

# Diabetes Dataset Testing

2023-09-24

## Functions

```
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))  
}
```

## 5. Data Processing

```
# 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)))
```

pregnant	glucose	pressure	triceps	insulin	mass	pedigree	age	diabetes
0	5	35	227	374	11	0	0	0

## Removing NA's

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

pregnant	glucose	pressure	triceps	insulin	mass	pedigree	age	diabetes
0	0	0	0	0	0	0	0	0

## Removing Outliers

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

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

## Data for models

```
# Split
set.seed(123)
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
#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

```
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)
confusionMatrix(predicted.classes.log, y.test, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction  0   1
##           0 63 10
##           1   6 19
##
##              Accuracy : 0.8367
##              95% CI : (0.7484, 0.9037)
##      No Information Rate : 0.7041
##      P-Value [Acc > NIR] : 0.001856
##
##              Kappa : 0.5919
##
##  Mcnemar's Test P-Value : 0.453255
##
##              Sensitivity : 0.6552
##              Specificity : 0.9130
##              Pos Pred Value : 0.7600
##              Neg Pred Value : 0.8630
##              Prevalence : 0.2959
##              Detection Rate : 0.1939
##      Detection Prevalence : 0.2551
##              Balanced Accuracy : 0.7841
##
##              '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$lambda.min), coef(cv.lasso.model, s = cv.lasso.model$lambda.1se))

## 9 x 2 sparse Matrix of class "dgCMatrix"
##              s1              s1
## (Intercept) -0.8646581 -0.7383796
## pregnant    .           .
## glucose      1.0258065  0.7428165
```

```
## pressure      .      .
## triceps       .      .
## insulin       .      .
## mass          0.4231952 0.1340846
## pedigree      0.2695704 .
## age           0.4144144 0.1527177

# 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)
confusionMatrix(predicted.classes.lasso.min, y.test, positive = "1")

## Confusion Matrix and Statistics
##
##              Reference
## Prediction  0  1
##           0 64  9
##           1  5 20
##
##              Accuracy : 0.8571
##              95% CI : (0.7719, 0.9196)
##      No Information Rate : 0.7041
##      P-Value [Acc > NIR] : 0.0003264
##
##              Kappa : 0.6429
##
##  Mcnemar's Test P-Value : 0.4226781
##
##              Sensitivity : 0.6897
##              Specificity : 0.9275
##      Pos Pred Value : 0.8000
##      Neg Pred Value : 0.8767
##              Prevalence : 0.2959
##      Detection Rate : 0.2041
##      Detection Prevalence : 0.2551
##      Balanced Accuracy : 0.8086
##
##      '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)
confusionMatrix(as.factor(predicted.classes.ridge), y.test, positive = "1")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##           0 64 10
##           1  5 19
##
##               Accuracy : 0.8469
##               95% CI : (0.7601, 0.9117)
##       No Information Rate : 0.7041
##       P-Value [Acc > NIR] : 0.0008073
##
##               Kappa : 0.6134
##
##  Mcnemar's Test P-Value : 0.3016996
##
##       Sensitivity : 0.6552
##       Specificity : 0.9275
##       Pos Pred Value : 0.7917
##       Neg Pred Value : 0.8649
##       Prevalence : 0.2959
##       Detection Rate : 0.1939
##       Detection Prevalence : 0.2449
##       Balanced Accuracy : 0.7914
##
##       'Positive' Class : 1
##
```

## Elastic Net

```
set.seed(123)

# CV and tuning grid

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)
confusionMatrix(as.factor(predicted.classes.elasticnet), y.test, positive = "1")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##           0 65 15
##           1  4 14
##
##           Accuracy : 0.8061
##           95% CI : (0.7139, 0.879)
##       No Information Rate : 0.7041
##       P-Value [Acc > NIR] : 0.01512
##
##           Kappa : 0.4773
##
##  Mcnemar's Test P-Value : 0.02178
##
##           Sensitivity : 0.4828
##           Specificity : 0.9420
##       Pos Pred Value : 0.7778
##       Neg Pred Value : 0.8125
##           Prevalence : 0.2959
##       Detection Rate : 0.1429
##   Detection Prevalence : 0.1837
##       Balanced Accuracy : 0.7124
##
##       'Positive' Class : 1
##

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

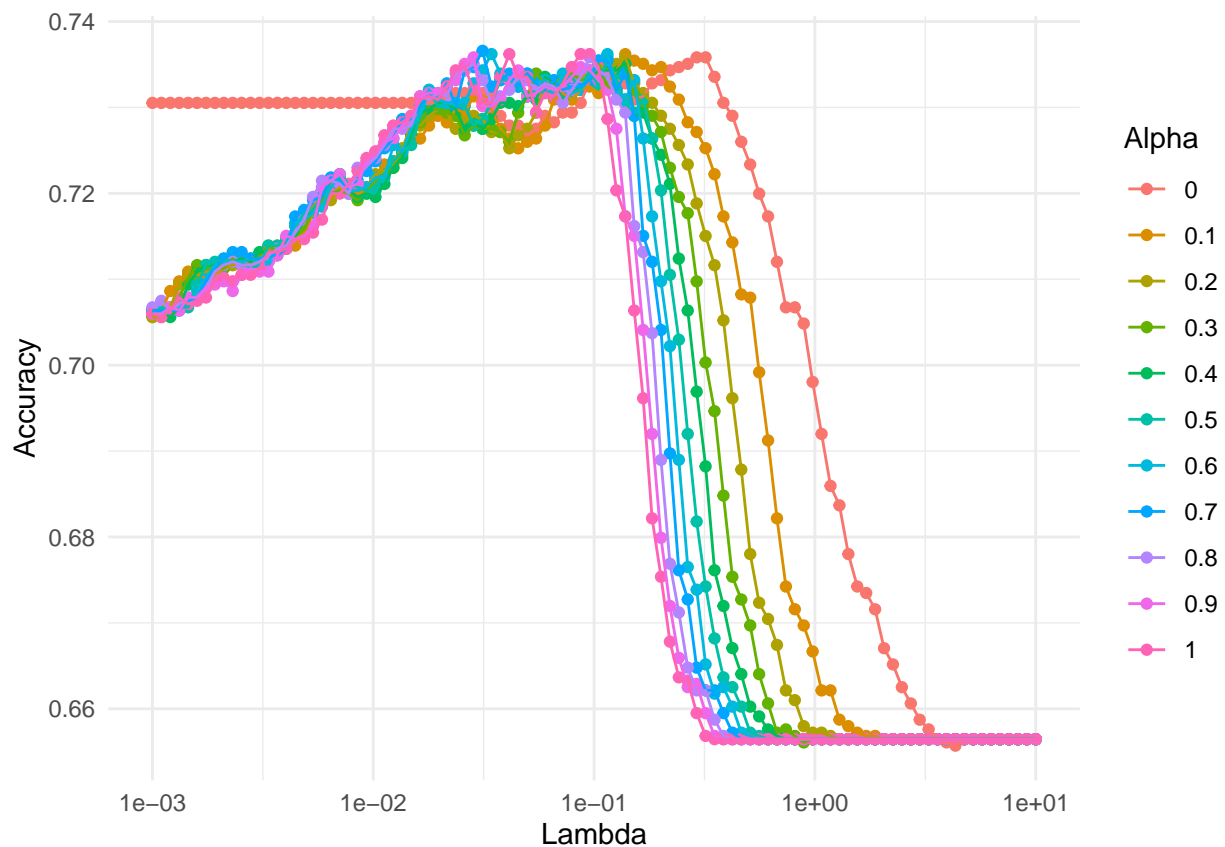
## 9 x 1 sparse Matrix of class "dgCMatrix"
##              s1
## (Intercept) -0.71479062
## pregnant    .
## glucose     0.66381573
## pressure    .

```

```
## triceps      .
## insulin     .
## mass        0.06264426
## pedigree    .
## age         0.09478631

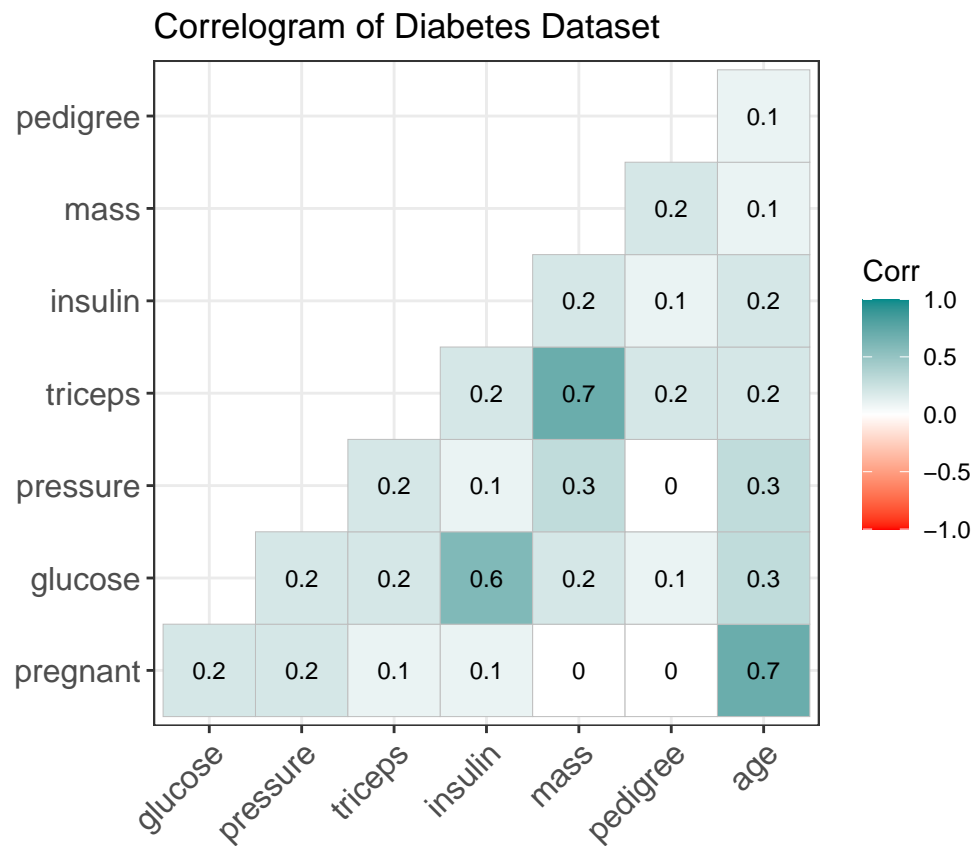
# Filter data to only include specific alpha values
filtered_data <- subset(elasticnet.model$results, alpha %in% seq(0, 1, by=0.1))

# 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())
```



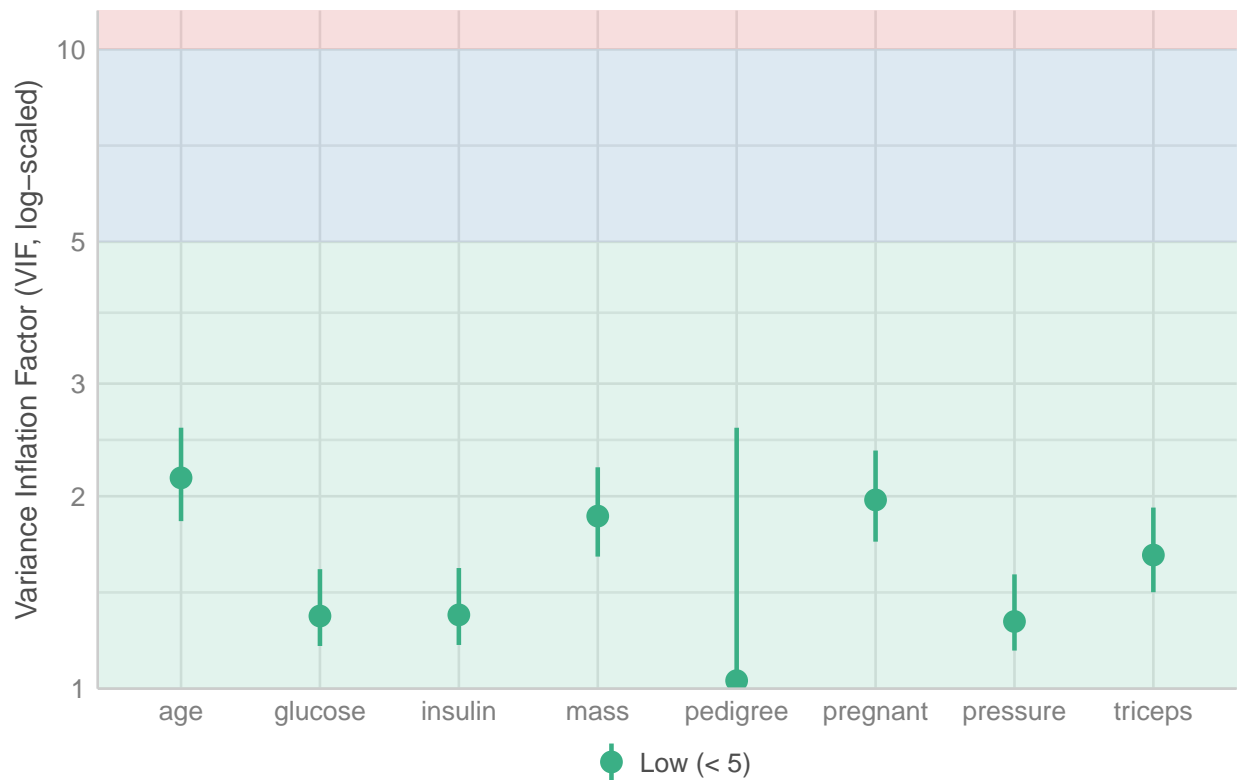
```
result <- check_collinearity(log.model)
plot(result)
```

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



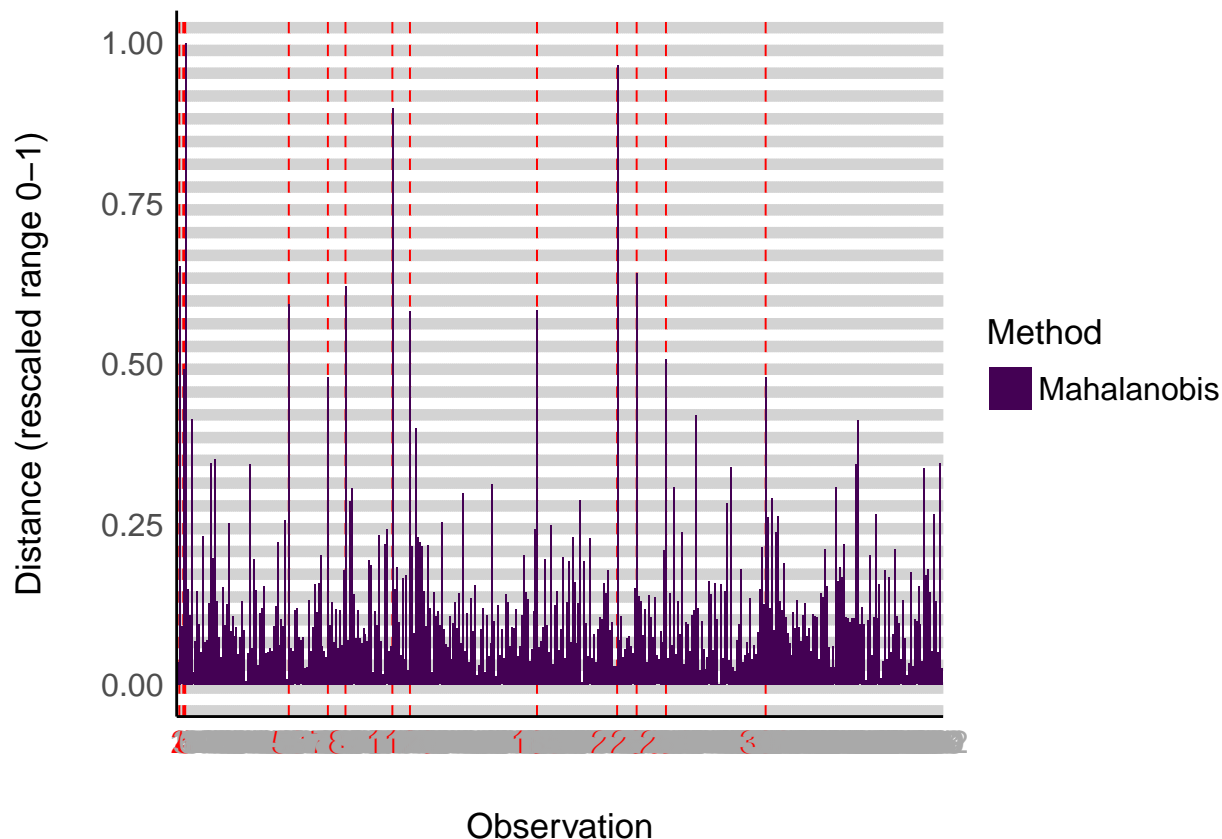
## Collinearity

High collinearity (VIF) may inflate parameter uncertainty



### Outliers

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



```
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.
```

```
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 FALSE TRUE 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
```

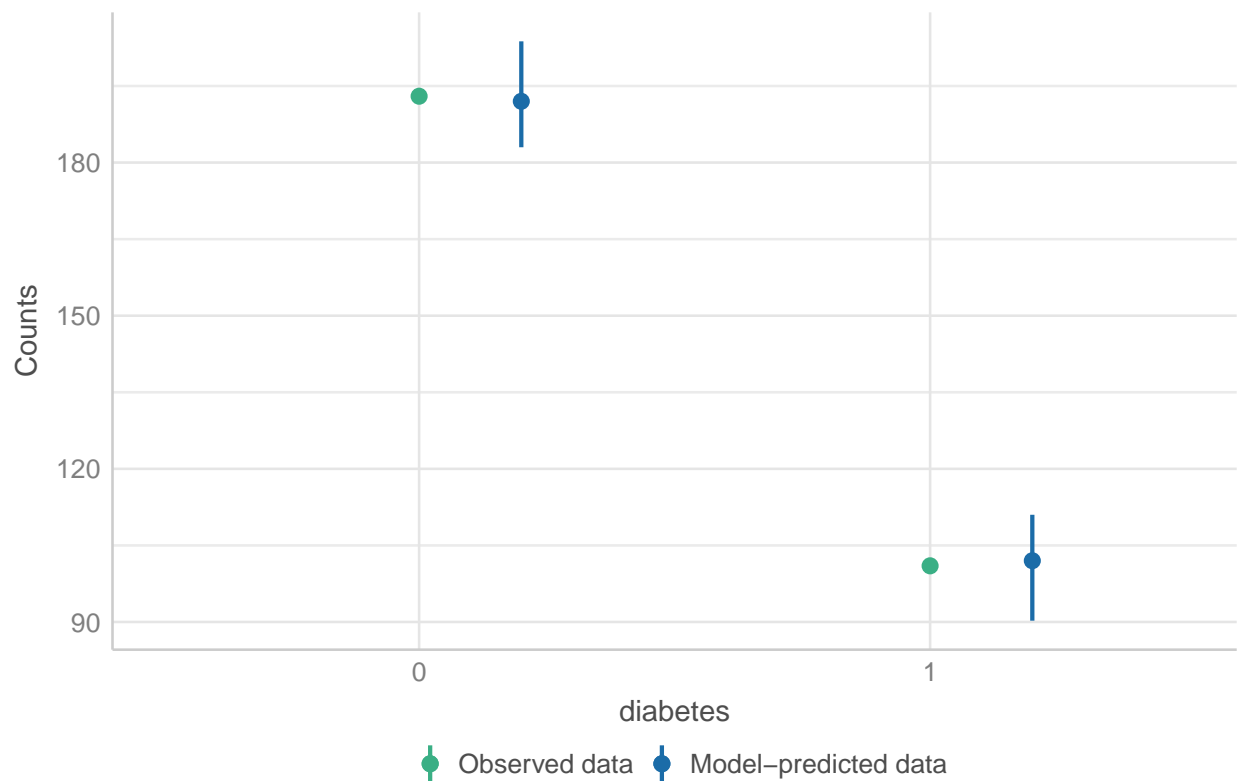
## Check model

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

```
## $PP_CHECK
```

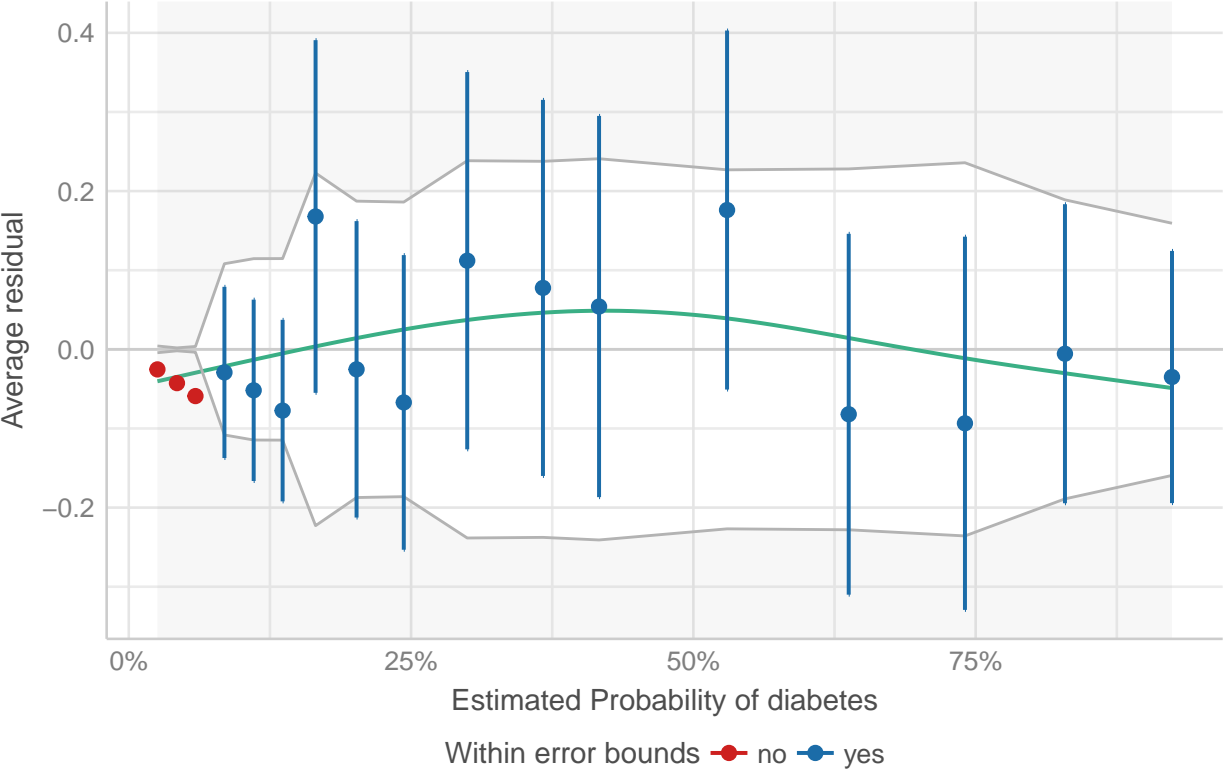
## Posterior Predictive Check

Model-predicted intervals should include observed data points



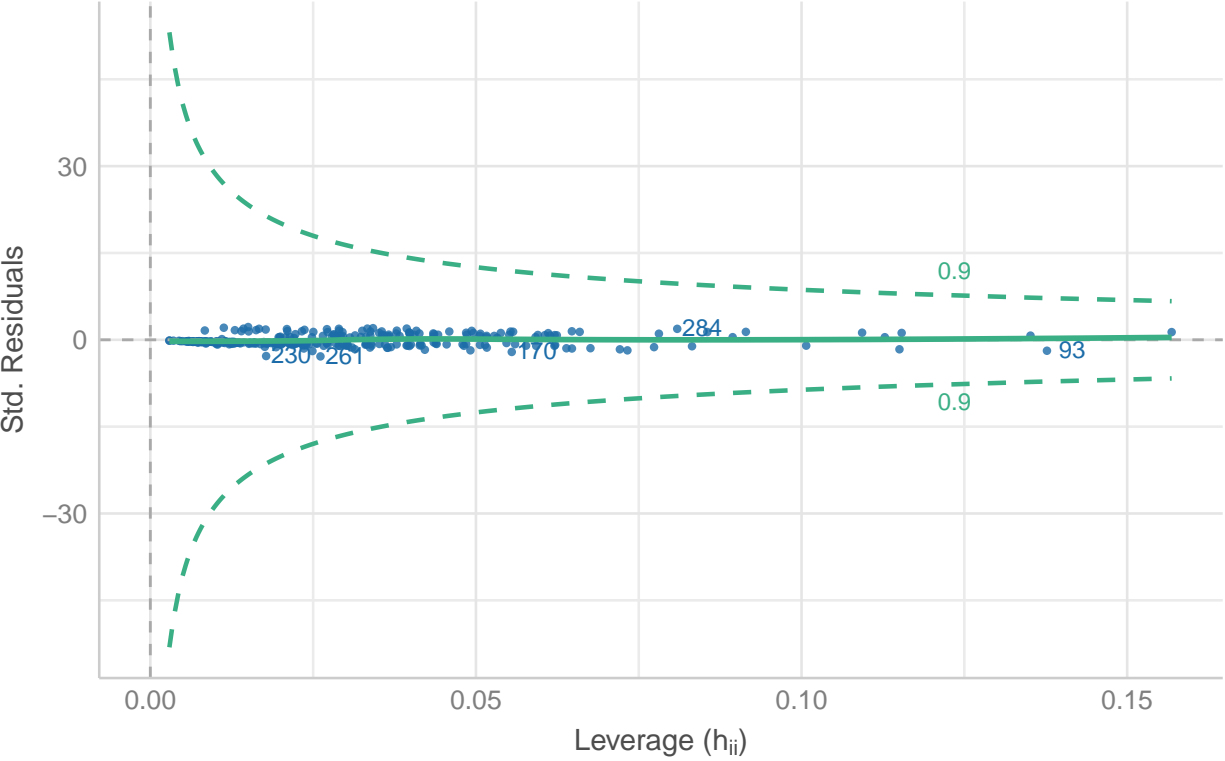
```
##
## $BINNED_RESID
```

Binned Residuals  
Points should be within error bounds



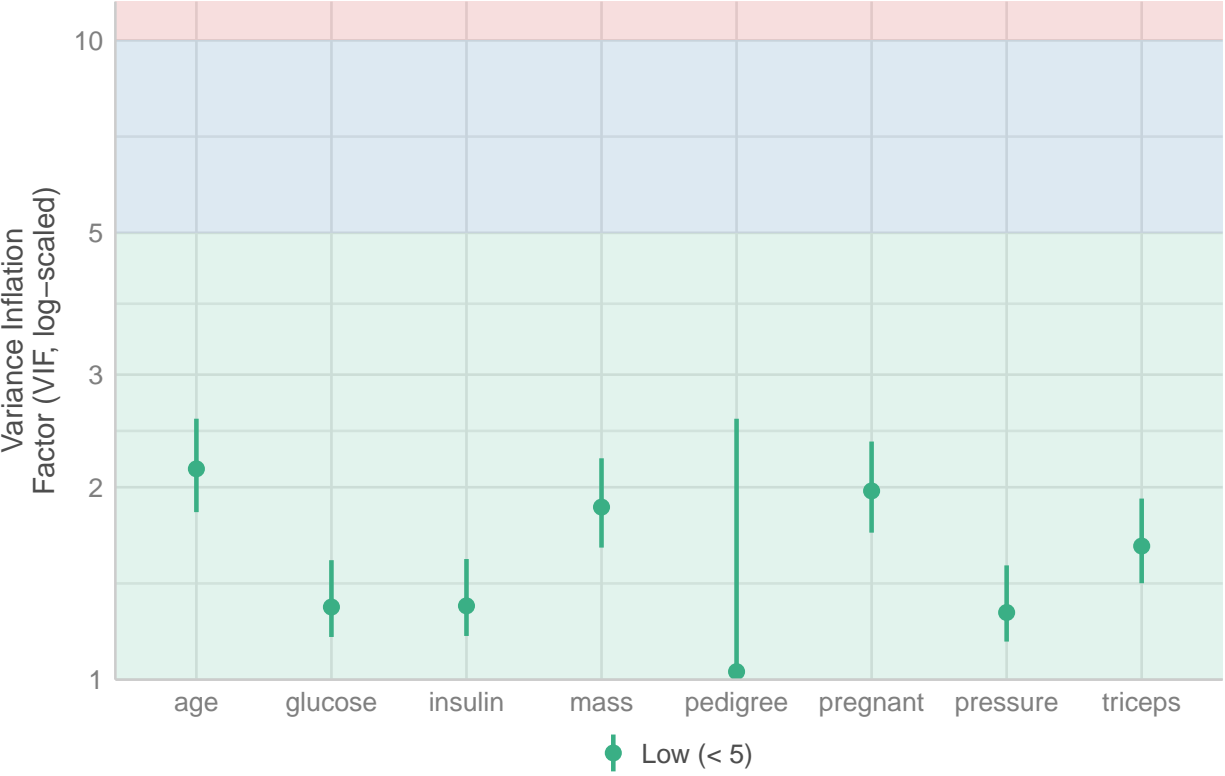
##  
## \$OUTLIERS

Influential Observations  
Points should be inside the contour lines



##  
## \$VIF

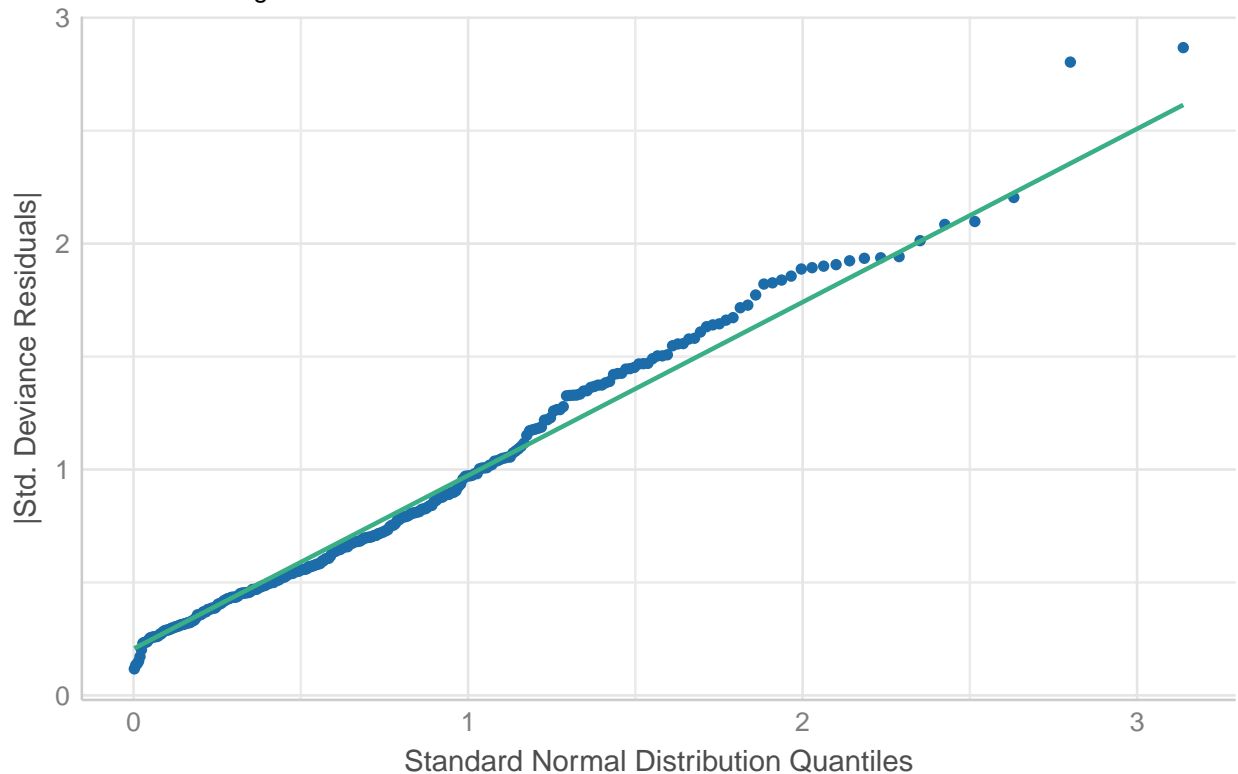
Collinearity  
High collinearity (VIF) may inflate parameter uncertainty



##  
## \$QQ

## Normality of Residuals

Dots should fall along the line



```
binmed_residuals(log.model)
```

```
## Warning: About 82% of the residuals are inside the error bounds (~95% or higher would be good).
```

## Calculate model metrics

```
# 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, metrics.elasticnet),
  nrow = 5, ncol = 5))
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          0.7841079 0.8085957 0.7568716 0.7913543 0.7123938
## Sensitivity  0.6551724 0.6896552 0.5862069 0.6551724 0.4827586
## Specificity  0.9130435 0.9275362 0.9275362 0.9275362 0.9420290
## F1           0.7037037 0.7407407 0.6666667 0.7169811 0.5957447
## Accuracy     0.8367347 0.8571429 0.8265306 0.8469388 0.8061224
```

## 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 $\lambda_{\min}$	Lasso $\lambda_{1se}$	Ridge	ElasticNet
<b>AUC</b>	0.784	0.809	0.757	0.791	0.712
<b>Sensitivity</b>	0.655	0.690	0.586	0.655	0.483
<b>Specificity</b>	0.913	0.928	0.928	0.928	0.942
<b>F1</b>	0.704	0.741	0.667	0.717	0.596
<b>Accuracy</b>	<b>0.837</b>	<b>0.857</b>	<b>0.827</b>	<b>0.847</b>	<b>0.806</b>

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

# Set the row names to be the names of the coefficients from one of the models
rownames(df_coef) <- names(coef.log)

df_coef
```



```
##           Logistic Lasso_min Lasso_1se   Ridge ElasticNet
## (Intercept) -0.9178   -0.8647   -0.7384 -0.7224   -0.7148
## pregnant    0.0327      .         .    0.093      .
## glucose     1.144     1.0258    0.7428  0.3495     0.6638
## pressure    -0.0037      .         .    0.0789      .
## triceps     0.0229      .         .    0.1042      .
## insulin     -0.0202      .         .    0.1477      .
## mass        0.5148     0.4232    0.1341  0.1587     0.0626
## pedigree    0.3822     0.2696      .    0.1316      .
## age         0.4724     0.4144    0.1527  0.187      0.0948
```

#### Latex Table Coef

```
coef_latex <- df_coef %>%
  kable(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 $\lambda_{\min}$	Lasso $\lambda_{1se}$	Ridge	ElasticNet
(Intercept)	-0.9178	-0.8647	-0.7384	-0.7224	-0.7148
pregnant	0.0327	.	.	0.093	.
glucose	1.144	1.0258	0.7428	0.3495	0.6638
pressure	-0.0037	.	.	0.0789	.
triceps	0.0229	.	.	0.1042	.
insulin	-0.0202	.	.	0.1477	.
mass	0.5148	0.4232	0.1341	0.1587	0.0626
pedigree	0.3822	0.2696	.	0.1316	.
age	0.4724	0.4144	0.1527	0.187	0.0948