvText <- ""
    } else nzvText <- ""

@
<<corrFilter, eval = FALSE, results = hide, echo = FALSE>>=
if(ppSteps["corr"])
  {
    ## OPTION:
    corrThresh <- .75
  }

```

```

highCorr <- findCorrelation(cor(trainX, use = "pairwise.complete.obs"),
                           corrThresh)
if(length(highCorr) > 0)
{
  corrVars <- names(trainX)[highCorr]
  trainX <- trainX[, -highCorr]
  corrText <- paste("There were ",
                   length(highCorr),
                   " predictors that were removed due to",
                   " large between--predictor correlations that",
                   " could negatively affect the model fit",
                   ifelse(length(highCorr) > 10,
                          ".",
                          ""),
                   paste(":",
                        listString(highCorr),
                        ".",
                        sep = "")),
                " Removing these predictors forced",
                " all pair--wise correlations to be",
                " less than ",
                corrThresh,
                ".",
                sep = "")
  if(pctTrain < 1) testX <- testX[, -highCorr]
} else corrText <- ""
} else corrText <- ""
@
<<preProc, eval = FALSE, echo = FALSE, results = hide>>=
ppMethods <- NULL
if(ppSteps["center"]) ppMethods <- c(ppMethods, "center")
if(ppSteps["scale"]) ppMethods <- c(ppMethods, "scale")
if(any(hasMissing) > 0) ppMethods <- c(ppMethods, "knnImpute")
##OPTION other methods, such as spatial sign, can be added to this list
if(length(ppMethods) > 0)
{
  ppInfo <- preProcess(trainX, method = ppMethods)
  trainX <- predict(ppInfo, trainX)
  if(pctTrain < 1) testX <- predict(ppInfo, testX)
  ppText <- paste("The following pre--processing methods were",
                 " applied to the training",
                 ifelse(pctTrain < 1, " and test", ""),
                 " data: ",
                 listString(ppMethods),
                 ".",
                 sep = "")
  ppText <- gsub("center", "mean centering", ppText)
  ppText <- gsub("scale", "scaling to unit variance", ppText)
  ppText <- gsub("knnImpute",
                paste(ppInfo$sk, "--nearest neighbor imputation", sep = ""),
                ppText)
  ppText <- gsub("spatialSign", "the spatial sign transformation", ppText)
  ppText <- gsub("pca", "principal component feature extraction", ppText)
  ppText <- gsub("ica", "independent component feature extraction", ppText)
} else {
  ppInfo <- NULL
  ppText <- ""
}
}
predictorNames <- names(trainX)
if(nzvText != "" | corrText != "" | ppText != "")
{
  varText <- paste("After pre--processing, ",
                  ncol(trainX),

```

```

      "predictors remained for modeling.")
    } else varText <- ""

@
%% \Sexpr{nzvText} \Sexpr{corrText} \Sexpr{ppText} \Sexpr{varText}
\clearpage
\section*{Model Building}
<<setupWorkers, echo = FALSE, results = tex>>=
numWorkers <- 1
##OPTION: turn up numWorkers to use MPI
if(numWorkers > 1)
{
  mpiCalcs <- function(X, FUN, ...)
  {
    theDots <- list(...)
    parLapply(theDots$cl, X, FUN)
  }
  library(snow)
  cl <- makeCluster(numWorkers, "MPI")
}
@
<<setupResampling, echo = FALSE, results = hide>>=
##OPTION: the resampling options can be changed. See
## ?trainControl for details
resampName <- "boot632"
resampNumber <- 10
numRepeat <- 1
resampP <- .75
modelInfo <- modelLookup(modName)

if(numClasses == 2)
{
  foo <- if(any(modelInfo$probModel)) twoClassSummary else twoClassNoProbs
} else foo <- defaultSummary
set.seed(3)
ctlObj <- trainControl(method = resampName,
  number = resampNumber,
  repeats = numRepeat,
  p = resampP,
  classProbs = any(modelInfo$probModel),
  summaryFunction = foo)
##OPTION select other performance metrics as needed
optMetric <- if(numClasses == 2 & any(modelInfo$probModel)) "ROC" else "Kappa"

if(numWorkers > 1)
{
  ctlObj$workers <- numWorkers
  ctlObj$computeFunction <- mpiCalcs
  ctlObj$computeArgs <- list(cl = cl)
}
@
<<setupGrid, results = hide, echo = FALSE>>=
##OPTION expand or contract these grids as needed (or
## add more models
gridSize <- 3
if(modName %in% c("svmPoly", "svmRadial", "svmLinear", "lvq", "ctree2", "ctree")) gridSize <- 5
if(modName %in% c("earth", "fda")) gridSize <- 7
if(modName %in% c("knn", "rocc", "glmboost", "rf", "nodeHarvest")) gridSize <- 10
if(modName %in% c("nb")) gridSize <- 2
if(modName %in% c("pam", "rpart")) gridSize <- 15
if(modName %in% c("pls")) gridSize <- min(20, ncol(trainX))
if(modName == "gbm")

```

```

{
  tGrid <- expand.grid(interaction.depth = -1 + (1:5)*2,
                      .n.trees = (1:10)*20,
                      .shrinkage = .1)
}
if(modName == "nnet")
{
  tGrid <- expand.grid(.size = -1 + (1:5)*2,
                      .decay = c(0, .001, .01, .1))
}
@
<<fitModel, results = hide, echo = FALSE, eval = TRUE>>=
##OPTION alter as needed
set.seed(4)
modelFit <- switch(modName,
  gbm =
  {
    mix <- sample(seq(along = trainY))
    train(
      trainX[mix,], trainY[mix], modName,
      verbose = FALSE,
      bag.fraction = .9,
      metric = optMetric,
      trControl = ctlObj,
      tuneGrid = tGrid)
  },
  multinom =
  {
    train(
      trainX, trainY, modName,
      trace = FALSE,
      metric = optMetric,
      maxiter = 1000,
      MaxNWts = 5000,
      trControl = ctlObj,
      tuneLength = gridSize)
  },
  nnet =
  {
    train(
      trainX, trainY, modName,
      metric = optMetric,
      linout = FALSE,
      trace = FALSE,
      maxiter = 1000,
      MaxNWts = 5000,
      trControl = ctlObj,
      tuneGrid = tGrid)
  },
  svmRadial =, svmPoly =, svmLinear =
  {
    train(
      trainX, trainY, modName,
      metric = optMetric,
      scaled = TRUE,
      trControl = ctlObj,
      tuneLength = gridSize)
  },
  {
    train(trainX, trainY, modName,
          trControl = ctlObj,
          metric = optMetric,

```

```

        tuneLength = gridSize)
    })
@
<<modelDescr, echo = FALSE, results = hide>>=
summaryText <- ""
resampleName <- switch(tolower(modelFit$control$method),
  boot = paste("the bootstrap (", length(modelFit$control$index), " reps)", sep = ""),
  boot632 = paste("the bootstrap 632 rule (", length(modelFit$control$index), " reps)", sep = ""),
  cv = paste("cross-validation (", modelFit$control$number, " fold)", sep = ""),
  repeatedcv = paste("cross-validation (", modelFit$control$number, " fold, repeated ",
    modelFit$control$repeats, " times)", sep = ""),
  lgocv = paste("repeated train/test splits (", length(modelFit$control$index), " reps, ",
    round(modelFit$control$p, 2), "%)", sep = ""))
tuneVars <- latexTranslate(tolower(modelInfo$label))
tuneVars <- gsub("\\#", "the number of ", tuneVars, fixed = TRUE)
if(ncol(modelFit$bestTune) == 1 && colnames(modelFit$bestTune) == ".parameter")
{
  summaryText <- paste(summaryText,
    "\n\n",
    "There are no tuning parameters associated with this model.",
    "To characterize the model performance on the training set,",
    resampleName,
    "was used.",
    "Table \\ref{T:resamps} and Figure \\ref{F:profile}",
    "show summaries of the resampling results. ")
} else {
  summaryText <- paste("There",
    ifelse(nrow(modelInfo) > 1, "are", "is"),
    nrow(modelInfo),
    ifelse(nrow(modelInfo) > 1, "tuning parameters", "tuning parameter"),
    "associated with this model:",
    listString(tuneVars, period = TRUE))
  paramNames <- gsub(".", "", names(modelFit$bestTune), fixed = TRUE)
  for(i in seq(along = paramNames))
  {
    check <- modelInfo$parameter %in% paramNames[i]
    if(any(check))
    {
      paramNames[i] <- modelInfo$label[which(check)]
    }
  }
  paramNames <- gsub("#", "the number of ", paramNames, fixed = TRUE)
  ## Check to see if there was only one combination fit
  summaryText <- paste(summaryText,
    "To choose",
    ifelse(nrow(modelInfo) > 1,
      "appropriate values of the tuning parameters,",
      "an appropriate value of the tuning parameter,"),
    resampleName,
    "was used to generated a profile of performance across the",
    nrow(modelFit$results),
    ifelse(nrow(modelInfo) > 1,
      "combinations of the tuning parameters.",
      "candidate values."),
    "Table \\ref{T:resamps} and Figure \\ref{F:profile} show",
    "summaries of the resampling profile. ",
    "The
final model fitted to the entire training set was:",
    listString(paste(latexTranslate(tolower(paramNames)), "=", modelFit$bestTune[1,]), period = TRUE))
}
@

```

```

\Sexpr{summaryText}
<<resampTable, echo = FALSE, results = tex>>=
tableData <- modelFit$results
if(all(modelInfo$parameter == "parameter"))
{
  tableData <- tableData[,-1, drop = FALSE]
  colNums <- c(length(modelFit$perfNames), length(modelFit$perfNames))
  colLabels <- c("Mean", "Standard Deviation")
  constString <- ""
} else {
  isConst <- apply(tableData[, modelInfo$parameter, drop = FALSE],
    2,
    function(x) length(unique(x)) == 1)
  numParamInTable <- sum(!isConst)
  if(any(isConst))
  {
    constParam <- modelInfo$parameter[isConst]
    constValues <- format(tableData[, constParam, drop = FALSE], digits = 4)[1,,drop = FALSE]
    tableData <- tableData[, !(names(tableData) %in% constParam), drop = FALSE]
    constString <- paste("The tuning",
      ifelse(sum(isConst) > 1,
        "parameters",
        "parameter"),
      listString(paste("", names(constValues), "", sep = "")),
      ifelse(sum(isConst) > 1,
        "were",
        "was"),
      "held constant at",
      ifelse(sum(isConst) > 1,
        "a value of",
        "values of"),
      listString(constValues[1,]))
  } else constString <- ""
  cn <- colnames(tableData)
  for(i in seq(along = cn))
  {
    check <- modelInfo$parameter %in% cn[i]
    if(any(check))
    {
      cn[i] <- modelInfo$label[which(check)]
    }
  }
  colnames(tableData) <- cn
  colNums <- c(numParamInTable,
    length(modelFit$perfNames),
    length(modelFit$perfNames),
    length(modelFit$perfNames))
  colLabels <- c("", "Mean", "Standard Deviation", "Apparant")
}
colnames(tableData) <- gsub("SD$", "", colnames(tableData))
colnames(tableData) <- gsub("Apparant$", "", colnames(tableData))
colnames(tableData) <- latexTranslate(colnames(tableData))
rownames(tableData) <- latexTranslate(rownames(tableData))
latex(tableData,
  rowname = NULL,
  file = "",
  cgroup = colLabels,
  n.cgroup = colNums,
  where = "h!",
  digits = 4,
  longtable = nrow(tableData) > 30,

```

```

caption = paste(resampleName, "results from the model fit.", constString),
label = "T:resamps")
@
\setkeys{Gin}{ width = 0.9\textwidth}
\begin{figure}[b]
\begin{center}
<<profilePlot, echo = FALSE, fig = TRUE, width = 8, height = 6>>=
trellis.par.set(caretTheme(), warn = TRUE)
if(all(modelInfo$parameter == "parameter") | all(isConst) | modName == "nb")
{
resultsPlot <- resampleHist(modelFit)
plotCaption <- paste("Distributions of model performance from the ",
"training set estimated using ",
resampleName)
} else {
if(modName %in% c("svmPoly", "svmRadial", "svmLinear"))
{
resultsPlot <- plot(modelFit,
metric = optMetric,
xTrans = function(x) log10(x))
resultsPlot <- update(resultsPlot,
type = c("g", "p", "l"),
ylab = paste(optMetric, " (", resampleName, ")", sep = ""))
} else {
resultsPlot <- plot(modelFit,
metric = optMetric)
resultsPlot <- update(resultsPlot,
type = c("g", "p", "l"),
ylab = paste(optMetric, " (", resampleName, ")", sep = ""))
}
plotCaption <- paste("A plot of the estimates of the",
optMetric,
"values calculated using",
resampleName)
}
print(resultsPlot)
@
\caption[Performance Plot]{\Sexpr{plotCaption}.}
\label{F:profile}
\end{center}
\end{figure}
<<stopWorkers, echo = FALSE, results = hide>>=
if(numWorkers > 1) stopCluster(cl)
@
<<testPred, results = tex, echo = FALSE>>=
if(pctTrain < 1)
{
cat("\clearpage\n\section*{Test Set Results}\n\n")
classPred <- predict(modelFit, testX)
cm <- confusionMatrix(classPred, testY)
values <- cm$overall[c("Accuracy", "Kappa", "AccuracyPValue", "McNemarPValue")]
values <- values[!is.na(values) & !is.nan(values)]
values <- c(format(values[1:2], digits = 3),
format.pval(values[-(1:2)], digits = 5))
nms <- c("the overall accuracy", "the Kappa statistic",
"the $p$--value that accuracy is greater than the no--information rate",
"the $p$--value of concordance from McNemar's test")
nms <- nms[seq(along = values)]
names(values) <- nms
if(any(modelInfo$probModel))
{

```



```

classProbs <- extractProb(list(fit = modelFit),
                           testX = testX,
                           testY = testY)
classProbs <- subset(classProbs, dataType == "Test")
if(numClasses == 2)
{
  tmp <- twoClassSummary(classProbs, lev = levels(classProbs$obs))
  tmp <- c(format(tmp, digits = 3))
  names(tmp) <- c("the sensitivity", "the specificity",
                 "the area under the ROC curve")
  values <- c(values, tmp)
}
probPlot <- plotClassProbs(classProbs)
}
testString <- paste("Based on the test set of",
                   nrow(testX),
                   "samples,",
                   listString(paste(names(values), "was", values), period = TRUE),
                   "The confusion matrix for the test set is shown in Table",
                   "\\ref{T:cm}.")
testString <- paste(testString,
                   " Using ", resampleName,
                   ", the training set estimates were ",
                   resampleStats(modelFit),
                   ". ",
                   sep = "")
if(any(modelInfo$probModel)) testString <- paste(testString,
                                                "Histograms of the class probabilities",
                                                "for the test set samples are shown in",
                                                "Figure \\ref{F:probs}",
                                                ifelse(numClasses == 2,
                                                      " and the test set ROC curve is in Figure \\ref{F:roc}.",
                                                      ". "))

latex(cm$stable,
      title = "",
      file = "",
      where = "h",
      cgroup = "Observed Values",
      n.cgroup = numClasses,
      caption = "The confusion matrix for the test set",
      label = "T:cm")
} else testString <- ""
@
\Sexpr{testString}
<<classProbsTex, results = tex, echo = FALSE>>=
if(any(modelInfo$probModel))
{
  cat(
    paste("\\begin{figure}[p]\n",
          "\\begin{center}\n",
          "\\includegraphics{classProbs}",
          "\\caption[PCA Plot]{Class probabilities",
          "for the test set. Each panel contains ",
          "separate classes}\n",
          "\\label{F:probs}\n",
          "\\end{center}\n",
          "\\end{figure}")
  )
}
if(any(modelInfo$probModel) & numClasses == 2)
{
  cat(
    paste("\\begin{figure}[p]\n",

```

```

        "\\begin{center}\\n",
        "\\includegraphics[clip, width = .8\\textwidth]{roc}",
        "\\caption[ROC Plot]{ROC Curve",
        "for the test set.}\\n",
        "\\label{F:roc}\\n",
        "\\end{center}\\n",
        "\\end{figure}")
    }
@
<<classProbsTex, results = hide, echo = FALSE>>=
if(any(modelInfo$probModel))
{
  pdf("classProbs.pdf", height = 7, width = 7)
  trellis.par.set(caretTheme(), warn = FALSE)
  print(probPlot)
  dev.off()
}
if(any(modelInfo$probModel) & numClasses == 2)
{ resPonse<-testY
  preDicator<-classProbs[, levels(trainY)[1]]
  pdf("roc.pdf", height = 8, width = 8)
# from pROC example at http://web.expasy.org/pROC/screenshots.htm
  plot.roc(resPonse, preDicator, # data
    percent=TRUE, # show all values in percent
    partial.auc=c(100, 90), partial.auc.correct=TRUE, # define a partial AUC (pAUC)
    print.auc=TRUE, #display pAUC value on the plot with following options:
    print.auc.pattern="Corrected pAUC (100-90%% SP):\\n%.1f%%", print.auc.col="#1c61b6",
    auc.polygon=TRUE, auc.polygon.col="#1c61b6", # show pAUC as a polygon
    max.auc.polygon=TRUE, max.auc.polygon.col="#1c61b622", # also show the 100% polygon
    main="Partial AUC (pAUC)")
  plot.roc(resPonse, preDicator,
    percent=TRUE, add=TRUE, type="n", # add to plot, but don't re-add the ROC itself (useless)
    partial.auc=c(100, 90), partial.auc.correct=TRUE,
    partial.auc.focus="se", # focus pAUC on the sensitivity
    print.auc=TRUE, print.auc.pattern="Corrected pAUC (100-90%% SE):\\n%.1f%%", print.auc.col="#008600",
    print.auc.y=40, # do not print auc over the previous one
    auc.polygon=TRUE, auc.polygon.col="#008600",
    max.auc.polygon=TRUE, max.auc.polygon.col="#00860022")
  dev.off()
}
# commenting old roc commands
# {
#   rocPoints <- as.data.frame(
#     roc(classProbs[, levels(trainY)[1]],
#       testY,
#       positive = levels(trainY)[1]))
#   pdf("roc.pdf", height = 8, width = 8)
#   plot(1 - rocPoints$specificity, rocPoints$sensitivity,
#     ylab = "Sensitivity", xlab = "1 - Specificity",
#     type = "n",
#     main = paste("AUC:", round(aucRoc(rocPoints), 3)))
#   abline(0, 1, lty = 2, col = "darkgrey")
#   points(1 - rocPoints$specificity, rocPoints$sensitivity,
#     type = "S")
#   #
#   dev.off()
# }
@
\\section*{Versions}
<<versions, echo = FALSE, results = tex>>=
toLatex(sessionInfo())
@

```

```

<<save-data, echo = FALSE, results = hide>>=
## change this to the name of modName...
plsFit<-modelFit
save(plsFit,file="plsFit.RData")
@
The model was built using Partial Least Squares and is saved as plsFit.RData for reuse. This contains the variable
plsFit.
\end{document}

```

## V. Code for ensemble method

```

listString <- function (x, period = FALSE, verbose = FALSE)
{
  if (verbose) cat("\n  entering listString\n")
  flush.console()
  if (!is.character(x))
    x <- as.character(x)
  numElements <- length(x)
  out <- if (length(x) > 0) {
    switch(min(numElements, 3), x, paste(x, collapse = " and "),
          {
            x <- paste(x, c(rep(" ", numElements - 2), " and", ""), sep = "")
            paste(x, collapse = " ")
          })
  }
  else ""
  if (period) out <- paste(out, ".", sep = "")
  if (verbose) cat("  leaving listString\n\n")
  flush.console()
  out
}

resampleStats <- function(x, digits = 3)
{
  bestPerf <- x$bestTune
  colnames(bestPerf) <- gsub("^\\.", "", colnames(bestPerf))
  out <- merge(x$results, bestPerf)
  out <- out[ colnames(out) %in% x$perfNames]
  out <- format(out, digits = digits)
  out
}

extractEnsProb1<-
function (models, testX = NULL, testY = NULL, unkX = NULL, unkOnly = !is.null(unkX) &
  is.null(testX), verbose = FALSE)
{
  objectNames <- names(models)
  if (is.null(objectNames))
    objectNames <- paste("Object", 1:length(models), sep = "")
  if (any(unlist(lapply(models, function(x) !caret::modelLookup(x$method)$probModel))))
    stop("only classification models that produce probabilities are allowed")
  obsLevels <- caret::getClassLevels(models[[1]])
  trainX <- models[[1]]$trainingData[, !(names(models[[1]]$trainingData) %in%
    ".outcome")]
  trainY <- models[[1]]$trainingData$.outcome
  if (verbose) {
    cat("Number of training samples:", length(trainY), "\n")
    cat("Number of test samples:  ", length(testY), "\n\n")
  }
  ensProb <- NULL
  predProb <- NULL
  predClass <- NULL
  obs <- NULL

```

```

modelName <- NULL
dataType <- NULL
objName <- NULL
if (!is.null(testX)) {
  if (!is.data.frame(testX))
    testX <- as.data.frame(testX)
  hasNa <- apply(testX, 1, function(data) any(is.na(data)))
  if (verbose)
    cat("There were ", sum(hasNa), "rows with missing values\n\n")
  flush.console()
}
for (i in seq(along = models)) {
  modelFit <- models[[i]]$finalModel
  method <- models[[i]]$method
  modelstats <- resampleStats(models[[i]])
  scalingFactor <- as.numeric(modelstats$ROC)
  methodActive <- NULL
  if (verbose)
    cat("starting ", models[[i]]$method, "\n")
  flush.console()
  if (!unkOnly) {
    if (!is.null(testX) & !is.null(testY)) {
      if (!is.data.frame(testX))
        testX <- as.data.frame(testX)
      tempX <- testX
      tempY <- testY
      tempX$outcome <- NULL
      if (is.null(models[[i]]$preProcess)) {
        tempTestPred <- caret:::predictionFunction(method,
          modelFit, tempX)
        tempTestProb <- caret:::probFunction(method, modelFit,
          tempX)
      }
      else {
        ppTest <- predict(models[[i]]$preProcess, tempX)
        tempTestPred <- caret:::predictionFunction(method,
          modelFit, ppTest)
        tempTestProb <- caret:::probFunction(method, modelFit,
          ppTest)
      }
      if (verbose)
        cat(models[[i]]$method, ":", length(tempTestPred),
          "test predictions were added\n")
      predProb <- if (is.null(predProb))
        tempTestProb
      else rbind(predProb, tempTestProb)
      predClass <- c(predClass, as.character(tempTestPred))
      obs <- c(obs, as.character(testY))
      modelName <- c(modelName, rep(models[[i]]$method,
        length(tempTestPred)))
      objName <- c(objName, rep(objectNames[[i]], length(tempTestPred)))
      dataType <- c(dataType, rep("Test", length(tempTestPred)))
      methodActive <- as.numeric(c(methodActive, tempTestProb$active))
    }
  }
  if (!is.null(unkX)) {
    if (!is.data.frame(unkX))
      unkX <- as.data.frame(unkX)
    tempX <- unkX
    tempX$outcome <- NULL
    if (is.null(models[[i]]$preProcess)) {
      tempUnkPred <- caret:::predictionFunction(method, modelFit,

```

```

        tempX)
    tempUnkProb <- caret:::probFunction(method, modelFit,
        tempX)
    }
    else {
        ppUnk <- predict(models[[i]]$preProcess, tempX)
        tempUnkPred <- caret:::predictionFunction(method, modelFit,
            ppUnk)
        tempUnkProb <- caret:::probFunction(method, modelFit,
            ppUnk)
    }
    if (verbose)
        cat(models[[i]]$method, ":", length(tempUnkPred),
            "unknown predictions were added\n")
    predProb <- if (is.null(predProb))
        tempUnkProb
    else rbind(predProb, tempUnkProb)
    predClass <- c(predClass, as.character(tempUnkPred))
    obs <- c(obs, rep(NA, length(tempUnkPred)))
    modelName <- c(modelName, rep(models[[i]]$method,
        length(tempUnkPred)))
    objName <- c(objName, rep(objectNames[[i]], length(tempUnkPred)))
    dataType <- c(dataType, rep("Unknown", length(tempUnkPred)))
    methodActive <- as.numeric(c(methodActive,tempUnkProb$active))
    }
    if (verbose){
        cat("String methodActive is of length :", length(methodActive),"and Prob length is ",
length(tempTestProb$active),"n")
    }
    if (is.null(ensProb)){
        ensProb<-data.frame(predProb)
        ensProb$dataType<-dataType
        predClass <- factor(predClass, levels = obsLevels)
        obs <- factor(obs, levels = obsLevels)
        ensProb$obs<-obs
        ensProb$pred<-predClass
        ensProb[method]<-methodActive*scalingFactor
        sumProb<-ensProb[method]
        sumscalingFactor<-scalingFactor
    }else{
        ensProb[method]<-methodActive*scalingFactor
        sumProb<-(sumProb+(methodActive*scalingFactor))
        sumscalingFactor<-sumscalingFactor+scalingFactor
    }
    }
    if (verbose)
        cat("Reached the end of model loop \n")
    # ensProb$active<-NULL
    #ensProb$inactive<-NULL
    #ensProb$pred<-NULL
    ensProb$active<-as.vector(sumProb/sumscalingFactor)
    ensProb$inactive<-as.vector(1-ensProb$active)
    pred<-ifelse(ensProb$active > 0.5,"active","inactive")
    ensProb$pred<-factor(pred,levels=obsLevels)
    ensProb
}

```