Building a model to predict whether a hospital patient (female) has diabetes based on patient information.

The diabetes.csv dataset consists of the following 9 variables:

1. Pregnancies: the number of pregnancies the patient has experienced in her lifetime
2. Glucose: plasma glucose concentration (2 hour oral test)
3. BloodPressure: diastolic blood pressure (mm Hg)
4. SkinThickness: triceps skin fold thickness (mm)
5. Insulin: 2 hour serum insulin (mu U/ml)
6. BMI: Body mass index (weight in kg/(height in m)^2)
7. DiabetesPedigree: Diabetes pedigree function
8. Age: Age in years
9. Outcome: 0 if the patient does not have diabetes, 1 if the patient has diabetes

Read Data

```
# Load relevant libraries (add here if needed)
library(car)
```

```
## Loading required package: carData
```

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(CombMSC)
```

```
##
## Attaching package: 'CombMSC'

## The following object is masked from 'package:car':
##
##     subsets

## The following object is masked from 'package:stats':
##
##     BIC
```

```
library(bestglm)
```

```
## Loading required package: leaps
```

```
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-2
```

```
library(boot)
```

```
##
## Attaching package: 'boot'

## The following object is masked from 'package:lattice':
##
##     melanoma

## The following object is masked from 'package:car':
##
##     logit
```

```
library(rpart)
library(rpart.plot)
library(leaps)
library(MASS)

# Ensure that the sampling type is correct
RNGkind(sample.kind="Rejection")

# Set seed
set.seed(0)

# Read the data
dataFull = read.csv("diabetes.csv", header=TRUE)

# Split data for training and testing
testRows = sample(nrow(dataFull), 0.2*nrow(dataFull))
dataTest = dataFull[testRows, ]
dataTrain = dataFull[-testRows, ]
```

Treating all variables as quantitative variables.

Full Model

```
# Code to fit logistic regression model (logit link function) model and display summary
model1 <- glm(Outcome~., family=binomial, data=dataTrain)
summary(model1) #summary table
```

```
##
## Call:
## glm(formula = Outcome ~ ., family = binomial, data = dataTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6053  -0.6538  -0.3791   0.6462   2.2566
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -9.9913904  0.8282718 -12.063  < 2e-16 ***
## Pregnancies       0.0500114  0.0384417   1.301  0.19327
## Glucose           0.0397656  0.0039708  10.015  < 2e-16 ***
## BloodPressure     0.0111516  0.0082437   1.353  0.17614
## SkinThickness     0.0198691  0.0114997   1.728  0.08402 .
## Insulin          -0.0009449  0.0009667  -0.978  0.32832
## BMI               0.0367133  0.0173341   2.118  0.03418 *
## DiabetesPedigree  0.8914450  0.2992210   2.979  0.00289 **
## Age               0.0335841  0.0128175   2.620  0.00879 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1048.44  on 827  degrees of freedom
## Residual deviance:  734.37  on 819  degrees of freedom
## AIC: 752.37
##
## Number of Fisher Scoring iterations: 5
```

Coefficients that are statistically significant at the 95% confidence level

```
p_values = summary(model1)$coef[,4]
alpha = 0.05
p_values <= alpha
```

```
##      (Intercept)      Pregnancies          Glucose    BloodPressure
##             TRUE            FALSE             TRUE            FALSE
##    SkinThickness          Insulin              BMI DiabetesPedigree
##            FALSE            FALSE             TRUE             TRUE
##              Age
##             TRUE
```

List of significant coefficients: Intercept, Glucose, BMI, DiabetesPedigree, and Age

Explanation: They have very low p-values that are less than the alpha value of 0.05 (95% confidence interval.

```
odd <-exp(0.0335841)
odd
```

```
## [1] 1.034154
```

The estimated coefficient of age which is 0.0335841 means that for one unit increase in the age of the patient,the log odds of pts having diabetes increases by 0.0335841, holding all other variables constant. Holding other predictors constant, an increase of one year of Age will increase the odds of having diabetes by 3.415% (e0.0335841 − 1 = 0.034151207). or one year increase in the age of the patient, the odds of the patient having diabetes increases by a factor of e^(0.0335841) = 1.03154, holding all other variables constant.

```
odd2 <- exp(0.0335841*5)-1
odd2
```

```
## [1] 0.1828426
```

Similarly, holding all other variables constant, an increase in age of 5 years will change the odds of the patient having diabetes by 18%

Deviance test for the overall regression of *model1*. What do you conclude using a significance level of 0.05? Include your reasoning.

```
gstat = model1$null.deviance- deviance(model1)
p_value = 1-pchisq(gstat,length(coef(model1)) -1)
cbind(gstat, p_value )
```

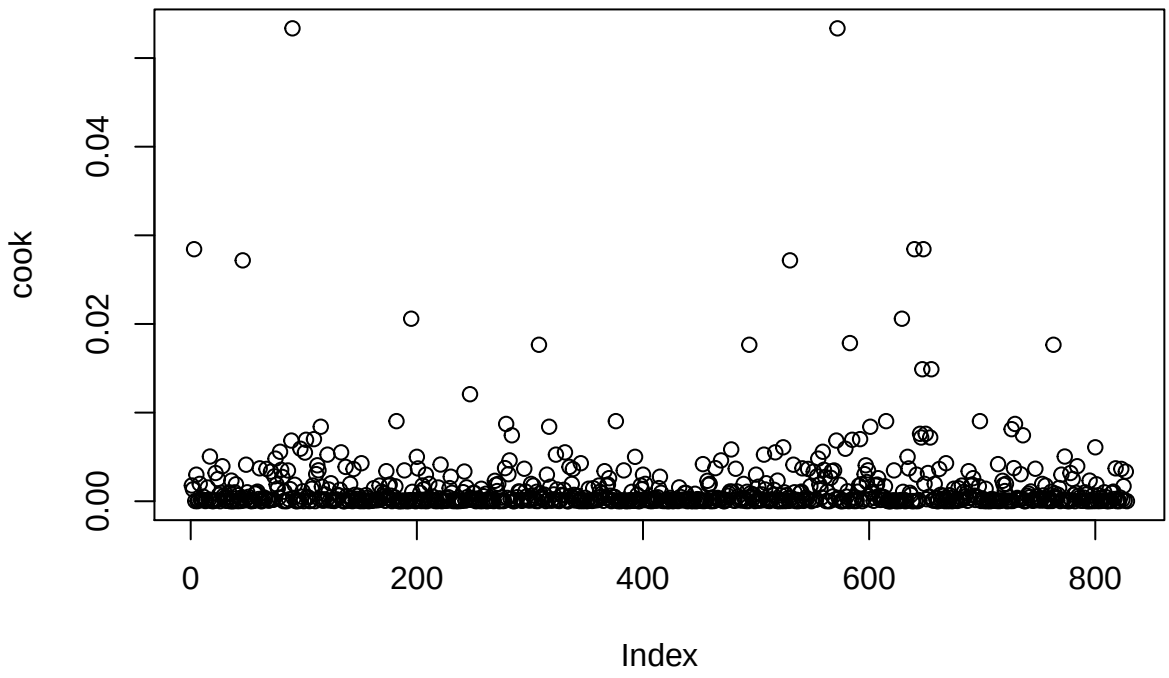
```
##        gstat p_value
## [1,] 314.064       0
```

test statistic=314.064 and p-value=0

The null hypothesis is that the all model coeff except for intercept are 0 and the alternative hypothesis suggests that at least one of the coeff is non-zero. Based on the low p-value of 0 which is less than 0.05, we can reject the null hypothesis and conclude that at least one of coeffs is non-zero and thus the overall model has explanatory power.

Outlier and Overdispersion Detection

```
# Code to calculate and plot Cook's distance
# Calculating Cook's distances
cook=cooks.distance(model1)
# Plotting Cook's distances
plot(cook)
abline(h=1, col="red")
```




```
#Identify outliers
cat("Observation", which(cook>0.004830918), "has a cook's distance that is greater than 0.004830918")
```

```
## Observation 3 17 46 79 89 90 97 101 102 109 115 121 133 182 195 200 247 279 284 308 317 323 331 376
```

```
n = nrow(dataTrain)
outliers <- which(cook > 4/n)
length(outliers)
```

```
## [1] 57
```

A lot of observations (about 57) are showing up as outliers based on 4/n threshold which is 0.004830918.

I will not remove the identified outliers because they might not be true outliers as it is likely for the model to show obs as outliers based on 4/n rule especially when the sample is large. Outliers usually occur in a very low frequency 1-4. They appear to be part of a heavy tail of this model rather than true outliers.

```
# Code for overdispersion calculation
# Calculate overdispersion parameter
model1$deviance/model1$df.res
```

```
## [1] 0.896668
```

The dispersion does not seem to be a problem for this model as it is close to 1 and is not greater than 2.

Variable Selection

A complete/exhaustive search to find the logistic submodel with the smallest AIC

```
# Code to conduct a complete search, fit model2 and display summary

bestAIC <- bestglm(dataTrain, IC="AIC", family = binomial)
```

```
## Morgan-Tatar search since family is non-gaussian.
```

```
bestAIC$BestModel
```

```
##
## Call:  glm(formula = y ~ ., family = family, data = Xi, weights = weights)
##
## Coefficients:
##      (Intercept)           Glucose     SkinThickness               BMI
##         -9.46903           0.03805           0.02037           0.03837
## DiabetesPedigree               Age
##          0.82376           0.04867
##
## Degrees of Freedom: 827 Total (i.e. Null);  822 Residual
## Null Deviance:       1048
## Residual Deviance: 739.1     AIC: 751.1
```

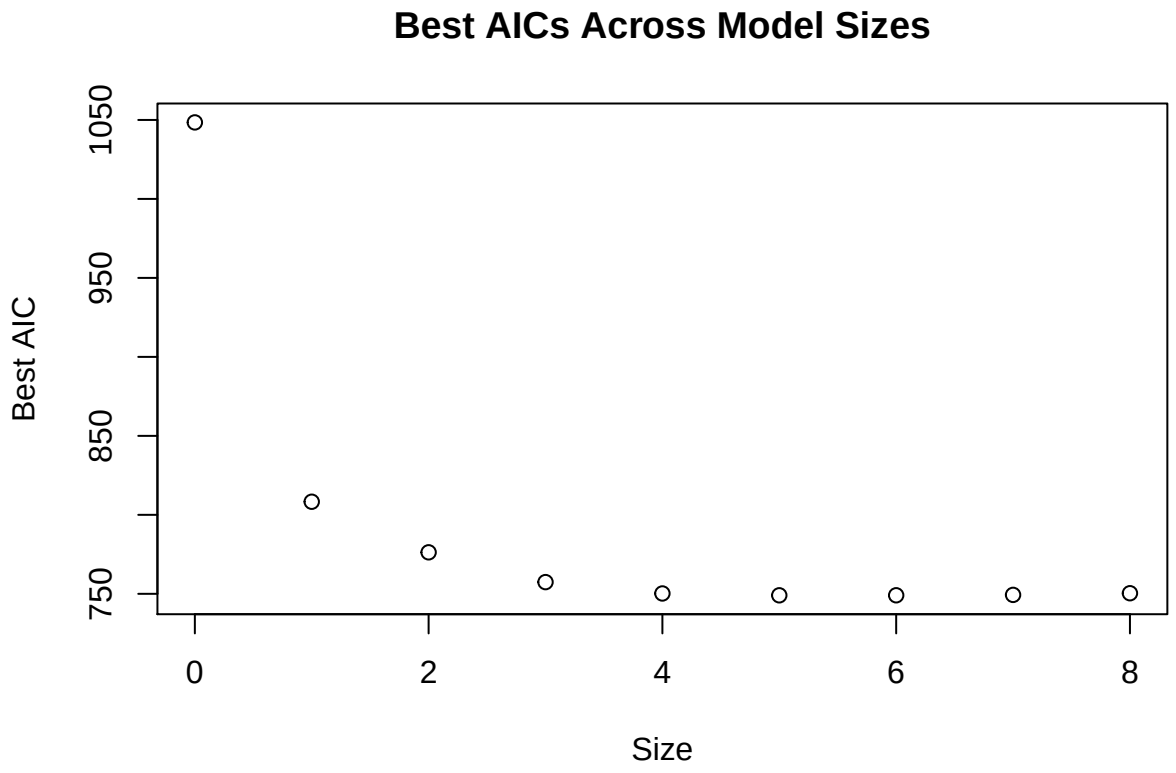
```
#model2 only using the variables from the bestAIC model

model2 <- glm(Outcome~Glucose + SkinThickness + BMI + DiabetesPedigree + Age,data=dataTrain, family="bi
summary(model2)
```

```
##
## Call:
## glm(formula = Outcome ~ Glucose + SkinThickness + BMI + DiabetesPedigree +
##     Age, family = "binomial", data = dataTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7138  -0.6746  -0.3866   0.6437   2.3020
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -9.469025   0.715495 -13.234  < 2e-16 ***
## Glucose           0.038051   0.003470  10.966  < 2e-16 ***
## SkinThickness     0.020373   0.011451   1.779  0.07521 .
## BMI               0.038365   0.016512   2.323  0.02016 *
## DiabetesPedigree  0.823759   0.293219   2.809  0.00496 **
## Age               0.048671   0.009407   5.174 2.29e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 1048.44  on 827  degrees of freedom
## Residual deviance:  739.06  on 822  degrees of freedom
## AIC: 751.06
##
## Number of Fisher Scoring iterations: 5
```

Variables selected by logistic submodel model2 are: Glucose, SkinThickness, BMI, DiabetesPedigree and Age

```
# Code to plot the AIC for the best model of each size
sizes <- cbind(0,1,2,3,4,5,6,7,8)
plot(sizes, bestAIC$Subsets$AIC, xlab="Size", ylab="Best AIC",
main="Best AICs Across Model Sizes")
```



This plot displays the AIC for the best model of each size (ranging from 0-8) produced by the best subset selection. As expected, the best model of size 0 (null model) has by far the largest AIC. AIC decreases as the number of variables increases; however, from the 4-variable model on, there is little improvement in AIC. The best models of sizes 4-8 have similar AICs, being the 5-variable model (best model), the model with the minimum AIC.

```
# Code to conduct forward stepwise, fit model3 and display summary
# Create the minimum model with only an intercept and use model1 as full model

set.seed(100)
n = nrow(dataTrain)
minmod <- glm(Outcome~1,data = dataTrain,family=binomial)
```

```
#Step forward from the minimum model using BIC as the complexity penalty
model3 = step(minmod, scope = list(lower = minmod, upper = model1), direction = "forward", k=log(n), tra
summary(model3)
```

```
##
## Call:
## glm(formula = Outcome ~ Glucose + Age + BMI + DiabetesPedigree,
##     family = binomial, data = dataTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7099  -0.6644  -0.3880   0.6567   2.2427
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -9.575146   0.711381 -13.460  < 2e-16 ***
## Glucose           0.038049   0.003457  11.008  < 2e-16 ***
## Age               0.051891   0.009241   5.616 1.96e-08 ***
## BMI               0.056487   0.013131   4.302 1.69e-05 ***
## DiabetesPedigree  0.864915   0.289803   2.984  0.00284 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1048.44  on 827  degrees of freedom
## Residual deviance:  742.22  on 823  degrees of freedom
## AIC: 752.22
##
## Number of Fisher Scoring iterations: 5
```

```
#names(which(summary(model3)$coefficients[,4]<.01))
```

Variables selected are Glucose, Age, BMI, DiabetesPedigree

Multicollinearity test on *model1*, *model2*, and *model3*

```
# Code to obtain VIF values
vif(model1)
```

```
##      Pregnancies          Glucose    BloodPressure    SkinThickness
##         1.896881         1.331417         1.181543         1.609046
##          Insulin              BMI DiabetesPedigree              Age
##         1.368060         1.721820         1.024210         1.979145
```

```
cat("VIF Threshold Model1:", max(10, 1/(1-summary(model1)$r.squared)), "\n")
```

```
## VIF Threshold Model1: 10
```

```r
# Code to obtain VIF values
vif(model2)
```

```
##          Glucose    SkinThickness              BMI DiabetesPedigree
##         1.011942         1.604694         1.595671         1.007578
##              Age
##         1.049536
```

```r
cat("VIF Threshold Model2:", max(10, 1/(1-summary(model2)$r.squared)), "\n")
```

```
## VIF Threshold Model2: 10
```

```r
# Code to obtain VIF values
vif(model3)
```

```
##          Glucose              Age              BMI DiabetesPedigree
##         1.010481         1.014190         1.007402         1.001934
```

```r
cat("VIF Threshold Model3:", max(10, 1/(1-summary(model3)$r.squared)), "\n")
```

```
## VIF Threshold Model3: 10
```

Multicollinearity do not seem to be the issue in any of the models because the VIF threshold is 10 for all three models and none of the predictors in any of the model surpass the threshold.

Regularized Regression

```r
##ridge regression with 10-fold cross validation

# Setting the seed
set.seed(0)

#Optimize lambda using cross validation
model4 = cv.glmnet(as.matrix(dataTrain[,-9]), dataTrain[,9],family="binomial",alpha=0,nfolds=10,nlambda=
cat("CV Optimized lambda:\n")
```

```
## CV Optimized lambda:
```

```r
## CV Optimized lambda:
model4$lambda.min
```

```
## [1] 0.05658218
```

Optimal lambda: 0.05658218

```
# Setting the seed
set.seed(0)

# coefficients at optimal lambda
ridge_coef=coef(model4,s=model4$lambda.min)
cbind(ridge_coef = as.vector(ridge_coef), model1_coef = model1$coef)
```

```
##                     ridge_coef  model1_coef
## (Intercept)      -7.6766992984 -9.991390385
## Pregnancies       0.0493263609  0.050011433
## Glucose           0.0255329209  0.039765633
## BloodPressure     0.0111465408  0.011151562
## SkinThickness     0.0170090947  0.019869137
## Insulin           0.0007210207 -0.000944938
## BMI               0.0270146165  0.036713348
## DiabetesPedigree  0.6043908255  0.891445035
## Age               0.0280541299  0.033584134
```

All the coefficients are slightly closer to zero than the model1 coefficients. They have shrunk due to the effect of the ridge penalty.

All nine variables' coefficients shrunk exactly to zero.

All variables are selected by the ridge regression which is expected as ridge regression does not do variable selection but only shrinks the coeffs. The coeff calculated using ridge regression at optimal lambda are much lower (~0) than the coeff generated using the logistic regression in model1 which again shows that the ridge regression shrank all of the coeffs to almost 0.

Prediction and Model Evaluation

Comparing the AICs and BICs of *model1 model2*, and *model3*

```
#Code to calculate and display AICs and BICs

set.seed(100)
n=nrow(dataTrain)
comp = rbind(model1.Full = c(AIC(model1,k=2), AIC(model1,k=log(n))),
model2.Complete.Search = c(AIC(model2,k=2), AIC(model2,k=log(n))),
model3.Forward = c(AIC(model3,k=2), AIC(model3,k=log(n))))
colnames(comp) = c("AIC","BIC")
comp
```

```
##                             AIC      BIC
## model1.Full            752.3711 794.8422
## model2.Complete.Search 751.0607 779.3748
## model3.Forward         752.2213 775.8163
```

Preferred model

model2 has the lowest AIC, and model3 has the lowest BIC. BIC tends to select smaller models than AIC, so since model3 has 4 variables selected, it has a lower BIC than model2 which has 5 variables selected.Both reduced models perform better than the full model on AIC and BIC even though we did not detect multicollinearity above while performing VIF.

Using models to classify (0/1) on the test dataset

```
# Code to calculate binary classifications. Using 0.5 as THE classification threshold.

predProb1 <- predict(model1, dataTest, type="response")
predClass1=rep(0,nrow(dataTest))
predClass1[predProb1>.5]=1
head(predClass1)
```

```
## [1] 0 0 0 1 0 0
```

```
predProb2 <- predict(model2, dataTest, type="response")
predClass2=rep(0,nrow(dataTest))
predClass2[predProb2>.5]=1
head(predClass2)
```

```
## [1] 0 0 0 1 0 0
```

```
predProb3 <- predict(model3, dataTest, type="response")
predClass3=rep(0,nrow(dataTest))
predClass3[predProb3>.5]=1
head(predClass3)
```

```
## [1] 0 0 0 1 0 0
```

```
#using optimal lambda value from model 4 to perform classification

predProb4 <- as.vector(predict(model4, as.matrix(dataTest[,-9]),
s=model4$lambda.min, type="response"))
predClass4=rep(0,nrow(dataTest))
predClass4[predProb4>.5]=1
head(predClass4)
```

```
## [1] 0 0 0 0 0 0
```

Calculate the accuracy, sensitivity and specificity of the predictions.

Note: Accuracy is the proportion of predictions that are correct overall, sensitivity is the proportion of actual positives that were correctly classified as positive, and specificity is the proportion of actual negatives which were correctly classified as negative.

```
#manual method

#pred_metrics_man = function(modelName, actualClass, predClass, totalClass) {
#cat(modelName, '\n')
#cat(c("Accuracy","Sensitivity","Specificity"), '\n')
#c(sum(predClass==actualClass)/totalClass,
#sum(predClass==actualClass&predClass==1)/nrow(dataTest[dataTest$Outcome==1,]),
#sum(predClass==actualClass&predClass==0)/nrow(dataTest[dataTest$Outcome==0,])
#)
#}

#pred_metrics_man("model1", dataTest$Outcome,
#predClass1, nrow(dataTest))
```

```r
# Code to calculate and display evaluation metrics
# Method 2- confusionMatrix()- Option 1
pred_metrics = function(modelName, actualClass, predClass) {
cat(modelName, '\n')
conmat <- caret::confusionMatrix(data = as.factor(predClass),
reference = as.factor(actualClass),
positive = "1")
c(conmat$overall["Accuracy"], conmat$byClass["Sensitivity"],
conmat$byClass["Specificity"])
}
pred_metrics("model1",dataTest$Outcome, predClass1)
```

```
## model1

##    Accuracy Sensitivity Specificity
##   0.8115942   0.6461538   0.8873239
```

```r
pred_metrics("model2",dataTest$Outcome, predClass2)
```

```
## model2

##    Accuracy Sensitivity Specificity
##   0.8019324   0.6307692   0.8802817
```

```r
pred_metrics("model3",dataTest$Outcome, predClass3)
```

```
## model3

##    Accuracy Sensitivity Specificity
##   0.8212560   0.7076923   0.8732394
```

```r
pred_metrics("model4",dataTest$Outcome, predClass4)
```

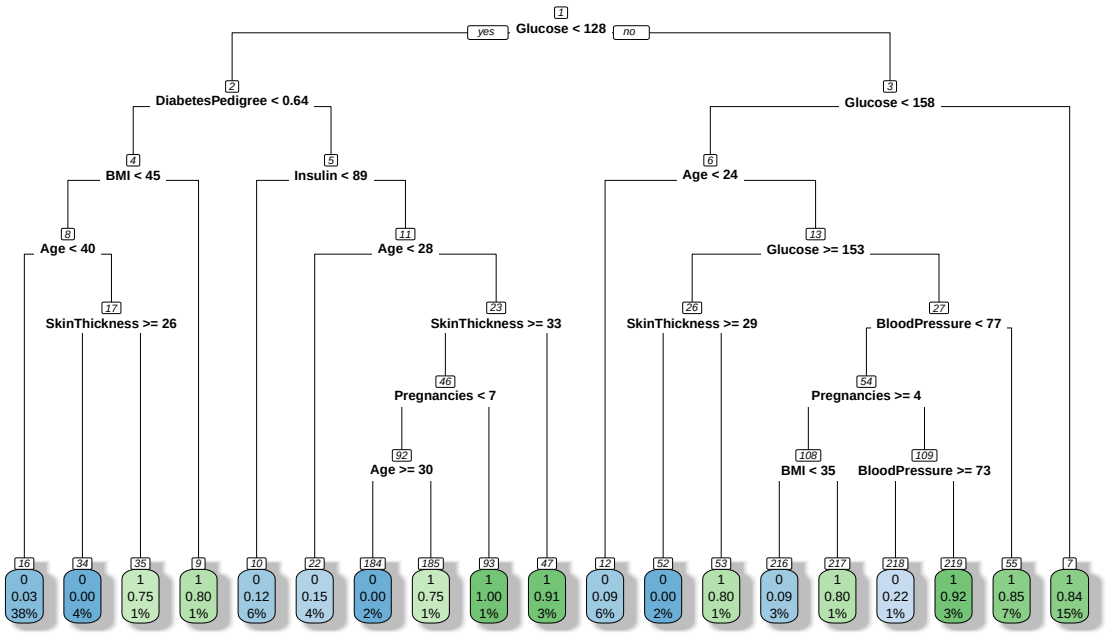
```
## model4

##    Accuracy Sensitivity Specificity
##   0.8260870   0.6461538   0.9084507
```

Model with highest prediction accuracy: Model with highest prediction accuracy model4 has the best prediction accuracy (0.8260870)

Given the context of the problem, sensitivity should be prioritized. Diabetes is a life threatening illness and testing negative for diabetes when a patient actually has diabetes could cause a lot of dangerous and expensive health complications. Specificity would control for false positives, which could result in a patient receiving expensive care for a condition they did not actually have, but this treatment would almost certainly not put their life in danger and the false positive could be reversed in a subsequent test during the course of their treatment.

Decision Trees

```
# Code to create and plot decision tree.
set.seed(300)
decision_tree <- rpart(Outcome~.,data=dataTrain, method="class")
#plotting model
rpart.plot(decision_tree, type=0,shadow.col="gray",nn=TRUE)
```




```
#box.palette= c(buzzgold,gtblue),
```

The most significant factor in predicting diabetes according to the decision tree is glucose level.

Advantages and disadvantages of decision tree models

With decision trees, it's visually easy to understand which factors are most relevant to predicting whether a new patient is likely to be diagnosed with diabetes or not. Decision trees in general also have very low bias, however they often have very large variance.

Goodness of Fit

We might not evaluate the goodness of fit of the logistic regression models because logistic regression does not have an error term. Additionally, we had data without replications, and replications are necessary for evaluating goodness of fit of logistic regression models. However, the data can be aggregated. As the data contains only numeric variables, they would need to be converted to categorical variables first, probably by grouping into bands (eg Age 18-24, Age 25-34, etc.).

One can perform goodness of fit using deviance residuals or even pearson residual analysis or even perform hypothesis testing using wald/z-test.