CalPred: A TOOL FOR THE PREDICTION OF EF-HAND CALCIUM-BINDING PROTEINS AND IDENTIFICATION OF CALCIUM-BINDING REGIONS USING MACHINE LEARNING TECHNIQUES

By

CHANDAN KUMAR-031507
KUNAL JAISWAL-031529

MAY-2007

DEPARTMENT OF BIOTECHNOLOGY & BIOINFORMATICS
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY-
WAKNAGHAT

CERTIFICATE

This is to certify that the work entitled, "**CalPred: A tool for the prediction of EF-hand calcium-binding proteins and identification of calcium-binding regions using machine learning techniques**" submitted by **Chandan Kumar** and **Kunal Jaiswal** in partial fulfillment for the award of degree of Bachelor of Technology in Bioinformatics of Jaypee University of Information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Naik

Dr. Pradeep Kumar Naik

(Project Coordinator)
Senior Lecturer
Department of Bioinformatics
Jaypee University of Information Technology
Waknaghat

ACKNOWLEDGMENT

Many people have contributed to this project in a variety of ways over the past few months. To the individuals who have helped us, we again express our appreciation. We also acknowledge the many helpful comments Received from our teachers of the bioinformatics department. We are indebted to all those who provided reviews & suggestions for improving the results and the topics covered in our project, and we extend our apologies to anyone we may have failed to mention.

Kunal Jaiswal Chandan

Kunal Jaiswal Chandan Kumar

CONTENTS

CERTIFICATE.. 2

ACKNOWLEDGMENT.. 3

LIST OF FIGURES... 6

ABBREVIATIONS..6

ABSTRACT... 7

CHAPTER 1: INTRODUCTION.. 8 - 26

DISCOVERY OF CALCIUM BINDING PROTEINS.. 8 - 9

EF-HAND CALCIUM BINDING PROTEINS... 9

STRUCTURAL FEATURES OF EF-HAND CA2+ BINDING PROTEINS.................. 10-11

MACHINE LEARNING TECHNIQUES.. 12 -22

NEURAL NETWORKS.. 12 - 16

SUPPORT VECTOR MACHINE... 16 – 22

IN SILICO PREDICTION OF PROTEIN MOTIFS.. 22 – 23

BASIC APPROACHES OF PREDICTION OF CA2+ BINDING MOTIFS..................... 23

REFERENCES... 23 – 26

OBJECTIVES.. 27

CHAPTER 2: CalPred: A TOOL FOR THE PREDICTION OF EF-HAND CALCIUM-BINDING PROTEINS AND IDENTIFICATION OF CALCIUM-BINDING REGIONS USING MACHINE LEARNING TECHNIQUES... 28 - 50

MOTIVATION... 29

INTRODUCTION...29 - 30

MATERIALS AND METHODS.. 30 - 35

RESULTS AND DISCUSSION.. 36 - 40

CONCLUSIONS... 40

ACKNOWLEDGEMENTS.. 40

REFERENCES...41

SCREENSHOTS.. 42 - 50

APPENDIX – I 51- 79

APPENDIX – II .... 80 - 81

APPENDIX – III 82

APPENDIX – IV .... 83 - 91

APPENDIX – V .... 92 - 100

APPENDIX – VI .... 101 - 102

BIBLIOGRAPHY 103

LIST OF FIGURES

Figure 1: EF-hand Calium-binding proteins. (a) Ribbon helix-loop-helix structure of EF-hand calcium binding proteins. (b) The structural display of EF-hand calcium binding protein. The calcium ion is bounded to one of the motifs in the protein through six oxygen atoms, one from each side chains of: Asp (D) 9, Asn (N) 11, Asp (D) 13, one from the main chain residue 15, and 2 from the side chain of Glu (E) 20.

Figure 2: Typical Artificial Neural Network setup.

Figure 3: Model showing different modules and their organization into the two layers.

Figure 4: Distribution of output values of *Homo sapiens* proteome for ANN*pepstats* module.

Figure5-6: Distribution of output values of *Homo sapiens* proteome for SVM*pepstats_linear* and SVM*pepstats_polynomial* modules.

Figure7: Distribution of output values of *Homo sapiens* proteome for SVM*pepstats_radial_basis* module

LIST OF ABBREVIATIONS

SVM:	Support Vector Machine.
ANN:	Artificial Neural Network.
EDTA:	Ethylenediaminetetraacetic acid
ER_SR:	Endo (sarco) plasmic reticulum.
InsP3:	Inositol 1-4-5 trisphosphate.
cADPr:	Cyclic ADP ribose.
CaM:	Calmodulin
CaBPs:	Ca2+-binding proteins.
GBP:	Galactose-binding protein.
PSSM:	Position Specific Scoring Matrix.

ABSTRACT

Motivation: Predicting calcium binding proteins and identifying calcium-binding sites is an important problem in the field of proteomics.Most of the currently used methods employs structural protein data to predict calcium-binding sites. Here we present a method developed to predict calcium-binding proteins and identify calcium-binding sites from protein sequence data using machine learning techniques such as neural networks and support vector machines.

Results: We have developed the novel application *CalPred*, having nine implemented algorithms divided into two filters. The first filter predicts proteins as a whole i.e. whether they are calcium-binding proteins or not, and the second filter predicts the specific calcium binding sites in the proteins which have passed through the first filter. The tool was able to pick sequences that were calcium-binding in nature but were not picked up by pattern sited as calcium-binding domain in PROSITE database. We also scanned four whole proteomes for potential calcium binding proteins.

Availability: The CalPred tool is available for free use to noncommercial users and can be downloaded to be used in-house as a stand alone server from following links.

http://www.juit.ac.in/calpred/index.html.

http://www.bioinformatics.org/calpred/

CHAPTER 1

INTRODUCTION

Calcium-binding proteins are proteins that participate in calcium cell signalling pathways by binding to Ca2+.The most ubiquitous Ca^{2+}-sensing protein, found in all eukaryotic organisms including yeasts, is calmodulin. Intracellular storage and release of Ca^{2+} from the sarcoplasmic reticulum is associated with the high-capacity, low-affinity calcium-binding protein, calsequestrin.With their role in signal transduction, calcium-binding proteins contribute to all aspects of the cell's functioning, from homeostasis to learning and memory. For example, the neuron-specific calexcitin has been found to have an excitatory effect on neurons, and interacts with proteins that control the firing state of neurons, such as the voltage-dependent potassium channel.

Discovery of Calcium Binding Proteins

The calcium binding proteins were discovered abruptly at the end of the 1950s, thanks to some seminal discoveries that paved the way for the acceptance of what is now called the calcium binding proteins. One was the demonstration by Weber (1) that the binding of Ca2+ to myofibrils activated actomyosin. Ebashi and Kodama (2) discovered troponin in crude tropomyosin preparations, followed by the demonstration that one of its subunits, troponin C, was the Ca2+ receptor that mediated myofibrillar contraction. Synthesis of EDTA and the characterization of its Ca2+-chelating properties by G. Schwarzenbachand Ackermann (3) provide another milestone in discovery of Calcium binding proteins. It was reported that removal of Ca2+ by the chelator relaxed muscle fibers.This reveled the signaling role of Ca2+ binding proteins. A breakthrough in this direction was the solution of the crystal structure of parvalbumin by Kretsinger (4) in 1972. Ca2+ binding protein belongs to a family of proteins known as EF hand proteins, which has now grown to nearly 600 members. EF hand proteins do buffer Ca2+ but also play another important role. They decode the information carried by Ca2+ and pass it on to targets. They do so by changing conformation after binding Ca2+ and after interacting with targets.

Essentially, EF hand proteins are more hydrophobic on the surface after complexing Ca2+, approach the target, and collapse around its binding domain. Thus, these proteins are better defined as Ca2+ modulated proteins, or Ca2+ sensors. Other proteins also decipher Ca2+ signals, e.g., the annexins, gelsolin, and proteins containing C2 domains, but the EF hand proteins are the most important. They may function as a committed separate subunit of a single (enzyme) protein or as a subunit that associates reversibly with different proteins (e.g., Cam). They may even be an integral portion of the sequence of enzymes (e.g., calpain) The control of Ca2+ concentration in the cytoplasm and organelles is instead the sole role of proteins that, as a rule, are intrinsic to the plasma membrane and to the membranes of organelles and transport Ca2+ across them. These proteins belong to various classes: Ca2+ channels in the plasma membrane are gated by voltage, by ligands, or by the emptying of internal Ca2+ stores. In the endo (sarco) plasmic reticulum (ER_SR), they are instead activated by the second ''messengers,'' inositol 1-4-5 trisphosphate (InsP3) and cyclic ADP ribose.

EF-hand Calcium Binding Proteins

The EF-hand proteins with a helix-loop-helix Ca2+ binding motif are one of the largest protein families and are involved in numerous biological processes. Ca2+, a second messenger in cellular signal transduction, functions as a pivotal regulator of the cellular life cycle including cell division, differentiation and apoptosis. (5-9) The regulatory effects of Ca2+ are influenced by the oscillation of intracellular Ca2+ concentration, which ranges from submicromolar to millimolar levels.(10) Ca2+ carries out its functions by binding to specific Ca2+ receptors or Ca2+-binding proteins (CaBPs). According to the role Ca2+ ions or the proteins play in the biological context, most Ca2+ binding proteins may fall into one of three categories: trigger or sensor proteins (e.g., calmodulin), (11) buffer proteins (e.g., S100G and parvalbumin),(12) or Ca2+-stabilized proteins (e.g., thermolysin).(13) The EF-hand proteins can be found in each category and constitute more than 50% of all well-characterized Ca2+-binding proteins. The EF-hand moiety is one of the most frequently used motifs in eukaryotic systems. (14)

Structural features of EF-hand Ca2+ binding proteins

Since the delineation of the EF-hand motif in 1973, the family of EF-hand proteins has expanded to include at least 66 subfamilies so far.(15 - 17) EF-hand motifs are divided into two major groups: the canonical EF-hands as seen in calmodulin (CaM) and the prokaryotic CaM-like protein calerythrin and the pseudo EF-hands exclusively found in the Ntermini of S100. and S100-like proteins. The major difference between these two groups lies in the Ca2+-binding loop: the 12 residue canonical EF-hand loop binds Ca2+ mainly via sidechain carboxylates or carbonyls (loop sequence positions 1, 3, 5, 12), whereas the 14-residue pseudo EF-hand loop chelates Ca2+ primarily via backbone carbonyls (positions 1, 4, 6, 9). The residue at the X axis coordinates the Ca2+ ion through a bridged water molecule. The EF-hand loop has a bidentate ligand (Glu or Asp) at Z axis. Among all the structures reported to date, the majority of EF-hand motifs are paired either between two canonical or one pseudo and one canonical motifs. For proteins with odd numbers of EF-hands, such as the penta-EF-hand calpain, EF-hand motifs were coupled through homo- or hetero-dimerization.(18-22) Recently, EF-hand-like proteins with diversified flanking structural elements around the Ca2+ binding loop have been reported in bacteria.(23-26) Several lines of evidence indicate that these prokaryotic EF-hand-like proteins are widely implicated in Ca2+ signaling and homeostasis in bacteria.(24,26-29) They contain flexible lengths of Ca2+-binding loops that differ from the EF-hand motifs. However, their coordination properties resemble classical EF-hand motifs. For example, the semi-continuous Ca2+-binding site in D-galactose-binding protein (GBP) contains a nine-residue loop (a.a.134-142). The Ca2+ ion is coordinated by seven protein oxygen atoms, five of which are from the loop mimicking the canonical EF loop whereas the other two are from the carboxylate group of a distant Glu (a.a. 205). Another example is a novel domain named Excalibur (extracellular Ca2+ binding region) isolated from *Bacillus subtilis*. This domain has a conserved 10-residue Ca2+-binding loop strikingly similar to the canonical 12-residue EF-hand loop.The diversity of the structure of the flanking region is illustrated by the discovery of EF-hand-like domains in bacterial proteins. For example, a helix-loop-strand instead of the helix-loop-helix structure is in periplasmic galactose-binding protein (S*almonella typhimurium,* 1gcg) or alginate-binding protein (*Sphingomonas sp*., 1kwh) ;(29) the entering helix is missing in protective antigen (*Bacillus anthracis*, 1acc) (30) or dockerin

(*Clostridium thermocellum*, 1daq).(31) With more and more EF-hand Ca2+ binding proteins being discovered and characterized in bacteria, archaea, and eukaryotes, structural and functional knowledge of the EF-hand proteins has expanded steadily in recent years. The EF-hand-like proteins contain Ca2+-binding sequences that closely resemble the canonical EF-hand motif yet with diversified flanking structural elements. The general structure of EF-Ca2+ binding protein is given below.

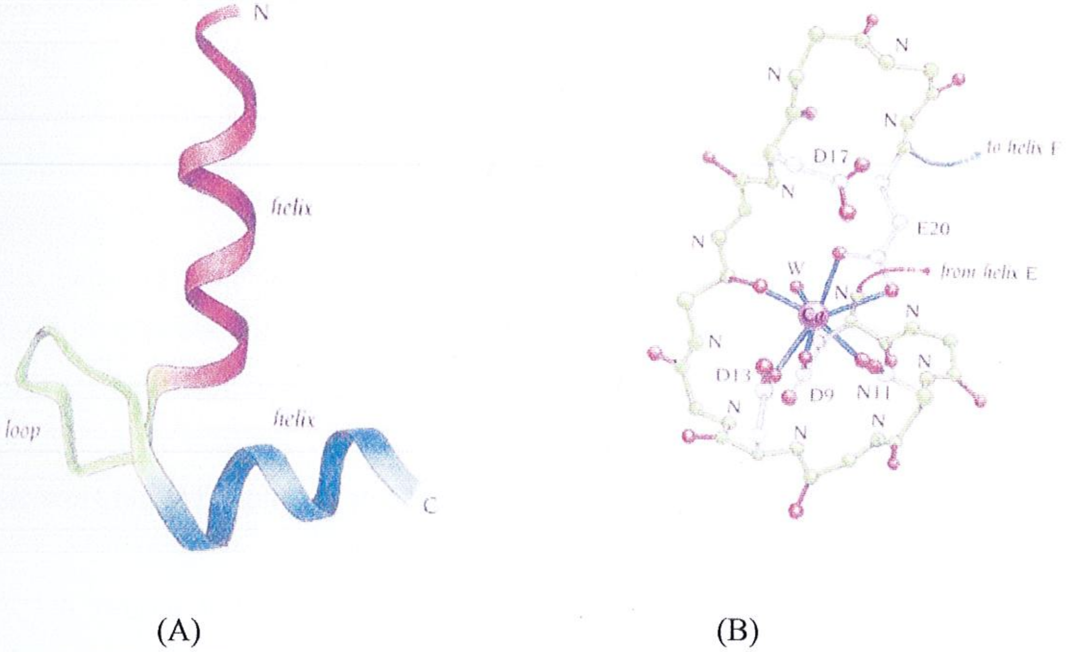


Figure 1: EF-hand Calium-binding proteins. (a) Ribbon helix-loop-helix structure of EF-hand calcium binding proteins. (b) The structural display of EF-hand calcium binding protein. The calcium ion is bounded to one of the motifs in the protein through six oxygen atoms, one from each side chains of: Asp (D) 9, Asn (N) 11, Asp (D) 13, one from the main chain residue 15, and 2 from the side chain of Glu (E) 20.

MACHINE LEARNING TECHNIQUES

NEURAL NETWORKS:

Neural Network or more appropriately Artificial Neural Network is basically a mathematical model of what goes in our mind (or brain). The brain of all the advanced living creatures consists of neurons, a basic cell, which when interconnected produces what we call Neural Network. The sole purpose of a Neuron is to receive electrical signals, accumulate them and see further if they are strong enough to pass forward. The basic functionality lies not in neurons but the complex pattern in which they are interconnected. NNs are just like a game of chess, easy to learn but hard to master. In the same way, a single neuron is useless. Well, practically useless. It is the complex connection between them and values attached with them which makes brains capable of thinking and having a sense of consciousness (much debated).

Basic working principle of a neuron

A neuron is basically a cell which accumulates electrical signals with different strengths. What it does more is that it compares the accumulated signal with one predefined value unique to every neuron. This value is called bias. Function of a neuron could be explained in the following diagram.

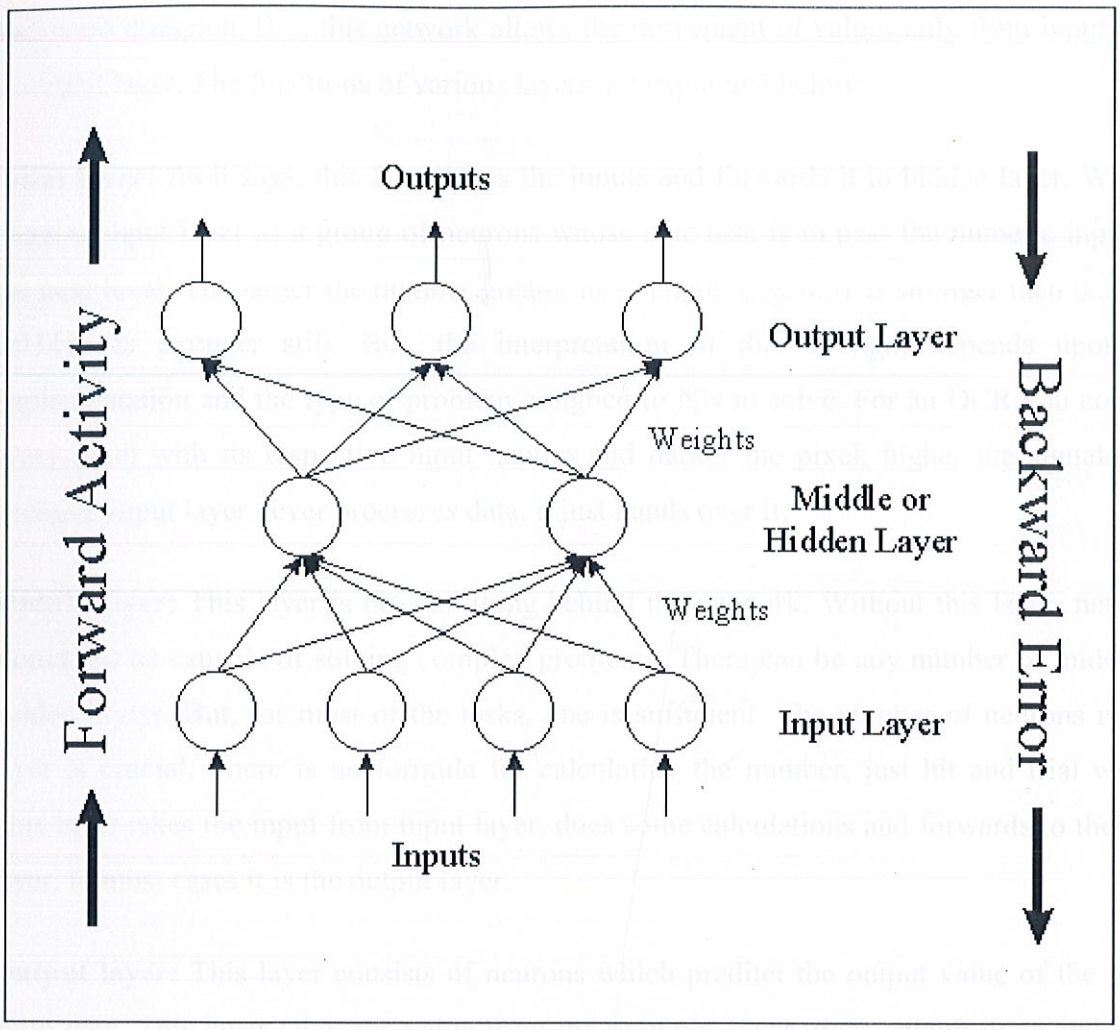


FIGURE 4.1. Typical Artificial Neural Network Setup (Caudill and Butler, 1992a).

Image Source: http://www.interwet.psu.edu/f41.gif

The circles in the image represent neurons. This network or more appropriately this network topology is called feed-forward multi layered neural network. It is the most basic and most widely used network. The network is called multi layered because it consists of more than two layers. The neurons are arranged in a number of layers, generally three. They are input, hidden/middle and output layers. The names signify the function of the layer. This network is feed-forward, means the values are propagated in one direction only. There are many other topologies in which values can be looped or move in both forward and backward direction. But, this network allows the movement of values only from input layer to output layer. The functions of various layers are explained below:

Input layer: As it says, this layer takes the inputs and forwards it to hidden layer. We can imagine input layer as a group of neurons whose sole task is to pass the numeric inputs to the next level. The larger the number greater its strength. E.g. 0.51 is stronger than 0.39 but 0.93412 is stronger still. But, the interpretation of this strength depends upon the implementation and the type of problem assigned to NN to solve. For an OCR you connect every pixel with its respective input neuron and darker the pixel, higher the signal/input strength. Input layer never processes data, it just hands over it.

Middle layer: This layer is the real thing behind the network. Without this layer, network would not be capable of solving complex problems. There can be any number or middle or hidden layers. But, for most of the tasks, one is sufficient. The number of neurons in this layer is crucial. There is no formula for calculating the number, just hit and trial works. This layer takes the input from input layer, does some calculations and forwards to the next layer, in most cases it is the output layer.

Output layer: This layer consists of neurons which preditct the output value of the given input data. This layer takes the value from the previous layer, **does calculations** and gives the final result. Basically, this layer is just like hidden layer but instead of passing values to the next layer, the values are treated as output.

Dendrites: These are straight lines joining two neurons of consecutive layers. They are just a passage (or method) through which values are passed from one layer to the next. There is a value attached with dendrite called **weight**. The weight associated with dendrites basically determines the importance of incoming value. A weight with larger value determines that the value from that particular neuron is of higher significance. To achieve this we do is multiply the incoming value with weight. So no matter how high the value is, if the weight is low the multiplication yields the final low value.

Training: Training is the most important part of a neural network and the one consisting of the most mathematics. It uses Backpropagation method for training the NN. The best example illustrating this principle is Charles Darwin(what?). Yes, at the time when he wrote '*On the Origin of Species*', DNA was not known. So, he propounded the evolution without even knowing the method of how it is done i.e. how traits are passed on from parents to offspring. Training a neural network model essentially means selecting one model from the set of allowed models (or, in a Bayesian framework, determining a distribution over the set of allowed models) that minimises the cost criterion. There are numerous algorithms available for training neural network models; most of them can be viewed as a straightforward application of optimization theory and statistical estimation. Most of the algorithms used in training artificial neural networks are employing some form of gradient descent. This is done by simply taking the derivative of the cost function with respect to the network parameters and then changing those parameters in a gradient-related direction. Evolutionary methods, simulated annealing, and expectation-maximization and non-parametric methods are among other commonly used methods for training neural networks. This training procedure must be repeated for larger number of samples so that the NN can produce accurate results for untrained input samples.

Applications: The utility of artificial neural network models lies in the fact that they can be used to infer a function from observations. This is particularly useful in applications where the complexity of the data or task makes the design of such a function by hand impractical.

Real life applications

The tasks to which artificial neural networks are applied tend to fall within the following broad categories:

- Function approximation, or regression analysis, including time series prediction and modeling.
- Classification, including pattern and sequence recognition, novelty detection and sequential decision making.
- Data processing, including filtering, clustering, blind source separation and compression.

Application areas include system identification and control (vehicle control, process control), game-playing and decision making (backgammon, chess, racing), pattern recognition (radar systems, face identification, object recognition and more), sequence recognition (gesture, speech, handwritten text recognition), medical diagnosis, financial applications, data mining (or knowledge discovery in databases, "KDD"), visualization and e-mail spam filtering.

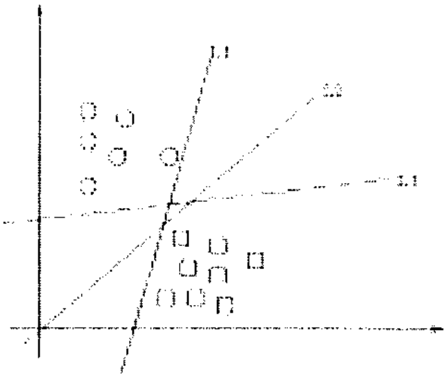
SUPPORT VECTOR MACHINE

Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. They belong to a family of generalized linear classifiers. They can also be considered a special case of Tikhonov regularization. A special property of SVMs is that they simultaneously minimize the empirical classification error and maximize the geometric margin; hence they are also known as maximum margin classifiers.

Support vector machines map input vectors to a higher dimensional space where a maximal separating hyperplane is constructed. Two parallel hyperplanes are constructed on each side of the hyperplane that separates the data. The separating hyperplane is the hyperplane that maximises the distance between the two parallel hyperplanes. An assumption is made that the larger the margin or distance between these parallel hyperplanes the better the generalisation error of the classifier will be.

Motivation

Often we are interested in classifying data as a part of a machine-learning process. This data will be represented as p-dimensional vector (a list of p numbers). We are interested in whether we can separate them with p-1 dimensional hyperplane. This is a typical form of linear classifier. There are many linear classifiers that might satisfy this property. However, we are additionally interested in finding out if we can achieve maximum separation (margin) between the two classes. By this we mean that we pick the hyperplane so that the distance from the hyperplane to the nearest data point is maximized. That is to say that the nearest distance between a point in one *separated* hyperplane and a point in the other *separated* hyperplane is maximized. Now, if such a hyperplane exists, it is clearly of interest and is known as the *maximum-margin hyperplane* and such a linear classifier is known as a maximum margin classifie



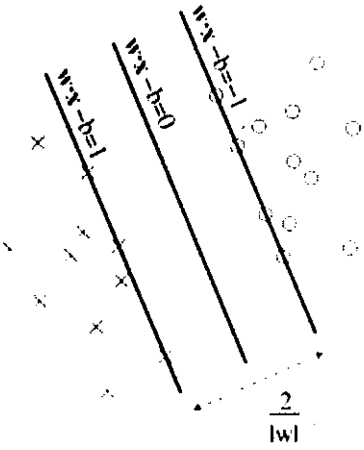
Many linear classifiers (hyperplanes) separate the data. However, only one achieves maximum separation

Classification scheme

We consider data points of the form:

$$\{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \ldots, (\mathbf{x}_n, c_n)\}$$

where the c_i is either 1 or −1, a constant denoting the class to which the point $\mathbf{x}_i$ belongs. Each $\mathbf{x}_i$ is a p-dimensional real vector, usually of normalised (Normalizing constant) [0, 1] or [-1, 1] values. The scaling is important to guard against variables (attributes) with larger variance that might otherwise dominate the classification. We can view this as *training data*, which denotes the correct classification which we would like the SVM to eventually distinguish, by means of the dividing (or separating) hyperplane, which takes the form



Maximum-margin hyperplanes for a SVM trained with samples from two classes. Samples along the hyperplanes are called the support vectors.

$$\mathbf{w} \cdot \mathbf{x} - b = 0.$$

The vector **w**points perpendicular to the separating hyperplane. Adding the offset parameter *b* allows us to increase the margin. In its absence, the hyperplane is forced to pass through the origin, restricting the solution.

As we are interested in the maximum margin, we are interested in the support vectors and the parallel hyperplanes (to the optimal hyperplane) closest to these support vectors in either class. It can be shown that these parallel hyperplanes can be described by equations

$$\mathbf{w} \cdot \mathbf{x} - b = 1,$$
$$\mathbf{w} \cdot \mathbf{x} - b = -1.$$

If the training data are linearly separable, we can select these hyperplanes so that there are no points between them and then try to maximize their distance. By using geometry, we find the distance between the hyperplanes is 2/|*w*|, so we want to minimize |*w*|. To exclude data points, we need to ensure that for all *i* either

$$\mathbf{w} \cdot \mathbf{x_i} - b \geq 1 \qquad \text{or}$$
$$\mathbf{w} \cdot \mathbf{x_i} - b \leq -1$$

This can be rewritten as:

$$c_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1, \quad 1 \leq i \leq n. \qquad (1)$$

Primal Form

The problem now is to minimize $|w|$ subject to the constraint (1). This is a quadratic programming (QP) optimization problem. More clearly,

minimize $(1/2)||\mathbf{w}||^2$**, subject to** $c_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1, \quad 1 \leq i \leq n.$.

The factor of 1/2 is used for mathematical convenience.

Dual Form

Writing the classification rule in its dual form reveals that classification is only a function of the *support vectors*, i.e., the training data that lie on the margin. The dual of the SVM can be shown to be:

$\max \sum_{i=1}^{n} \alpha_i - \sum_{i,j} \alpha_i \alpha_j c_i c_j \mathbf{x}_i^T \mathbf{x}_j$ **subject to** $\alpha_i \geq 0$,

where the α terms constitute a dual representation for the weight vector in terms of the training set:

$$\mathbf{w} = \sum_i \alpha_i c_i \mathbf{x}_i$$

Soft margin

In 1995, Corinna Cortes and Vladimir Vapnik suggested a modified maximum margin idea that allows for mislabeled examples. If there exists no hyperplane that can split the "yes" and "no" examples, the *Soft Margin* method will choose a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples. This work popularized the expression *Support Vector Machine* or *SVM*. The method introduces slack variables, ξ_i, which measure the degree of misclassification of the datum x_i

$$c_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1 - \xi_i \quad 1 \leq i \leq n \qquad (2).$$

The objective function is then increased by a function which penalises non-zero ξ_i, and the optimisation becomes a trade off between a large margin, and a small error penalty. If the penalty function is linear, the equation (3) now transforms to

$$\min ||\mathbf{w}||^2 + C\sum_i \xi_i \quad \textbf{such that } c_i(\mathbf{w}\cdot\mathbf{x_i} - b) \geq 1 - \xi_i \quad 1 \leq i \leq n$$

This constraint in (2) along with the objective of minimizing $|w|$ can be solved using Lagrange multipliers. The key advantage of a linear penalty function is that the slack variables vanish from the dual problem, with the constant *C* appearing only as an additional constraint on the Lagrange multipliers. Non-linear penalty functions have been used, particularly to reduce the effect of outliers on the classifier, but unless care is taken, the problem becomes non-convex, and thus it is considerably more difficult to find a global solution

Non-linear classification

The original optimal hyperplane algorithm proposed by Vladimir Vapnik in 1963 was a linear classifier. However, in 1992, Bernhard Boser, Isabelle Guyon and Vapnik suggested a way to create non-linear classifiers by applying the kernel trick (originally proposed by Aizerman) to maximum-margin hyperplanes. The resulting algorithm is formally similar, except that every dot product is replaced by a non-linear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in the transformed feature space. The transformation may be non-linear and the transformed space high dimensional; thus though the classifier is a hyperplane in the high-dimensional feature space it may be non-linear in the original input space.

If the kernel used is a Gaussian radial basis function, the corresponding feature space is a Hilbert space of infinite dimension. Maximum margin classifiers are well regularized, so the infinite dimension does not spoil the results. Some common kernels include,

- Polynomial (homogeneous): $k(\mathbf{x},\mathbf{x}') = (\mathbf{x}\cdot\mathbf{x}')^d$
- Polynomial (inhomogeneous): $k(\mathbf{x},\mathbf{x}') = (\mathbf{x}\cdot\mathbf{x}' + 1)^d$

- Radial Basis Function: $k(\mathbf{x},\mathbf{x}') = \exp(-\gamma \|\mathbf{x}-\mathbf{x}'\|^2)$, for $\gamma > 0$
- Gaussian Radial basis function: $k(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$
- Sigmoid: $k(\mathbf{x},\mathbf{x}') = \tanh(\kappa \mathbf{x}\cdot\mathbf{x}' + c)$, for some (not every) $\kappa > 0$ and $c < 0$

Regression

A version of a SVM for regression was proposed in 1996 by Vladimir Vapnik, Harris Drucker, Chris Burges, Linda Kaufman and Alex Smola. This method is called support vector regression (SVR). The model produced by support vector classification (as described above) only depends on a subset of the training data, because the cost function for building the model does not care about training points that lie beyond the margin. Analogously, the model produced by SVR only depends on a subset of the training data, because the cost function for building the model ignores any training data that are close (within a threshold ϵ) to the model prediction.

Properties of SVM

- Flexibility in choosing a similarity function
- Sparseness of solution when dealing with large data sets only support vectors are used to specify the separating hyperplane
- Ability to handle large feature spaces complexity does not depend on the dimensionality of the feature space
- Overfitting can be controlled by soft margin approach
- Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution
- Feature Selection

SVM Applications

- SVM has been used successfully in many real-world problems
  - Text (and hypertext) categorization
  - Image classification

- Bioinformatics (Protein classification, Cancer classification)
- Hand-written character recognition

In Silico prediction of protein motifs

Protein homology detection is one of the most important problems in computational biology. Homology is generally established by sequence similarity. Many methods for measuring similarity have been studied in the past two decades. The two most established methods are the Smith-Waterman algorithm (Smith and Waterman, 1981) and its heuristic, faster counterpart, BLAST (Altschul *et al.*, 1997). Protein sequence motifs are an alternative way of detecting sequence similarity. Motifs are usually constructed from multiple sequence alignments of related sequences. As a preliminary step, one extracts from an alignment 'blocks' which are ungapped regions of high sequence similarity. The blocks are then described either by Position Specific Scoring Matrices (PSSMs), which indicate the relative abundance of each amino acid at each position in the block, or by discrete sequence motifs, which indicate the possible amino acids at each position.

By focusing on limited, highly conserved regions of proteins, motifs can often reveal important clues to a protein's role even if it is not globally similar to any known protein (Nevill-Manning *et al.*, 1998). The motifs for most catalytic sites and binding sites are conserved over much wider taxonomic distances and evolutionary time than are the sequences of the proteins themselves. Thus, motifs often represent functionally important regions such as catalytic sites, binding sites, protein-protein interaction sites, and structural motifs.

The first database of protein motifs was the PROSITE database (Falquet *et al.*, 2002), whose discrete motifs are manually constructed. Other databases derived from multiple sequence alignments of protein families include BLOCKs+, PRINTs, pFAM, ProDom, DOMO, and InterPro; see Henikoff *et al.* (1999) and references therein. The BLOCKs+ database combines these databases (Henikoff *et al.*, 1999); the eMOTIF database contains discrete sequence motifs constructed from the blocks of BLOCKS+ (Huang and Brutlag, 2001). Unlike the PROSITE and PRINTS databases, eMOTIFs are constructed automatically in a principled approach that has been shown to perform well (Nevill-Manning *et al.*, 1998). More recently, the eBLOCKS database of blocks was constructed (Su *et al.*, 2003). It presents a systematic approach for creating blocks using groups of proteins constructed using clustered PSI-BLAST results. It is more comprehensive than BLOCKS+, and motifs constructed from it have better coverage than BLOCKS+ motifs. This paper uses discrete sequence motifs extracted from the eBLOCKs database using the eMOTIF method.

Basic approaches of prediction of Ca2+ binding motifs

In this paper we introduce a sequence similarity measure based on the motif content of a pair of sequences. A simple way to use a sequence similarity to annotate a novel protein is by using sequence derived features: decide on the annotation according to the annotations of the nearest neighbor(s). These sequence derived features are feeded into the machine learning tools like ANN and SVM in order to measeure the sequence similarity based on their learning experience. When a sequence similarity can be shown to be a dot product in some space, it is called a *kernel*. This is important since many successful machine learning methods such as SVMs are defined in terms of a kernel. In this paper we use protein motifs to construct a kernel that can be computed efficiently; we show that this kernel performs significantly better than a kernel based on BLAST or Smith-Waterman scores.We tested the methods on two tasks: prediction of Ca2+ binding proteins and motifs in a number of organism (proteome) as well as validation using prosite database. We find that good performance is obtained when using a combination of a good similarity measure (kernel), and a state of the art classifier: the performance motif-based kernel is significantly improved when coupled to an SVM classifier with ANN.

References

- Weber A. (1959) *J. Biol. Chem.* 234, 2764–2769.[1]
- Ebashi, S. & Kodama, A. (1965) *J. Biochem. (Tokyo)* 58, 107–108. [2]
- Schwarzenbach, v. G. & Ackermann, H. (1954) *Helv. Chim. Acta* 30, 1798–1804. [3]
- Kretsinger, R. H. (1972) *Nat. New Biol.* 240, 85–88.[4]

- Ernesto Carafoli, *PNAS 2002;* 99; 1115-1122; doi:10.1073/pnas.032427999.[5]
- Ermak G, Davies KJ. Mol Immunol 2002; 38(10):713-721.[6]
- Orrenius S, Zhivotovsky B, Nicotera P. Nat Rev Mol Cell Biol 2003; 4(7):552-565.[7]
- Dowd DR. Res 1995; 30:255-280.[8]
- Means AR, Rasmussen CD. Cell Calcium 1988; 9(5-6):313-319.[9]
- Tsien RW, Tsien RY. Annu Rev Cell Biol 1990;6:715-760.[10]
- Levine BA, Dalgarno DC, Esnouf MP, Klevit RE, Scott GM, Williams RJ.Ciba Found Symp 1983; 93:72- 97.[11]
- Schroder B, Schlumbohm C, Kaune R, Breves G. J Physiol 1996;492 (Pt 3):715-722.[12]
- Buchanan JD, Corbett RJ, Roche RS. Biophys Chem 1986;23(3-4):183-199.[13]
- Henikoff S, Greene EA, Pietrokovski S, Bork P, Attwood TK, Hood L. Science 1997; 278(5338):609-614. [14]
- Kawasaki H, Nakayama S, Kretsinger RH. Biometals 1998;11(4):277-295.[15]
- Kretsinger RH, Nockolds CE. J Biol Chem 1973;248(9):3313-3326.[16]
- Ravasi T, Hsu K, Goyette J, Schroder K, Yang Z, Rahimi F, Miranda LP, Alewood PF, Hume DA, Geczy C. Genomics 2004; 84(1):10-22.[17]
- Pal GP, Elce JS, Jia Z. J Biol Chem 2001;276(50):47233-47238.[18]
- Raser KJ, Buroker-Kilgore M, Wang KK. Biochim Biophys Acta 1996;1292(1):9-14.[19]
- 21. Ravulapalli R, Diaz BG, Campbell RL, Davies PL.Biochem J 2005;388(Pt 2):585- 591.[20]
- Hunter MJ, Chazin WJ. J Biol Chem 1998; 273(20):12427-12435.[21]
- Yap KL, Ames JB, Swindells MB, Ikura M. Proteins 1999; 37(3):499-507.[22]
- Rigden DJ, Jedrzejas MJ, Galperin MY. FEMS Microbiol Lett 2003; 221(1):103-110.[23]
- Rigden DJ, Jedrzejas MJ, Moroz OV, Galperin MY. Trends Microbiol 2003; 11(7):295-297.[24]
- Vyas NK, Vyas MN, Quiocho FA. Nature 1987; 327(6123):635-638.[25]

- Michiels J, Xi C, Verhaert J, Vanderleyden J. Trends Microbiol 2002; 10(2):87-93.[26]
- Rigden DJ, Galperin MY. J Mol Biol 2004; 343(4):971-984.[27]
- Yang K. J Mol Microbiol Biotechnol 2001;3(3):457-459.[28]
- Momma K, Mikami B, Mishima Y, Hashimoto W, Murata K. J Mol Biol 2002; 316(5):1051-1059.[29]
- Petosa C, Collier RJ, Klimpel KR, Leppla SH, Liddington RC.Nature 1997; 385(6619):833-838.[30]
- Lytle BL, Volkman BF, Westler WM, Heckman MP, Wu JH. J Mol Biol 2001; 307(3):745-753.[31]

- C.M. van der Walt and E. Barnard, "Data characteristics that determine classifier performance", in Proceedings of the Sixteenth Annual Symposium of the Pattern Recognition Association of South Africa, pp.160-165, 2006.
- B. E. Boser, I. M. Guyon, and V. N. Vapnik. *A training algorithm for optimal margin classifiers*. In D. Haussler, editor, 5th Annual ACM Workshop on COLT, pages 144-152, Pittsburgh, PA, 1992. ACM Press.
- Corinna Cortes and V. Vapnik, "Support-Vector Networks*, Machine Learning, 20, 1995. [1]*
- Christopher J. C. Burges. "A Tutorial on Support Vector Machines for Pattern Recognition". Data Mining and Knowledge Discovery 2:121 - 167, 1998 [2]
- Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000. ISBN 0-521-78019-5 *([3] SVM Book)*
- Harris Drucker, Chris J.C. Burges, Linda Kaufman, Alex Smola and Vladimir Vapnik (1997). "Support Vector Regression Machines". *Advances in Neural Information Processing Systems 9, NIPS 1996*, 155-161, MIT Press.
- Huang T.-M., Kecman V., Kopriva I. (2006), Kernel Based Algorithms for Mining Huge Data Sets, Supervised, Semi-supervised, and Unsupervised Learning, Springer-Verlag, Berlin, Heidelberg, 260 pp. 96 illus., Hardcover, ISBN 3-540-31681-7[4]

- Vojislav Kecman: "Learning and Soft Computing - Support Vector Machines, Neural Networks, Fuzzy Logic Systems", The MIT Press, Cambridge, MA, 2001.[5]
- Bernhard Schölkopf and A. J. Smola: *Learning with Kernels*. MIT Press, Cambridge, MA, 2002. *(Partly available on line: [6].)* ISBN 0-262-19475-9
- Bernhard Schölkopf, Christopher J.C. Burges, and Alexander J. Smola (editors). "Advances in Kernel Methods: Support Vector Learning". MIT Press, Cambridge, MA, 1999. ISBN 0-262-19416-3. [7]
- John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004. ISBN 0-521-81397-2 *([8] Kernel Methods Book)*
- P.J. Tan and D.L. Dowe (2004), MML Inference of Oblique Decision Trees, Lecture Notes in Artificial Intelligence (LNAI) 3339, Springer-Verlag, pp1082-1088. (This paper uses minimum message length (MML) and actually incorporates probabilistic support vector machines in the leaves of decision trees.)
- Vladimir Vapnik, S.Kotz "Estimation of Dependences Based on Empirical Data" Springer, 2006. ISBN: 0387308652, 510 pages [this is a reprint of Vapnik's early book describing philosophy behind SVM approach. The 2006 Appendix describes recent development].
- Dmitriy Fradkin and Ilya Muchnik "Support Vector Machines for Classification" in J. Abello and G. Carmode (Eds) "Discrete Methods in Epidemiology", DIMACS Series in Discrete Mathematics and Theoretical Computer Science, volume 70, pp. 13-20, 2006. [9]. succinctly describes theoretical ideas behind SVM.

Objectives

With the constantly increasing number of data deposited and the computational tools evolving, the focus of research has shifted from the study of a single gene towards an intra- and inter-species comparison of genes and gene products. This trend can also be seen in the field of structural biology, where the number of protein structures deposited in the Protein Data Bank, PDB is increasing rapidly. However, looking at the structure alone is not sufficient for a comprehensive study of the various types of relationships between proteins. Since the number of sequence deposited in the sequence data base like SWISSPROT and NCBI; out races the amount of protein structure deposited in the PDB. This demand the development of automated in silico tools for functional and structural annotations of proteins form the protein sequences only. Since the Ca2+ binding proteins are ubiquitously present among the organisms starting form bacteria to higher organism including human being. These proteins regulate most of the cellular process directly or indirectly, in silico annotation of these proiteins are very important. The objectives of the present study are as follows.

1. To prediction of Ca2+ binding proteins or not from the user input protein sequence. (User can give a single protein sequence or the entire proteome of an organism).

2. If the submitted sequence is a Ca2+ binding protein; to locate the Ca2+ binding motifs in the next step.

3. To develop an automated tool (CalPred 1) to implement above objectives using machine learning techniques: Artificial Neural Network (ANN) and Support Vector Machine (SVM).

4. To develop a second automated tool (CalPred 2) to classify the predicted Ca2+ binding protein into their corresponding families.

CHAPTER 2

CalPred: A tool for the prediction of EF-hand calcium-binding proteins and identification of calcium-binding regions using machine learning techniques.

CalPred2: A tool for the classification of predicted Ca2+ binding proteins into different Ca2+ binding families.

The tools developed in this study (CalPred and CalPred2) are uploaded in the University web server. The manuscript is communicated for publication in *Journal of Computational Biology*, Wiley Publisher.

Motivation: Predicting calcium binding proteins and identifying calcium-binding sites is an important problem in the field of proteomics.Most of the currently used methods employs structural protein data to predict calcium-binding sites. Here we present a method developed to predict calcium-binding proteins and identify calcium-binding sites from protein sequence data using machine learning techniques such as neural networks and support vector machines.

Results: We have developed the novel application *CalPred,* having nine implemented algorithms divided into two filters. The first filter predicts proteins as a whole i.e. whether they are calcium-binding proteins or not, and the second filter predicts the specific calcium binding sites in the proteins which have passed through the first filter. The tool was able to pick sequences that were calcium-binding in nature but were not picked up by pattern sited as calcium-binding domain in PROSITE database. We also scanned four whole proteomes for potential calcium binding proteins.

Availability: The CalPred tool is available for free use to noncommercial users and can be downloaded to be used in-house as a stand alone server from http://www.juit.ac.in/calpred/index.html.

1 INTRODUCTION:

Calcium plays an important role in many biological processes including cell signaling (Carafoli, 2002), apoptosis (Orrenius *et al.*, 2003) and cell differentiation (Hennings *et al.*, 1980). It performs its various functions by binding with Ca^{+2} receptors called the calcium binding proteins or the CaBPs. Thus it is important to predict the CaBPs and identify the regions in these proteins where Ca^{+2} ions bind. Attempts have been made to solve the problem of identifying protein calcium-binding sites or in general metal binding sites in proteins previously, but most of them used protein structure information. (Deng *et al.*, 2005; Wei *et al.*, 1999; Sodhi *et al.*, 2004; Liang *et al.*, 2003) Here we present a method to differentiate between CaBPs / non-CaBPs and find calcium-binding sites using protein sequence data. The CaBPs often share a common motif known as the EF-hand motif. The EF-hand motif is a helix turn-helix structural motif that is twelve to thirteen residues long and is cited in PROSITE database, (Hulo et al., 2006) under entry number PS00018. The simplest way to predict the EF-hand cal- cium binding proteins would be using pattern matching, but in many organisms EF-hand-like calcium-binding proteins with different structural elements around the Ca^{+2} ions binding loop regions have been identified (Rigden *et al.*, 2003; Rigden *et al.*, 2003); and many of them have flexible lengths of Ca+2-binding loops that are different from loops present in EF-hand motifs.Thus, there is a need to apply techniques on CaBPs problem; that are not entirely dependent on pattern matching techniques. These could be using statistical / machine learning techniques that try to capture global information of the protein sequences.

Various machine learning techniques have been applied to biological problems related to proteins including protein structure prediction (Rost, 1996), protein fold recognition (Ding and Dubchak, 2001), protein sub-cellular localization prediction (Hua and Sun, 2001) and prediction of proteasome cleavage motifs (Kesmir *et al.*, 2002). Here we applied two machine learning techniques namely, artificial neural networks (ANN) and support vector machines (SVM) to the problem of calcium-binding protein prediction and calcium-binding region identification.

2 MATERIALS AND METHODS:

2.1 Modules and filter layers

In this study along with the two machine learning techniques, we used three types of protein sequence encoding methods i.e. "pepstats","binary" and "pssm encoding" methods. Using a combination of these three methods, and the two techniques nine modules were created namely; ANN*pepstats*, ANN*binary*, ANN*pssm*, SVM*pepstats_linear*, SVM*pepstats_polynomial*, SVM*pepstats_radial_basis*, SVM*pepstats_sigmoidal_tanh*, SVM*binary* and SVM*pssm*. The nomenclature of these modules is according to the rule that first word in the name indicates the machine learning technique say ANN or SVM, and the second subscripted word signifies the encoding method; "pepstats", "binary" or "pssm encoding". In case of SVM, a third subscripted word was used indicating the kernel type. If it is absent as in case of SVMbinary and SVMpssm the kernel type, defaults to linear kernel. The modules associated with pepstats encoding method constitute the first level filter of

CalPred tool (Figure 1). The "pepstats"(Rice *et al.*, 2000) is an application from EMBOSS suite that calculates physicochemical properties of a protein sequence as a whole and thus it is used to predict the nature of a protein, whether it's a CaBP or non-CaBP. The other encoding methods work on a single amino acid of the sequence at a time, using the information from that particular residue and the seven neighbor residues on each side. So the modules associated with "binary" and "pssm"encoding methods are incorporated in second-level filter that is used to predict calcium binding regions.

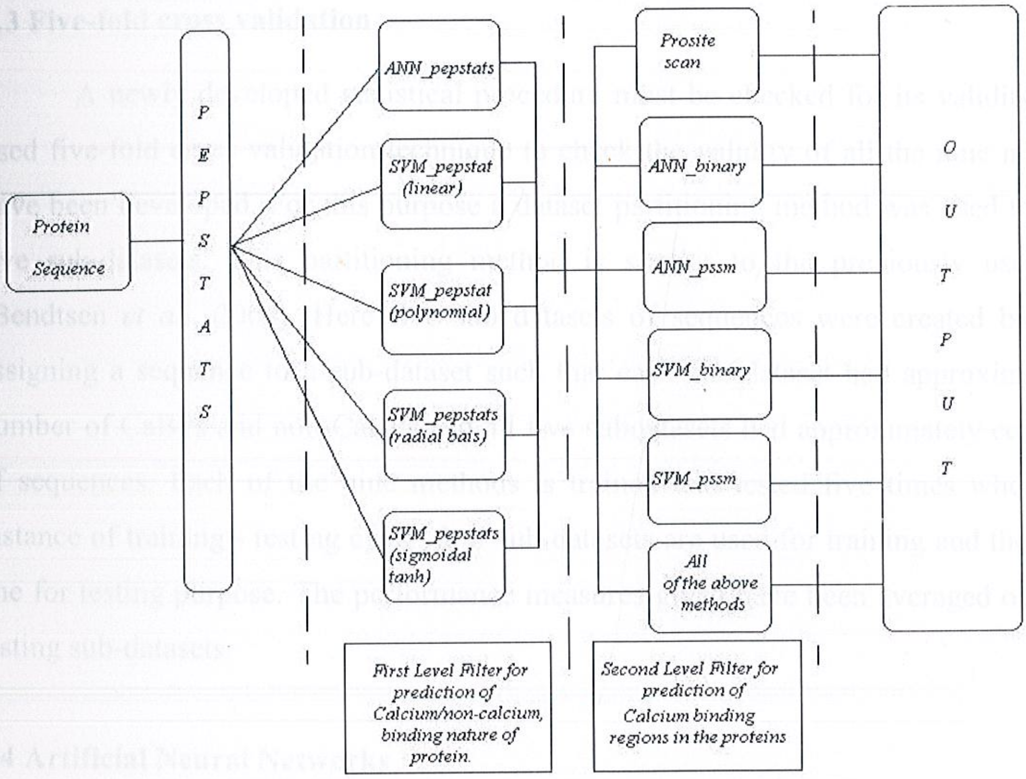

Protein Sequence
P E P S T A T S
ANN_pepstats
SVM_pepstat (linear)
SVM_pepstat (polynomial)
SVM_pepstats (radial bais)
SVM_pepstats (sigmoidal tanh)
Prosite scan
ANN_binary
ANN_pssm
SVM_binary
SVM_pssm
All of the above methods
O U T P U T
First Level Filter for prediction of Calcium/non-calcium, binding nature of protein.
Second Level Filter for prediction of Calcium binding regions in the proteins

Figure 1: Model showing different modules and their organization into the two layers.

2.2 Datasets

The initial dataset consisted of 306 CaBPs and 358 non-CaBP sequences. The CaBPs were obtained from EF-hand calcium binding proteins data library available at http://structbio.vanderbilt.edu/cabp_database/ and the non-CaBPs were obtained from Entrez protein database. The protein datasets used in the training-testing cycles of the nine modules were checked for sequence similarity to remove redundancy. The initial two datasets were filtered so that no protein sequence in the final datasets had more than 90% sequence similarity with sequence in that dataset. This was done using CD-HIT program (Li and Godzik, 2006) that clustered the sequences of a dataset in different clusters such that each cluster had sequences with more than 90%sequence similarity. Representative sequence from each cluster was taken to form the final datasets. After removing redundancy, the final datasets consisted of 188 CaBPs and 214 non-CaBP sequences. The sequence identifiers for these datasets are available in the supplementary material.

2.3 Five-fold cross validation

A newly developed statistical procedure must be checked for its validity. We have used five-fold cross validation technique to check the validity of all the nine modules that have been developed. For this purpose a dataset partitioning method was used to create the five sub-datasets. This partitioning method is similar to the previously used methods (Bendtsen *et al.*, 2004). Here five sub datasets of sequences were created by randomly assigning a sequence to a sub-dataset such that each sub-dataset had approximately equal number of CaBPs and non-CaBPs and all five subdatasets had approximately equal number of sequences. Each of the nine methods is trained and tested five times where, in each instance of training - testing cycle; four sub-datasets are used for training and the remaining one for testing purpose. The performance measures given have been averaged over the five testing sub-datasets.

2.4 Artificial Neural Networks

All the three neural network modules were implemented using the Stuttgart neural network simulator (Zell and Mamier, 1997). A feed-forward neural network with standard back-propagation algorithm is utilized in all cases but the architecture differs as their function differs.

In ANNpepstats module the neural network used had an architecture as 51-4-1 i.e. it had 51 nodes in input layer representing the values of physicochemical properties from the pepstats encoding method, 4 nodes in hidden layer and 1 node in output layer showing whether a given protein is CaBP or non-CaBP. The cut-off value used for prediction in this module is 0.9, i.e. a query protein is regarded as belonging to CaBP family if its score is greater than or equal to 0.9.

The ANNbinary and ANNpssm modules were incorporated in the second level filter of the CalPred tool; that functions to predict calcium binding regions in the given protein. The calcium-binding domain entry given in PROSITE database had a calcium-binding motif of thirteen residues length. Using this information; the window size in ANNbinary and ANNpssm modules was fixed at thirteen. In these modules, thirteen residues of a protein sequence are taken at a time; and the prediction is done for the middle residue i.e. seventh residue. Both the binary as well as pssm encoding methods, encodes the thirteen residues into 260 numeric values and thus input layer of these modules consisted of 260 nodes. The hidden layer in both modules had 20 nodes. The output layer of these modules consisted of a single node predicting whether the Ca+2 ions will bind to the seventh residue or not. Thus, the ANNbinary and ANNpssm modules share the same architecture of 260-20-1. The cut-off value used for prediction in both of these modules is 0.5, i.e. the seventh residue of a window is regarded as a Ca+2 ion binding residue if its score is greater than or equal to 0.5.

2.5 Support Vector machines

The support vector machines used in the SVM related modules first tried to map the input vector into high dimensional feature space, either linearly or by methods depending on kernel type chosen; such that error is minimized over the training dataset. Then an optimized division is sought between the positive and negative classes say "a CaBP and non-CaBP protein" or "a Ca+2 ion binding residue and non-Ca+2 ion binding residue". This was done by constructing a hyperplane that separated these two classes by the largest margin (Vapnik, 1998). Here we have used SVMlight software (Joachims, 1998) for implementing the support vector machine related modules. The SVMlight software allows users to choose from a number of available modes and kernel functions. In all the modules, a default classification mode is used while kernel functions are varied. The kernel functions available are linear, polynomial, radial basis and sigmoidal. In all the support vector machines the cut-off value used for prediction is 0, i.e. a query vector is regarded as member of positive dataset if its score is greater than 0 and is regarded as member of negative dataset if its score is less than 0. The ones having scored equal to zero are regarded as undefined.

2.6 Encoding methods

In this study, three types of encoding methods are used, these are "pepstats", "binary" and "pssm encoding" methods.

- Using the "pepstats encoding" method, the protein sequence is encoded into its physicochemical properties using pepstats application of EMBOSS suite of programs. Out of all the properties, 51 properties are then used to create an input vector for training and testing of the machine learning modules. These properties are normalized, prior to creation of the input vectors. The list of properties used for training and testing the modules and their associated normalization factors that are used in this study; are given in the supplementary material.

- The "binary encoding" method takes a window of thirteen amino acid residues at a time, and encodes every amino acid by a group of 20 units, each for a possible amino acid type that can be present at a particular position; thus creating thirteen binary vectors (1, 0, 0, . . .) and for a particular window totaling to 260 (13*20) units.

- The "pssm encoding" method uses position specific scoring matrices created by PSI-BLAST using three iterations and with default e-value threshold of 0.001. The matrix has 20 *M elements, where M is the length of query protein; and each element represents the frequency of occurrence of each of the 20 amino acids at one position in the alignment (Altschul *etal*, 1997). This encoding method then uses overlapping windows of 13*20 elements to create input vectors for machine learning modules.In both the binary and pssm encoding methods, the protein sequence is tagged with a stretch of 7 "Xs", such that information on the first and last seven residues is not lost. In these methods, the X's are encoded as vectors of twenty zero's.

2.7 Performance Measures

The performance measures used to evaluate the nine modules are listed below. These measures have been calculated for each module using five test sub-datasets; and the final measures (Table 1) given have been averaged over these five sub datasets.

- *Accuracy*: The accuracy of the modules have been calculated as:

$$Q_{Acc} = \frac{P + N}{P + N + O + U}$$

Where P and N refer to true positives and true negatives say correctly predicted calcium-binding and non calcium-binding proteins respectively; and O and U refer to false positives and false negatives i.e. incorrectly predicted calcium-binding and non calcium-binding proteins.

- *Specificity* (Qspec) and *sensitivity* (Qsens) of the modules are defined as:

$$Q_{spec} = \frac{N}{N + O} \qquad Q_{sens} = \frac{P}{P + U}$$

- The *Matthews correlation coefficient* (MCC) is defined as:

$$MCC = \frac{(P \times N) - (O \times U)}{\sqrt{(P + U) \times (P + O) \times (N + U) \times (N + O)}}$$

- QPred (*Probability of correct prediction*) is defined as:

$$Q_{pred} = \frac{P}{P + O} \times 100$$

3 RESULTS AND DISCUSSION

3.1 Performance over test dataset

The performance of the modules over the test dataset is given in Table 1. The accuracy of first-level filter modules on the test dataset varies from 57% to 94%. Support vector machine's polynomial kernel type performed the best on the test dataset amongst all the classifiers / modules in first-level filter. The other measures of SVM*pepstats_polynomial* module are also better than the modules in the first-level filter, with sensitivity of 0.917 and specificity of 1.00.

The performance measures for modules in second-level filter, used for predicting calcium-binding regions are also calculated. There is no perfect method to calculate the specific calcium binding regions in a protein, as one can observe from the pattern matching technique; using PROSITE database's calcium-binding domain entry which also gives proteins that are calcium-binding but are not picked up by the pattern. Therefore the performance measures of test dataset, for the second-level filter are calculated assuming; each residue belonging to a calcium-binding protein which is predicted as a calcium-binding site by the second-level filter is a true positive sample and a residue belonging to a non calcium-binding protein which is predicted as a non calcium binding site by the second-level filter is a true negative sample. Similarly, false positives and false negatives were calculated. Though this assumption does not make second-level filter an efficient filter, but due to lack of a standard method of finding calcium- binding regions in the protein; we made this assumption for the purpose of calculating the performance measures in the same manner as they were calculated in first-level filter. Due to this assumption, SVM related modules of the second-level filter show high performance measures on the test datasets (Table 1).

3.2 Performance over four organisms

The whole proteomes of four organisms namely, *Arabidopsis thaliana, Caenorhabditis elegans, Drosophila and Homo sapiens* were analyzed to predict the calcium-binding proteins in them. These proteomes were taken from Entrez protein database and analyzed using first-level filter modules. All modules having accuracy greater than 65%

were used in this analysis and the detailed results of proteomes; as data files and tabulated statistics are given in the supplementary material. A protein is regarded as a calcium binding protein if any of its score from first-level modules is above the cut-off score of that module. Performance measures are not calculated based on the data from whole proteomes, due to the large difference in number of sequences present in positive and negative datasets of some proteomes.

Name of the module	*Accuracy*	*Specificity*	*Sensitivity*	*Probability of correct prediction (QPred)*	*Matthews correlation coefficient (MCC)*
ANN*pepstats*	0.8815	0.8536	0.9176	79.64	0.7579
ANN*binary*	0.7131	0.7527	0.5907	43.64	0.3141
ANN*pssm*	0.8364	0.8790	0.7538	75.59	0.6327
SVM*pepstats_linear*	0.9406	0.9109	1.00	87.58	0.8927
SVM*pepstats_polynomial*	0.9453	0.9170	1.00	88.51	0.9006
SVM*pepstats_radial_basis*	0.7075	0.5535	0.7451	89.91	0.4487
SVM*pep-stats_sigmoidal_tanh*	0.5761	0.6508	0.9116	22.64	0.1763
SVM*binary*	0.9101	0.9239	0.8797	83.89	0.7935
SVM *sm*	0.9994	0.9993	0.9994	99.87	0.9986

Table 1. Summary of the prediction results of the nine modules used in first and second layers on the test dataset.

The plots for distribution of scores of proteins from ANN*pepstats,* SV*Mpepstats_linear,* SVM*pepstats_polynomial* and SVM*pepstats_radial_basis* for all the four proteomes are given in the supplementary material. Here we have given a brief overview of these plots for human proteome.

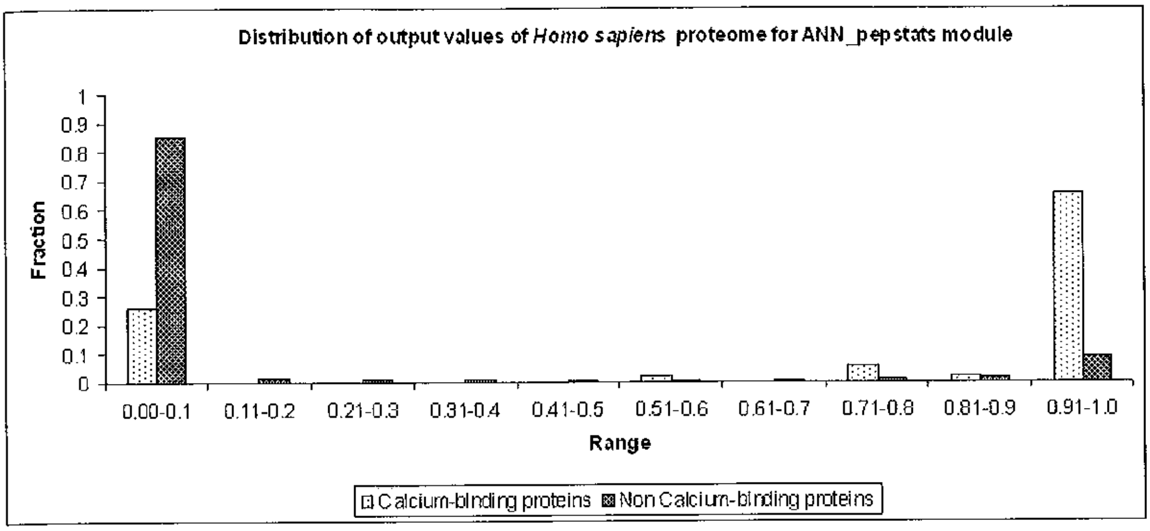
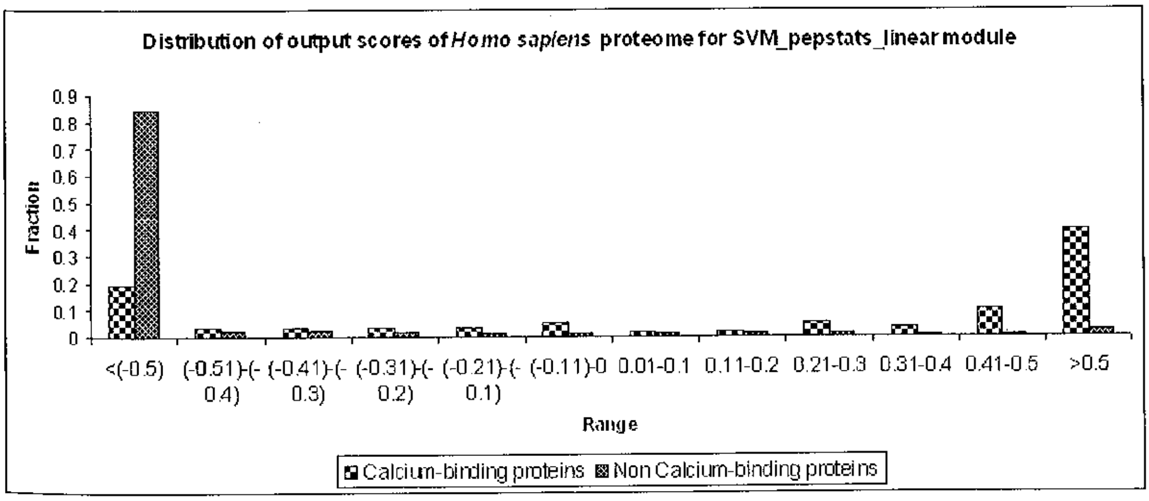


Figure 2: Distribution of output values of *Homo sapiens* proteome for ANN*pepstats* module.

*3.2.2 **The Human proteome**:* In the human proteome we analyzed 34,122 non-calcium binding proteins and 58 calcium-binding proteins. The plots for distribution of output scores of ANN*pepstats,* SV*Mpepstats_linear,* SVM*pepstats_polynomial* and SVM*pepstats_radial_basis* module for human proteome are given as Figures 2-5. It is clear from the plot (Figure 2) that ANN*pepstats* module separates the two datasets i.e. calcium-binding proteins and non calcium-binding proteins to a large extend. Mostly the calcium-binding proteins score above 0.9 value and the non calcium-binding proteins score below 0.1.

The SVM*pepstats_linear* and SVM*pepstats_polynomial* modules having zero threshold values, for classifying the proteins also identified the non calcium-binding proteins effectively (Figure 3 and 4) but



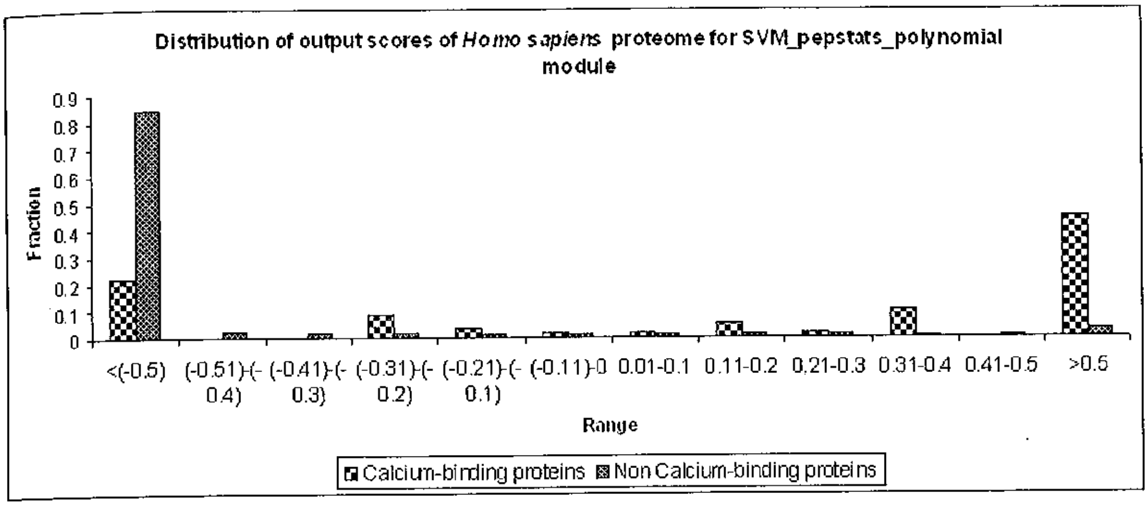


Figure 3-4: Distribution of output values of *Homo sapiens* proteome for SVM*pepstats_linear* and SVM*pepstats_polynomial* modules.

were found to be lesser efficient in screening calcium-binding proteins as the scores for the later were scattered over the whole range of -1 to 1, though majority of the scores were present above the threshold cut-off value. The SVM*pepstats_radial_basis* module was not able to distinguish the proteins efficiently in the proteome (Figure 5); this was expected as the module showed lesser accuracy on test dataset too.

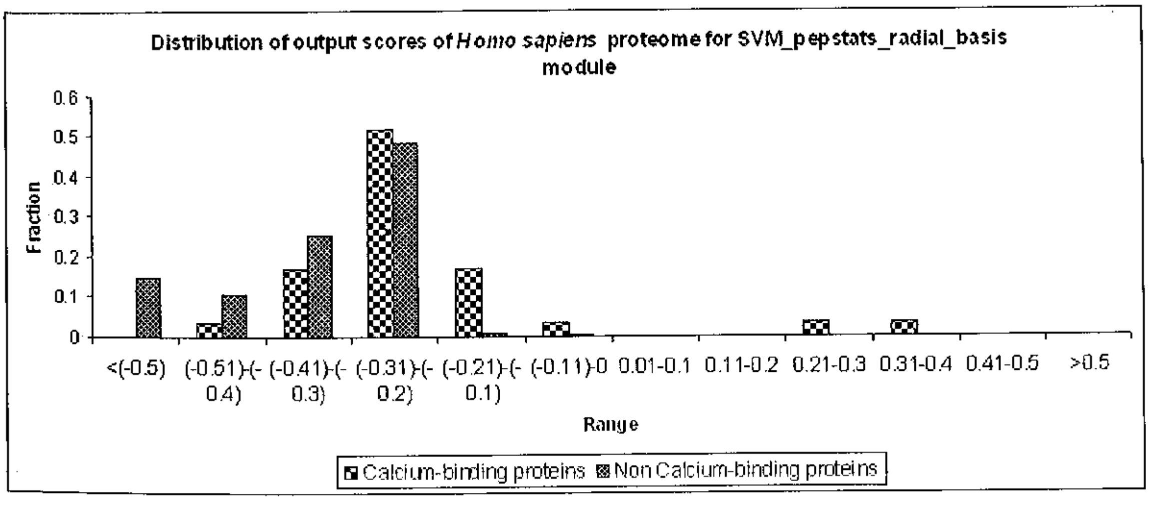


Figure 5: Distribution of output values of *Homo sapiens* proteome for SVM*pepstats_radial_basis* module.

3.3 Performance over PROSITE dataset

The calcium-binding proteins that were not picked up by the pattern cited as calcium-binding domain entry in PROSITE database with ID PS00018 were analyzed with the first- and second-level filters. There were 85 such proteins and these are listed in the entry itself. The first-level filter was able to identify 45 of the calcium binding proteins. The scores of these entries for first-level filter are given as data file in supplementary material. Here too we have used, only those first-level filter modules that showed greater than 65% accuracy on test dataset. The plots for distribution of scores of these entries with first-level filter for different modules are given in supplementary material. The detailed results for second-level filter are given in the supplementary material.

4 CONCLUSIONS

Analyzing calcium-binding proteins from sequence data is important in calcium-mediated biological studies. This approach is vital in cases where the structural data for proteins is unavailable. Intelligent systems like the ones used in this study, utilize global information of protein sequence data instead of using simple pattern matching techniques. Such systems can significantly increase the accuracy of calcium-binding proteins related studies. This study gives an insight into development of similar tools and techniques for other metal-binding proteins or for all metal-binding proteins in general.

ACKNOWLEDGEMENTS

We are thankful to Nelson *et al,* authors of EF-hand calcium binding protein data library for making the data library publicly available.

REFERENCES

Altschul,S.F., Madden,T.L., Alejandro,A.S., Zhang,J., Zheng,Z., Miller,W., and Lipman,D.J. (1997) Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Res.*, 25, 3389-3402.

Bendtsen,J.D., Jensen,L.J., Blom,N., Von,H.G. and Brunak,S. (2004) Feature-based prediction of non-classical and leaderless protein secretion. *Protein Eng. Des Sel*, 17, 349-356.

Carafoli,E. (2002) Calcium signaling: A tale for all seasons. *Proc Natl Acad Sci*, 99(3), 1115-1122.

Ding,C.H.Q. and Dubchak,I. (2001) Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 17(4), 349-358.

Deng,H., Liu,H. and Zhang,Y. (2005) Mining Calcium-binding Sites from Protein Structure Graphs. *ICNN&B '05 International Conference on Neural Networks and Brain*, 1980-85.

Hennings,H., Michael,D., Cheng,C., Steinert,P., Holbrook,K. and Yuspa,S.H. (1980) Calcium regulation of growth and differentiation of mouse epidermal cells in culture. *Cell*, 19, 245-254.

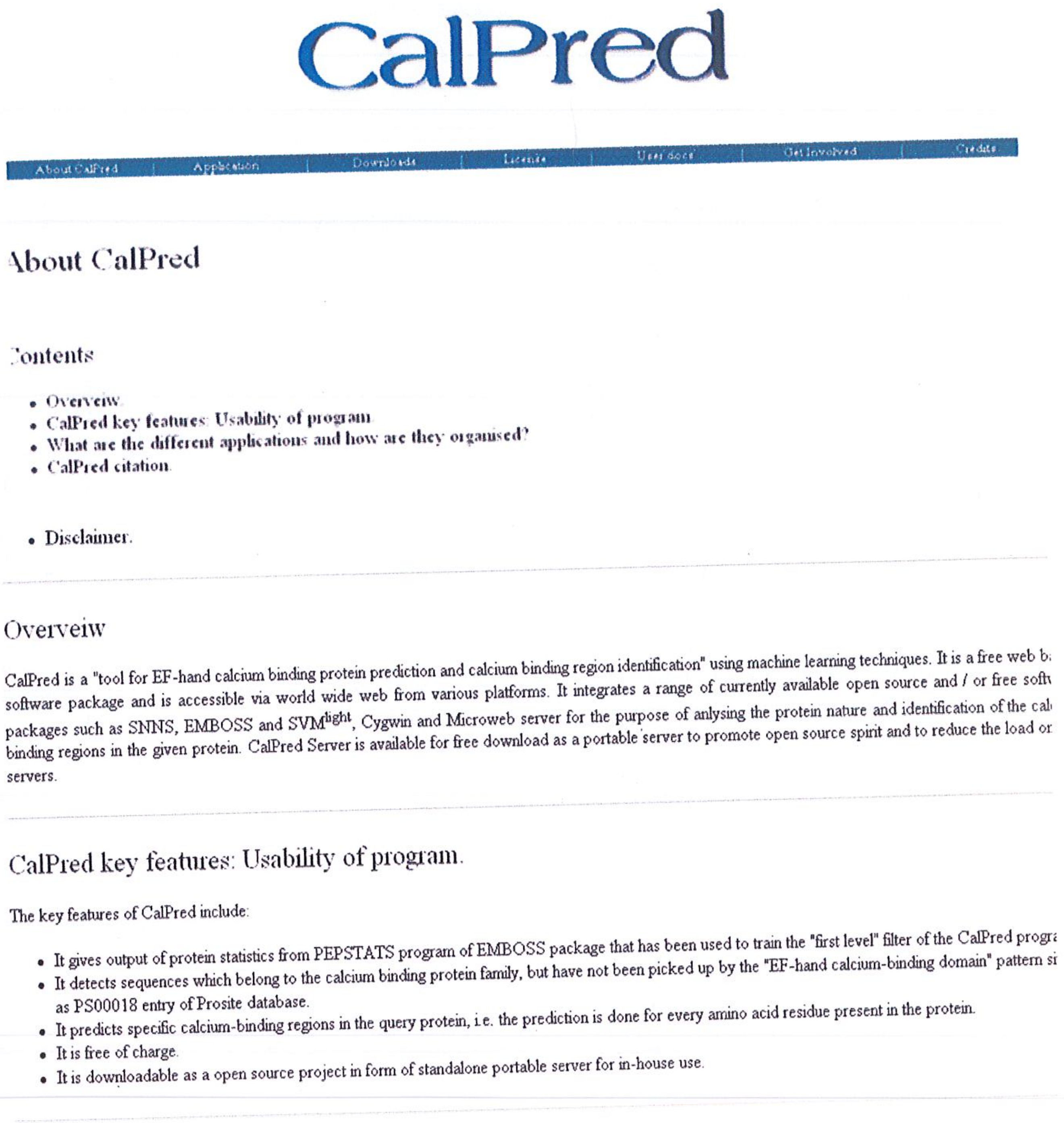
Hulo,N., Bairoch,A., Bulliard,V., Cerutti,L., De Castro,E., Langendijk-Genevaux,P.S., Pagni,M., Sigrist,C.J.A. (2006) The PROSITE database. *Nucleic Acids Res.*, 34, D227-D230.

Hua,S. and Sun,Z. (2001) Support vector machine approach for protein subcellular localization prediction. *Bioinformatics*, 17(8), 721-728.

Joachims,T. (1998) Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Proceedings of the European Conference on Machine Learning.*

Kesmir,C., Nussbaum,A.K., Schild,H., Detours,V., and Brunak,S. (2002) Prediction of proteasome cleavage motifs by neural networks. *Protein Engineering.* 15(4), 287-296.

SCREENSHOT 1(HOMEPAGE)


CalPred
About CalPred | Application | Downloads | License | User docs | Get Involved | Credits

\bout CalPred

:ontents

- Overveiw.
- CalPred key features: Usability of program.
- What are the different applications and how are they organised?
- CalPred citation.

- Disclaimer.

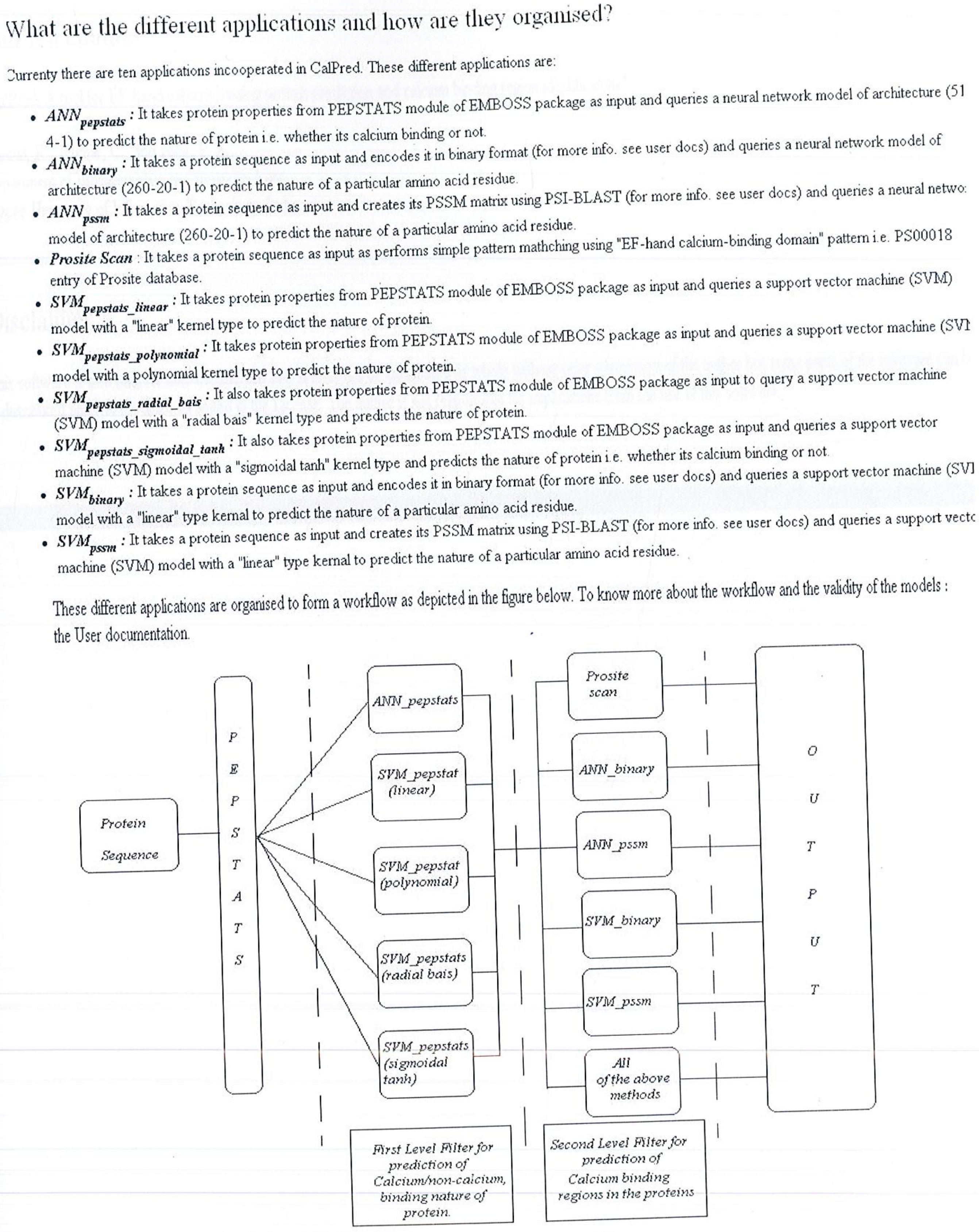
Overveiw

CalPred is a "tool for EF-hand calcium binding protein prediction and calcium binding region identification" using machine learning techniques. It is a free web b
software package and is accessible via world wide web from various platforms. It integrates a range of currently available open source and / or free soft
packages such as SNNS, EMBOSS and SVM^{light}, Cygwin and Microweb server for the purpose of anlysing the protein nature and identification of the cal
binding regions in the given protein. CalPred Server is available for free download as a portable server to promote open source spirit and to reduce the load or
servers.

CalPred key features: Usability of program.

The key features of CalPred include:

- It gives output of protein statistics from PEPSTATS program of EMBOSS package that has been used to train the "first level" filter of the CalPred progra
- It detects sequences which belong to the calcium binding protein family, but have not been picked up by the "EF-hand calcium-binding domain" pattern si
as PS00018 entry of Prosite database.
- It predicts specific calcium-binding regions in the query protein, i.e. the prediction is done for every amino acid residue present in the protein.
- It is free of charge.
- It is downloadable as a open source project in form of standalone portable server for in-house use.



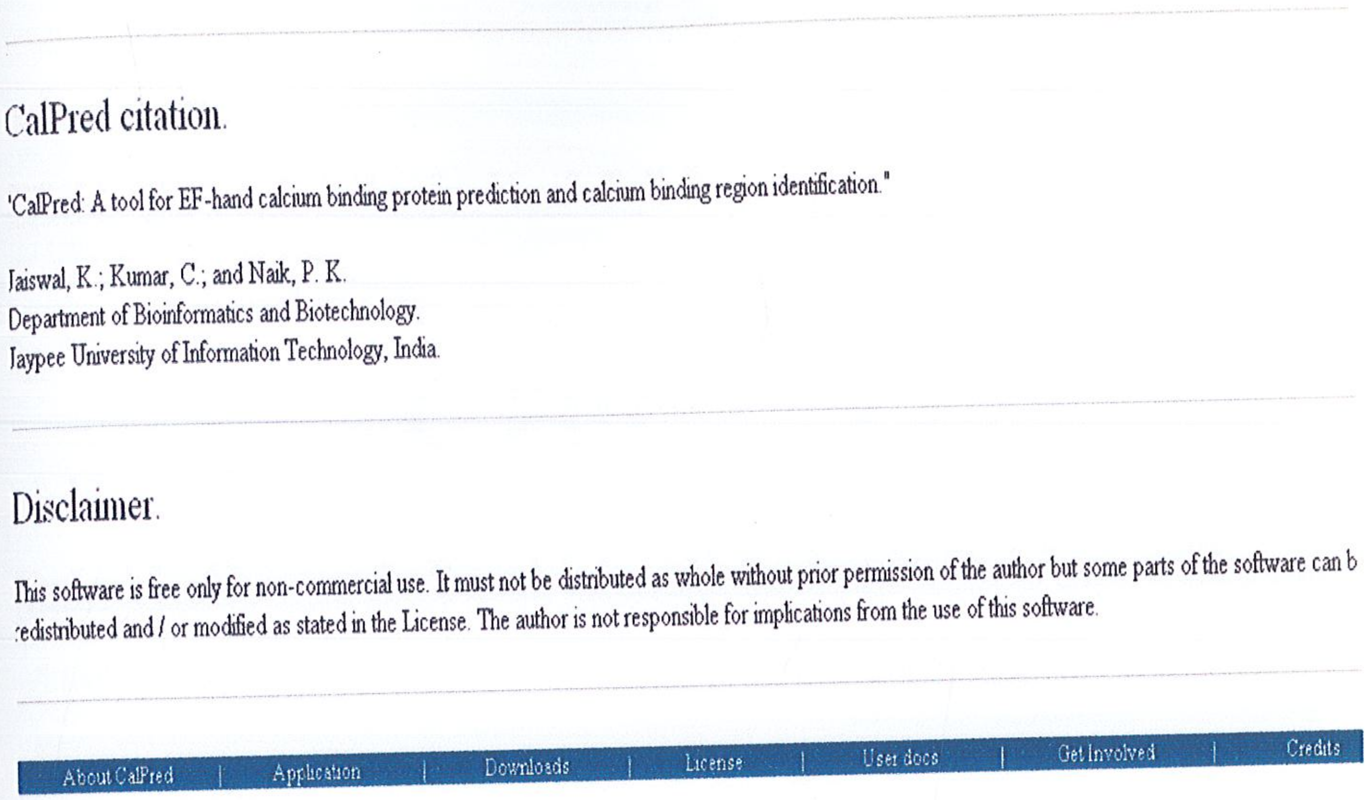
What are the different applications and how are they organised?

Currenty there are ten applications incooperated in CalPred. These different applications are:

- *ANN$_{pepstats}$* : It takes protein properties from PEPSTATS module of EMBOSS package as input and queries a neural network model of architecture (51 4-1) to predict the nature of protein i.e. whether its calcium binding or not.
- *ANN$_{binary}$* : It takes a protein sequence as input and encodes it in binary format (for more info. see user docs) and queries a neural network model of architecture (260-20-1) to predict the nature of a particular amino acid residue.
- *ANN$_{pssm}$* : It takes a protein sequence as input and creates its PSSM matrix using PSI-BLAST (for more info. see user docs) and queries a neural netwo: model of architecture (260-20-1) to predict the nature of a particular amino acid residue.
- ***Prosite Scan*** : It takes a protein sequence as input as performs simple pattern mathching using "EF-hand calcium-binding domain" pattern i.e. PS00018 entry of Prosite database.
- *SVM$_{pepstats\_linear}$* : It takes protein properties from PEPSTATS module of EMBOSS package as input and queries a support vector machine (SVM) model with a "linear" kernel type to predict the nature of protein.
- *SVM$_{pepstats\_polynomial}$* : It takes protein properties from PEPSTATS module of EMBOSS package as input and queries a support vector machine (SVI model with a polynomial kernel type to predict the nature of protein.
- *SVM$_{pepstats\_radial\_bais}$* : It also takes protein properties from PEPSTATS module of EMBOSS package as input to query a support vector machine (SVM) model with a "radial bais" kernel type and predicts the nature of protein.
- *SVM$_{pepstats\_sigmoidal\_tanh}$* : It also takes protein properties from PEPSTATS module of EMBOSS package as input and queries a support vector machine (SVM) model with a "sigmoidal tanh" kernel type and predicts the nature of protein i.e. whether its calcium binding or not.
- *SVM$_{binary}$* : It takes a protein sequence as input and encodes it in binary format (for more info. see user docs) and queries a support vector machine (SVI model with a "linear" type kernal to predict the nature of a particular amino acid residue.
- *SVM$_{pssm}$* : It takes a protein sequence as input and creates its PSSM matrix using PSI-BLAST (for more info. see user docs) and queries a support vectc machine (SVM) model with a "linear" type kernal to predict the nature of a particular amino acid residue.

These different applications are organised to form a workflow as depicted in the figure below. To know more about the workflow and the validity of the models : the User documentation.

Protein Sequence
P E P S T A T S
ANN_pepstats
SVM_pepstat (linear)
SVM_pepstat (polynomial)
SVM_pepstats (radial bais)
SVM_pepstats (sigmoidal tanh)
Prosite scan
ANN_binary
ANN_pssm
SVM_binary
SVM_pssm
All of the above methods
O U T P U T
First Level Filter for prediction of Calcium/non-calcium, binding nature of protein.
Second Level Filter for prediction of Calcium binding regions in the proteins



CalPred citation.

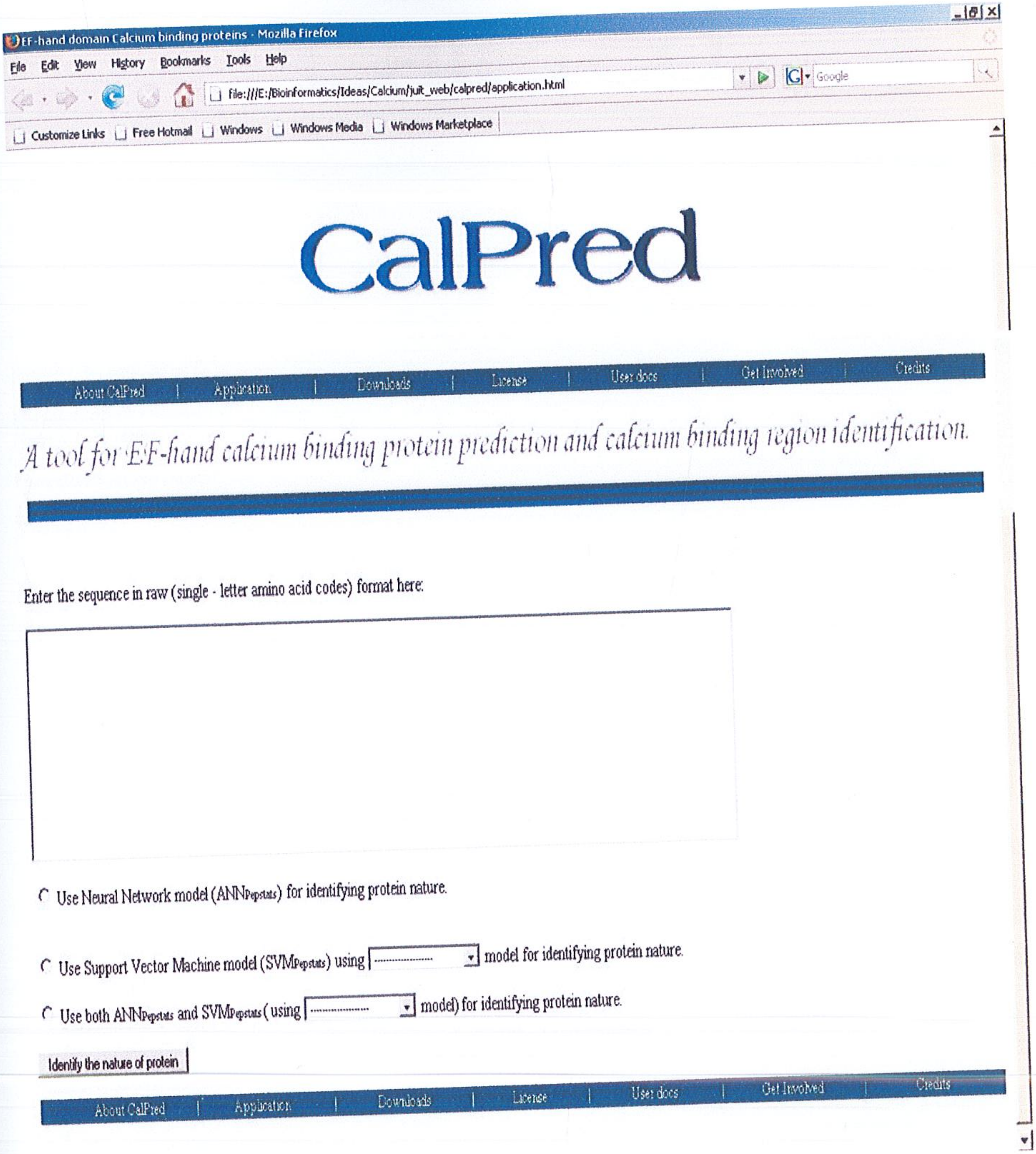
'CalPred: A tool for EF-hand calcium binding protein prediction and calcium binding region identification."

Jaiswal, K.; Kumar, C.; and Naik, P. K.
Department of Bioinformatics and Biotechnology.
Jaypee University of Information Technology, India.

Disclaimer.

This software is free only for non-commercial use. It must not be distributed as whole without prior permission of the author but some parts of the software can b redistributed and / or modified as stated in the License. The author is not responsible for implications from the use of this software.

SCREENSHOT 2(APPLICATION)


EF-hand domain Calcium binding proteins - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///E:/Bioinformatics/Ideas/Calcium/juit_web/calpred/application.html
Google
Customize Links
Free Hotmail
Windows
Windows Media
Windows Marketplace
About CalPred | Application | Downloads | License | User docs | Get Involved | Credits
Identify the nature of protein
About CalPred | Application | Downloads | License | User docs | Get Involved | Credits

CalPred

A tool for EF-hand calcium binding protein prediction and calcium binding region identification.

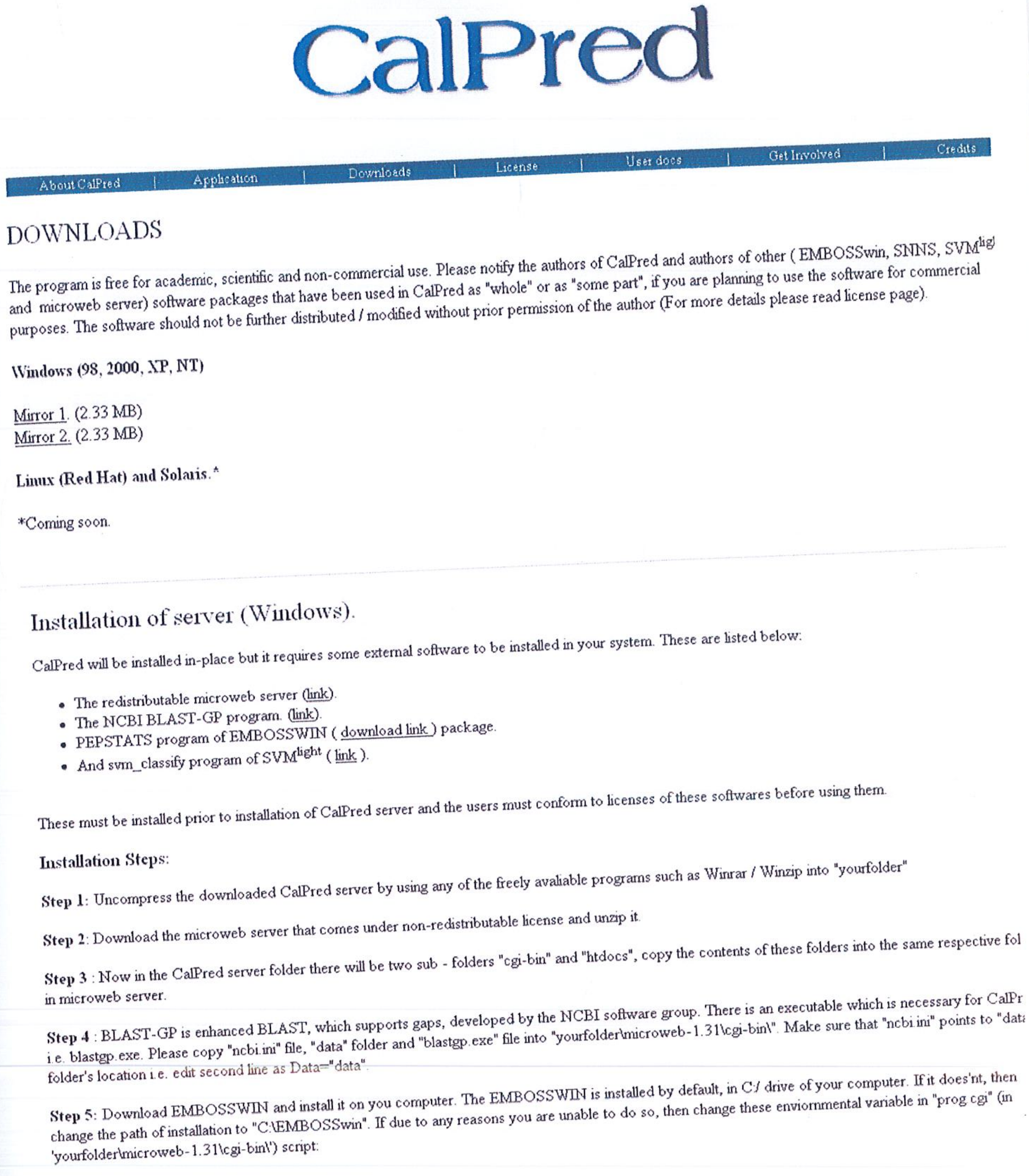
Enter the sequence in raw (single - letter amino acid codes) format here:

○ Use Neural Network model ($ANN_{Pepstats}$) for identifying protein nature.

○ Use Support Vector Machine model ($SVM_{Pepstats}$) using ------------ model for identifying protein nature.

○ Use both $ANN_{Pepstats}$ and $SVM_{Pepstats}$ (using ------------ model) for identifying protein nature.

SCREENSHOT 3(DOWNLOADS)


CalPred
About CalPred | Application | Downloads | License | User docs | Get Involved | Credits

DOWNLOADS

The program is free for academic, scientific and non-commercial use. Please notify the authors of CalPred and authors of other (EMBOSSwin, SNNS, SVM^{lig} and microweb server) software packages that have been used in CalPred as "whole" or as "some part", if you are planning to use the software for commercial and microweb server) software packages that have been used in CalPred as "whole" or as "some part", if you are planning to use the software for commercial purposes. The software should not be further distributed / modified without prior permission of the author (For more details please read license page).

Windows (98, 2000, XP, NT)

Mirror 1. (2.33 MB)
Mirror 2. (2.33 MB)

Linux (Red Hat) and Solaris.*

*Coming soon.

Installation of server (Windows).

CalPred will be installed in-place but it requires some external software to be installed in your system. These are listed below:

- The redistributable microweb server (link).
- The NCBI BLAST-GP program. (link).
- PEPSTATS program of EMBOSSWIN (download link) package.
- And svm_classify program of SVM^{light} (link).

These must be installed prior to installation of CalPred server and the users must conform to licenses of these softwares before using them.

Installation Steps:

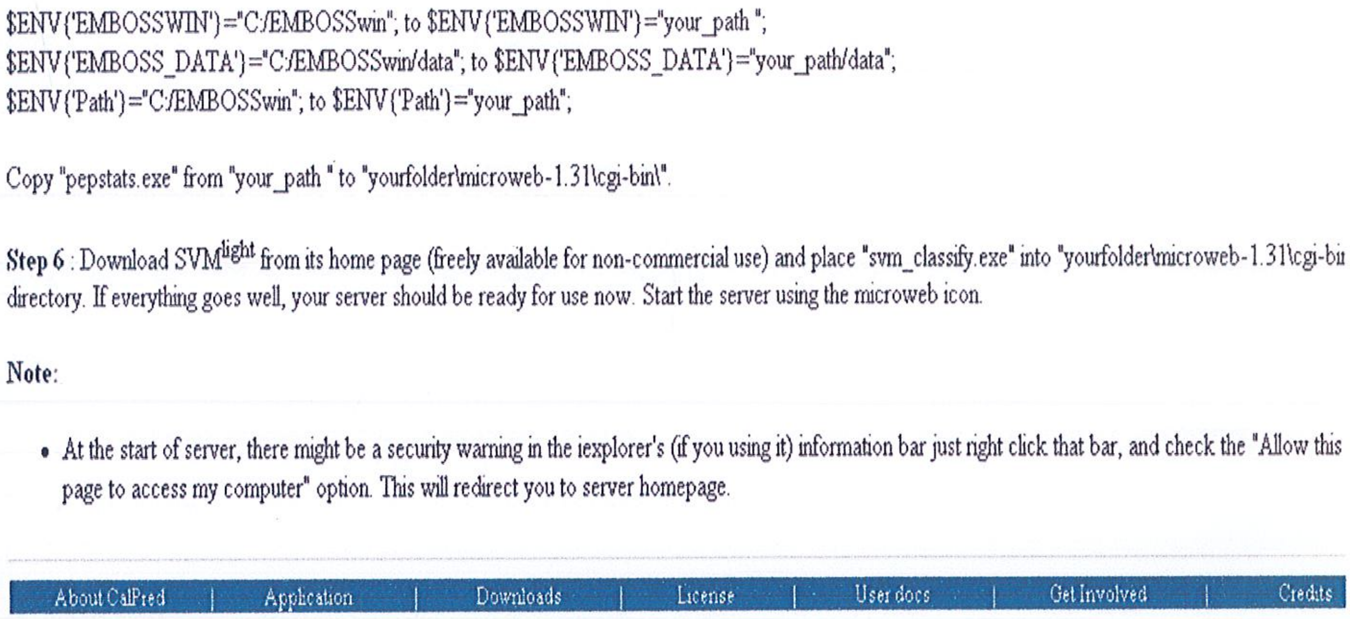
Step 1: Uncompress the downloaded CalPred server by using any of the freely avaliable programs such as Winrar / Winzip into "yourfolder"

Step 2: Download the microweb server that comes under non-redistributable license and unzip it.

Step 3 : Now in the CalPred server folder there will be two sub - folders "cgi-bin" and "htdocs", copy the contents of these folders into the same respective fol
in microweb server.

Step 4 : BLAST-GP is enhanced BLAST, which supports gaps, developed by the NCBI software group. There is an executable which is necessary for CalPr
i.e. blastgp.exe. Please copy "ncbi.ini" file, "data" folder and "blastgp.exe" file into "yourfolder\microweb-1.31\cgi-bin\". Make sure that "ncbi.ini" points to "data
folder's location i.e. edit second line as Data="data".

Step 5: Download EMBOSSWIN and install it on you computer. The EMBOSSWIN is installed by default, in C:/ drive of your computer. If it does'nt, then
change the path of installation to "C:\EMBOSSwin". If due to any reasons you are unable to do so, then change these enviornmental variable in "prog.cgi" (in
'yourfolder\microweb-1.31\cgi-bin\') script:



$ENV{'EMBOSSWIN'}="C:/EMBOSSwin"; to $ENV{'EMBOSSWIN'}="your_path ";
$ENV{'EMBOSS_DATA'}="C:/EMBOSSwin/data"; to $ENV{'EMBOSS_DATA'}="your_path/data";
$ENV{'Path'}="C:/EMBOSSwin"; to $ENV{'Path'}="your_path";

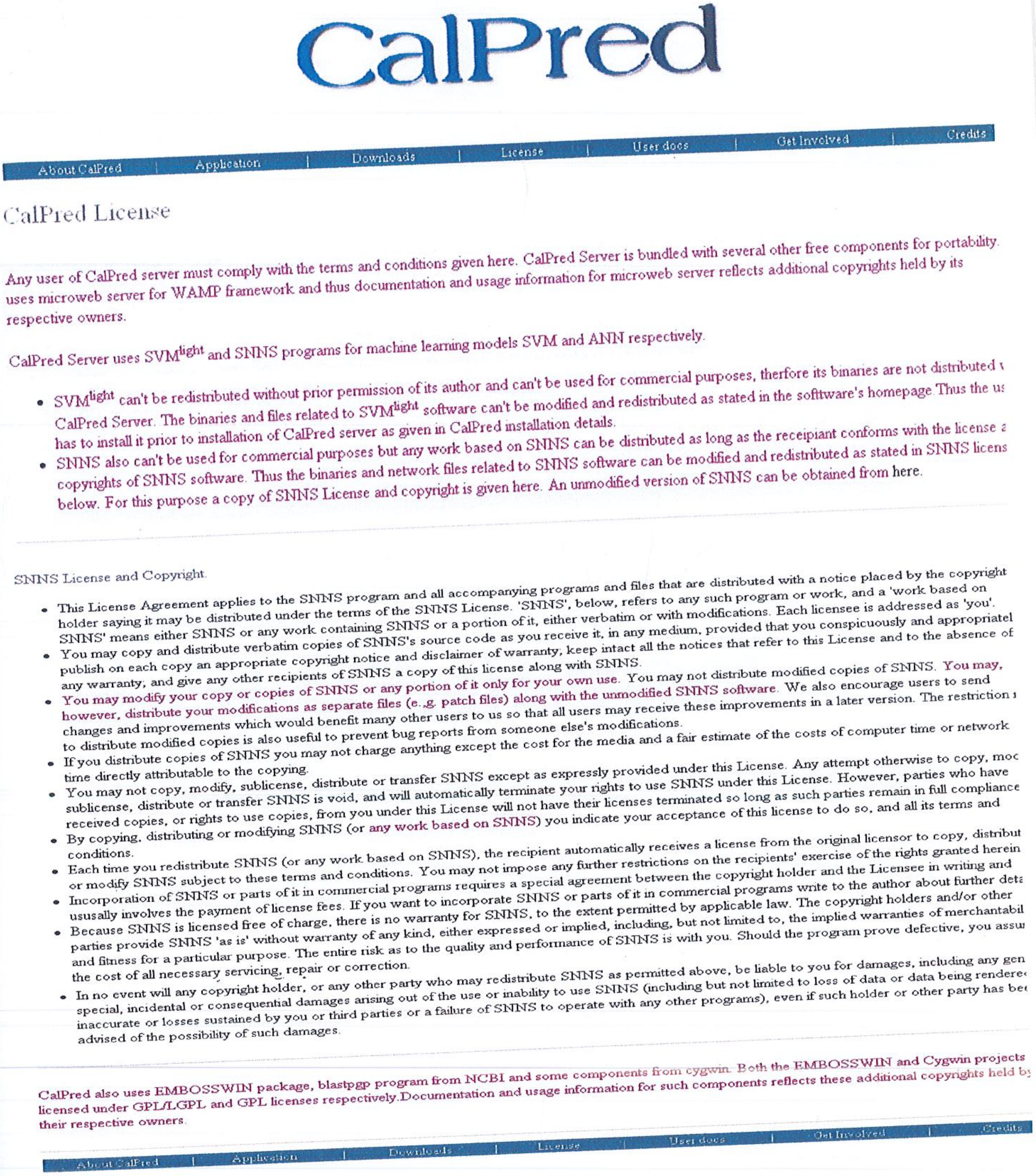
Copy "pepstats.exe" from "your_path " to "yourfolder\microweb-1.31\cgi-bin\".

Step 6 : Download SVM^{light} from its home page (freely available for non-commercial use) and place "svm_classify.exe" into "yourfolder\microweb-1.31\cgi-bi directory. If everything goes well, your server should be ready for use now. Start the server using the microweb icon.

Note:

- At the start of server, there might be a security warning in the iexplorer's (if you using it) information bar just right click that bar, and check the "Allow this page to access my computer" option. This will redirect you to server homepage.

SCREENSHOT 4(LICENSE)


CalPred
About CalPred | Application | Downloads | License | User docs | Get Involved | Credits

CalPred License

Any user of CalPred server must comply with the terms and conditions given here. CalPred Server is bundled with several other free components for portability.
uses microweb server for WAMP framework and thus documentation and usage information for microweb server reflects additional copyrights held by its
respective owners.

CalPred Server uses SVM^{light} and SNNS programs for machine learning models SVM and ANN respectively.

- SVM^{light} can't be redistributed without prior permission of its author and can't be used for commercial purposes, therfore its binaries are not distributed v
  CalPred Server. The binaries and files related to SVM^{light} software can't be modified and redistributed as stated in the softtware's homepage Thus the us
  has to install it prior to installation of CalPred server as given in CalPred installation details.
- SNNS also can't be used for commercial purposes but any work based on SNNS can be distributed as long as the receipiant conforms with the license a
  copyrights of SNNS software. Thus the binaries and network files related to SNNS software can be modified and redistributed as stated in SNNS licens
  below. For this purpose a copy of SNNS License and copyright is given here. An unmodified version of SNNS can be obtained from here.

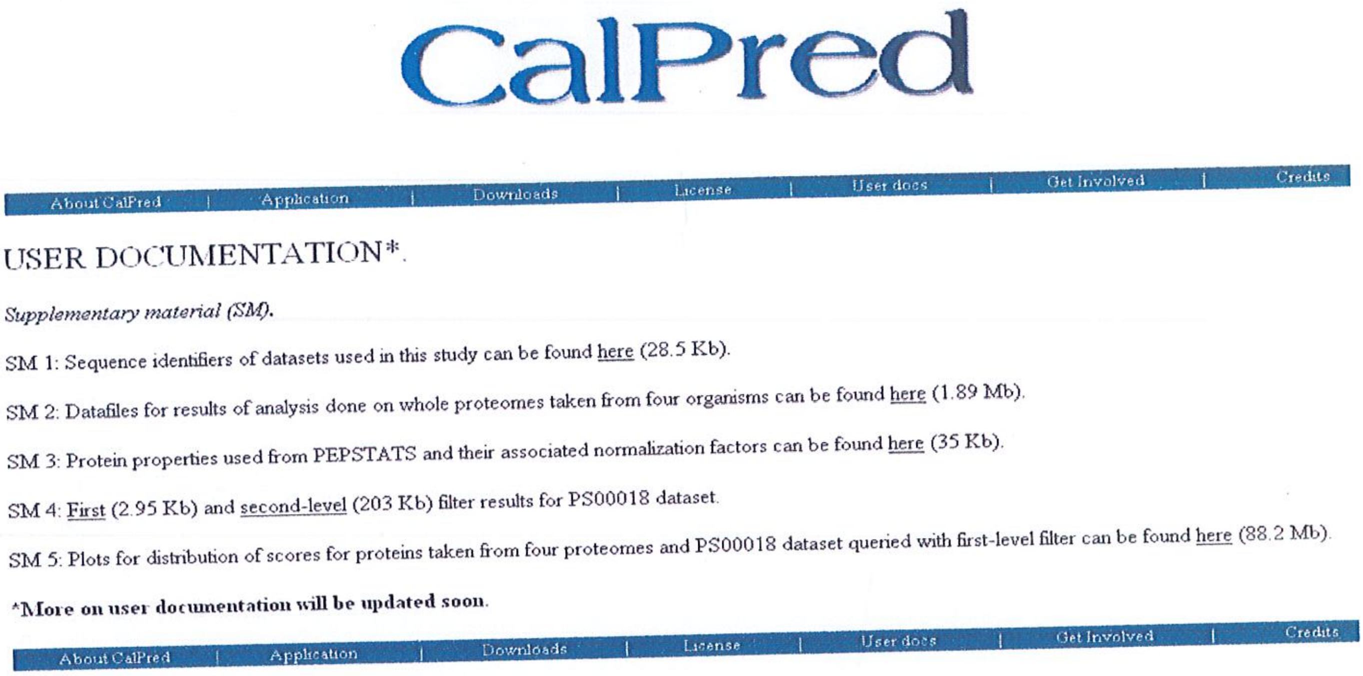
SNNS License and Copyright.

- This License Agreement applies to the SNNS program and all accompanying programs and files that are distributed with a notice placed by the copyright
  holder saying it may be distributed under the terms of the SNNS License. 'SNNS', below, refers to any such program or work, and a 'work based on
  SNNS' means either SNNS or any work containing SNNS or a portion of it, either verbatim or with modifications. Each licensee is addressed as 'you'.
- You may copy and distribute verbatim copies of SNNS's source code as you receive it, in any medium, provided that you conspicuously and appropriatel
  publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of
  any warranty; and give any other recipients of SNNS a copy of this license along with SNNS.
- You may modify your copy or copies of SNNS or any portion of it only for your own use. You may not distribute modified copies of SNNS. You may,
  however, distribute your modifications as separate files (e.,g. patch files) along with the unmodified SNNS software. We also encourage users to send
  changes and improvements which would benefit many other users to us so that all users may receive these improvements in a later version. The restriction t
  to distribute modified copies is also useful to prevent bug reports from someone else's modifications.
- If you distribute copies of SNNS you may not charge anything except the cost for the media and a fair estimate of the costs of computer time or network
  time directly attributable to the copying.
- You may not copy, modify, sublicense, distribute or transfer SNNS except as expressly provided under this License. Any attempt otherwise to copy, moc
  sublicense, distribute or transfer SNNS is void, and will automatically terminate your rights to use SNNS under this License. However, parties who have
  received copies, or rights to use copies, from you under this License will not have their licenses terminated so long as such parties remain in full compliance
- By copying, distributing or modifying SNNS (or any work based on SNNS) you indicate your acceptance of this license to do so, and all its terms and
  conditions.
- Each time you redistribute SNNS (or any work based on SNNS), the recipient automatically receives a license from the original licensor to copy, distribut
  or modify SNNS subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein
- Incorporation of SNNS or parts of it in commercial programs requires a special agreement between the copyright holder and the Licensee in writing and
  ususally involves the payment of license fees. If you want to incorporate SNNS or parts of it in commercial programs write to the author about further deta
- Because SNNS is licensed free of charge, there is no warranty for SNNS, to the extent permitted by applicable law. The copyright holders and/or other
  parties provide SNNS 'as is' without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantabil
  and fitness for a particular purpose. The entire risk as to the quality and performance of SNNS is with you. Should the program prove defective, you assu
  the cost of all necessary servicing, repair or correction.
- In no event will any copyright holder, or any other party who may redistribute SNNS as permitted above, be liable to you for damages, including any gen
  special, incidental or consequential damages arising out of the use or inability to use SNNS (including but not limited to loss of data or data being rendere
  inaccurate or losses sustained by you or third parties or a failure of SNNS to operate with any other programs), even if such holder or other party has bee
  advised of the possibility of such damages.

CalPred also uses EMBOSSWIN package, blastpgp program from NCBI and some components from cygwin. Both the EMBOSSWIN and Cygwin projects
licensed under GPL/LGPL and GPL licenses respectively.Documentation and usage information for such components reflects these additional copyrights held by
their respective owners.

About CalPred | Application | Downloads | License | User docs | Get Involved | Credits

SCREENSHOT 5(USER DOCUMENTATIONS)


CalPred
About CalPred | Application | Downloads | License | User docs | Get Involved | Credits

USER DOCUMENTATION*.

Supplementary material (SM).

SM 1: Sequence identifiers of datasets used in this study can be found here (28.5 Kb).

SM 2: Datafiles for results of analysis done on whole proteomes taken from four organisms can be found here (1.89 Mb).

SM 3: Protein properties used from PEPSTATS and their associated normalization factors can be found here (35 Kb).

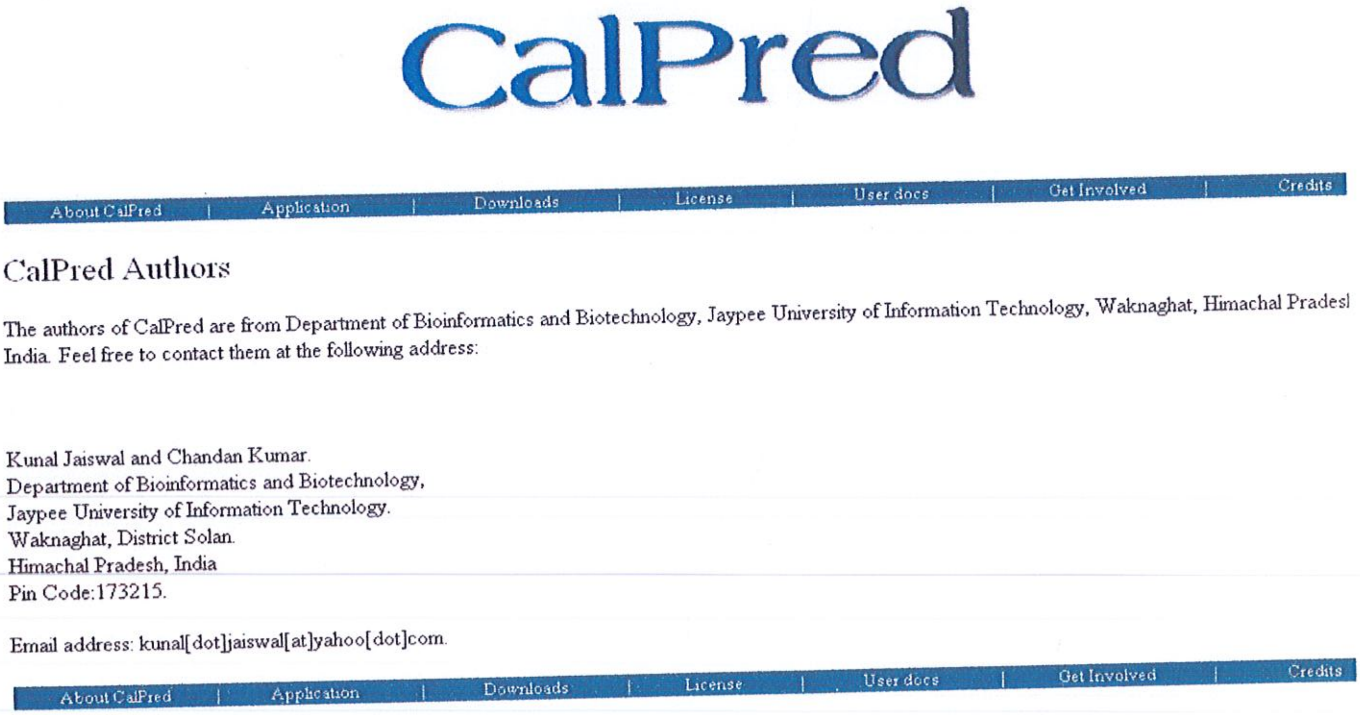
SM 4: First (2.95 Kb) and second-level (203 Kb) filter results for PS00018 dataset.

SM 5: Plots for distribution of scores for proteins taken from four proteomes and PS00018 dataset queried with first-level filter can be found here (88.2 Mb).

***More on user documentation will be updated soon.**

About CalPred | Application | Downloads | License | User docs | Get Involved | Credits

SCREENSHOT 6 (CREDITS)


CalPred
About CalPred | Application | Downloads | License | User docs | Get Involved | Credits

CalPred Authors

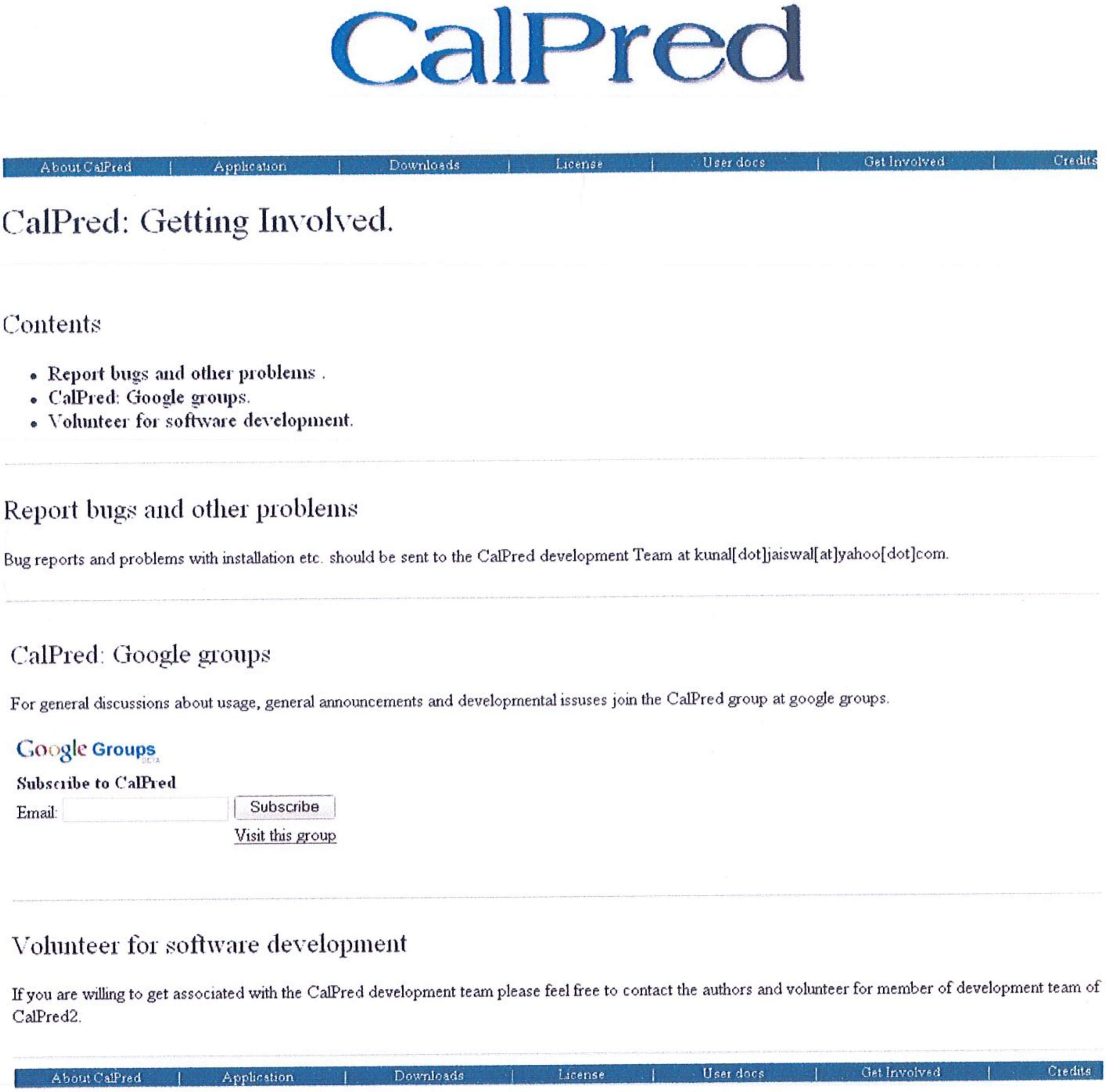
The authors of CalPred are from Department of Bioinformatics and Biotechnology, Jaypee University of Information Technology, Waknaghat, Himachal Prades India. Feel free to contact them at the following address:

Kunal Jaiswal and Chandan Kumar.
Department of Bioinformatics and Biotechnology,
Jaypee University of Information Technology.
Waknaghat, District Solan.
Himachal Pradesh, India
Pin Code:173215.

Email address: kunal[dot]jaiswal[at]yahoo[dot]com.

About CalPred | Application | Downloads | License | User docs | Get Involved | Credits

SCREENSHOT 7(GETTING INVOLVED)


CalPred
About CalPred | Application | Downloads | License | User docs | Get Involved | Credits

CalPred: Getting Involved.

Contents

- **Report bugs and other problems .**
- **CalPred: Google groups.**
- **Volunteer for software development.**

Report bugs and other problems

Bug reports and problems with installation etc. should be sent to the CalPred development Team at kunal[dot]jaiswal[at]yahoo[dot]com.

CalPred: Google groups

For general discussions about usage, general announcements and developmental issuses join the CalPred group at google groups.

Subscribe to CalPred

Email:

Volunteer for software development

If you are willing to get associated with the CalPred development team please feel free to contact the authors and volunteer for member of development team of CalPred2.

About CalPred | Application | Downloads | License | User docs | Get Involved | Credits

APPENDIX-I

PROGRAMS:

ANN

```
package ANN;

use strict;
use FileHandle;

sub new # constructor sub-routine
{
my $pattern={ sequence => '',
            };

bless ($pattern);
return $pattern;
}

sub setANN
{
      if(@_==2){
      my $self=shift();
      $self->sequence_method($_[0]);
      }

      else{
      print "Method setPattern_maker requires a argument\n";
      }
}

sub sequence_method
{

my $self=shift();
my $filename=shift() if(@_);
my $read = new FileHandle;

$read->open($filename) or
      die ("Could not open $filename");
while ( my $line = $read->getline() )
{
if($line=~ /^>.*/){next;} # for fasta line
elsif($line=~/^\s$/){next;} # for spaces
else{$self->{sequence}=$self->{sequence}.$line;}
}
$self->{sequence}=~ s/\s//gi;
return $self->{sequence};

}

sub print
{
my $self=shift();
```

```
print "The sequence is:\n".$self->{sequence};
}


sub pepstats
{
my $self=shift();
my $filename=shift() if(@_);
my $cmd="pepstats $filename pepstats_outfile.temp -auto 1";
my $out=`$cmd`;
my $read = new FileHandle;

$read->open("pepstats_outfile.temp") or
     die ("Could not open pepstats_outfile");
my @vector=();

while ( my $line = $read->getline() )
{
my @array=();
@array=split(' ',$line);
chomp(@array);

if($line=~ /Molecular.weight/)
{push(@vector,$array[3]/100000);}      #scaling the features and making the
classification vector

if($line=~ /Average/)
{push(@vector,$array[4]/1000);}

if($line=~ /Isoelectric/)
{push(@vector,$array[3]/10);}

if($line=~ /A280.Extinction/)
{push(@vector,$array[5]/100000);}

if( ($line=~ /A.=.Ala/) || ($line=~ /C.=.Cys/) || ($line=~ /D.=.Asp/)||
($line=~ /E.=.Glu/) || ($line=~ /F.=.Phe/) || ($line=~ /G.=.Gly/) ||
($line=~ /H.=.His/) || ($line=~ /I.=.Ile/) || ($line=~ /K.=.Lys/) ||
($line=~ /L.=.Leu/) || ($line=~ /M.=.Met/) || ($line=~ /N.=.Asn/) ||
($line=~ /P.=.Pro/) || ($line=~ /Q.=.Gln/) || ($line=~ /R.=.Arg/) ||
($line=~ /S.=.Ser/) || ($line=~ /T.=.Thr/) || ($line=~ /V.=.Val/) ||
($line=~ /W.=.Trp/) )
{push(@vector,$array[4]/10);
push(@vector,$array[5]);}

if($line=~ /Tiny/)
{push(@vector,$array[3]/100);}

if($line=~ /Small/)
{push(@vector,$array[3]/100);}

if($line=~ /Aliphatic/)
{push(@vector,$array[3]/100);}

if($line=~ /Aromatic/)
{push(@vector,$array[3]/100);}
```

```
if($line=~ /Non-polar/)
{push(@vector,$array[3]/100);}

if($line=~ /Polar/)
{push(@vector,$array[3]/100);}

if($line=~ /Charged/)
{push(@vector,$array[3]/100);}

if($line=~ /Basic/)
{push(@vector,$array[3]/100);}

if($line=~ /Acidic/)
{push(@vector,$array[3]/100);}

}#while

my $write= new FileHandle;

$write->open(">infile_pepstats_ann.temp") or
            die( "Could not open to write");

$write->autoflush(1);
$write->print("#Input_pattern_1:\n@vector");
my $command = "ann_pepstats.exe";
#my $output=`$command`;
system($command);
}



sub binary
{
my $self=shift();
my $seqe=$self->{sequence};
my $seq="XXXXXX".$seqe."XXXXXX";
my $offset=length($seq)-13;
my $write= new FileHandle;
$write->open(">infile_binary_ann.temp") or
            die( "Could not open to write");
$write->autoflush(1);
my $count=1;

foreach my $index(0..$offset)
{
my $subseq=substr($seq,$index,13);
$write->print("#Input_pattern_$count:\n");
my @inputs=();

foreach(0..12)
{
my $base=substr($subseq,$_,1);
my @input=qw(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0);
my @bases=qw(G A L M F W K Q E S P V I C Y H R N D T);
      foreach(0..19)
      {
            if($bases[$_] eq $base)
            {$input[$_]=1;}
      }#foreach 19
push(@inputs, @input);
}#foreach 12
$write->print("@inputs\n");
$count++;
}# foreach offset

my $cmd='ann_binary.exe';
my $out=`$cmd`;

my $len=length($seq)-12;
my $result='X';
foreach(1..($len-1))
{$result=$result.'X';}
my $count=0;
my @indices=();


my $read = new FileHandle;
$read->open("outfile_binary_ann.temp") or
      die ("Could not open outfile_binary_ann");
while ( my $line = $read->getline() )
{
      if($line)
      {
      chomp($line);
            if($line >= 0.5)
            {
            push(@indices,$count);
            }
      $count++;
      }
}

foreach(@indices)
{substr($result,$_,1,'C');}

my $final=substr($result,0,$len);

my $write= new FileHandle;
$write->open(">result_binary_ann.temp") or
die( "Could not open to write");
$write->autoflush(1);
$write->print(">Sequence\n$seqe\n>Binary ANN\n$final\n");
}

sub pssm
{
my $self=shift();
my $filename=shift() if(@_);
my $cmd="blastpgp -j 3 -d calcium -i $filename -Q pssm_matrix.temp";
my $out=`$cmd`;

my $zero='0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0';

my @array=();
my @pssmarray=();

foreach(1..6)
{push(@array,$zero);}

my $read = new FileHandle;
$read->open("pssm_matrix.temp") or
      die ("Could not open pssm_matrix");
while ( my $line = $read->getline() )
{
@pssmarray=();
@pssmarray=split(' ',$line);
my $twenty = '';
      if($pssmarray[0]=~ /\d/)
      {
      $twenty = '';
      foreach(2..21)
      {$twenty=$twenty." ".$pssmarray[$_];}
      }
if($twenty)
{push(@array,$twenty);}
}#while

foreach(1..6)
{push(@array,$zero);}

my $write= new FileHandle;
$write->open(">infile_pssm_ann.temp") or
die( "Could not open to write");
$write->autoflush(1);
chomp(@array);

my $l=scalar(@array);
$l=$l-12;
my $counter=0;
my $lcounter=0;
my $index=1;

foreach(1..$l)
{
$lcounter=$counter+13;
$write->print("\n#Input_pattern_$index:\n");
      foreach($counter..$lcounter)
      {$write->print("$array[$_] ");}
$counter++;
$index++;
}

my $cmd = 'ann_pssm.exe';
my $out = `$cmd`;
my $result='X';
foreach(1..($l-1))
{$result=$result.'X';}
my $count=0;
my @indices=();
my $read = new FileHandle;
$read->open("outfile_pssm_ann.temp") or
      die ("Could not open outfile_pssm_ann");
while ( my $line = $read->getline() )
{
      if($line)
      {
      chomp($line);
            if($line >= 0.5)
            {
            push(@indices,$count);
            }
      $count++;
      }
}
foreach(@indices)
{substr($result,$_,1,'C');}

my $final=substr($result,0,$l);

my $write= new FileHandle;
$write->open(">result_pssm_ann.temp") or
die( "Could not open to write");
$write->autoflush(1);
$write->print(">Sequence\n$self->{sequence}\n>PSSM ANN\n$final\n");

}
return 1;
```

SVM

```
package SVM;

use strict;
use FileHandle;

sub new  # constructor sub-routine
{
my $pattern={ sequence => '',
            };
bless ($pattern);
return $pattern;
}

sub setSVM
{
      if(@_==2){
      my $self=shift();
      $self->sequence_method($_[0]);
      }
      else{
      print "Method setPattern_maker requires a argument\n";
      }
}
```

```
sub sequence_method
{
my $self=shift();
my $filename=shift() if(@_);
my $read = new FileHandle;

$read->open($filename) or
     die ("Could not open $filename");
while ( my $line = $read->getline() )
{
if($line=~ /^>.*/){next;} # for fasta line
elsif($line=~/^\s$/){next;} # for spaces
else{$self->{sequence}=$self->{sequence}.$line;}
}
$self->{sequence}=~ s/\s//gi;
return $self->{sequence};
}

sub print
{
my $self=shift();
print "The sequence is:\n".$self->{sequence};
}


sub pepstats
{
my $self=shift();
my $filename=shift() if(@_);
my $modelname=shift() if(@_);
my $cmd="pepstats $filename pepstats_outfile.temp -auto 1";
my $out=`$cmd`;
my $read = new FileHandle;

$read->open("pepstats_outfile.temp") or
     die ("Could not open pepstats_outfile");
my @vector=();

while ( my $line = $read->getline() )
{
my @array=();
@array=split(' ',$line);
chomp(@array);

if($line=~ /Molecular.weight/)
{push(@vector,$array[3]/100000);}     #scaling the features and making the
classification vector

if($line=~ /Average/)
{push(@vector,$array[4]/1000);}

if($line=~ /Isoelectric/)
{push(@vector,$array[3]/10);}

if($line=~ /A280.Extinction/)
{push(@vector,$array[5]/100000);}
if( ($line=~ /A.=.Ala/) || ($line=~ /C.=.Cys/) || ($line=~ /D.=.Asp/)||
($line=~ /E.=.Glu/) || ($line=~ /F.=.Phe/) || ($line=~ /G.=.Gly/) ||
($line=~ /H.=.His/) || ($line=~ /I.=.Ile/) || ($line=~ /K.=.Lys/) ||
($line=~ /L.=.Leu/) || ($line=~ /M.=.Met/) || ($line=~ /N.=.Asn/) ||
($line=~ /P.=.Pro/) || ($line=~ /Q.=.Gln/) || ($line=~ /R.=.Arg/) ||
($line=~ /S.=.Ser/) || ($line=~ /T.=.Thr/) || ($line=~ /V.=.Val/) ||
($line=~ /W.=.Trp/) )
{push(@vector,$array[4]/10);
push(@vector,$array[5]);}

if($line=~ /Tiny/)
{push(@vector,$array[3]/100);}

if($line=~ /Small/)
{push(@vector,$array[3]/100);}

if($line=~ /Aliphatic/)
{push(@vector,$array[3]/100);}

if($line=~ /Aromatic/)
{push(@vector,$array[3]/100);}

if($line=~ /Non-polar/)
{push(@vector,$array[3]/100);}

if($line=~ /Polar/)
{push(@vector,$array[3]/100);}

if($line=~ /Charged/)
{push(@vector,$array[3]/100);}

if($line=~ /Basic/)
{push(@vector,$array[3]/100);}

if($line=~ /Acidic/)
{push(@vector,$array[3]/100);}

}#while

my $write= new FileHandle;

$write->open(">infile_pepstats_svm.temp") or
            die( "Could not open to write");

$write->autoflush(1);
my $count=1;
$write->print("#Testing dataset for SVM\n");
$write->print("0 ");
foreach(@vector)
{
$write->print("$count:$_ ");
$count++;
}
$write->print("\n");
my $command = "svm_classify infile_pepstats_svm.temp $modelname
outfile_pepstats_svm.temp";
```

```
my $output=`$command`;
}


sub binary
{
my $self=shift();
my $seqe=$self->{sequence};
my $seq="XXXXXX".$seqe."XXXXXX";
my $offset=length($seq)-13;
my $write= new FileHandle;
$write->open(">infile_binary_svm.temp") or
            die( "Could not open to write");
$write->autoflush(1);
my $count=1;
my $ccount=1;
$write->print("#Testing dataset for SVM\n");

foreach my $index(0..$offset)
{
my $subseq=substr($seq,$index,13);
my @inputs=();

foreach(0..12)
{
my $base=substr($subseq,$_,1);
my @input=qw(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0);
my @bases=qw(G A L M F W K Q E S P V I C Y H R N D T);
      foreach(0..19)
      {
            if($bases[$_] eq $base)
            {$input[$_]=1;}
      }#foreach 19
push(@inputs, @input);
}#foreach 12
$write->print("0 ");
foreach(@inputs)
{
$write->print("$ccount:$_ ");
$ccount++;
}
$write->print("\n");
$ccount=1;
$count++;
}# foreach offset

my $cmd='svm_classify infile_binary_svm.temp calcium_linear_binary
outfile_binary_svm.temp';
my $out=`$cmd`;

my $len=length($seq)-12;
my $result='X';
foreach(1..($len-1))
{$result=$result.'X';}
my $count=0;
my @indices=();
my $read = new FileHandle;
$read->open("outfile_binary_svm.temp") or
        die ("Could not open outfile_binary_svm");
while ( my $line = $read->getline() )
{
        if($line)
        {
        chomp($line);
                if($line >= 0)
                {
                push(@indices,$count);
                }
        $count++;
        }
}

foreach(@indices)
{substr($result,$_,1,'C');}

my $final=substr($result,0,$len);

my $write= new FileHandle;
$write->open(">result_binary_svm.temp") or
die( "Could not open to write");
$write->autoflush(1);
$write->print(">Sequence\n$seqe\n>Binary SVM\n$final\n");
}

sub pssm
{
my $self=shift();
my $filename=shift() if(@_);
my $cmd="blastpgp -j 3 -d calcium -i $filename -Q pssm_matrix.temp";
my $out=`$cmd`;

my $zero='0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0';

my @array=();
my @pssmarray=();

foreach(1..6)
{push(@array,$zero);}

my $read = new FileHandle;
$read->open("pssm_matrix.temp") or
        die ("Could not open pssm_matrix");
while ( my $line = $read->getline() )
{
@pssmarray=();
@pssmarray=split(' ',$line);
my $twenty = '';

        if($pssmarray[0]=~ /\d/)
        {
        $twenty = '';
        foreach(2..21)
```

```
      {$twenty=$twenty." ".$pssmarray[$_];}
      }
if($twenty)
{push(@array,$twenty);}
}#while

foreach(1..6)
{push(@array,$zero);}

my $write= new FileHandle;
$write->open(">infile_pssm_svm.temp") or
die( "Could not open to write");
$write->autoflush(1);
chomp(@array);
$write->print("#Testing dataset for SVM\n");

my $l=scalar(@array);
$l=$l-12;
my $counter=0;
my $lcounter=0;
my $index=1;

foreach(1..$l)
{
$lcounter=$counter+13;
      $write->print("0 ");
      my $newcount=1;

      foreach($counter..$lcounter)
      {
      my @newarray=();
      @newarray=split(' ',$array[$_]);
            foreach(@newarray)
            {
            $write->print("$newcount:$_ ");
            $newcount++;

            }

      }
      $write->print("\n");

$counter++;
$index++;
}

my $cmd='svm_classify infile_pssm_svm.temp calcium_linear_pssm
outfile_pssm_svm.temp';
my $out = `$cmd`;
my $result='X';
foreach(1..($l-1))
{$result=$result.'X';}
my $count=0;
my @indices=();

my $read = new FileHandle;
$read->open("outfile_pssm_svm.temp") or
```

```
        die ("Could not open outfile_pssm_svm");
while ( my $line = $read->getline() )
{
        if($line)
        {
        chomp($line);
                if($line >= 0.5)
                {
                push(@indices,$count);
                }
        $count++;
        }
}

foreach(@indices)
{substr($result,$_,1,'C');}

my $final=substr($result,0,$l);

my $write= new FileHandle;
$write->open(">result_pssm_svm.temp") or
die( "Could not open to write");
$write->autoflush(1);
$write->print(">Sequence\n$self->{sequence}\n>PSSM SVM\n$final\n");

}

return 1;
```

PATTERN

```
package pattern;

use strict;
use warnings;
use FileHandle;

sub new  # constructor sub-routine
{
my $pat={ sequence => '',
              motif=>'',};

bless ($pat);
return $pat;
}


sub setpattern
{
        if(@_==2){
        my $self=shift();
        $self->sequence_method($_[0]);
        $self->motif_method();
        }
```

```
      else{
      print "Method setPattern_maker requires a argument\n";
      }
}


sub sequence_method
{

my $self=shift();
my $filename=shift() if(@_);
my $read = new FileHandle;

$read->open($filename) or
      die ("Could not open $filename");
while ( my $line = $read->getline() )
{
chomp($line);
if($line=~ /^>.*/){next;} # for fasta line
elsif($line=~/^\s$/){next;} # for spaces
else{$self->{sequence}=$self->{sequence}.$line;}
}

$self->{sequence}=~ s/\s//gi;
$self->{sequence}=~ s/\n//gi;

return $self->{sequence};

}

sub motif_method
{

my $self=shift();
$self-
>{motif}='D[^W][DNS][^ILVFYW][DENSTG][DNQGHRK][^GP][LIVMC][DENQSTAGC]..[DE
][LIVMFYW]';
return $self->{motif};

}

sub pattern
{
my $self=shift();
my $filename=shift() if(@_);
my $seq=$self->{sequence};

my $index=0;
my @indices=();
my $fragment=$self->{motif};
my $len=length($seq);

while(substr($seq,$index,13))
{
$index++;
$_=substr($seq,$index,13);
      if($_=~ /$fragment/)
      {
      push(@indices,$index);
      }
}

my $result='X';
foreach(1..($len-1))
{$result=$result.'X';}

foreach(@indices)
{substr($result,$_,13,'CCCCCCCCCCCCC');}

my $write= new FileHandle;
$write->open(">result_pattern.temp") or
die( "Could not open to write");
$write->autoflush(1);
$write->print(">Sequence\n$seq\n>Prosite scan\n$result\n");

}

return 1;
```

PRINTER

```
package printer;

use strict;
use FileHandle;

sub new  # constructor sub-routine
{
my $print={ filename => '',
            method=>'',};

bless ($print);
return $print;
}

sub setprinter
{
      if(@_==2){
      my $self=shift();
      $self->file_method($_[0]);
      $self->method_method($_[0]);
      }

      else{
      print "Method setPattern_maker requires a argument\n";
      }
}

sub file_method
```

```
{
my $self=shift();
$self->{filename}=shift() if(@_);
return $self->{filename};
}

sub method_method
{
my $self=shift();
my $file=shift() if(@_);
my $method='';

if($file eq 'result_pattern.temp')
{$method='Prosite scan';}
if($file eq 'result_binary_ann.temp')
{$method='ANN using Binary model';}
if($file eq 'result_pssm_ann.temp')
{$method='ANN using PSSM model';}
if($file eq 'result_binary_svm.temp')
{$method='SVM using Binary model';}
if($file eq 'result_pssm_svm.temp')
{$method='SVM using PSSM model';}
$self->{method}=$method;
return $self->{method};
}

sub print {

my $self=shift();
my $method=$self->{method};
my $read = new FileHandle;
my @results=();
my @file=();
my $count=0;
my $flag=0;
my $blankline=0;
my $fastaline=0;
my @fasta=();
my $seq='';

$read->open($self->{filename}) or
        die ("Could not open $self->{filename}");
while ( my $line = $read->getline() )
{
$count++;
 if($line=~/^>.*/)
{$fastaline++; push(@fasta,$line); if($flag!=0){push(@file,$seq)} $seq="";
next;}
 elsif($line=~/^\s$/)
{$blankline++; next;}
else{$seq=$seq.$line;}
$flag++;
}
if( ($flag+$blankline+$fastaline)==$count){ push(@file,$seq);}
push(@results,'<table border ="1"><tr>');
my $index=0;
my $seq=$file[0];
my $result=$file[1];
my $len =length($file[0]);
my $round=$len/60;
$round++;

foreach(1..$round)
{
my $protein=substr($seq,$index,60);
my $patt=substr($result,$index,60);
##print "$index\n$protein\n$patt\n";
push(@results,'<td bgcolor="white">Sequence</td><td
bgcolor="white">'.$protein.'</td></tr><tr><td
bgcolor="#CC6666">'.$method.'</td><td
bgcolor="#CC6666">'.$patt.'</td></tr>');
$index=$index+60;
}
push(@results,'</table>');

my $write= new FileHandle;
$write->open(">result.temp") or
die( "Could not open to write");
$write->autoflush(1);
$write->print(@results);
}

sub multiprint
{
my $self=shift();

my $one=shift() if(@_);
my $two=shift() if(@_);
my $three=shift() if(@_);
my $four=shift() if(@_);
my $five=shift() if(@_);

$self->file_method($one);
$self->method_method($one);
my $onemethod=$self->{method};
my $read = new FileHandle;
my @results=();
my @file=();
my $count=0;
my $flag=0;
my $blankline=0;
my $fastaline=0;
my @fasta=();
my $seq='';
$read->open($self->{filename}) or
      die ("Could not open $self->{filename}");
while ( my $line = $read->getline() )
{
$count++;
 if($line=~/^>.*/)
{$fastaline++; push(@fasta,$line); if($flag!=0){push(@file,$seq)} $seq="";
next;}
```

```
 elsif($line=~/^\s$/)
{$blankline++; next;}
else{$seq=$seq.$line;}
$flag++;
}
if( ($flag+$blankline+$fastaline)==$count){ push(@file,$seq);}

$self->file_method($two);
$self->method_method($two);
my $twomethod=$self->{method};
$read = new FileHandle;
$count=0;
$flag=0;
$blankline=0;
$fastaline=0;
@fasta=();
$seq='';
$read->open($self->{filename}) or
      die ("Could not open $self->{filename}");
while ( my $line = $read->getline() )
{
$count++;
 if($line=~/^>.*/)
{$fastaline++; push(@fasta,$line); if($flag!=0){push(@file,$seq)} $seq="";
next;}
 elsif($line=~/^\s$/)
{$blankline++; next;}
else{$seq=$seq.$line;}
$flag++;
}
if( ($flag+$blankline+$fastaline)==$count){ push(@file,$seq);}

$self->file_method($three);
$self->method_method($three);
my $threemethod=$self->{method};
$read = new FileHandle;
$count=0;
$flag=0;
$blankline=0;
$fastaline=0;
@fasta=();
$seq='';
$read->open($self->{filename}) or
      die ("Could not open $self->{filename}");
while ( my $line = $read->getline() )
{
$count++;
 if($line=~/^>.*/)
{$fastaline++; push(@fasta,$line); if($flag!=0){push(@file,$seq)} $seq="";
next;}
 elsif($line=~/^\s$/)
{$blankline++; next;}
else{$seq=$seq.$line;}
$flag++;
}
if( ($flag+$blankline+$fastaline)==$count){ push(@file,$seq);}
```

```
$self->file_method($four);
$self->method_method($four);
my $fourmethod=$self->{method};
$read = new FileHandle;
$count=0;
$flag=0;
$blankline=0;
$fastaline=0;
@fasta=();
$seq='';
$read->open($self->{filename}) or
      die ("Could not open $self->{filename}");
while ( my $line = $read->getline() )
{
$count++;
 if($line=~/^>.*/)
{$fastaline++; push(@fasta,$line); if($flag!=0){push(@file,$seq)} $seq="";
next;}
 elsif($line=~/^\s$/)
{$blankline++; next;}
else{$seq=$seq.$line;}
$flag++;
}
if( ($flag+$blankline+$fastaline)==$count){ push(@file,$seq);}

$self->file_method($five);
$self->method_method($five);
my $fivemethod=$self->{method};
$read = new FileHandle;
$count=0;
$flag=0;
$blankline=0;
$fastaline=0;
@fasta=();
$seq='';
$read->open($self->{filename}) or
      die ("Could not open $self->{filename}");
while ( my $line = $read->getline() )
{
$count++;
 if($line=~/^>.*/)
{$fastaline++; push(@fasta,$line); if($flag!=0){push(@file,$seq)} $seq="";
next;}
 elsif($line=~/^\s$/)
{$blankline++; next;}
else{$seq=$seq.$line;}
$flag++;
}
if( ($flag+$blankline+$fastaline)==$count){ push(@file,$seq);}

push(@results,'<table border ="1"><tr>');
my $index=0;
my $seq=$file[0];
my $oneresult=$file[1];
my $tworesult=$file[3];
my $threeresult=$file[5];
my $fourresult=$file[7];
my $fiveresult=$file[9];
my $len =length($file[0]);
my $round=$len/60;
$round++;

foreach(1..$round)
{
my $protein=substr($seq,$index,60);
my $onepatt=substr($oneresult,$index,60);
my $twopatt=substr($tworesult,$index,60);
my $threepatt=substr($threeresult,$index,60);
my $fourpatt=substr($fourresult,$index,60);
my $fivepatt=substr($fiveresult,$index,60);

push(@results,'<td bgcolor="white">Sequence</td><td
bgcolor="white">'.$protein.'</td></tr><tr><td
bgcolor="#CC6666">'.$onemethod.'</td><td
bgcolor="#CC6666">'.$onepatt.'</td></tr>');
push(@results,'<tr><td bgcolor="#99CC00">'.$twomethod.'</td><td
bgcolor="#99CC00">'.$twopatt.'</td></tr>');
push(@results,'<tr><td bgcolor="#9999CC">'.$threemethod.'</td><td
bgcolor="#9999CC">'.$threepatt.'</td></tr>');
push(@results,'<td bgcolor="#FFFFCC">'.$fourmethod.'</td><td
bgcolor="#FFFFCC">'.$fourpatt.'</td></tr>');
push(@results,'</tr><tr><td bgcolor="#CCFFCC">'.$fivemethod.'</td><td
bgcolor="#CCFFCC">'.$fivepatt.'</td></tr>');

$index=$index+60;
}
push(@results,'</table>');

my $write= new FileHandle;
$write->open(">result.temp") or
die( "Could not open to write");
$write->autoflush(1);
$write->print(@results);


}

return 1;
```

MISC. PROGRAMS

```
#!c:/perl/bin/perl.exe
#!c:/perl/lib

use lib "E:/juit_website/calpred";
use CGI qw(:standard);
use FileHandle;
use ANN;
use SVM;
use pattern;
use printer;
```

```
$ENV{'EMBOSSWIN'}="C:/EMBOSSwin";
$ENV{'EMBOSS_DATA'}="C:/EMBOSSwin/data";
$ENV{'Path'}="C:/EMBOSSwin";

print header(),start_html("Results...");
print '<body vlink="#FFFFFF" alink="#FFFFFF" bgcolor="silver">';
$sequence=param('sequence');
$option=param('technique');

print hr(),hr(),'<p align="left"><font size="5" face="Monotype
Corsiva"><b>CalPred</b>: A tool for EF-hand calcium binding
    protein prediction and calcium binding region
identification.</font></p>',hr(),hr();
print '<p align="left"><font size="4" face="Monotype Corsiva">The Ca(+2)
binding site prediction results are as follows:</font></p>';

my $write= new FileHandle;
$write->open(">sequence.temp") or
die( "Could not open to write");
$write->autoflush(1);
$write->print($sequence);


if($option eq "prosite")  # simple pattern search
{
my $instance=new pattern;
$instance->setpattern("sequence.temp");
$instance->pattern();

my $anotherinstance=new printer;
$anotherinstance->setprinter("result_pattern.temp");
$anotherinstance->print();

my $read = new FileHandle;
$read->open("result.temp") or
die ("Could not open result file");
while($result = $read->getline())
{print "$result<br>";}
}

if($option eq "binary_ann")
{
my $instance=new ANN;
$instance->setANN("sequence.temp");
$instance->binary();

my $anotherinstance=new printer;
$anotherinstance->setprinter("result_binary_ann.temp");
$anotherinstance->print();

my $read = new FileHandle;
$read->open("result.temp") or
die ("Could not open result file");
while($result = $read->getline())
{print "$result<br>";}
}
```

```
if($option eq "pssm_ann")
{
my $instance=new ANN;
$instance->setANN("sequence.temp");
$instance->pssm("sequence.temp");

my $anotherinstance=new printer;
$anotherinstance->setprinter("result_pssm_ann.temp");
$anotherinstance->print();

my $read = new FileHandle;
$read->open("result.temp") or
die ("Could not open result file");
while($result = $read->getline())
{print "$result<br>";}
}


if($option eq "binary_svm")
{

my $instance=new SVM;
$instance->setSVM("sequence.temp");
$instance->binary();

my $anotherinstance=new printer;
$anotherinstance->setprinter("result_binary_svm.temp");
$anotherinstance->print();

my $read = new FileHandle;
$read->open("result.temp") or
die ("Could not open result file");
while($result = $read->getline())
{print "$result<br>";}

}

if($option eq "pssm_svm")
{

my $instance=new SVM;
$instance->setSVM("sequence.temp");
$instance->pssm("sequence.temp");

my $anotherinstance=new printer;
$anotherinstance->setprinter("result_pssm_svm.temp");
$anotherinstance->print();

my $read = new FileHandle;
$read->open("result.temp") or
die ("Could not open result file");
while($result = $read->getline())
{print "$result<br>";}

}
```

```
if($option eq "all")
{
my $instance=new pattern;
$instance->setpattern("sequence.temp");
$instance->pattern();

my $instance=new ANN;
$instance->setANN("sequence.temp");
$instance->binary();

my $instance=new ANN;
$instance->setANN("sequence.temp");
$instance->pssm("sequence.temp");

my $instance=new SVM;
$instance->setSVM("sequence.temp");
$instance->binary();


my $instance=new SVM;
$instance->setSVM("sequence.temp");
$instance->pssm("sequence.temp");

$pat=new printer;
$pat->multiprint("result_pattern.temp", "result_binary_ann.temp",
"result_pssm_ann.temp", "result_binary_svm.temp", "result_pssm_svm.temp");

my $read = new FileHandle;
$read->open("result.temp") or
die ("Could not open result file");
while($result = $read->getline())
{print "$result<br>";}

}



print hr(),hr();

$legend='
<table width="33%" border="1">
  <tr >
    <td colspan="3"><div align="center"><em>Legends for prediction
results</em></div></td>
    </tr>
  <tr bgcolor="#FFFFFF">
    <td width="10%">
      <div align="center"><strong>X</strong></div></td>
    <td width="90%">
      <div align="left"><em>No prediction.</em></div></td>
  </tr>
  <tr bgcolor="#CC6666">
    <td>
      <div align="center"><strong>C</strong></div></td>
    <td>
```

```
      <div align="left"><em>Calcium binding region
predicted.</em></div></td>
  </tr>
</table>';

print $legend;

print br(),'<p align="right"><a style="TEXT-DECORATION: none" href =
"./">Click here to go to home</a></p>';
print end_html();
```

```
#!c:/perl/bin/perl.exe
#!c:/perl/lib

use lib "E:/juit_website/calpred";
use CGI qw(:standard);
use FileHandle;
use ANN;
use SVM;

$ENV{'EMBOSSWIN'}="C:/EMBOSSwin";
$ENV{'EMBOSS_DATA'}="C:/EMBOSSwin/data";
$ENV{'Path'}="C:/EMBOSSwin";

print header(),start_html("Results...");
print '<body link="white" vlink="white" bgcolor="silver">';
$sequence=param('sequence');
$option=param('option');
$svmkernel=param('svmkernel');
$bothkernel=param('bothkernel');

print hr(),hr(),'<p align="left"><font size="5" face="Monotype
Corsiva"><b>CalPred</b>: A tool for EF-hand calcium binding
    protein prediction and calcium binding region
identification.</font></p>',hr(),hr();
print '<p align="left"><font size="4" face="Monotype Corsiva">The Protein
statistics along with machine learning technique\'s scores are as
follows:</font></p>';

if(!$sequence)
{print 'ERROR!!!!!!!!',br(),'Please enter a sequence!!',br(); exit;}

if(!$option)
{print 'ERROR!!!!!!!!',br(),'Please select a analyzation
technique!!',br(); exit;}

my $write= new FileHandle;
$write->open(">>calpred_log_user.txt") or
die( "Could not open to write");
$write->autoflush(1);
my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = gmtime();
$mon++;
$year=$year+1900;
$fulldate=$mday."/".$mon."/".$year;
```

```
$write-
>print("\n$fulldate\t$ENV{'REMOTE_ADDR'}\t$ENV{'HTTP_X_FORWARDED_FOR'}");


my $write= new FileHandle;
$write->open(">sequence.temp") or
die( "Could not open to write");
$write->autoflush(1);
$write->print($sequence);

if($option eq "ann")
{
my $pat= new ANN;
$pat->setANN("sequence.temp");
$pat->pepstats("sequence.temp");
my $read = new FileHandle;
$read->open("outfile_pepstats_ann.temp") or
die ("Could not open outfile_pepstats_ann");
$ann_res = $read->getline();
my $write= new FileHandle;
$write->open(">>pepstats_outfile.temp") or
die( "Could not open to write");
$write->autoflush(1);
$write->print("ANN Score $ann_res");
}

if($option eq "svm")
{

if($svmkernel=~ /-/)
{print "ERROR!!!!!!!!!<br>Please select a kernel type!!"; exit;}
if($svmkernel eq "linear")
{$kernel="calcium_linear";}
elsif($svmkernel eq "polynomial")
{$kernel="calcium_polynomial";}
elsif($svmkernel eq "radial")
{$kernel="calcium_radial_bais";}
elsif($svmkernel eq "sigmoidal")
{$kernel="calcium_sigmoidal_tanh";}

my $pat= new SVM;
$pat->setSVM("sequence.temp");
$pat->pepstats("sequence.temp",$kernel);

my $read = new FileHandle;
$read->open("outfile_pepstats_svm.temp") or
die ("Could not open outfile_pepstats_svm.t");
$svm_res = $read->getline();
my $write= new FileHandle;
$write->open(">>pepstats_outfile.temp") or
die( "Could not open to write");
$write->autoflush(1);
$write->print("SVM($kernel) Score $svm_res");
}

if($option eq "both")
{
```

```
my $annpat= new ANN;
$annpat->setANN("sequence.temp");
$annpat->pepstats("sequence.temp");
my $read = new FileHandle;
$read->open("outfile_pepstats_ann.temp") or
die ("Could not open outfile_pepstats_ann");
$ann_res = $read->getline();


if($bothkernel=~ /-/)
{print "ERROR!!!!!!!!!<br>Please select a kernel type!!"; exit;}
if($bothkernel eq "linear")
{$kernel="calcium_linear";}
elsif($bothkernel eq "polynomial")
{$kernel="calcium_polynomial";}
elsif($bothkernel eq "radial")
{$kernel="calcium_radial_bais";}
elsif($bothkernel eq "sigmoidal")
{$kernel="calcium_sigmoidal_tanh";}

my $pat= new SVM;
$pat->setSVM("sequence.temp");
$pat->pepstats("sequence.temp",$kernel);

my $read = new FileHandle;
$read->open("outfile_pepstats_svm.temp") or
die ("Could not open outfile_pepstats_svm.temp");
$svm_res = $read->getline();

my $write= new FileHandle;
$write->open(">>pepstats_outfile.temp") or
die( "Could not open to write");
$write->autoflush(1);
$write->print("ANN Score $ann_res");

my $write= new FileHandle;
$write->open(">>pepstats_outfile.temp") or
die( "Could not open to write");
$write->autoflush(1);
$write->print("SVM($kernel) Score $svm_res");

}

my $read = new FileHandle;
$read->open("pepstats_outfile.temp") or
      die ("Could not open pepstats_outfile");

print '<table border="1" cellspacing="2">';

while ( my $line = $read->getline() )
{
my @array=();
@array=split(' ',$line);
chomp(@array);

if($line=~ /Molecular.weight/)
{
```

```
print Tr('<td bgcolor="#999999">',b("Molecular Weight"),'</td>',
td($array[3]), '<td bgcolor="#999999">',b("Residues"),'</td>',
td($array[6]));
}

if($line=~ /Average/)
{
print Tr('<td bgcolor="#999999">',b("Average Residue Weight"),'</td>',
td($array[4]), '<td bgcolor="#999999">',b("Charge"),'</td>',
td($array[7]));
}

if($line=~ /Isoelectric/)
{
print Tr('<td bgcolor="#999999">',b("Isoelectric Point"),'</td>',
td($array[3]));
}

if($line=~ /A280.Molar/)
{
print Tr('<td bgcolor="#999999">',b("A280 Molar Extinction
Coefficient"),'</td>', td($array[5]));
}

if($line=~ /A280.Extinction/)
{
print Tr('<td bgcolor="#999999">',b("A280 Extinction Coefficient
1mg/ml"),'</td>', td($array[5]));
}

if($line=~ /^Residue/)
{
print Tr('<td bgcolor="#999999">',b("Residue"),'</td>', '<td
bgcolor="#999999">',b("Number"),'</td>', '<td
bgcolor="#999999">',b("Mole%"),'</td>', '<td
bgcolor="#999999">',b("DayhoffStat"),'</td>');
}

if( ($line=~ /A.=.Ala/) || ($line=~ /C.=.Cys/) || ($line=~ /D.=.Asp/)||
($line=~ /E.=.Glu/) || ($line=~ /F.=.Phe/) || ($line=~ /G.=.Gly/) ||
($line=~ /H.=.His/) || ($line=~ /I.=.Ile/) || ($line=~ /K.=.Lys/) ||
($line=~ /L.=.Leu/) || ($line=~ /M.=.Met/) || ($line=~ /N.=.Asn/) ||
($line=~ /P.=.Pro/) || ($line=~ /Q.=.Gln/) || ($line=~ /R.=.Arg/) ||
($line=~ /S.=.Ser/) || ($line=~ /T.=.Thr/) || ($line=~ /V.=.Val/) ||
($line=~ /W.=.Trp/) )
{
print Tr('<td bgcolor="#999999">',b("$array[0] $array[1]
$array[2]"),'</td>', td($array[3]), td($array[4]), td($array[5]));
push(@vector,$array[5]);
}

if($line=~ /^Property/)
{
print Tr('<td bgcolor="#999999">',b("Property"),'</td>', '<td
bgcolor="#999999">',b("Residues"),'</td>', '<td
bgcolor="#999999">',b("Number"),'</td>', '<td
bgcolor="#999999">',b("Mole%"),'</td>');
```

```
}
$tiny='Tiny';
$small='Small';

if($line=~ /Tiny/)
{
print Tr('<td bgcolor="#999999">',b("$array[0]"),'</td>', '<td
bgcolor="#999999">',$array[1],'</td>', td($array[2]), td($array[3]));
}

if($line=~ /Small/)
{
print Tr('<td bgcolor="#999999">',b("$array[0]"),'</td>', '<td
bgcolor="#999999">',$array[1],'</td>', td($array[2]), td($array[3]));
}

if($line=~ /Aliphatic/)
{
print Tr('<td bgcolor="#999999">',b("$array[0]"),'</td>', '<td
bgcolor="#999999">',$array[1],'</td>', td($array[2]), td($array[3]));
}

if($line=~ /Aromatic/)
{
print Tr('<td bgcolor="#999999">',b("$array[0]"),'</td>', '<td
bgcolor="#999999">',$array[1],'</td>', td($array[2]), td($array[3]));
}

if($line=~ /Non-polar/)
{
print Tr('<td bgcolor="#999999">',b("$array[0]"),'</td>', '<td
bgcolor="#999999">',$array[1],'</td>', td($array[2]), td($array[3]));
}

if($line=~ /Polar/)
{
print Tr('<td bgcolor="#999999">',b("$array[0]"),'</td>', '<td
bgcolor="#999999">',$array[1],'</td>', td($array[2]), td($array[3]));
}

if($line=~ /Charged/)
{
print Tr('<td bgcolor="#999999">',b("$array[0]"), '</td>','<td
bgcolor="#999999">',$array[1],'</td>', td($array[2]), td($array[3]));
}

if($line=~ /Basic/)
{
print Tr('<td bgcolor="#999999">',b("$array[0]"),'</td>', '<td
bgcolor="#999999">',$array[1],'</td>', td($array[2]), td($array[3]));
}

if($line=~ /Acidic/)
{
print Tr('<td bgcolor="#999999">',b("$array[0]"),'</td>', '<td
bgcolor="#999999">',$array[1],'</td>', td($array[2]), td($array[3]));
}
```

```
if($line=~ /ANN/)
{
print Tr('<td bgcolor="#999999">',b("$array[0] $array[1]"),'</td>',
td($array[2]));
}

if($line=~ /SVM/)
{
print Tr('<td bgcolor="#999999">',b("$array[0] $array[1]"),'</td>',
td($array[2]));
}

}#while
print '</table>';

print hr(),hr();

if(($ann_res >= 0.9) or ($svm_res > 0))
{
print '<p align="left"><font size="4" face="Monotype Corsiva">Your protein
is predicted to be an EF-hand Calcium binding protein in
nature.</font></p>';
}
elsif(($ann_res < 0.9) or ($svm_res < 0))
{
print '<p align="left"><font size="4" face="Monotype Corsiva">Your protein
is predicted to be an non EF-hand Calcium binding protein in
nature.</font></p>';
print hr(),hr(),br();

}

if(($ann_res >= 0.9) or ($svm_res > 0))
{
print '<p align="left"><font size="4" face="Monotype Corsiva">Identify the
Ca(+2) ions binding region in the protein:</font></p>';

$menu='<form name="form1" method="post" action="identify.pl">
<textarea name="sequence" cols="100" rows="10">'.$sequence.'</textarea>
            <select name="technique">
          <option value="-------------------" selected>-------------------
-</option>
          <option value="prosite">Using simple pattern matching using
Prosite entry PS00018.</option>
          <option value="binary_ann">Neural Network using binary encoding
method.</option>
          <option value="pssm_ann">Neural Network using PSSM matrices.
(Takes time !!)</option>
          <option value="binary_svm">Support Vector Machine using binary
encoding method. (Takes time !!)</option>
          <option value="pssm_svm">Support Vector Machine using PSSMs
matrices. (Takes time !!)</option>
          <option value="all">Using all the available methods. (Takes time
!!)</option>
        </select>
```

```
<input type="submit" name="Submit" value="Identify the Ca(+2) ion binding
regions in protein.">';

print $menu;

print hr(),hr(),br();
}

print br(),'<p align="right"><a style="TEXT-DECORATION: none" href =
"./">Click here to go to home</a></p>';
print end_html();
```

APPENDIX-II

Table for sequence identifiers of the proteins used in the study, here calcium-binding proteins are taken from EF-hand Calcium-binding protein data library whereas non calcium-binding proteins are taken from Entrez protein database.

Calcium-binding proteins identifiers (188)	*Non calcium-binding proteins identifiers (214)*
CABV_BOVIN, CABV_HUMAN, CART_RAT, CART_CHICK, ALG2_MOUSE, CAN2_CHICK, CAN1_HUMAN, CAN2_HUMAN, CAN_SCHMA, CAN3_CHICK, CAN3_RAT, CANS_HUMAN, GRAN_HUMAN, SORC_SCHJA, SORC_HUMAN, CDP1_ARATH, CDP1_ORYSA, CDP3_ORYSA, CDPK_SOYBN, CDPK_DAUCA, CDP2_ORYSA, CATR_ATRNU, CATR_CHLRE, CATR_DUNSA, CATR_GIALA, CAT1_HUMAN, CATR_NAEGR, CATR_SCHDU, CAT2_HUMAN, CC31_YEAST, CAT3_PARTE, CALM_ACHKL, CALM_MEDSA, CALM_CANAL, CALM_CHLRE, CALM_DICDI, CALM_EMENI, CALM_EUGGR, CALM_KLULA, CALF_NAEGR, CALM_PARTE, CALM_PLAFA, CALM_PNECA, CALM_SCHPO, CALL_CAEEL, CALL_HUMAN, CABO_LOLPE, TPCC_CHICK, TPC1_BALNU, TPC1_PONLE, TPC1_DROME, TPC1_HOMAM, TPC2_BALNU, TPC2_PONLE, TPC2_DROME, TPCS_PRODO, TPCS_RANES, TPCS_MELGA, TPC_BRALA, TPC_HALRO, TPC_TACTR, TPC_PATYE, MLEC_CHICK, MLEF_HUMAN, MLEL_DROPS, MLEP_DROSI, MLE_CAEEL, MLE_BRAFL, MLEX_CHICK, MLE_DICDI, MLE_HALRO, MLEN_HUMAN, MLE2_DROME, MLE_TODPA, MLE_PATYE, MLE1_CHICK, MLE1_HUMAN, MLEV_HUMAN, MLEY_HUMAN, MLES_CHICK, MLR_HALRO, MLR_CHLNI, MLR1_CAEEL, MLR_LUMTE, MLR_DICDI, MLR_TODPA, MLR_SPISA, MLR_PHYPO, MLR_DROME, MLRB_RAT, MLRB_CHICK, MLRA_PATYE, MLRA_CHICK, MLRT_RABIT, MLRN_DROME, MLRV_RAT, MLRS_CHICK, MLR5_HUMAN, HIPP_HUMAN, NCS1_CAEEL, NCS1_YEAST, NCS1_RAT, RECO_BOVIN, RECO_HUMAN, SMOD_RANCA, SPRC_CHICK, SPRC_CAEEL, SPRC_BOVIN, SPRC_XENLA, CALB_HUMAN, CALB_DROME, CALB_NAEGR, P01257, CALB_NEUCR, CALB_YEAST, SCPB_PENSP, SCPA_PENSP, SCP2_BRALA, SCP1_PONLE, SCP_NERDI, SCP_PATYE, SCP_PERVT, SPCN_CHICK, SPCA_DROME, SPCA_HUMAN,	AAR97543.1, AC016780_6, ABL75274.1, CAA54397.1, CAA92110.1, CAA97845.1, CAA96526.1, AAQ19039.1, AAQ19038.1, AAF78062.1, BAA37150.1, CAA78414.1, AAU11328.1, CAA47017.1, AAW80518.1, CAA41250.1, CAA47473.1, ABJ98330.1, CAL40958.1, 1814452D, AAB19835.2, AAP32204.1, AF399915_1, GRP2_SORBI, ABD78745.1, ABB78077.1, CAC40685.1, AF216868_1, AAS60207.1, XP_711307.1, CAA56641.1, AF465267_1, AF465266_1, AF465265_1, AAL66769.1, AAK30124.1, ABL96308.1, ABE98705.1, ABE98703.1, ABE98700.1, CAB46084.1, AAK59929.1, CAA48298.1, AF138704_1, 1EKJH, AAS94330.1, AAS94329.1, ABL63815.1, ABH85388.1, CAC80990.1, CAC80988.1, AAQ75613.1, ABJ90287.1, ABC49719.1, 1Z1FA, AAT39430.1, AAK67829.1, ABF93402.1, ABE68722.1, ABE68720.1, CAA06309.1, CAA06217.1, ABD16181.1, AAT45474.1, CAA07236.1, CAA10128.1, CAA09457.1, AAR19041.1, AAB91464.1, AAB91463.1, AAB91462.1, ABB82946.1, AAZ66349.1, ABB91438.1, CAA85775.1, CAA64799.1, AAD05567.1, ABL74481.1, ABL74480.1, ABL74479.1, ABD66305.1, CAB94692.1, AAK27151.1, AAK27150.1, AAB52550.1, BAA23741.1, 1EB9B, 1AVBB, CAA83001.1, CAA36853.1, BAF03553.1, CAA40073.1, CAA90585.1, CAA46016.1, ABA42952.1, AF506028_23, AF506028_16, AF506028_14, AF506028_10, AF506028_9, AF506028_8, AF506028_5, AF506028_4, ABD24226.1, AAY34565.1, ABF48718.1, ABC74528.1, CAB65335.1, CAA10217.1, CAB89831.1, CAA44054.1, CAA44055.1, CAA81749.1, CAA81748.1, CAA47846.1, CAA77741.1, CAA77742.1, CAA33517.1, AAF99703.1, BAD27384.1, BAD27380.1, BAD27379.1, BAD27376.1, BAD27374.1, BAD27373.1, BAD27370.1, BAD27364.1, BAD27359.1, BAD27356.1, ABA18633.1, ABD52008.2, AAZ81424.2, ABC55266.1, AAT66041.1, AAK26164.2, AAP57675.1, AAP57674.1, AAP57673.1, BAA08910.1,

PRVA_FELCA, PRVA_GERSP, PRVA_LATCH, PRVA_HUMAN, PRVA_CAVPO, PRVA_CYPCA, PRVA_AMPME, PRVB_AMPME, PRVA_RANES, PRVA_RANCA, PRVA_RAJCL, PRVA_RABIT, PRVA_ESOLU, PRVA_TRISE, PRVB_GADCA, PRVB_BOACO, ONCO_CAVPO, PRVB_MERME, PRVB_LEUCE, PRVB_LATCH, PRVB_GRAGE, PRVB_ESOLU, PRVB_RANES, PRVB_MERBI, PRVB_OPSTA, PRVB_MERMR, PRVB_XENLA, PRV1_SALSA, S10A_HUMAN, S110_CHICK, Release, S112_BOVIN, Release, Release, S102_BOVIN, S102_HUMAN, S103_HUMAN, S104_HUMAN, S105_MOUSE, S106_CHICK, S106_HORSE, S106_HUMAN, S107_BOVIN, S107_HUMAN, S108_HUMAN, S108_MOUSE, S108_RAT, S108_BOVIN, S108_HUMAN, S108_MOUSE, S108_RABIT, S108_RAT, S10B_BOVIN, S111_CHICK, S111_HUMAN, S111_MOUSE, S111_PIG, S111_RABIT, S10E_HUMAN, S10D_BOVIN, S10D_HUMAN, S10D_PIG, S10D_RAT.	CAA84632.1, CAA33465.1, CAA41434.1, AAY33493.1, ABB91437.1, ABB91436.1, ABB91435.1, ABB91434.1, ABB85234.1, ABB85232.1, ABB85228.1, ABB85227.1, ABB85214.1, ABB85212.1, ABB85210.1, ABB85208.1, ABB85207.1, ABB85194.1, ABB85188.1, ABF81469.1, ABF81467.1, ABF81453.1, ABF81450.1, ABF81448.1, ABF81444.1, ABF81443.1, ABF81442.1, ABF81426.1, ABF81425.1, ABF81424.1, ABF81421.1, AAP20702.1, AAP20701.1, AAO43264.1, AAM88375.1, ABC59509.1, ABC59507.1, ABC59506.1, ABC59505.1, ABC59504.1, ABC59503.1, ABC59501.1, ABC59500.1, ABC59499.1, ABC59498.1, ABC59496.1, ABC59495.1, ABC59493.1, ABC59488.1, ABC59482.1, ABC59477.1, ABC59472.1, ABC59471.1, ABC59470.1, ABC59469.1, ABC59467.1, AAZ99757.1, AAZ99791.1, AAZ99790.1, AAZ99789.1, AAZ99788.1, AAZ99787.1, AAZ99786.1, AAZ99774.1, AAZ99769.1, AAZ99766.1, AAZ99763.1, AAZ99761.1, AAU29362.1, AAU29361.1, AAN73010.1, AAN73009.1, AAN73008.1, AAN73007.1, AAC31553.1

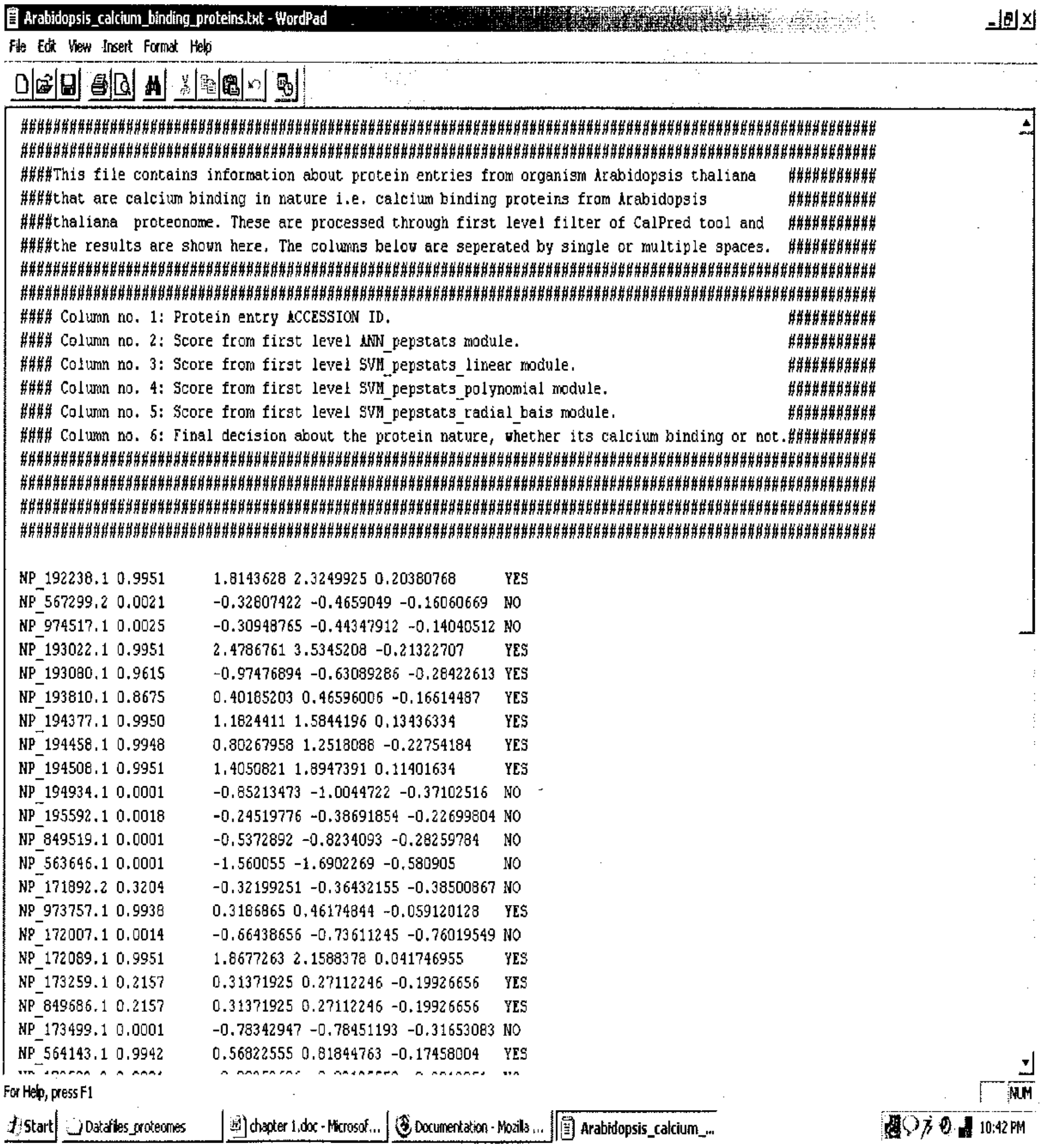
APPENDIX-III

Table for protein properties calculated using PEPSTATS and their associated normalization factors that were used in the study.

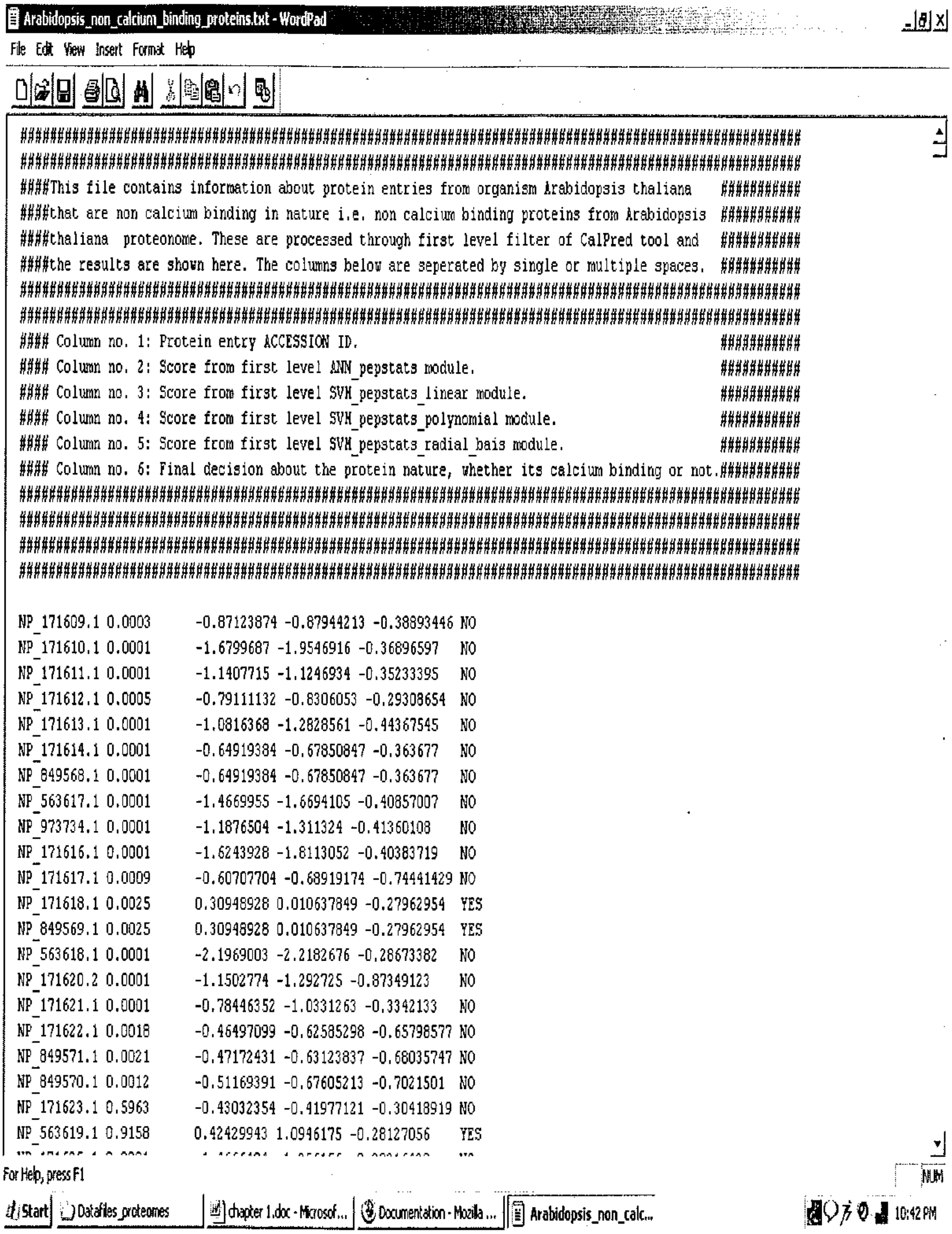
Property name	***Normalization factor***
Molecular weight.	10^5
Average Residue Weight.	10^3
Isoelectric Point.	10
A280 Extinction Coefficient 1mg/ml.	10^5
Mole % of 19 standard amino acids: Alanine, Cysteine, Aspartic Acid, Glutamic Acid, Phenylalanine, Glycine, Histidine, Isoleucine, Lysine, Leucine, Methionine, Asparagine, Proline, Glutamine, Arginine, Serine, Threonine, Valine and Tryptophan.	10
DayhoffStat of 19 standard amino acids: Alanine, Cysteine, Aspartic Acid, Glutamic Acid, Phenylalanine, Glycine, Histidine, Isoleucine, Lysine, Leucine, Methionine, Asparagine, Proline, Glutamine, Arginine, Serine, Threonine, Valine and Tryptophan.	1
Mole % of Tiny residues.	10^2
Mole % of Small residues.	10^2
Mole % of Aliphatic residues.	10^2
Mole % of Aromatic residues.	10^2
Mole % of Non-polar residues.	10^2
Mole % of Polar residues.	10^2
Mole % of Charged residues.	10^2
Mole % of Basic residues.	10^2
Mole % of Acidic residues.	10^2

APPENDIX-IV

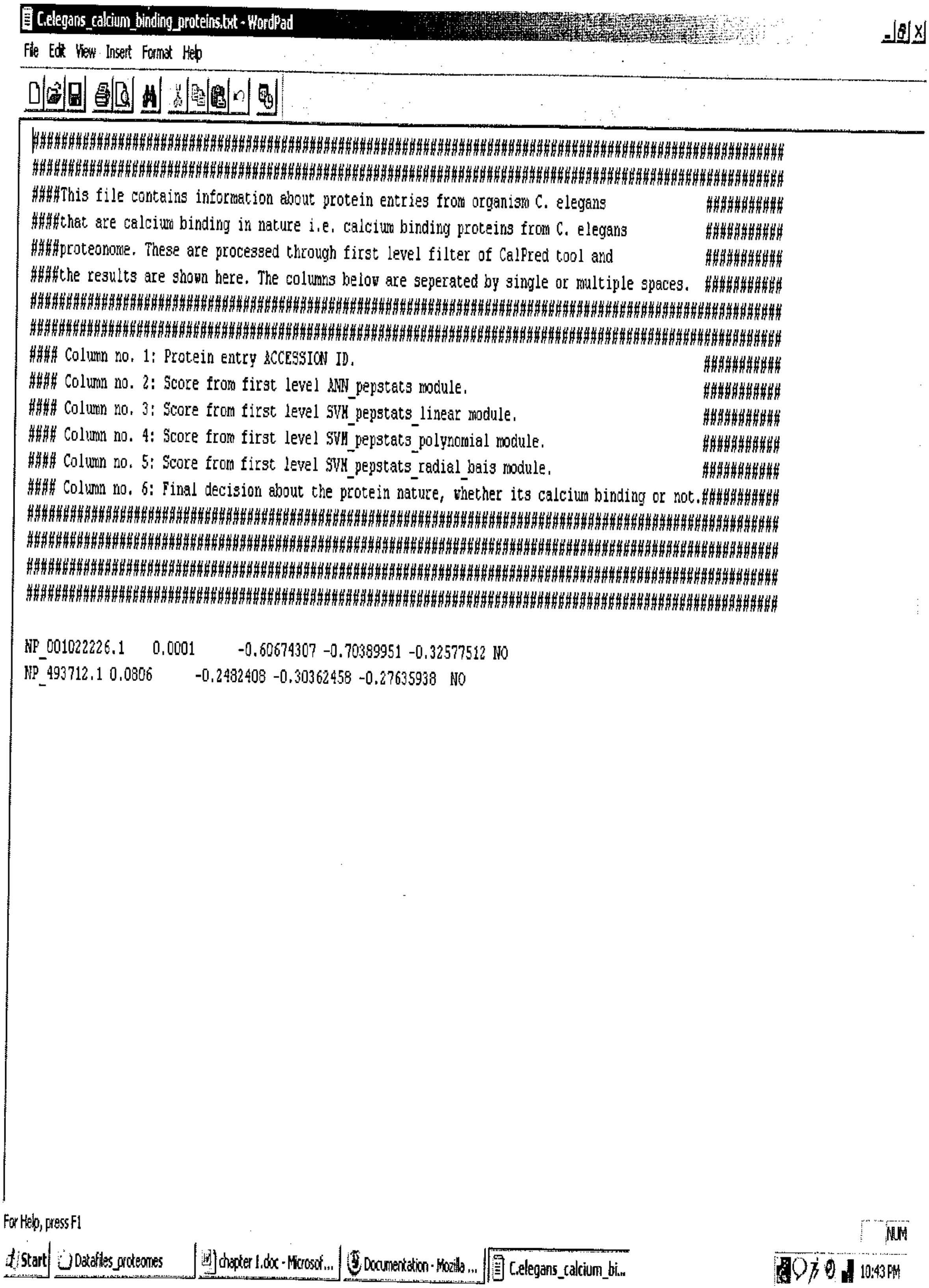
Screenshots of datafiles of the four proteasomes and the prosite dataset


Arabidopsis_calcium_binding_proteins.txt - WordPad
File Edit View Insert Format Help
For Help, press F1
NUM
Start
Datafiles_proteomes
chapter 1.doc - Microsof...
Documentation - Mozilla ...
Arabidopsis_calcium_...
10:42 PM

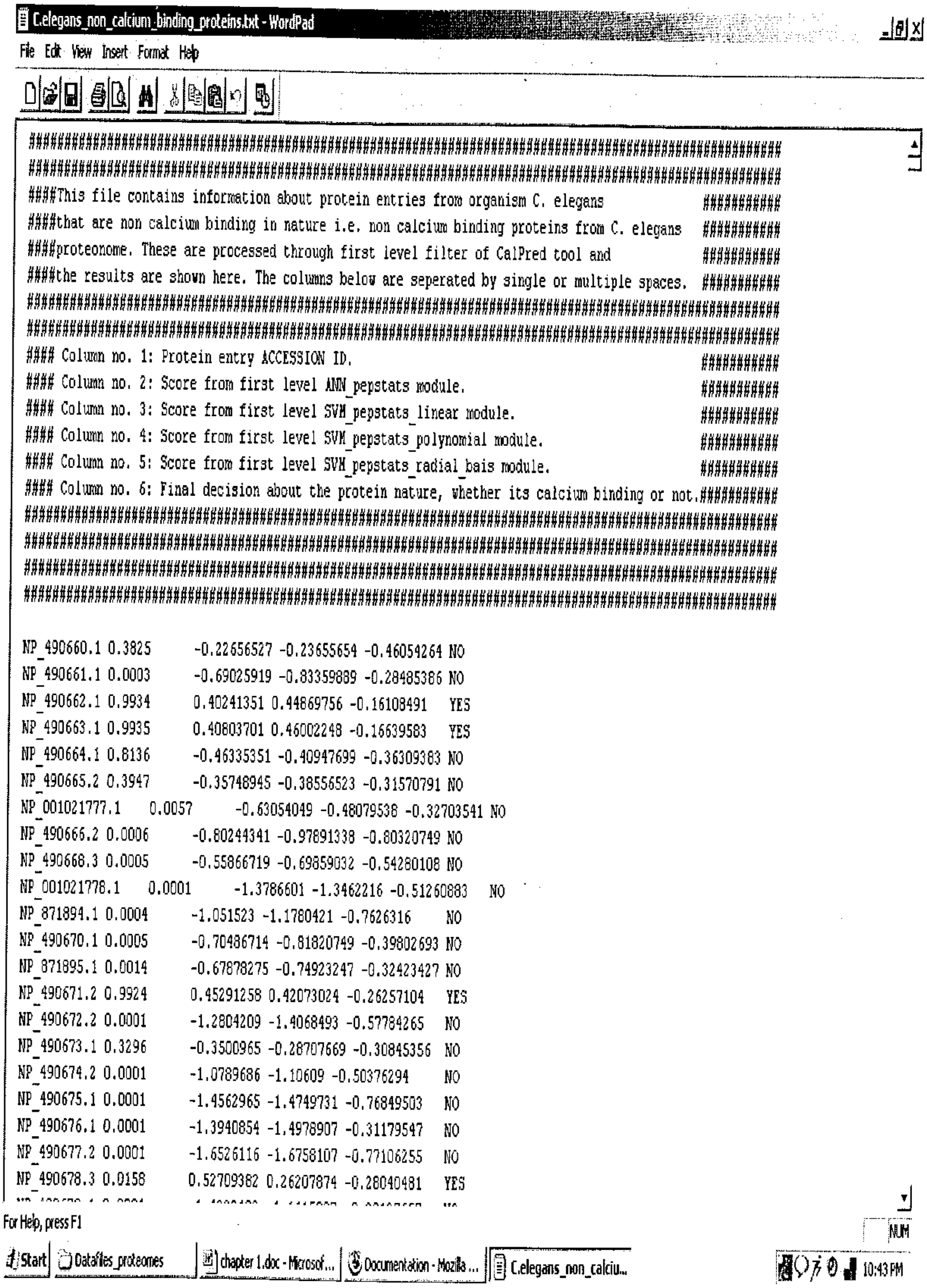
```
####################################################################################################
####################################################################################################
####This file contains information about protein entries from organism Arabidopsis thaliana      ###########
####that are calcium binding in nature i.e. calcium binding proteins from Arabidopsis            ###########
####thaliana  proteonome. These are processed through first level filter of CalPred tool and     ###########
####the results are shown here. The columns below are seperated by single or multiple spaces.   ###########
####################################################################################################
####################################################################################################
#### Column no. 1: Protein entry ACCESSION ID.                                                 ###########
#### Column no. 2: Score from first level ANN_pepstats module.                                 ###########
#### Column no. 3: Score from first level SVM_pepstats_linear module.                          ###########
#### Column no. 4: Score from first level SVM_pepstats_polynomial module.                      ###########
#### Column no. 5: Score from first level SVM_pepstats_radial_bais module.                     ###########
#### Column no. 6: Final decision about the protein nature, whether its calcium binding or not.###########
####################################################################################################
####################################################################################################
####################################################################################################
####################################################################################################

NP_192238.1 0.9951      1.8143628 2.3249925 0.20380768      YES
NP_567299.2 0.0021      -0.32807422 -0.4659049 -0.16060669  NO
NP_974517.1 0.0025      -0.30948765 -0.44347912 -0.14040512 NO
NP_193022.1 0.9951      2.4786761 3.5345208 -0.21322707     YES
NP_193080.1 0.9615      -0.97476894 -0.63089286 -0.28422613 YES
NP_193810.1 0.8675      0.40185203 0.46596006 -0.16614487   YES
NP_194377.1 0.9950      1.1824411 1.5844196 0.13436334      YES
NP_194458.1 0.9948      0.80267958 1.2518088 -0.22754184    YES
NP_194508.1 0.9951      1.4050821 1.8947391 0.11401634      YES
NP_194934.1 0.0001      -0.85213473 -1.0044722 -0.37102516  NO
NP_195592.1 0.0018      -0.24519776 -0.38691854 -0.22699804 NO
NP_849519.1 0.0001      -0.5372892 -0.8234093 -0.28259784   NO
NP_563646.1 0.0001      -1.560055 -1.6902269 -0.580905      NO
NP_171892.2 0.3204      -0.32199251 -0.36432155 -0.38500867 NO
NP_973757.1 0.9938      0.3186865 0.46174844 -0.059120128   YES
NP_172007.1 0.0014      -0.66438656 -0.73611245 -0.76019549 NO
NP_172089.1 0.9951      1.8677263 2.1588378 0.041746955     YES
NP_173259.1 0.2157      0.31371925 0.27112246 -0.19926656   YES
NP_849686.1 0.2157      0.31371925 0.27112246 -0.19926656   YES
NP_173499.1 0.0001      -0.78342947 -0.78451193 -0.31653083 NO
NP_564143.1 0.9942      0.56822555 0.81844763 -0.17458004   YES
```


Arabidopsis_non_calcium_binding_proteins.txt - WordPad
File Edit View Insert Format Help
For Help, press F1
NUM
Start
Datafiles_proteomes
chapter 1.doc - Microsof...
Documentation - Mozila ...
Arabidopsis_non_calc...
10:42 PM

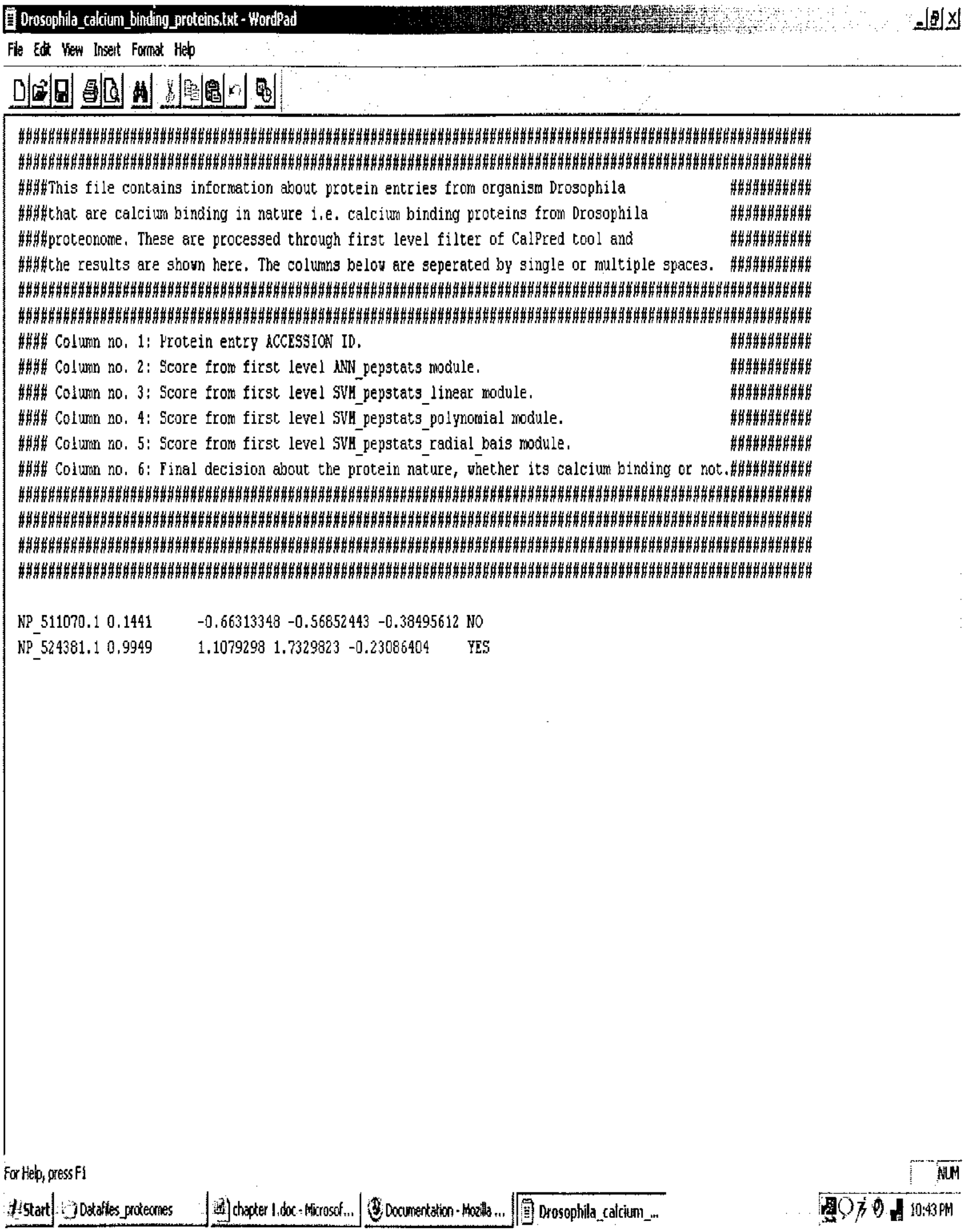
```
##########################################################################################################
##########################################################################################################
####This file contains information about protein entries from organism Arabidopsis thaliana    ###########
####that are non calcium binding in nature i.e. non calcium binding proteins from Arabidopsis  ###########
####thaliana  proteonome. These are processed through first level filter of CalPred tool and   ###########
####the results are shown here. The columns below are seperated by single or multiple spaces.  ###########
##########################################################################################################
##########################################################################################################
#### Column no. 1: Protein entry ACCESSION ID.                                                  ###########
#### Column no. 2: Score from first level ANN_pepstats module.                                  ###########
#### Column no. 3: Score from first level SVM_pepstats_linear module.                           ###########
#### Column no. 4: Score from first level SVM_pepstats_polynomial module.                       ###########
#### Column no. 5: Score from first level SVM_pepstats_radial_bais module.                      ###########
#### Column no. 6: Final decision about the protein nature, whether its calcium binding or not.###########
##########################################################################################################
##########################################################################################################
##########################################################################################################
##########################################################################################################

NP_171609.1 0.0003      -0.87123874 -0.87944213 -0.38893446 NO
NP_171610.1 0.0001      -1.6799687 -1.9546916 -0.36896597   NO
NP_171611.1 0.0001      -1.1407715 -1.1246934 -0.35233395   NO
NP_171612.1 0.0005      -0.79111132 -0.8306053 -0.29308654  NO
NP_171613.1 0.0001      -1.0816368 -1.2828561 -0.44367545   NO
NP_171614.1 0.0001      -0.64919384 -0.67850847 -0.363677   NO
NP_849568.1 0.0001      -0.64919384 -0.67850847 -0.363677   NO
NP_563617.1 0.0001      -1.4669955 -1.6694105 -0.40857007   NO
NP_973734.1 0.0001      -1.1876504 -1.311324 -0.41360108    NO
NP_171616.1 0.0001      -1.6243928 -1.8113052 -0.40383719   NO
NP_171617.1 0.0009      -0.60707704 -0.68919174 -0.74441429 NO
NP_171618.1 0.0025      0.30948928 0.010637849 -0.27962954  YES
NP_849569.1 0.0025      0.30948928 0.010637849 -0.27962954  YES
NP_563618.1 0.0001      -2.1969003 -2.2182676 -0.28673382   NO
NP_171620.2 0.0001      -1.1502774 -1.292725 -0.87349123    NO
NP_171621.1 0.0001      -0.78446352 -1.0331263 -0.3342133   NO
NP_171622.1 0.0018      -0.46497099 -0.62585298 -0.65798577 NO
NP_849571.1 0.0021      -0.47172431 -0.63123837 -0.68035747 NO
NP_849570.1 0.0012      -0.51169391 -0.67605213 -0.7021501  NO
NP_171623.1 0.5963      -0.43032354 -0.41977121 -0.30418919 NO
NP_563619.1 0.9158      0.42429943 1.0946175 -0.28127056    YES
```


C.elegans_calcium_binding_proteins.txt - WordPad
File Edit View Insert Format Help
For Help, press F1
NUM
Start
Datafiles_proteomes
chapter 1.doc - Microsof...
Documentation - Mozilla ...
C.elegans_calcium_bi...
10:43 PM

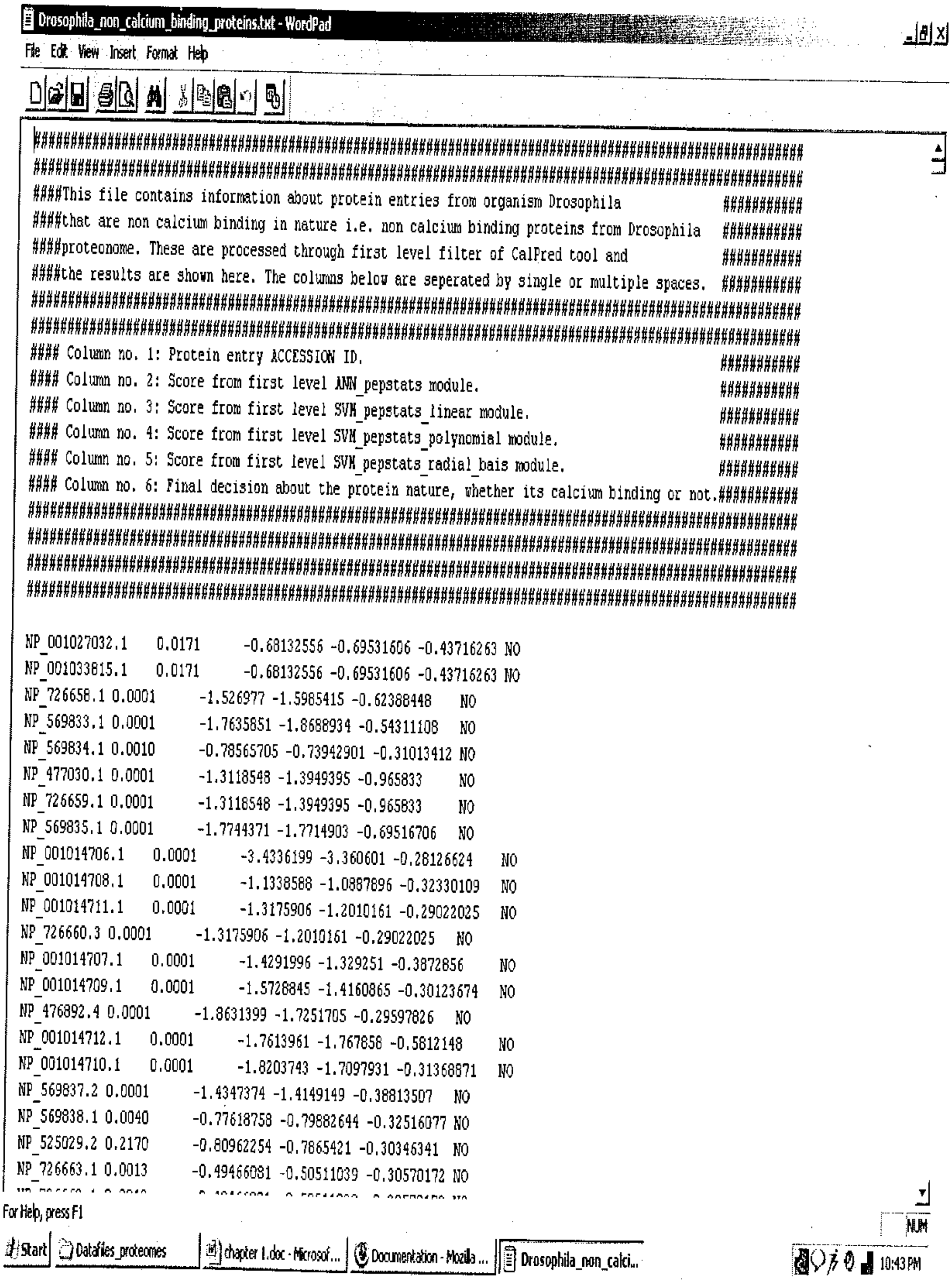
```
####################################################################################################
####################################################################################################
####This file contains information about protein entries from organism C. elegans          ###########
####that are calcium binding in nature i.e. calcium binding proteins from C. elegans        ###########
####proteonome. These are processed through first level filter of CalPred tool and          ###########
####the results are shown here. The columns below are seperated by single or multiple spaces. ###########
####################################################################################################
####################################################################################################
#### Column no. 1: Protein entry ACCESSION ID.                                              ###########
#### Column no. 2: Score from first level ANN_pepstats module.                              ###########
#### Column no. 3: Score from first level SVM_pepstats_linear module.                       ###########
#### Column no. 4: Score from first level SVM_pepstats_polynomial module.                   ###########
#### Column no. 5: Score from first level SVM_pepstats_radial_bais module.                  ###########
#### Column no. 6: Final decision about the protein nature, whether its calcium binding or not.###########
####################################################################################################
####################################################################################################
####################################################################################################
####################################################################################################

NP_001022226.1    0.0001       -0.60674307 -0.70389951 -0.32577512 NO
NP_493712.1 0.0806       -0.2482408 -0.30362458 -0.27635938  NO
```


C.elegans_non_calcium_binding_proteins.txt - WordPad
File Edit View Insert Format Help
For Help, press F1
NUM
Start
Datafiles_proteomes
chapter 1.doc - Microsof...
Documentation - Mozilla ...
C.elegans_non_calciu...
10:43 PM

```
###########################################################################################################
###########################################################################################################
####This file contains information about protein entries from organism C. elegans             ###########
####that are non calcium binding in nature i.e. non calcium binding proteins from C. elegans  ###########
####proteonome. These are processed through first level filter of CalPred tool and            ###########
####the results are shown here. The columns below are seperated by single or multiple spaces. ###########
###########################################################################################################
###########################################################################################################
#### Column no. 1: Protein entry ACCESSION ID.                                                ###########
#### Column no. 2: Score from first level ANN_pepstats module.                                ###########
#### Column no. 3: Score from first level SVM_pepstats_linear module.                         ###########
#### Column no. 4: Score from first level SVM_pepstats_polynomial module.                     ###########
#### Column no. 5: Score from first level SVM_pepstats_radial_bais module.                    ###########
#### Column no. 6: Final decision about the protein nature, whether its calcium binding or not.###########
###########################################################################################################
###########################################################################################################
###########################################################################################################
###########################################################################################################

NP_490660.1 0.3825      -0.22656527 -0.23655654 -0.46054264 NO
NP_490661.1 0.0003      -0.69025919 -0.83359889 -0.28485386 NO
NP_490662.1 0.9934      0.40241351 0.44869756 -0.16108491   YES
NP_490663.1 0.9935      0.40803701 0.46002248 -0.16639583   YES
NP_490664.1 0.8136      -0.46335351 -0.40947699 -0.36309383 NO
NP_490665.2 0.3947      -0.35748945 -0.38556523 -0.31570791 NO
NP_001021777.1    0.0057      -0.63054049 -0.48079538 -0.32703541 NO
NP_490666.2 0.0006      -0.80244341 -0.97891338 -0.80320749 NO
NP_490668.3 0.0005      -0.55866719 -0.69859032 -0.54280108 NO
NP_001021778.1    0.0001      -1.3786601 -1.3462216 -0.51260883   NO
NP_871894.1 0.0004      -1.051523 -1.1780421 -0.7626316     NO
NP_490670.1 0.0005      -0.70486714 -0.81820749 -0.39802693 NO
NP_871895.1 0.0014      -0.67878275 -0.74923247 -0.32423427 NO
NP_490671.2 0.9924      0.45291258 0.42073024 -0.26257104   YES
NP_490672.2 0.0001      -1.2804209 -1.4068493 -0.57784265   NO
NP_490673.1 0.3296      -0.3500965 -0.28707669 -0.30845356  NO
NP_490674.2 0.0001      -1.0789686 -1.10609 -0.50376294     NO
NP_490675.1 0.0001      -1.4562965 -1.4749731 -0.76849503   NO
NP_490676.1 0.0001      -1.3940854 -1.4978907 -0.31179547   NO
NP_490677.2 0.0001      -1.6526116 -1.6758107 -0.77106255   NO
NP_490678.3 0.0158      0.52709382 0.26207874 -0.28040481   YES
```

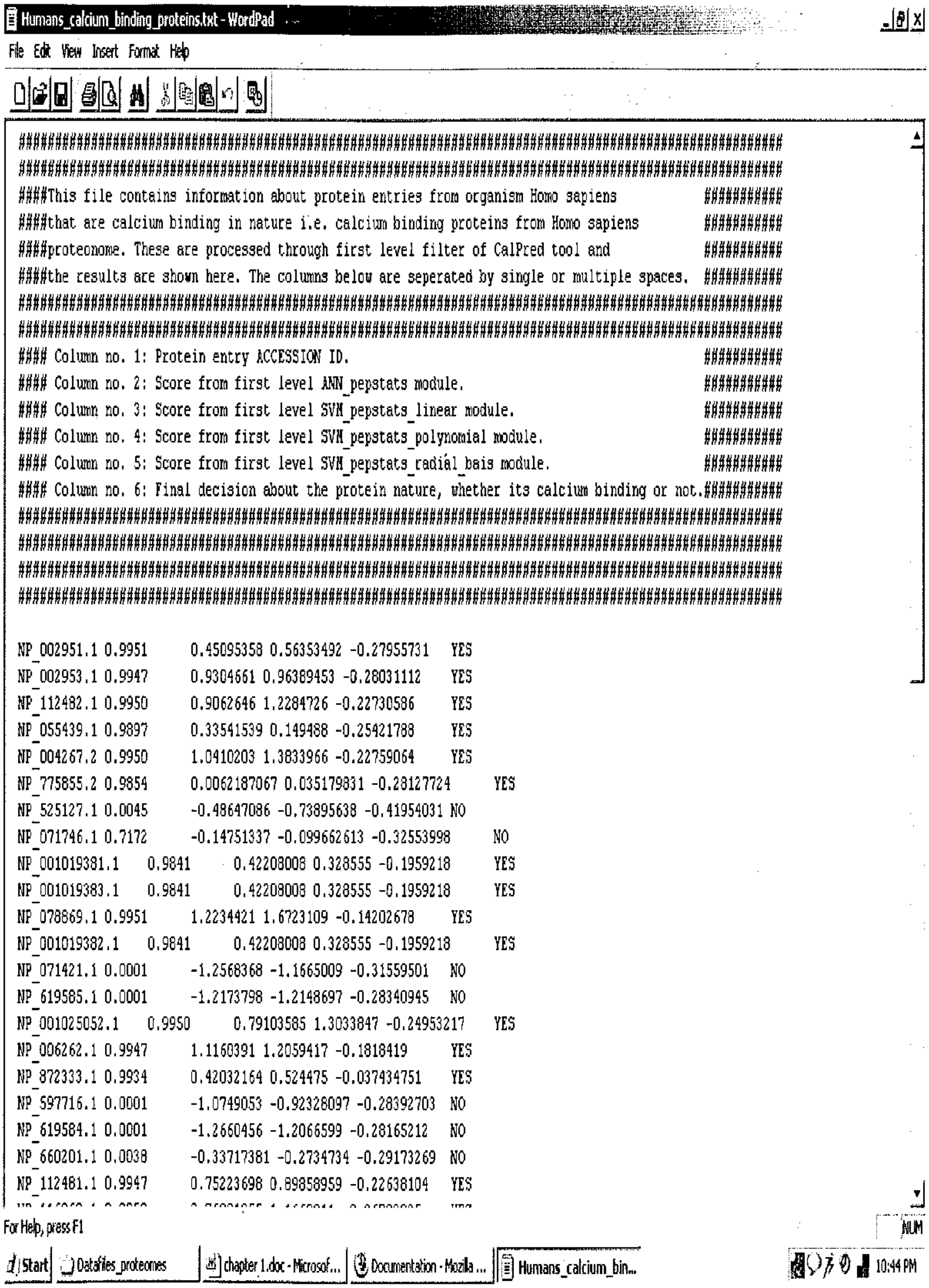

Drosophila_calcium_binding_proteins.txt - WordPad
File Edit View Insert Format Help
For Help, press F1
NUM
Start
Datafiles_proteomes
chapter 1.doc - Microsof...
Documentation - Mozilla ...
Drosophila_calcium_...
10:43 PM

```
#########################################################################################################
#########################################################################################################
####This file contains information about protein entries from organism Drosophila              ##########
####that are calcium binding in nature i.e. calcium binding proteins from Drosophila           ##########
####proteonome. These are processed through first level filter of CalPred tool and             ##########
####the results are shown here. The columns below are seperated by single or multiple spaces.  ##########
#########################################################################################################
#########################################################################################################
#### Column no. 1: Protein entry ACCESSION ID.                                                 ##########
#### Column no. 2: Score from first level ANN_pepstats module.                                 ##########
#### Column no. 3: Score from first level SVM_pepstats_linear module.                          ##########
#### Column no. 4: Score from first level SVM_pepstats_polynomial module.                      ##########
#### Column no. 5: Score from first level SVM_pepstats_radial_bais module.                     ##########
#### Column no. 6: Final decision about the protein nature, whether its calcium binding or not.##########
#########################################################################################################
#########################################################################################################
#########################################################################################################
#########################################################################################################

NP_511070.1 0.1441      -0.66313348 -0.56852443 -0.38495612 NO
NP_524381.1 0.9949      1.1079298 1.7329823 -0.23086404     YES
```

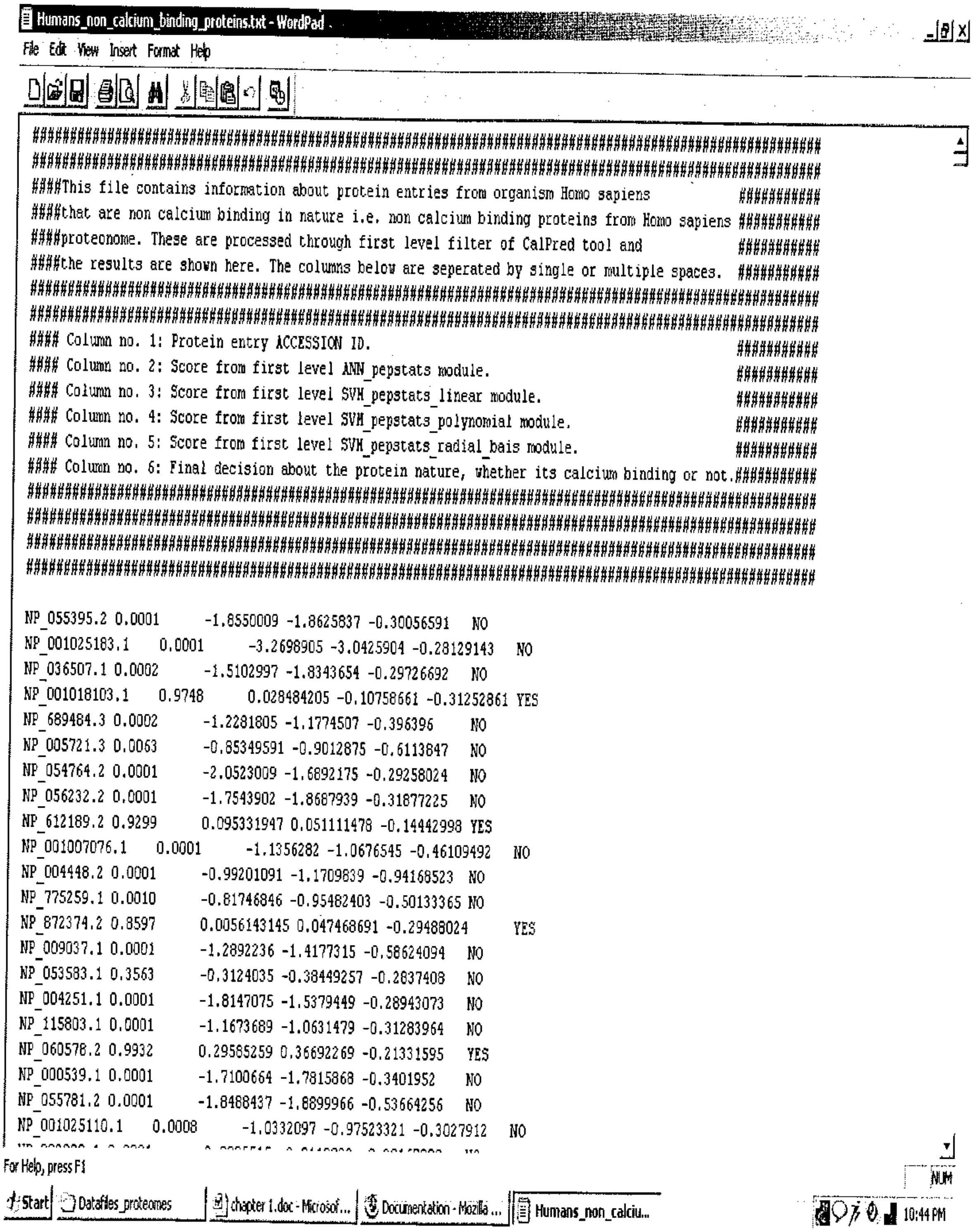


Drosophila_non_calcium_binding_proteins.txt - WordPad
File Edit View Insert Format Help
For Help, press F1
NUM
Start
Datafiles_proteomes
chapter 1.doc - Microsof...
Documentation - Mozilla ...
Drosophila_non_calci...
10:43 PM

```
####################################################################################################
####################################################################################################
####This file contains information about protein entries from organism Drosophila          ###########
####that are non calcium binding in nature i.e. non calcium binding proteins from Drosophila ###########
####proteonome. These are processed through first level filter of CalPred tool and          ###########
####the results are shown here. The columns below are seperated by single or multiple spaces. ###########
####################################################################################################
####################################################################################################
#### Column no. 1: Protein entry ACCESSION ID.                                              ###########
#### Column no. 2: Score from first level ANN_pepstats module.                              ###########
#### Column no. 3: Score from first level SVM_pepstats_linear module.                       ###########
#### Column no. 4: Score from first level SVM_pepstats_polynomial module.                   ###########
#### Column no. 5: Score from first level SVM_pepstats_radial_bais module.                  ###########
#### Column no. 6: Final decision about the protein nature, whether its calcium binding or not.###########
####################################################################################################
####################################################################################################
####################################################################################################
####################################################################################################

NP_001027032.1    0.0171      -0.68132556 -0.69531606 -0.43716263 NO
NP_001033815.1    0.0171      -0.68132556 -0.69531606 -0.43716263 NO
NP_726658.1 0.0001      -1.526977 -1.5985415 -0.62388448   NO
NP_569833.1 0.0001      -1.7635851 -1.8688934 -0.54311108  NO
NP_569834.1 0.0010      -0.78565705 -0.73942901 -0.31013412 NO
NP_477030.1 0.0001      -1.3118548 -1.3949395 -0.965833    NO
NP_726659.1 0.0001      -1.3118548 -1.3949395 -0.965833    NO
NP_569835.1 0.0001      -1.7744371 -1.7714903 -0.69516706  NO
NP_001014706.1    0.0001      -3.4336199 -3.360601 -0.28126624   NO
NP_001014708.1    0.0001      -1.1338588 -1.0887896 -0.32330109  NO
NP_001014711.1    0.0001      -1.3175906 -1.2010161 -0.29022025  NO
NP_726660.3 0.0001      -1.3175906 -1.2010161 -0.29022025  NO
NP_001014707.1    0.0001      -1.4291996 -1.329251 -0.3872856    NO
NP_001014709.1    0.0001      -1.5728845 -1.4160865 -0.30123674  NO
NP_476892.4 0.0001      -1.8631399 -1.7251705 -0.29597826  NO
NP_001014712.1    0.0001      -1.7613961 -1.767858 -0.5812148    NO
NP_001014710.1    0.0001      -1.8203743 -1.7097931 -0.31368871  NO
NP_569837.2 0.0001      -1.4347374 -1.4149149 -0.38813507  NO
NP_569838.1 0.0040      -0.77618758 -0.79882644 -0.32516077 NO
NP_525029.2 0.2170      -0.80962254 -0.7865421 -0.30346341  NO
NP_726663.1 0.0013      -0.49466081 -0.50511039 -0.30570172 NO
```

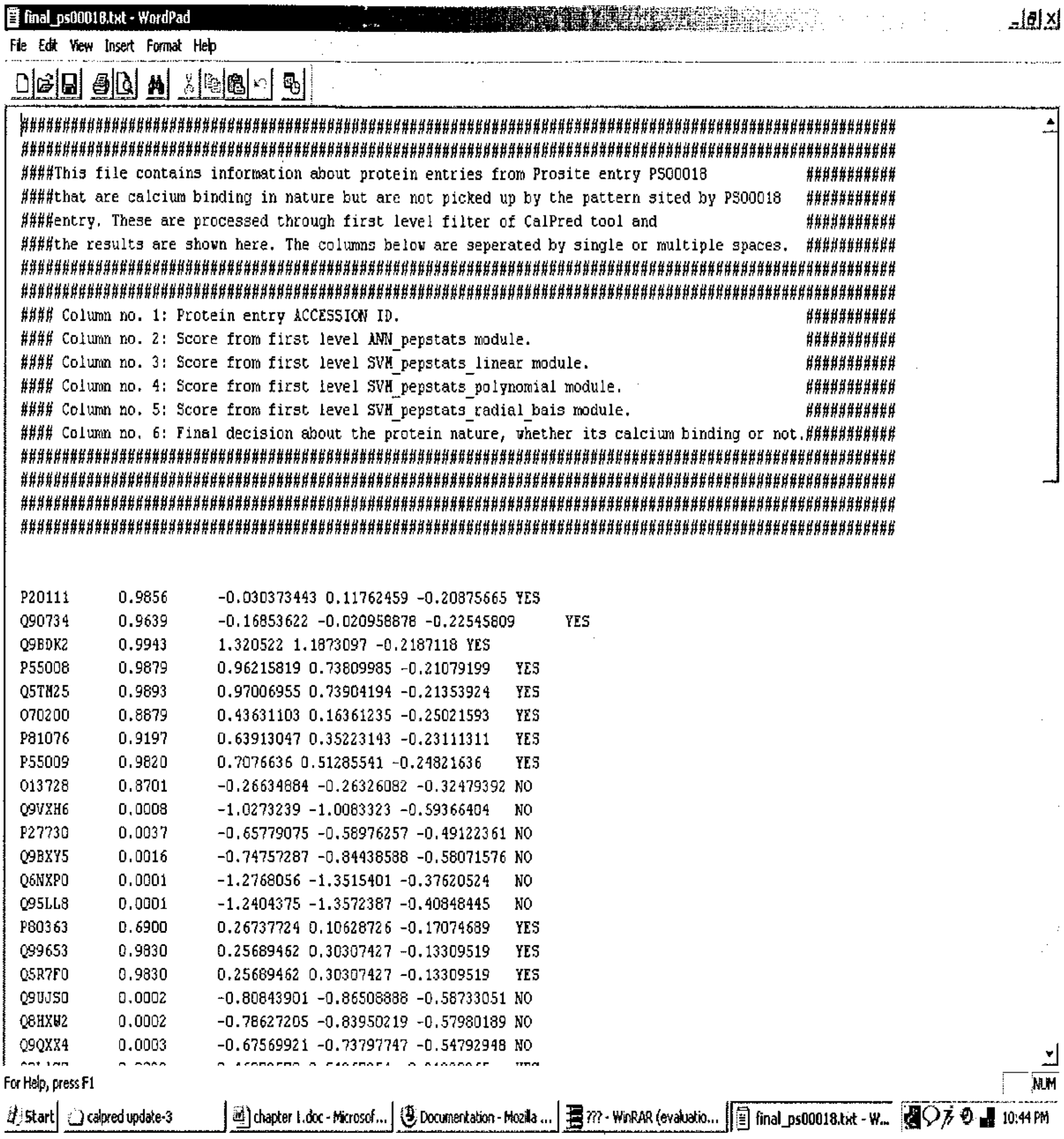

Humans_calcium_binding_proteins.txt - WordPad
File Edit View Insert Format Help
For Help, press F1
NUM
Start
Datafiles_proteomes
chapter 1.doc - Microsof...
Documentation - Mozila ...
Humans_calcium_bin...
10:44 PM

```
##########################################################################################################
##########################################################################################################
####This file contains information about protein entries from organism Homo sapiens          ###########
####that are calcium binding in nature i.e. calcium binding proteins from Homo sapiens       ###########
####proteonome. These are processed through first level filter of CalPred tool and           ###########
####the results are shown here. The columns below are seperated by single or multiple spaces. ###########
##########################################################################################################
##########################################################################################################
#### Column no. 1: Protein entry ACCESSION ID.                                              ###########
#### Column no. 2: Score from first level ANN_pepstats module.                              ###########
#### Column no. 3: Score from first level SVM_pepstats_linear module.                       ###########
#### Column no. 4: Score from first level SVM_pepstats_polynomial module.                   ###########
#### Column no. 5: Score from first level SVM_pepstats_radial_bais module.                  ###########
#### Column no. 6: Final decision about the protein nature, whether its calcium binding or not.###########
##########################################################################################################
##########################################################################################################
##########################################################################################################
##########################################################################################################


NP_002951.1 0.9951      0.45095358 0.56353492 -0.27955731   YES
NP_002953.1 0.9947      0.9304661 0.96389453 -0.28031112    YES
NP_112482.1 0.9950      0.9062646 1.2284726 -0.22730586     YES
NP_055439.1 0.9897      0.33541539 0.149488 -0.25421788     YES
NP_004267.2 0.9950      1.0410203 1.3833966 -0.22759064     YES
NP_775855.2 0.9854      0.0062187067 0.035179831 -0.28127724        YES
NP_525127.1 0.0045      -0.48647086 -0.73895638 -0.41954031 NO
NP_071746.1 0.7172      -0.14751337 -0.099662613 -0.32553998        NO
NP_001019381.1      0.9841      0.42208008 0.328555 -0.1959218      YES
NP_001019383.1      0.9841      0.42208008 0.328555 -0.1959218      YES
NP_078869.1 0.9951      1.2234421 1.6723109 -0.14202678     YES
NP_001019382.1      0.9841      0.42208008 0.328555 -0.1959218      YES
NP_071421.1 0.0001      -1.2568368 -1.1665009 -0.31559501   NO
NP_619585.1 0.0001      -1.2173798 -1.2148697 -0.28340945   NO
NP_001025052.1      0.9950      0.79103585 1.3033847 -0.24953217    YES
NP_006262.1 0.9947      1.1160391 1.2059417 -0.1818419      YES
NP_872333.1 0.9934      0.42032164 0.524475 -0.037434751    YES
NP_597716.1 0.0001      -1.0749053 -0.92328097 -0.28392703  NO
NP_619584.1 0.0001      -1.2660456 -1.2066599 -0.28165212   NO
NP_660201.1 0.0038      -0.33717381 -0.2734734 -0.29173269  NO
NP_112481.1 0.9947      0.75223698 0.89858959 -0.22638104   YES
```



Humans_non_calcium_binding_proteins.txt - WordPad
File Edit View Insert Format Help

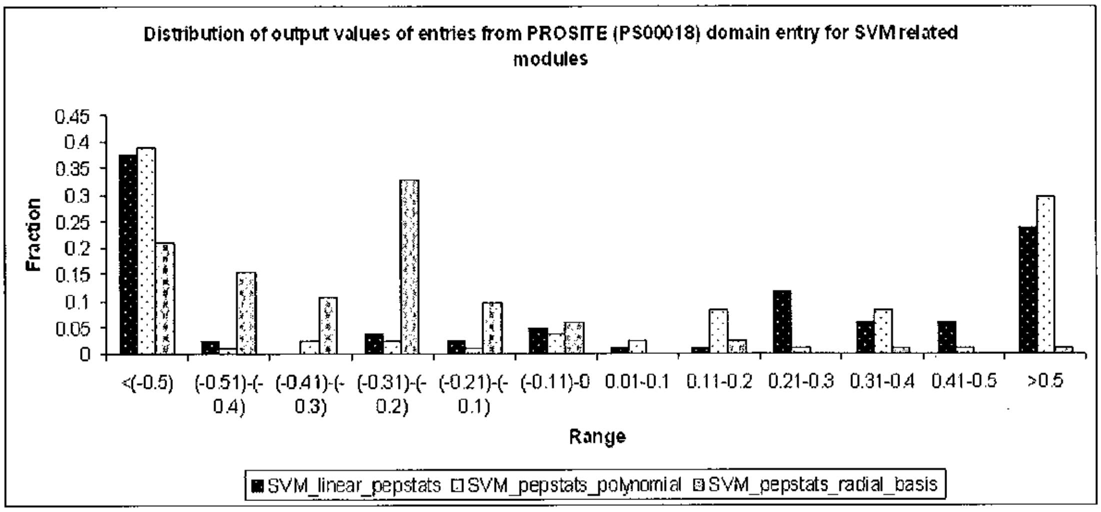
```
##########################################################################################################
##########################################################################################################
####This file contains information about protein entries from organism Homo sapiens              ###########
####that are non calcium binding in nature i.e. non calcium binding proteins from Homo sapiens ###########
####proteonome. These are processed through first level filter of CalPred tool and             ###########
####the results are shown here. The columns below are seperated by single or multiple spaces.   ###########
##########################################################################################################
##########################################################################################################
#### Column no. 1: Protein entry ACCESSION ID.                                                   ###########
#### Column no. 2: Score from first level ANN_pepstats module.                                   ###########
#### Column no. 3: Score from first level SVM_pepstats_linear module.                            ###########
#### Column no. 4: Score from first level SVM_pepstats_polynomial module.                        ###########
#### Column no. 5: Score from first level SVM_pepstats_radial_bais module.                       ###########
#### Column no. 6: Final decision about the protein nature, whether its calcium binding or not.###########
##########################################################################################################
##########################################################################################################
##########################################################################################################
##########################################################################################################

NP_055395.2 0.0001      -1.8550009 -1.8625837 -0.30056591   NO
NP_001025183.1    0.0001      -3.2698905 -3.0425904 -0.28129143   NO
NP_036507.1 0.0002      -1.5102997 -1.8343654 -0.29726692   NO
NP_001018103.1    0.9748      0.028484205 -0.10758661 -0.31252861 YES
NP_689484.3 0.0002      -1.2281805 -1.1774507 -0.396396     NO
NP_005721.3 0.0063      -0.85349591 -0.9012875 -0.6113847   NO
NP_054764.2 0.0001      -2.0523009 -1.6892175 -0.29258024   NO
NP_056232.2 0.0001      -1.7543902 -1.8687939 -0.31877225   NO
NP_612189.2 0.9299      0.095331947 0.051111478 -0.14442998 YES
NP_001007076.1    0.0001      -1.1356282 -1.0676545 -0.46109492   NO
NP_004448.2 0.0001      -0.99201091 -1.1709839 -0.94168523  NO
NP_775259.1 0.0010      -0.81746846 -0.95482403 -0.50133365 NO
NP_872374.2 0.8597      0.0056143145 0.047468691 -0.29488024       YES
NP_009037.1 0.0001      -1.2892236 -1.4177315 -0.58624094   NO
NP_053583.1 0.3563      -0.3124035 -0.38449257 -0.2837408   NO
NP_004251.1 0.0001      -1.8147075 -1.5379449 -0.28943073   NO
NP_115803.1 0.0001      -1.1673689 -1.0631479 -0.31283964   NO
NP_060578.2 0.9932      0.29585259 0.36692269 -0.21331595   YES
NP_000539.1 0.0001      -1.7100664 -1.7815868 -0.3401952    NO
NP_055781.2 0.0001      -1.8488437 -1.8899966 -0.53664256   NO
NP_001025110.1    0.0008      -1.0332097 -0.97523321 -0.3027912   NO
```

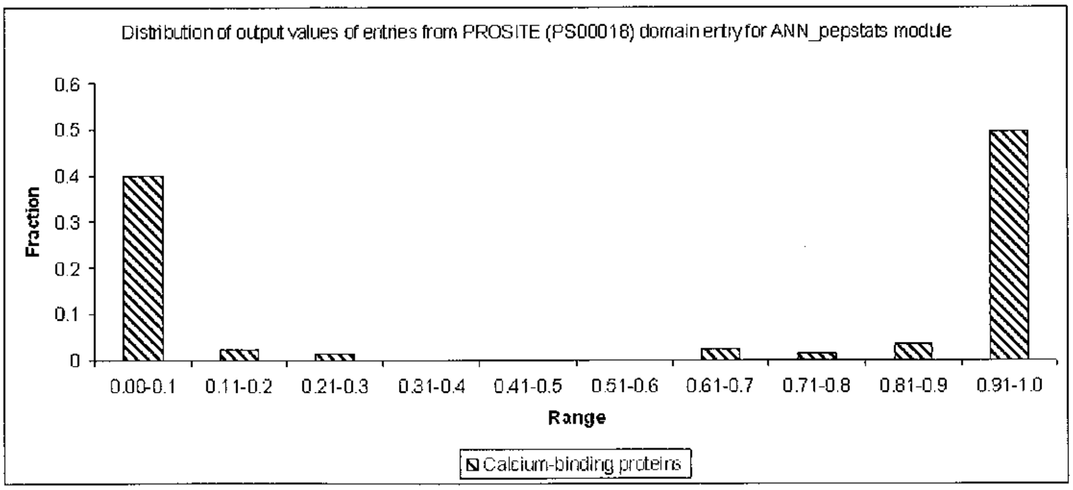

final_ps00018.txt - WordPad
File Edit View Insert Format Help
For Help, press F1
NUM
Start
calpred update-3
chapter 1.doc - Microsof...
Documentation - Mozilla ...
??? - WinRAR (evaluatio...
final_ps00018.txt - W...
10:44 PM

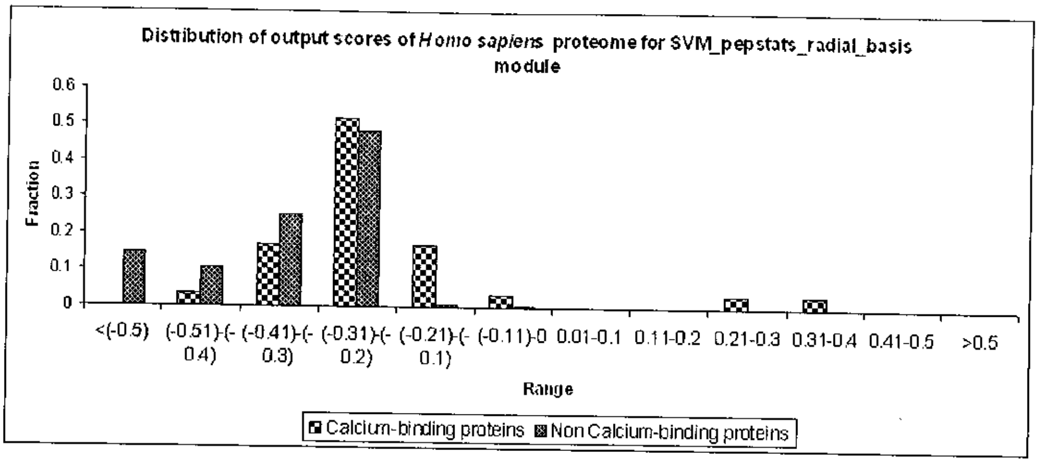
```
##########################################################################################################
##########################################################################################################
####This file contains information about protein entries from Prosite entry PS00018              ###########
####that are calcium binding in nature but are not picked up by the pattern sited by PS00018  ###########
####entry. These are processed through first level filter of CalPred tool and                 ###########
####the results are shown here. The columns below are seperated by single or multiple spaces.  ###########
##########################################################################################################
##########################################################################################################
#### Column no. 1: Protein entry ACCESSION ID.                                                 ###########
#### Column no. 2: Score from first level ANN_pepstats module.                                 ###########
#### Column no. 3: Score from first level SVM_pepstats_linear module.                          ###########
#### Column no. 4: Score from first level SVM_pepstats_polynomial module.                      ###########
#### Column no. 5: Score from first level SVM_pepstats_radial_bais module.                     ###########
#### Column no. 6: Final decision about the protein nature, whether its calcium binding or not.###########
##########################################################################################################
##########################################################################################################
##########################################################################################################
##########################################################################################################


P20111      0.9856      -0.030373443 0.11762459 -0.20875665 YES
Q90734      0.9639      -0.16853622 -0.020958878 -0.22545809      YES
Q9BDK2      0.9943      1.320522 1.1873097 -0.2187118 YES
P55008      0.9879      0.96215819 0.73809985 -0.21079199   YES
Q5TM25      0.9893      0.97006955 0.73904194 -0.21353924   YES
O70200      0.8879      0.43631103 0.16361235 -0.25021593   YES
P81076      0.9197      0.63913047 0.35223143 -0.23111311   YES
P55009      0.9820      0.7076636 0.51285541 -0.24821636    YES
O13728      0.8701      -0.26634884 -0.26326082 -0.32479392 NO
Q9VXH6      0.0008      -1.0273239 -1.0083323 -0.59366404   NO
P27730      0.0037      -0.65779075 -0.58976257 -0.49122361 NO
Q9BXY5      0.0016      -0.74757287 -0.84438588 -0.58071576 NO
Q6NXP0      0.0001      -1.2768056 -1.3515401 -0.37620524   NO
Q95LL8      0.0001      -1.2404375 -1.3572387 -0.40848445   NO
P80363      0.6900      0.26737724 0.10628726 -0.17074689   YES
Q99653      0.9830      0.25689462 0.30307427 -0.13309519   YES
Q5R7F0      0.9830      0.25689462 0.30307427 -0.13309519   YES
Q9UJS0      0.0002      -0.80843901 -0.86508888 -0.58733051 NO
Q8HXW2      0.0002      -0.78627205 -0.83950219 -0.57980189 NO
Q9QXX4      0.0003      -0.67569921 -0.73797747 -0.54792948 NO
```

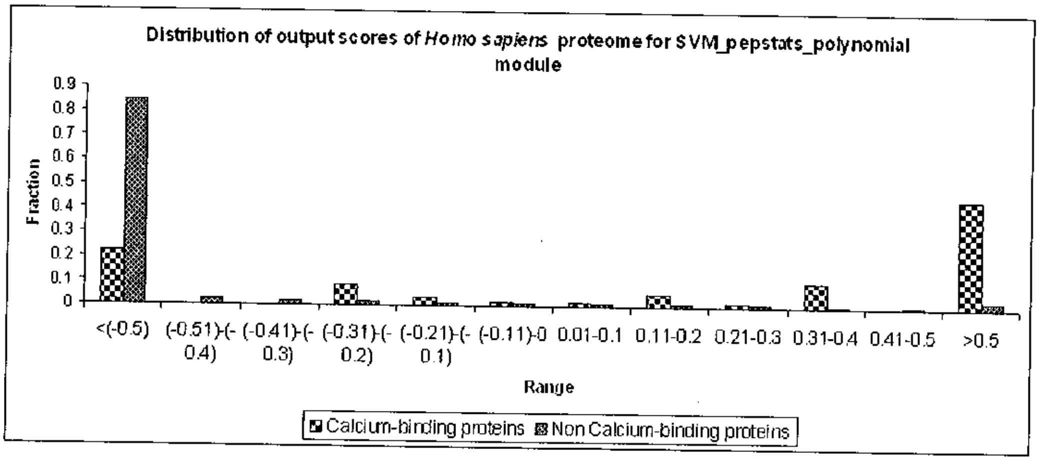
APPENDIX-V

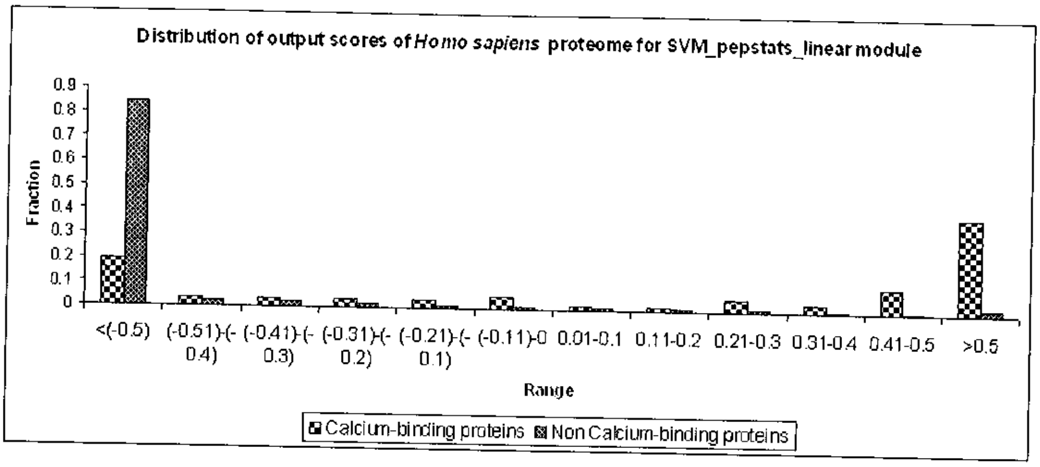
Plots for distribution of output scores of proteins from proteomes of four organisms and PS00018 dataset, which were queried with the first-level filter

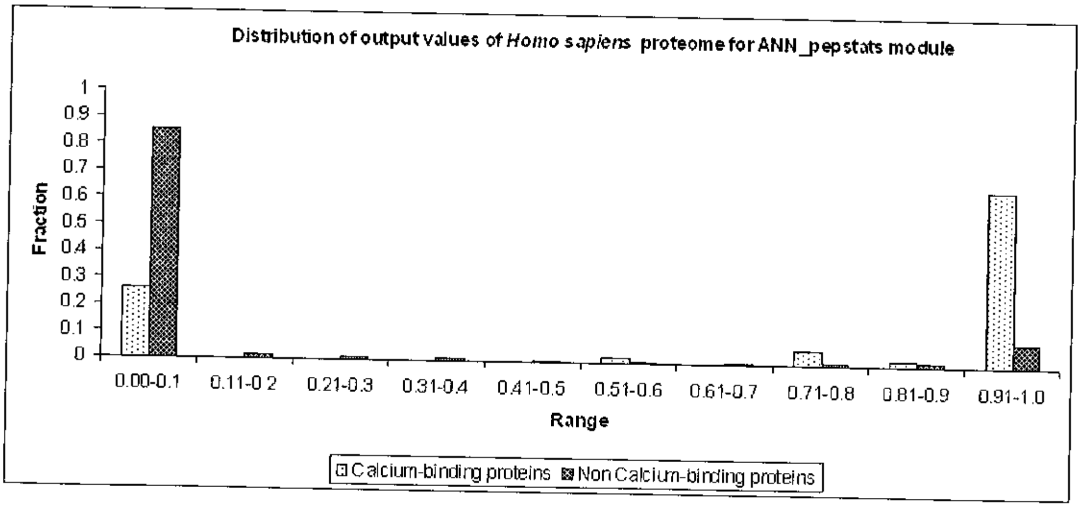


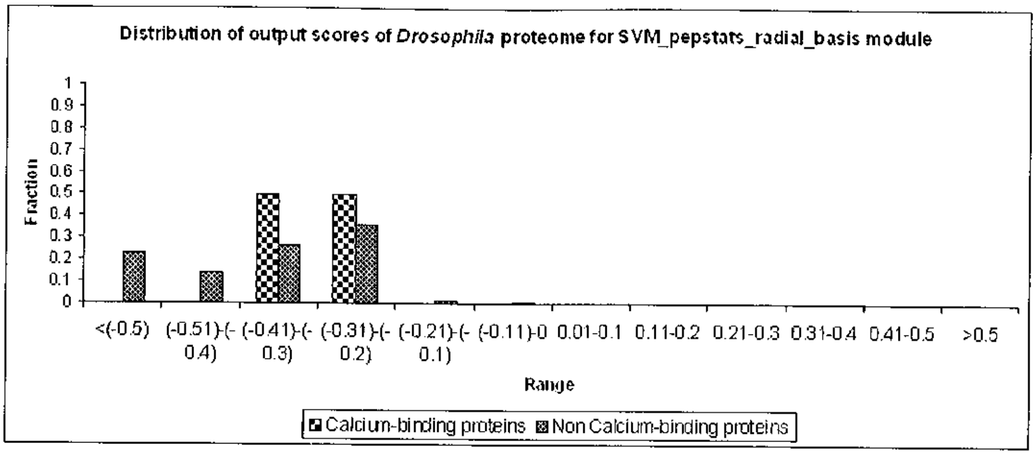


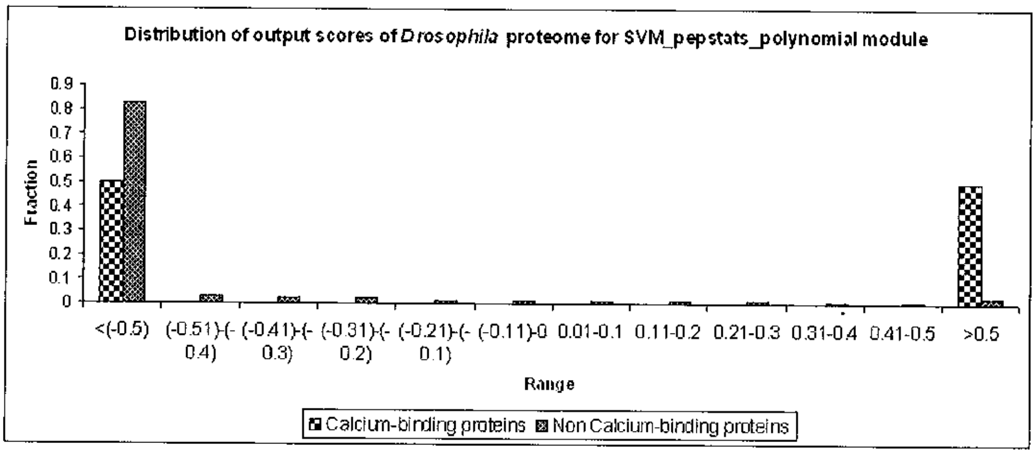

Distribution of output scores of *Homo sapiens* proteome for SVM_pepstats_radial_basis module
Fraction
0.6
0.5
0.4
0.3
0.2
0.1
0
<(-0.5)
(-0.51)-(-0.4)
(-0.41)-(-0.3)
(-0.31)-(-0.2)
(-0.21)-(-0.1)
(-0.11)-0
0.01-0.1
0.11-0.2
0.21-0.3
0.31-0.4
0.41-0.5
>0.5
Range
▨ Calcium-binding proteins ▨ Non Calcium-binding proteins

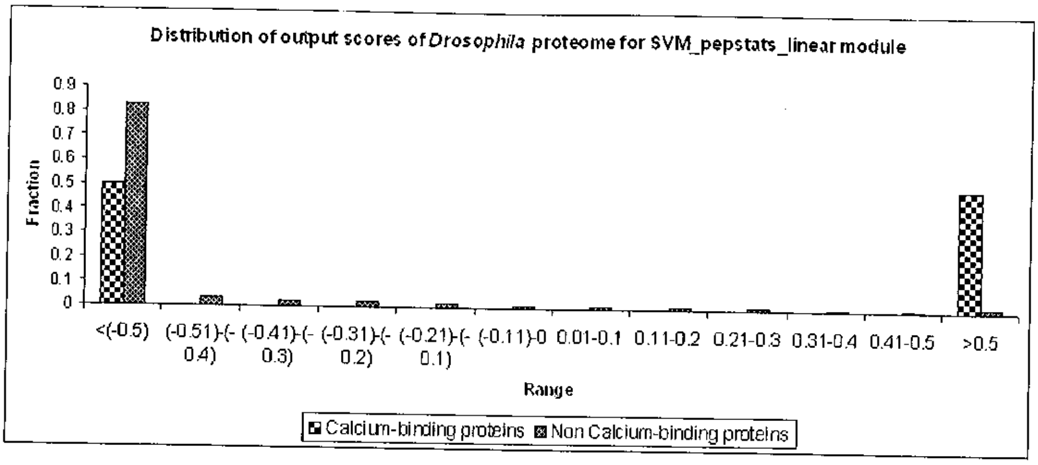

Distribution of output scores of *Homo sapiens* proteome for SVM_pepstats_polynomial module
Fraction
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
<(-0.5)
(-0.51)-(-0.4)
(-0.41)-(-0.3)
(-0.31)-(-0.2)
(-0.21)-(-0.1)
(-0.11)-0
0.01-0.1
0.11-0.2
0.21-0.3
0.31-0.4
0.41-0.5
>0.5
Range
▨ Calcium-binding proteins ▨ Non Calcium-binding proteins

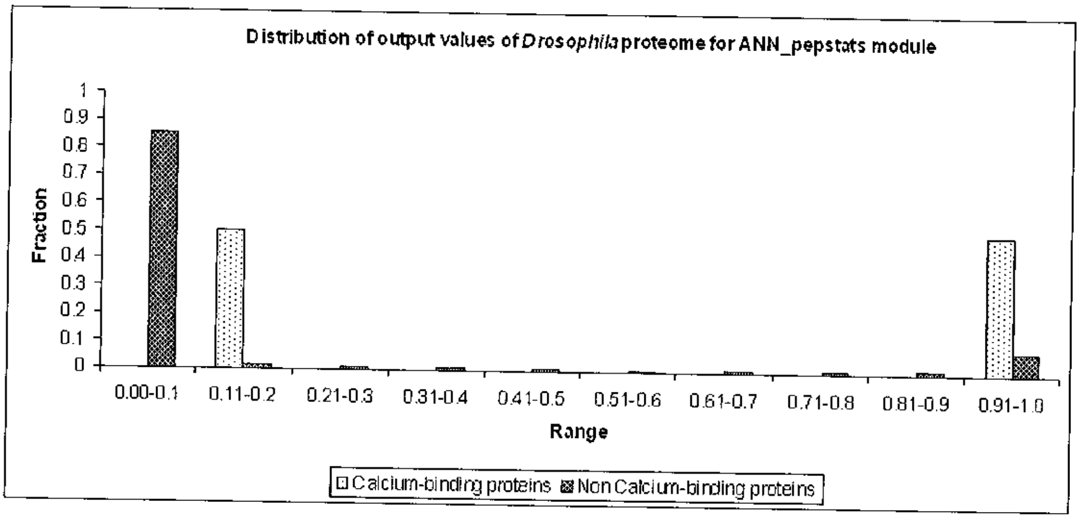

Distribution of output scores of *Homo sapiens* proteome for SVM_pepstats_linear module
Fraction
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
<(-0.5)
(-0.51)-(-0.4)
(-0.41)-(-0.3)
(-0.31)-(-0.2)
(-0.21)-(-0.1)
(-0.11)-0
0.01-0.1
0.11-0.2
0.21-0.3
0.31-0.4
0.41-0.5
>0.5
Range
▩ Calcium-binding proteins ▩ Non Calcium-binding proteins

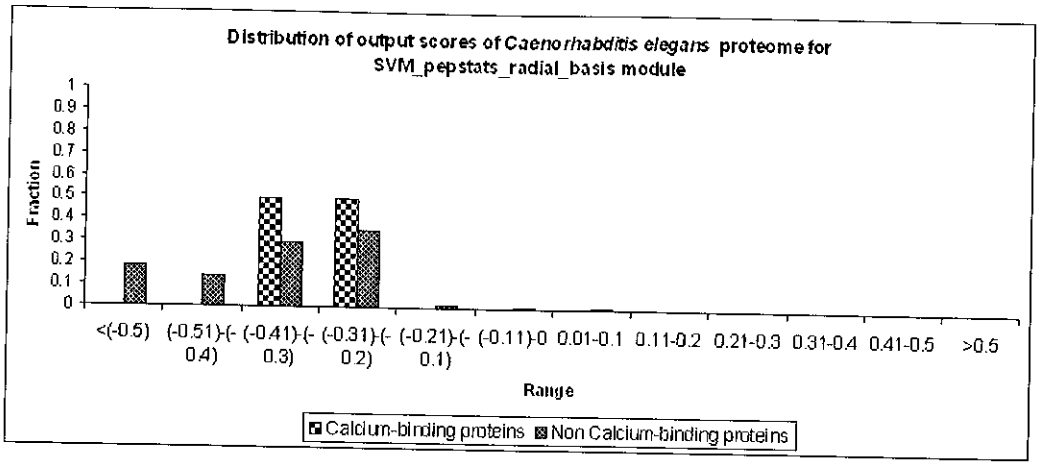

Distribution of output values of *Homo sapiens* proteome for ANN_pepstats module
Fraction
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.00-0.1
0.11-0.2
0.21-0.3
0.31-0.4
0.41-0.5
0.51-0.6
0.61-0.7
0.71-0.8
0.81-0.9
0.91-1.0
Range
□ Calcium-binding proteins ▩ Non Calcium-binding proteins


Distribution of output scores of *Drosophila* proteome for SVM_pepstats_radial_basis module

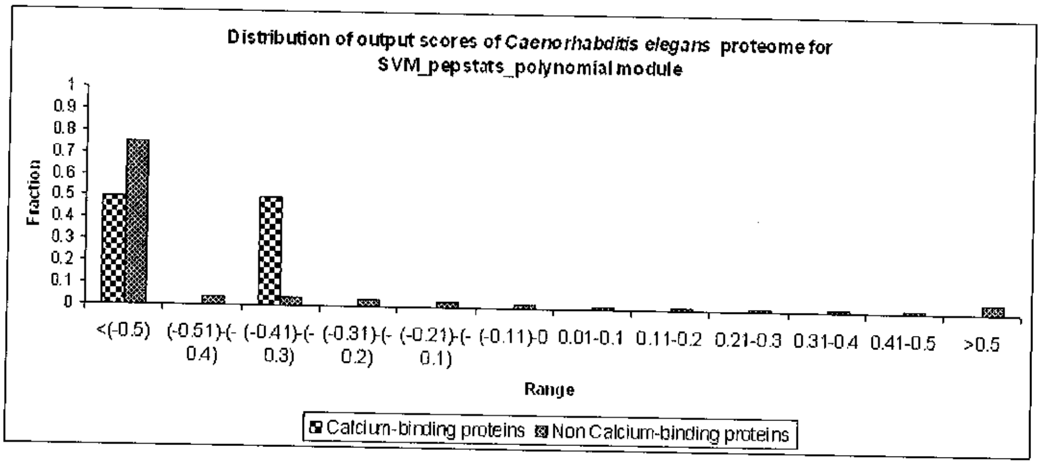

Distribution of output scores of *Drosophila* proteome for SVM_pepstats_polynomial module


Distribution of output scores of *Drosophila* proteome for SVM_pepstats_linear module
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Fraction
<(-0.5)
(-0.51)-(-0.4)
(-0.41)-(-0.3)
(-0.31)-(-0.2)
(-0.21)-(-0.1)
(-0.11)-0
0.01-0.1
0.11-0.2
0.21-0.3
0.31-0.4
0.41-0.5
>0.5
Range
▨ Calcium-binding proteins ▨ Non Calcium-binding proteins

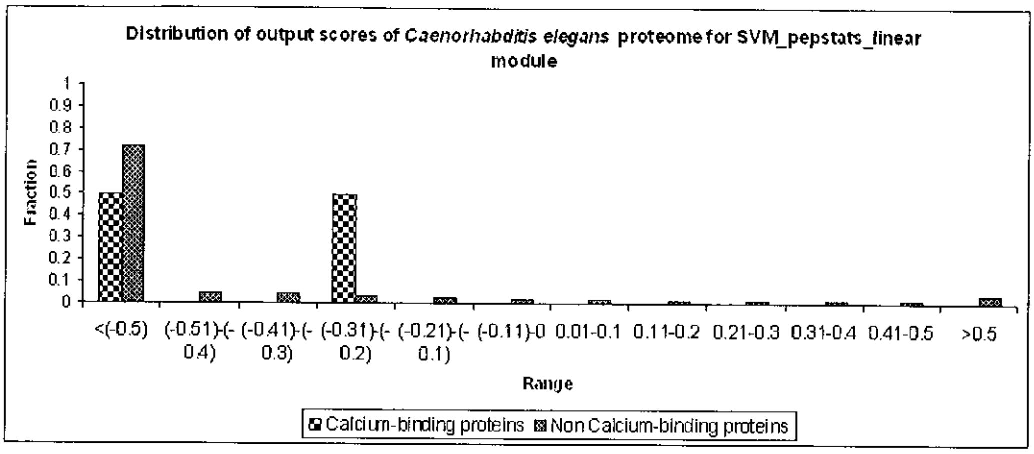

Distribution of output values of *Drosophila* proteome for ANN_pepstats module
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Fraction
0.00-0.1
0.11-0.2
0.21-0.3
0.31-0.4
0.41-0.5
0.51-0.6
0.61-0.7
0.71-0.8
0.81-0.9
0.91-1.0
Range
□ Calcium-binding proteins ▨ Non Calcium-binding proteins

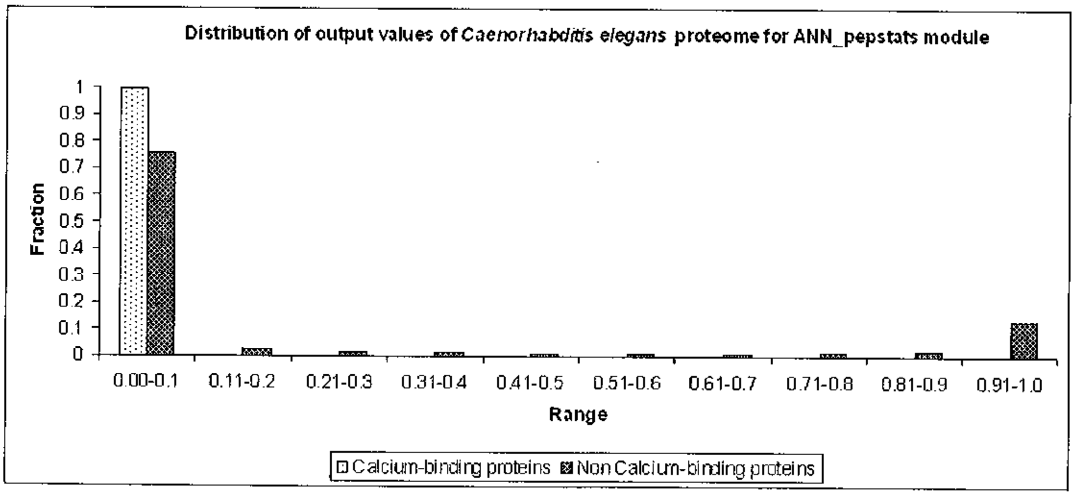


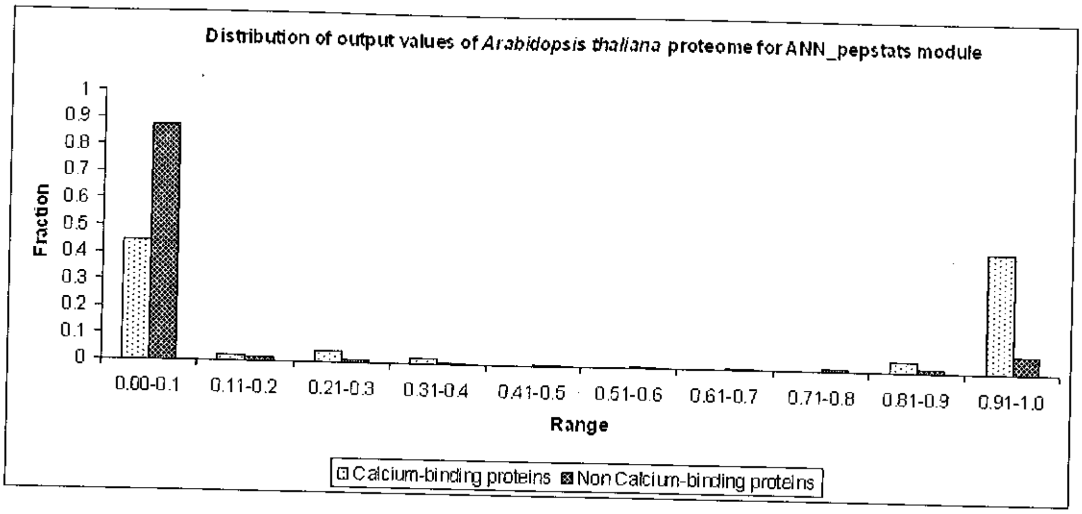
Distribution of output scores of *Caenorhabditis elegans* proteome for SVM_pepstats_radial_basis module

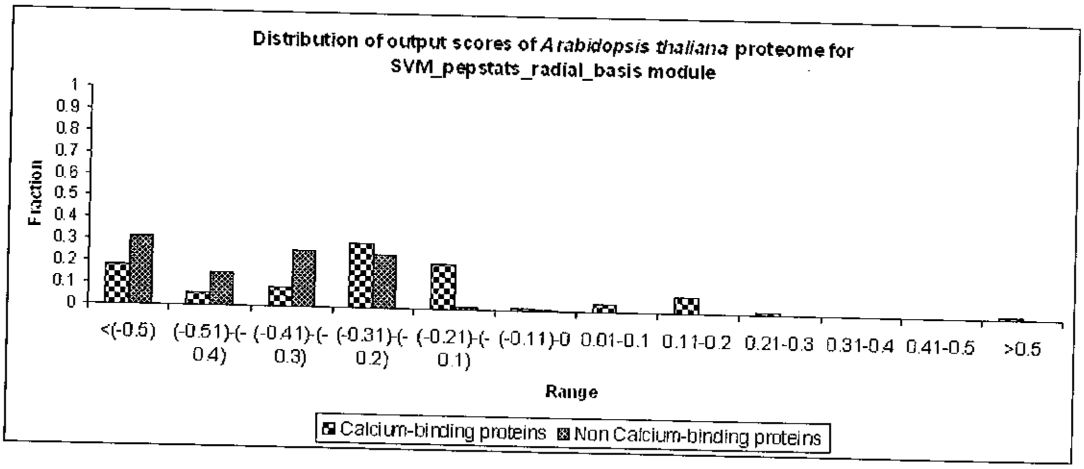


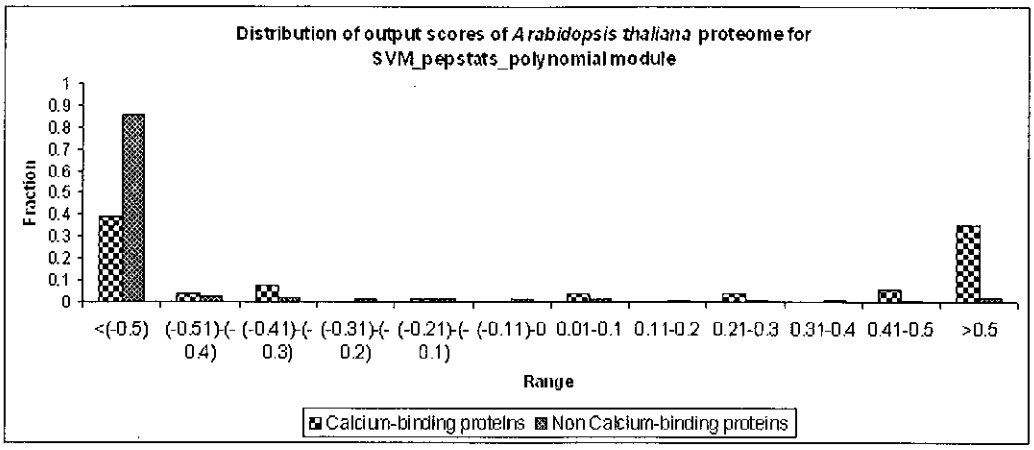
Distribution of output scores of *Caenorhabditis elegans* proteome for SVM_pepstats_polynomial module

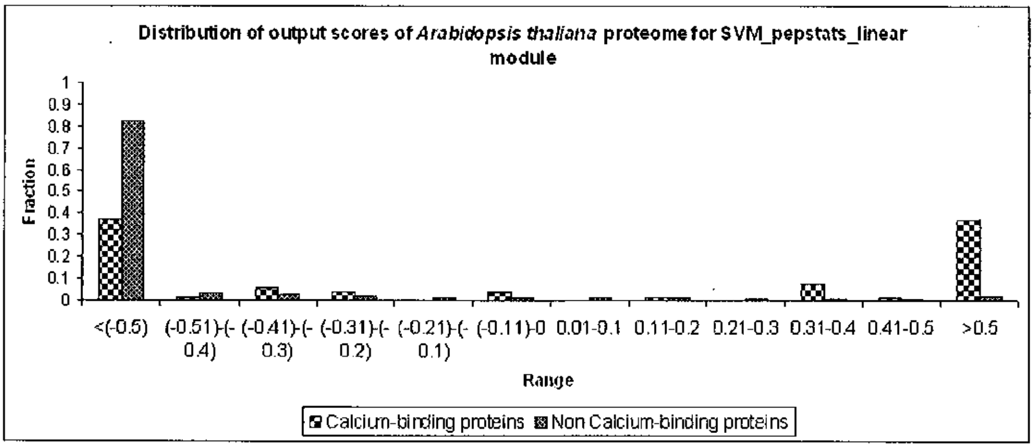

Distribution of output scores of *Caenorhabditis elegans* proteome for SVM_pepstats_linear module
Fraction
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
<(-0.5)
(-0.51)-(-0.4)
(-0.41)-(-0.3)
(-0.31)-(-0.2)
(-0.21)-(-0.1)
(-0.11)-0
0.01-0.1
0.11-0.2
0.21-0.3
0.31-0.4
0.41-0.5
>0.5
Range
Calcium-binding proteins
Non Calcium-binding proteins


Distribution of output values of *Caenorhabditis elegans* proteome for ANN_pepstats module
Fraction
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.00-0.1
0.11-0.2
0.21-0.3
0.31-0.4
0.41-0.5
0.51-0.6
0.61-0.7
0.71-0.8
0.81-0.9
0.91-1.0
Range
□ Calcium-binding proteins
Non Calcium-binding proteins


Distribution of output values of *Arabidopsis thaliana* proteome for ANN_pepstats module
Fraction
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.00-0.1
0.11-0.2
0.21-0.3
0.31-0.4
0.41-0.5
0.51-0.6
0.61-0.7
0.71-0.8
0.81-0.9
0.91-1.0
Range
□ Calcium-binding proteins ▨ Non Calcium-binding proteins

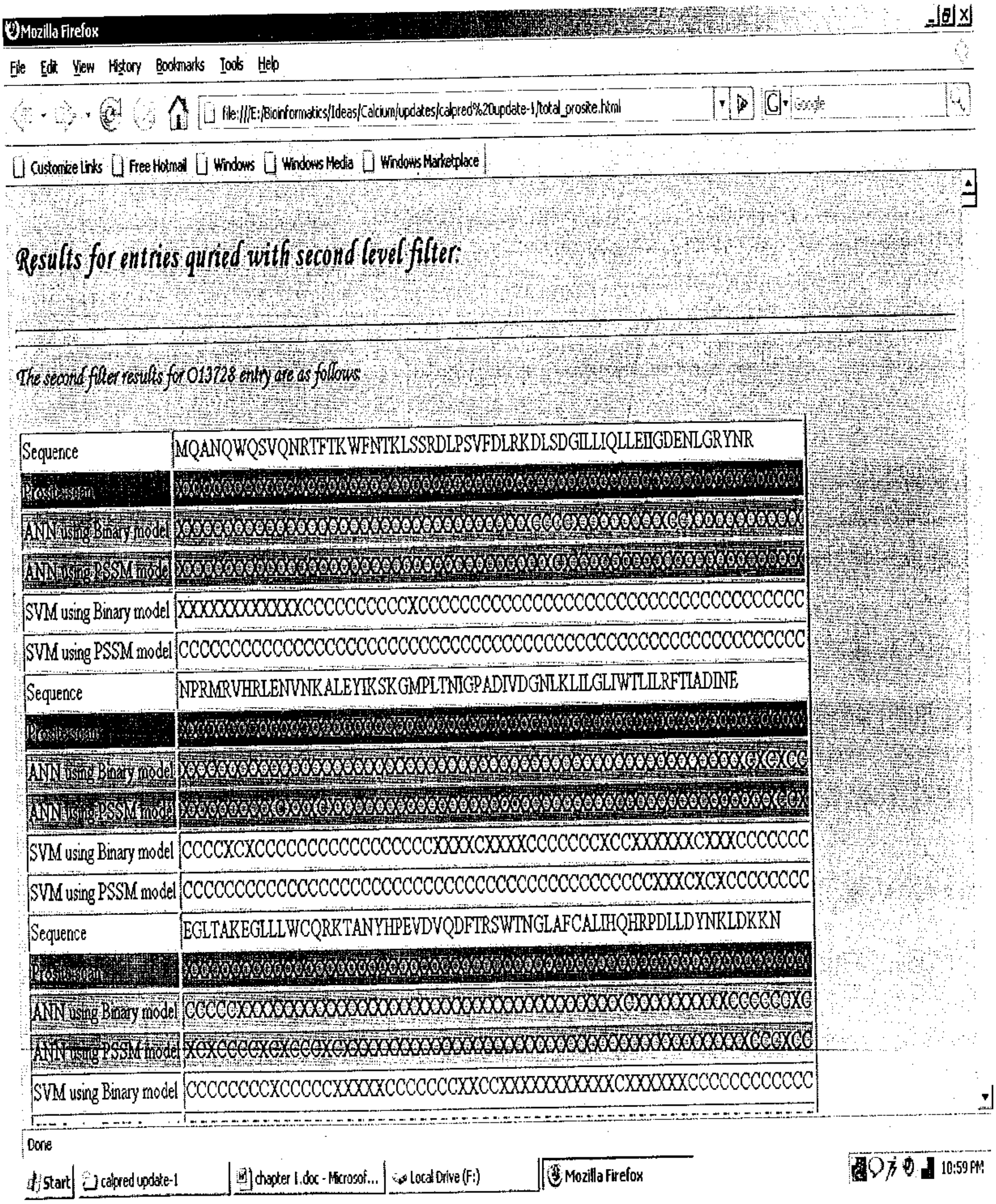

Distribution of output scores of *Arabidopsis thaliana* proteome for SVM_pepstats_radial_basis module
Fraction
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
<(-0.5)
(-0.51)-(-0.4)
(-0.41)-(-0.3)
(-0.31)-(-0.2)
(-0.21)-(-0.1)
(-0.11)-0
0.01-0.1
0.11-0.2
0.21-0.3
0.31-0.4
0.41-0.5
>0.5
Range
▨ Calcium-binding proteins ▨ Non Calcium-binding proteins


Distribution of output scores of *Arabidopsis thaliana* proteome for SVM_pepstats_polynomial module
Fraction
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
<(-0.5)
(-0.51)-(-0.4)
(-0.41)-(-0.3)
(-0.31)-(-0.2)
(-0.21)-(-0.1)
(-0.11)-0
0.01-0.1
0.11-0.2
0.21-0.3
0.31-0.4
0.41-0.5
>0.5
Range
▨ Calcium-binding proteins ▩ Non Calcium-binding proteins


Distribution of output scores of *Arabidopsis thaliana* proteome for SVM_pepstats_linear module
Fraction
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
<(-0.5)
(-0.51)-(-0.4)
(-0.41)-(-0.3)
(-0.31)-(-0.2)
(-0.21)-(-0.1)
(-0.11)-0
0.01-0.1
0.11-0.2
0.21-0.3
0.31-0.4
0.41-0.5
>0.5
Range
▨ Calcium-binding proteins ▩ Non Calcium-binding proteins

APPENDIX-VI

Some snapshots for second-level (203 Kb) filter results for PS00018 dataset.

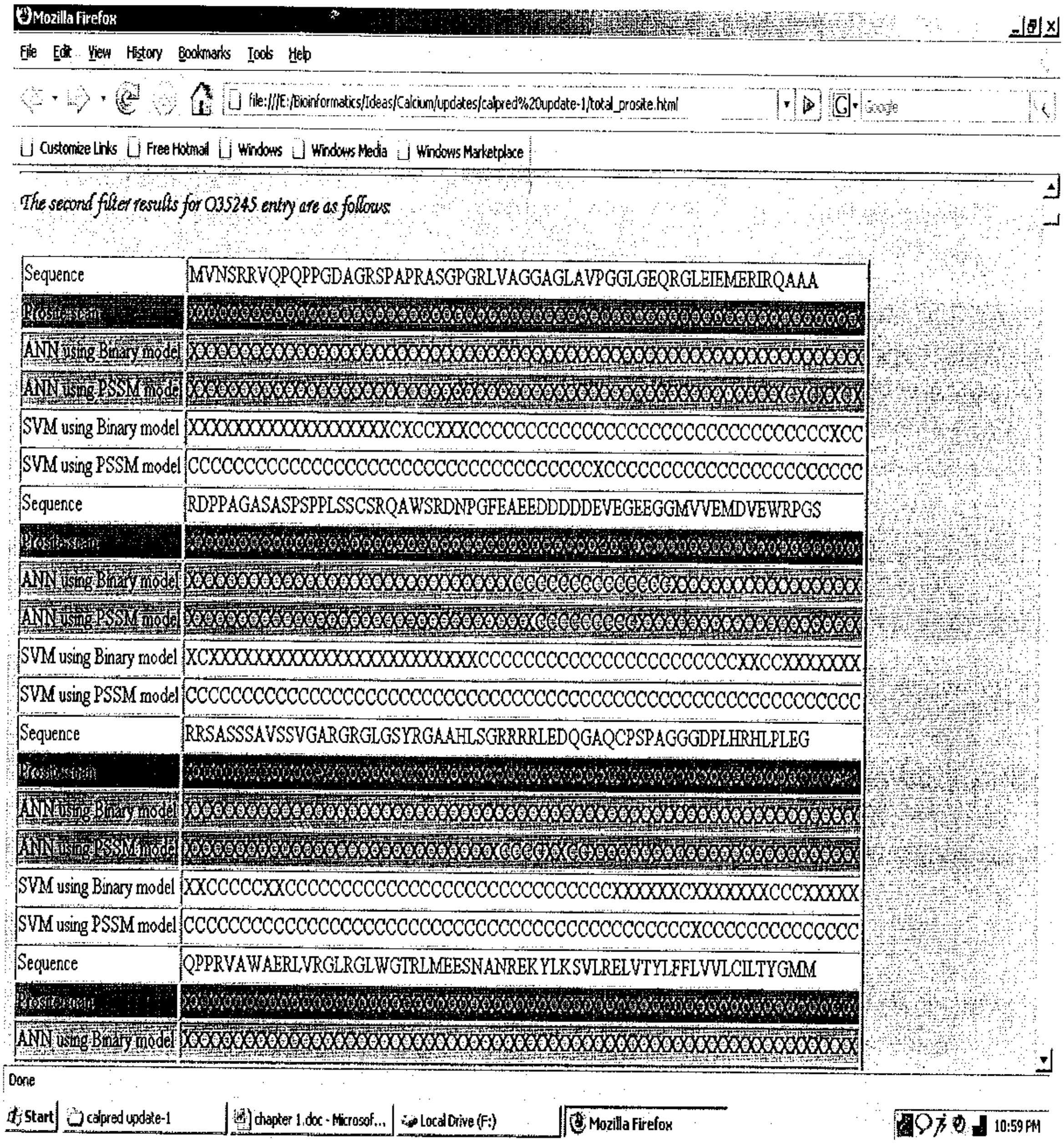


Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///E:/Bioinformatics/Ideas/Calcium/updates/calpred%20update-1/total_prosite.html
Google
Customize Links Free Hotmail Windows Windows Media Windows Marketplace

Results for entries quried with second level filter:

The second filter results for O13728 entry are as follows

Sequence	MQANQWQSVQNRTFTKWFNTKLSSRDLPSVFDLRKDLSDGILLIQLLEIIGDENLGRYNR
Prosite scan	[illegible]
ANN using Binary model	[illegible]
ANN using PSSM model	[illegible]
SVM using Binary model	XXXXXXXXXXXXCCCCCCCCCXCCCCCCCCCCCCCCCCCCCCCCCCCCCC
SVM using PSSM model	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
Sequence	NPRMRVHRLENVNKALEYIKSKGMPLTNIGPADIVDGNLKLILGLIWTLILRFTIADINE
Prosite scan	[illegible]
ANN using Binary model	[illegible]
ANN using PSSM model	[illegible]
SVM using Binary model	CCCCXCXCCCCCCCCCCCCCCXXXXCXXXXCCCCCCCXCCXXXXXXCXXXCCCCCCC
SVM using PSSM model	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCXXXCXCXCCCCCCCC
Sequence	EGLTAKEGLLLWCQRKTANYHPEVDVQDFTRSWTNGLAFCALIHQHRPDLLDYNKLDKKN
Prosite scan	[illegible]
ANN using Binary model	[illegible]
ANN using PSSM model	[illegible]
SVM using Binary model	CCCCCCCCXCCCCCXXXXXCCCCCCCXXCCXXXXXXXXXXXCXXXXXXCCCCCCCCCCC



Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///E:/Bioinformatics/Ideas/Calcium/updates/calpred%20update-1/total_prosite.html
Google
Customize Links
Free Hotmail
Windows
Windows Media
Windows Marketplace
Done
Start
calpred update-1
chapter 1.doc - Microsof...
Local Drive (F:)
Mozilla Firefox
10:59 PM

The second filter results for O35245 entry are as follows:

Sequence	MVNSRRVQPQPPGDAGRSPAPRASGPGRLVAGGAGLAVPGGLGEQRGLEIEMERIRQAAA
Prosite scan	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ANN using Binary model	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ANN using PSSM model	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXCXCXXCX
SVM using Binary model	XXXXXXXXXXXXXXXXXXCXCCXXXCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCXCC
SVM using PSSM model	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCXCCCCCCCCCCCCCCCCC
Sequence	RDPPAGASASPSPPLSSCSRQAWSRDNPGFEAEEDDDDDEVEGEEGGMVVEMDVEWRPGS
Prosite scan	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ANN using Binary model	XXXXXXXXXXXXXXXXXXXXXXXXXXXXCCCCCCCCCCCCCCCCCXXXXXXXXXXXXXXX
ANN using PSSM model	XXXXXXXXXXXXXXXXXXXXXXXXXXXXCCCCCCCCCCCCCCCCCCCXXXXXXXXXXXXX
SVM using Binary model	XCXXXXXXXXXXXXXXXXXXXXXXXXXCCCCCCCCCCCCCCCCCCCCCCXXCCXXXXXXX
SVM using PSSM model	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
Sequence	RRSASSSAVSSVGARGRGLGSYRGAAHLSGRRRRLEDQGAQCPSPAGGGDPLHRHLPLEG
Prosite scan	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ANN using Binary model	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ANN using PSSM model	XXXXXXXXXXXXXXXXXXXXXXXXXXXCCCCXCCXXXXXXXXXXXXXXXXXXXXXXXXXX
SVM using Binary model	XXCCCCCXXCCCCCCCCCCCCCCCCCCCCCCCCCCCCCXXXXXXCXXXXXXXCCCXXXXX
SVM using PSSM model	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCXCCCCCCCCCCCCCC
Sequence	QPPRVAWAERLVRGLRGLWGTRLMEESNANREKYLKSVLRELVTYLFFLVVLCILTYGMM
Prosite scan	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ANN using Binary model	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

BIBLIOGRAPHY

WEB PAGES:

http://expasy.org/prosite/

http://structbio.vanderbilt.edu/cabp_database/