CHRONIC KIDNEY DISEASE PREDICTION USING CLASSIFICATION ALGORITHMS- NAIVE BAYES, SVM, ARTIFICIAL NEURAL NETWORKS

SAN JOSÉ STATE UNIVERSITY

Submitted To : Prof. Chandrashekhar Vuppalapati

Team Members	**SJSU ID**
Mahitha Byreddy	010732787
Charmili Narra	010641891
Shreya Prabhu	010374507

Table of Contents

PROJECT DESCRIPTION

In the healthcare industry, the data mining is essentially used for prediction of diseases and their analysis so as to determine the impact factors and causes of the disease. A myriad of data mining techniques are used for predicting diseases namely classification, association rules, clustering, regression, summarizations. The fundamental objective of this project is to predict and analyze the causes and impact factors of chronic kidney diseases using some classification algorithms like Artificial Neural Network, Support Vector Machine and Naïve Bayes. Our project is primarily focused on finding the finest classification algorithm authorized using performance factors like the execution time and classification accuracy. Based on the experimental results the performance of the artificial neural networks is recognized to be better than the remaining classification algorithms.

Chronic Kidney Disease (CKD) can result in mortality as kidneys are most important in maintaining stability in the physiological processes. CKD can affect the entire body system. Hence, the prediction of CKD in its early stage is very important. By detecting CKD, early treatment can be given to the patients and disease progression could be monitored. Though there were tests to identify CKD, not everyone will be tested. The ubiquity of CKD is rising around the world. More than 20 million patients in the USA have CKD according to the recent statistics. So we can say that there is an immediate need of a screening tool which will predict the disease at its early stage.

The purpose of our project is to develop a model which can be used to identify patients who are prone to risk by using simple patient data. In our project we tried to evaluate the CKD based on different variables which the patients can easily get from their physician. Our work predominantly focused on the prediction and analysis of chronic kidney diseases using classification techniques of data mining. To attain our goal of the project we have focused on the data taken from the survey conducted by National Center for Health Statistics. We combined all the possible risk factors and calculated the overall risk faced by the patients. For this, we have used different classifiers such as Artificial Neural Network (ANN), Naïve Bayes and Support Vector Machine. The performance of all these classifiers is evaluated based on certain measures.

Evaluation Metrics:

The quality of the solution we developed is accessed by using the below metrics:

1. Accuracy
2. Sensitivity
3. Specificity

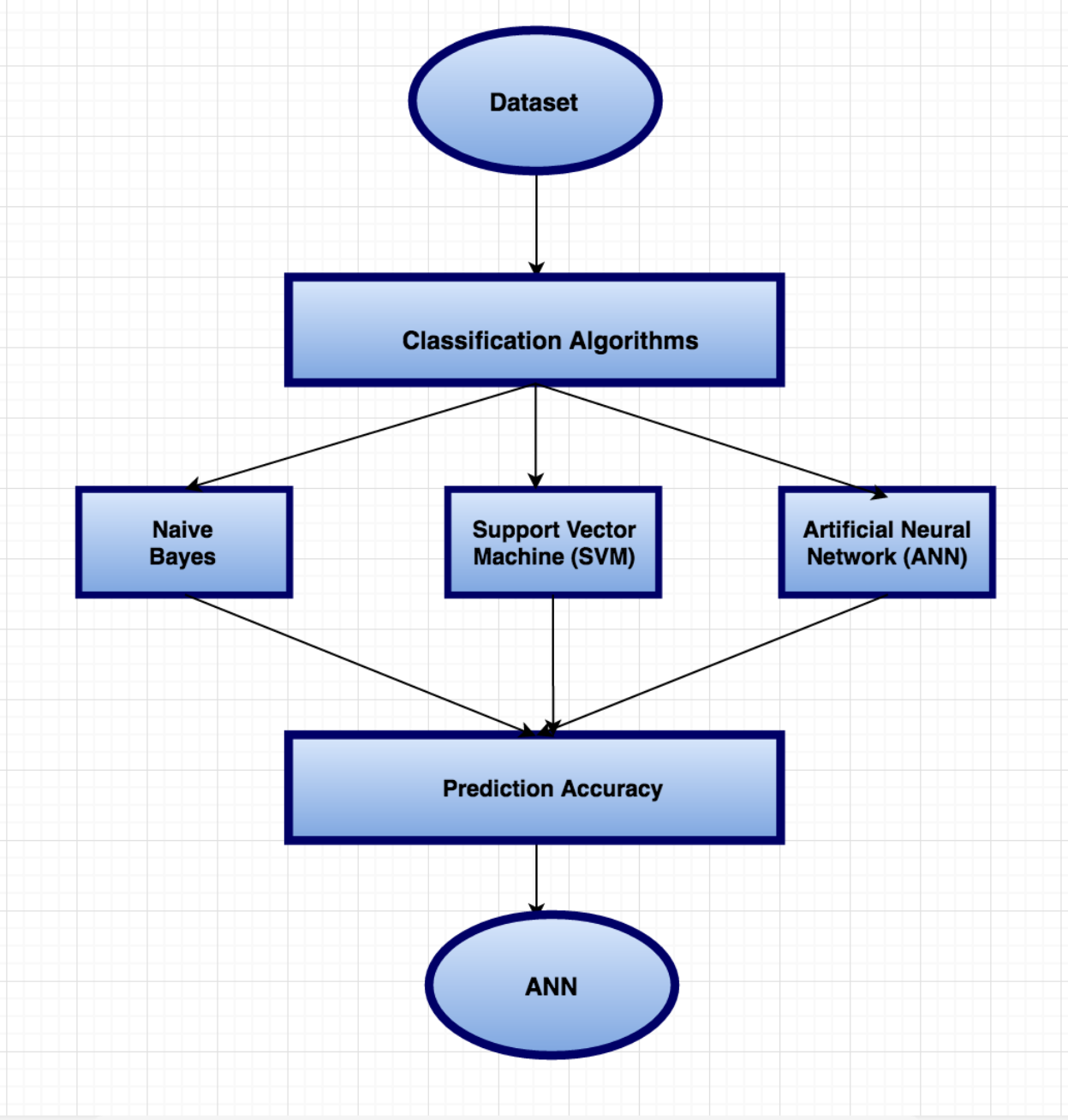


Figure 1: Project Overview Architecture Diagram

REQUIREMENTS

The Causes of CKD3:

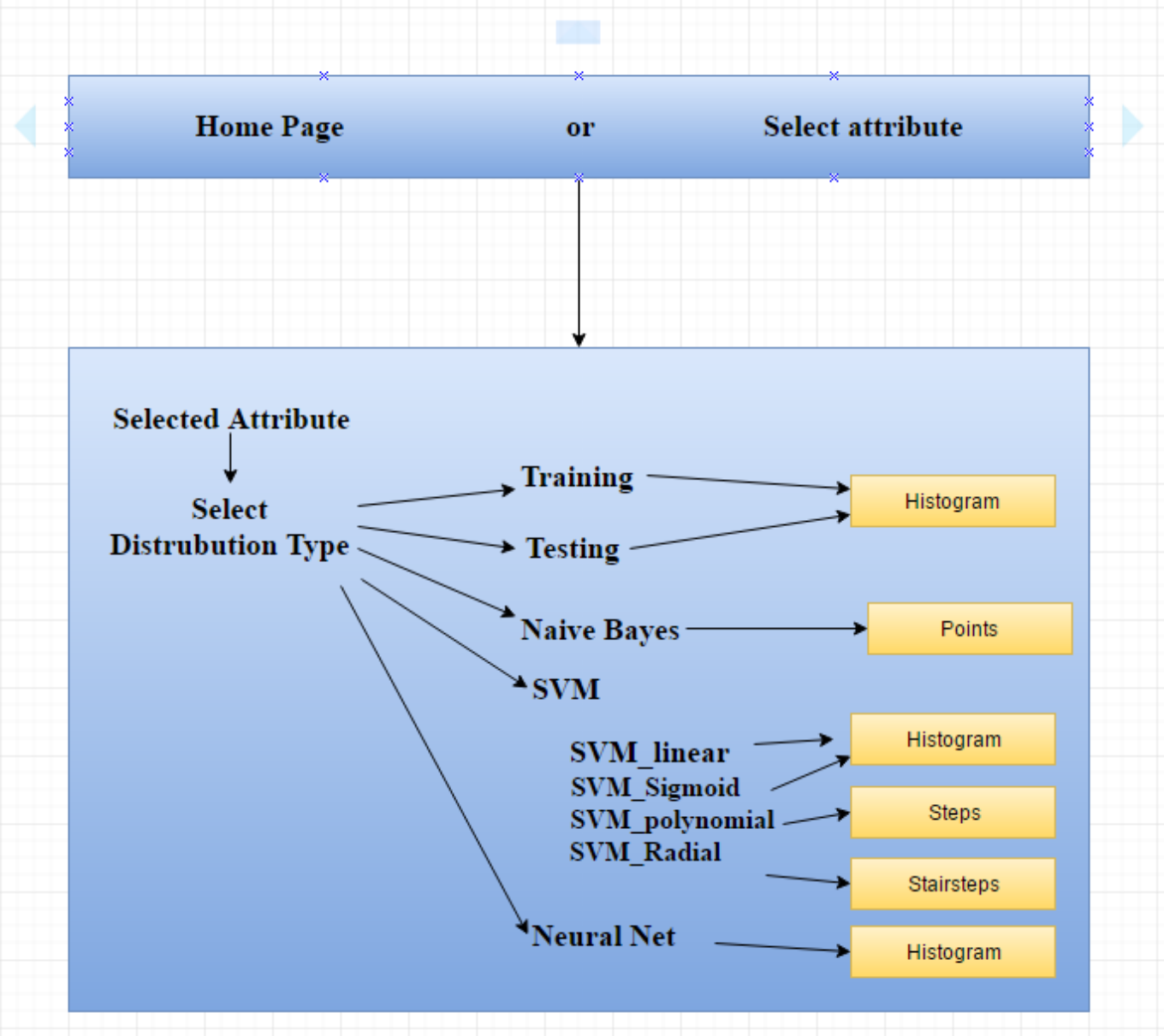
The two main causes of chronic kidney disease are diabetes and high blood pressure, which are responsible for up to two-thirds of the cases. Diabetes happens when your blood sugar is too high, causing damage to many organs in your body, including the kidneys and heart, as well as blood vessels, nerves, and eyes. High blood pressure, or hypertension, occurs when the pressure of your blood against the walls of your blood vessels increases. If uncontrolled, or poorly

controlled, high blood pressure can be a leading cause of heart attacks, strokes, and chronic kidney disease. Also, chronic kidney disease can cause high blood pressure.
Other conditions that affect the kidneys are:
• Glomerulonephritis, a group of diseases that cause inflammation and damage to the kidney's filtering units. These disorders are the third most common type of kidney disease.
• Inherited diseases, such as polycystic kidney disease, which causes large cysts to form in the kidneys and damage the surrounding tissue.
• Malformations that occur as a baby develops in its mother's womb. For example, a narrowing may occur that prevents normal outflow of urine and causes urine to flow back up to the kidney. This causes infections and may damage the kidneys.
• Lupus and other diseases that affect the body's immune system.
• Obstructions caused by problems like kidney stones, tumors, or an enlarged prostate gland in men.
• Repeated urinary infections.

While anyone at any age can develop chronic kidney disease (CKD), a number of risk factors have been identified that may lead to possible problems with your kidneys. These include:
• Diabetes. Diabetes is the leading cause of CKD. If you have diabetes, talk with your doctor about how to keep your blood glucose as close to normal as possible to ensure your diabetes is under control.
• Hypertension. Hypertension, also called high blood pressure, is the second-highest cause of CKD. Keep your blood pressure under control. A number of effective medications are available to help you with this task. Your doctor will help you to determine which medication is right for you.
• Cardiovascular disease. In addition to hypertension, other diseases of the heart and blood vessels may increase your risk for kidney disease. People who have had heart attacks or strokes, congestive heart failure, coronary artery disease, or peripheral vascular disease need to be monitored carefully for kidney problems.
• Family history of kidney disease. Some kidney diseases are genetic. People with a mother, father, brother, or sister who has had a kidney disease are more likely to develop problems with their kidneys.
• Age. People 60 years and older are at a higher risk for developing CKD.
• Race. People belonging to certain ethnic groups, such as First Nations (Canadian aboriginal peoples) and Pacific Islanders, are at a higher risk for developing this disease.

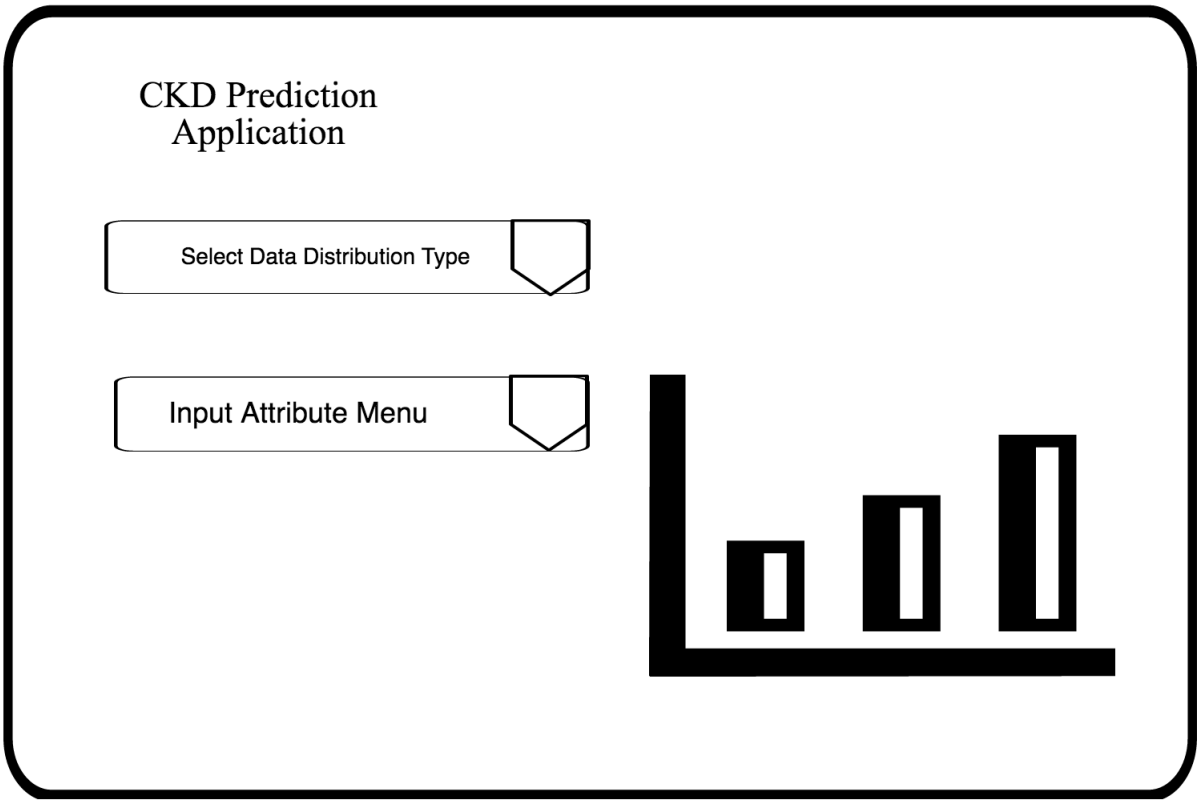
UI DESIGN PRINCIPLES

Storyboard



Wireframes

Main Application Page



HIGH LEVEL ARCHITECTURE DESIGN

In our architecture diagram we are using a Shiny Server and Shiny tool as the front end. On the server side, we are using R Studio which has built in function modules for Naive Bayes, Support Vector Machine, Artificial Neural Network Classifiers. We are using Naive Bayes classifier in R Studio and trained it using its built classifier. We have created UI using shiny and we provide with data visualization for each contributing impact factor in chronic kidney disease by the means of interactive graphs.

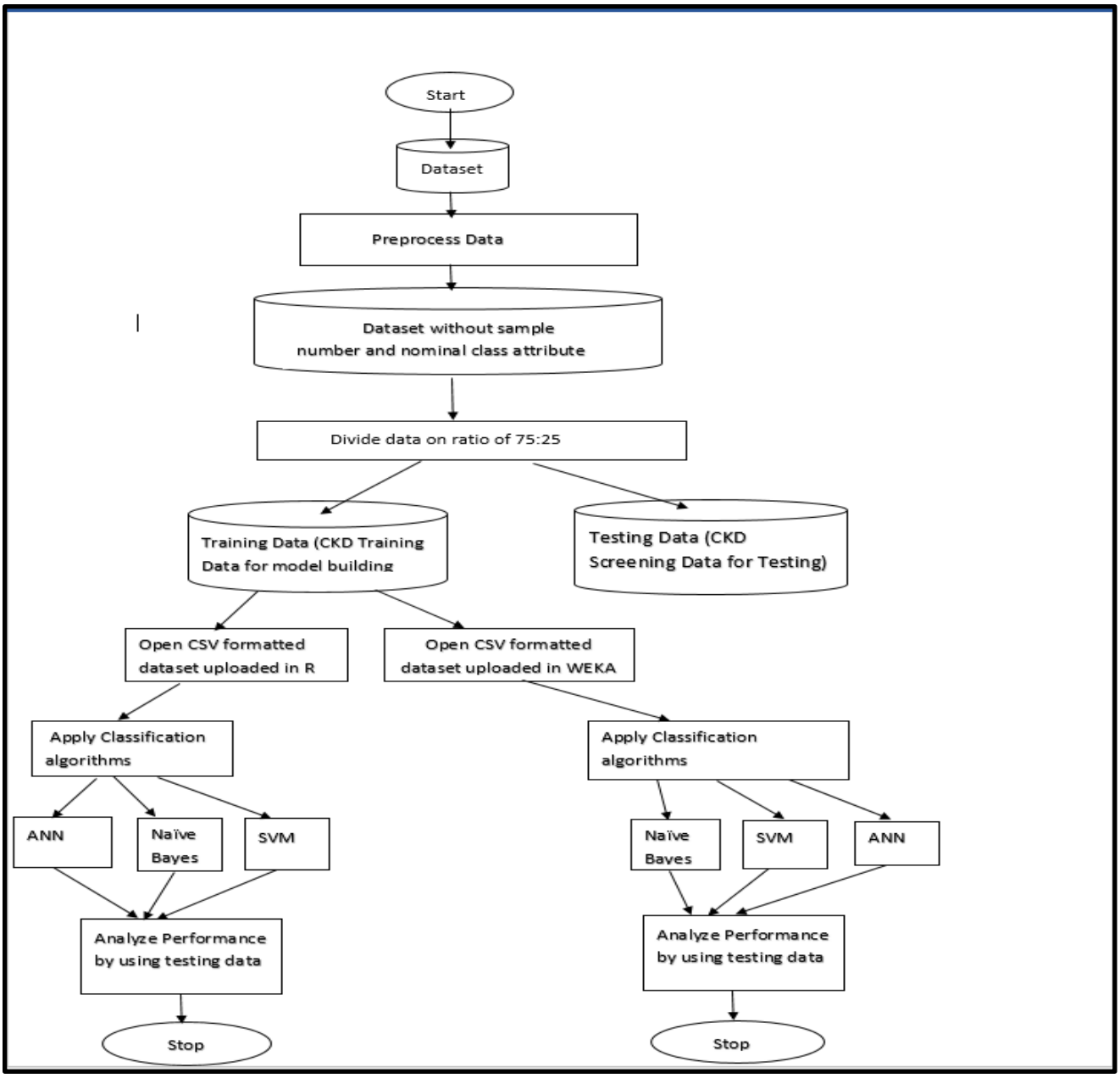


Figure 2: High Level Architecture Diagram

DATASETS & DATA PATTERNS

Since 1975, the National Center for Health Statistics of the Centers for Disease Control and Prevention has conducted nationwide surveys of U.S. adults. Using trained personnel, the center collected a wide variety of demographic and health information using direct interviews, examinations, and blood samples. In this project, the dataset was populated using the data from National Center for Health Statistics of the Centers for Disease Control "http://www.cdc.gov/nchs/ ".

Data from National Center for Health Statistics of the Centers for Disease Control was in csv format: the issue with the data was that records were in row format. However, to access the data in Weka or RStudio we required the data to be in a column format so we set the byrow value to TRUE so that the data gets organized according to rows.

We concentrated data given for chronic kidney disease patients 20 years of age or older taken from the surveys; we had around 10219 records for different patients. Pre-processing was performed to remove duplicate records and we found for few missing values, we have populated the dataset with a constant zero value for such records.
The sample subjects were randomly divided into two pools: a 6,000-case training set and a 2,819-case validation sample. A test for CKD was administered to everyone in the study population. The variable of interest is CDK, a 0/1 dummy variable indicating whether or not the subject had CKD.Notice that variables in columns A through J are demographic in nature, K through V were collected during the physical exam, and W through AH are based, in part, on self-reported health histories.
Resultant dataset consists of 8219 records.
Now for these reviews we have extracted the sentiments and calculated the gap and derived the class variables, below chart display the distribution of instances against the class.

Training And Test Data

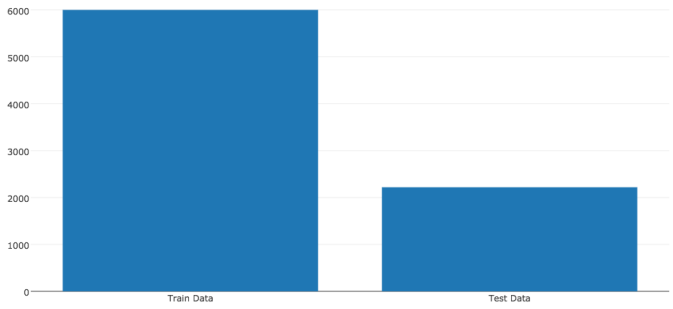


Figure 3: Graph showing bifurcation of model data between Training and Test Data

After data preprocessing & feature extraction, we ended up selecting the below features for the Training our machine learning models.

Table 1: Attributes Established for our project from the dataset

Column	Variable	Definition
A	ID	Identification Number
B	Age	Age(years)
C	Female	1 if female
D	Racegrp	Self reported race/ ethnic group(White, black, hispanic, other)
E	Educ	1 if more than high school
F	Unmarried	1 if unmarried
G	Income	1 if household income is above the median
H	CareSource	Self reported source of medical care(Dr./HMO, clinic, noplace, other)
I	Insured	1 if covered by health insurance
J	Weight	Weight(kg)
K	Height	Height(cm)
L	BMI	Body Mass Index(kg/m2)
M	Obese	1 if BMI is greater than 30kg/m2
N	Waist	Waist circumference(cm)
O	SBP	Systolic blood pressure(max)
P	DBP	Diastolic blood pressure(min)
Q	HDL	(mg/dL)the “good” cholesterol
R	LDL	(mg/dL) the “bad” cholesterol
S	Total chol	(mg/dL) the sum of good and bad cholesterol

T	Dyslipidemia	Too high LDL or too low HDL
U	PVD	Peripheral vascular disease reflected by reduced SBP at the leg relative to the arm.
V	Activity	Mostly sit (1); stand or walk a lot (2); lift light loads or climb stairs often (3); heavy work and heavy loads (4).
W	Poor Vision	Self-reported poor vision
X	Smoker	Smoked at least 100 cigarettes.
Y	HyperTension	The presence of at least one of four indicators of high blood pressure.
Z	Fam Hypertension	Family history of hypertension (high blood pressure)
AA	Diabetes	Self-reported physician diagnosed or lab test result
AB	Fam Diabetes	Family history of diabetes
AC	Stroke	Self-reported response to "Has a doctor ever told you that you had a stroke?"
AD	CVD	Response to "Has a doctor ever told you that you had angina pectoris, myocardial infarction, or stroke?"
AE	Fam CVD	Family history of cardiovascular disease
AF	CHF	Self-reported response to "Has a doctor ever told you that you had congestive heart failure?"

AG	Anaemia	Treatment for anemia received in past 3 months or hemoglobin at exam lower than 11g/dL
AH	CKD	Chronic kidney disease as indicated by measured serum creatinine.

Data Flow Diagrams & Architecture

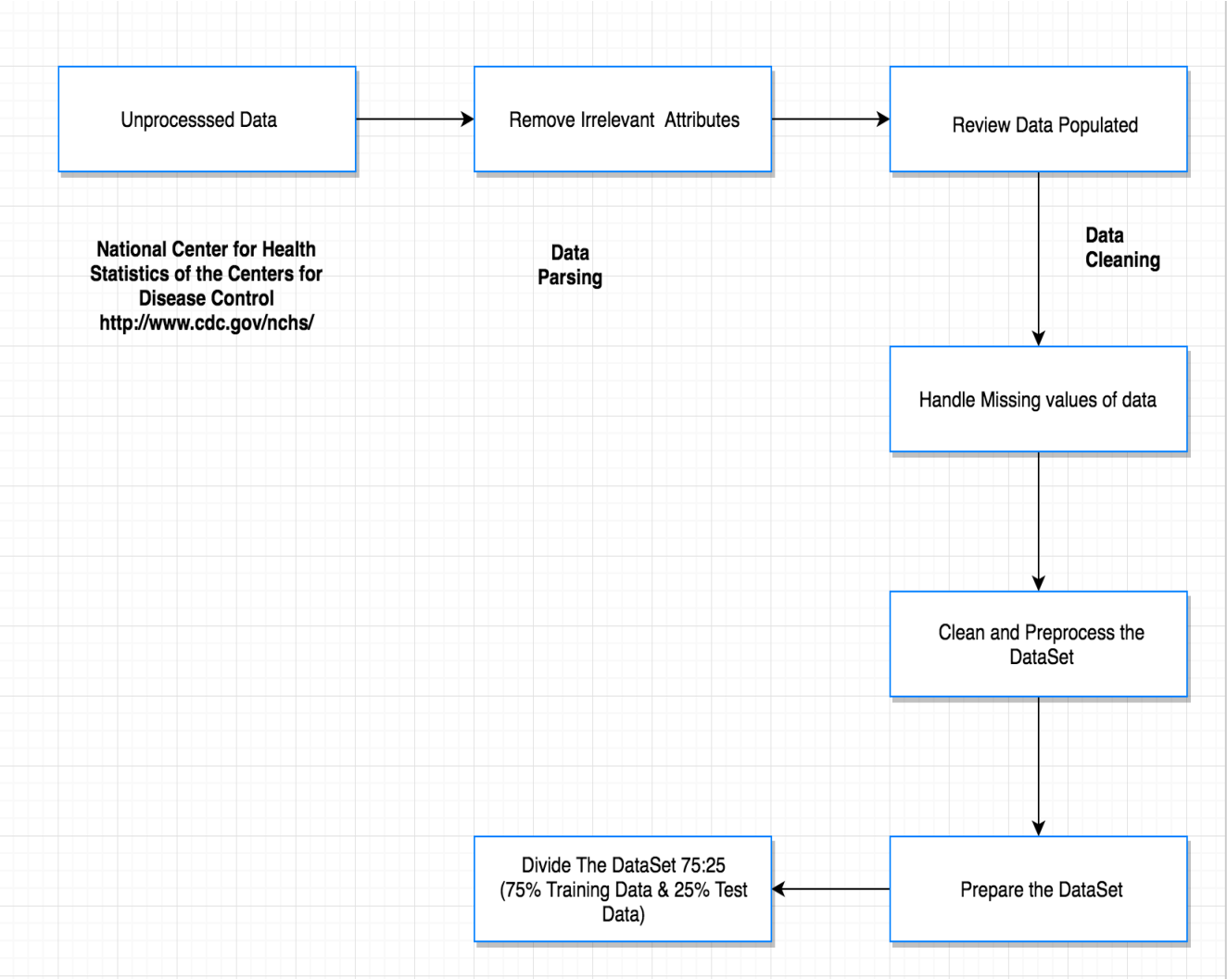


Figure 4: Steps Data Modeling Process Diagram

INFOGRAPHICS

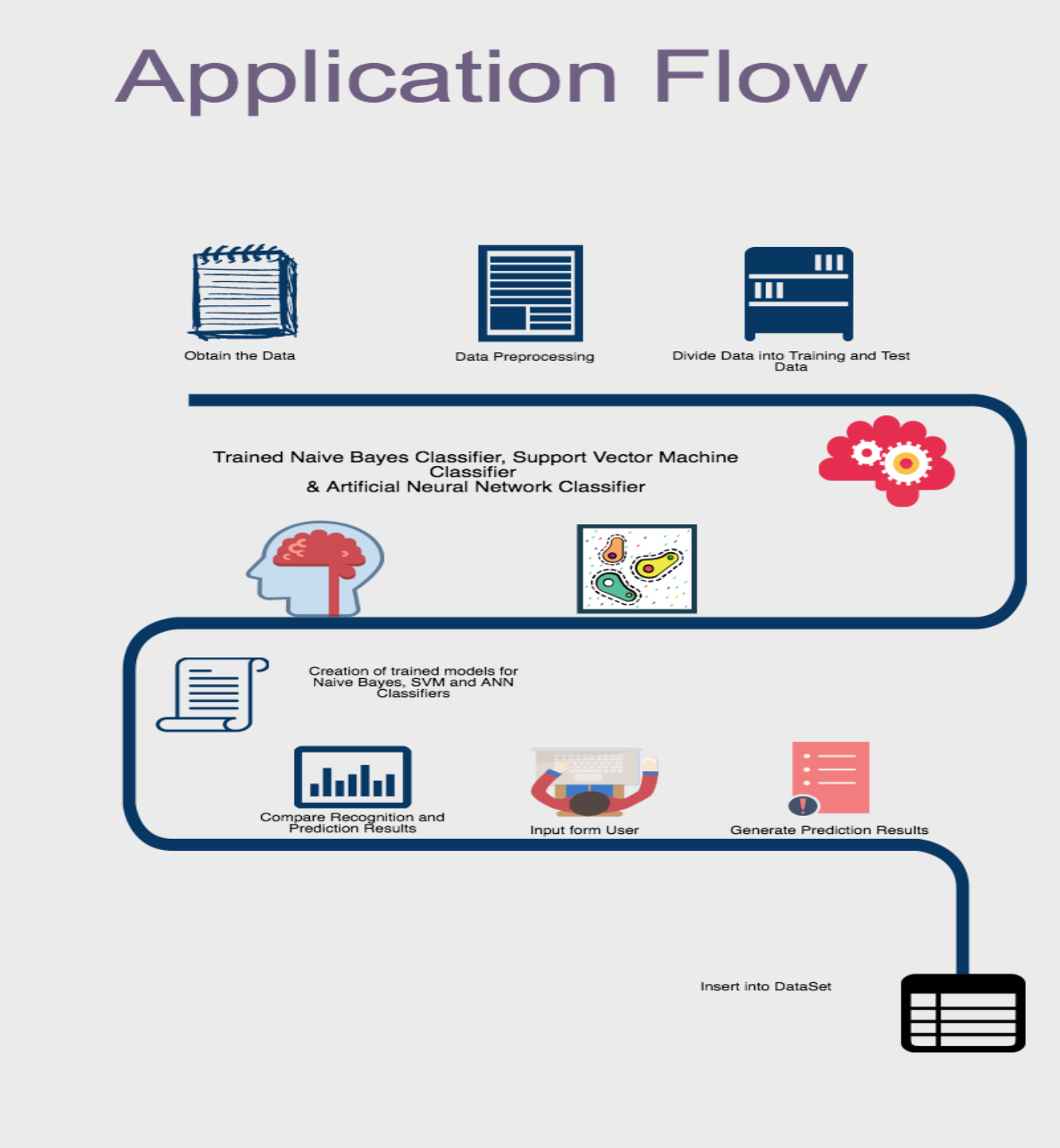


Figure 5: Infographics for our Project

DATA MINING PRINCIPLES AND ALGORITHMS

We have used the following classifiers in our project:

Naïve Bayes: Naïve Bayes is a probabilistic classifier that relies on the Bayes theorem to find the probability and uses this calculated probability to make its decisions. This model uses the frequency of impact factors.

Support Vector Machine: SVM creates a multi-dimensional hyperplane to separate the data points. A separation is acceptable if there is maximum distance between the support vectors.

Artificial Neural Network : ANN involves the creation of extensive parallel networks and these networks will be trained to elucidate specific problems.

Naïve Bayes:

Naïve Bayes Classification utilizes the Bayesian theorem. It is suitable for the when the dimensionality of the input data is high. An advantage of this classifier is that it doesn't require huge amounts of training data but it only approximates the parameters necessary for classification by taking a small amount of training data. The probability that a specific feature in the data appears as a member part in the set of probabilities and is derived by calculating the frequency of each feature value within a class of a training data set. The training dataset is a subset and it is used to train a classifier algorithm by using known values to predict the unknown values.

For naive bayes, the evaluation on training set summary is as follows:

Correctly classified instances	94.75%
Incorrectly classified instances	5.25%
Kappa statistic	0.891
Mean absolute error	0.049
Root mean squared error	0.210
Relative absolute error	10.6129%
Root relative squared error	43.54%

Table 2: This table gives the results of the training set on the application of Naive Bayes classifier from Weka

Results for naive Bayes in RStudio:
Code Snippet:

>set.seed(0305)
We have to set the seed every time you want to get a reproducible random result. Any random number can be taken inside the seed.

>index <- createDataPartition(ModelDataFactored$CKD, p = 0.75, list = FALSE)
The createDataPartition is present in caret package. It takes parameters out of which the ModelDataFactored will be the vector of outcomes and the 'p' value will be the percentage of data that goes to the training part and ' list' specifies whether the results should be in alist format or not.

>training <- ModelDataFactored[index,]
For instance, 3102 observations of 33 variables

>testing <- ModelDataFactored[-index,]
For instance, 1033 observations of 33 variables

>nb <- naiveBayes(CKD~., data=training,laplace=1)
The naivebayes is present in e1071 package. This takes as parameters, the first one being CKD~. which specifies that the class attribute or formula object on the left side taken with the explanatory variables where '.' specifies that all the variables are taken into account.The 'data' specifies the data.frame or the table and 'laplace' specifies positive double controlling laplace smoothing. The nb object gives a list of four on the console. The naivebayes on the whole computes conditional-a posteriori probabilities of categorical class variables given independent predictor variables using bayes rule.

>predVector<-predict(nb, newdata=testing)
The predict method will predict the given training set or the data frame or the table against the specified testing set. The predict takes parameters as 'nb' the object of class which has the previous naivebayes result and 'newdata' specifies optional data.frame in which to look for variables with which to predict.

>confusionMatrix(predVector, testing$CKD, positive="No")
The confusionMatrix is present in the caret package. It calculates the cross tabulation of observed and predicted classes with associated statistics. It takes parameters like 'predvector' which is the data which is factor of predicted classes or an object of the class table. The 'testing$CKD' is a factor of classes to be used as true results. The 'positive' field is "yes" if there are only two levels for data, first level will be used as positive level.

>print("Naive Bayes results:")
>print("Accuracy: 86%")
>print("Sensitivity: 87%")
>print("Specificity: 62%")
The above print statements are used to get result when the test file is executed.

>TestDataNB <- TestDataFactored
The TestDataFactored is present in TestFactorize.R file which will get 1907 observations of the 32 variables.

>TestDataNB$CKD <- predict(nb, newdata=TestDataNB)
The predict method will predict the given training set or the data frame or the table against the specified testing set. The predict takes parameters as 'nb' the object of class which has the previous naivebayes result and 'newdata' specifies optional data.frame in which to look for variables with which to predict.

>write.csv(TestDataNB, file = "NaiveBayes.csv", row.names = F, quote = F)
This method writes the data into the .csv files. The 'TestDataNB' decides the object to be written to the file. The 'file' specifies the File to which the object has to be copied. The 'row.names' specifies whether row names should be written along with TestDataNB. The 'quote' specifies the 'T' when the factor or character columns get quoted.

>TestDataNB <- NULL
This specifies to set the TestDataNB object to NULL

The output when the above code snippet gets executed is as follows:
Accuracy : 0.8742
95% CI : (0.8524, 0.8938)
No Information Rate : 0.9361
P-Value [Acc > NIR] : 1
Kappa : 0.3336
Mcnemar's Test P-Value : 1.21e-12
Sensitivity : 0.8904
Specificity : 0.6364
Pos Pred Value : 0.9729
Neg Pred Value : 0.2838
Prevalence : 0.9361
Detection Rate : 0.8335
Detection Prevalence : 0.8567
Balanced Accuracy : 0.7634
'Positive' Class : No

Accuracy:86%
Sensitivity:87%
Specificity: 62%

Support Vector Machine:

The Support Vector Machine produces input – output mapping functions from a set of labeled training data. It can be used to predict categorical variable or a continuous variable. It is a best suitable technique when we have large number of predictor variables (dependent variables) as in the case of our project. The SVM algorithm performs a classification by constructing a multi-dimensional hyperplane that separates two classes.This algorithm accomplishes high discriminative power by utilizing unique nonlinear capacities or functions called Kernels to change the input space into a multidimensional space.

SVM model attempts to locate the best hyper plane that augments the margin between categories while adjusting the tradeoff of conceivably over fitting the data. The narrower the margin between the support vectors, the more accurate the model will be.

This technique has already used to improve methods for detecting diseases in clinical settings. Also, SVM has exhibited superior in taking care of order issues

Correctly classified instances	98.75%
Incorrectly classified instances	1.25%
Kappa statistic	0.973
Root relative squared error	23.094%
Root mean squared error	0.111
Mean absolute error	0.012
Relative absolute error	2.665%

Table 3: This table gives the results of the training set on the application of SVM classifier from Weka

Results for SVM in RStudio:

Code Snippet:

```
>set.seed(0305)
```

We have to set the seed every time you want to get a reproducible random result. Any random number can be taken inside the seed.

```
>index <- createDataPartition(ModelDataFactored$CKD, p = 0.75, list = FALSE)
```

The createDataPartition is present in caret package. It takes parameters out of which the ModelDataFactored will be the vector of outcomes and the ‘p’ value will be the percentage of data that goes to the training part and ‘ list’ specifies whether the results should be in alist format or not.

>*training <- ModelDataFactored[index,]*
For instance, 3102 observations are created.

>*testing <- ModelDataFactored[-index,]*
For instance, 1033 observations were created.

##LINEAR KERNEL
>*obj_linear <- best.tune(svm, CKD~., data = training, kernel = "linear")*
This generic function tunes hyperparameters of statistical methods using a grid search over supplied parameter ranges. Here ‘svm’ specifies the function to be tuned and CKD~. gives the formula or a matrix of predictors but here CKD value is validated against the all other attributes. The data specifies if a formula interface is used. The ‘kernel’ specifies the model for the svm which can be either linear or sigmoid or radial basis function.

>*obj_linear*
Displays the object obj_linear

>*SVM_linear <- svm(CKD~., data = training, kernel = "linear", gamma = 0.02564103, cost = 1)*
This is present in e1071 package. Svm is used to train a support vector machine. It can be used to carry out general regression and classification as well as destiny estimation. A formula interface is provided.

>*predVector1 <- predict(SVM_linear, newdata = testing)*
The ‘SVM_linear’ gives the object of the class which is the previous SVM linear function’s object and newdata is an optional frame in which to look for variables with which to predict.

>*table1 <- table(predVector1, testing$CKD)*
Table uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels. The parameters specifies the factors whose components can be interpreted.

>*confusionMatrix(table1, positive = "No")*
The confusionMatrix is present in the caret package. It calculates the cross tabulation of observed and predicted classes with associated statistics. It takes parameters like ‘table1’ which

is the data which is factor of predicted classes or an object of the class table. The ‘positive’ field is “yes” if there are only two levels for data, first level will be used as positive level.

>print("Support Vector Machines (linear) results:")
>print("Accuracy: 93.6%")
>print("Sensitivity: 100%")
>print("Specificity: 0%")
The above print statements are used to get result when the test file is executed.

>TestDataSVMLinear <- TestDataFactored
The ‘TestDataFactored’ is present in the TestFactorize.R file which contains 1907 observations of 33 variables.

>TestDataSVMLinear$CKD <- predict(SVM_linear, newdata = TestDataSVMLinear)
This method predicts the object of the class which is previously obtained using the SVM_linear function and new data gives the optional dataframe in which to look for variables with which to predict.

>write.csv(TestDataSVMLinear, file = "SVM_Linear.csv", row.names = F, quote = F)
This method writes to a .csv file where in the ‘TestDataSVMLinear’ is the object to be written to the file and file specifies the ‘file’ to which this info has to be copied and ‘row.names’ specifies whether row names should be written along with the test data. The quotes are not quoted usually but if true, factor or the character columns get quoted.

##RADIAL KERNEL

>obj_radial <- best.tune(svm, CKD~., data = training, kernel = "radial")
This generic function tunes hyperparameters of statistical methods using a grid search over supplied parameter ranges. Here ‘svm’ specifies the function to be tuned and CKD~. gives the formula or a matrix of predictors but here CKD value is validated against the all other attributes. The data specifies if a formula interface is used. The ‘kernel’ specifies the model for the svm which can be either linear or sigmoid or radial basis function.

>obj_radial
Displays the object obj_radial

>SVM_radial <- svm(CKD~., data = training, kernel = "radial", gamma = 0.02564103, cost = 1)
This is present in e1071 package. Svm is used to train a support vector machine. It can be used to carry out general regression and classification as well as destiny estimation. A formula interface is provided.

>predVector2 <- predict(SVM_radial, newdata = testing)
The ‘SVM_radial’ gives the object of the class which is the previous SVM radial function’s object and newdata is an optional frame in which to look for variables with which to predict.

>table2 <- table(predVector2, testing$CKD)
Table uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels. The parameters specifies the factors whose components can be interpreted.

>confusionMatrix(table2, positive = "No")
The confusionMatrix is present in the caret package. It calculates the cross tabulation of observed and predicted classes with associated statistics. It takes parameters like ‘table2’ which is the data which is factor of predicted classes or an object of the class table. The ‘positive’ field is “yes” if there are only two levels for data, first level will be used as positive level.

>print("Support Vector Machines (radial) results:")
>print("Accuracy: 93.6%")
>print("Sensitivity: 100%")
>print("Specificity: 0%")
The above print statements are used to get result when the test file is executed.

>TestDataSVMRadial <- TestDataFactored
The ‘TestDataFactored’ is present in the TestFactorize.R file which contains 1907 observations of 33 variables.

>TestDataSVMRadial$CKD <- predict(SVM_radial, newdata = TestDataSVMRadial)
This method predicts the object of the class which is previously obtained using the SVM_radial function and new data gives the optional data.frame in which to look for variables with which to predict.

>write.csv(TestDataSVMRadial, file = "SVM_Radial.csv", row.names = F, quote = F)
This method writes to a .csv file where in the ‘TestDataSVMRadial’ is the object to be written to the file and file specifies the ‘file’ to which this info has to be copied and ‘row.names’ specifies whether row names should be written along with the test data. The quotes are not quoted usually but if true, factor or the character columns get quoted.

##POLYNOMIAL KERNEL

>obj_polynomial <- best.tune(svm, CKD~., data = training, kernel = "polynomial")

This generic function tunes hyperparameters of statistical methods using a grid search over supplied parameter ranges. Here ‘svm’ specifies the function to be tuned and CKD~. gives the formula or a matrix of predictors but here CKD value is validated against the all other attributes. The data specifies if a formula interface is used. The ‘kernel’ specifies the model for the svm which can be either linear or sigmoid or radial basis function.

>obj_polynomial
Displays the object object_polynomial

>SVM_polynomial <- svm(CKD~., data = training, kernel = "polynomial", gamma = 0.02564103, cost = 1, degree = 3, coef.0 = 0)
This is present in e1071 package. Svm is used to train a support vector machine. It can be used to carry out general regression and classification as well as destiny estimation. A formula interface is provided.

>predVector3 <- predict(SVM_polynomial, newdata = testing)
The ‘SVM_polynomial’ gives the object of the class which is the previous SVM polynomial function’s object and newdata is an optional frame in which to look for variables with which to predict.

>table3 <- table(predVector3, testing$CKD)
Table uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels. The parameters specifies the factors whose components can be interpreted.

>confusionMatrix(table3, positive = "No")
The confusionMatrix is present in the caret package. It calculates the cross tabulation of observed and predicted classes with associated statistics. It takes parameters like ‘table3’ which is the data which is factor of predicted classes or an object of the class table. The ‘positive’ field is “yes” if there are only two levels for data, first level will be used as positive level.

>print("Support Vector Machines (polynomial) results:")
>print("Accuracy: 93.6%")
>print("Sensitivity: 100%")
>print("Specificity: 0%")
The above print statements are used to get result when the test file is executed.

>TestDataSVMPolynomial <- TestDataFactored
The ‘TestDataFactored’ is present in the TestFactorize.R file which contains 1907 observations of 33 variables.

>TestDataSVMPolynomial$CKD<-predict(SVM_polynomial,newdata= TestDataSVMPolynomial)

This method predicts the object of the class which is previously obtained using the SVM_polynomial function and new data gives the optional data.frame in which to look for variables with which to predict.

>write.csv(TestDataSVMPolynomial, file = "SVM_Polynomial.csv", row.names = F, quote = F)

This method writes to a .csv file where in the 'TestDataSVMPolynomial' is the object to be written to the file and file specifies the 'file' to which this info has to be copied and 'row.names' specifies whether row names should be written along with the test data. The quotes are not quoted usually but if true, factor or the character columns get quoted.

##SIGMOID KERNEL

>obj_sigmoid <- best.tune(svm, CKD~., data = training, kernel = "sigmoid")

This generic function tunes hyperparameters of statistical methods using a grid search over supplied parameter ranges. Here 'svm' specifies the function to be tuned and CKD~. gives the formula or a matrix of predictors but here CKD value is validated against the all other attributes. The data specifies if a formula interface is used. The 'kernel' specifies the model for the svm which can be either linear or sigmoid or radial basis function.

>obj_sigmoid

Displays the object obj_sigmoid

>SVM_sigmoid <- svm(CKD~., data = training, kernel = "sigmoid", gamma = 0.02564103, cost = 1, coef.0 = 0)

This is present in e1071 package. Svm is used to train a support vector machine. It can be used to carry out general regression and classification as well as destiny estimation. A formula interface is provided.

>predVector4 <- predict(SVM_sigmoid, newdata = testing)

The 'SVM_polynomial' gives the object of the class which is the previous SVM polynomial function's object and newdata is an optional frame in which to look for variables with which to predict.

>table4 <- table(predVector4, testing$CKD)

Table uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels. The parameters specifies the factors whose components can be interpreted.

>confusionMatrix(table4, positive = "No")

The confusionMatrix is present in the caret package. It calculates the cross tabulation of observed and predicted classes with associated statistics. It takes parameters like ‘table4’ which is the data which is factor of predicted classes or an object of the class table. The ‘positive’ field is “yes” if there are only two levels for data, first level will be used as positive level.

>print("Support Vector Machines (sigmoid) results:")
>print("Accuracy: 91.5%")
>print("Sensitivity: 97.1%")
>print("Specificity: 10.6%")
The above print statements are used to get result when the test file is executed.

>TestDataSVMSigmoid <- TestDataFactored
The ‘TestDataFactored’ is present in the TestFactorize.R file which contains 1907 observations of 33 variables.

>TestDataSVMSigmoid$CKD <- predict(SVM_sigmoid, newdata = TestDataSVMSigmoid)
This method predicts the object of the class which is previously obtained using the SVM_sigmoid function and new data gives the optional data.frame in which to look for variables with which to predict.

>write.csv(TestDataSVMSigmoid, file = "SVM_Sigmoid.csv", row.names = F, quote = F)
This method writes to a .csv file where in the ‘TestDataSVMSigmoid’ is the object to be written to the file and file specifies the ‘file’ to which this info has to be copied and ‘row.names’ specifies whether row names should be written along with the test data. The quotes are not quoted usually but if true, factor or the character columns get quoted.

Code results:
Parameters:
SVM-Type: C-classification
SVM-Kernel: linear
cost: 1
gamma: 0.02564103
Number of Support Vectors: 575
Confusion Matrix and Statistics

predVector1 No Yes
No 967 66
Yes 0 0

Accuracy : 0.9361

95% CI : (0.9194, 0.9502)
No Information Rate : 0.9361
P-Value [Acc > NIR] : 0.5327
Kappa : 0
Mcnemar's Test P-Value : 1.235e-15
Sensitivity : 1.0000
Specificity : 0.0000
Pos Pred Value : 0.9361
Neg Pred Value : NaN
Revalence : 0.9361
Detection Rate : 0.9361
Detection Prevalence : 1.0000
Balanced Accuracy : 0.5000
'Positive' Class : No

Support Vector Machines (linear) results:
Accuracy: 93.6%
Sensitivity: 100%
Specificity: 0%
Call:
best.tune(svm, CKD ~ ., data = training, kernel = "radial")
Parameters:
SVM-Type: C-classification
SVM-Kernel: radial
cost: 1
gamma: 0.02564103
Number of Support Vectors: 569

Artificial Neural Networks:

Artificial Neural Networks are rough electronic models based on the neurological structure of the brain. These biologically encouraged methods of computing are perceived to be the next major headway in the computing industry. This new way of classification does not utilize conventional programming methods but involves the creation of extensive parallel networks and these networks are trained to elucidate specific problems.

Various inputs to the network are represented by the mathematical symbol, which are multiplied by a connection weight. In the elementary case, these products are summated, input through a transfer function to generate a by-product, and then the final output. The electronic execution is possible with other network structures which utilize divergent summation functions as well as distinct transfer functions.

Correctly classified instances	*100%*
Incorrectly classified instances	*0%*
Kappa statistic	*1*
Mean absolute error	*0.0009*
Root mean squared error	*0.003*
Relative absolute error	*0.183%*
Root relative squared error	*0.747%*

Table 4 : This table gives the results of the training set on the application of ANN classifier from Weka

Results using RStudio:
Code snippet:

>*set.seed(0305)*
We have to set the seed every time we want to get a reproducible random result. The value given as parameter can be any random number.

>*ModelDataNum <- as.data.frame(sapply(ModelDataFactored, as.numeric))*
This is a method to check if an object is a data.frame or coerce it if possible. It takes as arguments any R object. 'Sapply' is user-friendly version of 'lapply' and also a wrapper of 'lapply' by default returning a vector or matrix. It now converts the 'ModelDataFactored' vector or object to a numeric form and then sets it as a data. frame.

>*index <- createDataPartition(ModelDataFactored$CKD, p = 0.75, list = FALSE)*
This createDataPartition is present in the 'caret' package. It takes a vector of outcomes which is the 'ModelDataFactored' as a parameter and the value of 'p' specifies the percentage of data that goes to the training set and 'list' specifies whether the results should be in a list or not.

>*training <- ModelDataNum[index,]*
The training object gets 3102 observations of 33 variables.

>*testing <- ModelDataNum[-index,]*
The testing object gets 1033 objects of 33 variables.

>n <- names(training)
‘Names’ is a function to get or set names of an object. So for the training object, the names are set and then assigned to another object names ‘n’.

>names <- paste(n[1:32], collapse = ' + ')
‘Paste’ is used to concatenate vectors after converting to character. It takes as parameters the objects to be converted to character form and ‘collapse’ specifies the optional character string to separate the results.

>f <- paste("CKD ~ ", names)
‘Paste’ is used to concatenate vectors after converting to character. It takes as parameters the objects to be converted to character form and here the objects that are to be converted will be the CKD~. which specifies the class attribute or the formula object with respect to all the other explanatory variables and ‘names’ the previous object.

>nn <- neuralnet(f, training, hidden = c(5,4), threshold = 0.03, stepmax = 5e+07, learningrate = 0.05, lifesign = "full")
‘Neuralnet’ is present in the neuralnet package and it is for training of the neural networks. It takes as arguments, the ‘formula’ which is a symbolic description of the model to be fitted and ‘data’ gives information regarding the data frame containing the variables specified in the formula, ‘hidden’ gives a vector of integers specifying the number of hidden neurons in each layer, ‘threshold’ is a numeric value specifying the threshold for the partial derivatives of the error function as stopping criteria, ‘stepmax’ gives the maximum steps for the training of neural network, ‘lifesign’ is a string specifying how much the function will print during the calculation of neural network and here it prints full.

>nn
Displays the nn object previously generated.

>imnn.results <- compute(nn, testing[-33])
‘Compute’ is present in CRAN package. This computes the effective sizes in R. This package provides a comprehensive set of tools/functions to easily derive and/or convert statistics generated from one’s study to all of the common effect size estimates, along with their variances, confidence intervals, and p-values.

>predVector <- (imnn.results$net.result < 1.5)
This assigns the net.result in the object imnn.result whose value is less than 1.5 to another object which is the predVector.

>table <- table(Predicted = (predVector), Actual = (testing$CKD == 1))

Table uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels. The parameters specifies the factors whose components can be interpreted.

>table
Displays the table object.

>print("Neural Network results:")
>print("Sensitivity: 100%")
>print("Specificity: 0%")
The above print statements are used to get result when the test file is executed.

>TestDataNN <- TestDataFactored
The 'TestDataFactored' is present in the TestFactorize.R file which contains 1907 observations of 33 variables.

>TestDataNN <- as.data.frame(sapply(TestDataNN, as.numeric))
This is a method to check if an object is a data.frame or coerce it if possible. It takes as arguments any R object. 'Sapply' is user-friendly version of 'lapply' and also a wrapper of 'lapply' by default returning a vector or matrix. It now converts the 'TestDataNN' vector or object to a numeric form and then sets it as a data. frame.

>nn.predNew <- compute(nn, TestDataNN)
'Compute' is present in CRAN package. This computes the effective sizes in R. This package provides a comprehensive set of tools/functions to easily derive and/or convert statistics generated from one's study to all of the common effect size estimates, along with their variances, confidence intervals, and p-values.

>TestDataNN <- round(nn.predNew$net.result)
This 'round' method is used for rounding of numbers. So here it rounds the values from the object nn.predNew from the previous case with respect to the net.result and then stores the rounded values in TestDataNN.

>write.csv(TestDataNN, file = "NeuralNet.csv", row.names = F, quote = F)
This method writes to a .csv file where in the 'TestDataNN' is the object to be written to the file and file specifies the 'file' to which this info has to be copied and 'row.names' specifies whether row names should be written along with the test data. The quotes are not quoted usually but if true, factor or the character columns get quoted.

>TestDataNN <- NULL
The TestDataNN object is set to NULL.

>*testing <- NULL*
The testing object is set to NULL.

>*training <- NULL*
The training object is set to NULL.

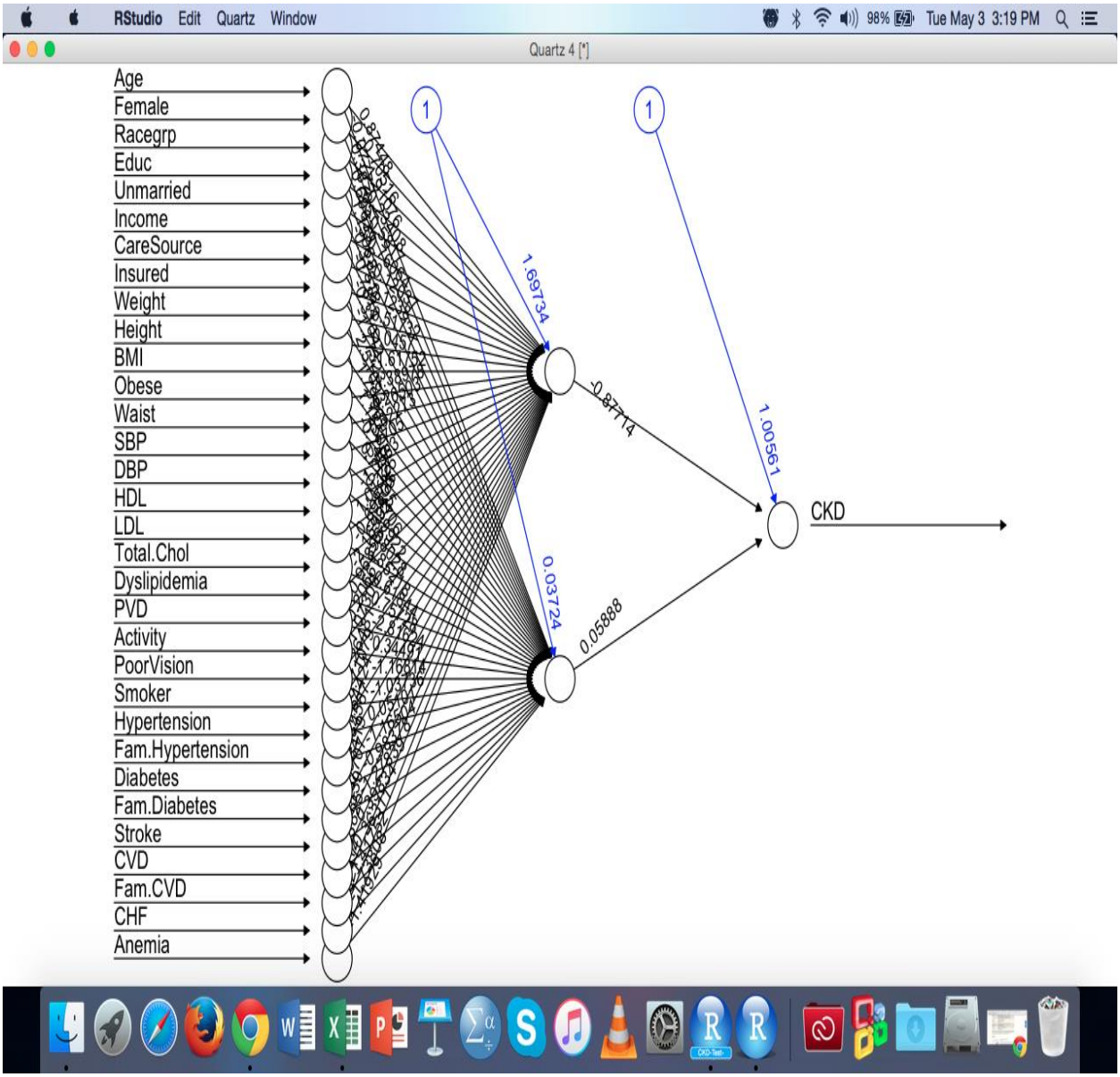


Figure :6 Plot of Neural Network.

KDD PRINCIPLES

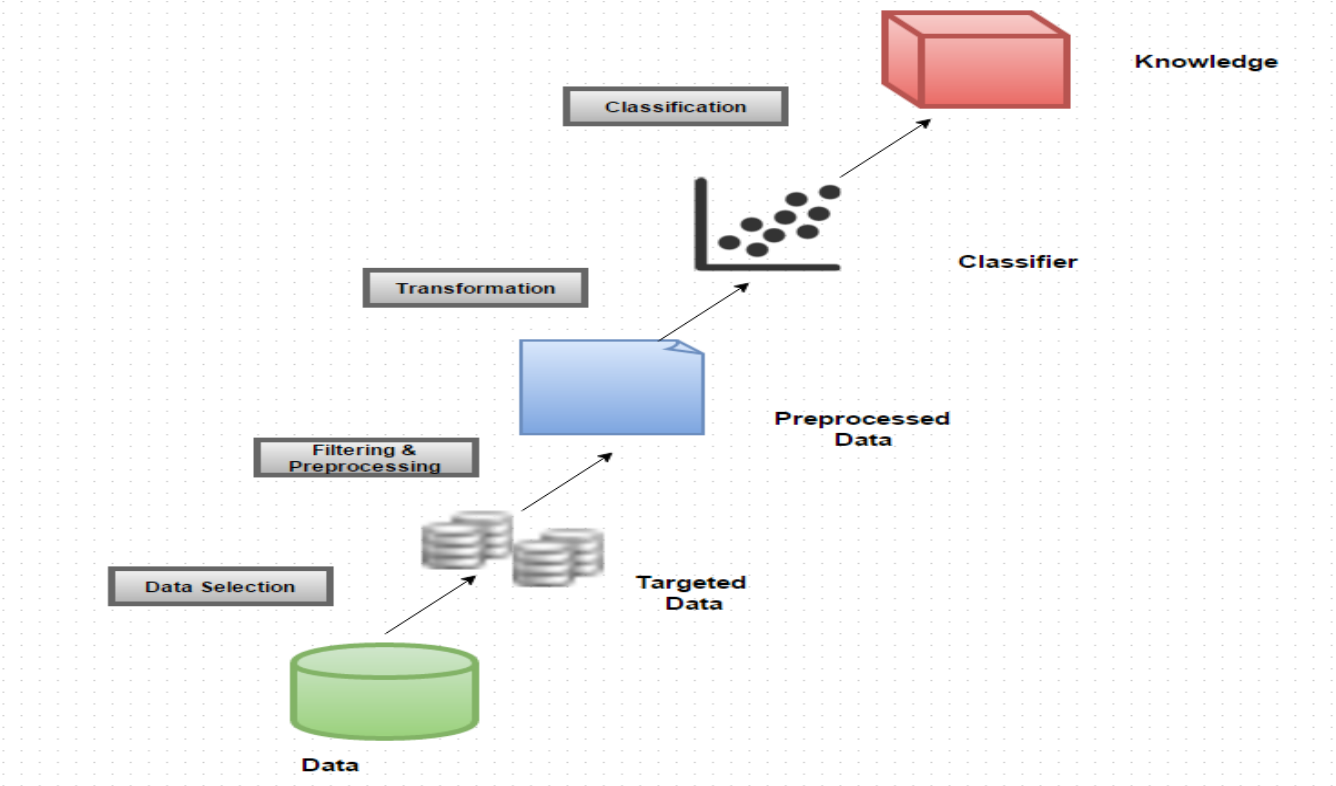


Figure 7 :KDD Architecture

1. Data Cleansing:

Remove the noise and inconsistent data, we have removed unwanted columns and unrelated columns and finally created the data set which has to be trained.

2. Data Integration:

Data Integration block is not used in our project as we only have one data source i.e. National Center for Health Statistics of the Centers for Disease Control
and Prevention

3. Data selection:

Selected the data that is required for predicting the likelihood of getting CKD. Hence we have created the database for using fields.This is our target dataset. We have a CSV file from the CSV

file we have extracted useful columns. Thus we have created the data set and prepared it for preprocessing.

4. Data Transformation: (where data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations, for instance

Our program in Weka and R studio will generate data with following columns:

- Weight
- Height
- BMI
- Obese
- SBP
- DBP
- HDL
- LDL
- Total Cholesterol
- Dyslipidemia
- PVD
- Activity
- Smoker
- Age
- Gender
- Marital Status
- CVD
- CHF
- Race Group
- Income
- Family History of Diabetes, Hypertension and CVD

This data is used for processing and analysis and prediction purposes.

5. Classification:

As a part of data mining principles we are using the Naive Bayes Classifier, SVM and ANN classifier to predict the likelihood of getting chronic kidney disease given the symptoms.

6.Knowledge presentation:

We are predicting the likelihood of getting chronic kidney disease given the symptoms and giving comparison of prediction accuracies of different classifiers.

Thus we have used the above data mining data principles to classify the data.

DATA TOOLS:

Implementation in WEKA:

1. The dataset is split into 75% training data and 25% testing data.
2. In the preprocessing step click on “open file” which opens the .CSV file.

3. For classification click on the classify tab and select the Naive Bayes Classifier and click on start and then we get the results for the Naive Bayes Classification.
4. For classification click on trees and then select ID3 classifier click on start and then we get the results for the Decision tree Classification.
5. For classification click on functions and then select SVM classifier click on start and then we get the results for the SVM Classification.
6. For classification click on functions and then select Multi Layer perceptron classifier click on start and then we get the results for the ANN Classification.

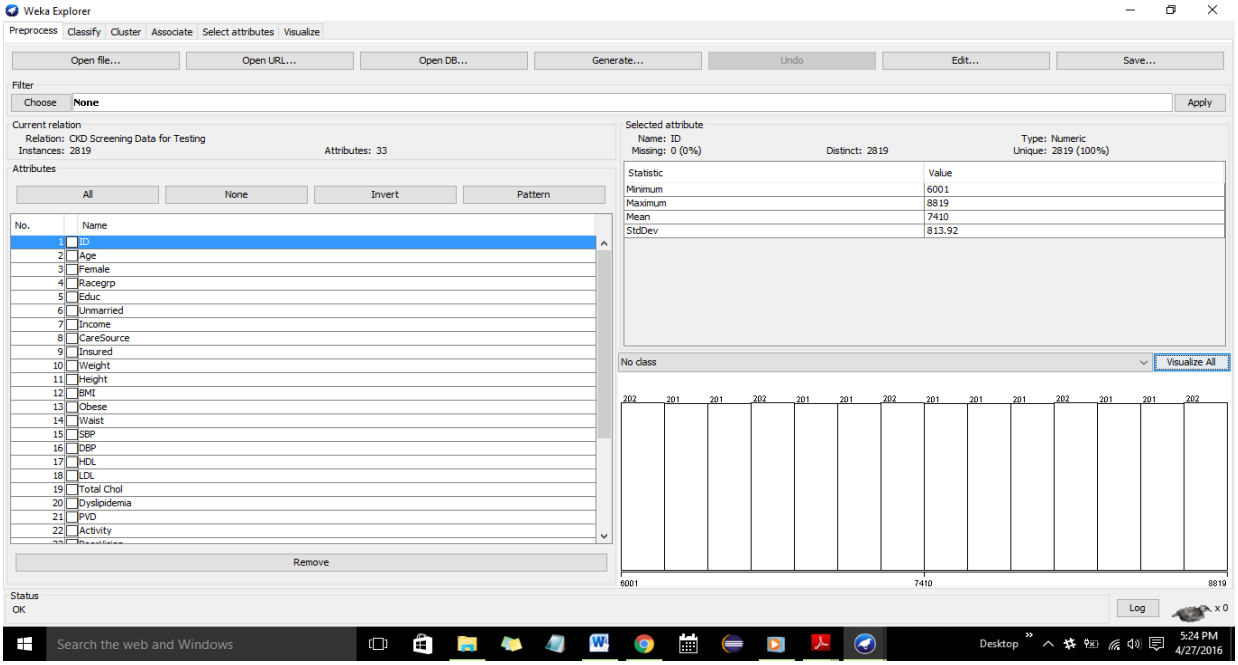


Figure 8 : Loading Of Attributes in the Weka Tool

Figure 9: Visualization of Attributes in the Weka Tool

Implementation in R Studio

1. The dataset is split into 75% training data and 25% testing data.
2. Install the packages respective to the algorithms like Naive bayes, SVM and Neural Networks.
3. Implement the Naive bayes by using the predefined function that takes formula and the training data as parameters and stores the results of the naive bayes algorithm in an object.
4. Implement the SVM by using the predefined function SVM that takes formula and training data as parameters and stores the results based on the kernel values. The kernel values for linear, polynomial,sigmoid and radial as into different objects like SVM_Linear, SVM_Polynomial, SVM_Sigmoid, SVM_Radial.
5. Implement the artificial neural networks by using the predefined function neuralnet that takes the formula and training data as parameters and stores the output of the neuralnet method in an object named nn.
6. Now implement the User interface using Shiny by installing the shiny package and then creating two files-one for server which is server.R and the other for client which is the ui.R. These files contain the user interface design.
7. The user selects the distribution type on the page and then the distribution of the attributes with respect to the class attributes are presented in a graphical format and the drop down list contains features of distribution like- training, testing, NaiveBayes, SVM_linear, SVM_polynomial, SVM_radial, SVM_sigmoid, NeuralNet which includes all the algorithms that we implemented in the project.

8. The user is given a chance to enter the attributes so that the user's data can be validated against the already available data and then prediction can be made as of whether the user has a probability of getting the disease or not.
9. Whenever the user chooses an attribute and gives its value, the data gets entered into the dataset and then the prediction is done with the added data so that the user will be presented with the graphical plots as of which attribute's value has more probability of resulting in a Chronic Kidney Disease.

USER INTERFACE DESIGN

We have used the Shiny Server and Shiny tool in RStudio for data visualization for each impact factor of Chronic Kidney Disease at the front end.

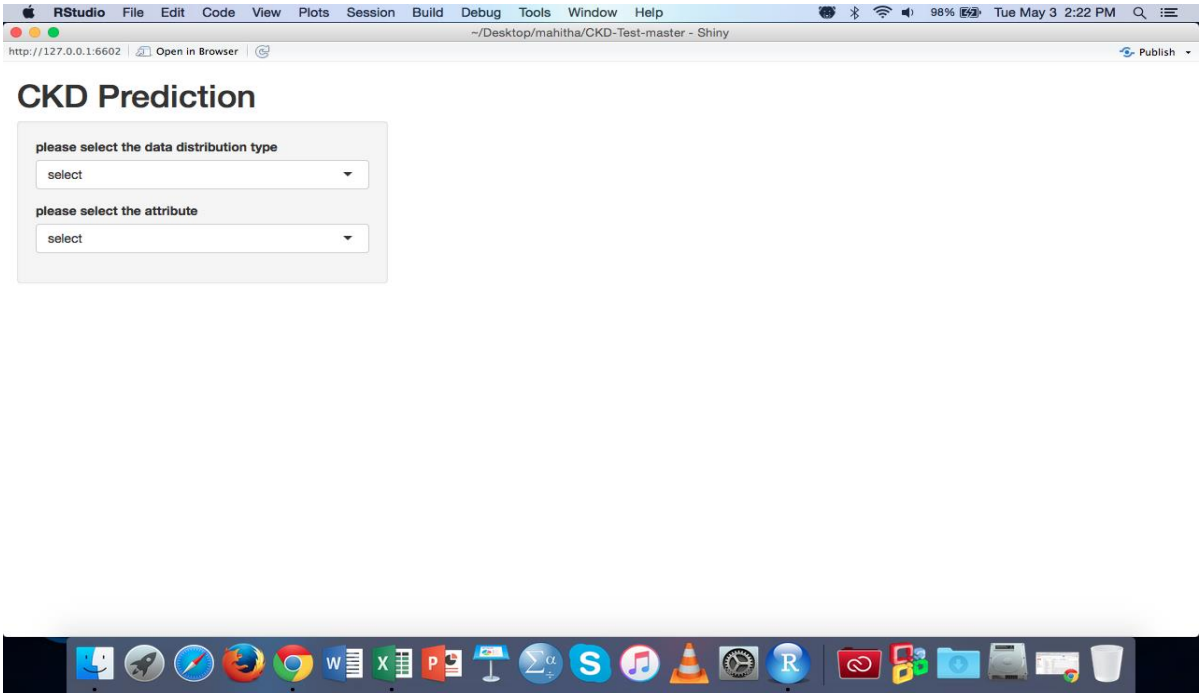


Figure 10 :Main Application Page

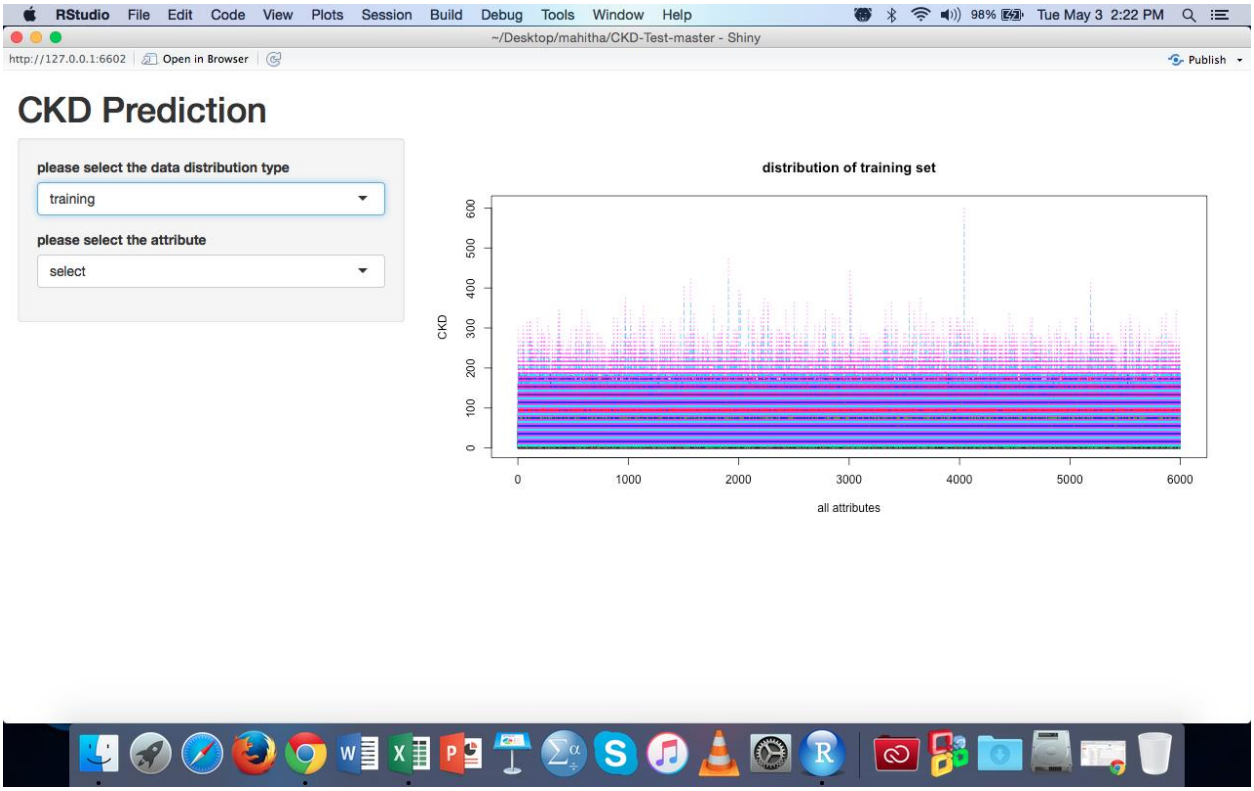


Figure 11: Visualization of the Training DataSet

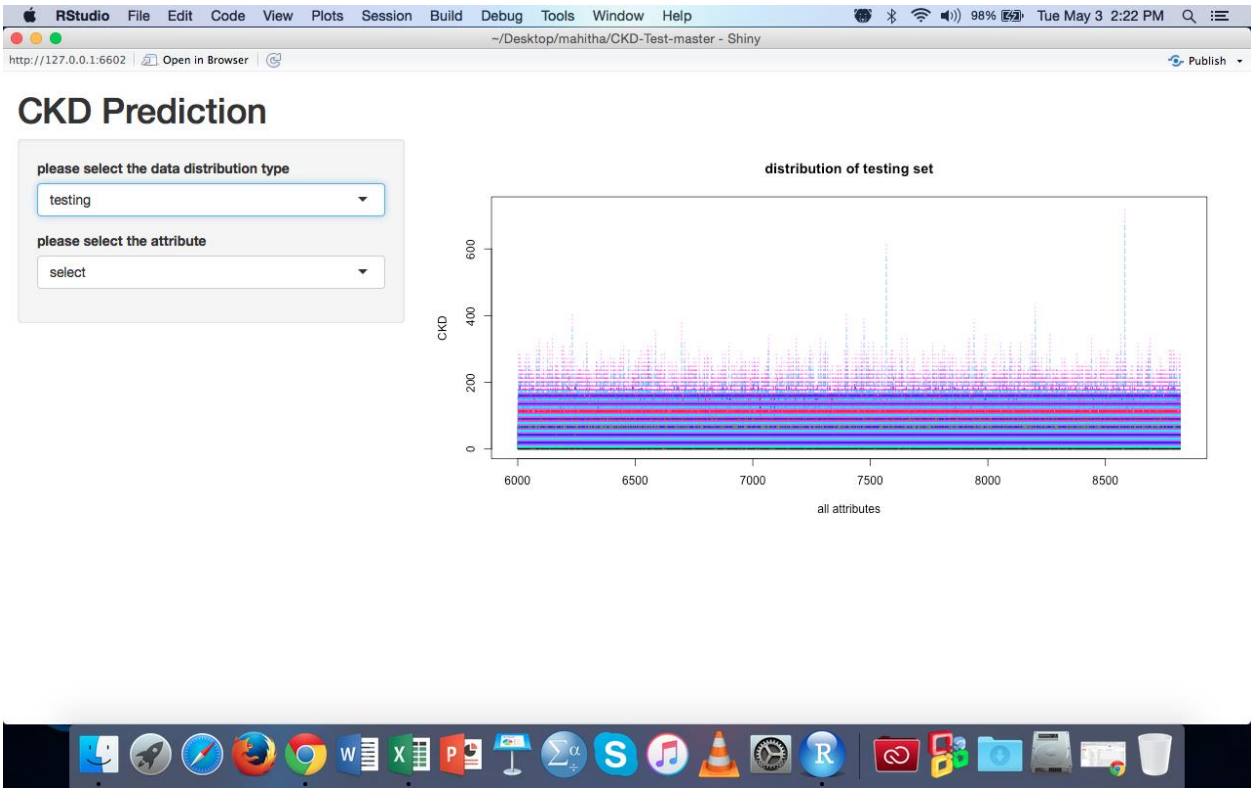


Figure 12: Visualization of the Testing DataSet

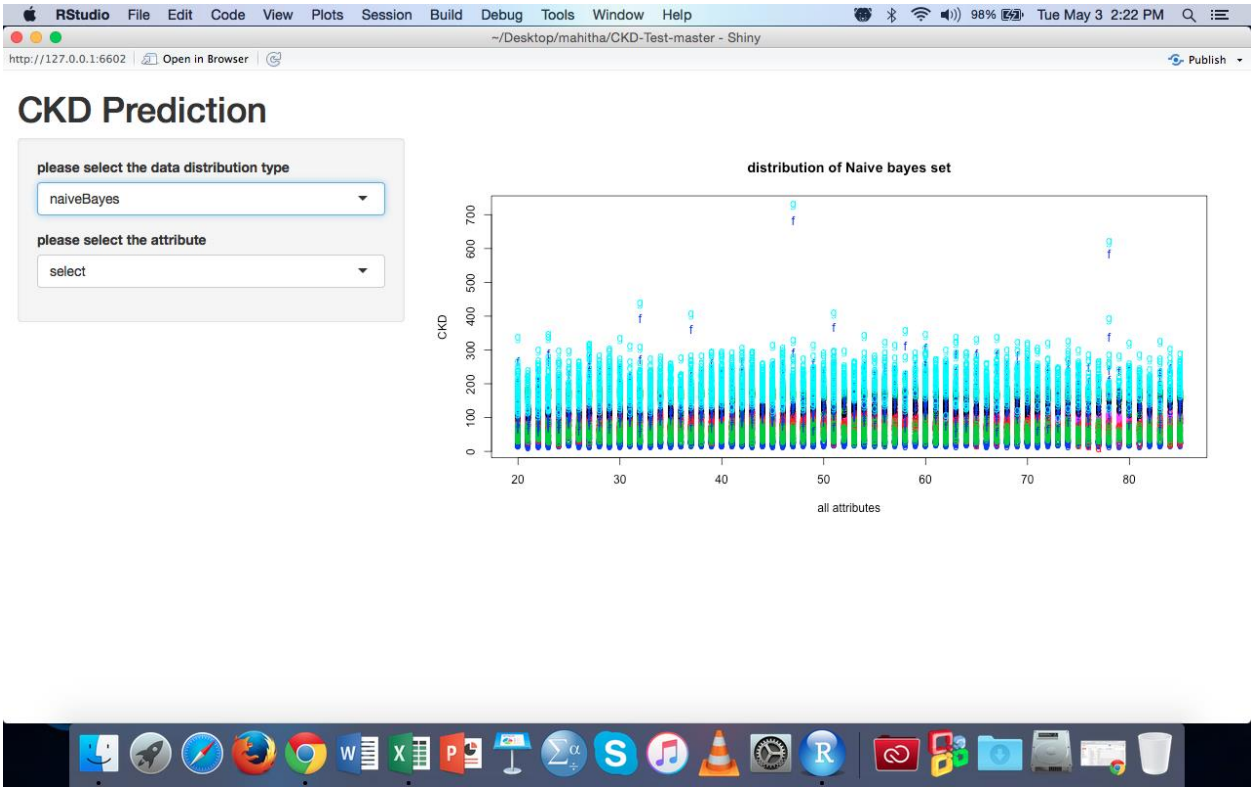


Figure 13: Visualization of the Naive Bayes Model

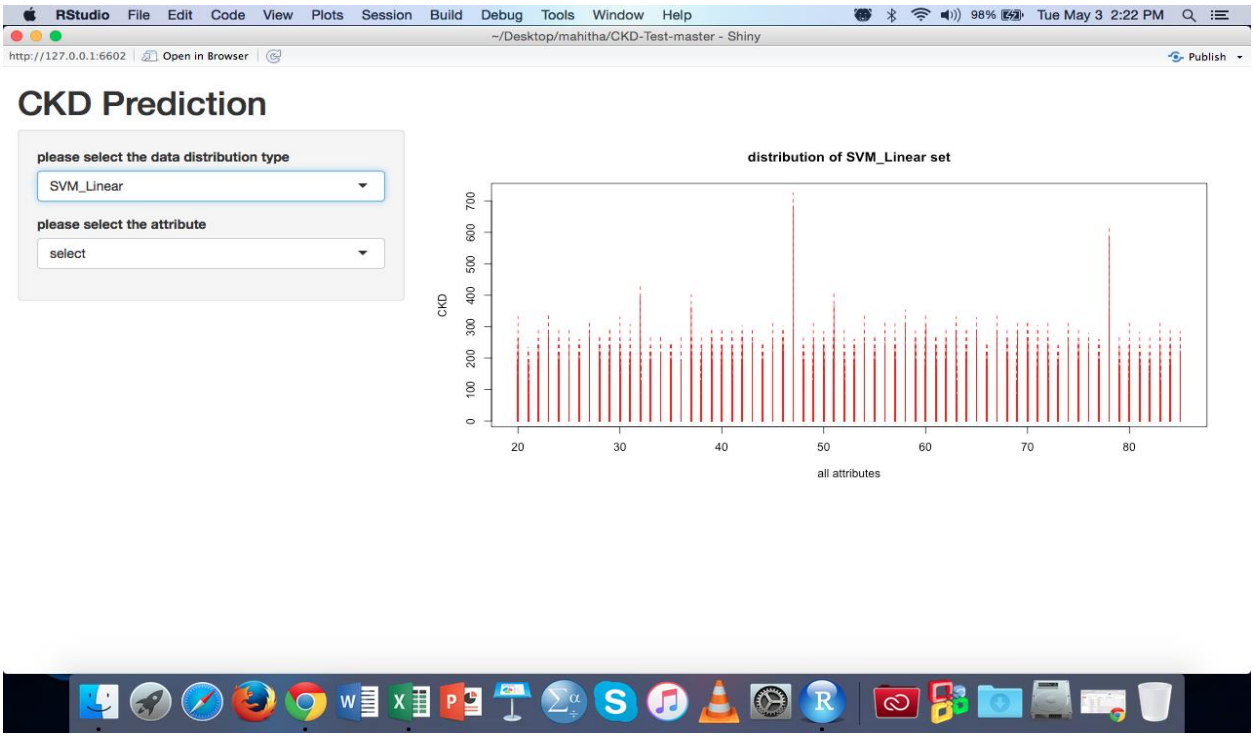


Figure 14: Visualization of the SVM Linear Model

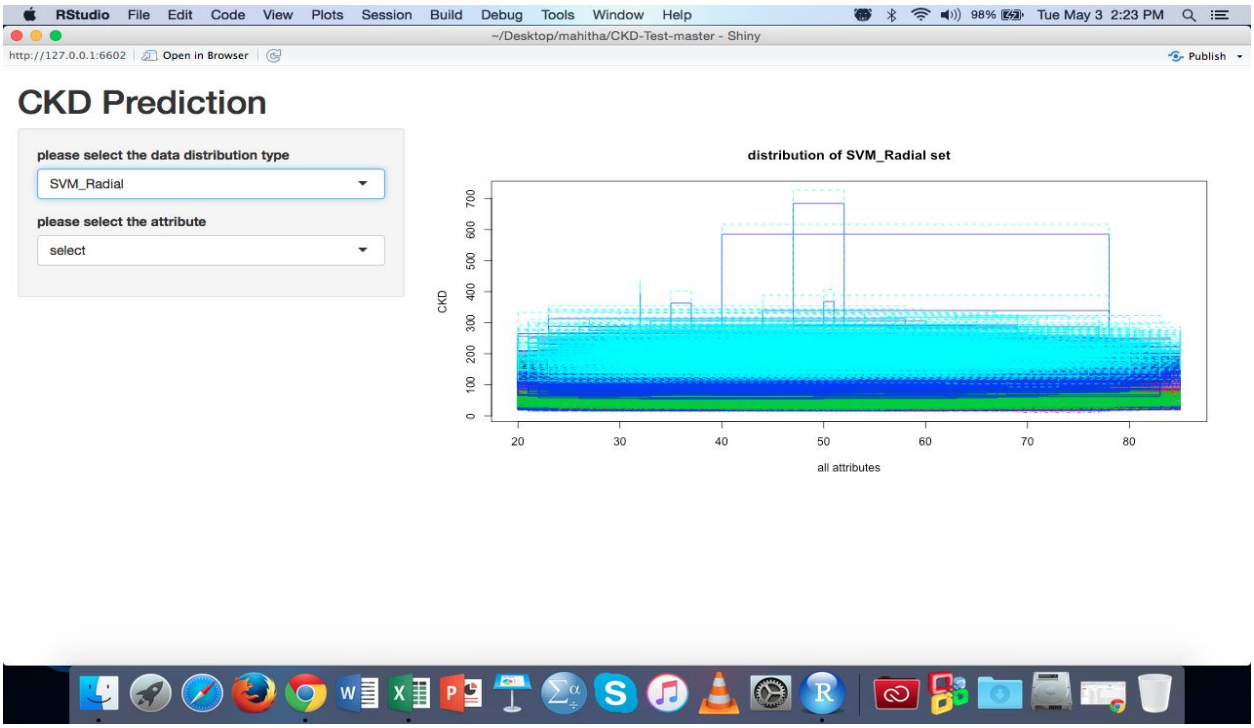


Figure 15: Visualization of the SVM Radial Model

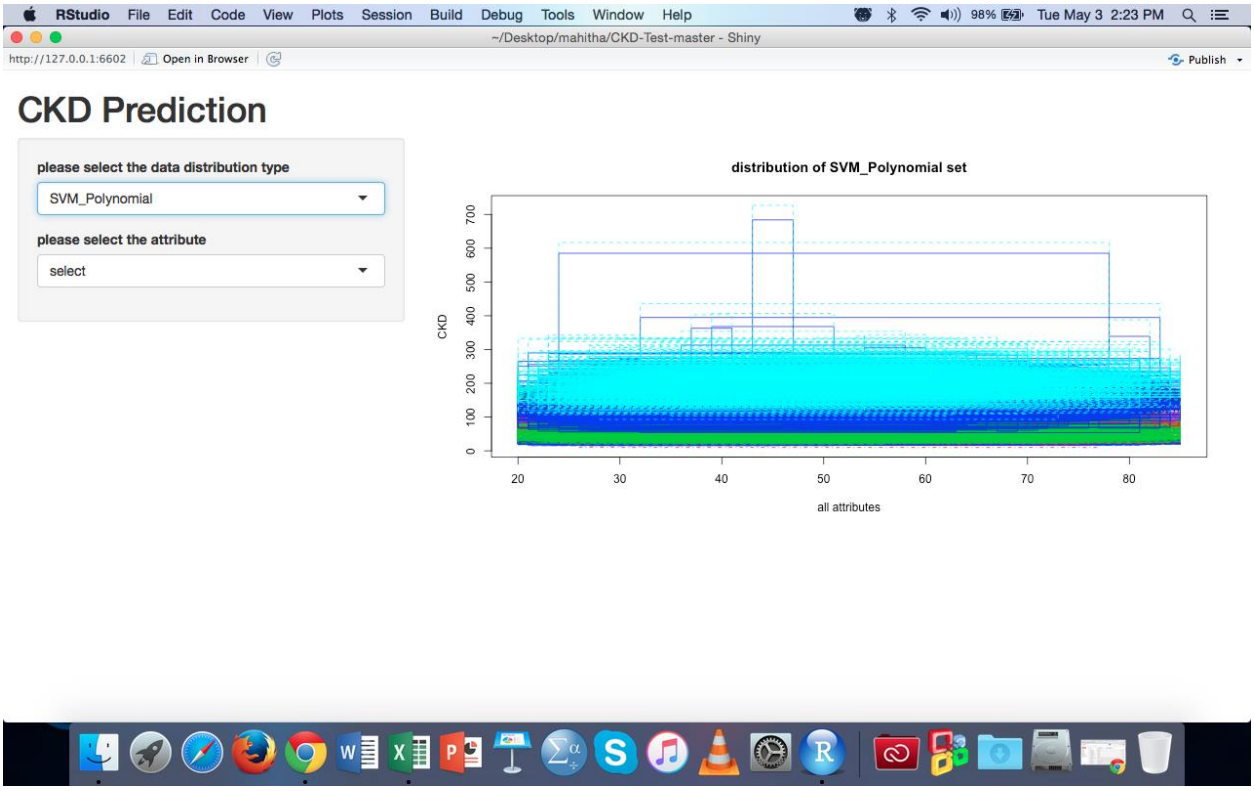


Figure 16: Visualization of the SVM Polynomial Model

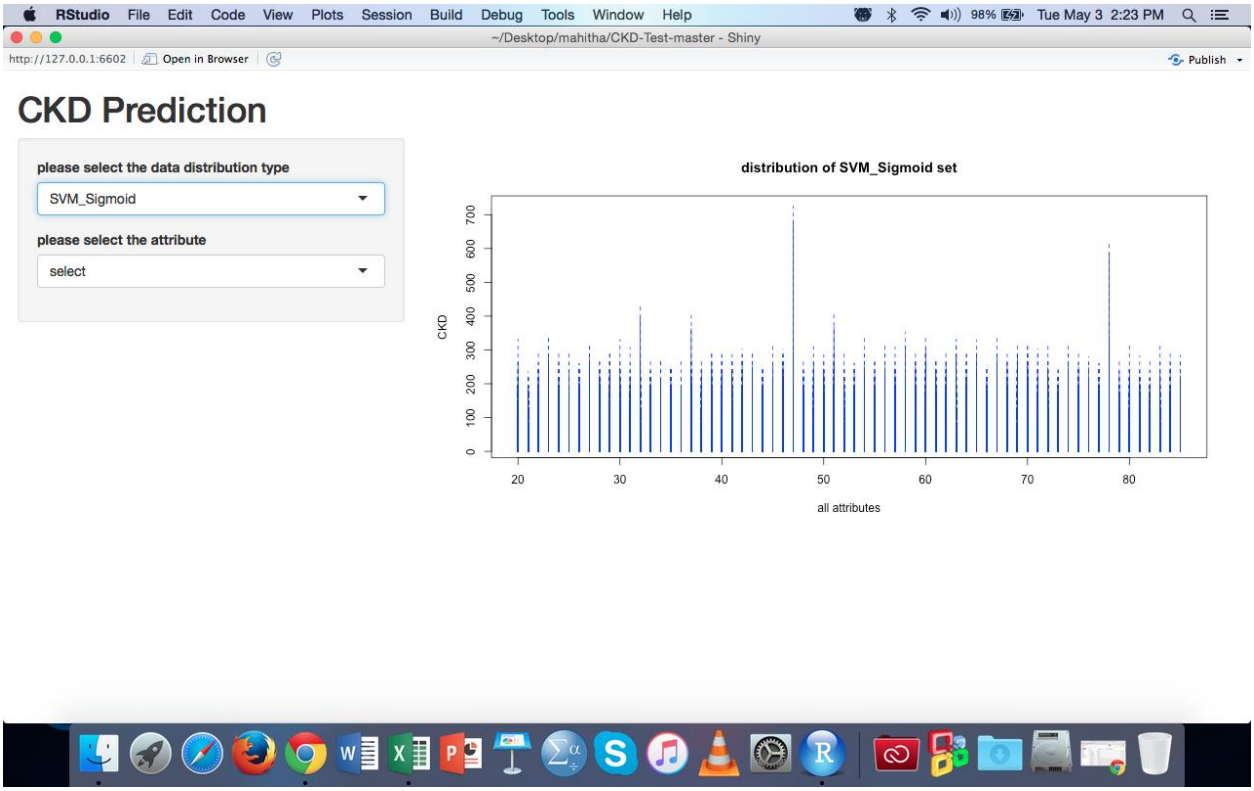


Figure 17: Visualization of the SVM Sigmoid Model

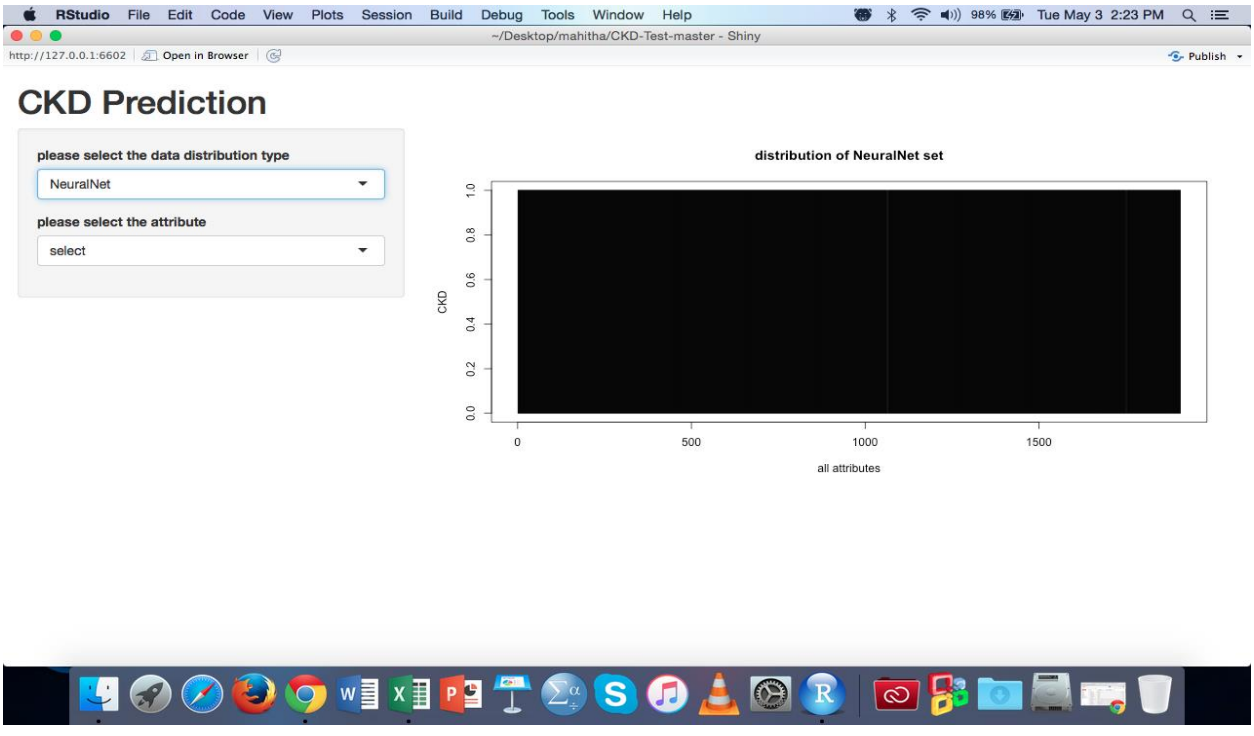


Figure 18: Visualization of the Neural Net Model

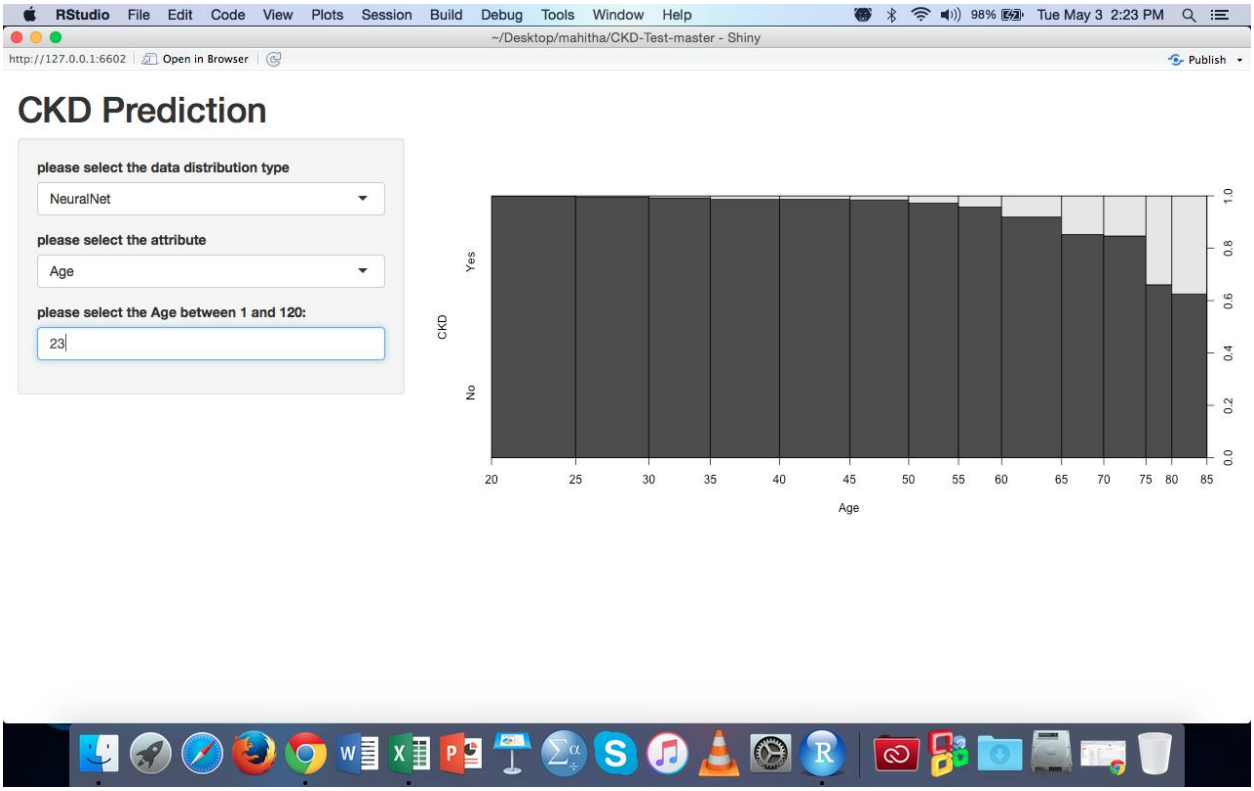


Figure 19 :Graph for Age Factor

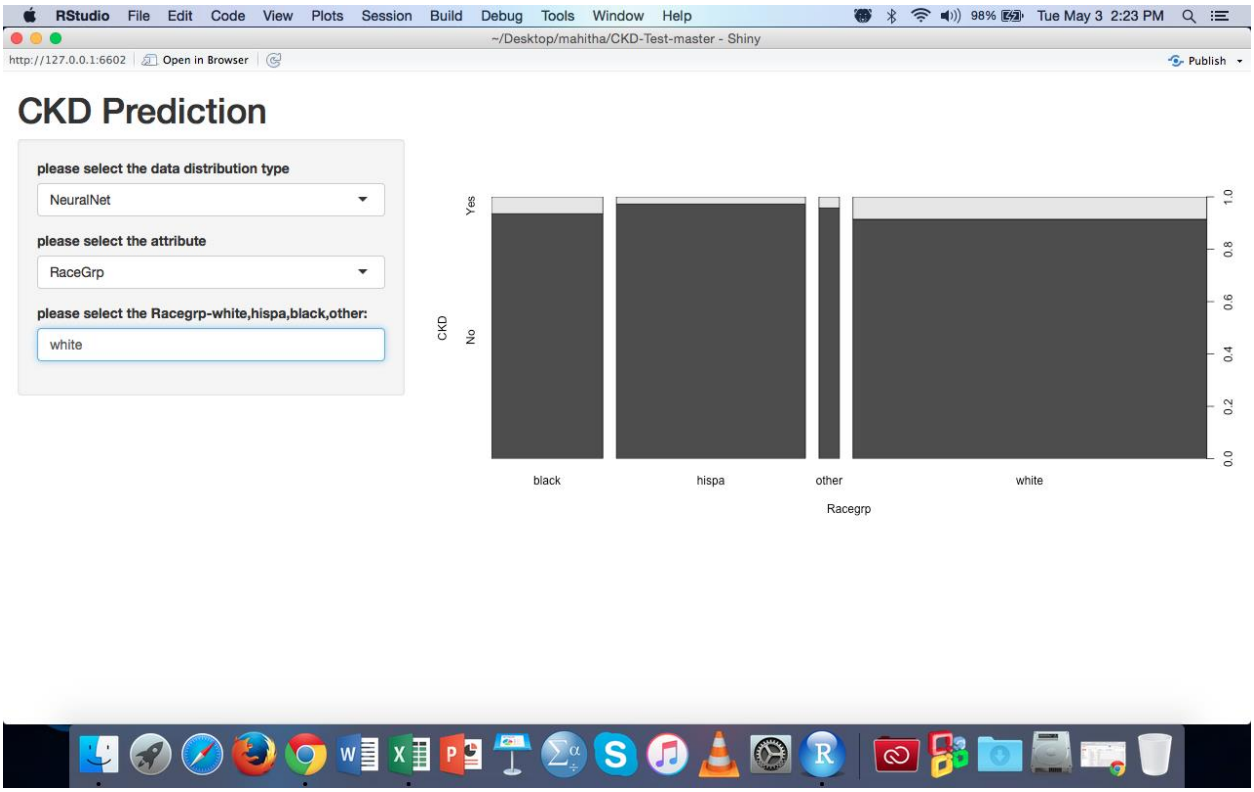


Figure 20 :Graph for Race Group Factor

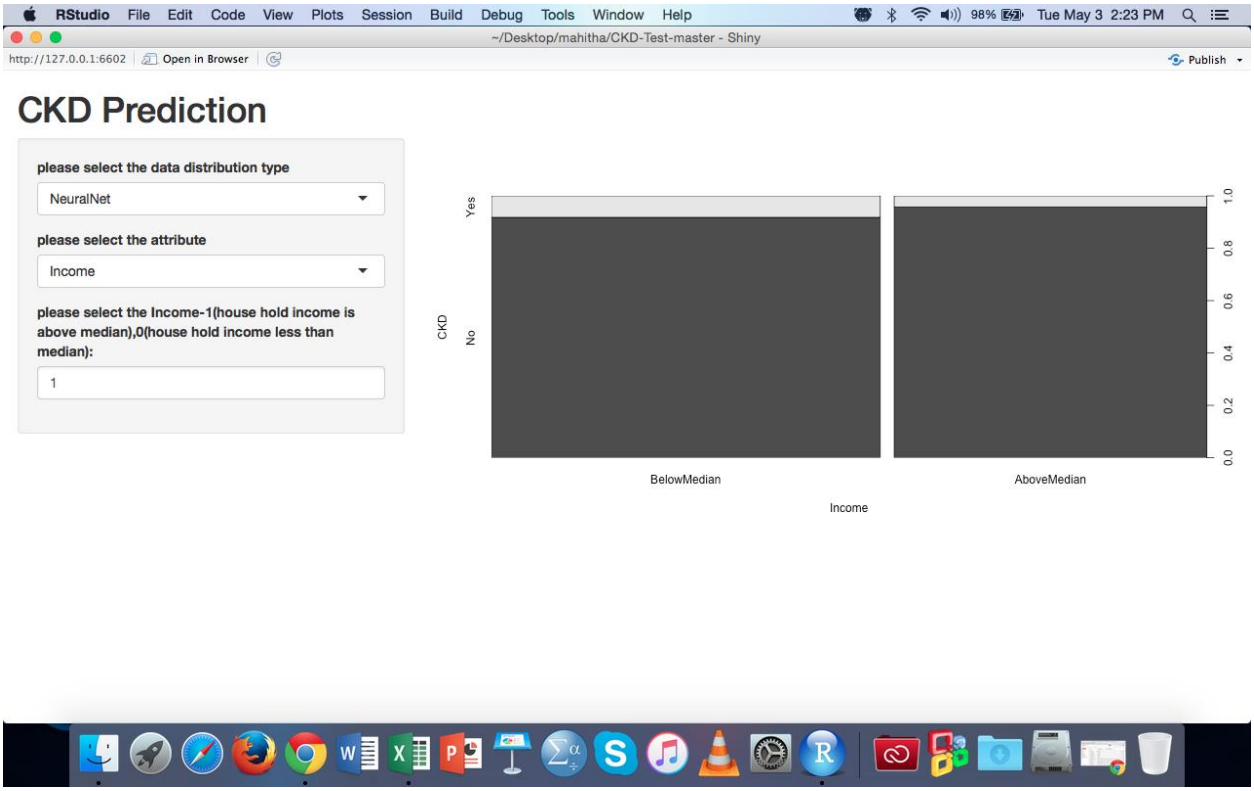


Figure 21 :Graph for Income Factor

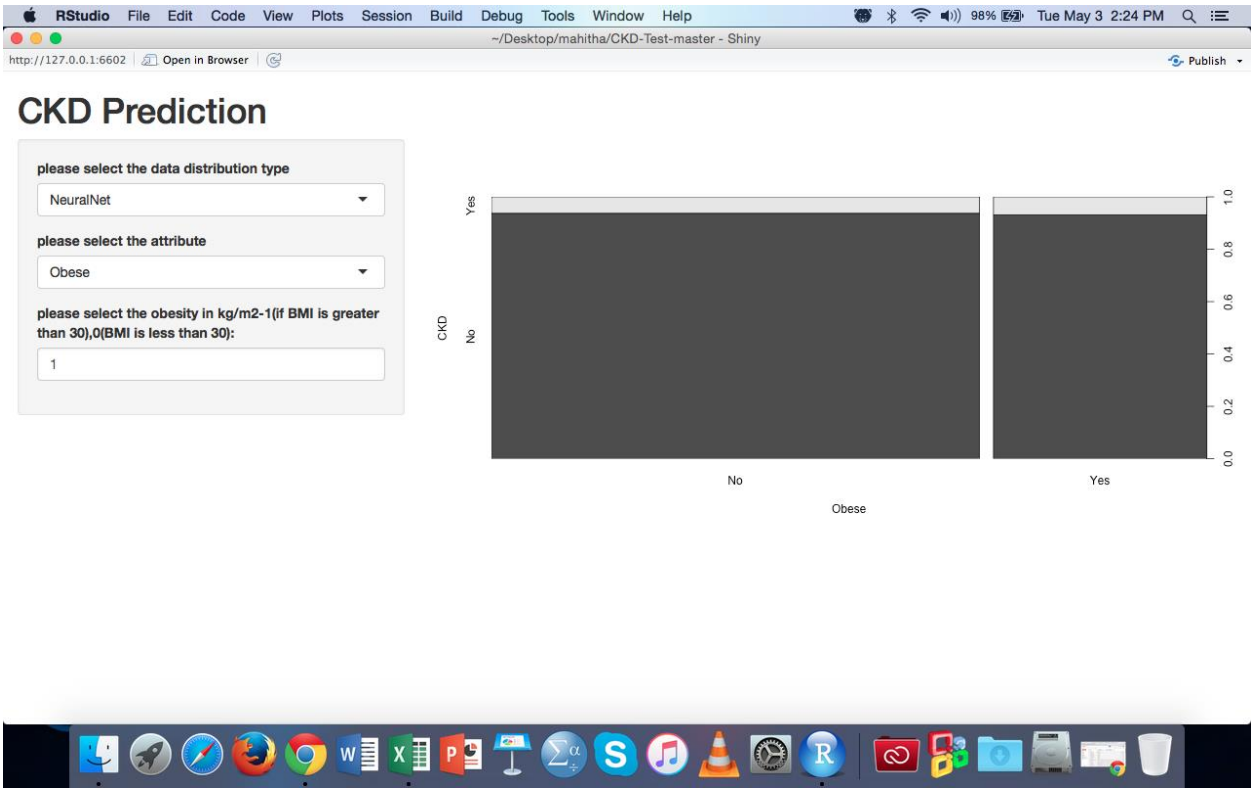


Figure 22 :Graph for Obesity Factor

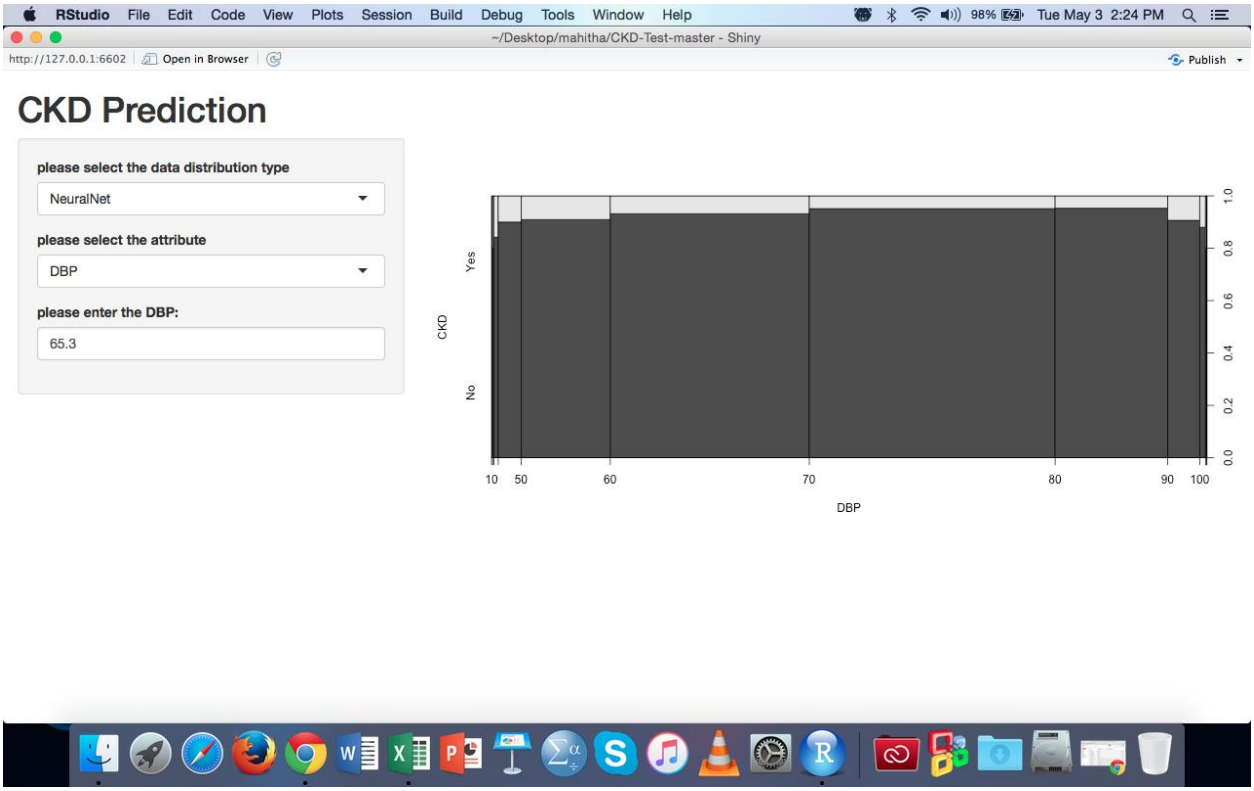


Figure 23 :Graph for SBP Factor

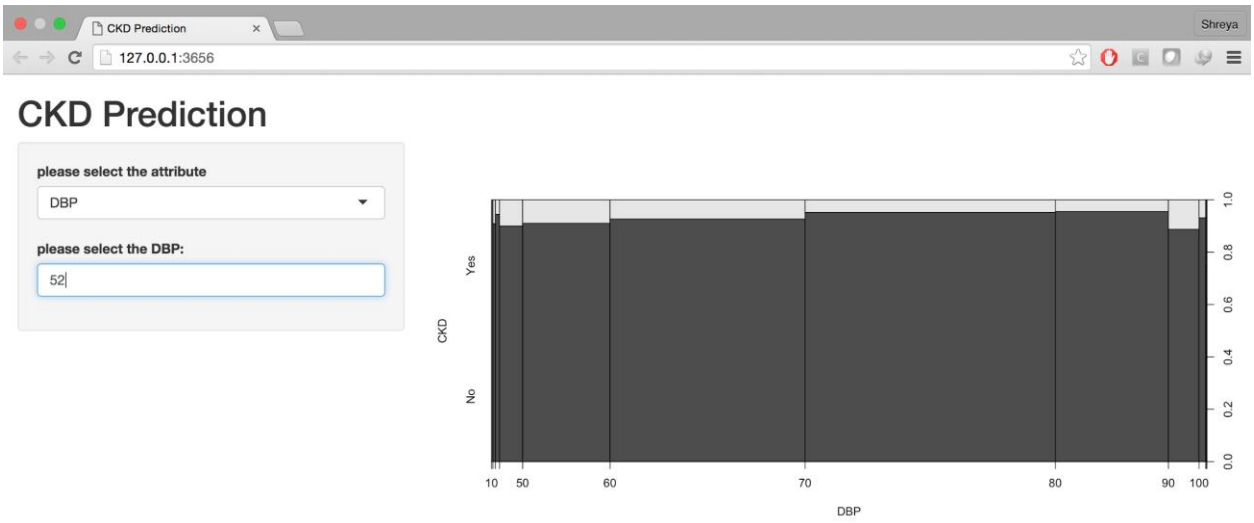


Figure 24 :Graph for DBP Factor

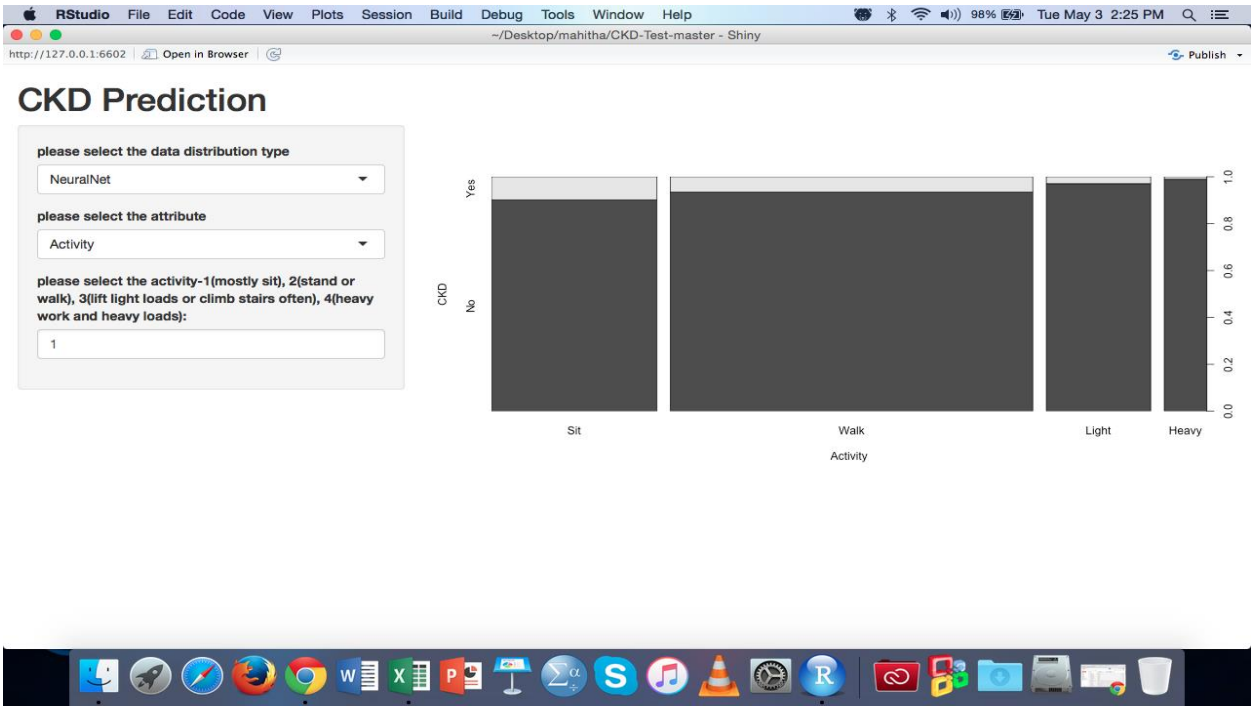


Figure 25 :Graph for Activity Factor

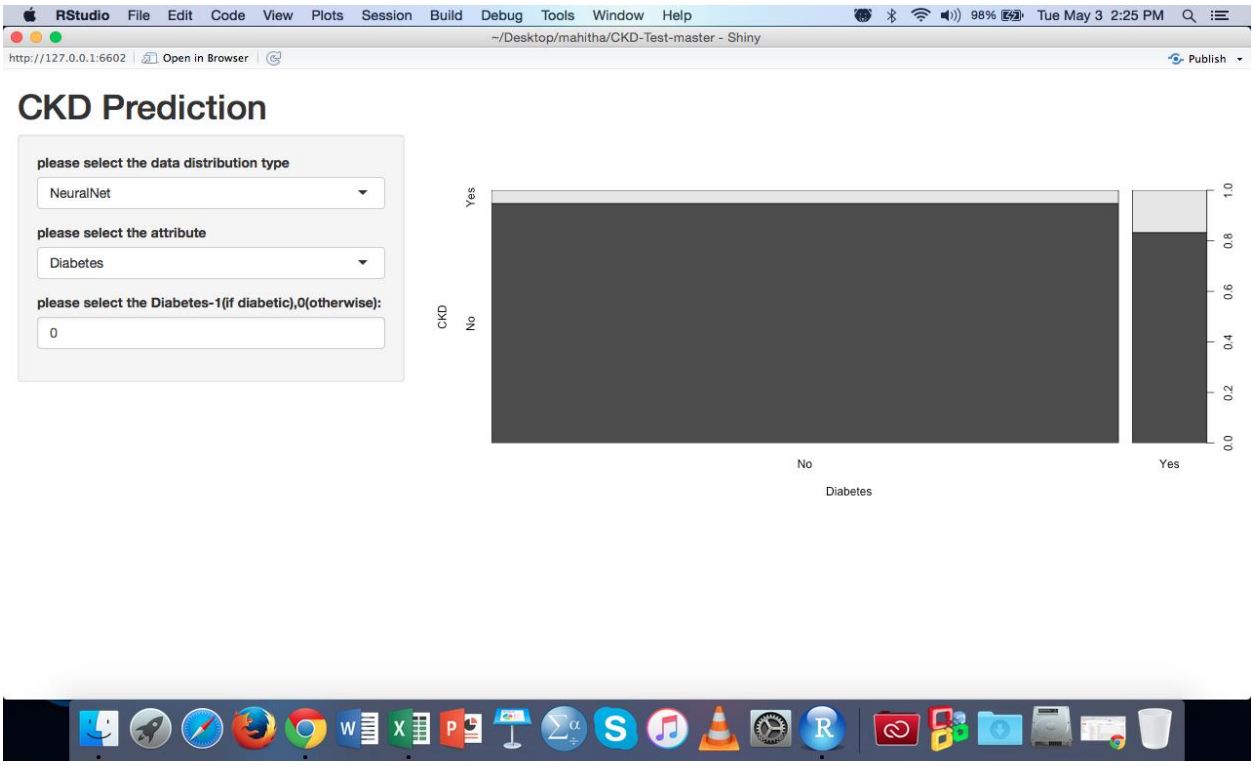


Figure 26: Graph for Diabetes Factor

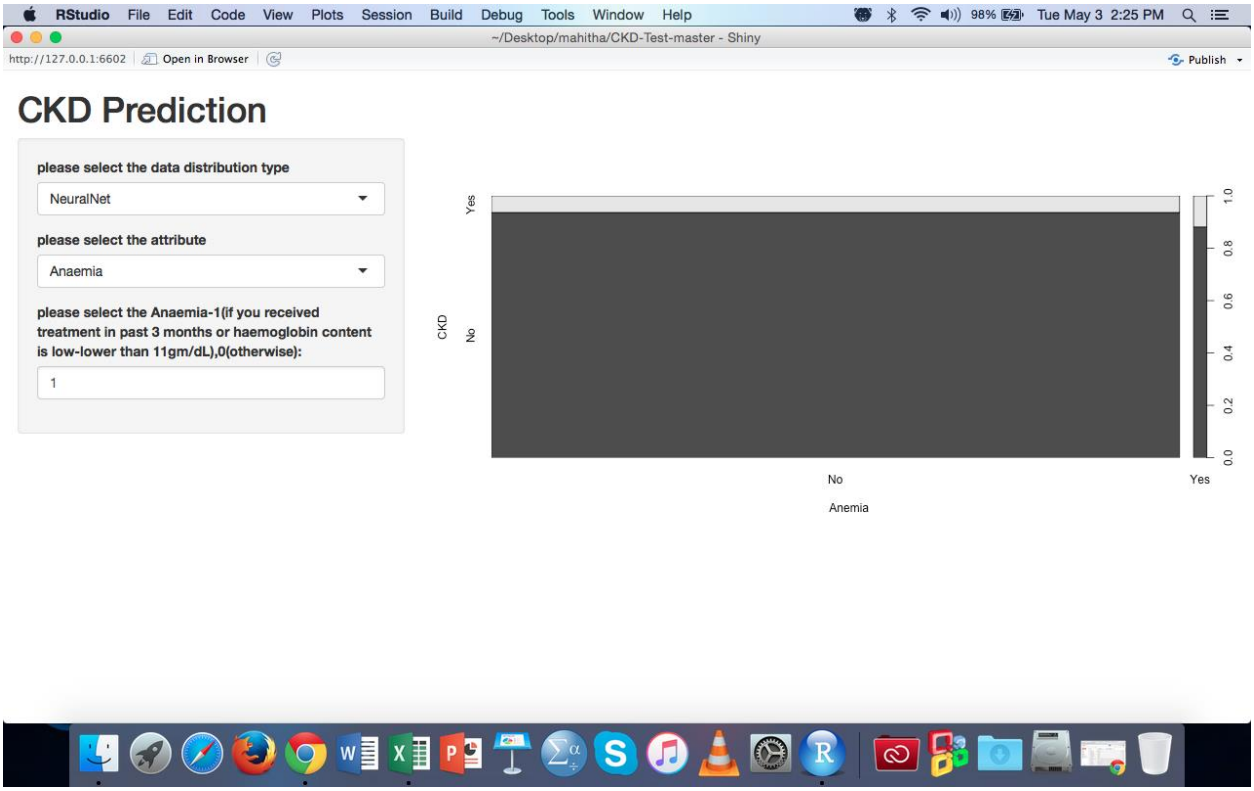


Figure 27 :Graph for Anaemia Factor

DESIGN PATTERNS USED

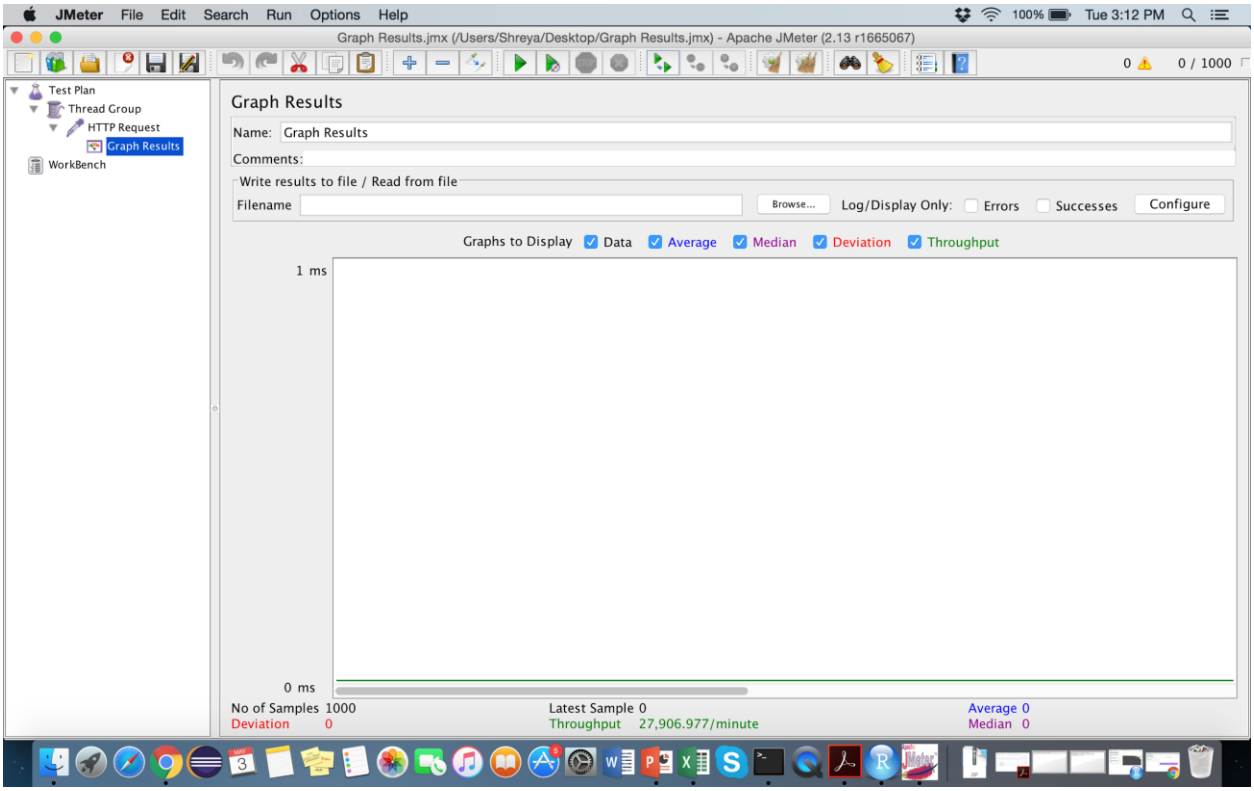
Model view controller

MVC is a design pattern widely used in enterprise softwares in order to divide the responsibilities using a layered approach. It is a widely used design pattern adopted across many languages and frameworks. The purpose of MVC is to decouple data from the view. Model represents the business logic and data. View represents visualization of the data. Controller is the connecting link between Model and View and it contains the application logic of the software. It manipulates data using the actions of the user.

In our project the View is ui.R, the Model is server.R, and the Controller is embedded in ui.R and denoted by the input argument passed to server.R in the R studio at the back end and Shiny Server and tool at the front end.

TESTING

We have done the load testing using Jmeter. Please find below the screen shots for the test results. We have done load testing for 1000 Users.



INDIVIDUAL CONTRIBUTION

Name	**Responsibilities**
Mahitha Byreddy	Wrote scripts in R Studio and Weka for Naive Bayes classifier, Project Report,Shiny UI
Charmili Narra	Wrote scripts in R Studio and Weka for Support Vector Machine (SVM) classifier, Project Report
Shreya Prabhu	Wrote scripts in R Studio and Weka for Artificial Neural Network (ANN) classifier, Project Report, Dataset Management