**STAT 860**

**STATISTICAL MACHINE LEARNING**

**FINAL PROJECT REPORT**

**SUBMITTED BY**

**YING LIN, MORUF DISU, MICHAEL HEIM**

**SUBMITTED TO**

**MINGAO YUAN (Ph.D.)**

**SUBMITTED ON**

**MAY 12TH, 2020**

**Research Question and Data**

Obesity has become a major public health concern in the U.S. in recent years. As a result of this, Type 2 diabetes has become more prevalent as well. Many previous studies suggest that Type 2 diabetes does have some underlying genetic factors, and certain populations may be more at risk than others. There are also environmental factors associated with 21st Century living and the American lifestyle that contribute to the prevalence of obesity and diabetes in today’s society.

A study conducted in 2004 studied diabetes in the Pima Indians. This population was of particular interest because the group has limited genetic and environmental variability. They also have the highest reported prevalence of diabetes of any population in the world. The study collected data, particularly from female members aged 21 and over, to investigate genetic factors related to Type 2 diabetes in this population. The variables collected include:

* Number of pregnancies
* Plasma glucose 2 hours in an oral glucose test
* Diastolic blood pressure (in mmHg)
* Triceps skin fold thickness test (in mm)
* 2 hour serum insulin (in muU/ml)
* Body Mass Index (weight in kg/(height in m2)
* Diabetes Pedigree Function
* Age
* Diabetes presence

With this data, we would like to investigate which of these variables can be used as predictors for developing Type 2 diabetes, and apply statistical machine learning techniques and algorithms to predict the onset of Type 2 diabetes in this population.

**Methods and Results**

Linear Discriminant Analysis

Linear discriminant analysis attempts to express or predict a response variable or group membership as a linear combination of some explanatory variables. The method assumes the density function of the variables within each group is Gaussian, or multivariate normal, and that there is equality of covariance of matrices across the groups.

Given a set of data, the density function, covariance matrix, and group probabilities are estimated. To predict the class of a new variable, the log-odds of groups are compared, and the new variable is assigned to the group which maximizes the probability of the new variable belonging in said group.

First, the covariance matrices of the data on Pima Indians Type II Diabetes were tested to see if it was reasonable to assume there was a single equal covariance matrix between the groups of people who tested positive and negative for Type II diabetes. This assumption was met. Then, the data was randomly broken into a test and training set (split 50/50). The training set was used to develop the LDA classifier, and the test set was used to evaluate prediction error rates.

The prior probability of testing negative estimated from the training set was about 66%. After constructing the classifier and using the test set for prediction, it was found that LDA was 77% accurate in predicting the test status.

Quadratic Discriminant Analysis

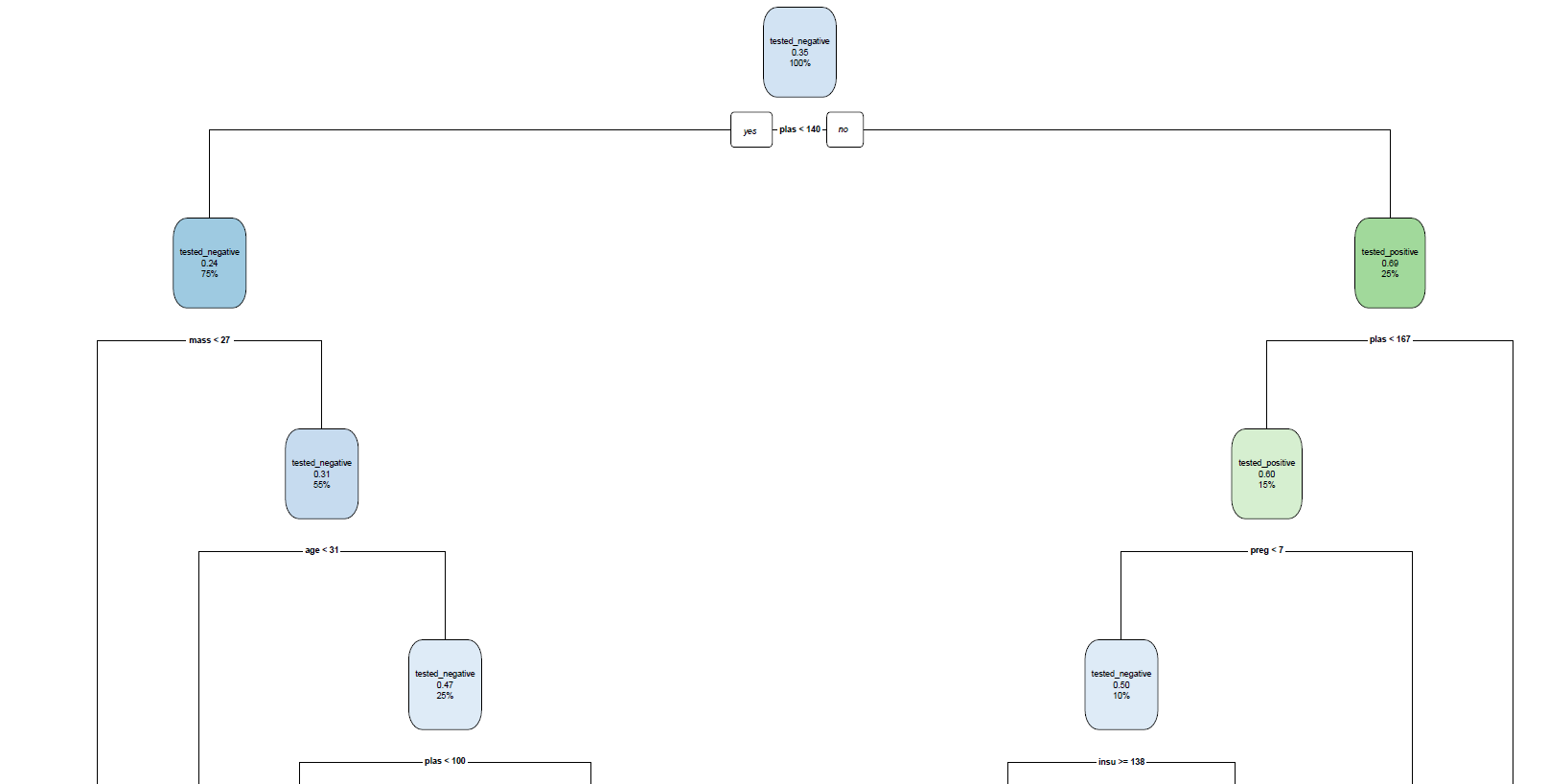
Similar to linear discriminant analysis, quadratic discriminant analysis attempts to predict class membership using a linear combination of explanatory variables. QDA has slightly more lax assumptions, as it doesn’t require equality of covariance matrices between the classes. Thus, the differences in LDA and QDA are small, but QDA can be more desirable when this assumption is not met.

In the Pima Indians Diabetes data, the covariance matrices can be assumed equal between those who tested negative and positive for Type II diabetes. The prior probability of testing negative for diabetes was the same as LDA, about 66%. After constructing the model and using the test set for prediction, QDA was 74% accurate in predicting the test status. This is comparable to LDA, but slightly lower. Likely, the reason for LDA performing better is because of the assumption of equality of covariance matrices between those who tested positive and negative for diabetes. Since the assumption is safely met, LDA is more appropriate and provides better accuracy, although QDA is still a valid method.

Decision Tree

A decision tree is a process by which the set of splitting rules is used to segment the predictor space and summarize it in a tree. The process of building a decision tree takes a top-down, greedy approach that is known as the recursive binary splitting. It’s said to be top-down because it begins from the top of the tree and then successively splits the predictor. The approach is said to be greedy because, at each step of the tree building process, the best split is made at that particular step. Decision trees can be applied to regression and classification problems.

This study makes use of the classification tree approach that is used to predict a qualitative response. This approach not only focuses on the class of prediction corresponding to a particular node but also, aims at interpreting the proportion among the training observations. Classification trees make use of the error rate, Gini index, or the cross-entropy for the recursive binary approach. We use the Gini Index approach that is defined as the measure of total variance across the classes.

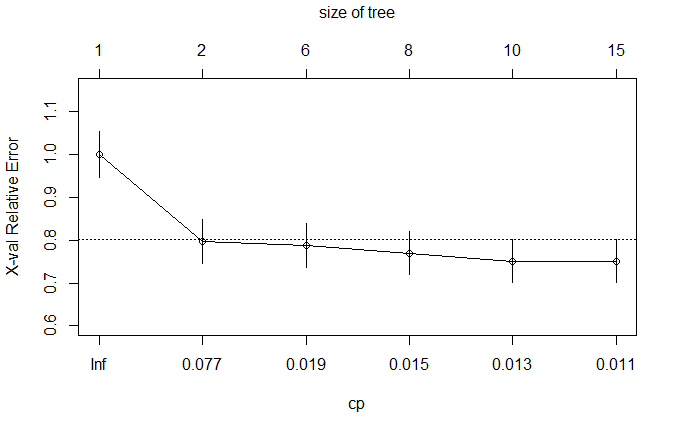
The dataset was preprocessed for any missing value or if there were outliers in the data, but after careful processing, there were no missing observations and no outliers in the data. The data is then split into 80% training data and the rest as the test data. The tree classification below Shows the proportion and percentage of variables that contributed to the criteria of type II diabetes that is either tested positive or tested negative. It shows that 75% of those with less than 140 plasma glucose concentration is tested negative while the remaining 25% of patients with more than 140 plasma glucose concentration are tested positive for type II diabetes. Also, we can infer from the tree that those patients with body mass index more than 27kg/m2 that are tested negative but are older than 31 years old have 25% tested negative while patients with age less than 31 years have 30% tested positive. The root node error of the model is 35.34%.

Decision Tree Plot

|  |  |  |
| --- | --- | --- |
| Model Prediction | | |
|  | Tested Negative | Tested Positive |
| Tested Negative | 90 | 13 |
| Tested Positive | 19 | 32 |

Table 1: Confusion Matrix

The table shows that 90 patients are tested negative, 32 patients are tested positives, 13 of the patients are tested negative but are positive while 19 patients are misclassified as tested positive but are tested negative. This indicates that using 614 of the training data on 154 test data, we can predict that the sensitivity and the specificity of the data is approximately 79%.



The above plot shows the Cp value with the size of the tree and the x-val relative error, minimum value of cp is desired but the full model uses the minimum model with a better prediction accuracy.

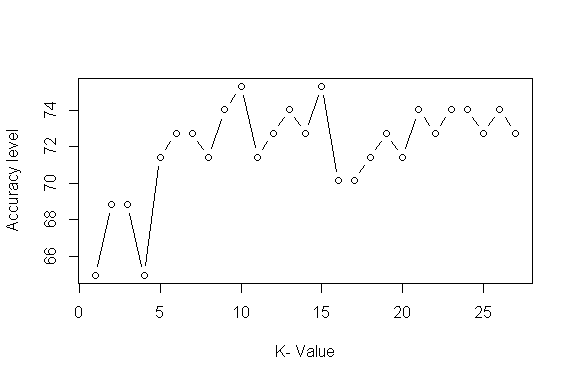
Random Forest

Random forest is used as an improvement to the other trees by growing n- different number of trees. The results of each tree from random forest is tally to which output has the important votes. The output with the most votes is then the output of the forest. We focus on the use of Random Forest classification.

The out of bag estimation error rate of the model using 80% of the train data is approximately 24%. Using the 80% of the train data on the 20% of the test data, it can be predicted that out of the 154 patients of the tested data , 88 patients are tested to be true negative, 33 patients are tested positive, 11 patients are tested positive but are negative while 22 patients are tested negative but are positive. The specificity and sensitivity of the model is approximately 79%.

K-Nearest Neighbors

The algorithm of kNN is that a case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. Since all predictors are continuous, we can use euclidean distance and standardize the input features to have mean zero and variance 1.



According to the accuracy level plot, k=10 is the optimal choice. The table of confusion matrix shows that 23 cases of the 77 test samples are misclassified, and the remaining 54 cases are correctly classified. So the test error rate is 29.87% and the accuracy rate is 70.13%.

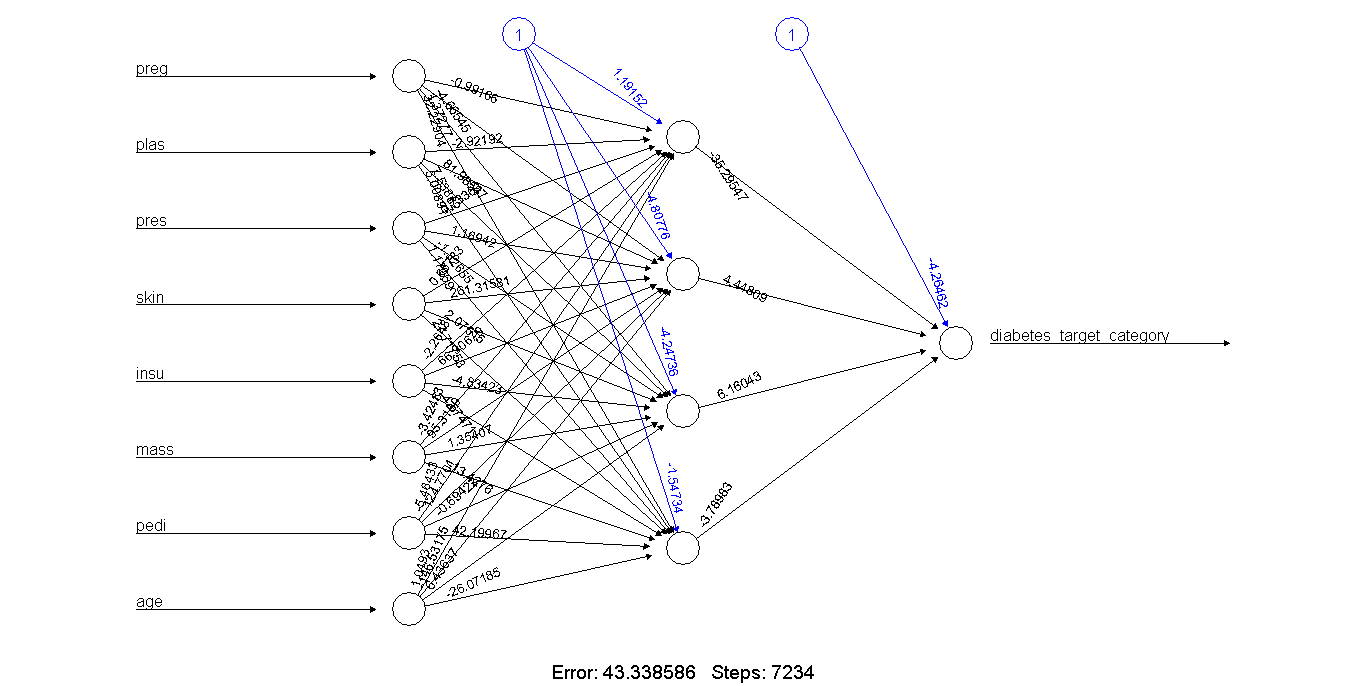
|  |  |  |  |
| --- | --- | --- | --- |
|  | | Observed Label | |
| Tested Negative | Tested Positive |
| Predicted Label | Tested Negative | 40 | 15 |
| Tested Positive | 8 | 14 |

Table : Confusion matrix for kNN

Neural Network

Neural networks are more flexible and can be used with both regression and classification problems. This approach is good for the nonlinear dataset with a large number of inputs, and it can work with any number of inputs and layers.

The following figure visualizes the computed neural network. Our model has 4 units in its hidden layer.



Then we can predict the values for the test set and calculate the test error rate by comparing the estimated labels yielded from the NN to the actual label in the test samplings. This table indicates that the accuracy rate is 74.03% based on the fact that 20 cases of the 77 test samples are misclassified, and the remaining 57 cases are correctly classified.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Observed Label | |
| Tested Negative | Tested Positive |
| Predicted Label | Tested Negative | 38 | 10 |
| Tested Positive | 10 | 19 |

Table : Confusion matrix for NN

**Conclusion**

Comparing the six approaches usedd, Random forest and Decision Tree are more accurate than other methods to predict Type 2 diabetes with genetic factors.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Method | LDA | QDA | RF | DT | kNN | NN |
| Accuracy | 77% | 74% | 79% | 79% | 70.13% | 74.03% |

**Resources**

Baier, L., & Hanson, R. (2004, May 01). Genetic Studies of the Etiology of Type 2 Diabetes

Pima Indians. Retrieved May 06, 2020, from

https://diabetes.diabetesjournals.org/content/53/5/1181

**Appendix: Code and Output**

LDA & QDA

library(HDtest)

library(MASS)

diabetes.pos=diabetes[diabetes$class=="tested\_positive",]

covar.pos=var(diabetes.pos[,1:8])

diabetes.neg=diabetes[diabetes$class=="tested\_negative",]

covar.neg=var(diabetes.neg[,1:8])

test=testCov(covar.pos, covar.neg, method='HD')

ind=sample(1:nrow(diabetes), nrow(diabetes)/2, replace=FALSE)

lda.fit=lda(class~., data=diabetes, subset=ind)

lda.pred=predict(lda.fit, diabetes.test)

table(lda.pred$class, diabetes.test$class)

lda.acc=mean(lda.pred$class==diabetes.test$class)

qda.fit=qda(class~., data=diabetes, subset=ind)

qda.pred=predict(qda.fit, diabetes.test)

table(qda.pred$class, diabetes.test$class)

qda.acc=mean(qda.pred$class==diabetes.test$class)

Decision Tree

library(ISLR)

library(readxl)

library(rpart)

library(rpart.plot)

set.seed(51)

diabetes <- read\_excel("diabetes.xlsx")

head(diabetes)

summary(diabetes)

set.seed(1)

dia <- sample(1:nrow(diabetes))

head(dia)

diabetes.n <- diabetes[dia,]

head(diabetes.n)

create.train.test <- function(data, size = 0.8, train = TRUE) {

n.row = nrow(data)

t.row = size \* n.row

train.sample = 1: t.row

if (train == TRUE) {

return (data[train.sample, ])

} else {

return (data[-train.sample, ])

}

}

set.seed(12)

train.data <- create.train.test(diabetes.n, 0.8, train = TRUE)

test.data <- create.train.test(diabetes.n, 0.8, train = FALSE)

fit.model <- rpart(class~., data = train.data, method = 'class')

rpart.plot(fit.model, extra = 106)

rsq.rpart(fit.model)[1]

plotcp(fit.model)

model.predict <-predict(fit.model, test.data, type = 'class')

table.matr <- table(test.data$class, model.predict)

table.matr

Test.accuracy <- sum(diag(table.matr)) / sum(table.matr)

print(paste('Accuracy for test', round(Test.accuracy,4)))

#TUNING THE MODEL

control <- rpart.control(minsplit = 9, minbucket = round(9/3),maxdepth = 4,cp = 0)

model.tune.fit <- rpart(class~., data = train.data, method = 'class', control = control)

rpart.plot(model.tune.fit)

model.tune.predict <-predict(model.tune.fit, test.data, type = 'class')

tune.table.matr <- table(test.data$class, model.tune.predict)

tune.table.matr

tune.Test.accuracy <- sum(diag(tune.table.matr)) / sum(tune.table.matr)

print(paste('Accuracy for test', round(tune.Test.accuracy,4)))

Random Forest

library(randomForest)

library(caret)

set.seed(21)

diabetes.samples <- sample(1:nrow(diabetes.n), nrow(diabetes.n) \*0.8, replace = FALSE)

rf.train <- diabetes.n[diabetes.samples, ]

rf.test <- diabetes.n[-diabetes.samples, ]

rf.fit <- randomForest(as.factor(class) ~ ., data = rf.train)

(rf.fit)

prediction <- predict(rf.fit, rf.test)

conf.table<- table(rf.test$class, prediction)

conf.table

sum(diag(conf.table)) / sum(conf.table)

kNN and NN

library(readr)

diabetes <- read\_csv("diabetes.csv")

ran <- sample(1:nrow(diabetes), 0.9 \* nrow(diabetes))

##the normalization function is created

nor <-function(x) { (x -min(x))/(max(x)-min(x)) }

##Run nomalization on first 4 coulumns of dataset because they are the predictors

diabetes\_norm <- as.data.frame(lapply(diabetes [,c(1,2,3,4,5,6,7,8)], nor))

summary(diabetes\_norm)

diabetes\_train <- diabetes\_norm[ran,]

diabetes\_test <- diabetes\_norm[-ran,]

diabetes\_target\_category=diabetes$class[ran]

diabetes\_test\_category=diabetes$class[-ran]

##load the package class

library(class)

i=1

k.optm=1

for (i in 1:27){

knn.mod <- knn(train=diabetes\_train, test=diabetes\_test, cl=diabetes\_target\_category, k=i)

k.optm[i] <- 100 \* sum(diabetes\_test\_category == knn.mod)/NROW(diabetes\_test\_category)

k=i

cat(k,'=',k.optm[i],'

')

}

plot(k.optm, type="b", xlab="K- Value",ylab="Accuracy level")

pr <- knn(diabetes\_train,diabetes\_test,cl=diabetes\_target\_category,k=10)

##create confusion matrix

tab <- table(pr,diabetes\_test\_category)

tab

accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) \* 100}

accuracy(tab)

################################

install.packages("neuralnet")

library(neuralnet)

library(nnet)

diabetes$class=1\*(diabetes$class=='tested\_positive')

diabetes\_target\_category=diabetes$class[ran]

diabetes\_test\_category=diabetes$class[-ran]

diabetes\_train=cbind(diabetes\_train,diabetes\_target\_category)

nn <- neuralnet(diabetes\_target\_category ~ ., data=diabetes\_train, hidden=c(4), linear.output=FALSE, threshold=0.01)

plot(nn)

nn.results <- compute(nn, diabetes\_test)

results <- data.frame(actual = diabetes\_test\_category, prediction = nn.results$net.result)

roundedresults<-sapply(results,round,digits=0)

roundedresultsdf=data.frame(roundedresults)

attach(roundedresultsdf)

tab2=table(actual,prediction)

accuracy(tab2)