Diabetes Dataset Testing

2023-09-24

Functions

```r
count_na_per_column <- function(df) {
    sapply(df, function(x) sum(is.na(x)))
}

# Create a function to calculate metrics
calc_metrics <- function(pred, true) {
  confusion <- table(pred, true)
  TP <- confusion[2, 2]
  FP <- confusion[2, 1]
  TN <- confusion[1, 1]
  FN <- confusion[1, 2]
  Sensitivity <- TP / (TP + FN)
  Specificity <- TN / (TN + FP)
  Accuracy <- (TP + TN) / (TP + FP + TN + FN)
  Precision <- TP / (TP + FP)
  F1 <- 2 * (Precision * Sensitivity) / (Precision + Sensitivity)
  pred_obj <- prediction(as.numeric(pred), as.numeric(true))
  perf <- performance(pred_obj, "auc")
  AUC <- as.numeric(perf@y.values)
  return(c(AUC, Sensitivity, Specificity, F1, Accuracy))
}


### A string like "Logistic" has to be put in. <--- from the colnames of df_coef
importance_plot <- function(model_string) {

# Create a data frame for plotting
coef_df <- data.frame(
  Variable = rownames(df_coef)[-1],  # Exclude the intercept
  Importance = abs(df_coef[-1, model_string])  # Exclude the intercept
)

# Order the variables by importance
coef_df <- coef_df[order(coef_df$Importance), ]

# Plot
ggplot(coef_df, aes(x = reorder(Variable, Importance), y = Importance, fill = Importance)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "lightblue2", high = "lightblue3") +
  theme_minimal() +
  theme(axis.text.y = element_text(size = 12),
        axis.text.x = element_text(size = 10),
        title = element_text(size = 15),
        axis.title.y = element_blank(),
        axis.title.x = element_blank(),
        legend.position = "none") +
  coord_flip() +
  ggtitle(paste(model_string, "Model - Variable Importance"))

}
```

5. Data Processing

```r
# Loading the data
data("PimaIndiansDiabetes2", package = "mlbench")
diabetes <- PimaIndiansDiabetes2
diabetes$diabetes <- as.factor(ifelse(diabetes$diabetes == "pos", 1, 0))
#diabetes$diabetes <- ifelse(diabetes$diabetes == "pos", 1, 0)
#kable(t(count_na_per_column(diabetes)))
```

Removing NA's

```r
diabetes <- na.omit(diabetes)
#kable(t(count_na_per_column(diabetes)))
```

Removing Outliers

```r
outliers <- check_outliers(diabetes, method = "mahalanobis")
#plot(outliers)
outliers <- as.vector(outliers)

#diabetes <- diabetes[!outliers, ]
```

Data for models

```r
# Split
set.seed(222)
n <- nrow(diabetes)
training.samples <- sample(1:n, size = 0.75 * n)
train.data <- diabetes[training.samples, ]
scaled_train.data <- scale(train.data[, 1:8])
train.data[, 1:8] <- scaled_train.data
#train.data <- smote(diabetes ~ ., train.data, perc.over = 1)
test.data <- diabetes[-training.samples, ]
scaled_test.data <- scale(test.data[, 1:8])
test.data[, 1:8] <- scaled_test.data
#test.data <- na.omit(test.data)



# Handle NA's in training set
#mice <- complete(mice(subset(train.data, select = -c(triceps, insulin)), method='rf', seed = 123))
#mice <- complete(mice(train.data, method='rf', seed = 123))

#train.data$glucose <- mice$glucose
```

```r
#train.data$pressure <- mice$pressure
#train.data$mass <- mice$mass

#train.data <- na.omit(train.data)

#mice.triceps <- complete(mice(subset(train.data, select = -insulin), method='rf', seed = 123))
#train.data$triceps <- mice.triceps$triceps

#mice.insulin <- complete(mice(train.data, method='rf', seed = 123))
#train.data$insulin <- mice.insulin$insulin

# Train
X.train <- model.matrix(diabetes~., data = train.data)[,-1]
#X.train <- scale(X.train)
y.train <- train.data$diabetes

# Test
X.test <- model.matrix(diabetes ~., data = test.data)[,-1]
#X.test <- scale(X.test)
y.test <- test.data$diabetes
```

Logistic Regression

```r
set.seed(123)
log.model <- glm(diabetes ~., data = train.data, family = "binomial")

# Make predictions
probabilities <- predict(log.model, newdata = test.data, type = "response")
predicted.classes.log <- as.factor(ifelse(probabilities > 0.5, 1, 0))

# Accuracy
# mean(predicted.classes == y.test)
log.conf <- confusionMatrix(predicted.classes.log, y.test, positive = "1")
log.conf
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction  0  1
##          0 61 10
##          1  6 21
##
##                Accuracy : 0.8367
##                  95% CI : (0.7484, 0.9037)
##     No Information Rate : 0.6837
##     P-Value [Acc > NIR] : 0.0004538
##
##                   Kappa : 0.609
##
##  Mcnemar's Test P-Value : 0.4532547
##
##             Sensitivity : 0.6774
##             Specificity : 0.9104
##          Pos Pred Value : 0.7778
##          Neg Pred Value : 0.8592
##              Prevalence : 0.3163
##          Detection Rate : 0.2143
##    Detection Prevalence : 0.2755
##       Balanced Accuracy : 0.7939
##
##        'Positive' Class : 1
##
```

LASSO

```
set.seed(123)
cv.lasso.model <- cv.glmnet(X.train, y.train, alpha = 1, family = "binomial",
                            intercept = T)
#plot(cv.lasso.model)
cbind(coef(cv.lasso.model, s = cv.lasso.model$lambda.min), coef(cv.lasso.model, s = cv.lasso.model$lamb
```

```
## 9 x 2 sparse Matrix of class "dgCMatrix"
##                     s1          s1
## (Intercept) -0.9087548 -0.79367862
## pregnant     0.2500270  0.04713706
## glucose      1.0571306  0.80577102
## pressure     .          .
## triceps      0.0626410  .
## insulin      .          .
## mass         0.4184043  0.19426613
## pedigree     0.3622318  0.10361089
## age          0.2235214  0.17759602
```

```
# Make predictions
probabilities <- predict(cv.lasso.model, newx = X.test, s = cv.lasso.model$lambda.min, type = "response
predicted.classes.lasso.min <- as.factor(ifelse(probabilities > 0.5, 1, 0))

probabilities <- predict(cv.lasso.model, newx = X.test, s = cv.lasso.model$lambda.1se, type = "response
predicted.classes.lasso.1se <- as.factor(ifelse(probabilities > 0.5, 1, 0))

# Accuracy
#mean(predicted.classes.lasso.min == y.test)
lasso.min.conf <- confusionMatrix(predicted.classes.lasso.min, y.test, positive = "1")
lasso.min.conf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 61 10
##          1  6 21
##
##                Accuracy : 0.8367
##                  95% CI : (0.7484, 0.9037)
##     No Information Rate : 0.6837
##     P-Value [Acc > NIR] : 0.0004538
##
##                   Kappa : 0.609
```

```
##
##  Mcnemar's Test P-Value : 0.4532547
##
##             Sensitivity : 0.6774
##             Specificity : 0.9104
##          Pos Pred Value : 0.7778
##          Neg Pred Value : 0.8592
##              Prevalence : 0.3163
##          Detection Rate : 0.2143
##    Detection Prevalence : 0.2755
##       Balanced Accuracy : 0.7939
##
##        'Positive' Class : 1
##
```

```
lasso.1se.conf <- confusionMatrix(predicted.classes.lasso.1se, y.test, positive = "1")
lasso.1se.conf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 62 12
##          1  5 19
##
##                Accuracy : 0.8265
##                  95% CI : (0.7369, 0.8956)
##     No Information Rate : 0.6837
##     P-Value [Acc > NIR] : 0.001064
##
##                   Kappa : 0.573
##
##  Mcnemar's Test P-Value : 0.145610
##
##             Sensitivity : 0.6129
##             Specificity : 0.9254
##          Pos Pred Value : 0.7917
##          Neg Pred Value : 0.8378
##              Prevalence : 0.3163
##          Detection Rate : 0.1939
##    Detection Prevalence : 0.2449
##       Balanced Accuracy : 0.7691
##
##        'Positive' Class : 1
##
```

in data: set.seed(123) and train 0.75 all others see(123) i like it a lot. min and 1se very sparse and good pred, just not much difference in amount of sparsity and amount of FN

in data: set.seed(2) and train 0.75 all others see(123) i like it a lot. 1se sparse and more FN but same pred

in data: set.seed(222) and train 0.75 all others see(123) pretty good similar too above

in data: set.seed(6) and train 0.75 all others see(123) gives very sparse 1se lasso and okay pred

in data: set.seed(13) and train 0.75 all others see(123) pretty nice! sparse 1se, but pred is same

in data: set.seed(42) and train 0.75 all others see(123) pretty nice! very very sparse 1se, more FN in 1se some things we see:

- the more conservative model (1se) usually has more FN —> it makes the safe/conservative choice of going for the class that has 2/3 of observations

Ridge

```
set.seed(123)
cv.ridge.model <- cv.glmnet(X.train, y.train, alpha = 0, family = "binomial", intercept = T)
#plot(cv.ridge)

# Make predictions
probabilities <- predict(cv.ridge.model, newx = X.test, s = cv.ridge.model$lambda.min, type = "response
predicted.classes.ridge <- as.factor(ifelse(probabilities > 0.5, 1, 0))

# Accuracy
#mean(predicted.classes.ridge == y.test)
ridge.conf <- confusionMatrix(as.factor(predicted.classes.ridge), y.test, positive = "1")
ridge.conf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 62 10
##          1  5 21
##
##                Accuracy : 0.8469
##                  95% CI : (0.7601, 0.9117)
##     No Information Rate : 0.6837
##     P-Value [Acc > NIR] : 0.0001806
##
##                   Kappa : 0.6301
##
##  Mcnemar's Test P-Value : 0.3016996
##
##             Sensitivity : 0.6774
##             Specificity : 0.9254
##          Pos Pred Value : 0.8077
##          Neg Pred Value : 0.8611
##              Prevalence : 0.3163
##          Detection Rate : 0.2143
##    Detection Prevalence : 0.2653
##       Balanced Accuracy : 0.8014
##
##        'Positive' Class : 1
##
```

Elastic Net

```
set.seed(123)

# CV and tuning grid
```

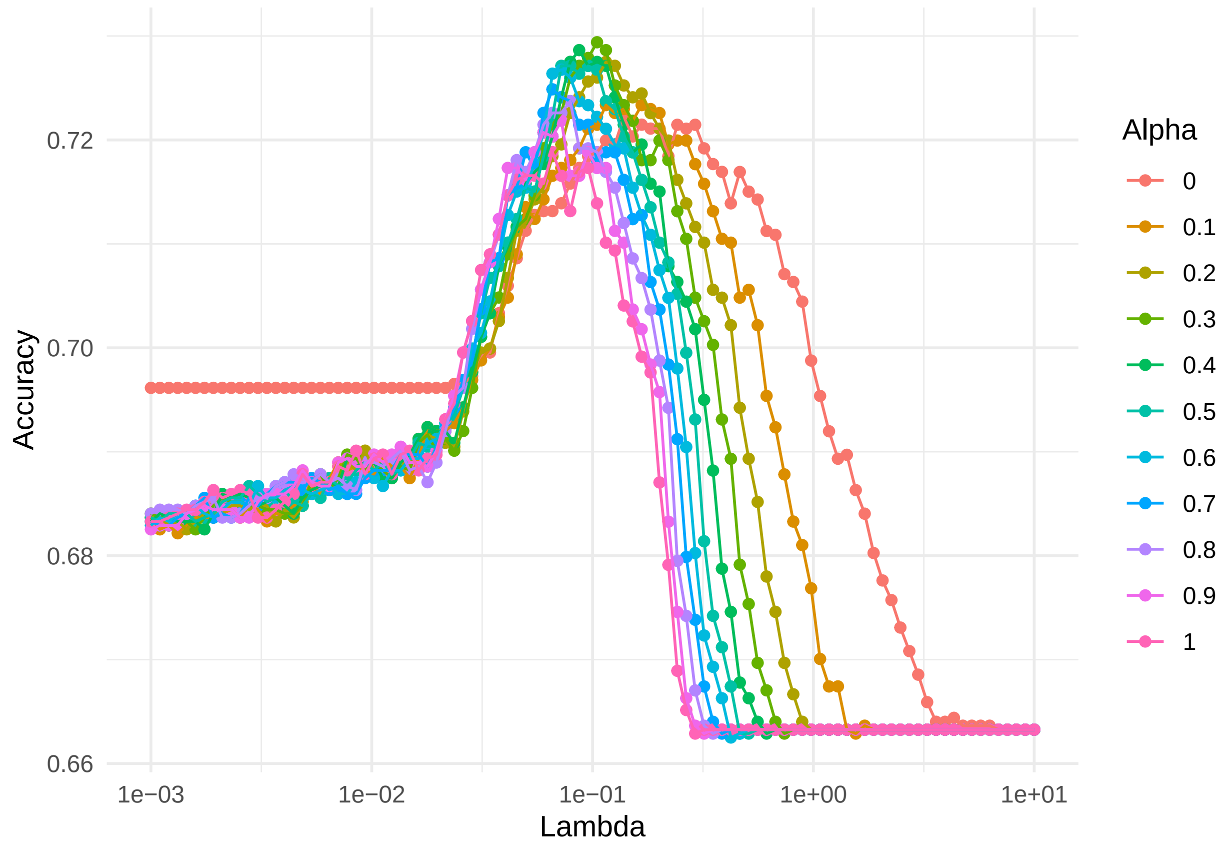
```r
myFolds <- createFolds(y.train, k = 10, list = TRUE)
myControl <- trainControl(index = myFolds)

#cvControl <- trainControl(method = "cv", number = 10)
tuneGrid <- expand.grid(alpha = seq(0, 1, by = 0.05), lambda = 10^seq(1, -3, length=100))

# Train model
elasticnet.model <- train(X.train, y.train, method = "glmnet", trControl = myControl,
                          tuneGrid = tuneGrid, intercept = T, family = "binomial")

# Make predictions
probabilities <- predict(elasticnet.model, newdata = X.test, type = "prob") # "raw" already outputs pre
predicted.classes.elasticnet <- ifelse(probabilities$`1` > 0.5, 1, 0)

# Accuracy
# mean(predicted.classes == y.test)
elasticnet.conf <- confusionMatrix(as.factor(predicted.classes.elasticnet), y.test, positive = "1")
elasticnet.conf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 63 11
##          1  4 20
##
##                Accuracy : 0.8469
##                  95% CI : (0.7601, 0.9117)
##     No Information Rate : 0.6837
##     P-Value [Acc > NIR] : 0.0001806
##
##                   Kappa : 0.6233
##
##  Mcnemar's Test P-Value : 0.1213353
##
##             Sensitivity : 0.6452
##             Specificity : 0.9403
##          Pos Pred Value : 0.8333
##          Neg Pred Value : 0.8514
##              Prevalence : 0.3163
##          Detection Rate : 0.2041
##    Detection Prevalence : 0.2449
##       Balanced Accuracy : 0.7927
##
##        'Positive' Class : 1
##
```

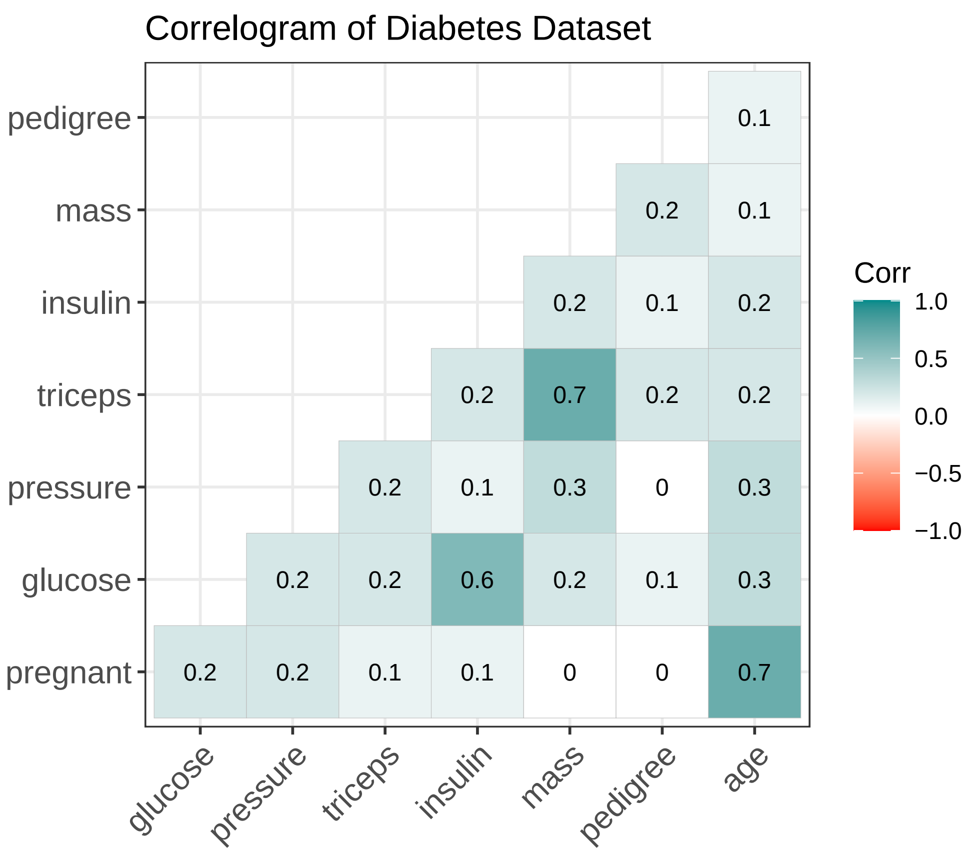
```r
# Output coefficients
coef(elasticnet.model$finalModel, s = elasticnet.model$bestTune$lambda)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) -0.783644314
## pregnant     0.124519132
## glucose      0.616288277
## pressure     0.003625147
## triceps      0.073458775
## insulin      0.067361284
## mass         0.200698201
## pedigree     0.158043114
## age          0.188599082
```

```
# Filter data to only include specific alpha values
filtered_data <- subset(elasticnet.model$results, alpha %in% seq(0, 1, by=0.1))

# Create the plot
ggplot(filtered_data, aes(x=lambda, y=Accuracy, color=factor(alpha))) +
  geom_point() +
  geom_line() +
  scale_x_log10() +
  scale_color_discrete(name = "Alpha") +
  labs(x = "Lambda", y = "Accuracy") +
  theme_minimal()
```

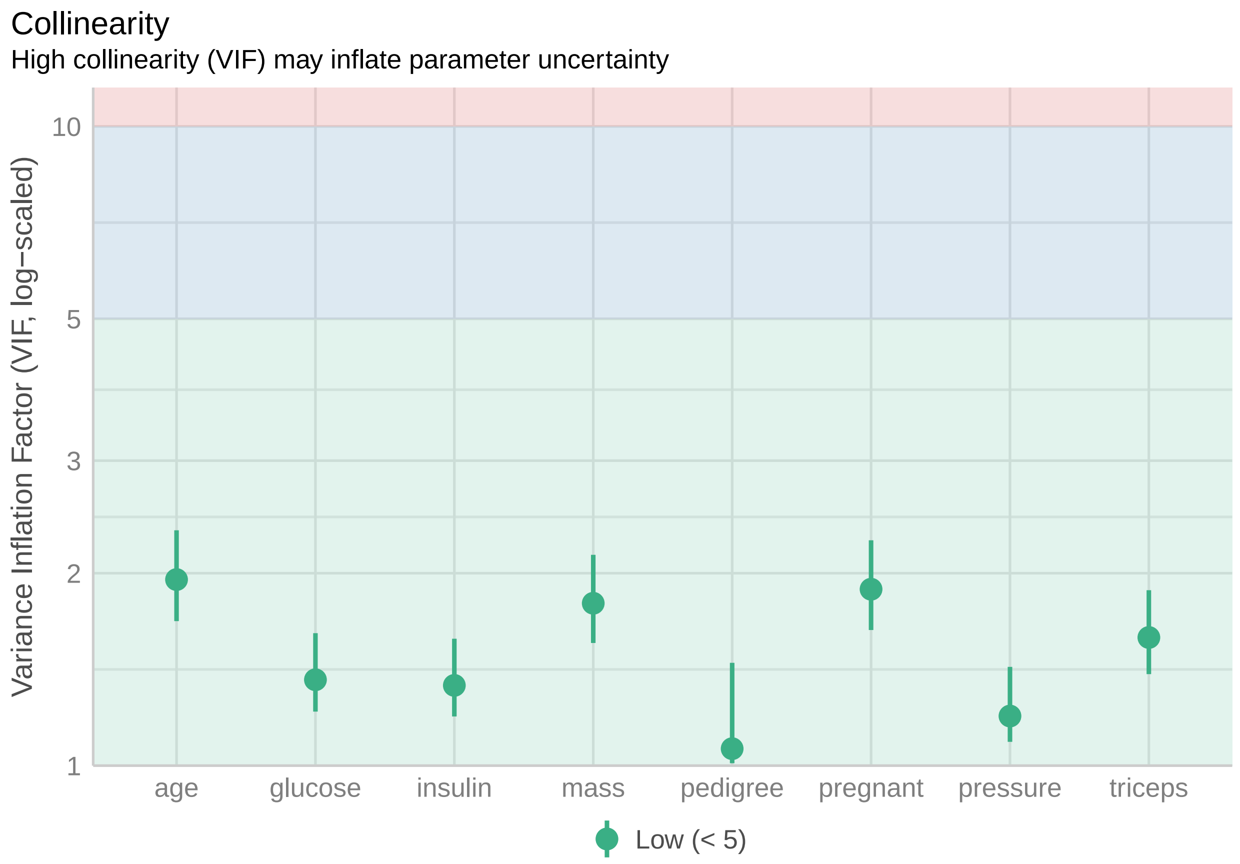


```
library(ggcorrplot)
# Ensure that all remaining columns after subsetting are numeric
numeric_data <- subset(diabetes, select = -c(diabetes))
corr <- round(cor(numeric_data), 1)
ggcorrplot(corr,
           type = "lower",
           lab = TRUE,
           lab_size = 3,
           colors = c("red", "white", "cyan4"),
           title = "Correlogram of Diabetes Dataset",
           ggtheme = theme_bw())
```



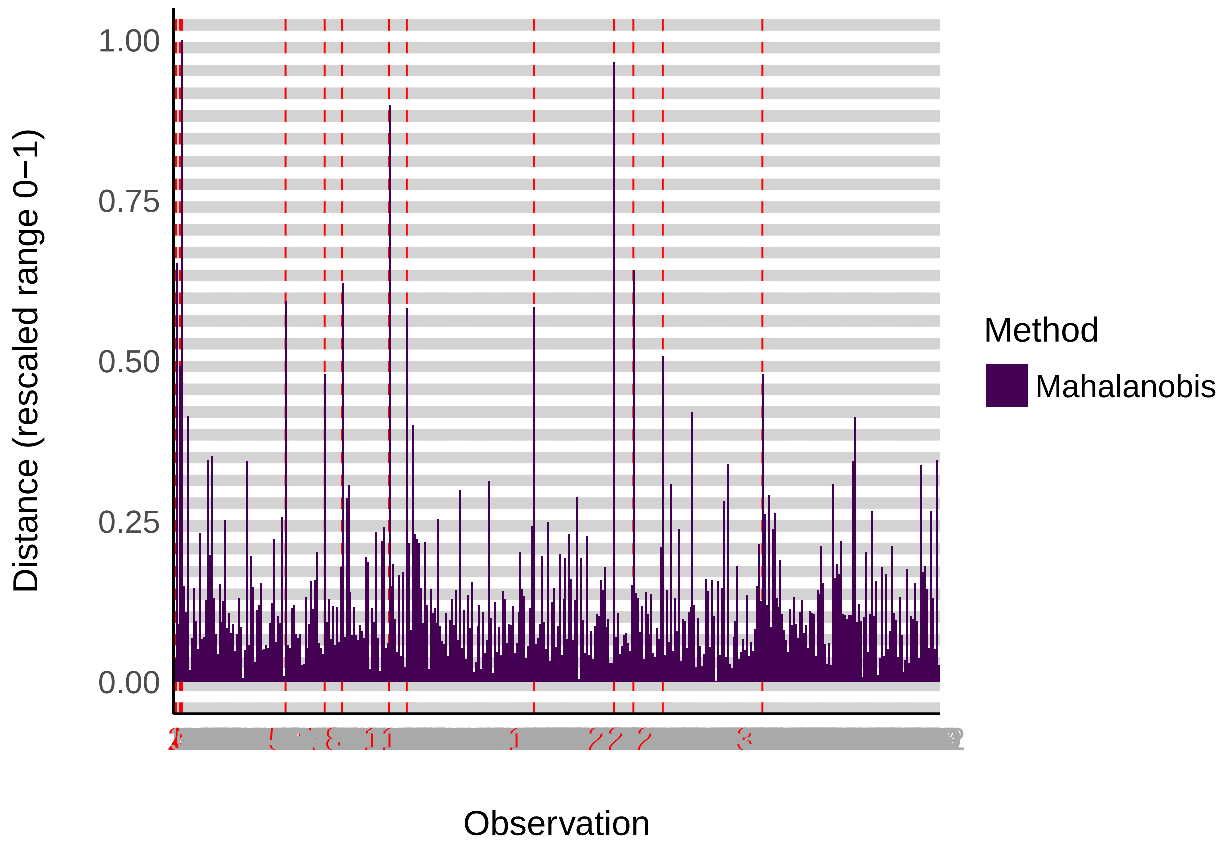
```
result <- check_collinearity(log.model)
plot(result)
```

```
## Variable ‘Component‘ is not in your data frame :/
```



### Outliers

```r
result <- check_outliers(diabetes, method = "mahalanobis")
plot(result, type = "dots")
```



```r
result
```

```
## 13 outliers detected: cases 2, 4, 5, 58, 78, 87, 111, 120, 185, 226,
##   236, 251, 302.
## - Based on the following method and threshold: mahalanobis (30).
## - For variables: pregnant, glucose, pressure, triceps, insulin, mass,
##   pedigree, age.
```

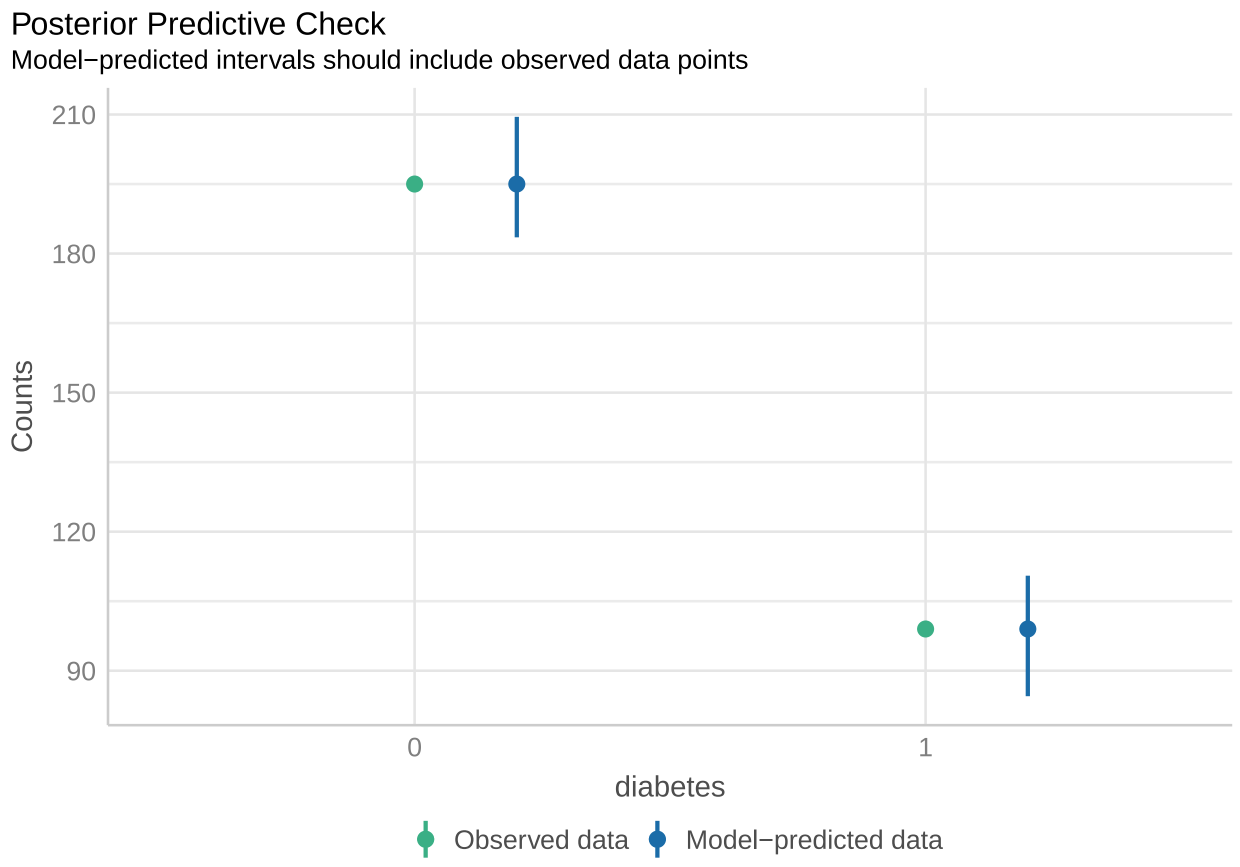
```r
as.vector(result)
```

```
##   [1] FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
##  [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [73] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
##  [85] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [109] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
## [121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [145] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [157] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [169] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [181] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [193] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [205] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [217] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
## [229] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [241] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
## [253] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [265] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [277] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [289] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [301] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [313] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [325] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [337] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [349] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [361] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [373] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [385] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

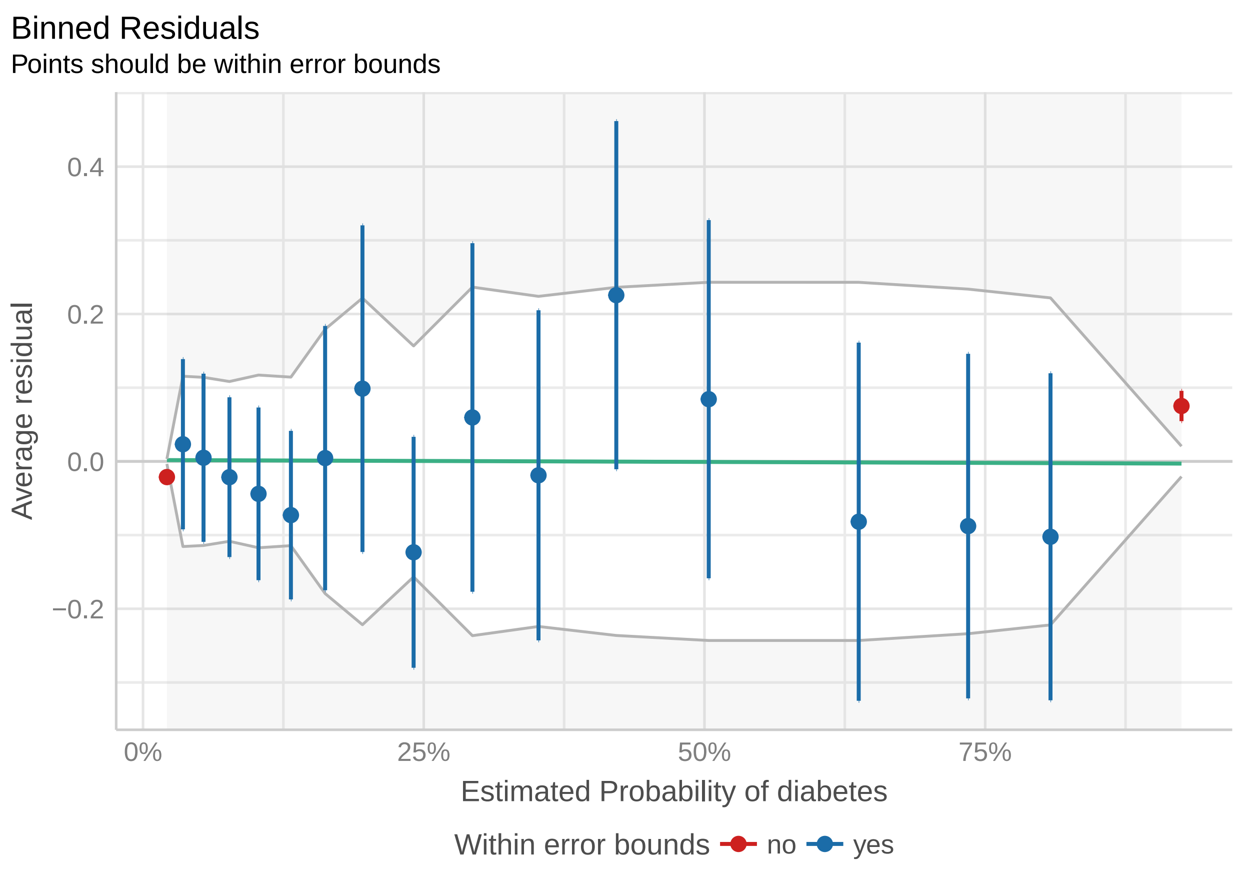
Check model

```r
check <- check_model(log.model, panel = F)
plot(check)
```

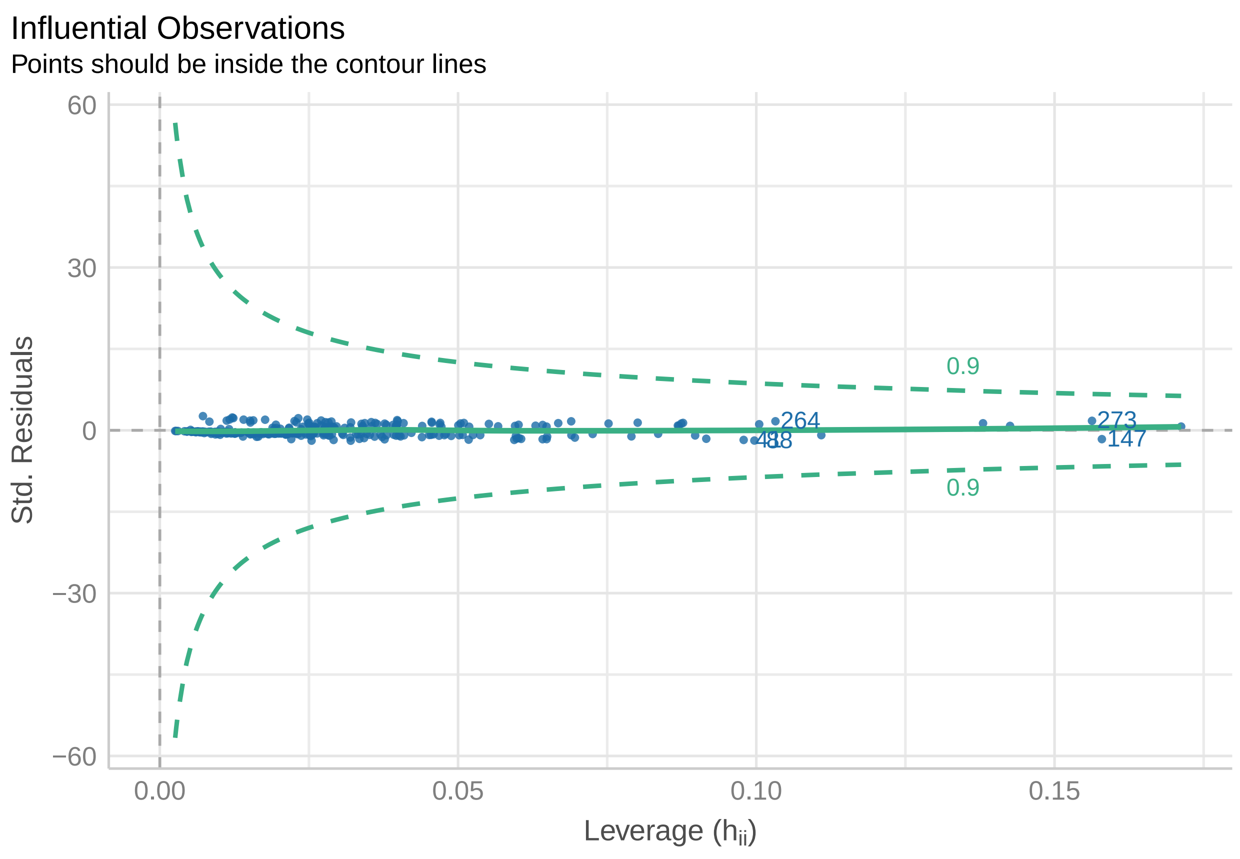
```
## $PP_CHECK
```



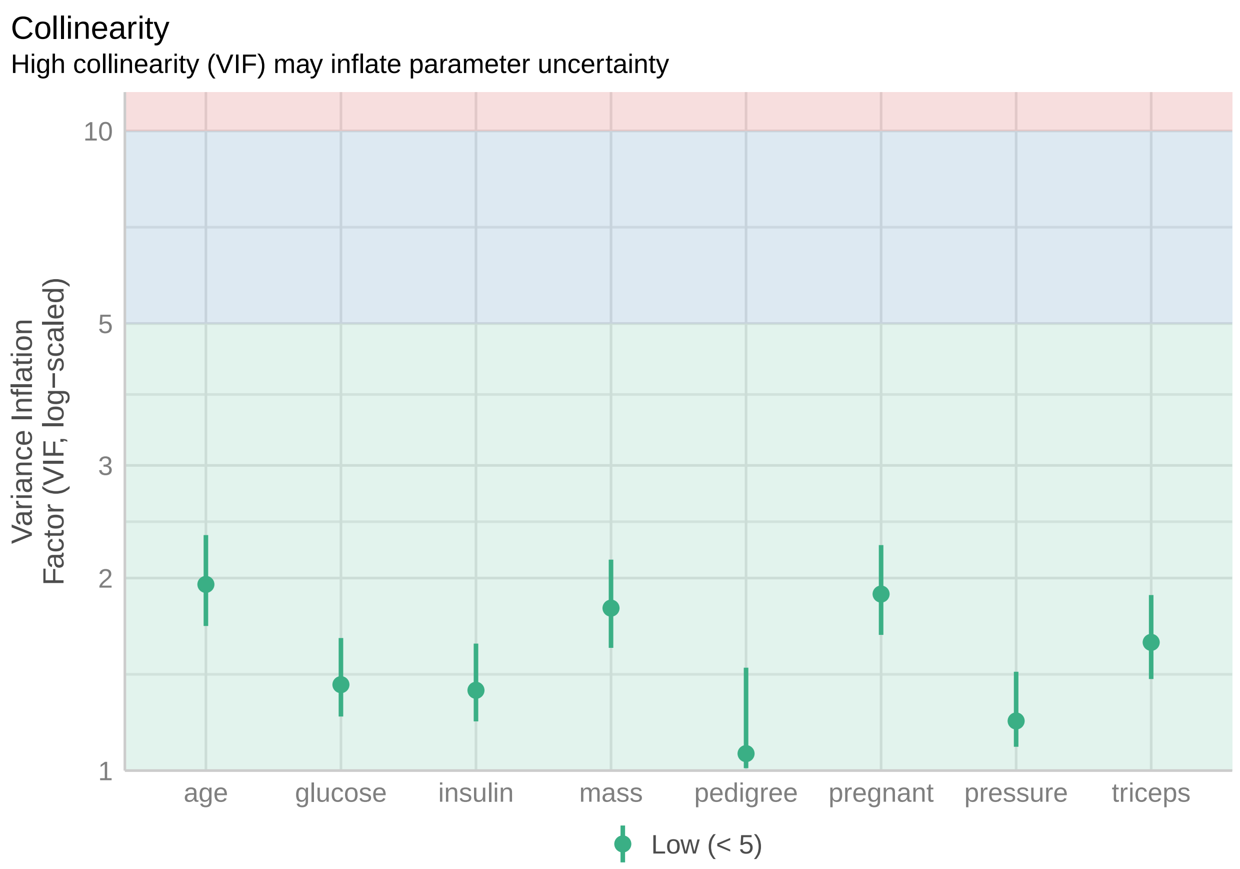
```
##
## $BINNED_RESID
```


Binned Residuals
Points should be within error bounds
Average residual
0.4
0.2
0.0
−0.2
0%
25%
50%
75%
Estimated Probability of diabetes
Within error bounds
no
yes

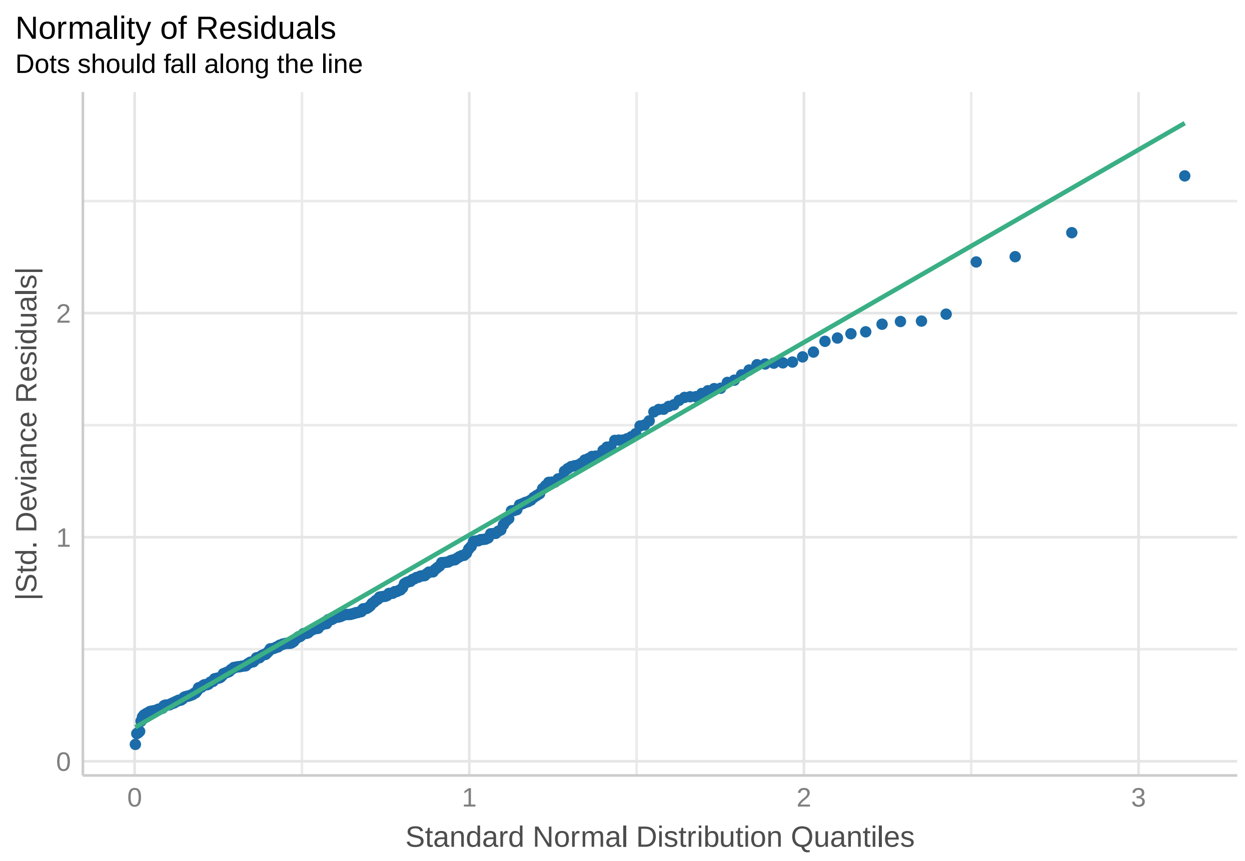
```
##
## $OUTLIERS
```


Influential Observations
Points should be inside the contour lines
60
30
0
−30
−60
Std. Residuals
0.00
0.05
0.10
0.15
Leverage (h_{ii})
0.9
0.9
264
488
273
147

```
##
## $VIF
```


Collinearity
High collinearity (VIF) may inflate parameter uncertainty
Variance Inflation Factor (VIF, log−scaled)
10
5
3
2
1
age
glucose
insulin
mass
pedigree
pregnant
pressure
triceps
Low (< 5)

```
##
## $QQ
```



```r
binned_residuals(log.model)
```

```
## Warning: About 88% of the residuals are inside the error bounds (~95% or higher would be good).
```

Calculate model metrics

```r
# Calculate metrics for each model
metrics.log <- calc_metrics(predicted.classes.log, y.test)
metrics.lasso.min <- calc_metrics(predicted.classes.lasso.min, y.test)
metrics.lasso.1se <- calc_metrics(predicted.classes.lasso.1se, y.test)
metrics.ridge <- calc_metrics(predicted.classes.ridge, y.test)
metrics.elasticnet <- calc_metrics(predicted.classes.elasticnet, y.test)

# Store in a dataframe
df_metrics <- as.data.frame(matrix(c(metrics.log, metrics.lasso.min, metrics.lasso.1se, metrics.ridge, 
colnames(df_metrics) <- c("Logistic", "Lasso_min", "Lasso_1se", "Ridge", "ElasticNet")
rownames(df_metrics) <- c("AUC", "Sensitivity", "Specificity", "F1", "Accuracy")
df_metrics
```

```
##              Logistic Lasso_min Lasso_1se     Ridge ElasticNet
## AUC         0.7939336 0.7939336 0.7691382 0.8013962  0.7927299
## Sensitivity 0.6774194 0.6774194 0.6129032 0.6774194  0.6451613
## Specificity 0.9104478 0.9104478 0.9253731 0.9253731  0.9402985
## F1          0.7241379 0.7241379 0.6909091 0.7368421  0.7272727
## Accuracy    0.8367347 0.8367347 0.8265306 0.8469388  0.8469388
```

Latex Table Metrics

```
metrics_latex <- df_metrics %>%
  kable(format = "latex", booktabs = TRUE, align = c('c'), digits = 3, escape = FALSE,
        col.names = c("Logistic", "Lasso $\\lambda_{\\text{min}}$",
                      "Lasso $\\lambda_{\\text{1se}}$", "Ridge", "ElasticNet")) %>%
  kable_styling(latex_options = c("striped", "scale_down", "hold_position"), position = "center") %>%
  row_spec(0, bold = TRUE) %>%
  column_spec(1, bold = TRUE) %>%
  add_header_above(c(" " = 1, "Models" = 5), bold = TRUE) %>%
  row_spec(nrow(df_metrics) - 1, extra_latex_after = "\\midrule[.08em]") %>%
  row_spec(nrow(df_metrics), bold = TRUE)

print(metrics_latex)
```

	Models				
	Logistic	**Lasso λ_{min}**	**Lasso λ_{1se}**	**Ridge**	**ElasticNet**
AUC	0.794	0.794	0.769	0.801	0.793
Sensitivity	0.677	0.677	0.613	0.677	0.645
Specificity	0.910	0.910	0.925	0.925	0.940
F1	0.724	0.724	0.691	0.737	0.727
Accuracy	**0.837**	**0.837**	**0.827**	**0.847**	**0.847**

Data Frame of all model coefficients

```
coef.log <- coef(log.model)
coef.lasso.min <- (coef(cv.lasso.model, s = cv.lasso.model$lambda.min))
coef.lasso.1se <- coef(cv.lasso.model, s = cv.lasso.model$lambda.1se)
coef.ridge <- coef(cv.ridge.model)
coef.elasticnet <- coef(elasticnet.model$finalModel, s = elasticnet.model$bestTune$lambda)

# Create a data frame to store coefficients
df_coef <- data.frame(
  Logistic = as.vector(coef.log),
  Lasso_min = as.vector(as.matrix(coef.lasso.min)),
  Lasso_1se = as.vector(as.matrix(coef.lasso.1se)),
  Ridge = as.vector(as.matrix(coef.ridge)),
  ElasticNet = as.vector(as.matrix(coef.elasticnet))
)
rownames(df_coef) <- names(coef.log)

options(scipen = 999)
df_coef_sparse <- round(df_coef, 4)
df_coef_sparse[] <- apply(df_coef, 2, function(x) ifelse(x == 0, '.', x))

df_coef_sparse
```

```
##                          Logistic          Lasso_min           Lasso_1se
## (Intercept)   -0.976333973381479 -0.908754819703348 -0.793678618299817
## pregnant       0.348383290980628   0.25002699073021 0.0471370598584918
## glucose         1.19662756034941   1.05713059005187  0.805771016005861
## pressure    -0.00787072837112941                  .                  .
## triceps       0.0990151553153173 0.0626409984004955                  .
## insulin      -0.0108165468422877                  .                  .
## mass           0.527217247967629  0.418404324962646   0.19426613242444
## pedigree       0.488903494680658  0.362231763229765  0.103610887531999
## age             0.24003005800779   0.22352140524853  0.177596021359124
##                           Ridge          ElasticNet
## (Intercept) -0.776920592058855  -0.783644314126479
## pregnant     0.156750171833101   0.124519132142544
## glucose      0.437363939996492   0.616288276600248
## pressure     0.084266432087758 0.00362514721615527
## triceps      0.121285665042735  0.0734587750905991
## insulin      0.157046443194511  0.0673612838310804
## mass         0.181212025582025   0.200698200930468
## pedigree     0.176979262379118   0.158043114063933
## age          0.186020503416058     0.1885990817328
```

Latex Table Coef

```r
coef_latex <- df_coef_sparse %>%
  kable(digits = 3, format = "latex", booktabs = TRUE, align = c('c'), escape = FALSE,
        col.names = c("Logistic", "Lasso $\\lambda_{\\text{min}}$",
                      "Lasso $\\lambda_{\\text{1se}}$", "Ridge", "ElasticNet")) %>%
  kable_styling(latex_options = c("striped", "scale_down", "hold_position"), position = "center") %>%
  row_spec(0, bold = TRUE) %>%
  column_spec(1, bold = TRUE) %>%
  add_header_above(c(" " = 1, "Models" = 5), bold = TRUE)

print(coef_latex)
```

	Models				
	Logistic	**Lasso λ_{min}**	**Lasso λ_{1se}**	**Ridge**	**ElasticNet**
(Intercept)	-0.976333973381479	-0.908754819703348	-0.793678618299817	-0.776920592058855	-0.783644314126479
pregnant	0.348383290980628	0.25002699073021	0.0471370598584918	0.156750171833101	0.124519132142544
glucose	1.19662756034941	1.05713059005187	0.805771016005861	0.437363939996492	0.616288276600248
pressure	-0.00787072837112941	.	.	0.084266432087758	0.00362514721615527
triceps	0.0990151553153173	0.0626409984004955	.	0.121285665042735	0.0734587750905991
insulin	-0.0108165468422877	.	.	0.157046443194511	0.0673612838310804
mass	0.527217247967629	0.418404324962646	0.19426613242444	0.181212025582025	0.200698200930468
pedigree	0.488903494680658	0.362231763229765	0.103610887531999	0.176979262379118	0.158043114063933
age	0.24003005800779	0.22352140524853	0.177596021359124	0.186020503416058	0.1885990817328

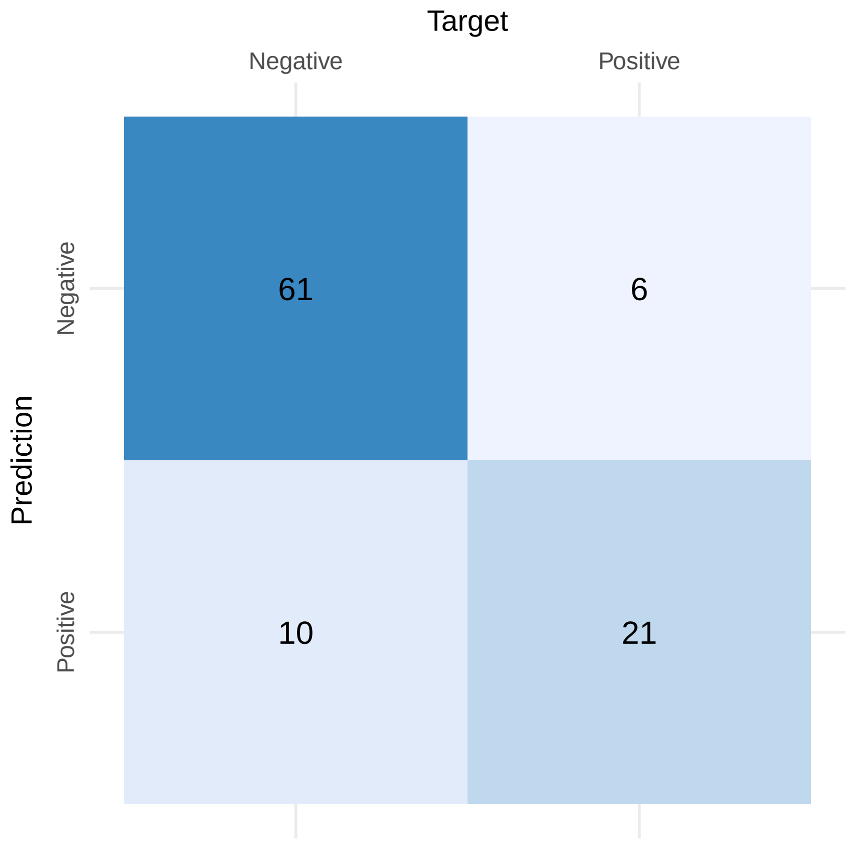
Confusion matrix latex - log.model

```r
# Convert the table to a data frame
log.conf.df <- as.data.frame(log.conf$table)

# Replace the numeric labels with text labels
log.conf.df$Prediction <- factor(log.conf.df$Prediction, levels = c("0", "1"), labels = c("Negative", "
log.conf.df$Reference <- factor(log.conf.df$Reference, levels = c("0", "1"), labels = c("Negative", "Po

# Rename columns as required by cvms
```

```r
names(log.conf.df) <- c('Target','Prediction','N')
# Plot the confusion matrix using cvms
plot_confusion_matrix(log.conf.df, add_normalized = FALSE,
  add_row_percentages = FALSE, add_col_percentages = FALSE, rotate_y_text = TRUE,
  place_x_axis_above = TRUE, class_order = c("Positive", "Negative"))
```

```
## Warning in plot_confusion_matrix(log.conf.df, add_normalized = FALSE,
## add_row_percentages = FALSE, : 'ggimage' is missing. Will not plot arrows and
## zero-shading.
```

```
## Warning in plot_confusion_matrix(log.conf.df, add_normalized = FALSE,
## add_row_percentages = FALSE, : 'rsvg' is missing. Will not plot arrows and
## zero-shading.
```

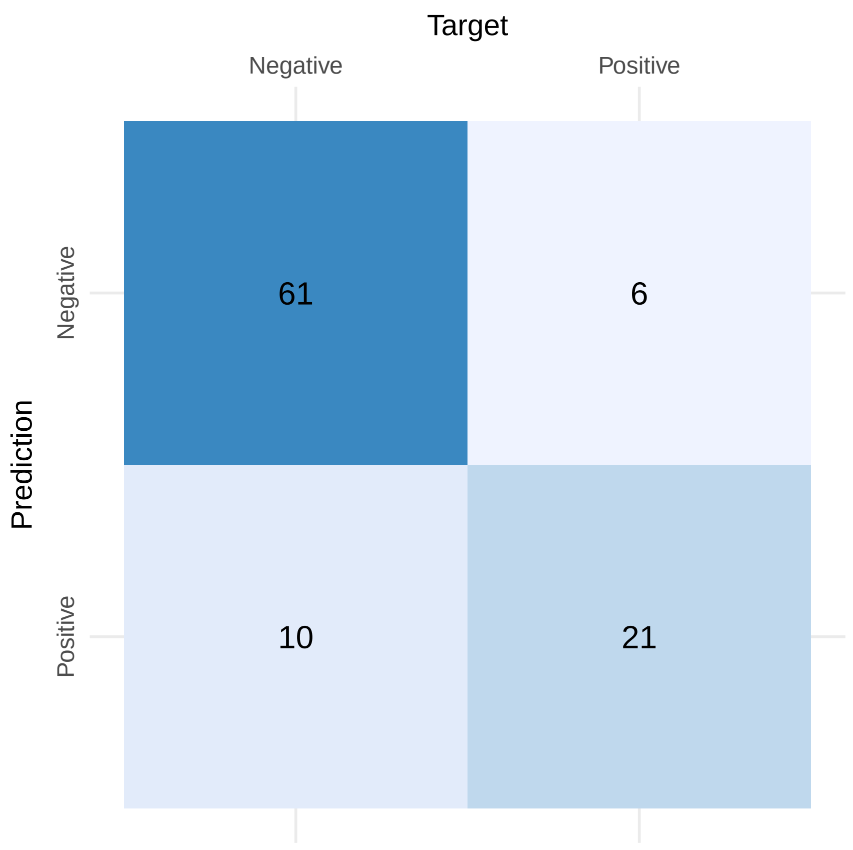


### Confusion matrix latex - lasso.min.model

```r
# For lasso.min.conf
lasso.min.conf.df <- as.data.frame(lasso.min.conf$table)
lasso.min.conf.df$Prediction <- factor(lasso.min.conf.df$Prediction, levels = c("0", "1"), labels = c("
lasso.min.conf.df$Reference <- factor(lasso.min.conf.df$Reference, levels = c("0", "1"), labels = c("Ne
names(lasso.min.conf.df) <- c('Target','Prediction','N')
plot_confusion_matrix(lasso.min.conf.df, add_normalized = FALSE, add_row_percentages = FALSE, add_col_p
```

```
## Warning in plot_confusion_matrix(lasso.min.conf.df, add_normalized = FALSE, :
## 'ggimage' is missing. Will not plot arrows and zero-shading.
```

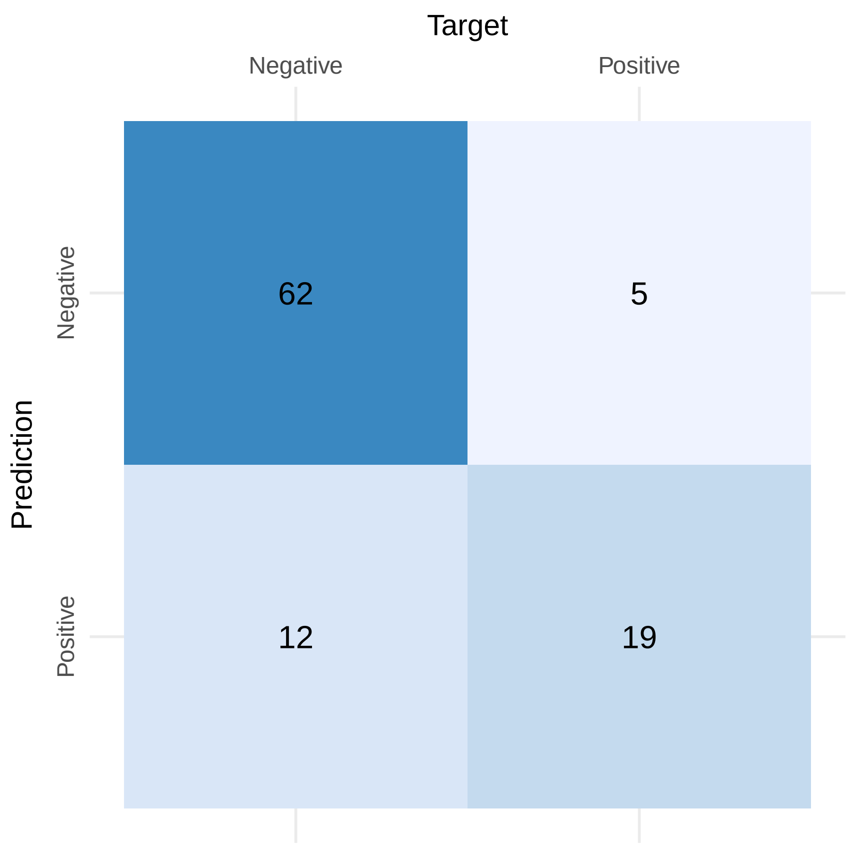
```
## Warning in plot_confusion_matrix(lasso.min.conf.df, add_normalized = FALSE, :
## 'rsvg' is missing. Will not plot arrows and zero-shading.
```



Confusion matrix latex - lasso.1se.model

```
lasso.1se.conf.df <- as.data.frame(lasso.1se.conf$table)
lasso.1se.conf.df$Prediction <- factor(lasso.1se.conf.df$Prediction, levels = c("0", "1"), labels = c("N
lasso.1se.conf.df$Reference <- factor(lasso.1se.conf.df$Reference, levels = c("0", "1"), labels = c("Neg
names(lasso.1se.conf.df) <- c('Target','Prediction','N')
plot_confusion_matrix(lasso.1se.conf.df, add_normalized = FALSE, add_row_percentages = FALSE, add_col_pe
```

```
## Warning in plot_confusion_matrix(lasso.1se.conf.df, add_normalized = FALSE, :
## 'ggimage' is missing. Will not plot arrows and zero-shading.

## Warning in plot_confusion_matrix(lasso.1se.conf.df, add_normalized = FALSE, :
## 'rsvg' is missing. Will not plot arrows and zero-shading.
```

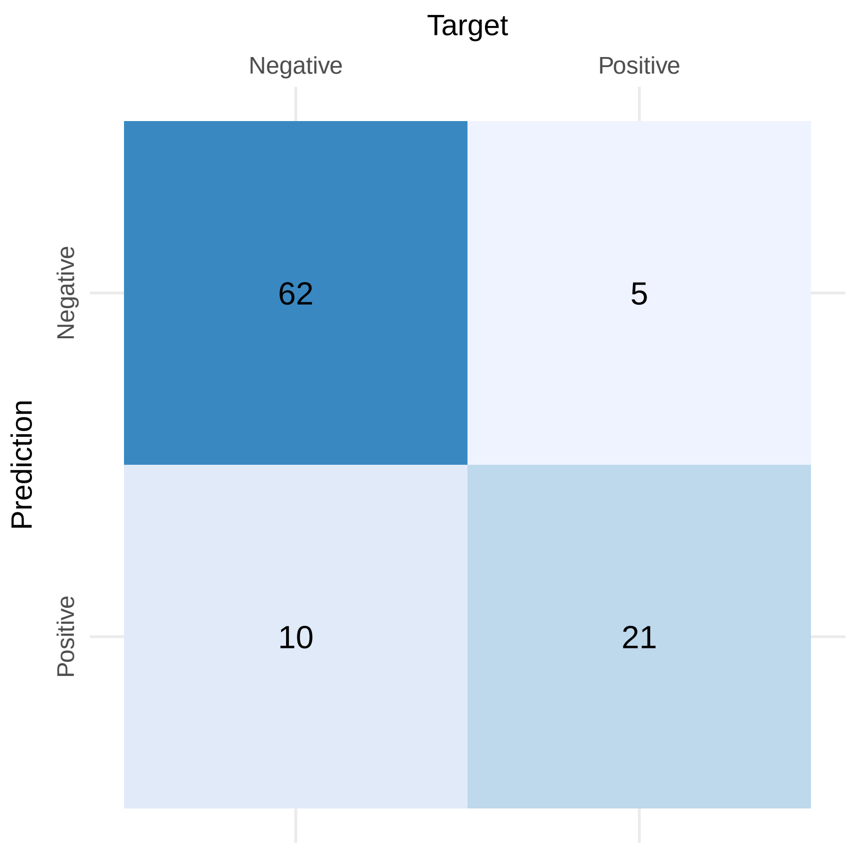


Confusion matrix latex - ridge.model

```r
# For ridge.conf
ridge.conf.df <- as.data.frame(ridge.conf$table)
ridge.conf.df$Prediction <- factor(ridge.conf.df$Prediction, levels = c("0", "1"), labels = c("Negative
ridge.conf.df$Reference <- factor(ridge.conf.df$Reference, levels = c("0", "1"), labels = c("Negative",
names(ridge.conf.df) <- c('Target','Prediction','N')
plot_confusion_matrix(ridge.conf.df, add_normalized = FALSE, add_row_percentages = FALSE, add_col_perce
```

```
## Warning in plot_confusion_matrix(ridge.conf.df, add_normalized = FALSE, :
## 'ggimage' is missing. Will not plot arrows and zero-shading.
```

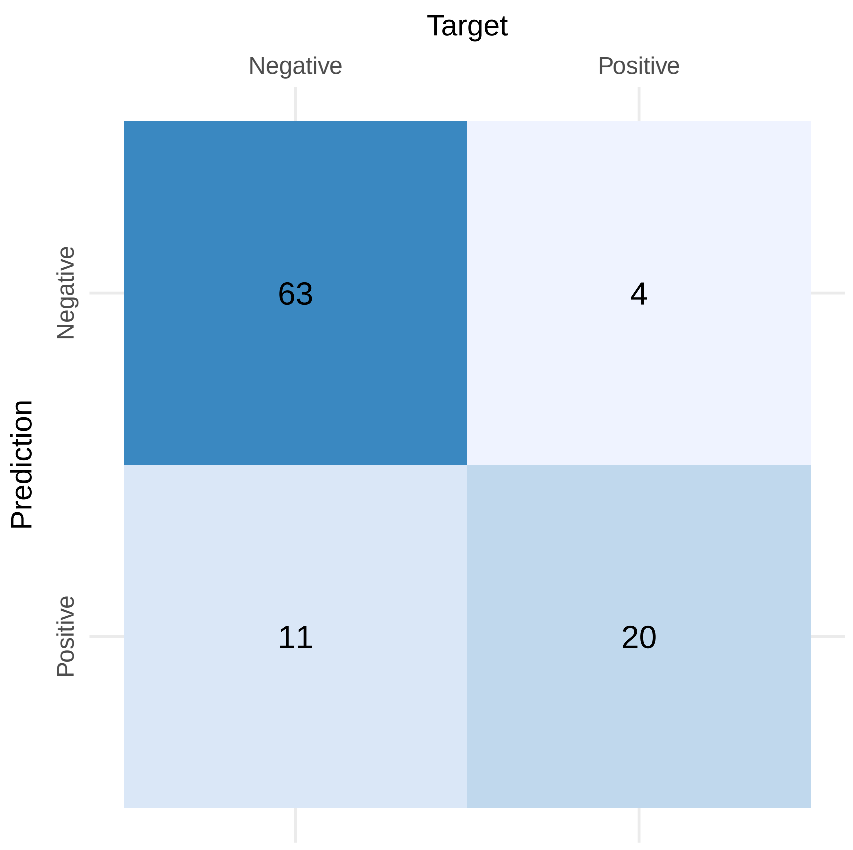
```
## Warning in plot_confusion_matrix(ridge.conf.df, add_normalized = FALSE, :
## 'rsvg' is missing. Will not plot arrows and zero-shading.
```



Confusion matrix latex - elasticnet.model

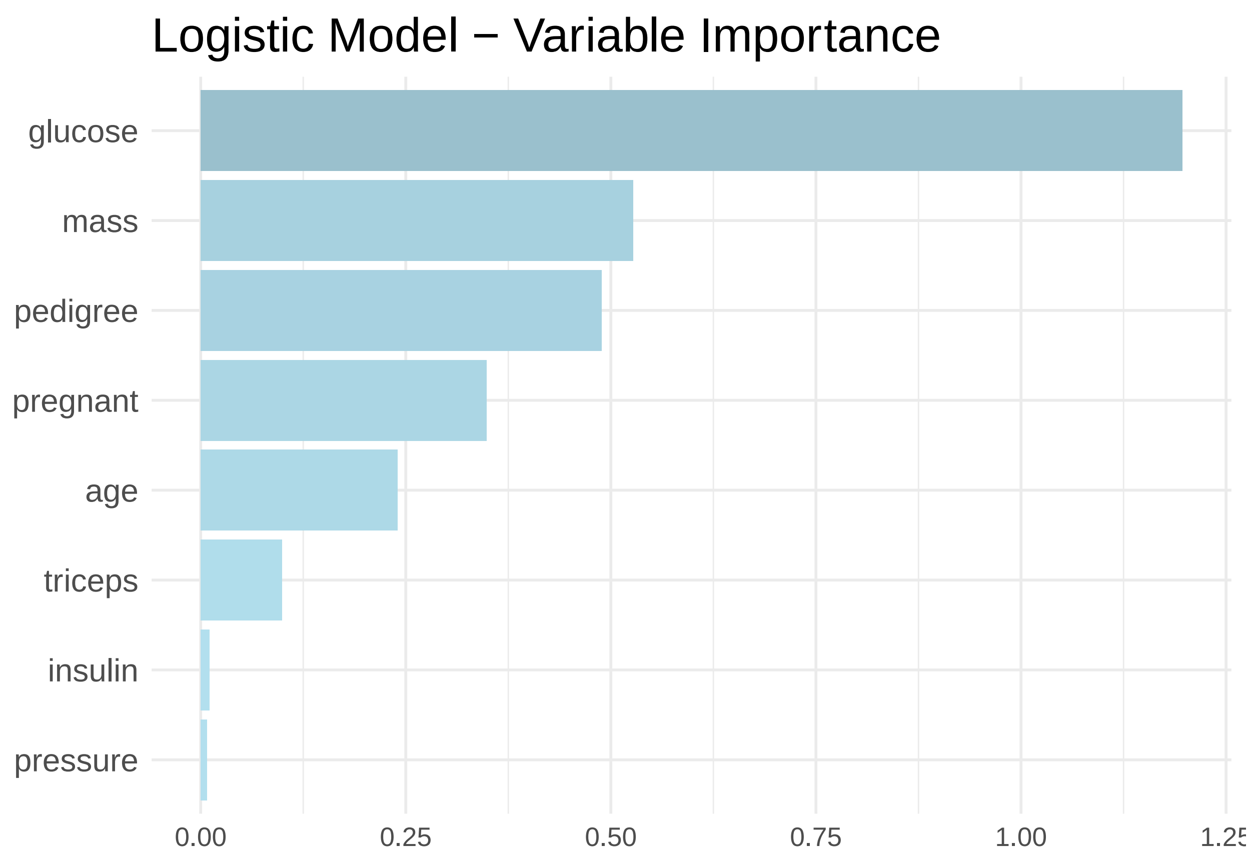
```
# For elasticnet.conf
elasticnet.conf.df <- as.data.frame(elasticnet.conf$table)
elasticnet.conf.df$Prediction <- factor(elasticnet.conf.df$Prediction, levels = c("0", "1"), labels = c
elasticnet.conf.df$Reference <- factor(elasticnet.conf.df$Reference, levels = c("0", "1"), labels = c("
names(elasticnet.conf.df) <- c('Target','Prediction','N')
plot_confusion_matrix(elasticnet.conf.df, add_normalized = FALSE, add_row_percentages = FALSE, add_col_
```

```
## Warning in plot_confusion_matrix(elasticnet.conf.df, add_normalized = FALSE, :
## 'ggimage' is missing. Will not plot arrows and zero-shading.

## Warning in plot_confusion_matrix(elasticnet.conf.df, add_normalized = FALSE, :
## 'rsvg' is missing. Will not plot arrows and zero-shading.
```

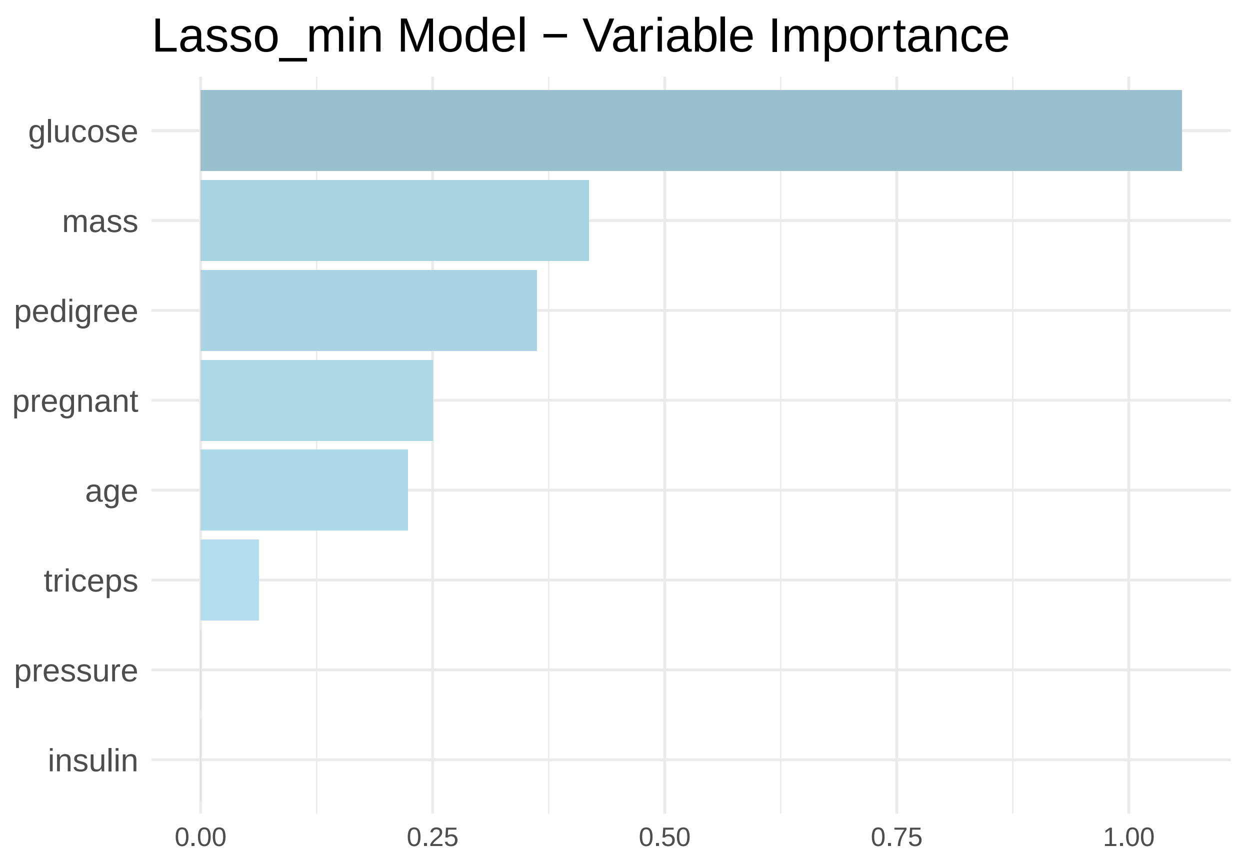


### Variable Importance

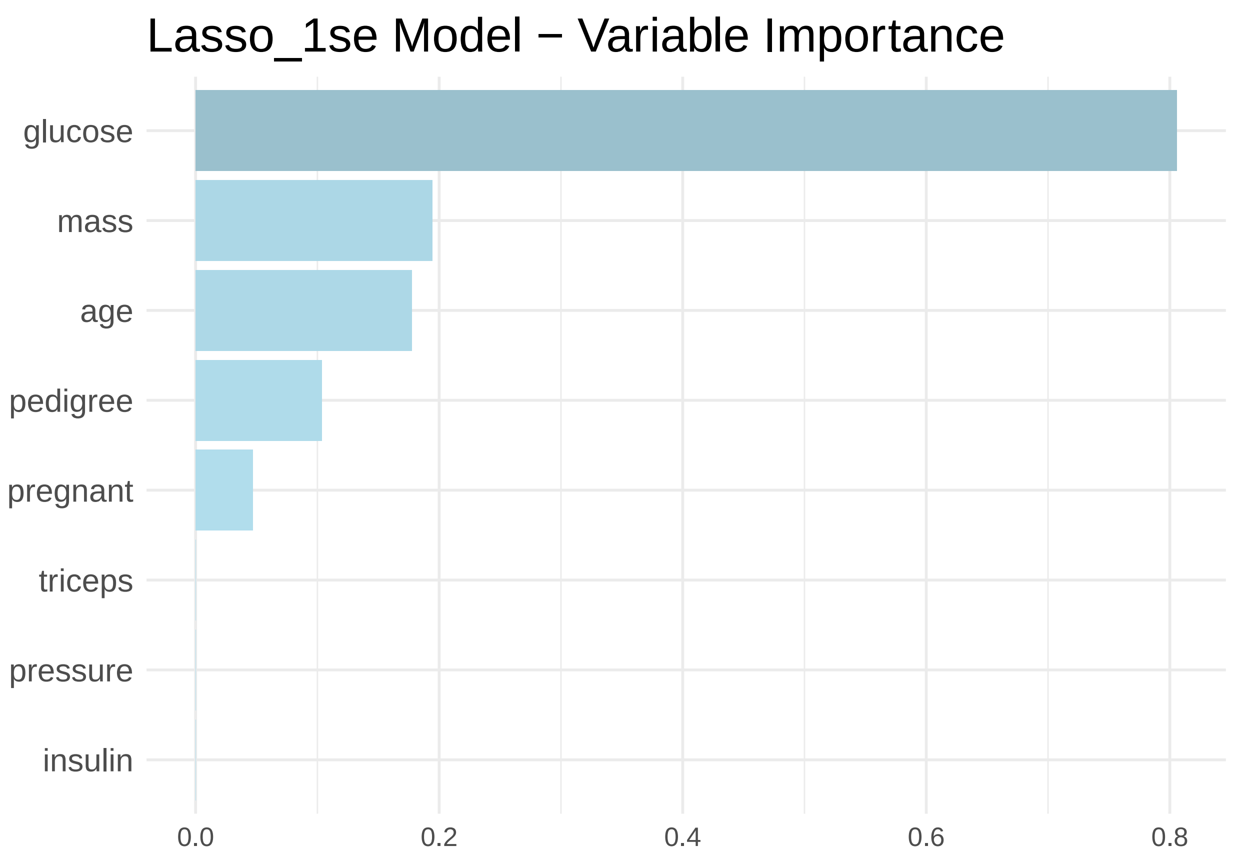
```
importance_plot("Logistic")
```



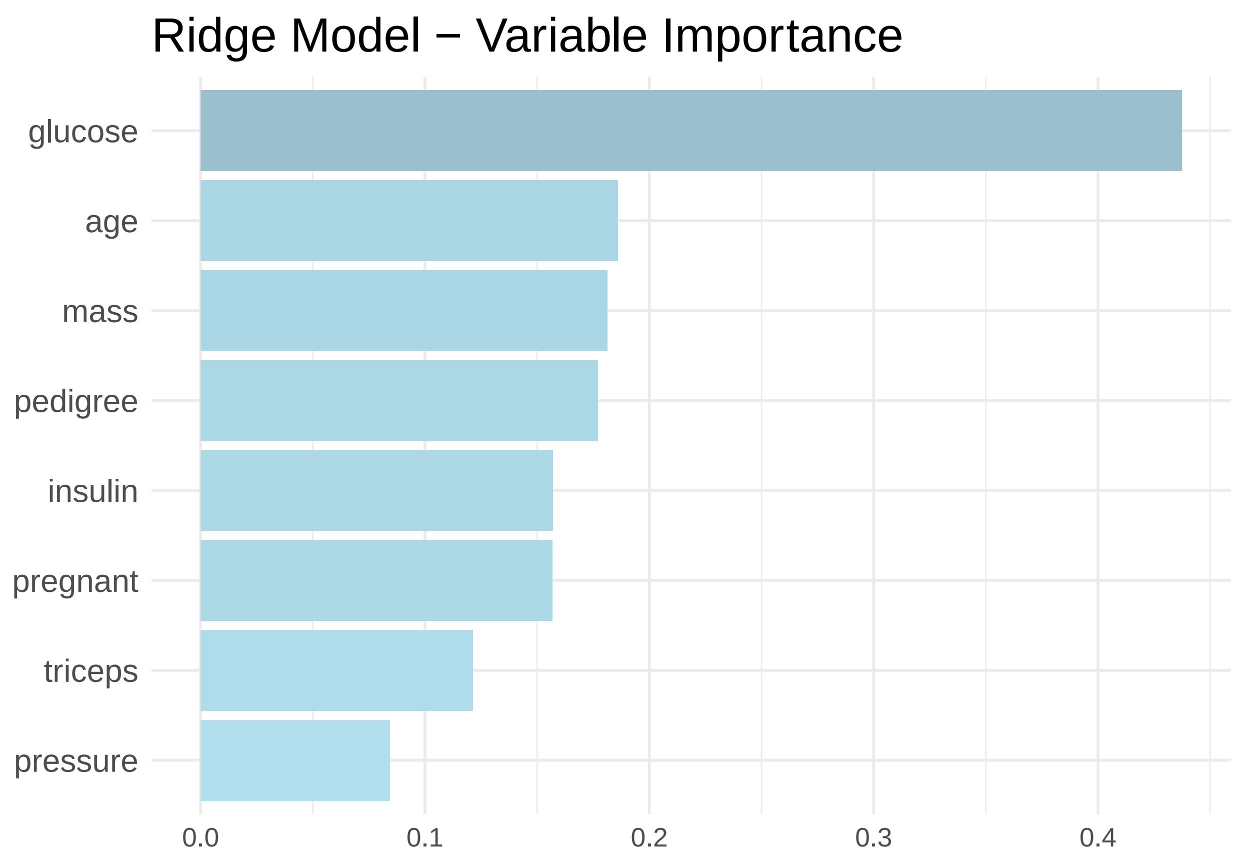
```r
importance_plot("Lasso_min")
```



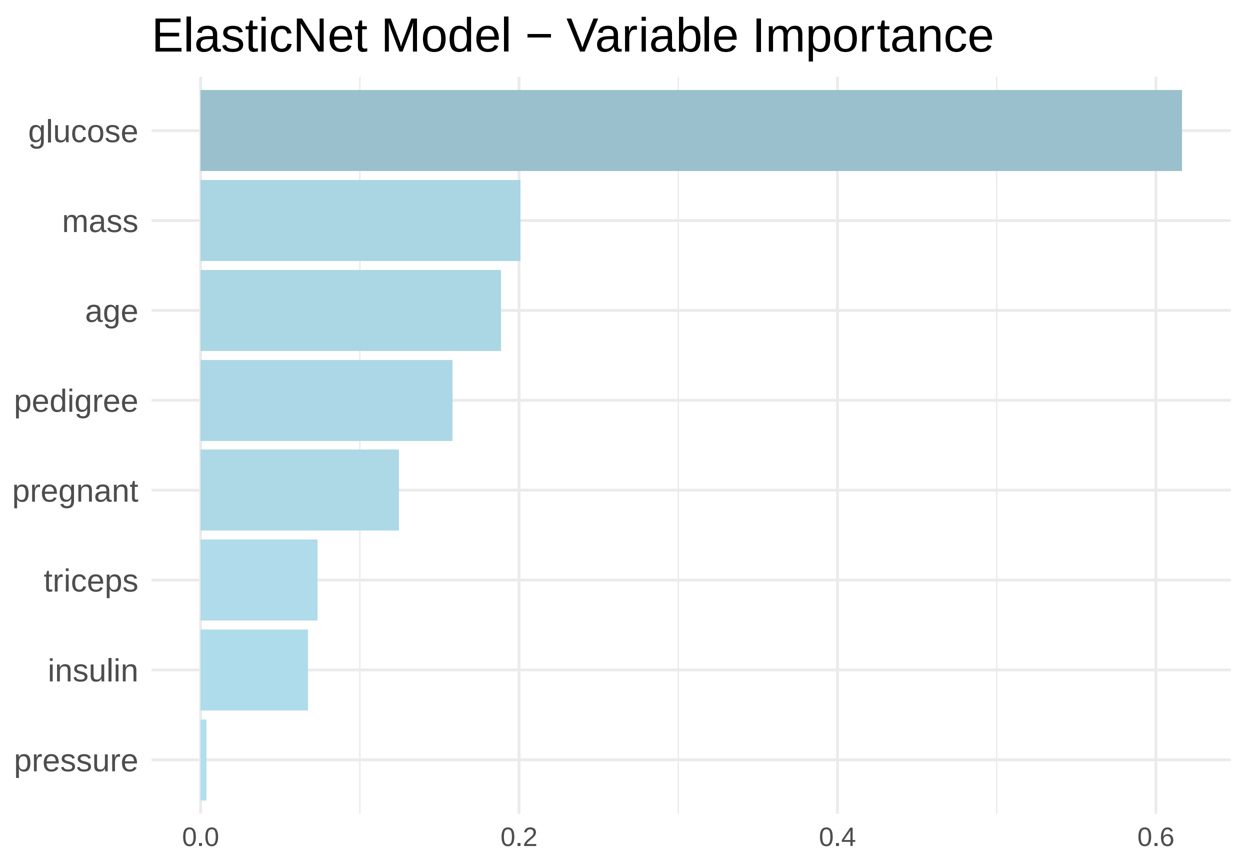
```r
importance_plot("Lasso_1se")
```



```r
importance_plot("Ridge")
```

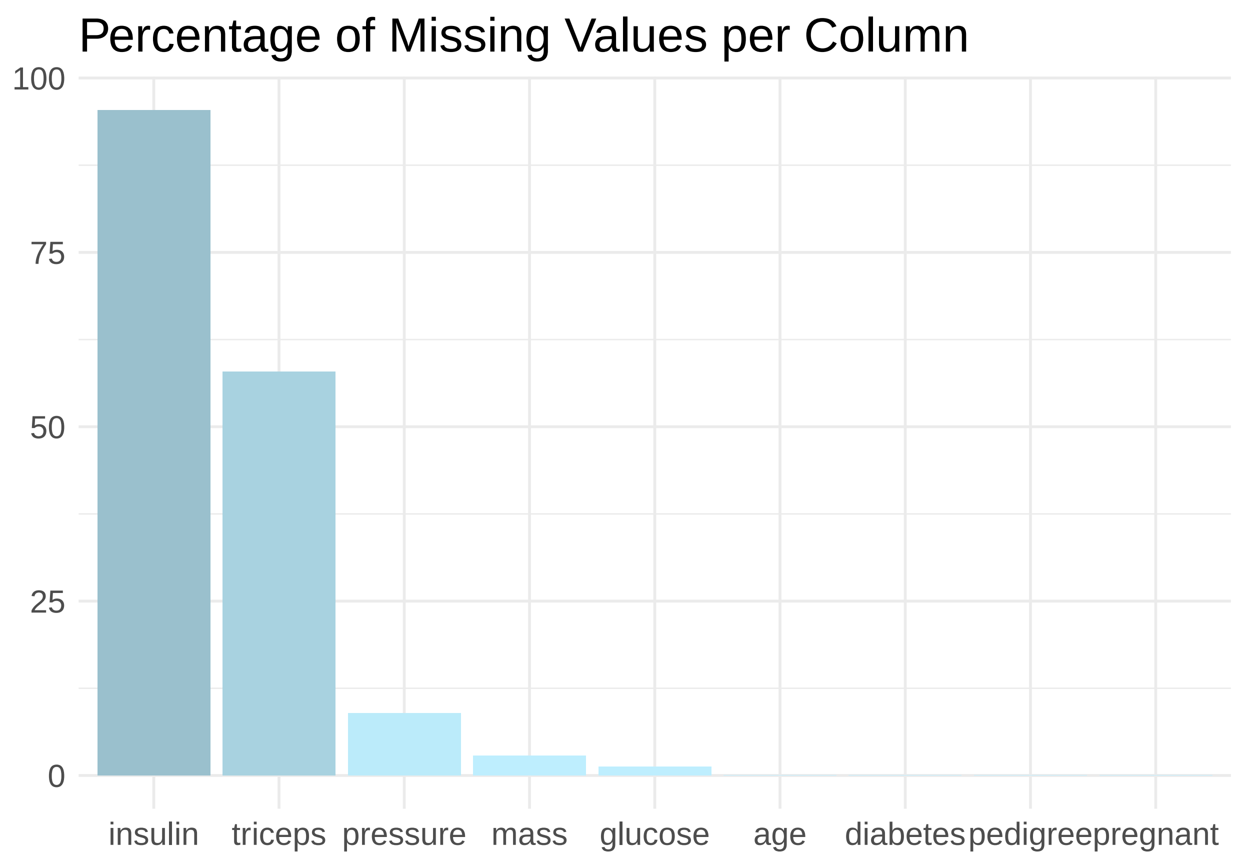


```
importance_plot("ElasticNet")
```



Perc missing

```r
library(ggplot2)

# Count of missing values per column
count_na <- c(pregnant = 0, glucose = 5, pressure = 35, triceps = 227, insulin = 374, mass = 11, pedigr

# Total number of rows in the data
n_rows <- nrow(diabetes)

# Convert to data frame and calculate percentages
df_na <- data.frame(Column = names(count_na), Count = as.numeric(count_na))
df_na$Percentage <- (df_na$Count / n_rows) * 100

# Plot
ggplot(df_na, aes(x = reorder(Column, -Percentage), y = Percentage, fill = Percentage)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "lightblue1", high = "lightblue3") +
  theme_minimal() +
  theme(
    axis.text.y = element_text(size = 12),
    axis.text.x = element_text(size = 12),
    title = element_text(size = 15),
    axis.title.y = element_blank(),
    axis.title.x = element_blank(),
    legend.position = "none"
  ) +
  ggtitle("Percentage of Missing Values per Column")
```



log model summary table latex

```r
# Extract coefficients from the model summary
coef_summary <- summary(log.model)$coefficients

# Convert the matrix to a data frame for kable()
coef_summary <- round(as.data.frame(coef_summary)[, c(1, 4)], 3)
colnames(coef_summary) <- c("Estimate", "p-value")

coef_summary$`p-value` <- cell_spec(coef_summary$`p-value`, "latex",
                                    bold = ifelse(coef_summary$`p-value` < 0.05, TRUE, FALSE))
# Create the LaTeX table
kable(coef_summary, "latex", booktabs = TRUE, align = "c", escape = F) %>%
  kable_styling(latex_options = c("striped", "scale_down", "hold_position")) %>%
  row_spec(0, bold = TRUE)
```

	Estimate	**p-value**
(Intercept)	-0.976	**0**
pregnant	0.348	0.087
glucose	1.197	**0**
pressure	-0.008	0.963
triceps	0.099	0.622
insulin	-0.011	0.95
mass	0.527	**0.016**
pedigree	0.489	**0.004**
age	0.240	0.251