Final Project_Mouseketeers

Three Mouseketeers (Co)

2025-12-10

```
library(caret)
```

```
## Loading required package: ggplot2
```

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

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-10
```

```
install.packages("pROC", repos = "https://cran.rstudio.com/")
```

```
##
## The downloaded binary packages are in
##  /var/folders/m0/q7ycmxr14311yyb1w1lfh5680000gn/T//RtmpxdaYqV/downloaded_packages
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(ggplot2)
```

1. Compiled dataframe to include both summaries of glucose reading and patient demographic data.

```
#Downloading the dataframe by Connor, compiled from multiple data frames, including variables
MasterFrame <- read.csv("~/Desktop/Final Project/combined_patient_data.csv")
```

2. Based on literature review and domain knowledge, identify percent out of range as the predicted value (ideal range is between 70 and 180).

```r
#Connor was able to identify percent out of range in the pct_70_180 column.
#I will create a new column "Uncontrolled" which is a binary variable.
#It will be set as 1 if it is uncontrolled and 0 if it is controlled.
#Uncontrolled is defined as pct_70_180 >70 (%)
MasterFrame$Uncontrolled <- ifelse(MasterFrame$pct_70_180 < 70, 1, 0)
```

3. Based on literature review and domain knowledge, include these variables as predictors

Connor has chosen these variables based on his expertise as an endocrinologist. (predictor_vars)

```r
predictor_vars <- c(
  # Demographics
  "AgeAtEnrollment",
  "Gender",
  "Ethnicity",
  "Race",
  "DiagAge",
  "DiagAgeApprox",
  "Weight_kg",
  "Height_cm",
  "BMI",
  "BldPrSys",
  "BldPrDia",
  "EducationLevel",
  "AnnualIncome",
  "InsuranceType",

  # Baseline clinical data
  "HbA1c_Screening",
  "NumMedicalConditions",
  "NumMedications",

  # TDD metrics
  "tdd_mean",
  "tdd_median",
  "tdd_sd",
  "tdd_cv",
  "TDD_per_kg",
  "TDD_per_BMI",

  # Basal insulin metrics
  "basal_mean",
  "basal_median",
  "basal_sd",
  "basal_pct",

  # Bolus insulin metrics
  "bolus_mean",
  "bolus_median",
  "bolus_sd",
  "bolus_pct",
  "mean_boluses_per_day",
```

```
  # Data collection duration
  "n_days"
)
```

I have some concerns regarding the use of tdd and insulin metrics, as it may be a source of leakage for prediction models. (insulin_vars)

```
insulin_vars <- c(
  "tdd_mean", "tdd_median", "tdd_sd", "tdd_cv", "TDD_per_kg", "TDD_per_BMI",
  "basal_mean", "basal_median", "basal_sd", "basal_pct",
  "bolus_mean", "bolus_median", "bolus_sd", "bolus_pct",
  "mean_boluses_per_day"
)
```

This chunk of code will narrow down the table to only include these variables and the Uncontrolled outcome variable.

```
ModelDF_full <- MasterFrame[, c(predictor_vars, "Uncontrolled")]
ModelDF_no_insulin <- MasterFrame[, c(setdiff(predictor_vars, insulin_vars), "Uncontrolled")]

#Drop observations with no outcome
ModelDF_full_outcome <- ModelDF_full[!is.na(ModelDF_full$Uncontrolled), ]
ModelDF_no_insulin_outcome <- ModelDF_no_insulin[!is.na(ModelDF_no_insulin$Uncontrolled), ]

clean_impute <- function(df) {
#Impute missing predictor variables as median
  preProc <- preProcess(df, method = "medianImpute")
  df <- predict(preProc, df)
#Change characters as factors (to prepare for glm)
  char_cols <- sapply(df, is.character)
  df[, char_cols] <- lapply(df[, char_cols], as.factor)

  return(df)
}

ModelDF_full_clean <- clean_impute(ModelDF_full_outcome)
ModelDF_no_insulin_clean <- clean_impute(ModelDF_no_insulin_outcome)
```

4. Split dataframes to Train/Test

```
set.seed(1)

# library(caret) already downloaded above
# Partition function ->  To ensure acceptable ratio of controlled/uncontrolled in train/test sets

#Split Full set into train and test with 80/20 ratio
train_index_full <- createDataPartition(ModelDF_full_clean$Uncontrolled, p = 0.8, list = FALSE)
Train_full <- ModelDF_full_clean[train_index_full, ]
Test_full  <- ModelDF_full_clean[-train_index_full, ]

# Same for No Insulin Set
train_index_no_insulin <- createDataPartition(ModelDF_no_insulin_clean$Uncontrolled, p = 0.8, list = FA
```

```r
Train_no_insulin <- ModelDF_no_insulin_clean[train_index_no_insulin, ]
Test_no_insulin  <- ModelDF_no_insulin_clean[-train_index_no_insulin, ]

# Optional: check proportions
prop.table(table(Train_full$Uncontrolled))
```

```
##
##         0         1
## 0.5666667 0.4333333
```

```r
prop.table(table(Test_full$Uncontrolled))
```

```
##
##         0         1
## 0.6363636 0.3636364
```

```r
prop.table(table(Train_no_insulin$Uncontrolled))
```

```
##
##         0         1
## 0.5888889 0.4111111
```

```r
prop.table(table(Test_no_insulin$Uncontrolled))
```

```
##
##         0         1
## 0.5454545 0.4545455
```

5. Ridge/Lasso to narrow down variables to 6 most relevant to prediction

Options for variable selection include: Subset selection (not good for variables that might have multicollinearity), ridge/lasso (shrinking some variables close to zero and zero respectively) and principle component analysis (PCA, which creates new variables which are amalgamations of other variables, not very interpretable).

Ridge/Lasso is a good balance between interpretability and handling collinearity.

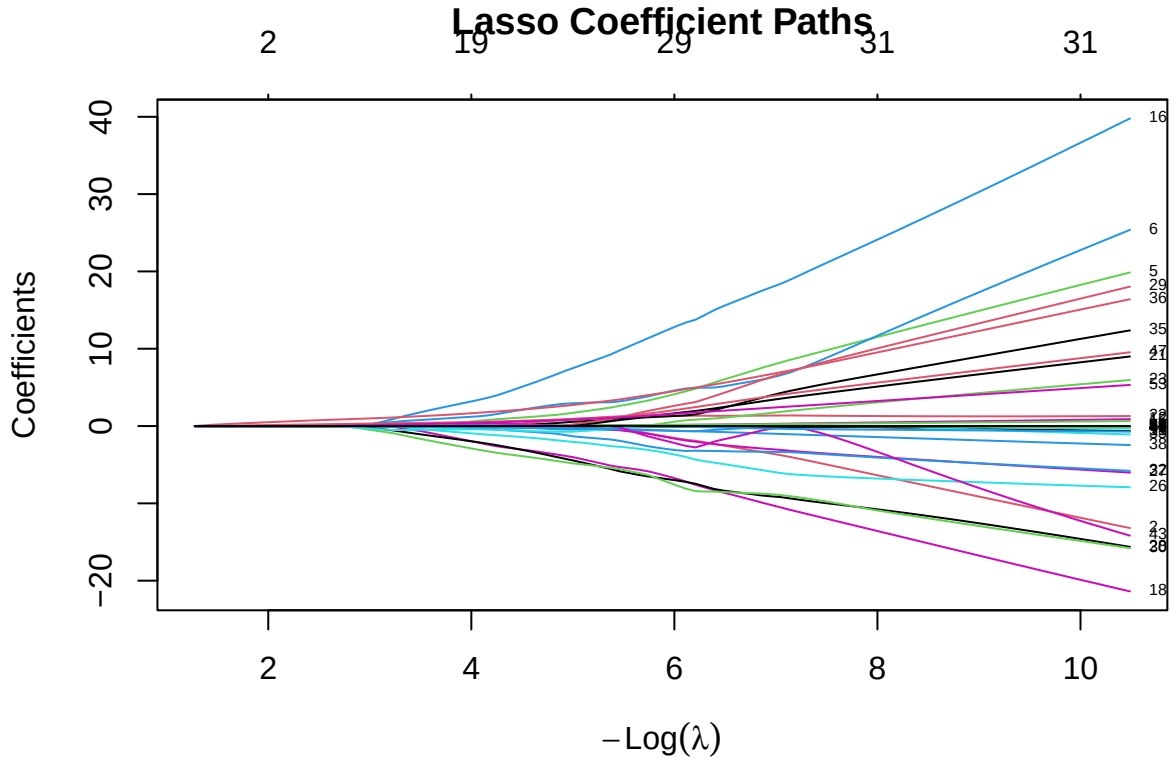
```r
# library(glmnet) downloaded above

# setting the predictors (x) and outcome (y)
x <- model.matrix(Uncontrolled ~ ., data = Train_full)[,-1]
y <- Train_full$Uncontrolled

# Cross-validation to find best lambda
cv.lasso <- cv.glmnet(x, y, alpha = 1, family = "binomial")
best_lambda_lasso <- cv.lasso$lambda.min
best_lambda_lasso
```

```
## [1] 0.06909099
```

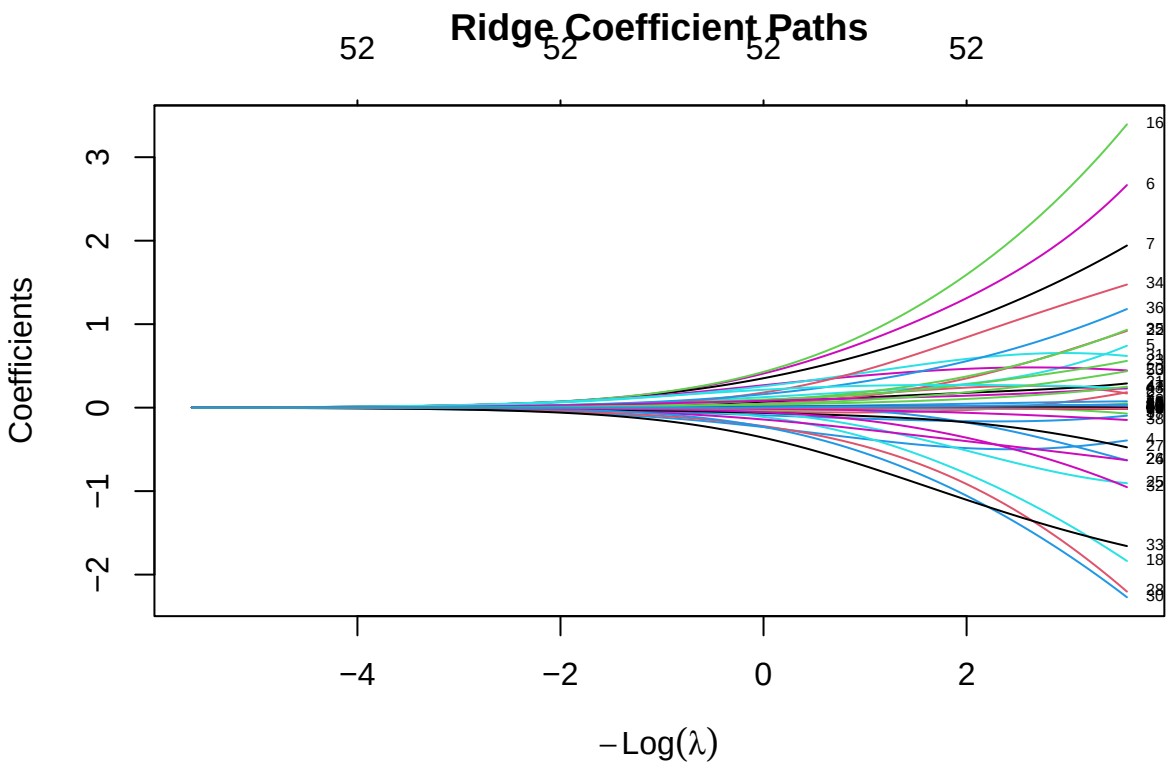
```r
# Cross-validation for best lambda
cv.ridge <- cv.glmnet(x, y, alpha = 0, family = "binomial")
best_lambda_ridge <- cv.ridge$lambda.min
best_lambda_ridge
```

```
## [1] 0.2159588
```

```r
# Lasso coefficient path
lasso.mod <- glmnet(x, y, alpha = 1, family = "binomial")
plot(lasso.mod, xvar = "lambda", label = TRUE, main = "Lasso Coefficient Paths")
```




```r
# Ridge coefficient path
ridge.mod <- glmnet(x, y, alpha = 0, family = "binomial")
plot(ridge.mod, xvar = "lambda", label = TRUE, main = "Ridge Coefficient Paths")
```

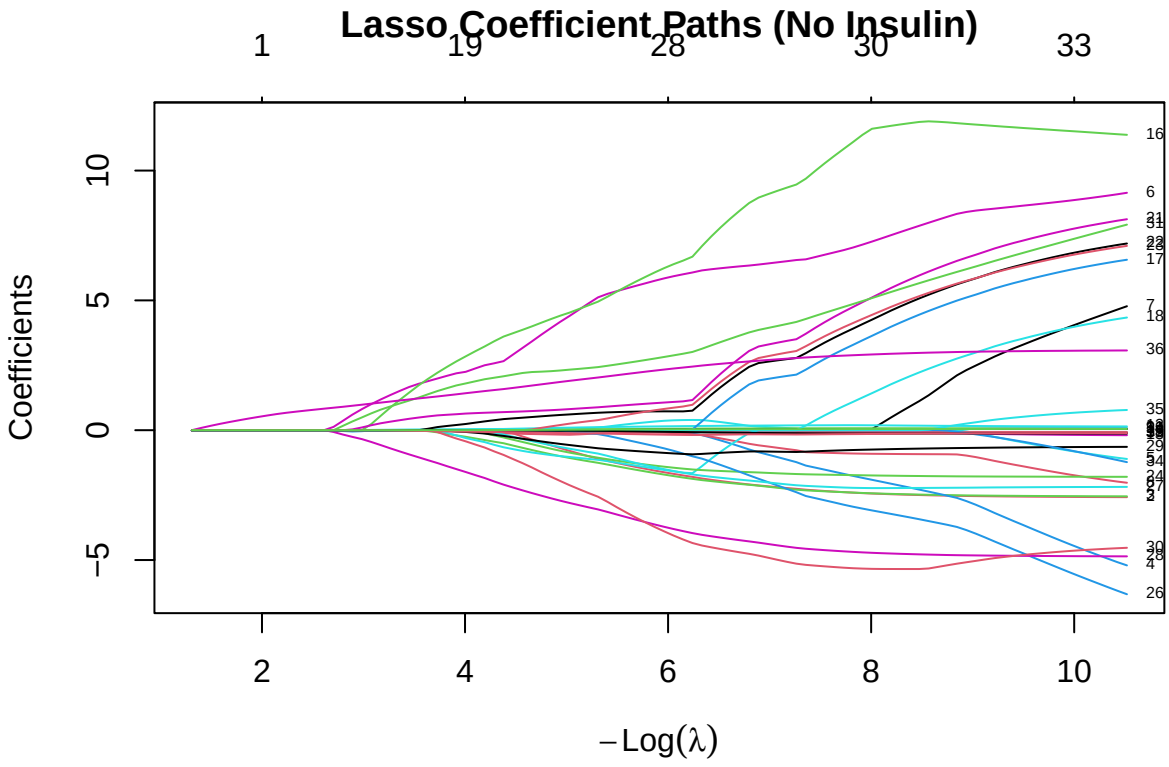



```
#I ultimately chose to use lasso, which sets other variables to zero
coef_lasso <- predict(lasso.mod, type = "coefficients", s = best_lambda_lasso)
nonzero_vars_full <- rownames(coef_lasso)[coef_lasso[,1] != 0]
print(nonzero_vars_full)
```

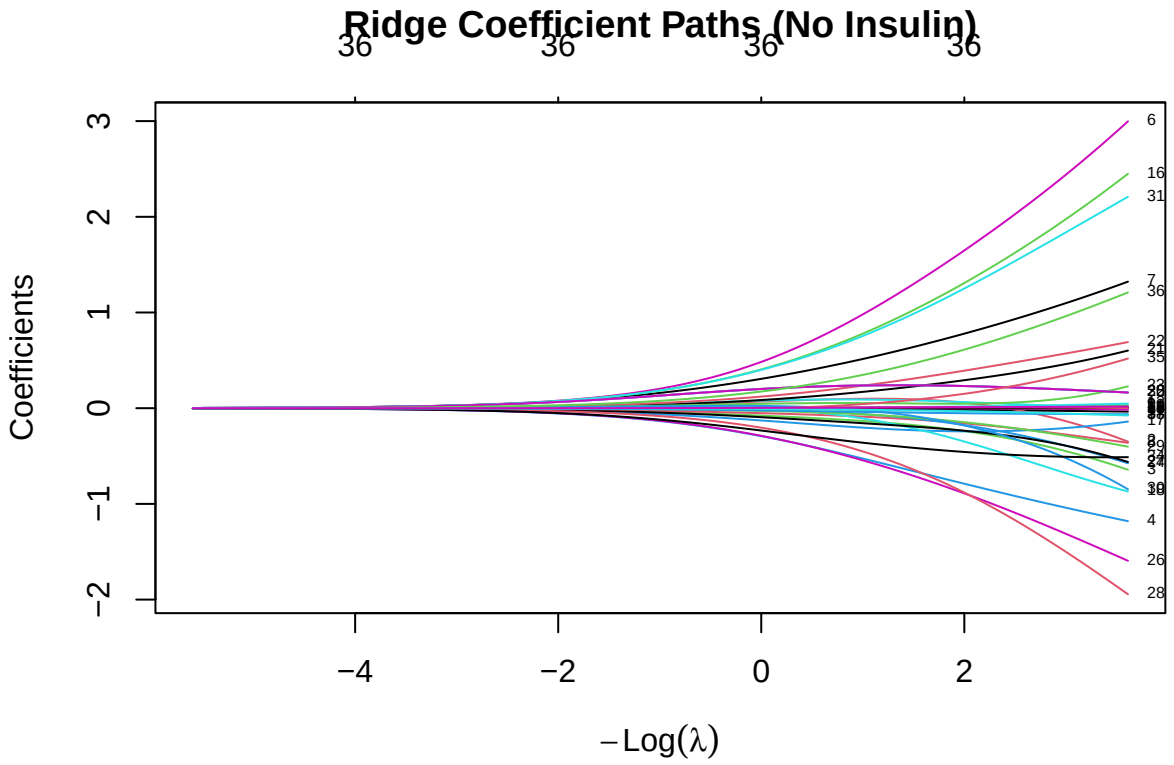
```
## [1] "(Intercept)"          "HbA1c_Screening"      "basal_sd"
## [4] "mean_boluses_per_day"
```

Do the same for the no insulin set to see if there are the same predictors

```
x_no_insulin <- model.matrix(Uncontrolled ~ ., data = Train_no_insulin)[,-1]
y_no_insulin <- Train_no_insulin$Uncontrolled

# Lasso
cv.lasso_no_insulin <- cv.glmnet(x_no_insulin, y_no_insulin, alpha = 1, family = "binomial")
best_lambda_lasso_no_insulin <- cv.lasso_no_insulin$lambda.min

# Ridge
cv.ridge_no_insulin <- cv.glmnet(x_no_insulin, y_no_insulin, alpha = 0, family = "binomial")
best_lambda_ridge_no_insulin <- cv.ridge_no_insulin$lambda.min

# Visualization
lasso.mod_no_insulin <- glmnet(x_no_insulin, y_no_insulin, alpha = 1, family = "binomial")
plot(lasso.mod_no_insulin, xvar = "lambda", label = TRUE, main = "Lasso Coefficient Paths (No Insulin)")
```




```
ridge.mod_no_insulin <- glmnet(x_no_insulin, y_no_insulin, alpha = 0, family = "binomial")
plot(ridge.mod_no_insulin, xvar = "lambda", label = TRUE, main = "Ridge Coefficient Paths (No Insulin)")
```




```
# Get nonzeros from lasso with best lambda
coef_lasso_no_insulin <- predict(lasso.mod_no_insulin, type = "coefficients", s = best_lambda_lasso_no_
nonzero_vars_no_insulin <- rownames(coef_lasso_no_insulin)[coef_lasso_no_insulin[,1] != 0]
print(nonzero_vars_no_insulin)
```

```
## [1] "(Intercept)"     "AgeAtEnrollment" "HbA1c_Screening"
```

6. Create a glm model based on the selected variables using the train data set

```
glm_model_full <- glm(Uncontrolled ~
                                  HbA1c_Screening +
                                  basal_sd +
                                  mean_boluses_per_day,
                                  data = Train_full,
                                  family = binomial)

summary(glm_model_full)
```

```
##
## Call:
## glm(formula = Uncontrolled ~ HbA1c_Screening + basal_sd + mean_boluses_per_day,
##     family = binomial, data = Train_full)
##
## Coefficients:
```

```
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -22.5882     5.4244  -4.164 3.13e-05 ***
## HbA1c_Screening        1.9522     0.5453   3.580 0.000344 ***
## basal_sd               0.6073     0.1539   3.945 7.98e-05 ***
## mean_boluses_per_day   0.4621     0.1957   2.361 0.018235 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 123.162  on 89  degrees of freedom
## Residual deviance:  63.387  on 86  degrees of freedom
## AIC: 71.387
##
## Number of Fisher Scoring iterations: 6
```

Same for no insulin

```
glm_model_no_insulin <- glm(Uncontrolled ~ AgeAtEnrollment +
                                           HbA1c_Screening,
                                           data = Train_no_insulin,
                                           family = binomial)
summary(glm_model_no_insulin)
```

```
##
## Call:
## glm(formula = Uncontrolled ~ AgeAtEnrollment + HbA1c_Screening,
##     family = binomial, data = Train_no_insulin)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -10.22352    2.87993  -3.550 0.000385 ***
## AgeAtEnrollment  -0.02915    0.01747  -1.669 0.095134 .
## HbA1c_Screening   1.41895    0.36472   3.891 0.000100 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 121.907  on 89  degrees of freedom
## Residual deviance:  86.486  on 87  degrees of freedom
## AIC: 92.486
##
## Number of Fisher Scoring iterations: 5
```

7. Cross validate Full

```
# 1. Predict probabilities on Test_full
prob_full <- predict(glm_model_full, newdata = Test_full, type = "response")

# 2. Convert probabilities to binary predictions (threshold 0.5)
pred_full <- ifelse(prob_full > 0.5, 1, 0)
```

```r
# 3. Create Confusion Matrix
conf_matrix_full <- table(Actual = Test_full$Uncontrolled, Predicted = pred_full)
print(conf_matrix_full)
```

```
##       Predicted
## Actual  0  1
##      0 10  4
##      1  2  6
```

```r
accuracy_full <- sum(diag(conf_matrix_full)) / sum(conf_matrix_full) * 100
print(accuracy_full)
```

```
## [1] 72.72727
```

```r
# 1. Predict probabilities on Test_no_insulin
prob_no_insulin <- predict(glm_model_no_insulin, newdata = Test_no_insulin, type = "response")

# 2. Convert probabilities to binary predictions (threshold 0.5)
pred_no_insulin <- ifelse(prob_no_insulin > 0.5, 1, 0)

# 3. Create Confusion Matrix
conf_matrix_no_insulin <- table(Actual = Test_no_insulin$Uncontrolled, Predicted = pred_no_insulin)
print(conf_matrix_no_insulin)
```

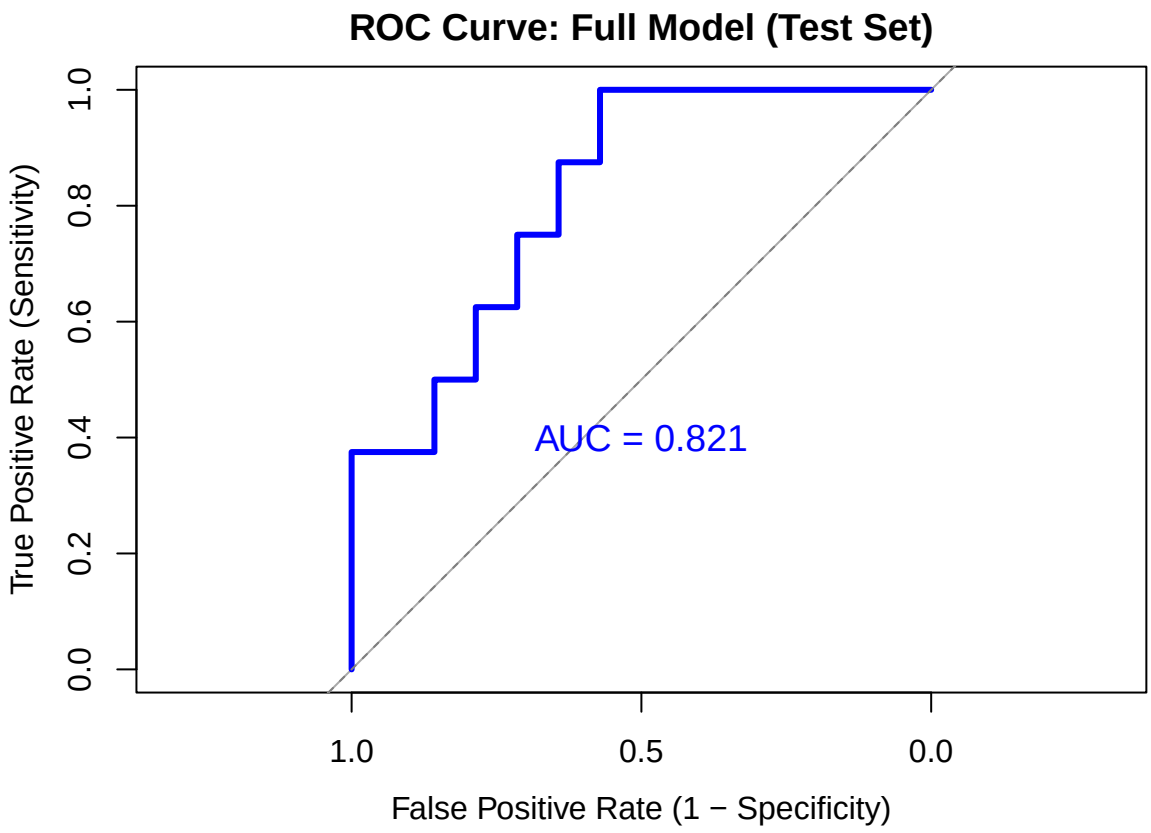
```
##       Predicted
## Actual 0 1
##      0 9 3
##      1 5 5
```

```r
accuracy_no_insulin <- sum(diag(conf_matrix_no_insulin)) / sum(conf_matrix_no_insulin) * 100
print(accuracy_no_insulin)
```

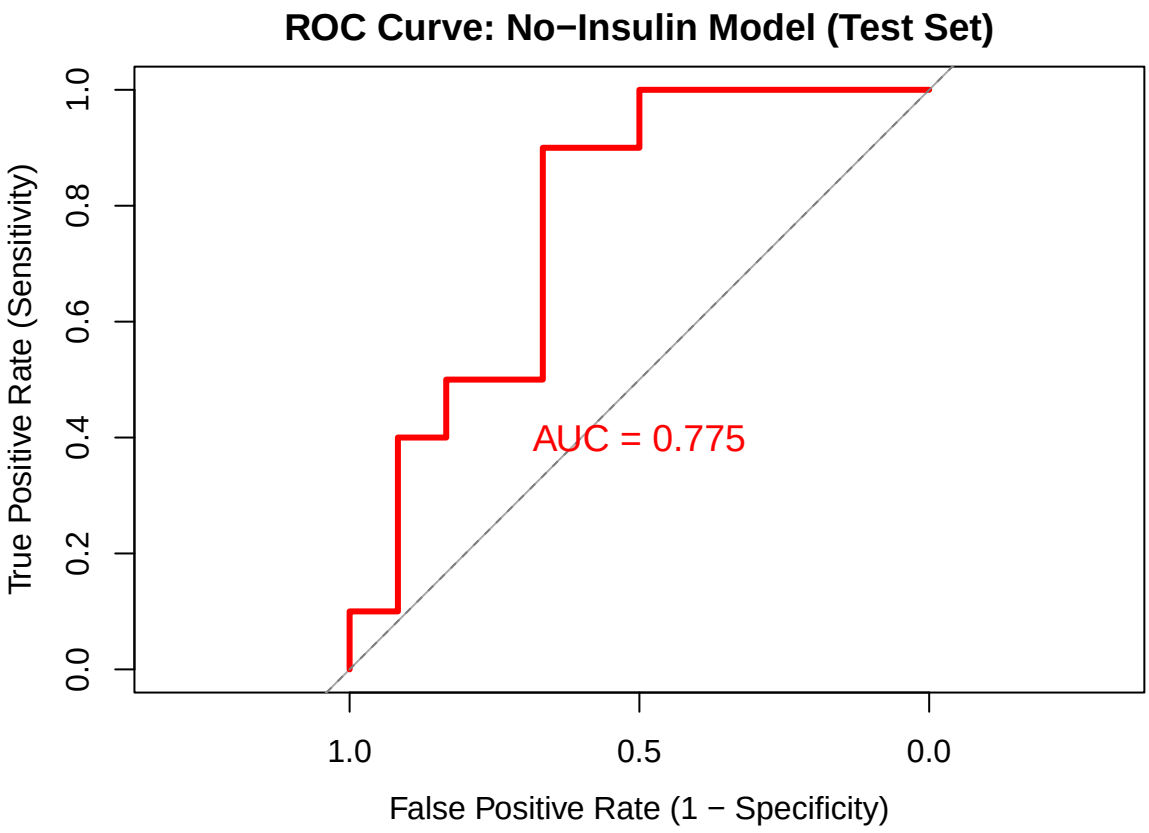
```
## [1] 63.63636
```

8. Creating AUC

```r
# already installed library(pROC)
roc_full <- roc(response = Test_full$Uncontrolled, predictor = prob_full, quiet = TRUE)
plot(roc_full,
     col = "blue",
     lwd = 3,
     main = "ROC Curve: Full Model (Test Set)",
     xlab = "False Positive Rate (1 - Specificity)",
     ylab = "True Positive Rate (Sensitivity)")

# Add the diagonal line (random guess)
abline(a = 1, b = -1, lty = 2, col = "gray50")

text(x = 0.5, y = 0.4,
     labels = paste("AUC =", round(auc(roc_full), 3)),
     col = "blue",
     cex = 1.2)
```




```
roc_no_insulin <- roc(response = Test_no_insulin$Uncontrolled, predictor = prob_no_insulin, quiet = TRU

plot(roc_no_insulin,
     col = "red",
     lwd = 3,
     main = "ROC Curve: No-Insulin Model (Test Set)",
     xlab = "False Positive Rate (1 - Specificity)",
     ylab = "True Positive Rate (Sensitivity)")

# Add the diagonal line (random guess)
abline(a = 1, b = -1, lty = 2, col = "gray50")

# Add AUC as text on the plot
text(x = 0.5, y = 0.4,
     labels = paste("AUC =", round(auc(roc_no_insulin), 3)),
     col = "red",
     cex = 1.2)
```


ROC Curve: No−Insulin Model (Test Set)
True Positive Rate (Sensitivity)
0.0
0.2
0.4
0.6
0.8
1.0
AUC = 0.775
1.0
0.5
0.0
False Positive Rate (1 − Specificity)