Class Activity 25

Your name here

May 19 2024

Group Activity 1

In this activity, we will calculate the probability of diabetes for a glucose level of 150 mg/dL using the logistic regression coefficients $\beta_0 = -5.61$ and $\beta_1 = 0.0392$.

a. Calculate Log Odds

First, calculate the log odds for a glucose level of 150 mg/dL.

```
log_odds <- -5.61 + (0.0392 * 150)
log_odds
[1] 0.27
```

b. Convert Log Odds to Odds

```
odds <- exp(log_odds)
odds
[1] 1.309964
```

c. Convert Odds to Probability

Finally, convert the odds to probability.

```
probability <- odds / (1 + odds)
probability
[1] 0.5670929
```

The probability of having diabetes at a glucose level of 150 mg/dL is calculated to be 0.5670929.

Group Activity 2

a. Let's fit the logistic regression model.

```
set.seed(12345)
db_single <- db %>% select(diabetes, glucose)
db_split <- initial_split(db_single, prop = 0.80)
```

```
# Create training data
db_train <- db_split %>% training()

# Create testing data
db_test <- db_split %>% testing()

fitted_logistic_model <- logistic_reg() %>% # Call the model function
        # Set the engine/family of the model
        set_engine("glm") %>%
        # Set the mode
        set_mode("classification") %>%
        # Fit the model
        fit(diabetes~., data = db_train)

tidy(fitted_logistic_model)
# A tibble: 2 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)  -5.61     0.678       -8.28 1.20e-16
2 glucose       0.0392   0.00514      7.62 2.55e-14
```

b. We are interested in predicting the diabetes status of patients depending on the amount of glucose. Verify that the glucose value of 143.11 gives the probability of having diabetes as 1/2.

Answer:

$$log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x$$

```
(p <- round(exp(-5.61 + 0.0392* 143.11) / (1 + exp(-5.61 + 0.0392* 143.11)),2))
[1] 0.5
```

c. What value of glucose is needed to have a probability of diabetes of 0.5?

Answer:

```
p <- 0.5
(x <- (log(p/(1-p)) - (-5.61))/0.0392)
[1] 143.1122
```

d. Make a classifier that classifies the diabetes status of new patients with a threshold of 0.5, i.e, a new patient is classified as negative if the estimated class probability is less than 0.75. Also, create a confusion matrix of the resulting predictions. Evaluate the model based on accuracy, sensitivity, specificity, and ppv.

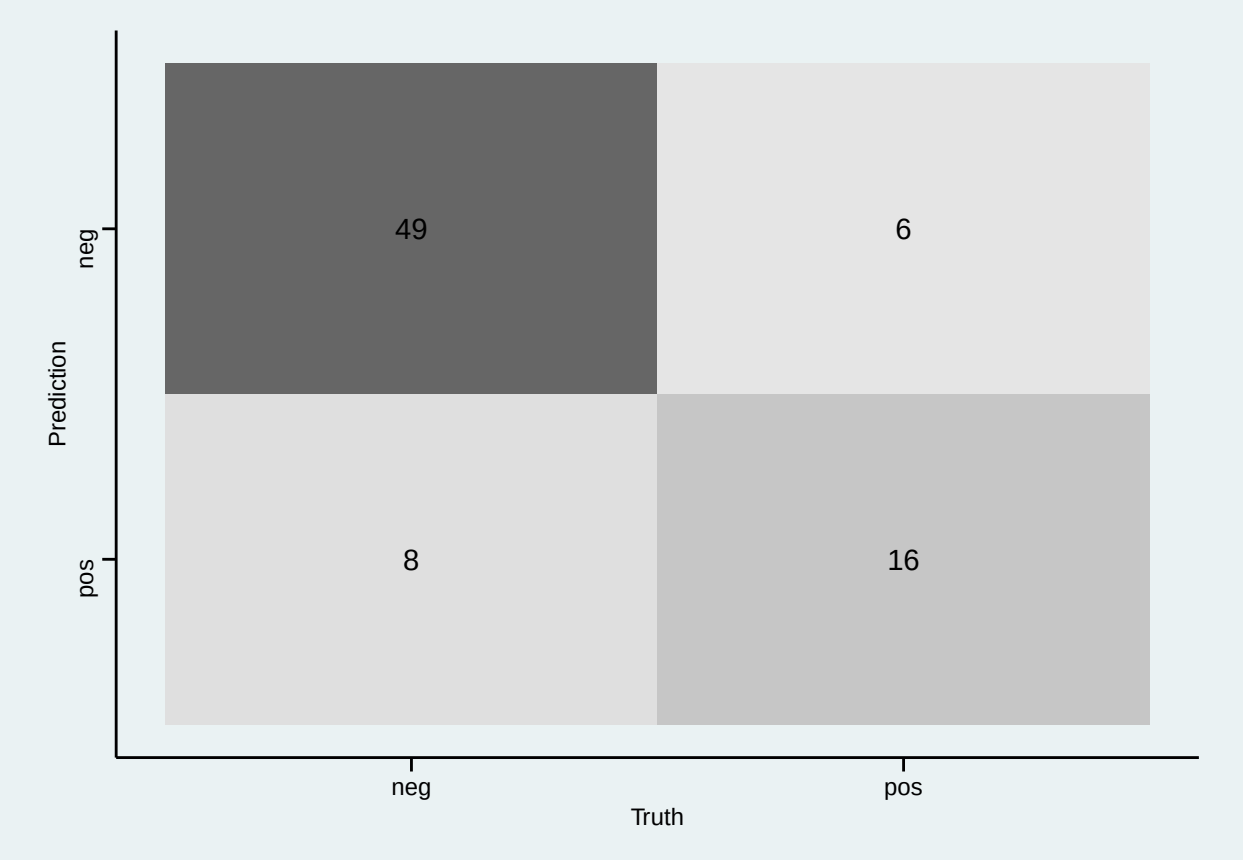
```
# Prediction Probabilities
library(probably)

pred_prob <- predict(fitted_logistic_model,  new_data = db_test,   type = "prob")

db_results <- db_test %>% bind_cols(pred_prob) %>%
  mutate(.pred_class = make_two_class_pred(.pred_neg, levels(diabetes), threshold = .55)) %>%
  select(diabetes, glucose, contains(".pred"))


db_results %>%
```

```
  conf_mat(diabetes,.pred_class) %>%
  autoplot(type = "heatmap")
```




```
# Evaluating the model
eval_metrics <- metric_set(accuracy, sensitivity, specificity, ppv)

eval_metrics(data = db_results,
             truth = diabetes,
             estimate = .pred_class) %>% select(-2)
# A tibble: 4 x 2
  .metric     .estimate
  <chr>           <dbl>
1 accuracy        0.823
2 sensitivity     0.860
3 specificity     0.727
4 ppv             0.891
```

e. Evaluate the performance of a diabetes prediction model at different classification thresholds and visualize how various metrics such as accuracy, sensitivity, and PPV change across these thresholds. Use a sequence of threshold values, apply each one to classify test data, calculate the performance metrics for each classification, and then create a line plot to illustrate the results.

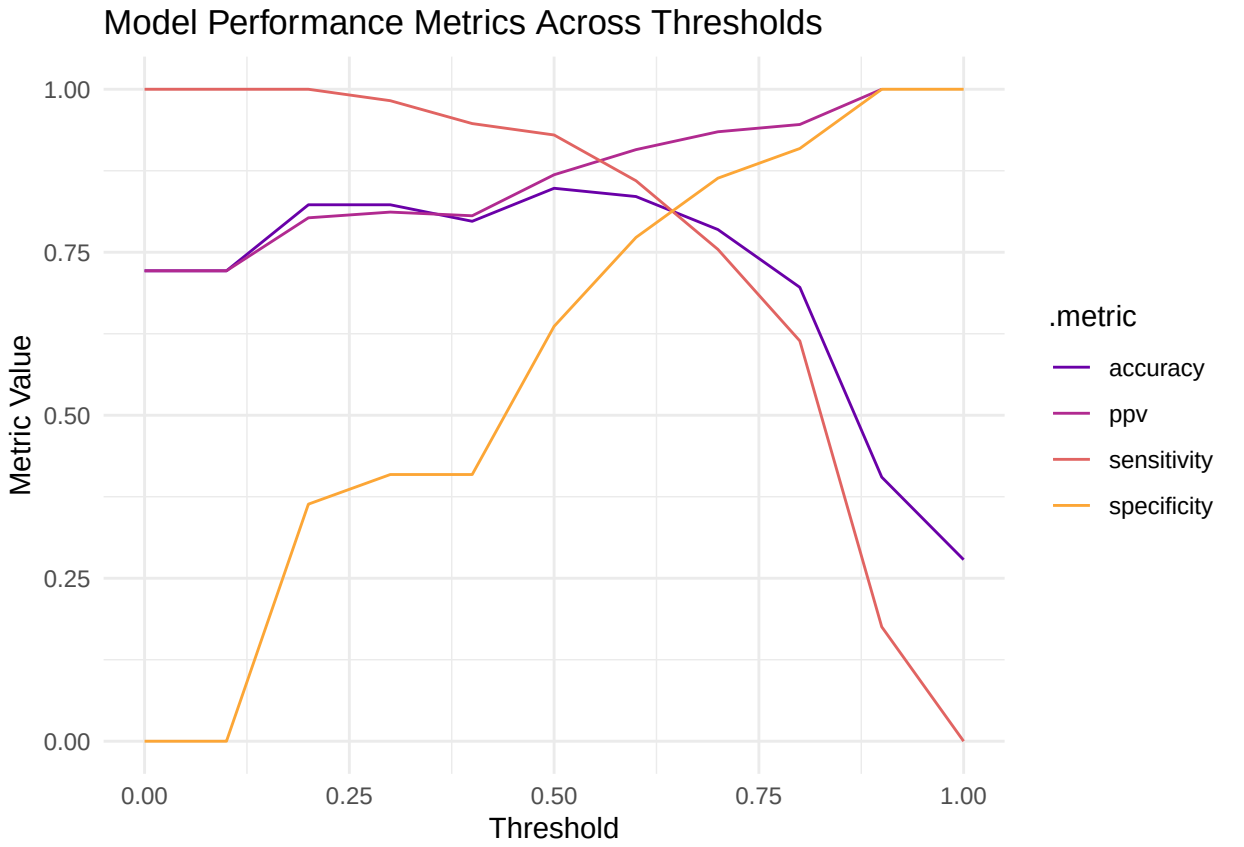
```
# Step 1: Generate a sequence of thresholds
thresholds <- seq(0, 1, by = 0.1)

# Step 2: Calculate metrics for each threshold using map
```

```r
metrics_list <- map_df(thresholds, ~{
  db_results <- db_test %>%
    bind_cols(pred_prob) %>%
    mutate(.pred_class = make_two_class_pred(.pred_neg, levels(diabetes), threshold = .x)) %>%
    select(diabetes, glucose, contains(".pred"))

  metrics <- eval_metrics(data = db_results, truth = diabetes, estimate = .pred_class) %>%
    mutate(threshold = .x) %>%
    select(-2)
  return(metrics)
})

# Step 3: Plot the metrics across thresholds
ggplot(metrics_list, aes(x = threshold, y = .estimate, color = .metric)) +
  geom_line() +
  labs(title = "Model Performance Metrics Across Thresholds",
       x = "Threshold",
       y = "Metric Value") +
  theme_minimal() +
  scale_color_viridis_d(begin = 0.2, end = 0.8, direction = 1, option = "C")
```



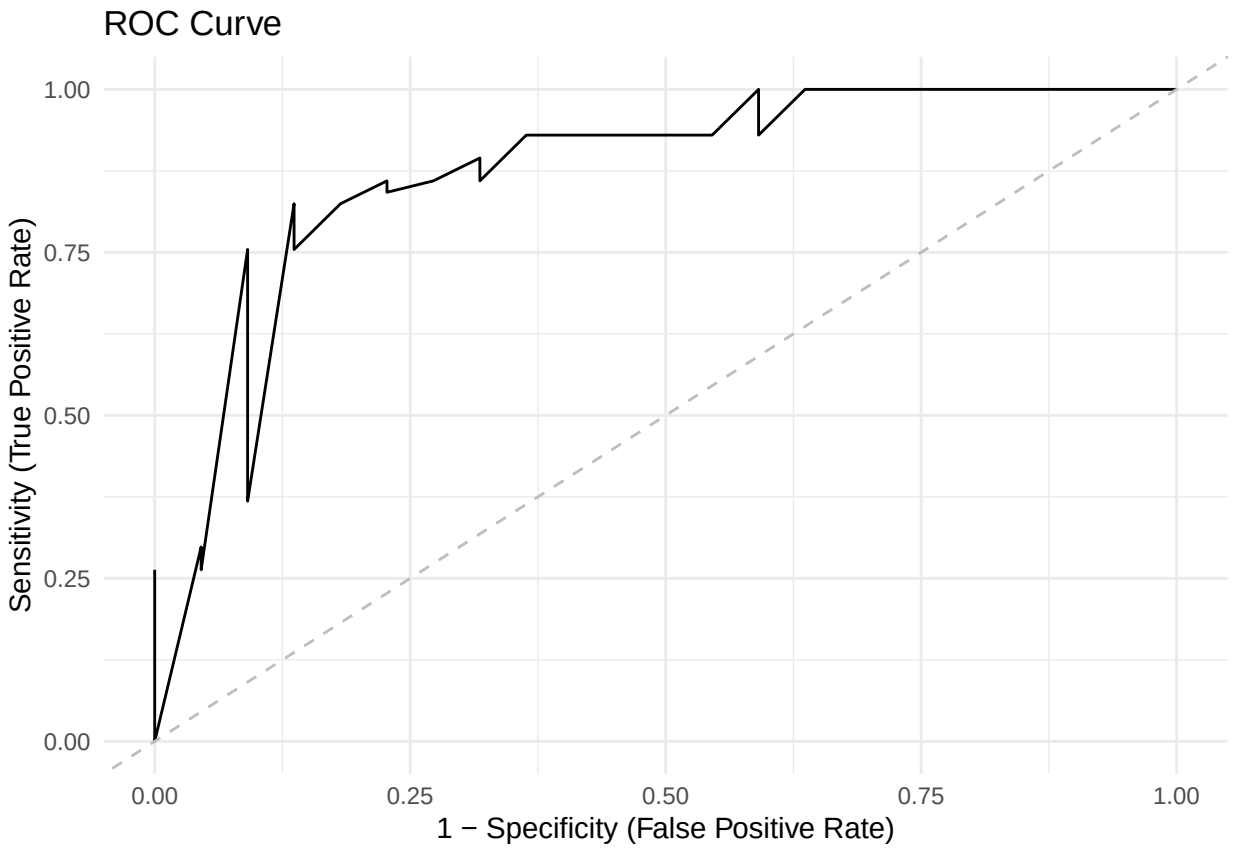
e. Generate a ROC Curve and Determine the Optimal Threshold: Evaluate the performance of your diabetes prediction model by plotting a ROC curve. Use the curve to identify the point that is closest to the top-left corner (maximizing sensitivity and minimizing 1 - specificity), and back-calculate to find the corresponding optimal threshold. This threshold represents the best balance between sensitivity (true positive rate) and specificity (false positive rate).

```r
library(yardstick)

# Predict probabilities on the test set
diabetes_prob <- predict(fitted_logistic_model, db_test, type = "prob")
diabetes_results <- db_test %>%
  select(diabetes) %>%
  bind_cols(diabetes_prob)

# Generate ROC curve data
roc_data <- roc_curve(diabetes_results, truth = diabetes, .pred_neg)


# Plot the ROC curve
roc_plot <- ggplot(roc_data, aes(x = 1 - specificity, y = sensitivity)) +
  geom_line() +
  geom_abline(linetype = "dashed", color = "gray", slope = 1) +
  theme_minimal() +
  labs(title = "ROC Curve", x = "1 - Specificity (False Positive Rate)", y = "Sensitivity (True Positive
roc_plot
```




```r
# Identify the closest point to the top-left corner
optimal_point <- roc_data %>%
  mutate(distance = sqrt((1 - specificity)^2 + (sensitivity - 1)^2)) %>%
  arrange(distance) %>%
  slice(1)
optimal_point
```

```
# A tibble: 1 x 4
  .threshold specificity sensitivity distance
       <dbl>       <dbl>       <dbl>    <dbl>
1      0.681       0.864       0.825    0.222
```