2018-01-01

Integrated Statistical And Machine Learning Algorithms For Predicting And Classifying G Protein-Coupled Receptors

Fredrick Ayivor
University of Texas at El Paso, fredrickayivor@gmail.com

INTEGRATED STATISTICAL AND MACHINE LEARNING ALGORITHMS FOR PREDICTING AND CLASSIFYING G PROTEIN-COUPLED RECEPTORS

FREDRICK AYIVOR

Master's Program in Computational Science

APPROVED:

Ming-Ying Leung, Chair, Ph.D.

Charlotte M. Vines, Ph.D.

Sangjin Kim , Ph.D.

Thompson Sarkodie-Gyan, Ph.D.

Charles Ambler, Ph.D.
Dean of the Graduate School

INTEGRATED STATISTICAL AND MACHINE LEARNING ALGORITHMS FOR PREDICTING AND CLASSIFYING G PROTEIN-COUPLED RECEPTORS

by

FREDRICK AYIVOR, BSc.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Computational Science Program

THE UNIVERSITY OF TEXAS AT EL PASO

December 2018

Acknowledgements

I would like to express my deep-felt gratitude to my advisor, Dr Ming-Ying leung of the Computational Science Department at The University of Texas at El Paso, for her advice, encouragement, enduring patience and constant support. She was never ceasing in her belief in me, always providing clear explanations when I was lost, constantly driving me with energy when I was tired, and always, *always* giving me her time, in spite of anything else that was going on.

I also wish to thank the other members of my committee, Dr. Charlotte Vines of the Biological Sciences Department, Dr. Sangjin Kim of the Mathematical Sciences Department and Dr. Sarkodie-Gyan Thompson of the Electrical & Computer Engineering all at The University of Texas at El Paso. Their suggestions, comments and additional guidance were invaluable to the completion of this work.

My heartfelt gratitude also goes to the research group members; Jonathan Mohl, Khodeza Begum and Eder Perez for their enormous participation in putting together this work.

Also not forgetting the tremendous support of Grant 5G12MD007592 to the Border Biomedical Research Center from the National Institute on Minority Health and Health Disparities, a component of the National Institutes of Health.

Additionally, I want to thank The University of Texas at El Paso Computational Science Department professors and staff for all their hard work and dedication, providing me the means to complete my degree.

And finally, I must thank my dear wife for putting up with me during the development of this work with continuing, loving support and no complaint. I do not have the words to express all my feelings here, only that I love you, Gladys V. Matiedje Ayivor!

Abstract

G protein-coupled receptors (GPCRs) are transmembrane proteins with important functions in signal transduction and often serve as drug targets. With increasing availability of protein sequence information, there is much interest in computationally predicting GPCRs and classifying them according to their biological roles. Such predictions are cost-efficient and can be valuable guides for designing wet lab experiments to help elucidate signaling pathways and expedite drug discovery. There are existing computational tools of GPCR prediction that involve principal component analysis (PCA), intimate sorting (IS), support vector machine, and random forest (RF) techniques using various sequence derived features. While accuracies of over 90% were reported for their own test datasets, the capabilities in distinguishing GPCRs from transmembrane non-GPCRs had not been measured in any of these tools. Furthermore, no direct comparison of the different approaches has been conducted. In this project, we have established two new GPCR prediction algorithms that integrate combinations of PCA, IS, and RF with the univariate feature selection method that has not been used for GPCR predictions before. The same 1355 sequence features are used uniformly with a test dataset with 2179 positive examples of confirmed GPCRs, and 3781 negative examples including transmembrane non-GPCRs. Overall prediction accuracies are over 90%, and the false positive rates among the transmembrane non-GPCRs are substantially lower than those in existing tools. These results suggest that integrated algorithms perform well with GPCR prediction. We plan to further explore different integrated prediction approaches and apply them to the GPCR classification problem in the future.

Table of Contents

List of Tables

Chapter 1

Introduction

In biology and biochemistry, the term "receptor" is used to denote a class of cellular macromolecules that are specifically or directly involved in chemical signaling between and within cells. As a receptor, it does not only recognize the particular molecules that activate it but also does alter, when recognition occurs, cell function by causing, e.g., a metamorphosis in membrane permeability or an alternation in gene transcription. Therefore, the interaction of a hormone, neurotransmitter, or intracellular messenger with its receptor(s) may lead to a change in cellular activity.

Of the cell surface receptors, the G protein-coupled receptors have occupied a special place of importance. This is because they have the following important features;

1. **Signal transduction function:** G-protein-coupled receptors constitute a unifying signal transduction mechanism. A wide range of neurotransmitters, neuropeptides, polypeptide hormones, inflammatory mediators, and other bioactive molecules transmit their signals to the intracellular environment by specific interaction with a class of receptor that relies upon interaction with GTP-binding protein (G-protein) for activation of intracellular effector systems.

2. **One of the largest protein families:** The number of G-protein-coupled receptors is increasing rapidly. It is estimated that more than a thousand different G protein-coupled receptors exist in mammals, thus constituting one of the largest protein families in nature.

1.1 GPCR characteristics and classification

G protein-coupled receptors have seven-helix-bundled structure embedded in the cell membrane. GPCRs consist of a single polypeptide chain of variable length that traverses the lipid bilayer seven times, forming characteristic transmembrane helices and alternating extracellular and intracellular sequences . The significance of the seven transmembrane motif is not fully clear, but it must be uniquely suited for transmitting a signal from the external surface to the internal surface of the plasma membrane through a ligand-induced conformational change.

GPCRs' integral membrane proteins regulate many important physiological processes including automatic nervous system transmission, sense of smell, and regulation of immune system activity. GPCRs are the largest known class of cell surface receptors (Strader et al. 1994) and represent 1% of the total mammalian genes (Barreda-Gmez et al. 2010). GPCRs can thus play an effective role in secretion, proliferation, chemotaxis, heart rate, and neuro-transmission (Spiegel et al. 1992). Ligands, a heterogeneous set of molecules, consisting of ions, peptides, and proteins are bounded to GPCRs and consequently activate GPCRs by allowing it to bind with G proteins. The binding interactions of these receptors with G proteins can be understood by means of structural bioinformatics (Chou 2005a).

Several classification systems have been used to sort out this superfamily. One of the most frequently used systems called the IUPHAR by Horn et al. (2003) divided GPCRs into six major classes, A, B, C, D, E, and F based on sequence homology and functional similarity and subclasses are assigned using roman number nomenclature (Attwood and Findlay 1994; Kolakowski, 1994). This AF system is designed to cover all GPCRs, in both vertebrates and invertebrates. However, some families in the AF system do not exist in humans (D-E). This is why another classification system designed towards mammalian species has been devised; The GRAFS system clusters GPCRs in five main families that we term glutamate (G), rhodopsin (R), adhesion (A), frizzled/taste2 (F), and secretin (S). (Fredriksson and Lagerstrom 2003).

The **GRAFS system (Vertebrate GPCRs)**

- Glutamate
- Rhodopsin-like
- Adhesion
- Frizzled/Taste2
- Secretin

The **IUPHAR system (Vertebrate an Invertebrate)**

- Class A (Rhodopsin-like)
- Class B (Secretin-like)
- Class C (Metabotrophic glutamate)
- Class D (fungal mating pheromone receptors)
- Class E (cAMP receptors)
- Class F (Frizzled/Smoothened)

Horn et al. (2003) divided GPCRs into six major classes based on sequence homology and functional similarity which is known as the IUPHAR system above.

The rhodopsin-like receptors, known as class A, comprise 80% of all the GPCRs and can transduce a range of stimuli including peptide hormone, light, nucleotides, and chemokines (Bryson-Richardson et al. 2004). The human non-olfactory receptors of rhodopsin-like class bind peptides and biogenic amines (Fridmanis et al. 2006).

Class B secretin-like, second family of GPCRs, contains receptors like glucagon, glucagon-like peptide 1(GLP1), calcitonin, vasoactive intestine peptide, growth hormone-releasing factor, parathyroid hormone, and pituitary cyclase-activating polypeptide (PACAP) (Liu

et al. 1999). Both class A and class B GPCRs have no clear sequence homology but have same functions, identical transmembrane topology, and common extracellular loop 1 and 2 in transmembrane domain.

Class C, metabotropic glutamate receptors (mGluRs) have been implicated for neuronal functions (Dolen and Bear 2008) and play a central role in modulation of nociperception (Hu et al. 2007).

Class D, pheromone receptors are a small class of GPCRs used for chemical communication by the organisms and play a role in controlling interactions between individuals of a single species (Martini et al. 2001).

Class E, cAMP receptors contribute towards chemotactic signaling system of slime molds (Prabhu and Eichinger 2006).

Class F, Frizzled/ smoothened is required for the hedgehog signaling. These classes are called the families of GPCRs and are denoted as family level GPCRs in our work. The decomposition of these classes into their subfamilies, sub-subfamilies, and subtypes are termed as subfamily, sub-subfamily, and subtype levels of GPCRs.

1.2 Computational tools to predict and classify GPCRs

In recent times, researchers are more interested in the functional roles of GPCRs at the finest subtype level. Because each subtype has its own characteristic ligand-binding property, coupling partners of trimeric G-proteins, and interaction partners of oligomerization (Kristiansen 2004), prediction of GPCRs at the fifth level becomes significant in the effort to decipher GPCRs.

It is however a challenging task. Fortunately, more GPCR sequences are now being accumulated into the GPCRDB database, which makes it possible to accurately predict GPCRs at all the five levels. GPCR classification plays a crucial role in predicting the function of the protein and a step towards applications of GPCRs.

The prediction of GPCRs, based on structure and function is possible because of the

production of very large-scale genome sequencing projects (Vaidehi et al. 2002). However, it is difficult to develop a comprehensive classification system for all the subtypes of GPCR due to its inherent diversity (Daives et al. 2007).

Many GPCR prediction methods have been proposed, in which some of the methods are also based on the sequence similarity searching in protein database using alignment tools (Pearson 2000). However, one of the major problems of sequence similarity search-based methods is that it fails if the tested protein sequences have no match to the database sequences. In addition, in case of GPCRs, function similarity relationship is still unclear.

In addition, in case of GPCRs, function similarity relationship is still unclear. Other methods include the use of covariant discriminant algorithm (Elrod and Chou 2002), support vector machine (SVM) (Karchin et al. 2002), k-nearest neighbors (KNN) (Gao and Wang 2006; Khan et al. 2008a), statistical analysis method (Chou and Elrod 2002), Hidden Markov Models (Qian et al. 2003), and binary topology pattern (Inoue et al. 2004). Ensemble approaches have also been used for protein identification (Huang et al. 2004; Shen and Chou 2007; Shen et al. 2007).

A number of computational methods have been developed to predict GPCRs based on their sequences (Guo et al. 2006; Wen et al. 2007; Chou 2005b). These computational methods are now frequently used to facilitate the identification and characterization of receptors, which could lead to the discovery of novel signal transduction pathways and provide new insights into the disease process and drug discovery.

Also a number of tools (webservers) have been developed from all these methods for the prediction and classification GPCRs, which includes the GPCRTm, GPCRpred, PCA-GPCR, PredGPCR, SVMProt, GPCRGIA and GPCRCA. A brief description of these programs and the algorithms they use are as follows;

1. **GPCRTm** is a membrane protein topology prediction method based on a hidden Markov model. This program is for prediction of transmembrane helices in proteins. One of the main advantages of an HMM is that it is possible to model helix length, which has only been done fairly crudely in most other methods, by setting upper

and lower limits for the length of a membrane helix. The HMM is very well suited for prediction of transmembrane helices because it can incorporate hydrophobicity, charge bias, helix lengths, and grammatical constraints into one model for which algorithms for parameter estimation and prediction already exist (see e.g. Durbin et al. (1998)). We further apply TMHMM to predict all membrane proteins in a large collection of mostly fully sequenced genomes, and present statistics on the frequency of proteins with different topologies.

2. **GPCRpred** is a support vector machine (SVM)-based method, developed for predicting families and subfamilies of GPCRs from the dipeptide composition of proteins. The method is further able to predict five major classes or families of GPCRs with an overall Matthews correlation coefficient (MCC) and accuracy of 0.81 and 97.5% respectively.

3. **PCA-GPCR** predicts GPCRs using a comprehensive set of 1497 sequence-derived features. The principal component analysis is first employed to reduce the dimension of the feature space to 32. Then, the resulting 32-dimensional feature vectors are fed into a simple yet powerful classification algorithm, called intimate sorting, to predict GPCRs at five levels. The prediction at the first level determines whether a protein is a GPCR or a non-GPCR. If it is predicted to be a GPCR, then it will be further predicted into certain family, subfamily, sub-subfamily and subtype by the classifiers at the second, third, fourth, and fifth levels, respectively.

4. **PredGPCR** system is based on a probabilistic method that uses family specific profile HMMs in order to determine to which GPCR family a query sequence belongs or resembles. The approach proposed in this method exploits the descriptive power of profile HMMs along with an exhaustive discrimination assessment method to select only highly selective and sensitive profiles, for each family. The collection of these profiles constitutes a signature library, which is scanned, for significant matches with a given query sequence.

5. **SVM-Prot** is a web-based software that employs a machine learning method, support vector machines (SVM), for predicting protein functional families from protein sequences irrespective of sequence or structural similarity, which have shown good predictive performances to complement other methods or as part of the integrated approaches in predicting the function of diverse classes of proteins including the distantly-related proteins and homologous proteins of different functions. Given that the functional prediction capacity could be enhanced by the integration of multiple methods, two machine learning prediction tools, K nearest neighbor (kNN) and probabilistic neural networks (PNN), were integrated into this version of SVM-Prot to facilitate the collective assessment of protein functional families.

6. **GPCR-GIA** was developed for identifying GPCR proteins and their family classes solely based on the sequence information. GPCR-GIA was established on the following two cornerstones: one is the combination of the Pseudo amino acide composition (PseAAC) with the grey incidence analysis (GIA) which is used as a distance measure ('nearness'), and the other is the fusion of many individual basic classifiers such as the kNN.

7. **GPCRCA** where "CA" stands for "Cellular Automaton" (Wolfram et al. 1984), meaning that the CA images have been utilized to reveal the pattern features hidden in piles of long and complicated protein sequences. Meanwhile, the graylevel cooccurrence matrix factors extracted from the CA images are used to represent the samples of proteins through their pseudo amino acid composition. GPCRCA is a twolayer predictor: the first layer prediction engine is for identifying a query protein as GPCR on non-GPCR; if it is a GPCR protein, the process will be automatically continued with the secondlayer prediction engine to further identify its type among the following six functional classes: (a) rhodopsinlike, (b) secretinlike, (c) metabotrophic/glutamate/pheromone; (d) fungal pheromone, (e) cAMP receptor, and (f) frizzled/smoothened family.

Of these entire classification approaches, SVM is quite promising regarding its performance. However, SVM is a binary classifier while GPCRs classification is a multi-class problem. The selective top-down approach by Daives et al. (2007) is also very effective regarding GPCRs classification. They have employed selective top-down approach. In their approach, a numeric feature vector is constructed, whereby 5 z-values such as lipophilicity (z1), bulk and polarisability (z2), polarity (z3), and electronics effects (z4 and z5) are derived from 26 real physiochemical properties of amino acids.

Yet there is no SVM program developed purposefully for the prediction and classification of GPCRs.

With the tools available, there are some limitations and shortcomings which are;

1. No testing with TM non-GPCRs which was evident when the PCA-GPCR webserver was tested with several sets of 10 TM proteins each but produced a false positives ranging from 70% to 100%. as determined by one member of our research group, Perez during the testing all these tools with the sequences.

2. Many algorithms are published but the actual programs are either not available, or cannot be used for large scale testing, SVM being a typical example.

3. No direct comparison of different algorithms with the same test dataset has been conducted. That is, the tools stated above all used their own datasets.

1.3 Research objectives

Despite the existence of a good variety of published GPCR prediction and classification algorithms as mentioned above, there are several problems with their usability and reliability. First, in many published algorithms, no executable programs or even source codes are no longer available or supported. So, they cannot be used for any practical purposes.

Second, all the available GPCR prediction web servers give a high false positive prediction rate among transmembrane non-GPCRs. Third, the available web-based programs, while performing well in terms of GPCR prediction, generally do not have very good accuracy in GPCR classification. The eventual goal of my project is to analyze the discriminative power of the existing strategies and devise an integrated algorithm to overcome the above problems and produce a reliable prediction and classification system that can be widely used by scientists for GPCR research.

Within the timeframe of my masters thesis, my work focuses on GPCR prediction, which refers to predicting whether a protein is a GPCR or not when given its amino acid sequence. The specific aims are:

1. Organize a set of confirmed GPCR and non-GPCR proteins to serve as the training and testing data sets for both existing and new algorithms for GPCR prediction and classification. The non-GPCR dataset will contain transmembrane proteins.

2. Compile from existing literature all the protein features (e.g., amino acid and dipeptide composition) that have been used for GPCR prediction and classification.

3. Implement and compare the performance of several integrated statistical and machine learning computational approaches locally on our bioinformatics computer network using the dataset obtained in Aim 1 for both training and testing. During the performance evaluation process, relative importance of those sequence features compiled in Aim 2 will be assessed in terms of their contribution to prediction accuracy.

When the above specific aims are achieved, I will be able to use the analysis results of the GPCR prediction to inform the design of additional computational approaches to tackle the GPCR classification problem to complete the research for my Ph.D. dissertation. The proposed work plan for my PhD dissertation is laid out in Chapter 5.

Chapter 2

Literature Review

This chapter accounts for the various approaches and algorithms related to the classification of GPCRs into different levels. The first section highlights on the computational algorithms which use machine learning and statistical approaches for the classification and prediction. It also entails the accuracies by the various algorithms. The second section covers the combination of different methods.

2.1 Overview of GPCR prediction and classification

GPCRs as membrane proteins, are very difficult to crystallize and most of them will not dissolve in normal solvents [Xiao X, Wang Pet al.,2009]. Accordingly, the 3D structure of only squid rhodopsin, β_1, β_2 adrenergic receptor and the A2A adenosine receptor have been solved to data. In contrast, the amino acid sequences of more than 1000 GPCRs are known with the rapid accumulation of data of new protein sequence produced by the high-throughput sequencing technology. In view of the extremely unbalanced state, it is vitally important to develop a computational method that can fast and accurately predict the structure and function of GPCRs from sequence information. Many predictive methods have been developed, which in general, can be roughly divided into three categories. The first one is proteochemometric approach developed by Lapinsh [Lapinsh M et al., 2005]. However, the methods need structural information of organic compounds. The second one is based on similarity searches using primary database search tools (e.g. BLAST, FASTA) and such database searches coupled with searches of pattern databases (PRINTS) [Lapinsh

M et al.,2002]. However, they do not seem to be sufficiently successful for comprehensive functional identification of GPCRs, since GPCRs make up a highly divergent family, and even when they are grouped according to similarity of function, their sequences share strikingly little homology or similarity to each other [Inoue Y et al., 2004]. The third one is based on statistical and machine learning method, including support vector machines (SVM) [Karchin R et al.,2002], hidden Markov models (HMMs) [Qian B et al., 2003], covariant discriminant (CD) [Chou KC et al., 2002], nearest neighbor (NN) [Gao QB et al.,2006] and other techniques .

2.2 Protein features commonly used in prediction and classification.

The basic compositions of life, proteins play a central role in most cellular functions such as gene regulation, metabolism and cell proliferation. As a typical pattern recognition problem, computational methods for protein structural class prediction consist of three main steps: i) protein feature representation; ii) algorithm selection for classification; iii) optimal feature selection.

Among the three steps, feature extraction is the most critical factor for the success of protein structural class prediction. Feature extraction means how to extract features from protein sequences so that each protein is represented as a fixed-length numerical vector (Zhen-Ling Peng et al.,2010). The commonly-used feature extraction methods are based on amino acid composition (Chou KC et al.,2004) and dipeptide composition (Qao QB et al.,2006), and more complicated ones include Chous pseudo amino acid composition (Lin WZ et al.,2006), the cellular automaton image approach (Xiao X et al.,2009), profile hidden Markov models (Papasaikas PK et al.,2003), fast Fourier transform (Guo YZ et al.,2006), wavelet-based time series analysis (Gupta R et al.,2008) and Fisher Score Vectors (Karchin R et al, Karplus K, Haussler D).

Eight sets of commonly-used structural and physicochemical features, including 1355 descriptor values, can be divided into four groups each of which has been used as an independent set of features for predicting proteins and peptides of various profiles by using statistical learning methods. The first group includes two features, which are the amino acid composition and dipeptide composition, with 2 descriptors and 420 descriptor values (Shepherd et al., 2003). Each of the second, third and fourth group contains a different autocorrelation feature: normalized Moreau-Broto autocorrelation (Reczko et al.,1994), Moran autocorrelation (Horne et al.,1998) and Geary autocorrelation (Soka et al.,2006). Each of these features has 8 descriptors and 240 descriptor values. The fifth group contains two sequence-order feature sets (Chou et al.,2000), one is sequence-order-coupling number with 2 descriptors and 60 descriptor values, and the other is quasi-sequence-order with 2 descriptors and 100 descriptor values. Apart from these descriptors, it can also compute previous autocorrelation descriptors based on user-defined properties.

Amino acid and dipeptide composition are simplistic descriptors of protein sequence features (Shepherd et al., 2003), which have been used for predicting protein fold and structural classes (Grassmann et al.,1999), functional classes (Bhasin et al.,2004) and subcellular locations (Hua et al.,2001) at accuracy levels of $72\%-95\%$, $83-97\%$ and $79-91\%$, respectively. Amino acid composition is the fraction of each amino acid type in a sequence: $f(r) = N_r/N$, where r = 1, 2, 3, ... , 20, N_r is the number of amino acid of type r, and N is the length of the sequence. A total of 20 descriptor values are computed for the 20 types of amino acids. Dipeptide composition is defined as: $fr(r, s) = N_{rs}/(N-1)$, where $r, s = 1, 2, 3, ..., 20$, and N_{rs} is the number of dipeptides of amino acid type r and s (Bhasin et al.,2004). A total of 400 descriptor values are computed for the 20×20 amino acid combinations.

Autocorrelation features describe the level of correlation between two objects (protein or peptide sequences) in terms of their specific structural or physicochemical property (Broto et al.,1984), which are defined based on the distribution of amino acid properties along the sequence (Kawashima et al.,2000). There are eight amino acid properties used for deriving

these autocorrelation descriptors. The first is hydrophobicity scale derived from the bulk hydrophobic character for the 20 types of amino acids in 60 protein structures (Cid,H. et al.,1992). The second is the average flexibility index derived from the statistical average of the B-factors of each type of amino acids in the available protein X-ray crystallographic structures (Bhaskaran et al.,1988). The third is the polarizability parameter computed from the group molar refractivity values originally provided by Hansch et al. (Charton et al.,1982). The fourth is the free energy of amino acid solution in water measured by Hutchins (Charton et al.,1982). The fifth is the residue accessible surface areas taken from average values from folded proteins (Chothia et al.,1976). The sixth is the amino acid residue volumes measured by Fisher (Bigelow et al.,1967). The seventh is the steric parameters derived from the van der Waals raddi of amino acid side-chain atoms (Charton et al.,1981). The eighth is the relative mutability obtained by multiplying the number of observed mutations by the frequency of occurrence of the individual amino acids (Dayhoff et al.,1978).

2.3 Computational algorithms

2.3.1 Supervised machine learning algorithms

Support Vector Machines

Support vector machines (SVMs) are a set of related supervised learning methods used for classification, regression and novelty detection. They belong to a family of generalized linear classifiers and are all about decision boundaries so do not provide posterior probabilities. They can also be considered a special case of Tikhonov regularization. A special property of SVMs is that they simultaneously minimize the empirical classification error and maximize the geometric margin; hence they are also known as maximum margin classifiers.

For a two-class classification problem using using linear models of the form

$$y(x) = w^T \phi(x) + b \tag{2.1}$$

where $\phi(x)$ denotes a fixed feature-space transformation, and let the bias parameter b be explicit. Note that to avoid working explicitly in feature space a dual representation expressed in terms of kernel functions is used. The training data set comprises N input vectors $x_1, ..., x_N$, with corresponding target values $t_1, ..., t_N$ where $t_n \in \{1, 1\}$, and new data points x are classified according to the sign of y(x).

Let's assume for a moment that the training data set is linearly separable in feature space, so that by definition there exists at least one choice of the parameters w and b such that a function of the form (2.1) satisfies $y(x_n) > 0$ for points having $t_n = +1$ and $y(x_n) < 0$ for points having $t_n = 1$, so that $t_n y(x_n) > 0$ for all training data points. The support vector machine approaches the problem of multiple solutions (all classify the training data set exactly) through the concept of the margin, which is defined to be the smallest distance between the decision boundary and any of the samples.

In support vector machines the decision boundary is chosen to be the one for which the margin is maximized. The maximum margin solution can be motivated using computational learning theory, also known as statistical learning theory. However, a simple insight into the origins of maximum margin has been given by Tong and Koller (2000) who consider a framework for classification based on a hybrid of generative and discriminative approaches. They first model the distribution over input vectors x for each class using a Parzen density estimator with Gaussian kernels having a common parameter σ^2. Together with the class priors, this defines an optimal misclassification-rate decision boundary. However, instead of using this optimal boundary, they determine the best hyperplane by minimizing the probability of error relative to the learned density model. In the limit $\sigma^2 \to 0$, the optimal hyperplane is shown to be the one having maximum margin. The intuition behind this result is that as σ^2 is reduced, the hyperplane is increasingly dominated by nearby data points relative to more distant ones. In the limit, the hyperplane becomes independent of

data points that are not support vectors.

In an experiment conducted by Rachel et al.,(2002) to classify Protein sequences of the GPCR superfamily (Watson and Arkinstall,1994). Specifically to recognize small subfamilies of GPCRs that bind the same ligand. This requires extension of the two-class problem to a k-class problem. They chose the simplest approach to multi-class SVMs by training k one-to-rest classifiers.An initial step was required in which each protein sequence is transformed into a fixed-length feature vector. Next a two-class SVM was trained for each GPCR subfamily.

Each of these SVMs learns to distinguish between subfamily members and non-members by making several passes through training set and using the kernel function to compute a weight for each sequence. They described the results of a two-fold cross-validation experiments that compare multi-class SVMs with two popular classification methods. They found that one of the methods are better GPCR discriminators than SVMs at the class (super family) level. However, SVMs make significantly fewer errors than the other methods when applied to the problem of discriminating subfamilies of GPCRs, particularly when methods are evaluated by their specificity, or performance in the low false-positive range. That is, the errors per sequence at the Minimum Error Point (MEP) were 13.7% for multi-class SVMs, 17.1% for their SVMtree method of hierarchical multi-class SVM classification as compared to 30% and 49% for the other two methods. In general, SVM makes its prediction using a mathematical tool known as a kernel function. Compared to BLAST (Altschu et al., 1990) and Hidden Markov Models (HMMs ; Durbin et al.,1998) this method is computationally expensive.

Covariant-discriminant algorithm

Linear and Quadratic Discriminant analysis (LDA/QDA) are common tools for classification problems. For these methods we assume observations are normally distributed within group. We estimate a mean and covariance matrix for each group and classify using Bayes theorem. With LDA, we estimate a single, pooled covariance matrix, while for QDA we estimate a separate covariance matrix for each group. Rarely do we believe in a homogeneous covariance structure between groups, but often there is insufficient data to separately estimate covariance matrices.

Consider the usual two class problem: our data consists of n observations, each observation with a known class label $\in \{1, 2\}$, and p covariates measured per observation. Let y denote the n-vector corresponding to class (with n_1 observations in class 1 and n_2 in class 2), and X, the n by p matrix of covariates. We would like to use this information to classify future observations.

We further assume that, given class y(l), each observation, x_l , is independently normally distributed with some class specific mean $\mu_y(l) \in \mathbf{R}^p$ and covariance $\sum_{y(l)}$, and that y(l) has prior probability π_1 of coming from class 1 and π_2 from class 2. From here we estimate the two mean vectors, covariance matrices, and prior probabilites and use these estimates with Bayes theorem to classify future observations. In the past a number of different methods have been proposed to estimate these parameters. The simplest is Quadratic Discriminant Analysis (QDA).which estimates the parameters by their maximum likelihood estimates

$$\pi_k = \frac{n_k}{n}$$

$$\hat{\mu_k} = \frac{1}{n_k} \sum_{y(l)=k} x_l$$

and

$$\hat{L_k} = \frac{1}{n_k} \sum_{y(l)=k} (x_l - \mu_k)(x_l - \mu_k)^T$$

To classify a new observation x, one finds the class with the highest posterior probability. This is equivalent in the two class case to considering

$$D(x) = \log(\frac{\pi_1}{\pi_2}) - \frac{1}{2}(x - \mu_1)^T L_1^{-1}(x - \mu_1) + \frac{1}{2}(x - \mu_2)^T L_1^{-1}(x - \mu_2) + \log det L_1^{-1/2} L_2^{1/2}$$

and if D(x) > 0 then classifying to class 2, otherwise to class 1.

Linear Discriminant Analysis (LDA) is a similar but more commonly used method. It estimates the parameters by a restricted MLE the covariance matrices in both classes are constrained to be equal. So, for LDA

$$\hat{L_1} = \hat{L_2} = \frac{1}{n} \sum_{y(l)=k} (x_l - \mu_{y(l)})(x_l - \mu_{y(l)})^T$$

While one rarely believes that the covariance matrices are exactly equal, often the decreased variance from pooling the estimates greatly outweights the increased bias.

In the study on correlation of G-protein-coupled receptors types with amino acid composition, the covariant-discriminant algorithm proposed by Chou and Elrod (1999) was used. The rhodopsin-like amine according to the GPCRDB (Horn et al.,1998) can be classified into six classes which are acetylcholine, adrenoceptor, dopamine, histamine, serotonin and

octopamine. The sequesnces from GPCRDB described as being one of the six classes were selected and all of the incomplete sequences containing only fragments of the receptors were removed.Also NRDB progam (Gish,1999) was used to check that none of the sequences was identical to any others in the data set. After the screening procedure, two of the receptor types (hystamine and octopamine) were left out of the statistical analysis since they contained only few sequences. The covariant-discriminant algorithm was utilized to analyze the GPCR based on their amine acid compositions. The overall Jackknife test rate for the data set of 167 was 83.23%. This suggests that the types of GPCR are predictable to s considerably accurate extent of a complete or quasi-training data set be establised.

Bagging Classification Tree

Tree-based methods for regression and classification involve stratifying or segmenting the predictor space into a number of simple regions. In order to make a prediction for a given observation, we typically use the mean or the mode of the training observations in the region to which it belongs. There are several tree construction algorithms, such as CART Classification and Regression Tree) , OC1, ID3 and C4.5 (Breiman et al., 1984;Quinlan, 1993; Salzberg et al., 1998). An advantage of the tree-based classification algorithm is its relative ease of interpretation, which can help the user to understand and improve the classification rules. Here we will concentrate on the CART and the C4.5 algorithm for classification tree construction. Classification tree is a useful qualitative-response prediction algorithm that has been used in many areas of bioinformatics, such as gene finding, tumor classification, etc. (Salzberg et al., 1998; Zhang et al., 2001; Dudoit et al., 2002). We predict that each observation belongs to the most commonly occuring class of training observation in the region to which it belongs. In interpreting the results of a classification tree, we are often interested not only in the class prediction corresponding to a particular terminal node region, but also in the class proportions among the training observations that fall into that region. Since we plan classification to assign an observation in a given region to the most commonly occurring error rate class of training observations in that

region, the classification error rate is simply the fraction of the training observations in that region that do not belong to the most common class:

$$E = 1 - \max_k(\hat{p}_{mk})$$

where $\hat{p}_{mk}$ represents the proportion of training observations in the mth region that are from the kth class. However, it turns out that classification error is not sufficiently sensitive for tree-growing, and in practice two other measures are preferable.
The Gini index is defined by

$$G = \sum_{k=1}^{K} \hat{p}_{mk}$$

a measure of total variance across the K classes.There is also the cross entropy index. Therefore, like the Gini index, the cross-entropy will take on a small value if the mth node is pure.

When building a classification tree, either the Gini index or the cross-entropy are typically used to evaluate the quality of a particular split, since these two approaches are more sensitive to node purity than is the classification error rate. Any of these three approaches might be used when pruning the tree, but the classification error rate is preferable if prediction accuracy of the final pruned tree is the goal.

The C4.5 algorithm uses a divide-and-conquer approach to growing decision trees. This algorithm selects a feature (an amino acid composition) to split the training data into subsets (nodes of the decision tree). The default criterion used by C4.5 for feature selection is information gain ratio, a measure based on information theory (Shannon, 1948). This measure can quantify how well a given feature separates the training data. At each node the training dataset will be further divided until some stopping criteria are satisfied. Then each terminal subset (leaf node) is assigned to a class label (receptor type). After generating the maximal classification tree, a pruning technique is used to simplify the classification

tree and avoid over-fitting. Pruning a tree consists of replacing a whole sub-tree by a leaf node. The replacement takes place if the expected error rate in the subtree is greater than that in the single leaf. We also present C4.5 algorithm has also been applied to genomic sequence annotation and protein phosphorylation sites identification (Kretschmann et al., 2001; Muggleton et al., 2001; Berry et al., 2004).

The use of tree based algorithm for prediction and classification is very effective and is used to address several classification problems. Huang et al. (2004) describes a bagging classification tree for classifying GPCRs into sub-families and sub-sub-families based on the amino-acid composition. Here it uses the C4.5 algorithm which is used to generate a decision tree. In total 4395 sequences were classified into sub-families and 4036 sequences were classified into sub-sub-families with an accuracy of 91.1% and 82.4% respectively.

2.3.2 Unsupervised machine learning algorithms

Principal Component Analysis(PCA)

PCA refers to the process by which principal components are computed, and the subsequent use of these components in understanding the data. PCA is an unsupervised approach, since it involves only a set of features $X_1, X_2, ..., X_p$ and no associated response Y. PCA finds a low-dimensional representation of a data set that contains as much as possible of the variation. Each of the dimensions found by PCA is a linear combination of the p features. Principal components (PCs) allow us to summarize large set of correlated variables with a smaller number of representative variables that collectively explain most of the variability. PCs are directions in feature space along which original data are highly variable and also define lines and subspaces that are as close as possible to the data cloud.

The first principal component of a set of features $X_1, X_2, ..., X_p$ is the normalized linear

combination of the features.

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + ... + \phi_{p1}X_p \tag{2.2}$$

that has the largest variance. By normalized, we mean that $\sum_{j=1}^{p} \phi_{ji}^2 = 1$ where the $\phi_{11}, ..., \phi_{p1}$ are the loadings of the first principal components.

Given a n $\times$ p data set $\mathbf{X}$, how how do we compute the first PC? Since we are only interested in variance, we assume that each of the variables in $\mathbf{X}$ has been centered to have mean zero (that is the column means of $\mathbf{X}$ are zero). We then look for the linear combinations of the sample feature values of the form

$$z_{i1} = \phi_{11}X_{i1} + \phi_{21}X_{i2} + ... + \phi_{p1}X_{ip} \tag{2.3}$$

that has the largest sample variance, subject to the constraints that $\sum_{j=1}^{p} \phi_{ji}^2 = 1$

In other words, the first PC loading vector solves the optimization problem

$$\max_{\phi_{11},...,\phi_{p1}} \frac{1}{n}\sum_{i=1}^{n} z_{i1}^2$$

$$\text{subject to} \quad \sum_{j=1}^{p} \phi_{ji}^2 = 1 \tag{2.4}$$

since $\frac{1}{n}\sum_{i=1}^{n} x_{ij} = 0$, the average of $z_{11}, ..., z_{n1}$ will be zero as well. Hence the objective of maximizing (2.4) is just the sample variance of n values of z_{i1}.

Peng et al. (2010) proposed a method called PCA-GPCR for predicting and classifying GPCRs into family, sub-family, sub-sub-family, subtype from a large dataset. The study was done on 1497 sequence derived features which was further reduced into 32-dimensional feature vectors using the PCA. By the first level classifier a new protein sequence is predicted to be either a GPCR or a non-GPCR. If it is predicted to be a GPCR, it will be further classified into one of the four families, which is done by the second-level classifier.

The third-level classifiers hence determine which subfamily the protein belongs to. For some subfamilies, the fourth-level classifiers are used to determine the sub-subfamily of the protein. Finally, the fifth-level classifiers determine the subtypes of the protein, if any. Jackknife tests show that the overall accuracies of PCA-GPCR are 99.5%, 88.8%, 80.47%, 80.3%, and 92.34% for the five levels, respectively.

K-Nearest Neighbor(KNN)

The KNN algorithm is a robust and versatile classifier that is often used as a benchmark for more complex classifiers such as Artificial Neural Networks (ANN) and Support Vector Machines (SVM).

Let x denote a feature (predictor, attribute) and y to denote the target (label, class) we are trying to predict.KNN falls in the supervised learning family of algorithms. Informally, this means that we are given a labelled dataset consisting of training observations (x,y) and would like to capture the relationship between x and y. More formally, our goal is to learn a function h:X$\mapsto$Y so that given an unseen observation x, h(x) can confidently predict the corresponding output y.

The KNN classifier is also a non parametric(means it makes no explicit assumptions about the functional form of h, avoiding the dangers of mismodeling the underlying distribution of the data) and instance-based learning algorithm (means that our algorithm doesnt explicitly learn a model. Instead, it chooses to memorize the training instances which are subsequently used as knowledge for the prediction phase).

In the classification setting, the K-nearest neighbor algorithm essentially boils down to forming a majority vote between the K most similar instances to a given unseen observation. Similarity is defined according to a distance metric between two data points. A popular choice is the Euclidean distance given by

$$d(x, x') = \sqrt{(x_1 - x'_2)^2 + (x_2 - x'_2)^2 + ... + (x_n - x'_n)^2}$$

but other measures can be more suitable for a given setting and include the Manhattan, Chebyshev and Hamming distance

More formally, given a positive integer K, an unseen observation x and a similarity metric d, KNN classifier performs the following two steps:

- It runs through the whole dataset computing **d** between **x** and each training observation. Well call the **K** points in the training data that are closest to **x** the set $\mathcal{A}$. Note that K is usually odd to prevent tie situations.

- It then estimates the conditional probability for each class, that is, the fraction of points in A with that given class label. (Note $\mathbf{I}(\mathbf{x})$ is the indicator function which evaluates to 1 when the argument x is true and 0 otherwise)

$$P(y = j|X = x) = \frac{1}{K} \sum_{i \in \mathcal{A}} I(y^{(i)} = j)$$

  Finally, our input x gets assigned to the class with the largest probability.

  In a paper by Qing-Bing Gao et al.,(2006), a nearest neighbor method was introduced to discriminate GPCRs from non-GPCRs and subsequently classify GPCRs at four levels on the basis of amino acid composition and dipeptide composition of proteins. The prediction performance was evaluated on a non-redundant dataset consisted of 1406 GPCRs for six families and 1406 globular proteins using the jackknife test. A high overall accuracy has been achieved in each step using dipeptide-based method. Moreover, comparisons with existing methods in the literature show that their method achieves great competitive performance.

2.3.3 Statistical methods

Family Specific Motif

Due to the significant role GPCR act both as receptors for outside ligands (ligands range from photons inducing sight to small peptides inducing neurological effects) and actuators for internal processes, it was very important to be able to distinguish which ligands that a specific GPCR interacts with and which parts of the sequence have a particularly important role. As a result, there were two presiding goals for computational methods in GPCR research: first, to classify GPCR sequences with respect to subfamilies within Family A which contains more than 80 percent of human GPCRs, second and most importantly to identify the key ligand-interacting sites using the sequence alone.

In response to the above requirements,Murat Can Cobanoglu et al., (2011) developed a classification technique that also pinpoints ligand-receptor interaction sites. Their proposed technique involved identifying the frequent residue triplets in the sequence using sequence-derived motifs, calculating their distinguishing power among the subfamilies by the Distinguishing Power Evaluation (DPE) technique they proposed, and deducing rules from this information.

The classifier proposed here was called GPCRBind. It is a rule extraction method, and while training takes time on the order of hours, classification of a sequence takes milliseconds. This property of GPCRBind makes it suitable for being used as a classification server. The GPCRBind method, requires random partitioning. Due to this randomness the results of two successive runs are not identical. Therefore, they repeated the whole method 100 times and reported the average accuracy. They implemented the proposed methods and tested them on real data sets. The results were compared with the state-of-the-art GPCR classification techniques. Experiments showed that GPCRBind outperformed the state-of-the-art classification techniques.

Binary Topology Pattern

In a paper by Inoue et al.,(2004) ,they proposed a novel and simple (i.e., fast) method for the functional classification and identification of mammalian-type GPCRs based on the TM topology patterns (Poluliakh et al., 2000; Sugiyama et al., 2003; Tusndy et al., 1997; Clements and Martin, 2002), instead of using techniques based on functional domain detection or sequence similarity searches. In their method, each loop length is expressed as 1 for a long loop or 0 for a short loop, and the TM topology pattern was expressed as a string of binarized loop lengths (the binary topology pattern, BTP). The BTP method was expected to even identify and classify GPCRs with low amino acid sequence similarities, that sequence similarity searches such as BLAST (Altschul et al., 1997) cannot easily identify or classify, e.g., searching for mammalian-type GPCRs in invertebrate organisms. Comprehensive functional classification and identification of mammalian-type GPCRs were carried out by applying the BTP method to 10 different eukaryotic species (H. sapiens, M. musculus, F. rubripes, C. intestinalis, A. thaliana, D. melanogaster, A. gambiae, C. elegans, P. falciparum, S. cerevisiae) for which the complete genome sequences or draft sequences have already been released.

They described it as a stepwise method where the first step works with three unified functional groups, (i) Class A and Non-GPCR, (ii) Class B + Class C and (iii) Frizzled/Smoothened, with a certain threshold value assigned. In the next step, it works with classifying Class B and Class C, and in step three, Class C is divided into three functional groups followed by Step four, five and six determining the rest of the functional groups along with the identification of the mammalian-type GPCRs. The accuracy is 100% for three groups, and four groups with more than 80%.

Statistical Encoding Method

In a work by Muhammad Javed Iqbal et al, (2016), a statistical distance-based encoding method is used to represent a variable length lower similarity protein sequence with a fixed

size vector. The protein sequences of three GPCRs families were investigated in the experiments due to their importance in the pharmaceutical industry. The sequences belong to the families of GPCRs having very weak sequence similarity which exhibits difficulties in classifying them into different classes. Each phase of their proposed framework was employed to perform a specific task of classifying protein sequences into their corresponding families/subfamilies.

The classification results obtained were the average results of various rounds of the classifier. Different performance measurement metrics such as accuracy, true positive rate (TPR), false positive rate (FPR), specificity, sensitivity, recall, F-measure and Mathews Correlation Coefficient (MCC) were investigated for performance evaluation of the protein sequence classification technique. Some performance measured metrics were computed which includes classification accuracy, specificity, sensitivity, recall, f-measure and MCC. The best accuracy results on Datasets 1 and 2 were in the range of 94% to 97.9%, which shows some improvement from the accuracy results which were in the range of 90.7% to 95% in the previous studies (X. Zhou et al., 2010 and M. C. Cobanoglu et al., 2011).

Structural Regional lengths

In the study by Sahin et al. (2014), they proposed a method named GPCRsort, which determines the class of a GPCR using its structural properties. Specifically, they used the lengths of the transmembrane and loop regions of the GPCR structure. The method first determined the transmembrane regions of GPCRs, and constructs feature vectors using the lengths of the regions. Random Forest (Breiman, 2001) classifier is employed in the learning and decision parts of the method. The method can predict GPCRs at every level of the GPCR class hierarchy tree. This shows the generality of GPCRsort, as the other techniques are inadequate to make the exact classification. The experimental results show that GPCRsort is very effective and outperforms the current known techniques for GPCR classification. Highest accuracy, 97.3%, is achieved with GPCRsort method when compared

to the other techniques under the same experimental conditions. Besides these advantages, running in minutes makes GPCRsort a faster prediction algorithm among others.

2.3.4 Combination of tools

Genetic Ensemble

Naveed and Khan (2012) introduced GPCR-MPredictor which predicts and classifies GPCRs into five levels (family, sub-family, sub-sub-family, subtype) including the prediction. In order to handle the dimesionality problem of the dipeptide compositions, they used genetic algorithm (GA)-based feature selection to reduce the dimensionality and to improve the classification accuracy. Their approach is an ensembled approach where individual classifiers like k-nearest neighbor, support vector machine, probabilistic neural networs, J48, Adaboost and Naives Bayes classifiers have been used. However, they observed that SVM performs better on dipeptide composition among all of these classifiers. Amino acid composition, pseudo amino acid composition and dipeptide composition are the three features used to predict and classify GPCRs. The proposed method is compared with the PCA-GPCR approach (Peng et al. 2010). The predictive performance of their approach is comparatively higher than that of PCA-GPCR at all levels of GPCRs. The GPCR-MPredictor yields an accuracy of 99.75, 92.45, 87.80, 83.57, and 96.17% at the five levels, respectively.

Fast Fourier Transform With Support Vector Machine

Guo et al. (2006) introduced a fast Fourier transform based support vector machine method to classify GPCRs and Nuclear Receptors (NRs) based on the hydrophobicity of proteins. The three principal properties of hydrophobicity represented by hydrophobicity model, electron-ion interaction potential model and c-p-v model are used to transform the protein sequences into numerical sequences. Three hydrophobicity scales were selected for the optimization as hydrophobicity can vary due to different experimental conditions, different

organic solvents and computing approaches. For performance measurement, Jackknife test was used as well as for prediction quality, accuracy, total accuracy and Matthews correlation coefficient were evaluated. Higher accuracy is achieved with the hydrophobicity scale than c-p-v or electron-ion interaction potential model.

Support Vector Machine with different approaches

Yabuki et al. (2005) has described a system called GRIFFIN (G-Protein and Receptor Interaction Feature Finding Instrument) which uses Support Vector Machines and Hidden Markov Model to predict GPCR and G-Protein coupling selectivity along with a hierarchical SVM classifier including the feature vectors to predict Class A GPCRs. For the other type of families (Opsins, Ofactory subfamilies of Class A, Class B, Class C, frizzled and smoothened) HMM is used. As BLAST and FASTA uses sequence similarity for predicting the protein, yet it is not clear to predict GPCRs based on sequence similarity relationship. This system is unique as it uses information from GPCR ligand information and GPCR sequence. SwissProt and TrEMBL databases are used as both ligand and sequence information are present. In total, they have used twenty-four features for ligands and GPCRs.

Karchin et al. (2002) has used a simple nearest neighbor approach (BLAST), Hidden Markov Model (HMM) and support vector machines (SVMs) which is a group of statistical algorithms for recognizing superfamilies and the small subfamilies of GPCRs that bind the same ligand. The work is focused on comparing different methods with SVMs to observe which one is better computationally. For the classification of GPCRs, the primary sequence information is used which required the extension of the two-class problem to a k-class problem. Karchin et al. (2002) have used the simplest approach to multi-class SVMs by training k one-to-rest classifiers. SVM is computationally expensive but it has significantly less Minimum Error Point (MEP) than the other methods especially in the case for classifying subfamilies. It is also observed that the higher classification with good approximation can be achieved using SVMtree method. The future work is focused on classifying the subfamilies based on the suitable feature selection for the subfamilies along

with the biological knowledge of each subfamilys transmembrane topology.

Li et al. (2010) proposed a three-layer classifier for GPCRs based on the combination of SVM with feature selection method. The method holds high accuracy for classifying into superfamily, family and subfamily of GPCRs. In every level, minimum redundancy maximum relevance (mRMR) (Peng et al.,2005) is utilized to pre-evaluate features with discriminative information. After that, to further improve the prediction accuracy and to obtain the most important features, genetic algorithms (GA) (JH 2005) is applied to feature selection. Finally, three models based on SVM are constructed and used to identify whether a query protein is GPCR and which family or subfamily the protein belongs to. The prediction quality evaluated on a non-redundant dataset by the jackknife cross-validation test exhibited significant improvement compared with published results. Higher accuracy is observed with the proposed method named GPCR-SVMFS than GPCR-CA and GPCRPred.

Another algorithm has been developed by [Liao, Ju, & Zou, 2016] which uses the features from SVM-Prot [Y. H. Li et al., 2016] and Random forest algorithm to identify GPCRs from non-GPCRs with an accuracy of 91.61%.

Chapter 3

Materials and Methods

In this chapter we describe the construction of the dataset; GPCR and Non GPCRs. This is followed by a description of the features used for predicting and classifying the GPCRs and how the features have been analyzed. The last section is the description of the integrated algorithms used.

3.1 Collection of dataset

I constructed a non-redundant dataset consisting of GPCRs and non-GPCR protein sequences downloaded from the GPCR Prediction Ensemble database (GPCR-PEnDb) developed by Begum et al. (2018) to evaluate and train the classifiers for the GPCR prediction. I also downloaded transmembrane proteins that are not GPCRs with a CDHIT of 90% from Uniprot. A CDHIT of x% essentially produces a set of protein sequences that are at most (100-x)% similar to one another. The entire data collection include 2776 positive examples of confirmed GPCRs, and three sets of negative examples. The first two sets of negative examples contain 2179 transmembrane (TM) non-GPCRs, 1602 non-TM non-GPCRs respectively. The third set is the union of the previous two sets, resulting in a mix of 3781 TM and non-TM non-GPCR proteins in total.

3.2 Feature extraction

In order to predict and classify GPCRs, different sets of features are used by different algorithms to measure the accuracies of the methods as well as the importance of those

features in increasing performance. It is therefore imperative to extract features of each observation (sequence) in the dataset. The objective here is to identify distinctive features used in different studies and later generate them for our dataset using a suitable programming language, Python. Python programming language was used to generate the "CSV" (comma separated values) file that contains the sequence IDs and percentages.

The sequence-derived structural and physicochemical features such as hydrophobicity scales, average flexibility indices, polarizability parameter etc are highly useful for representing and distinguishing proteins or peptides of different structural, functional and interaction profiles. Currently, these structural and physicochemical features of proteins and peptides were routinely used to characterize target proteins in drugtarget pairs and predict new drugtarget associations to identify potential drug targets (He et al., 2010), following the spirit of chemogenomics.

Several programs for computing protein structural and physicochemical features have been developed (Du et al., 2012; Holland et al., 2008; Li et al., 2006); however, they are not comprehensive and can only be limited to a certain kind of features. Additionally, these are not freely and easily accessible.

Liang et al. (2013) implemented a selection of sophisticated protein features and provided them as a package for the free and open source software environment Python. The Propy package aims at providing the user with comprehensive implementations of these descriptors in a unified framework to allow easy and transparent computation. Propy is the first open source package computing a large number of protein features based on user-defined structural and physicochemical properties. From the propy package, four groups of features were used. In total, there are 1355 features.

Propy package description

The propy package can compute a large number of structural and physicochemical properties (used to capture as much information of protein sequences as possible) from amino acid sequence (Cao et al., 2013).The features used can be divided into four groups,

each of which has been independently predicting protein- and peptide-related problems by using machine-learning methods.

1. The first group includes two features, amino acid composition and, dipeptide composition, with two descriptors and 420 descriptor values.

2. The second group consists of three different autocorrelation features: normalized Moreau-Broto autocorrelation, Moran autocorrelation and Geary autocorrelation. The autocorrelation features describe the level of correlation between two protein or peptide sequences in terms of their specific structural or physicochemical property. Each of these features has eight descriptors and 240 descriptor values.

3. The third group includes two sequence-order feature sets, one is sequence-order-coupling number with two descriptors and 90 descriptor values, and the other is quasi-sequence-order with two descriptors and 100 descriptor values. These features are derived from both SchneiderWrede physicochemical distance matrix and Grantham chemical distance matrix.

4. The fourth group contains pseudo- amino acid compositions with 25 descriptor values. Eight types of physicochemical properties have been used for calculating these features.

The propy package contains several functions and modules manipulating proteins and peptides. To obtain protein sequences easily, propy provides a download module, by which the user could easily get protein sequences from the Uniprot website by providing Uniprot IDs or a file containing Uniprot IDs. A check module (Procheck) is also provided to ensure whether an input protein is a valid amino acid sequence or not. That is, if it has no special characters (X, B, Z, J, ϕ, ω, ψ, π, ζ, +, -). To facilitate the accessibility of the property or distance matrix of amino acids, propy provides an AAIndex module, which helps the user automatically download the needed property from the AAindex database. There are two means to compute these structural and physicochemical features from protein or peptide

sequences. One is to use the built-in modules in the propy package. The instruction for each module is provided in the form of HTML in propy. We could import related functions to compute these features as needed. The other is to call the GetProDes class by importing the PyPro module, which encapsulates commonly used descriptor calculation methods. I constructed both online and offline GetProDes object (Appendix A) with a protein sequence input, which calls corresponding methods to calculate these features.

Additionally, the main advantage of propy is that the users themselves could specify some sets of amino acid properties in the form of dictionary (a data structure in python). More conveniently, the output from the AAIndex module could be directly used as the user-defined property to calculate the aforementioned descriptors, greatly enlarging the applications to our calculated features. A list of physicochemical properties for feature extractions covered by the current version of propy is summarized in Table below.

Table 3.1: The physicochemical properties of the amino acids

Order	Physicochemical property	Range of property
1	Hydrophobicity scales	[-1.14, 1.81]
2	Average flexibility indices	[0.295, 0.544]
3	Polarizability parameter	[0, 0.409]
4	Free energy of solution in water	[-2.24, 4.91]
5	Residue accessible surface area in tripeptide	[75, 255]
6	Residue volume	[36.3, 135.4]
7	Steric parameter	[0, 1.02]
8	Relative mutability	[18, 134]

To this end, lets investigate the features, where the parameters are set to the same values as in (Li et al., 2010).

3.2.1 Amino acid composition (AAC) and dipeptide composition (DC)

Amino acid composition is defined as the occurrence frequencies of 20 amino acids in a protein sequence. That is,

$$f_A(i) = \frac{n_A(i)}{L},$$

where each $i = 1, 2, ..., 20$ corresponds to a distinct amino acid and $n_A(i)$ is the number of amino acid i occurring in the protein sequence of length L. Amino acid composition was widely used to transform GPCR sequences into 20-dimension numerical vectors (Chou KC et. al). However, the sequence order information would be completely lost. In order to address this issue, dipeptide composition was proposed to represent GPCR sequences by 400-dimension vectors, which capture local-order information and have been reported to improve classifications. Similarly, dipeptide composition is defined as the occurrence frequencies of the 400 dipeptides (i.e., 400 amino acid pairs). That is,

$$f_D(i) = \frac{n_D(i)}{L-1},$$

where each $i = 1, 2, ..., 400$ corresponds to one of the 400 dipeptides and $n_D(i)$ is the number of dipeptide i occurring in the sequence.

3.2.2 Autocorrelation descriptors (AD)

There are three autocorrelation descriptors normalized Moreau-Broto autocorrelation descriptors, Moran autocorrelation descriptors and Geary autocorrelation descriptors. They are all defined based on the value distributions of the eight physicochemical properties of amino acids along a protein sequence . The measurement values of these properties are first standardized before proceeding to calculate the autocorrelation descriptors. The standardization is performed as follows.

$$P(i) = \frac{P_0(i) - \bar{P}_0}{\sigma}$$

where $P_0(i)$ are the property value of the amino acid i,

$$\bar{P}_0 = \frac{1}{20}\sum_{i=1}^{20} P_0(i), \qquad \sigma = \sqrt{\frac{1}{20}\sum_{i=1}^{20}(P_0(i) - \bar{P}_0)^2}$$

Normalized Moreau-Broto autocorrelation descriptors are defined as:

$$NMBA(d) = \frac{MBA(d)}{L-d} \qquad d = 1, 2, 3, ...30$$

where

$$MBA(d) = \sum_{i=1}^{L-d} P(R_i)P(R_{i+d}),$$

R_i and R_{i+d} are the amino acids at position i and $i+d$ along the protein sequence, respectively.

Moran autocorrelation descriptors are defined as:

$$MA(d) = \frac{\frac{1}{L-d}\sum_{i=1}^{L-d}(P(R_i) - \bar{P})(P(R_{i+d}) - \bar{P})}{\frac{1}{L}\sum_{i=1}^{L}(P(R_i) - \bar{P})^2} \qquad d = 1, 2, 3, ...30$$

where $\bar{P}_0 = \frac{1}{L}\sum_{i=1}^{L} P(R_i)$ is the average value of the property of interest along the sequence.

Geary autocorrelation descriptors are defined as:

$$MA(d) = \frac{\frac{1}{2(L-d)}\sum_{i=1}^{L-d}(P(R_i) - P(R_{i+d}))^2}{\frac{1}{L-1}\sum_{i=1}^{L}(P(R_i) - \bar{P})^2} \qquad d = 1, 2, 3, ...30$$

3.2.3 Pseudo amino acid composition (PseAAC)

This set of features were originally developed by Chou and have been used widely to predict various attributes of proteins, such as outer membrane protein (Cai et al. 2003),

nuclear receptors (Gao et al., 2006), and protein structural classes (Xiao et al. 2005). The pseudo amino acid composition descriptors are defined similarly as the quasi-sequence order descriptors. The difference lies in the coupling number $\tau(j)$, which is modified to:

$$\theta(d) = \frac{1}{L-1}\sum_{i=1}^{L-d}\Theta(R_i, R_{i+d}), \qquad d = 1, 2, ...30$$

where $\Theta(R_i, R_{i+d})$ is the dth-tier correlation factor that reflects the sequence order correlation between all the most contiguous residues along a protein chain. It is defined as:

$$\Theta(R_i, R_{i+d}) = \frac{1}{3}\sum_{k=1}^{3}[H_k(R_i) - H_k(R_{i+d})]^2$$

where $H_1(R_i)$, $H_2(R_i)$ and $H_3(R_i)$ are the hydrophobicity, hydrophilicity, and side-chain mass of amino acid, respectively (Chou KC: Prediction of protein cellular attributes using pseudo amino acid composition). Their original values are standardized. Finally, the Chous pseudo amino acid composition descriptors are defined as:

$$PseAAC(i) = \begin{cases} \frac{f_A(i)}{\sum_{j=1}^{20} f_A(i) + \omega \sum_{d=1}^{30}\theta(d)} & 1 \le i \le 20 \\ \frac{\omega\theta(i)}{\sum_{j=1}^{20} f_A(i) + \sum_{d=1}^{30}\theta(d)} & 21 \le i \le 50 \end{cases}$$

where ω is a weighting factor (default $\omega = 0.1$).

3.2.4 Sequence-Order descriptors (SD)

In deriving the sequence-order descriptors, we depend on two distance measures for amino acid pairs which are the Grantham chemical distance matrix (Li et al., 2006), and the other called the Schneider-Wrede physicochemical distance matrix. Then, the jth-rank sequence-order coupling number is defined as:

$$\tau(j) = \sum_{i=1}^{L-j}(d(R_i, R_{i+j}))^2, \qquad j = 1, 2, ...30$$

where $d(R_i, R_{i+j})$ is one of the above distances between the two amino acids R_i and R_{i+j} located at position i and $i + j$, respectively.

The quasi-sequence-order descriptors are defined as:

$$QSO(i) = \begin{cases} \frac{f_A(i)}{\sum_{j=1}^{20} f_A(j) + \omega \sum_{j=1}^{30} \tau(j)} & 1 \leq i \leq 20 \\ \frac{\omega \tau(i)}{\sum_{j=1}^{20} f_A(j) + \sum_{j=1}^{30} \tau(j)} & 21 \leq i \leq 50 \end{cases}$$

where ω is a weighting factor (default $= 0.1$). We end up with 90 ($= 30 \times 3$) sequence-order-coupling numbers and 100 ($= 50 \times 2$) quasi-sequence-order descriptors. In total, there are 190 features extracted from the sequence-order descriptors

3.3 Methods

The three integrated methods are;

- PCA & intimate sorting algorithm
- Univariate Feature Selection, PCA & intimate sorting algorithm
- PCA & Random Forest

.

3.3.1 Intimate sorting algorithm (ISA)

In this study, we use a simple yet powerful algorithm called intimate sorting (Chou KC et. al.). This algorithm is easy to implement and does not need to set any parameters as some other algorithms (e.g., support vector machines).

Suppose that a training set consists of N proteins P_1, P_2, ..., P_N, each of which P_i is a λ-dimension vector, $\mathbf{P_i}$= $(p_{i,1}, p_{i,2}, \ . \ . \ ., p_{i,\lambda})^T$. The GPCR classes of these proteins are already known, and each protein belongs to exactly one of the μ classes. The intimate sorting algorithm aims to place a query protein $\mathbf{P}$= $(p_1, p_2, \ . \ . \ ., p_\lambda)^T$ into one of the μ classes based on the information from the N proteins in the training set. To this end, a measure of similarity score between P and P_i is defined as

$$\Phi(P, P_i) = \frac{P.P_i}{||P||||P_i||}, \qquad\qquad i = 1, 2...N$$

where P.P_i= $\sum_{j=1}^{\lambda} p_j.p_{i,j}$ and $||P||$= $\sqrt{\sum_{j=1}^{\lambda} p_j^2}$. When P $\equiv$ P_i, it can be easily seen that $\Phi(P, P_i)$= 1, suggesting that they are most likely to belong to the same class. In general, we have -1$\leq\Phi(P, P_i)\leq$1. The higher the $\Phi(P, P_i)$ value, the more likely two proteins belong to the same class. Among the N proteins in the training set, the one with the highest score with the query protein P is picked out, which we denote by $\mathbf{P_k}$, k $\in$ [1, N]. If there is a tie, we would randomly select one of them. In the final step, the intimate sorting algorithm simply assigns P into the same GPCR class as $\mathbf{P_k}$.

3.3.2 PCA with ISA

This method proposed by Peng et al., (2010) uses the PCA to reduce the dimension then later apply the IS algorithm on the principal components.

3.3.3 Univariate feature selection, PCA and ISA

A good feature subset can be defined as one that contains features highly correlated with (predictive of) outcome, yet uncorrelated (independent) with (not predictive of) each other. Nevertheless, the existing diversity of Feature Selection (FS) methods makes it challenging to choose the correct one for the task at hand. Firstly, we consider the univariate feature selection which is known as univariate correlation filtering, this step removes all features

that are not directly related to their class variables. Filter methods are generally used as a preprocessing step. The built-in function SelectKBest class just scores the features using a function (in this case f_classif which uses the statistical method ANOVA to test whether the means of several groups are equal or not) and then "removes all but the k highest scoring features".

In addition to the univariate filter approach, I implemented and studied the use of PCA as a multivariate filter to reduce further the number of features. PCA reduces the dimensionality of the data while retaining most of the variation in the predictor variables. Thus, by using a few components, PCA can represent each sample by using relatively few (new) variables instead of (potentially) thousands of them. The number of the latent features (factors) can be much lower than the number of original features, so that the data can be visualized in a much lower-dimensional space. The final stage is the application of the ISA model.

3.3.4 PCA with random forest (RF)

In this method, pca is first implemented to have our reduced set of variables called pcs then the random forest is applied as the classifier.

3.4 Comparative measures and assessment procedure for prediction accuracy

3.4.1 Comparative measures

Table 3.2: Comparative Measures

		Predicted		
		GPCR	Non-GPCR	**Total**
Actual	GPCR	True Positive (TP)	False Negative (FN)	$TP + FN$
	Non-GPCR	False Positive (FP)	True Negative (TN)	$FP + TN$
	Total	TP+FP	FN+TN	P+N

1. Condition Positive (P) is the number of real positive cases in the data.

2. Condition Negative (N) is the number of real negative cases in the data.

3. True positive (TP) is the number of samples that were predicted as GPCRs (positives) and were actually (truly) GPCRs which is equivalent with the hit.

4. False negative (FN) is the number of samples that were predicted as Non-GPCRs (negatives) but were actually (falsely) GPCRs which is equivalent with Type II error.

5. False positive (FP) is the number of samples that were predicted as GPCRs (positives) but were actually (falsely) Non-GPCRs which is equivalent with false alarm or Type I error. The ratio of false positives to the sum of FP and TN is false positive rate (FPR)

$$\mathbf{FPR} = \frac{\mathbf{FP}}{\mathbf{FP + TN}} \times 100\%$$

6. True negative (TN) is the number that were predicted as Non-GPCRs and actually (Truly) Non-GPCRs which is equivalent with correct rejections.

7. Sensitivity also known as recall, hit rate or true positive rate (TPR) is given by;

$$\textbf{Sensitivity} = \frac{\textbf{TP}}{\textbf{TP + FN}} \times 100\%$$

8. Specificity also known as selectivity or true negative rate (TNR) is given by;

$$\textbf{Specificity} = \frac{\textbf{TN}}{\textbf{TN + FP}} \times 100\%$$

9.

$$\textbf{Accuracy} = \frac{\textbf{TP + TN}}{\textbf{P + N}} \times 100\%$$

3.4.2 Accuracy assessment procedure

In assessing the accuracy, the data is randomly partitioned into different sets of training and test within a for-loop, in which case the algorithms are being run and then accuracy based on each test set produced. A loop produces a random partition of training and test data. Each training set and test set were randomly chosen to be 70% and 30% of the data sequences respectively. I finally took average of all the accuracy results from the different test sets. The number of iterations is user-defined but I did set mine to 10 which means 10 different sets of random partitions.

Chapter 4

Results and Discussion

In this chapter we have the datasets, results of the features collected and overall results of the various integrated Machine learning methods that have been achieved from the different feature extraction, classification strategies, and the results obtained by analyzing the features have been discussed.

4.1 Dataset

Below is a table of the proteins that we have collected in our dataset. The GPCRs are distributed in into these six major classes, A, B, C, D, E, and F based on sequence homology and functional similarity according to the IUPHAR classification system (Horn et al. 2003). They are highly dominated by class A which is almost 85% of all the GPCRs collected followed by class B which is 7%. The least represented is class E which is just 0.4%. Now the Non-GPCRs are in two parts, we have the transmembranes (TM) which are not characterised by seven-helix-bundled structure embedded in the cell membrane and other protein sequences which are not transmembranes.

Table 4.1: Non-redundant dataset

Proteins	Family	Number of sequences
GPCR	Class A	2358
	Class B	203
	Class C	112
	Class D	13
	Class E	11
	Class F	79
Non TM Non-GPCR		1602
TM Non-GPCR		2179

4.2 Features collected

Here we have a list of 1355 propy computed features for each protein sequence in our dataset and a brief description of the features can be seen in section 3.2. When tested with the PCA-ISA algorithm, the order of importance among the four groups of features as shown in Table 4.2 are sequence composition, pseudo amino acid composition, quasi-sequence order composition, and autocorrelation composition. Applying PCA-ISA on just the sequence composition features produce an overall accuracy of 97% with a false positive rate (FPR) around 6% among non-TM non-GPCR's and when the pseudo amino acid composition is added, there is an improvement in both the FPR (lowered) and accuracy (increased). Now using the single amino acid with just the pseudo amino compositions produce an accuracy of around 98% and further decrease the FPR to 4%. This then explains further how a loss in sequence-oder information in the calculation of amino acid which is being accounted for by the pseudo amino acid plays an important role. In that sense, the performance of pseudo amino acid with dipeptide is better than just amino acid with dipeptide. The negative samples used in these analysis are non TM nonGPCRs.

Table 4.2: Feature groups and number of descriptors

Feature groups	Features	No. of descriptors
Sequence compositions	Amino acid composition	20
	Dipeptide composition	400
Pseudo amino acid composition	Pseudo amino acid	25
Quasi-sequence order	Sequence order coupling number	90
	Quasi-sequence order descriptors	100
Autocorrelation	Normalized Moreau-Broto autocorrelation	240
	Moran autocorrelation	240
	Geary autocorrelation	240

4.3 Prediction accuracies

Here we display the results of all the predicting methods in a table for easy comparisons.

4.3.1 Predicting GPCR vs TM Non-GPCR

We will first start with the prediction of whether a given sequence is GPCR or non-TM non-GPCR using all the classification proposed methods. In all, there was a total of 4955 examples of which 30% (1487) of the entire data was used as testing set.

Table 4.3: GPCR vs TM Non-GPCR

Methods	**Sensitivity (%)**	**Specificity (%)**	**False Positive Rate(%)**	**Accuracy (%)**
ISA	90.15	86	14	88.01
RF	91.20	85.50	14.50	88.15
PCA & ISA	97.73	88.52	11.48	93.73
PCA & ISA &UFS	98	89.74	10.26	94.43
PCA & RF	97.98	91.93	8.07	95.36

4.3.2 Predicting GPCR vs Non-TM Non-GPCR

This is the results from combining GPCRs with non-TM non-GPCR and predicting whether a given protein sequence is a GPCR. There was a total of 4378 examples of which 30% (1314) of the entire data was used as testing set.

Table 4.4: GPCR vs Non-TM Non-GPCR

Methods	**Sensitivity (%)**	**Specificity (%)**	**False Positive Rate(%)**	**Accuracy (%)**
ISA	93	90.15	9.85	91.88
RF	92	89.10	10.9	90.5
PCA & ISA	99.29	95.09	4.91	97.81
PCA & ISA &UFS	99.29	96.75	3.25	98.45
PCA & RF	99.28	95	5	97.72

4.3.3 Predicting GPCR vs Mixed TM & Non-TM Non-GPCR

This is the results from combining GPCRs with TM non-GPCRs and non-TM non-GPCRs then predicting whether a given protein sequence is a GPCR. There was a total of 6557 samples with 30% (1967) of the entire dataset been set aside as testing set.

Table 4.5: GPCR vs Mixed TM & Non-TM Non-GPCR

Methods	**Sensitivity (%)**	**Specificity (%)**	**False Positive Rate(%)**	**Accuracy (%)**
ISA	90	87	13	88.25
RF	91.07	88.73	11.27	89.37
PCA & ISA	98.03	92.81	7.19	95.07
PCA & ISA &UFS	98.59	93.0	7.0	95.43
PCA & RF	93.96	95.90	4.1	95.06

From the tables above, we notice that the integrated algorithms performed better than

the individual classifiers in terms of overall accuracies and false positive rates.That is the false positive rate ranges from 3 to 5% in terms of Non-TM Non-GPCR and from 9 to 12% in terms of TM Non-GPCR. We also notice that, using TM non-GPCRs as negative examples had the worst performance which is an indication that the methods most times predicts the negative examples as positives. In as much as the difference is not so great, in demonstrating the superior performance, we made comparisons with a number of existing online web-servers which had a different dataset from what I trained with. A balanced dataset was used to test the performance of these web-servers and the results showed that;

1. Web-servers with greater than 90% overall accuracy were TMHMM, GPCRPred, and GPCR-GIA and all other servers had accuracy between 83% and 90%.

2. Sensitivity ranged from 68 to 100%, specificity from 73 to 98%, positive predictive value from 79 to 98%, and negative predictive value from 76 to 100%.

3. False positive rate for transmembrane non-GPCR ranged from 10 (PredGPCR) to 100% (PCAGPCR), with average difference between transmembrane non-GPCRs and all non-GPCRs at 31%. Accuracy for GPCR family classification varied from 60 to 81% due to each web-servers variation of their use of the IUPHAR system.

Chapter 5

Conclusion and Future Work

5.1 Conclusion based on results to date

The experimental results show that UFS, PCA and ISA integrated method had averagely the lowest false positive rate then followed by PCA & RF which did great with TM proteins. This shows that, integrating some unsupervised algorithms with the classifiers (supervised) helps in improving the overall accuracies (both in sensitivities and specificities) leading to a reduced false positive rate. It is able to distinguish transmembrane protein sequences which are actually GPCRs.

There are existing computational tools of GPCR prediction that involve principal component analysis (PCA), intimate sorting (IS), support vector machine, and random forest (RF) techniques using various sequence derived features. While accuracies of over 90% were reported for their own test datasets, the capabilities in distinguishing GPCRs from transmembrane non-GPCRs had not been measured in any of these tools.

5.2 Future research goals and proposed approaches

The specific aims of my dissertation research are:

1. Survey and assess the available neural network (NN) and support vector machine (SVM) type approaches.

   In addition to the PCA, ISA, RF, and UFS described in the previous chapter, NN and SVM approaches are also popularly used in GPCR prediction (Seo et al 2018, Nievola

et al. 2016, Ren et al 2013) . I plan to extend my assessment of GPCR prediction approaches to include them in the comparison, using the same dataset, which contains TM non-GPCRs among the negative examples, as that used for the PCA, ISA, RF, and UFS approaches described earlier. Based on this more exhaustive comparison, we will design the best integrated statistical and machine learning algorithm that can produce the highest GPCR prediction accuracy overall.

2. Address the GPCR family and subfamily classification problem.

   There are six families named Class A - F in the IUPHAR system. Preliminary results obtained in our group [41] on classification accuracies of currently available web servers are mostly in the 60 - 80% range (although higher accuracies are observed in Class A GPCRs). This is likely caused by the biased data used in training the classification algorithms, the vast majority of which are Class A GPCRs while examples in Classes D and E are scanty. Using the examples in our current GPCR collection, I will create more balanced datasets for training and testing. This may involve using only a subset instead of the entire collection of Class A GPCRs from our dataset, and possible grouping of all the Class D, E, and F GPCRs into one category. Then I will explore a few multinomial classification schemes including ensemble random forest, multinomial logistic regression, and penalized regression models to help design an integrated classification algorithm to optimize the classification accuracies. Finally, I will investigate the possibilities of further improving the classification accuracies by including class-specific sequence motifs and loop length characteristics as sequence features in our classification scheme.

3. Software implementation and distribution.

   Once we have decided on what the final integrated algorithm is, we will implement it using Python, along with the Propy and R packages. It will be connected to the website gpcr.utep.edu via Webpy and distributed as part of the web-based GPCR-PEn package on the UTEP bioinformatics computer network so that it can be accessed

by GPCR researchers worldwide. I also plan to upload the source code of my program to GitHub so that it can be modified by other computational scientists for further improvements or adapted to be used for other purposes.

5.3 Expected results and possible pitfalls

For specific aim 1, we expect that the false positive rate for each individual approach among the TM non-GPCRs will be substantially reduced by introducing such proteins into the training dataset. Overall prediction accuracies of the different approaches are anticipated to stay above 90%. While there may not be any large variations among the different approaches, it would be reasonable to expect that the highest accuracy is attained by one of the integrated algorithms.

In specific aim 2, I will have to implement several multinomial classification schemes. While these general approaches have been established for some time, whether they will perform well on GPCR prediction has yet to be demonstrated. If their accuracies turned out to be less than desirable, we may have to return to binary classification techniques such as those that were used in GPCR prediction and iteratively carry out binary classifications at several levels. We can first classify a GPCR to be either Class A or not Class A. For those that are not in Class A, we move on to the next level of classifying each to be in Class B or not Class B, etc.

Once we have identified one or more classification schemes that produce sufficiently high prediction accuracy, we can link the program to the gpcr.utep.edu webpage maintained by our group so that the programs can be used by others. It is expected that when more people use the program, errors will be detected and these need to be continuously corrected for some time before the program becomes bug-free and be formally published.

5.4 Timeline

The following table shows the tentative timeline for the research goals to be completed

Table 5.1: Timeline for the completion of research goals

Date	**Tasks to complete**
January 2019 - May 2019	Survey and assess the available neural network (NN) and support vector machine (SVM)
June 2019 - August 2019	Address the GPCR family and subfamily classification problem
September 2019 - December 2019	Software implementation and distribution.
January 2020 - May 2020	Finalize Ph.D. Dissertation and prepare for defense

References

[1] Attwood,T.K. (2001) A compendium of specific motifs for diagnosing GPCR subtypes. Pharmacol Sci., 22,162-5.

[2] Begum, K., Mohl, J. E., and Leung, M.-Y., Sixteenth Annual Rocky Mountain Bioinformatics Conference 2018, Aspen/Snowmass, Colorado, "GPCR-PEnDB: A database of protein sequences and derived features to facilitate prediction and classification of G protein-coupled receptors," International Society for Computational Biology. (December 6, 2018).

[3] Bhasin, M., & Raghava, G. P. S. (2004). GPCRpred: An SVM-based method for prediction of families and subfamilies of G-protein coupled receptors. Nucleic Acids Research, 32(WEB SERVER ISS.), 383389. https://doi.org/10.1093/nar/gkh416

[4] Bryson-Richardson, Darren Logan, Peter David Currie, Ian J Jackson. (2004). Large-scale analysis of gene structure in rhodopsin-like GPCRs: Evidence for widespread loss of an ancient intron.

[5] Cai, Y., Zhou, G. (2003). Support Vector Machines for Predicting Membrane Protein Types by Using Functional Domain Composition, 84(5): 32573263. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1302886/

[6] Chou, K.-C., & Elrod, D. W. (1999). Protein subcellular location prediction. Protein Engineering, Design and Selection, 12(2), 107118. https://doi.org/10.1093/protein/12.2.107

[7] Chou, K.-C. (2001). Prediction of protein cellular attributes using pseudo-amino acid composition. Proteins: Structure, Function, and Genetics, 43(3), 246255. https://doi.org/10.1002/prot.1035

[8] Chou KC (2005a). Prediction of G-protein-coupled receptor classes. J. Proteome Res., 2005, 4 (4), pp 14131418

[9] Cobanoglu, M. C., Saygin, Y., & Sezerman, U. (2011). Classification of GPCRs using family specific motifs. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 8(6), 14951508. https://doi.org/10.1109/TCBB.2010.101

[10] Davies, M. N., Secker, A., Halling-Brown, M., Moss, D. S., Freitas, A. a, Timmis, J., Flower, D. R. (2008). GPCRTree: online hierarchical classification of GPCR function. BMC Research Notes, 1, 67. https://doi.org/10.1186/1756-0500-1-67

[11] Eilers, M., Hornak, V., Smith, S. O., & Konopka, J. B. (2005). Comparison of class A and D G protein-coupled receptors: common features in structure and activation. Biochemistry, 44(25), 895975. https://doi.org/10.1021/bi047316u

[12] Elrod, D. W., & Chou, K. C. (2002). A study on the correlation of G-protein-coupled receptor types with amino acid composition. Protein Eng, 15(9), 713715.

[13] Frank,E. and Witten,I.H. (1998) Generating Accurate Rule Sets without Global Optimization. Fifteenth International Conference on Machine Learning

[14] Fredriksson, R. et al (2003). The G-protein-coupled receptors in the human genome form five main families. Phylogenetic analysis, paralogon groups, and fingerprints. Molecular Pharmacology;63(6):1256-72. https://www.ncbi.nlm.nih.gov/pubmed/12761335

[15] Fridmanis, D. et al. (2006) Formation of new genes explains lower intron density in mammalian Rhodopsin G protein-coupled receptors. Mol Phylogenet Evol., 43, 864-80.

[16] Gao, Q. Bin, & Wang, Z. Z. (2006). Classification of G-protein coupled receptors at four levels. Protein Engineering, Design and Selection, 19(11), 511516. https://doi.org/10.1093/protein/gzl038

[17] Gether,U. et al. (2002) Structural basis for activation of G-proteincoupled receptors. Pharmacol Toxicol., 91, 304-312.

[18] Gloriam, D.E. et al. (2005) Nine new human Rhodopsin family Gprotein coupled receptors: identification, sequence characterisation and evolutionary relationship. Biochim Biophys Acta, 1722, 235-46

[19] Goudet C, et al. Heptahelical domain of metabotropic glutamate receptor 5 behaves like rhodopsin-like receptors. Proc Natl Acad Sci U S A. 2004;101:378383

[20] Guerrero, F. D., Kellogg, A., Ogrey, A. N., Heekin, A. M., Barrero, R., Bellgard, M. I., Leung, M.-Y. (2016). Prediction of G protein-coupled receptor encoding sequences from the synganglion transcriptome of the cattle tick, Rhipicephalus microplus. Ticks and Tick-Borne Diseases, 7(5), 670677. https://doi.org/10.1016/j.ttbdis.2016.02.014

[21] Guo, Y. Z., Li, M., Lu, M., Wen, Z., Wang, K., Li, G., & Wu, J. (2006). Classifying G protein-coupled receptors and nuclear receptors on the basis of protein power spectrum from fast Fourier transform. Amino Acids, 30(4), 397402. https://doi.org/10.1007/s00726-006-0332-z

[22] Guo,Y.Z. et al. (2005) Fast fourier transform-based support vector machine for prediction of G-protein coupled receptor subfamilies Acta Biochim Biophys Sin (Shanghai), 37, 759-66.

[23] Hamann, J., Aust, G., Arac, D., Engel, F. B., Formstone, C., Fredriksson, R., Schioth, H. B. (2015). International Union of Basic and Clinical Pharmacology. XCIV. Adhesion G Protein-Coupled Receptors. Pharmacological Reviews, 67(2), 338367. https://doi.org/10.1124/pr.114.009647

[24] Hebert,T.E. and Bouvier,M. (1998) Structural and functional aspects of G protein-coupled receptor oligomerization. Biochem Cell Biol. 76, 1-11

[25] Horn, F. (2003). GPCRDB information system for G protein-coupled receptors. Nucleic Acids Research, 31(1), 294297. https://doi.org/10.1093/nar/gkg103

[26] Horn, F., Weare, J., Beukers, M. W., Hrsch, S., Bairoch, A., Chen, W., Vriend, G. (1998). GPCRDB: an information system for G protein-coupled receptors. Nucleic Acids Research, 26(1), 275279. https://doi.org/10.1093/nar/26.1.275

[27] Huang, Y., Cai, J., Ji, L., & Li, Y. (2004). Classifying G-protein coupled receptors with bagging classification tree. Computational Biology and Chemistry, 28(4), 275280. https://doi.org/10.1016/j.compbiolchem.2004.08.001

[28] Inoue, Y., Ikeda, M.,& Shimizu, T. (2004). Proteome-wide classification and identification of mammalian-type GPCRs by binary topology pattern. Computational Biology and Chemistry, 28(1), 3949. https://doi.org/10.1016/j.compbiolchem.2003.11.003

[29] Iqbal, M. J., Faye, I., & Samir, B. B. (2016). Classification of GPCRs proteins using a statistical encoding method. Proceedings of the International Joint Conference on Neural Networks, 2016Octob, 12241228. https://doi.org/10.1109/IJCNN.2016.7727337

[30] Isberg, V., Mordalski, S., Munk, C., Rataj, K., Harpse, K., Hauser, A. S., Gloriam, D. E. (2016). GPCRdb: an information system for G protein-coupled receptors. Nucleic Acids Research, 44(D1), D356-64. https://doi.org/10.1093/nar/gkv1178

[31] Karchin,R. et al. (2002) Classifying G-protein coupled receptors with support vector machines. Bioinformatics, 18, 147-159.

[32] Keerthi,S.S. et al. (2001) Improvements to Platt's SMO Algorithm for SVM Classifier Design. Neural Computation, 13, 637-649.

[33] Klabunde,T. and Hessler,G. (2002) Drug design strategies for targeting G-protein-coupled receptors. ChemBioChem 2002, 3, 928 944.

[34] Langenhan, T., Aust, G.,& Hamann, J. (2013). Sticky signaling–adhesion class G protein-coupled receptors take the stage. Science Signaling, 6(276), re3. https://doi.org/10.1126/scisignal.2003825

[35] Li, Y. H., Xu, J. Y., Tao, L., Li, X. F., Li, S., Zeng, X., Chen, Y. Z. (2016). SVM-Prot 2016: A Web-Server for Machine Learning Prediction of Protein Functional Families from Sequence Irrespective of Similarity. PLOS ONE, 11(8), e0155290. https://doi.org/10.1371/journal.pone.0155290

[36] Li, Z., Zhou, X., Dai, Z., & Zou, X. (2010). Classification of G-protein coupled receptors based on support vector machine with maximum relevance minimum redundancy and genetic algorithm. BMC Bioinformatics, 11, 325. https://doi.org/10.1186/1471-2105-11-325

[37] Liao, Z., Ju, Y., & Zou, Q. (2016). Prediction of G Protein-Coupled Receptors with SVM-Prot Features and Random Forest. Scientifica, 2016, 110. https://doi.org/10.1155/2016/8309253

[38] Liu, Y., An, S., Ward, R., Yang, Y., Guo, X.-X., Li, W., & Xu, T.-R. (2016). G protein-coupled receptors as promising cancer targets. Cancer Letters, 376(2), 226239. https://doi.org/10.1016/j.canlet.2016.03.031

[39] Li ZR, et al. PROFEAT: a web server for computing structural and physicochemical features of proteins and peptides from amino acid sequence, Nucleic Acids Res. , 2006, vol. 34 (pg. W32-W37)

[40] Naveed, M., & Khan, A. U. (2012). GPCR-MPredictor: Multi-level prediction of G protein-coupled receptors using genetic ensemble. Amino Acids, 42(5), 18091823. https://doi.org/10.1007/s00726-011-0902-6

[41] Palczewski, K., Kumasaka, T., Hori, T., Behnke, C. A., Motoshima, H., Fox, B. A.,

Miyano, M. (2000). Crystal Structure of Rhodopsin: A G Protein-Coupled Receptor. Science, 289(5480), 739745. https://doi.org/10.1126/science.289.5480.739

[42] Pearson, W. R., & Lipman, D. J. (1988). Improved tools for biological sequence comparison. Proceedings of the National Academy of Sciences of the United States of America, 85(8), 24448. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/3162770

[43] Peng, Z.-L., Yang, J.-Y., & Chen, X. (2010). An improved classification of G-protein-coupled receptors using sequence-derived features. BMC Bioinformatics, 11(1), 420. https://doi.org/10.1186/1471-2105-11-420

[44] Perez,E., Begum K, Mohl, J., & Leung, M., (2018). Assessment of web-based G protein-coupled receptor prediction and classification bioinformatics tools. 23rd Joint UTEP/NMSU Conference on Mathematics, Computer Science and Computational Sciences. http://www.cs.utep.edu/vladik/utepnmsu18.html

[45] Poyner, D. R., & Hay, D. L. (2012). Secretin family (Class B) G protein-coupled receptors - from molecular to clinical perspectives. British Journal of Pharmacology, 166(1), 13. https://doi.org/10.1111/j.1476-5381.2011.01810.x

[46] Qian, B., Soyer, O. S., Neubig, R. R., & Goldstein, R. A. (2003). Depicting a proteins two faces: GPCR classification by phylogenetic tree-based HMMs. FEBS Letters, 554(12), 9599. https://doi.org/10.1016/S0014-5793(03)01112-8

[47] Raisley, B., Zhang, M., Hereld, D., & Hadwiger, J. A. (2004). A cAMP receptor-like G protein-coupled receptor with roles in growth regulation and development. Developmental Biology, 265(2), 43345. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/14732403

[48] Sahin, M. E., Can, T., & Son, C. D. (2014). GPCRsort-responding to the next generation sequencing data challenge: prediction of G protein-coupled receptor classes

using only structural region lengths. Omics: A Journal of Integrative Biology, 18(10), 636644. https://doi.org/10.1089/omi.2014.0073

[49] Spiegel,A.M., Shenker,A. and Weinstein,L.S. (1992)Endocr. Rev., 13, 536565

[50] Strader,C.D., Fong,T.M., Tota,M.R. and Underwood,D. (1994) Annu. Rev. Biochem., 63, 101132.

[51] Xiao,X., Shao,S., Ding,Y., Huang,Z., Huang,Y. and Chou,K.C. (2005) Amino Acids, 28, 5761.

Chapter 6

Appendices

Appendix A - Script for Extracting various descriptors

```
import pandas as pd
from propy.GetProteinFromUniprot import GetProteinSequence as gps
from propy.PyPro import GetProDes

data = pd.read_csv('Final_correected_GPCR_TransNOnGPCR.csv')
protein_id = data['ID']
protein_id_list = list(protein_id)

result = []
protein_id_error_index=[]
for idx,pro_id in enumerate(protein_id_list):
print(idx)
protein_sequence = gps(pro_id)
descriptor = GetProDes(protein_sequence)
try:
PAAC= descriptor.GetPAAC()
#SOCN=descriptor.GetSOCN()
#PAAC = descriptor.GetPAAC(lamda=5,weight=0.1)
except Exception as e:
# raise e
print('Error occurred finding the protein descriptors for '+pro_id)
```

```
protein_id_error_index.append(idx)
continue
result.append(PAAC)
protein_id = protein_id.drop(labels=protein_id_error_index).
reset_index()
pd.concat([protein_id,pd.DataFrame(result)],axis=1).
drop(labels=['index'],axis=1).to_csv('GPCR_TRANS_PAAC.csv')
pd.DataFrame({'Protein_id_error':protein_id_error_index}).
to_csv('GPCR_TRANS_PAACerror1.csv')
```

Appendix B - UFS, PCA & ISA

```
import pandas as pd
import numpy as np
import random as rand
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from random import choice, seed
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.feature_selection import chi2

##dataset = pd.read_csv('gpcr_nongpcr_new.csv', header=None)

np.random.seed(75)
dataset = pd.read_csv('GSE4115.csv', header=None, skiprows=1)
 # The main dataset to be split into training and test
#X = dataset.iloc[:,:-1]
#1165 446 1357

array = dataset.values
X = array[:,1:-1]
Y = array[:,-1]
#X = dataset.iloc[:,1:1356]
#y = dataset.iloc[:,-1]

#randomly generate the rows to pick out for the trainingset and the
rest are used for the testset
```

```
# training_set , test_set , training_labels , test_labels =
train_test_split(X,Y , test_size=0.3,
#
random_state=42)
# #print(test_set)
# # Saving the trainingset and testset
# pd.concat([training_set , training_labels], axis=1).to_csv
('Training_Set.csv', header=0) # the first column of this
 file contains the indices showing how random the rows were selected
# pd.concat([test_set , test_labels], axis=1).
to_csv('Test_Set.csv', header=0)

# Feature Extraction with Univariate Statistical Tests
(Chi-squared for classification)

# load data

# array = dataframe.values
# X = array[:,0:8]
# Y = array[:,8]
# feature extraction

test = SelectKBest(score_func=f_classif , k=700)
fit = test.fit(X, Y)
# summarize scores
np.set_printoptions(precision=3)
print(fit.scores_)
features = fit.transform(X)
```

```
#ft=np.concatenate((features,Y,axis=1)
# summarize selected features
print(features[0:5,:])
np.savetxt('features_set.csv', features,'%s',',')

#randomly generate the rows to pick out for the trainingset
and the rest are used for the testset
training_set,test_set,training_labels,test_labels =
train_test_split(features,Y ,test_size=0.3,
random_state=75)
# #print(test_set)
# # Saving the trainingset and testset
training_labels = np.reshape(training_labels,
(training_labels.shape[0],1))
test_labels = np.reshape(test_labels,(test_labels.shape[0],1))
np.savetxt('Training_Set.csv',np.concatenate(
(training_set,training_labels),axis=1),'%s',',')
 # the first column of this file contains the indices
 showing how random the rows were selected
np.savetxt('Test_Set.csv',np.concatenate(
(test_set,test_labels),axis=1),
'%s',',')
# convert the two datasets(dataframe) into numpy arrays
testArrays = test_set
trainingArrays = training_set

#feature scaling
sc=StandardScaler()
```

```
trainingArrays=sc.fit_transform(trainingArrays)
#print(trainingArrays)
testArrays=sc.transform(testArrays)

#applying pca
pca = PCA(n_components =95)
trainingArrays = pca.fit_transform(trainingArrays)
#print(trainingArrays)
testArrays = pca.transform(testArrays)
explained_variance = pca.explained_variance_ratio_
#print(explained_variance)

predicted_class_list = []

for index, testRow in enumerate(testArrays):
# keeps track of the similarity scores for each row
(the row in the test dataset)
similarity_score_list = []
for row in trainingArrays:
norm = np.linalg  # class to access norm-2

# calculates the similarity score
simi_score = np.divide(np.dot(testRow, row),
np.multiply(norm.norm(testRow), norm.norm(row)))
similarity_score_list.append(simi_score)

index_maxvalue = similarity_score_list.
index(max(similarity_score_list))
```

```
class_maxvalue = training_labels[index_maxvalue]
 # gets the class of the maximum similarity score
predicted_class_list.append(class_maxvalue)

# print('\nThe test protein[',index+1,']
belongs to :',class_maxvalue)

# This adds the predicted classes to the last
#column in the testArrays
testArrays = np.concatenate((testArrays, test_labels,
np.array(predicted_class_list)), axis=1)

# Checking the accuracy
true_positives = 0  # variable to hold the
number of occurances where
 actual is equal to predicted
pred_classes_count_dict = {}
test_classes_count_dict = {}
for i in range(len(test_labels)):
testLabel = str(test_labels[i]).strip().lower()
# count the classes in test
if testLabel in test_classes_count_dict.keys():
test_classes_count_dict[testLabel] += 1
else:
test_classes_count_dict[testLabel] = 1

if testLabel == str(predicted_class_list[i]).strip().lower():
true_positives += 1
```

```
# count the predicted classes
if testLabel in pred_classes_count_dict.keys():
pred_classes_count_dict[testLabel] += 1
else:
pred_classes_count_dict[testLabel] = 1

for key, value in pred_classes_count_dict.items():
print('The Accuracy of class [', key, '], is :',
pred_classes_count_dict[key] /
 test_classes_count_dict[key] * 100)
print('The Overall Accuracy is :', true_positives /
len(testArrays) * 100)

# save the predictions made to a file
np.savetxt('Test_Set_predicted_VI.csv', testArrays, '%s', ',')
```

Appendix C - PCA & ISA

```
''' After successful running of this code the accuracy is printed
 to the console
and three files named as:
1. 'Training_Set.csv'
2. 'Test_Set.csv'
3. 'Test_Set_predicted.csv'  are produced'''

import pandas as pd
import numpy as np
import random as rand
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from random import choice, seed

np.random.seed(75)
dataset = pd.read_csv('GSE4115.csv', header=0)
 # The main dataset to be split into training and test
#X = dataset.iloc[:,:-1]
#1165 446 1357
X = dataset.iloc[:,:-1]
y = dataset.iloc[:,-1]

#randomly generate the rows to pick out for the training set
and the rest are used for the testset
training_set,test_set,training_labels,test_labels =
```

```
 train_test_split(X,y ,test_size=0.3,
random_state=75)

np.savetxt('Test_Set_check.csv', test_set,'%s',',')
#print(test_set)
# Saving the trainingset and testset
pd.concat([training_set,training_labels],axis=1).to_csv
('Training_Set.csv',header=0) # the first column of this file
 contains the indices showing how random the rows were selected
pd.concat([test_set,test_labels],axis=1).
to_csv('Test_Set.csv',header=0)

# convert the two datasets(dataframe) into numpy arrays
testArrays = test_set.iloc[:,1:].values
trainingArrays = training_set.iloc[:,1:].values

#feature scaling
sc=StandardScaler()
trainingArrays=sc.fit_transform(trainingArrays)
#print(trainingArrays)
testArrays=sc.transform(testArrays)

#applying pca
pca = PCA(n_components=400)
trainingArrays = pca.fit_transform(trainingArrays)
#print(trainingArrays)
testArrays = pca.transform(testArrays)
explained_variance = pca.explained_variance_ratio_
```

```
#print(explained_variance)


# this list holds the predicted classes for the test proteins
predicted_class_list=[]

for index,testRow in enumerate(testArrays):
# keeps track of the similarity scores for each row(the row in
 the test dataset)
similarity_score_list = []
for row in trainingArrays:
norm = np.linalg # class to access norm-2

# calculates the similarity score
simi_score = np.divide(np.dot(testRow,row),
np.multiply(norm.norm(testRow),norm.norm(row)))
similarity_score_list.append(simi_score)



index_maxvalue = similarity_score_list.
index(max(similarity_score_list))
class_maxvalue = training_labels.iloc[index_maxvalue] # gets the
 class of the maximum similarity score
predicted_class_list.append(class_maxvalue)
```

```
# print('\nThe test protein[',index+1,'] belongs to :',class_maxvalue)

# This adds the predicted classes to the last
# column in the testArrays
test_set_row = test_set.iloc[:,0].shape[0]
testArrays = np.concatenate((testArrays,test_labels.
values.reshape
([len(test_labels),1]),
np.array([predicted_class_list]).T,test_set.iloc[:,0].
values.reshape
([test_set_row,1])),axis=1)

# Checking the accuracy
true_positives = 0 # variable to hold the number of occurances
where actual is equal to predicted
pred_classes_count_dict = {}
test_classes_count_dict = {}
for i in range(len(test_labels)):
testLabel = str(test_labels.iloc[i]).strip().lower()
# count the classes in test added str in line 119 and 112
if testLabel in test_classes_count_dict.keys():
test_classes_count_dict[testLabel]+=1
else:
test_classes_count_dict[testLabel]=1

if testLabel== str(predicted_class_list[i]).strip().lower():
true_positives +=1
# count the predicted classes
```

```
if testLabel in pred_classes_count_dict.keys():
pred_classes_count_dict[testLabel]+=1
else:
pred_classes_count_dict[testLabel]=1

for key,value in pred_classes_count_dict.items():
print('The Accuracy of class [',key,'], is :',
pred_classes_count_dict[key]/test_classes_count_dict[key]*100)
print('The Overall Accuracy is :',
true_positives/len(testArrays)*100)


#save the predictions made to a file
np.savetxt('Test_Set_predicted_simi.csv', testArrays,'%s',',')
```

Appendix D- PCA & Random Forest

```
#importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#import the data set

np.random.seed(75)

dataset=pd.read_csv('Final_correected_GPCR_TransNOnGPCR.csv',header=0)
X=dataset.iloc[:,1:-1].values
y=dataset.iloc[:,-1].values

#splitting the dataset into the Training set and test set
from sklearn.model_selection import train_test_split
X_train,X_test, y_train,y_test=
train_test_split(X,y ,test_size=0.3,random_state=75)

#feature scaling

from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
#print(X_train)
X_test=sc.transform(X_test)
```

```
#applying pca
from sklearn.decomposition import PCA
pca = PCA(n_components = 75)
X_train = pca.fit_transform(X_train)
print(X_train)
X_test = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_
print(explained_variance)

# Fitting RandomForst Classifier to the Training set
from sklearn.ensemble import RandomForestClassifier as RFC
classifier = RFC(n_estimators=1000,criterion='entropy',75)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

Curriculum Vitae

Fredrick Ayivor was born on December 08, 1988. The first son of Emmanuel Ayivor and Vida Amankwah, he graduated from St Thomas High School at Osu, Ghana, in 2007. He entered Kwame Nkrumah University of Science & Technology (KNUST) in the fall of 2008. After pursuing his bachelor's degree in Mathematics in the summer of 2012, he worked as a Teaching Assistant, at the same institution (KNUST).

In the fall of 2014, he entered the Graduate School of New Mexico State University at las cruces, New Mexico. While pursuing a master's degree in Mathematical Science he worked as a Teaching Assistant, and as the Laboratory Instructor for the institution. Fredrick got accepted in the Computational Science program at University of Texas at El Paso to pursue his Doctoral degree in fall 2016 and currently resides in El Paso with his wife.

Email address: **fayivor@miners.utep.edu**