Discriminating between surfaces of peripheral membrane proteins and reference proteins using machine learning algorithms

Fengming Miao

February 25, 2019

Master's thesis

Department of Informatics
University of Bergen

Abstract

In biology, the cell membrane is an important component of a cell and usually works as a "fence" to distinguish the inside and outside of a cell. The key role is to protect the cells from being interfered by their surroundings by preventing the molecules that will enter into the cell. However as we know, cells need to keep communicating with their surroundings to acquire nutrition and other necessary molecules in order to stay alive and grow. Due to this reason, membrane proteins are used as molecular carriers to participate the molecular communication and regulate the biological activities. There are two kinds of membrane proteins: integral and peripheral. In this project, we only focus on the latter.

Unlike the integral membrane proteins which can go across the whole membrane, peripheral membrane proteins can only attach to the surface of the membrane through various interactions. Because peripheral proteins are also soluble, it is difficult to differentiate them from other kinds of proteins (i.e. non membrane-binding) from sequence or structure. In this project, we will develop a method to predict from its structure wether a protein is membrane-binding protein or not based on two machine learning algorithms: k-nearest neighbors(KNN) and support vector machine(SVM). We use them to train the data and create two models respectively, which will be used to classify new proteins as well as compare their performance.

By for example collecting different features of proteins, adjusting the parameters of the algorithms or changing size and structure of the dataset, we can improve the performances of the algorithms as well as predict the protein type more accurately. We also use ROC curve and AUC to present the performance in overview, and cross validation to verify the result.

For the problems in this field, several challenges should be considered as well, such as collecting of features, analysis and dealing with the huge variety of data, as well as the choice of machine learning algorithms for a design based on functional requirements, data structure, efficiency and other factors. In this project, we will encounter these challenges and solve them by effective methods.

Acknowledgement

I would like first to thank my supervisor Professor Nathalie Reuter in the Computational Biology Unit at the University of Bergen. I can not find a word to express all my gratitude to her. Thanks for giving me the opportunity to work under her instruction and all the patient guidance and everlasting optimistic attitude to my scientific work. The door to Prof. Nathalie's office was always open whenever I ran into a trouble spot or had a question about my research or writing. She really gave me so much advice on my project and guided me to finish the thesis. She would correct me in time every time I had deviations on my work. Her way of doing science impressed me very much and what I learned from her will be a great treasure in my life.

I would also say thanks to Takaya Saito who is an associate Professor in UiB and is extremely experienced in machine learning. He gave me a lot of advice about how to implement machine learning algorithms on my data. He always answered me patiently both by e-mails and in person whenever i had questions.

Another I would thank is Edvin Fuglebakk who offered me the original protein dat sets without reservation. Based on these sets I extracted a part of peripheral membrane proteins and a part of reference proteins with several features, and finally finished my research work.

Last, I also want to say thanks to my colleagues in the office: Hanif Muhammad Khan, Qaiser Waheed and Emmanuel Edouard Moutoussamy. They always gave me help passionately when I had questions. I really had a nice time with them.

Contents

1 Introduction **6**
- 1.1 Background . . . 6
- 1.2 Objectives . . . 6
- 1.3 Thesis overview . . . 6

2 Proteins **7**
- 2.1 Biosynthesis and definition . . . 7
- 2.2 Categories . . . 7
- 2.3 Structures . . . 7
- 2.4 Cell membrane . . . 8
- 2.5 Peripheral membrane proteins . . . 9

3 Machine learning **10**
- 3.1 Definition . . . 10
- 3.2 Categories . . . 10
  - 3.2.1 Supervised learning . . . 10
  - 3.2.2 Unsupervised learning . . . 10
  - 3.2.3 Reinforcement learning . . . 12

4 Collecting of data and features **13**
- 4.1 Definitions . . . 14
- 4.2 First feature . . . 14
  - 4.2.1 ASA . . . 14
- 4.3 Second feature . . . 15
- 4.4 Third feature . . . 15
- 4.5 Fourth feature . . . 17
- 4.6 Feature summary . . . 17

5 Algorithm selection **18**
- 5.1 KNN . . . 18
  - 5.1.1 Principle . . . 18
  - 5.1.2 Distance function . . . 18
  - 5.1.3 Factor K . . . 19
  - 5.1.4 Application . . . 21
- 5.2 SVM . . . 21
  - 5.2.1 Principle . . . 22
  - 5.2.2 Linear classification . . . 22
  - 5.2.3 Non-linear classification . . . 25
  - 5.2.4 γ and C . . . 28
  - 5.2.5 Applications . . . 29

6 Feature analysis **30**
6.1 Why . . . 31
6.2 Features visualization . . . 31
6.3 Features selection . . . 33

7 Algorithm implementation **36**
7.1 KNN function . . . 36
7.2 SVM function . . . 36
7.3 Noise elimination . . . 38
7.4 Feature scaling . . . 38
7.5 Balanced dataset . . . 38
7.6 Performance formula . . . 39
7.7 Wrapper method . . . 40
7.8 KNN performance . . . 41
7.8.1 Alternative dataset . . . 41
7.9 SVM performance . . . 41
7.10 ROC-AUC curve . . . 44

8 Result verification **47**
8.1 Cross-Validation . . . 47
8.2 Steps . . . 47
8.3 Cross-Validation on KNN . . . 47
8.4 Cross-Validation on SVM . . . 48

9 Conclusion and future work **51**
9.1 Conclusion . . . 51
9.2 Future work . . . 51

1 Introduction

1.1 Background

Along with the development of society and technologies, traditional analysis tools could not satisfy the requirements of modern data. The volume of data is becoming bigger and bigger and the quality is becoming more and more complicated. Therefore the choice of a intelligent analytical method has been a top priority. Machine learning is an ideal choice for solving this kind of data. It is becoming more and more popular and can be used in many different fields, such as industry, service, agriculture and so on. In this article, we will implement machine learning in biology and give an overview about how it is capable of classifying peripheral membrane proteins based on protein data. The proper implementation of machine learning can liberate the workforce, save production costs, improve work efficiency and finally change our world.

1.2 Objectives

Peripheral membrane proteins have complicated and various structures, with more and more new proteins are found, it is time-consuming to determine the type(peripheral membrane proteins or not) using traditional methods. Moreover, unlike transmembrane proteins which can be identified by their sequences, peripheral proteins are arduous to detect using normal detection methods. Therefore we aim at finding an efficient way in which we can find internal rules and make an accurate prediction of all these new proteins rapidly through plenty of data collected from thousands of proteins of whose structures are already known. That is why we chose a machine learning algorithm by which we train the collected data and form a model to predict new proteins.

1.3 Thesis overview

In this article, first of all we will talk about the definition of proteins, their structures and functions. In third chapter, we will introduce machine learning as well as its categorization. In fourth chapter, we collect data and features of proteins from a original dataset. When the data is ready, suitable algorithms are then picked up according to the structure of dataset we have created. In sixth chapter, we will carry on features analysis to see which features are representative and choose those that give us the best performance through machine learning algorithms. After that, we come to the core part in this article–implementing the machine learning algorithms, we will try two algorithms in order to compare their performances. A model validation techniques will be tried as well to estimate how accurately a predictive model from machine learning will perform in practice. At last, the conclusion analysis will be included to display the exciting results from our model by various ways. We will also present the work we will do for potential improvement in the future.

2 Proteins

2.1 Biosynthesis and definition

Proteins are large biomolecules and play a critical role in the biological processes. Amino acids chains are the main components of them. As we know, the formation of proteins has gone through a complex process. Briefly it follows a central dogma: "*DNA makes RNA and RNA makes protein*"[1]. DNA contains two strands which are composed of four nucleotide bases(A-adenine, T-thymine, G-guanine, C-cytosine) in a certain order. These two strands are combined together to form the double strands DNA based on pairing rule of bases(A corresponds T and G corresponds C) and hydrogen bonds. The process from DNA to RNA is called transcription[2], where one of strands is transcribed into a RNA single chain according to base complementarity principle[3] by RNA polymerase[4].

The process from RNA to protein is called translation[5], where base sequences of mRNA are decoded to generate corresponding amino acids. Every amino acid contains three nucleotide bases and are called a codon[6] in the gene fragment of DNA. Since there are four bases, therefore the total number of amino acids should be 64($4\times4\times4$). However, in biology the number of amino acids is 20, which indicates that there is a superabundance in the representation of bases(some amino acids are represented by more than one codon). For example: ACT represents Threonine[6], but ACC, ACA, ACG also mean Threonine.

A short sequence of amino acids is called a peptide[7] and a long sequence is called a polypeptide[7] or a chain. Each protein contains one or more chains. Difference sequences of amino acids generate different proteins.

2.2 Categories

In biology, there are many kinds of proteins with various functions. For example: antibody[8] is a kind of protein and can be used to neutralize viruses, which will keep our bodies from being infected diseases; Also a part of proteins can work as signaling for biological processes; and some can help to transport molecules and other materials from one place to another; In addition, most of enzymes[9] are also proteins which influence plenty of biological reactions.

2.3 Structures

Most proteins can fold into 3-dimensional structures. There are four different structures: Primary structure; Secondary structure; Tertiary structure and Quaternary structure. From figure 2.1, you can see how it looks like for every stage.

We focus on the protein tertiary structure in this project. Our extraction of protein features is based on this.

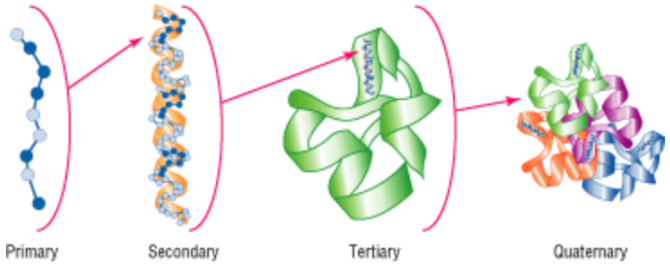

Figure 2.1: The four different levels of protein structures.(Source: "*The Four Levels of Protein Structure*"[10])

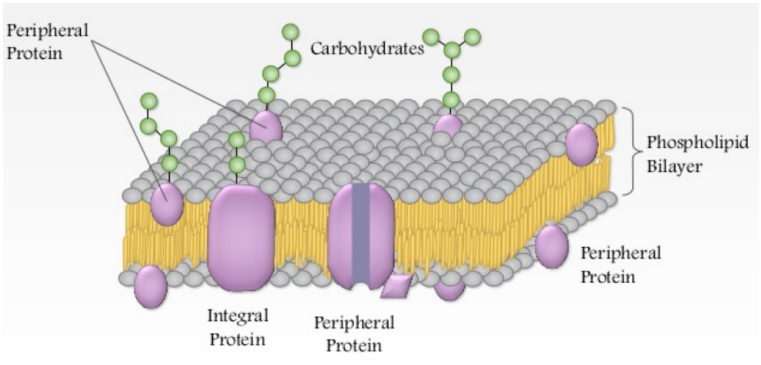

Figure 2.2: Integral proteins and peripheral proteins: Integral proteins are fully embedded in the bilayer of membrane, while peripheral proteins only attach on its surface. (Source: "*Integral and peripheral membrane proteins*"[15])

2.4 Cell membrane

As we know, cell membrane[11] is the "*fence*" to separate the interior of cells from their exterior. In addition, it also participates many biological processes and serves as the attachment points for peripheral proteins.[12]

The main component of a cell membrane is phospholipids[13] which contain both a hydrophilic head and a hydrophobic tail. Hydrophobic tails usually gather together to form a 'sheet' via non-covalent interactions[14]. Two 'sheets' then compose a lipid bilayer with their tails facing inward and heads outward(see figure 2.2). The area between two layers works as a barrier to protect the cell.

2.5 Peripheral membrane proteins

Membrane proteins are proteins that interact with membranes. There are two different membrane proteins in the lipid bilayer: integral membrane proteins[16] and peripheral membrane proteins. Integral membrane proteins are placed across the whole lipid bilayer, while peripheral membrane proteins(PMP) only attach on the surface of the membranes(see figure 2.2).[17][18] This kind of attachment plays an important role for many intracellular and extracellular activities.

Membrane proteins are very common in protein families. A lot of genes in genomes are used to encode membrane proteins.[19][20]

3 Machine learning

3.1 Definition

Machine Learning (ML) covers a large and diverse range of methods involving algorithm, probability and statistics etc. It shows how computers learn the regular pattern of existing knowledge in different fields and apply the obtained rules on new knowledge. It is the central part of artificial intelligence(AI)[21].

3.2 Categories

It can be divided into the following three categories.

3.2.1 Supervised learning

Supervised learning is a kind of learning based on both input and output values of samples. It trains a model based on samples by finding out the rules in the data, and use this model to predict new samples.

For instance, let's say you want to decide if you will do an outdoor activity one day according to the weather situation, such as temperature, humidity, wind power, traffic condition etc. Then your model will be trained on historical data with these features and use this model to decide a new day on which you will do a outdoor activity or not. Hence the model is supervised.

Typical supervised learning algorithms: Support vector machine, Naive Bayes, Decision Tree, Neural network, K-Nearest Neighbors and Gradient boosting.[22]

3.2.2 Unsupervised learning

In contrast, unsupervised learning is a learning where we only have input values and don't have output values in the dataset. The purpose of unsupervised learning is to find out the distribution pattern of data. It contains two types: clustering problem and association problem[23]. Figure 3.2 is a example of clustering.

For a clustering problem, we take the same example as we used for supervised learning. This time we will automatically partition people into different groups(people in this group like this kind of activity and people in another group like other activities) according to several variables like their hobbies, weather condition, traffic condition etc, based on an unsupervised learning algorithm. For an association problem, we further want to know the possibility that they will do another activity based on what they like.

The main algorithms for unsupervised learning are k-means and Association Rules.[23]

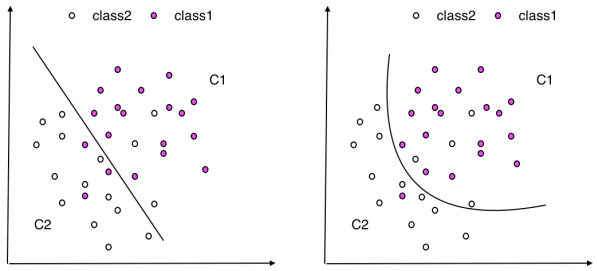

Figure 3.1: The figure above shows a classification problem for samples. There are two classes(C1 and C2). The straight line represents a linear boundary(left) and the curve represents a quadratic boundary(right), which are used to define the regions C1 and C2. New observations will be classified into the class C1 or C2 depending on which region they will fall into. Classifiers are not perfect, because they can not classify the points totally, some points are classified wrongly.

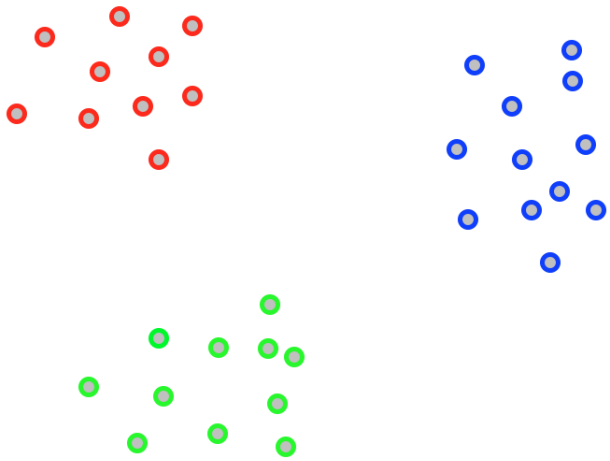

Figure 3.2: There are three different groups in the data, they can be partitioned clearly using unsupervised learning

3.2.3 Reinforcement learning

Reinforcement learning[24] is the main learning of artificial intelligence(AI). The machine based on this learning needs to make different decisions. For instance, a self-driving car is driving on the road and needs to decide when and where it should turn or when it must be stopped. By continuously collecting information from environment to train models for itself using this learning, the car can make accurate decisions.

The most typical algorithm of this learning is Q-Learning, Deep Adversarial Networks.[23]

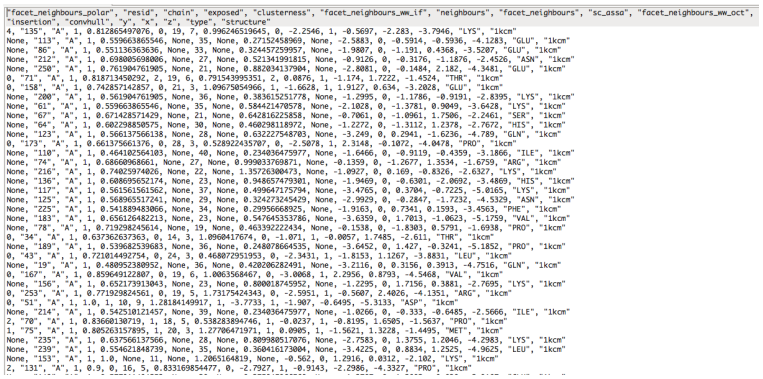

Figure 4.1: The original protein structure dataset with features.(Source: "*A model for hydrophobic protrusions on peripheral membrane proteins*"[25])

4 Collecting of data and features

In order to implement machine learning algorithms, we obviously need to collect data from the structure of proteins. Since data is the basis of any data analysis, it is also ideal that we have enough data for machine learning algorithms to learn and form a model that can reflect the pattern of the data. Moreover, the data we collect must reflect the essential structure of proteins and should be obviously different between peripheral proteins and reference proteins(those are not membrane-binding proteins).

In this article, we use protein structure dataset collected by Edvin Fuglebakk who was a former member of the research group. He formulated a model for protrudings on protein surface using properties of the vertices of a convex hull defined by the c_α and c_β atoms of the protein in the dataset.

This dataset contains amino acids from a set of peripheral membrane proteins and a set of reference proteins and lists many features of proteins. Figure 4.1 shows how the dataset looks like.

There are 17 features in the original dataset. According to the advice from my supervisor, we have chosen 6 features that we will use in this project:

A: **structure** is the code of protein in OPM database, which is consistent with PDB file. It specifies which protein the amino acid belongs to.

B: **type** is three letter code of residue.

C: **exposed** is 1 if this residue is exposed on the surface, otherwise 0.

D: **neighbours** is the number of c_α and c_β atoms this residue has within 1 nm. It reflects the protein density around the amino acid.

E: **convhull** is 1 if residues is on the convex hull, otherwise 0.

F: **facet-neighbours-ww-if** is the number of vertices that share a facet with this residue on the convex hull. 'None' means the residue is not on the convex hull. Here we

only restrict to co-insertable hydrophobic protruding residues. We will interpret all these items in the following except A and B which we have expounded in chapter 2.

We take the dataset of Edvin Fuglebakk as an original dataset. The following part will show you how we transform these 6 features from this original dataset into 4 new features in the new dataset.

4.1 Definitions

Before we start, we need to introduce several concepts in order to understand the new features.

C_α and C_β -atoms In organic molecules, "*C_α refers to the first carbon atom that attaches to a functional group, such as a carbonyl, and the carbon next to it is called C_β atom.*"[26] The definitions of C_α and C_β -atoms can also be applied to proteins and amino acids. All the amino acids contain C_α and C_β -atoms except glycine.

Convex hull: In mathematics, "*the convex hull of a set of points(we call this set X) is the smallest convex set that contains X.[27] It can be considered as the set of all convex points in X*".[28] For the convex hull of proteins, it is defined as the smallest possible set of atoms c_α and c_β in the protein domain.

Protruding residues Intuitively, the protrusion is a part that protrudes from an object. We identify protruding residues "*via the calculation of the convex hull of the C_α and C_β coordinates in the protein. The residues whose C_β atom is a vertex of this convex hull are defined as vertex residues and vertex residues are defined as protruding residues if they have low local protein densities which are defined as the number(n) of neighboring C_α and C_β atoms within a distance c(= 1nm) of their C_β atom*".[25]

4.2 First feature

As the first feature, we are going to calculate the frequency of five amino acids(Phenylalanine, Tryptophan, Cysteine, Leucine and Methionine) on the surface of tertiary structure of proteins. The reason why we choose this property as a feature is that these amino acids are hydrophobic and membranes also have a hydrophobic core. The frequency of these five amino acids is higher in the set of peripheral proteins than in the reference set(R).

4.2.1 ASA

We need to define solvent-accessible surface area in order to count the amino acids on the surface.

The accessible surface area (ASA) is the surface area of a protein molecule that is accessible to a solvent. It is normally decided by rolling a ball with a specific radius to 'probe' the surface of the molecule.[29] The radius of the ball used here is 0.14nm.

Another feasible solution is that you can filter the amino acids by calculating the ASA values based on softwares, such as: Freesasa,[30] VMD,[31] NACCESS[32] etc. After calculating the ASA values, a threshold should be set up to identify wether the residue is exposed on the surface or not according to the value. If the ASA value is bigger than this threshold, it is; otherwise, it's not.

We pick up original feature A, B and C to build this new feature. For every protein we calculate the number of those five amino acids that are exposed on the surface(the one whose value is 1 for feature C in original dataset), and divide it by the total number of residues on the surface. The outcome is treated as the value of the first feature in our training set.

4.3 Second feature

As the second feature, we calculate the number of hydrophobic protruding residues. According to the experimental result of Edvin Fuglebakk, there are much more hydrophobic protruding residues on the surface of peripheral proteins than reference proteins.

Since we have defined what are protruding residues above, now we designate the local protein density(d) to be low if d < n. Here the values of c and n are 1nm and 22 respectively, "*which were manually chosen based on a set of six different families of peripheral membrane proteins (C2-domain, PX-domain, Discodin domain, ENTH domain, Lipoxygenases and a Bacterial Phospholipase C)*".[25](See figure 4.2)

Moreover, "*an amino acid is defined to be hydrophobic if it contributes to membrane interface partitioning of peptides. These amino acids are: leucine, isoleucine, phenylalanine, tyrosine, tryptophan, cysteine and methionine*".[25]

Original features D and E are chosen to create the second feature. For every protein, we check every residue if it is on the convex hull and if the number of neighboring C_α and C_β -atoms is less than 22. In addition, we also have to restrict to those listed seven amino acids above. As a result, the count of residues under these three limitations works as the value of this feature.

4.4 Third feature

The third feature is extracted partly based on the definition of second feature. We further restrict only to count the number of co-insertable hydrophobic protruding residues(figure 4.3). This kind of residues are "*hydrophobic protruding residues that connect at least one other residue using a straight line which is an edge of the convex hull*".[25]

We use feature F for this feature. Only the residues values greater than zero are selected and counted.

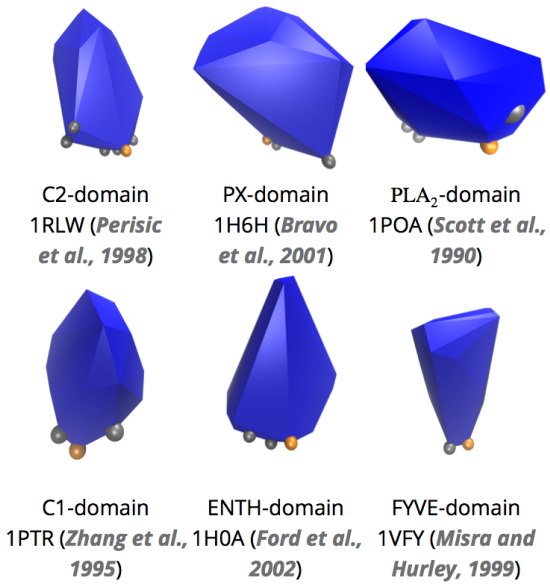

Figure 4.2: Protruding hydrophobes are found on the membrane binding sites of membrane binding domains. The figure shows the convex hull (in blue) of the C_α and C_β atoms of several peripheral membrane binding domains. The C_β atoms of the likely inserted hydrophobe are shown as orange spheres and C_β atoms of experimentally identified membrane-binding residues as gray spheres. 1RLW: C2 domain of human phospholipase A2;[33] 1H6H: PX domain of P40PHOX;[34] 1POA: snake phospholipase A2;[35] 1PTR: C1 domain of protein kinase C delta;[36] 1H0A: Epsin ENTH domain;[37] 1VFY: FYVE domain of yeast vacuolar protein sorting-associated protein 27.[38] (Source: "*A model for hydrophobic protrusions on peripheral membrane proteins*"[25])

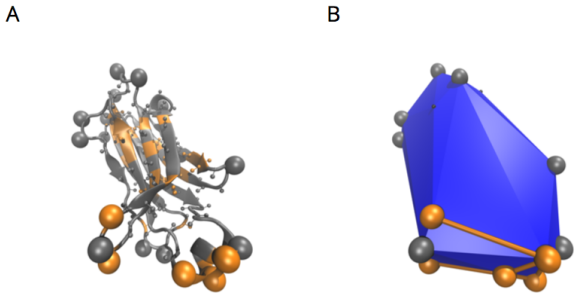

Figure 4.3: Panel A shows a cartoon representation of the C2 domain of human phospholipase A2 (PDB ID: 1RLW), and panel B show the convex hull for the same protein. Co-insertable protruding hydrophobes are connected by orange lines.(Source: "*A model for hydrophobic protrusions on peripheral membrane proteins*"[25])

4.5 Fourth feature

The net charge of the amino acids on the surface of protein is our last feature.[39]

As we know, the quantity of amino acids that make up the proteins is 20, among which only 4 are charged. They are: lysine (+), arginine (+), aspartate (-) and glutamate (-). Thus the net charge is the sum of charged amino acids on the surface of protein.

4.6 Feature summary

Until now, we have extracted 4 features for both peripheral proteins and reference proteins from the original dataset. The first feature is the frequency of those 5 amino acids on the surface of protein; the second feature is the number of hydrophobic protruding residues; the third feature is the number of co-insertable hydrophobic protruding residues and the fourth feature is the net charge of amino acids on the surface. In chapter 6, you will see an example of the new dataset. We will also analyze these features and present an overview of how much they can contribute to the model in that chapter.

5 Algorithm selection

As we have stated in chapter 1, our goal is to develop a method to predict from structure whether a given protein is membrane-binding or not, apparently this is a classification problem. Therefore the machine learning algorithms we choose should be powerful for classification. In addition, the protein dataset on which our classification problem is based has label values and all the feature values are continuous. So the machine learning algorithm we consider for our specific issue should be supervised. Moreover, another factor that need to be considered is that the size of dataset we have created is not so large.

Therefore based on the situations mentioned above, we pick up two algorithms KNN and SVM to implement and compare their performances. Although each learning algorithm fits a specific dataset, they still have some properties in common. So that both of them can be used for this project:

1) Both of them can be used for both classification and regression problems in various fields.

2) For classification problems, both of them can be used either for two classes or multi classes.

3) Both of them are suitable for datasets with small sizes.

5.1 KNN

5.1.1 Principle

The k-nearest neighbor (KNN)[40] algorithm is one of the most traditional algorithms for both classification and regression. Based on the purpose of this project, we only focus on its classification.

As described before, the output values in our dataset are classes(YES means membrane-binding protein and NO means non membrane-binding protein). For every new-added sample, the KNN algorithm will calculate the distance between this sample and all other samples in the dataset through the feature space to find its k-nearest neighbors. Then the class with most samples among these neighbors will be assigned to this new sample.

5.1.2 Distance function

Since KNN is based on calculating the distances between new sample and each of the training samples to decide the final classification output. So how can we calculate the distance between two samples? A feasible solution is: we imagine that for every new sample with N features, the values of features are the coordinates in N-dimensional space and are used to calculate the distance according to distance formula. In figure 5.1, the new sample(* star) will be classified positive(if we take small squares as positive samples and small circles as negative samples). Because among its 5 nearest neighbors, the number

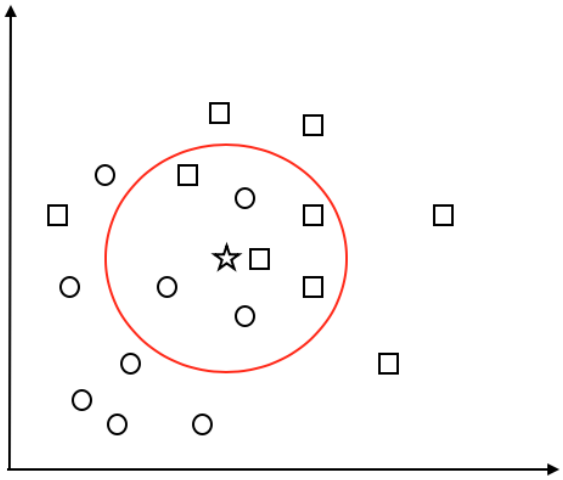

Figure 5.1: k-Nearest neighbor classification. The new sample will be classified into a positive sample.

of positive samples is larger than the number of negative samples based on the voting mechanism.

There are many different functions for calculating the distance, among which Euclidean distance function is used widely and fits our dataset:

Euclidean distance:

$$dist(A, B) = \sqrt{\frac{\Sigma_{i=1}^{m}(x_i - y_i)^2}{m}} \tag{5.1}$$

where A = (x_1 , x_2, ..., x_m), B = (y_1, y_2, ..., y_m) and m is the dimensionality of the space. However, it is worth to try all of them in case another function could give better performance.

5.1.3 Factor K

After learning the principle of KNN, maybe someone would ask: does the choice of k really affect the result? And how to find the optimal k in order to get the best performance of KNN?

Firstly, the choice of the parameter k is very crucial and is somehow the most important parameter in this algorithm, and the choice strictly relies on what kind of data you have.

If you implement KNN with different k values on a two-classes dataset, you will get different boundaries which separate two classes and the boundary will become more and more gentle if k is increased gradually. Figure 5.2 shows this change.

Figure 5.2: The influence of different k to boundaries which separate the two classes.

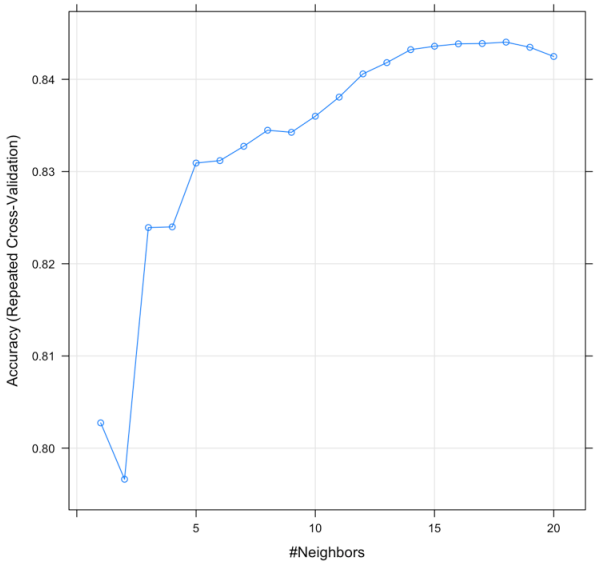

Figure 5.3: The cross-validation accuracy plot with different values of k. In this plot the interval of x-axis(the range of k) is [1 20].

Secondly, there are multiple methods to find the optimal k, such as K-Fold Cross Validation[41] which we will talk about later. The best k value can be found by plotting the cross-validation accuracy with different k. As you can see in figure 5.3 which is calculated using 15 repeats of 10-fold cross-validation. By visualizing the plot, the k value with highest accuracy seems to be located between 15 and 20. Figure 5.4 gives you the resampling results across tuning parameters, where k = 18 corresponds to the highest cross-validation accuracy. You can also extend the range of k values(X-axis in figure 5.3) to give you a more accurate k value, but the running time will be prolonged accordingly.

5.1.4 Application

KNN is an powerful algorithm to solve real world problems in many fields. For instance, KNN can be used for "*visual pattern recognition to scan and detect hidden packages in the bottom of a shopping cart at check-out*".[42] Another application is to for example predict the incidence of some diseases by collecting the medical data of patients. Proper usage of machine learning algorithms can really bring you surprisingly effects.

5.2 SVM

Support vector machine (SVM)[43] is another popular machine learning algorithm for both classification and regression.

```
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.8027407  0.6054815
   2  0.7966296  0.5932593
   3  0.8239259  0.6478519
   4  0.8240000  0.6480000
   5  0.8309259  0.6618519
   6  0.8311852  0.6623704
   7  0.8327407  0.6654815
   8  0.8344815  0.6689630
   9  0.8342593  0.6685185
  10  0.8360000  0.6720000
  11  0.8380741  0.6761481
  12  0.8405926  0.6811852
  13  0.8418148  0.6836296
  14  0.8432222  0.6864444
  15  0.8435926  0.6871852
  16  0.8438519  0.6877037
  17  0.8438889  0.6877778
  18  0.8440370  0.6880741
  19  0.8434815  0.6869630
  20  0.8424815  0.6849630

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was k = 18.
```

Figure 5.4: The resampling results across tuning parameters where we can find the optimal k.

5.2.1 Principle

The principle of SVM for classification is: we consider the features of samples as coordinates and map them into a N-dimensional space(N is decided by the number of features). According to the data and the kernel function in the algorithm, SVM will train a model and classify samples into different classes with the help of a margin and its boundaries. A new sample will be assigned the class where it falls in. Similar with KNN, we need to find the optimal parameters(margin and boundaries).

There are two forms of classifications in SVM: Linear classification and Non-linear classification.[43]

5.2.2 Linear classification

Linear classification indicates that two classes can be divided by a "margin hyperplane". It has two situations: Linear separable and Non-linear separable.[43]

Linear separable Let us first see a simple example in which positive samples and negative samples can be separated totally using a straight line or a hyperplane. We call this situation linear separation. See figure (5.5).

In this situation, you can draw so many multiple separating lines or hyperplanes as you want. Figure (5.6) illustrates this situation. All the straight lines in this figure are meaningful since they can separate the classes totally. In fact there is an infinity. Now the question is: which one is the best?

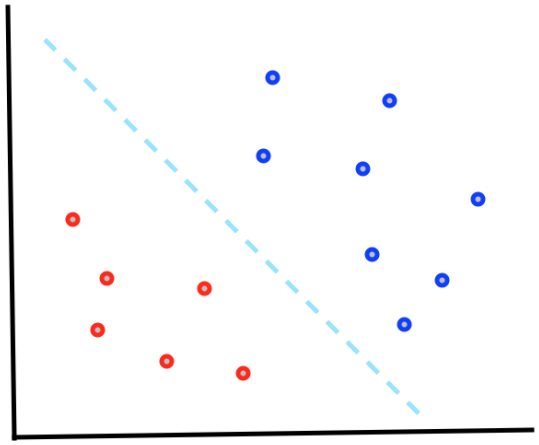

Figure 5.5: In this example red points and blue points can be easily separated using a straight line between them.

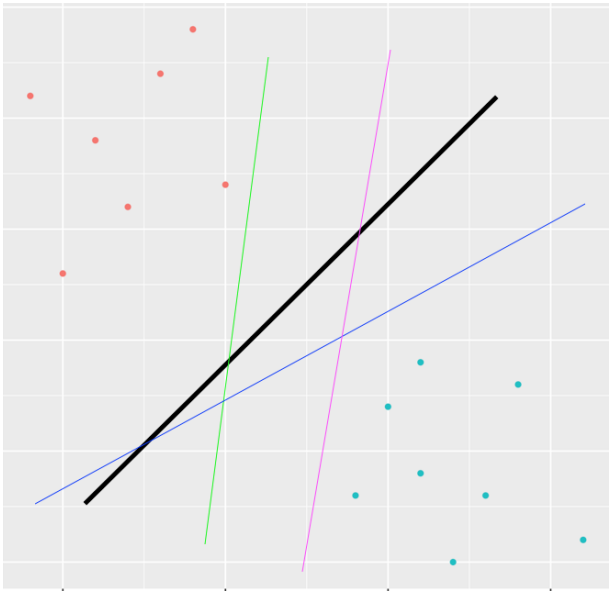

Figure 5.6: Multiple separation lines. More than one straight line can separate the data totally, but only one of them is optimal.

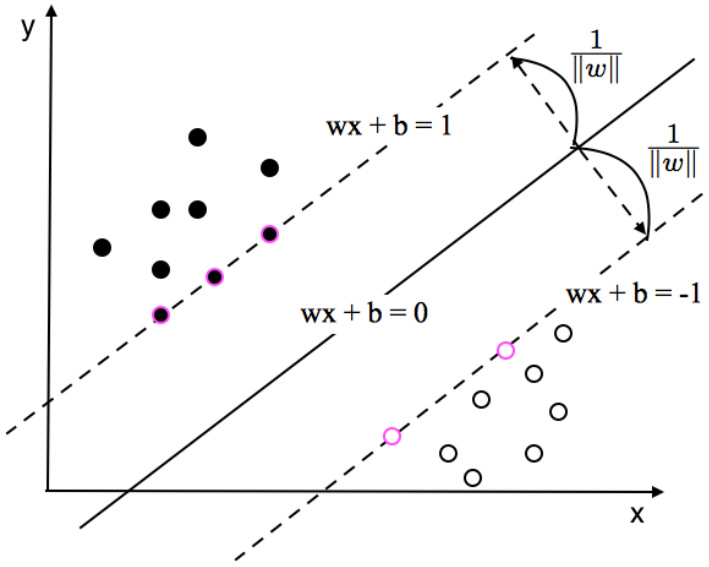

Figure 5.7: In this figure, the two dashed lines are optimal boundaries which satisfy the shown equations respectively, where all points locating on and above the upper boundary belong to one class and all points locating on and below the lower boundary belong to another class. The solid line is the optimal hyperplane which separates two classes. X represents the data point in the feature space and w is its vector to that line. There will always be some points (from both sides) that are closest to the optimal boundaries (the red points in the figure), these points are called support vectors[43]. The straight distance between support vectors and the hyperplane is $\frac{1}{\|w\|}$, which can be calculated by mathematical methods.

Here we define the strategy of finding the best straight line or hyperplane: we first need to find the maximum margin whose vertical distance from its boundaries to the nearest data point is minimized. In another word, the vertical distance from its hyperplane(In geometry, hyperplane locates in the center of margin and has equal distance to both parallel boundaries) to the nearest data point is maximized. This kind of hyperplane is considered to be optimal. Figure (5.7) gives you an overview.

It is easy to see that the max-margin hyperplane depends only on those support vectors. This means that unlike other classification algorithms, this classifier does not rely on other points in the dataset.

Although the above separation is great, the data in practice is not so perfect as we expect and it can not be separated linearly. So, what can we do when we deal with these kinds of datasets?

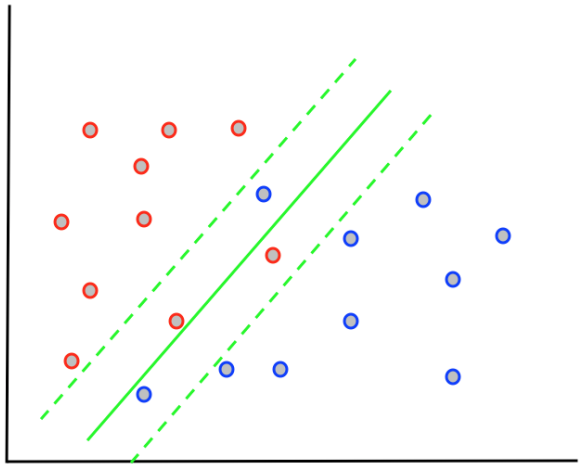

Figure 5.8: Non-linear separation. You can see that several points are misclassified. One blue point locates on the red side and one red point locates on the blue side.

Non-linear separable Another situation is non-linear separation in which it is feasible for some points to be misclassified because of the complexity of data. As you can see in 5.8

The hyperplane in this situation can be represented by the following equation[43]:

$$\left[\frac{1}{n}\sum_{i=1}^{m} max(0, 1 - y_i(\vec{w} \cdot \vec{x_i} + b))\right] + \lambda\|\vec{w}\|^2 \tag{5.2}$$

where λ is a parameter that controls the hyperplane(Source: "`https://en.wikipedia.org/wiki/Support-vector_machine.`").

5.2.3 Non-linear classification

Notice that the separation boundaries which are produced by the previous process in linear calssification do not always work because of the complexity in real life data(See figure 5.9). A way to "*create non-linear classifiers by applying the kernel trick (originally proposed by Aizerman et al.[43]) to maximum-margin hyperplanes was suggested by Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik*".[44] As mentioned in the principle of SVM part, we mapp the features of samples into a N-dimensional space and use the number of features as the dimension of space. For the data points that can not be linearly separated(5.9), the strategy of kernel function is to increase the dimension of space, which can greatly reduce the complexity of the problem(as you can see in figure 5.10).

Some common kernels functions[45] include:

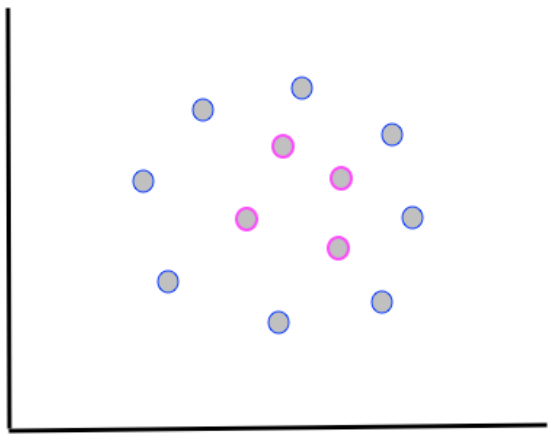

Figure 5.9: Non-linearly separable data. The points in this set can not be separated by a straight line.

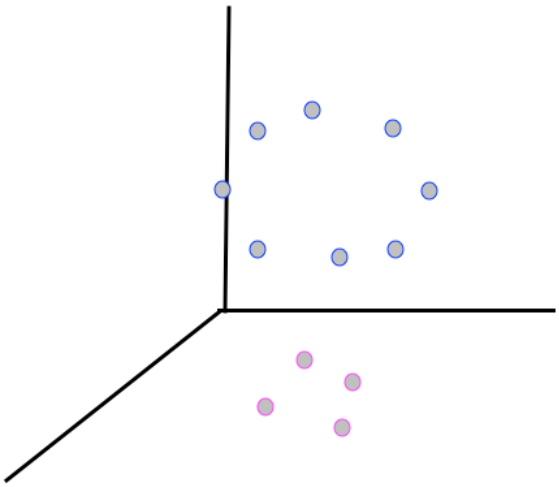

Figure 5.10: The transformed space. In this space, the number of space dimensions increased from 2 to 3, which greatly simplifies the complexity of the problem, because the data can be separated by a linear hyperplane in the new space.

Linearl:

$$K\left(X_i, X_j\right) = {X_i}^T X_j \tag{5.3}$$

Polynomial:

$$K\left(X_i, X_j\right) = \left(\gamma {X_i}^T X_j + r\right)^d, \gamma > 0 \tag{5.4}$$

Radial basis function(RBF):

$$K\left(X_i, X_j\right) = \exp\left(-\gamma \|X_i - X_j\|^2\right), \gamma > 0 \tag{5.5}$$

Sigmoid:

$$K\left(X_i, X_j\right) = \tanh\left(\gamma {X_i}^T X_j + r\right) \tag{5.6}$$

The first is used for linear separation and the last three for non-linear separation. Furthermore, RBF has been proven to be most effective among these functions and has been applied most frequently in practice. We will test all these three functions and give their corresponding performance in the following part.

5.2.4 γ and C

Parameter γ In equation 5.5, parameter γ controls the separation boundaries. Intuitively, it decide the distance between the samples and the boundaries, Low γ leads to long distance and high γ can lead to short distance. If the γ is too high, the kernel function will shrink and will not recapitulate the data totally, and thus will cause a risk of underfitting. On the other hand, if the γ is too low, the kernel function will extend and more data points will be included in the margin, so that it will work like a linear separation, and will obviously have a hazard of overfitting.

Parameter C Another important factor in kernel functions is C, which is the number of misclassified data points and can be seen as "*the penalty of making an error*".[46] The increasing of C indicates that the margin is becoming bigger and bigger, so that we penalize more and more points. In contrast, the decreasing of C means that the margin is getting smaller and smaller, and we are penalizing less points. We don't want to penalize too many points in order to have more data points to train and get an accurate model, and in the meanwhile we also want to get a big enough margin to generalize the dataset as much as possible. So it's a tradeoff between having more data points to train and generalizing the dataset as much as possible.

Both parameter λ and parameter C are pivotal to SVM. Therefore we need to find the optimal γ and C in order to exert the best performance of this algorithm.

5.2.5 Applications

As one of the most popular ML algorithms, SVM can be used in various fields:

1) SVM can be used to solve various real world problems.
2) SVM are useful for text categorization and image identification.
3) Hand-written characters can also be recognized through SVM[47].

	A
1	proteincode,feature1,feature2,feature3,feature4
2	1kcm,0.179,4,14,-3
3	1tgx,0.245,5,19,9
4	3tgo,0.102,3,10,-5
5	1eci,0.104,1,5,9
6	4kvl,0.129,3,25,-2
7	1jza,0.159,1,12,1
8	1dab,0.083,4,16,0
9	2qog,0.194,2,21,11
10	1dkc,0.08,1,6,4
11	2a1l,0.162,2,18,0
12	3n5a,0.11,2,20,14
13	1hxi,0.221,5,21,-6
14	3biw,0.097,2,14,-14
15	1as5,0.25,2,7,4
16	3g7j,0.129,0,12,-15
17	1co6,0.098,1,5,4
18	1ozy,0.165,4,14,5
19	3hhm,0.127,3,26,-2
20	2whx,0.118,7,25,1

Figure 6.1: The new dataset on which machine learning algorithms are based. This set contains four protein features and a part of data.

6 Feature analysis

Normally when you implement a machine learning algorithm on data, several procedures should be followed.

1) Data collection.
2) Data transformation.
3) Algorithm selection.
4) Feature analysis and extraction.
5) Data preprocessing(noise elimination and data normalization).
6) Implementation(of machine learning algorithm).
7) Result analysis and verification.

The original dataset has been provided by Edvin Fuglebakk, so we don't need to collect data. We have finished step 2 in chapter 4 and step 3 in chapter 5. In this chapter, we will carry out features analysis and extraction. In chapter 7, we will go through the data preprocessing as well as Implementation, and carry on the result verification in chapter 8.

First of all, let's see the new dataset we have created in chapter 4(figure 6.1). There are 5 columns which represent different implications: first column refers the protein code which is consistent with protein databases such as OPM[48] and PDB,[49] the remaining four columns correspond to the four new protein features we have created in chapter 4. We extracted 1000 peripheral proteins and 1000 reference proteins totally from the original dataset and stored them in CSV files.

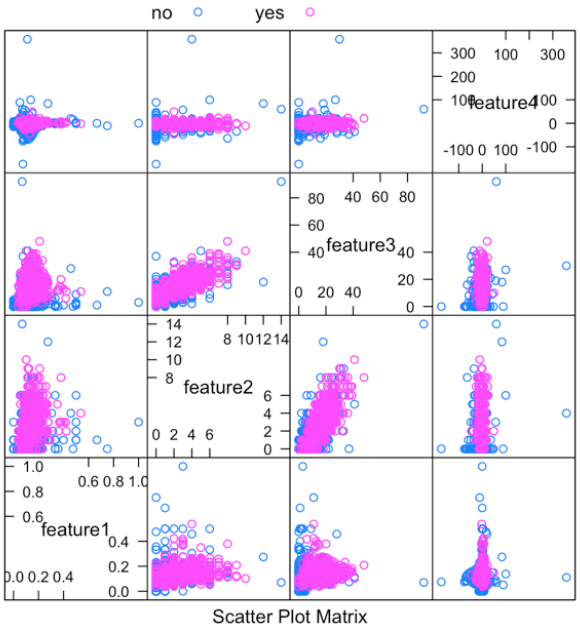

Figure 6.2: The scatter plot matrix. This matrix shows the correlation between every two features.

In this chapter, we will analyze these features and select those that are representative and can improve the model instead of keeping all features no matter if they are useful or not.

6.1 Why

The primary purpose of feature analysis is to do the feature selection. The following reasons explain why we need a feature selection:

1) More features might contain more noise in the data, which will influence the performance of model. Reducing irrelevant features can improve the performance and avoid overfitting.

2) The simplified data structure can reduce the running time so that you can try more complicated algorithms.

3) The simplified data structure can also make it easy for people to understand. A dataset with 5 features is surely more succinct than with 10.

6.2 Features visualization

Before we do the selection, it is necessary to visualize the features to get more insights of them. In R a package ggplot2 can be used to plot features in different forms. Such as:

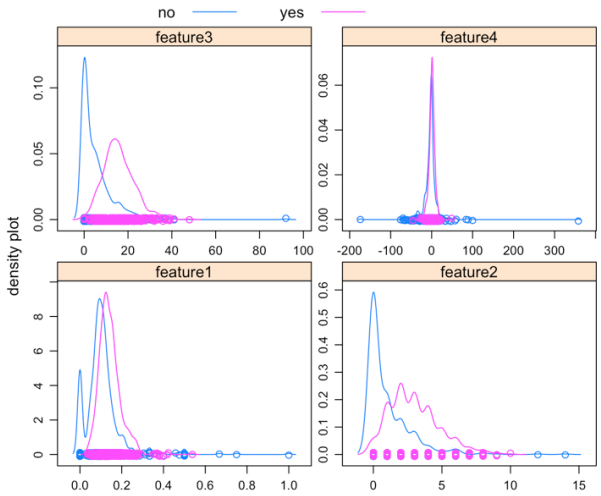

Figure 6.3: The density plot.

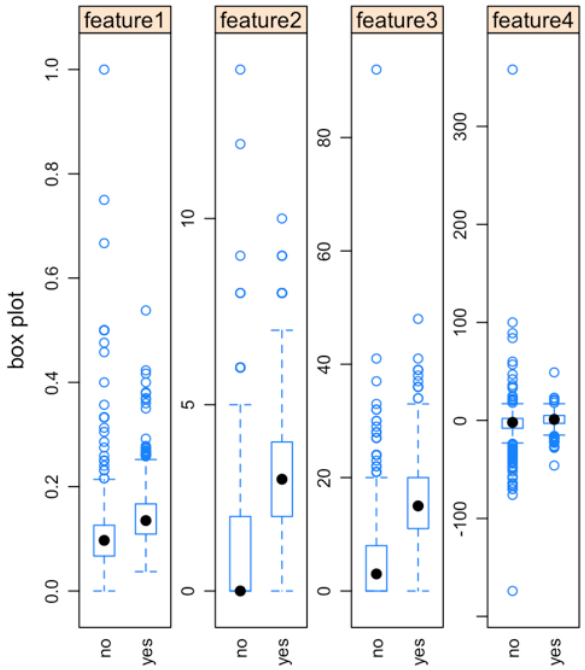

Figure 6.4: The box plot.

scatter plot, density plot and box plot.

In scatter plot(figure 6.2), the correlation between every two features is shown in every cell. For example, the scatter plot on leftmost in first row implies the correlation of feature 1 and feature 4. It is not hard to see that most of samples(both positive and negative) gather together disorderly, which means feature 4 does not have obvious change along with increment of values in feature 1, and anyone of them don't actually influence another. Same interpretation in other cells except cell(2, 2) that shows a special correlation between feature 2 and 3, where samples show an increasing trend(if you put an approximate imaginary straight line in the middle), which suggests that the relationship between them is proportional.

In contrast to the scatter plot matrix that shows the correlation between features. density plot in figure 6.3 and box plot in figure 6.4 indicate the properties within every feature. In figure 6.3, The blue parabola visualises the distribution of negative samples over a continuous interval and the red one visualises the distribution of positive samples. We take feature 3 as an example, notes that the directrix for blue parabola is far away from the one for red parabola, which means the main distribution interval of negative samples has an offset to the main distribution interval of positive samples even through there is a small overlapping interval in feature 3. The bigger offset between the two directrix, the more representative the feature is and the better model it forms. For feature 1 and 2, there also exist offset from positive samples to negative more or less while in feature 4, their main distribution intervals are almost overlapping. Figure 6.4 has a more clear display, the two rectangles represent the main distribution intervals of negative and positive samples within every feature. Less common area tell us they have less overlapping distribution interval, which means this feature is more representative. In above figure, we can easier to see feature 3 is the best and feature 4 is the worst.

All the analysis above only depends on the visualization of features. We don't have any credible evidences to support which features we should select and which one we should discard. In the next paragraph we will carry out data analysis for the selection to see if the result is consistent with the visualization.

6.3 Features selection

We first filter features by computing specific scores using statistical methods.

Table 1: Feature analysis methods based on different type of features.
(Source: 'Introduction to Feature Selection methods with an example')[50]

Features\Response	Continuous	Categorical
Continuous	Pearson's Correlation	LDA
Categorical	Anova	Chi-Square

```
Call:
lda(output ~ ., data = trainset, prior = c(1, 1)/2, CV = FALSE)

Prior probabilities of groups:
 no yes
0.5 0.5

Group means:
    feature1 feature2  feature3   feature4
no  0.105800 1.026667  5.363333 -3.5000000
yes 0.144755 2.975000 15.896667  0.5466667

Coefficients of linear discriminants:
                 LD1
feature1 3.440223199
feature2 0.043445787
feature3 0.115431321
feature4 0.005119763
```

Figure 6.5: LDA output. This method is used to find a linear combination of variables that can separate different classes.

Filter Methods There are many methods to achieve it. Table 1 can guide your choice. The methods in this table try to find the correlation or association between features. Since all our features are continuous and outputs are categorical, therefore LDA(linear discriminant analysis) is the matching one. The R output based on this method is displayed as follows(figure 6.5):

The first part in this figure tells us that the sizes of positive samples and negative samples are equal. Second part demonstrates the mean values of these two categories in every feature. It is necessary to mention that the difference of mean values between the two groups in feature 3 is obvious, which means that this feature will have an apparent impact on the model comparing with other features. In the last part, coefficient indicates the relationship between every feature and its corresponding output, and represents the slope of the linear equation.

$$y = coefficient * feature$$

Another way in which we evaluate features is to select those features that have the strongest relationship with the output values by calculating the feature scores. The scikit-learn library in Python provides the SelectKBest class that can be used to do that. Below(figure 6.6) are functions and result, where f_classif is one of internal functions.

By using score function fit(X, Y), we got the scores of four features. The feature with highest score is the best and has the strongest relationship with output.

```
91
92      test = SelectKBest(score_func=f_classif, k=4)
93      fit = test.fit(X, Y)
94      print fit.scores_
95
[ 54.37105803 284.69347632 466.58165788   9.27433091]
[Finished in 0.8s]
```

Figure 6.6: The Python code for calculating feature scores. SelectKBest is a class. fit(X, Y) is the score function. X and Y indicate input and output value respectively.

Wrapper Methods Again filter methods can show us comparison of features. It seems like we should drop the last feature since it got the lowest score. However, a tricky issue is how we should define the threshold with which we filter features? Do we need to filter the first feature as well? We require a robust and powerful method to decide that.

A direct method is wrapper[50] that relies on the performance of machine learning models. We train a model by adding features one by one to see if it helps to improve the performance. If it does, then we keep it; otherwise, we drop it. We will implement this method after we introduce the main functions of KNN and SVM in the next chapter.

Some common methods are: forward feature selection, backward feature elimination and recursive feature elimination etc.[50]

7 Algorithm implementation

In this chapter we will introduce the main functions of KNN and SVM in R and display results by adjusting different parameters.

7.1 KNN function

$$knn\_model \leftarrow knn(\text{Trainset.scaled, Testset.scaled, labels\_train,}$$
$$k = 6, \text{prob} = \text{TRUE},$$
$$\text{algorithm} = c(\text{“kd\_tree”, “cover\_tre”, “brute”}))$$

Tthe KNN function used in the project is shown above. Where:

• Trainset.scaled: iscaled training set;

• Testset.scaled: scaled testset;

• labels_train: label values of training set;

• k: number of neighbours we take. See chapter 5.1.3. The K value here is just a default value;

• prob: If it is true, the algorithm will also calculate the voting ratio of winning class;

• algorithm: is the strategy to search nearest neighbors.

7.2 SVM function

$$SVM\_model \leftarrow svm(\text{trainset.scale\$label .}, \text{data} = \text{trainset.scale}, \text{scale} = \text{TRUE},$$
$$tolerance = 0.001, epsilon = 0.1,$$
$$cross = 0, nu = 0.5, \text{probability} = \text{FALSE},$$
$$\text{na.action} = \text{na.omit}, degree = 3, coef0 = 0,$$
$$gamma = 1/4, cost = 1, \text{class.weights} = \text{NULL},$$
$$\text{method} = \text{”C-classification”}, \text{kernel} = \text{”sigmoid”})$$

Above shows the SVM function[51] used in the project. Where:

• trainset: the dataset we used to train the model.

• V8: the output values in the training set.

• scale: indicating if the data should be scaled.

• tolerance: the termination point of iteration for calculating the error rate in order to find the optimal solution to SVM. This iteration would never stop without this setting because of the floating point type of errors.

• epsilon: the model goes through a process of gradual optimization to be good enough by calculating the loss using the loss function in the algorithm, parameter epsilon in that function is used to stop the optimization process.

• cross: k-fold cross validation that is used to verify the model. We will talk about it in details in the following chapter.

• nu: a parameter for nu-classification, nu-regression, and one-classification. Our problem is c-classification.

• probability: indicating whether the model will calculate the class probability of new samples or not.

• na.action: specifying what the function will do if there are NA values in the training set. "na.omit" means that it will reject samples with missing values, while "na.fail" means that it will cause an error.

• gamma: parameter that is used to control the margin for all kernel function except linear. See the section 5.2.4.

• cost: specifying the number of misclassified points. Also see the section 5.2.4.

• degree and coef0: parameters for polynomial and sigmoid kernel functions. We will not talk about them in details since they didn't performance very well for our dataset

• class.weights: indicating whether the training set is balanced or not. If it is, this parameter is NULL; otherwise, the ratio of classes should be supplied.

• method: indicating the type of task(classification or regression) depending on the output values.

• kernel: kernel function.

Note that in SVM function, we used default values for gamma and cost. Recall chapter 5 where we have declared that optimal gamma and cost need to be found in order to avoid underfitting and overfitting, Therefore, a tune function is used to find the optimal parameter gamma and cost, in other words to capture the best performance of model. See the equation below.

$$
\begin{aligned}
tune\_out \leftarrow tune.svm(&\text{x} = \text{trainset.scale[,-train\_ColNum]},\\
&\text{y} = \text{factor(trainset.scale[,train\_ColNum])},\\
&\text{sampling} = \text{"Bootstrapping"},\\
&gamma = 10 \wedge (-3:3), cost = 10 \wedge (0:3),\\
&\text{kernel="radial")}
\end{aligned}
$$

Where:

• x: the variables of training set.

• y: the output of training set.

• sampling: how to sample for the dataset. Some typical methods are cross, Bootstrapping, Bagging and Ensembling.

• gamma: the grid of gamma, the function will find the optimal gamma in this range.

• cost: the grid of cost, the function will find the optimal cost in this range.

• kernel: kernel function.

7.3 Noise elimination

Noise elimination is critical as well if your data contains noise. In machine learning, the quality of data will greatly affect the model performance. For example: noise can influence the distribution of data points, especially the data in the domain area of dataset, and thus will affect the performance of model.

As we know, noise can appear both in input value and output value, where subjective judgments, typing errors, information insufficient, missing values or incomplete values can be common.

Noise elimination is really tough. Different noise requires different techniques. It is hard to clean up all the noise, and is also hard to distinguish between noisy and true exceptions. Despite this, several methods can be proposed to avoid or deal with noise to some extent.

1) All the label values and feature values should be acquired through scientific ways or theories instead of subjective assumptions.

2) Training models like Regression or Bayes with normal data to predict label values can be an approach.

3) Copying values from a similar normal sample in the dataset to fill the missing values is also possible.

4) For feature noise, we may replace it with a specific value such as mean or median, or according to a voting mechanism based on the type of features.

5) We can also implement algorithms that are tolerant to noise. Such as deep neural network.[52]

7.4 Feature scaling

Feature scaling is a method that is used to scale the features and is usually considered as a data preprocessing step. It is also called data normalization.[53]

Some machine learning algorithms would not work properly without normalization because of the diversity of features. For example, some classifiers calculate the distance between two data points through a kernel function. If the range of one feature is extremely bigger than others, then the distance will be dominated by this feature. Therefore, all features should be normalized ahead of time so that each feature contributes proportionally to the model.

We use min-max normalization[53] to scale the data. See the equation 7.1

$$\acute{x} = \frac{x - min(x)}{max(x) - min(x)} \tag{7.1}$$

7.5 Balanced dataset

Balanced dataset refers to a dataset in which the classes are distributed equally. This is an unignored factor which can influence the final result. Imbalanced dataset might produce

a bias classifier, which means the result will be dominated by the majority class. For instance, you have 200 samples in a dataset in which 150 samples belong to class A and others belong to class B. The ratio of these two classes is 3:1. Thus even though you got a excellent performance from this model, this performance may only reflects the dominant class distribution instead of entire data.

However in real-world data, imbalanced data is inevitable, such as medical data from patients, or real-time data from production. They are either expensive or difficult to collect. How can we implement the machine learning in this situation and meanwhile ensure the reliability of the model? So in case you have an imbalanced dataset, several measures can be attempted. For example:

1) Try as many different algorithms as possible. Some of them might perform better on imbalanced dataset.

2) Try to collect more data. A larger dataset can make the imbalance less obvious.

3) Use other performance analysis. Such as sensitivity-precision curve.[54]

4) Resample dataset. This may help you to get more samples in minority class or get rid of samples from majority class.[55]

We prepared 2000 protein samples(1000 peripheral membrane proteins and 1000 reference proteins) and two algorithms(KNN and SVM). We will run both algorithms with different data sizes and parameters. In order to guarantee the accuracy, we will use balanced dataset.

7.6 Performance formula

In this chapter we will get to the core part of this article: the implementation of machine learning based on dataset we have created and preprocessed.

As we have said before we divide the dataset into training set and test set. The training set is used to train a model and test set to test performance. The proportion of training set and test set is adjustable. Popular options can be 0.7/0.3, 0.8/0.2 or 0.9/0.1. If you have a big dataset, then you can use more data as test set; otherwise, less samples can also be feasible.

We display performance by calculating accuracy, sensitivity, specificity and precision. They can be achieved via the following formula:

$$Accuracy = (TP + TN)/(TP + FP + TN + FN) \tag{7.2}$$

$$Sensitivity = TP/(TP + FN) \tag{7.3}$$

$$Specificity = TN/(TN + FP) \tag{7.4}$$

$$Precision = TP/(TP + FP) \tag{7.5}$$

where:

True positive(TP): is the number of positive samples that were classified into positive in test set.

False negative(FN): is the number of positive samples that were classified into negative in test set.

True negative(TN): is the number of negative samples that were classified into negative in test set.

False positive(FP): is the number of negative samples that were classified into positive in test set.

Thus in classification problems:

Accuracy: is the ratio of correctly classified samples by the model over the whole test set.

Sensitivity: is the ratio of correctly classified samples within positive class over all the positive samples in the test set.

Specificity: is the ratio of correctly classified samples within negative class over all the negative samples in the test set.

Precision: is the ratio of correctly classified samples within positive class over all the classified positive samples.

The matrix in Table 2 can help to understand.

Table 2: Confusion matrix.

Predicted\Test set	Pos	Neg
Pos	TP	FP
Neg	FN	TN

7.7 Wrapper method

As we promised above, we will implement the common wrapper method based on a model for feature selection. By adding one feature each time, we calculate the model's performance. The final results from KNN are illustrated in the Table 3.

In the Table 3, all the results are based on a dataset with 500 proteins(250 peripheral membrane proteins and 250 reference proteins) in which 400 proteins are for training set and 100 proteins for test set. We calculated accuracy, sensitivity, specificity and precision by adding features one by one. There is an obvious upward trend on these performances along with the accumulation of features except feature 4. The including of feature 4 leads

Table 3: Wrapper method based on a KNN model. See section 4.2 - 4.5 for feature description.

Performance	Accuracy	Sensitivity	Specificity	Precision	k value
Feature 1	0.68	0.72	0.64	0.67	5
Feature 1, 2	0.74	0.76	0.72	0.73	10
Feature 1, 2, 3	0.85	0.82	0.84	0.84	10
Feature 1, 2, 3, 4	0.82	0.82	0.82	0.82	16

to the decay of model's performance in overall. This conclusion is consistent with the feature analysis in chapter 6. The K values in last column are optimal K values for every implementation. In the following analysis, we will exclude feature 4(net charge of amino acids exposed on the surface) and focus on rest of them.

7.8 KNN performance

This time we will extend the dataset size to see any changes on performance. We also use a different ratio of training set and test set, the test set occupies 1/10 of whole set, and this test set is an independent set and is not used to train the model. So that we can keep more data for both training set and validation set which we will talk about in the result verification section. See the performances in figure 7.1.

7.8.1 Alternative dataset

In figure 7.1 four performance curves are displayed for KNN. Along with the growing of dataset, they would decline. What caused this diversification? A possible reason is: In order to collect more data to train a reliable model as well as have a balanced dataset, we collected many proteins from a alternative dataset as a part of reference proteins. This dataset was collected with different criterions and has more exceptions, which means that the features values we transformed from this alternative dataset are not so representative for reference proteins as what we did from the original reference dataset.

Therefore, it shows an overall decline with the extending of dataset. their values of accuracy, sensitivity, specificity and precision at size 2000 are 0.85, 0.89, 0.80 and 0.82 correspondingly.

7.9 SVM performance

We first assay all the kernel functions except Linearl. The comparison based on 500 proteins is shown in Table 4. Obviously, RBF has the best performances among them.

The SVM's performance based on RBF and the extending of data is shown in figure 7.2. Each plot contains the comparison of 'before tune' and 'after tune'.

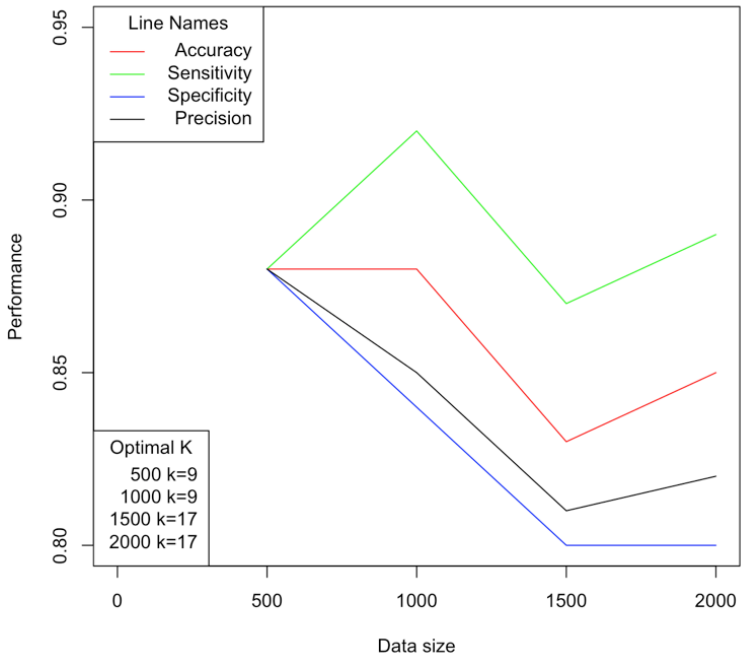

Figure 7.1: KNN performance with 3 features.

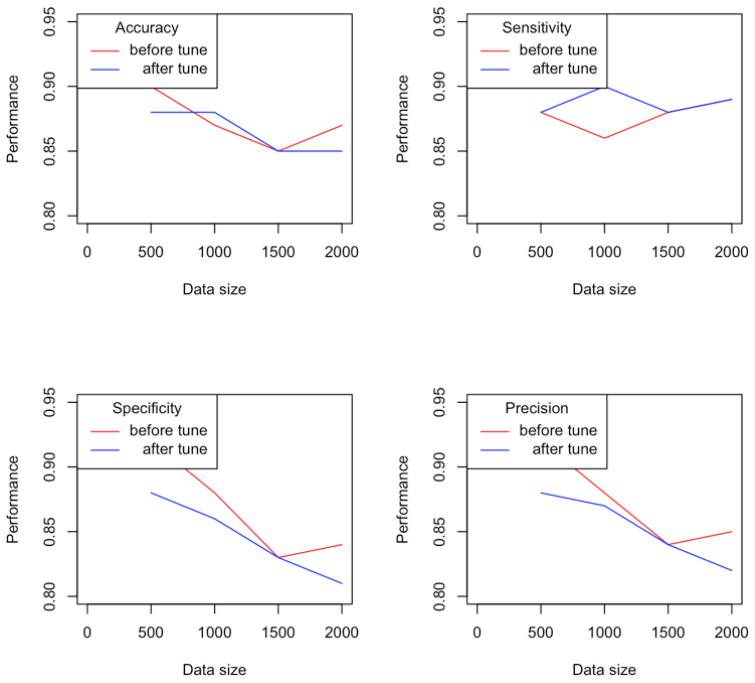

Figure 7.2: SVM performance with 3 features.

Table 4: The comparison of kernel functions

Kernel	Accuracy	Sensitivity	Specificity	Precision
Polynomial	0.82	0.96	0.68	0.75
RBF	0.88	0.88	0.88	0.88
Sigmoid	0.7	0.8	0.6	0.67

Note that these four performance plots have downward trends after tuning. This also validates the characteristic of our dataset which we have stated in previous paragraph. The values of accuracy, sensitivity, specificity and precision after tuning at size 2000 are 0.85, 0.89, 0.81 and 0.82.

The confusion matrix(figure 7.3) offers a panoramic view of the SVM performance at size 2000 where 200 proteins are used as a independent test set.

7.10 ROC-AUC curve

Another method that can be used to evaluate the performance of model is ROC-AUC(receiver operating characteristic-area under the curve) curves.[56]

In statistics, ROC curves show the trend of model performance with different thresholds for classifying the classes. It reflects the mutual relationship between true positive rates(TPR) and false positive rates (FPR) in the model, where true positive rate is actually the sensitivity and false positive rates can be acquired by equation: TN/(TN+FP). You can see the curve in figure 7.4.

AUC can be used to get the whole performance of classifier. By running statistical function in R, the area under this curve is 0.86. The higher, the better.

With the development of technology, ROC-AUC analysis has been applied in medicine, agriculture, commerce and bioinformatics etc, and is widely used in machine learning and data analysis.

However, it is worth to mention that if you have unbalanced classes(unequal number of positive and negative samples), ROC curves can be misleading and should not be used, a sensitivity-precision curve can be more reasonable. An example is given in figure 7.5.

```
Confusion Matrix and Statistics

          Reference
Prediction no yes
       no  81  11
       yes 19  89

               Accuracy : 0.85
                 95% CI : (0.7928, 0.8965)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7
 Mcnemar's Test P-Value : 0.2012

            Sensitivity : 0.8100
            Specificity : 0.8900
         Pos Pred Value : 0.8804
         Neg Pred Value : 0.8241
             Prevalence : 0.5000
         Detection Rate : 0.4050
   Detection Prevalence : 0.4600
      Balanced Accuracy : 0.8500

       'Positive' Class : no
```

Figure 7.3: Confusion Matrix of the SVM output.

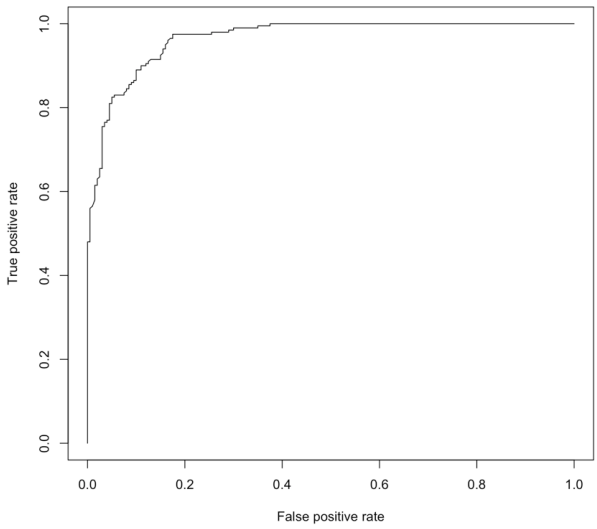

Figure 7.4: ROC curve.

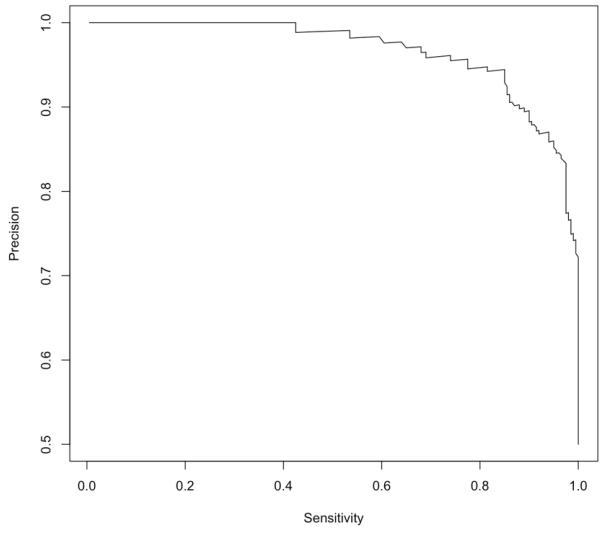

Figure 7.5: Sensitivity-Precision plot.

8 Result verification

Having obtained the results from both KNN and SVM. We are now going to use a method to verify them.

8.1 Cross-Validation

Since we obtained good performances, we now use a method to verify this result. We still use SVM performance as our verification object. A popular verification method is Cross-validation.

Cross-validation is a model validation technique to evaluate the model we get when we apply it to a totally different dataset. Intuitively, it is used to estimate the accurateness of the model. For a machine learning prediction problem, we prepare a training set and a test set. We aim at testing this model by cross-validation to avoid bias.

8.2 Steps

The Cross-Validation can be divided into several steps.

1) Partition the training set into K groups with equal sizes. Each group has N/K samples(we assume the total size of training set is N). The K value is normally 10.

2) We define these groups as group_1, group_2, ..., group_K, then choose one of them as a validation set one by one starting from group_1 and rest groups as training set.

3) Every time when we get a new training set and validation set, we will train a model using this training set and test on the validation set.

4) Keep iterating K times, we get K best performances from those models.

5) We average these performances and get ultimate result.

Figure 8.1 can give you an overview about how we divide the data for Cross-Validation.

8.3 Cross-Validation on KNN

We choose the function that is used to find the optimal K to implement the cross-validation. The function is illustrated below.

$$
\begin{aligned}
&model \leftarrow train(labels\_train\ .,\\
&\qquad \text{method} = \text{"knn"},\\
&\qquad \text{tuneGrid} = \text{expand.grid}(k = 1:50),\\
&\qquad \text{trControl} = trainControl(\text{method} = \text{"repeatedcv"}, number = 10, repeats = 15),\\
&\qquad \text{metric} = \text{"Accuracy"},\\
&\qquad \text{data} = \text{Trainset.scaled})
\end{aligned}
$$

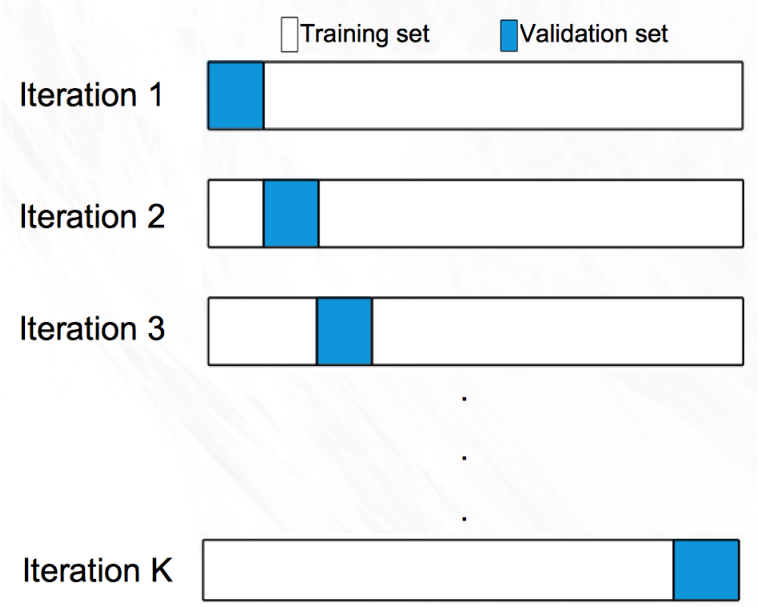

Figure 8.1: Cross-Validation.

Where:

• labels_train: is the label values of dataset which is used for cross-validation.

• method: is the machine learning algorithm.

• tuneGrid: is the tuneGrid for K.

• trControl: is the train control. Where the method is cross-validation, we divide data into 10 groups to run the cross-validation and repeat the operation15 times.

• metric: indicating we will calculate the accuracy performance of cross-validation.

• data: is scaled dataset.

See the output depending on 1800 training samples and 3 features in figure 8.2, where the accuracies for 10-fold cross-validation is listed and the average accuracy is 0.845.

8.4 Cross-Validation on SVM

Remember that in the SVM function there is a parameter "cross" which is defined to implement the cross-validation. The outcome based on 1800 training samples and 3 features is illustrated in figure 8.3. The average accuracy is 0.848.

We notice that the performances from cross-validation for both KNN and SVM are slightly lower than those from test set. This is normal. Cross-validation almost always lead to lower estimated errors on an unseen test set. The reason is that the model is not so generalized. It is specalized to the structure of the training set.

```
> confusionMatrix(model)
Cross-Validated (10 fold, repeated 15 times) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction   no  yes
       no  40.3  5.8
       yes  9.7 44.2

 Accuracy (average) : 0.8449
```

Figure 8.2: KNN Cross-Validation output.

```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  10
      gamma:  1

Number of Support Vectors:  653

 ( 321 332 )


Number of Classes:  2

Levels:
 no yes

10-fold cross-validation on training data:

Total Accuracy: 84.83333
Single Accuracies:
 88.33333 87.22222 77.77778 85 84.44444 89.44444 83.33333 82.22222 84.44444 86.11111
```

Figure 8.3: SVM Cross-Validation result.

The small deviation between test set and validation set indicates that this model is reliable. Big deviation means the model you have trained is overfitting. However if your cross-validation performance is higher than test performance, the reason can be various, such as data size or splitting ratio.

9 Conclusion and future work

9.1 Conclusion

Till now we have implemented these two algorithms based on the data we have collected in chapter 4. By visualizing the results, we can summarize that:

1) After the feature visualization and feature selection, we believe that feature1, feature2 and feature3 we created in chapter 4 are positive for building the model, while feature4 is harmful to the model.

2) KNN and SVM have similar performance track on the data. Both of their performances declined along with the extending of data because of the characteristic of dataset(As we have mentioned in 7.8, the data from alternative dataset contains more exceptions).

3) They also have approximative performances in accuracy, sensitivity, specificity and precision, since both are typical for classification problems.

4) More specifically, SVM is slightly higher in specificity and precision, but KNN is better in sensitivity, and they have equal accuracy.

5) Their cross-validation accuracies are also proximate and a little bit lower than their models.

6) Another need to be mentioned is that RBF gave the best performance among those kernel functions in SVM based on our data.

7) After verifying the model using cross-validation, we see that the models we have trained are accurate. Because validation accuracy is close to test accuracy.

9.2 Future work

So far we have completed the necessary steps for a machine learning algorithm. From the data collecting in the beginning to feature selection and visualization, from the choice of algorithm to implementation, and from result display to verification, every step refers to different methods. It is unrealistic to attempt every method, but we used the most suitable one for every step.

We will stop this project here, However we still have a lot of work to do in the future even through we got good models and performances based on our data. We want to further improve the model. Thus the following work can be done to improve the model.

First of all, try to collect more proteins. The original dataset contains only 1000 peripheral membrane proteins and over 2000 reference proteins. So next we can collect more peripheral proteins from protein databases such as PDB(Protein data bank) and OPM(Orientations of Proteins in Membranes (OPM) database) to support our model. The more data you have, a more reliable model you will get.

Second, try to use different protein data to test and verify the model, since the model is specalized to the training set. By this way we can test if the model er reliable.

Last, based on a larger dataset, we can try different machine learning algorithms and compare their performance extensively.

List of Figures

2.1 The four different levels of protein structures.(Source: "*The Four Levels of Protein Structure*"[10]) 8

2.2 Integral proteins and peripheral proteins: Integral proteins are fully embedded in the bilayer of membrane, while peripheral proteins only attach on its surface. (Source: "*Integral and peripheral membrane proteins*"[15]) 8

3.1 The figure above shows a classification problem for samples. There are two classes(C1 and C2). The straight line represents a linear boundary(left) and the curve represents a quadratic boundary(right), which are used to define the regions C1 and C2. New observations will be classified into the class C1 or C2 depending on which region they will fall into. Classifiers are not perfect, because they can not classify the points totally, some points are classified wrongly. 11

3.2 There are three different groups in the data, they can be partitioned clearly using unsupervised learning 11

4.1 The original protein structure dataset with features.(Source: "*A model for hydrophobic protrusions on peripheral membrane proteins*"[25]) 13

4.2 Protruding hydrophobes are found on the membrane binding sites of membrane binding domains. The figure shows the convex hull (in blue) of the C_α and C_β atoms of several peripheral membrane binding domains. The C_β atoms of the likely inserted hydrophobe are shown as orange spheres and C_β atoms of experimentally identified membrane-binding residues as gray spheres. 1RLW: C2 domain of human phospholipase A2;[33] 1H6H: PX domain of P40PHOX;[34] 1POA: snake phospholipase A2;[35] 1PTR: C1 domain of protein kinase C delta;[36] 1H0A: Epsin ENTH domain;[37] 1VFY: FYVE domain of yeast vacuolar protein sorting-associated protein 27.[38] (Source: "*A model for hydrophobic protrusions on peripheral membrane proteins*"[25]) 16

4.3 Panel A shows a cartoon representation of the C2 domain of human phospholipase A2 (PDB ID: 1RLW), and panel B show the convex hull for the same protein. Co-insertable protruding hydrophobes are connected by orange lines.(Source: "*A model for hydrophobic protrusions on peripheral membrane proteins*"[25]) 17

5.1 k-Nearest neighbor classification. The new sample will be classified into a positive sample. 19

5.2 The influence of different k to boundaries which separate the two classes. . 20

5.3 The cross-validation accuracy plot with different values of k. In this plot the interval of x-axis(the range of k) is [1 20]. 21

5.4 The resampling results across tuning parameters where we can find the optimal k. 22

5.5 In this example red points and blue points can be easily separated using a straight line between them. 23
5.6 Multiple separation lines. More than one straight line can separate the data totally, but only one of them is optimal. 23
5.7 In this figure, the two dashed lines are optimal boundaries which satisfy the shown equations respectively, where all points locating on and above the upper boundary belong to one class and all points locating on and below the lower boundary belong to another class. The solid line is the optimal hyperplane which separates two classes. X represents the data point in the feature space and w is its vector to that line. There will always be some points (from both sides) that are closest to the optimal boundaries (the red points in the figure), these points are called support vectors[43]. The straight distance between support vectors and the hyperplane is $\frac{1}{\|w\|}$, which can be calculated by mathematical methods. 24
5.8 Non-linear separation. You can see that several points are misclassified. One blue point locates on the red side and one red point locates on the blue side. 25
5.9 Non-linearly separable data. The points in this set can not be separated by a straight line. 26
5.10 The transformed space. In this space, the number of space dimensions increased from 2 to 3, which greatly simplifies the complexity of the problem, because the data can be separated by a linear hyperplane in the new space. 27
6.1 The new dataset on which machine learning algorithms are based. This set contains four protein features and a part of data. 30
6.2 The scatter plot matrix. This matrix shows the correlation between every two features. 31
6.3 The density plot. 32
6.4 The box plot. 32
6.5 LDA output. This method is used to find a linear combination of variables that can separate different classes. 34
6.6 The Python code for calculating feature scores. SelectKBest is a class. fit(X, Y) is the score function. X and Y indicate input and output value respectively. 35
7.1 KNN performance with 3 features. 42
7.2 SVM performance with 3 features. 43
7.3 Confusion Matrix of the SVM output. 45
7.4 ROC curve. 46
7.5 Sensitivity-Precision plot. 46
8.1 Cross-Validation. 48
8.2 KNN Cross-Validation output. 49
8.3 SVM Cross-Validation result. 49

List of Tables

1 Feature analysis methods based on different type of features. 33
2 Confusion matrix. 40
3 Wrapper method based on a KNN model. See section 4.2 - 4.5 for feature description. 41
4 The comparison of kernel functions 44

List of Equations

5.1 Euclidean distance 19
5.2 Equation for computing the (soft-margin) SVM classifier 25
5.3 Linear kernel 28
5.4 Polynomial kernel 28
5.5 RBF kernel 28
5.6 Sigmoid kernel 28
7.0 KNN function 36
7.0 SVM function 36
7.0 tune.svm function 37
7.1 Normalization function 38
7.2 Accuracy function 39
7.3 Sensitivity function 39
7.4 Specificity function 40
7.5 Precision function 40
8.0 train function 47

References

[1] Central Dogma, https://en.wikipedia.org/wiki/Central_dogma_of_molecular_biology.

[2] Transcription, https://en.wikipedia.org/wiki/Transcription_(biology).

[3] Base Complementarity Principle, https://en.wikipedia.org/wiki/Complementarity_(molecular_biology).

[4] RNA Polymerase, https://en.wikipedia.org/wiki/RNA_polymerase.

[5] Translation, https://en.wikipedia.org/wiki/Translation_(biology).

[6] Codon, https://en.wikipedia.org/wiki/DNA_codon_table.

[7] Peptide and Polypeptide, https://en.wikipedia.org/wiki/Peptide.

[8] Antibody, https://en.wikipedia.org/wiki/Antibody.

[9] Enzymes, https://en.wikipedia.org/wiki/Enzyme.

[10] Mckenzie, *"The Four Levels of Protein Structure".* http://kenzie-skye.weebly.com/blog/the-four-levels-of-protein-structure. 9/11/2012

[11] Cell Membrane, https://en.wikipedia.org/wiki/Cell_membrane.

[12] Peripheral Proteins, https://en.wikipedia.org/wiki/Peripheral_membrane_protein.

[13] Phospholipids, https://en.wikipedia.org/wiki/Phospholipid.

[14] Non-covalent Interaction, https://en.wikipedia.org/wiki/Non-covalent_interactions.

[15] Integral and Peripheral Membrane Proteins, https://www.slideshare.net/Medical_PPT_Images/integral-and-peripheral-membrane-proteins-medical-images-for-power-point. SlideShare. Jul 18, 2014

[16] Integral Membrane Proteins, https://en.wikipedia.org/wiki/Integral_membrane_protein.

[17] Johnson JE, Cornell RB (1999). *"Amphitropic proteins: regulation by reversible membrane interactions (review)".* Mol. Membr. Biol. 16 (3): 217-235. doi:10.1080/096876899294544. PMID10503244.

[18] Alenghat, Francis J.; Golan, David E. (2013). *"Membrane Protein Dynamics and Functional Implications in Mammalian Cells"*. Current Topics in Membranes. Current Topics in Membranes.

[19] Krogh, A.; Larsson, B. R.; Von Heijne, G.; Sonnhammer, E. L. L. (2001). *"Predicting transmembrane protein topology with a hidden markov model: Application to complete genomes"*. Journal of Molecular Biology. 305 (3): 567-580. doi:10.1006/jmbi.2000.4315. PMID11152613.

[20] Liszewski, Kathy (1 October 2015). *"Dissecting the Structure of Membrane Proteins"*. Genetic Engineering & Biotechnology News (paper). 35 (17): 1, 14, 16?17. doi:10.1089/gen.35.17.02.

[21] AI, `https://en.wikipedia.org/wiki/Artificial_intelligence.`

[22] Sunil Ray *"Essentials of Machine Learning Algorithms (with Python and R Codes)"*. `https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/.` Analytics Vidhya. September 9, 2017

[23] David Fumo *"Types of Machine Learning Algorithms You Should Know"*. `https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861.` Towards Data Science. Jun 15, 2017

[24] Reinforcement Learning, `https://en.wikipedia.org/wiki/Reinforcement_learning.`

[25] Edvin Fuglebakk, Nathalie Reuter. *"A model for hydrophobic protrusions on peripheral membrane proteins"*. PLOS Computational Biology. July 26, 2018

[26] Alpha and beta carbon, `https://en.wikipedia.org/wiki/Alpha_and_beta_carbon#endnote_Hackhb.`

[27] de Berg et al. (2000), p. 3.

[28] Convex Hull, `https://en.wikipedia.org/wiki/Convex_hull.`

[29] Accessible Surface Area, `https://en.wikipedia.org/wiki/Accessible_surface_area.`

[30] Freesasa. *https://freesasa.github.io/doxygen/.*

[31] Theoretical and Computational Biophysics Group, *"VMD(Visual Molecular Dynamics)". http://www.ks.uiuc.edu/Research/vmd/.*

[32] NACCESS. *http://wolf.bms.umist.ac.uk/naccess/.*

[33] Perisic O, Fong S, Lynch DE, Bycroft M, Williams RL. *"Crystal structure of a calcium-phospholipid binding domain from cytosolic phospholipase A2".* J Biol Chem. 1998 Jan; 273(3):1596-1604.

[34] Bravo J, Karathanassis D, Pacold CM, Pacold ME, Ellson CD, Anderson KE, Butler PJ, Lavenir I, Perisic O, Hawkins PT, Stephens L, Williams RL. *"The crystal structure of the PX domain from p40(phox) bound to phosphatidylinositol 3-phosphate".* Mol Cell. 2001 Oct; 8(4):829?839.

[35] Scott DL, White SP, Otwinowski Z, Yuan W, Gelb MH, Sigler PB. *"Interfacial catalysis: the mechanism of phospholi- pase A2".* Science. 1990 Dec; 250(4987):1541?1546.

[36] Zhang G, Kazanietz MG, Blumberg PM, Hurley JH. *"Crystal structure of the cys2 activator-binding domain of protein kinase C delta in complex with phorbol ester".* Cell. 1995 Jun; 81(6):917-924.

[37] Ford MGJ, Mills IG, Peter BJ, Vallis Y, Praefcke GJK, Evans PR, McMahon HT. *"Curvature of clathrin-coated pits driven by epsin".* Nature. 2002 Sep; 419(6905):361-366.

[38] Misra S, Hurley JH. *"Crystal structure of a phosphatidylinositol 3-phosphate-specific membrane-targeting motif, the FYVE domain of Vps27p".* Cell. 1999 May; 97(5):657-666.

[39] Nitin Bhardwaj, Robert V. Stahelin, Robert E. Langlois, Wonhwa Cho, and Hui Lu. *"Structural Bioinformatics Prediction of Membrane-Binding Proteins".* Journal of molecular biology. Volume 359, Issue 2, 2 June 2006, Pages 486-495

[40] KNN, `https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm`.

[41] K-Fold Cross ValidationNN, `https://en.wikipedia.org/wiki/Cross-validation_(statistics)`.

[42] Lillian Pierson, *"Solving Real-World Problems with Nearest Neighbor Algorithms".* `https://www.dummies.com/programming/big-data/data-science/solving-real-world-problems-with-nearest-neighbor-algorithms/`. Data Science For Dummies.

[43] SVM, `https://en.wikipedia.org/wiki/Support-vector_machine`.

[44] Aizerman, Mark A.; Braverman, Emmanuel M. & Rozonoer, Lev I. (1964). *"Theoretical foundations of the potential function method in pattern recognition learning".* Automation and Remote Control. 25: 821-837.

[45] DataFlair Team, *"Kernel Functions-Introduction to SVM Kernel & Examples".* https://data-flair.training/blogs/svm-kernel-functions/. DataFlair. August 12, 2017

[46] Kailash Awati, *"A gentle introduction to support vector machines using R". https://eight2late.wordpress.com/2017/02/07/a-gentle-introduction-to-support-vector-machines-using-r/.* Eight to Late. February 7, 2017

[47] DeCoste, Dennis (2002). *"Training Invariant Support Vector Machines".* Machine Learning. 46: 161-190.

[48] University of Michigan, *"Orientations of Proteins in Membranes (OPM) database".* https://opm.phar.umich.edu/.

[49] Cambridge Crystallographic Data Centre, Brookhaven National Laboratory, *"Protein Data Bank".* https://www.rcsb.org/.

[50] Saurav Kaushik, Introduction to Feature Selection methods with an example (or how to select the right variables?). https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/. Analytics Vidhya. December 1, 2016

[51] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, Friedrich Leisch, Chih-Chung Chang, Chih-Chen Lin, *"Package 'e1071'(PDF)". https://cran.r-project.org/web/packages/e1071/e1071.pdf.* January 21, 2019

[52] Deep Neural Network, https://en.wikipedia.org/wiki/Deep_learning.

[53] Min-max Normalization, https://en.wikipedia.org/wiki/Feature_scaling.

[54] Precision and recall, https://en.wikipedia.org/wiki/Precision_and_recall.

[55] Jason Brownlee, *"Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset".* https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/. August 19, 2015 in Machine Learning Process

[56] Receiver Operating Characteristic, https://en.wikipedia.org/wiki/Receiver_operating_characteristic.