EVALUATING TRAINING PROCEDURES IN A NAIVE BAYES APPROACH TO PATHOGENICITY PREDICTION: THE CASE OF THE FAMILY OF SODIUM CHANNEL PROTEINS

by

Jing Li

A Thesis Submitted to the Faculty of the

Graduate Interdisciplinary Program in Statistics

In Partial Fulfillment of the Requirements

For the Degree of

MASTER OF SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

2018

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

The thesis titled ***Evaluating Training Procedures in a Naive Bayes Approach to Pathogenicity Prediction: The Case of the Family of Sodium Channel Proteins*** prepared by ***Jing Li*** has been submitted in partial fulfillment of requirements for a master's degree at the University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that an accurate acknowledgement of the source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: Jing Li

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

Joseph C. Watkins

Joseph Watkins, Professor of Mathematics

Defense date
May 10th, 2018

Acknowledgements

I want to express my deep gratitude for my advisor Professor Joseph Watkins, who provides me with the opportunity to research into this topic. His detailed guidance is always very helpful while I'm finishing this research project. Every discussion with Dr. Watkins motivates me to work towards better results. Also I would like to my committee members for their helpful discussion while writing this paper and the faculty members in the GIDP statistics program who gives me help and guidance during my study in the program. Finally I would like to thank my fellow students in the program who gave me accompany, joy and encouragement in the effort of pursuing this degree. Thank you all very much.

Contents

List of Figures

List of Tables

Abstract

PolyPhen-2 is a software that could help predict the pathogeneicity of mutations. We used it for prediction for sodium channel protein data after comparison with a few other prediction including Grantham Scores, SIFT, phyloP, PolyPhen-2, and CADD. But it's still working with limited accuracy. We are primarily concerned about is that our problem is trying to study the pathogenicity of sodium channel proteins and we are questioning that if these softwares will help us predicting these mutation as well as the training set is based on the whole genome. So we tried to modify the software by training a data set from sodium channel protein mutations using naive Bayes algorithms. Then we compared the prediction results from our classifier and the prediction results of PolyPhen-2, then we could see that the classifier trained by our methods can make prediction results with significantly better accuracy.

1 Introduction

1.1 Basic Concepts

The central dogma of molecular biology states that genes encoding DNA are transcribed into messenger RNA which find their way to the ribosomes. Here, transfer RNA will decode the nucleotide sequence into an amino acid sequence, which leads to the synthesis of a protein. A point mutation refers to a genetic mutation in which a single nucleotide is changed, inserted or deleted from DNA or RNA. Point mutations may have a variety of effects on the downstream protein product. If the mutation does not change the amino acid sequence, we call the mutation synonymous. If a mutation results in a change in the amino acid sequence, the mutation is called non-synonymous and the the protein structure varies from previous pattern. This may result in a difference in protein function and could lead to certain disease conditions.

In general, the term pathogenicity of a mutation refers whether the mutation is causing a certain type of disease. For genetic diseases, such as sickle cell anemia, Marfan Syndrome, etc. We want to know if a mutation will result in such disease. Hence in this context we use the term pathogenicity to denote whether a point mutation will end in an amino acid sequence or protein that would result in such diseases as we are concerned. Geneticists and bioinformaticians from all over the world have been trying to build models or software to make such prediction possible and accurate.

Pathogenicity prediction algorithms include Grantham (1974), SIFT (2003), phyloP (2006), PolyPhen-2 (2010), and CADD (2013). For example, CADD builds a model applying support vector machine for classification, and gives output called C-scores to help evaluate the pathogenicity of a mutation. Some of these models and associated software use datasets based on the entire human genome as well as genomic data from other species to predict the pathogenicity of a mutation. In Chapter 2, we will describe these methods in more detail, briefly compare them and discuss how they relate to our problem.

In this thesis, we focus on the family of sodium channel proteins. Sodium channels are proteins that form ion channels, conducting sodium ions (Na^+)

through a cell's plasma membrane. They are classified according to the trigger that opens the channel, i.e. either a voltage-change or a binding of a substance (a ligand) to the channel (ligand-gated sodium channels). In excitable cells such as neurons, myocytes, and certain types of glia, sodium channels are responsible for the regulating action potentials. Humans have nine different sodium channel proteins, *SCN1A*, *SCN2A*,..., and *SCN9A*. We are focus on point mutations in the genes *SCN1A*, *SCN2A*, and *SCN8A*. These channels consist of four domains, each having six linked transmembrane segments. Domains are connected by a sequence of amino acids called a linker. Mutations in these proteins are know to cause diseases such as epilepsy and autism spectrum disorder.

1.2 Overview of Thesis

The goal of this thesis to test the idea of improving pathogenicity prediction based on a more focused training set rather using the entire genome. We will explore this idea in the context of sodium channels and the pathogenicity prediction algorithm PolyPhen-2. We still adopt the naïve Bayes methods used in PolyPhen-2.

1.3 Pathogenicity Prediction

When we detect a point mutation, a natural question would be whether it will cause disease, or, more precisely, what is the probability that the carrier remains healthy. We give a brief overview of several approaches to address this question

1.3.1 Grantham Scores

The Grantham Scores predict the distance between two amino acids and whether such a mutation is pathogenic in the evolutionary sense. It gives a measure of distance of two amino acids based on molecular polarity, composition, and volume. Polarity refers to energetically favorable contact with water. Composition is measured by assigning different scores for different side chain of amino acids. Molecular volume is also considered. The Grantham score is obtained by summing the squares of these three features

for the two amino acids under consideration. It holds the idea that the more distant two amino acid are the more pathogenic is their substitution. The distance scores published by R. Grantham are between 5 and 215. For example, a substitution of isoleucine for leucine, or of leucine for isoleucine, has a score of 5 (and is predicted to be tolerated). A substitution cysteine for tryptophan, or of tryptophan for cysteine, has a score of 215. Any variation involving cysteine has a high or very high Grantham score (and is predicted to be deleterious).

1.3.2 SIFT

SIFT (Sorting Intolerant From Tolerant) is also a program trying to predict whether an amino acid substitution is pathogenic. It is mainly applied to human protein to identify deleterious mutation and can be used for other species as well. It uses sequence homology of protein family members to compute whether an amino acid substitution will affect protein function. For a specific mutation, it uses BLAST (Basic Local Alignment Search Tool) to find homologous sequences across species and then it examines whether they are conserved or polymorphic. It holds the idea that residues that are conserved in protein family tends to affect function more if there is an amino-acid change. SIFT presumes that important amino acids will remain fixed in the protein family. So changes at well-conserved positions tend to be deleterious.

1.3.3 PhyloP

PhyloP is a phylogenetic analysis of mutation sites that comes from Haussler's group at UC Santa Cruz. It's output is a score ranging from -14 to 3 that helps evaluate how conserved is an individual alignment sites. Positive scores would indicate more conserved sites while negative scores would show the sites are more rapidly-evolving. This is also based on the principle that mutation on more conserved sites are more likely to be pathogenic.

1.3.4 PolyPhen-2

PolyPhen-2 is an updated version of the original Polyphen, which is an open-source software that can be found at genetics.bwh.harvard.edu. It will extract some corresponding features of amino acid sequence and compute to tell whether a new mutant amino acid will bring disease based on historical data. These features includes change in residue side chain volume, CpG context, normalized accessible surface area, etc, which will be further explained in section 2.2. Its computing algorithm is a naive Bayes classifier which predicts the outcome probability based on these features. An entropy-based discretization is applied to continuous variables.

1.3.5 CADD

CADD stands for Combined Annotation Dependent Depletion and its output is usually called C-score. It was developed by developed by Martin Kircher, Philipp Rentzsch, Daniela M. Witten, Gregory M. Cooper, and Jay Shendure at University of Washington. It includes about 19.4 million variants and 63 distinct annotations. A support vector machine is trained from these annotations and related features. For a later version of CADD, a logistic regression method is also used. For a variant, CADD can provide both raw score and scaled scores, which tell the percentage of a mutant pathogenicity, ranging from 0 to 100. Here score 0 means benign and score 100 means definitely deleterious.

1.4 Sodium Channel Proteins

Our data comes from the Sodium Channel Protein genes, *SCN1A*, *SCN2A*, and *SCN8A*. Here is a brief description of their common structures.

Sodium channels are integral membrane proteins that form ion channels conducting sodium ion (Na^+) through a cells plasma membrane. Their location is shown in Figure 1. They are classified according to the trigger that opens the channel for such ions, i.e. either a voltage-change or a binding of a substance (a ligand) to the protein.

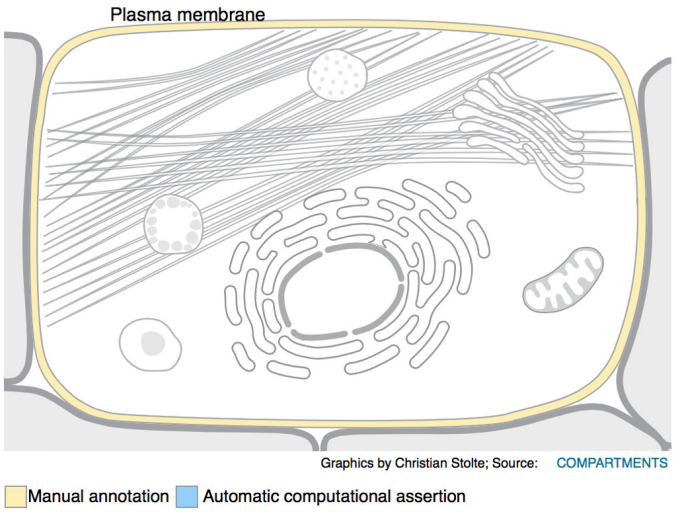

Figure 1: SCN Protein Location
-Pic source: Protein data bank, https://www.rcsb.org

The voltage-gated sodium channel has several functional parts that determine its ion selectivity. This particular channel is specifically selective for sodium ions. Even the chemical similar potassium ions cannot pass through the channel. Another portion of the channel protein serves as a gate that can open and close. For many ion channels, the gate opens in response to regulatory molecules that specifically bind to either the inside or outside of the channel. But in the case of the voltage-gated sodium channel, the gate is controlled by a voltage sensor, which responds to the level of the membrane potential, as shown in the picture below.

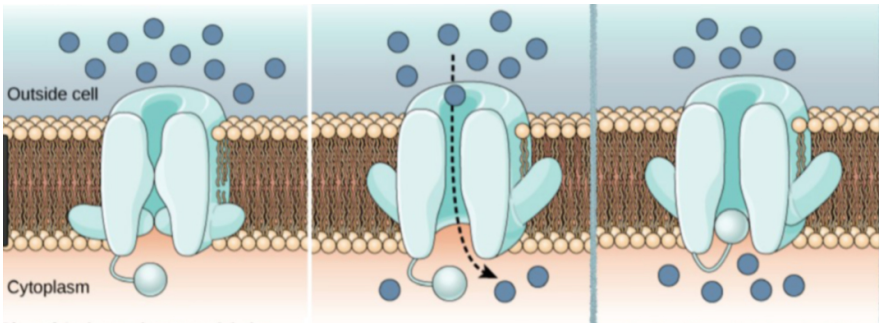

Figure 2: Sodium Channel
-Pic Source: OpenStax Biology, https://cnx.org/contents/

The voltage-gated sodium channel is a specific example of a transmembrane protein. This type of protein is found in nerve and muscle cells and is necessary for rapid electrical signaling. The principle subunits of the voltage-gated sodium channel is a polypeptide chain of more than 1800 amino acids. When the amino acid sequence of any protein embedded in a cell membrane is examined, typically one or more segments of polypeptide chain are found to be comprised largely of amino acid with non-polar side chains. Each of these segment coils is what is called a transmembrane segment with a length approximately the width of the membrane. Moreover, within a transmembrane segment the side chains necessarily face outward where they readily interact with the lipids of the membrane. By contrast, the peptide bonds face inward, and are separated from the lipid environmental of the membrane. Assuming opened or closed mechanism in response to the voltage difference across the membrane, the protein constructs a sodium selective channel through which Na^+ ions may pass in accordance with their electrochemical gradient. In the case of the voltage-gated sodium channels, there are 24 such transmembrane segments, six in each of four domains, in the polypeptide chain, as shown in the following picture.

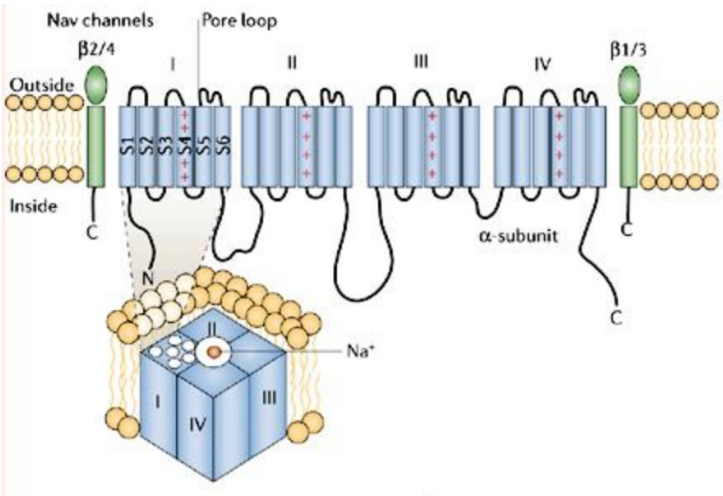

Figure 3: Sodium Channel Structure
-Pic source: H. Lai, L. Jan, Nature Review, 2006

We could find all the major features of sodium channel protein. And these features are used in the naive Bayes algorithms to train our model when computing the possibility whether a mutant allele will cause disease. We'll discuss this more in detail in the following section.

2 Material and Methods

2.1 Description of the Data

Our data on genetic variants of healthy (control) individuals is taken from the ExAC database. The Exome Aggregation Consortium (ExAC) database is maintained by Broad Institute to aggregate and arrange exome sequencing data. Thus, we download mutation information the ExAC database for the proteins *SCN1A*, *SCN2A*, *SCN8A*. Each sample has detailed information on point mutations, i.e., the change of nucleotides.

In addition, we have currently have 3 different datasets, i.e. from *SCN1A*, *SCN2A*, *SCN8A* for individuals having a genetic disease. So we have 450 to 470 samples in total (together with the benign samples from ExAC) for each of the three proteins in our study. For each sample we know the exact

change of amino acid and its exact location in the DNA sequence. We also know that whether the corresponding individual is carrying disease or not. For SCN1A samples, about 1/4 samples have a pathogenic condition. For SCN2A, the individuals with disease are approximately 1/3 of all samples. For SCN8A, the ratio of people with disease is even higher, reaching almost one half of all samples. For each of the three proteins, we have two comparison groups, the first is the dataset with nucleotide mutations known to be pathogenic, while the comparison group consist of samples with nucleotide mutations which are known to be benign. We'll randomly select samples from both group for training and testing.

Our training data sets come from the output of PolyPhen-2. A batch query into PolyPhen-2 returns the prediction results, posterior probability, as well as the features that are used for making these predictions. These can be downloaded as a full report from the PolyPhen-2 platform. Then we examine the output data frames to investigate what features are good for training our new model.

2.2 Motivation and Methods

In the following table, there are three categories of outcome, Correct Prediction, Non Accurate Prediction and Wrong Prediction. A mutation is classified as probably damaging if its probabilistic score is above 0.85. A mutation is classified as possibly damaging if its posterior probability is above 0.15 true positive rates are below 90% corresponding to the fraction of false positive under 18%. The remaining mutations are classified as benign. There are three possibilities of PolyPhen-2 categorical outcome, benign, possibly damaging, and probably damaging. Correct Prediction means the output category given by PolyPhen-2 is exactly the same as the pathogenicity of mutant/wild allele. Wrong Prediction gives the number that PolyPhen-2 gives the opposite results, i.e. benign predicted to be probably damaging or causing disease predicted to be benign. For either benign or pathogenic mutation (by its nature) if its predicted to be 'possibly damaging' we counted that as Non-Accurate. We could see that these results are not very reliable. For example, we can see from the following table that the incorrect prediction for benign samples are very high, especially for SCN8A and SCN2A

groups. For example, in the first row of the table, it shows how well benign samples are predicted by PolyPhen-2. For SCN1A data, we have 345 benign samples in total, 186 of them are predicted as benign, 96 of them are predicted as possible damaging, and 62 of them are predicted as probably damaging/pathogenic, and similar countings are presented in the following rows,

	Correct Prediction	Non-Acccurate Prediction	Wrong Prediction	Total
SCN1A benign	180	96	62	345
SCN1A pathogenic	109	6	7	122
SCN2A benign	141	42	103	288
SCN2A pathogenic	129	15	21	167
SCN8A benign	128	53	67	249
SCN8A pathogenic	129	38	27	211
Total benign	455	191	232	882
Total pathogenic	367	59	55	500

Table 1: Summary of PolyPhen-2 Output

2.3 Features Used for Prediction

In the first process of training our new model, we are examining about 40 features that could be seen from the PolyPhen-2 output table. We retain 16 features selected from the 40 features (including the values based on the PSIC score) features available. We choose these 16 features because the data are relatively complete. They are listed as follows:

1. **site**–substitution SITE annotation, meaning that whether the substitution happening at the site or not.

2. **region**–substitution REGION annotation. The values can be transmembrane, repeat, or none. So it's a feature having 3 discrete values.

3. **PHAT**–PHAT matrix element to estimate the probability an amino acid substitution in the transmembrane region. PHAT stands for predicted hydrophobic and transmembrane matrix. It's a 20 by 20 triangular matrix with integer values. It's treated as discrete features in our computation because elements in the matrix are indicators of probabilities and that there are only finite possible indices in the matrix for this feature.

4. **score1**–PSIC score for wild type amino acid residue, PSIC stands for position specific independent counts. It is a computed score based on the frequency of an amino acid at a specific position. It's a continuous feature given by the log ratio of likelihood based on search in NRDB (Natural Resources Database, http://www.nrdb.org).

5. **score2**–PSIC score for mutant amino acid residue. And similarly it's a continuous feature.

6. **dscore**–difference of PSIC scores for two amino acid residue variants score1-score2, which is also a continuous feature.

7. **MSAv**–version of the multiple sequence alignment used in conservation scores calculations, there are totally 3 versions used here, 1) pairwise BLAST HSP obsolete, 2) MAFFT Leon Cluspack default, 3) MultiZ CDS. It's a discrete variable with 3 values. This value shows which version is used so the value would be 1,2 or 3.

8. **Nobs**–number of residues observed at the substitution position in multiple alignments without gaps. (in human proteins) It is treated as a continuous feature.

9. **Transv**–whether substitution is a transversion.

10. **CodPos**–position of the substitution within a codon. We know that a codon is formed by 3 nucleotides and we marked the value as 0 if it happens at the start of the 3-sequence, 1 if in the middle, and 2 if it's at the end. It is a discrete feature with 3 values.

11. **CpG**–whether substitution changes CpG context: 0, non-CpG context retained; 1, removes CpG site; 2, creates new CpG site; 3, CpG context retained.

12. **MinDJxn**–substitution distance from closest exon/intron junction. This distance is measured by the number of amino acid from the substitution position and is treated as a continuous feature.
13. **Pfamhit**–Pfam identifier of the query protein. It is a unique identifier of protein in the Pfam database. Pfam is a protein family database which tries to put protein into clans or proteome so that various proteins are more easily to be identified by its clan. It's searching and comparing algorithm is based on hidden Markov model. So the identifier here would be an indicator about which family does the protein belong to. So it's treated as a discrete feature.
14. **IDQmin**–query sequence identity with the closest homologue deviating from the wild type amino acid residue. It's a continuous feature.
15. **IdPmax**–It gives an index of maximum prediction congruency of the mutant amino acid residue to all sequences. It's also a continuous feature.
16. **IDSNP**–maximum congruency of the mutant amino acid residue to the sequences in multiple alignment with the mutant residue. It is a continuous feature.

We have to note that these features used here are different from the features used in PolyPhen-2. A few features used in PolyPhen-2 are not included because these data are incomplete, which are not usable for computation. We are using a few more features available because we want to include as much information as possible.

2.4 Naive Bayes Classification

PolyPhen-2 selected 11 items from all the possible variables to be considered, based on an iterative greedy algorithm. We are using a similar training method, i.e. naive Bayes methods (but not including greedy algorithm), as described more in detail in the following sections.

2.4.1 Bayes Theorem and Classification

Given events A and B, we have:

$$P(A|B) = P(A) \cdot P(B|A)/P(B)$$

This can be derived from the general multiplication formula for AND events:

$$\begin{aligned} P(A \cap B) &= P(A) \cdot P(B|A) \\ P(B|A) &= P(A \cap B)/P(A) \\ P(B|A) &= P(B) \cdot P(A|B)/P(A) \end{aligned}$$

The naive Bayes classifier calculates posterior conditional probabilities for a class outcome given prior information (our features in this case). The term naïve is used because the prior probability assume independence between features when in reality they may be dependent in some way. Here we just assume all the features we used are independent with each other.

This assumption allows us to calculate the probability of the evidence by multiplying the individual probabilities of each piece of evidence occurring together using the simple multiplication rule for independent events. The values for the features are very unlikely to satisfy the independence assumption. Nevertheless naive Bayes has shown itself to be a good approximation for the posterior probability and is adequate for the purposes of classification. Because the Naive Bayes classifier has proven to be highly effective, it is commonly deployed in classification problems.

If we write the formulas with words expression as needed for the problem, the Naive Bayes formula becomes:

$$P(Class, Features) = P(Class|Features)P(Features) = P(Features|Class)P(Class)$$

hence,

$$P(Class|Features) = P(Class) \cdot P(Features|Class)/P(Features)$$

In the equations above, 'Features' is short for values of different features. The main aim in the Naive Bayes algorithm is to calculate the conditional probability of a sample with a feature vector.

2.4.2 Discrete and Continuous Variables

Let C_0, C_1 denote 2 different classes, C_0 denotes a benign class while C_1 denotes the class that has disease, then for $X_1, X_2, ...X_m$, discrete features, they have conditional joint mass function $f_i^d(u_1, ...u_n)$, for class i. This is assumed to be independent and so becomes a product of mass functions, $\prod_{j=1}^{m} f_i^j(u_j)$.

For continuous features $Y_1, ...Y_c$, we assumed them to be independent and they have joint conditional density $g_i^c(v_1, ...v_c)$ for class i, and it is equivalent to a prodect of density functions, $\prod_{j=1}^{c} g_i^j(v_j)$.

The algorithm that deals with continuous data from R package assumes that a variable follows normal distribution. It will automatically gives an estimate of mean and variance, applying maximum likelihood estimation. Then we could get an estimate of posterior probability applying the kernel method. The kernel method, which is also called the kernel trick is using a function that is similar to probability function to compute the posterior probability. Here we use the density to substitute the probability interval around a point. Hence the y-value on the density curve is retained to denote the value of probability around a small area around x, then we could do naive bayes computation using these approximations and get outcome values.

Naive Bayes method assumes each feature is independent hence we have,

$$P(C_i|X_1 = u_1, ...X_m = u_m, Y_1 = v_1, ...Y_n = v_n)$$

$$\propto f_i^d(u_1, u_2, ..., u_m) g_i^c(v_1, v_2, ...v_n) P(C_i)$$

$$\propto \prod_{j=1}^{m} f_i^j(u_j) \prod_{k=1}^{n} g_i^k(v_k) P(C_i)$$

In the product above, the value of $f_i^j(u_j)$ in the naive Bayes classifier is estimated by the fraction of counts in the data, $g_i^k(v_k)$ is a density function for the normal distribution where mean and variance are given by MLE derived from our data.

In our training set, any missing data-point will not be counted into computation of conditional probability if it's a discrete feature, or it won't be counted into MLE estimate of $\hat{\mu}$ or $\hat{\sigma}$ if it's a continuous feature. And for the prediction for testing test if any feature is missing then it would be omitted in the product, or say treated as 1, when computing posterior probability.

2.5 3-fold Cross Validation

We first separate our data sets into 2/3 training set and 1/3 testing set to train a new model. Then we apply 3-fold cross validation and obtain a new prediction result on the whole data-set, which is compared with the output of PolyPhen-2. We'll illustrate this more in detail in Chapter 3.

3 Results and Comparison

3.1 Quality of Predictions from PolyPhen-2

A summary of PolyPhen-2 output is given in section 2.2. Beside giving prediction probability, PolyPhen-2 divides posterior probability into 3 categories, benign, possibly damaging, and damaging. So for the possibly damaging part there's no clear true or false, we make assignment of a Non-Accurate prediction. For example for SCN1A benign samples, if they are predicted to be benign then the prediction is correct, if PP2 says it's 'probably damaging' then it's a wrong prediction, and if PP2 shows its 'POSSIBLY damaging', then we'll add one more count to the number of non-accurate predictions.

3.2 Training Methods Comparison

We start with a relative simple approach. For each of the models in this section we use 2/3 data for training and 1/3 for testing. Then we apply 3-fold cross validation. And for the outcome we assign a natural cutoff of 0.5, i.e. if the prediction probability is bigger than 0.5 we predict that to be pathogenic or benign otherwise. Then we'll have a look at the confusion matrices of these models to determine whether we should train 3 dataset (SCN1A, SCN2A, SCN8A) all together or we'd better train them separately. And in section 3.3 we would apply 3-fold cross-validation and give more detailed evaluations of training and testing results.

We first train SCN1A data separately and get the following confusion matrix. For SCN1A data, when predicting the testing set, 81 samples were incorrectly predicted, the rest 399 were correctly predicted, Then the following two columns present true positive rate, false negative rate, true negative rate and false positive rate, respectively.

SCN1A Model	Test Total	Pathogenic Sample	Benign Sample	TP FN	TN FP
All Sample	480	131	349		
Correct Prediction	399	112	287	85.50%	82.23%
Wrong Prediction	81	19	62	14.50%	17.77%

Table 2: SCN1A Model Confusion Matrix

For the SCN2A model, when predicting the testing set, 94 samples were wrongly predicted, 343 samples were correctly predicted, and we have a similar table as follows,

SCN2A Model	Test Total	Pathogenic Sample	Benign Sample	TP FN	TN FP
All Sample	437	157	280		
Correct Prediction	343	146	197	92.99%	70.36%
Wrong Prediction	94	11	83	7.01%	29.64%

Table 3: SCN2A Model Confusion Matrix

For the SCN8A model, when predicting the testing set, 122 samples were wrongly predicted, 348 were correctly predicted, and we have a more detailed table for theses counts and rates as follows,

SCN8A Model	Test Total	Pathogenic Sample	Benign Sample	TP FN	TN FP
All Sample	470	222	248		
Correct Prediction	348	158	190	71.17%	76.61%
Wrong Prediction	122	64	58	28.23%	23.39%

Table 4: SCN8A Model Confusion Matrix

Next we use the entire sodium channel data set to train and test on each of the three sodium channels individuals. The results are contained in the following table. Similar to table 2,3,4, which show summary with data trained separately from 3 proteins, table 5, table 6, and table 7 give the details how different datasets are predicted in this same model.

All-Trained Model	Test Total	Pathogenic Sample	Benign Sample	TP FN	TN FP
SCN1A data	465	126	339		
Correct Prediction	380	101	279	80.15%	79.35%
Wrong Prediction	85	25	60	19.85%	20.65%

Table 5: SCN Model Confusion Matrix– 1A data

All-Trained Model	Test Total	Pathogenic Sample	Benign Sample	TP FN	TN FP
SCN2A data	490	181	309		
Correct Prediction	335	149	186	82.32%	60.19%
Wrong Prediction	155	32	123	17.68%	39.81%

Table 6: SCN Model Confusion Matrix– 2A data

All-Trained Model	Test Total	Pathogenic Sample	Benign Sample	TP FN	TN FP
SCN8A data	439	191	248		
Correct Prediction	282	66	216	34.56%	87.10%
Wrong Prediction	157	125	32	65.44%	13.90%

Table 7: SCN Model Confusion Matrix– 8A data

Here are three initial observations we can make from the confusion matrices above,

1) After training our data based on the specific protein, the results show that the false-positive rate was reduced to an extent that is more acceptable. We could see that for each of the 3 specific data sets, i.e. SCN1A model, SCN2A model, SCN8A model, we have a higher correction rates, comparing to the results in table 1. So we would give more discussion about these results in the following sections.

2) When we train the data all-together, as shown in Table 5, Table 6, Table 7, we could see that the correct rates are not raised obviously than the result of PolyPhen-2, unlike the situation when separate the 3 groups data and counting the prediction results consecutively on the same testing sets (i.e. for SCN1A, SCN2A, SCN8A), as shown in Table 2 , Table 3, and Table 4. So we could assume that these new models trained are no more accurate for prediction than PolyPhen-2, and that this is not a better approach than training the 3 set separately.

3) To analyze the difference between training all data together and training them together, we propose the following tests. For pathogenic samples, For example for SCN1A, we hold the idea that a better classifier will give more prediction of 1s instead of 0s, so we would subtract the prediction of training all together from the prediction of training SCN1A itself, then we'll look at the result sequence to see difference in proportions, applying a one-sample z-test. And similarly we could test the subtraction results for benign samples, so here H_0 is that of difference (whole training classification results -separate training classification results) is 0 or greater than 0, and H_α is that results from training on specific protein is better. And the p-values for these tests are list in the table below.

Table 8: Test of Difference in Proportions

proportion and p-values	SCN1A	SCN2A	SCN8A
separate-training proportion(TP)	85.50%	92.99%	71.17%
all-training proportion(TP)	80.15%	82.32%	34.56%
correlation-pathogenic sample predictions	0.4860	0.5038	0.4368
pathogenic sample z-test	0.0225	0.0661	2^*10^{-16}
separate-training proportion(TN)	82.23%	70.36%	76.61%
all-training proportion(TN)	79.35%	60.19%	87.10%
correlation-benign sample predictions	0.9478	0.7260	0.3870
benign sample z-test	0.3277	0.0004	≈ 1

From this table we can see that for SCN1A pathogenic samples, SCN2A benign samples, SCN8A pathogenic samples, separate-training models seems to have significant improvement than all-trained models, with significance level 0.05. This result can also be verified by the confusion matrices above. For SCN1A testing set, true positive rates increases most by 5 percent by training them separately. And for SCN2A data-set, true negative rates increases by 10 percent. For SCN8A dataset, true positive rates between separate-training model and whole-training model differ by more than 35 percent, and the very small p-value would be a good indicator of the difference here while true negative rate for this dataset does not increase by training data separately. So we would conclude that separate-training models give more accurate prediction in general and that 3 data-set would have different improvement based on cases (pathogenic) and controls (benign).

Correlation of two proportions is also shown in the table. We could see that for SCN1A data we have a very high correlation between two models when prediction benign samples, i.e. 0.9478; while the correlation value is not so high for pathogenic samples, with a value of 0.4860. Therefore we would conclude that for SCN1A benign samples two model are pretty much the same while for SCN1A pathogenic samples they are giving different predictions. This is consistent with the p-values given in the z-test. And for SCN2A data correlation for benign samples is still higher than pathogenic samples i.e. 0.5038. For SCN8A both pathogenic and benign samples have a relatively low correlation value, i.e. 0.4368 and 0.3870, respectively. This can also be verified by their ROC curves. More quantitative discussion will come in the following sections, given ROC curves of different models.

3.3 Quality of Predictions from Sodium Channel Data

If we focus on the prediction probabilities and false positive rates, we could compare our prediction results with the PolyPhen-2 prediction probabilities as follows. For example, for a designated false positive rate, such as 0.05, 0.10, or 0.20, we could know the classifier's true positive rate corresponding to that threshold, and therefore compare the true positive rates of different classifiers based on the same false positive thresholds level. And below is the the table of True Positive Rate with the same False Positive Rate Level/Threshold,

training set	whole genome			sodium channels			3 sodium channels		
FPR	0.05	0.10	0.20	0.05	0.10	0.20	0.05	0.10	0.20
SCN1A	0.23	0.44	0.78	0.44	0.60	0.84	0.28	0.49	0.61
SCN2A	0.17	0.33	0.52	0.42	0.51	0.85	0.27	0.33	0.49
SCN8A	0.15	0.35	0.58	0.36	0.58	0.73	0.38	0.43	0.50

Table 9: True Positive Rates (TPR) with False Positive Rates (FPR) Thresholds

For SCN1A testing set, we can see that separate training classifier generally gives a higher TPR than PolyPhen-2, while all-trained classifier does not give a higher rate than PolyPhen-2 at 3 certain thresholds. So we believe that for SCN1A set separate-training model is better. Adding the information

from Table 8, we could attribute this to the improvement from pathogenic samples in SCN1A pool. And we could see for SCN2A data-set, all-trained classifier shows a higher TPR rates but not always, while separate-training classifier generally gives a higher TPR at 3 thresholds. Comparing to Table 8 we could conclude this comes from the that separate-training classifier gives more true negative prediction, For SCN8A testing set, TPR of all-trained model is better than PolyPhen-2 occasionally while separate-trained model is always better. Similarly we could see that this comes from the improvement of pathogenic-sample prediction, combining the information given in Table 8.

So with Table 9 and Table 8, we could summarize that separate training model would increase the prediction accuracy of 3 datasets, but in different mechanism and that this improvement is not well- realized if we train them all together. For example, it might be that SCN2A pathogenic sample data won't help with SCN1A pathogenic sample prediction, and if we train them all together these advances will be reduced. And more quantitative illustration of the differences will be given by ROC curves in the following section.

3.4 Receiver Operating Characteristic Curves

When we obtain the posterior probability from the Naive Bayes classifier applied to a set of data or testing set, the receiver operating characteristic or ROC curve is defined to be a plot of the true positive rates (TPR) for each value of the false positive rate (FPR), or we could denote it as sensitivity versus (1-specificity). So we could list all the posterior probabilities from lowest to highest, give the thresholds of prediction cutoffs and plot the TPR versus FPR.

For example, for a designated false positive rate, such as 0.05, 0.10, or 0.20, we could know the classifier's true positive rate corresponding to that threshold, and therefore compare the true positive rates of different classifiers based on the same false positive thresholds level. These probabilities are summarized in Table 9. And we could see the ROC curves as a overall characteristic of a classifier, compute its AUC (area under curve) and compare different classifiers' prediction ability based on their AUCs. A better

classifier tends to achieve AUC close to 1 while a random classifier only holds AUC equal to 0.5. Then we could apply DeLong's test ([11], Delong, 1988) to determine whether the AUCs are significantly different from each other. More details will be discussed in section 3.5.

For SCN1A model and testing sets, we got the following curves. The blue line shows the ROC of SCN1A model while the black line shows the ROC of PolyPhen-2 on the same testing set. The picture of two ROC shows the results of a 3-fold cross validation results. The black line is result for PolyPhen-2 and the blue line is result for our model. The purple line shows the prediction given by model trained by all 3 datasets (SCN1A SCN2A SCN8A).They are drawn over the same data-set. And SCN1A rates table 9 is part of the information shown in these curves. We can see that all trained-model (purple line) is twisted with PolyPhen-2 outcome (black line), while SCN1A-trained classifier(blue line) seems to do slightly better.

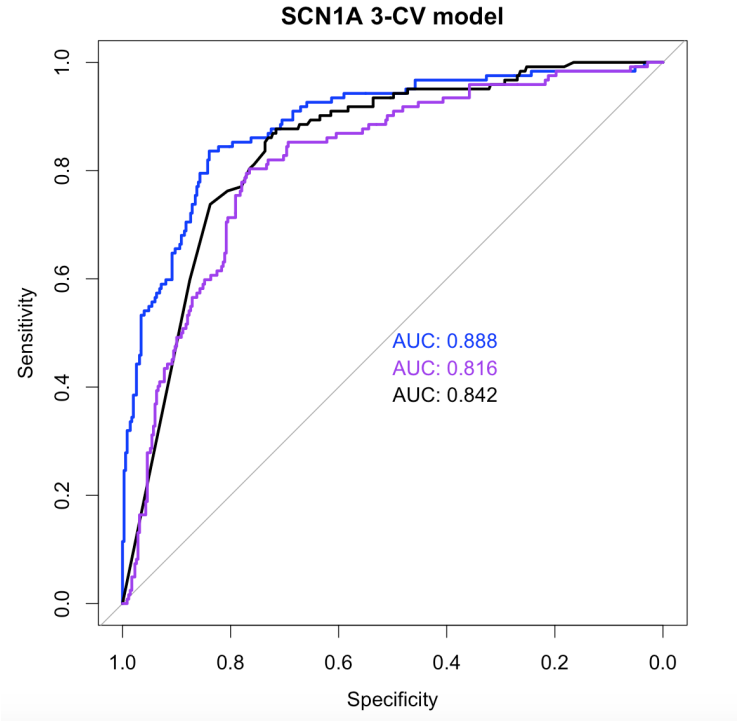

Figure 4: SCN1A model ROC curves

Similary for SCN2A curves, we got the following curves, with blue line showing the SCN2A ROC and black line showing the PolyPhen-2 ROC, purple line showing results from training on 3 datasets, on the same dataset. True positive rates at certain thresholds can also be found in Table 9 and we can see that all-trained model is better than PolyPhen-2 at lower FPRs while SCN2A-trained model is doing much better.

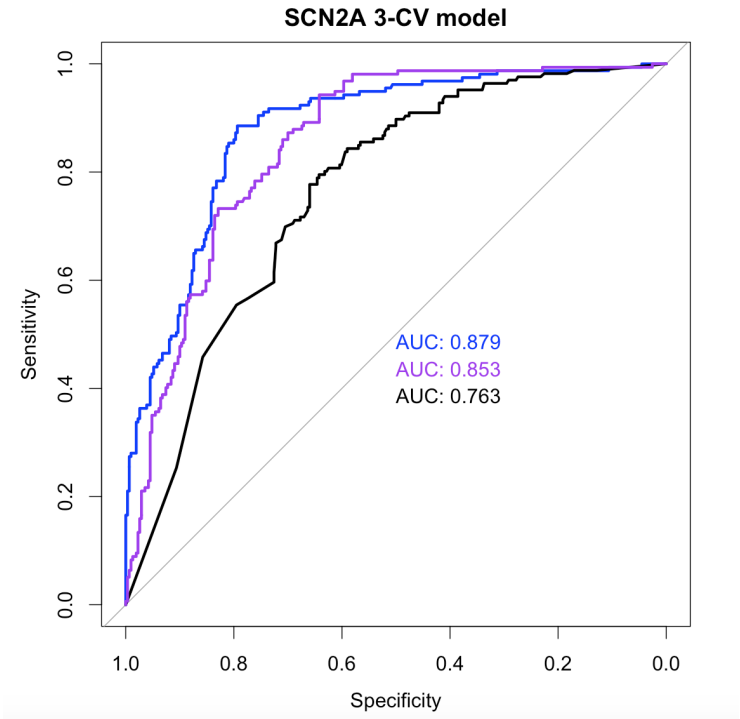

Figure 5: SCN2A model ROC curves

And the following picture gives the SCN8A curves, with blue line coming from the SCN8A model and PolyPhen-2 showing in black line. Similarly purple line shows the result from 3-dataset-training. Similary all-trained model is more twisted with PolyPhen-2 results while SCN8A model generally produces higher TPRs.

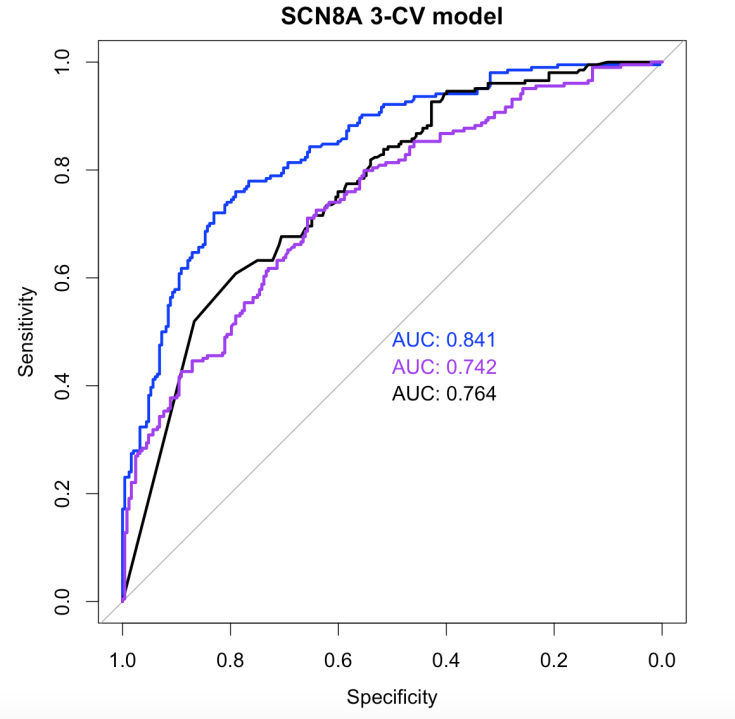

Figure 6: SCN8A model ROC curves

3.5 Test of Differences in ROC Curves

We want to look at the differences of the ROC curves more in detail so we compute the AUC, and conduct DeLong's tests and we write it in the following table. We could see that the p-values are all very small so we could conclude that the training results in our work is significantly different from the PolyPhen-2 outcome.

We are using DeLong's test to examine whether the two ROCs are significantly different. The deLong test uses U statistics to obtain a 2-dimensional central limit theorem. Taking into account the correlation, we can compute a z-statistic, which follows a chi-square distribution. Results of two-samples test of z-statistics are shown in the following table,

Classifier/ Test-Set	AUC separate-trained	AUC PolyPhen-2	p-value
SCN1A	0.88776	0.84249	0.04451
SCN2A	0.87889	0.76269	0.000015
SCN8A	0.84134	0.76358	0.003276

Table 10: Delong test for ROC curves - Separate training vs PolyPhen-2

From ROC curves and deLong's tests, we could see that we could get obvious improvement on SCN2A and SCN8A data. Hence we could conclude that the mutation structure are very helpful for getting to know if the mutation is pathogenic or not, while for SCN1A it does not improve very much so we could conclude that the pathogenicity of SCN1A mutation is not likely to have strong relation with its own amino acid structure, hence we should probably try to seek other methods to get better prediction results. Therefore for SCN2A and SCN8A we can tell in a higher accuracy level that a mutation/change of amino acid will result in function disorder.

Classifier/ Test-Set	AUC all-trained	AUC PolyPhen-2	p-value
SCN1A	0.81627	0.84249	0.1875
SCN2A	0.85326	0.76269	0.04457
SCN8A	0.74180	0.76358	0.2465

Table 11: Delong test for ROC curves - All training vs PolyPhen-2

We can run similar tests between model trained by all data and PolyPhen-2, and get Table 11. We could see that for SCN1A and SCN8A testing sets there are no significant different between the model trained and PolyPhen-2 at α -level 0.05. And for SCN2A testing set the result is better than PolyPhen-2 (α=0.05). But we could also notice that the model trained with separate methods have a much smaller p-value for SCN2A data.

Comparing the p-values given in Table 8, we could find that AUC of SCN1A rises from 0.816 to 0.887 by training separately, and that this difference comes from the rise of accuracy in predicting pathogenic samples,

with p-value 0.02. And for SCN2A data AUC increases to 0.879 from 0.853 by separate-training, and this increase comes mostly from benign samples. For SCN8A data AUC is 0.741 when training separately and becomes 0.841 when training separately, and similarly we could read from Table 8 that this difference comes mostly from pathogenic samples with a p-value of z-test very close to 0.

To sum up, we could conclude that separate training model performs best on all 3 datasets, whole-training model works slightly better than PolyPhen-2 on SCN2A data-set, while the rest datasets do not have significant improvement (α-level 0.05).

4 Discussion

4.1 More to Do with Our Model

More features-selection methods could be applied to explore better accuracy. If we know more about the mechanism of how these proteins are interacting with ion gate or how will protein function change with amino acid change, we could analyze these structural data more effectively. If we do not have more information from biological process, we can still apply mathematical methods such as greedy algorithm to see which combination may give highest power when doing prediction, while results like this will only serve as a good indication of a better classifier based on our dataset but still need further test and biological foundation to confirm the inference.

4.2 Other Methods to Be Considered

There are a few more methods that could be tried with more information and we think a logistic regression method or recursive partition would be good ideas to explore with. Here are our thoughts and concerns.

4.2.1 Additive Model of Logistic Regression

There's some parts of missing data in out data-set. If our data-set is complete it would be perfect to try a logistic regression of additive models. So even for the data-columns with relative complete data there might be slightly amount of data missing from that column. In this case it's risky to conduct a logistic regression to get prediction probabilities as it's hard to give the missing part a value. If the data is more complete or if there are good methodologies to make up the missing entries, a logistic regression additive model would be a good choice to explore.

4.2.2 Random Forests/Recursive Partitioning

Recursive partitioning could also be performed for this data-set and it should be an interesting direction to try. If we know more about biological organism of different features,it would be easier to select a subsets to do a recursive partitioning. If we have more information about how to put different weight on different features, we could probably build a effective tree applying the recursive partitioning model with different levels to analyze the possible prediction results step by step.

So given more time and information the two methods above could be tried and we believe that will give interesting results or better accuracy.

5 Appendix – R Code

```
#################
## SCN1A model ##
#################
set.seed(1000)
library(ggplot2)
library(e1071)
library(lattice)
library(caret)
library(ISLR)
library(tree)
SCN1full<-read.csv("file:///Users/jingli/Desktop/readydata/1all.csv",
                   header=T,na.strings=c("?",NA))
SCN1d<-
 SCN1full[,c(12,21,22,23,24,25,26,27,28,49,50,51,52,53,54,55,56)]


#choose the columns needed
#SCN1d[is.na(SCN1d)] <- 0
SCN1d[is.na(SCN1d)] <- 0
      IdP1<-SCN1d$IdPmax      #change the string types of data-columns
      SCN1d$IdPmax<-as.numeric(IdP1)
      IdQ1<-SCN1d$IdQmin
      SCN1d$IdQmin<-as.numeric(IdQ1)
      IdPSNP<-SCN1d$IdPSNP
      IdPS1<-SCN1d$IdPSNP
      SCN1d$IdPSNP<-as.numeric(IdPS1)
      MinD1<-SCN1d$MinDJxn
      SCN1d$MinDJxn<-as.numeric(MinD1)
      path1<-SCN1d$pathogenic
      SCN1d$pathogenic<-as.factor(path1)
       SCN1d$MSAv<-as.factor(SCN1d$MSAv)
 SCN1d$Transv<-as.factor(SCN1d$Transv)
 SCN1d$CodPos<-as.factor(SCN1d$CodPos)
 SCN1d$CpG<-as.factor(SCN1d$CpG)
 SCN1d$IdQmin<-as.numeric(SCN1d$IdQmin)


    allrows1<-1:nrow(SCN1d)
    trainrows11<-  #randomly choose 2/3 as train rows
    sample(allrows1,replace=F,size=2/3*length(allrows1))
    test_cvrows1<-allrows[-trainrows1]
    testrows1<-
```

```
 sample(test_cvrows1,replace=F,size=0.5*length(test_cvrows1))
        train1SCN<-SCN1d[trainrows1,]
           test1SCN<-SCN1d[testrows1,]
           test11<-allrows[-trainrows1]

test11SCN<-SCN1d[-trainrows11,]          #divide into 3 part, part1/3
SCN1model<-naiveBayes(train1SCN[,-1],train1SCN[,1])
SCN1_predictions<-predict(SCN1model,test11SCN,type="raw")
#SCN1_predictions                       #part1/3 predictions
SCN1_predictions1<-predict(SCN1model,test11SCN)
SCN1_predictions1
test12<-sample(trainrows1,replace=F,size=0.5*length(trainrows81))
trainrows13<-c(test12,test11) #3-fold Cross Validation Sets
test13<-allrows1[-trainrows13]
trainrows12<-c(test11,test13)
testrowscv1<-sample(test11,replace=F,size=0.5*length(test_cvrows1))

trainSCN1<-SCN1d[trainrows11,]
trainSCN12<-SCN1d[trainrows12,]
trainSCN13<-SCN1d[trainrows13,]
   test12SCN<-SCN1d[test12,]    #2/3
   test13SCN<-SCN1d[test13,]    #3/3
SCN1model1<-naiveBayes(trainSCN1[,-1],trainSCN1[,1],na.action = na.pass)
SCN1model2<-naiveBayes(trainSCN12[,-1],trainSCN12[,1],na.action = na.pass)
SCN1model3<-naiveBayes(trainSCN13[,-1],trainSCN13[,1],na.action = na.pass)
    SCN1_predictions1<-predict(SCN1model1,test11SCN)
    SCN1_predictions1        #SCN1A data 0-1 prediction for part 1/3 with cut-off 0.5
    SCN1_predictions2<-predict(SCN1model2,test12SCN)
    SCN1_predictions2        #part 2/3
    SCN1_predictions3<-predict(SCN1model3,test13SCN)
    SCN1_predictions3        #part 3/3
    SCN1_predictionr1<-predict(SCN1model1,test11SCN,type="raw")
   ##SCN1_predictionr1       #raw predictions for part 1/3 with probabilities
SCN1_predictionr2<-predict(SCN1model2,test12SCN,type="raw")
   ##SCN1_predictionr2       #part 2/3
SCN1_predictionr3<-predict(SCN1model3,test13SCN,type="raw")
   ##SCN1_predictionr3       #part 3/3
length(SCN1_predictions1)
length(trainrows1)
dim(test11SCN)

 itp1=0    #numbers of TP TN FP FN in part 1/3
 jtp1=0    #variables set up for counting in the for-loop
 itn1=0
```

```
 jtn1=0
 ifn1=0
 jfn1=0
 ifp1=0
 jfp1=0
 for(itp1 in 1:156){  #COUNTING TRUE POSITIVE from part 1/3 of SCN1A
       if(SCN1_predictions1[itp1]==test11SCN
 $pathogenic[itp1]&&SCN1_predictions1[itp1]==1){jtp1=jtp1+1}}
  for(itn1 in 1:156){ #TRUE NEGATIVE
       if(SCN1_predictions1[itn1]==test11SCN
 $pathogenic[itn1]&&SCN1_predictions1[itn1]==0){jtn1=jtn1+1}}
  for(ifn1 in 1:156){ #FALSE NEGATIVE
       if(SCN1_predictions1[ifn1]!=test11SCN
 $pathogenic[ifn1]&&SCN1_predictions1[ifn1]==0){jfn1=jfn1+1}}
  for(ifp1 in 1:156){ #FALSE POSITIVE
       if(SCN1_predictions1[ifp1]!=test11SCN
 $pathogenic[ifp1]&&SCN1_predictions1[ifp1]==1){jfp1=jfp1+1}}
 jtp1
 jtn1
 jfn1
 jfp1

jtp1/(jtp1+jfn1) #sensitivity TP/(TP+FN)
jtn1/(jtn1+jfp1) #specificity TN/(TN+FP)

 library(pROC)
 library(stats)
 plot(roc(test11SCN$pathogenic, SCN1_predictions[,2], direction="<"),
         col="blue", lwd=3, main="SCN1A model",print.auc=TRUE)
  lines(roc(SCN1full[-trainrows1,]$pathogenic, SCN1full[-trainrows1,]$pph2_prob,
         direction="<"),
         col="black", lwd=3, main="SCN1A model",print.auc=TRUE)

itp12=0  #testing 2/3 summary
jtp12=0  #numbers of TP TN FP FN in part 2/3
itn12=0  #variables setup for counting
jtn12=0
ifn12=0
jfn12=0
ifp12=0
jfp12=0
dim(test12SCN)
for(itp12 in 1:150){ #COUNTING TRUE POSITIVE from part 2/3 of SCN1A
if((SCN1_predictions2[itp12]==test12SCN$pathogenic[itp12])
```

```
&&(SCN1_predictions2[itp12]==1)){jtp12=jtp12+1}}
 for(itn12 in 1:150){ #TRUE NEGATIVE
      if((SCN1_predictions2[itn12]==test12SCN$pathogenic[itn12])
&&(SCN1_predictions2[itn12]==0)){jtn12=jtn12+1}}
 for(ifn12 in 1:150){ #FALSE NEGATIVE
      if((SCN1_predictions2[ifn12]!=test12SCN$pathogenic[ifn12])
                  &&(SCN1_predictions2[ifn12]==0)){jfn12=jfn12+1}}
 for(ifp12 in 1:150){ #FALSE POSITIVE
      if((SCN1_predictions2[ifp12]!=test12SCN
$pathogenic[ifp12])&&(SCN1_predictions2[ifp12]==1)){jfp12=jfp12+1}}

jtp12
jtn12
jfn12
jfp12


itp13=0  #testing 3/3 summary
jtp13=0  #numbers of TP TN FP FN in part 3/3
itn13=0  #variables set-up for counting
jtn13=0
ifn13=0
jfn13=0
ifp13=0
jfp13=0
dim(test13SCN)
for(itp13 in 1:165){ #COUNTING TRUE POSITIVE from part 3/3 of SCN1A
if((SCN1_predictions3[itp13]==test83SCN$pathogenic[itp13])
&&(SCN1_predictions3[itp13]==1)){jtp13=jtp13+1}}
 for(itn13 in 1:165){ #TRUE NEGATIVE
      if((SCN1_predictions3[itn13]==test13SCN$pathogenic[itn13])
&&(SCN1_predictions3[itn13]==0)){jtn13=jtn13+1}}
 for(ifn13 in 1:165){ #FALSE NEGATIVE
      if((SCN1_predictions3[ifn13]!=test13SCN$pathogenic[ifn13])
                    &&(SCN1_predictions3[ifn13]==0)){jfn13=jfn13+1}}
 for(ifp13 in 1:165){ #FALSE POSITIVE
      if((SCN1_predictions3[ifp13]!=test13SCN
$pathogenic[ifp13])&&(SCN1_predictions3[ifp13]==1)){jfp13=jfp13+1}}

jtp13
jtn13
jfn13
jfp13
SCN1TP<-jtp1+jtp12+jtp13   #ADD ALL COUNTINGS TOGETHER TRUE POSITIVE
```

```
SCN1TN<-jtn1+jtn12+jtn13   #TRUE NEGATIVE
SCN1FN<-jfn1+jfn12+jfn13   #FALSE NEGATIVE
SCN1FP<-jfp1+jfp12+jfp13    #FALSE POSITIVE
 SCN1TP  #NUMBERS OUTPUT
 SCN1TN
 SCN1FN
 SCN1FP
Test1SCN<-rbind(test11SCN,test12SCN,test13SCN)  #test set summary total
dim(Test1SCN)
SCN1PR<-rbind(SCN1_predictionr1,SCN1_predictionr2,SCN1_predictionr3)
plot(roc(Test1SCN$pathogenic, SCN1PR[,2], direction="<"),
         col="blue", lwd=3, main="SCN1A 3-CV model",print.auc=TRUE)
lines(roc(SCN1full$pathogenic, SCN1full$pph2_prob, direction="<"),
         col="black", lwd=3,print.auc=TRUE)
roc.test(roc(Test1SCN$pathogenic, SCN1PR[,2]),roc(SCN1full$pathogenic,
                           SCN1full$pph2_prob),method="delong",alternative="greater")

############
#  SCN2A  #
############

set.seed(1000)
SCN2full<-read.csv("file:///Users/jingli/Desktop/readydata/2all.csv",
                                       header=T,na.strings=c("?",NA))
SCN2d<-
      SCN2full[,c(12,21,22,23,24,25,26,27,28,49,50,51,52,53,54,55,56)]
SCN2d[is.na(SCN2d)] <- 0

    IdP2<-SCN2d$IdPmax
    SCN2d$IdPmax<-as.numeric(IdP2)
    IdQ2<-SCN2d$IdQmin
    SCN2d$IdQmin<-as.numeric(IdQ2)
    IdPSNP<-SCN2d$IdPSNP
    IdPS2<-SCN2d$IdPSNP
    SCN2d$IdPSNP<-as.numeric(IdPS2)
    MinD2<-SCN2d$MinDJxn
    SCN2d$MinDJxn<-as.numeric(MinD2)
    path2<-SCN2d$pathogenic
    SCN2d$pathogenic<-as.factor(path2)
    SCN2d$MSAv<-as.factor(SCN2d$MSAv)
    SCN2d$Transv<-as.factor(SCN2d$Transv)
    SCN2d$CodPos<-as.factor(SCN2d$CodPos)
    SCN2d$CpG<-as.factor(SCN2d$CpG)
    SCN2d$IdQmin<-as.numeric(SCN2d$IdQmin)
```

```
     allrows2<-1:nrow(SCN2d)
     trainrows2<-sample(allrows2,replace=F,size=2/3*length(allrows2))
test21<-allrows[-trainrows2]
test_cvrows2<-allrows2[-trainrows2]
     testrows2<-
 sample(test_cvrows2,replace=F,size=0.5*length(test_cvrows2))
     train2SCN<-SCN2d[trainrows2,]
        test2SCN<-SCN2d[testrows2,]
        SCN2model<-naiveBayes(train2SCN[,-1],train2SCN[,1])
  test21SCN<-SCN2d[-trainrows2,]
  SCN2_predictionss<-predict(SCN2model,test21SCN,type="raw")
  SCN2_prediction21<-predict(SCN2model,test21SCN)
  SCN2_prediction21
  ength(SCN2_prediction21)

dim(test2SCN)
dim(test21SCN)

#######
 test22<-sample(trainrows1,replace=F,size=0.5*length(trainrows2))
       #3-fold Cross Validation Set
 trainrows23<-c(test22,test21)
 test23<-allrows1[-trainrows23]
 trainrows22<-c(test21,test23)
 testrowscv1<-sample(test11,replace=F,size=0.5*length(test_cvrows1))
 trainSCN2<-SCN2d[trainrows11,]
 trainSCN22<-SCN2d[trainrows12,]
 trainSCN23<-SCN2d[trainrows13,]
test22SCN<-SCN2d[test12,]    #2/3
test23SCN<-SCN2d[test13,]    #3/3

###model###
SCN2model1<-naiveBayes(trainSCN2[,-1],trainSCN2[,1],na.action = na.pass)
SCN2model2<-naiveBayes(trainSCN22[,-1],trainSCN22[,1],na.action = na.pass)
SCN2model3<-naiveBayes(trainSCN23[,-1],trainSCN23[,1],na.action = na.pass)
    SCN2_predictions1<-predict(SCN2model1,test21SCN)
    SCN2_predictions1
    SCN2_predictions2<-predict(SCN2model2,test22SCN)
    SCN2_predictions2
    SCN2_predictions3<-predict(SCN2model3,test23SCN)
    SCN2_predictions3
    SCN2_predictionr1<-predict(SCN2model1,test21SCN,type="raw")
    ##SCN2_predictionr1
```

```
    SCN2_predictionr2<-predict(SCN2model2,test22SCN,type="raw")
    ##SCN2_predictionr2
    SCN2_predictionr3<-predict(SCN2model3,test23SCN,type="raw")
    ##SCN2_predictionr3
length(SCN2_predictions1)

#####
length(SCN2_prediction22)
dim(test2SCN)

 itp2=0
 jtp2=0
 itn2=0
 jtn2=0
 ifn2=0
 jfn2=0
 ifp2=0
 jfp2=0
 for(itp2 in 1:152){ #TRUE POSITIVE
       if(SCN2_prediction21[itp2]==test21SCN
 $pathogenic[itp2]&&SCN2_prediction21[itp2]==1){jtp2=jtp2+1}}
  for(itn2 in 1:152){ #TRUE NEGATIVE
       if(SCN2_prediction21[itn2]==test21SCN
 $pathogenic[itn2]&&SCN2_prediction21[itn2]==0){jtn2=jtn2+1}}
  for(ifn2 in 1:152){ #FALSE NEGATIVE
       if(SCN2_prediction21[ifn2]!=test21SCN
 $pathogenic[ifn2]&&SCN2_prediction21[ifn2]==0){jfn2=jfn2+1}}
  for(ifp2 in 1:152){ #FALSE POSITIVE
       if(SCN2_prediction21[ifp2]!=test21SCN
 $pathogenic[ifp2]&&SCN2_prediction21[ifp2]==1){jfp2=jfp2+1}}

 jtp2
 jtn2
 jfn2
 jfp2

 jtp2/(jtp2+jfn2) #sensitivity TP/(TP+FN)
 jtn2/(jtn2+jfp2) #specificity TN/(TN+FP)

#####
itp22=0
jtp22=0
itn22=0
jtn22=0
```

```
ifn22=0
jfn22=0
ifp22=0
jfp22=0

test22SCN$pathogenic[is.na(test22SCN$pathogenic)] <- 0
 for(itp22 in 1:150){ #TRUE POSITIVE
       if(SCN2_predictions2[itp22]==test22SCN
 $pathogenic[itp22]&&SCN2_predictions2[itp22]==1){jtp22=jtp22+1}}
  for(itn22 in 1:150){ #TRUE NEGATIVE
       if(SCN2_predictions2[itn22]==test22SCN
 $pathogenic[itn22]&&SCN2_predictions2[itn22]==0){jtn22=jtn22+1}}
  for(ifn22 in 1:150){ #FALSE NEGATIVE
       if(SCN2_predictions2[ifn22]!=test22SCN
 $pathogenic[ifn22]&&SCN2_predictions2[ifn22]==0){jfn22=jfn22+1}}
  for(ifp22 in 1:150){ #FALSE POSITIVE
       if(SCN2_predictions2[ifp22]!=test22SCN
 $pathogenic[ifp22]&&SCN2_predictions2[ifp22]==1){jfp22=jfp2+1}}
 jtp22
 jtn22
 jfn22
 jfp22
 jtp22/(jtp22+jfn22) #sensitivity TP/(TP+FN)
 jtn22/(jtn22+jfp22) #specificity TN/(TN+FP)
####
itp23=0
jtp23=0
itn23=0
jtn23=0
ifn23=0
jfn23=0
ifp23=0
jfp23=0

test23SCN$pathogenic[is.na(test23SCN$pathogenic)] <- 0
 for(itp23 in 1:150){ #TRUE POSITIVE
       if(SCN2_predictions3[itp23]==test23SCN
 $pathogenic[itp23]&&SCN2_predictions3[itp23]==1){jtp23=jtp23+1}}
  for(itn23 in 1:150){ #TRUE NEGATIVE
       if(SCN2_predictions3[itn23]==test23SCN
 $pathogenic[itn23]&&SCN2_predictions3[itn23]==0){jtn23=jtn23+1}}
  for(ifn23 in 1:150){ #FALSE NEGATIVE
       if(SCN2_predictions3[ifn23]!=test23SCN
 $pathogenic[ifn23]&&SCN2_predictions3[ifn23]==0){jfn23=jfn23+1}}
```

```
  for(ifp23 in 1:150){ #FALSE POSITIVE
       if(SCN2_predictions3[ifp23]!=test23SCN
 $pathogenic[ifp23]&&SCN2_predictions3[ifp23]==1){jfp23=jfp2+1}}

 jtp23 #outputs of counting
 jtn23
 jfn23
 jfp23

 jtp23/(jtp23+jfn23) #sensitivity TP/(TP+FN)
 jtn23/(jtn23+jfp23) #specificity TN/(TN+FP)

#####
SCN2TP<-jtp2+jtp22+jtp23
SCN2TN<-jtn2+jtn22+jtn23
SCN2FN<-jfn2+jfn22+jfn23
SCN2FP<-jfp2+jfp22+jfp23
SCN2TP
SCN2TN
SCN2FN
SCN2FP
Test2SCN<-rbind(test21SCN,test22SCN,test23SCN)
dim(Test2SCN)
SCN2PR<-rbind(SCN2_predictionr1,SCN2_predictionr2,SCN2_predictionr3)
plot(roc(Test2SCN$pathogenic, SCN2PR[,2], direction="<"),
         col="blue", lwd=3, main="SCN2A 3-CV model",print.auc=TRUE)
lines(roc(SCN2full$pathogenic, SCN2full$pph2_prob, direction="<"),
         col="black", lwd=3,print.auc=TRUE)

###DeLong test###
roc.test(roc(Test2SCN$pathogenic, SCN2PR[,2]),roc(SCN2full$pathogenic,
            SCN2full$pph2_prob),method="delong",alternative="greater")

############
#  SCN8A  #
############

set.seed(1000)
library(ggplot2)
library(e1071)
library(lattice)
library(caret)
library(ISLR)
library(tree)
```

```
   SCN8full<-read.csv("file:///Users/jingli/Desktop/readydata/8all.csv",
       header=T,na.strings=c("?",NA))
   SCN8d<-SCN8full[,c(12,21,22,23,24,25,26,27,28,49,50,51,52,53,54,55,56)]
dim(SCN8d)
str(SCN8d)
       IdP8<-SCN8d$IdPmax
       SCN8d$IdPmax<-as.numeric(IdP8)
       IdQ8<-SCN8d$IdQmin
       SCN8d$IdQmin<-as.numeric(IdQ8)
       IdPSNP<-SCN8d$IdPSNP
       IdPS8<-SCN8d$IdPSNP
       SCN8d$IdPSNP<-as.numeric(IdPS8)
       MinD8<-SCN8d$MinDJxn
       SCN8d$MinDJxn<-as.numeric(MinD8)
       path8<-SCN8d$pathogenic
       SCN8d$pathogenic<-as.factor(path8)
       ###

SCN8d$MSAv<-as.factor(SCN8d$MSAv)
SCN8d$Transv<-as.factor(SCN8d$Transv)
SCN8d$Nobs<-as.numeric(SCN8d$Nobs)
SCN8d$CodPos<-as.factor(SCN8d$CodPos)
SCN8d$CpG<-as.factor(SCN8d$CpG)
SCN8d$IdQmin<-as.numeric(SCN8d$IdQmin)
     path8<-SCN8d$pathogenic
SCN8d$pathogenic<-as.factor(path8)

 str(SCN8d)

 allrows8<-1:nrow(SCN8d)
 trainrows81<-sample(allrows8,replace=F,size=2/3*length(allrows8))
 test81<-allrows8[-trainrows81]   #1/3
 test82<-sample(trainrows8,replace=F,size=0.5*length(trainrows81))
 #3-fold Cross Validation Set
 trainrows83<-c(test82,test81)
 test83<-allrows8[-trainrows83]
 trainrows82<-c(test81,test83)
 testrowscv8<-sample(test81,replace=F,size=0.5*length(test_cvrows8))
 trainSCN8<-SCN8d[trainrows81,]
 trainSCN82<-SCN8d[trainrows82,]
 trainSCN83<-SCN8d[trainrows83,]
    test88SCN<-SCN8d[-trainrows81,]
    test82SCN<-SCN8d[test82,]    #2/3
```

```
    test83SCN<-SCN8d[test83,]    #3/3
SCN8model1<-naiveBayes(trainSCN8[,-1],trainSCN8[,1],na.action = na.pass)
SCN8model2<-naiveBayes(trainSCN82[,-1],trainSCN82[,1],na.action = na.pass)
SCN8model3<-naiveBayes(trainSCN83[,-1],trainSCN83[,1],na.action = na.pass)
      SCN8_predictions1<-predict(SCN8model1,test88SCN)
      SCN8_predictions1
      SCN8_predictions2<-predict(SCN8model2,test82SCN)
      SCN8_predictions2
      SCN8_predictions3<-predict(SCN8model3,test83SCN)
      SCN8_predictions3
      SCN8_predictionr1<-predict(SCN8model1,test88SCN,type="raw")
      ##SCN8_predictionr1
      SCN8_predictionr2<-predict(SCN8model2,test82SCN,type="raw")
      ##SCN8_predictionr2
      SCN8_predictionr3<-predict(SCN8model3,test83SCN,type="raw")
      ##SCN8_predictionr3
length(SCN8_predictions1)
length(trainrows8)
itp8=0  #testing 1/3 summary
jtp8=0
itn8=0
jtn8=0
ifn8=0
jfn8=0
ifp8=0
jfp8=0
for(itp8 in 1:151){ #TRUE POSITIVE
if((SCN8_predictions1[itp8]==test88SCN$pathogenic[itp8])
&&(SCN8_predictions1[itp8]==1)){jtp8=jtp8+1}}
 for(itn8 in 1:151){ #TRUE NEGATIVE
      if((SCN8_predictions1[itn8]==test88SCN$pathogenic[itn8])
&&(SCN8_predictions1[itn8]==0)){jtn8=jtn8+1}}
 for(ifn8 in 1:151){ #FALSE NEGATIVE
      if((SCN8_predictions1[ifn8]!=test88SCN$pathogenic[ifn8])
      &&(SCN8_predictions1[ifn8]==0)){jfn8=jfn8+1}}
 for(ifp8 in 1:151){ #FALSE POSITIVE
      if((SCN8_predictions1[ifp8]!=test88SCN
$pathogenic[ifp8])&&(SCN8_predictions1[ifp8]==1)){jfp8=jfp8+1}}

jtp8
jtn8
jfn8
jfp8
jtp8/(jtp8+jfn8) #sensitivity TP/(TP+FN)
```

```
jtn8/(jtn8+jfp8) #specificity TN/(TN+FP)
library(pROC)
library(stats)
plot(roc(test88SCN$pathogenic, SCN8_predictions[,2], direction="<"),
        col="blue", lwd=3, main="SCN8A model",print.auc=TRUE)
 lines(roc(SCN8full[-trainrows81,]$pathogenic, SCN8full[-trainrows81,]$pph2_prob,
        direction="<"),
        col="black", lwd=3, main="SCN8A model",print.auc=TRUE)

roc.test(roc(SCN8full[-trainrows81,]$pathogenic, SCN8_predictions[,2]),
 roc(SCN8full[-trainrows81,]$pathogenic, SCN8full[-trainrows81,]$pph2_prob),
 paired=TRUE, method="delong",alternative="greater")
itp82=0  #testing 2/3 summary
jtp82=0
itn82=0
jtn82=0
ifn82=0
jfn82=0
ifp82=0
jfp82=0
dim(test82SCN)
for(itp82 in 1:150){ #TRUE POSITIVE
if((SCN8_predictions2[itp82]==test82SCN$pathogenic[itp82])
&&(SCN8_predictions2[itp82]==1)){jtp82=jtp82+1}}
 for(itn82 in 1:150){ #TRUE NEGATIVE
      if((SCN8_predictions2[itn82]==test82SCN$pathogenic[itn82])
&&(SCN8_predictions2[itn82]==0)){jtn82=jtn82+1}}
 for(ifn82 in 1:150){ #FALSE NEGATIVE
      if((SCN8_predictions2[ifn82]!=test88SCN$pathogenic[ifn82])
      &&(SCN8_predictions2[ifn82]==0)){jfn82=jfn82+1}}
 for(ifp82 in 1:150){ #FALSE POSITIVE
      if((SCN8_predictions2[ifp82]!=test82SCN
$pathogenic[ifp82])&&(SCN8_predictions2[ifp82]==1)){jfp82=jfp82+1}}

jtp82
jtn82
jfn82
jfp82
itp83=0  #testing 3/3 summary
jtp83=0
itn83=0
jtn83=0
ifn83=0
jfn83=0
```

```
ifp83=0
jfp83=0
dim(test83SCN)
for(itp83 in 1:151){ #TRUE POSITIVE
if((SCN8_predictions3[itp83]==test83SCN$pathogenic[itp83])
&&(SCN8_predictions3[itp83]==1)){jtp83=jtp83+1}}
 for(itn83 in 1:151){ #TRUE NEGATIVE
      if((SCN8_predictions3[itn83]==test83SCN$pathogenic[itn83])
&&(SCN8_predictions3[itn83]==0)){jtn83=jtn83+1}}
 for(ifn83 in 1:151){ #FALSE NEGATIVE
      if((SCN8_predictions3[ifn83]!=test83SCN$pathogenic[ifn83])
      &&(SCN8_predictions3[ifn83]==0)){jfn83=jfn83+1}}
 for(ifp83 in 1:151){ #FALSE POSITIVE
      if((SCN8_predictions3[ifp83]!=test83SCN
$pathogenic[ifp83])&&(SCN8_predictions3[ifp83]==1)){jfp83=jfp83+1}}

jtp83 #summary total
jtn83
jfn83
jfp83

SCN8TP<-jtp8+jtp82+jtp83
SCN8TN<-jtn8+jtn82+jtn83
SCN8FN<-jfn8+jfn82+jfn83
SCN8FP<-jfp8+jfp82+jfp83
SCN8TP
SCN8TN
SCN8FN
SCN8FP

Test8SCN<-rbind(test88SCN,test82SCN,test83SCN)
dim(Test8SCN)
SCN8PR<-rbind(SCN8_predictionr1,SCN8_predictionr2,SCN8_predictionr3)
plot(roc(Test8SCN$pathogenic, SCN8PR[,2], direction="<"),
         col="blue", lwd=3, main="SCN8A 3-CV model",print.auc=TRUE)
lines(roc(SCN8full$pathogenic, SCN8full$pph2_prob, direction="<"),
         col="black", lwd=3,print.auc=TRUE)
roc.test(roc(Test8SCN$pathogenic, SCN8PR[,2]),roc(SCN8full$pathogenic,
SCN8full$pph2_prob),method="delong",alternative="greater")

################
# all data model
################
set.seed(1000)
```

```
traintl13SCN<-rbind(trainSCN1,trainSCN2,trainSCN8) ###CV on all data-part 1/3

 traintl23SCN<-rbind(trainSCN12,trainSCN22,trainSCN82) ###CV on all data-part 2/3
 traintl33SCN<-rbind(trainSCN13,trainSCN23,trainSCN83) ###CV on all data-part 3/3


 SCNtlmodel13<-naiveBayes(traintl13SCN[,-1],traintl13SCN[,1])
 SCNtlPrediction11<-predict(SCNtlmodel13,test11SCN)      #Cross Validation Part 1/3
 SCNtlmodel23<-naiveBayes(traintl23SCN[,-1],traintl23SCN[,1])
 SCNtlPrediction12<-predict(SCNtlmodel23,test12SCN)       #Cross Validation Part 2/3
 SCNtlmodel33<-naiveBayes(traintl33SCN[,-1],traintl33SCN[,1])
 SCNtlPrediction13<-predict(SCNtlmodel33,test13SCN)      #Cross Validation Part 3/3
                                                  #raw prediction with probabilities
 SCNtlPrediction11r<-predict(SCNtlmodel13,test11SCN,type="raw")
 SCNtlPrediction12r<-predict(SCNtlmodel23,test12SCN,type="raw")
 SCNtlPrediction13r<-predict(SCNtlmodel33,test13SCN,type="raw")
 SCN1PRT<-rbind(SCNtlPrediction11r,SCNtlPrediction12r,SCNtlPrediction23r)
 plot(roc(Test1SCN$pathogenic, SCN1PRT[,2], direction="<"),
          col="purple", lwd=3, main="All data 3-CV model - SCN1A",print.auc=TRUE)



SCNtlPrediction21<-predict(SCNtlmodel13,test21SCN)
SCNtlPrediction22<-predict(SCNtlmodel23,test22SCN)
SCNtlPrediction23<-predict(SCNtlmodel33,test23SCN)

Test2TSCN<-c(SCNtlPrediction21, SCNtlPrediction22, SCNtlPrediction23)

SCNtlPrediction21r<-predict(SCNtlmodel13,test21SCN,type="raw")
SCNtlPrediction22r<-predict(SCNtlmodel23,test22SCN,type="raw")
SCNtlPrediction23r<-predict(SCNtlmodel33,test23SCN,type="raw")
SCN2PRT<-rbind(SCNtlPrediction21r,SCNtlPrediction22r,SCNtlPrediction23r)
plot(roc(Test2SCN$pathogenic, SCN2PRT[,2], direction="<"),
          col="purple", lwd=3, main="All data 3-CV model - SCN2A",print.auc=TRUE)



SCNtlPrediction81<-predict(SCNtlmodel13,test88SCN)
SCNtlPrediction82<-predict(SCNtlmodel23,test82SCN)
SCNtlPrediction83<-predict(SCNtlmodel33,test83SCN)


SCNtlPrediction81r<-predict(SCNtlmodel13,test88SCN,type="raw")
SCNtlPrediction82r<-predict(SCNtlmodel23,test82SCN,type="raw")
```

```
SCNtlPrediction83r<-predict(SCNtlmodel33,test83SCN,type="raw")
SCN8PRT<-rbind(SCNtlPrediction81r,SCNtlPrediction82r,SCNtlPrediction83r)
plot(roc(Test8SCN$pathogenic, SCN8PRT[,2], direction="<"),
          col="purple", lwd=3, main="All data 3-CV model - SCN8A",print.auc=TRUE)


################################
# all trained model SCN1A summary
################################

itpa1=0  #testing 1/3 summary
jtpa1=0  #variables set up for counting loop
itna1=0  #i for loop, j for counting
jtna1=0
ifna1=0
jfna1=0
ifpa1=0
jfpa1=0
dim(test11SCN)
for(itpa1 in 1:156){ #COUNTING TRUE POSITIVE part 1/3 of SCN1A data
    if((SCNtlPrediction11[itpa1]==test11SCN$pathogenic[itpa1])
       &&(SCNtlPrediction11[itpa1]==1)){jtpa1=jtpa1+1}}
for(itna1 in 1:156){ #TRUE NEGATIVE
    if((SCNtlPrediction11[itna1]==test11SCN$pathogenic[itna1])
       &&(SCNtlPrediction11[itna1]==0)){jtna1=jtna1+1}}
for(ifna1 in 1:156){ #FALSE NEGATIVE
    if((SCNtlPrediction11[ifna1]!=test11SCN$pathogenic[ifna1])
       &&(SCNtlPrediction11[ifna1]==0)){jfna1=jfna1+1}}
for(ifpa1 in 1:156){ #FALSE POSITIVE
    if((SCNtlPrediction11[ifpa1]!=test11SCN
        $pathogenic[ifpa1])&&(SCNtlPrediction11[ifpa1]==1)){jfpa1=jfpa1+1}}

jtpa1
jtna1
jfna1
jfpa1
jtpa1/(jtpa1+jfna1) #sensitivity TP/(TP+FN)
jtna1/(jtna1+jfpa1) #specificity TN/(TN+FP)


itpa12=0  #testing 2/3 summary
jtpa12=0  #variable set up
itna12=0
jtna12=0
```

```
ifna12=0
jfna12=0
ifpa12=0
jfpa12=0
dim(test12SCN)
for(itpa12 in 1:150){ #COUNTING TRUE POSITIVE part 2/3 of 1A data
    if((SCNtlPrediction12[itpa12]==test12SCN$pathogenic[itpa12])
       &&(SCNtlPrediction12[itpa12]==1)){jtpa12=jtpa12+1}}
for(itna12 in 1:150){ #TRUE NEGATIVE
    if((SCNtlPrediction12[itna12]==test12SCN$pathogenic[itna12])
       &&(SCNtlPrediction12[itna12]==0)){jtna12=jtna12+1}}
for(ifna12 in 1:150){ #FALSE NEGATIVE
    if((SCNtlPrediction12[ifna12]!=test12SCN$pathogenic[ifna12])
       &&(SCNtlPrediction22[ifna12]==0)){jfna12=jfna12+1}}
for(ifpa12 in 1:150){ #FALSE POSITIVE
    if((SCNtlPrediction12[ifpa12]!=test12SCN
        $pathogenic[ifpa12])&&(SCNtlPrediction22[ifpa12]==1)){jfpa12=jfpa12+1}}

jtpa12
jtna12
jfna12
jfpa12
itpa13=0  #testing 3/3 summary
jtpa13=0
itna13=0
jtna13=0
ifna13=0
jfna13=0
ifpa13=0
jfpa13=0
dim(test13SCN)
for(itpa13 in 1:165){ #COUNTING TRUE POSITIVE part 3/3 of 1A data
    if((SCNtlPrediction13[itpa13]==test13SCN$pathogenic[itpa13])
       &&(SCNtlPrediction13[itpa13]==1)){jtpa13=jtpa13+1}}
for(itna13 in 1:165){ #TRUE NEGATIVE
    if((SCNtlPrediction13[itna13]==test13SCN$pathogenic[itna13])
       &&(SCNtlPrediction13[itna13]==0)){jtna13=jtna13+1}}
for(ifna13 in 1:165){ #FALSE NEGATIVE
    if((SCNtlPrediction13[ifna13]!=test13SCN$pathogenic[ifna13])
       &&(SCNtlPrediction13[ifna13]==0)){jfna13=jfna13+1}}
for(ifpa13 in 1:165){ #FALSE POSITIVE
    if((SCNtlPrediction13[ifpa13]!=test23SCN
        $pathogenic[ifpa13])&&(SCNtlPrediction13[ifpa13]==1)){jfpa13=jfpa13+1}}
```

```
jtpa13 # Part 3/3 of 1A data output
jtna13
jfna13
jfpa13

SCN1aTP<-jtpa1+jtpa12+jtpa13   #Adding part 1/3 2/3 3/3 together True Positive SCN1A
SCN1aTN<-jtna1+jtna12+jtna13   #Adding part 1/3 2/3 3/3 together True Negative SCN1A
SCN1aFN<-jfna1+jfna12+jfna13  #False Negative
SCN1aFP<-jfpa1+jfpa12+jfpa13   #False Positive

SCN1aTP # Total Output
SCN1aTN
SCN1aFN
SCN1aFP


########################
#all trained 2A summary
########################

itpa2=0  #testing 1/3 summary
jtpa2=0  #variables set up for counting loop
itna2=0
jtna2=0
ifna2=0
jfna2=0
ifpa2=0
jfpa2=0
dim(test21SCN)
for(itpa2 in 1:151){ #COUNTING TRUE POSITIVE part1/3 of SCN2A data
    if((SCNtlPrediction21[itpa2]==test21SCN$pathogenic[itpa2])
       &&(SCNtlPrediction21[itpa2]==1)){jtpa2=jtpa2+1}}
for(itna2 in 1:151){ #TRUE NEGATIVE
    if((SCNtlPrediction21[itna2]==test21SCN$pathogenic[itna2])
       &&(SCNtlPrediction21[itna2]==0)){jtna2=jtna2+1}}
for(ifna2 in 1:151){ #FALSE NEGATIVE
    if((SCNtlPrediction21[ifna2]!=test21SCN$pathogenic[ifna2])
       &&(SCNtlPrediction21[ifna2]==0)){jfna2=jfna2+1}}
for(ifpa2 in 1:151){ #FALSE POSITIVE
    if((SCNtlPrediction21[ifpa2]!=test21SCN
        $pathogenic[ifpa2])&&(SCNtlPrediction21[ifpa2]==1)){jfpa2=jfpa2+1}}

jtpa2
jtna2
```

```
jfna2
jfpa2
jtpa2/(jtpa2+jfna2) #sensitivity TP/(TP+FN)
jtna2/(jtna2+jfpa2) #specificity TN/(TN+FP)


itpa22=0  #testing 2/3 summary
jtpa22=0
itna22=0
jtna22=0
ifna22=0
jfna22=0
ifpa22=0
jfpa22=0
dim(test22SCN)
for(itpa22 in 1:150){ #COUNTING TRUE POSITIVE part 2/3 of SCN2A data
    if((SCNtlPrediction22[itpa22]==test22SCN$pathogenic[itpa22])
       &&(SCNtlPrediction22[itpa22]==1)){jtpa22=jtpa22+1}}
for(itna22 in 1:150){ #TRUE NEGATIVE
    if((SCNtlPrediction22[itna22]==test22SCN$pathogenic[itna22])
       &&(SCNtlPrediction22[itna22]==0)){jtna22=jtna22+1}}
for(ifna22 in 1:150){ #FALSE NEGATIVE
    if((SCNtlPrediction22[ifna22]!=test22SCN$pathogenic[ifna22])
       &&(SCNtlPrediction22[ifna22]==0)){jfna22=jfna22+1}}
for(ifpa22 in 1:150){ #FALSE POSITIVE
    if((SCNtlPrediction22[ifpa22]!=test22SCN
        $pathogenic[ifpa22])&&(SCNtlPrediction22[ifpa22]==1)){jfpa22=jfpa22+1}}

jtpa22
jtna22
jfna22
jfpa22

itpa23=0  #testing 3/3 summary
jtpa23=0
itna23=0
jtna23=0
ifna23=0
jfna23=0
ifpa23=0
jfpa23=0
dim(test23SCN)
for(itpa23 in 1:165){ #COUNTING TRUE POSITIVE part 3/3 of SCN2A data
    if((SCNtlPrediction23[itpa23]==test23SCN$pathogenic[itpa23])
```

```
       &&(SCNtlPrediction23[itpa23]==1)){jtpa23=jtpa23+1}}
for(itna23 in 1:165){ #TRUE NEGATIVE
    if((SCNtlPrediction23[itna23]==test23SCN$pathogenic[itna23])
       &&(SCNtlPrediction23[itna23]==0)){jtna23=jtna23+1}}
for(ifna23 in 1:165){ #FALSE NEGATIVE
    if((SCNtlPrediction23[ifna23]!=test23SCN$pathogenic[ifna23])
       &&(SCNtlPrediction23[ifna23]==0)){jfna23=jfna23+1}}
for(ifpa23 in 1:165){ #FALSE POSITIVE
    if((SCNtlPrediction23[ifpa23]!=test23SCN
        $pathogenic[ifpa23])&&(SCNtlPrediction23[ifpa23]==1)){jfpa23=jfpa23+1}}

jtpa23 #summary total
jtna23
jfna23
jfpa23
SCN2aTP<-jtpa2+jtpa22+jtpa23   #Adding part 1/3 2/3 3/3 together True Positive SCN2A
SCN2aTN<-jtna2+jtna22+jtna23   #Adding part 1/3 2/3 3/3 together True Negative SCN2A
SCN2aFN<-jfna2+jfna22+jfna23
SCN2aFP<-jfpa2+jfpa22+jfpa23
SCN2aTP
SCN2aTN
SCN2aFN
SCN2aFP


############################
#all trained SCN8A summary
############################


itpa8=0  #part 1/3 summary
jtpa8=0
itna8=0
jtna8=0
ifna8=0
jfna8=0
ifpa8=0
jfpa8=0
for(itpa8 in 1:151){ #TRUE POSITIVE
    if((SCNtlPrediction81[itpa8]==test88SCN$pathogenic[itpa8])
       &&(SCNtlPrediction81[itpa8]==1)){jtp8=jtp8+1}}
for(itna8 in 1:151){ #TRUE NEGATIVE
    if((SCNtlPrediction81[itna8]==test88SCN$pathogenic[itna8])
       &&(SCNtlPrediction81[itna8]==0)){jtna8=jtna8+1}}
```

```
for(ifna8 in 1:151){ #FALSE NEGATIVE
    if((SCNtlPrediction81[ifna8]!=test88SCN$pathogenic[ifna8])
       &&(SCNtlPrediction81[ifna8]==0)){jfna8=jfna8+1}}
for(ifpa8 in 1:151){ #FALSE POSITIVE
    if((SCNtlPrediction81[ifpa8]!=test88SCN
        $pathogenic[ifpa8])&&(SCNtlPrediction81[ifpa8]==1)){jfpa8=jfpa8+1}}

jtpa8
jtna8
jfna8
jfpa8
jtpa8/(jtpa8+jfna8) #sensitivity TP/(TP+FN)
jtna8/(jtna8+jfpa8) #specificity TN/(TN+FP)


itpa82=0  #testing 2/3 summary
jtpa82=0
itna82=0
jtna82=0
ifna82=0
jfna82=0
ifpa82=0
jfpa82=0
dim(test82SCN)
for(itpa82 in 1:150){ #TRUE POSITIVE
    if((SCNtlPrediction82[itpa82]==test82SCN$pathogenic[itpa82])
       &&(SCNtlPrediction82[itpa82]==1)){jtpa82=jtpa82+1}}
for(itna82 in 1:150){ #TRUE NEGATIVE
    if((SCNtlPrediction82[itna82]==test82SCN$pathogenic[itna82])
       &&(SCNtlPrediction82[itna82]==0)){jtna82=jtna82+1}}
for(ifna82 in 1:150){ #FALSE NEGATIVE
    if((SCNtlPrediction82[ifna82]!=test88SCN$pathogenic[ifna82])
       &&(SCNtlPrediction82[ifna82]==0)){jfna82=jfna82+1}}
for(ifpa82 in 1:150){ #FALSE POSITIVE
    if((SCNtlPrediction82[ifpa82]!=test82SCN
        $pathogenic[ifpa82])&&(SCNtlPrediction82[ifap82]==1)){jfpa82=jfpa82+1}}

jtpa82
jtna82
jfna82
jfpa82
itpa83=0  #testing 3/3 summary
jtpa83=0
itna83=0
```

```
jtna83=0
ifna83=0
jfna83=0
ifpa83=0
jfpa83=0
dim(test83SCN)
for(itpa83 in 1:151){ #TRUE POSITIVE
    if((SCNtlPrediction83[itp83]==test83SCN$pathogenic[itp83])
       &&(SCNtlPrediction83[itp83]==1)){jtp83=jtp83+1}}
for(itna83 in 1:151){ #TRUE NEGATIVE
    if((SCNtlPrediction83[itn83]==test83SCN$pathogenic[itn83])
       &&(SCNtlPrediction83[itn83]==0)){jtn83=jtn83+1}}
for(ifna83 in 1:151){ #FALSE NEGATIVE
    if((SCNtlPrediction83[ifn83]!=test83SCN$pathogenic[ifn83])
       &&(SCNtlPrediction83[ifn83]==0)){jfn83=jfn83+1}}
for(ifpa83 in 1:151){ #FALSE POSITIVE
    if((SCNtlPrediction83[ifp83]!=test83SCN
        $pathogenic[ifp83])&&(SCNtlPrediction83[ifp83]==1)){jfp83=jfp83+1}}

jtpa83 #summary total
jtna83
jfna83
jfpa83
SCN8ATP<-jtpa8+jtpa82+jtpa83   #Adding part 1/3 2/3 3/3 together True Positive
SCN8ATN<-jtna8+jtna82+jtna83   #Adding part 1/3 2/3 3/3 together True Negative
SCN8AFN<-jfna8+jfna82+jfna83   #False Negative
SCN8AFP<-jfpa8+jfpa82+jfpa83   #False Positive
SCN8ATP                           #Numbers Output
SCN8ATN
SCN8AFN
SCN8AFP


######
itpa83=0  #all-data training- testing 3/3 summary
jtpa83=0
itna83=0
jtna83=0
ifna83=0
jfna83=0
ifpa83=0
jfpa83=0
dim(test83SCN)
for(itpa83 in 1:151){ #TRUE POSITIVE
```

```
if((SCNtlPrediction83[itpa83]==test83SCN$pathogenic[itpa83])
&&(SCNtlPrediction83[itpa83]==1)){jtpa83=jtpa83+1}}
  for(itna83 in 1:150){ #TRUE NEGATIVE
       if(SCNtlPrediction83[itna23]==test23SCN
 $pathogenic[itna83]&&SCNtlPrediction83[itna83]==0){jtna83=jtna83+1}}


for(ifna83 in 1:151){ #FALSE NEGATIVE
      if((SCNtlPrediction83[ifna83]!=test83SCN$pathogenic[ifna83])
      &&(SCNtlPrediction83[ifna83]==0)){jfna83=jfna83+1}}
 for(ifpa83 in 1:151){ #FALSE POSITIVE
      if((SCNtlPrediction83[ifpa83]!=test83SCN
$pathogenic[ifpa83])&&(SCNtlPrediction83[ifpa83]==1)){jfpa83=jfpa83+1}}

jtpa83 #summary total
jtna83
jfna83
jfpa83



itpa8=0  #testing 1/3 summary
jtpa8=0
itna8=0
jtna8=0
ifna8=0
jfna8=0
ifpa8=0
jfpa8=0
for(itpa8 in 1:151){ #TRUE POSITIVE
    if((SCNtlPrediction81[itpa8]==test88SCN$pathogenic[itpa8])
       &&(SCNtlPrediction81[itpa8]==1)){jtp8=jtp8+1}}
for(itna8 in 1:151){ #TRUE NEGATIVE
    if((SCNtlPrediction81[itna8]==test88SCN$pathogenic[itna8])
       &&(SCNtlPrediction81[itna8]==0)){jtna8=jtna8+1}}
for(ifna8 in 1:151){ #FALSE NEGATIVE
    if((SCNtlPrediction81[ifna8]!=test88SCN$pathogenic[ifna8])
       &&(SCNtlPrediction81[ifna8]==0)){jfna8=jfna8+1}}
for(ifpa8 in 1:151){ #FALSE POSITIVE
    if((SCNtlPrediction81[ifpa8]!=test88SCN
        $pathogenic[ifpa8])&&(SCNtlPrediction81[ifpa8]==1)){jfpa8=jfpa8+1}}

jtpa8
jtna8
```

```
jfna8
jfpa8
jtpa8/(jtpa8+jfna8) #sensitivity TP/(TP+FN)
jtna8/(jtna8+jfpa8) #specificity TN/(TN+FP)


itpa82=0  #testing 2/3 summary
jtpa82=0
itna82=0
jtna82=0
ifna82=0
jfna82=0
ifpa82=0
jfpa82=0
dim(test82SCN)
for(itpa82 in 1:150){ #TRUE POSITIVE
    if((SCNtlPrediction82[itpa82]==test82SCN$pathogenic[itpa82])
       &&(SCNtlPrediction82[itpa82]==1)){jtpa82=jtpa82+1}}
for(itna82 in 1:150){ #TRUE NEGATIVE
    if((SCNtlPrediction82[itna82]==test82SCN$pathogenic[itna82])
       &&(SCNtlPrediction82[itna82]==0)){jtna82=jtna82+1}}
for(ifna82 in 1:150){ #FALSE NEGATIVE
    if((SCNtlPrediction82[ifna82]!=test88SCN$pathogenic[ifna82])
       &&(SCNtlPrediction82[ifna82]==0)){jfna82=jfna82+1}}
for(ifpa82 in 1:150){ #FALSE POSITIVE
    if((SCNtlPrediction82[ifpa82]!=test82SCN
        $pathogenic[ifpa82])&&(SCNtlPrediction82[ifpa82]==1)){jfpa82=jfpa82+1}}

jtpa82
jtna82
jfna82
jfpa82
itpa83=0  #testing 3/3 summary
jtpa83=0
itna83=0
jtna83=0
ifna83=0
jfna83=0
ifpa83=0
jfpa83=0
dim(test83SCN)
for(itpa83 in 1:151){ #TRUE POSITIVE
    if((SCNtlPrediction83[itpa83]==test83SCN$pathogenic[itpa83])
       &&(SCNtlPrediction83[itpa83]==1)){jtpa83=jtpa83+1}}
```

```
for(itna83 in 1:151){ #TRUE NEGATIVE
    if((SCNtlPrediction83[itna83]==test83SCN$pathogenic[itna83])
       &&(SCNtlPrediction83[itna83]==0)){jtna83=jtna83+1}}
for(ifna83 in 1:151){ #FALSE NEGATIVE
    if((SCNtlPrediction83[ifna83]!=test83SCN$pathogenic[ifna83])
       &&(SCNtlPrediction83[ifna83]==0)){jfna83=jfna83+1}}
for(ifpa83 in 1:151){ #FALSE POSITIVE
    if((SCNtlPrediction83[ifpa83]!=test83SCN
        $pathogenic[ifpa83])&&(SCNtlPrediction83[ifpa83]==1)){jfpa83=jfpa83+1}}

jtpa83 #summary total
jtna83
jfna83
jfpa83
SCN8ATP<-jtpa8+jtpa82+jtpa83
SCN8ATN<-jtna8+jtna82+jtna83
SCN8AFN<-jfna8+jfna82+jfna83
SCN8AFP<-jfpa8+jfpa82+jfpa83
SCN8ATP
SCN8ATN
SCN8AFN
SCN8AFP



###########
# Table 8 #
###########


d101<-as.numeric(SCN1_predictions1)-as.numeric(SCNtlPrediction11)
summary(test11SCN$pathogenic)
test11SCN$pathogenic
p11<-d101[1:50]
b11<-d101[51:156]

d102<-as.numeric(SCN1_predictions2[order(test12SCN$pathogenic)]
)-as.numeric(SCNtlPrediction12[order(test12SCN$pathogenic)])
 d102
 summary(test12SCN$pathogenic[order(test12SCN$pathogenic)])
   test12SCN$pathogenic[order(test12SCN$pathogenic)]
 b12<-d102[1:113]
 p12<-d102[114:150]
```

```
 d103<-as.numeric(SCN1_predictions3)-as.numeric(SCNtlPrediction13)
 summary(test13SCN$pathogenic)
 test13SCN$pathogenic
 p13<-d101[1:35]
 b13<-d101[36:165]

 benign1A<-c(b11,b12,b13)  #SCN1A benign group
 patho1A<-c(p11,p12,p13)    #SCN1A pathogenic group


d201<-as.numeric(SCN2_predictions1)-as.numeric(SCNtlPrediction21)
summary(test21SCN$pathogenic)
test21SCN$pathogenic
p21<-d201[1:53]
b21<-d201[54:152]

d202<-as.numeric(SCN2_predictions2[order(test22SCN$pathogenic)]
)-as.numeric(SCNtlPrediction22[order(test22SCN$pathogenic)])
 d202
 summary(test22SCN$pathogenic[order(test22SCN$pathogenic)])
   test22SCN$pathogenic[order(test22SCN$pathogenic)]
 b22<-d202[1:99]
 p22<-d202[100:150]


 d203<-as.numeric(SCN2_predictions3)-as.numeric(SCNtlPrediction23)
 summary(test23SCN$pathogenic)
 test23SCN$pathogenic
 p23<-d203[1:53]
 b23<-d203[54:165]

 benign2A<-c(b21,b22,b23)
 patho2A<-c(p21,p22,p23)


d801<-as.numeric(SCN8_predictions1)-as.numeric(SCNtlPrediction81)
summary(test88SCN$pathogenic)
test88SCN$pathogenic
p81<-d801[1:68]
b81<-d801[69:151]

d802<-as.numeric(SCN8_predictions2[order(test82SCN$pathogenic)]
)-as.numeric(SCNtlPrediction82[order(test82SCN$pathogenic)])
```

```
d802
summary(test82SCN$pathogenic[order(test82SCN$pathogenic)])
test82SCN$pathogenic[order(test82SCN$pathogenic)]
b82<-d802[1:82]
p82<-d802[83:150]


d803<-as.numeric(SCN8_predictions3)-as.numeric(SCNtlPrediction83)
summary(test83SCN$pathogenic)
test83SCN$pathogenic
p83<-d803[1:68]
b83<-d803[69:151]

benign8A<-c(b81,b82,b83)                    #SCN8A benign group
patho8A<-c(p81,p82,p83)                     #SCN8A pathogenic group
```

6 References

[1] I.A. Adzhubei, S. Schmit, S. Sunyaev, *A Method and Server for Predicting Damaging Missense Mutations* Nature Methods. 7(4): 248-9; Apr 2010.

[2] I.A. Azhubei, *Predicting Functional Effect of Human Missense Mutations Using PolyPhen-2* Curr Protoc Hum Genet, Chapter 7: Unit 7.20, 2013.

[3] R. Grantham, *Amino Acid Difference Formula to Help Explain Protein Evolution* Science 06, Sep 1974.

[4] M. Kercher, D. Witten, P. Jain, B.J. O'Roak, G.M. Copper, J. Shendure, *A General Framework for Estimating the Relative Pathogenicity of Human Genetic Variants* Nature Genetic, Feb 2, 2014.

[5] P. Kumar, S. Henikoff, P.C. Ng. *Predicting the Effects of Coding Non-Synonymous Variants on Protein Function Using the SIFT Algorithm* Nature Protoc, 4(7), 2009.

[6] S. Sunyaev, F. Eisenhaber, *PSIC, Profile Extraction from Sequence Alignments with Position-Specific Counts of Independent Observations* Protein Engineering vol.12 no.5 pp. 387-394, 1999.

[7] P.C. Ng, J.G. Henikoff, S. Henikoff, *PHAT: A Transmembrane-Specific Substitution Matrix. Predicted Hydrophobic and Transmembrane* Bioinformatics, Sep 2000.

[8] R.D. Finn, P. Coggill, R.Y. Eberhardt, S.R. Eddy, J. Mistry, A.L. Mitchell, S.C. Potter, M. Punta, M. Qureshi, A. Sangrador-Vegas, G.A. SalaZar, J. Tate, A. Bateman *The Pfam Protein Families Database: Towards a More Sustainable Future* Nucleic Acids Research, Database Issue 44: D279-D285, 2016.

[9] V. Ramensky, P. Bork, S. Sunyaev, *Human Non-Synonymous SNPs: Server and Survey.* Nucleic Acids Res, 30(17): 3894-3900, 2002.

[10] M. Lek, K. Karczewski *Analysis of Protein-Coding Genetic Variation in 6070 Humans* Nature 536, 285-291, August 18, 2016.

[11] E. Delong, D. Delong, C. Pearson, *Comparing the Areas Under Two or More Correlated Receiver Operating Characteristic Curves: A Nonparametric Approach* Biometrics, Sep; 44(3): 837-45, 1988.

[12] H.C. Lai, L.Y. Jan, *The distribution and targeting of neuronal voltage-gated ion channels.* Nature Reviews Neuroscience 7, 548-562,2006.