STAT 4255 Final Project

Emily Briggs, Jessica Zambuto

12/12/2022

STAT4255 Project- Diabetes Prediction

Report

Introduction:

Diabetes is a chronic health condition that affects millions of people globally. It is a disease that occurs when the body cannot properly produce insulin to break down glucose in the bloodstream. For clinical practice, it is important to understand the factors that can determine whether or not an individual has diabetes. Certain characteristics can make someone more likely to develop type 2 diabetes, and we would like to analyze those predictors in this report. Furthermore, we would like to find the most appropriate model that will classify whether or not someone is likely to have diabetes, given a certain set of predictors.

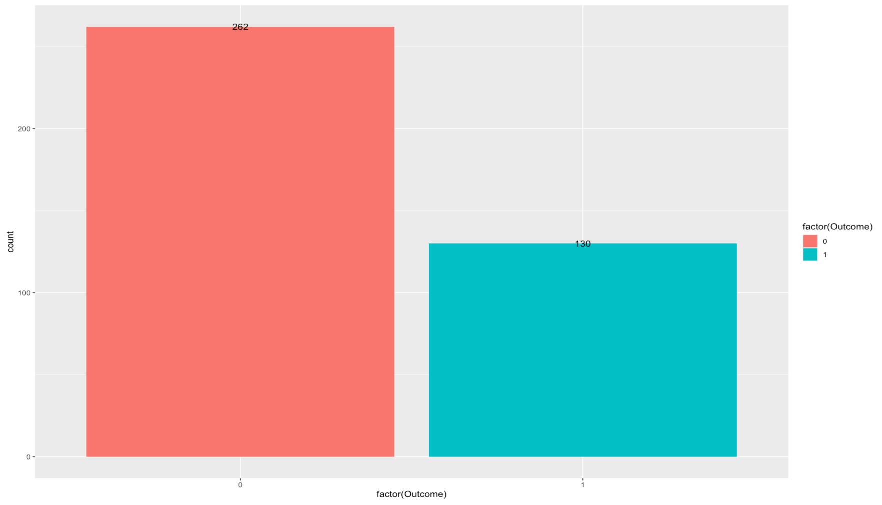
For our project, we looked at the "Diabetes Dataset" from user Akshay Dattatray Khare on Kaggle. It was pulled from a larger dataset created by the National Institute of Diabetes and Digestive and Kidney Diseases. In this smaller dataset that we will be using, only females aged 21 and older of Pima Indian heritage were included. The data set contains 768 observations of 9 variables. 8 of the variables are independent, and only one is dependent- the Outcome variable. The variables are, as follows:
- Pregnancies - Number of pregnancies
- Glucose - the Glucose level in blood
- Blood Pressure - the Blood pressure measurement
- SkinThickness - the thickness of the skin
- Insulin - the Insulin level in blood
- BMI - Body mass index
- DiabetesPedigreeFunction - function which scores likelihood of diabetes based on family history
- Age - age of the patient
- Outcome - To express the final result (1 is Yes = positive for diabetes and 0 is No = negative for diabetes)

These variables are all specific to each patient, and each row in the dataset represents one patient.

This data is supervised. We have a set of predictors that presumably lead us to a certain outcome. In this case, Y is the outcome measurement and takes on a finite, unordered set (e.g diagnosed with diabetes or not diagnosed). We will split the dataset into training data and test data. On the basis of the training data, we aim to accurately predict unseen test cases, understand which inputs affect the outcome, and how, and assess the quality of our predictions and inferences. This is also a classification problem. We want to build a function C(X) that takes as input the feature vector X and predicts its value for Y (whether or not the person has diabetes).

Exploratory Analysis

Before attempting to model the data, our first step was to perform a preliminary exploratory analysis, and cleanse the data if necessary. We found that there were indeed 9 variables with 768 total observations. We also noticed that all of the variables are numerical types, including the Outcome (dependent) variable. For now, this is okay, but in later modeling methods we will factorize Outcome into a binary response variable. Upon further exploratory analysis, I noticed a slight problem. Although there was not any null values in the data set, it appeared that missing data had instead been coded as '0'. For variables such as BMI and Age, this does not make sense. Therefore, I removed observations containing a 0 for any of the variables in order the clean the data. We were left with 392 observations. Out of these 392 observations, our constructed bar chart shows that there were 262 outcomes of '0' (patient did not have diabetes) and 130 outcomes of '1' (patient did have diabetes).



Modeling

For the purpose of this project, the main question we were trying to answer is: What model has the highest prediction accuracy for classifying whether or not an individual has diabetes? To do this, we will look at 9 classification methods/models: linear regression, best subset selection, logistic regression, LDA, QDA, Naive Bayes', K-nearest neighbors, classification trees, and generalized boosted regression. Our plan is to create each of these statistical models using our data, and then create a testing and training set to determine the prediction accuracy. From there, we will compare between the models and decide on the best fit for this data.

Linear Regression

One approach we used to answer our question was multiple linear regression. We used this because there was more than one predictor variable of interest and it would produce a model that could be used to predict the likelihood of someone developing diabetes based on certain risk factors. This approach could be used because our response variable was binary, there was a linear relationship between the independent and dependent variables (as shown in the scatter plots), the residuals were assumed to be normally distributed, and there was no presence of collinearity. Collinearity was checked by calculating the variance influential factors. All values were between 1 and 2 indicating moderate correlation between variables, but since they were all less than 5, all variables could be used in the model. R calculated the regression coefficient estimates using values that minimized the RSS. The P-values for the F-Statistics were also calculated in order to determine which variables were statistically significant. In this case, glucose, BMI, diabetes pedigree function, and age were significant as they all had a p-value less than .05 and thus, should be included in the model.

Best Subset

In order to determine the best model, we also used best subset selection. Adjusted R^2 was calculated and used to select the best model size, which was 5 variables. A graph was produced to demonstrate that the greatest R^2 value corresponded to 5 variables. However, it should be noted that this R^2 of .335 is very small, indicating that multiple linear regression may not be an appropriate model for the data. BIC was also calculated to determine how many predictor variables should be used to produce the best model and the corresponding coefficient estimates.

Logistic Regression

Logistic regression is a well-liked method for classification analysis, because p(X) will always have a value between 0 and 1, making it more easy to distinguish whether or not there is success. To fit the model, we used the glm function, which fits logistic regression models by maximum likelihood. We first fit a model using all 8 predictors, and then determined that the two most significant predictors were Glucose and DiabetesPedigreeFunction (lowest p-values). From there, we fit a reduced model using those predictors, and then split the data into a training and test set (70% training, 30% testing). With this split data, we determined a prediction accuracy of .805 for the reduced logistic regression model. A decision boundary is also shown in the appendix.

LDA

Linear Discriminant Analysis is a linear model for classification and dimensionality reduction. It focuses on maximizing the separability among known categories from the data. To classify at the value X = x, this method works by assigning x to the class with the largest discriminant score. Mathematically, this score is calculated using Gaussian density and plugging it into Bayes formula. LDA focuses on maximizing the distance between means and minimizing scatter within each category. In our analysis, we used the lda function from the MASS package to build our LDA model. We included an ROC curve, which indicated fairly good performance (curve is close to top left corner). We then used our reduced model and split data to calculate prediction accuracy, which came out to be .805, exactly the same as the logisitic regression model.

QDA

Another approach we used was quadratic discriminant analysis in which it was assumed that the observations within each class were drawn from a multivariate Gaussian distribution and each class had its own covariate matrix. When there is substantial separation between two classes, the parameter estimates for the logistic regression model are unstable, but QDA does not suffer from this problem, so we wanted to see if this model would be more accurate. The two variables of interest were glucose and diabetes pedigree function since they had the most significant p-values from the logistic regression analysis. The prediction accuracy, .7966, was slightly lower than that of the linear discriminant analysis.

Naive Bayes

We also were interested in using the naive bayes classifier to determine the best model for the data. This approach assumes that within the kth class, the p predictors are independent. This typically works best when n is not large enough relative to p to effectively estimate the joint distribution of the predictors within each class. It also reduces variance, so we thought this method would be appropriate. This method produced the same prediction accuracy and decision boundary as the QDA method.

KNN

K nearest neighbors is a nonparametric method in which given a value for K and a prediction point x_0 KNN regression first identifies the K training observations that are closest to x_0 represented by N_0. It then estimates f(x_0) using the average of all the training responses in N_0. KNN involves classifying a data point by looking at the nearest annotated data point, also known as the nearest neighbor. We divided the data into a training and test set. The test set was categorized using the k-nearest neighbors algorithm and compared to the training set to see how well the model performed. When K=3, the prediction accuracy was the highest, at .763.

Classification Trees

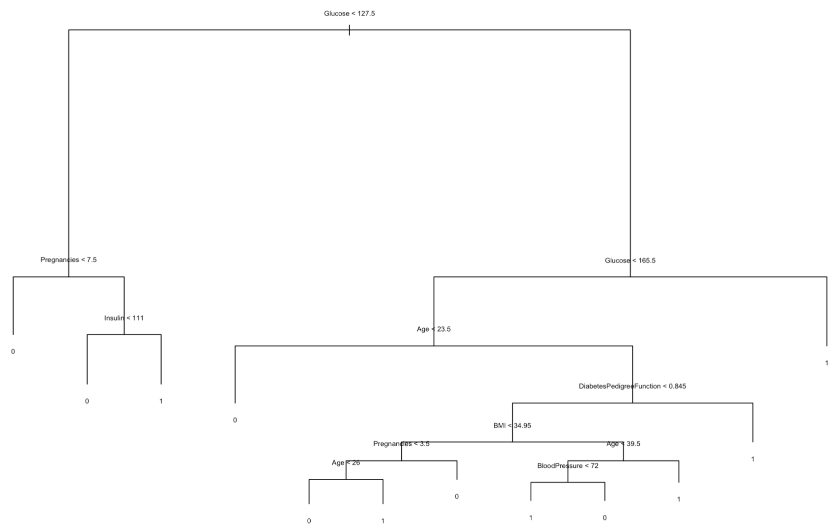
Tree-based methods involve stratifying or segmenting the predictor space into a number of regions, which help classify the data. For a classification tree, we predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs. To grow a classification tree, we use recursive binary splitting. In our analysis, we used functions from the tree package to build and prune our decision tree for diabetes outcome. Using the split data, we ended up having a pruned treewith 12 nodes, and a prediction accuracy of .814, the highest of any of the methods we used. We also used boosting on this data.

Summary of Final Results

We will now look at an executive summary of the predictive power (prediction accuracy) of each model we used to predict whether or not an individual has diabetes:
+ Linear Regression: adjusted R^2 = .332
+ Best Subset Selection: adjusted R^2 = .335
+ Logistic Regression: prediction accuracy = .805
+ Linear Discriminant Analysis: prediction accuracy = .805
+ Quadratic Discriminant Analysis: prediction accuracy = .797
+ Naive Bayes: prediction accuracy = .797
+ K-Nearest Neighbours: prediction accuracy = .763
+ Classification Trees: prediction accuracy = .813

Final Approach

Based on prediction accuracies, we decided to use classification trees as our final approach. Classification trees have several advantages over other methods. They are very simple to visualize and make sense of, and easy to explain to others. From a medical standpoint, this could be a useful model in clinical practice because of its simplicity. Additionally, trees closely mirror human decision-making, allowing for easier comprehension. They can be displayed graphically, and can easily handle qualitative variables, without the need to create dummy variables. There is concern that trees generally do not have the same level of predictive accuracy as other methods, but we can see from our results that the classification tree actually had the highest level of predictive accuracy for our diabetes data. For these reasons, we decided on the classification tree with 12 branches as our chosen model to predict whether or not an individual has diabetes, given our set of 8 predictor variables. We should note here that some of the prediction accuracies calculated above were from models containing all of the predictors, while others only used Glucose and DiabetesPedigreeFunction. We still stand by our results, because we noticed that going from the full model to the reduced model in most cases only slightly changed the prediction accuracy. The tree we have chosen as our method for classification can be pictured as follows:



Conclusion

In conclusion, after observing a selection of classification methods and fitting various models to our data, we have decided that using a classification tree with 12 branches will result in the highest prediction accuracy. Especially in clinical practice, it is extremely important to have predictive accuracy. When giving someone a diagnosis that could potentially have a huge impact on the course of their life is an enormous responsibility, and care should be taken to reduce error as much as possible. False negatives or false positives in a clinical setting could be disastrous. That is why we went with the model that is easy to comprehend and share with others, as well as the model with the highest predictive accuracy. We should note that a limitation of our data set is that they only looked at women over age 21 of Pima Indian heritage, so extrapolations to other groups of people should be done with caution.

Appendix

Exploratory Analysis

```
library(readr)
diabetes <- read.csv('diabetes.csv')

head(diabetes, 10)
```

```
##    Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
## 1            6     148            72            35       0 33.6
## 2            1      85            66            29       0 26.6
## 3            8     183            64             0       0 23.3
## 4            1      89            66            23      94 28.1
## 5            0     137            40            35     168 43.1
## 6            5     116            74             0       0 25.6
## 7            3      78            50            32      88 31.0
## 8           10     115             0             0       0 35.3
## 9            2     197            70            45     543 30.5
## 10           8     125            96             0       0  0.0
##    DiabetesPedigreeFunction Age Outcome
## 1                     0.627  50       1
## 2                     0.351  31       0
## 3                     0.672  32       1
## 4                     0.167  21       0
## 5                     2.288  33       1
## 6                     0.201  30       0
## 7                     0.248  26       1
## 8                     0.134  29       0
## 9                     0.158  53       1
## 10                    0.232  54       1
```

```
dim(diabetes)
```

```
## [1] 768   9
```

```
names(diabetes)
```

```
## [1] "Pregnancies"              "Glucose"
## [3] "BloodPressure"            "SkinThickness"
## [5] "Insulin"                  "BMI"
## [7] "DiabetesPedigreeFunction" "Age"
## [9] "Outcome"
```

```
str(diabetes)
```

```
## 'data.frame':    768 obs. of  9 variables:
##  $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
##  $ Glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
##  $ BloodPressure           : int  72 66 64 66 40 74 50 0 70 96 ...
##  $ SkinThickness           : int  35 29 0 23 35 0 32 0 45 0 ...
##  $ Insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
##  $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
##  $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
##  $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
##  $ Outcome                 : int  1 0 1 0 1 0 1 0 1 1 ...
```

```
summary(diabetes)
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin           BMI       DiabetesPedigreeFunction      Age
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00
##  Median : 30.5   Median :32.00   Median :0.3725           Median :29.00
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##     Outcome
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.349
##  3rd Qu.:1.000
##  Max.   :1.000
```

```
unique(diabetes$Pregnancies)
```

```
##  [1]  6  1  8  0  5  3 10  2  4  7  9 11 13 15 17 12 14
```

```
diabetes$age
```

```
## NULL
```

```
sum(is.na(diabetes))
```

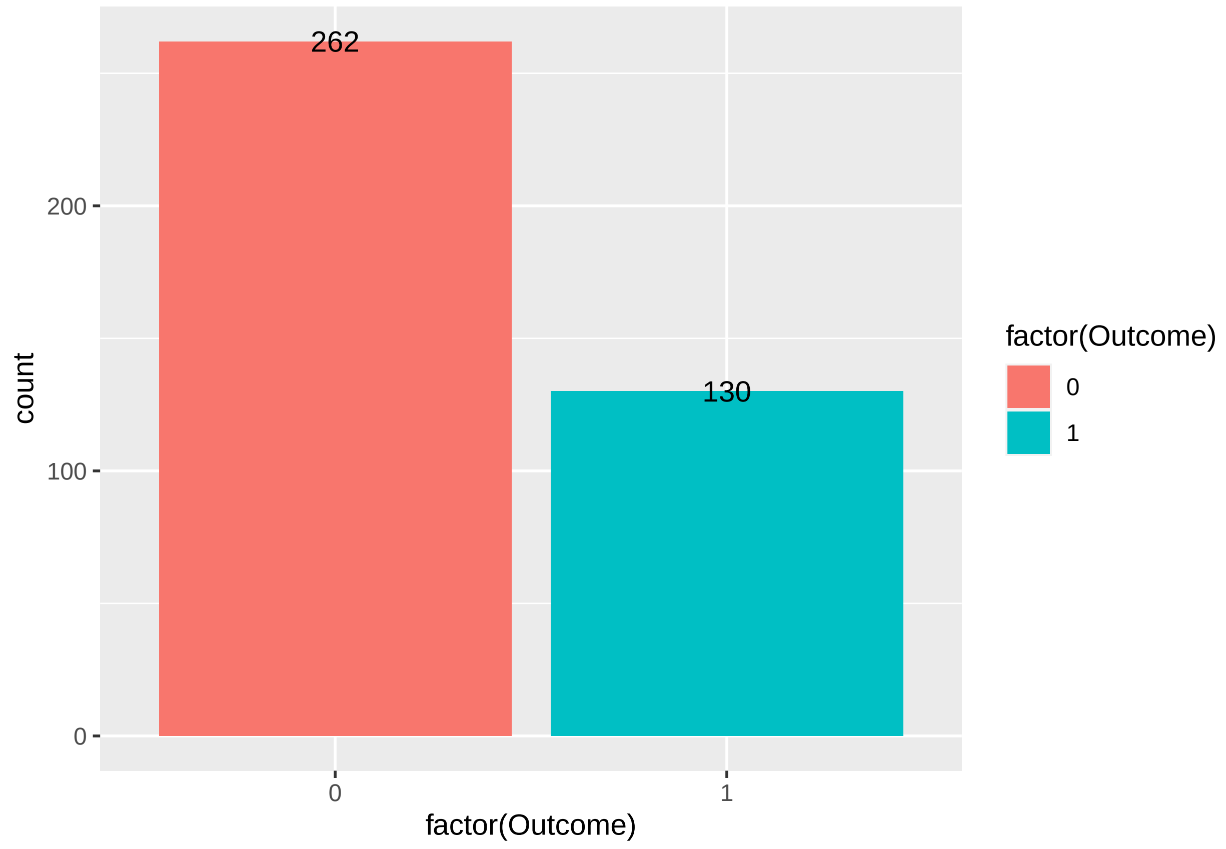
```
## [1] 0
```

```
diabetes = diabetes[diabetes$BloodPressure !=0, ]
diabetes = diabetes[diabetes$Glucose !=0, ]
diabetes = diabetes[diabetes$SkinThickness !=0, ]
diabetes = diabetes[diabetes$Insulin !=0, ]
diabetes = diabetes[diabetes$BMI !=0, ]
diabetes = diabetes[diabetes$Age !=0, ]
diabetes = diabetes[diabetes$DiabetesPedigreeFunction !=0, ]
dim(diabetes)
```

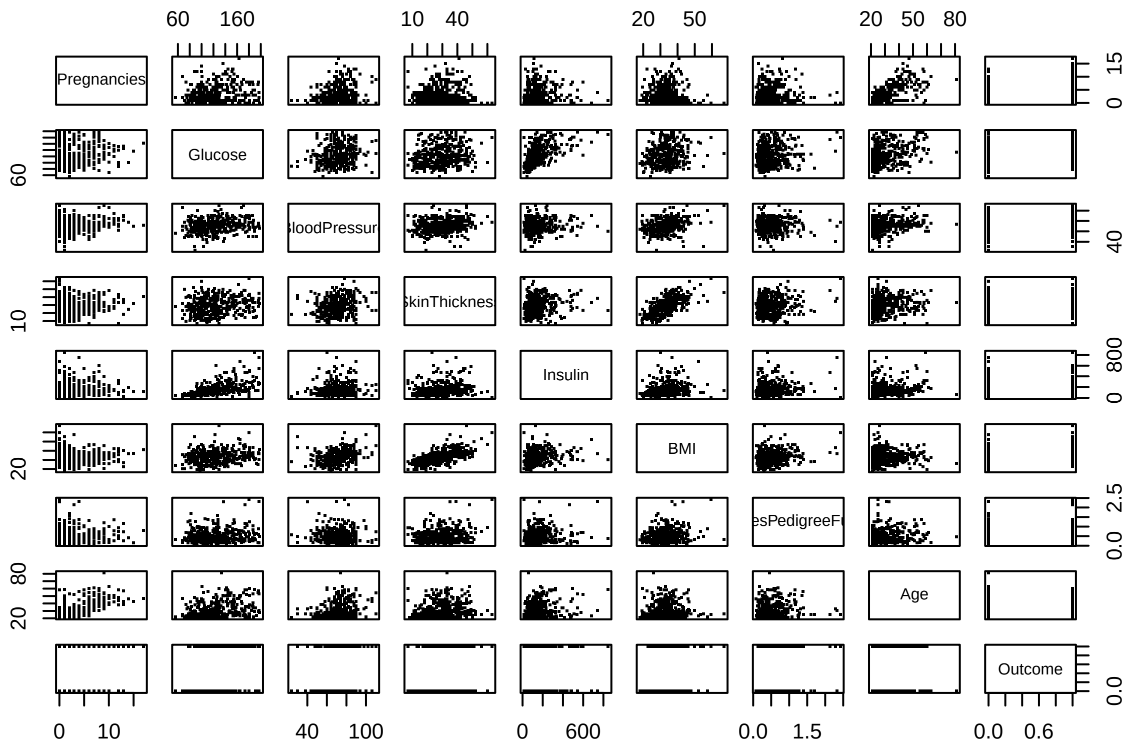
```
## [1] 392   9
```

```
#Bar chart
library(ggplot2)

(ggplot(diabetes, aes(factor(Outcome), fill=factor(Outcome)))
 + geom_bar()
 + geom_text(
     aes(label=after_stat(count)),
     stat='count',
     nudge_y=0.125,
     va='bottom'
 )
)
```

```
## Warning: Ignoring unknown parameters: va
```



```
#Scatterplot Matrix
pairs(diabetes, pch = '.')
```



```r
#Correlation Matrix
cor(diabetes[, -9])
```

```
##                          Pregnancies   Glucose BloodPressure SkinThickness
## Pregnancies              1.000000000 0.1982910     0.2133548     0.0932094
## Glucose                  0.198291043 1.0000000     0.2100266     0.1988558
## BloodPressure            0.213354775 0.2100266     1.0000000     0.2325712
## SkinThickness            0.093209397 0.1988558     0.2325712     1.0000000
## Insulin                  0.078983625 0.5812230     0.0985115     0.1821991
## BMI                     -0.025347276 0.2095159     0.3044034     0.6643549
## DiabetesPedigreeFunction 0.007562116 0.1401802    -0.0159711     0.1604985
## Age                      0.679608470 0.3436415     0.3000389     0.1677611
##                             Insulin         BMI DiabetesPedigreeFunction
## Pregnancies              0.07898363 -0.02534728              0.007562116
## Glucose                  0.58122301  0.20951592              0.140180180
## BloodPressure            0.09851150  0.30440337             -0.015971104
## SkinThickness            0.18219906  0.66435487              0.160498526
## Insulin                  1.00000000  0.22639652              0.135905781
## BMI                      0.22639652  1.00000000              0.158771043
## DiabetesPedigreeFunction 0.13590578  0.15877104              1.000000000
## Age                      0.21708199  0.06981380              0.085029106
##                                 Age
## Pregnancies              0.67960847
## Glucose                  0.34364150
## BloodPressure            0.30003895
## SkinThickness            0.16776114
## Insulin                    0.21708199
## BMI                        0.06981380
## DiabetesPedigreeFunction 0.08502911
## Age                        1.00000000
```

Linear Model

```
#Multiple Linear Regression
lm_fit <- lm(Outcome ~ ., data=diabetes)
summary(lm_fit)
```

```
##
## Call:
## lm(formula = Outcome ~ ., data = diabetes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.07966 -0.25711 -0.06177  0.25851  1.03750
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -1.103e+00  1.436e-01  -7.681 1.34e-13 ***
## Pregnancies               1.295e-02  8.364e-03   1.549  0.12230
## Glucose                   6.409e-03  8.159e-04   7.855 4.07e-14 ***
## BloodPressure             5.465e-05  1.730e-03   0.032  0.97482
## SkinThickness             1.678e-03  2.522e-03   0.665  0.50631
## Insulin                  -1.233e-04  2.045e-04  -0.603  0.54681
## BMI                       9.325e-03  3.901e-03   2.391  0.01730 *
## DiabetesPedigreeFunction  1.572e-01  5.804e-02   2.708  0.00707 **
## Age                       5.878e-03  2.787e-03   2.109  0.03559 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3853 on 383 degrees of freedom
## Multiple R-squared:  0.3458, Adjusted R-squared:  0.3321
## F-statistic:  25.3 on 8 and 383 DF,  p-value: < 2.2e-16
```

```
car::vif(lm_fit)
```

```
##              Pregnancies                  Glucose            BloodPressure
##                 1.900719                 1.670072                 1.231815
##            SkinThickness                  Insulin                      BMI
##                 1.852772                 1.556143                 1.979596
## DiabetesPedigreeFunction                      Age
##                 1.059315                 2.129433
```

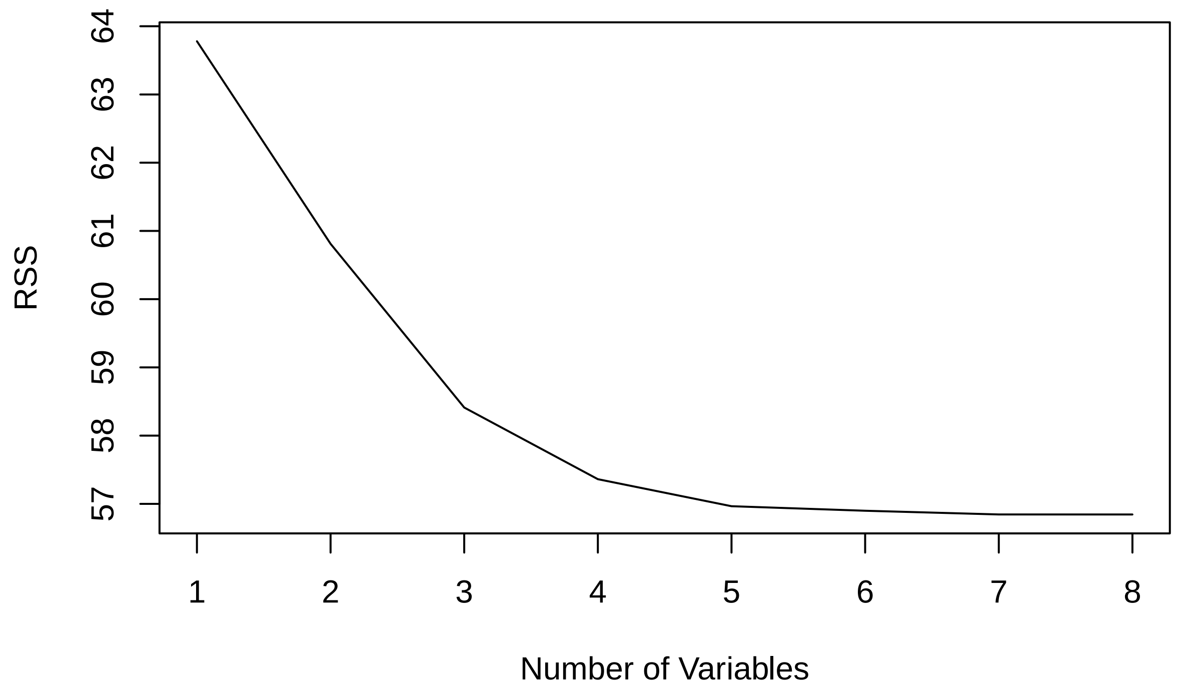
```
confint(lm_fit)
```

```
##                                  2.5 %        97.5 %
## (Intercept)              -1.3849532086 -0.8204004116
## Pregnancies              -0.0034924773  0.0293979543
## Glucose                   0.0048044659  0.0080127213
## BloodPressure            -0.0033476837  0.0034569829
## SkinThickness            -0.0032806439  0.0066356919
## Insulin                  -0.0005254366  0.0002787638
## BMI                       0.0016557622  0.0169943666
## DiabetesPedigreeFunction  0.0430729012  0.2713108875
## Age                       0.0003981287  0.0113580402
```

```
#Subset Selection
library(leaps)
best_subset <- regsubsets(Outcome ~ ., data = diabetes, nvmax=8)
bss_summary <-summary(best_subset)
names(bss_summary)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

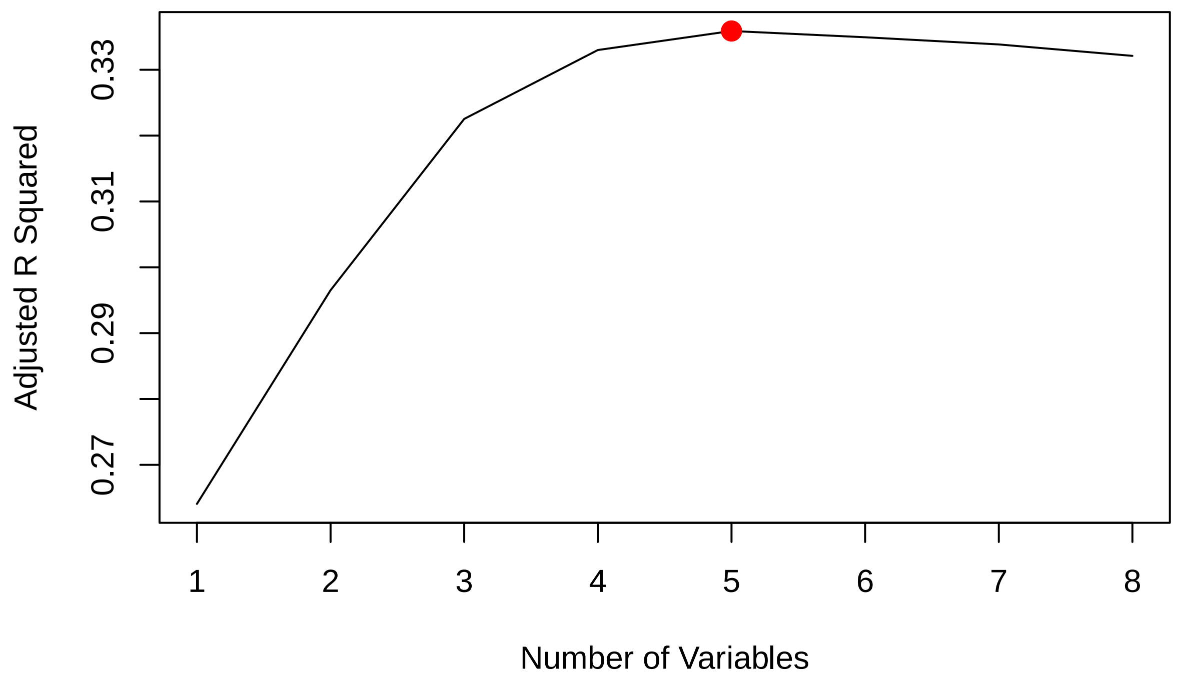
```
plot(bss_summary$rss, xlab = "Number of Variables", ylab = "RSS", type = "l")
```



```
plot(bss_summary$adjr2, xlab = "Number of Variables",
     ylab = "Adjusted R Squared", type = "l")
(best_modsize <- which.max(bss_summary$adjr2))
```

```
## [1] 5
```

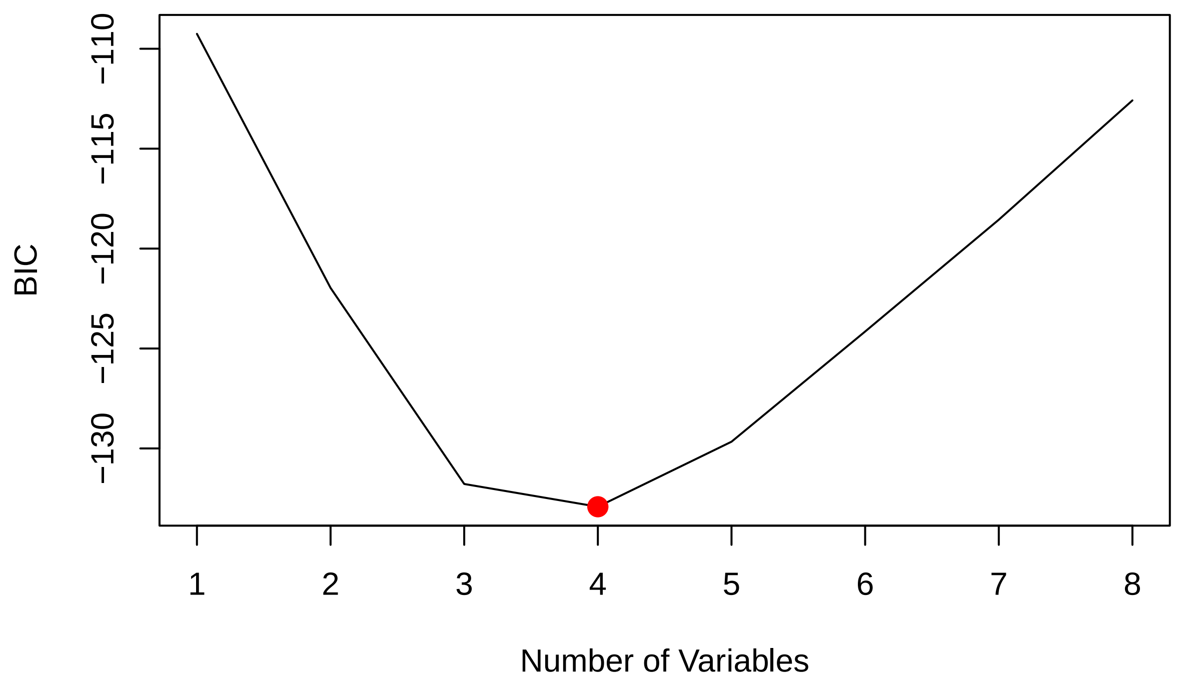
```r
points(best_modsize, bss_summary$adjr2[best_modsize],
       col="red", cex = 2, pch = 20)
```



```r
plot(bss_summary$bic, xlab = "Number of Variables",
     ylab = "BIC", type = "l")
(best_modsize <- which.min(bss_summary$bic))
```

```
## [1] 4
```

```r
points(best_modsize, bss_summary$bic[best_modsize],
       col="red", cex = 2, pch = 20)
```



```
#Coefficients for Best Model
coef(best_subset, best_modsize)
```

```
##              (Intercept)                  Glucose                      BMI
##             -1.118373857              0.006131381              0.010382152
## DiabetesPedigreeFunction                      Age
##              0.152978038              0.008897177
```

Logistic Regression Model

```
diabetes$Outcome <- as.factor(diabetes$Outcome)
#Fitting
glm_fit <- glm(Outcome ~ ., data = diabetes, family = binomial)
summary(glm_fit)
```

```
##
## Call:
## glm(formula = Outcome ~ ., family = binomial, data = diabetes)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7823  -0.6603  -0.3642   0.6409   2.5612
##
```

```
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -1.004e+01  1.218e+00  -8.246  < 2e-16 ***
## Pregnancies               8.216e-02  5.543e-02   1.482  0.13825
## Glucose                   3.827e-02  5.768e-03   6.635 3.24e-11 ***
## BloodPressure            -1.420e-03  1.183e-02  -0.120  0.90446
## SkinThickness             1.122e-02  1.708e-02   0.657  0.51128
## Insulin                  -8.253e-04  1.306e-03  -0.632  0.52757
## BMI                       7.054e-02  2.734e-02   2.580  0.00989 **
## DiabetesPedigreeFunction  1.141e+00  4.274e-01   2.669  0.00760 **
## Age                       3.395e-02  1.838e-02   1.847  0.06474 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 498.10  on 391  degrees of freedom
## Residual deviance: 344.02  on 383  degrees of freedom
## AIC: 362.02
##
## Number of Fisher Scoring iterations: 5
```

```r
#Model Summary
coef(glm_fit)
```

```
##              (Intercept)              Pregnancies                  Glucose
##            -1.004074e+01             8.215942e-02             3.826952e-02
##            BloodPressure            SkinThickness                  Insulin
##            -1.420290e-03             1.122139e-02            -8.253128e-04
##                      BMI DiabetesPedigreeFunction                      Age
##             7.053758e-02             1.140909e+00             3.395162e-02
```

```r
summary(glm_fit)$coefficients
```

```
##                               Estimate  Std. Error    z value     Pr(>|z|)
## (Intercept)              -1.004074e+01 1.217674335 -8.2458330 1.640136e-16
## Pregnancies               8.215942e-02 0.055425546  1.4823385 1.382502e-01
## Glucose                   3.826952e-02 0.005767709  6.6351344 3.242069e-11
## BloodPressure            -1.420290e-03 0.011833396 -0.1200239 9.044642e-01
## SkinThickness             1.122139e-02 0.017083709  0.6568474 5.112790e-01
## Insulin                  -8.253128e-04 0.001306439 -0.6317270 5.275653e-01
## BMI                       7.053758e-02 0.027342138  2.5798122 9.885405e-03
## DiabetesPedigreeFunction  1.140909e+00 0.427433723  2.6692059 7.603082e-03
## Age                       3.395162e-02 0.018381721  1.8470318 6.474254e-02
```

```r
summary(glm_fit)$coefficients[, 4] #p-values
```

```
##              (Intercept)              Pregnancies                  Glucose
##             1.640136e-16             1.382502e-01             3.242069e-11
##            BloodPressure            SkinThickness                  Insulin
##             9.044642e-01             5.112790e-01             5.275653e-01
##                      BMI DiabetesPedigreeFunction                      Age
##             9.885405e-03             7.603082e-03             6.474254e-02
```

```r
#Prediction Accuracy
glm_probs <-  predict(glm_fit, type = 'response')
glm_probs[1:10]
```

```
##          4          5          7          9         14         15         17 
## 0.02711452 0.89756363 0.03763559 0.85210016 0.79074609 0.71308818 0.40253966 
##         19         20         21 
## 0.21114313 0.21942842 0.41601896
```

```r
glm_pred <- rep('0', 392)
glm_pred[glm_probs > 0.5] <- '1'
table(glm_pred, diabetes$Outcome)
```

```
## 
## glm_pred   0   1
##        0 233  56
##        1  29  74
```

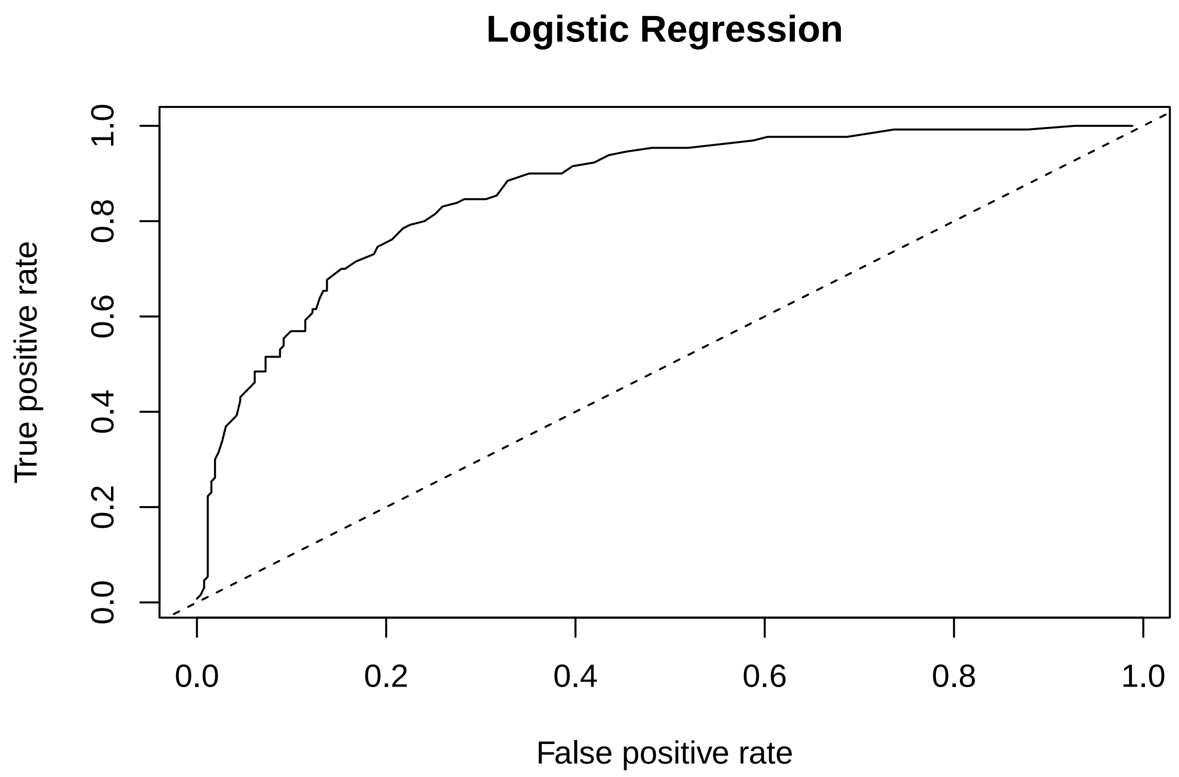
```r
((233 + 74) / 392) #prediction accuracy
```

```
## [1] 0.7831633
```

```r
mean(glm_pred == diabetes$Outcome)
```

```
## [1] 0.7831633
```

```r
#Performance Evaluation (ROC curve)
roc <- function(threshold, pred_pr, Outcome) {
  pred_class <- rep("0", length(Outcome))
  pred_class[pred_pr > threshold] <- "1"
  FP_rate <- sum(pred_class != Outcome & pred_class == "1")/sum(Outcome == "0")
  TP_rate <- sum(pred_class == Outcome & pred_class == "1")/sum(Outcome == "1")
  return(c(FP_rate, TP_rate))
}
THRESHOLD <- seq(0.01, 0.99, by = 0.01)
## Each row records the FP and FN rates for one threshold
Rates <- matrix(NA, nrow = length(THRESHOLD), ncol = 2)
for (i in seq_along(THRESHOLD)) {
  threshold <- THRESHOLD[i]
  Rates[i, ] <- roc(threshold, glm_probs, diabetes$Outcome)
}
plot(Rates[, 2] ~ Rates[, 1], type = "l", xlab = "False positive rate",
     ylab = "True positive rate", main = "Logistic Regression")
abline(0, 1, lty = 2)
```



```
#Data Splitting
#Dividing data into a training sample (70%) and a validation sample (30%)
set.seed(1234)
train <- sample(nrow(diabetes), 0.7*nrow(diabetes))
diab.train <- diabetes[train,]
diab.test <- diabetes[-train,]
Outcome_test <- diabetes$Outcome[-train]

#Fit logistic regression on training data
glm_train <- glm(Outcome ~ . , data = diab.train, family = binomial)
#Predict on test data
glm_probs <- predict(object = glm_train, newdata = diab.test, type = "response")
glm_pred <- rep("0", 118)
glm_pred[glm_probs > 0.5] <- "1"
table(glm_pred, Outcome_test)
```

```
##          Outcome_test
## glm_pred  0  1
##         0 71 14
##         1  9 24
```

```
mean(glm_pred == Outcome_test) #Prediction Accuracy
```

```
## [1] 0.8050847
```

```r
mean(glm_pred != Outcome_test) #Misclassification Error Rate
```

```
## [1] 0.1949153
```

```r
#Reduced model, using two predictors with most significant p-values
glm.reduc <- glm(Outcome ~ Glucose + DiabetesPedigreeFunction,
                 data = diabetes, family = binomial, subset = train)
#Predict on test data
glm_probs <- predict(object = glm.reduc, newdata = diab.test, type = "response")
glm_pred <- rep("0", 118)
glm_pred[glm_probs > 0.5] <- "1"
table(glm_pred, Outcome_test)
```

```
##         Outcome_test
## glm_pred  0  1
##        0 74 17
##        1  6 21
```

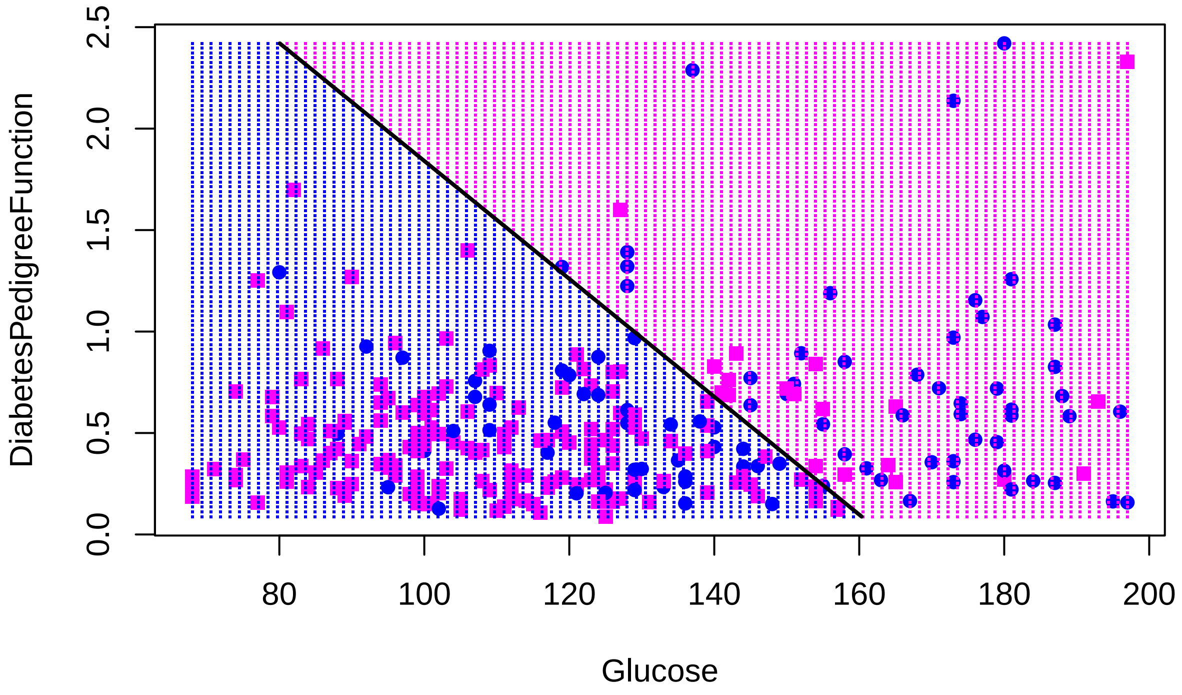
```r
mean(glm_pred == Outcome_test) #Prediction Accuracy
```

```
## [1] 0.8050847
```

```r
(21 / (21 + 17))  #true positive rate
```

```
## [1] 0.5526316
```

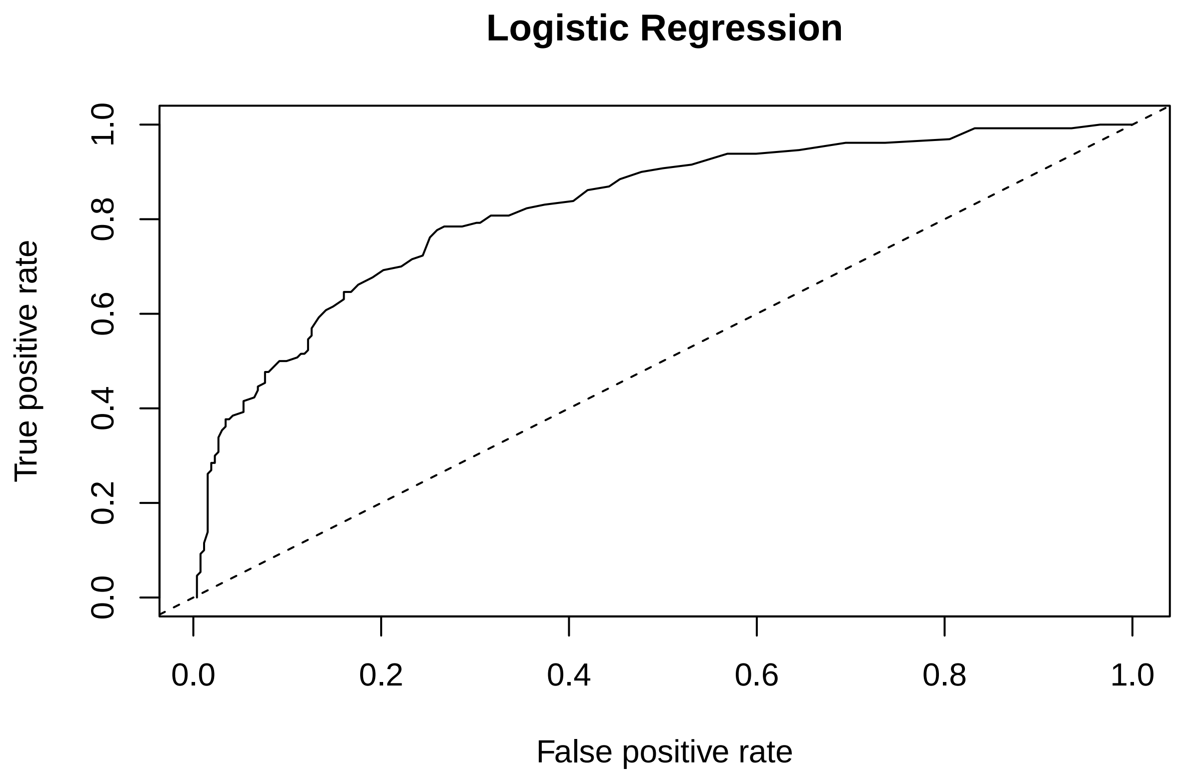
```r
#Decision Boundary (having trouble)
plot(DiabetesPedigreeFunction ~ Glucose, data = diabetes, subset = train,
     pch = c(15, 16)[diabetes$Outcome[train]],
     col = c("magenta", "blue")[diabetes$Outcome[train]])

grid_gluc <- seq(min(diabetes$Glucose[train]),
                 max(diabetes$Glucose[train]),
                 length.out = 100)
grid_dpf <- seq(min(diabetes$DiabetesPedigreeFunction[train]),
                max(diabetes$DiabetesPedigreeFunction[train]),
                length.out = 100)
gridpts <- expand.grid(Glucose = grid_gluc, DiabetesPedigreeFunction = grid_dpf)

pred_pr <- predict(glm.reduc, gridpts, type = "response")
pred_class <- as.factor(ifelse(pred_pr > 0.5, "1", "0"))

points(gridpts, col = c("blue", "magenta")[pred_class], pch = ".", cex = 1)

contour(x = grid_gluc, y = grid_dpf,
        z = matrix(pred_pr, nrow = 100),
        levels = 0.5, lwd = 2, drawlabels = FALSE, add = TRUE)
```



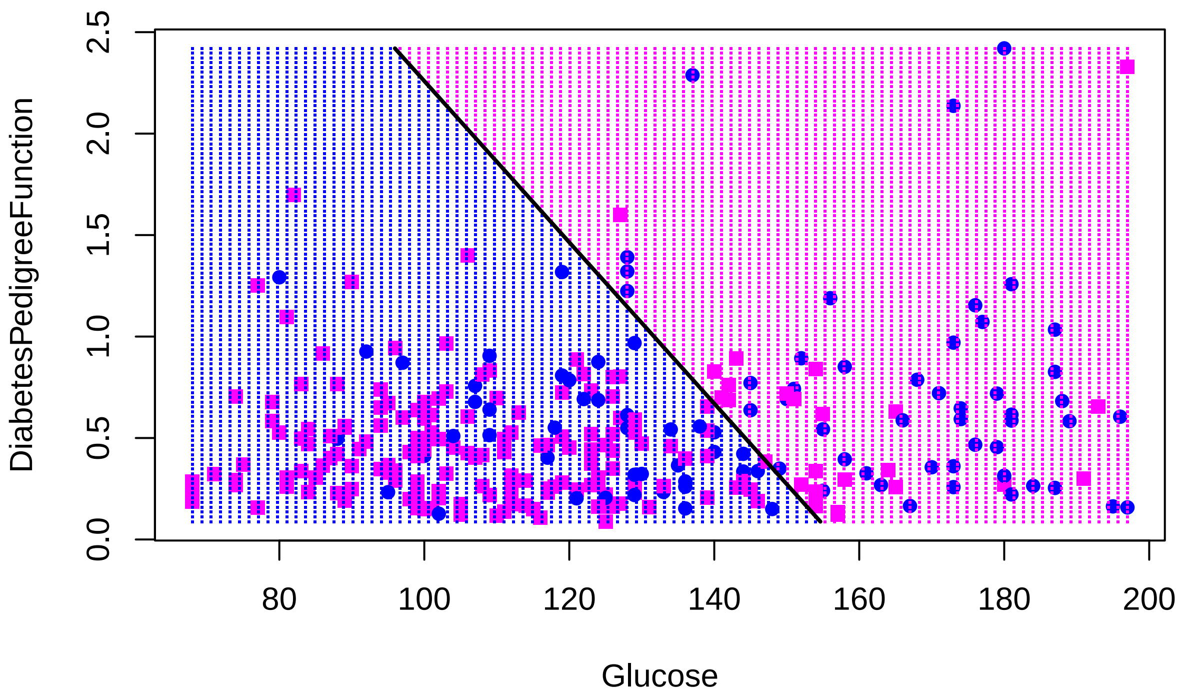
Linear Discriminant Analysis

```
library(MASS)
lda_fit <- lda(Outcome ~ Glucose + DiabetesPedigreeFunction, data = diabetes)
lda_pred <- predict(lda_fit)
lda_probs <- lda_pred$posterior[, 2]

## ROC curve for LDA
roc <- function(threshold, pred_pr, Outcome) {
  pred_class <- rep("0", length(Outcome))
  pred_class[pred_pr > threshold] <- "1"
  FP_rate <- sum(pred_class != Outcome & pred_class == "1")/sum(Outcome == "0")
  TP_rate <- sum(pred_class == Outcome & pred_class == "1")/sum(Outcome == "1")
  return(c(FP_rate, TP_rate))
}
THRESHOLD <- seq(0.01, 0.99, by = 0.01)
## Each row records the FP and FN rates for one threshold
Rates <- matrix(NA, nrow = length(THRESHOLD), ncol = 2)
for (i in seq_along(THRESHOLD)) {
  threshold <- THRESHOLD[i]
  Rates[i, ] <- roc(threshold, lda_probs, diabetes$Outcome)
}
plot(Rates[, 2] ~ Rates[, 1], type = "l", xlab = "False positive rate",
     ylab = "True positive rate", main = "Logistic Regression")
```

```r
abline(0, 1, lty = 2)
```



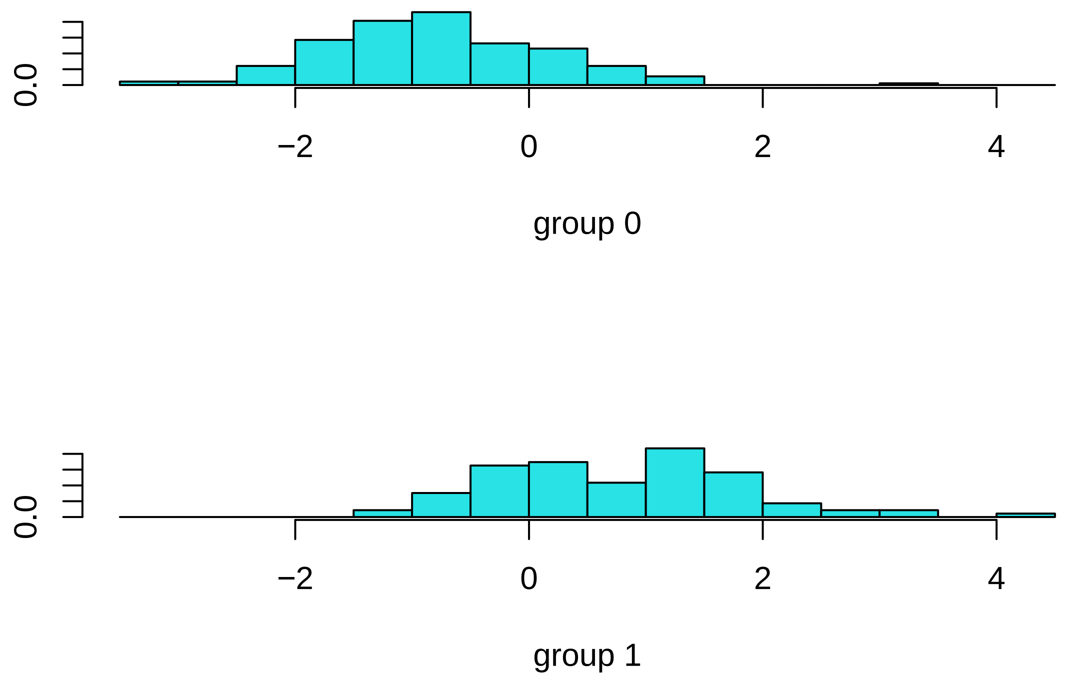
```r
#Decision Boundary
plot(DiabetesPedigreeFunction ~ Glucose, data = diabetes, subset = train,
     pch = c(15, 16)[diabetes$Outcome[train]],
     col = c("magenta", "blue")[diabetes$Outcome[train]])
preds <- predict(lda_fit, gridpts)
points(gridpts, col = c("blue", "magenta")[preds$class], pch = ".", cex = 1)
contour(x = grid_gluc, y = grid_dpf,
        z = matrix(preds$posterior[, 2], nrow = 100),
        levels = 0.5, lwd = 2, drawlabels = FALSE, add = TRUE)
```



```r
#Data splitting
lda_train <- lda(Outcome ~ . , data = diabetes, subset = train)
lda_train
```

```
## Call:
## lda(Outcome ~ ., data = diabetes, subset = train)
##
## Prior probabilities of groups:
##         0         1
## 0.6642336 0.3357664
##
## Group means:
##   Pregnancies  Glucose BloodPressure SkinThickness  Insulin      BMI
## 0    2.868132 112.7198      68.45604      27.91758 130.2308 31.75824
## 1    4.565217 145.4891      74.22826      33.00000 207.8043 35.96739
##   DiabetesPedigreeFunction      Age
## 0                0.4679176 28.64835
## 1                0.6407826 35.73913
##
## Coefficients of linear discriminants:
##                                    LD1
## Pregnancies                0.0793474599
## Glucose                    0.0256691901
## BloodPressure              0.0082991028
## SkinThickness             -0.0040812125
## Insulin                   -0.0001935807
## BMI                          0.0649687367
## DiabetesPedigreeFunction     0.8874579661
## Age                          0.0208765556
```

```
plot(lda_train)
```



```
lda_pred <- predict(lda_train, diab.test)
lda_class <- lda_pred$class
table(lda_class, Outcome_test)
```

```
##          Outcome_test
## lda_class  0  1
##         0 71 14
##         1  9 24
```

```
mean(lda_class == Outcome_test) #Prediction Accuracy
```

```
## [1] 0.8050847
```

```
head(lda_pred$posterior)
```

```
##             0          1
## 4  0.9839307 0.01606934
## 7  0.9820331 0.01796693
## 14 0.2050382 0.79496183
## 19 0.8961522 0.10384779
## 20 0.8240412 0.17595878
## 25 0.1768224 0.82317759
```

```
lda_pred$posterior[1:20, 1]
```

```
##         4         7        14        19        20        25        28        33
## 0.9839307 0.9820331 0.2050382 0.8961522 0.8240412 0.1768224 0.9776355 0.9834651
##        40        51        53        58        71        88        99       100
## 0.3137581 0.9756983 0.9696402 0.6047193 0.8632242 0.8901534 0.9604146 0.4893895
##       110       111       115       121
## 0.9304469 0.4762874 0.3418373 0.1083661
```

```
lda_class[1:20]
```

```
##  [1] 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 1 0 1 1 1
## Levels: 0 1
```

```
sum(lda_pred$posterior[, 1] >= 0.5) # num. of obs. classified to "0" group
```

```
## [1] 85
```

```
sum(lda_pred$posterior[, 1] < 0.5)
```

```
## [1] 33
```

Quadratic Discriminant Analysis

```
qda_fit <- qda(Outcome ~ Glucose + DiabetesPedigreeFunction, data = diabetes,
               subset = train)
qda_fit
```

```
## Call:
## qda(Outcome ~ Glucose + DiabetesPedigreeFunction, data = diabetes,
##     subset = train)
##
## Prior probabilities of groups:
##         0         1
## 0.6642336 0.3357664
##
## Group means:
##    Glucose DiabetesPedigreeFunction
## 0 112.7198                0.4679176
## 1 145.4891                0.6407826
```

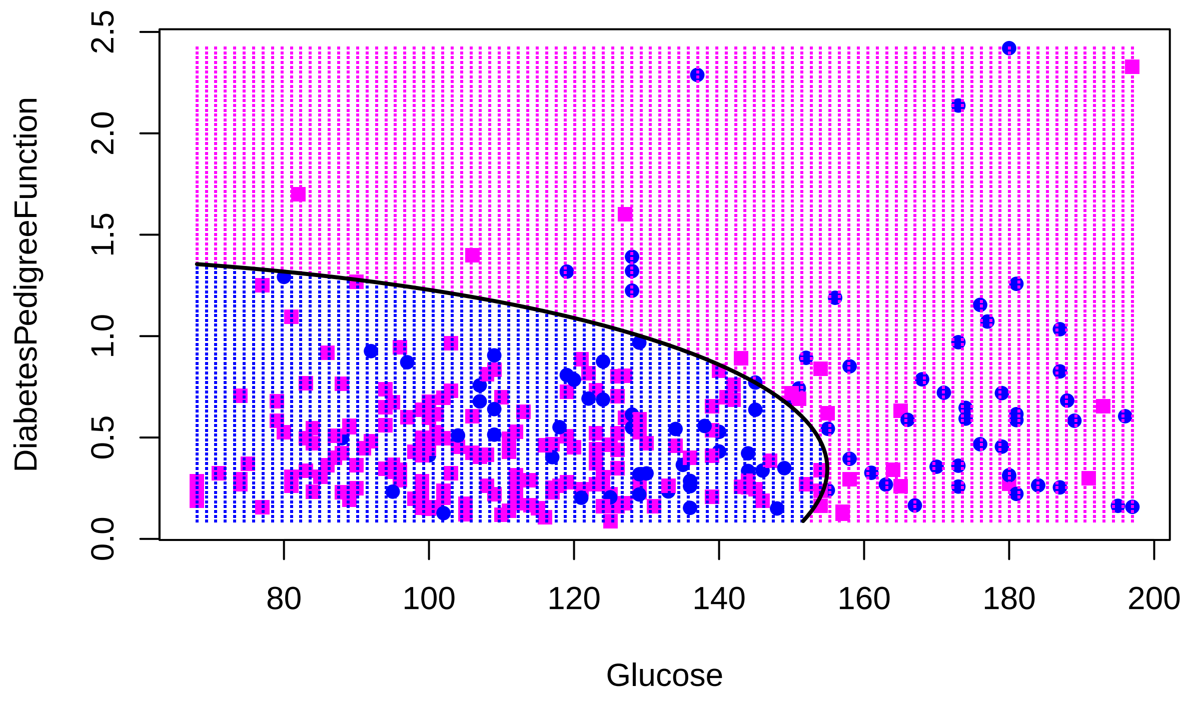
```r
#Prediction Accuracy
qda_class <- predict(qda_fit, diab.test)$class
table(qda_class, Outcome_test)
```

```
##          Outcome_test
## qda_class  0  1
##         0 73 17
##         1  7 21
```

```r
mean(qda_class == Outcome_test)
```

```
## [1] 0.7966102
```

```r
#Decision Boundary
plot(DiabetesPedigreeFunction ~ Glucose, data = diabetes, subset = train,
     pch = c(15, 16)[diabetes$Outcome[train]],
     col = c("magenta", "blue")[diabetes$Outcome[train]])
preds <- predict(qda_fit, gridpts)
points(gridpts, col = c("blue", "magenta")[preds$class], pch = ".", cex = 1)
contour(x = grid_gluc, y = grid_dpf,
        z = matrix(preds$posterior[, 2], nrow = 100),
        levels = 0.5, lwd = 2, drawlabels = FALSE, add = TRUE)
```



Naive Bayes

```
library(e1071)
(nb_fit <- naiveBayes(Outcome ~ Glucose + DiabetesPedigreeFunction,
                      data = diabetes, subset = train))
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1
## 0.6642336 0.3357664
##
## Conditional probabilities:
##    Glucose
## Y       [,1]     [,2]
##   0 112.7198 25.64749
##   1 145.4891 29.62049
##
##    DiabetesPedigreeFunction
## Y        [,1]      [,2]
##   0 0.4679176 0.3055408
##   1 0.6407826 0.4375197
```

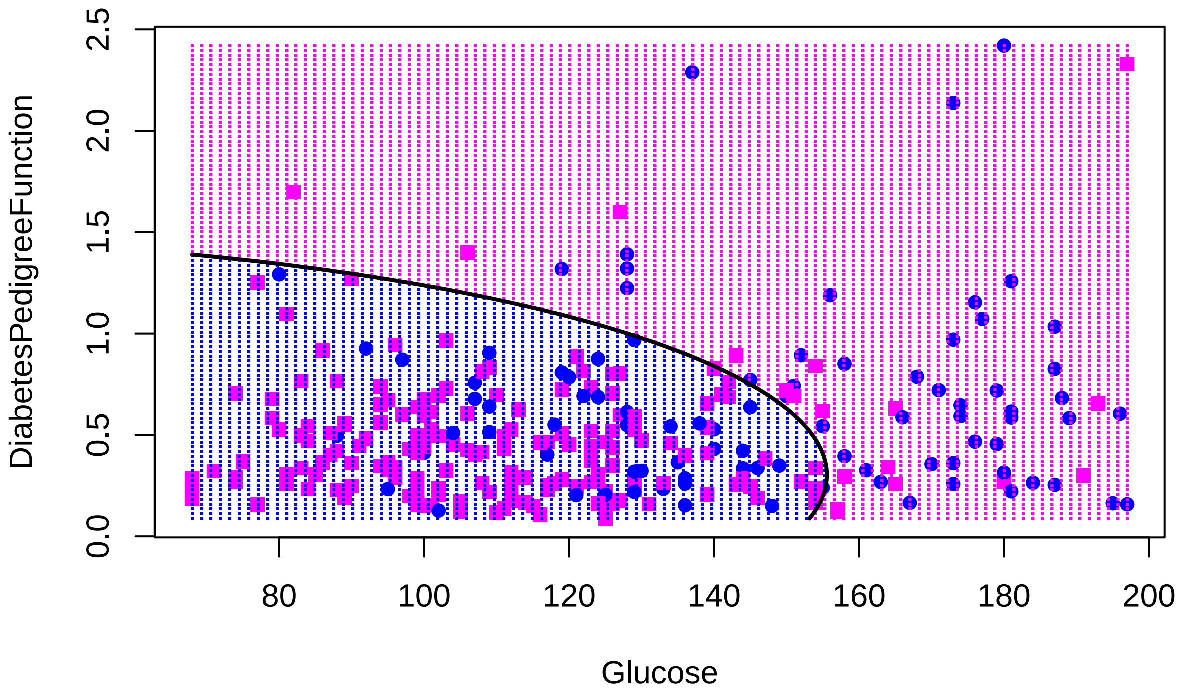
```
#Prediction Accuracy
nb_class <- predict(nb_fit, diab.test)
table(nb_class, Outcome_test)
```

```
##         Outcome_test
## nb_class  0  1
##        0 73 17
##        1  7 21
```

```
mean(nb_class == Outcome_test)
```

```
## [1] 0.7966102
```

```
#Decision Boundary
plot(DiabetesPedigreeFunction ~ Glucose, data = diabetes, subset = train,
     pch = c(15, 16)[diabetes$Outcome[train]],
     col = c("magenta", "blue")[diabetes$Outcome[train]])
pred_class <- predict(nb_fit, gridpts)
pred_pr <- predict(nb_fit, gridpts, type = "raw")
points(gridpts, col = c("blue", "magenta")[pred_class], pch = ".", cex = 1)
contour(x = grid_gluc, y = grid_dpf,
        z = matrix(pred_pr[, 2], nrow = 100),
        levels = 0.5, lwd = 2, drawlabels = FALSE, add = TRUE)
```



K-Nearest Neighbors

```
library(class)
train_X <- cbind(diabetes$Glucose, diabetes$DiabetesPedigreeFunction)[train, ]
test_X <- cbind(diabetes$Glucose, diabetes$DiabetesPedigreeFunction)[-train, ]
train_Direction <- diabetes$Outcome[train]
set.seed(1) # obs. could tie as nearest neighbors
knn_pred <- knn(train_X, test_X, train_Direction, k = 1)
table(knn_pred, Outcome_test)
```

```
##         Outcome_test
## knn_pred  0  1
##        0 63 16
##        1 17 22
```

```
#Prediction Accuracy
mean(knn_pred == Outcome_test)
```

```
## [1] 0.720339
```

```
#Trying k = 3
knn_pred3 <- knn(train_X, test_X, train_Direction, k = 3)
table(knn_pred3, Outcome_test)
```

```
##          Outcome_test
## knn_pred3  0  1
##         0 69 17
##         1 11 21
```

```
#Prediction Accuracy
mean(knn_pred3 == Outcome_test)
```

```
## [1] 0.7627119
```

```
#Trying k = 5
knn_pred5 <- knn(train_X, test_X, train_Direction, k = 5)
table(knn_pred5, Outcome_test)
```

```
##          Outcome_test
## knn_pred5  0  1
##         0 71 22
##         1  9 16
```

```
#Prediction Accuracy
mean(knn_pred5 == Outcome_test)
```

```
## [1] 0.7372881
```

Classification Trees

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v purrr   0.3.4     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::select() masks MASS::select()
```

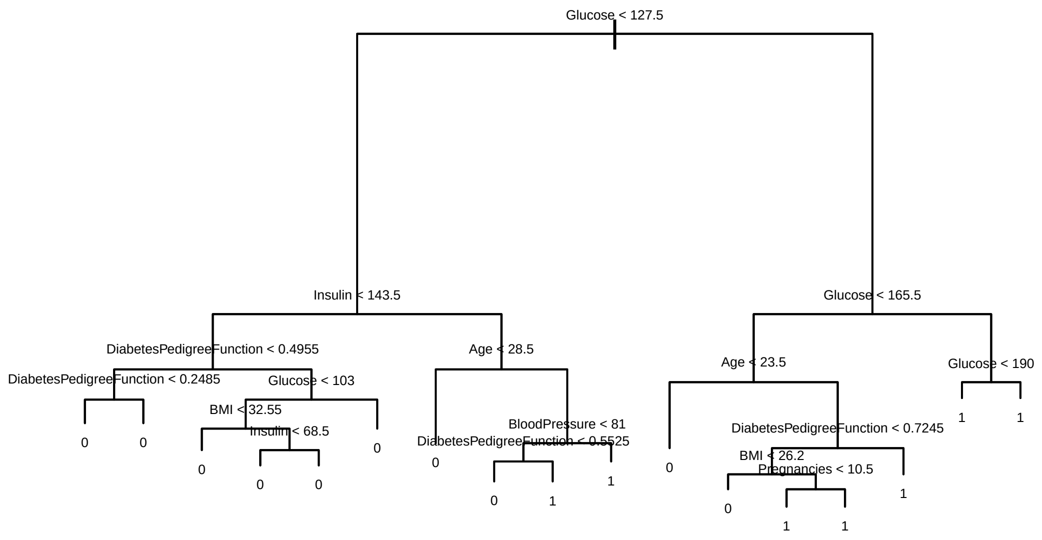
```
library(tree)
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
tree_outcome <- tree(Outcome ~ ., data = diabetes)
summary(tree_outcome)
```

```
##
## Classification tree:
## tree(formula = Outcome ~ ., data = diabetes)
## Variables actually used in tree construction:
## [1] "Glucose"                  "Insulin"
## [3] "DiabetesPedigreeFunction" "BMI"
## [5] "Age"                      "BloodPressure"
## [7] "Pregnancies"
## Number of terminal nodes:  17
## Residual mean deviance:  0.6294 = 236 / 375
## Misclassification error rate: 0.148 = 58 / 392
```

```r
plot(tree_outcome)
text(tree_outcome, cex = 0.45)
```



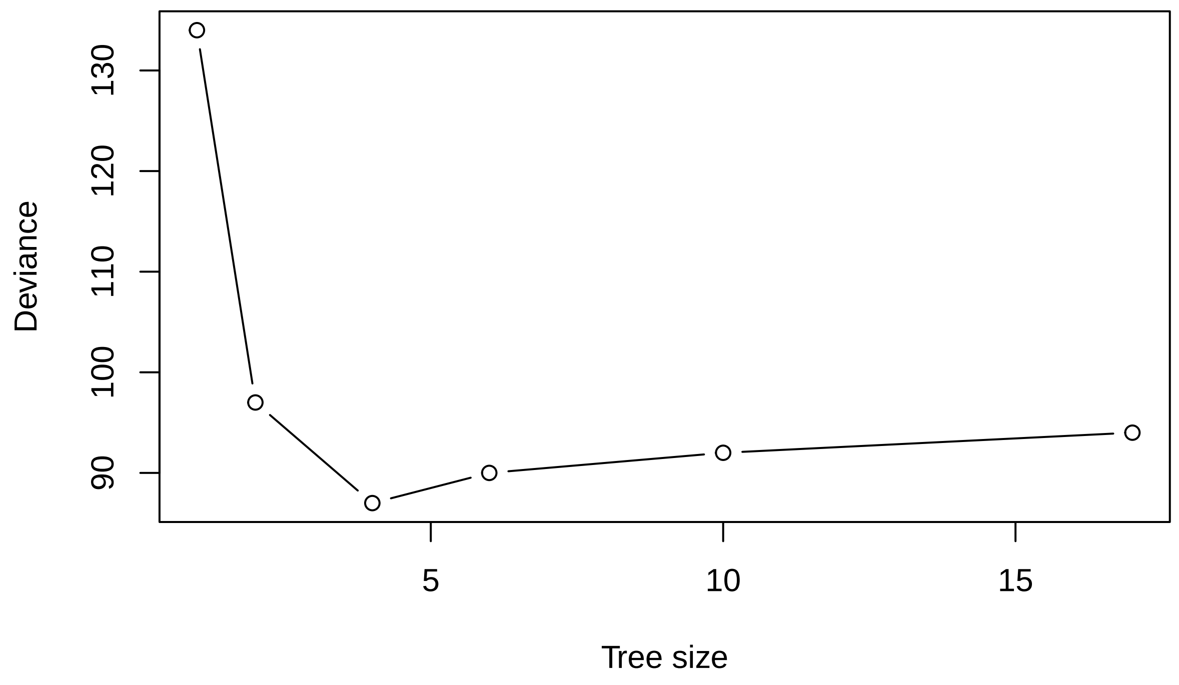
```r
#Gini Index
gini_outcome <- tree(Outcome ~ ., data = diabetes, split = "gini")
summary(gini_outcome)
```

```
##
## Classification tree:
## tree(formula = Outcome ~ ., data = diabetes, split = "gini")
## Variables actually used in tree construction:
## [1] "Glucose"                  "Insulin"
## [3] "DiabetesPedigreeFunction" "BloodPressure"
## [5] "Pregnancies"              "Age"
## [7] "SkinThickness"
## Number of terminal nodes:  37
## Residual mean deviance:  0.5856 = 207.9 / 355
## Misclassification error rate: 0.1429 = 56 / 392
```

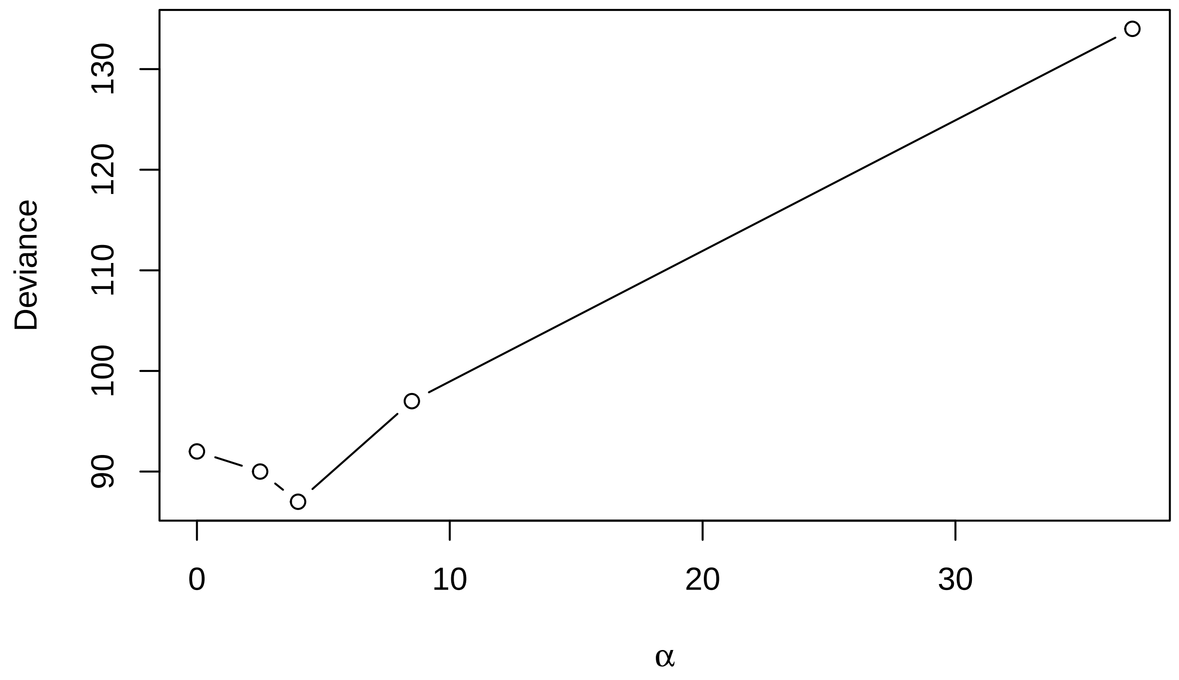
```
#Pruning
set.seed(2019)
cv_outcome <- cv.tree(tree_outcome, FUN = prune.misclass)
str(cv_outcome)
```

```
## List of 4
##  $ size  : int [1:6] 17 10 6 4 2 1
##  $ dev   : num [1:6] 94 92 90 87 97 134
##  $ k     : num [1:6] -Inf 0 2.5 4 8.5 ...
##  $ method: chr "misclass"
##  - attr(*, "class")= chr [1:2] "prune" "tree.sequence"
```

```
plot(cv_outcome$size, cv_outcome$dev, type = "b",
     xlab = "Tree size", ylab = "Deviance")
```



```
plot(cv_outcome$k, cv_outcome$dev, type = "b",
     xlab = expression(alpha), ylab = "Deviance")
```



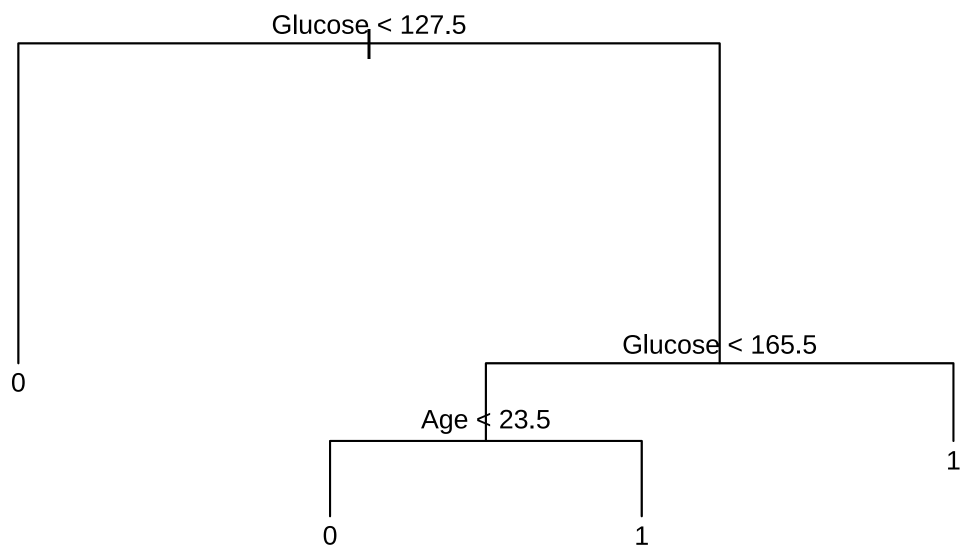
```r
cv_outcome
```

```
## $size
## [1] 17 10  6  4  2  1
##
## $dev
## [1]  94  92  90  87  97 134
##
## $k
## [1] -Inf  0.0  2.5  4.0  8.5 37.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

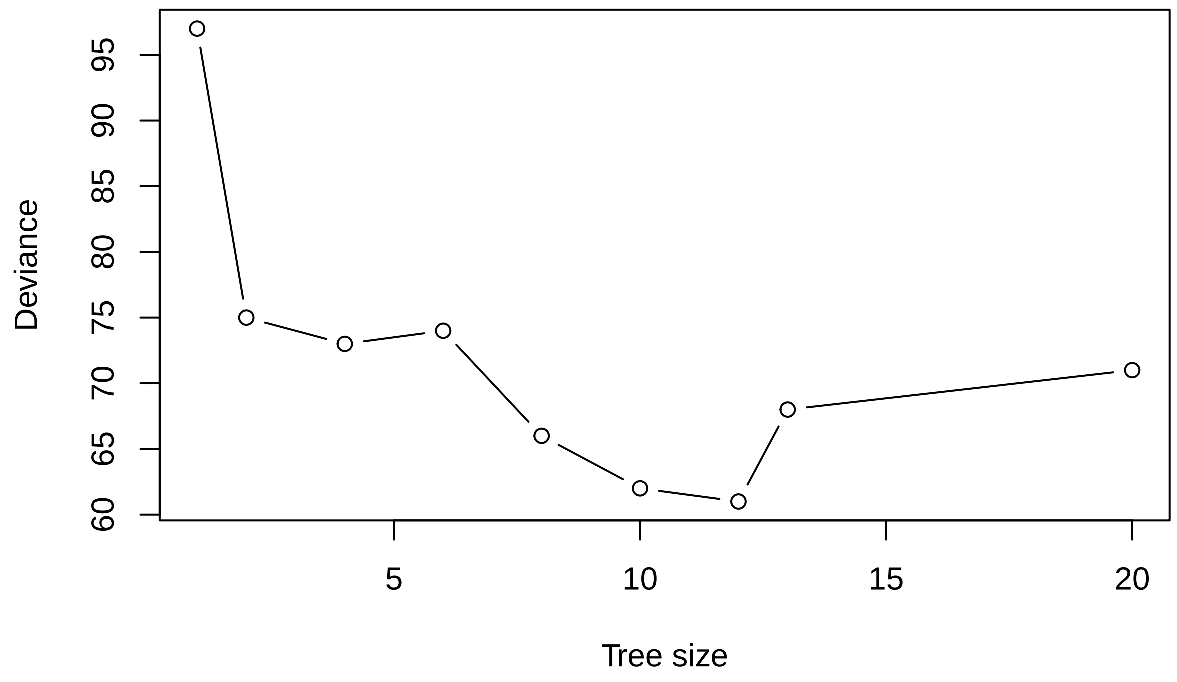
```r
(best_size <- with(cv_outcome, rev(size)[which.min(rev(dev))]))
```

```
## [1] 4
```

```r
prunetree_oc <- prune.misclass(tree_outcome, best = best_size)
plot(prunetree_oc)
text(prunetree_oc, cex = 0.8)
```



```
#Using Split Data
tree_oc <- tree(Outcome ~ ., data = diabetes, subset = train)
cvtree_oc <- cv.tree(tree_oc, FUN = prune.misclass)
plot(cvtree_oc$size, cvtree_oc$dev, type = "b",
     xlab = "Tree size", ylab = "Deviance")
```



```
cvtree_oc
```

```
## $size
## [1] 20 13 12 10  8  6  4  2  1
##
## $dev
## [1] 71 68 61 62 66 74 73 75 97
##
## $k
## [1] -Inf  0.0  1.0  1.5  3.0  3.5  4.0  5.0 25.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
(best_size2 <- with(cvtree_oc, rev(size)[which.min(rev(dev))]))
```

```
## [1] 12
```

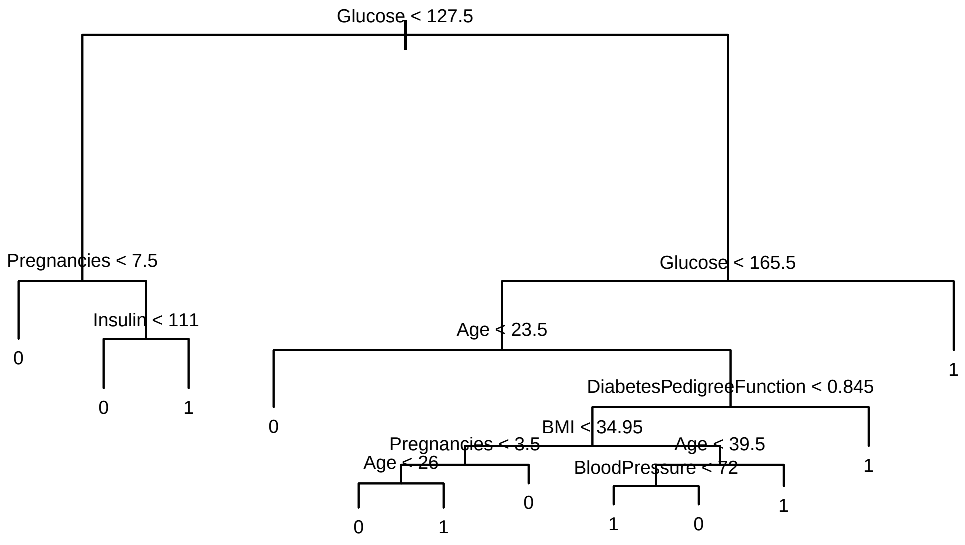
```
prunetree_out <- prune.misclass(tree_oc, best = best_size2)
tree_pred <- predict(prunetree_out, newdata = diab.test, type = "class")
table(tree_pred, Outcome_test)
```

```
##          Outcome_test
## tree_pred  0  1
##         0 77 19
##         1  3 19
```

```
#Classification Accuracy
mean(tree_pred == Outcome_test)
```

```
## [1] 0.8135593
```

```
plot(prunetree_out)
text(prunetree_out, cex = 0.6)
```



Boosting

```
library(gbm)
```

```
## Loaded gbm 2.1.8.1
```

```
library(caret)
```

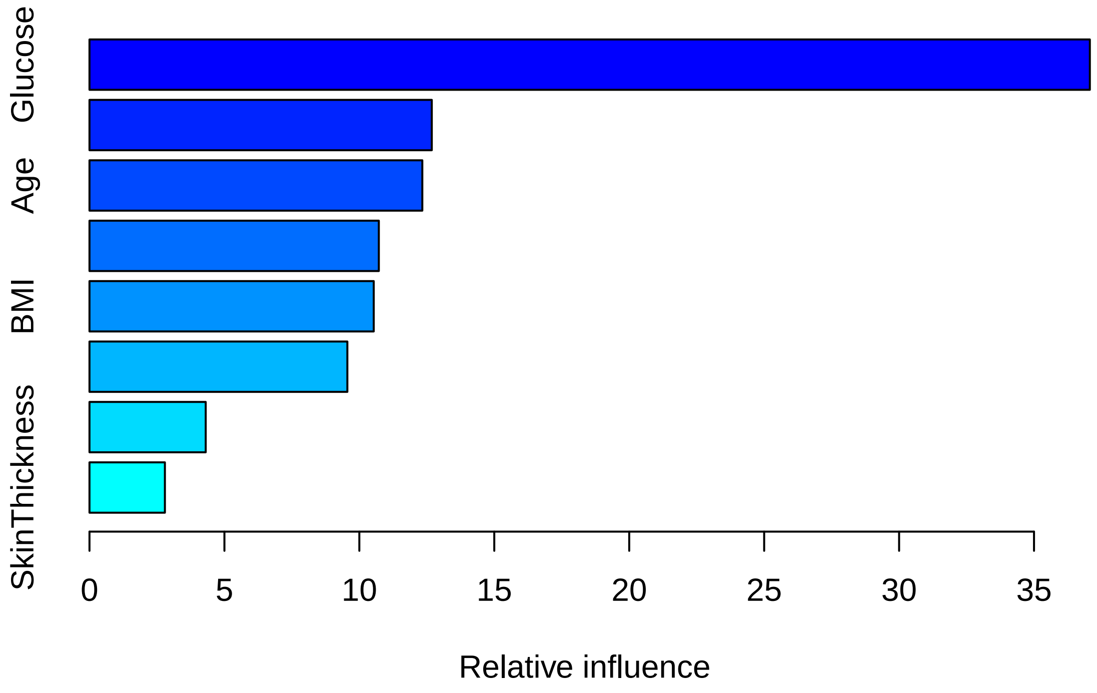
```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
set.seed(1031)

model_gbm = gbm(Outcome ~.,
                data = diab.train,
                distribution = "multinomial",
                cv.folds = 10,
                shrinkage = .01,
                n.trees = 500)       # 500 tress to be built
```

```
## Warning: Setting `distribution = "multinomial"` is ill-advised as it is
## currently broken. It exists only for backwards compatibility. Use at your own
## risk.
```

```
summary(model_gbm)
```



```
##                                                       var   rel.inf
## Glucose                                           Glucose 37.068847
## Insulin                                        Insulin 12.684732
## Age                                                Age 12.333478
## DiabetesPedigreeFunction DiabetesPedigreeFunction 10.722606
## BMI                                                BMI 10.533328
## Pregnancies                                Pregnancies  9.555765
## BloodPressure                            BloodPressure  4.306840
## SkinThickness                            SkinThickness  2.794403
```