ABSTRACT

SAHA, SOMA. Building an Essential Gene Classification Framework (under the direction of Dr. Steffen Heber).

The analysis of gene deletions is a fundamental approach for investigating gene function. We applied machine learning techniques to predict phenotypic effects of gene deletions in yeast. We created a dataset containing features that potentially have predictive power and then used feature processing techniques to improve the dataset and identify features that are important for our classification problem. We evaluated four different classification algorithms, K-Nearest Neighbors, Support Vector Machine, Decision Tree, and Random Forest, with respect to this problem. We used our framework to complement the set of experimentally determined essential yeast genes produced by the *Saccharomyces* Genome Deletion Project and produce more than 2000 annotations for genes that might cause morphological alterations in yeast.

Building an Essential Gene Classification Framework

by

Soma Saha

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial satisfaction of the
requirements for the Degree of
Master of Science in Computer Science

Computer Science

Raleigh

2005

Approved By:

Dr. Xiaosong Ma

Dr. Dennis Bahler

Dr. Steffen Heber
Chair of Advisory Committee

Biography

Soma Saha was born on the 16th of February, 1981. She is from Kolkata, India and is the second and youngest child of Sunil Kumar Saha and Jharna Saha. "Soma" means like the moon in Bengali and she was so named because she was born on a Monday. She received her Bachelor of Engineering in Computer Science and Engineering from R. V. College of Engineering, Bangalore, India in June 2003. She started her graduate studies at the North Carolina State University in Fall 2003. With the defense of this thesis, she is receiving the Master of Science in Computer Science.

Acknowledgements

I would like to thank Dr. Steffen Heber for guiding me throughout this project. I would like to thank Dr. Dennis Bahler and Dr. Xiaosong Ma for being on my advisory committee. I would like to acknowledge the support and encouragement my parents, Sunil Kumar Saha and Jharna Saha, and my brother, Sumit Kumar Saha, have given me throughout my student life. I would also like to mention the encouragement and help my friends have given me during the duration of my Master's. I would especially like to thank Anubhav Dhoot for the many technical discussions we had.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Saccharomyces cerevisiae was the first completely sequenced eukaryote, and it is one of the most studied model organisms to date. Despite its importance, we hardly know anything about phenotypic effects of most of its genes, and about one third of its 6600 genes have no functional annotation at all. The fundamental approach to understanding the function of a gene is to study the effect the deletion of that gene has on the organism. The *Saccharomyces* Genome Deletion Project [WSAL99] tried to collect as complete a set as possible of yeast deletion strains, using a PCR-based gene deletion strategy [BOKD+93, WBPP94]. The goal of this effort was to assign functions to yeast genes through phenotypic analysis of deletion mutants, and to identify essential genes. Essential genes are genes that an organism cannot survive without. In other words, the essential genes for an organism form the minimal set of genes that are required to support the life of an organism. Since genes contribute by performing certain functions, the set of essential genes is in effect the set of functions, which are indispensable for the organism. Giaever *et al.* visually screened 4401 of the yeast deletion mutants to identify genes involved in specifying cell shape and size. The deletion mutant morphologies were grouped into seven classes: elongated, round, small, large, pointed, clumped and other (mutants with more than three kinds of morphological phenotypes).

The PCR-based experimental method used to identify essential yeast genes could not be applied to the entire gene set. The dataset produced by the *Saccharomyces* Genome Deletion Project has around 1098 essential genes, 5198 non-essential genes, and 360 unclassified genes. We built a dataset consisting of features based on which essential genes could be predicted. We then applied supervised learning techniques to produce predictions for the unclassified set. We identified features that are important for essential gene prediction. We also used these features to produce morphological alteration predictions for genes that were not included in the study done by Giaever *et al.* We did this work using the R environment [R D04] and in the process, made additions to it that are useful for classification tasks. A part of this work is contained in the paper titled "*In Silico* Prediction of Yeast Deletion Phenotypes", which has been accepted for publication in the journal, *Genetics and Molecular Research* [SH].

1.2 Outline

In the following sections of this chapter, we briefly describe the biological background of the thesis and how computational techniques are used in the field of biology. Chapter 2 describes the dataset we built for our classification task. Chapter 3 describes the classification algorithms and techniques we used. Feature processing steps are discussed in chapter 4. In chapter 5 we talk about the environment in which we did all our work and our additions to it. In chapter 6 we present the evaluation of our techniques and the predictions we obtained. In chapter 7 we present some of the related work in this area. Finally, in chapter 8, we summarize our achievements and suggest future directions.

1.3 Biological Background

1.3.1 DNA

DNA(deoxyribonucleic acid) is the building block of life in almost all life forms. It contains all the information that these living things need to function cor-

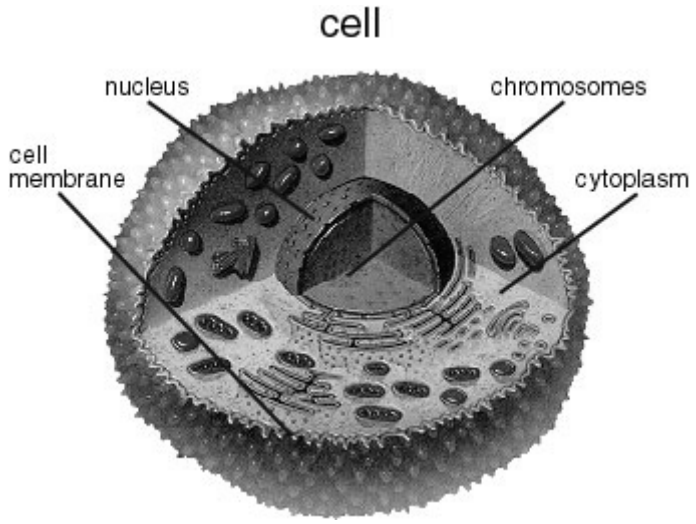


Figure 1.1: Structure of a cell showing the nucleus and chromosomes (http://www.harcourtschool.com)

rectly. It is also the chemical responsible for preserving, copying and transmitting information within cells from generation to generation. In higher organisms, DNA is contained in **chromosomes**, which are structures present in the nucleus of cells (Figure 1.1).

Nucleic acids are linear polymers of nucleotides. Nucleotides consist of three parts:

1. A five-carbon sugar(a pentose). DNA contains the pentose **deoxyribose**.

2. A nitrogenous ring structure called a **base**. In DNA, four different bases are found:

   - two purines, called **Adenine (A)** and **Guanine (G)**
   - two pyrimidines, called **Thymine (T)** and **Cytosine (C)**

   The combination of a base and a pentose is called a nucleoside.

3. A **phosphate** group.

In 1953, Watson and Crick first proposed the famous **"Double Helix"** structure of DNA. DNA consists of two long nucleotide chains, each twisted around the other to

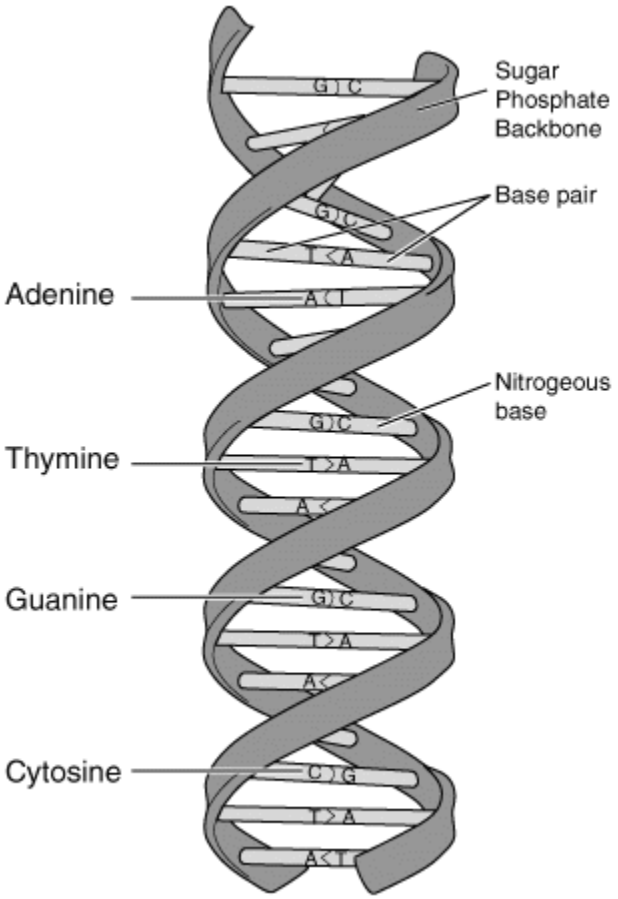


Figure 1.2: Structure of DNA showing Watson-Crick pairing (http://whyfiles.org)

form a double-stranded helix (Figure 1.2). At any position on the paired strands, the bases occur according to the **Watson-Crick pairing rules**. The rules are:

- the purine adenine (A) is paired with the pyrimidine thymine (T)
- the pyrimidine cytosine (C) is paired with the purine guanine (G)

The nucleotides found in the DNA strands are referred to by the letters A, T, C and G, representing the bases they contain. A **DNA sequence** is the precise sequence of nucleotides present in a DNA sample, represented by a succession of the four letters.

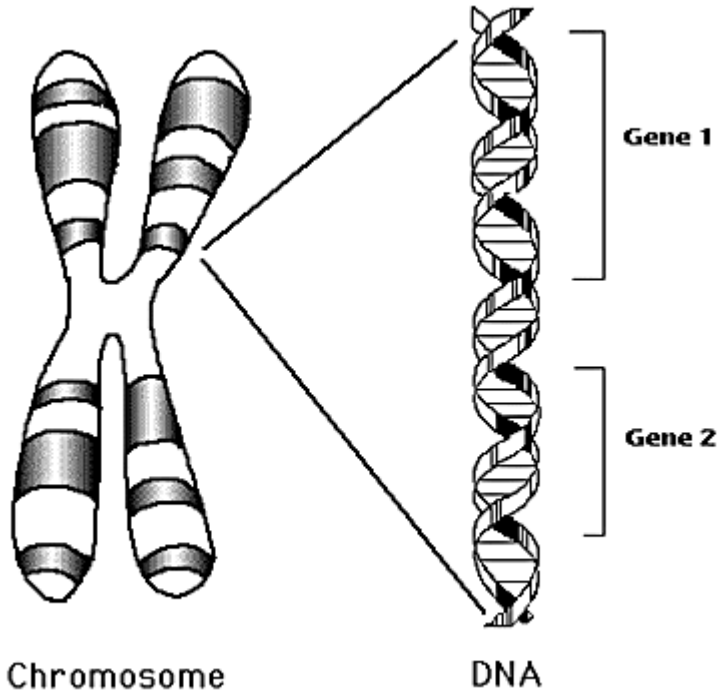


Figure 1.3: Location of DNA and genes on a chromosome (http://www.accessexcellence.org)

1.3.2 Genes, Gene Expression and Proteins

Genes are locatable regions of the DNA that correspond to the basic units of heredity in living cells (Figure 1.3). These entities encode information essential for the construction and regulation of proteins and other molecules that determine the growth and functioning of the organism.

Gene expression is the process by which information contained in a gene is converted into the structures and functions of a cell (mostly proteins). It involves the process of **transcription** followed by **translation**. Along with the process of **replication**, it forms the **central dogma** of molecular biology (Figure 1.4).

Transcription generates a **RNA (ribonucleic acid)** strand from a DNA template (a single strand of DNA obtained after the double helix of DNA opens up). The structure of RNA is similar to, but not identical with that of DNA. RNA contains

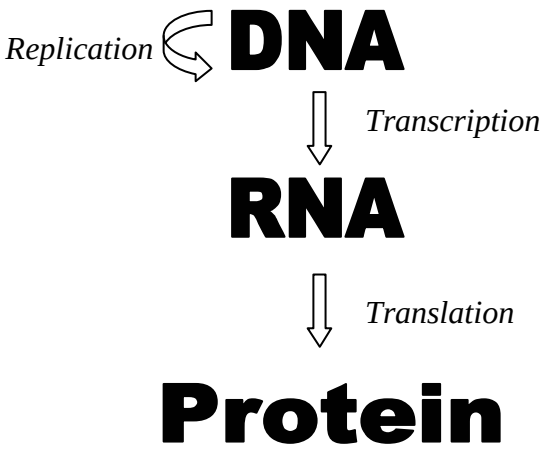


Figure 1.4: The Central Dogma of Molecular Biology

the sugar **ribose** instead of deoxyribose. It is usually single-stranded and it contains the base **Uracil (U)** instead of thymine (T). The base sequence in the RNA strand formed during transcription is complementary (in the Watson-Crick pairing sense) to that in the DNA template, except that U is paired with A. The three principal classes of RNA involved in the synthesis of proteins are **messenger RNA (mRNA)**, **transfer RNA (tRNA)**, and **ribosomal RNA (rRNA)**.

Translation converts the nucleotide sequence of RNA into the sequence of amino acids comprising a protein. The mRNA is translated into a protein sequence while the tRNA and the rRNA provide other components needed for protein synthesis. The entire length of the mRNA is not translated, but it contains at least one coding region that is related to a protein sequence by the **genetic code**: each nucleotide triplet, **codon**, of the coding region represents one amino acid. Some combinations of nucleotides specify the same amino acid, as a result there are 64 possible three-base combinations but only 20 amino acids. The process takes place on a ribosome, which combines with the mRNA and moves along it from one end to the other in steps of three nucleotides at a time (codon by codon), as shown in Figure 1.5. The amino acids are also represented by single letters and **protein sequences** are strings formed from these letters, which give the amino acid composition of the proteins.

Replication is the process by which the DNA double helix unwinds and makes

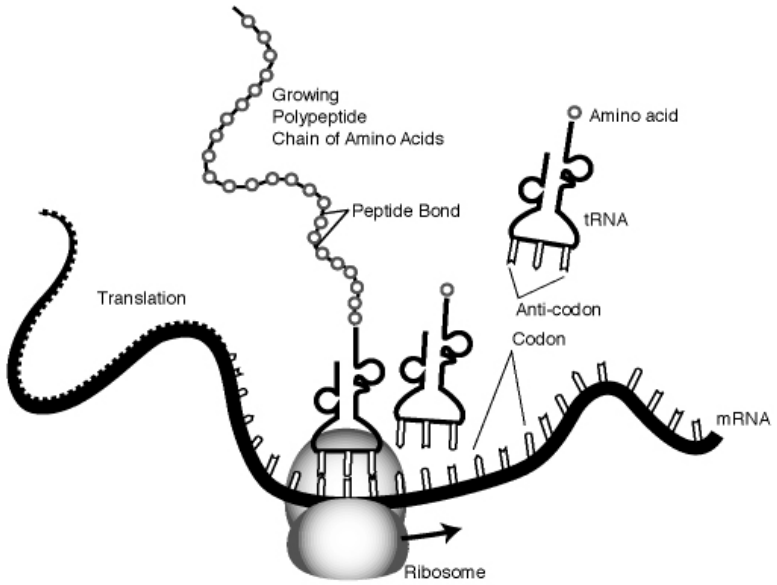


Figure 1.5: The Translation Process (http://en.wikipedia.org)

a copy of itself. This is how information is passed to new cells formed during cell division.

1.3.3 Evolutionary Relationships

Organisms, even very distinct ones, share many features at the genetic level. This is because of the evolutionary relationships that exist among these organisms. From an evolutionary perspective, this unity of fundamental molecular processes is derived by inheritance from a distant common ancestor in which the molecular mechanisms were already in place.

Homologs are genes related by descent from a common ancestral DNA sequence. Homologs can be of two types: **orthologs** and **paralogs**. Orthologs are genes in different species that evolved from a common ancestral gene by speciation (origin of new species). Paralogs are genes related by duplication within a genome (genetic material of an organism). Figure 1.6 shows the difference between the two.

Living organisms can be divided into three major domains: **bacteria** (simple

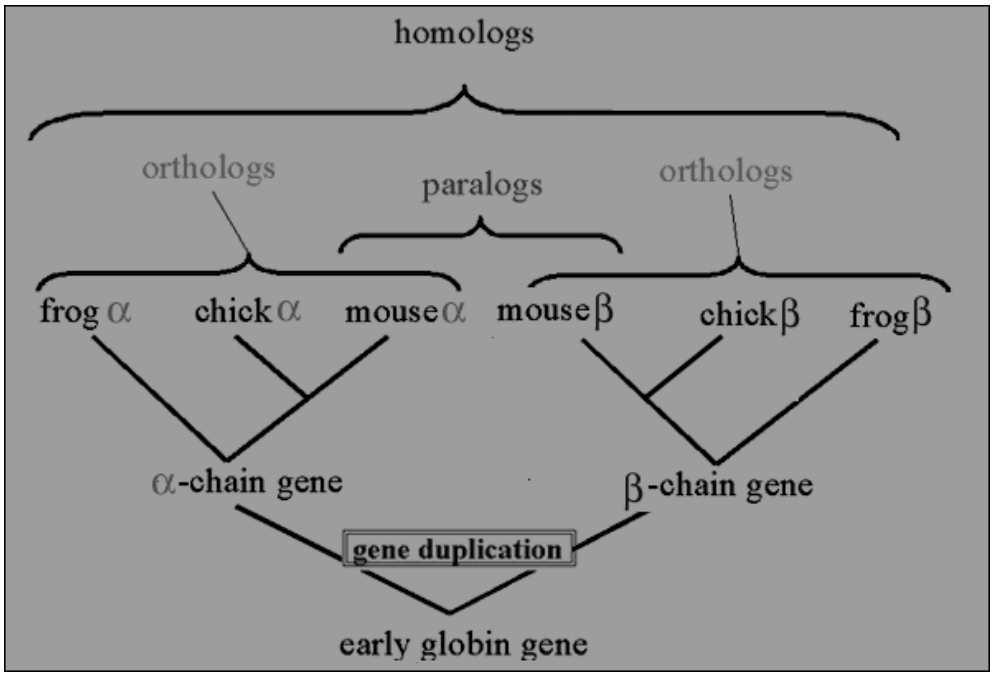


Figure 1.6: Formation of orthologs and paralogs (http://www.ncbi.nlm.nih.gov)

single-celled organisms lacking a nucleus), **archaea** (similar in structure to bacteria but resemble eukaryotes in their genetic transcription and translation), and **eukaryota** (organisms with complex cells in which is genetic material organized inside membrane-bound nuclei). The eukaryota domain contains all plants, animals, fungi and protista. Viruses are generally not classified as living, even though they have genes and show inheritance, because they are dependent on host cells for producing new generations of viruses.

1.4 Role of Bioinformatics

Biological databases contain a large volume of information and are growing faster than we can keep up with them. Several genomes have been completely sequenced thus enabling genome-wide investigations to take place. This much data could be overwhelming but with the right tools there exists the possibility of uncovering potentially useful information lying hidden in all this data. In the field of bioinformatics, we ap-

ply techniques from computer science and mathematics in order to develop methods to store, handle, and analyze this data. These methods can speed up investigations into biological problems or complement other experimental methods.

Chapter 2

Dataset Description

We downloaded the Yeast Deletion Collection of the *Saccharomyces* Genome Deletion Project (http://www-sequence.stanford.edu/group/yeast_deletion_project/deletions3.html) [WSAL99]. The dataset consisted of 1098 essential genes, 5198 non-essential genes, and 360 genes not available in the Yeast Deletion Collection. In addition, we downloaded an accompanying screen of 4401 deletion mutants for altered cell morphologies from the Supplementary Material of Giaever *et al.* [GCN+02].

We also collected features based on which we could try to classify the yeast genes. We included features that could potentially contain information about gene essentiality. Some of these features were based on related work on gene classification [LBO04, PMT+99, PATO03]. We collected a total of 1817 features. The following sections describe them.

2.1 Phylogenetic Profile

Functionally linked proteins usually evolve in a correlated fashion and therefore they might have homologs in the same subset of organisms. Conventional sequence alignment techniques cannot be used to identify functionally linked proteins because in general, pairs of such proteins have no amino acid sequence similarity with each other. Phylogeny is the evolutionary relationship among organisms. A phylogenetic profile of a protein describes the presence or absence of homologs of the protein in

other organisms or groups of organisms. Proteins with similar phylogenetic profiles have often been found to be functionally linked [PMT+99]. We hypothesize that phylogenetic profile might contain information about essentiality.

We created a protein phylogenetic profile for yeast genes by looking for homologs in the following subsets: viruses, archaea, bacteria, protista, fungi (excluding ascomycota), ascomycota (excluding *Saccharomyces cerevisiae*), plants, invertebrates, vertebrates (excluding mammals), and mammals. We obtained all the protein sequences available for the aforementioned subsets in Swissprot-Trembl [GGH+03] and CoGenT [JEA+03] databases. Tables 2.1, 2.2, and 2.3 show the number of sequences in each major domain of living organisms and in viruses that were present in the version of the above databases that we used.

Table 2.1: Distribution of sequences in CoGenT release 228

Domain	No. of Proteins
Archaea	46320
Bacteria	531920
Eukaryota	243875
Total	822115

Table 2.2: Distribution of sequences in Swiss-Prot release 46.1

Domain	No. of Proteins
Virus	9133
Archaea	9062
Bacteria	75183
Eukaryota	76762
Total	170140

Table 2.3: Distribution of sequences in TrEMBL release 29.1

Domain	No. of Proteins
Virus	292054
Archaea	45353
Bacteria	609846
Eukaryota	665523
Total	1612776

	viruses	bacteria	archaea	protista	fungi	plants	vertebrates	invertebrates
protein xyz	0.18	0.43	0.22	0.56	0.79	0.08	0.87	0.64

Figure 2.1: Example of a protein phylogenetic profile

We downloaded the *blastp* program, which compares an amino acid query sequence against a protein sequence database, from the National Center for Biotechnology Information (NCBI) website (http://www.ncbi.nlm.nih.gov). We performed a search of each yeast protein against the databases we created with the proteins in each group using this program with the BLOSUM62 matrix (a scoring matrix that assigns scores to amino acid pairs aligned together during the blast search) and an e-value threshold (a parameter that describes the number of hits one can expect to see just by chance) of 10^{-6}. A score between 0 and 1 was obtained in each category for each yeast gene by dividing the score of the best blast hit in each category (which represents the closest homolog) by the blast score of the protein against itself [LBO04]. This method avoids the need to specify a threshold for blast similarity scores, which is needed in binary phylogenetic profiles, where only 0 and 1 are used to indicate absence or presence of homologs. Figure 2.1 shows what a phylogenetic profile looks like.

2.2 Protein Domains

A protein has a unique structure determined by its amino acid sequence. The shape into which a protein naturally folds is known as its native state. A domain is structurally and functionally defined region in a protein. A domain can act as an independent unit, to the extent that it can be excised from the rest of the protein, and still be shown to fold correctly, and often still exhibit biological activity. Many domains are not unique to the protein products of one gene but instead appear in a variety of proteins. Some domains that occur frequently and figure prominently in biological functions are named and singled out. In multi-domain proteins, the combination of domains determines the function of the protein. For this reason domains might be important features for essential gene prediction.

The Protein Families (Pfam) database [BBD+00] contains a large collection of

common protein domains. We included domains from this database as features in our dataset. The domains were identified by their Pfam IDs and a 0 or a 1 was used to indicate the presence or absence of a particular domain, respectively. We obtained 1801 Pfam domains for the genes in our dataset.

2.3 Other Features

There are several features in our dataset other the ones obtained in the two categories described in the preceding sections. These features are described in the followings paragraphs. All these features, except protein-protein interactions, were obtained from the Ensembl database [HAC+05].

Paralogs are obtained through gene duplication and they reveal preferential enrichment in certain important functional classes. Therefore, we included degree of paralogy assessment in our dataset. This was calculated in a manner similar to the way the degree of conservation was calculated but instead of doing a blast for each yeast protein against proteins of other organisms, a blast search was done against the rest of the yeast proteins.

Protein interactions are involved in almost any physiological process. The number of interactions might indicate the importance of the protein. We obtained data on the number **protein-protein interactions** of yeast proteins from the Database of Interacting Proteins (DIP) [XSD+02].

We included **protein length** as a feature as conserved proteins have been observed to be, on average, longer than poorly conserved ones [LSK+02].

The presence or absence of **transmembrane and signal domains** are two other features in our dataset. Transmembrane domains are domains that extend across a membrane, usually referring to protein subunits that are exposed on both sides of a cell membrane. Signal domains are domains associated with signal transduction at the cellular level, i.e., movement of signals across cell boundaries.

The **GC content** percentage reveals the gene composition. The rest is made up of AT.

Chapter 3

Classification

3.1 Machine Learning

The construction of machines capable of learning from experience is an intriguing area of Artificial Intelligence (AI). These systems are of strategic importance as there are many tasks that cannot be solved using classical programming techniques. Handwriting recognition and classification problems are examples of such tasks.

When computers are usually used to solve problems, the methods required to derive outputs from given inputs are explicitly described. However, in some complex cases these methods cannot be described because there is no precise way to compute the outputs from the inputs. In such cases, an alternative strategy is employed. The computer is made to learn input-output functionality from examples. This is called the **learning methodology**. The examples form the **training data** for the machine.

We used **supervised learning**, in which the training set has pre-assigned output labels (labeled examples are not available in unsupervised learning). The classification problem is a supervised learning problem. A model is learnt from training data, which has class labels, and this is used to classify new data into one of the existing classes.

In supervised learning, a **learning algorithm** is used to find functional mappings between input output pairings in the training data. This process returns a **decision function**. This decision function represents the solution of the learning problem and is used to produce outputs for unseen data. It is chosen from a set of candidate

functions, known as **hypotheses**, each of which form valid mappings.

3.2 Classification Algorithms

We evaluated four different classification algorithms, k-nearest neighbor, support vector machine, decision tree, and random forest. All these classifiers are available as add-on packages in the R software (see chapter 5). We also tried the naïve bayes classifier. It is a simple probabilistic classifier based on Bayes theorem, which assumes that the effect of a feature on a given class is independent of the other features. This assumption is called class conditional independence. This assumption is a fairly strong assumption and is often not applicable. The naïve bayes classifier performed poorly on our dataset, so the observations have not been reported.

3.2.1 K-Nearest Neighbors

K-Nearest Neighbor (KNN) is a simple instance-based learning algorithm, which is a variant of the nearest neighbor classification method. The difference lies in the fact that rather than assigning a label based upon the class of the nearest neighbor this algorithm assigns a label based on the class of the k nearest neighbors, where k is a parameter.

Each feature is assigned a dimension to form a multidimensional feature space and all objects are plotted as points within this feature space. The distance between any two points is regarded as a measure for similarity of the objects they represent. The distance is normally calculated using an Euclidean distance measure, where distance between two points $\mathbf{x}$ and $\mathbf{y}$ in the vector space $\mathbb{R}^n$ is given by,

$$d(a, b) = |\mathbf{x} - \mathbf{y}| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}. \tag{3.1}$$

KNN has no separate training phase but the storing of the training data could be taken as the "training phase". Calculations are delayed until queries occur and no hypothesis in the usual sense is returned. The training dataset and the rules new

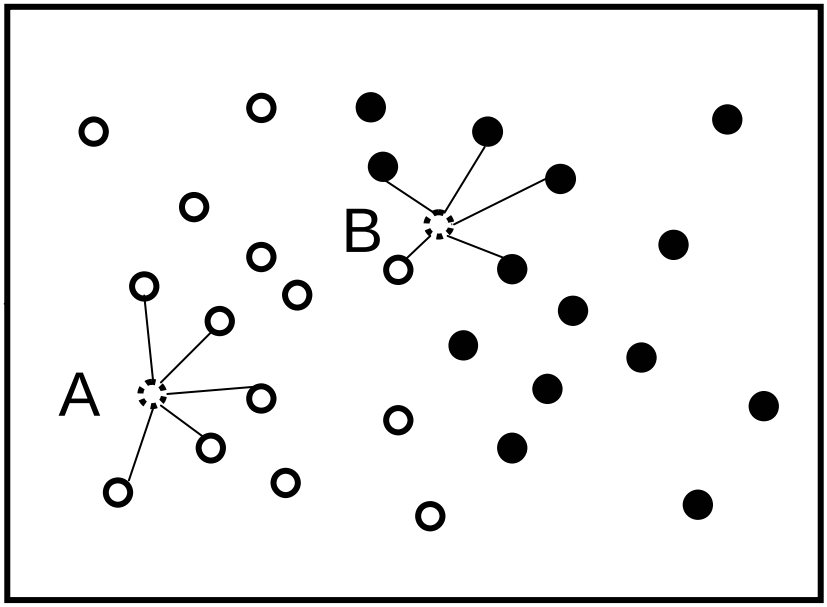


Figure 3.1: K-Nearest Neighbors Classification

instances are classified by form the implicit hypothesis. When a new instance needs to be classified, the algorithm locates the k closest neighbors in the training dataset and assigns it to the class to which most of its neighbors belong. The classification process is shown in Figure 3.1. The classification of B demonstrates the difference between KNN algorithm and the nearest neighbor algorithm. The nearest neighbor of B is a white dot but it is surrounded by black dots and it most probably belongs to the black class. The single white dot might be an anomaly but the nearest neighbor algorithm will pick the white class, as it does not use the voting mechanism used by KNN.

A major problem of the simple approach of KNN is that the vector distance may not be a suitable similarity measure, especially if irrelevant attributes are present.

3.2.2 Support Vector Machine

Support Vector Machine (SVM) is a relatively new learning algorithm that solves two-class pattern recognition problems [Vap99, CST00]. This algorithm also plots the training data as points in a high-dimensional space but the classification

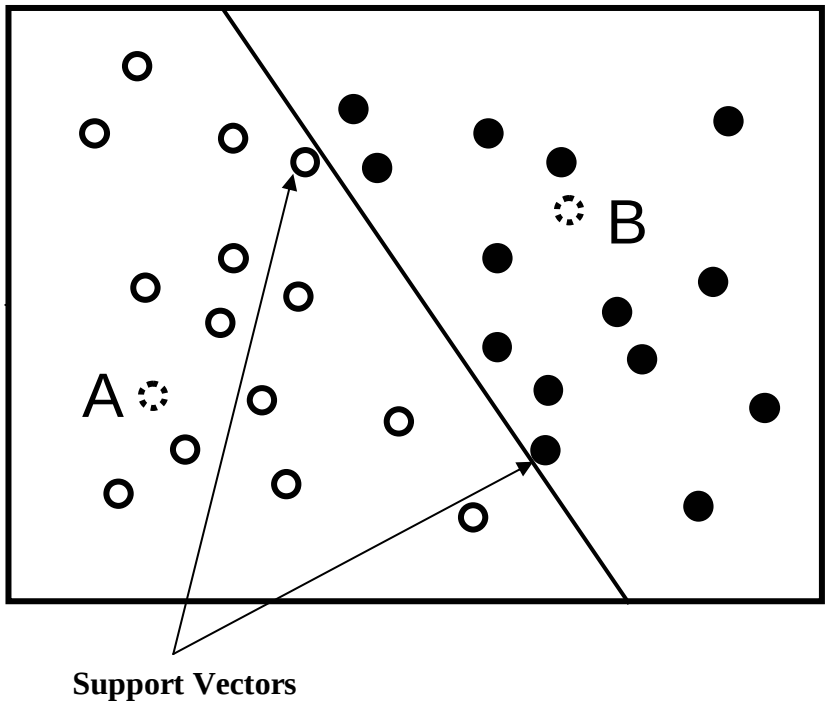


Figure 3.2: Classification using SVM model

takes place with the help of a decision hyperplane that separates the two classes in the training data in this space. A new object is classified by plotting it in the same high-dimensional space and determining on which side of the separating hyperplane it falls. The confidence of the classification is indicated by how far away it is from the plane.

Figure 3.2 shows how unknown points, A and B, are classified depending on their respective positions with respect to the hyperplane. The distance between the two planes that are parallel to the separating hyperplane and going through the vectors that are closest to it on either side, is called the margin of the SVM. The special vectors that constrain the margin of the classifier are called support vectors.

During the training phase of SVM, the algorithm searches the hypothesis space for the optimal hyperplane that separates the input vectors in such a way that cases with one label are on one side of the plane and cases with the other label are on the other size of the plane. However, the input points might be separated by a nonlinear

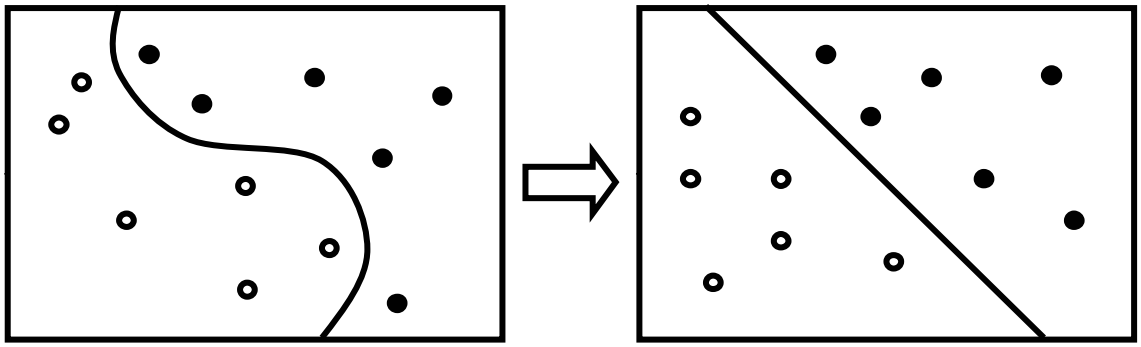


Figure 3.3: Using feature mapping to simplify classification task in SVM

region as shown on the left of Figure 3.3. Rather than fitting nonlinear curves to the data, SVM handles this using the "kernel trick". A kernel function (we used the radial basis function) projects the data into a feature space, where a hyperplane can be used to do the separation. The dimensionality of the feature space might differ from that of the input space. This process, called feature mapping is shown in Figure 3.3. The concept of a feature mapping is very powerful. It allows SVM models to perform separations even for datasets with very complex boundaries.

The kernel function usually takes a few parameters and the accuracy of an SVM model is largely dependent on the selection of the kernel parameters. This process is usually time consuming as the search space is big. SVM is most suited for two-class classification problems but multi-class classification problems might also be solved using a "one against rest" strategy.

3.2.3 Decision Tree

Decision Tree classifies instances by sorting them down a tree from the root to some leaf node, which provides the classification of the instance. The tree used for classification is characterized by the following:

- Each non-leaf node is associated with a feature.
- Each leaf node is associated with a class label.

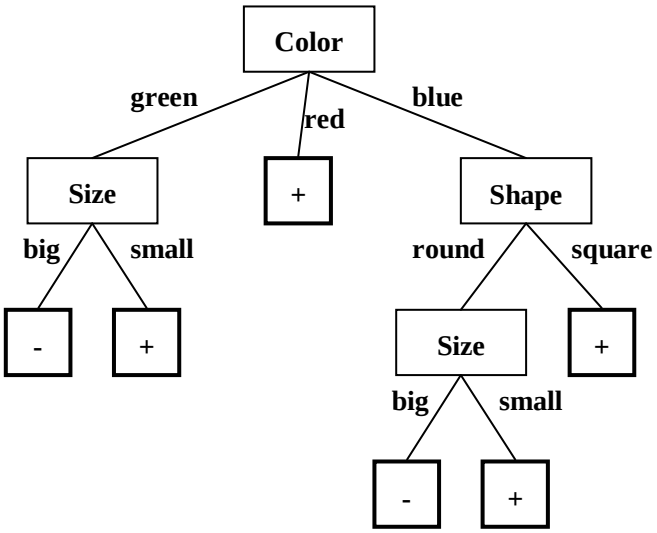


Figure 3.4: Decision Tree Classification

- Each branch (arc) of the tree corresponds to a possible value of the feature specified by the node from which it descends.
- Each path from the root to a leaf node corresponds to a valid conjunction of feature tests and the entire tree is a disjunction of these conjunctions.

The tree is constructed during the training phase. A decision tree can be learned by splitting the training set into subsets based on the value of a feature. This process is repeated in a recursive manner until splitting is either not feasible, or a unique classification is available for each instance of the training set.

Figure 3.4 is an example of a decision tree. In this example the two possible classifications are '+' and '-'. The features are color (with values green, red, and blue), size (with values big and small) and shape (with values round and square).

Large feature sets might cause the algorithm to return very large trees, which might cause overfitting (see section 3.3). In such cases decision trees are often pruned to produce smaller trees. Pruning might result in some loss of information.

3.2.4 Random Forest

Random Forest classifiers are constructed using an algorithm developed by Leo Brieman and Adele Cutler [Bre01]. The classifier uses a large number of individual decision trees and decides the class by choosing the mode (most frequently occurring) of the classes as determined by the individual trees. The individual trees are formed by selecting random vectors from the input vector for training. Growing an ensemble of trees in this manner usually results in significant improvements in classification accuracy.

The design of this algorithm has several advantages. It can handle thousands of input variables without variable deletion and it does not overfit. Each tree is grown to the largest extent possible and pruning is not used. Large input vectors can still be handled with ease because the individual trees use smaller subsets of the original vectors, thus making this classifier work efficiently even with large datasets.

3.3 Performance Estimation and Generalization

3.3.1 Classification Outcomes and Confusion Matrix

The classification process involves an estimation of how well a classification algorithm is performing with the given dataset. True positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), are the four possible outcomes of prediction for a two-class case with "positive" and "negative" classes. These outcomes are usually entered in a **confusion matrix**, where each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class (Figure 3.5). Various performance measures can be derived from the confusion matrix. The 3 used in this thesis are:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.2}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{3.3}$$

$$Specificity = \frac{TN}{TN + FP} \tag{3.4}$$

		Predicted Class	
		Yes	No
Actual Class	Yes	TP	FN
	No	FP	TN

Figure 3.5: Confusion Matrix Example (http://www.gepsoft.com)

Sensitivity is also called true positive rate. We used accuracy as a basis for comparison of performance in this thesis but we will report sensitivity and specificity as well. If we optimize sensitivity or specificity we might get trivial classifications (like all positives or all negatives) and still have very good performance values (100% sensitivity or 100% specificity, respectively).

3.3.2 ROC Curves

Receiver Operating Characteristic (ROC) curves are another way to examine the performance of classifiers [Swe88]. A ROC curve is a plot with the false positive rate ($FP/(TN + FP)$) on the X axis and the true positive rate ($TP/(TP + FN)$) on the Y axis. It shows how the threshold parameter of a classifier can be adjusted to increase true positive rate at the cost of increasing false positive rate and vice versa. A non-parametric classifier is represented by a single point. The point $(0, 1)$ corresponds to a perfect classifier: it classifies all positive cases and negative cases correctly. The point $(0, 0)$ corresponds to a classifier that predicts all cases to be negative, while the point $(1, 1)$ corresponds to a classifier that predicts every case to be positive. Point $(1, 0)$ corresponds to a classifier that is incorrect for all instances.

ROC curves provide a visual tool for examining the tradeoff between the ability of a classifier to correctly identify positive cases and the number of negative cases that are incorrectly classified. The area under the ROC curve describes the discriminatory power of the predictor, the greater the area, the better prediction algorithm. Figure

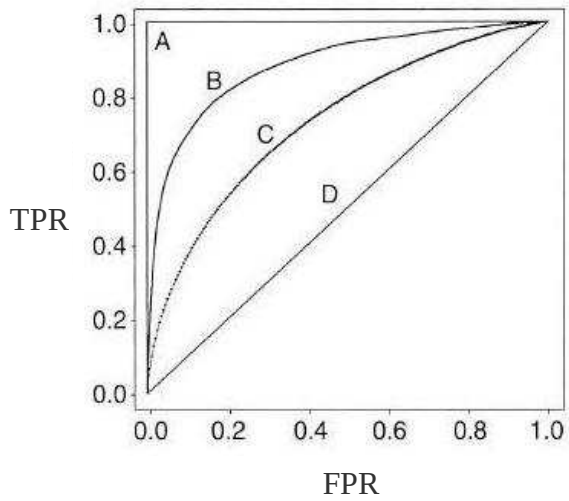


Figure 3.6: Example of ROC curves: the four curves in order of decreasing predictive power are A, B, C and D. (http://www.clinic-clinic.com)

3.6 shows an example.

3.3.3 Generalization

Generalization is the ability of a hypothesis to correctly classify data not seen during the training phase. Overfitting is the phenomenon where a learning algorithm adapts too well to a training set. Consequently, the performance on the test set suffers and the hypothesis does not generalize well.

One of the basic ways to measure overfitting is to keep a few training instances away from the classifier and use them to test the classifier, once the model is built. This is called the holdout mechanism. There are many variations to how holdout is actually performed. Cross-validation is one of these several approaches. The data set is divided into k subsets of approximately equal size, and training and testing of the algorithm is repeated k times. Each time, one of the k subsets is used as the test set and the other $k-1$ subsets are put together to form a training set. The average error across all k trials is then computed. The advantage of this method is that it matters less how the data is divided. Every data point gets to be in a test set exactly once, and gets to be in a training set $k-1$ times. The variance of the resulting estimate

is reduced as k is increased. The disadvantage of this method is that the training algorithm has to be rerun from scratch k times, which means it takes k times as much computation to make an evaluation.

If the algorithm is unstable with a particular dataset then cross validation performance estimate is likely to be unreliable because of high variance. Averaging multiple cross validation repetitions using different splits in the dataset might reduce variance and result in better estimation of performance [Koh95]. Repetition can be performed until desired reduction in variance is achieved. But this reduction comes at the added cost of a huge increase in computation time if multiple repetitions are required.

We perform k-fold cross validation with k set to 10 and also do variance reduction as described by Kohavi [Koh95] until the deviation from mean falls below 5%, to attain some stability in our accuracy estimates. If the variance is not sufficiently reduced after 10 attempts we report this observation and continue.

3.3.4 Imbalanced datasets

Many real life datasets are imbalanced i.e., at least one of the classes constitutes only a very small minority of the data. For such datasets, the interest usually leans towards correct classification of the rare class. Our current dataset is an example of this. Out of about 6000 yeast genes only around 1000 are known to be essential. Most commonly used classification algorithms do not work well for such problems because of the lack of examples for the rare class compared to size of the entire dataset.

There are two common approaches to handle the problem of extremely imbalanced data. One is based on cost sensitive learning: assigning a high cost to misclassification of the minority class and trying to minimize the overall cost. The other approach is to use a sampling technique: either down-sampling the majority class or over-sampling the minority class or both. We use the sampling approach and down-sample the majority class.

Chapter 4

Feature Processing

Feature processing is a very important phase in classification tasks. It tries to find a near optimal feature configuration (subset and feature transformations), which reduces computation time and increases classification performance. There are two distinct approaches for this processing phase, the filter approach and the wrapper approach [KJ98]. Filter approach attempts to judge the best feature set from data alone without any regards to the classification algorithm being used. The wrapper approach attempts to select features by using the classification algorithm to evaluate the selected features during the process. This difference is illustrated in 4.1.

The filter approach is usually much simpler and faster but it does not take the classification algorithm into account and does not try to tailor the feature set according to the bias of the algorithm. On the other hand, the wrapper approach does take the algorithm but it could be expensive to run.

We used a hybrid approach to feature processing: the initial steps were filter-based while the latter ones were wrapper-based. Our starting dataset had 1817 features. We used a filter approach in the first step to reduce computation time and quickly eliminate some of the redundant features. First, we combined some of the binary features, then we did feature selection step and finally we adjusted the weights of the features that were selected in the previous step.

F-score is an important tool we used in the first two steps. It measures the discrimination existing between two sets of real numbers (the positive and negative

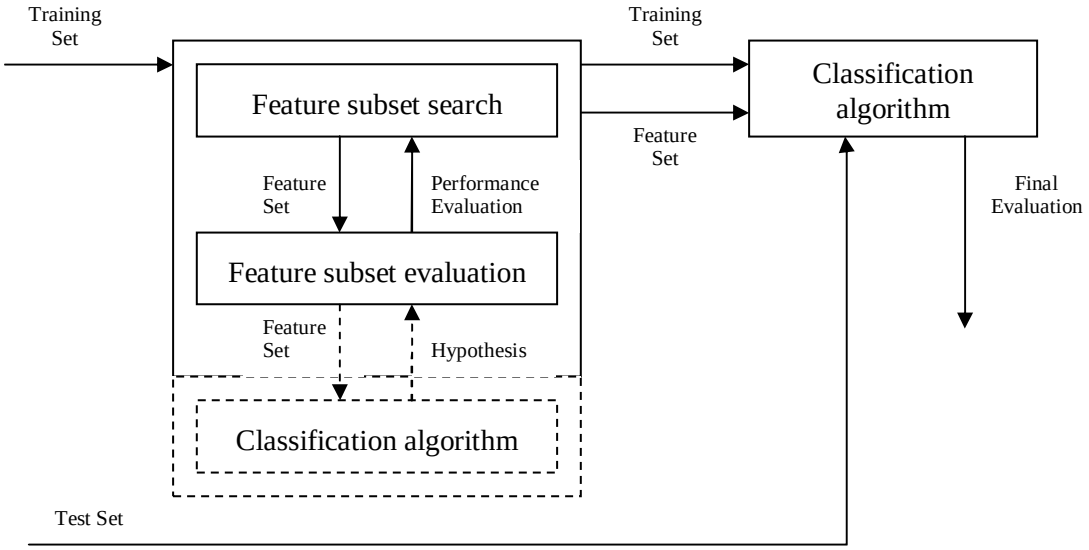


Figure 4.1: Difference between filter approach and wrapper approach; the dotted lines show the extra processing in the wrapper approach; in the filter approach the feature subset evaluation is independent of the classification algorithm while in the wrapper approach the classification algorithm is used in the feature subset evaluation

sets in our case). F-score is a simple filter-based method that does not take into account either the classification algorithm or how different features interact together. However, it is a quick method to estimate relative relevance of the features. Features with low f-scores are not likely to be very useful for classification. F-score is given by the following equation,

$$f = \frac{\left(\bar{x}^{(+)} - \bar{x}\right)^2 + \left(\bar{x}^{(-)} - \bar{x}\right)^2}{\frac{1}{n_+ - 1}\sum_{k=1}^{n_+}\left(x_k^{(+)} - \bar{x}^{(+)}\right)^2 + \frac{1}{n_- - 1}\sum_{k=1}^{n_-}\left(x_k^{(-)} - \bar{x}^{(-)}\right)^2} \tag{4.1}$$

where x is the entire set, $x^{(+)}$ is the positive set, $x^{(-)}$ is the negative set, n_+ is the number of instances in the positive set, and n_- is the number of instances in the negative set.

4.1 Binary Feature Combination

Binary features are features which can take on only two values (e.g., 0 and 1). If there are many such features present in the feature set, then we try to logically combine the information contained in those features to get a smaller feature set. This might result in a possible loss of information but it also reduces the dimensionality, which might help the classification process. The trade-off will vary from dataset to dataset.

There are many binary protein domain features in our dataset, most of which are present only in a few genes. We form sets of these domains and each set forms a new feature that replaces the individual domains contained in the set. If one or more domains contained in a set is present in a gene, then the feature corresponding to the set gets a value of 1. This is nothing but the logical "OR" operation. We use a greedy algorithm to combine the protein domains into domain sets for gene classification. First, the domains were ordered according to their f-scores. After that, sets of domains were created such that combining the domains in these sets resulted in a higher f-score than the f-score of the domains taken individually. If adding more domains to a particular set did not result in a higher f-score then we stopped adding

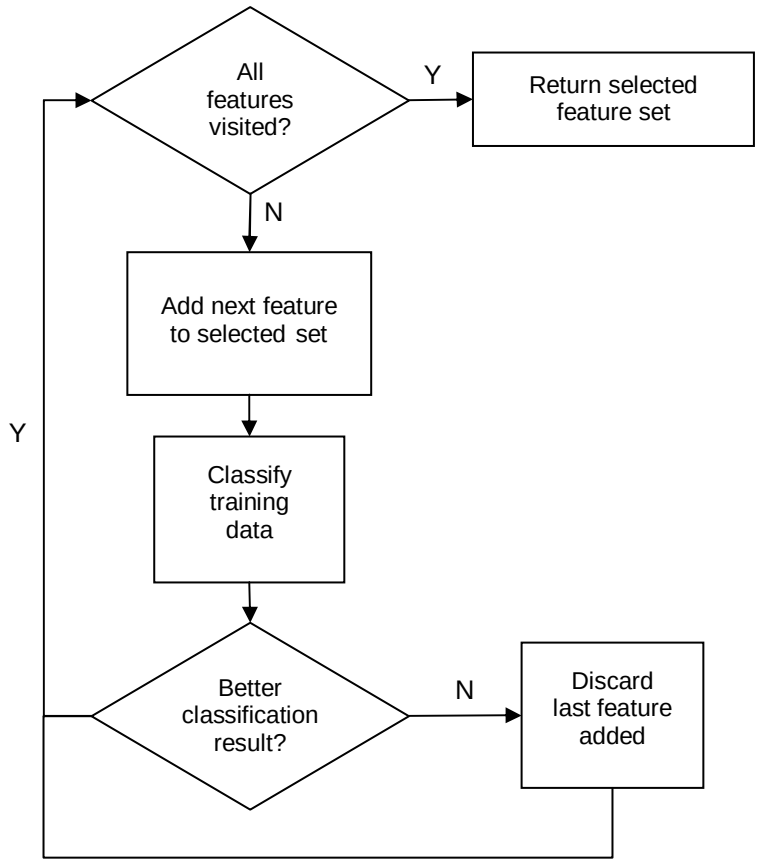


Figure 4.2: Forward Selction Flowchart

domains to that set and started a new set. This process can be repeated until there is no significant change in the feature set.

We repeated this procedure twice and then added these combined domain attributes to the other attributes to create our new feature set.

4.2 Feature Selection

Datasets may contain hundreds of features, many of which may be irrelevant or redundant. Feature selection reduces the number of features by removing such features from the dataset. Feature selection is an instance of the common subset selection problem. It is necessary step because each feature used as a part of a

classification procedure can increase the running time and irrelevant features might also result in poor classification. The latter is especially true when the sample size is small because the dataset has so many degrees of freedom that it would require many examples to learn the range of possible values. This is known as the "curse of dimensionality". However, it is also necessary to include sufficient features to enable the classification algorithm to recognize the classes.

Finding the best subset is usually an intractable problem because if there are m objects in a set then are a total of $2^m - 1$ subsets, so heuristics are usually applied to search a small but hopefully interesting fraction of the space of all subsets. One of these heuristics is forward selection, which starts with an empty subset, and at each stage tries to include a feature. This is a greedy approach and features once added are never removed. Forward selection returns a near optimal feature set in feasible amount of time.

We first ordered the features according to the f-score values and then used a wrapper approach to determine whether to add a feature to our selected set or not.

4.3 Feature Weighting

Adjusting the relative weights of the features is an optimization step that might improve the classification power of the feature set. Simulated Annealing (SA) is a global optimization algorithm [KGV83], which we used to adjust the weights of the features. The special characteristic of this algorithm is that while better solutions are always accepted, poorer solutions are also accepted with a non-zero probability. This prevents the algorithm from being stuck in a local optimum. We used a wrapper approach in this step also. In each iteration of simulated annealing a feature is selected for weight adjustment. The feature weights range from 0 and 1.

We blended some techniques of Tabu search [Glo86] with our algorithm in order to increase the optimization performance. Our algorithm stores recently visited attributes in a tabu list. The attributes in the tabu list cannot be selected for weight adjustment for the tabu tenure. The direction for weight adjustment, either positive or negative, is chosen based on the expected gain in classification accuracy. We then

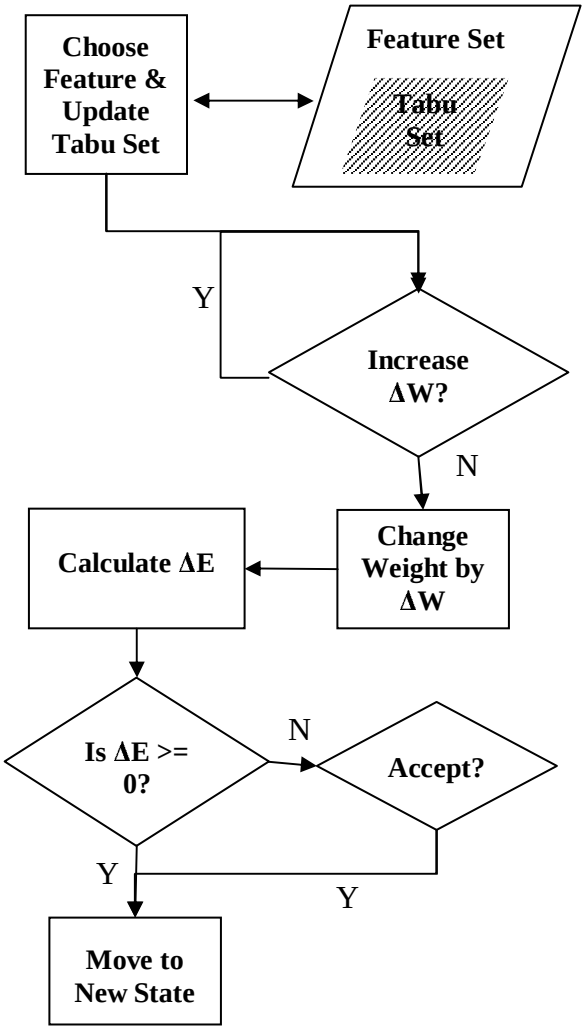


Figure 4.3: Flowchart showing a single iteration of Simulated Annealing: ΔW is the change in weight; ΔE is the change in objective function; Tabu Set represents the set of features that are excluded from consideration during the current iteration

keep adjusting the weight in small increments using hill climbing until we see no further improvement in the objective function.

Figure 4.3 shows the flowchart for a single iteration of the simulated annealing algorithm for feature weighting. At the end of the simulation the algorithm reports the best accuracy and corresponding set of weights.

There are various parameters which can be adjusted in our simulated annealing algorithm: the maximum number of iterations, the tabu length, the unit change in weight and the stopping criterion. We tried multiple combinations of these parameters. We varied the maximum iterations up to 6000, used 1/2, 1/3, and 1/4 the size of the feature set as our tabu length, and used 0.01, 0.05, and 0.1 as the unit change in weight.

Chapter 5

The R Framework

R is a free software for statistical computing and graphics, available under the GNU General Public License in source code form [R D04]. It is similar to the S language, which was developed at Bell Laboratories.

We chose to build our framework using R because there are a number of advantages. R is a free software and it is available for a wide variety of platforms, including Linux, Windows and several UNIX-based systems. R comes with several classification functions and graphics tools. It has very simple and effective data handling and storage facilities and convenient vector and matrix operations. In addition to these features, it is also a well-developed programming language, which can be used to extend the functions and write scripts.

All the four classification algorithms we used are available as add-on packages [VR02, DHL+05, Rip05, LW02]. We also used the ROCR package to produce ROC curves [SSBL05].

5.1 Additions to R

The following additions were made to R in the form of library functions. Some of these functions implement the processing steps described in the previous chapters and others are wrappers around existing functions available in R. These simplify the whole process of building feature sets, and training and testing classifiers.

- A wrapper for the classification algorithms available in R. This function takes the name of a classification algorithm as a parameter and hides the details from the user.
- A simple k-fold cross validation framework. It takes positive and negative sets, the number of folds, maximum variance tolerance and classification algorithm as arguments and returns the cross validation performance. It internally splits the datasets and creates training and test sets.
- A function to calculate f-scores for all features in a dataset.
- A feature combination function.
- Functions for forward selection and backward deletion. Both are wrapper-based: the classification algorithm is passed as a parameter.
- Functions for wrapper-based simulated annealing for feature selection and weighting.

Chapter 6

Results

The overall structure of our framework is shown in Figure 6.1. We performed all the processing steps (feature combination, feature selection and feature weighting) on our training sets and used the feature sets thus produced to train the four classifiers, KNN, SVM, decision tree and random forest. We then used the trained classifiers to produce predictions for the genes that could not be experimentally tested. We produced two sets of new annotations: a set of newly predicted essential genes and a set of genes causing morphological alterations.

However, we first present the results of our validation experiments. To validate each feature processing step we used a set containing 1098 essential genes and 1098 randomly sampled non-essential genes. We then evaluated each step by doing a 10-fold cross validation to estimate the performance. This outer cross validation loop is different from the internal cross validation that we perform on each training set while evaluating performance in wrapper-based approaches.

Table 6.1 shows the cross validation accuracy values obtained using the original dataset and Tables 6.2, 6.3, and 6.4 show the same for each feature processing step. The average number of features after feature combination is 97. This is a big reduction from the 1817 features we started with. The feature selection step is a wrapper-based one, so the number of features selected varied with each algorithm. The average number of features is 12 for decision tree, 20 for SVM and random forest, and 21 for KNN (Figure 6.2).

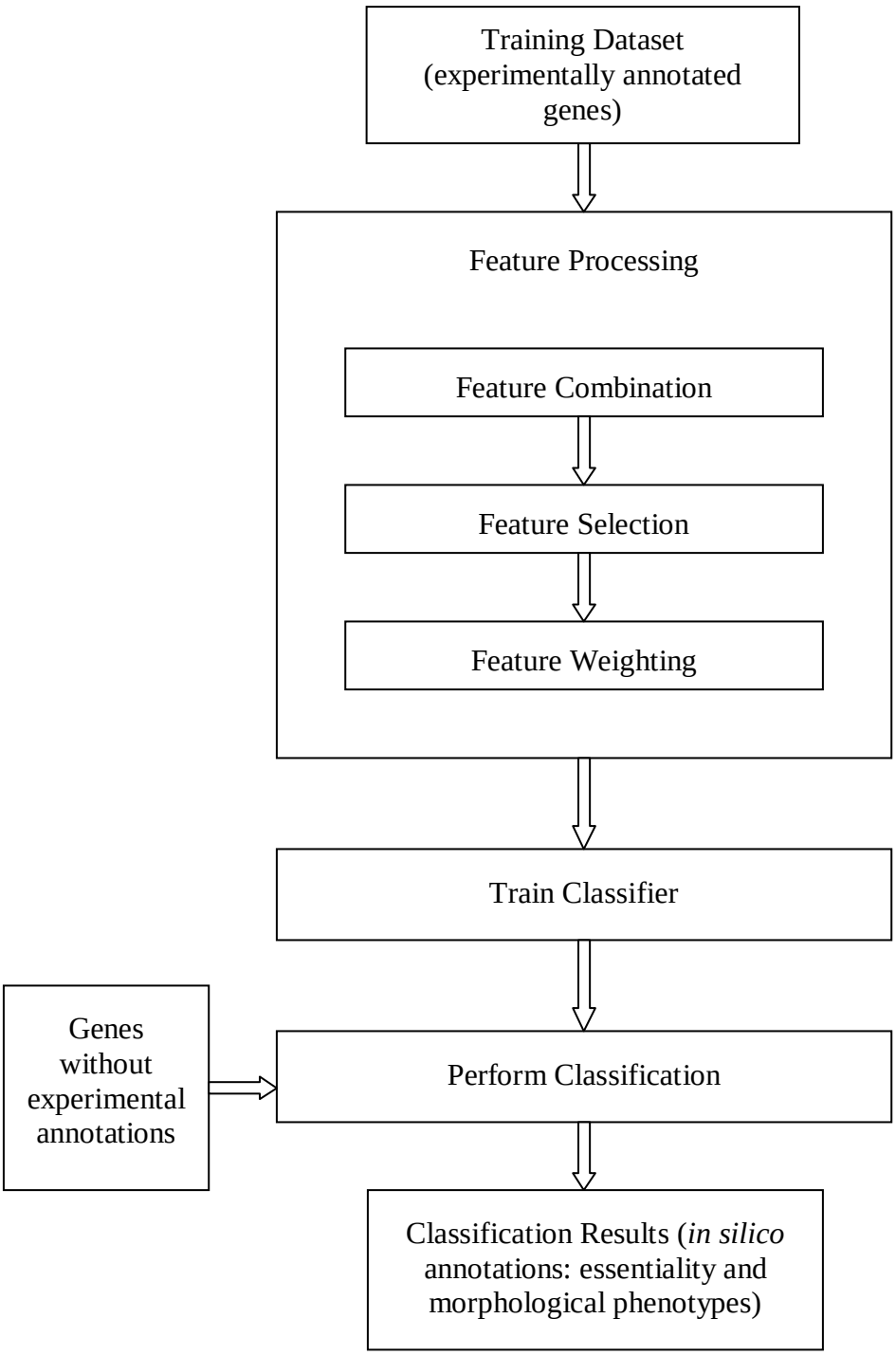


Figure 6.1: Overall structure of our framework

Table 6.1: Accuracy values with the 1817 original features

Round	**knn**	**svm**	**dt**	**rf**
1	60.55	67.88	64.22	64.22
2	61.00	64.67	74.31	71.11
3	67.43	71.55	67.88	68.34
4	73.39	75.68	64.67	69.72
5	70.60	75.22	71.10	72.93
6	75.22	74.31	74.77	75.68
7	70.18	77.52	68.34	72.93
8	66.97	75.22	66.51	69.72
9	75.68	77.98	73.39	76.14
10	68.80	70.64	66.51	72.47
Average	68.98	73.06	69.17	71.32
SD	5.25	4.31	3.95	3.54

Table 6.2: Accuracy values after feature combination

Round	**knn**	**svm**	**dt**	**rf**
1	65.59	66.05	62.84	68.34
2	65.13	63.30	72.93	71.55
3	72.47	73.85	69.72	74.77
4	71.10	75.22	64.22	72.93
5	70.18	72.01	69.26	77.52
6	72.01	72.93	74.31	77.06
7	73.85	78.44	71.55	78.89
8	71.55	72.01	65.13	76.60
9	73.85	76.60	72.47	77.06
10	70.18	72.93	70.18	76.60
Average	70.59	72.33	69.26	75.13
SD	3.03	4.57	3.93	3.25

Table 6.3: Accuracy values after feature selection

Round	**knn**	**svm**	**dt**	**rf**
1	64.22	63.76	64.67	68.80
2	72.01	72.47	73.39	70.64
3	72.47	73.39	68.80	75.22
4	65.13	70.64	64.22	70.64
5	74.77	70.64	69.72	71.10
6	76.14	78.89	73.85	76.60
7	75.68	76.14	71.55	74.77
8	71.55	72.93	63.30	73.39
9	76.14	71.10	71.10	72.47
10	72.70	70.64	70.18	74.31
Average	72.08	72.06	69.07	72.79
SD	4.26	3.97	3.79	2.47

Table 6.4: Accuracy values after feature weighting

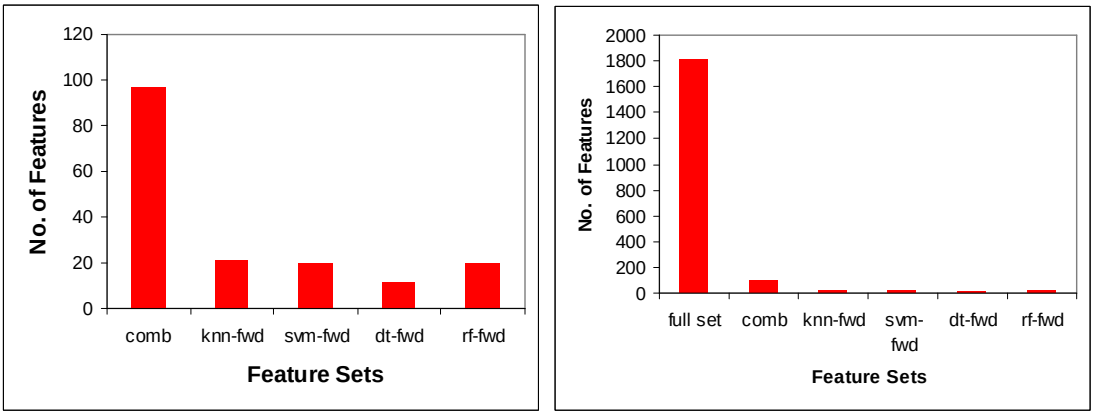
Round	**knn**	**svm**	**dt**	**rf**
1	65.59	63.76	64.67	67.88
2	72.93	72.47	73.39	71.1
3	74.77	73.39	68.8	76.14
4	67.43	70.64	64.22	71.55
5	76.14	70.64	69.72	71.55
6	75.68	78.89	73.85	77.98
7	75.22	76.14	71.55	75.22
8	71.1	72.93	63.3	73.39
9	77.52	71.1	69.72	72.47
10	72.47	70.64	70.18	74.31
Average	72.88	72.06	68.94	73.15
SD	3.87	3.97	3.73	2.89

Figure 6.2 shows the number of features at the start, after feature combination and after forward selection using each of the four classifiers. As is evident from the figure, there was a huge reduction after the combination step and another big reduction after feature selection. The two steps together reduced the number of features from 1817 to a number between 12–21. The corresponding effect on the time taken by the classifiers is shown in Figure 6.3. These measurements were taken on the cluster we used for all our experiments, under normal load. Measurements may vary with the speed of the processor and load present while taking them but they indicate the relative speed of the classifiers.

If we compare the accuracy values in Figure 6.4, we can see that the average accuracy value increased by 5.6% for KNN after feature processing. The timing change (Figure 6.3) is noticeable and the performance of KNN is on par with SVM and random forest. Decision tree is the lowest performing classifier of the four we tried. While the change in timing is similar to KNN, there is no corresponding change in accuracy. If we now look at the performance of SVM and random forest, we see that there is a slight decrease and a slight increase in performance, respectively. While neither the increase nor the decrease are significant, we see a very big improvement in the timing chart (Figure 6.3). These classifiers took prohibitively long times with the larger feature sets but the timings recorded with the smallest set are feasible.

Overall, feature processing had a positive impact. The number of features was drastically reduced and the timing improved while maintaining good performance. However, contrary to our expectations, the feature weighting step did not have significant effect on the current dataset.

Figure 6.7 shows the ROC curves for the cross validation runs for each classifier. Most of the cross validation curves are grouped close together, which means that the classifiers performed consistently with each cross validation test set. Figure 6.8 show the average ROC curves of all the four classifiers in a single graph. This figure enables us to visually compare the performance of the classifiers. The curves confirm our observations from the accuracy value comparisons. KNN, SVM, and random forest have similar curves, while the decision tree curve is below the other curves. This means the discriminatory power of decision tree is lower than that of the other



A B

Figure 6.2: Feature Set Comparisons A: feature sets obtained after combination and forward selection. B: comparison of same feature sets with original feature set

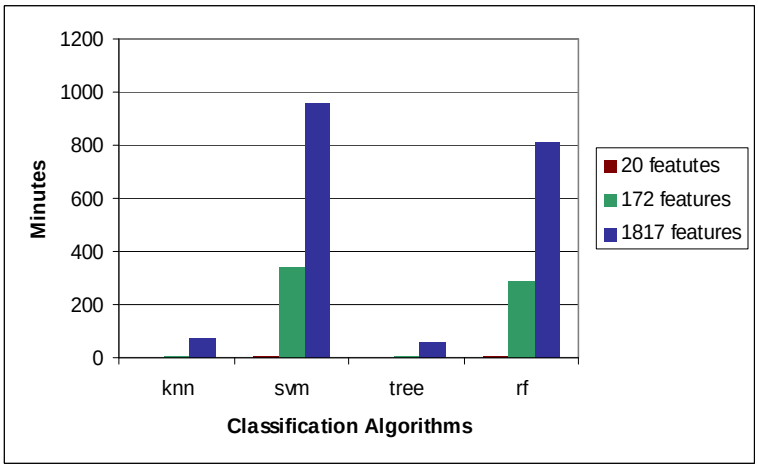


Figure 6.3: Timing of 100 iterations for the four classifiers using feature sets of different sizes; measurements shown are the averages over three runs

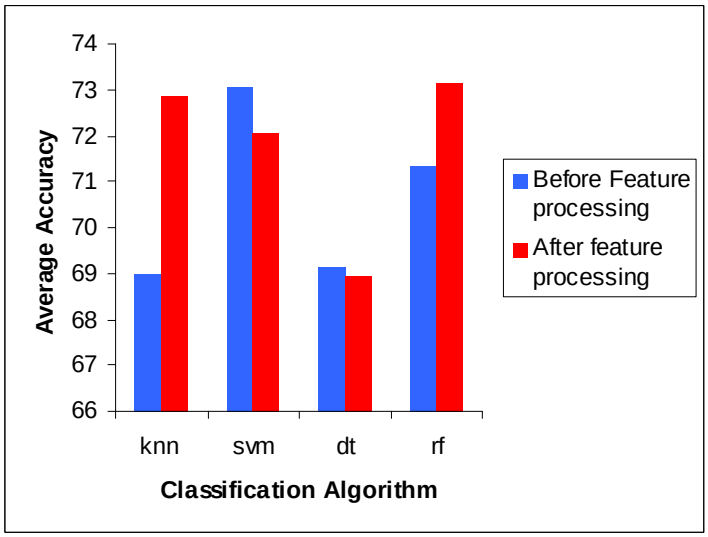


Figure 6.4: Accuracy of the classifiers before and after feature processing

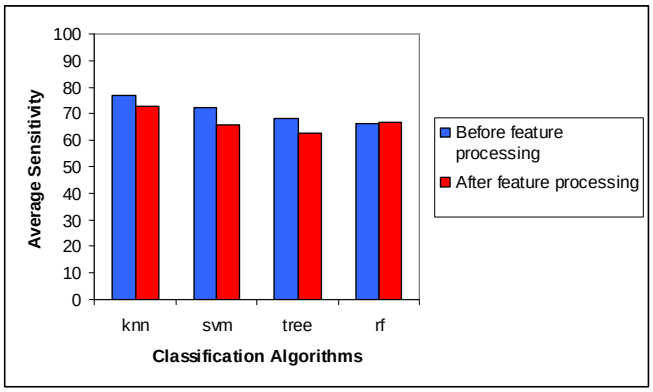


Figure 6.5: Sensitivity of the classifiers before and after feature processing

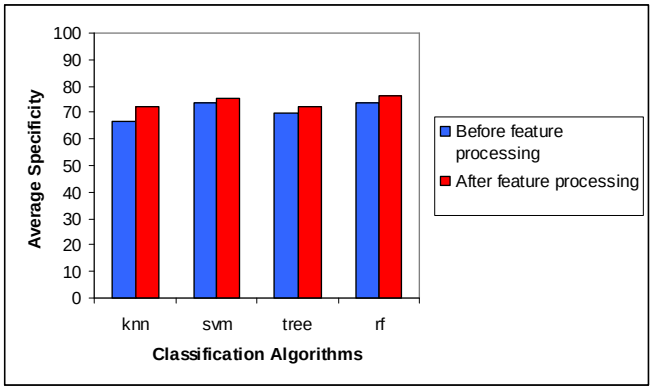


Figure 6.6: Specificity of the classifiers before and after feature processing

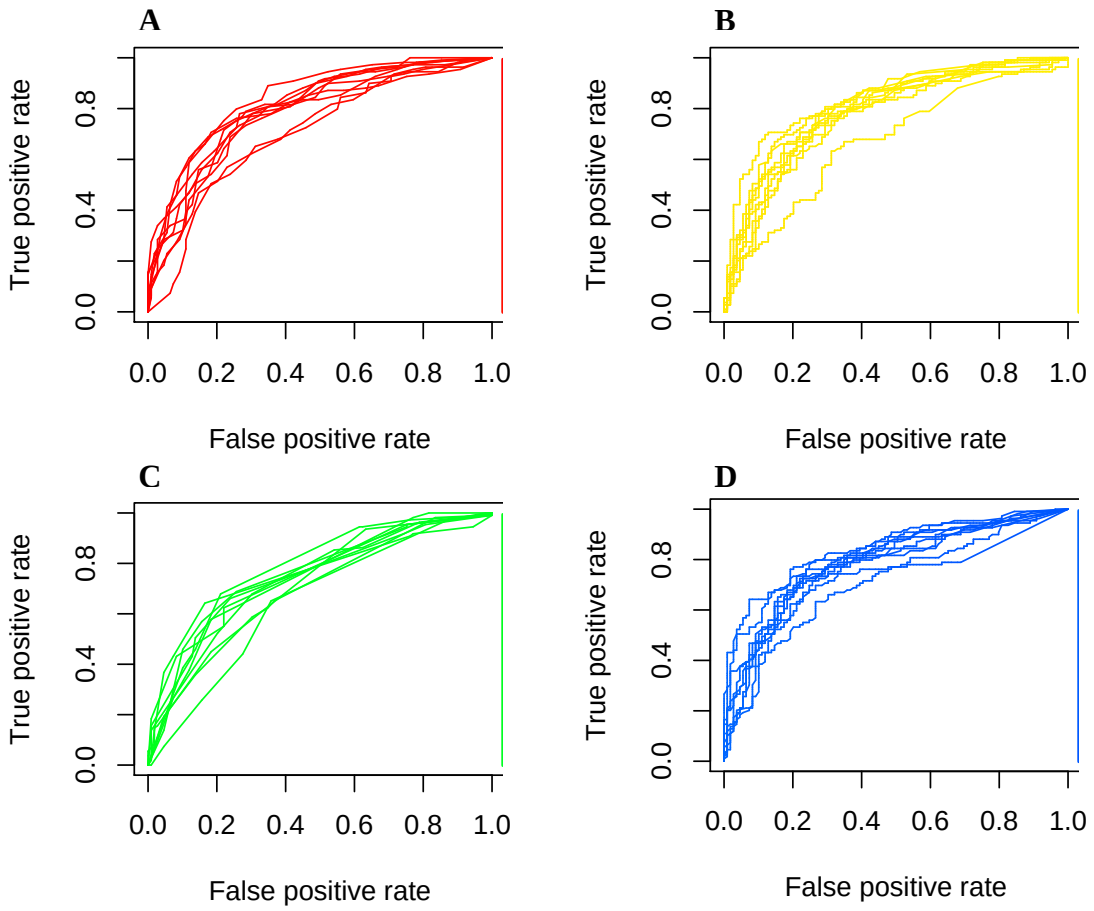


Figure 6.7: ROC curves for 10-fold cross validation for A: KNN, B: SVM, C: Decision Tree, and D: Random Forest

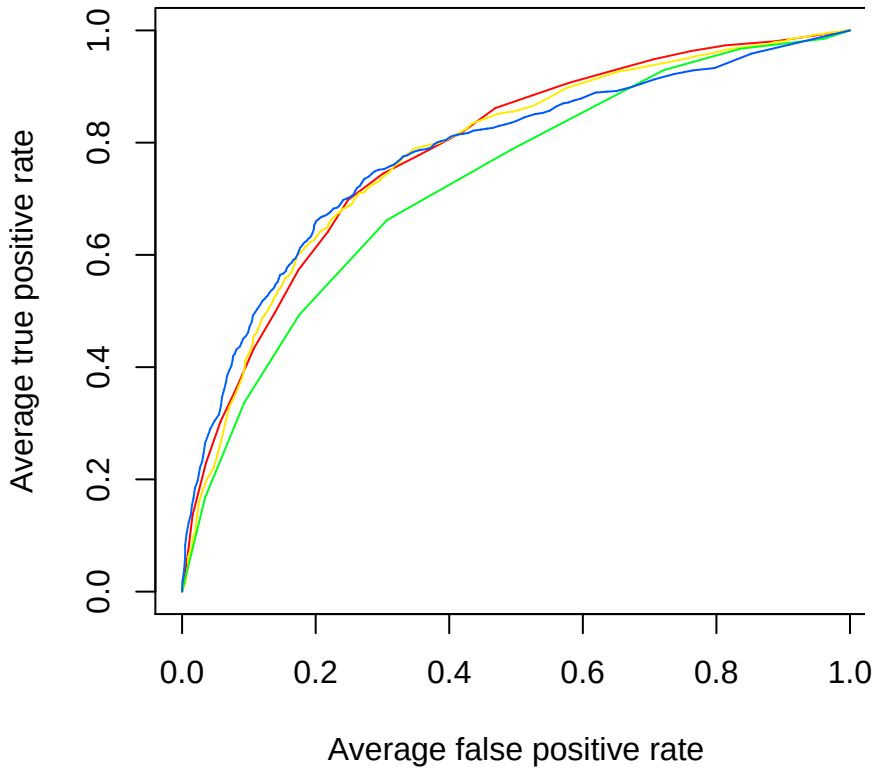


Figure 6.8: Average ROC curves of the four classifiers, KNN (red), SVM (yellow), Decision Tree (green), and Random Forest (blue)

classifiers.

Table 6.5 shows the average area under the ROC curves. This table confirms our previous observations. We can see that the values for KNN, SVM and random forest are almost the same. Decision tree has a lower value.

Table 6.5: Area under ROC curve (AUC) for all four classifiers

Classifier	AUC
KNN	0.7843
SVM	0.7833
Decision Tree	0.7321
Random Forest	0.7805

Table 6.6 shows the features selected for the four classifiers. We can see some differences in the features sets but degree of conservation in mammals, number of protein-protein interactions, degree of paralogy and multiple combined protein domains are present in all the sets.

Table 6.6: Features selected most frequently for the four classifiers including the average number of combined domains. The features are degree of conservation in mammals (m), vertebrates (ve), invertebrates (i), fungi (f), plants (pl), and protista (pr), number of protein-protein interactions (pi), and degree of paralogy (p).

Classifier	Selected Features
KNN	m, ve, i, pl, pr, pi, p, 13 combined domains
SVM	m, pl, pi, p, 14 combined domains
Decision Tree	m, ve, f, pi, p, 6 combined domains
Random Forest	m, f, pi, p, 14 combined domains

As stated in chapter 2 there are around 360 unclassified genes in the Yeast Deletion Collection. These are the genes for which the PCR-based deletion experiment could not be performed. We tried to predict essential genes in this set using our computational method. We trained the four classifiers using the set of experimentally characterized genes. Table 6.7 shows the genes that were predicted as essential along with a brief description for them, obtained from Ensembl [HAC+05]. To increase reliability of the predictions, only genes that were predicted as essential by at least 3 classifiers (42 in number) are shown.

Our results also contain information about one gene with unknown function (YJR138W). Experimental annotations for unknown genes are usually time consuming and hard to obtain. Thus, we see how computational techniques can speed up experimental ones.

Although consensus predictions of at least three independent classifiers are used, our results have to be interpreted carefully. Computational techniques are not a 100% accurate, so some follow up studies are still needed. For example, the description of YGL190C says it is non-essential but it was predicted as essential by all four classifiers.

Table 6.7: Newly predicted essential genes; the classifiers are KNN(K), SVM(S), Decision Tree(T), Random Forest(R)

Gene ID	**Classifiers**	**Description**
YAL044W-A	KTR	Putative DNA repair protein.
YBL071W-A	KSTR	Protein required for synthesis of diphthamide.
YCR042C	KSTR	Involved in RNA polymerase II transcription initiation.
YEL020W-A	KSTR	Mitochondrial intermembrane space protein.
YEL021W	KSTR	Catalyzes the sixth enzymatic step in the de novo biosynthesis of pyrimidines.
YER015W	KTR	Involved in the activation of endogenous pools of fatty acids.
YER029C	KSTR	Core Sm protein Sm B; part of heteroheptameric complex.
YER100W	STR	Ubiquitin-conjugating enzyme involved in ER-associated protein degradation.
YFL017W-A	KSTR	Core Sm protein Sm G; part of heteroheptameric complex.
YFL034C-B	KSTR	Component of the RAM signaling network.
YGL100W	KSTR	Nuclear pore protein.

Table 6.7: (continued)

Gene ID	Classifiers	Description
YGL106W	KSTR	Essential light chain for myosin Myo2p.
YGL119W	KSTR	Protein required for ubiquinone (coenzyme Q) biosynthesis and for respiratory growth.
YGL178W	KSR	Has roles in longevity, in maintenance of cell wall integrity, and in sensitivity to and recovery from pheromone arrest.
YGL183C	KSTR	Protein required for recombination and meiotic nuclear division.
YGL187C	KSR	Subunit of the terminal member of the mitochondrial inner membrane electron transport chain.
YGL190C	KSTR	Non-essential regulatory subunit B of protein phosphatase 2A.
YGL192W	KSTR	Required for IME1 transcript accumulation and for sporulation.
YHL004W	KSTR	Mitochondrial ribosomal protein.
YHR027C	KSTR	May participate in the recognition of several ligands of the proteasome; a site for protein-protein interactions.
YHR039C-A	KSTR	Involved in vacuolar acidification.
YHR052W	KSTR	Essential protein that interacts with proteasome components.
YHR072W-A	KSTR	Constituent of particles which are required for pseudouridylation and processing of pre-18S rRNA.
YHR089C	KSTR	Involved in the modification and cleavage of the 18S pre-rRNA.

Table 6.7: (continued)

Gene ID	Classifiers	Description
YHR098C	KSTR	Forms a complex that is involved in sorting of Pma1p into COPII vesicles.
YHR099W	KSTR	Interacts with acidic activators (e.g., Gal4p) which leads to transcription activation.
YHR102W	KSTR	Required for cell integrity possibly through regulating 1,6-beta-glucan levels in the wall.
YHR119W	KTR	Required in transcriptional silencing near telomeres and at the silent mating type loci.
YHR128W	KTR	Synthesizes UMP from uracil; involved in the pyrimidine salvage pathway.
YHR165C	KSTR	Involved in the second catalytic step of splicing.
YHR169W	KSTR	Involved in biogenesis of the 40S ribosomal subunit.
YHR187W	KST	Subunit of a histone acetyltransferase.
YJR132W	KSTR	A carrier protein involved in nuclear import of proteins.
YJR135W-A	KSTR	Mitochondrial intermembrane space protein mediating import and insertion of polytopic inner membrane proteins.
YJR138W	KSTR	Protein of unknown function.
YLR245C	KST	Catalyzes the modification of cytidine to uridine in vitro but native RNA substrates have not been identified.
YLR383W	KSTR	Protein involved in structural maintenance of chromosomes; required for interchromosomal and sister chromatid recombination.
YLR419W	KSR	Hypothetical protein.

Table 6.7: (continued)

Gene ID	**Classifiers**	**Description**
YNL048W	STR	Catalyzes addition of the terminal alpha 1,2-Man to the Man5GlcNAc2-PP-dolichol intermediate.
YOL142W	KSTR	Protein involved in rRNA processing.
YOL145C	KSTR	Component of a large complex that binds to and modulates the activity of RNA polymerase II; required for expression of a subset of genes.
YPR081C	KSR	Protein with sequence similarity to Grs1p, which is a glycyl-tRNA synthetase.

In addition, we also investigated genes causing morphological alterations. We used the dataset produced by Giaever *et al.* as training data for our predictions. We produced a total of 2126 new annotations. A summary of the results is shown in Table 6.8.

Table 6.8: Morphological Alteration Predictions

Mutant Phenotype	No. of genes
Clumped	292
Elongated	315
Large	287
Pointed	276
Round	368
Small	389
Other	199

Chapter 7

Related Work

Our current work is an extension the work carried out by the *Saccharomyces* Genome Deletion Project [WSAL99], which experimentally deleted yeast genes using a PCR-based gene deletion strategy, and thereby identifying many yeast essential genes. They also tried to assign functions to the genes by studying the mutants obtained from the deletion process. Giaever *et al.* [GCN+02] studied the mutant phenotypes to identify genes causing morphological alterations

Chen and Xu [CX05] did a study on protein dispensability in yeast in which it is represented as a fitness score that is measured by the growth rate of gene deletion mutants. Neural network and support vector machine were applied to predict protein dispensability through high-throughput data. In good concordance with our results, they found that a protein's dispensability shows significant correlations with its evolutionary rate, duplication rate, and connectivity in protein-protein interaction network and gene-expression correlation network.

López-Bigas and Ouzounis [LBO04] classified human disease genes using protein length, degree of paralogy and phylogenetic profile using nine different taxonomic groups: viruses, archaea, bacteria, protista, fungi, plants, invertebrates, vertebrates (excluding mammals) and mammals (excluding humans). They carefully selected these 11 features and did not use any feature selection. They conducted extensive statistical tests to test the differences in the distributions of each of the features between disease genes and other human genes. They achieved 68% accuracy using

the decision tree classifier.

Kuramochi and Karypis [KK01] did a feasibility study on using expression profiles for gene classification. They used a dataset consisting of 2467 yeast genes that had expression data available. They used 79 different measures but not all genes had the entire set of 79 measurements. They used zeroes for missing values. They tried to classify their dataset into some 249 different functional classes. They reported precision values between 26.0% and 41.6% and recall values between 37.0% and 61.3%.

John *et al.* [JKP94] looked at theoretical notions of relevance and irrelevance of features and evaluated feature selection with various artificial and real datasets. The datasets they used had 6–16 features and sample sizes up to 1024. They used two different algorithms that generate decision trees, ID3 and C4.5. They found that the error rates were similar after feature selection and concluded that the main advantage of the procedure is that smaller structures were created.

Koller and Sahami [KS96] examined a theoretically optimal but computationally intractable method for feature selection. Then they used forward selection and backward deletion heuristics to eliminate features that gave little or no additional information. First, they observed the effects on four datasets (both artificial and real) with features ranging from 6 to 180 in number and sample sizes ranging from 32 to 3200. In addition, they also constructed two datasets from the Reuters text dataset in which the features were presence or absence of words and classes were document topics. The two datasets, Reuters1 and Reuters2, had 1675 and 1646 features and 337 and 379 samples, respectively. After reduction the two datasets had 675 and 646 features, respectively. In all the six datasets, the feature reductions were achieved without no or little loss in classification accuracy. The classification algorithms they used were naïve bayes and C4.5.

Chapter 8

Conclusions and Future Work

We described a framework to predict essential genes in yeast. We used feature processing techniques to improve our starting dataset and evaluated four different classifiers. The feature combination and feature selection steps drastically reduced the number of features and the time taken by the classification algorithms, while maintaining good performance. The feature weighting step did not improve performance for our dataset. We found that gene conservation, degree of paralogy, number of protein-protein interactions and protein domains are important features for predicting essential genes. We produced predictions for genes that were not classified by the *Saccharomyces* Genome Deletion Project and created a list of 42 additional essential genes, including one gene with unknown function. We produced annotations for 2126 genes which possibly cause morphological alterations. We also made additions to the R framework, which help in various stages of the classification process.

Computational methods such as ours require training data and may require further experimental verification. However, they still provide significant advantage as they complement and speed up other approaches in many applications. Essential genes of other organisms could be predicted using similar features and the techniques described here. For example, more than 900 *E. coli* genes are unclassified [HIM+]. These techniques could also be used for other gene classification problems such as human disease gene prediction. We already created a dataset with similar features for the human gene set, which contains more than 22000 genes. The feature combination

technique could be applied to other binary features, like the presence or absence of words in text databases.

Bibliography

[BBD+00] A. Bateman, E. Birney, R. Durbin, S. R. Eddy, K. L. Howe, and E. L. Sonnhammer. The pfam protein families database. *Nucleic Acids Res.*, 28:263–266, 2000.

[BOKD+93] A. Baudin, O. Ozier-Kalogeropoulos, A. Denouel, F. Lacroute, and C. Cullin. A simple and efficient method for direct gene deletion in *Saccharomyces cerevisiae*. *Nucleic Acids Res*, 21(14):3329–3330, 1993.

[Bre01] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[CST00] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[CX05] Yu Chen and Dong Xu. Understanding protein dispensability through machine-learning analysis of high-throughput data. *Bioinformatics*, 21(5):575–581, March 2005.

[DHL+05] Evgenia Dimitriadou, Kurt Hornik, Friedrich Leisch, David Meyer, and Andreas Weingessel. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*, 2005. R package version 1.5-8.

[GCN+02] Guri Giaever, Angela M. Chu, Li Ni, Carla Connelly, and Linda Riles. Functional profiling of the *Saccharomyces cerevisiae* genome. *Nature*, 418:387–391, 2002.

[GGH+03] Elisabeth Gasteiger, Alexandre Gattiker, Christine Hoogland, Ivan Ivanyi, Ron D. Appel, and Amos Bairoch. ExPASy: the proteomics server for in-depth protein knowledge and analysis. *Nucleic Acids Res*, 31(13):3784–3788, 2003.

[Glo86] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.*, 13(5):533–549, 1986.

[HAC+05] T. Hubbard, D. Andrews, M. Caccamo, G. Cameron, Y. Chen, M. Clamp, L. Clarke, G. Coates, T. Cox, F. Cunningham, V. Curwen, T. Cutts, T. Down, R. Durbin, X. M. Fernandez-Suarez, J. Gilbert, M. Hammond, J. Herrero, H. Hotz, K. Howe, V. Iyer, K. Jekosch, A. Kahari, A. Kasprzyk, D. Keefe, S. Keenan, F. Kokocinsci, D. London, I. Longden, G. McVicker, C. Melsopp, P. Meidl, S. Potter, G. Proctor, M. Rae, D. Rios, M. Schuster, S. Searle, J. Severin, G. Slater, D. Smedley, J. Smith, W. Spooner, A. Stabenau, J. Stalker, R. Storey, S. Trevanion, A. Ureta-Vidal, J. Vogel, S. White, C. Woodwark, and E. Birney. Ensembl 2005. *Nucleic Acids Res*, 33(Database issue), January 2005.

[HIM+] Masayuki Hashimoto, Toshiharu Ichimura, Hiroshi Mizoguchi, Kimie Tanaka, Kazuyuki Fujimitsu, Kenji Keyamura, Tomotake Ote, Takehiro Yamakawa, Yukiko Yamazaki, Hideo Mori, Tsutomu Katayama, and Jun-Ichi Kato. Cell size and nucleoid organization of engineered escherichia coli cells with a reduced genome. *Molecular Microbiology*, 55(1):137+.

[JEA+03] P. Janssen, A. J. Enright, B. Audit, I. Cases, L. Goldovsky, N. Harte, V. Kunin, and C. A. Ouzounis. COmplete GENome Tracking (COGENT): a flexible data environment for computational genomics. *Bioinformatics*, 19(11):1451–1452, 2003.

[JKP94] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *International Conference on Machine*

Learning, pages 121–129, 1994. Journal version in AIJ, available at http://citeseer.nj.nec.com/13663.html.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[KJ98] R. Kohavi and G. John. The wrapper approach, 1998.

[KK01] Michihiro Kuramochi and George Karypis. Gene classification using expression profiles: A feasibility study. In *IEEE International Conference on Bioinformatics and Biomedical Egineering*, pages 191–200, 2001.

[Koh95] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, pages 1137–1145, 1995.

[KS96] Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *International Conference on Machine Learning*, pages 284–292, 1996.

[LBO04] N. López-Bigas and C. A. Ouzounis. Genome-wide identification of genes likely to be involved in human genetic disease. *Nucleic Acids Res*, 32(10):3108–3114, 2004.

[LSK+02] David J. Lipman, Alexander Souvorov, Eugene V. Koonin, Anna R. Panchenko, and Tatiana A. Tatusova. The relationship of protein conservation and sequence length. *BMC Evol Biol*, 2(20), 2002.

[LW02] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[PATO03] José Manuel Peregrin-Alvarez, Sophia Tsoka, and Christos A. Ouzounis. The phylogenetic extent of metabolic enzymes and pathways. *Genome Research*, 13(3):422–427, 2003.

[PMT+99] Matteo Pellegrini, Edward M. Marcotte, Michael J. Thompson, David Eisenberg, and Todd O. Yeates. Assigning protein functions by comparative genome analysis: Protein phylogenetic profiles. *Proc Natl Acad Sci U S A*, 96(8):4285–4288, 1999.

[R D04] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2004. ISBN 3-900051-07-0.

[Rip05] Brian Ripley. *tree: Classification and regression trees*, 2005. R package version 1.0-19.

[SH] Soma Saha and Steffen Heber. *In Silico* prediction of yeast deletion phenotypes. *Genetic and Molecular Research*. Accepted.

[SSBL05] Tobias Sing, Oliver Sander, Niko Beerenwinkel, and Thomas Lengauer. *ROCR: Visualizing the performance of scoring classifiers.*, 2005. R package version 1.0-1.

[Swe88] J. A. Swets. Measuring the accuracy of diagnostic systems. *Science*, 240(4857):1285–1293, 1988.

[Vap99] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1999.

[VR02] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. ISBN 0-387-95457-0.

[WBPP94] A. Wach, A. Brachat, R. Pohlmann, and P. Philippsen. New heterologous modules for classical or PCR-based gene disruptions in *Saccharomyces cerevisiae*. *Yeast*, 10(13):1793–1808, 1994.

[WSAL99] Elizabeth A. Winzeler, Daniel D. Shoemaker, Anna Astromoff, and Hong Liang. Functional characterization of the *S. cerevisiae* genome by gene deletion and parallel analysis. *Science*, 285(5429):901–906, 1999.

[XSD+02] Ioannis Xenarios, Lukasz Salwínski, Xiaoqun Joyce Duan, Patrick Higney, Sul-Min Kim, and David Eisenberg. DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Res*, 30(1):303–305, 2002.