# COMPARISON OF CLASSIFICATION RATES AMONG LOGISTIC REGRESSION, NEURAL NETWORK AND SUPPORT VECTOR MACHINES IN THE PRESENCE OF MISSING DATA

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Sudhi Upadhyaya

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Statistics

May 2014

Fargo, North Dakota

# North Dakota State University
## Graduate School

**Title**

Comparison of Classification Rates Among Logistic Regression, Neural Network and Support Vector Machines in the Presence of Missing Data

**By**

Sudhi Upadhyaya

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

**MASTER OF SCIENCE**

SUPERVISORY COMMITTEE:

Dr.Rhonda Magel
Chair

Mr. Curt Doetkott

Dr. Tatjana Miljkovic

Dr. Kambiz Farahmand

Approved:

May 12th 2014
Date

Dr.Rhonda Magel
Department Chair

## ABSTRACT

Statistical models such as Logistic Regression (LR), Neural Network (NN) and Support Vector Machines (SVM) often use datasets with missing values while making inferences regarding the population. When inferences are made based on the data set used, the presence of missing data can severely skew the results and distort the efficiency of the model. Our objective was to identify a robust model among LR, NN, SVM in the presence of missing data. The study was conducted by simulating observations based on Monte Carlo methods and missing data was introduced randomly at 10% level. Single mode imputation was used to impute missing values. Simple random samples of 120, 240 and 500 observations were chosen and these three models were fit for two scenarios. Results showed that the performance of SVM was far superior compared to LR or NN models. However, the classification accuracy of SVM gradually decreased as sample size increased.



**Key words** : Missing data, Single mode imputation, Logistic Regression, Neural Network, Support Vector Machine, classification probability

**TABLE OF CONTENTS**

ABSTRACT……iii

LIST OF TABLES……..vi

LIST OF FIGURES……vii

CHAPTER 1.INTRODUCTION…….1

CHAPTER 2. BACKGROUND…….4

CHAPTER 3. DESCRIPTION OF THE METHODS COMPARED…….7

3.1. Neural Network ……7

3.2. Logistic Regression……12

3.3. Support Vector Machine…….13

CHAPTER 4. DESIGN OF SIMULATION STUDY……..16

4.1. Step 1: Choosing a Dataset……..16

4.2. Step 2: Choosing Variables From This Dataset…..17

4.3. Step 3: Identifying Frequency Distribution…..18

4.4. Step 4: Monte Carlo Simulation ……..20

4.5. Step 5: Simulating Missing Values…..21

4.6. Step 6: Mode Imputation……21

CHAPTER 5. RESULTS……..23

CHAPTER 6. CONCLUSIONS……30

REFERENCES……..32

APPENDIX A. SAS CODE FOR SAMPLE SIZE 120 , SCENARIO 1……..38

APPENDIX B. SAS CODE FOR SAMPLE SIZE 240 , SCENARIO 1……..48

APPENDIX C. SAS CODE FOR SAMPLE SIZE 500 , SCENARIO 1…….58

APPENDIX D. SAS CODE FOR SAMPLE SIZE 120 , SCENARIO 2……..…..68

APPENDIX E. SAS CODE FOR SAMPLE SIZE 240 , SCENARIO 2…….…..……...78

APPENDIX F. SAS CODE FOR SAMPLE SIZE 500 , SCENARIO 2…….…..…...88

## LIST OF TABLES

Table Page

1: Summary of Features Used to Build NN Model……..…….....12

2: Summary of Features Used To Build The SVM Model……...……...15

3: Description of The Variables and Their Levels Used In The Study……..……..17

4: Simulating 10% Missing Data……..…..…….21

5: Decision Matrix of Diagnosis……..….……..23

## LIST OF FIGURES

Figures Page

1: Overview Of A Neural Network Model ……..…….…….... 7

2: List of Activation and Combination Functions……...…….9

3: Logistic Regression ……...……..…….....13

4: Linear Support Vector Machine, Adopted from Smola, & Schölkopf (2004)….....15

5: Outline Of The Design Of Simulation ……...…….…….....16

6: Summary Of Proportion Of Various Levels For Each Variable …….……..…….....18

7: Mode Values For Each Independent Variable…….……..….....22

8: Classification Table Logistic Regression……...…….…….....24

9: Classification Table Output for Neural Network…….…..…….….……..….24

10: Classification Table Output for SVM…….……...…….....…….25

11: Percent Correct Classification, LR, NN and SVM, Sample Size 120, Scenario 1……....26

12: Percent Correct Classification, LR, NN and SVM, Sample Size 240, Scenario 1.....26

13: Percent Correct Classification, LR, NN and SVM, Sample Size 500, Scenario 1.....27

14: Percent Correct Classification, LR, NN and SVM, Sample Size 120, Scenario 2……....28

15: Percent Correct Classification, LR, NN and SVM, Sample Size 500, Scenario 2……....28

16: Percent Correct Classification, LR, NN and SVM, Sample Size 500, Scenario 2……....29

# CHAPTER 1. INTRODUCTION

Analyses of incomplete data has been the object of many studies. The term missing data refers to situations where one or more data values for variables are not observed. Missing data are ubiquitous and the causes of missingness are numerous. This could be due to the general study design, simply due to chance, or both. For instance, some information is not collected from all subjects, some participants may decline to provide all information, and some information may be purposely deleted to protect confidentiality (Horton & Kleinman, 2007). In some situations, missing data is due to non-compliance, dropouts, or non-response.

Over the past century, statisticians and other scientists not only have invented numerous methods for handling missing/incomplete data, but also have invented many ways to impute missing data. These include data augmentation, hidden states, latent variables, potential outcome, and auxiliary variables. Purposely constructing unobserved/unobservable variables offers an extraordinarily flexible and powerful framework for both scientific modeling and computation and is one of the central statistical contributions to natural, engineering, and social sciences. In parallel, much research has been devoted to better understanding and modeling of real-life missing-data mechanisms; that is, the unintended data selection process that prevents us from observing our intended data.

There are four main sources of missing data. The first is item total nonresponse that results from subjects refusal to participate in the survey. The second source of missing data is due to non-coverage. This occurs when some elements of inference are not included in the survey. For example: A survey conducted across the nation elicits responses whether or not a subject climbed Mt.Everest. From the total responses received if the survey contains answers from very few women, these responses are discarded. A third source of missing data is item

nonresponse. In this case the participant responses are partially complete and do not answer all the elements in the survey. A fourth source of missing data is partial nonresponse. Partial nonresponse falls between total and item nonresponse. Total nonresponse relates to failure to obtain any response from a sampled element and item non-response implies the failure to obtain responses for only a small number of survey items. Partial nonresponse involves a substantial number of item nonresponses. An example of partial nonresponse is when a respondent cuts off an interview in the middle (Brick & Kalton, 1996 ; Rubin,1976).

Apart from the main sources of missingness, the pattern of missingness itself can be classified into three categories; Missing Completely at Random (MCAR), Missing at Random (MAR), Not Missing at Random (NMAR). Let $x_{ij}$ represent complete data and $m_{ij}$ represent missing data vectors respectively. If $f(m_{ij}|\theta) = f(m_{ij}|x_{obs}, x_{miss}, \theta) \ \forall \ x_{ij\,(}x_{obs}, x_{miss)}, \theta$ the data is missing completely at random (MCAR) because the missing data does not depend on either observed($x_{obs}$) or missing values($x_{miss}$) of $x_{ij}$ . If $f(m_{ij}|\theta) = f(m_{ij}|\, x_{obs}, \theta) \ \forall \ x_{miss}, \theta$ the data is missing at random (MAR) because the missing data depends only on the observed component of $x_{ij}$, $x_{obs}$ and not on the missing component of $x_{ij}$, $x_{miss}$ . If $f(m_{ij}|x_{ij}, \theta) = f(m_{ij}|\, x_{miss}, \theta) \ \forall \ x_{ij}, \theta$ the data is not missing at random (NMAR) because here the missing value depends on the missing values of $y_{ij}$ (Little & Rubin, 2002).

Accordingly there are several statistical methods that can be applied to obtain either an unbiased estimator or a model with robust prediction accuracy in the presence of missing data. This depends on the assumption regarding missingness (MCAR, MAR or NMAR), but in practice, analyzing the reason for incomplete data is very difficult since in most cases the incomplete data themselves contain little or no information about the missing data mechanism ( Lu & Copas, 2004). According to Meng (2000), the missing-data mechanism is the very core

of statistics, and among the nastiest and the most deceptive area of statistics; "*why on earth should anyone be concerned with data that one does not even have*?" , (p.1329).

## CHAPTER 2. BACKGROUND

A problem that is equally severe, if not greater than missing data, is the performance of statistical models that use these datasets with missing values. The quality of the results obtained using these models mainly depends on the quality of the data set used, and the presence of missing data can severely skew the results and reduce the efficiency of the models (Gao & Hui , 1997). Hence, it is very important to choose a prediction model that is very robust and efficient in the presence of missing data.

Logistic Regression (LR), Artificial Neural Network (ANN) and Support Vector Machine (SVM) are becoming increasingly popular tools for classification in many areas of science and technology. Several studies have compared the performance of logistic regression models to address various issues (Steyerberg, Eijkemans, Harrell, & Habbema, 2000 ; Steyerberg, Eijkemans, Van Houwelingen, Lee, & Habbema, 2000 ; Lachin, 2008; Bellazzi & Zupan, 2008). These studies in general can be classified as : 1) Studies that compare the variables identified by logistic regression with variables identified by other models ; 2) Studies that compare the sensitivity and specificity by logistic regression with other models; 3) Studies that compare the logistic regression Receiver Operating Characteristics (ROC) or the Area Under the Curve (AUC) with ROC or AUC values from other models (Austin, 2007).

During the past few years, the application of artificial neural networks (ANNs) for prognostic and diagnostic classification in clinical medicine has attracted growing interest. Preliminary studies have shown that neural networks are more accurate than physicians in identifying acute myocardial infarction in patients presenting to the emergency department with anterior chest pain (Faraggi & Simon, 1995; Faraggi, LeBlanc & Crowley, 2001 ; Biganzoli, Boracchi, Mariani & Marubini, 1998).

Support Vector Machine (SVM) is a related supervised learning algorithm used for classification of both linear and nonlinear data based on a statistical learning theory. The basic idea is that a SVM constructs a hyperplane or a set of hyperplanes in a high-dimensional space, which can be further used for classification, regression or any other related tasks. The hyperplane built by SVM optimally splits the training dataset between a set of objects belonging to different groups (Czibula, Czibula, & Găceanu, 2014). SVM has an extensive scope and has been applied in a number of areas such as healthcare (Dukart, Mueller, Barthel, Villringer, Sabri, & Schroeter, 2013), accident prevention (Suárez Sánchez, Riesgo Fernández, Sánchez Lasheras, de Cos Juez, & García Nieto, 2011), bankruptcy prediction (Ravi Kumar & Ravi, 2007) among others.

There have been several attempts to compare the classification efficiency of these discriminant models, including:

- Nearest-neighbor, linear discriminant analysis, and classification tree while imputing missing values based on k nearest-neighbor algorithm method (Dudoit, Fridlyand, & Speed, 2006).
- Missing data techniques for generalized linear models, comparing four common approaches for inference in generalized linear models (GLMs): maximum likelihood (ML), multiple imputation (MI), fully Bayesian (FB), and weighted estimating equations (WEEs) , (Ibrahim, Chen, Lipsitz & Herring, 2007).
- Comparing the accuracy of neural network, support vector machine and logistic regression but missing data was not part of their objective, (Muniz, Liu, Lyons, Pahwa, Liu, Nobre, & Nadal, 2010; Westreich, Lessler, & Funk, 2010; Chen, & Du, 2009; Chen et.al, 2009).

However, to the best of our knowledge the classification efficiency of LR, NN and SVM models were not compared in the presence of missing data to identify which among these was more efficient when the mode imputation technique was applied. Therefore the goal of our research was to identify which among these ( LR, NN, SVM) are more efficient classifiers in the presence of missing data when the mode based single imputation technique is used.

## CHAPTER 3. DESCRIPTION OF THE METHODS COMPARED.

### 3.1. Neural Network

A typical NN model can accommodate multiple input features, $x_i$, to predict multiple outputs of interest, $y_i$. It differs from conventional modeling approaches in its ability to learn about the system that can be modeled without prior knowledge of the process relationships. The prediction by a well-trained NN is normally much faster than the conventional simulation programs or mathematical models because NN does not require lengthy iterative calculations to solve differential equations or other complex estimation procedures (Oğuz, Sarıtas, & Baydan , 2010).

Generally, NN is composed of three types of layers: an input layer, a hidden layer and an output layer as shown in Figure 1. Each layer has a certain number of components called neurons or nodes which are linked to each other. The neurons in the input layer represent the independent variables and the neurons in the output layer represent the dependent variable. Each neuron is linked to others with communication links accompanied by an associated weight (Ercan, 2011). The number of neurons in each layer depends on the desired level of accuracy.

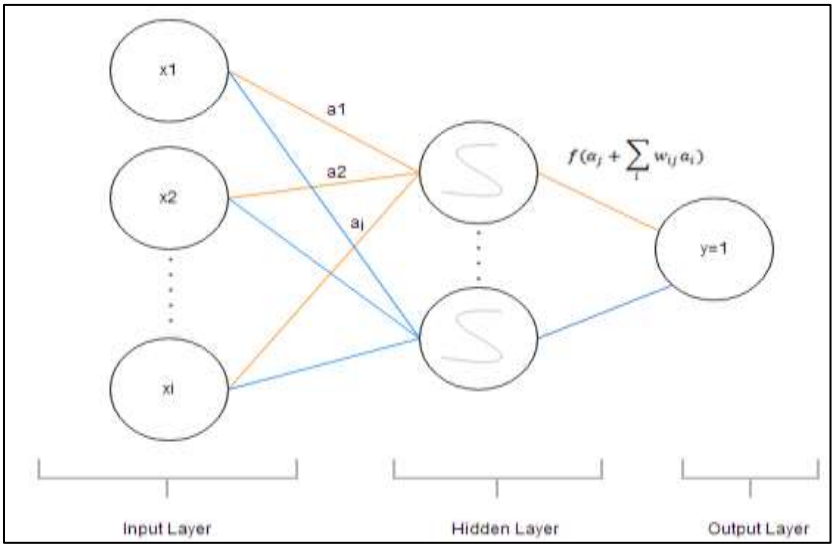


**Figure 1: Overview Of A Neural Network Model**

Each neuron performs two functions: First, it adds all the inputs from the lower layers based on their weights according to the equation $\sum_i w_{ij} a_i$ where, $w_{ij}$ represents the weight connecting neurons in two adjacent layers i and j and $a_j$ is the output from the jth neuron. Weights in neural networks are similar to coefficients in regression models and these weights are calculated based on an equation which depends on the type of training technique used to train the network (Ercan , 2011).

There are two types of training: supervised and unsupervised. We used supervised training in our analysis because it is performed when the observations consist of both independent and target variables $\{ x_1, x_2, x_3, y_i \}$ and the training algorithm seeks a function where $f(.)$ results in $y$ based on $f(x,y)$. There are various supervised training algorithms including Trust-Region Levenberg-Marquardt, Quasi-Newton, Conjugate Gradient, DBLDOG, BackProp, RPROP, QPROP, among others. Training is initialized by setting the threshold of the nodes and the strengths of the links to a small random value. The network is trained with a set of training cases. Each training case contains a set of input patterns and a corresponding desired output pattern, $\{x_1=1, x_2=0, \ldots . x_i=1, y=1 \} \{x_1=0, x_2=1, \ldots . x_i=0, y=1\}$. Training is continued to achieve good performance. Activity is then propagated from the hidden layer to the output layer. The actual activities (predicted values) of the output nodes are then compared with the target values and the error is computed (Ercan, 2011).

During the training process, the learning rule specifies how the weights of the connections in the network are adjusted. The weights are usually adjusted in a large number of small steps. We used Levenberg-Marquardt (LVM) training method in our study and hence weights were calculated based on LVM method. When the NN is trained to satisfactory levels,

the weighted links between various neurons are saved. These weights are then used to predict results from a new set of input data (Ercan , 2011).

A second function of a neuron in the neural network is that it processes the input it receives as a sum using a function based on the equation $a_j = f(\alpha_j + \sum_i w_{ij} a_i)$ where $\alpha_j$ is threshold value for the $i^{th}$ neuron and $a_i$ is the output of the ith neuron. The function *f*(.) is commonly referred to as transfer function or activation function. The purpose of the transfer or activation function is to transform the weighted sum of values from the previous neurons, either using a linear or non-linear relation and transfer the output to the next layer only if it crosses a certain threshold value Ercan (2011). Figure 2 provides a brief overview of some of the hidden layer combinations and transfer/activation functions and target layer combinations and transfer/activation functions.

| Hidden Combination Function | Hidden Activation Function | Target Combination Function | Target Activation Function |
|---|---|---|---|
| Linear: $bias + w_i x_i$ | Linear: $x$ | Linear: $bias + w_i x_i$ | Logistic: $\frac{1}{1+exp^{-x}}$ |
| | Logistic: $\frac{1}{1+exp^{-x}}$ | | Mlogistic: $\frac{e^x}{exponentials}$ |
| | arctan: $Tan(x)$ | | Softmax: $\frac{e^x}{exponentials}$ |
| | Hyperbolic angent: $Tanh\ x = \frac{e^x - e^{-x}}{e^x - e^{-x}}$ | | Gauss: $e^{-x^2}$ |

**Figure 2: List of Activation and Combination Functions**

There are several important issues in building a model. One issue is related to the model itself, such as architecture, selection of hidden nodes, input features, training function, transfer function and others. Other issues are those related to the analysis of the output, such as, the reliability and interpretation of results (Hung, Shankar, Hu, 2002).

### *3.1.1. Neural Network-Architecture*

The type of connections between the neurons define the architecture of a Neural Network. Neural Networks where the neurons are connected to other units in the same layer, to neurons in the preceding layer or even to themselves are termed recurrent networks. A feedforward network is one where units in one layer are connected only to units in the next layer and not to units in a preceding layer or units in the same layer. Multilayer Perceptron (MLP) is a type of feedforward network. The Multi-Layer Perceptron (MLP) is the most frequently used neural network technique in pattern recognition and classification problems (Rupérez, Martín-Guerrero, Monserrat & Alcañiz , 2011). MLP is both simple and based on solid mathematical framework. The inputs are processed through successive layers of neurons. In this study we used a three layer MLP feedforward neural network architecture.

### *3.1.2. Neural Network- Hidden Layer*

The hidden layer in a NN captures the nonlinear relations among various input features. There is no specific rule for the selection of the number of neurons in the hidden layer or the number of hidden layers that will produce the optimum results for a given problem ( Lolas, & Olatunbosun, 2008). Several authors have proposed various methods to calculate the number of hidden nodes. For example, Qiu, Tao, Tan, & Wu (2007) calculated the number of hidden nodes based on the formula, $h = \sqrt{(N+M)} + \alpha$ where, h is the number of hidden nodes in hidden layer, N is the number of input neural elements, M is the number of output neural elements, α is a constant between 1 and 10.

Hanafizade, Ravasan, & Khaki (2010), stated that the range of the number of hidden layers was between *2N +1* and *O(N+1)* where N is the number of input nodes and *O* is the number of output nodes. Emin Tagluk, Akin, & Sezgin (2010) state that no precise formula

exists. In our study, the number of neurons in the hidden layer was determined empirically. Results showed that out of the 2, 4, 6, 8, 10, 12, 16, 18, 20 hidden nodes tested , the NN model built using 10 hidden nodes resulted in the highest sensitivity (0.9966) and specificity (0.9918) and the maximum ROC.

*3.1.3. Neural Network-Training*

Training the neural network is defined as the process of identifying the optimal parameters such as weights, biases, and other parameters so that the model can accurately predict the outcome. Wilamowski, Iplikci, & Efe (2001) stated that neural network models trained using the LVM algorithm have better classification accuracy compared to those trained based on conventional back-propagation (BP) algorithm. Although the LVM algorithm has much higher computational requirements, it is more efficient and faster in achieving global minimization of the performance function. Therefore, we used the LVM algorithm to predict the outcome.

*3.1.4. Neural Network-Target Layer, Combination, Activation and Error function*

Neural network modeling depends on correctly specifying the transfer combination and error function which depends on the level of measurement of the target variable in the neural network model. It was recommended that a general linear combination function is applied to hidden-to-target layers for nominal type target values and a Softmax transformation function is applied in the case of hidden-to-target layer for nominal or binary target values (SAS Institute Inc., 2012). There are many different error functions that can be applied to hidden-to-target layers: BiWeight, Bernoulli, Binomial, Cauchy, Entropy, Gamma, Huber, Logistic, MBernoulli, MEntropy, Multinomial, Normal, Poisson and Wave. We tested our neural network for each of these error functions while the combination function was set to Linear and transfer function was set to Softmax. The results showed that Gamma , MBernoulli and Multinomial hidden-to-target

layers functions yield the highest classification efficiency with sensitivity 0.9966 and specificity 0.9918 and ROC=1. Table 1 below summarizes the features used to build the NN model.

**Table 1: Summary of Features Used to Build NN Model**

| Features | Options |
|---|---|
| Architecture: | *Multi Layer Perceptron* |
| Training | LVM |
| Direction Connection: | *No* |
| Number of Hidden Nodes: | 10 |
| Input Standardization: | *None* |
| Hidden Bias: | *Yes* |
| Target Layer Combination Function: | *Linear* |
| Target Layer Activation Function: | *Softmax* |
| Target Layer Error Function: | *Gamma, MBernoulli, Multinomial* |
| Target Bias: | *Yes* |
| Maximum iterations: | *1000* |

### 3.2. Logistic Regression

LR is a regression method for predicting a binary dependent variable, y, by modeling this as function of one or more independent features, $x_i$. The LR model yields a probability value which predicts the chances of the event occuring based on the best subset of predictors. This is expressed as $p_i = 1/(1 + e^{\alpha + \beta_i x_i})$ were $p_i$ represents the probability that the event occurs. This is calculated based on α which a constant which represents the intercept, $\beta_i$ is the vector of estimates for independent variables $x_i$ (Agresti , 1990) which is as shown in Figure 3.

Thus the process of classification ultimately reduces to the process of

a) Identifying the subset of covariates ($x_i$) that aid in classification.
b) Estimating the coefficients ($\alpha, \beta_i$) associated with theintercept and independent variables.
c) Calculating the probability p of the event. If $p > 0.5$, event occurs; if $p \leq 0.5$ event does not occur.

$p = 1/1+e^{-(\alpha+\beta x)}$ implies that probability increases or decreases in a sigmoid shape. This is a function of covariates $x_i$ where α is the intercept constant and $\beta_1, \beta_2, \beta_3, \ldots \beta_i$ are the estimated regression coefficients.

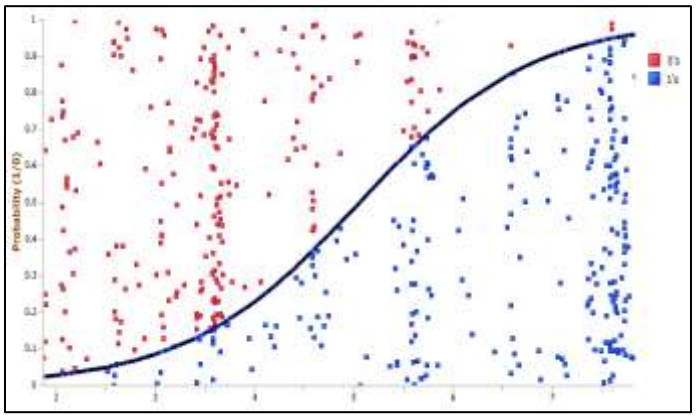


**Figure 3: Logistic Regression**

Depending on the value of p, a decision is made as to which subset of the population the subject belongs to on the basis of observed x. This classification into either $p_1$ or $p_0$ is the partitioning of Rm into two regions, $D_1$ and $D_0$ such that a subject is classified as belonging to $p_1$ if $x \in D_1$. The model is considered efficient and accurate if it minimizes the probability of misclassification of randomly selected subjects (O'Neill, 1980).

### 3.3. Support Vector Machine

Support Vector Machine is a machine-learning algorithm that aids in generalizing the charecteristics of input dataset. Consider a dataset with $\{x_i, y_i\}_{i=1}^{N}$ with input vector $x_i \epsilon R^d$ and $y_i$ transformed to {-1, 1 }. SVM maps the d-dimensional input vector $x_i$ from the input space to a feature space using linear/non-linear function $\Phi(.)$. The separating hyperplane in the feature space is based on equation $\Phi(x_i)^T \Phi(x_i) + b = 0$, b is some constant where $b \epsilon R$. If $\Phi(x_i)^T \Phi(x_i) + b \geq 1$ then that data is assigned to a group 1 and if $\Phi(x_i)^T \Phi(x_i) + b \leq -1$ then the data is assigned to group 2 (Louis, Agrawal & Khadikar, 2010; Smola, & Schölkopf , 2004)

A perfect classifier will result in a classification equation $\Phi(x_i)^T \Phi(x_i) + b \geq 1$ for y = 1 and $\Phi(x_i)^T \Phi(x_i) + b \leq -1$ for y = -1 . However, in realility, there are always some data points close to the separating hyperplane that could belong either to group 1 or to group 2 and this often causes misclassification or classification error. This introduces a new scenario where the objective is still classification while error is minimum and this is written as *Minimize* : $z(w,e) = \frac{1}{2}(w^T w) + k\sum_{i=1}^{N} e_i)$ such that $\Phi(x_i)T\ \Phi(x_i) + b + e_i \geq 1$ for $y_i = 1$ and $\Phi(x_i)^T \Phi(x_i) + b + e_i \leq 1$ for $y_i = -1$ where *k* is a regularization constant and $e_i$ is a slack variable to handle misclassification. The values of $e_i$ represents the distance of $x_i$ with respect to the decision boundary . This is as shown in Figure 4 below. (Louis, Agrawal & Khadikar, 2010; Smola, & Schölkopf , 2004)

There are three scenarios depending on the value of $e_i$

- If $e_i \geq 1$ when y = 1 and $\Phi(x_i)^t \Phi(x_i) + b \leq 1$ or $e_i \leq 1$ when y = -1 and $\Phi(x_i)^t \Phi(x_i) + b \geq 1$ then these scenarios indicate misclassification.
- If $0 \leq e_i \leq 1$ this indicates $x_i$ is correctly classified, but lies inside the margin.
- If $e_i = 0$ then $x_i$ is correctly classified and on the margin boundary.

Based on these notations, SVM is solved as a constrained primal optimization problem written in a dual space and solved as a quadratic programing problem with lagrangian multiplier $\alpha_i$ thus taking the form $\sum_{i=1}^{\#SV} \alpha_i \Phi(x_i)^t \Phi(x_i) + b$ (Louis, Agrawal & Khadikar, 2010; Smola, & Schölkopf , 2004)

The idea behind SVM is to find a (linear/nonlinear) mapping function $\Phi(x_i)$ that transforms data in input space to data in feature space in such a way as to render a problem either linearly or non-linearly separable. $\Phi(x_i)$ are the features of input variables $x_i$ after kernel transformation, $\alpha_i$ and b are the coefficients. Kernel transformation can be performed either

using linear, polynomial or radial bias functions (RBF). In our study we identified the fifth order polynomial kernel as providing the best results. After kernel transformation, the new feature space allows the data to be linearly or non-linearly separable, Louis, Agrawal & Khadikar, 2010; Smola, & Schölkopf , 2004.

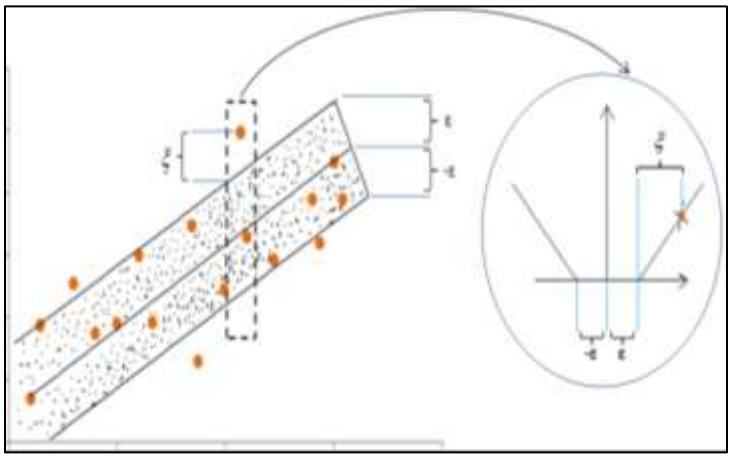

**Figure 4 : Linear Support Vector Machine, Adopted from Smola, & Schölkopf (2004)**

We tested our SVM algorithm with various kernel transformation functions and identified the fifth order polynomial as the most efficient kernel that resulted in highest classification results. Table 2 below shows the various options that we used in developing our SVM algorithm.

**Table 2: Summary of Features Used To Build The SVM Model**

| Features | Options |
|---|---|
| Kernel: | *Polynomial* |
| Polynomial Order: | *5* |
| Estimation Method | LSVM |
| Tuning Method: | *Grid Search* |
| Regularization Parameter: | *Tuning* |

## CHAPTER 4. DESIGN OF SIMULATION STUDY

The steps in our study design are outlined in the flowchart as shown in Figure 5.

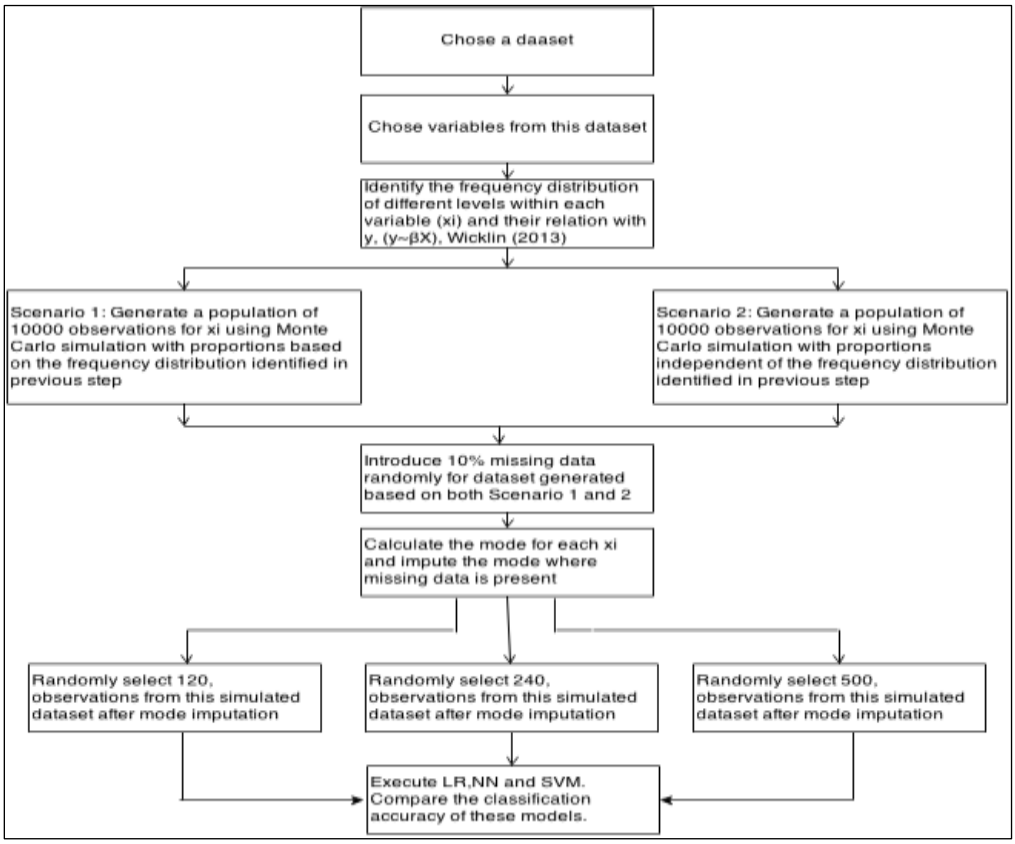


**Figure 5: Outline Of The Design Of Simulation**

### 4.1. Step 1: Choosing a Dataset

We selected a data set that stored information about the health and wellbeing of Native American elders. This dataset is based on a survey titled "Identifying Our Needs: A Survey of Elders" that was conducted by the National Resource Center on Native American Aging. The goal of the survey was to identify AI/AN elders with chronic illness such as diabetes, cancer and other health related issues. The dataset includes responses from 18,078 AI/AN elders (6573 males, 10919 females and 586 did not indicate their gender .

### 4.2. Step 2: Choosing Variables From This Dataset

**Table 3: Description of The Variables and Their Levels Used In The Study**

| Variable Name | Description | Levels |
|---|---|---|
| Falls | # of times an elder experienced falling during the past year | 1,2,3,4 or 5 |
| EyesPastYear | Whether the subject had his eyes examined during the past year | 1 = Yes<br>0 = No |
| LastDrank | Alcohol consumption | 1=during the past 30 days<br>2=during the past 1 year<br>3=sometime during the past 3 or more years |
| AgeGrp | Age group | 1='55 to 59'<br>2='60 to 69'<br>3='70 to 79'<br>4='80+' |
| CHF | Coronary Heart Failure | 1 = Yes<br>0 = No |
| BMIGrp | Bod Mass Index | 1= low<br>2= normal<br>3= obese<br>4= morbid obesity |
| Diab_Bernoulli_Sim | Diabetes | 1= Yes<br>0 = No |

Six independent variables: Body Mass Index (BMI), Falling, Alcohol consumption, Age Group, Coronary Heart Failure (CHF), Eyes check past year and one dependent variable Diabetes were considered for our study from this database. The description of these variables and the different levels are shown in Table 3.

### 4.3. Step 3: Identifying Frequency Distribution

Two different scenarios were used in determining the frequency distribution for independent variables in our study.

*4.3.1 Scenario 1- Frequency Distribution for independent variables*

The frequency distribution of different levels ($p_1, p_2, \ldots p_j$) within each variable ($x_i$, $y_i$) listed in Table 3 was determined and the summarized results are as shown in Figure 6. The largest percentage of observations in each category for the variables were, subjects between 60-69 in age (45.95%), reported being obese (40.85%), falling once during the past year (65.98%), very rarely consumed alcohol sometime during the past 3 years, (56.74), not experienced coronary heart failure (87.77%) and had their eyes examined (62.88%) .

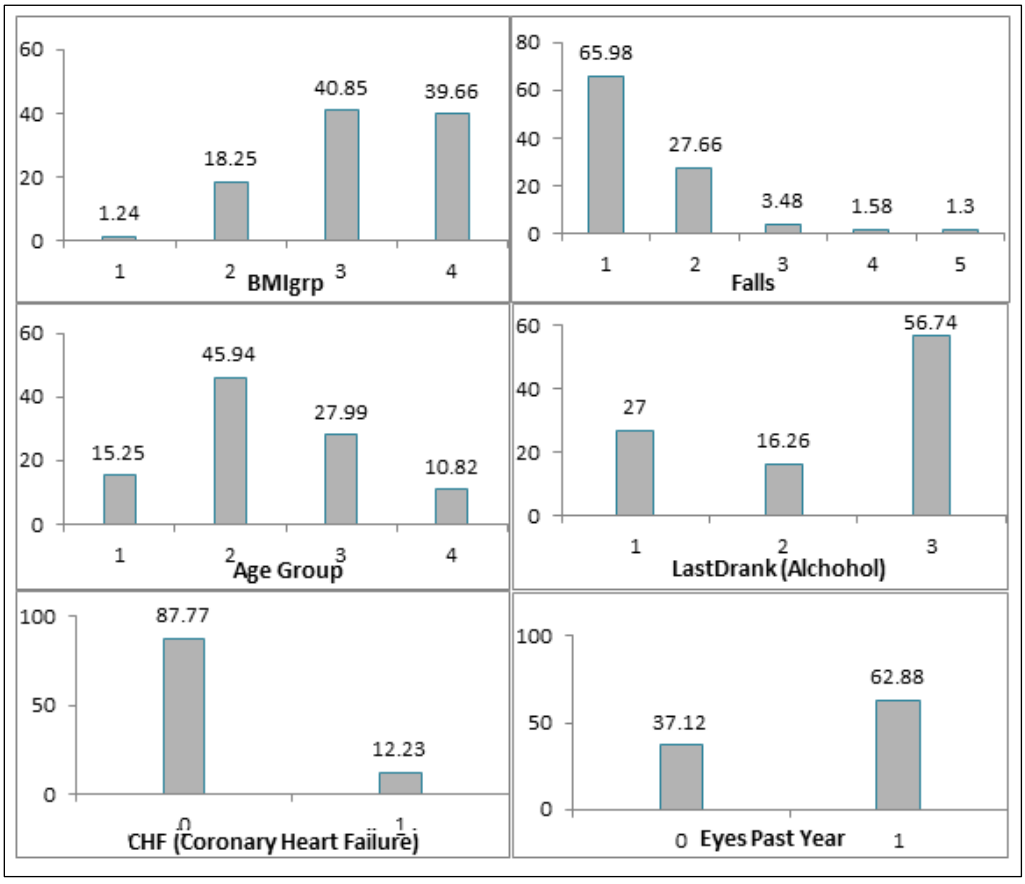


**Figure 6: Summary Of Proportion Of Various Levels For Each Variable**

*4.3.2 Scenario 2-Frequency Distribution for independent variables*

The dataset used for this study stored self-reported data related to subjects health and well being. In an attempt to assess the performance of these three classification methods we generated another dataset which is different from the dataset in Scenario1. Tobias et.al (2014) investigated the relation the between various levels of BMI and mortality among diabetes patients and reported that there is direct linear relationship between mortality and diabetic subjects with higher BMI. The frequency distribution of various levels of BMI in our original dataset displayed an *S* -shaped relation between BMI and diabetes. Based on this information we generated data for BMI ($x_{BMI}$) by changing the proportions from 0.0124, 0.1825, 0.4085, 0.3966 to 0.012, 0.18, 0.35, 0.458. This new distribution represented a linear relation between BMI and Diabetes compared to the *S*- shaped relationship in Scenario 1.

A prospective study of 3,075 men and women aged 70 –79 years recruited at the University of Pittsburgh, and the University of Tennessee, analyzed the overlap between risk of falling and diabetes and concluded that older adults with diabetes are more likely to fall (Schwartz et.al , 2008). Thus we generated new data for Fall ($x_{fall}$) by changing the proportions from 0.63, 0.26, 0.03, 0.07, 0.01 to 0.01, 0.03, 0.07, 0.26, 0.63.

Age was simulated based on four levels in Scenario 1 as shown in Table 3. However, a paper published by Koopman, Mainous, Diaz & Geesey (2005) stated that diabetes could be occurring at a higher frequency in youth and in young adults. Results from their study showed that age of diagnosis decreased from 52.0 to 46.0 years independent of race and ethnicity. So in Scenario 2 we generated data for the age ($x_{age}$) by modifying the proportions from 0.1525, 0.4594, 0.2799, 0.1082 to 0.2025, 0.4594, 0.2799, 0.0582.

Howard, Arnsten & Gourevitch (2008) conducted a study to understand the effect of alcohol consumption on Diabetes and concluded that moderate alcohol consumption is associated with 55%-79% reduction in risk for death due to coronary heart disease. Thus based on this information we generated data for alcohol consumption ($x_{alcohol}$) by modifying the proportions from 0.27, 0.16, 0.56 to 0.6, 0.1, 0.3.

That diabetes can cause heart failure was discovered by Kannel, Hjortland & Castelli (1974) and the results of their study were published as part of the very famous Framingham Heat Study. This study concluded that diabetics are twice as likely to die due to congestive heart failure as their non-diabetic cohorts. Based on this information we generated a new data for the CHF ($x_{chf}$) by changing the proportions from 0.87 to 0.95 in scenario 2.

Diabetic retinopathy is the most common factor contributing to blindness among diabetics and a study by Fong et.al (2004) stated that nearly 86% of blindness was attributed to diabetes and based on this information we generated new data for EyesPastYear ($x_{eyes}$) by changing the proprotions from 0.37 to 0.85 in scenario 2.

### 4.4. Step 4: Monte Carlo Simulation

A Monte Carlo simulation was conducted to generate 10000 observations that became the population in our study. These observations consisted of six independent variables (Falls, EyesPastYear, LastDrank, AgeGrp, CHF, BMIGrp) and dependent variable Diabetes based on the proportions defined in both Scenario 1 and Scenario 2. This resulted in two different of populations. For independent variables with more than two levels, we used the multinomial distribution with parameter $p_{i.}$ For eyes check and CHF, we used Bernoulli distribution with proportion parameter p. The proportion values, $p_i$, were based on the proportion of each level within each independent variable identified in Step2. The response variable, y, was generated

based on the equation $\text{logit}(p_i) = \beta_0 + \beta_1 x_{(bmi)} + \beta_2 x_{(age)} + \beta_3 x_{(Fall)} + \beta_4 x_{(alcohol)} + \beta_5 x_{(eyes)} + \beta_6 x_{(chf)}$ , the $p_i$ from this equation was used as the input in the Bernoulli distribution which was used to generate y based on the equation y = rand("Bernoulli", $p_i$) (Wicklin, 2013).

### 4.5. Step 5: Simulating Missing Values

In order to recreate the missing data scenario, missing values were simulated by randomly generating binary data (1,0) based on Monte Carlo simulation using the binomial distribution and the parameter p was set to 10%. This is represented as $z_i$. This process generated observations with approximately 90% 0s and 10% 1s. Each $z_i$ containing 1s and 0s was associated with an independent variable $x_i$. Cell values of $x_i$ were set to missing where $z_i = 1$, ($x_{i.} = . \mid z_i = 1$) . This is represented in Table 4.

**Table 4: Simulating 10% Missing Data**

| BMIGrp | z |
|---|---|
| 1 | 0 |
| 3 | 0 |
| 2 | 0 |
| 3 | 1 |
| 3 | 0 |
| 4 | 1 |

| BMIGrp | z |
|---|---|
| 1 | 0 |
| 3 | 0 |
| 2 | 0 |
| . ← | 1 |
| 3 | 0 |
| . ← | 1 |

### 4.6. Step 6: Mode Imputation

Let $x_{ij}$ represent the value of x for the $i^{th}$ respondent, i=1….n , $j^{th}$ covariate, j=1….6.. Single mode imputation substitutes the mode of variable $x_j$ for cells with missing data for that $j^{th}$ variable. For an equally weighted observation, the mode may be estimated by the mode of the observed data values. We decided to use single mode imputation for our study because:

a. Mode imputation uses the same value for that specific $x_i$ .This process underestimates the variance by reducing the discriminant power of a statistical model significantly.

b. This is a simple method. It is very easy to implement and avoids the issue with reduced sample size due to deleting incomplete records (Tian, Yu, Yu & Ma , 2013).

Figure 7 shows the values of mode for the six variables in our dataset. Results show that mode values were: subjects experiencing fall at least once, getting their eyes checked during the past year, rarely consume alcohol, between 60-69 years and not suffering from CHF for both diabetic and non-diabetic subjects. The mode for BMI was obese.

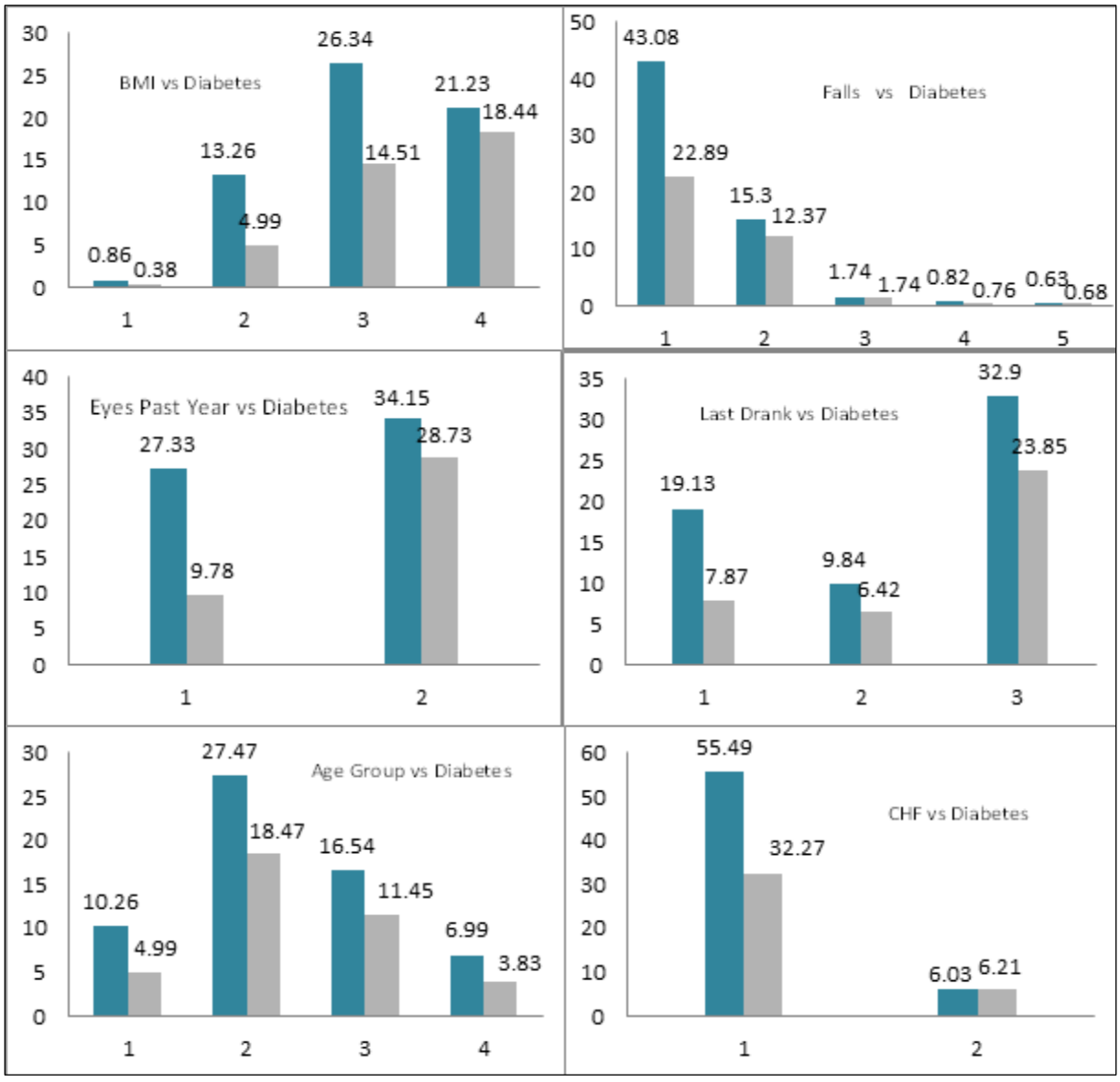


**Figure 7: Mode Values For Each Independent Variable**

## CHAPTER 5. RESULTS

The efficiency and accuracy of a diagnostic test can be defined in many different ways. For a diagnostic test that uses binary data, the results of the tests are classified as true positive, true negative, false positive or false negative as shown in Table 5 below.

**Table 5: Decision Matrix of Diagnosis**

| | **Patient Diabetic** | **Patient Not Diabetic** |
|---|---|---|
| **Model predicts diabetes** | True Positive | False Positive. |
| **Model does not predict diabetes** | False Negative | True Negative |

A perfect test will predict the outcome correctly all the time. An inadequate test will provide unreliable results. Thus the classification probabilities for true positive or true negative are considered more relevant because they quantify how well the test reflects the true status of the outcome. Also, there is a direct relation between the classification probabilities and the predictive values (Pepe, 2003).

For the LR model, we used the option CTABLE in proc Logistic to obtain a set of true positive, false positive, true negative and false negative values at different probability levels. The value in the *percentage of correct classification* column corresponding to the row with highest sensitivity and specificity was chosen and this value represented the classification efficiency of the LR model. For example, the highest values of specificity and sensitivity were 60.0 and 62.9 respectively. For these values of specificity and sensitivity the corresponding values of True Positive , True Negative, False Positive and False Negative are as shown in Figure 8.

| Probability Level | Number of Correct Events | Number of Correct Non events | Number of Incorrect Events | Number of Incorrect Non Events | Percentage of Correct Classification | Sensitivity % | Specificity % | Percentage of False Positive | Percentage of False Negative | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.9 | 0 | 69 | 1 | 50 | 57.5 | 0 | 98.6 | 100 | 42 | 98.571 |
| 0.7 | 8 | 60 | 10 | 42 | 56.7 | 16 | 85.7 | 55.6 | 41.2 | 101.71 |
| 0.8 | 5 | 68 | 2 | 45 | 60.8 | 10 | 97.1 | 28.6 | 39.8 | 107.14 |
| 0.1 | 48 | 12 | 58 | 2 | 50 | 96 | 17.1 | 54.7 | 14.3 | 113.14 |
| 0.4 | 30 | 39 | 31 | 20 | 57.5 | 60 | 55.7 | 50.8 | 33.9 | 115.71 |
| 0.6 | 18 | 56 | 14 | 32 | 61.7 | 36 | 80 | 43.8 | 36.4 | 116 |
| 0.2 | 43 | 23 | 47 | 7 | 55 | 86 | 32.9 | 52.2 | 23.3 | 118.86 |
| 0.3 | 38 | 31 | 39 | 12 | 57.5 | 76 | 44.3 | 50.6 | 27.9 | 120.29 |
| 0.5 | 27 | 48 | 22 | 23 | 62.5 | 54 | 68.6 | 44.9 | 32.4 | 122.57 |
| 0.45 | 30 | 44 | 26 | 20 | 61.7 | 60 | 62.9 | 46.4 | 31.3 | 122.86 |

**Figure 8: Classification Table Logistic Regression**

The highlighted row in Figure 9 shows that 30 events and 44 nonevents were accurately classified out of a total of 120 observations, resulting in a classification accuracy of 74/120 = 61.67%.

| _NAME_ | Train: Total Degrees of Freedom | Train: of Fr for E | Train: Degr of Fr | Tr: W | Tra: Aka Inf Cri | Tr: Scl Bayesi Criterion | Tr: Sq Error | Tra: Abs for A | Trai: of F | Trai: Aver Erro | Train: of Sq Error | Train: of C Wei Time | Train: Final Pred Error | Train: Mean Error | Tra: Roc Pre Err | Tra: Roc Squared Error | Train: F Func | Train: Misclassification Rate | Train: Number of Wrong Classifications |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Diab_Bernoulli_Sim | 120 | -51 | 171 | . | . | 0 | 1 | *** | *** | 0 | 59 | *** | . | . | . | . | 0.34 | 83 | 0.45 | 54 |
| OVERALL | 120 | -51 | 171 | * | . | 0 | 1 | *** | *** | 0 | 59 | *** | . | . | . | . | 0.34 | 83 | 0.45 | 54 |

**Figure 9: Classification Table Output for Neural Network**

For the NN model we used the option OUTFIT in proc Neural which stored the results of classification table as shown in Figure 9. This table contained data regarding *total number of wrong classifications* and *total observations*. So the overall classification accuracy was calculated as 1- (# of Wrong classification/Total number of observations), i.e, 1- (54/120) = 0.55

| _TARGET_ | _Y_SCALE_ | _X_SCALE_ | _DATA_ | _TYPE_ | _TASK_ | _METHOD_ | _KERNEL_ | _CONVER_ | N Interval vars | N Class vars | N Observations | N Traningobs | N_C_levels | N_Y_levels | Classification Error |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Diab_Bernoulli_Sim | BINARY | SCALE_X | TRAINING | _FITIND_ | C_CLAS | LSVM | Polynomial | _CONVER_ | 6 | 1 | 120 | 120 | 0 | 2 | 25 |
| Diab_Bernoulli_Sim | BINARY | SCALE_X | TRAINING | _ACCTAB_ | C_CLAS | LSVM | Polynomial | _CONVER_ | 29 | 12 | 0 | 13 | 66 | 0 | 0 |

**Figure 10: Classification Table Output for SVM**

For the SVM model we used the option OUTFIT in proc SVM and this option stored the data related to classification table which is as shown in Figure 10. This table contains information regarding N observations and Classification Error. The classification accuracy was calculated as 1- (Classification Error/Total number of observations), i.e, 1- (25/120) = 0.79

Thus performances of LR, NN and SVM were assessed using three different sample sizes 120, 240 and 500. Observations were drawn using simple random sampling for each iteration (sample) from a population that was generated based on different proportions described in Scenario1. Such a sampling approach was recommended to estimate the performance of the classifier by Sahiner, Chan & Hadjiiski (2008). The results of 1000 iterations for the performance of LR, NN, SVM when 10% data was missing and mode imputation was used are as shown in Figures 11, 12 &13 for three different sample sizes ,120,240 and 500, respectively.

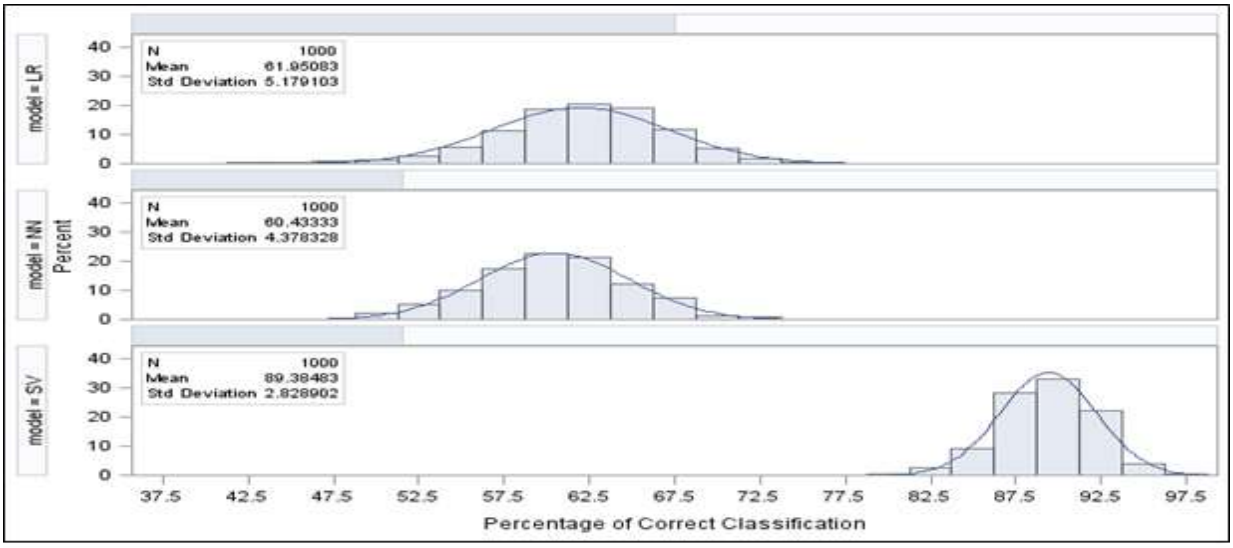


**Figure 11: Percent Correct Classification, LR, NN and SVM, Sample Size 120, Scenario 1**

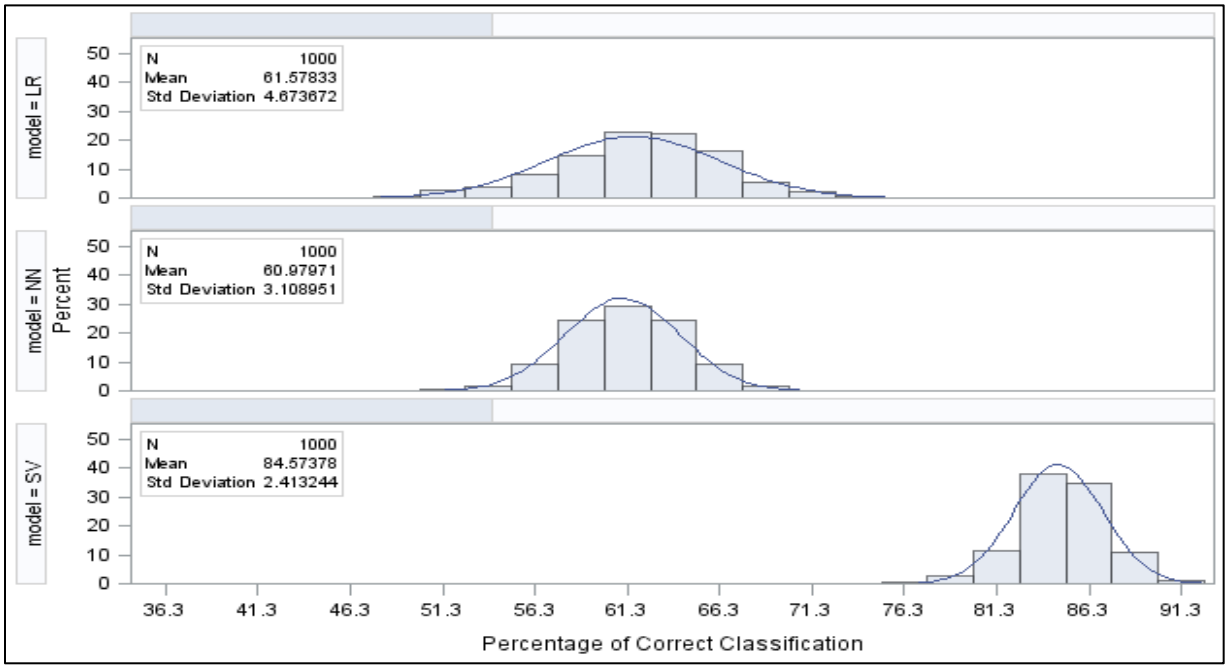


**Figure 12: Percent Correct Classification, LR, NN and SVM, Sample Size 240, Scenario 1**

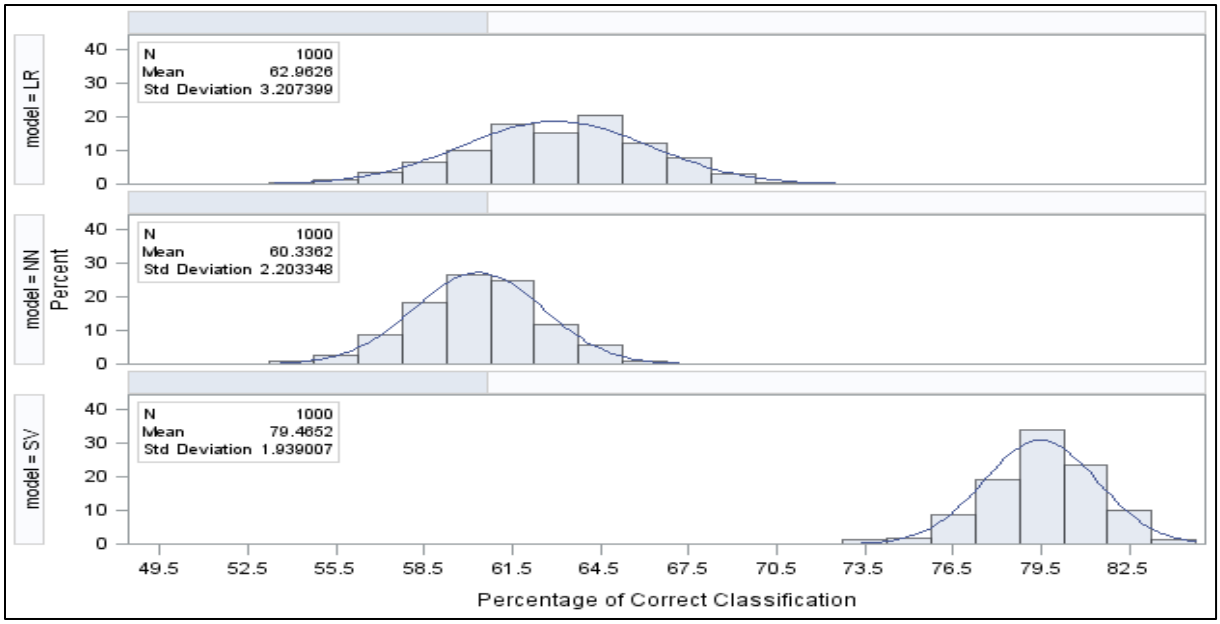


**Figure 13: Percent Correct Classification, LR, NN and SVM, Sample Size 500, Scenario 1**

Similarly the performances of LR, NN and SVM were assessed using three different sample sizes 120, 240 and 500. The observations were again drawn using simple random sampling for each iteration (sample) from a population that was generated based on different proportions described in Scenario 2. The results for the performance of LR, NN, SVM when 10% data was missing and mode imputation was used in Scenario 2 are as shown in Figures 14,15 &16 for three different sample sizes : 120,240 and 500, respectively.

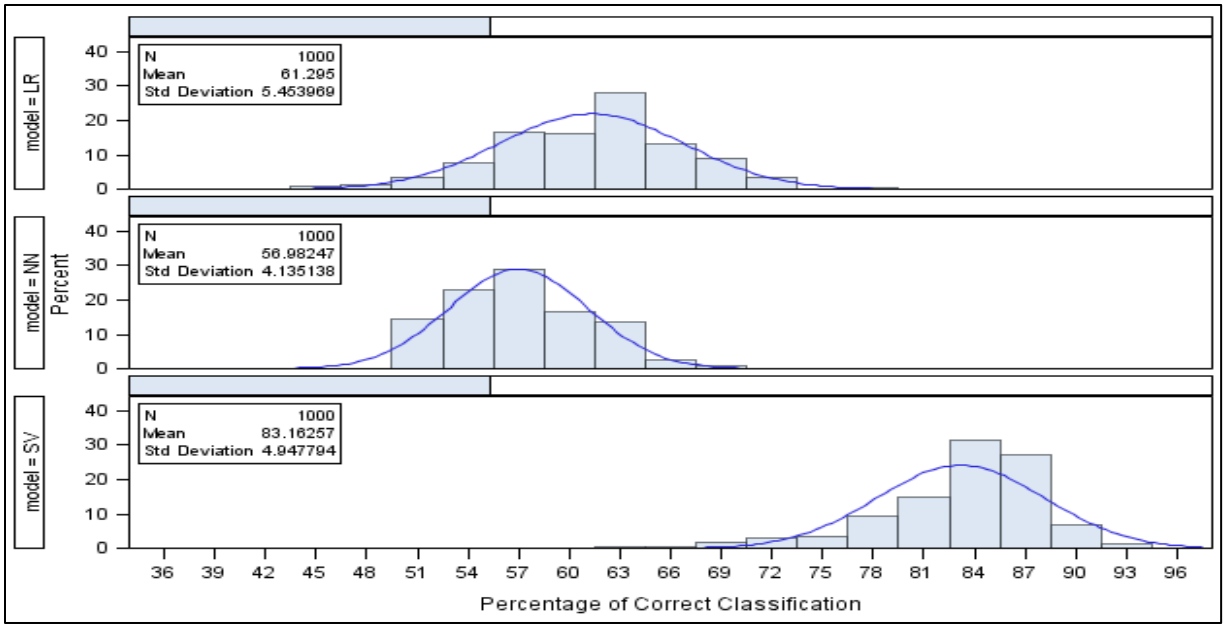


**Figure 14: Percent Correct Classification, LR, NN and SVM, Sample Size 120, Scenario 2**

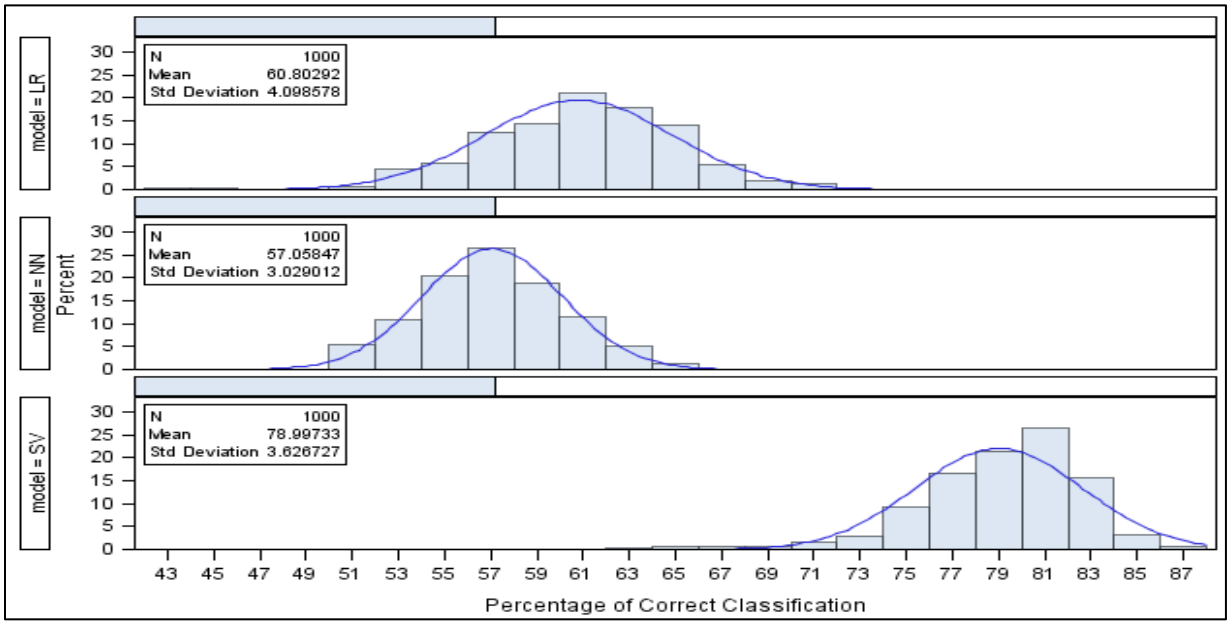


**Figure 15: Percent Correct Classification, LR, NN and SVM, Sample Size 500, Scenario 2**

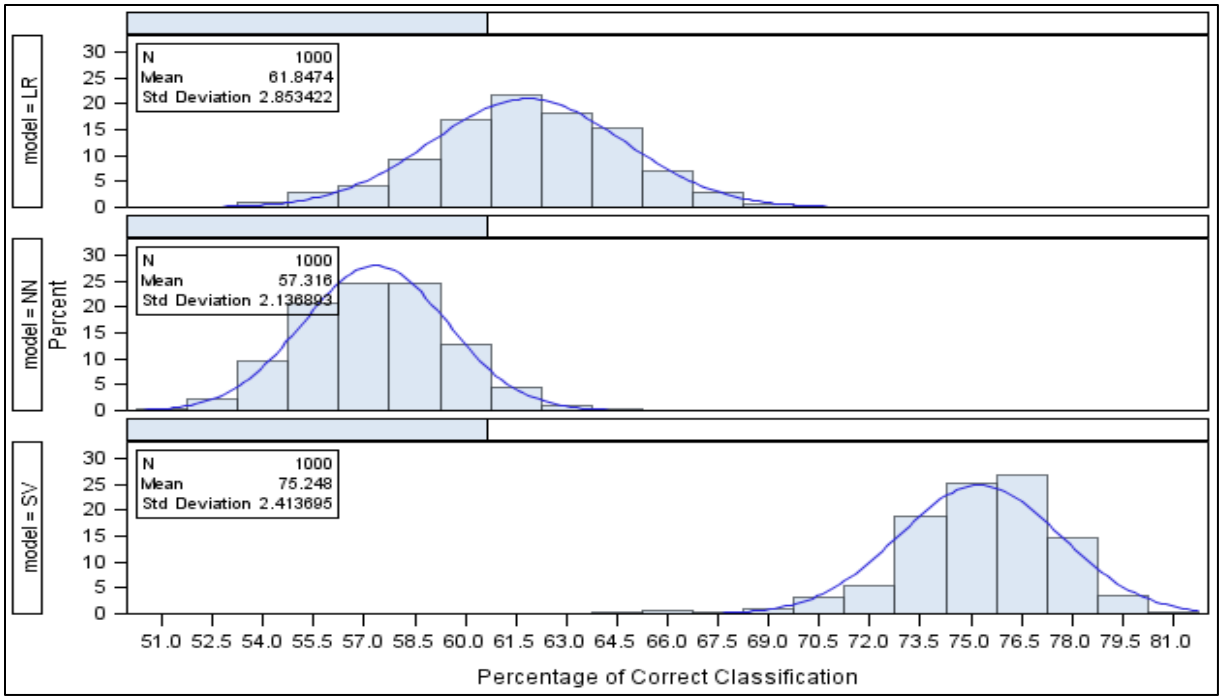


**Figure 16: Percent Correct Classification, LR, NN and SVM, Sample Size 500, Scenario 2**

For all three sample sizes, 120, 240 and 500 and when tested on two different populations generated based on two different frequency distributions, results consistently show that the performance of SVM (Scenario1: 89.1%, 84.43%, 79.48%, Scenario 2: 83.16 ,78.99,75.24) was far superior compared to LR (Scenario 1: 61.5%, 62.14%, 62.9%, Scenario 2: 61.29, 60.80, 61.84) or NN (Scenario 1: 59.39%, 60.65%, 60. 34%, Scenario 2: 56.98, 57.05, 57.31) models.

# CHAPTER 6. CONCLUSIONS

While classical LR is the default model of choice in many research areas such as healthcare and sports because of its ability to predict the probability of an outcome as a function of independent variables, results from our study showed that SVM is a more efficient and robust classification model in the presence of 10% missing data when mode imputation was used. This is because SVM generalizes the model efficiently by minimizing the classification error and minimizing the complexity of the model. A disadvantage of the LR model is that this technique is unable to identify nonlinear relationships whereas SVM uses nonlinear or more complex decision boundaries to classify the dataset based on various kernel functions. Unlike LR, SVM can prevent the model from being very sensitive to outliers in the data, resulting in a model that is capable of making good predictions for prospective analyses.

Although results from our study showed that SVM is a more effective classification model compared to LR or NN, there are several drawbacks related to SVM and NN models. First is the time required to identify the correct combination of various parameters to build a very efficient SVM and NN model, second is the increasing sample size that results in degradation of performance in SVM model and third is the computational burden of SVM and NN models compared to LR model.

The accuracy of a SVM depends on the choice of a kernel function, training method, estimation method and choice of parameter. Similarly the accuracy of the NN model depends on the right combination of architecture, number of hidden nodes, target layer combination function, hidden and target layer activation function, target layer error function, and training technique. For SVM if the kernel function is not chosen properly, SVM will be unable to find a separating hyperplane in feature space. Similarly for NN, inaccurate choice of architecture may

result in failure to achieve appropriate convergence. Also the power of a NN over a main effects LR model lies in the use of hidden nodes. These hidden nodes generate additional sets of parameter estimates that result in a more complex regression equation but there are no clear rules or formulas that help in determining the correct number of hidden nodes in the hidden layer.

Thus choosing the best combination of these parameters is a very difficult task because of the sensitivity of SVM and NN algorithm to these choices. Different choices often yield completely different results. Hence, it is necessary to first execute SVM and NN models based on different combinations of these parameters to identify the best possible combination that provides highest classification efficiency. This is a very time consuming and tedious process.

There was an increasing degradation of SVM model (Scenario1: 89.1%, 84.43%, 79.48%, Scenario 2: 83.16 ,78.99,75.24) as the sample size increased. One possible hypothesis for this phenomenon is because as sample size increases we can expect more data points closer to the hyperplane which could contribute to misclassification and hence gradually lead to deterioration in performance as sample size increases. Another major drawback of SVM and NN models are the higher computational burden compared to LR model. During our analysis, if LR models for sample sizes of 120, 240 and 500 took between 10-30 minutes to execute, it took somewhere between 90 – 240 minutes to execute NN models for sample sizes of 120,240 and 500 and between 180 - 1320 minutes to execute SVM models. All these processing times were for 1000 iterations and excluded the setup time that involves identifying the right combination of various parameters for NN and SVM models as mentioned earlier.

However our attempt in using mode based single imputation is just the first step towards identifying a very robust classification model in the presence of missing data and requires some consideration in relation to the plethora of options that are available.

## REFERENCES


Agresti, A., 1990, Categorical data analysis, New York: Wiley, 1990

Austin, P. C. (2007). A comparison of regression trees, logistic regression, generalized additive models, and multivariate adaptive regression splines for predicting AMI mortality. Statistics in Medicine, 26(15), 2937-2957.

Bellazzi, R., & Zupan, B. (2008). Predictive data mining in clinical medicine: current issues and guidelines. International journal of medical informatics,77(2), 81-97.

Biganzoli, E., Boracchi, P., Mariani, L., & Marubini, E. (1998). Feed forward neural networks for the analysis of censored survival data: a partial logistic regression approach. Statistics in Medicine, 17(10), 1169-1186.

Brick, J. M., & Kalton, G. (1996). Handling missing data in survey research. Statistical Methods in Medical Research, 5(3), 215-238.Chicago.

Chen, W. S., & Du, Y. K. (2009). Using neural networks and data mining techniques for the financial distress prediction model. Expert Systems with Applications, 36(2), 4075-4086.

Chen, S. T., Hsiao, Y. H., Huang, Y. L., Kuo, S. J., Tseng, H. S., Wu, H. K., & Chen, D. R. (2009). Comparative analysis of logistic regression, support vector machine and artificial neural network for the differential diagnosis of benign and malignant solid breast tumors by the use of three-dimensional power Doppler imaging. Korean Journal of Radiology, 10(5), 464-471.

Czibula, G., Czibula, I. G., & Găceanu, R. D. (2014). A Support Vector Machine Model For Intelligent Selection of Data Representations. Applied Soft Computing.

Dudoit, S., Fridlyand, J., & Speed, T. P. (2002). Comparison of discrimination methods for the classification of tumors using gene expression data. Journal of the American Statistical Association, 97(457), 77-87.

Dukart, J., Mueller, K., Barthel, H., Villringer, A., Sabri, O., & Schroeter, M. L. (2013). Meta-analysis based SVM classification enables accurate detection of Alzheimer's disease across different clinical centers using FDG-PET and MRI.Psychiatry Research: Neuroimaging, 212(3), 230-236.

Emin Tagluk, M., Akin, M., & Sezgin, N. (2010). Classıfication of sleep apnea by using wavelet transform and NNs. Expert Systems with Applications, 37(2), 1600-1607.

Ercan, O. (2011) Neural Network based modelling of the Marshall Stability of Asphalt Concrete. Expert Systems with Applications 38: 6025-6030.

Faraggi, D., & Simon, R. (1995). A neural network model for survival data.Statistics in Medicine, 14(1), 73-82.

Faraggi, D., LeBlanc, M., & Crowley, J. (2001). Understanding neural networks using regression trees: an application to multiple myeloma survival data.Statistics in medicine, 20(19), 2965-2976.

Fong, D. S., Aiello, L., Gardner, T. W., King, G. L., Blankenship, G., Cavallerano, J. D., ... & Klein, R. (2004). Retinopathy in diabetes. Diabetes Care, 27(suppl 1), s84-s87.

Gao, S., & Hui, S. L. (1997). Logistic regression models with missing covariate values for complex survey data. Statistics in Medicine, 16(21), 2419-2428.

Hanafizadeh P., Ravasan A.Z., and Khaki H. R., (2010). An expert system for perfume selection using NN, Expert Systems with Applications, 37(12), pp. 8879-8887.

Horton, N.J, Kleinman K,P. 2007. Much ado about nothing: A Comparison of Missing Data Methods and Software to fit Incomplete Data Regression Models. Am. Stat. 61:79–90.

Howard, A. A., Arnsten, J. H., & Gourevitch, M. N. (2004). Effect of Alcohol Consumption on Diabetes MellitusA Systematic Review. Annals of Internal Medicine, 140(3), 211-219.

Hung, M. S., Shanker, M., & Hu, M. Y. (2002). Estimating Breast Cancer Risks using Neural Networks. Journal of the Operational Research Society, 222-231.

Ibrahim, J. G., Chen, M. H., Lipsitz, S. R., & Herring, A. H. (2005). Missing-data methods for generalized linear models: A comparative review. Journal of the American Statistical Association, 100(469), 332-346.

Kannel, W. B., Hjortland, M., & Castelli, W. P. (1974). Role of diabetes in congestive heart failure: the Framingham study. The American Journal of Cardiology, 34(1), 29-34.

Koopman, R. J., Mainous, A. G., Diaz, V. A., & Geesey, M. E. (2005). Changes in age at diagnosis of type 2 diabetes mellitus in the United States, 1988 to 2000. The Annals of Family Medicine, 3(1), 60-63.

Lachin, J. M. (2008). Sample size evaluation for a multiply matched case–control study using the score test from a conditional logistic (discrete Cox PH) regression model. Statistics in Medicine, 27(14), 2509-2523.

Little, R. J., & Rubin, D. B. (2002). Statistical Analysis with Missing Data.

Lolas, S., & Olatunbosun, O. A. (2008). Prediction of vehicle reliability performance using Neural Networks. Expert Systems with Applications, 34(4), 2360-2369.

Louis, B., Agrawal, V. K., & Khadikar, P. V. (2010). Prediction of intrinsic solubility of generic drugs using MLR, ANN and SVM analyses. European journal of medicinal chemistry, 45(9), 4018-4025.

Lu, G., & Copas, J. B. (2004). Missing at random, likelihood ignorability and model completeness. The Annals of Statistics, 32(2), 754-765.

Meng, X. L. (2000). Missing data: dial M for???. Journal of the American Statistical Association, 95(452), 1325-1330.Chicago.

Muniz, A. M. S., Liu, H., Lyons, K. E., Pahwa, R., Liu, W., Nobre, F. F., & Nadal, J. (2010). Comparison among probabilistic neural network, support vector machine and logistic regression for evaluating the effect of subthalamic stimulation in Parkinson disease on ground reaction force during gait. Journal of Biomechanics, 43(4), 720-726.

Oğuz, H., Sarıtas, I., & Baydan, H. E. (2010). Prediction of diesel engine performance using biofuels with artificial neural network. Expert Systems with Applications, 37(9), 6579-6586.

O'Neill, T. J. (1980). The general distribution of the error rate of a classification procedure with application to LR discrimination. Journal of the American Statistical Association, 75(369), 154-160.

Pepe, M. S. (2003). *The statistical evaluation of medical tests for classification and prediction*. Oxford University Press.

Qiu, X., Tao, N., Tan, Y., & Wu, X. (2007). Constructing of the risk classification model of cervical cancer by NN. Expert Systems with Applications, 32(4), 1094-1099.

Ravi Kumar, P., & Ravi, V. (2007). Bankruptcy prediction in banks and firms via statistical and intelligent techniques–A review. European Journal of Operational Research, 180(1), 1-28.

Rubin, D. B. (1976). Inference and missing data. Biometrika, 63(3), 581-592.

Rupérez, M. J., Martín-Guerrero, J. D., Monserrat, C., & Alcañiz, M. (2011). Artificial neural networks for predicting dorsal pressures on the foot surface while walking. Expert Systems with Applications.

SAS Institute Inc. 2012. SAS/STAT 9.3 User's Guide. Cary, NC: SAS Institute Inc.

Sahiner, B., Chan, H., Hadjiiski, L., 2008. Classifier performance prediction for computer-aided diagnosis using a limited dataset. Medical Physics 35 (4), 1559–1570.

Schwartz, A.V., Vittinghoff, E., Sellmeyer, D.E., Feingold, K.R., de RN, Strotmeyer, E.S., et al. 2008, "Diabetes-related complications, glycemic control, and falls in older adults." Diabetes Care 2008; 31: 391–6.

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression.Statistics and Computing, 14(3), 199-222.

Steyerberg, E. W., Eijkemans, M. J., Harrell, F. E., & Habbema, J. D. F. (2000). Prognostic modelling with logistic regression analysis: a comparison of selection and estimation methods in small data sets. Statistics in Medicine, 19(8), 1059-1079.

Steyerberg, E. W., Eijkemans, M. J. C., Van Houwelingen, J. C., Lee, K. L., & Habbema, J. D. F. (2000). Prognostic models based on literature and individual patient data in logistic regression analysis. Statistics in Medicine, 19(2), 141-160.

Suárez Sánchez, A., Riesgo Fernández, P., Sánchez Lasheras, F., de Cos Juez, F. J., & García Nieto, P. J. (2011). Prediction of work-related accidents according to working conditions using support vector machines. Applied Mathematics and Computation, 218(7), 3539-3552.

Tian, J., Yu, B., Yu, D., & Ma, S. (2013). Clustering-Based Multiple Imputation via Gray Relational Analysis for Missing Data and Its Application to Aerospace Field. The Scientific World Journal, 2013.

Tobias, D. K., Pan, A., Jackson, C. L., O'Reilly, E. J., Ding, E. L., Willett, W. C., ... & Hu, F. B. (2014). Body-Mass Index and Mortality among Adults with Incident Type 2 Diabetes. New England Journal of Medicine, 370(3), 233-244.

Westreich, D., Lessler, J., & Funk, M. J. (2010). Propensity score estimation: neural networks, support vector machines, decision trees (CART), and meta-classifiers as alternatives to logistic regression. Journal of Clinical Epidemiology,63(8), 826-833.24.

Wicklin, R. (2013). Simulating Data with SAS. SAS Institute. Cary, NC.

Wilamowski, B. M., Iplikci, S., & Efe, M. O. (2001). An algorithm for fast convergence in training neural networks. In Neural Networks, 2001. Proceedings. IJCNN'01.International Joint Conference on (Vol. 3, pp. 1778-1782). IEEE.

## APPENDIX A. SAS CODE FOR SAMPLE SIZE 120 , SCENARIO 1

```
/* Creating a Library(Folder) for all dataset related to this work */
LIBNAME Sim 'T:\ms\Simulation\SampleSize120';


/* Importing the 6572 male dataset with missing data */
PROC IMPORT
      OUT= Sim.Datamale
      DATAFILE= "T:\ms\Simulation\datamales.csv"
      DBMS=CSV REPLACE;
      GETNAMES=YES;
      DATAROW=2;
RUN;

/* Proc Univariate to generate Mode for the 6 covariates and store them
Sim.Model1*/
ods output BasicMeasures=Sim.Mode1;
proc univariate data=Sim.Datamale ;
   var BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;
run;

/* Clean up code to just retain mode and delete all other unwanted
information */
data Sim.Mode (keep=VarName LocMeasure LocValue);
set Sim.Mode1;
where LocMeasure='Mode';
run;

/* Getting rid of Model1 table because the cleaner version of this is Model
*/
proc datasets library=Sim;
   delete Mode1;
run;

/* Ploting histogram to extract the proportions for BMI, CHF, Age, Eyes,
Falling, Alchohol_Consmp and levels from 6572 record dataset*/
/*Proc Gchart data=Sim.Datamale ;
      hbar BMIgrp /DISCRETE;
      hbar CHF /DISCRETE;
      hbar AgeGrp /DISCRETE;
      hbar EyesPastYear /DISCRETE;
      hbar Falls /DISCRETE;
      hbar LastDrank /DISCRETE;
run;*/

/* Using the proportion from the previous Proc Gchart to simulate Falling
data the relationship between
different levels { 1,2,3,4,5 } is maintained approximately like the
proportion in Datamale dataset */
data Sim.DataX1 (drop = i) ;
array prob [5] (0.63,0.26,0.03,0.07,0.01);
      call streaminit(1234);
      do i = 1 to 10000;
            Falls = rand("Table", of prob[*]);
```

```
      output;
end;
run;

/* Using the proportion from the previous Proc Gchart to simulate Eyes,
Alchohol, Age, Heart, BMI data
the proprotion with different levels are maintained approximately like the
proportion in Datamale dataset */
data Sim.DataX2 (drop = i) ;
      call streaminit(1234);
      do i = 1 to 10000;
            int = 1;
            EyesPastYear =rand("Bernoulli",0.3712);
            LastDrank = rand("Table", 0.27, 0.1626, 0.5674);
            AgeGrp = rand("Table", 0.1525, 0.4594, 0.2799, 0.1082);
            CHF =rand("Bernoulli",0.8777);
            BMIGrp = rand("Table", 0.0124, 0.1825, 0.4085, 0.3966);
      output;
end;
run;

/* Combining the Sim.Datax1 which contains Sim_Data_X_Y values for Falling
and
Combining Sim.Datax2 which contains Sim_Data_X_Y values for Eyes,
Alchohol,Age, Heart, and BMI into Sim.Data_X*/
data Sim.Data_X (drop= prob1 prob2 prob3 prob4 prob5);
merge Sim.DataX1 Sim.DataX2;
run;

/* Deleting Datax1 and Datax2 because they are already merged into Data_X */
proc datasets library=Sim;
   delete DataX1 DataX2;
run;

/* Run Proc Logistic on Clean Male Dataset with not missing data and store
the parameter estimates in
Estimates dataset */
proc logistic data=Sim.Datamale  DESCENDING  outest=Sim.Estimates ;
Class
      AgeGrp(Ref='1') BMIgrp(Ref='2') CHF(Ref='0') EyesPastYear(Ref='0')
Falls(Ref='0') LastDrank(Ref='1')
      / param=ref;
      model Diabetes = BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;
run;

/* Transform the Data_X dataset wich contains the Sim_Data_X_Y values for
Fall, Age, Alchohol, BMI into a
1 0 1, 1 1 1, 0 1 1, 0 0 0......etc matrix like this and store the values in
Transform10100 dataset*/
data Sim.Transform10100 (drop= BMIgrp CHF AgeGrp EyesPastYear  Falls
LastDrank) ;
set Sim.Data_X ;

      if BMIGrp=1 then
      do;
      BMIgrp1=1;
      BMIgrp3=0;
```

```
  BMIgrp4=0;
  end;
  else if BMIGrp=2 then
  do;
  BMIgrp1=0;
  BMIgrp3=0;
  BMIgrp4=0;
  end;
  else if BMIGrp=3 then
  do;
  BMIgrp1=0;
  BMIgrp3=1;
  BMIgrp4=0;
  end;
  else if BMIGrp=4 then
  do;
  BMIgrp1=0;
  BMIgrp3=0;
  BMIgrp4=1;
  end;


if LastDrank=1 then
  do;
  LastDrank2=0;
  LastDrank3=0;
  end;
  if LastDrank=2 then
  do;
  LastDrank2=1;
  LastDrank3=0;
  end;
  else if LastDrank=3 then
  do;
  LastDrank2=0;
  LastDrank3=1;
  end;


  if Falls=1 then
  do;
  Falls2=0;
  Falls3=0;
  Falls4=0;
  Falls5=0;
  end;
  else if Falls=2 then
  do;
  Falls2=1;
  Falls3=0;
  Falls4=0;
  Falls5=0;
  end;
  else if Falls=3 then
  do;
  Falls2=0;
  Falls3=1;
```

```
Falls4=0;
Falls5=0;
end;
else if Falls=4 then
do;
Falls2=0;
Falls3=0;
Falls4=1;
Falls5=0;
end;
else if Falls=5 then
do;
Falls2=0;
Falls3=0;
Falls4=0;
Falls5=1;
end;

if EyesPastYear=1 then
do;
EyesPastYear1=1;
end;
else if EyesPastYear=0 then
do;
EyesPastYear1=0;
end;

if AgeGrp=2 then
do;
AgeGrp2=1;
AgeGrp3=0;
AgeGrp4=0;
end;
else if AgeGrp=3 then
do;
AgeGrp2=0;
AgeGrp3=1;
AgeGrp4=0;
end;
else if AgeGrp=4 then
do;
AgeGrp2=0;
AgeGrp3=0;
AgeGrp4=1;
end;
else if AgeGrp=1 then
do;
AgeGrp2=0;
AgeGrp3=0;
AgeGrp4=0;
end;


if CHF=1 then
do;
CHF1=1;
end;
```

```
      else if CHF=0 then
      do;
      CHF1=0;
      end;


run;


/* Use Proc IML perform some array operations */
proc iml ;

      use sim.Estimates; /* Read all the parameter estimates stored in
Estimates dataset into an array z2 */
      read all ;
            z2 = Intercept || BMIgrp1 || BMIgrp3 || BMIgrp4 ||  LastDrank2 ||
LastDrank4 || Falls1 || Falls2 || Falls3 || Falls4 || EyesPastYear1 ||
AgeGrp2 || AgeGrp3 ||AgeGrp4 ||CHF1 ;

      use sim.Transform10100; /* Read all the Transform10100 dataset that
contains transformed Falls , BMI, Alchohol variables and store them in array
z4 */
      read all ;
            z4 =  int || BMIgrp1 || BMIgrp3 || BMIgrp4 || LastDrank2 ||
LastDrank3 || Falls2 ||  Falls3 || Falls4 || Falls5 || EyesPastYear1 ||
AgeGrp2 || AgeGrp3 || AgeGrp4 || CHF1 ;
            /* scoring on the Sim_Data_X_Y dataset */
            z5 = z4*t(z2);
            p = exp(z5)/(1+exp(z5));
            /* Generating data for Diabetes based on Simulation ....not very
strongly related to previous steps but just for further analysis */
            Diabetes = rand("Bernoulli",p);
            z6= p || Diabetes ; /* Merge two arrays p and Diabetes into one
z6 */

      /* putting these arrays back into a dataset Test 6 */
      create Sim.Test6 from z6;
      append from z6;
      close Sim.Test6;

quit;


/* Final Sim_Data_X_Y Data , contains both p and diabetes based on Bernouli
simuation*/
data Sim.Sim_Data_X_Y (rename=(Col1=p Col2=Diab_Bernoulli_Sim ) );
merge Sim.Data_X Sim.Test6;
if Col1<=0.5 then Diab_p=0;
else if Col1 > 0.5 then Diab_p=1;
id_Sim=_N_;
run;


/* Deleting Test6 because they are already merged into Sim_Data_X_Y */
proc datasets library=Sim;
   delete Test6 Data_X;
run;
```

```
proc logistic data=Sim.Sim_Data_X_Y  DESCENDING ;
Class
      AgeGrp(Ref='1') BMIgrp(Ref='2') CHF(Ref='0') EyesPastYear(Ref='0')
Falls(Ref='1') LastDrank(Ref='1') / param=ref;
      model Diab_Bernoulli_Sim = BMIgrp CHF AgeGrp EyesPastYear Falls
LastDrank ;
run;

/*---------------------Start------Generating Missing Data--------------------
----------*/

data Sim.DataMiss1 ;
      call streaminit(1234);
      do id_miss = 1 to 10000;
            EyesPastYea       =     rand("Bernoulli",0.1);
            LastDran          =     rand("Bernoulli",0.1);
            AgeGp                   =     rand("Bernoulli",0.1);
            CH                      =     rand("Bernoulli",0.1);
            BMIGp                   =     rand("Bernoulli",0.1);
            Fall              =     rand("Bernoulli",0.1);
      output;
end;
run;

data Sim.Sim_Miss_Data (keep= BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank
Diab_Bernoulli_Sim);
set Sim.DataMiss1  ; set Sim.Sim_Data_X_Y ;
if id_miss=id_Sim and EyesPastYea=1 then EyesPastYear=.;
if id_miss=id_Sim and AgeGp=1             then AgeGrp=.;
if id_miss=id_Sim and LastDran=1    then LastDrank=.;
if id_miss=id_Sim and CH=1                then CHF=.;
if id_miss=id_Sim and BMIGp=1             then BMIgrp=.;
if id_miss=id_Sim and Fall=1        then Falls=.;
run;

data Sim.Sim_Impute_Mode;
set Sim.Sim_Miss_Data;
if EyesPastYear=. then  EyesPastYear=1;
if AgeGrp=.             then  AgeGrp=2;
if LastDrank=.          then  LastDrank=4;
if BMIgrp=.             then  BMIgrp=3 ;
if Falls=.              then  Falls=0;
if CHF=.                then  CHF=0;
run;



proc datasets library=Sim;
   delete DataMiss1 Transform10100;
run;


%macro sim;

%let iterations = 1000;
```

```
%do i = 1 %to &iterations;


proc surveyselect data=Sim.Sim_Impute_Mode
  method = SRS
  reps = 1
  seed = 0
  N = 120
  out = Sim.Sample_Data;
run;

ods output  Classification=Sim.ctable1;;
proc logistic data=Sim.Sample_Data  DESCENDING ;
Class
      AgeGrp(Ref='1') BMIgrp(Ref='2') CHF(Ref='0') EyesPastYear(Ref='0')
Falls(Ref='1') LastDrank(Ref='1') / param=ref;
      model Diab_Bernoulli_Sim = BMIgrp CHF AgeGrp EyesPastYear Falls
LastDrank / ctable
      pprob=0.1 0.2 0.3 0.4 0.45 0.5 0.6 0.7 0.8 0.9;
run;


data Sim.ctable2;
set Sim.ctable1;
total=Sensitivity + Specificity;
run;

proc sort data=Sim.ctable2;
    by total ProbLevel;
run;

data Sim.ctable3;
   set Sim.ctable2;
   if _n_ = 10 then output;
   keep Correct;
run;

proc append base=Sim.LR_120_Classification data=Sim.ctable3 force;
                  run;

%end;
run;
%mend sim;

%sim;

proc datasets library=Sim;
   delete Ctable1 Ctable2 Ctable3;
run;

%macro sim;

%let iterations = 1000;

%do i = 1 %to &iterations;

                        proc surveyselect data=Sim.Sim_impute_mode
```

```
                                method = SRS
                                reps = 1
                                N = 120
                                out = Sim.ANN_Imputed_Data;
                  run;

                  proc dmdb batch data= Sim.ANN_Imputed_Data
out=Sim.ANN_DM_Smp10 dmdbcat=Sim.ANN_CAT_Smp10;
                                          CLASS    Diab_Bernoulli_Sim (DESC)
BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;
                                          TARGET   Diab_Bernoulli_Sim;

                        run;

                        proc neural data= Sim.ANN_Imputed_Data
dmdbcat=Sim.ANN_CAT_Smp10;
                                       NETOPTIONS DECAY=0 INVALIDTARGET=OMITCASE
OBJECT=LIKE;
                                          INPUT BMIgrp CHF AgeGrp
EyesPastYear Falls LastDrank /level= nominal id=inp STD=NO ;
                                          TARGET      Diab_Bernoulli_Sim /
level=nominal id=trg STD=NO ACT=SOFTMAX COMBINE=LINEAR ERROR=MBERNOULLI;
                                          ARCHI MLP   HIDDEN = 10 ;
                                          TRAIN TECHNIQUE= LEVMAR
MAXITER=1000 ;
                                          INITIAL;
                                          SCORE data= Sim.ANN_Imputed_Data
out=Sim.ANN_OUT_Smp1 outfit=Sim.ANN_FIT_Smp1 role=TRAIN;;


                        run;

                        data Sim.temp;
                              set Sim.ANN_FIT_Smp1;
                              concordant = round((1 - _misc_)*100,.01);
                              if _n_ = 2 then output;
                              keep concordant;
                  run;

                        proc append base=Sim.ann_concordant data=Sim.temp
force;
                  run;

%end;
run;
%mend sim;

%sim;

data Sim.ANN_120_Classification(rename=(concordant=Correct));
    set Sim.ann_concordant;
run;

%macro svm;

%let iterations = 1000;
```

```
%do i = 1 %to &iterations;

proc surveyselect data=Sim.Sim_impute_mode
                                 method = SRS
                                 reps = 1
                                 N = 120
                                 out = Sim.SVM_Imputed_Data;
                  run;



proc dmdb
      batch
      data= Sim.SVM_Imputed_Data
      out=Sim.SVM_DM_Smp10
      dmdbcat=Sim.SVM_CAT_Smp10;

      CLASS
      Diab_Bernoulli_Sim (DESC) ;

      VAR
      BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;

      TARGET
      Diab_Bernoulli_Sim;

run;



PROC SVM DATA=Sim.SVM_DM_Smp10 DMDBCAT=Sim.SVM_CAT_Smp10 CV=SPLIT FOLD=10
METHOD=LSVM KERNEL=POLYNOM K_PAR =5
OUT=Sim.SVM_OUT OUTEST= Sim.SVM_EST OUTFIT=Sim.SVM_FIT
TUN=GRID;
VAR BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;

TARGET Diab_Bernoulli_Sim;

C 0.1 TO 1.0 by 0.1;
RUN;

                        data Sim.svm_temp;
                                 set Sim.SVM_FIT;
                                 concordant = round((1 -
((_ACCU_)/(_NTRAIN_)))*100,0.01);
                                 if _n_ = 1 then output;
                                 keep concordant;
                  run;

                        proc append
                                          base=Sim.svm_120_Classification
                                          data=Sim.svm_temp force;
                  run;

%end;
run;
%mend svm;
```

```
%svm;

data Sim.ann_concordant(rename=(concordant=ann_correct));
    set Sim.Ann_120_Classification;
run;

data Sim.Svm_Classification_120_1000(rename=(concordant=correct));
set Sim.Svm_120_Classification;
model='SVM';
run;

data sim.LR_Classification_120_1000 ;
set sim.LR_120_Classification;
model='LR';
run;

data Sim.ann_Classification_120_1000(rename=(ann_correct=correct));
set Sim.Ann_120_Classification;
model='NN';
run;

data Classification_120_1000;
   merge sim.LR_Classification_120_1000 Sim.ann_Classification_120_1000
Sim.Svm_Classification_120_1000;
   by model;
run;

proc univariate data=Classification_120_1000 noprint;
   class model;
   histogram Correct / nrows        = 3
                       intertile    = 1
                       cprop
                       normal(noprint);
   inset n = "N" mean std / pos = nw;
run;
```

## APPENDIX B. SAS CODE FOR SAMPLE SIZE 240 , SCENARIO 1

```
/* Creating a Library(Folder) for all dataset related to this work */
LIBNAME Sim 'T:\ms\Simulation\SampleSize240';


/* Importing the 6572 male dataset with missing data */
PROC IMPORT
      OUT= Sim.Datamale
      DATAFILE= "T:\ms\Simulation\datamales.csv"
      DBMS=CSV REPLACE;
      GETNAMES=YES;
      DATAROW=2;
RUN;

/* Proc Univariate to generate Mode for the 6 covariates and store them
Sim.Model1*/
ods output BasicMeasures=Sim.Mode1;
proc univariate data=Sim.Datamale ;
   var BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;
run;

/* Clean up code to just retain mode and delete all other unwanted
information */
data Sim.Mode (keep=VarName LocMeasure LocValue);
set Sim.Mode1;
where LocMeasure='Mode';
run;

/* Getting rid of Model1 table because the cleaner version of this is Model
*/
proc datasets library=Sim;
   delete Mode1;
run;

/* Ploting histogram to extract the proportions for BMI, CHF, Age, Eyes,
Falling, Alchohol_Consmp and levels from 6572 record dataset*/
/*Proc Gchart data=Sim.Datamale ;
      hbar BMIgrp /DISCRETE;
      hbar CHF /DISCRETE;
      hbar AgeGrp /DISCRETE;
      hbar EyesPastYear /DISCRETE;
      hbar Falls /DISCRETE;
      hbar LastDrank /DISCRETE;
run;*/

/* Using the proportion from the previous Proc Gchart to simulate Falling
data the relationship between
different levels { 1,2,3,4,5 } is maintained approximately like the
proportion in Datamale dataset */
data Sim.DataX1 (drop = i) ;
array prob [5] (0.63,0.26,0.03,0.07,0.01);
      call streaminit(1234);
      do i = 1 to 10000;
```

```
            Falls = rand("Table", of prob[*]);
      output;
end;
run;

/* Using the proportion from the previous Proc Gchart to simulate Eyes,
Alchohol, Age, Heart, BMI data
the proprotion with different levels are maintained approximately like the
proportion in Datamale dataset */
data Sim.DataX2 (drop = i) ;
      call streaminit(1234);
      do i = 1 to 10000;
            int = 1;
            EyesPastYear =rand("Bernoulli",0.3712);
            LastDrank = rand("Table", 0.27, 0.1626, 0.5674);
            AgeGrp = rand("Table", 0.1525, 0.4594, 0.2799, 0.1082);
            CHF =rand("Bernoulli",0.8777);
            BMIGrp = rand("Table", 0.0124, 0.1825, 0.4085, 0.3966);
      output;
end;
run;

/* Combining the Sim.Datax1 which contains Sim_Data_X_Y values for Falling
and
Combining Sim.Datax2 which contains Sim_Data_X_Y values for Eyes,
Alchohol,Age, Heart, and BMI into Sim.Data_X*/
data Sim.Data_X (drop= prob1 prob2 prob3 prob4 prob5);
merge Sim.DataX1 Sim.DataX2;
run;

/* Deleting Datax1 and Datax2 because they are already merged into Data_X */
proc datasets library=Sim;
   delete DataX1 DataX2;
run;

/* Run Proc Logistic on Clean Male Dataset with not missing data and store
the parameter estimates in
Estimates dataset */
proc logistic data=Sim.Datamale  DESCENDING  outest=Sim.Estimates ;
Class
      AgeGrp(Ref='1') BMIgrp(Ref='2') CHF(Ref='0') EyesPastYear(Ref='0')
Falls(Ref='0') LastDrank(Ref='1')
      / param=ref;
      model Diabetes = BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;
run;

/* Transform the Data_X dataset wich contains the Sim_Data_X_Y values for
Fall, Age, Alchohol, BMI into a
1 0 1, 1 1 1, 0 1 1, 0 0 0......etc matrix like this and store the values in
Transform10100 dataset*/
data Sim.Transform10100 (drop= BMIgrp CHF AgeGrp EyesPastYear  Falls
LastDrank) ;
set Sim.Data_X ;

      if BMIGrp=1 then
      do;
      BMIgrp1=1;
```

```
  BMIgrp3=0;
  BMIgrp4=0;
  end;
  else if BMIGrp=2 then
  do;
  BMIgrp1=0;
  BMIgrp3=0;
  BMIgrp4=0;
  end;
  else if BMIGrp=3 then
  do;
  BMIgrp1=0;
  BMIgrp3=1;
  BMIgrp4=0;
  end;
  else if BMIGrp=4 then
  do;
  BMIgrp1=0;
  BMIgrp3=0;
  BMIgrp4=1;
  end;


if LastDrank=1 then
  do;
  LastDrank2=0;
  LastDrank3=0;
  end;
  if LastDrank=2 then
  do;
  LastDrank2=1;
  LastDrank3=0;
  end;
  else if LastDrank=3 then
  do;
  LastDrank2=0;
  LastDrank3=1;
  end;


  if Falls=1 then
  do;
  Falls2=0;
  Falls3=0;
  Falls4=0;
  Falls5=0;
  end;
  else if Falls=2 then
  do;
  Falls2=1;
  Falls3=0;
  Falls4=0;
  Falls5=0;
  end;
  else if Falls=3 then
  do;
  Falls2=0;
```

```
Falls3=1;
Falls4=0;
Falls5=0;
end;
else if Falls=4 then
do;
Falls2=0;
Falls3=0;
Falls4=1;
Falls5=0;
end;
else if Falls=5 then
do;
Falls2=0;
Falls3=0;
Falls4=0;
Falls5=1;
end;

if EyesPastYear=1 then
do;
EyesPastYear1=1;
end;
else if EyesPastYear=0 then
do;
EyesPastYear1=0;
end;

if AgeGrp=2 then
do;
AgeGrp2=1;
AgeGrp3=0;
AgeGrp4=0;
end;
else if AgeGrp=3 then
do;
AgeGrp2=0;
AgeGrp3=1;
AgeGrp4=0;
end;
else if AgeGrp=4 then
do;
AgeGrp2=0;
AgeGrp3=0;
AgeGrp4=1;
end;
else if AgeGrp=1 then
do;
AgeGrp2=0;
AgeGrp3=0;
AgeGrp4=0;
end;


if CHF=1 then
do;
CHF1=1;
```

```
      end;
      else if CHF=0 then
      do;
      CHF1=0;
      end;


run;


/* Use Proc IML perform some array operations */
proc iml ;

      use sim.Estimates; /* Read all the parameter estimates stored in
Estimates dataset into an array z2 */
      read all ;
            z2 = Intercept || BMIgrp1 || BMIgrp3 || BMIgrp4 ||  LastDrank2 ||
LastDrank4 || Falls1 || Falls2 || Falls3 || Falls4 || EyesPastYear1 ||
AgeGrp2 || AgeGrp3 ||AgeGrp4 ||CHF1 ;

      use sim.Transform10100; /* Read all the Transform10100 dataset that
contains transformed Falls , BMI, Alchohol variables and store them in array
z4 */
      read all ;
            z4 =  int || BMIgrp1 || BMIgrp3 || BMIgrp4 || LastDrank2 ||
LastDrank3 || Falls2 ||  Falls3 || Falls4 || Falls5 || EyesPastYear1 ||
AgeGrp2 || AgeGrp3 || AgeGrp4 || CHF1 ;
            /* scoring on the Sim_Data_X_Y dataset */
            z5 = z4*t(z2);
            p = exp(z5)/(1+exp(z5));
            /* Generating data for Diabetes based on Simulation ....not very
strongly related to previous steps but just for further analysis */
            Diabetes = rand("Bernoulli",p);
            z6= p || Diabetes ; /* Merge two arrays p and Diabetes into one
z6 */

      /* putting these arrays back into a dataset Test 6 */
      create Sim.Test6 from z6;
      append from z6;
      close Sim.Test6;

quit;


/* Final Sim_Data_X_Y Data , contains both p and diabetes based on Bernouli
simuation*/
data Sim.Sim_Data_X_Y (rename=(Col1=p Col2=Diab_Bernoulli_Sim ) );
merge Sim.Data_X Sim.Test6;
if Col1<=0.5 then Diab_p=0;
else if Col1 > 0.5 then Diab_p=1;
id_Sim=_N_;
run;


/* Deleting Test6 because they are already merged into Sim_Data_X_Y */
proc datasets library=Sim;
   delete Test6 Data_X;
```

```
run;

proc logistic data=Sim.Sim_Data_X_Y  DESCENDING ;
Class
      AgeGrp(Ref='1') BMIgrp(Ref='2') CHF(Ref='0') EyesPastYear(Ref='0')
Falls(Ref='1') LastDrank(Ref='1') / param=ref;
      model Diab_Bernoulli_Sim = BMIgrp CHF AgeGrp EyesPastYear Falls
LastDrank ;
run;

/*----------------------Start------Generating Missing Data--------------------
----------*/

data Sim.DataMiss1 ;
      call streaminit(1234);
      do id_miss = 1 to 10000;
            EyesPastYea       =     rand("Bernoulli",0.1);
            LastDran          =     rand("Bernoulli",0.1);
            AgeGp                   =     rand("Bernoulli",0.1);
            CH                      =     rand("Bernoulli",0.1);
            BMIGp                   =     rand("Bernoulli",0.1);
            Fall              =     rand("Bernoulli",0.1);
      output;
end;
run;

data Sim.Sim_Miss_Data (keep= BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank
Diab_Bernoulli_Sim);
set Sim.DataMiss1  ; set Sim.Sim_Data_X_Y ;
if id_miss=id_Sim and EyesPastYea=1 then EyesPastYear=.;
if id_miss=id_Sim and AgeGp=1             then AgeGrp=.;
if id_miss=id_Sim and LastDran=1    then LastDrank=.;
if id_miss=id_Sim and CH=1                then CHF=.;
if id_miss=id_Sim and BMIGp=1             then BMIgrp=.;
if id_miss=id_Sim and Fall=1        then Falls=.;
run;

data Sim.Sim_Impute_Mode;
set Sim.Sim_Miss_Data;
if EyesPastYear=. then  EyesPastYear=1;
if AgeGrp=.             then  AgeGrp=2;
if LastDrank=.          then  LastDrank=4;
if BMIgrp=.             then  BMIgrp=3 ;
if Falls=.              then  Falls=0;
if CHF=.                then  CHF=0;
run;



proc datasets library=Sim;
   delete DataMiss1 Transform10100;
run;

/*----------------------End---------Generating Missing Data------------------
-------------*/
```

```
/*-------------Start-------Randomly generate 10000 samples each sample of
size 120--------*/

%macro sim;

%let iterations = 1000;

%do i = 1 %to &iterations;


proc surveyselect data=Sim.Sim_Impute_Mode
  method = SRS
  reps = 1
  seed = 0
  N = 240
  out = Sim.Sample_Data;
run;

ods output  Classification=Sim.ctable1;;
proc logistic data=Sim.Sample_Data  DESCENDING ;
Class
      AgeGrp(Ref='1') BMIgrp(Ref='2') CHF(Ref='0') EyesPastYear(Ref='0')
Falls(Ref='1') LastDrank(Ref='1') / param=ref;
      model Diab_Bernoulli_Sim = BMIgrp CHF AgeGrp EyesPastYear Falls
LastDrank / ctable
      pprob=0.1 0.2 0.3 0.4 0.45 0.5 0.6 0.7 0.8 0.9;
run;


data Sim.ctable2;
set Sim.ctable1;
total=Sensitivity + Specificity;
run;

proc sort data=Sim.ctable2;
    by total ProbLevel;
run;

data Sim.ctable3;
   set Sim.ctable2;
   if _n_ = 10 then output;
   keep Correct;
run;

proc append base=Sim.LR_240_Classification data=Sim.ctable3 force;
                  run;

%end;
run;
%mend sim;

%sim;

proc datasets library=Sim;
   delete Ctable1 Ctable2 Ctable3;
run;
```

```
%macro sim;
%let iterations = 1000;

%do i = 1 %to &iterations;

                        proc surveyselect data=Sim.Sim_impute_mode
                              method = SRS
                              reps = 1
                              N = 240
                              out = Sim.ANN_Imputed_Data;
                  run;

                  proc dmdb batch data= Sim.ANN_Imputed_Data
out=Sim.ANN_DM_Smp10 dmdbcat=Sim.ANN_CAT_Smp10;
                                          CLASS    Diab_Bernoulli_Sim (DESC)
BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;
                                          TARGET   Diab_Bernoulli_Sim;

                        run;

                        proc neural data= Sim.ANN_Imputed_Data
dmdbcat=Sim.ANN_CAT_Smp10;
                                    NETOPTIONS DECAY=0 INVALIDTARGET=OMITCASE
OBJECT=LIKE;
                                          INPUT BMIgrp CHF AgeGrp
EyesPastYear Falls LastDrank /level= nominal id=inp STD=NO ;
                                          TARGET      Diab_Bernoulli_Sim /
level=nominal id=trg STD=NO ACT=SOFTMAX COMBINE=LINEAR ERROR=MBERNOULLI;
                                          ARCHI MLP   HIDDEN = 10 ;
                                          TRAIN TECHNIQUE= LEVMAR
MAXITER=1000 ;
                                          INITIAL;
                                          SCORE data= Sim.ANN_Imputed_Data
out=Sim.ANN_OUT_Smp1 outfit=Sim.ANN_FIT_Smp1 role=TRAIN;;

                        run;

                        data Sim.temp;
                              set Sim.ANN_FIT_Smp1;
                              concordant = round((1 - _misc_)*100,.01);
                              if _n_ = 2 then output;
                              keep concordant;
                  run;

                        proc append base=Sim.ann_concordant data=Sim.temp
force;
                  run;

%end;
run;
%mend sim;

%sim;

data Sim.ANN_240_Classification(rename=(concordant=Correct));
    set Sim.ann_concordant;
run;
```

```
%macro svm;

%let iterations = 1000;
%do i = 1 %to &iterations;
proc surveyselect data=Sim.Sim_impute_mode
                              method = SRS
                              reps = 1
                              N = 240
                              out = Sim.SVM_Imputed_Data;
run;

proc dmdb
      batch
      data= Sim.SVM_Imputed_Data
      out=Sim.SVM_DM_Smp10
      dmdbcat=Sim.SVM_CAT_Smp10;

      CLASS
      Diab_Bernoulli_Sim (DESC) ;

      VAR
      BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;

      TARGET
      Diab_Bernoulli_Sim;

run;

PROC SVM DATA=Sim.SVM_DM_Smp10 DMDBCAT=Sim.SVM_CAT_Smp10 CV=SPLIT FOLD=10
METHOD=LSVM KERNEL=POLYNOM K_PAR =5
OUT=Sim.SVM_OUT OUTEST= Sim.SVM_EST OUTFIT=Sim.SVM_FIT
TUN=GRID;
VAR BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;

TARGET Diab_Bernoulli_Sim;

C 0.1 TO 1.0 by 0.1;
RUN;

                        data Sim.svm_temp;
                              set Sim.SVM_FIT;
                              concordant = round((1 -
((_ACCU_)/(_NTRAIN_)))*100,0.01);
                              if _n_ = 1 then output;
                              keep concordant;
                  run;

                        proc append
                                          base=Sim.svm_240_Classification
                                          data=Sim.svm_temp force;
                  run;

%end;
run;
%mend svm;
%svm;
```

```
data Sim.ann_concordant(rename=(concordant=ann_correct));
    set Sim.Ann_240_Classification;
 run;

data Sim.Svm_Classification_240_1000(rename=(concordant=correct));
set Sim.Svm_240_Classification;
model='SVM';
run;

data sim.LR_Classification_240_1000 ;
set sim.LR_240_Classification;
model='LR';
run;

data Sim.ann_Classification_240_1000(rename=(ann_correct=correct));
set Sim.Ann_240_Classification;
model='NN';
run;

data Classification_240_1000;
   merge sim.LR_Classification_240_1000 Sim.ann_Classification_240_1000
Sim.Svm_Classification_240_1000;
   by model;
run;

proc univariate data=Classification_240_1000 noprint;
   class model;
   histogram Correct / nrows        = 3
                       intertile    = 1
                       cprop
                       normal(noprint);
   inset n = "N" mean std / pos = nw;
run;
```

## APPENDIX C. SAS CODE FOR SAMPLE SIZE 500 , SCENARIO 1

```
/* Creating a Library(Folder) for all dataset related to this work */
LIBNAME Sim 'T:\ms\Simulation\SampleSize500';


/* Importing the 6572 male dataset with missing data */
PROC IMPORT
      OUT= Sim.Datamale
      DATAFILE= "T:\ms\Simulation\datamales.csv"
      DBMS=CSV REPLACE;
      GETNAMES=YES;
      DATAROW=2;
RUN;

/* Proc Univariate to generate Mode for the 6 covariates and store them
Sim.Model1*/
ods output BasicMeasures=Sim.Mode1;
proc univariate data=Sim.Datamale ;
   var BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;
run;

/* Clean up code to just retain mode and delete all other unwanted
information */
data Sim.Mode (keep=VarName LocMeasure LocValue);
set Sim.Mode1;
where LocMeasure='Mode';
run;

/* Getting rid of Model1 table because the cleaner version of this is Model
*/
proc datasets library=Sim;
   delete Mode1;
run;

/* Ploting histogram to extract the proportions for BMI, CHF, Age, Eyes,
Falling, Alchohol_Consmp and levels from 6572 record dataset*/
/*Proc Gchart data=Sim.Datamale ;
      hbar BMIgrp /DISCRETE;
      hbar CHF /DISCRETE;
      hbar AgeGrp /DISCRETE;
      hbar EyesPastYear /DISCRETE;
      hbar Falls /DISCRETE;
      hbar LastDrank /DISCRETE;
run;*/

/* Using the proportion from the previous Proc Gchart to simulate Falling
data the relationship between
different levels { 1,2,3,4,5 } is maintained approximately like the
proportion in Datamale dataset */
data Sim.DataX1 (drop = i) ;
array prob [5] (0.63,0.26,0.03,0.07,0.01);
      call streaminit(1234);
      do i = 1 to 10000;
            Falls = rand("Table", of prob[*]);
      output;
```

```
end;
run;

/* Using the proportion from the previous Proc Gchart to simulate Eyes,
Alchohol, Age, Heart, BMI data
the proprotion with different levels are maintained approximately like the
proportion in Datamale dataset */
data Sim.DataX2 (drop = i) ;
      call streaminit(1234);
      do i = 1 to 10000;
            int = 1;
            EyesPastYear =rand("Bernoulli",0.3712);
            LastDrank = rand("Table", 0.27, 0.1626, 0.5674);
            AgeGrp = rand("Table", 0.1525, 0.4594, 0.2799, 0.1082);
            CHF =rand("Bernoulli",0.8777);
            BMIGrp = rand("Table", 0.0124, 0.1825, 0.4085, 0.3966);
      output;
end;
run;

/* Combining the Sim.Datax1 which contains Sim_Data_X_Y values for Falling
and
Combining Sim.Datax2 which contains Sim_Data_X_Y values for Eyes,
Alchohol,Age, Heart, and BMI into Sim.Data_X*/
data Sim.Data_X (drop= prob1 prob2 prob3 prob4 prob5);
merge Sim.DataX1 Sim.DataX2;
run;

/* Deleting Datax1 and Datax2 because they are already merged into Data_X */
proc datasets library=Sim;
   delete DataX1 DataX2;
run;

/* Run Proc Logistic on Clean Male Dataset with not missing data and store
the parameter estimates in
Estimates dataset */
proc logistic data=Sim.Datamale  DESCENDING  outest=Sim.Estimates ;
Class
      AgeGrp(Ref='1') BMIgrp(Ref='2') CHF(Ref='0') EyesPastYear(Ref='0')
Falls(Ref='0') LastDrank(Ref='1')
      / param=ref;
      model Diabetes = BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;
run;

/* Transform the Data_X dataset wich contains the Sim_Data_X_Y values for
Fall, Age, Alchohol, BMI into a
1 0 1, 1 1 1, 0 1 1, 0 0 0......etc matrix like this and store the values in
Transform10100 dataset*/
data Sim.Transform10100 (drop= BMIgrp CHF AgeGrp EyesPastYear  Falls
LastDrank) ;
set Sim.Data_X ;

      if BMIGrp=1 then
      do;
      BMIgrp1=1;
      BMIgrp3=0;
      BMIgrp4=0;
```

```
  end;
  else if BMIGrp=2 then
  do;
  BMIgrp1=0;
  BMIgrp3=0;
  BMIgrp4=0;
  end;
  else if BMIGrp=3 then
  do;
  BMIgrp1=0;
  BMIgrp3=1;
  BMIgrp4=0;
  end;
  else if BMIGrp=4 then
  do;
  BMIgrp1=0;
  BMIgrp3=0;
  BMIgrp4=1;
  end;


if LastDrank=1 then
  do;
  LastDrank2=0;
  LastDrank3=0;
  end;
  if LastDrank=2 then
  do;
  LastDrank2=1;
  LastDrank3=0;
  end;
  else if LastDrank=3 then
  do;
  LastDrank2=0;
  LastDrank3=1;
  end;


  if Falls=1 then
  do;
  Falls2=0;
  Falls3=0;
  Falls4=0;
  Falls5=0;
  end;
  else if Falls=2 then
  do;
  Falls2=1;
  Falls3=0;
  Falls4=0;
  Falls5=0;
  end;
  else if Falls=3 then
  do;
  Falls2=0;
  Falls3=1;
  Falls4=0;
```

```
Falls5=0;
end;
else if Falls=4 then
do;
Falls2=0;
Falls3=0;
Falls4=1;
Falls5=0;
end;
else if Falls=5 then
do;
Falls2=0;
Falls3=0;
Falls4=0;
Falls5=1;
end;

if EyesPastYear=1 then
do;
EyesPastYear1=1;
end;
else if EyesPastYear=0 then
do;
EyesPastYear1=0;
end;

if AgeGrp=2 then
do;
AgeGrp2=1;
AgeGrp3=0;
AgeGrp4=0;
end;
else if AgeGrp=3 then
do;
AgeGrp2=0;
AgeGrp3=1;
AgeGrp4=0;
end;
else if AgeGrp=4 then
do;
AgeGrp2=0;
AgeGrp3=0;
AgeGrp4=1;
end;
else if AgeGrp=1 then
do;
AgeGrp2=0;
AgeGrp3=0;
AgeGrp4=0;
end;


if CHF=1 then
do;
CHF1=1;
end;
else if CHF=0 then
```

```
      do;
      CHF1=0;
      end;


run;


/* Use Proc IML perform some array operations */
proc iml ;

      use sim.Estimates; /* Read all the parameter estimates stored in
Estimates dataset into an array z2 */
      read all ;
            z2 = Intercept || BMIgrp1 || BMIgrp3 || BMIgrp4 ||  LastDrank2 ||
LastDrank4 || Falls1 || Falls2 || Falls3 || Falls4 || EyesPastYear1 ||
AgeGrp2 || AgeGrp3 ||AgeGrp4 ||CHF1 ;

      use sim.Transform10100; /* Read all the Transform10100 dataset that
contains transformed Falls , BMI, Alchohol variables and store them in array
z4 */
      read all ;
            z4 =  int || BMIgrp1 || BMIgrp3 || BMIgrp4 || LastDrank2 ||
LastDrank3 || Falls2 ||  Falls3 || Falls4 || Falls5 || EyesPastYear1 ||
AgeGrp2 || AgeGrp3 || AgeGrp4 || CHF1 ;
            /* scoring on the Sim_Data_X_Y dataset */
            z5 = z4*t(z2);
            p = exp(z5)/(1+exp(z5));
            /* Generating data for Diabetes based on Simulation ....not very
strongly related to previous steps but just for further analysis */
            Diabetes = rand("Bernoulli",p);
            z6= p || Diabetes ; /* Merge two arrays p and Diabetes into one
z6 */

      /* putting these arrays back into a dataset Test 6 */
      create Sim.Test6 from z6;
      append from z6;
      close Sim.Test6;

quit;


/* Final Sim_Data_X_Y Data , contains both p and diabetes based on Bernouli
simuation*/
data Sim.Sim_Data_X_Y (rename=(Col1=p Col2=Diab_Bernoulli_Sim ) );
merge Sim.Data_X Sim.Test6;
if Col1<=0.5 then Diab_p=0;
else if Col1 > 0.5 then Diab_p=1;
id_Sim=_N_;
run;


/* Deleting Test6 because they are already merged into Sim_Data_X_Y */
proc datasets library=Sim;
   delete Test6 Data_X;
run;
```

```
proc logistic data=Sim.Sim_Data_X_Y  DESCENDING ;
Class
      AgeGrp(Ref='1') BMIgrp(Ref='2') CHF(Ref='0') EyesPastYear(Ref='0')
Falls(Ref='1') LastDrank(Ref='1') / param=ref;
      model Diab_Bernoulli_Sim = BMIgrp CHF AgeGrp EyesPastYear Falls
LastDrank ;
run;

/*--------------------Start------Generating Missing Data-----------------*/

data Sim.DataMiss1 ;
      call streaminit(1234);
      do id_miss = 1 to 10000;
            EyesPastYea       =     rand("Bernoulli",0.1);
            LastDran          =     rand("Bernoulli",0.1);
            AgeGp                   =     rand("Bernoulli",0.1);
            CH                      =     rand("Bernoulli",0.1);
            BMIGp                   =     rand("Bernoulli",0.1);
            Fall              =     rand("Bernoulli",0.1);
      output;
end;
run;

data Sim.Sim_Miss_Data (keep= BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank
Diab_Bernoulli_Sim);
set Sim.DataMiss1  ; set Sim.Sim_Data_X_Y ;
if id_miss=id_Sim and EyesPastYea=1 then EyesPastYear=.;
if id_miss=id_Sim and AgeGp=1             then AgeGrp=.;
if id_miss=id_Sim and LastDran=1    then LastDrank=.;
if id_miss=id_Sim and CH=1                then CHF=.;
if id_miss=id_Sim and BMIGp=1             then BMIgrp=.;
if id_miss=id_Sim and Fall=1        then Falls=.;
run;

data Sim.Sim_Impute_Mode;
set Sim.Sim_Miss_Data;
if EyesPastYear=. then  EyesPastYear=1;
if AgeGrp=.             then  AgeGrp=2;
if LastDrank=.          then  LastDrank=4;
if BMIgrp=.             then  BMIgrp=3 ;
if Falls=.              then  Falls=0;
if CHF=.                then  CHF=0;
run;

proc datasets library=Sim;
   delete DataMiss1 Transform10100;
run;


%macro sim;

%let iterations = 1000;

%do i = 1 %to &iterations;


proc surveyselect data=Sim.Sim_Impute_Mode
```

```
  method = SRS
  reps = 1
  seed = 0
  N = 500
  out = Sim.Sample_Data;
run;

ods output  Classification=Sim.ctable1;;
proc logistic data=Sim.Sample_Data  DESCENDING ;
Class
      AgeGrp(Ref='1') BMIgrp(Ref='2') CHF(Ref='0') EyesPastYear(Ref='0')
Falls(Ref='1') LastDrank(Ref='1') / param=ref;
      model Diab_Bernoulli_Sim = BMIgrp CHF AgeGrp EyesPastYear Falls
LastDrank / ctable
      pprob=0.1 0.2 0.3 0.4 0.45 0.5 0.6 0.7 0.8 0.9;
run;


data Sim.ctable2;
set Sim.ctable1;
total=Sensitivity + Specificity;
run;

proc sort data=Sim.ctable2;
    by total ProbLevel;
run;

data Sim.ctable3;
   set Sim.ctable2;
   if _n_ = 10 then output;
   keep Correct;
run;

proc append base=Sim.LR_500_Classification data=Sim.ctable3 force;
                  run;

%end;
run;
%mend sim;

%sim;

proc datasets library=Sim;
   delete Ctable1 Ctable2 Ctable3;
run;

%macro sim;

%let iterations = 1000;

%do i = 1 %to &iterations;

                        proc surveyselect data=Sim.Sim_impute_mode
                              method = SRS
                              reps = 1
                              N = 500
                              out = Sim.ANN_Imputed_Data;
```

```
                  run;

                  proc dmdb batch data= Sim.ANN_Imputed_Data
out=Sim.ANN_DM_Smp10 dmdbcat=Sim.ANN_CAT_Smp10;
                                          CLASS    Diab_Bernoulli_Sim (DESC)
BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;
                                          TARGET   Diab_Bernoulli_Sim;

                        run;

                        proc neural data= Sim.ANN_Imputed_Data
dmdbcat=Sim.ANN_CAT_Smp10;
                                       NETOPTIONS DECAY=0 INVALIDTARGET=OMITCASE
OBJECT=LIKE;
                                          INPUT BMIgrp CHF AgeGrp
EyesPastYear Falls LastDrank /level= nominal id=inp STD=NO ;
                                          TARGET      Diab_Bernoulli_Sim /
level=nominal id=trg STD=NO ACT=SOFTMAX COMBINE=LINEAR ERROR=MBERNOULLI;
                                          ARCHI MLP   HIDDEN = 10 ;
                                          TRAIN TECHNIQUE= LEVMAR
MAXITER=1000 ;
                                          INITIAL;
                                          SCORE data= Sim.ANN_Imputed_Data
out=Sim.ANN_OUT_Smp1 outfit=Sim.ANN_FIT_Smp1 role=TRAIN;;


                        run;

                        data Sim.temp;
                              set Sim.ANN_FIT_Smp1;
                              concordant = round((1 - _misc_)*100,.01);
                              if _n_ = 2 then output;
                              keep concordant;
                  run;

                        proc append base=Sim.ann_concordant data=Sim.temp
force;
                  run;

%end;
run;
%mend sim;

%sim;

data Sim.ANN_500_Classification(rename=(concordant=Correct));
    set Sim.ann_concordant;
run;

%macro svm;

%let iterations = 1000;

%do i = 1 %to &iterations;

proc surveyselect data=Sim.Sim_impute_mode
                              method = SRS
```

```
                                reps = 1
                                N = 500
                                out = Sim.SVM_Imputed_Data;
                  run;



proc dmdb
      batch
      data= Sim.SVM_Imputed_Data
      out=Sim.SVM_DM_Smp10
      dmdbcat=Sim.SVM_CAT_Smp10;

      CLASS
      Diab_Bernoulli_Sim (DESC) ;

      VAR
      BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;

      TARGET
      Diab_Bernoulli_Sim;

run;



PROC SVM DATA=Sim.SVM_DM_Smp10 DMDBCAT=Sim.SVM_CAT_Smp10 CV=SPLIT FOLD=10
METHOD=LSVM KERNEL=POLYNOM K_PAR =5
OUT=Sim.SVM_OUT OUTEST= Sim.SVM_EST OUTFIT=Sim.SVM_FIT
TUN=GRID;
VAR BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;

TARGET Diab_Bernoulli_Sim;

C 0.1 TO 1.0 by 0.1;
RUN;

                        data Sim.svm_temp;
                              set Sim.SVM_FIT;
                              concordant = round((1 -
((_ACCU_)/(_NTRAIN_)))*100,0.01);
                              if _n_ = 1 then output;
                              keep concordant;
                  run;

                        proc append
                                          base=Sim.svm_500_Classification
                                          data=Sim.svm_temp force;
                  run;

%end;
run;
%mend svm;

%svm;
```

```
data Sim.ann_concordant(rename=(concordant=ann_correct));
    set Sim.Ann_500_Classification;
run;

data Sim.Svm_Classification_500_1000(rename=(concordant=correct));
set Sim.Svm_500_Classification;
model='SVM';
run;

data sim.LR_Classification_500_1000 ;
set sim.LR_500_Classification;
model='LR';
run;

data Sim.ann_Classification_500_1000(rename=(ann_correct=correct));
set Sim.Ann_500_Classification;
model='NN';
run;

data Classification_500_1000;
   merge sim.LR_Classification_500_1000 Sim.ann_Classification_500_1000
Sim.Svm_Classification_500_1000;
   by model;
run;

proc univariate data=Classification_500_1000 noprint;
   class model;
   histogram Correct / nrows        = 3
                       intertile    = 1
                       cprop
                       normal(noprint);
   inset n = "N" mean std / pos = nw;
run;
```

## APPENDIX D. SAS CODE FOR SAMPLE SIZE 120 , SCENARIO 2

```
/* Creating a Library(Folder) for all dataset related to this work */
LIBNAME Sim 'T:\ms\Simulation\Scenario2\SampleSize120';


/* Importing the 6572 male dataset with missing data */
PROC IMPORT
      OUT= Sim.Datamale
      DATAFILE= "T:\ms\Simulation\datamales.csv"
      DBMS=CSV REPLACE;
      GETNAMES=YES;
      DATAROW=2;
RUN;

/* Proc Univariate to generate Mode for the 6 covariates and store them
Sim.Model1*/
ods output BasicMeasures=Sim.Mode1;
proc univariate data=Sim.Datamale ;
   var BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;
run;

/* Clean up code to just retain mode and delete all other unwanted
information */
data Sim.Mode (keep=VarName LocMeasure LocValue);
set Sim.Mode1;
where LocMeasure='Mode';
run;

/* Getting rid of Model1 table because the cleaner version of this is Model
*/
proc datasets library=Sim;
   delete Mode1;
run;

/* Ploting histogram to extract the proportions for BMI, CHF, Age, Eyes,
Falling, Alchohol_Consmp and levels from 6572 record dataset*/
/*Proc Gchart data=Sim.Datamale ;
      hbar BMIgrp /DISCRETE;
      hbar CHF /DISCRETE;
      hbar AgeGrp /DISCRETE;
      hbar EyesPastYear /DISCRETE;
      hbar Falls /DISCRETE;
      hbar LastDrank /DISCRETE;
run;*/

/* Using the proportion from the previous Proc Gchart to simulate Falling
data the relationship between
different levels { 1,2,3,4,5 } is maintained approximately like the
proportion in Datamale dataset */
data Sim.DataX1 (drop = i) ;
array prob [5] (0.01, 0.03, 0.07, 0.26, 0.63);
      call streaminit(1234);
      do i = 1 to 10000;
            Falls = rand("Table", of prob[*]);
```

```
      output;
end;
run;

/* Using the proportion from the previous Proc Gchart to simulate Eyes,
Alchohol, Age, Heart, BMI data
the proprotion with different levels are maintained approximately like the
proportion in Datamale dataset */
data Sim.DataX2 (drop = i) ;
      call streaminit(1234);
      do i = 1 to 10000;
            int = 1;
            EyesPastYear =rand("Bernoulli",0.85);
            LastDrank = rand("Table", 0.6,0.1,0.3);
            AgeGrp = rand("Table", 0.2025, 0.4594, 0.2799, 0.0582);
            CHF =rand("Bernoulli",0.95);
            BMIGrp = rand("Table", 0.012 , 0.18, 0.35, 0.458);
      output;
end;
run;

/* Combining the Sim.Datax1 which contains Sim_Data_X_Y values for Falling
and
Combining Sim.Datax2 which contains Sim_Data_X_Y values for Eyes,
Alchohol,Age, Heart, and BMI into Sim.Data_X*/
data Sim.Data_X (drop= prob1 prob2 prob3 prob4 prob5);
merge Sim.DataX1 Sim.DataX2;
run;

/* Deleting Datax1 and Datax2 because they are already merged into Data_X */
proc datasets library=Sim;
   delete DataX1 DataX2;
run;

/* Run Proc Logistic on Clean Male Dataset with not missing data and store
the parameter estimates in
Estimates dataset */
proc logistic data=Sim.Datamale  DESCENDING  outest=Sim.Estimates ;
Class
      AgeGrp(Ref='1') BMIgrp(Ref='2') CHF(Ref='0') EyesPastYear(Ref='0')
Falls(Ref='0') LastDrank(Ref='1')
      / param=ref;
      model Diabetes = BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;
run;

/* Transform the Data_X dataset wich contains the Sim_Data_X_Y values for
Fall, Age, Alchohol, BMI into a
1 0 1, 1 1 1, 0 1 1, 0 0 0......etc matrix like this and store the values in
Transform10100 dataset*/
data Sim.Transform10100 (drop= BMIgrp CHF AgeGrp EyesPastYear  Falls
LastDrank) ;
set Sim.Data_X ;

      if BMIGrp=1 then
      do;
      BMIgrp1=1;
      BMIgrp3=0;
```

```
  BMIgrp4=0;
  end;
  else if BMIGrp=2 then
  do;
  BMIgrp1=0;
  BMIgrp3=0;
  BMIgrp4=0;
  end;
  else if BMIGrp=3 then
  do;
  BMIgrp1=0;
  BMIgrp3=1;
  BMIgrp4=0;
  end;
  else if BMIGrp=4 then
  do;
  BMIgrp1=0;
  BMIgrp3=0;
  BMIgrp4=1;
  end;


if LastDrank=1 then
  do;
  LastDrank2=0;
  LastDrank3=0;
  end;
  if LastDrank=2 then
  do;
  LastDrank2=1;
  LastDrank3=0;
  end;
  else if LastDrank=3 then
  do;
  LastDrank2=0;
  LastDrank3=1;
  end;


  if Falls=1 then
  do;
  Falls2=0;
  Falls3=0;
  Falls4=0;
  Falls5=0;
  end;
  else if Falls=2 then
  do;
  Falls2=1;
  Falls3=0;
  Falls4=0;
  Falls5=0;
  end;
  else if Falls=3 then
  do;
  Falls2=0;
  Falls3=1;
```

```
Falls4=0;
Falls5=0;
end;
else if Falls=4 then
do;
Falls2=0;
Falls3=0;
Falls4=1;
Falls5=0;
end;
else if Falls=5 then
do;
Falls2=0;
Falls3=0;
Falls4=0;
Falls5=1;
end;

if EyesPastYear=1 then
do;
EyesPastYear1=1;
end;
else if EyesPastYear=0 then
do;
EyesPastYear1=0;
end;

if AgeGrp=2 then
do;
AgeGrp2=1;
AgeGrp3=0;
AgeGrp4=0;
end;
else if AgeGrp=3 then
do;
AgeGrp2=0;
AgeGrp3=1;
AgeGrp4=0;
end;
else if AgeGrp=4 then
do;
AgeGrp2=0;
AgeGrp3=0;
AgeGrp4=1;
end;
else if AgeGrp=1 then
do;
AgeGrp2=0;
AgeGrp3=0;
AgeGrp4=0;
end;


if CHF=1 then
do;
CHF1=1;
end;
```

```
      else if CHF=0 then
      do;
      CHF1=0;
      end;


run;


/* Use Proc IML perform some array operations */
proc iml ;

      use sim.Estimates; /* Read all the parameter estimates stored in
Estimates dataset into an array z2 */
      read all ;
            z2 = Intercept || BMIgrp1 || BMIgrp3 || BMIgrp4 ||  LastDrank2 ||
LastDrank4 || Falls1 || Falls2 || Falls3 || Falls4 || EyesPastYear1 ||
AgeGrp2 || AgeGrp3 ||AgeGrp4 ||CHF1 ;

      use sim.Transform10100; /* Read all the Transform10100 dataset that
contains transformed Falls , BMI, Alchohol variables and store them in array
z4 */
      read all ;
            z4 =  int || BMIgrp1 || BMIgrp3 || BMIgrp4 || LastDrank2 ||
LastDrank3 || Falls2 ||  Falls3 || Falls4 || Falls5 || EyesPastYear1 ||
AgeGrp2 || AgeGrp3 || AgeGrp4 || CHF1 ;
            /* scoring on the Sim_Data_X_Y dataset */
            z5 = z4*t(z2);
            p = exp(z5)/(1+exp(z5));
            /* Generating data for Diabetes based on Simulation ....not very
strongly related to previous steps but just for further analysis */
            Diabetes = rand("Bernoulli",p);
            z6= p || Diabetes ; /* Merge two arrays p and Diabetes into one
z6 */

      /* putting these arrays back into a dataset Test 6 */
      create Sim.Test6 from z6;
      append from z6;
      close Sim.Test6;

quit;


/* Final Sim_Data_X_Y Data , contains both p and diabetes based on Bernouli
simuation*/
data Sim.Sim_Data_X_Y (rename=(Col1=p Col2=Diab_Bernoulli_Sim ) );
merge Sim.Data_X Sim.Test6;
if Col1<=0.5 then Diab_p=0;
else if Col1 > 0.5 then Diab_p=1;
id_Sim=_N_;
run;


/* Deleting Test6 because they are already merged into Sim_Data_X_Y */
proc datasets library=Sim;
   delete Test6 Data_X;
run;
```

```
proc logistic data=Sim.Sim_Data_X_Y  DESCENDING ;
Class
      AgeGrp(Ref='1') BMIgrp(Ref='2') CHF(Ref='0') EyesPastYear(Ref='0')
Falls(Ref='1') LastDrank(Ref='1') / param=ref;
      model Diab_Bernoulli_Sim = BMIgrp CHF AgeGrp EyesPastYear Falls
LastDrank ;
run;

/*---------------------Start------Generating Missing Data--------------------
----------*/

data Sim.DataMiss1 ;
      call streaminit(1234);
      do id_miss = 1 to 10000;
            EyesPastYea       =     rand("Bernoulli",0.1);
            LastDran          =     rand("Bernoulli",0.1);
            AgeGp                   =     rand("Bernoulli",0.1);
            CH                      =     rand("Bernoulli",0.1);
            BMIGp                   =     rand("Bernoulli",0.1);
            Fall              =     rand("Bernoulli",0.1);
      output;
end;
run;

data Sim.Sim_Miss_Data (keep= BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank
Diab_Bernoulli_Sim);
set Sim.DataMiss1  ; set Sim.Sim_Data_X_Y ;
if id_miss=id_Sim and EyesPastYea=1 then EyesPastYear=.;
if id_miss=id_Sim and AgeGp=1             then AgeGrp=.;
if id_miss=id_Sim and LastDran=1    then LastDrank=.;
if id_miss=id_Sim and CH=1                then CHF=.;
if id_miss=id_Sim and BMIGp=1             then BMIgrp=.;
if id_miss=id_Sim and Fall=1        then Falls=.;
run;

data Sim.Sim_Impute_Mode;
set Sim.Sim_Miss_Data;
if EyesPastYear=. then  EyesPastYear=1;
if AgeGrp=.             then  AgeGrp=2;
if LastDrank=.          then  LastDrank=4;
if BMIgrp=.             then  BMIgrp=3 ;
if Falls=.              then  Falls=0;
if CHF=.                then  CHF=0;
run;



proc datasets library=Sim;
   delete DataMiss1 Transform10100;
run;

%macro sim;

%let iterations = 1000;

%do i = 1 %to &iterations;
```

```
proc surveyselect data=Sim.Sim_Impute_Mode
  method = SRS
  reps = 1
  seed = 0
  N = 120
  out = Sim.Sample_Data;
run;

ods output  Classification=Sim.ctable1;;
proc logistic data=Sim.Sample_Data  DESCENDING ;
Class
      AgeGrp(Ref='1') BMIgrp(Ref='2') CHF(Ref='0') EyesPastYear(Ref='0')
Falls(Ref='1') LastDrank(Ref='1') / param=ref;
      model Diab_Bernoulli_Sim = BMIgrp CHF AgeGrp EyesPastYear Falls
LastDrank / ctable
      pprob=0.1 0.2 0.3 0.4 0.45 0.5 0.6 0.7 0.8 0.9;
run;


data Sim.ctable2;
set Sim.ctable1;
total=Sensitivity + Specificity;
run;

proc sort data=Sim.ctable2;
    by total ProbLevel;
run;

data Sim.ctable3;
   set Sim.ctable2;
   if _n_ = 10 then output;
   keep Correct;
run;

proc append base=Sim.LR_120_Classification data=Sim.ctable3 force;
                  run;

%end;
run;
%mend sim;

%sim;

proc datasets library=Sim;
   delete Ctable1 Ctable2 Ctable3;
run;



%macro sim;

%let iterations = 1000;

%do i = 1 %to &iterations;
```

```
                        proc surveyselect data=Sim.Sim_impute_mode
                              method = SRS
                              reps = 1
                              N = 120
                              out = Sim.ANN_Imputed_Data;
                  run;

                  proc dmdb batch data= Sim.ANN_Imputed_Data
out=Sim.ANN_DM_Smp10 dmdbcat=Sim.ANN_CAT_Smp10;
                                          CLASS    Diab_Bernoulli_Sim (DESC)
BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;
                                          TARGET   Diab_Bernoulli_Sim;

                        run;

                        proc neural data= Sim.ANN_Imputed_Data
dmdbcat=Sim.ANN_CAT_Smp10;
                                       NETOPTIONS DECAY=0 INVALIDTARGET=OMITCASE
OBJECT=LIKE;
                                          INPUT BMIgrp CHF AgeGrp
EyesPastYear Falls LastDrank /level= nominal id=inp STD=NO ;
                                          TARGET      Diab_Bernoulli_Sim /
level=nominal id=trg STD=NO ACT=SOFTMAX COMBINE=LINEAR ERROR=MBERNOULLI;
                                          ARCHI MLP   HIDDEN = 10 ;
                                          TRAIN TECHNIQUE= LEVMAR
MAXITER=1000 ;
                                          INITIAL;
                                          SCORE data= Sim.ANN_Imputed_Data
out=Sim.ANN_OUT_Smp1 outfit=Sim.ANN_FIT_Smp1 role=TRAIN;;


                        run;

                        data Sim.temp;
                              set Sim.ANN_FIT_Smp1;
                              concordant = round((1 - _misc_)*100,.01);
                              if _n_ = 2 then output;
                              keep concordant;
                  run;

                        proc append base=Sim.ann_concordant data=Sim.temp
force;
                  run;

%end;
run;
%mend sim;

%sim;

data Sim.ANN_120_Classification(rename=(concordant=Correct));
    set Sim.ann_concordant;
run;
```

```
%macro svm;

%let iterations = 1000;

%do i = 1 %to &iterations;

proc surveyselect data=Sim.Sim_impute_mode
                                method = SRS
                                reps = 1
                                N = 120
                                out = Sim.SVM_Imputed_Data;
                  run;



proc dmdb
      batch
      data= Sim.SVM_Imputed_Data
      out=Sim.SVM_DM_Smp10
      dmdbcat=Sim.SVM_CAT_Smp10;

      CLASS
      Diab_Bernoulli_Sim (DESC) ;

      VAR
      BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;

      TARGET
      Diab_Bernoulli_Sim;

run;



PROC SVM DATA=Sim.SVM_DM_Smp10 DMDBCAT=Sim.SVM_CAT_Smp10 CV=SPLIT FOLD=10
METHOD=LSVM KERNEL=POLYNOM K_PAR =5
OUT=Sim.SVM_OUT OUTEST= Sim.SVM_EST OUTFIT=Sim.SVM_FIT
TUN=GRID;
VAR BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;

TARGET Diab_Bernoulli_Sim;

C 0.1 TO 1.0 by 0.1;
RUN;

                        data Sim.svm_temp;
                                set Sim.SVM_FIT;
                                concordant = round((1 -
((_ACCU_)/(_NTRAIN_)))*100,0.01);
                                if _n_ = 1 then output;
                                keep concordant;
                  run;

                        proc append
                                        base=Sim.svm_120_Classification
                                        data=Sim.svm_temp force;
                  run;
```

```
%end;
run;
%mend svm;

%svm;


data Sim.ann_concordant(rename=(concordant=ann_correct));
    set Sim.Ann_120_Classification;
run;

data Sim.Svm_Classification_120_1000(rename=(concordant=correct));
set Sim.Svm_120_Classification;
model='SVM';
run;

data sim.LR_Classification_120_1000 ;
set sim.LR_120_Classification;
model='LR';
run;

data Sim.ann_Classification_120_1000(rename=(ann_correct=correct));
set Sim.Ann_120_Classification;
model='NN';
run;



data Classification_120_1000;
   merge sim.LR_Classification_120_1000 Sim.ann_Classification_120_1000
Sim.Svm_Classification_120_1000;
   by model;
run;

proc univariate data=Classification_120_1000 noprint;
   class model;
   histogram Correct / nrows        = 3
                       intertile    = 1
                       cprop
                       normal(noprint);
   inset n = "N" mean std / pos = nw;
run;
```

## APPENDIX E. SAS CODE FOR SAMPLE SIZE 240 , SCENARIO 2

```
/* Creating a Library(Folder) for all dataset related to this work */
LIBNAME Sim 'T:\ms\Simulation\Scenario2\SampleSize240';

 /* Importing the 6572 male dataset with missing data */
PROC IMPORT
      OUT= Sim.Datamale
      DATAFILE= "T:\ms\Simulation\datamales.csv"
      DBMS=CSV REPLACE;
      GETNAMES=YES;
      DATAROW=2;
RUN;

/* Proc Univariate to generate Mode for the 6 covariates and store them
Sim.Model1*/
ods output BasicMeasures=Sim.Mode1;
proc univariate data=Sim.Datamale ;
   var BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;
run;

/* Clean up code to just retain mode and delete all other unwanted
information */
data Sim.Mode (keep=VarName LocMeasure LocValue);
set Sim.Mode1;
where LocMeasure='Mode';
run;

/* Getting rid of Model1 table because the cleaner version of this is Model
*/
proc datasets library=Sim;
   delete Mode1;
run;

/* Ploting histogram to extract the proportions for BMI, CHF, Age, Eyes,
Falling, Alchohol_Consmp and levels from 6572 record dataset*/
/*Proc Gchart data=Sim.Datamale ;
      hbar BMIgrp /DISCRETE;
      hbar CHF /DISCRETE;
      hbar AgeGrp /DISCRETE;
      hbar EyesPastYear /DISCRETE;
      hbar Falls /DISCRETE;
      hbar LastDrank /DISCRETE;
run;*/

/* Using the proportion from the previous Proc Gchart to simulate Falling
data the relationship between
different levels { 1,2,3,4,5 } is maintained approximately like the
proportion in Datamale dataset */
data Sim.DataX1 (drop = i) ;
array prob [5] (0.01, 0.03, 0.07, 0.26, 0.63);
      call streaminit(1234);
      do i = 1 to 10000;
            Falls = rand("Table", of prob[*]);
      output;
```

```
end;
run;

/* Using the proportion from the previous Proc Gchart to simulate Eyes,
Alchohol, Age, Heart, BMI data
the proprotion with different levels are maintained approximately like the
proportion in Datamale dataset */
data Sim.DataX2 (drop = i) ;
      call streaminit(1234);
      do i = 1 to 10000;
            int = 1;
            EyesPastYear =rand("Bernoulli",0.85);
            LastDrank = rand("Table", 0.6,0.1,0.3);
            AgeGrp = rand("Table", 0.2025, 0.4594, 0.2799, 0.0582);
            CHF =rand("Bernoulli",0.95);
            BMIGrp = rand("Table", 0.012 , 0.18, 0.35, 0.458);
      output;
end;
run;

/* Combining the Sim.Datax1 which contains Sim_Data_X_Y values for Falling
and
Combining Sim.Datax2 which contains Sim_Data_X_Y values for Eyes,
Alchohol,Age, Heart, and BMI into Sim.Data_X*/
data Sim.Data_X (drop= prob1 prob2 prob3 prob4 prob5);
merge Sim.DataX1 Sim.DataX2;
run;

/* Deleting Datax1 and Datax2 because they are already merged into Data_X */
proc datasets library=Sim;
   delete DataX1 DataX2;
run;

/* Run Proc Logistic on Clean Male Dataset with not missing data and store
the parameter estimates in
Estimates dataset */
proc logistic data=Sim.Datamale  DESCENDING  outest=Sim.Estimates ;
Class
      AgeGrp(Ref='1') BMIgrp(Ref='2') CHF(Ref='0') EyesPastYear(Ref='0')
Falls(Ref='0') LastDrank(Ref='1')
      / param=ref;
      model Diabetes = BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;
run;

/* Transform the Data_X dataset wich contains the Sim_Data_X_Y values for
Fall, Age, Alchohol, BMI into a
1 0 1, 1 1 1, 0 1 1, 0 0 0......etc matrix like this and store the values in
Transform10100 dataset*/
data Sim.Transform10100 (drop= BMIgrp CHF AgeGrp EyesPastYear  Falls
LastDrank) ;
set Sim.Data_X ;

      if BMIGrp=1 then
      do;
      BMIgrp1=1;
      BMIgrp3=0;
      BMIgrp4=0;
```

```
  end;
  else if BMIGrp=2 then
  do;
  BMIgrp1=0;
  BMIgrp3=0;
  BMIgrp4=0;
  end;
  else if BMIGrp=3 then
  do;
  BMIgrp1=0;
  BMIgrp3=1;
  BMIgrp4=0;
  end;
  else if BMIGrp=4 then
  do;
  BMIgrp1=0;
  BMIgrp3=0;
  BMIgrp4=1;
  end;


if LastDrank=1 then
  do;
  LastDrank2=0;
  LastDrank3=0;
  end;
  if LastDrank=2 then
  do;
  LastDrank2=1;
  LastDrank3=0;
  end;
  else if LastDrank=3 then
  do;
  LastDrank2=0;
  LastDrank3=1;
  end;


  if Falls=1 then
  do;
  Falls2=0;
  Falls3=0;
  Falls4=0;
  Falls5=0;
  end;
  else if Falls=2 then
  do;
  Falls2=1;
  Falls3=0;
  Falls4=0;
  Falls5=0;
  end;
  else if Falls=3 then
  do;
  Falls2=0;
  Falls3=1;
  Falls4=0;
```

```
Falls5=0;
end;
else if Falls=4 then
do;
Falls2=0;
Falls3=0;
Falls4=1;
Falls5=0;
end;
else if Falls=5 then
do;
Falls2=0;
Falls3=0;
Falls4=0;
Falls5=1;
end;

if EyesPastYear=1 then
do;
EyesPastYear1=1;
end;
else if EyesPastYear=0 then
do;
EyesPastYear1=0;
end;

if AgeGrp=2 then
do;
AgeGrp2=1;
AgeGrp3=0;
AgeGrp4=0;
end;
else if AgeGrp=3 then
do;
AgeGrp2=0;
AgeGrp3=1;
AgeGrp4=0;
end;
else if AgeGrp=4 then
do;
AgeGrp2=0;
AgeGrp3=0;
AgeGrp4=1;
end;
else if AgeGrp=1 then
do;
AgeGrp2=0;
AgeGrp3=0;
AgeGrp4=0;
end;


if CHF=1 then
do;
CHF1=1;
end;
else if CHF=0 then
```

```
      do;
      CHF1=0;
      end;


run;

/* Use Proc IML perform some array operations */
proc iml ;

      use sim.Estimates; /* Read all the parameter estimates stored in
Estimates dataset into an array z2 */
      read all ;
            z2 = Intercept || BMIgrp1 || BMIgrp3 || BMIgrp4 ||  LastDrank2 ||
LastDrank4 || Falls1 || Falls2 || Falls3 || Falls4 || EyesPastYear1 ||
AgeGrp2 || AgeGrp3 ||AgeGrp4 ||CHF1 ;

      use sim.Transform10100; /* Read all the Transform10100 dataset that
contains transformed Falls , BMI, Alchohol variables and store them in array
z4 */
      read all ;
            z4 =  int || BMIgrp1 || BMIgrp3 || BMIgrp4 || LastDrank2 ||
LastDrank3 || Falls2 ||  Falls3 || Falls4 || Falls5 || EyesPastYear1 ||
AgeGrp2 || AgeGrp3 || AgeGrp4 || CHF1 ;
            /* scoring on the Sim_Data_X_Y dataset */
            z5 = z4*t(z2);
            p = exp(z5)/(1+exp(z5));
            /* Generating data for Diabetes based on Simulation ....not very
strongly related to previous steps but just for further analysis */
            Diabetes = rand("Bernoulli",p);
            z6= p || Diabetes ; /* Merge two arrays p and Diabetes into one
z6 */

      /* putting these arrays back into a dataset Test 6 */
      create Sim.Test6 from z6;
      append from z6;
      close Sim.Test6;

quit;


/* Final Sim_Data_X_Y Data , contains both p and diabetes based on Bernouli
simuation*/
data Sim.Sim_Data_X_Y (rename=(Col1=p Col2=Diab_Bernoulli_Sim ) );
merge Sim.Data_X Sim.Test6;
if Col1<=0.5 then Diab_p=0;
else if Col1 > 0.5 then Diab_p=1;
id_Sim=_N_;
run;


/* Deleting Test6 because they are already merged into Sim_Data_X_Y */
proc datasets library=Sim;
   delete Test6 Data_X;
run;

proc logistic data=Sim.Sim_Data_X_Y  DESCENDING ;
```

```
Class
      AgeGrp(Ref='1') BMIgrp(Ref='2') CHF(Ref='0') EyesPastYear(Ref='0')
Falls(Ref='1') LastDrank(Ref='1') / param=ref;
      model Diab_Bernoulli_Sim = BMIgrp CHF AgeGrp EyesPastYear Falls
LastDrank ;
run;

/*---------------------Start------Generating Missing Data--------------------
----------*/

data Sim.DataMiss1 ;
      call streaminit(1234);
      do id_miss = 1 to 10000;
            EyesPastYea       =     rand("Bernoulli",0.1);
            LastDran          =     rand("Bernoulli",0.1);
            AgeGp                   =     rand("Bernoulli",0.1);
            CH                      =     rand("Bernoulli",0.1);
            BMIGp                   =     rand("Bernoulli",0.1);
            Fall              =     rand("Bernoulli",0.1);
      output;
end;
run;

data Sim.Sim_Miss_Data (keep= BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank
Diab_Bernoulli_Sim);
set Sim.DataMiss1  ; set Sim.Sim_Data_X_Y ;
if id_miss=id_Sim and EyesPastYea=1 then EyesPastYear=.;
if id_miss=id_Sim and AgeGp=1             then AgeGrp=.;
if id_miss=id_Sim and LastDran=1    then LastDrank=.;
if id_miss=id_Sim and CH=1                then CHF=.;
if id_miss=id_Sim and BMIGp=1             then BMIgrp=.;
if id_miss=id_Sim and Fall=1        then Falls=.;
run;

data Sim.Sim_Impute_Mode;
set Sim.Sim_Miss_Data;
if EyesPastYear=. then  EyesPastYear=1;
if AgeGrp=.             then  AgeGrp=2;
if LastDrank=.          then  LastDrank=4;
if BMIgrp=.             then  BMIgrp=3 ;
if Falls=.              then  Falls=0;
if CHF=.                then  CHF=0;
run;



proc datasets library=Sim;
   delete DataMiss1 Transform10100;
run;


%macro sim;

%let iterations = 1000;

%do i = 1 %to &iterations;
```

```
proc surveyselect data=Sim.Sim_Impute_Mode
  method = SRS
  reps = 1
  seed = 0
  N = 240
  out = Sim.Sample_Data;
run;

ods output  Classification=Sim.ctable1;;
proc logistic data=Sim.Sample_Data  DESCENDING ;
Class
      AgeGrp(Ref='1') BMIgrp(Ref='2') CHF(Ref='0') EyesPastYear(Ref='0')
Falls(Ref='1') LastDrank(Ref='1') / param=ref;
      model Diab_Bernoulli_Sim = BMIgrp CHF AgeGrp EyesPastYear Falls
LastDrank / ctable
      pprob=0.1 0.2 0.3 0.4 0.45 0.5 0.6 0.7 0.8 0.9;
run;


data Sim.ctable2;
set Sim.ctable1;
total=Sensitivity + Specificity;
run;

proc sort data=Sim.ctable2;
    by total ProbLevel;
run;

data Sim.ctable3;
   set Sim.ctable2;
   if _n_ = 10 then output;
   keep Correct;
run;

proc append base=Sim.LR_240_Classification data=Sim.ctable3 force;
                  run;

%end;
run;
%mend sim;

%sim;

proc datasets library=Sim;
   delete Ctable1 Ctable2 Ctable3;
run;

%macro sim;

%let iterations = 1000;

%do i = 1 %to &iterations;

                        proc surveyselect data=Sim.Sim_impute_mode
                              method = SRS
                              reps = 1
```

```
                                N = 240
                                out = Sim.ANN_Imputed_Data;
                    run;

                    proc dmdb batch data= Sim.ANN_Imputed_Data
out=Sim.ANN_DM_Smp10 dmdbcat=Sim.ANN_CAT_Smp10;
                                            CLASS    Diab_Bernoulli_Sim (DESC)
BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;
                                            TARGET   Diab_Bernoulli_Sim;

                            run;

                            proc neural data= Sim.ANN_Imputed_Data
dmdbcat=Sim.ANN_CAT_Smp10;
                                        NETOPTIONS DECAY=0 INVALIDTARGET=OMITCASE
OBJECT=LIKE;
                                            INPUT BMIgrp CHF AgeGrp
EyesPastYear Falls LastDrank /level= nominal id=inp STD=NO ;
                                            TARGET      Diab_Bernoulli_Sim /
level=nominal id=trg STD=NO ACT=SOFTMAX COMBINE=LINEAR ERROR=MBERNOULLI;
                                            ARCHI MLP   HIDDEN = 10 ;
                                            TRAIN TECHNIQUE= LEVMAR
MAXITER=1000 ;
                                            INITIAL;
                                            SCORE data= Sim.ANN_Imputed_Data
out=Sim.ANN_OUT_Smp1 outfit=Sim.ANN_FIT_Smp1 role=TRAIN;;


                            run;

                            data Sim.temp;
                                  set Sim.ANN_FIT_Smp1;
                                  concordant = round((1 - _misc_)*100,.01);
                                  if _n_ = 2 then output;
                                  keep concordant;
                    run;

                            proc append base=Sim.ann_concordant data=Sim.temp
force;
                    run;

%end;
run;
%mend sim;

%sim;

data Sim.ANN_240_Classification(rename=(concordant=Correct));
    set Sim.ann_concordant;
run;


%macro svm;

%let iterations = 1000;

%do i = 1 %to &iterations;
```

```
proc surveyselect data=Sim.Sim_impute_mode
                                method = SRS
                                reps = 1
                                N = 240
                                out = Sim.SVM_Imputed_Data;
                  run;



proc dmdb
      batch
      data= Sim.SVM_Imputed_Data
      out=Sim.SVM_DM_Smp10
      dmdbcat=Sim.SVM_CAT_Smp10;

      CLASS
      Diab_Bernoulli_Sim (DESC) ;

      VAR
      BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;

      TARGET
      Diab_Bernoulli_Sim;

run;



PROC SVM DATA=Sim.SVM_DM_Smp10 DMDBCAT=Sim.SVM_CAT_Smp10 CV=SPLIT FOLD=10
METHOD=LSVM KERNEL=POLYNOM K_PAR =5
OUT=Sim.SVM_OUT OUTEST= Sim.SVM_EST OUTFIT=Sim.SVM_FIT
TUN=GRID;
VAR BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;

TARGET Diab_Bernoulli_Sim;

C 0.1 TO 1.0 by 0.1;
RUN;

                        data Sim.svm_temp;
                              set Sim.SVM_FIT;
                              concordant = round((1 -
((_ACCU_)/(_NTRAIN_)))*100,0.01);
                              if _n_ = 1 then output;
                              keep concordant;
                  run;

                        proc append
                                          base=Sim.svm_240_Classification
                                          data=Sim.svm_temp force;
                  run;

%end;
run;
%mend svm;
```

```
%svm;



data Sim.ann_concordant(rename=(concordant=ann_correct));
    set Sim.Ann_240_Classification;
run;

data Sim.Svm_Classification_240_1000(rename=(concordant=correct));
set Sim.Svm_240_Classification;
model='SVM';
run;

data sim.LR_Classification_240_1000 ;
set sim.LR_240_Classification;
model='LR';
run;

data Sim.ann_Classification_240_1000(rename=(ann_correct=correct));
set Sim.Ann_240_Classification;
model='NN';
run;

data Classification_240_1000;
   merge sim.LR_Classification_240_1000 Sim.ann_Classification_240_1000
Sim.Svm_Classification_240_1000;
   by model;
run;

proc univariate data=Classification_240_1000 noprint;
   class model;
   histogram Correct / nrows        = 3
                       intertile    = 1
                       cprop
                       normal(noprint);
   inset n = "N" mean std / pos = nw;
run;
```

## APPENDIX F. SAS CODE FOR SAMPLE SIZE 500 , SCENARIO 2

```
/* Creating a Library(Folder) for all dataset related to this work */
LIBNAME Sim 'T:\ms\Simulation\Scenario2\SampleSize500';

 /* Importing the 6572 male dataset with missing data */
PROC IMPORT
      OUT= Sim.Datamale
      DATAFILE= "T:\ms\Simulation\datamales.csv"
      DBMS=CSV REPLACE;
      GETNAMES=YES;
      DATAROW=2;
RUN;

/* Proc Univariate to generate Mode for the 6 covariates and store them
Sim.Model1*/
ods output BasicMeasures=Sim.Mode1;
proc univariate data=Sim.Datamale ;
   var BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;
run;

/* Clean up code to just retain mode and delete all other unwanted
information */
data Sim.Mode (keep=VarName LocMeasure LocValue);
set Sim.Mode1;
where LocMeasure='Mode';
run;

/* Getting rid of Model1 table because the cleaner version of this is Model
*/
proc datasets library=Sim;
   delete Mode1;
run;

/* Ploting histogram to extract the proportions for BMI, CHF, Age, Eyes,
Falling, Alchohol_Consmp and levels from 6572 record dataset*/
/*Proc Gchart data=Sim.Datamale ;
      hbar BMIgrp /DISCRETE;
      hbar CHF /DISCRETE;
      hbar AgeGrp /DISCRETE;
      hbar EyesPastYear /DISCRETE;
      hbar Falls /DISCRETE;
      hbar LastDrank /DISCRETE;
run;*/

/* Using the proportion from the previous Proc Gchart to simulate Falling
data the relationship between
different levels { 1,2,3,4,5 } is maintained approximately like the
proportion in Datamale dataset */
data Sim.DataX1 (drop = i) ;
array prob [5] (0.01, 0.03, 0.07, 0.26, 0.63);
      call streaminit(1234);
      do i = 1 to 10000;
            Falls = rand("Table", of prob[*]);
      output;
```

```
end;
run;

/* Using the proportion from the previous Proc Gchart to simulate Eyes,
Alchohol, Age, Heart, BMI data
the proprotion with different levels are maintained approximately like the
proportion in Datamale dataset */
data Sim.DataX2 (drop = i) ;
      call streaminit(1234);
      do i = 1 to 10000;
            int = 1;
            EyesPastYear =rand("Bernoulli",0.85);
            LastDrank = rand("Table", 0.6,0.1,0.3);
            AgeGrp = rand("Table", 0.2025, 0.4594, 0.2799, 0.0582);
            CHF =rand("Bernoulli",0.95);
            BMIGrp = rand("Table", 0.012 , 0.18, 0.35, 0.458);
      output;
end;
run;

/* Combining the Sim.Datax1 which contains Sim_Data_X_Y values for Falling
and
Combining Sim.Datax2 which contains Sim_Data_X_Y values for Eyes,
Alchohol,Age, Heart, and BMI into Sim.Data_X*/
data Sim.Data_X (drop= prob1 prob2 prob3 prob4 prob5);
merge Sim.DataX1 Sim.DataX2;
run;

/* Deleting Datax1 and Datax2 because they are already merged into Data_X */
proc datasets library=Sim;
   delete DataX1 DataX2;
run;

/* Run Proc Logistic on Clean Male Dataset with not missing data and store
the parameter estimates in
Estimates dataset */
proc logistic data=Sim.Datamale  DESCENDING  outest=Sim.Estimates ;
Class
      AgeGrp(Ref='1') BMIgrp(Ref='2') CHF(Ref='0') EyesPastYear(Ref='0')
Falls(Ref='0') LastDrank(Ref='1')
      / param=ref;
      model Diabetes = BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;
run;

/* Transform the Data_X dataset wich contains the Sim_Data_X_Y values for
Fall, Age, Alchohol, BMI into a
1 0 1, 1 1 1, 0 1 1, 0 0 0......etc matrix like this and store the values in
Transform10100 dataset*/
data Sim.Transform10100 (drop= BMIgrp CHF AgeGrp EyesPastYear  Falls
LastDrank) ;
set Sim.Data_X ;

      if BMIGrp=1 then
      do;
      BMIgrp1=1;
      BMIgrp3=0;
      BMIgrp4=0;
```

```
  end;
  else if BMIGrp=2 then
  do;
  BMIgrp1=0;
  BMIgrp3=0;
  BMIgrp4=0;
  end;
  else if BMIGrp=3 then
  do;
  BMIgrp1=0;
  BMIgrp3=1;
  BMIgrp4=0;
  end;
  else if BMIGrp=4 then
  do;
  BMIgrp1=0;
  BMIgrp3=0;
  BMIgrp4=1;
  end;


if LastDrank=1 then
  do;
  LastDrank2=0;
  LastDrank3=0;
  end;
  if LastDrank=2 then
  do;
  LastDrank2=1;
  LastDrank3=0;
  end;
  else if LastDrank=3 then
  do;
  LastDrank2=0;
  LastDrank3=1;
  end;


  if Falls=1 then
  do;
  Falls2=0;
  Falls3=0;
  Falls4=0;
  Falls5=0;
  end;
  else if Falls=2 then
  do;
  Falls2=1;
  Falls3=0;
  Falls4=0;
  Falls5=0;
  end;
  else if Falls=3 then
  do;
  Falls2=0;
  Falls3=1;
  Falls4=0;
```

```
Falls5=0;
end;
else if Falls=4 then
do;
Falls2=0;
Falls3=0;
Falls4=1;
Falls5=0;
end;
else if Falls=5 then
do;
Falls2=0;
Falls3=0;
Falls4=0;
Falls5=1;
end;

if EyesPastYear=1 then
do;
EyesPastYear1=1;
end;
else if EyesPastYear=0 then
do;
EyesPastYear1=0;
end;

if AgeGrp=2 then
do;
AgeGrp2=1;
AgeGrp3=0;
AgeGrp4=0;
end;
else if AgeGrp=3 then
do;
AgeGrp2=0;
AgeGrp3=1;
AgeGrp4=0;
end;
else if AgeGrp=4 then
do;
AgeGrp2=0;
AgeGrp3=0;
AgeGrp4=1;
end;
else if AgeGrp=1 then
do;
AgeGrp2=0;
AgeGrp3=0;
AgeGrp4=0;
end;


if CHF=1 then
do;
CHF1=1;
end;
else if CHF=0 then
```

```
      do;
      CHF1=0;
      end;


run;


/* Use Proc IML perform some array operations */
proc iml ;

      use sim.Estimates; /* Read all the parameter estimates stored in
Estimates dataset into an array z2 */
      read all ;
            z2 = Intercept || BMIgrp1 || BMIgrp3 || BMIgrp4 ||  LastDrank2 ||
LastDrank4 || Falls1 || Falls2 || Falls3 || Falls4 || EyesPastYear1 ||
AgeGrp2 || AgeGrp3 ||AgeGrp4 ||CHF1 ;

      use sim.Transform10100; /* Read all the Transform10100 dataset that
contains transformed Falls , BMI, Alchohol variables and store them in array
z4 */
      read all ;
            z4 =  int || BMIgrp1 || BMIgrp3 || BMIgrp4 || LastDrank2 ||
LastDrank3 || Falls2 ||  Falls3 || Falls4 || Falls5 || EyesPastYear1 ||
AgeGrp2 || AgeGrp3 || AgeGrp4 || CHF1 ;
            /* scoring on the Sim_Data_X_Y dataset */
            z5 = z4*t(z2);
            p = exp(z5)/(1+exp(z5));
            /* Generating data for Diabetes based on Simulation ....not very
strongly related to previous steps but just for further analysis */
            Diabetes = rand("Bernoulli",p);
            z6= p || Diabetes ; /* Merge two arrays p and Diabetes into one
z6 */

      /* putting these arrays back into a dataset Test 6 */
      create Sim.Test6 from z6;
      append from z6;
      close Sim.Test6;

quit;


/* Final Sim_Data_X_Y Data , contains both p and diabetes based on Bernouli
simuation*/
data Sim.Sim_Data_X_Y (rename=(Col1=p Col2=Diab_Bernoulli_Sim ) );
merge Sim.Data_X Sim.Test6;
if Col1<=0.5 then Diab_p=0;
else if Col1 > 0.5 then Diab_p=1;
id_Sim=_N_;
run;

/* Deleting Test6 because they are already merged into Sim_Data_X_Y */
proc datasets library=Sim;
   delete Test6 Data_X;
run;

proc logistic data=Sim.Sim_Data_X_Y  DESCENDING ;
```

```
Class
      AgeGrp(Ref='1') BMIgrp(Ref='2') CHF(Ref='0') EyesPastYear(Ref='0')
Falls(Ref='1') LastDrank(Ref='1') / param=ref;
      model Diab_Bernoulli_Sim = BMIgrp CHF AgeGrp EyesPastYear Falls
LastDrank ;
run;

/*---------------------Start------Generating Missing Data--------------------
----------*/

data Sim.DataMiss1 ;
      call streaminit(1234);
      do id_miss = 1 to 10000;
            EyesPastYea       =     rand("Bernoulli",0.1);
            LastDran          =     rand("Bernoulli",0.1);
            AgeGp                   =     rand("Bernoulli",0.1);
            CH                      =     rand("Bernoulli",0.1);
            BMIGp                   =     rand("Bernoulli",0.1);
            Fall              =     rand("Bernoulli",0.1);
      output;
end;
run;

data Sim.Sim_Miss_Data (keep= BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank
Diab_Bernoulli_Sim);
set Sim.DataMiss1  ; set Sim.Sim_Data_X_Y ;
if id_miss=id_Sim and EyesPastYea=1 then EyesPastYear=.;
if id_miss=id_Sim and AgeGp=1             then AgeGrp=.;
if id_miss=id_Sim and LastDran=1    then LastDrank=.;
if id_miss=id_Sim and CH=1                then CHF=.;
if id_miss=id_Sim and BMIGp=1             then BMIgrp=.;
if id_miss=id_Sim and Fall=1        then Falls=.;
run;

data Sim.Sim_Impute_Mode;
set Sim.Sim_Miss_Data;
if EyesPastYear=. then  EyesPastYear=1;
if AgeGrp=.             then  AgeGrp=2;
if LastDrank=.          then  LastDrank=4;
if BMIgrp=.             then  BMIgrp=3 ;
if Falls=.              then  Falls=0;
if CHF=.                then  CHF=0;
run;



proc datasets library=Sim;
   delete DataMiss1 Transform10100;
run;

%macro sim;

%let iterations = 1000;

%do i = 1 %to &iterations;
```

```
proc surveyselect data=Sim.Sim_Impute_Mode
  method = SRS
  reps = 1
  seed = 0
  N = 500
  out = Sim.Sample_Data;
run;

ods output  Classification=Sim.ctable1;;
proc logistic data=Sim.Sample_Data  DESCENDING ;
Class
      AgeGrp(Ref='1') BMIgrp(Ref='2') CHF(Ref='0') EyesPastYear(Ref='0')
Falls(Ref='1') LastDrank(Ref='1') / param=ref;
      model Diab_Bernoulli_Sim = BMIgrp CHF AgeGrp EyesPastYear Falls
LastDrank / ctable
      pprob=0.1 0.2 0.3 0.4 0.45 0.5 0.6 0.7 0.8 0.9;
run;


data Sim.ctable2;
set Sim.ctable1;
total=Sensitivity + Specificity;
run;

proc sort data=Sim.ctable2;
    by total ProbLevel;
run;

data Sim.ctable3;
   set Sim.ctable2;
   if _n_ = 10 then output;
   keep Correct;
run;

proc append base=Sim.LR_500_Classification data=Sim.ctable3 force;
                  run;

%end;
run;
%mend sim;

%sim;

proc datasets library=Sim;
   delete Ctable1 Ctable2 Ctable3;
run;

%macro sim;

%let iterations = 1000;

%do i = 1 %to &iterations;

                        proc surveyselect data=Sim.Sim_impute_mode
                              method = SRS
                              reps = 1
                              N = 500
```

```
                              out = Sim.ANN_Imputed_Data;
                  run;

                  proc dmdb batch data= Sim.ANN_Imputed_Data
out=Sim.ANN_DM_Smp10 dmdbcat=Sim.ANN_CAT_Smp10;
                                          CLASS    Diab_Bernoulli_Sim (DESC)
BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;
                                          TARGET   Diab_Bernoulli_Sim;

                        run;

                        proc neural data= Sim.ANN_Imputed_Data
dmdbcat=Sim.ANN_CAT_Smp10;
                                       NETOPTIONS DECAY=0 INVALIDTARGET=OMITCASE
OBJECT=LIKE;
                                          INPUT BMIgrp CHF AgeGrp
EyesPastYear Falls LastDrank /level= nominal id=inp STD=NO ;
                                          TARGET      Diab_Bernoulli_Sim /
level=nominal id=trg STD=NO ACT=SOFTMAX COMBINE=LINEAR ERROR=MBERNOULLI;
                                          ARCHI MLP   HIDDEN = 10 ;
                                          TRAIN TECHNIQUE= LEVMAR
MAXITER=1000 ;
                                          INITIAL;
                                          SCORE data= Sim.ANN_Imputed_Data
out=Sim.ANN_OUT_Smp1 outfit=Sim.ANN_FIT_Smp1 role=TRAIN;;


                        run;

                        data Sim.temp;
                              set Sim.ANN_FIT_Smp1;
                              concordant = round((1 - _misc_)*100,.01);
                              if _n_ = 2 then output;
                              keep concordant;
                  run;

                              proc append base=Sim.ann_concordant data=Sim.temp
force;
                  run;

%end;
run;
%mend sim;

%sim;

data Sim.ANN_500_Classification(rename=(concordant=Correct));
    set Sim.ann_concordant;
run;



%macro svm;

%let iterations = 1000;

%do i = 1 %to &iterations;
```

```
proc surveyselect data=Sim.Sim_impute_mode
                                method = SRS
                                reps = 1
                                N = 500
                                out = Sim.SVM_Imputed_Data;
                  run;



proc dmdb
      batch
      data= Sim.SVM_Imputed_Data
      out=Sim.SVM_DM_Smp10
      dmdbcat=Sim.SVM_CAT_Smp10;

      CLASS
      Diab_Bernoulli_Sim (DESC) ;

      VAR
      BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;

      TARGET
      Diab_Bernoulli_Sim;

run;

PROC SVM DATA=Sim.SVM_DM_Smp10 DMDBCAT=Sim.SVM_CAT_Smp10 CV=SPLIT FOLD=10
METHOD=LSVM KERNEL=POLYNOM K_PAR =5
OUT=Sim.SVM_OUT OUTEST= Sim.SVM_EST OUTFIT=Sim.SVM_FIT
TUN=GRID;
VAR BMIgrp CHF AgeGrp EyesPastYear Falls LastDrank ;

TARGET Diab_Bernoulli_Sim;

C 0.1 TO 1.0 by 0.1;
RUN;

                        data Sim.svm_temp;
                                set Sim.SVM_FIT;
                                concordant = round((1 -
((_ACCU_)/(_NTRAIN_)))*100,0.01);
                                if _n_ = 1 then output;
                                keep concordant;
                  run;

                        proc append
                                              base=Sim.svm_500_Classification
                                              data=Sim.svm_temp force;
                  run;

%end;
run;
%mend svm;

%svm;
```

```
data Sim.ann_concordant(rename=(concordant=ann_correct));
    set Sim.Ann_500_Classification;
run;


data Sim.Svm_Classification_500_1000(rename=(concordant=correct));
set Sim.Svm_500_Classification;
model='SVM';
run;

data sim.LR_Classification_500_1000 ;
set sim.LR_500_Classification;
model='LR';
run;

data Sim.ann_Classification_500_1000(rename=(ann_correct=correct));
set Sim.Ann_500_Classification;
model='NN';
run;

data Classification_500_1000;
   merge sim.LR_Classification_500_1000 Sim.ann_Classification_500_1000
Sim.Svm_Classification_500_1000;
   by model;
run;

proc univariate data=Classification_500_1000 noprint;
   class model;
   histogram Correct / nrows        = 3
                       intertile    = 1
                       cprop
                       normal(noprint);
   inset n = "N" mean std / pos = nw;
run;
```