**STAT 652 – STATISTICAL LEARNING**

**Project Report**

***Name: BUI LE LINH***

***Student ID: 301298917***

Contents

[1. Introduction 1](#_Toc497995516)

[2. Dataset 1](#_Toc497995517)

[3. Methods 1](#_Toc497995518)

[3.1. Data Exploration and Pre-processing 1](#_Toc497995519)

[3.2. Model Evaluation Criteria 3](#_Toc497995520)

[3.3. Statistical Modelling 3](#_Toc497995521)

[4. Results 3](#_Toc497995522)

[4.1. Summary of Model Performance 3](#_Toc497995523)

[4.2. Model Interpretation 5](#_Toc497995524)

[4.3. Model Checking 5](#_Toc497995525)

[5. Conclusion 5](#_Toc497995526)

[REFERENCES 6](#_Toc497995527)

[APPENDIX – R CODE 6](#_Toc497995528)

# Introduction

The Pima Indians of Arizona have the highest reported prevalence of diabetes of any population in the world [1]. In this analysis, we aim to predict whether a person is tested positive for type 2 diabetes based on several clinical measurements. In addition, we also try to gain insight on the underlying data generating process in this dataset. For that purpose, we will use the Pima Indians Diabetes dataset, which is described in more details in section 2. Section 3 explains methodologies used in our analysis. We interpret and discuss the results in Section 4. In Section 5, We summarize our findings and limitations.

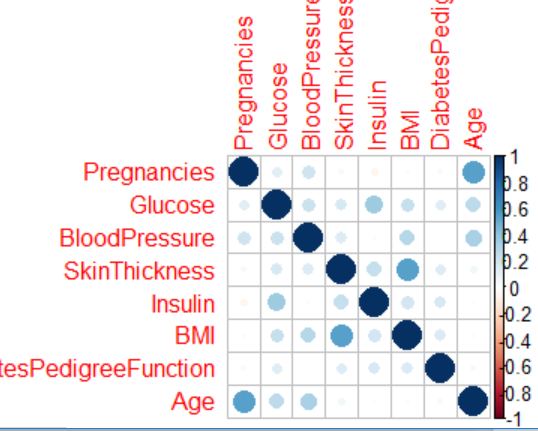
# Dataset

The Pima Indians Diabetes dataset contains diagnostics information of 768 females at least 21 years old and of Pima Indian heritage. Full description of the dataset is available on UCI Machine Learning Repository website [2]. A summary of variables in the dataset is described below:

* *Pregnancies*: Number of times pregnant
* Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
* *BloodPressure*: Diastolic blood pressure (mm Hg)
* *SkinThickness*: Triceps skin fold thickness (mm)
* *Insulin*: 2-Hour serum insulin (mu U/ml)
* *BMI*: Body mass index (weight in kg/(height in m)^2)
* *DiabetesPedigreeFunction*: Diabetes pedigree function (a synthesis of the diabetes mellitus history in relatives and the genetic relationship of those relatives to the subject)
* *Age*: Age (in years)
* *Outcome*: Whether a subject is tested positive for type 2 diabetes (0-No or 1-Yes)

# Methods

## Data Exploration and Pre-processing

We look at several summary statistics to understand the data ranges and check missing vales. *Outcome* only 34.9% is Class 1 (Yes), indicating that the dataset is imbalance. *Glucose*, *BloodPressure*, *SkinThickness*, *Insulin* and *BMI* all contains zero values, which are invalid for these clinical measurements. These values should be treated as missing values. We could simply remove observations with missing values, but that would eliminate about one third of our data. Instead, we will perform imputation for these missing values using median of corresponding variables.

Examining correlation among predictors, we can see a few pairs with high correlation: *Age* and *Pregnancies*, *BMI* and *SkinThickness*, as shown in Figure 3.1.1:

Figure 3.1.1 – Correlation among predictors

For each predictor, we look at the data distribution's differences between the two *Outcome* classes:

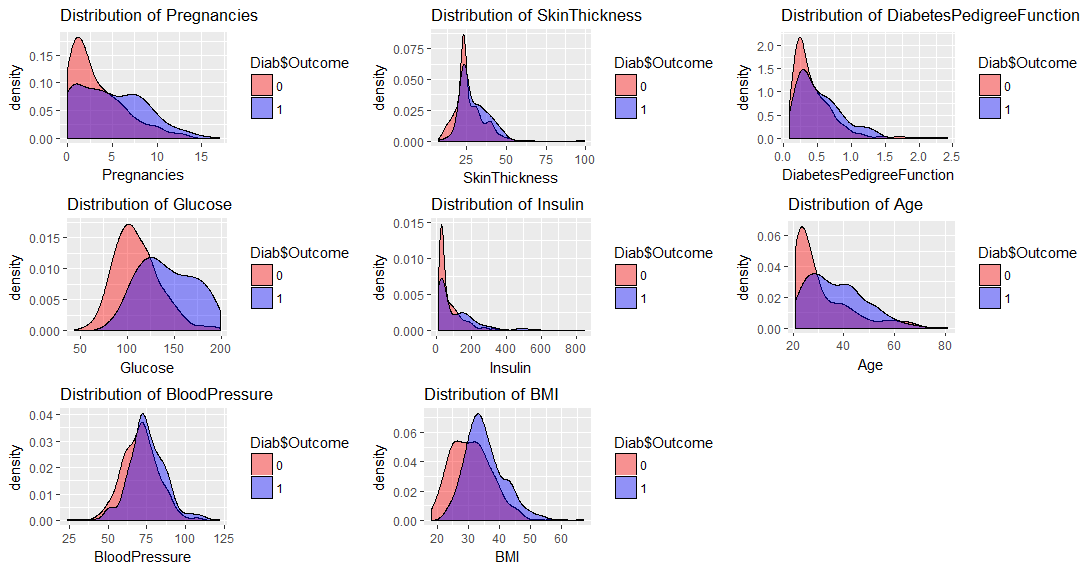


Figure 3.1.2 – Approximated density function for each predictor per *Outcome* class

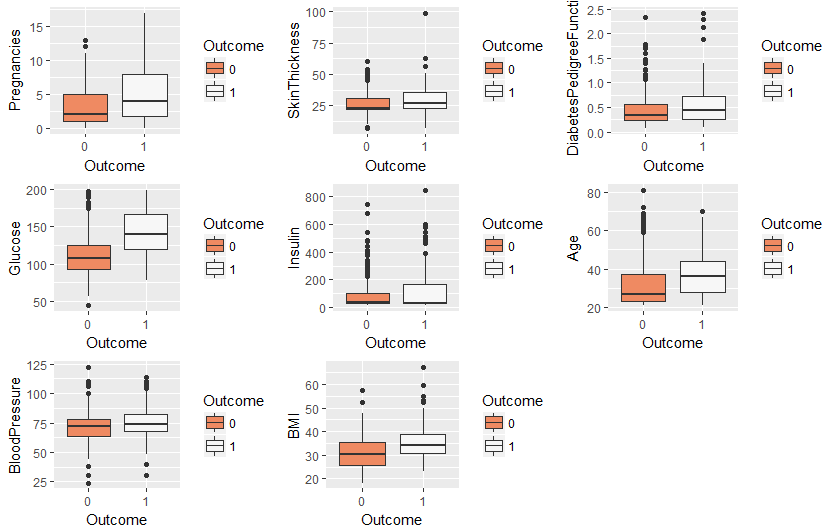


Figure 3.1.3 – Box-plot for each predictor per *Outcome* class

For each predictor, there does not seem to have noticeable differences in variances between *Outcome* classes, while the mean in class 1 might be higher than in class 0.

## Model Evaluation Criteria

To evaluate how well a model can predict whether a person is tested positive for type 2 diabetes, we split the dataset into 80% train set and 20% test set using stratified sampling to maintain class ratio in these datasets. We train different statistical models using train set and evaluate the models' performances on the hold-out test set. Since the dataset is imbalance, simple accuracy or misclassification rate is not a suitable evaluation criterion. There are several evaluation criteria we can use for imbalance dataset, such as Kappa, f-score, true positive and true negative rates, and Area Under the Curve (AUC) [2]. In this analysis, we will use AUC as our evaluation criteria.

## Statistical Modelling

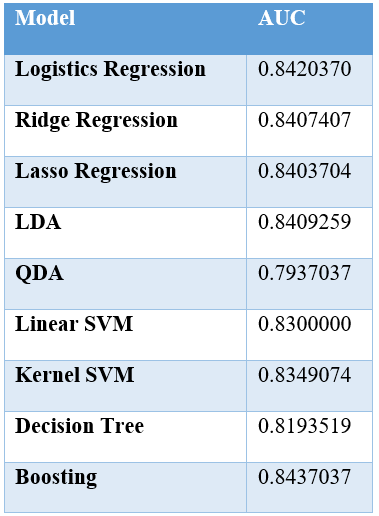
We fit various statistical learning models to predict the *Outcome* variable based on predictors in the dataset. If a model has a hyperparameter, the hyperparameter will be chosen using 10-fold cross validation.

* **Logistic regression** models the conditional probability of *Outcome* being 1, given the predictors.
* **Ridge logistic regression** is a regularized version of logistic regression with L2 shrinkage penalty.
* **Lasso logistic regression** is another regularized version of logistic regression with L1 shrinkage penalty.
* **Linear Discriminant Analysis** (LDA) first models the conditional probability of predictors given the response variable *Outcome* and then use Bayes’ theorem to estimate the conditional probability of *Outcome* given the predictors. It assumes that the observations within each class are drawn from a multivariate Gaussian distribution with a class specific mean vector and a covariance matrix that is common to all classes.
* **Quadric Discriminant Analysis** (QDA) is like Linear Discriminant Analysis but it assumes that each class has its own covariance matrix.
* **Decision Tree** consists of a series of splitting rules on predictors to predict *Outcome*.
* **Linear Support Vector Machine** (SVM) aims to find a linear hyper-plane in the original feature space that best separates observations between two classes.
* **Kernel Support Vector Machine** uses a basis function (radial basis function in this case) to enlarge the feature space to capture a non-linear decision boundary in the original feature space.
* **Boosted Tree** involves fitting multiple decision trees sequentially, in which each tree is grown using information from previously grown trees to correct mistakes made by the previous trees.

# Results

## 4.1. Summary of Model Performance

Model performance in terms of Area Under the Curve (AUC) on test set of various statistical learning models are shown below:

* Ridge regression and Lasso regression's performances are a little worse compared to Logistics Regression. This is probably because the decrease in variance is less than the increase in bias when we make the models less flexible using regularization. In this case, such trade-off is more severe for Lasso which tries to yield a simpler model at the expense of higher bias.
* The performance of Linear Discriminant Analysis is close to Logistics Regression. Quadric Discriminant Analysis has worse performance compared to Linear Discriminant Analysis, suggesting that the quadratic form assumed by QDA may not capture the true relationship compared to the linear forms assumed by LDA and logistic regression, leading to high variance.
* Kernel SVM with radial basis function has higher performance compared to Linear SVM, suggesting that the true decision boundary might be non-linear.
* Decision Tree has quite low performance, which is expected because Decision Tree usually does not possess high prediction accuracy.
* Boosting model also has better performance compared to Decision Tree Logistics Regression because boosting technique usually manages to reduce both bias and variance.

Although Boosting has higher AUC compared to Logistic Regression, the difference is very small. In addition, Boosting is difficult to interpret and computationally intensive. Therefore, we select **Logistics Regression as our best model**.

At threshold of 0.5, the Logistics Regression model has the below estimated performance on test set: Accuracy of 79.22%, Sensitivity (True Positive Rate) of 90.00%, and Specificity (1-False Positive Rate) of 59.26%.

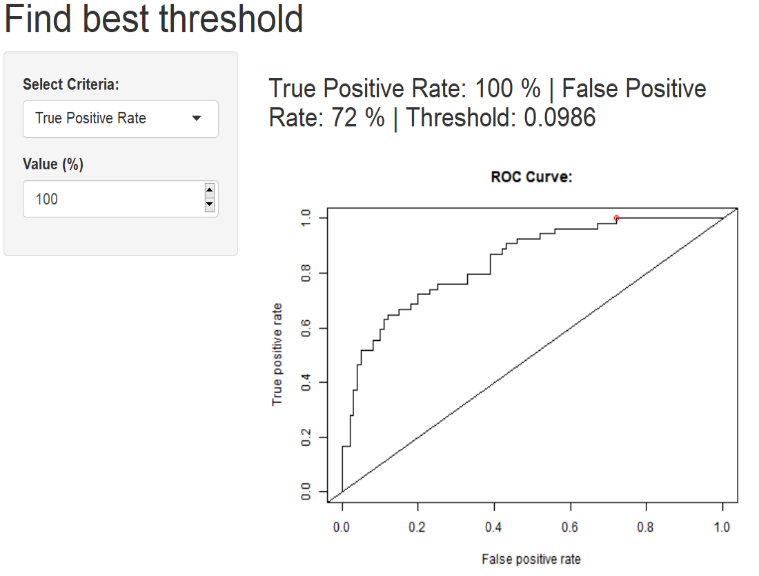
Receiver Operating Characteristic (ROC) curve plots true positive rate again false positive rate at different thresholds. We implement a simple interactive dashboard with R Shiny to find the best threshold given a specific True Positive or False Positive Rate, as illustrated in Figure 4.1.1. Looking at the ROC curve, we can see that Logistic Regression's performance in predicting diabetes is moderate: To achieve 100% True Positive Rate, we would have to accept 72% False Positive Rate, while to achieve 0% False Positive Rate, we would have to accept 17% True Positive Rate.

Figure 4.1.1 – ROC curve interactive dashboard

## 4.2. Model Interpretation

We fit the Logistics Regression model using all available data:

*Pregnancies*, *Glucose*, *BMI*, *DiabetesPedigreeFunction* are important in the model:

* Increase *Pregnancies* by 1 would increase the estimated odd of having diabetes by 1.13 times, given other predictors remain the same. 95% confidence interval for odd ratio is (1.06, 1.21)
* Increase *Glucose* by 1 would increase the estimated odd of having diabetes by 1.04 times, given other predictors remain the same. 95% confidence interval for odd ratio is (1.03, 1.05)
* Increase *BMI* by 1 would increase the estimated odd of having diabetes by 1.10 times, given other predictors remain the same. 95% confidence interval for odd ratio is (1.07, 1.14)
* Increase *DiabetesPedigreeFunction* by 1 would increase the estimated odd of having diabetes by 2.50 times, given other predictors remain the same. 95% confidence interval for odd ratio is (1.39, 4.50)

## 4.3. Model Checking

Goodness-of-fit (GOF) statistics are often computed as more objective measures of the overall fit of a model. With model containing continuous variables, the Hosmer and Lemeshow test [3] would be an appropriate Goodness of Fit test. Hosmer and Lemeshow test statistics has p-value = 0.182 > 0.05, suggesting the model fit is not bad. Look at standardized residuals, we can see there are 38 potential outliers (0.04%) with standardized residual beyond 2 and 10 extreme outliers (0.01%) with standardized residual beyond 3. We also identify 62 potential influential observation using several criteria, including Leverage and Cook’s Distance [4].

# Conclusion

In summary, we have analyzed the Pima Indians Diabetes dataset to gain insight on underlying data distribution. We apply different statistical learning techniques to predict whether a person is tested positive for type 2 diabetes based on clinical measurements. There is no major difference in model performance in terms of Area Under the Curve (AUC) among these models except for Quadric Discriminant Analysis which has very low performance. Boosting marginally outperforms Logistics Regression, but Logistics Regression model is much more interpretable. We implement a simple interactive dashboard to understand the trade-off between True Positive Rate and False Positive Rate of the Logistics Regression model. The Logistics Regression model is also used to interpret the relationship between predictors and the response variable *Outcome*. *Pregnancies*, *Glucose*, *BMI*, *DiabetesPedigreeFunction* are important in the model and the all increase the estimated odd of having diabetes given other predictors remain the same. The model fit is not bad, but we do have 10 extreme outliers and 62 potential influential points in the dataset.

Due to lack of understanding on how the dataset was collected, we are unable to investigate outliers. In addition, it would make sense to experiment with variable transformation techniques (basis function, step function) on predictors in future research. If the goal is mainly prediction, it might be a good idea to perform model stacking to improve on prediction.

# REFERENCES

1. *Leslie J. Baier, and Robert L. Hanson (2004),* Genetic Studies of the Etiology of Type 2 Diabetes in Pima Indians*, American Diabetes Association*
2. *László A. Jeni, Jeffrey F. Cohn, and Fernando D. Torre1 (2015),* Facing Imbalanced Data Recommendations for the Use of Performance Metrics*, National Center for Biotechnology Information*
3. *Christopher R. Bilder and Thomas M. Loughin*, Analysis of Categorical Variable with R, *Page 293*.
4. *Christopher R. Bilder and Thomas M. Loughin*, Analysis of Categorical Variable with R, *Page 296*.

# APPENDIX – R CODE

**R version:** 3.4.0

**Packages:**

* corrplot
* ggplot2
* caret
* glmnet
* MASS
* rpart
* e1071
* gbm
* shinyApp

**External codes:**

* multiplot.R: <http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/>
* AllGOFTests.R: <http://www.chrisbilder.com/categorical/Chapter5/AllGOFTests.R>
* glmDiagnostics.R: <http://www.chrisbilder.com/categorical/Chapter5/AllGOFTests.R>

**List of R code files:**

* project.Rmd: The main project R markdown file to generate above analyses and charts. The time to knit the code should be about 2-3 minutes. In addition to the R packages, the code also relies on three external R code files listed above for plotting and model diagnosis.
* shinnyApp.R: The main app to run interactive dashboard shown in Figure 4.1.1
  + shinnyApp/ui.R: R file to define UI for the interactive dashboard
  + shinnyApp/server.R: R file to define logic for the interactive dashboard

**project.R**

**#Data Pre-processing and Exploration#**

```{r}

#Read data

Diab <- read.csv("pima-diabetes.csv")

summary(Diab)

```

```{r}

#Data Pre-processing

Diab$Glucose[Diab$Glucose==0] <- median(Diab$Glucose)

Diab$BloodPressure[Diab$BloodPressure==0] <- median(Diab$BloodPressure)

Diab$Insulin[Diab$Insulin==0] <- median(Diab$Insulin)

Diab$BMI[Diab$BMI==0] <- median(Diab$BMI)

Diab$SkinThickness[Diab$SkinThickness==0] <- median(Diab$SkinThickness)

Diab2 <- Diab #data used for boosting later since gbm cannot handle factor variable

Diab$Outcome <- as.factor(Diab$Outcome)

```

```{r}

#Correlation Plot

library(corrplot)

corr <- cor(Diab[, -9])

corrplot(corr)

```

```{r}

#Density Function for each predictor ~ Outcome

library(ggplot2)

source('multiplot.R')

p1 <- ggplot(Diab,aes(x=Pregnancies,fill=Diab$Outcome))+geom\_density(alpha=0.4)+scale\_fill\_manual(values=c("red", "blue"))+labs(title="Distribution of Pregnancies")

p2 <- ggplot(Diab,aes(x=Glucose,fill=Diab$Outcome))+geom\_density(alpha=0.4)+scale\_fill\_manual(values=c("red", "blue"))+labs(title="Distribution of Glucose")

p3 <- ggplot(Diab,aes(x=BloodPressure,fill=Diab$Outcome))+geom\_density(alpha=0.4)+scale\_fill\_manual(values=c("red", "blue"))+labs(title="Distribution of BloodPressure")

p4 <- ggplot(Diab,aes(x=SkinThickness,fill=Diab$Outcome))+geom\_density(alpha=0.4)+scale\_fill\_manual(values=c("red", "blue"))+labs(title="Distribution of SkinThickness")

p5 <- ggplot(Diab,aes(x=Insulin,fill=Diab$Outcome))+geom\_density(alpha=0.4)+scale\_fill\_manual(values=c("red", "blue"))+labs(title="Distribution of Insulin")

p6 <- ggplot(Diab,aes(x=BMI,fill=Diab$Outcome))+geom\_density(alpha=0.4)+scale\_fill\_manual(values=c("red", "blue"))+labs(title="Distribution of BMI")

p7 <- ggplot(Diab,aes(x=DiabetesPedigreeFunction,fill=Diab$Outcome))+geom\_density(alpha=0.4)+scale\_fill\_manual(values=c("red", "blue"))+labs(title="Distribution of DiabetesPedigreeFunction")

p8 <- ggplot(Diab,aes(x=Age,fill=Diab$Outcome))+geom\_density(alpha=0.4)+scale\_fill\_manual(values=c("red", "blue"))+labs(title="Distribution of Age")

multiplot(p1, p2, p3, p4, p5, p6, p7, p8, cols=2)

```

```{r}

#Box-plot for each predictor ~ Outcome

p1 <- ggplot(Diab,aes(x=Outcome,y=Pregnancies,fill=Outcome))+geom\_boxplot()+scale\_fill\_brewer(palette="RdBu")

p2 <- ggplot(Diab,aes(x=Outcome,y=Glucose,fill=Outcome))+geom\_boxplot()+scale\_fill\_brewer(palette="RdBu")

p3 <- ggplot(Diab,aes(x=Outcome,y=BloodPressure,fill=Outcome))+geom\_boxplot()+scale\_fill\_brewer(palette="RdBu")

p4 <- ggplot(Diab,aes(x=Outcome,y=SkinThickness,fill=Outcome))+geom\_boxplot()+scale\_fill\_brewer(palette="RdBu")

p5 <- ggplot(Diab,aes(x=Outcome,y=Insulin,fill=Outcome))+geom\_boxplot()+scale\_fill\_brewer(palette="RdBu")

p6 <- ggplot(Diab,aes(x=Outcome,y=BMI,fill=Outcome))+geom\_boxplot()+scale\_fill\_brewer(palette="RdBu")

p7 <- ggplot(Diab,aes(x=Outcome,y=DiabetesPedigreeFunction,fill=Outcome))+geom\_boxplot()+scale\_fill\_brewer(palette="RdBu")

p8 <- ggplot(Diab,aes(x=Outcome,y=Age,fill=Outcome))+geom\_boxplot()+scale\_fill\_brewer(palette="RdBu")

multiplot(p1, p2, p3, p4, p5, p6, p7, p8, cols=2)

```

**#Evaluation Criteria Definition#**

```{r}

#Train-Test Split

set.seed(13)

library(caret)

testset <- createDataPartition(Diab$Outcome, p = 0.2, list = FALSE)

Diab.test <- Diab[testset,]

Diab.train <- Diab[-testset,]

```

```{r}

#AUC

library(ROCR)

caculate\_auc <- function(probs) {

pred <- prediction(probs, Diab.test$Outcome)

auc.perf = performance(pred, measure = "auc")

return (auc.perf@y.values[[1]])

}

```

#Statistical Modelling#

```{r}

#Logistics Regression

mod.lm <- glm(Outcome ~ ., data = Diab.train, family = binomial(link='logit'))

pred.probs = predict(mod.lm, Diab.test, type="response")

auc.lm <- caculate\_auc(pred.probs)

```

```{r}

#Ridge logistics regression

library(glmnet)

X.train <- model.matrix(Outcome ~ ., data = Diab.train)

X.test <- model.matrix(Outcome ~ ., data = Diab.test)

lambdas <- 10^{seq(from=-2,to=5,length=100)}

cv.ridge <- cv.glmnet(X.train, Diab.train$Outcome, alpha = 0, lambda = lambdas, standardize=TRUE, family="binomial")

lambda.best <- cv.ridge$lambda.min

pred.probs <- predict(cv.ridge, s = lambda.best, newx = X.test, type="response")

auc.ridge <- caculate\_auc(pred.probs)

```

```{r}

#Lasso logistics regression

lambdas <- 10^{seq(from=-2,to=5,length=100)}

cv.lasso <- cv.glmnet(X.train, Diab.train$Outcome, alpha = 1, lambda = lambdas, standardize=TRUE, family="binomial")

lambda.best <- cv.lasso$lambda.min

pred.probs <- predict(cv.lasso, s = lambda.best, newx = X.test, type="response")

auc.lasso <- caculate\_auc(pred.probs)

```

```{r}

#Linear Discriminant Analysis

library(MASS)

mod.lda <- lda(Outcome ~ ., data = Diab.train)

pred.probs = predict(mod.lda, Diab.test, type="response")$posterior[ ,2]

auc.lda <-caculate\_auc(pred.probs)

```

```{r}

#Quadric Discriminant Analysis

mod.qda <- qda(Outcome ~ ., data = Diab.train)

pred.probs = predict(mod.qda, Diab.test, type="response")$posterior[ ,2]

auc.qda <- caculate\_auc(pred.probs)

```

```{r}

#Decision Tree

library(rpart)

mod.tree <- rpart(Outcome ~ ., data = Diab.train, method = "class")

pred.probs <- predict(mod.tree, Diab.test, type="prob")[,2]

auc.tree <-caculate\_auc(pred.probs)

#plot(mod.tree)

#text(mod.tree)

```

```{r}

#Linear Support Vector Machine

library(e1071)

tune.linear <- tune(svm, Outcome ~ ., data = Diab.train, kernel = "linear", probability=TRUE, scale=TRUE,

ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)))

svm.linear <- tune.linear$best.model

pred.probs <- predict(svm.linear, Diab.test, type="response", probability=TRUE)

pred.probs <- attr(pred.probs,"probabilities")[,2]

auc.svm.linear <- caculate\_auc(pred.probs)

```

```{r}

#Kernel Support Vector Machine

tune.radial <- tune(svm, Outcome ~ ., data = Diab.train, kernel = "radial", probability=TRUE, scale=TRUE,

ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100),

gamma = c(0.001, 0.01, 1, 5, 10, 100)))

svm.radial <- tune.radial$best.model

pred.probs <- predict(svm.radial, Diab.test, type="response", probability=TRUE)

pred.probs <- attr(pred.probs,"probabilities")[,2]

auc.svm.radial <-caculate\_auc(pred.probs)

```

```{r}

#Boosting

library(gbm)

Diab2.test <- Diab2[testset,]

Diab2.train <- Diab2[-testset,]

n.trees.list <- c(100, 500, 1000, 2000, 5000)

cv.error.list <- rep(NA, length(n.trees.list))

for (i in 1:length(n.trees.list)) {

mod.boost <- gbm(Outcome ~ ., data = Diab2.train, distribution = "bernoulli",

cv.folds = 10, n.trees = n.trees.list[i], shrinkage = 0.01)

cv.error <- mod.boost$cv.error[gbm.perf(mod.boost, plot.it = FALSE, method="cv")]

cv.error.list[i] <- cv.error

}

n.trees.best <- n.trees.list[which.min(cv.error.list)]

pred.probs = predict(mod.boost, Diab2.test, n.trees = n.trees.best, type="response")

auc.boosting <- caculate\_auc(pred.probs)

```

**#Model Performance Summary#**

```{r}

#Summary of model performance

models <-c("Logistics Regression", "Ridge Regression", "Lasso Regression",

"LDA", "QDA",

"Linear SVM", "Kernel SVM", "Decision Tree", "Boosting")

auc.list <-c(auc.lm, auc.ridge, auc.lasso, auc.lda, auc.qda, auc.svm.linear, auc.svm.radial, auc.tree, auc.boosting)

auc.df <- data.frame(models, auc.list)

names(auc.df) <- c("Model", "AUC")

print(auc.df)

```

```{r}

#Confusion Matrix for the Logistics Regression model:

pred.probs = predict(mod.lm, Diab.test, type="response")

pred.label <- ifelse(pred.probs > 0.5, 1, 0)

xtab <- table(pred.label, Diab.test$Outcome)

confusionMatrix(xtab)

```

```{r}

#ROC Curve for the Logistics Regression model:

pred <- prediction(pred.probs, Diab.test$Outcome)

perf <- performance(pred, "tpr", "fpr")

plot(perf)

abline(a=0, b= 1)

title(paste("Logistic Regression - AUC:",round(auc.lm,4)))

```

```{r}

# Save ROC data for interactive dashboard (optional)

save(perf, file = "shinyApp/perf.RData")

```

**#Model Interpretation and Diagnosis#**

```{r}

#Model Interpretation

mod.fit <- glm(Outcome ~ ., data = Diab, family = binomial(link='logit'))

summary(mod.fit)

```

```{r}

#Goodness of fit

source("AllGOFTests.R")

HL <- HLTest(obj=mod.fit, g = 100)

HL

```

```{r}

#Outliers

s.res = rstandard(mod.fit, type="pearson")

s.res.beyond2 <- which(s.res > 2 | s.res < -2)

paste("Number of observations with standardized residual beyond 2:", length(s.res.beyond2))

s.res.beyond3 <- which(s.res > 3 | s.res < -3)

paste("Number of observations with standardized residual beyond 3:", length(s.res.beyond3))

```

```{r}

#Influence

source("glmDiagnostics.R")

influentials <- glmInflDiag(mod.fit = mod.fit, print.output=FALSE, which.plots = FALSE)

paste("Number of influential points:", nrow(influentials))

```

**shinnyApp.R**

library(shiny)

runApp("shinyApp")

**shinnyApp/ui.R**

library(shiny)

# Define UI for miles per gallon application

shinyUI(pageWithSidebar(

# Application title

headerPanel("Find best threshold"),

sidebarPanel(

selectInput("variable", "Select Criteria:",

list("True Positive Rate" = "tpr",

"False Positive Rate" = "fpr")),

numericInput("rate", "Value (%)", 100, min = 0, max = 100, step = NA, width = NULL)

),

mainPanel(

h3(textOutput("caption")),

plotOutput("mpgPlot")

)

))

**shinnyApp/server.R**

library(shiny)

#Read ROC data

load("perf.RData")

thresholds <- data.frame(threshold=perf@alpha.values[[1]], fpr=perf@x.values[[1]],

tpr=perf@y.values[[1]])

# function to find best threshold, tpr and fpr based a specific criteria

get\_point <- function(variable, rate) {

if (variable=="fpr") {

thresholds2 <- thresholds[order(thresholds$tpr, decreasing=TRUE),]

thresholds\_filter <- thresholds2[thresholds2$fpr <= rate,]

rownames(thresholds\_filter) <- 1:nrow(thresholds\_filter)

best\_threshold <- thresholds\_filter[1,1]

if (is.infinite(best\_threshold)) {best\_threshold <- 1}

best\_tpr <- thresholds\_filter[1,3]

best\_fpr <- thresholds\_filter[1,2]

return(c(best\_threshold, best\_tpr, best\_fpr))

}

else {

thresholds2 <- thresholds[order(thresholds$fpr, decreasing=FALSE),]

thresholds\_filter <- thresholds2[thresholds2$tpr >= rate,]

rownames(thresholds\_filter) <- 1:nrow(thresholds\_filter)

best\_threshold <- thresholds\_filter[1,1]

if (is.infinite(best\_threshold)) {best\_threshold <- 1}

best\_tpr <- thresholds\_filter[1,3]

best\_fpr <- thresholds\_filter[1,2]

return(c(best\_threshold, best\_tpr, best\_fpr))

}

}

# Define server logic required to plot

shinyServer(function(input, output) {

# get best threshold, tpr and fpr

currentFib <- reactive({ get\_point(input$variable, input$rate/100) })

# Compute the forumla text in a reactive expression

formulaText <- reactive({

text\_threshold <- paste("Threshold: ", round(currentFib()[1], 4))

text\_tpr <- paste("True Positive Rate: ", round(currentFib()[2]\*100,2), "%")

text\_fpr <- paste("False Positive Rate: ", round(currentFib()[3]\*100,2), "%")

return(paste(text\_tpr, text\_fpr, text\_threshold, sep=" | "))

})

# Return the formula text for printing as a caption

output$caption <- renderText({

formulaText()

})

# Generate a plot

output$mpgPlot <- renderPlot({

plot(perf)

abline(a=0, b= 1)

points(x=currentFib()[3], y=currentFib()[2], type="p", pch=1, col="red", cex=1)

title(paste("ROC Curve:"))

})

})