4-13-2017

Prediction of High-Throughput Protein-Protein Interactions for Large Datasets Using Repeated Random Sampling with Replacement of Short Linear Motifs

Behzad Rezaei
University of Windsor

Prediction of High-Throughput Protein-Protein Interactions for Large Datasets Using Repeated Random Sampling with Replacement of Short Linear Motifs

By

Behzad Rezaei

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2017

Prediction of High-Throughput Protein-Protein Interactions for Large Datasets Using Repeated Random Sampling with Replacement of Short Linear Motifs

by

Behzad Rezaei

APPROVED BY:

S. Nkurunziza
Department of Mathematics and Statistics

R. Gras
School of Computer Science

A. Ngom, Advisor
School of Computer Science

L. Rueda, co-Advisor
School of Computer Science

January 12, 2017

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyones copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Wet-lab experimental methods for prediction of Protein-Protein Interactions (PPIs), as a decisive problem in biology, are labor demanding and costly, and usually comprise high false-negative and false-positive rates [20]. Therefore, computational methods have been extensively used as faster, less-expensive and more accurate alternatives [1]. Among all different computational approaches for predicting PPIs, methods based on protein sequences information are more common than the others [16]. While such methods do not need any extra knowledge or data about the proteins rather than their sequences' amino acids information, they have shown to be promising about predicting PPIs [16].

Basically, these methods try to find patterns spread over interacting and non-interacting proteins' sequences, take them as features, and use them for predicting PPIs. Motifs, as common patterns of amino acids between a group of sequences [33], have been recently used for this purpose. There are some algorithms and tools for obtaining motifs from protein sequences. However, most of them have limitations on size of the datasets they can deal with, and also depend on datasets of pre-found motifs. One of the most popular algorithms which is capable of handling big datasets is Multiple EM for Motif Elucidation (MEME). Nevertheless, even for powerful tools like MEME, finding large number of motifs from such datasets would be time-wise infeasible.

We proposed a new method which is able to extract large amount of motifs from a large dataset using MEME, in reasonable period of time. We tested our method on a PPIs dataset of size 5000 (2500 positive and 2500 negative pairs of protein sequences) to obtain 5000 motifs. Then, we used acquired motifs as features to represent our PPI dataset based on them. Finally, using machine learning techniques, we classified our dataset with some of the well-known classifiers like K-nearest neighbour (K-NN), Random Forest, and Support Vector Machine (SVM). Results not only prove the accuracy of our method, which is above 93%, but they also show that the proposed method for finding motifs from big datasets is effective and can be applied for prediction of PPIs.

AKNOWLEDGEMENTS

I would like to express my deepest appreciation to my supervisor Dr. Alioune Ngom, and my co-supervisor Dr. Luis Rueda for their perpetual supports, patience, and inspiration during my Master's program in University of Windsor. It was such an honor for me to know you, and also be in your research team. Thank you so much for giving me an opportunity to learn from you.

Secondly, I would also like to express my gratitude to my committee members Dr. Severien Nkurunziza, and Dr. Robin Gras for their beneficial advices and suggestions to my thesis.

Meanwhile I would like to express my special thanks to my friends for helping me during past two years.

Finally, I would like to express my greatest appreciation to my family for all unconditional love, support, patience, encouragement, and kindness they gave me during my whole life.

DEDICATION

I would like to dedicate my thesis to my dear sisters, brother, father, and specially to my mother who always persuaded me to take this program, but unfortunately I was not lucky enough to have her to see my graduation. Remembering her wonderful and gentle soul will forever remain in my heart. May she rest in peace.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Protein-Protein Interaction

Binding two or more proteins with each other is called protein-protein interaction (PPIs) [26]. There are many indispensable biological processes happening in every living cell, which are affected by PPIs. Thus, to comprehend fundamental systems engaged in cellular precesses studying these biological interactions is very important [28]. In other words, since for many proteins the only way to play their role in a cell is to interact with other companion proteins, Protein-Protein Interactions (PPIs) are, as a result, critical in discerning most of the biological processes existing in the cell [1]. PPIs analysis can assist both anticipating the function of the proteins that have not been discovered yet, and also distinguishing fundamental pathways and processes at the cellular level [1]. Besides, information obtained from protein-protein interactions also helps to describe the function of a protein by its position in the protein-protein interaction network. Having this information will likely make a contribution to finding new drug targets [41].

Basically, common understanding of PPIs is mainly extracted from experimental methods or computational indicator techniques [1]. Experimental methods are costly, labor-intensive, and usually suffering from high false-positive and false-negative rates. Therefore, establishing trustworthy computational methods for predicting PPIs is of great importance [25].

Based on the features that computational methods use for predicting PPIs, they can be practically divided into three categories. First, second and third classes are methods based on sequence information, unification of sequence information and inferior structural

information, and integration of sequence information and 3D structural information, respectively [23]. However, the first method is more common due to the fact that it does not depend on further information about proteins [16].

1.2 Motifs

Patterns outspread over a set of proteins which are functionally linked or may share biological characteristics are called motifs [33]. In other words, motifs are frequent subsequences occurring the most among a group of protein sequences [33]. Motifs can perform in an organized manner to show the complicatedness of practical regulatory inside the cell. Therefore, motif analysis will increase the knowledge about main process that runs protein-protein interactions [18].

1.2.1 Short Linear Motifs

While a motif consists of a sequence pattern of 3-20 amino acids, Short linear motifs (SLiMs) or minimotifs are referred to motifs with length of 3-10 amino acids[11], often with a mixture of fixed positions and wildcards[13].

SLiMs have been found to be decisive due to their capability of domain binding, conversion, cleavage, and targeting, which are all critical in signalling in cells [18]. A motif can be shown in two different ways, with its logo and its regular expression.

FIGURE 1.2.1: A Short Linear Motif (SLiM) of length 10, shown with logo and its corresponding regular expression.

1.2.2 Tools for Finding Motifs

In last decades more research have been done on protein sequences and motifs, because of the important role that PPIs and SLiMs play in the function and formation of a protein. Having different ideas and algorithms finally led to have many databases and tools for extracting motifs from the protein sequences. Most of these databases and tools are presented to the researchers and users only in a web-based platform, whereas some of them have provided a standalone version of their tools as well. The advantage of using web-based tools, specially for small tasks, is that there is no need to go through any installation process, all users have to do is to enter the sequences they want to extract SLiMs from, customize their search parameters, submit the request to the server, and get the results on the browser. However all the web-based tools have limitation on the size of the input dataset.

Some of the main and popular tools for motif discovery are as follows:

- SLiMSearch
- SLiMFinder
- SLiMScape
- Eukaryotic Linear Motif (ELM)
- Minimotif Miner (MnM)
- Nomad
- MEME suite

According to Davey et al. [10], there are different web-based tools like ELM, MnM, SIRW, ScanProsite, and QuasiMotifFinder, which use databases of acknowledged motifs to examine and match known SLiMs with the sequences existing in dataset provided by users. As the authors claim, these tools depend on the libraries of pre-found motifs. They also add that users are able to find new sites of user-specified SLiMs in a group of protein sequences using their web-based designed tool called SLiMSearch. As they state, in their algorithm, first a fast search is done to find equal patterns in order to label new sites of

user-decided SLiMs, then all the sites are scored using pre-schemed databases. Acording to the authors, SLiMSearch is developed to help clarifying the outputs of another SLiM discovery tool designed by the same authors called SLiMFinder.

According to Edwards et al. [13] SLiMFinder is a web-based and also standalone tool suitable for exploring high-throughput motifs. As they explain, SLiMFinder consists of two algorithms named SLiMBuild and SLiMChance. The authors claim that SLiMBuild first finds all the fixed length motifs, and keeps only those have been repeated between the sequences for adequate number of times. Then, it merges the selected motifs to obtain final ones containing wildcards. As they state, SLiMChance algorithm then evaluates probability of gained motifs, fixes their length and arrangements, and finally scores all the motifs.

Both SLiMSearch and SLiMFinder are valuable tools for motif discovery. however, as the authors claim [10] they both are designed for small protein datasets (up to 100 proteins for SLiMFinder [13]).

According to O'Brien et al. [43] SLiMScape is an add-on for Cytoscape which allows users to both search for sites of known motifs and also detecting new motifs. As the authors explain, SLiMScape has two main search tool. First, SLiMFinder which discovers new motifs by investigation a network of protein interactions, and second, SLimSearch which is able to detect sites of known or promising motifs through the same network. As they state, in last step results are illustrated by Cytoscape imagery features. The authors also state that internet connection is necessary while using SLiMScape, since it relies upon some websites like SMART domain database, Uniprot protein database, DBFetch, SLiMFinder, and SLiMSearch.

According to Dinkel et al. [12] Eukaryotic Linear Motif (ELM) has two main parts. First, a database of known motifs, and second, a web-based tool that uses this database to find potential motifs in provided protein sequence dataset. As Gould et al. [15] explain, users can provide protein sequences via ELM main page and will obtain the results of applicant motifs. The authors state that for educational goals, ELM also produces typed analysis and links to the literatures related to the roles that LM plays in the cell.

According to Schiller et al. [37] Minimotif Miner (MnM) is a web-based tool and database for extracting SLiMs from the protein sequences. The authors claim that Minimo-

tif Miner 3.0, the latest version released, has around 300,000 minimotifs from any species. As they explain while MnM 1.0 as the first version of this web service, did not have any filter on false positive cases and used to score the minimotifs based on the complicatedness of the sequences, MnM 2.0 let users to filter the results as needed. As the authors claim, basically filtering will not cut out the false positives, and it just assists the end user to filter the output based on his/her judgement. As the authors state, MnM 3.0 has significantly enhanced over MnM 2.0 in terms of accuracy of the minimotif searching, and also the size of the known minimotifs database.

According to Hernandez et al. [19] Nomad (Neighborhood Optimization for Multiple Alignment Discovery) is a local tool and also web interface that uses hill-climbing strategy and Ungapped Local Multiple Alignment (ULMA) to find non-overlapping fixed-size sites. As they explain, Nomad uses frequency of symbols of each position in ULMA to score obtained occurrences. The authors also claim that Nomad outperforms MEME and Gibbs Site Sampler in cases that sequences are distantly-related.

According to Bailey et al. [4, 3], MEME Suite , which is a local and web-based tool for novel ungapped motif discovery, was first described in 1994 and has been constantly maintained and enhanced for more than 20 years. As the authors explain, after 2010 they enhanced the Expectation Maximization (EM)-based MEME algorithm to take advantage of position-specific priors, which significantly increased MEME's ability of motif detection. They also clarify that position-specific priors have been proved to be a good method to enhance the performance of Gibbs sampler-based motif discovery algorithms. As they explain, MEME finds motifs after progressing three phases. In first phase it chooses starting points for different mixture of requested number-of-sites and motifs length. In the second phase, MEME applies EM optimization algorithm for each of the starting points and generates candidate PSPM for the motifs. In the third phase, MEME scores the candidate PSPMs based on the relative entropy of predicted occurrences, and selects those candidate motifs which have highest scores as the final discovered motifs.

Tool's name	Platform	Type of discovered motif	Limitation
SLiMSearch	web-based	acknowledged	depends on pre-found motifs, designed for small protein datasets up to 100 proteins
SLiMFinder	web-based, standalone	acknowledged	depends on pre-found motifs, designed for small protein datasets up to 100 proteins
SLiMScape	web-based	acknowledged, novel	size of the input dataset
ELM	web-based	acknowledged	depends on pre-found motifs
MnM	web-based	acknowledged	depends on pre-found motifs
Nomad	web-based, standalone	novel	—
MEME	web-based, standalone	novel	—

TABLE 1.2.1: Comparing different motif discovery tools based on their platforms, type of the motif can be discovered by them, and their limitations.

For our experiment we need a motif discovery tool which has two characteristics. First, since our dataset is large and also the number of motifs to be found is large, query can not be processed via web-based platforms. Thus, the tool has to have standalone (local) version with no limitation on the size of either dataset or the number of requested motifs. Second, for prediction of PPIs we need to discover novel motifs. Therefore, the tool should be capable of finding novel motifs. Taking these facts and Table 1.2.1 into consideration, we decided to use MEME and Nomad for our experiments.

1.3 Machine Learning

Machine learning, as one of the most intriguing targets in artificial intelligence, is learning methods that make a machine to be able to anticipate a case correctly, based on previous information [36]. The target of machine learning is to set up rules that make the predictions

as precise as possible [36]. In Machine learning, whenever the class labels of the data are known in advance (supervised learning), classification methods are used for prediction, otherwise (unsupervised learning) clustering techniques are needed [35].

1.4 WEKA as a classification and feature selection tool

Waikato Environment for Knowledge Analysis (WEKA) is one of the best and well-known tools that is widely used for machine learning and data mining [17]. With providing complete selection of machine learning algorithms, and data preparing gadgets, it helps users to easily classify new datasets with different machine learning algorithms and compare the results [17]. Besides, having an open source code has given users the ability of making and developing new projects which helps WEKA to enhance even more [17]. In classification, first a model is learned by training instances, then the learned model is used to classify the new examples into the acknowledged classes [39]. Classification process is as follows :

1. Building a training dataset
2. Analyzing the class feature and classes
3. Analyzing effective attributes for classification
4. Learning the model by the training samples in training set
5. Using the model to analyze the undiscovered data samples [39]

1.4.1 Classification algorithms

While many known classification algorithms (such as NaiveBayes, K-nearest neighbour (KNN), and Random Forest) have been inserted into WEKA and can be easily selected and used for classification, some of them (like Support Vector Machines (SVM)) need to be added to WEKA as new packages (LibSVM). The following is a brief explanation of the classifiers used in our proposed method:

Bayesian classifiers, which rely on Bayes' theory, are statistical classifiers [22]. To facilitate the associated computation, Naive Bayesian classifiers consider that impact of a class or feature value is autonomous and separated from other features [22]. Despite of this feeble independence hypothesis, the results of this classifier have shown to be promising and even in some cases comparable with more complicated methods [31]. Naive Bayes has been found to be efficient in many sensible functions such as medical analysis, and text labelling [31].

K-nearest neighbour (KNN) is one of the aged and elementary classification algorithms, and is one of the best options specially when there is no previous information about the data distribution [29]. KNN uses distance metrics (usually Euclidean distances) to find K-nearest neighbours of unknown samples in the data distribution space, to finally decide which class it belongs to based on the classes of its neighbours [46].

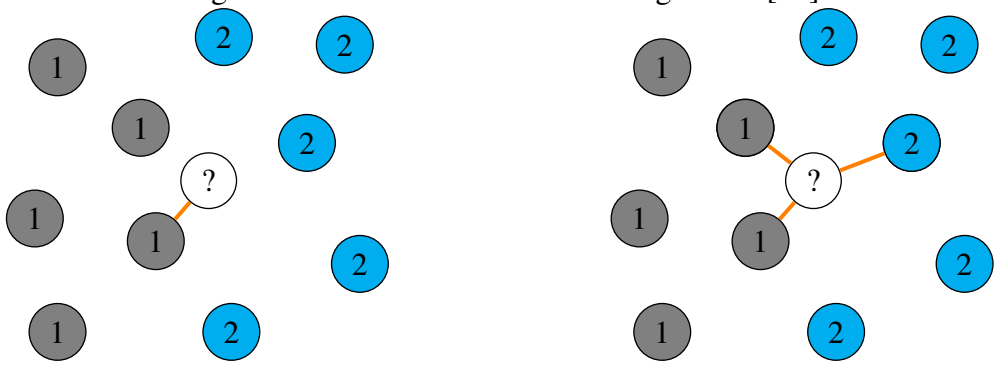

FIGURE 1.4.1: Symbolic view of KNN for K=1, and K=3.

Random Forest (RF) is an ensemble tree-based classifier that applies Bootstrap aggregating (bagging) method to make training sets [7]. It consists of two main techniques. First, random feature subspace which helps to build the trees faster, and second out-of-bag error which increases the chance of assessing the context of the features [7]. Generally, Random Forest is not parametric, has good accuracy, and able to discover importance of the features [32].

Support Vector Machine (SVM) is one of the most broadly used classifications. It performs well with small-scale training datasets, as well as datasets with considerable number of features, and it has a great generalization capacity. SVM is used for both linearly and

non-linearly separable data. Considering that Support Vectors (SV) are referred to the data points that lie on the very edge of each class, for linearly separable cases SVM tries to find a hyperplane that separates classes with the maximum margin [21]. Margin is any positive distance between the hyperplane and any data points. Once the optimum hyperplane is found, the resolution is defined by a linear mixture of involved support vectors [21].

However, the most realistic problems are non-linearly separable. For these cases SVM uses kernels to map the data onto higher dimensional space and then tries to find separating hyperplane in the new space [21]. Thus, a linear solution in the new feature space corresponds to non-linear function in the initial space [21]. There are different kernels (like Polynomial, Radial Basis Function (RBF), Spline) with various options implemented for SVM. Therefore, in order to find the one that works the best for a given dataset, testing kernels with different combination of their options is suggested.

1.4.2 Feature selection

Since many of the classification methods are not initially devised to deal with multiple unrelated features, it is essential to mix them with feature selection (FS) methods [35]. Feature selection, which can be applied on both supervised and unsupervised learning, has three main goals:

1. to prevent over-fitting and promote model and prediction efficiency
2. to produce more agile and cost-efficient models
3. to obtain better understanding of how data is created [35]

However, when a feature selection method is trying to find a subgroup of related features, it brings an extra level of complicatedness to the modelling process [35]. There are three different FS methods For classification purposes, filter, wrapper, and embedded. Each one of them has its own advantages and disadvantage and based on the data being classified one may work better [35].

1.4.3 Evaluation method

The method that is usually used for evaluating classifiers is called M-fold cross validation. In M-fold cross validation, which is a generalization of cross validation, first dataset is divided into *m* disarranged sets of balanced size. Then the classifier is trained *m* times such that at each iteration *m-1* folds are used to train the dataset and 1 fold is left out for testing. Finally, measures obtained from all M-folds are averaged (Figure 1.4.2).

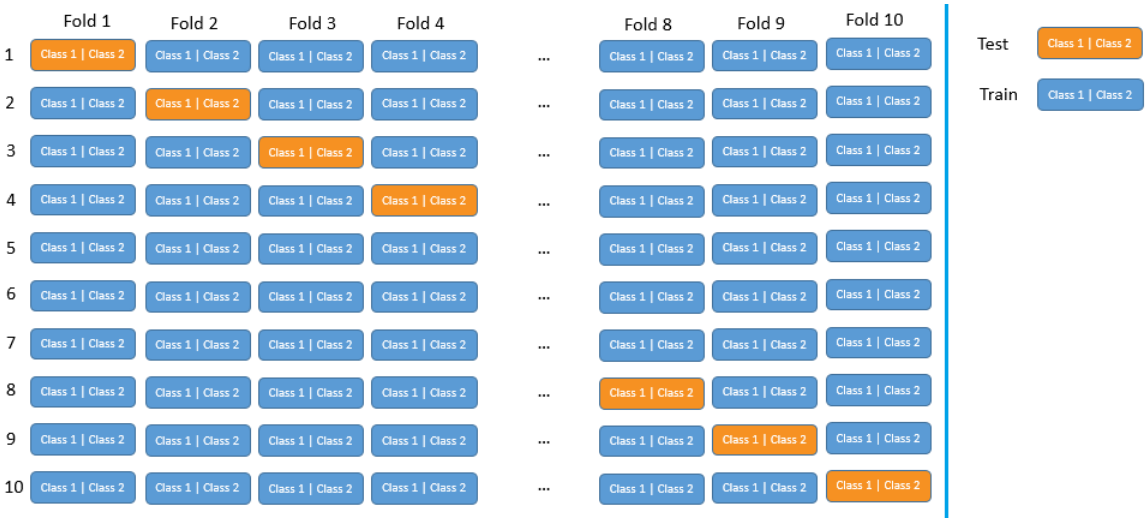

FIGURE 1.4.2: *10*-fold cross validation scheme.

Measures usually used for evaluating performance of a classifier are *Accuracy*, *Sensitivity* (*Recall*), *Specificity*, *Precision*, and *Matthews Correlation Coefficient* (MCC) which all are computed based on *Confusion Matrix*. Considering we have two classes, positive and negative, confusion matrix consists of four elements, TP, TN, FP, and FN [45]:

- True Positives (TP) are positive samples classified as positive
- True Negatives (TN) are negative samples classified as negative
- False Positives (FP) are negative samples classified as positive
- False Negatives (FN) are positive samples classified as negative

We have the following formulas [45] :

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1.4.1}$$

$$Sensitivity(Recall) = \frac{TP}{TP + FN} \tag{1.4.2}$$

$$Specificity = \frac{TN}{FP + TN} \tag{1.4.3}$$

$$Precision = \frac{TP}{TP + FP} \tag{1.4.4}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{1.4.5}$$

Sensitivity (Recall) shows true positive rate, Specificity reveals true negative rate, Precision offers positive predicted value [45], and MCC, which is a value between -1 and +1, presents the correlation coefficient among the classes and predicted samples. Where -1 means no correlation between predicted classes and actual classes, 0 means the performance of the classifier is not better than randomly classifying the samples, and +1 means precise prediction.

1.5 Motivation of this Thesis

Wet-lab experimental methods for prediction of Protein-Protein Interactions (PPIs), as an important problem in biology, are labor demanding and costly, and usually comprise high false-negative and false-positive rates [20]. Therefore, computational methods have been extensively used as faster, less-expensive and more accurate alternatives [1]. Among all different computational approaches for predicting PPIs, methods based on protein sequences information are more common than the others [16].

Basically, these methods try to find patterns spread over interacting and non-interacting proteins' sequences, take them as features, and use them for predicting PPIs. Motifs, as common patterns of amino acids between a group of sequences [33] , have recently been used for this purpose.

Many different tools have been developed for motif discovery. However, most of them usually have two major drawbacks for predicting PPIs using novel motifs. First, they have been designed to match possible motifs with a dataset of known motifs instead of uncovering new motifs, which leads to using motifs that are less appropriate for PPI prediction. Second, they have limitations on the size of the input dataset as well as on number of requested motifs, which leads to not having enough features for PPI prediction. Furthermore, even for a few tools that do not have these disadvantages (such as MEME), finding only hundreds of motifs may take months or years. Considering the fact that researchers in this area always tend to enlarge the dataset they are working on to obtain as close results as possible to the actual PPI datasets, existing boundaries have always inhibited them from achieving their goal.

In this thesis, we propose a method for obtaining large number of motifs from a large dataset of interacting and non-interacting proteins using MEME in much faster time. We obtained 5000 motifs from a database of size 5000 (pairs of proteins) and used discovered motifs to predict PPIs using classification methods. Our method proves to be encouraging specially considering the time spent to uncover 5000 novel motifs, and indicates that SLiMs are highly suitable for accurate prediction of PPIs.

In Chapter 2 we review some of the related works for predicting PPIs using SLiMs, and in Chapters 3, 4, and 5 we discuss the proposed method, results, and conclusion respectively.

CHAPTER 2

Literature review

Among all the studies that have been done regarding using protein sequence information for predicting protein-protein interactions (PPI), using SLiMs has been the most popular. In this chapter we review some of the literature about short linear motifs for prediction of PPIs.

2.1 Approaches for Prediction of PPIs

2.1.1 Prediction of High-throughput Protein-Protein Interactions and Calmodulin-Binding Using Short-Linear Motifs

According to Y. Li [24], prediction of protein-protein interactions (PPIs) and also Calmodulin Binding Proteins (CaM-binding) are two vital problems in biology. Existing methods for PPIs prediction are not usually precise enough and suffer from significant rates of FP and FN, besides developed methods for CaM-binding prediction are not advanced enough. The author states that in proposed method novel SLiMs found by MEME are used to predict PPIs and CaM-binding.

Previous work and shortcomings by others referred to by the authors

The author of [24] refers to related work that addresses the related problem of prediction of PPIs using information from simple codon pairs [48], and prediction of PPIs using information from protein sequences [25] and states that the shortcomings of previous work are poor coverage and low accuracy which is around 80%.

The new idea that the authors proposed

As Y. Li [24] claims, she has applied two different methods for obtaining SLiMs using MEME tool, SM and CM. She explains that SM means obtaining SLiMs from interacting and non-interacting datasets individually, while CM means putting both datasets together for SLiMs discovery. As the author describes, first she has applied SM and CM on both PPIs and CaM-binding datasets to discover 100 novel SLiMs for each set, then she scores the occurrences (sites) with 5 different functions, and finally uses gained scores to build the final datasets for prediction.

Materials and methods

As the author claims [24], for PPIs dataset 50 protein pairs has been downloaded from PrePPI dataset and set as positive samples, and 38 negative protein pairs has been obtained from Negatome Database version 2.0. On the other hand, for CaM-binding dataset 194 positive samples have been chosen from Calmodulin Target Database and 193 negative instances have been selected from Uniprot database. As the author explains, in proposed method MEME tool is used for discovering 100 motifs of length 3 to 10 for each dataset and each method (SM, and CM). Then 5 different methods have been used for scoring the occurrences (sites) as follows:

1. Counting sites

2. Scoring sites with I formula

$$I(a|\ X) = -\sum_{i=1}^{l} P(a_i) \times \log(P(a_i)) \tag{2.1.1}$$

3. Scoring sites with $\hat{\text{I}}$ formula

$$\hat{I}(a|\ X) = -\frac{1}{l} \times \sum_{i=1}^{l} P(a_i) \times \log(P(a_i)) \tag{2.1.2}$$

4. Scoring sites with $\hat{\text{I}}$ formula / counting of sites

5. SlidingWindow Scoring method

As the author clarifies, first method simply tries to find obtained motifs in original protein sequences and count the number of occurrences, while in second and third methods mentioned formulas has been used for calculating the scores of sites. As the author states [24], "in the formulas X is the profile sequence, $P(a_i)$ is the probability (of the i^{th} residue of a)", and *l* is the length of the motif. According to the author, while the fourth method is actually the third scoring formula divided by the first one with the aim of checking the effects of counting sites on the Î formula, the fifth method considers every sub-sequence of length *l* in a sequence to likely be a site. Thus, a Sliding Window has been used to score all the possible sites.

According to the author, in next step final datasets has been built based on the obtained scores, and different classification methods such as SVM-Polynomial kernel, Random Forest, KNN, and Multilayer Perceptron have been applied on the original dataset as well as dataset filtered by feature selection method.

Results that the authors claim to have achieved

As the author of [24] states, since the results of all 5 different scoring functions were almost similar, results of the best method (Sliding Window Scoring (SWS) is illustrated only. The Author claims to have obtained the following results for PPI prediction using proposed methods on the datasets mentioned above:

Dataset for Classification	Classifier	Accuracy (%)	Recall (%)	MCC
S score matrix	SVM-Polynomial ($c = 1, g = 0$)	76.1	76.1	0.525
	Random Forest	76.1	76.1	0.509
	k-NN ($k = 1$)	75.0	75.0	0.488
	Decision Tree	64.8	64.8	0.270
	Multilayer Perceptron	**79.5**	79.5	0.586
T score matrix	SVM-Polynomial ($c = 1, g = 0$)	79.5	79.5	0.583
	Random Forest	73.9	73.9	0.462
	k-NN ($k = 1$)	**81.8**	81.8	0.651
	Decision Tree	67.0	67.0	0.331
	Multilayer Perceptron	78.4	78.4	0.557
S score matrix subset selected by FS	SVM-Polynomial ($c = 1, g = 0$)	78.4	78.4	0.562
	Random Forest	79.5	79.5	0.580
	k-NN ($k = 1$)	**80.7**	80.7	0.611
	Decision Tree	75.0	75.0	0.499
	Multilayer Perceptron	79.5	79.5	0.591
T score matrix subset selected by FS	SVM-Polynomial ($c = 1, g = 0$)	**84.1**	84.1	0.675
	Random Forest	81.8	81.8	0.629
	k-NN ($k = 1$)	81.8	81.8	0.636
	Decision Tree	79.5	79.5	0.581
	Multilayer Perceptron	79.5	79.5	0.583

FIGURE 2.1.1: Classification results for the score matrices with SLiMs obtained from the CM approach.

Dataset for Classification	Classifier	Accuracy (%)	Recall (%)	MCC
S score matrix	SVM-Polynomial ($c = 1, g = 0$)	73.9	73.9	0.492
	Random Forest	72.7	72.7	0.439
	k-NN ($k = 1$)	72.7	72.7	0.439
	Decision Tree	58.0	58.0	0.146
	Multilayer Perceptron	**81.8**	81.8	0.632
T score matrix	SVM-Polynomial ($c = 1, g = 0$)	56.8	56.8	0.000
	Random Forest	**78.4**	78.4	0.564
	k-NN ($k = 1$)	77.3	77.3	0.533
	Decision Tree	63.6	63.6	0.244
	Multilayer Perceptron	70.5	70.5	0.390
S score matrix subset selected by FS	SVM-Polynomial ($c = 1, g = 0$)	70.5	70.5	0.392
	Random Forest	**78.4**	78.4	0.558
	k-NN ($k = 1$)	72.7	72.7	0.453
	Decision Tree	77.3	77.3	0.537
	Multilayer Perceptron	77.3	77.3	0.545
T score matrix subset selected by FS	SVM-Polynomial ($c = 1, g = 0$)	56.8	56.8	0.000
	Random Forest	75.0	75.0	0.486
	k-NN ($k = 1$)	79.5	79.5	0.587
	Decision Tree	75.0	75.0	0.491
	Multilayer Perceptron	**80.7**	80.7	0.604

FIGURE 2.1.2: Classification results for the score matrices with SLiMs obtained from the SM approach.

		C=1	C=10	C=100	C=1,000
S score matrix	gamma=0	61.4	61.4	61.4	61.4
	gamma=0.01	61.4	61.4	61.4	61.4
	gamma=0.1	61.4	61.4	61.4	61.4
	gamma=1	61.4	61.4	61.4	61.4
	gamma=10	61.4	61.4	61.4	61.4
	gamma=100	61.4	61.4	61.4	61.4
	gamma=1,000	61.4	61.4	61.4	61.4
T score matrix	gamma=0	56.8	56.8	60.2	71.6
	gamma=0.01	56.8	56.8	60.2	71.6
	gamma=0.1	56.8	60.2	71.6	76.1
	gamma=1	59.1	71.6	76.1	71.6
	gamma=10	71.6	76.1	72.7	70.5
	gamma=100	78.4	78.4	79.5	79.5
	gamma=1,000	62.5	63.6	63.6	63.6
S score matrix subset selected by FS	gamma=0	61.4	61.4	61.4	61.4
	gamma=0.01	63.6	63.6	63.6	63.6
	gamma=0.1	61.4	61.4	61.4	61.4
	gamma=1	61.4	61.4	61.4	61.4
	gamma=10	61.4	61.4	61.4	61.4
	gamma=100	61.4	61.4	61.4	61.4
	gamma=1,000	61.4	61.4	61.4	61.4
T score matrix subset selected by FS	gamma=0	56.8	56.8	58.0	69.3
	gamma=0.01	56.8	56.8	56.8	56.8
	gamma=0.1	56.8	56.8	56.8	69.3
	gamma=1	56.8	56.8	69.3	78.4
	gamma=10	56.8	69.3	**84.1**	83.0
	gamma=100	72.7	**84.1**	79.5	78.4
	gamma=1,000	75.0	76.1	68.2	65.9

FIGURE 2.1.3: Prediction of PPIs using SVM-Polynomial (C = 1, 10, 100, 1000, gamma = 0.01, 0.1, 0, 1, 10, 100, 1000) with SLiMs obtained from SM.

		C=1	C=10	C=100	C=1,000
S score matrix	gamma=0	76.1	76.1	76.1	76.1
	gamma=0.01	76.1	76.1	76.1	76.1
	gamma=0.1	76.1	76.1	76.1	76.1
	gamma=1	76.1	76.1	76.1	76.1
	gamma=10	76.1	76.1	76.1	76.1
	gamma=100	76.1	76.1	76.1	76.1
	gamma=1,000	76.1	76.1	76.1	76.1
T score matrix	gamma=0	79.5	78.4	77.3	77.3
	gamma=0.01	68.2	79.5	76.1	77.3
	gamma=0.1	77.3	77.3	77.3	77.3
	gamma=1	77.3	77.3	77.3	77.3
	gamma=10	77.3	77.3	77.3	77.3
	gamma=100	77.3	77.3	77.3	77.3
	gamma=1,000	77.3	77.3	77.3	77.3
S score matrix subset selected by FS	gamma=0	78.4	78.4	78.4	78.4
	gamma=0.01	78.4	78.4	78.4	78.4
	gamma=0.1	78.4	78.4	78.4	78.4
	gamma=1	78.4	78.4	78.4	78.4
	gamma=10	78.4	78.4	78.4	78.4
	gamma=100	78.4	78.4	78.4	78.4
	gamma=1,000	78.4	78.4	78.4	78.4
T score matrix subset selected by FS	gamma=0	84.1	84.1	**86.4**	84.1
	gamma=0.01	56.8	56.8	60.2	85.2
	gamma=0.1	85.2	84.1	85.2	**86.4**
	gamma=1	**86.4**	80.7	78.4	77.3
	gamma=10	79.5	79.5	79.5	79.5
	gamma=100	75.0	75.0	75.0	75.0
	gamma=1,000	79.5	79.5	79.5	79.5

FIGURE 2.1.4: Prediction of PPIs using SVM-Polynomial (C = 1, 10, 100, 1000, gamma = 0.01, 0.1, 0, 1, 10, 100, 1000) with SLiMs obtained from CM.

As the author states, the best result for PPIs prediction is obtained from classifying the dataset using SVM-Polynomial (gamma=1, and cost=1) with 86.4% accuracy. As she claims, results show that generally CM method works better than SM for PPI prediction.

The Author also claims to have obtained the following results for CaM-Binding prediction using her methods on the datasets mentioned above:

Dataset for Classification	Classifier	Accuracy (%)	Recall (%)	MCC
S score matrix	SVM-Polynomial ($c = 1, g = 0$)	72.6	72.6	0.453
	Random Forest	73.1	73.1	0.463
	k-NN ($k = 1$)	**80.6**	80.6	0.612
	Decision Tree	72.9	72.9	0.466
	Multilayer Perceptron	76.0	76.0	0.533
T score matrix	SVM-Polynomial ($c = 1, g = 0$)	55.0	55.0	0.105
	Random Forest	68.5	68.5	0.375
	k-NN ($k = 1$)	59.7	59.7	0.275
	Decision Tree	68.2	68.2	0.364
	Multilayer Perceptron	**75.7**	75.7	0.518
S score matrix subset selected by FS	SVM-Polynomial ($c = 1, g = 0$)	56.1	56.1	0.122
	Random Forest	**77.8**	77.8	0.556
	k-NN ($k = 1$)	77.0	77.0	0.542
	Decision Tree	74.2	74.2	0.495
	Multilayer Perceptron	76.2	76.2	0.545
T score matrix subset selected by FS	SVM-Polynomial ($c = 1, g = 0$)	64.9	64.9	0.297
	Random Forest	**69.3**	69.3	0.385
	k-NN ($k = 1$)	66.4	66.4	0.330
	Decision Tree	66.7	66.7	0.334
	Multilayer Perceptron	68.0	68.0	0.360

FIGURE 2.1.5: Prediction of CaM-binding proteins classification results for the score matrices with SLiMs obtained from SM.

Dataset for Classification	Classifier	Accuracy (%)	Recall (%)	MCC
S score matrix	SVM-Polynomial ($c = 1, g = 0$)	72.6	72.6	0.453
	Random Forest	74.7	74.7	0.494
	k-NN ($k = 1$)	**78.3**	78.3	0.566
	Decision Tree	71.3	71.3	0.436
	Multilayer Perceptron	76.5	76.5	0.553
T score matrix	SVM-Polynomial ($c = 1, g = 0$)	57.6	57.6	0.213
	Random Forest	69.3	69.3	0.395
	k-NN ($k = 1$)	58.1	58.1	0.261
	Decision Tree	65.1	65.1	0.303
	Multilayer Perceptron	**71.3**	71.3	0.436
S score matrix subset selected by FS	SVM-Polynomial ($c = 1, g = 0$)	62.0	62.0	0.240
	Random Forest	**80.1**	80.1	0.603
	k-NN ($k = 1$)	78.6	78.6	0.571
	Decision Tree	72.1	72.1	0.455
	Multilayer Perceptron	77.0	77.0	0.560
T score matrix subset selected by FS	SVM-Polynomial ($c = 1, g = 0$)	60.2	60.2	0.210
	Random Forest	**70.5**	70.5	0.415
	k-NN ($k = 1$)	68.7	68.7	0.382
	Decision Tree	68.7	68.6	0.379
	Multilayer Perceptron	68.5	68.5	0.370

FIGURE 2.1.6: Prediction of CaM-binding proteins classification results for the score matrices with SLiMs obtained from CM.

		C=1	C=10	C=100	C=1,000
	gamma=0	72.6	72.6	72.6	72.6
	gamma=0.01	72.6	72.6	72.6	72.6
S score	gamma=0.1	72.6	72.6	72.6	72.6
	gamma=1	72.6	72.6	72.6	72.6
matrix	gamma=10	72.6	72.6	72.6	72.6
	gamma=100	72.6	72.6	72.6	72.6
	gamma=1,000	72.6	72.6	72.6	72.6
	gamma=0	55.0	69.8	**75.2**	73.6
	gamma=0.01	55.0	70.3	**75.5**	73.4
T score	gamma=0.1	73.4	70.5	68.0	68.7
	gamma=1	68.7	68.7	68.7	68.7
matrix	gamma=10	68.7	68.7	68.7	68.7
	gamma=100	68.7	68.7	68.7	68.7
	gamma=1,000	68.7	68.7	68.7	68.7
	gamma=0	45.0	45.0	45.0	45.0
	gamma=0.01	42.4	69.3	61.5	61.5
S score	gamma=0.1	62.8	62.8	62.8	62.8
matrix subset	gamma=1	45.0	45.0	45.0	45.0
selected by FS	gamma=10	49.9	49.9	49.9	49.9
	gamma=100	48.3	48.3	48.3	48.3
	gamma=1,000	56.1	56.1	56.1	56.1
	gamma=0	53.0	62.5	62.5	62.5
	gamma=0.01	53.0	53.0	53.0	53.0
T score	gamma=0.1	53.0	62.5	62.5	62.5
matrix subset	gamma=1	63.0	66.9	67.2	64.6
selected by FS	gamma=10	66.7	66.4	66.9	65.4
	gamma=100	64.3	63.6	58.4	64.3
	gamma=1,000	58.7	66.4	64.1	61.8

FIGURE 2.1.7: Prediction of CaM-binding proteins using SVMPolynomial (C = 1, 10, 100, 1000, gamma = 0.01, 0.1, 0, 1, 10, 100, 1000) with SLiMs obtained from SM.

		C=1	C=10	C=100	C=1,000
	gamma=0	72.6	72.6	72.6	72.6
	gamma=0.01	72.6	72.6	72.6	72.6
S score	gamma=0.1	72.6	72.6	72.6	72.6
	gamma=1	72.6	72.6	72.6	72.6
matrix	gamma=10	72.6	72.6	72.6	72.6
	gamma=100	72.6	72.6	72.6	72.6
	gamma=1,000				
	gamma=0	57.6	71.3	**72.9**	71.3
	gamma=0.01	57.6	71.8	**72.9**	71.3
T score	gamma=0.1	71.3	68.5	70.3	70.3
	gamma=1	70.3	70.3	70.3	70.3
matrix	gamma=10	70.3	70.3	70.3	70.3
	gamma=100	70.3	70.3	70.3	70.3
	gamma=1,000	70.3	70.3	70.3	70.3
	gamma=0	62.0	62.0	62.0	62.0
	gamma=0.01	44.7	44.7	44.7	44.7
S score	gamma=0.1	52.7	52.7	52.7	52.7
matrix subset	gamma=1	49.1	49.1	49.1	49.1
selected by FS	gamma=10	66.9	66.9	66.9	66.9
	gamma=100	55.3	55.3	55.3	55.3
	gamma=1,000	33.6	33.6	33.6	33.6
	gamma=0	60.2	63.3	64.6	65.6
	gamma=0.01	58.1	58.1	58.1	60.2
T score	gamma=0.1	60.2	63.3	64.6	65.6
matrix subset	gamma=1	65.6	69.5	73.9	73.1
selected by FS	gamma=10	70.8	66.7	68.5	69.5
	gamma=100	71.8	69.8	67.4	69.8
	gamma=1,000	66.4	65.6	65.9	63.8

FIGURE 2.1.8: prediction of CaM-binding proteins using SVMPolynomial (C = 1, 10, 100, 1000, gamma = 0.01, 0.1, 0, 1, 10, 100, 1000) with SLiMs obtained from CM.

As the author states, the best result for CaM-Binding prediction is obtained from classifying the dataset using KNN (k=1) with 80.6% accuracy on SM method. She also claims that feature selection had an essential effects on most of the applied classifiers such that

they got better accuracy compared to result obtained from original dataset.

2.1.2 A model based on minimotifs for classification of stable protein-protein complexes

According to L.Rueda et al. [34] prediction of obligate and non-obligate proteins is an important problem in biology. As they explain, between all different existing problems in this matter, they aimed their attention at the problem of distinguishing the immobility of protein structure, and the transportation from non-obligate to obligate. As the authors clarify, obligate interactions are easier to investigate due to the fact that they are continuous, while non-obligate interactions are acknowledged to be either long-lasting or short-term.

Previous work and shortcomings by others referred to by the authors

The authors [34] refer to the related work that addresses the related problem of prediction of protein-protein interaction types using association rule based classification [27], and prediction of biological protein-protein interactions using atom-type and amino acid properties [2], and also predicting and analyzing protein-protein interaction types using electrostatic energies [44], and state that the shortcomings of previous work are first, methods using protein structures are restricted to structural information of the proteins which with current knowledge is accessible for a small number of proteins, and second such methods are basically slow and time-absorbing.

The new idea that the authors proposed

The authors [34] state that their proposed method uses short linear motifs (SLiMs) for PPI prediction. As they explain, their method uses an information-content-based scoring function that creates features for both obligate and non-obligate samples by scoring SLiMs obtained by MEME. As they explain, k-NN, LDR and SVM classification methods, and cross validation and leave-one-out validation methods have been used for evaluating their method.

Materials and methods

As the authors [34] claim, they have used two different datasets named ZH and MW. As they explain, ZH includes 75 obligate and 62 non-obligate curated pairs of proteins obtained from [49], while MW contains 115 obligate and 212 non-obligate curated pairs of protein. They also explain that they have used MEME to obtain two series of 1000 SLiMs, first of length 3-10 , and second of length 2-7 for each ZH and MW, separately.

According to the authors [34], after discovering the motifs each sequence is divided into all attainable overlapping small frames of length *l* (equal to SLiMs length), then using uncovered SLiMs information content for each frame is determined by the following formula, and finally best 20 values are used to build 20 feature vector for each pair of protein.

$$\hat{I}(a|\ X) = -\frac{1}{l} \times \sum_{i=1}^{l} P(a_i) \times \log(P(a_i)) \tag{2.1.3}$$

As the authors add, since log(1) = 0, in order to avoid losing information the following cases have been applied while scoring the frames:

$$\log P(a_i) = \begin{cases} log(0.99) & \text{if } P(a_i) = 1 \\ log(P(a_i)) & \text{otherwise} \end{cases} \tag{2.1.4}$$

As the authors [34] state, for classification part two validation methods have been used. As they explain first method is leave-one-out approach which is used by a k-NN classifier. As they add, in this technique a pair of protein is picket for classification, then k-NN using Euclidean distance discovers the nearest neighbour among the rest of the pairs that have not been classified yet, and Second method is cross-validation used with SVM (with linear kernel) and LDR classifiers to evaluate the efficiency of their proposed method. As they mention, for this method they used SLiMs obtained from MW training set to test the ZH, and the other way around for testing MW. They also state that obtained results not only show the capability of the proposed method to amend the PPIs prediction, but also presents

its capacity of generalization and independence of acknowledged features.

Results that the authors claim to have achieved

The Author claims to have obtained the following results using their methods on the datasets mentioned above:

	length (ℓ)	ZH (%)	MW (%)
$\ell = 3-10$ 1,000 SLiMs	10	98.54	98.77
	9	99.27	99.07
	8	95.62	99.27
	7	99.27	96.31
	6	99.27	96.62
	5	99.27	99.27
	4	99.27	97.54
$\ell = 2-7$ 1,000 SLiMs	7	98.35	98.46
	6	98.54	96.62
	5	99.27	98.77
	4	96.35	99.27
	3	93.43	65.64

FIGURE 2.1.9: KNN classification results for the datasets using PPI-SLIM-SEQ.

	(ℓ)	SVM	LDR	
			Quadratic	Linear
ZH Dataset MW SLiMs	5	95.62	97.81	97.08
	4	97.81	99.27	97.81
MW Dataset ZH SLiMs	6	98.77	98.77	98.47
	5	98.47	99.08	98.47

FIGURE 2.1.10: SVM and LDR classification results for the ZH and MW datasets with the MW and ZH SLiMs respectively.

	NOXClass	Aziz *et al.* [7]	Vasudev and Rueda [8]	PPI-SLiM -Seq with *k*-NN
Zhu *et al.* [6]	88.32	82.13	96.17	99.27
Mintseris *et al.* [17]		80.86	97.36	99.07

FIGURE 2.1.11: Comparison of classification accuracy with other related works.

As the authors state, using KNN (with k = 1,5,10,15,20,25,30) and leave-one-out validation method, while almost all the results are more than 93%, the best accuracy obtained

is 99.27% for l=9,7,6,5 for ZH dataset and the same accuracy for l=8,5 for MW dataset. They also claim that best obtained results for SVM is 98.77% accuracy which is obtained from MW dataset, using ZH SLiMs and with l=6, while for LDR classifier the best gained result is for quadratic kernel with 99.27% accuracy from the ZH dataset, using MW SLiMs.

2.2 Inspiration from the Previous Works

The main inspiration from previous works comes from the size of the datasets and the number of discovered motifs that they used in their experiment. In both experiments, these two factors were tried to be kept small enough in order to make the motif discovery part feasible, while removing this limitation will help the experiment to be more realistic and enriched. Thus, we decided to propose a method to deal with this problem, such that prediction of PPIs can be done using much larger number of motifs obtained from much larger datasets.

CHAPTER 3

Materials and Methods

In this chapter we describe the dataset and methods we used in our experiments step by step. As illustrated in Figure 3.0.1, Java is used for most of the parts that required text-file processing and scoring. We used Python to extract the sequences from the corresponding proteins XML files downloaded from UniProt website [www.Uniprot.org] (a hub for protein information) [9]. BLASTp software and also Java used for purifying the dataset, and MEME is used for obtaining the SLiMs. We benefited from java.regex for finding the sites, and used WEKA for classification and feature selection purposes.

3.1 Datasets

The datasets used in our experiment have been created by selecting samples from known interacting and non-interacting PPIs datasets. The interacting (we refer them as positive) pairs (Figure 3.1.1) have been selected from ”New Human Protein Interaction Set” of PrePPI (a structure-informed database of proteinprotein interactions) [47] database, while non-interacting (we refer them as negative) complexes (Figure 3.1.2) have been selected from Negatome (a database of non-interacting proteins) version 2.0 [5] . PrePPI has 23779 pairs of interacting proteins, while Negatome v 2.0 only has 4397 non-interacting protein pairs.

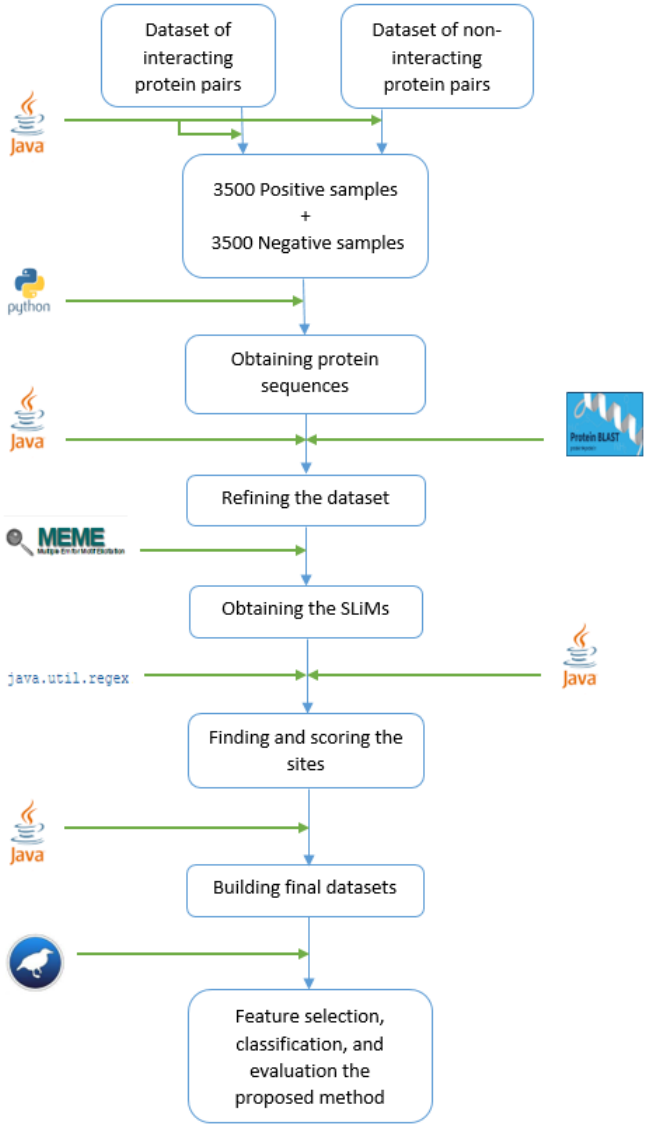

FIGURE 3.0.1: Scheme of the steps involved in the proposed method, and the tools have been used for each part.

DBHSN0000007	A0JLT2|O00571	IntAct	pubmed:15175163
DBHSN0000008	A0JLT2|O14645	IntAct	pubmed:15175163
DBHSN0000030	A0JLT2|P26599	IntAct	pubmed:15175163
DBHSN0000033	A0JLT2|P31689	IntAct	pubmed:15175163
DBHSN0000186	A8MT69|O15360	BioGRID	pubmed:20347429
DBHSN0000187	A8MT69|P04637	HPRD	pubmed:17347673
DBHSN0000188	A8MT69|P62988	BioGRID	pubmed:11278694
DBHSN0000190	A8MT69|Q13547	BioGRID	pubmed:10737769
DBHSN0000191	A8MT69|Q96ST3	BioGRID	pubmed:10737769
DBHSN0000192	A8MT69|Q9NW38	BioGRID	pubmed:20347429
DBHSN0000196	A9YTQ3|P03372	BioGRID	pubmed:18565642
DBHSN0000197	A9YTQ3|P56524	BioGRID	pubmed:17949687

FIGURE 3.1.1: Samples from PrePPI dataset. Each row indicates there is an interaction between specified pair of proteins.

P01636	P98002	1ar1	MI:0114	x-ray crystallography
P01644	P38383	2zjs	MI:0114	x-ray crystallography
P01647	P07143	1ezv	MI:0114	x-ray crystallography
P01647	P07256	1ezv	MI:0114	x-ray crystallography
P01647	P07257	1ezv	MI:0114	x-ray crystallography
P01647	P08525	1ezv	MI:0114	x-ray crystallography
P01647	P22289	1ezv	MI:0114	x-ray crystallography
P01675	P04004	3bt2	MI:0114	x-ray crystallography
P01730	P01857	2qad	MI:0114	x-ray crystallography
P01730	P01857	2QAD	MI:0114	x-ray crystallography
P01730	P02789	1jl4	MI:0114	x-ray crystallography
P01730	P03437	3s4s,3s5l	MI:0114	x-ray crystallography

FIGURE 3.1.2: Samples from Negatome v 2.0 dataset. Each row indicates there is no interaction between specified pair of proteins.

Since we wanted to deal with balanced datasets, we selected the same number of protein pairs from each datasets (PrePPI and Negatome). Ideally, we should have selected as much pairs as existing in the smaller dataset which is 4397 (from Negatome). However, in order to make our datasets more manageable we reduce the size of each dataset to 3500. Using Java Random class, we randomly selected 3500 positive protein pairs from PrePPI dataset as well as 3500 negative protein pairs from Negatome dataset.

3.2 Obtaining protein sequences

PrePPI and Negatome only list proteins names while for predicting PPIs we need to process protein sequences. Therefore, after selecting the samples from original databases we used Python to extract sequences for all the existing proteins in our dataset. For this matter, using Python codes first we downloaded a XML (in *ProteinName*.xml format) file from Uniprot (www.Uniprot.org) for each protein in our dataset, then from those xml files we extracted proteins sequences information from *Sequnce* tag.

FIGURE 3.2.1: *Sequence* tag in P22619.xml file (downloaded from Uniprot.org) contains the sequence information for P22619 protein.

After obtaining protein sequences we changed our datasets format from the original one (Figure 3.1.1, and 3.1.2) to FASTA format (Figure 3.2.2). In FASTA format first line of each entry, which is indicated by a $>$ symbol, is a description line for that entry, and is followed by sequence information of corresponding protein in next line(s). The reason we changed our datasets to FASTA format is that FASTA is one of the layouts that is acceptable by most of the motif discovery tools as well as blast software.

```
>A0JLT2
MENFTALFGAQADPPPPPTALGFGPGKPPPPPPPPAGGGPGTAPPPTAATAPPGADKSGAGCGPFYLMRELPGSTELTGSTNLITHYNLEQAYNKFCGKKV
KEKLSNFLPDLPGMIDLPGSHDNSSLRSLIEKPPILSSSFNPITGTMLAGFRLHTGPLPEQCRLMHIQPPKKKNKHKHKQSRTQDPVPPETPSDSDHKKKK
KKKEEDPDRKRKKKEKKKKKNRHSPDHPGMGSSQASSSSSLR

>Q13838
MAENDVDNELLDYEDDEVETAAGGDGAEAPAKKDVKGSYVSIHSSGFRDFLLKPELLRAIVDCGFEHPSEVQHECIPQAILGMDVLCQAKSGMGKTAVFVL
ATLQQLEPVTGQVSVLVMCHTRELAFQISKEYERFSKYMPNVKVAVFFGGLSIKKDEEVLKKNCPHIVVGTPGRILALARNKSLNLKHIKHFILDECDKML
EQLDMRRDVQEIFRMTPHEKQVMMFSATLSKEIRPVCRKFMQDPMEIFVDDETKLTLHGLQQYYVKLKDNEKNRKLFDLLDVLEFNQVVIFVKSVQRCIAL
AQLLVEQNFPAIAIHRGMPQEERLSRYQQFKDFQRRILVATNLFGRGMDIERVNIAFNYDMPEDSDTYLHRVARAGRFGTKGLAITFVSDENDAKILNDVQ
DRFEVNISELPDEIDISSYIEQTR


>A0JLT2
MENFTALFGAQADPPPPPTALGFGPGKPPPPPPPPAGGGPGTAPPPTAATAPPGADKSGAGCGPFYLMRELPGSTELTGSTNLITHYNLEQAYNKFCGKKV
KEKLSNFLPDLPGMIDLPGSHDNSSLRSLIEKPPILSSSFNPITGTMLAGFRLHTGPLPEQCRLMHIQPPKKKNKHKHKQSRTQDPVPPETPSDSDHKKKK
KKKEEDPDRKRKKKEKKKKKNRHSPDHPGMGSSQASSSSSLR

>Q14978
MADAGIRRVVPSDLYPLVLGFLRDNQLSEVANKFAKATGATQQDANASSLLDIYSFWLKSAKVPERKLQANGPVAKKAKKKASSSDSEDSSEEEEEVQGPP
AKKAAVPAKRVGLPPGKAAAKASESSSSEESSDDDDEEDQKKQPVQKGVKPQAKAAKAPPKKAKSSDSDSDSSSEDEPPKNQKPKITPVTVKAQTKAPPKP
ARAAPKIANGKAASSSSSSSSSSSSDDSEEEKAAATPKKTVPKKQVVAKAPVKAATTPTRKSSSSEDSSSDEEEEQKKPMKNKPGPYSSVPPPSAPPPKKS
LGTQPPKKAVEKQQPVESSEDSSDESDSSSEEEKKPPTKAVVSKATTKPPPAKKAAESSSDSSDSDSSEDDEAPSKPAGTTKNSSNKPAVTTKSPAVKPAA
APKQPVGGGQKLLTRKADSSSSEEESSSSEEEKTKKMVATTKPKATAKAALSLPAKQAPQGSRDSSSDSDSSSSEEEEEKTSKSAVKKKPQKVAGGAAPSK
PASAKKGKAESSNSSSSDDSSEEEEEKLKGKGSPRPQAPKANGTSALTAQNGKAAKNSEEEEEEKKKAAVVVSKSGSLKKRKQNEAAKEAETPQAKKIKLQ
TPNTFPKRKKGEKRASSPFRRVREEEIEVDSRVADNSFDAKRGAAGDWGERANQVLKFTKGKSFRHEKTKKKRGSYRGGSISVQVNSIKFDSE
```

FIGURE 3.2.2: Our datasets view after finding the sequences information and changing their formats to FASTA.

3.3 Refining the datasets

In this part we curated our datasets by finding repeated and duplicate pairs in our datasets in order to have clean datasets with unique instances.

1. Duplicate protein pairs

   As we noticed both PrePPI and Negatome databases happens to have duplicate protein pairs such that two or more different rows indicate interact or non-interact interaction between same pair of proteins. In order to refine the datasets, using Java codes (Figure 3.3.2) we found these repeated samples, kept one for each pair and removed duplicates from our datasets. In this code, we compared each pair of proteins with rest of the pairs to see if we could find cases such that proteins in a pair are repeated with any order in another pair.

1	A0N6Y3	P00127	2ibz	MI:0114	x-ray crystallography
2	P00127	A0N6Y3	2IBZ	MI:0114	x-ray crystallography
3	A0N6Y3	P00128	2ibz	MI:0114	x-ray crystallography
4	P00128	A0N6Y3	2IBZ	MI:0114	x-ray crystallography
5	A5GZW8	Q0QF01	1zoy	MI:0114	x-ray crystallography
6	Q0QF01	A5GZW8	1ZP0	MI:0114	x-ray crystallography
7	O14746	P61769	2bck	MI:0114	x-ray crystallography
8	P61769	O14746	2BCK	MI:0114	x-ray crystallography
9	A2NTU7	P01887	2ckb	MI:0114	x-ray crystallography
10	A2NTU7	P01887	2CKB	MI:0114	x-ray crystallography

FIGURE 3.3.1: Samples of duplicate protein pairs in Negatome dataset.

```
...
for (int i =1 ; i<5001 ; i++){
  Protein1 = FirstProtein[i];
  Protein2 = SecondProtein[i];
  counter = 0;
  for (int j =1 ; j<5001 ; j++){
    Protein3 = FirstProtein[j];
    Protein4 = SecondProtein[j];
    if ((Protein1==Protein3 && Protein2==Protein4)||
       (Protein1==Protein4 && Protein2==Protein3)){
            counter = counter + 1;
    }
  }
}
...
```

FIGURE 3.3.2: Algorithm used for finding duplicate protein pairs in our datasets.

2. Different proteins with identical sequences

   There are some proteins in both PrePPI and Negatome database such that they have different names, they have identical sequences though. Since our method depends on protein sequences, having such instances can affect the SLiMs discovery part and eventually our classification results. In order to avoid these proteins, we applied a BLASTp query on our datasets, found such samples, saved one for each pair and eliminated remaining duplicates.

```
P1:D0VWR9
P2:F5H8P3
-----------------------------------
P3:D0VWR9
P4:P19054
-----------------------------------
Similarity between D0VWR9,D0VWR9 is:100%
Similarity between F5H8P3,P19054 is:100%
===================================
P1:A1KKF3
P2:A5U4D5
-----------------------------------
P3:A1KKF3
P4:A5WP83
-----------------------------------
Similarity between A1KKF3,A1KKF3 is:100%
Similarity between A5U4D5,A5WP83 is:100%
===================================
```

FIGURE 3.3.3: Examples of identical protein pairs found using BLASTp results.

```
>D0VWR9
MEVNQLGFIATALFVLVPSVFLIILYVQTESQQKSS

>F5H8P3
KLPEAYAIFDPLVDVLPVIPVLFFALAFVVQAAVGFR
----------------------------------------
>D0VWR9
MEVNQLGFIATALFVLVPSVFLIILYVQTESQQKSS

>P19054
KLPEAYAIFDPLVDVLPVIPVLFFALAFVVQAAVGFR
----------------------------------------
```

```
>A1KKF3
MTWPLPDRLSINSLSGTPAVDLSSFTDFLRRQAPELLPASISGGAPLAGGDAQLPHGTTIVALKYPGGVVMAGDRRSTQGNMISG
RDVRKVYITDDYTATGIAGTAAVAVEFARLYAVELEHYEKLEGVPLTFAGKINRLAIMVRGNLAAAMQGLLALPLLAGYDIHASD
PQSAGRIVSFDAAGGWNIEEEGYQAVGSGSLFAKSSMKKLYSQVTDGDSGLRVAVEALYDAADDDSATGGPDLVRGIFPTAVIID
ADGAVDVPESRIAELARAIIESRSGADTFGSDGGEK

>A5U4D5
MSFPYFISPEQAMRERSELARKGIARAKSVVALAYAGGVLFVAENPSRSLQKISELYDRVGFAAAGKFNEFDNLRRGGIQFADTR
GYAYDRRDVTGRQLANVYAQTLGTIFTEQAKPYEVELCVAEVAHYGETKRPELYRITYDGSIADEPHFVVMGGTTEPIANALKES
YAENASLTDALRIAVAALRAGSADTSGGDQPTLGVASLEVAVLDANRPRRAFRRITGSALQALLVDQESPQSDGESSG
----------------------------------------------------------------------------------------
>A1KKF3
MTWPLPDRLSINSLSGTPAVDLSSFTDFLRRQAPELLPASISGGAPLAGGDAQLPHGTTIVALKYPGGVVMAGDRRSTQGNMISG
RDVRKVYITDDYTATGIAGTAAVAVEFARLYAVELEHYEKLEGVPLTFAGKINRLAIMVRGNLAAAMQGLLALPLLAGYDIHASD
PQSAGRIVSFDAAGGWNIEEEGYQAVGSGSLFAKSSMKKLYSQVTDGDSGLRVAVEALYDAADDDSATGGPDLVRGIFPTAVIID
ADGAVDVPESRIAELARAIIESRSGADTFGSDGGEK

>A5WP83
MSFPYFISPEQAMRERSELARKGIARAKSVVALAYAGGVLFVAENPSRSLQKISELYDRVGFAAAGKFNEFDNLRRGGIQFADTR
GYAYDRRDVTGRQLANVYAQTLGTIFTEQAKPYEVELCVAEVAHYGETKRPELYRITYDGSIADEPHFVVMGGTTEPIANALKES
YAENASLTDALRIAVAALRAGSADTSGGDQPTLGVASLEVAVLDANRPRRAFRRITGSALQALLVDQESPQSDGESSG
```

FIGURE 3.3.4: Corresponding sequences of discovered identical pairs.

After deleting all duplicate samples from both datasets as mentioned, we obtained around 2580 protein pairs in our negative (non-interacting) dataset and 2690 protein pairs in our positive (interacting) dataset. In order to balance the dataset, using Java codes

we randomly chose 2500 protein pairs from remaining pairs in each class (Figure 3.3.5). RAND function in Java produces a random number between 0 and the number provided for rand.nextInt() - 1.

```
...
for (int i=1 ; i<2501 ; i++){
    int Random = rand.nextInt(2690)+1; % for Positive
    writer.println(Pair(Random));
}
for (int j=1 ; j<2501 ; j++){
    int Random = rand.nextInt(2580)+1; % for Negative
    writer.println(Pair(Random));
}
...
```

FIGURE 3.3.5: Algorithm used for randomly selecting the positive and negative samples.

3.4 Obtaining the SLiMs

As mentioned earlier, we used MEME for motif discovery part. However, regardless of all the benefits that MEME has such as having standalone version which removes limitations on input dataset size and also capability of discovering novel motifs, finding even hundreds of motifs would be infeasible due to the long time that it takes. Considering for our case we have to look for motifs of length 3 to 10, the time needed to obtain motifs gets even longer than usual since all the process has to be repeated over and over again for each value of the length of the motif.

For solving this problem we proposed a method of dividing the whole dataset into smaller and manageable sub-datasets of equal size, and discovering motifs of length 3 to 10 for sub-datasets (Figure 3.4.2). Although SLiMs discovered by this method may be different from ones obtained from the whole dataset, time complexity will be significantly reduced such that it makes the case to be feasible.

Undoubtedly, the obtained motifs from a set of proteins depends on existing proteins in that set. In order to check how different grouping (making subsets randomly) may change the motifs and eventually the classification results we decided to obtain four different series of 100 subsets with completely unlike arrangement. Thus, after dividing the dataset into 100 subsets we shuffled the whole dataset and repeated grouping part to create second, third, and fourth random series of 100 subsets (Figure 3.4.2). While finding 5000 motifs from a dataset of size 5000 (protein pairs) may take several months or years, we obtained 20,000 motifs (5000 motifs from each series) only in 40 days.

For each series we passed each subset to MEME separately to obtain 50 motifs of length 3 to 10 using the following command:

meme Dataset.txt -o SubsetName -mod anr -nmotifs 50 -minw 3 -maxw 10

where Dataset.txt is the dataset created for each subset (25 positive and 25 negative pairs), -o indicates the name of the output folder, -mod signifies the mode for obtaining the motifs, -nmotifs specifies the number of motifs that need to be discovered, and -minw and -maxw are the minimum and maximum length of motifs, respectively.

MEME has three different modes for discovering motifs:

1. Zero or one occurrence per sequence (ZOOPS)
2. One occurrence per sequence (OOPS)
3. Any number of repetitions (ANR)

Using the first mode, MEME will find zero or one site per sequence in the dataset, while the second mode does not allow sequences to have less or more than one occurrences. On the other hand, the third mode does not put any limitation on the number of sites found in each sequence. For this reason we used the third mode for discovering motifs.

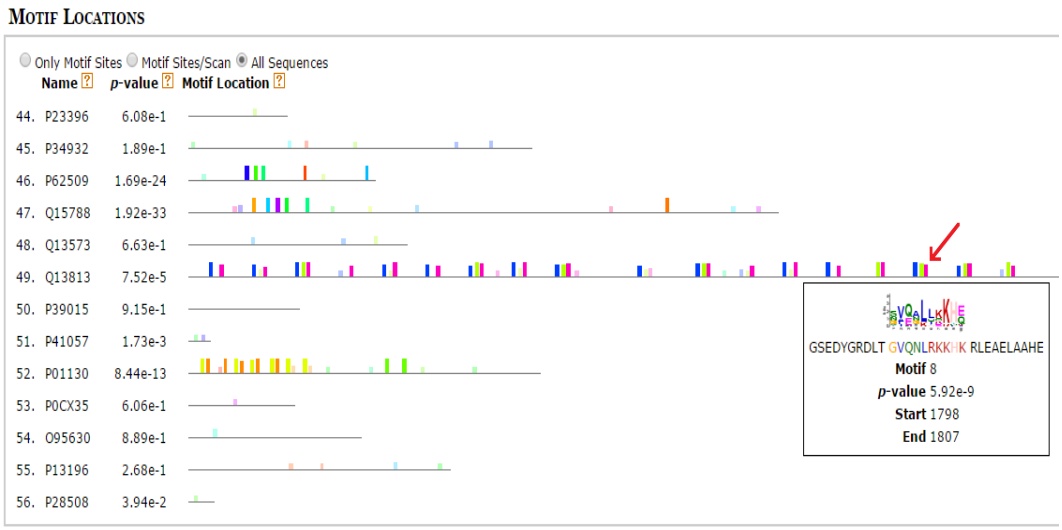

FIGURE 3.4.1: Samples of discovered motif locations after using ANR mode in MEME.

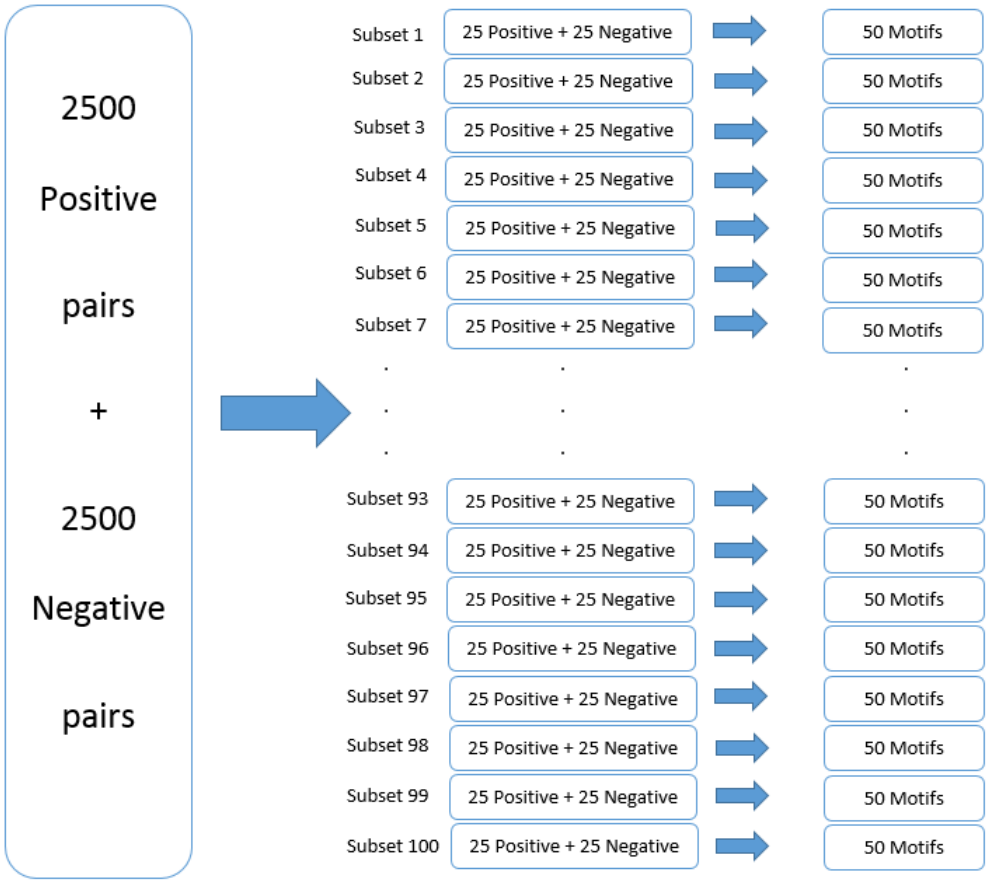

FIGURE 3.4.2: Dividing the whole dataset into 100 subsets of size 50 pairs of protein (25 positive and 25 negative) in order to pass each subset to MEME separately and obtain 50 motifs of length 3 to 10.

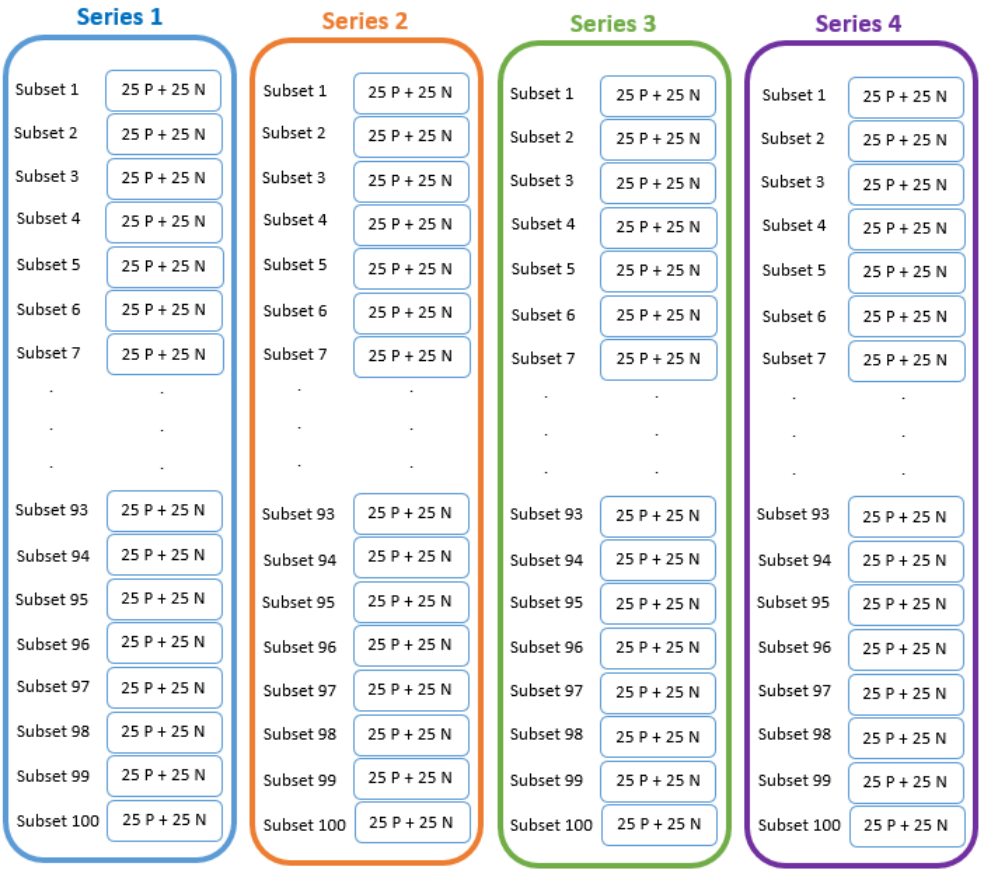

FIGURE 3.4.3: Creating four different series of 100 subsets by shuffling the dataset, randomly selecting protein pairs, and putting them into subsets for each series.

MEME will create a logo for each one of the discovered motifs, as well as one output text file for each executed and terminated command (in our case for each subset). This text file contains information about sequences existing in the input dataset, letter frequency in the dataset, as well as Position Specific Probability Matrix (PSPM) for each discovered motif. For all discovered motifs we used PSPM to uncover their regular expression.

Regular expressions can be easily obtained from PSPMs. In PSPM, each row corresponds to one position of the pattern. Thus, a PSPM with ten rows indicates that its pattern has ten positions. Since we have twenty different residues (ACDEFGHIKLMNPQRSTVWY) each column in PSPM also corresponds to one of them. By finding all non-zero elements of each row, we can discover corresponding residues for that position, such that if the probability is 1, that position in the pattern has a fixed card of the corre-

sponding residue, otherwise the probability is distributed between more than one residue, which means that position in the pattern has a wild-card of all corresponding residues. Finally, the regular expression is obtained by attaching all fixed and wild cards from the first position to the last one.

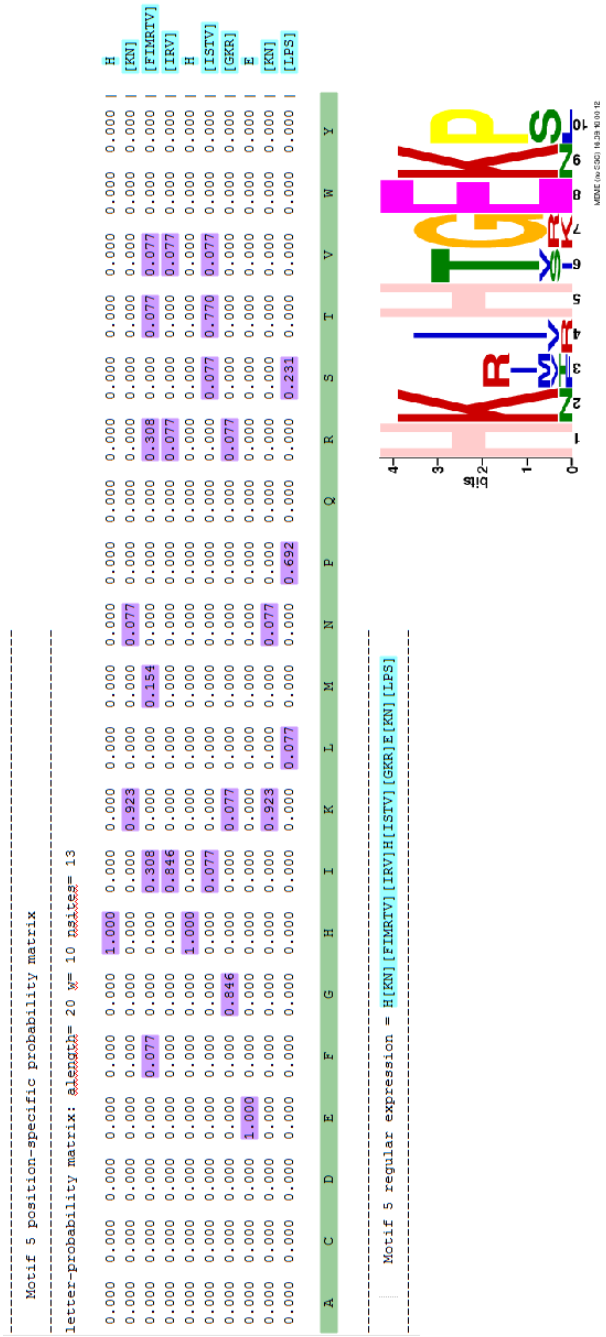

FIGURE 3.4.4: Position Specific Probability Matrix (PSPM) for one discovered motif, and the regular expression obtained by that.

3.5 Finding and scoring the sites

After obtaining regular expression of all discovered motifs, we used Java regular expression (java.util.regex package) to find and score the sites. Java Regex is a package that helps to find a substring in another string. All it needs is a pattern written in regular expression form, and a string that is going to be searched. The former is called *Pattern*, and the latter is named *Matcher*. For example, running following pattern and matcher in java will give us the number of matches found (here is 2) as well as starting position of the matches:

Pattern r = Pattern.compile("M[LRC]V");
Matcher m = r.matcher("PDTMLVCSVLVLLLRRNMRVNGDS");
While (m.find( ))

Thus, in order to build our final dataset based on the discovered motifs, for each pair of protein we passed all 5000 discovered motifs to the pattern separately to count all matches in both protein sequences.

```
--------------------------------
Motif 1 = C[IQTVY]C[FILTVY]G[AEFGNQR][GQS][IKRSV][GS][CEFHRW]
--------------------------------
Motif 2 = [DGNPR]L[EKLMQRST][PT][AGT][CLRSTV][DEKQ]Y[IKNSTV][ILTVY]
--------------------------------
Motif 3 = [GRV][KMNQSTV][FGHLMNSTV][EY][AHKLQRSTV][IV][GN][DEMQV][EKPQSTY][WY]
--------------------------------
Motif 4 = [AGHNQS][AIV][ENQ][AGNTV]L[ILQR][DEKRT][KR][FHQ][ADEKQ]
--------------------------------
.
.
.
--------------------------------
Motif 4997 = [LY][ELRST]C[AFHNR][EFPQRST][CDEHIRS][KMNR][DGHNQ][CH][ITV]
--------------------------------
Motif 4998 = P[EK][IKT]V[KTY]W[DM]R[DR]M
--------------------------------
Motif 4999 = TP[KQ]IQVYSRH
--------------------------------
Motif 5000 = IC[DY][IM][AL][IN]NW
--------------------------------
```

FIGURE 3.5.1: Samples of listed discovered motifs from series 1.

For example for A0JLT2-P52292 pair of protein, first we pass the first motif's regular expression to the "Pattern", then we pass A0JLT2's sequence to the "Matcher" and try to find and score the sites. Before proceeding to the next motif, we also pass P52292's

sequence to the ”Matcher” and try to do the same. Eventually, we do the same for all 5000 motifs (Figure 3.5.2).

```
...
for (int i =1 ; i<5001 ; i++){
      score=0;
      Pattern r = Pattern.compile(Motif[i]);
      Matcher m = r.matcher(A0JLT2-Sequence);
      while (m.find( ))
      {
      score=score+1;
      }
      m = r.matcher(P52292-Sequence);
      while (m.find( ))
      {
      score=score+1;
      }
      ...
}
```

FIGURE 3.5.2: Algorithm used for scoring the sites.

Using regular expression pattern and matcher, this code will find all the sites existing in both proteins sequences and count number of matches for each motif (Figure 3.5.3).

```
>A0JLT2
         Motif 16                                        Motif 250
MENFTALFGAQADPPPPPTALGFGPGKPPPPPPPPAGGGPGTAPPPTAATAPPGADKSGAGCGPFYLMRELPGSTELTGSTNLITHYNLEQAYNKFCGKK
                          Motif 3018                                                  Motif 1298
VKEKLSNFLPDLPGMIDLPGSHDNSSLRSLIEKPPILSSSFNPITGTMLAGFRLHTGPLPEQCRLMHIQPPKKKNKHKHKQSRTQDPVPPETPSDSDHKK

KKKKKEEDPDRKRKKKEKKKKNRHSPDHPGMGSSQASSSSSLR
--------------------------------------------------------------------------------------------------------

>P52292
                                                                  Motif 250
MSTNENANTPAARLHRFKNKGKDSTEMRRRRIEVNVELRKAKKDDQMLKRRNVSSFPDDATSPLQENRNNQGTVNWSVDDIVKGINSSNVENQLQATQAA

RKLLSREKQPPIDNIIRAGLIPKFVSFLGRTDCSPIQFESAWALTNIASGTSEQTKAVVDGGAIPAFISLLASPHAHISEQAVWALGNIAGDGSVFRDLV
                                  Motif 16
IKYGAVDPLLALLAVPDMSSLACGYLRNLTWTLSNLCRNKNPAPPIDAVEQILPTLVRLLHHDDPEVLADTCWAISYLTDGPNERIGMVVKTGVVPQLVK
Motif 4893
LLGASELPIVTPALRAIGNIVTGTDEQTQVVIDAGALAVFPSLLTNPKTNIQKEATWTMSNITAGRQDQIQQVVNHGLVPFLVSVLSKADFKTQKEAVWA
                          Motif 4893
VTNYTSGGTVEQIVYLVHCGIIEPLMNLLTAKDTKIILVILDAISNIFQAAEKLGETEKLSIMIEECGGLDKIEALQNHENESVYKASLSLIEKYFSVEE

EEDQNVVPETTSEGYTFQVQDGAPGTFNF
--------------------------------------------------------------------------------------------------------
```

FIGURE 3.5.3: Using regular expression we found sites of all the motifs in each protein pair.

3.6 Building final datasets

Using mentioned method (Figure 3.5.2), we built our final dataset such that for each pair of proteins in our dataset we assigned score of each motif in its corresponding position.

	Motif 1	Motif 2	...	Motif 16	...	Motif 250	...	Motif 1298	...	Motif 3018	...	Motif 4893	...	Motif 4999	Motif 5000
A0JLT2-P52292 :	0	0		2		2		1		1		2		0	0
A0JLT2-P31689 :	0	4		0		1		0		0		0		1	0
P61457-P35680 :	1	0		0		0		0		0		0		0	0
.															
.															
.															
P01887-P38158 :	0	0		0		0		0		0		6		0	1
Q6R327-Q16531 :	11	0		2		0		3		0		0		0	0
Q5RIG3-Q61026 :	3	0		8		0		1		0		0		12	2

FIGURE 3.6.1: Building the final dataset based on the scores provided by the regular expression.

Since we had four different series of 5000 motifs, we created one dataset, as explained, for each series to see how different grouping changes the classification results. Furthermore, at this point we decided to score all 20,000 motifs and created two more datasets based on the flexibility of motifs patterns.

It has been proved that motifs can be scored using information theory [14]. Thus we use following formula to score each motif (all logarithms are base 2):

$$I(P) = \sum_{i} H(M) - H(K_i) \tag{3.6.1}$$

$$H(P) = -\sum_{a \in C} P_a log P_a \tag{3.6.2}$$

where C is a set of symbols $\{a\}$, which each of them has a background probability $\{Pa\}$. i also runs over all the positions in the pattern K, and M is a set of all amino acids existing in the patterns (in our case M = {A,C,D,E,F,G,H,I,K,L,M,N,P,Q,R,S,T,V,W,Y}).

Since in our experiment we have motifs of different lengths (mostly 7 to 10), we divided

I(P) by the length of the motifs to normalize the scores. Therefore, we have:

$$I(P) = \frac{\sum_i H(M) - H(K_i)}{l} \tag{3.6.3}$$

where *l* is the length of each motif.

For example, for pattern K = C[DG][AHQ]D, while *M* = {A,C,D,P,G,H,Q}, and their background probabilities are $P_A = P_C = P_D = P_G = P_H = P_Q = \frac{1}{6}$, and considering that probabilities of letters in a wild-card are equal, scoring would be as follows:

$$I(P) = \frac{\sum_i H(M) - H(K_i)}{l} = \frac{H(M) - H(C) + H(M) - H([DG]) + H(M) - H([AHQ]) + H(M) - H(D)}{l} = \frac{-6(\frac{1}{6}log\frac{1}{6}) + 1log1 - 6(\frac{1}{6}log\frac{1}{6}) + 2(\frac{1}{2}log\frac{1}{2}) - 6(\frac{1}{6}log\frac{1}{6}) + 3(\frac{1}{3}log\frac{1}{3}) - 6(\frac{1}{6}log\frac{1}{6}) + 1log1}{4} = \frac{7.7548}{4} = 1.9387$$

As mentioned earlier, using (3.6.3) we scored all 20,000 motifs obtained from the four series. Then, we used the 5000 top scored ones to create a new dataset named (Stiff-Motifs), and also used 5000 low scored ones to create another dataset named (Flexible-Motifs). The reason for choosing these names is that the motifs with higher scores have more fixed-card and do not have flexible positions (Figure 3.6.2), while motifs with lower scores have more wild-cards and as a result they are more flexible (Figure 3.6.3). It means the lower score a motif has, there is more chance to find sites using that motif, because it has more wild-cards.

Finally, to be able to compare the quality of the motifs discovered by MEME and Nomad, we used Nomad to discover same number of motifs (5000) from our refined dataset, and created our last dataset (named "Nomad-Motifs") using regular expression.

```
Motif 17353:          QNANPDCKTI
I(P) = 4.1774907
--------------------------------
Motif 12768:          FIDNIFRFTQ
I(P) = 4.1774907
--------------------------------
.
.
.
--------------------------------
Motif 7720:           PQEDMCQTFG
I(P) = 4.176589
--------------------------------
Motif 267:            FN[QR]AFGFM
I(P) = 4.051271
--------------------------------
.
.
.
--------------------------------
Motif 13271:          FT[FR][KW][EQ]WFD
I(P) = 3.8012712
--------------------------------
Motif 15:             [DKR][TWY]CA[TV][CT][HNQ][DNQ]
I(P) = 3.2234514
--------------------------------
.
.
.
```

FIGURE 3.6.2: Selecting the first 5000 motifs with the most fixed-cards to build dataset entitled ”Stiff-Motifs”.

```
Motif 1601:
C[ADEFGHIKLNPQSTV][AEFGHIKLNPQSTV]C[ACDEFGHIKLMPQRSTY][DEFGHIKLMNPQRST][ADEFHKLMNPRTVY][ACEFHIKLMQSTVW][ACDEGHIKLNQRSTVWY][ADEGHIKLNPQRST]
I(P) = 1.2945489
--------------------------------
Motif 7568:
[ACDFGILMPQRTVY][FGHKLRVWY][ADEFGHIKLMNQRSTVY][CF][ADEGKNPQRST][ACEFGHIKLNQSVY][CS][ADEFGHKLMNQRSW][ACDHIKLNPQRSTY][EFHKLNPRSTVY]
I(P) = 1.4294226
--------------------------------
  .  .  .
--------------------------------
Motif 19451:
[FHILMTVWY][ACDEFHIKLNPQRSTV][CH][ADEGHKNPQSTV][AEFGHIKLMNQRSVY][CE][ADEFGHKLNQRTVW][ADEFGHKMNQRSVWY][AEFGHKLMNPQRSTVY][ADFGIKLMRSTV]
I(P) = 1.4955812
--------------------------------
Motif 1603:
[ACFLNQRTVWY][ACEHKMPQRSTVY]C[ACDEFGIKLNPQRSTV][AEFGHIKLMNPQSTVY]C[DEGKMNQRS][ADEGIKLMNRSTVWY][ACEFGHKLMNQRSVY][AFGIKLNTWY]
I(P) = 1.5694661
--------------------------------
  .  .  .
--------------------------------
Motif 15:
[DEKP][EGQ][ADEGIMPSTV][ILNTV][FIKMNV][EGKNT][PST][CW][CFL][CEGW]
I(P) = 2.9851136
--------------------------------
Motif 14871:
[AIKV][ALPTV]E[EK][EKRV][AIKPV][AHLPRS][AELPTV][AIKLPV][AEFIKPRV]
I(P) = 3.00958
--------------------------------
  .  .  .
```

FIGURE 3.6.3: Selecting the first 5000 motifs with the most wild-cards to build dataset entitled ”Flexible-Motifs”

3.7 Classification

For the machine learning phase, we used different classifiers such as Naive Bayes, K-nearest neighbour (KNN) with K values from 1 to 70 ($\sqrt{5000}$), Random Forest, SVM-Polynomial with default parameters, and SVM-RBF with different combinations of c (cost) and g (gamma) values, for all the six datasets created with motifs obtained from the MEME, and dataset created with motifs obtained from NOMAD. 10-fold cross validation is also used for training and testing all classifiers. For each classifier we obtained the confusion matrix, TP Rate, FN Rate, TN Rate, FP Rate, Accuracy, Precision, Recall, and MCC value. We also used mRmR feature selection to see how the selected features affect accuracy of mentioned classifiers on our experiment. The results of all mentioned methods are displayed in next chapter.

CHAPTER 4

Results

For classification purposes, we used different classifiers such as Naive Bayes, K-nearest neighbour (KNN), Random Forest, and SVM with two different kernels, Polynomial and Radial Basis Function (RBF) on all of our datasets. As mentioned earlier, we had six datasets created by the SLiMs obtained using MEME(four datasets from four series of subsets and two datasets using scoring function, named Stiff-Motifs and Flexible-Motifs), and one dataset which was built using SLiMs discovered by Nomad.

After obtaining the results, we also applied Feature Selection on all datasets with the aim of removing possible noise and obtaining better results. For mRmR we selected "*WrapperSubsetEval*" as "*Attribute evaluator*" and "*RerankingSearch*" as its search method. Furthermore, we chose Random Forest as wrapper's classifier, and *Accuracy* for its evaluation measure. Moreover, we used mRmR as our ranking method.

We used the features selected by mRmR to filter our datasets, and applied all mentioned classifiers once again on filtered datasets to be able to compare both methods. The results of classifying all datasets (original and filtered) using the mentioned classifiers, as well as comparison between the two methods are listed and discussed in this chapter.

4.1 Classification results on the original datasets

4.1.1 NaiveBayes

As can be seen in Table 4.1.1, NaiveBayes classified Series 1, Series 2, Series 3, Series 4, Flexible-Motifs, and Nomad-Motifs datasets with accuracy values between 71% to 74%. The best result is achieved from classifying Flexible-Motifs with 73.52% accuracy and

0.528 MCC. Besides, NaiveBayes could not classify Stiff-Motifs with better than 61.14% accuracy.

Naive Bayes	Confusion Matrix		Accuracy(%)	Precision(%)	Recall(%)	MCC
Series 1	1208	1292	72.46	79.30	72.50	0.513
	85	2415				
Series 2	1277	1223	73.38	79.20	73.40	0.522
	108	2392				
Series 3	1248	1252	73.28	79.80	73.30	0.527
	84	2416				
Series 4	1176	1324	71.60	78.50	71.60	0.496
	96	2404				
Stiff-Motifs	2478	22	61.14	76.30	61.10	0.343
	1921	579				
Flexible-Motifs	1271	1229	73.52	79.60	73.50	0.528
	95	2405				
Nomad-Motifs	2496	4	71.88	81.80	71.90	0.528
	1402	1098				

TABLE 4.1.1: Results of running Naive Bayes classifier on series 1 to 4, Stiff-Motifs, Flexible-Motifs, and Nomad-Motifs datasets. (Best, second best, and third best).

4.1.2 KNN

Table 4.1.2 reveals the results of applying KNN (k=1 to k=70) classifier on our datasets. Best value of k is indicated for each dataset. As shown in the table, while accuracy of KNN for all Series 1, Series 2, Series 3, Series 4, and Flexible-Motifs datasets is around 90%, the best result is acquired from Series 2 dataset with 90.96% accuracy and 0.821 MCC. However, accuracy gained for Stiff-Motifs and Nomad-Motifs datasets are only 72.24% and 78.98%, respectively.

KNN	Confusion Matrix		Accuracy(%)	Precision(%)	Recall(%)	MCC
Series 1 (k=2)	2226	274	90.52	90.60	90.50	0.811
	200	2300				
Series 2 (k=3)	2201	299	90.96	91.10	91.00	0.821
	153	2347				
Series 3 (k=1)	2178	322	90.78	91.00	90.80	0.818
	139	2361				
Series 4 (k=2)	2242	258	90.62	90.60	90.60	0.813
	211	2289				
Stiff-Motifs (k=1)	2464	36	72.24	80.80	72.20	0.523
	1352	1148				
Flexible-Motifs (k=2)	2210	290	90.68	90.80	90.70	0.814
	176	2324				
Nomad-Motifs (k=1)	1517	983	78.98	83.50	79.00	0.623
	68	2432				

TABLE 4.1.2: Results of running KNN classifier (k=1 to k=70) on series 1 to 4, Stiff-Motifs, Flexible-Motifs, and Nomad-Motifs datasets. (Best, second best, and third best).

4.1.3 Random Forest

As shown in Table 4.1.3, applying Random Forest classifiers on 5 datasets Series 1, Series 2, Series 3, Series 4, and Flexible-Motifs gave us almost the same accuracy of 92%. While the best result is 92.36% accuracy and 0.847 MCC for Series 3, Random Forest could not classify Flexible-Motifs and Nomad-Motifs better than 72.52% and 79.30% accuracy, respectively.

4.1.4 SVM

For SVM classifier we used two different kernels, Polynomial and Radial Basis Function (RBF). For Polynomial kernel we used the default settings, however for SVM-RBF in order to find the best combination of cost and gamma we did a grid search. First we fixed the cost (c) to 10 and changed the value of gamma (g) from beginning to the end of set {0.01, 0.1, 1, 10, 100, 1000, 5000, 10000, 20000, 100000}. Then we did the same thing with setting gamma to 0.01 and changing the cost in this order {1, 10, 100, 1000, 10000, 100000, 1000000}. As can be seen from Table 4.1.4, performance of polynomial kernel for almost all the datasets is weak with accuracy around 52%. However, SVM-RBF has much better

Random Forest	Confusion Matrix		Accuracy(%)	Precision(%)	Recall(%)	MCC
Series 1	2292	208	92.04	92.00	92.00	0.841
	190	2310				
Series 2	2298	202	92.08	92.10	92.10	0.842
	194	2306				
Series 3	2317	183	92.36	92.40	92.40	0.847
	199	2301				
Series 4	2282	218	91.50	91.50	91.50	0.830
	207	2293				
Stiff-Motifs	2462	38	72.52	80.80	72.50	0.527
	1336	1164				
Flexible-Motifs	2281	219	91.76	91.80	91.80	0.835
	193	2307				
Nomad-Motifs	1524	976	79.30	83.90	79.30	0.630
	59	2441				

TABLE 4.1.3: Results of running Random Forest classifier on series 1 to 4, Stiff-Motifs, Flexible-Motifs, and Nomad-Motifs datasets. (Best, second best, and third best).

performance.

As shown, while the best accuracy for each dataset has been obtained with different mixture of cost and gamma, the best result for four out of seven datasets (Series 1, Series 3, Series 4, and Flexible-Motifs) has been obtained setting cost to 10 and gamma to 0.01. Considering that accuracy for all these 4 datasets are the best 4 accuracies among all 7 datasets we can realize that c=10 and g=0.01 is the best combination between all the ones we tried.

The best accuracy, precision, recall, and MCC value for all the datasets after being classified by SVM-RBF has been demonstrated in Table 4.1.5. Clearly, accuracy of Series 1, Series 2, Series 3, and Series 4 datasets are all around 92% and the best result is gained from classifying Flexible-Motifs dataset using SVM-RBF (c=10 and g=0.01) with 93.70% accuracy. Stiff-Motifs and Nomad-Motifs datasets also got 72.92% and 81.0% accuracy.

SVM Grid-Search	Series 1	Series 2	Series 3	Series 4	Stiff-Motifs	Flexible-Motifs	Nomad-Motifs
SVM-Polynomial	51.00	52.40	51.30	51.10	50.10	56.30	50.02
SVM-RBF c=10, g=0.01	92.24	92.22	92.32	91.24	72.60	**93.70**	78.18
SVM-RBF c=10, g=0.1	91.60	91.76	92.06	90.68	72.62	87.60	74.32
SVM-RBF c=10, g=1	82.82	82.18	78.78	82.00	71.02	63.40	76.42
SVM-RBF c=10, g=10	71.96	70.74	66.58	68.98	69.90	52.80	81.00
SVM-RBF c=10, g=100	71.96	70.74	66.58	68.98	69.90	52.80	81.00
SVM-RBF c=10, g=1,000	71.96	70.74	66.58	68.98	69.90	52.80	81.00
SVM-RBF c=10, g=5,000	71.96	70.74	66.58	68.98	69.90	52.80	81.00
SVM-RBF c=10, g=10,000	71.96	70.74	66.58	68.98	69.90	52.80	81.00
SVM-RBF c=10, g=20,000	71.96	70.74	66.58	68.98	69.90	52.80	81.00
SVM-RBF c=10, g=100,000	71.96	70.74	66.58	68.98	69.90	52.80	81.00
SVM-RBF g=0.01 , c=1	89.96	89.14	89.72	88.18	68.20	92.40	72.98
SVM-RBF g=0.01 , c=10	92.24	92.22	92.32	91.24	72.60	93.70	78.18
SVM-RBF g=0.01 , c=100	91.80	92.64	92.26	91.12	72.92	93.40	79.38
SVM-RBF g=0.01 , c=1,000	91.48	90.10	91.74	90.36	72.86	92.80	79.20
SVM-RBF g=0.01 , c=10,000	90.52	90.10	90.32	88.46	72.90	92.50	79.20
SVM-RBF g=0.01 , c=100,000	89.40	89.46	89.94	87.84	72.90	92.50	79.20
SVM-RBF g=0.01 , c=1,000,000	89.02	88.78	89.94	87.94	72.90	92.50	79.20

TABLE 4.1.4: Grid-search on SVM reveals the best obtained results for each of seven datasets. Values are gained accuracy(%) after running SVM on datasets.

SVM-RBF	Confusion Matrix		Accuracy(%)	Precision(%)	Recall(%)	MCC
Series 1 (c=10, g=0.01)	2251	249	92.24	92.30	92.20	0.846
	139	2361				
Series 2 (c=100, g=0.01)	2276	224	92.64	92.70	92.60	0.853
	144	2356				
Series 3 (c=10, g=0.01)	2244	256	92.32	92.40	92.30	0.848
	128	2372				
Series 4 (c=10, g=0.01)	2209	291	91.24	91.40	91.20	0.826
	147	2353				
Stiff-Motifs (c=100, g=0.01)	2445	55	72.92	80.50	72.90	0.528
	1299	1201				
Flexible-Motifs (c=10, g=0.01)	**2360**	**140**	**93.70**	**93.70**	**93.70**	**0.874**
	175	**2325**				
Nomad-Motifs (c=10, g=10)	2074	426	81.00	81.00	81.00	0.620
	524	1976				

TABLE 4.1.5: Results of running SVM-RBF classifier on series 1 to 4, Stiff-Motifs, Flexible-Motifs, and Nomad-Motifs datasets. (Best, second best, and third best).

4.2 Classification results on datasets after feature selection

As we mentioned, we used mRmR feature selection method with setting Random Forest as its wrapper's classifier. The number of features that mRmR chose between all the features for each dataset (5000) is as follows:

Feature selection with mRmR	Number of selected features
Series 1	25
Series 2	24
Series 3	27
Series 4	28
Stiff-Motifs	2
Flexible-Motifs	20
Nomad-Motifs	1

TABLE 4.2.1: Number of features selected by mRmR for each dataset.

Using mRmR results, we filtered all 7 datasets such that we kept selected features, and removed the remaining ones. Then we classified filtered datasets with same classifiers to

see how feature selection affects the classification results.

4.2.1 Results of applying NaiveBayes on filtered datasets

As can be seen in Table 4.2.2, NaiveBayes classified F-Series 1, F-Series 2, F-Series 3, F-Series 4, and Flexible-Motifs datasets with accuracy between 69% to 74%. The best result is achieved from classifying F-Series 3 with 73.76% accuracy and 0.534 MCC. Besides, NaiveBayes could not classify F-Stiff-Motifs, and F-Nomad-Motifs datasets with better than 52.86%, and 51.98% accuracy.

Naive Bayes	Confusion Matrix		Accuracy(%)	Precision(%)	Recall(%)	MCC
F-Series 1	1174	1326	71.92	79.20	71.90	0.506
	78	2422				
F-Series 2	1275	1225	73.76	80.00	73.80	0.534
	87	2413				
F-Series 3	1177	1323	72.06	79.40	72.10	0.509
	74	2426				
F-Series 4	1171	1329	71.72	78.90	71.70	0.501
	85	2415				
F-Stiff-Motifs	2498	2	52.86	75.10	52.90	0.169
	2355	145				
F-Flexible-Motifs	1043	1457	69.26	77.60	69.30	0.462
	80	2420				
F-Nomad-Motifs	2498	2	51.98	74.50	52.00	0.139
	2399	101				

TABLE 4.2.2: Results of running Naive Bayes classifier on series 1 to 4, Stiff-Motifs, Flexible-Motifs, and Nomad-Motifs datasets after getting filtered using feature selection results. (Best, second best, and third best).

4.2.2 Results of applying KNN on filtered datasets

Table 4.2.3 reveals the results of applying KNN (k=1 to k=70) classifier on our filtered datasets. Best value of k is indicated for each dataset. As shown in the table, while accuracy of KNN for all F-Series 1, F-Series 2, F-Series 3, F-Series 4, and F-Flexible-Motifs datasets is around 85%, the best result is acquired from F-Series 3 dataset with 86.54% accuracy and 0.733 MCC. However, accuracy gained for Stiff-Motifs and Nomad-Motifs datasets are as low as 54.08% and 51.98%, respectively.

KNN (k=1 to k=70)	Confusion Matrix		Accuracy(%)	Precision(%)	Recall(%)	MCC
F-Series 1 (k=2)	2041	459	85.96	86.20	86.00	0.722
	243	2257				
F-Series 2 (k=3)	2004	496	84.70	85.00	84.70	0.697
	269	2231				
F-Series 3 (k=1)	2067	433	86.54	86.80	86.50	0.733
	240	2260				
F-Series 4 (k=2)	2033	467	85.16	85.40	85.20	0.705
	275	2225				
F-Stiff-Motifs (k=1)	2497	3	54.08	75.40	54.10	0.203
	2293	207				
F-Flexible-Motifs (k=4)	2088	412	85.14	85.20	85.10	0.703
	331	2169				
F-Nomad-Motifs (k=1)	2498	2	51.98	74.50	52.00	0.139
	2399	101				

TABLE 4.2.3: Results of running KNN classifier on series 1 to 4, Stiff-Motifs, Flexible-Motifs, and Nomad-Motifs datasets after getting filtered using feature selection results. (Best, second best, and third best).

4.2.3 Results of applying Random Forest on filtered datasets

As shown in Table 4.2.4, applying Random Forest classifiers on 5 datasets F-Series 1, F-Series 2, F-Series 3, F-Series 4, and F-Flexible-Motifs gave us accuracy between 85% and 88%. While the best result is 88.06% accuracy and 0.763 MCC for F-Series 3, Random Forest could not classify Stiff-Motifs and Nomad-Motifs better than 54.08% and 51.98% accuracy, respectively.

4.2.4 Results of applying SVM on filtered datasets

As shown in Table 4.2.5, we did a grid search on filtered datasets with same values of cost and gamma. Obviously, the best result for F-Serties 1, F-Series 2, and F-Series 3 datasets has been obtained setting cost to 10 and gamma to 1, while for the rest of the datasets has been gained from setting cost to 10 and gamma to 0.1. Thus, based on the results, between all the values of cost and gamma that we tried, c=10 is the best value for cost, and g=0.1 and g=1 are two best values for gamma.

The best accuracy, precision, recall, and MCC value for all the datasets after being

Random Forest	Confusion Matrix		Accuracy(%)	Precision(%)	Recall(%)	MCC
F-Series 1	2033	467	86.00	86.30	86.00	0.723
	233	2267				
F-Series 2	2044	456	85.88	86.10	85.90	0.720
	250	2250				
F-Series 3	2128	372	88.06	88.20	88.10	0.763
	225	2275				
F-Series 4	2069	431	86.22	86.40	86.20	0.726
	258	2242				
F-Stiff-Motifs	2497	3	54.08	75.40	54.10	0.203
	2293	207				
F-Flexible-Motifs	2182	318	86.76	86.80	86.80	0.735
	344	2156				
F-Nomad-Motifs	2498	2	51.98	74.50	52.00	0.139
	2399	101				

TABLE 4.2.4: Results of running Random Forest classifier on series 1 to 4, Stiff-Motifs, Flexible-Motifs, and Nomad-Motifs datasets after getting filtered using feature selection results. (Best, second best, and third best).

classified by SVM-RBF has been demonstrated in Table 4.2.6. Clearly, the accuracy of F-Series 1, F-Series 2, F-Series 3, F-Series 4, and F-Flexible-Motifs datasets are all around 86% and the best result is gained from classifying F-Flexible-Motifs dataset using SVM-RBF (c=10 and g=0.1) with 87.92% accuracy. Stiff-Motifs and Nomad-Motifs datasets also got 54.08% and 51.98% accuracy.

SVM Grid-Search	F-Series 1	F-Series 2	F-Series 3	F-Series 4	F-Stiff-Motifs	F-Flexible-Motifs	F-Nomad-Motifs
SVM-Polynomial	70.24	75.10	74.40	74.90	54.08	80.34	51.98
SVM-RBF c=10, g=0.01	84.18	83.88	85.86	84.50	54.08	84.88	51.98
SVM-RBF c=10, g=0.1	85.52	85.42	87.30	86.20	54.08	**87.92**	51.98
SVM-RBF c=10, g=1	86.08	85.76	87.78	86.08	54.08	85.58	51.98
SVM-RBF c=10, g=10	85.80	84.92	87.34	84.72	54.08	79.52	51.98
SVM-RBF c=10, g=100	85.80	84.92	87.34	84.72	54.08	79.52	51.98
SVM-RBF c=10, g=1,000	85.80	84.92	87.34	84.72	54.08	79.52	51.98
SVM-RBF c=10, g=5,000	85.80	84.92	87.34	84.72	54.08	79.52	51.98
SVM-RBF c=10, g=10,000	85.80	84.92	87.34	84.72	54.08	79.52	51.98
SVM-RBF c=10, g=20,000	85.80	84.92	87.34	84.72	54.08	79.52	51.98
SVM-RBF c=10, g=100,000	85.80	84.92	87.34	84.72	54.08	79.52	51.98
SVM-RBF g=0.01 , c=1	82.94	83.06	84.66	82.94	53.76	83.06	51.98
SVM-RBF g=0.01 , c=10	84.18	83.88	85.86	84.50	54.08	84.88	51.98
SVM-RBF g=0.01 , c=100	84.34	84.42	85.98	84.66	54.08	85.18	51.98
SVM-RBF g=0.01 , c=1,000	84.02	84.48	86.14	84.94	54.08	85.40	51.98
SVM-RBF g=0.01 , c=10,000	84.38	83.82	85.76	84.58	54.08	84.70	51.98
SVM-RBF g=0.01 , c=100,000	84.92	83.68	86.06	83.88	54.08	83.94	51.98
SVM-RBF g=0.01 , c=1,000,000	82.70	76.66	84.36	82.16	54.08	78.44	51.98

TABLE 4.2.5: Grid-search on SVM reveals the best obtained results for each of 7 datasets filtered using feature selection results . Values are gained accuracy(%) after running SVM on datasets.

SVM-RBF	Confusion Matrix		Accuracy(%)	Precision(%)	Recall(%)	MCC
Series 1(c=10, g=1)	2067	433	86.08	86.20	86.10	0.723
	263	2237				
Series 2 (c=10, g=1)	2092	408	85.76	85.80	85.80	0.716
	304	2196				
Series 3 (c=10, g=1)	2164	336	87.78	87.80	87.80	0.756
	275	2225				
Series 4 (c=10, g=0.1)	2010	490	86.20	86.70	86.20	0.729
	200	2300				
Stiff-Motifs (c=10, g=0.1)	2497	3	54.08	75.40	54.10	0.203
	2293	207				
Flexible-Motifs (c=10, g=0.1)	**2125**	**375**	**87.92**	**88.00**	**87.90**	**0.760**
	229	**2271**				
Nomad-Motifs (c=10, g=0.1)	2498	2	51.98	74.50	52.00	0.139
	2399	101				

TABLE 4.2.6: Results of running SVM-RBF classifier on series 1 to 4, Stiff-Motifs, Flexible-Motifs, and Nomad-Motifs datasets after getting filtered using feature selection results. (Best, second best, and third best).

4.3 Comparison

4.3.1 Comparison of classifiers performances on original datasets

As illustrated in Figure 4.3.1, regardless of which classifier is used, results of Series 1, Series 2, Series 3, and Series 4 datasets are always among the best with around 2 to 3 percent difference. Besides, since the best results for three out of five classifiers (NaiveBayes, SVM-Polynomial, and SVM-RBF), including the best result achieved in our experiment with 93.7% accuracy (SVM-RBF), are obtained classifying the Flexible-Motifs dataset, it can be concluded that creating a dataset using low scored motifs in some cases can enhance the classification results. Even in KNN, and Random Forest results that Flexible-Motifs dataset is not the best one, its results are much closer to the best ones with less than 1% difference. This is because low scored motifs have more wild-cards in their pattern, and having more wild-cards increases the chance of finding sites while creating the final datasets, which eventually leads to a better dataset. On the other hand, none of the classifiers could have a good performance on Stiff-Motifs. This also shows that using high

scored motifs, which have more fixed-cards in their patterns than wild-cards, decreases the chance of finding sites, and consequently the quality of the dataset. Indeed, the best results for our last dataset, Nomad-Motifs, obtained from KNN, Random Forest, and SVM-RBF with almost 80% accuracy.

Finally, among all the classifiers we used in our experiment, SVM-Polynomial was the weakest and KNN, Random Forest, and SVM-RBF all performed very well. From another point of view, among all our datasets, Series 1, Series 2, Series 3, Series 4, Flexible-Motifs were almost equally the best datasets.

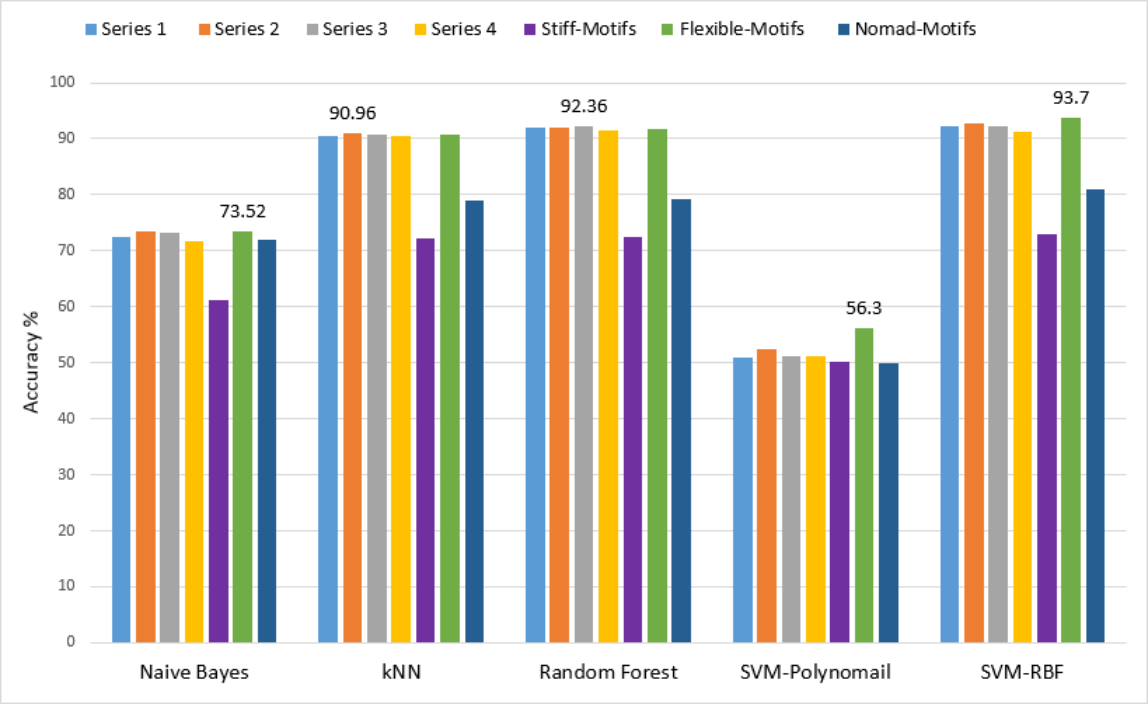

FIGURE 4.3.1: Comparing performance of each classifier over all original datasets.

4.3.2 Comparison of classifiers performances on filtered datasets

As shown in Figure 4.3.2, KNN, Random Forest, and SVM-RBF could classify F-Series 1, F-Series 2, F-Series 3, F-Series 4, and Flexible-Motifs filtered datasets all with around 85% accuracy, including the best result obtained from classifying F-Series 3 with Random Forest with 88.06% accuracy.

While NaiveBayes could not classify datasets with more that 73.76% accuracy, SVM-Polynomial achieved 80.34% accuracy classifying F-Flexible-Motifs dataset.

It can be concluded from the figure that KNN, Random Forest, and SVM-RBF all performed well classifying filtered datasets. Besides, among all filtered datasets F-Series 3, and Flexible-Motifs were the best ones.

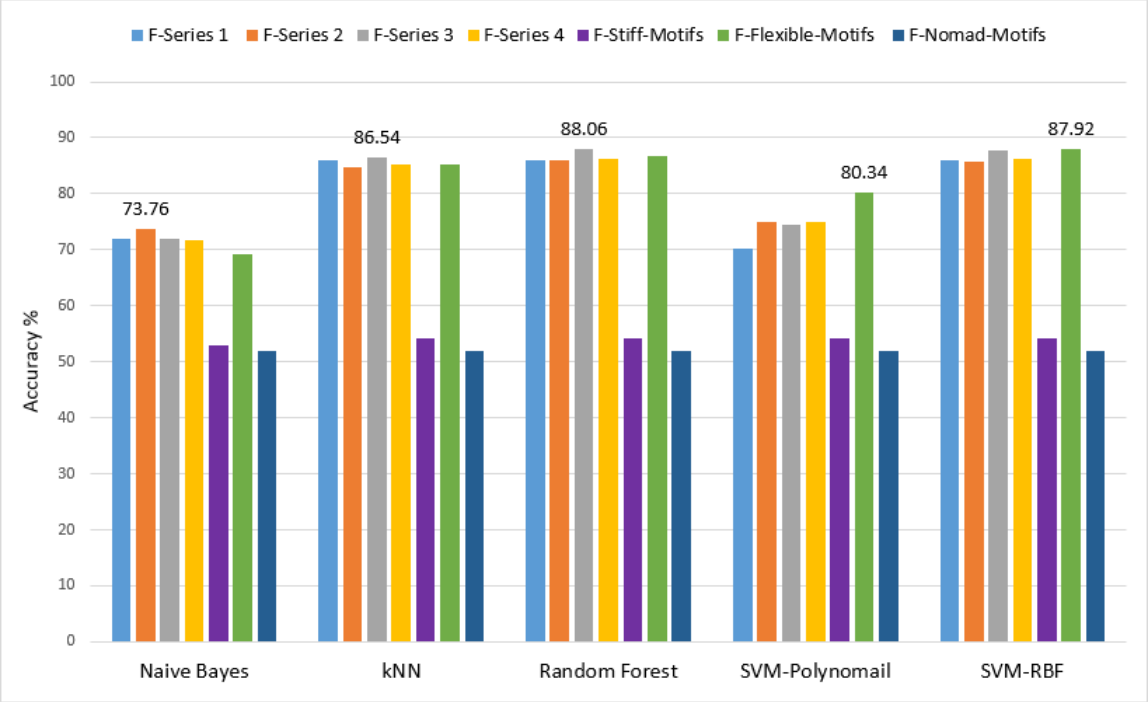

FIGURE 4.3.2: Comparing performance of each classifier over all filtered datasets.

4.3.3 Original datasets VS filtered datsets

As shown in Figure 4.3.3, accuracy of KNN, Random Forest, and SVM-RBF dropped by almost 5% for Series 1, Series 2, Series 3, Series 4, and Flexible-Motifs after feature selection. Besides, while after feature selection accuracy of NaiveBayes either did not change or dropped, feature selection surprisingly enhanced SVM-Polynomial performance for all the datasets. This enhancement significantly increased the SVM-Polynomial performance for some datasets like Series 1, Series 2, Series 3, Series 4, and Flexible-Motifs by up to 24%. However, even the best results obtained by SVM-Polynomial, which is 80.34% accuracy for Flexible-Motifs dataset, is not as good as best results obtained from KNN, Random Forest, and SVM-RBF.

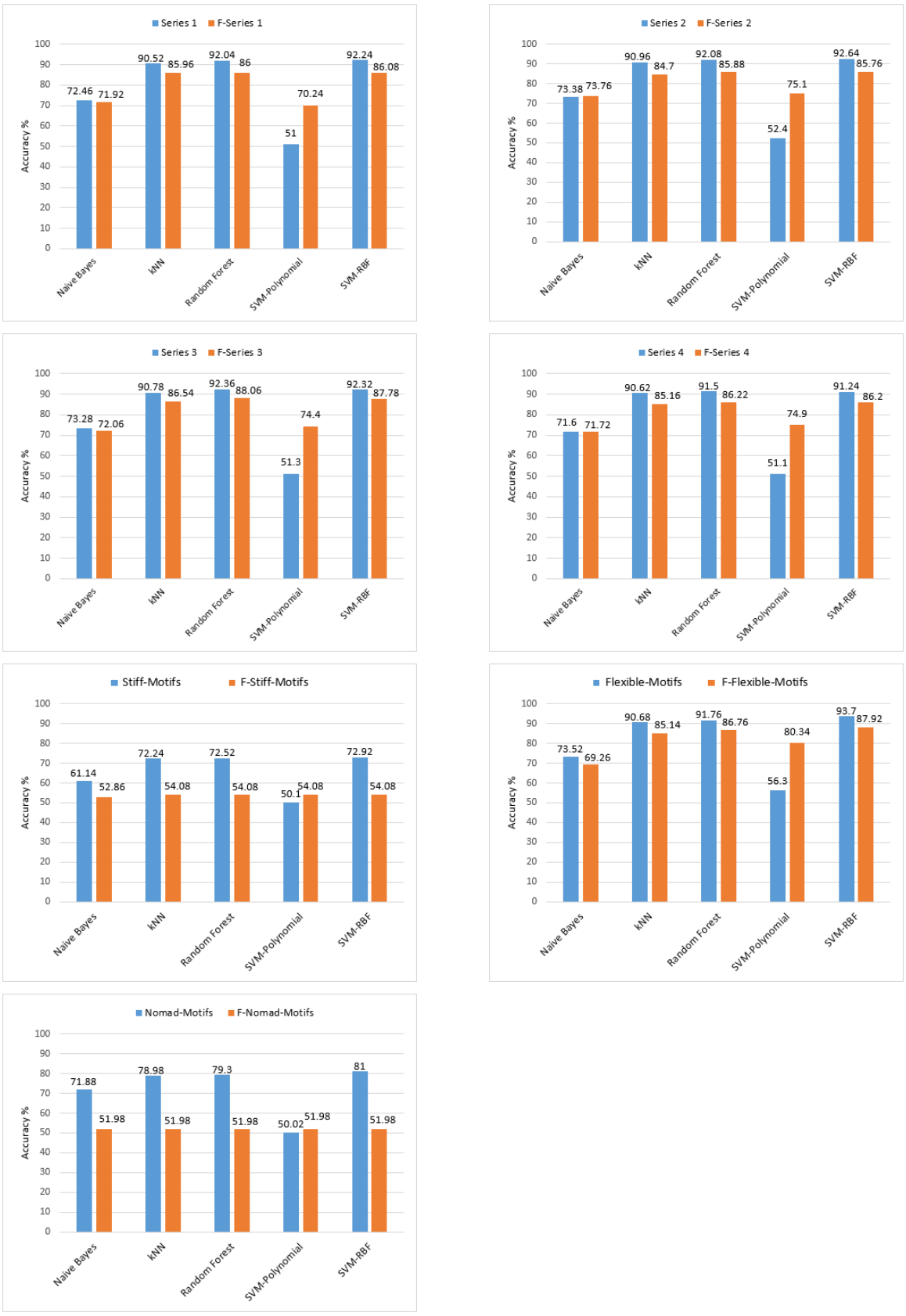

FIGURE 4.3.3: Comparing results of classifying each dataset, before and after feature selection.

4.3.4 Motifs VS Nomad

As illustrated in Figure 4.3.4, Nomad is much faster than MEME in terms of discovering motifs, such that finding 5000 motifs with Nomad only takes less than 2 days, while it takes around 10 days to discover 5000 motifs with MEME.

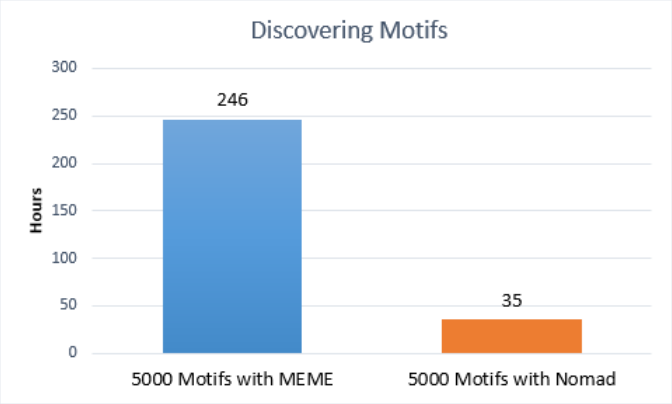

FIGURE 4.3.4: Time spent to discover 5000 motifs with MEME and Nomad.

However, as shown in Figure 4.3.5, the best results that we could achieve among all the datasets created by motifs discovered by MEME was from classifying Flexible-Motifs dataset with SVM-RBF with almost 94% accuracy, while Nomad-Motifs dataset could never be classified with any classifier with more than 81% accuracy. Thus, in our case MEME proved to be a better tool for motif discovery.

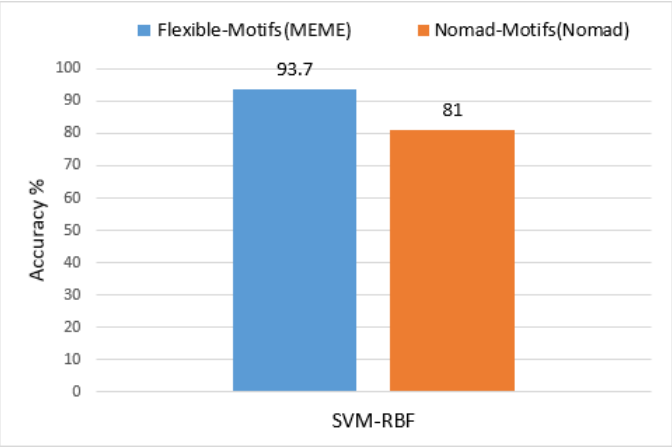

FIGURE 4.3.5: Comparing the best result obtained from motifs discovered by MEME, with the best result obtained from Nomad-Motifs dataset.

CHAPTER 5

Conclusion and Future Work

5.1 Contributions

We proposed a novel method to deal with the problem of finding large number of motifs from large datasets, and use them for prediction of protein-protein interactions. In our method, first we chose 2500 interacting and 2500 non-interacting protein pairs, and after curating the dataset, we divided the whole dataset into 100 small subsets and randomly selected 25 interacting and 25 non-interacting for each subset. Using the same idea we created three more series of subsets to see how different grouping changes the classification results. At this point, instead of passing the whole dataset to MEME, we separately passed subsets of each series to MEME to discover novel motifs. As explained earlier, we used a function to score all the motifs and created two more datasets based on the flexibility of the motifs. We also used Nomad to discover motifs from our original dataset to be able to compare the results of MEME and Nomad. After that we used five different classifiers to predict protein-protein interactions. We also used mRmR feature selection to see if it can help the classifiers with removing the noises.

The fact that results of Series 1, 2, 3, and 4 datasets were almost the same regardless of which classifier is used, shows that changing the orders of protein pairs in the subsets does not have so much effects on classification results. However, considering the results obtained from these datasets have always been among top three and above 90%, it can be concluded that the proposed method is effective. Furthermore, the results obtained from Stiff-Motifs and Flexible-Motifs datasets reveals the importance of motifs wild-cards. While the accuracy of classifying Stiff-Motifs dataset never exceeded 73%, results obtained for classifying

Flexible-Motif have always been either the best, or so close to the best. This proves that using flexible motifs for creating the dataset enhances the performance of classifies, because having patterns with lower scores means having more wild-cards, which eventually leads to find more sites and have better dataset.

Although feature selection significantly enhanced the performance of SVM-Polynomial, the accuracy of other classifiers decreased by almost 5%. As a result, we state that in our case feature selection could not help classifiers to obtain better results in total.

5.2 Future Work

I divided the dataset into hundred subsets of size fifty protein pairs (half interacting and half non-interacting). Other combination of the number of subsets and their size can be taken into consideration for further studies. Besides, I simply added up the number of sites I found in each protein pairs to create final datasets. However, scoring the sites with existing formulas from other works may be used. Furthermore, the motifs selected by feature selection can somehow be related to each other. Studying their relation can be a possible extension to this work. Finally, other feature selection methods can be used with the aim of obtaining better results. Therefore, all options for extending this work can be summarized as follows:

- Changing the subsets number and size to see how enlarging or shrinking the subsets might change the classification results.
- Scoring the sites with different scoring functions.
- The relationship between the discovered motifs can be taken into consideration for further investigation.
- Other feature selection methods can be used.

REFERENCES

[1] Ahmed, H. R. and Glasgow, J. I. J. (2014). Pattern discovery in protein networks reveals high-confidence predictions of novel interactions. *AAAI*, pages 2938–2945.

[2] Aziz, M., Maleki, M., Rueda, L., Raza, M., Banerjee, S., et al. (2011). Prediction of biological protein–protein interactions using atom-type and amino acid properties. *Proteomics*, 11(19):3802–3810.

[3] Bailey, T. L., Bodén, M., Whitington, T., and Machanick, P. (2010). The value of position-specific priors in motif discovery using meme. *BMC Bioinformatics*, 11(1):1.

[4] Bailey, T. L., Johnson, J., Grant, C. E., and Noble, W. S. (2015). The meme suite. *Nucleic Acids Research*, 43(W1):W39–W49.

[5] Blohm, P., Frishman, G., Smialowski, P., Goebels, F., Wachinger, B., Ruepp, A., and Frishman, D. R. (2014). Negatome 2.0: a database of non-interacting proteins derived by literature mining, manual annotation, and protein structure analysis. *Nucleic Acids Research*, page gkt1079.

[6] Browne, F., Zheng, H., Wang, H., and Azuaje, F. (2010). From experimental approaches to computational techniques: a review on the prediction of protein-protein interactions. *Advances in Artificial Intelligence*, 2010:7.

[7] Chan, J. C.-W. and Paelinckx, D. (2008). Evaluation of random forest and adaboost tree-based ensemble classification and spectral band selection for ecotope mapping using airborne hyperspectral imagery. *Remote Sensing of Environment*, 112(6):2999–3011.

[8] Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

[9] Consortium, T. U. (2014). Uniprot: a hub for protein information. *Nucleic Acids Research*, page gku989.

[10] Davey, N. E., Haslam, N. J., Shields, D. C., and Edwards, R. J. (2011a). SLiMSearch 2.0: biological context for short linear motifs in proteins. *Nucleic Acids Research*, 39(suppl 2):W56–W60.

[11] Davey, N. E., Travé, G., and Gibson, T. J. (2011b). How viruses hijack cell regulation. *Trends in Biochemical Sciences*, 36(3):159–169.

[12] Dinkel, H., Van Roey, K., Michael, S., Kumar, M., Uyar, B., Altenberg, B., Milchevskaya, V., Schneider, M., Kühn, H., Behrendt, A., et al. (2015). ELM 2016data update and new functionality of the eukaryotic linear motif resource. *Nucleic Acids Research*, page gkv1291.

[13] Edwards, R. J., Davey, N. E., and Shields, D. C. (2007). SLiMFinder: a probabilistic method for identifying over-represented, convergently evolved, short linear motifs in proteins. *PloS one*, 2(10):e967.

[14] Eidhammer, I., Jonassen, I. T., William, R., and Inge Jonassen, W. R. T. (2004). *Protein Bioinformatics: An algorithmic approach to sequence and structure analysis*. John Wiley and Sons, Ltd.

[15] Gould, C. M., Diella, F., Via, A., Puntervoll, P., Gemünd, C., Chabanis-Davidson, S., Michael, S., Sayadi, A., Bryne, J. C., Chica, C., et al. (2009). ELM: the status of the 2010 eukaryotic linear motif resource. *Nucleic Acids Research*, page gkp1016.

[16] Guo, Y., Yu, L., Wen, Z., and Li, M. (2008). Using support vector machine combined with auto covariance to predict protein-protein interactions from protein sequences. *Nucleic Acids Research*, 36(9):3025–3030.

[17] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

[18] Haslam, J., N., Shields, and C., D. (2012). Profile-based short linear protein motif discovery. *Bmc Bioinformatics*, 13(1):1.

[19] Hernandez, D., Gras, R., and Appel, R. (2008). Neighborhood functions and hill-climbing strategies dedicated to the generalized ungapped local multiple alignment. *European Journal of Operational Research*, 185(3):1276–1284.

[20] Jurisica, I. and Wigle, D. (2005). *Knowledge Discovery in Proteomics*. CRC press.

[21] Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. (2007). Supervised machine learning: A review of classification techniques.

[22] Leung, K. M. (2007). Naive bayesian classifier. *Polytechnic University Department of Computer Science/Finance and Risk Engineering*.

[23] Li, B.-Q., Feng, K.-Y., Chen, L., Huang, T., and Cai, Y.-D. (2012). Prediction of protein-protein interaction sites by random forest algorithm with mrmr and ifs. *PloS one*, 7(8):e43927.

[24] Li, Y. (2016). Prediction of high-throughput protein-protein interactions and calmodulin binding using short linear motifs. *Thesis Dissertation*.

[25] Li, Y., Rezaei, B., Ngom, A., and Rueda, L. (2015). Prediction of high-throughput protein-protein interactions based on protein sequence information. *Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–6.

[26] Nooren, I. M. and Thornton, J. M. (2003). Diversity of protein–protein interactions. *The EMBO Journal*, 22(14):3486–3492.

[27] Park, S. H., Reyes, J. A., Gilbert, D. R., Kim, J. W., and Kim, S. (2009a). Prediction of protein-protein interaction types using association rule based classification. *BMC Bioinformatics*, 10(1):1.

[28] Park, S. H., Reyes, J. A., Gilbert, D. R., Kim, J. W., and Kim, S. J. (2009b). Prediction of protein-protein interaction types using association rule based classification. *BMC Bioinformatics*, 10(1):1.

[29] Peterson, L. E. (2009). K-nearest neighbor. *Scholarpedia*, 4(2):1883.

[30] Ren, S., Yang, G., He, Y., Wang, Y., Li, Y., and Chen, Z. (2008). The conservation pattern of short linear motifs is highly correlated with the function of interacting protein domains. *BMC Genomics*, 9(1):1.

[31] Rish, I. (2001). An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM New York.

[32] Rodriguez-Galiano, V. F., Ghimire, B., Rogan, J., Chica-Olmo, M., and Rigol-Sanchez, J. P. (2012). An assessment of the effectiveness of a random forest classifier for land-cover classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 67:93–104.

[33] Rueda, L. and Pandit, M. (2014a). A model based on minimotifs for classification of stable protein-protein complexes. *Computational Intelligence in Bioinformatics and Computational Biology, 2014 IEEE Conference*, pages 1–6.

[34] Rueda, L. and Pandit, M. (2014b). A model based on minimotifs for classification of stable protein-protein complexes. *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB 2014), Hawaii, USA*, pages 1–6.

[35] Saeys, Y., Inza, I., and Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517.

[36] Schapire, R. E. (2003). The boosting approach to machine learning: An overview. In *Nonlinear estimation and classification*, pages 149–171. Springer.

[37] Schiller, M. R., Mi, T., Merlin, J. C., Deverasetty, S., Gryk, M. R., Bill, T. J., Brooks, A. W., Lee, L. Y., Rathnayake, V., Ross, C. A., et al. (2011). Minimotif miner 3.0:

database expansion and significantly improved reduction of false-positive predictions from consensus sequences. *Nucleic Acids Research*, page gkr1189.

[38] Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N., Schwikowski, B., and Ideker, T. (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504.

[39] Sharma, T. C. and Jain, M. (2013). Weka approach for comparative study of classification algorithm. *International Journal of Advanced Research in Computer and Communication Engineering*, 2(4):1925–1931.

[40] Sigrist, C. J., De Castro, E., Cerutti, L., Cuche, B. A., Hulo, N., Bridge, A., Bougueleret, L., and Xenarios, I. (2012). New and continuing developments at prosite. *Nucleic Acids Research*, page gks1067.

[41] Skrabanek, L., Saini, H. K., Bader, G. D., and Enright, A. J. (2008). Computational prediction of protein–protein interactions. *Molecular Biotechnology*, 38(1):1–17.

[42] Stelzl, U., Worm, U., Lalowski, M., Haenig, C., Brembeck, F. H., Goehler, H., and Timm, J. (2005). A human protein-protein interaction network: a resource for annotating the proteome. *Cell*, 122(6):957–968.

[43] T OBrien, K., Haslam, N. J., and Shields, D. C. (2013). SLiMScape: a protein short linear motif analysis plugin for Cytoscape. *BMC Bioinformatics*, 14(1):1.

[44] Vasudev, G. and Rueda, L. (2012). A model to predict and analyze protein-protein interaction types using electrostatic energies. In *Bioinformatics and Biomedicine (BIBM), 2012 IEEE International Conference on*, pages 1–5. IEEE.

[45] Veropoulos, K., Campbell, C., Cristianini, N., et al. (1999). Controlling the sensitivity of support vector machines. In *Proceedings of the international joint conference on AI*, pages 55–60.

[46] Weinberger, K. Q., Blitzer, J., and Saul, L. K. (2005). Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480.

[47] Zhang, Q. C., Petrey, D., Garzón, J. I., Deng, L., and Honig, B. (2012). PrePPI: a structure-informed database of protein-protein interactions. *Nucleic Acids Research*, page gks1231.

[48] Zhou, Y., Zhou, Y. S., He, F., Song, J., and Zhang, Z. (2012). Can simple codon pair usage predict protein-protein interaction? *Molecular BioSystems*, 8(5):1396–1404.

[49] Zhu, H., Domingues, F. S., Sommer, I., and Lengauer, T. (2006). NOXclass: prediction of protein-protein interaction types. *BMC Bioinformatics*, 7(1):1.

VITA AUCTORIS

NAME:	Behzad Rezaei
PLACE OF BIRTH:	Ahvaz, Khouzestan, Iran.
EDUCATION:	Associate Degree in Computer-Software, Jihad university, Ahvaz, Khouzestan, Iran, 2006.
	Bachelor of Science in Computer Science, Payam Golpaygan university , Golpaygan, Isfahan, Iran, 2009.
	Master of Science in Computer Science, University of Windsor, Windsor, Ontario, Canada, 2017.