

# Regression and Classification

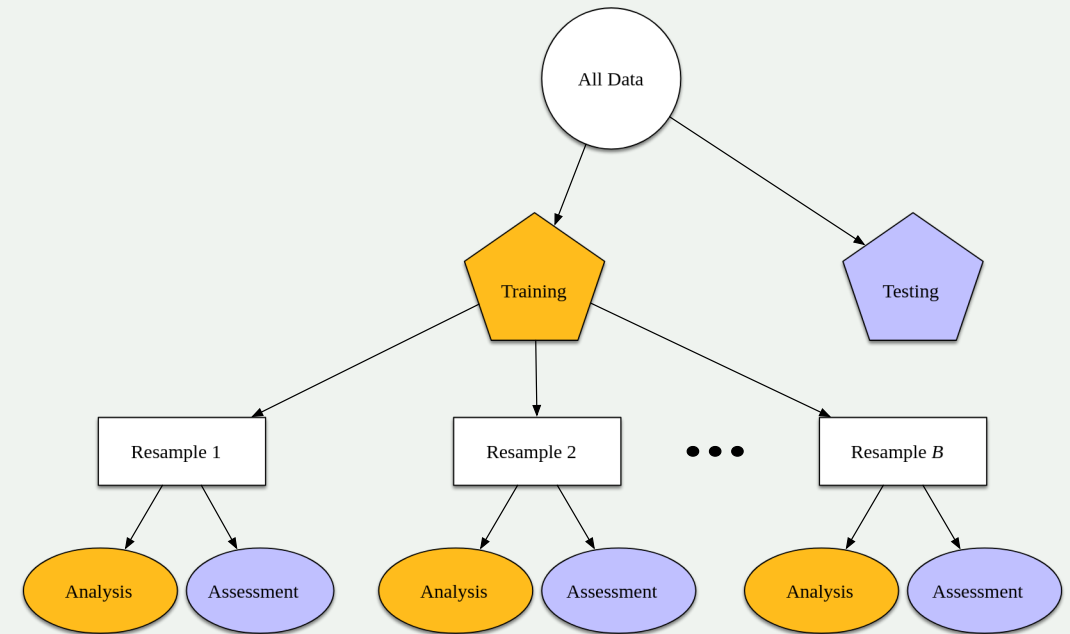
Stat 220

Bastola

March 02 2022

# Resampling methods

Create a series of data sets similar to the training/testing split, always used with the training set



## First, simple linear regression (SLR)

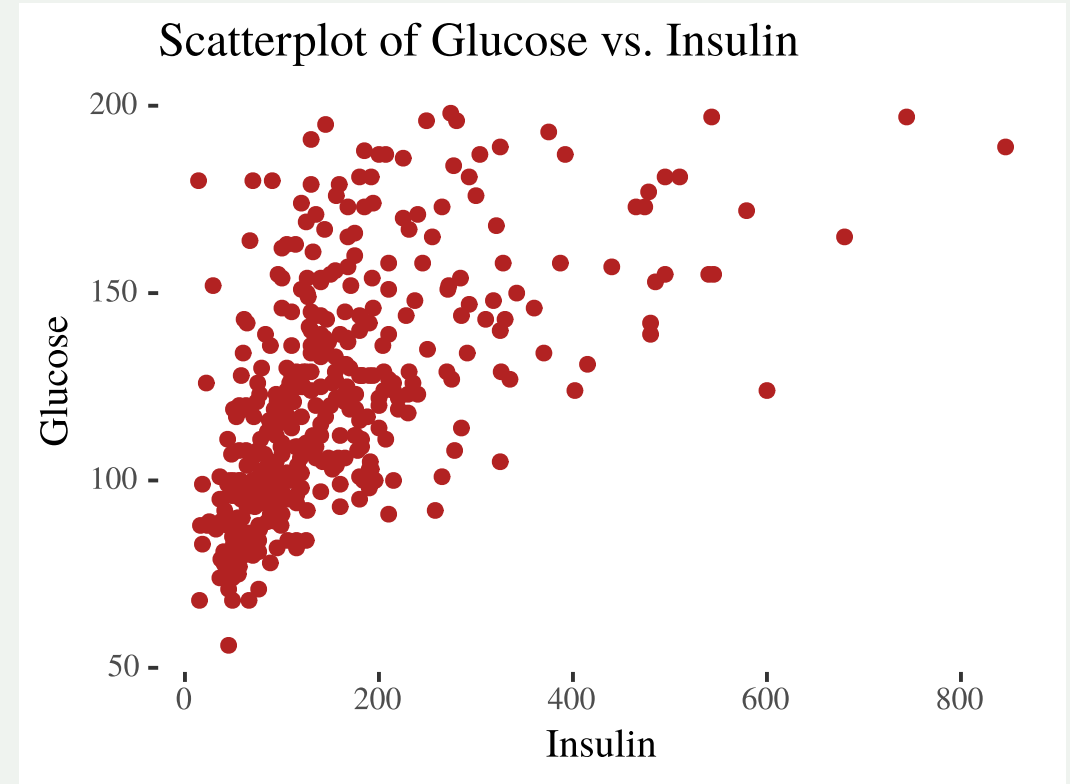
Predicting a numeric outcome when there is just one predictor

$$Y = \beta_0 + \beta_1 X$$

- $\beta$  values are the coefficients and  $X$  is the only model predictor or feature.

# Bivariate data from PimaIndiansDiabetes2

	glucose	insulin
4	89	94
5	137	168
7	78	88
9	197	543
14	189	846
15	166	175
17	118	230
19	103	83
20	115	96
21	126	235
25	143	146
26	125	115
28	97	140
29	145	110
32	158	245



# Specification for a linear regression model

```
lm_spec <- linear_reg() %>%  
  set_mode("regression") %>%  
  set_engine("lm")
```

```
lm_spec  
Linear Regression Model Specification (regression)  
  
Computational engine: lm
```

# Fitting the model

```
lm_fit <- lm_spec %>%  
  fit(glucose ~ insulin, data = db_slr)  
  
lm_fit  
parsnip model object  
  
Fit time: 4ms  
  
Call:  
stats::lm(formula = glucose ~ insulin, data = data)  
  
Coefficients:  
(Intercept)      insulin  
    99.0737      0.1509
```

# Getting the fit

```
lm_fit %>%  
  pluck("fit")
```

Call:

```
stats::lm(formula = glucose ~ insulin, data = data)
```

Coefficients:

(Intercept)	insulin
99.0737	0.1509

```
lm_fit %>%  
  pluck("fit") %>%  
  summary()
```

Call:

```
stats::lm(formula = glucose ~ insulin, data = data)
```

Residuals:

Min	1Q	Median	3Q	Max
-65.633	-17.361	-5.807	12.626	78.813

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	99.0737	2.0979	47.23	<2e-16 ***
insulin	0.1509	0.0107	14.11	<2e-16 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 25.14 on 390 degrees of freedom

Multiple R-squared: 0.3378, Adjusted R-squared: 0.3361

F-statistic: 199 on 1 and 390 DF, p-value: < 2.2e-16



```
predict(lm_fit, new_data = db_slr)
```

```
# A tibble: 392 × 1
```

```
  .pred
```

```
<dbl>
```

```
1  113.
```

```
2  124.
```

```
3  112.
```

```
4  181.
```

```
5  227.
```

```
6  125.
```

```
7  134.
```

```
8  112.
```

```
9  114.
```

```
10 135.
```

```
# ... with 382 more rows
```

# Confidence and Prediction intervals

```
predict(lm_fit, new_data = db_slr,  
        type = "conf_int")
```

```
# A tibble: 392 × 2
```

	.pred_lower <dbl>	.pred_upper <dbl>
1	110.	116.
2	122.	127.
3	109.	115.
4	173.	190.
5	212.	241.
6	123.	128.
7	131.	137.
8	109.	115.
9	111.	116.
10	132.	138.

```
# ... with 382 more rows
```

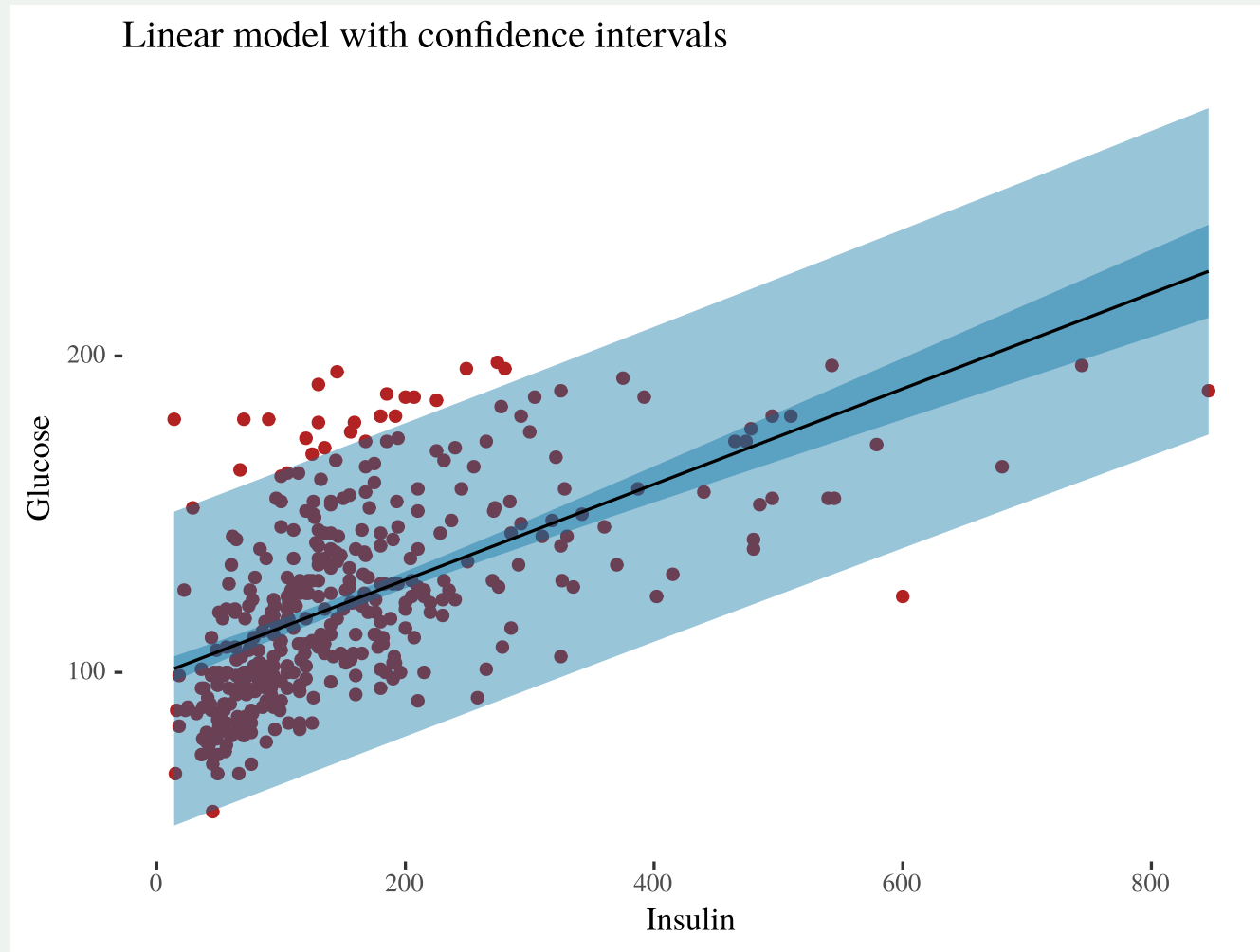
```
predict(lm_fit, new_data = db_slr,  
        type = "pred_int")
```

```
# A tibble: 392 × 2
```

	.pred_lower <dbl>	.pred_upper <dbl>
1	63.7	163.
2	74.9	174.
3	62.8	162.
4	131.	231.
5	175.	278.
6	76.0	175.
7	84.3	183.
8	62.1	161.
9	64.0	163.
10	85.0	184.

```
# ... with 382 more rows
```

# Confidence and Prediction intervals



## Multiple linear regression (MLR)

Predicting a continuous response with a set of  $p$  predictors. predic

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k$$

- $\beta_i$ 's are the coefficients of the model and  $X_i$ 's are the predictors.

## Fitting a MLR

```
db_mlr <- db %>% select(-diabetes)

lm_fit2 <- lm_spec %>%
  fit(glucose ~ ., data = db_mlr)
```

# Extract parameter estimates

```
tidy(lm_fit2)
```

```
# A tibble: 8 × 5
```

	term <chr>	estimate <dbl>	std.error <dbl>	statistic <dbl>	p.value <dbl>
1	(Intercept)	60.0	8.44	7.11	5.65e-12
2	pregnant	0.0738	0.523	0.141	8.88e- 1
3	pressure	0.213	0.108	1.98	4.82e- 2
4	triceps	0.0743	0.158	0.471	6.38e- 1
5	insulin	0.133	0.0108	12.3	1.62e-29
6	mass	0.130	0.244	0.535	5.93e- 1
7	pedigree	4.18	3.62	1.15	2.50e- 1
8	age	0.577	0.172	3.36	8.57e- 4

# Predict new values

```
predict(lm_fit2, new_data = db_mlr)
# A tibble: 392 × 1
  .pred
  <dbl>
1  105.
2  128.
3  105.
4  186.
5  227.
6  136.
7  138.
8  106.
9  115.
10 137.
# ... with 382 more rows
```

# Actual and predicted values

```
bind_cols(  
  predict(lm_fit, new_data = db_mlr), db_mlr) %>% select(glucose, .pred)  
# A tibble: 392 × 2  
  glucose .pred  
  <dbl> <dbl>  
1      89  113.  
2     137  124.  
3      78  112.  
4     197  181.  
5     189  227.  
6     166  125.  
7     118  134.  
8     103  112.  
9     115  114.  
10     126  135.  
# ... with 382 more rows
```



# Data Splitting

```
set.seed(1234)

db_split <- initial_split(db_mlr,
                          prop = 0.80,
                          strata = age,
                          breaks = 5)

db_train <- training(db_split)
db_test  <- testing(db_split)
```

# Recipe

```
db_recipe <- recipe(glucose ~ ., data = db_train) %>%  
  step_scale(all_predictors()) %>%  
  step_center(all_predictors()) %>% prep()
```

db\_recipe  
Recipe

Inputs:

	role	#variables
outcome		1
predictor		7

Training data contained 311 data points and no missing data.

Operations:

Scaling for pregnant, pressure, triceps, insulin, mass, ped... [trained]  
Centering for pregnant, pressure, triceps, insulin, mass, ped... [trained]

# Model Building

```
lm_spec <- # your model specification  
  linear_reg() %>% # model type  
  set_engine(engine = "lm") %>% # model engine  
  set_mode("regression") # model mode
```

```
# Show your model specification  
lm_spec  
Linear Regression Model Specification (regression)  
  
Computational engine: lm
```

## Create workflow

```
lm_wflow <-  
  workflow() %>%  
  add_model(lm_spec) %>%  
  add_recipe(db_recipe)
```

# Create Validation Set

```
set.seed(1234)

cv_folds <- vfold_cv(db_train,
  v = 5,
  strata = age,
  breaks = 5)
```

```
cv_folds
# 5-fold cross-validation using stratification
# A tibble: 5 × 2
  splits          id
  <list>        <chr>
1 <split [247/64]> Fold1
2 <split [247/64]> Fold2
3 <split [249/62]> Fold3
4 <split [250/61]> Fold4
5 <split [251/60]> Fold5
```

# Common metrics for regression

## Root mean square error (RMSE)

- the standard deviation of the residuals (prediction errors)
- smaller is better

## Coefficient of determination, $R^2$

- proportion of the variation in the outcome that is predictable from the predictors
- larger is better

# Fit the model

```
get_model <- function(x) {  # Function to extract fit
  extract_fit_parsnip(x) %>% tidy()
}
```

```
lm_wflow_eval <- lm_wflow %>%
  fit_resamples(
    resamples = cv_folds,
    metrics = metric_set(rmse, rsq),
    control = control_resamples(
      save_pred = TRUE,
      extract = get_model)
  )
lm_wflow_eval%>%collect_metrics()
# A tibble: 2 × 6
  .metric .estimator   mean     n std_err .config
  <chr>   <chr>       <dbl> <int>   <dbl> <chr>
1 rmse    standard    24.8     5  0.870 Preprocessor1_Model1
2 rsq     standard    0.417     5  0.0442 Preprocessor1_Model1
```

# Last fit and evaluation

```
last_fit_lm <- last_fit(lm_wflow, split = db_split)
```

```
last_fit_lm %>%  
  collect_metrics()  
# A tibble: 2 × 4  
  .metric .estimator .estimate .config  
  <chr>    <chr>         <dbl> <chr>  
1 rmse     standard      24.4   Preprocessor1_Model1  
2 rsq      standard       0.312  Preprocessor1_Model1
```



# Extract the estimates

```
lm_wflow_eval$.extracts[[1]][[1]]  
[[1]]  
# A tibble: 8 × 5  
  term          estimate std.error statistic    p.value  
  <chr>          <dbl>      <dbl>    <dbl>    <dbl>  
1 (Intercept)  124.         1.56     79.5 6.74e-174  
2 pregnant    -1.19         2.17    -0.547 5.85e- 1  
3 pressure     4.08         1.73     2.36 1.91e- 2  
4 triceps      0.392        2.13     0.184 8.54e- 1  
5 insulin     15.8         1.68     9.42 4.13e- 18  
6 mass         0.923        2.18     0.424 6.72e- 1  
7 pedigree     0.323        1.61     0.201 8.41e- 1  
8 age          5.85         2.25     2.60 9.92e- 3
```

# Logistic Regression

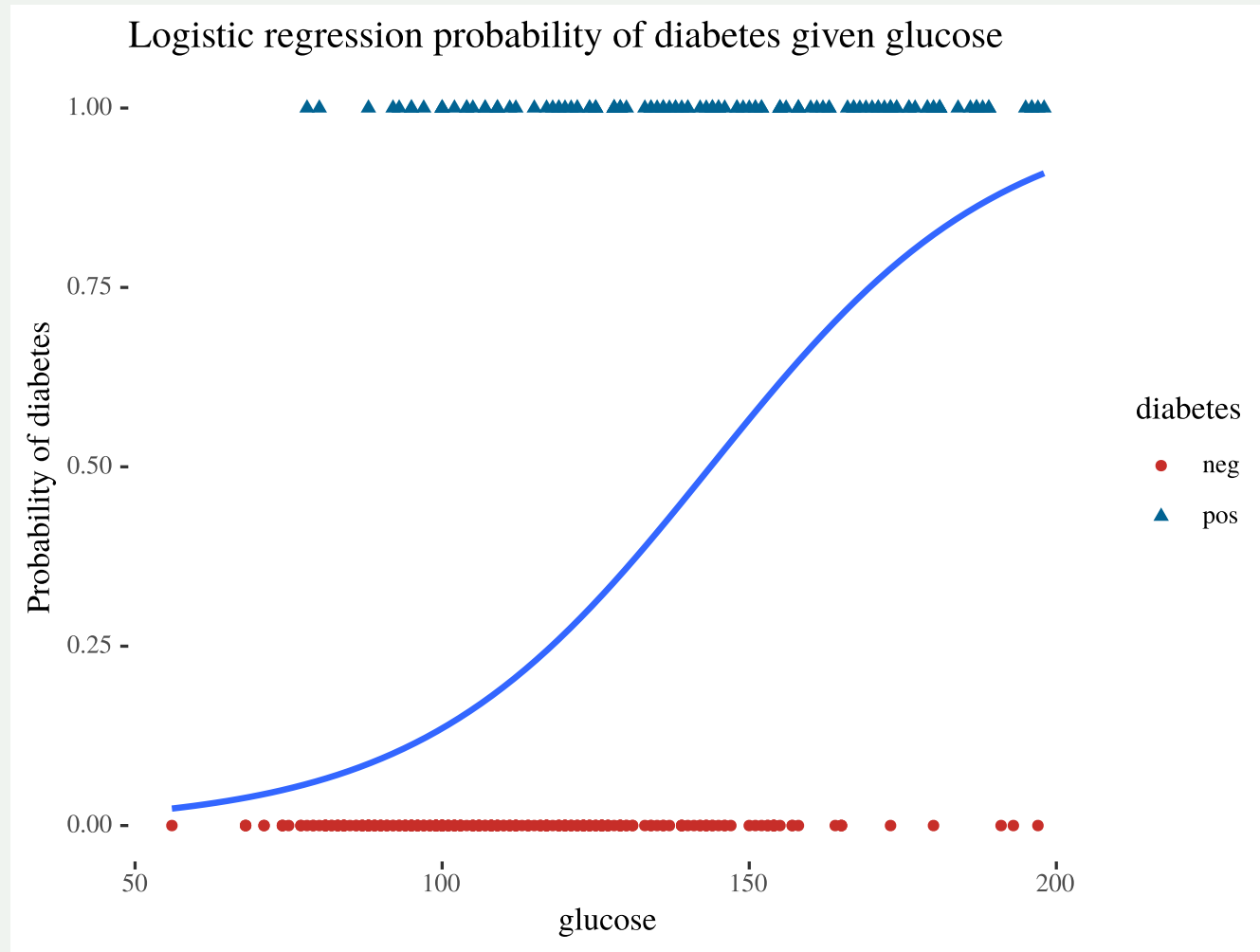
Binary response,  $Y$ , with a set of  $p$  explanatory (predictor, features) variables,  $X_1, \dots, X_p$ . We model the probability that  $Y$  belongs to a particular category.

$$P(Y = 1) = \frac{e^{\beta_0 + \beta_1 + \dots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 + \dots + \beta_p X_p}}$$

$$\text{Odds} = \frac{P(Y = 1)}{1 - P(Y = 1)} = e^{\beta_0 + \beta_1 + \dots + \beta_p X_p}$$

$$\text{Log Odds} = \beta_0 + \beta_1 + \dots + \beta_p X_p$$

# Logistic regression with just one predictor



# Train and Test Split

```
# Create data split for train and test
set.seed(1234)
db_single <- db %>% select(diabetes, glucose)
db_split <- initial_split(db_single, prop = 0.80, strata = diabetes)
```

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

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

## Steps

1. Call the model function
2. Supply the family of the model
3. Supply the type of model you want to fit
4. Fit the model

```
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 the Summary

```
tidy(fitted_logistic_model)
# A tibble: 2 × 5
  term          estimate std.error statistic  p.value
<chr>         <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)  -6.26      0.718     -8.72  2.90e-18
2 glucose      0.0439    0.00545     8.04  8.81e-16
```

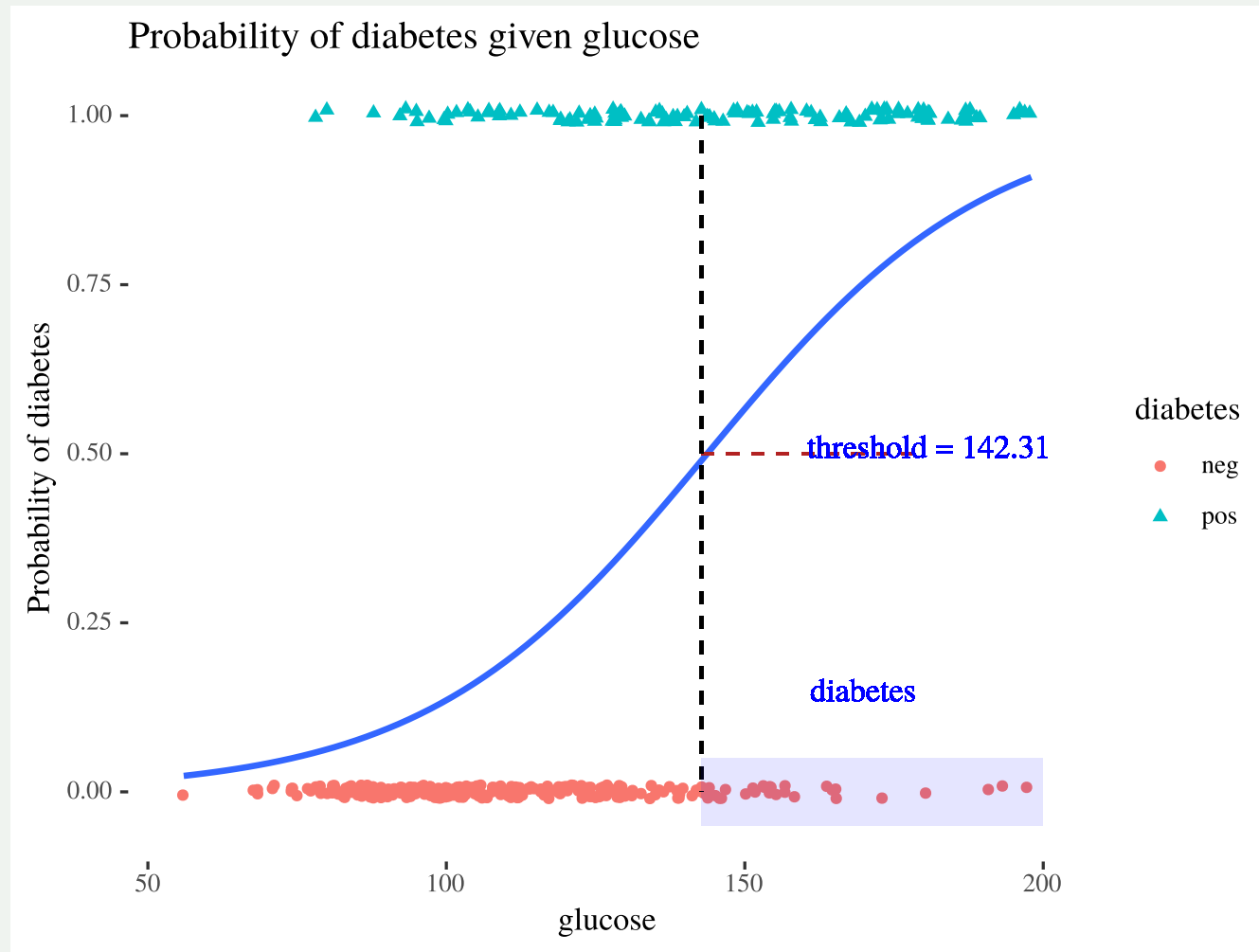
# Odds Ratio

$$ODDS = \frac{probability}{1 - probability}$$

```
tidy(fitted_logistic_model, exponentiate = TRUE)
# A tibble: 2 × 5
  term          estimate std.error statistic  p.value
<chr>         <dbl>    <dbl>    <dbl>    <dbl>
1 (Intercept)  0.00191    0.718    -8.72  2.90e-18
2 glucose      1.04      0.00545    8.04  8.81e-16
```



# Threshold for classification



## Your Turn 1

05:00

Please clone the repository on [logistic regression](#) to your local folder.

$$P(Y = 1) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

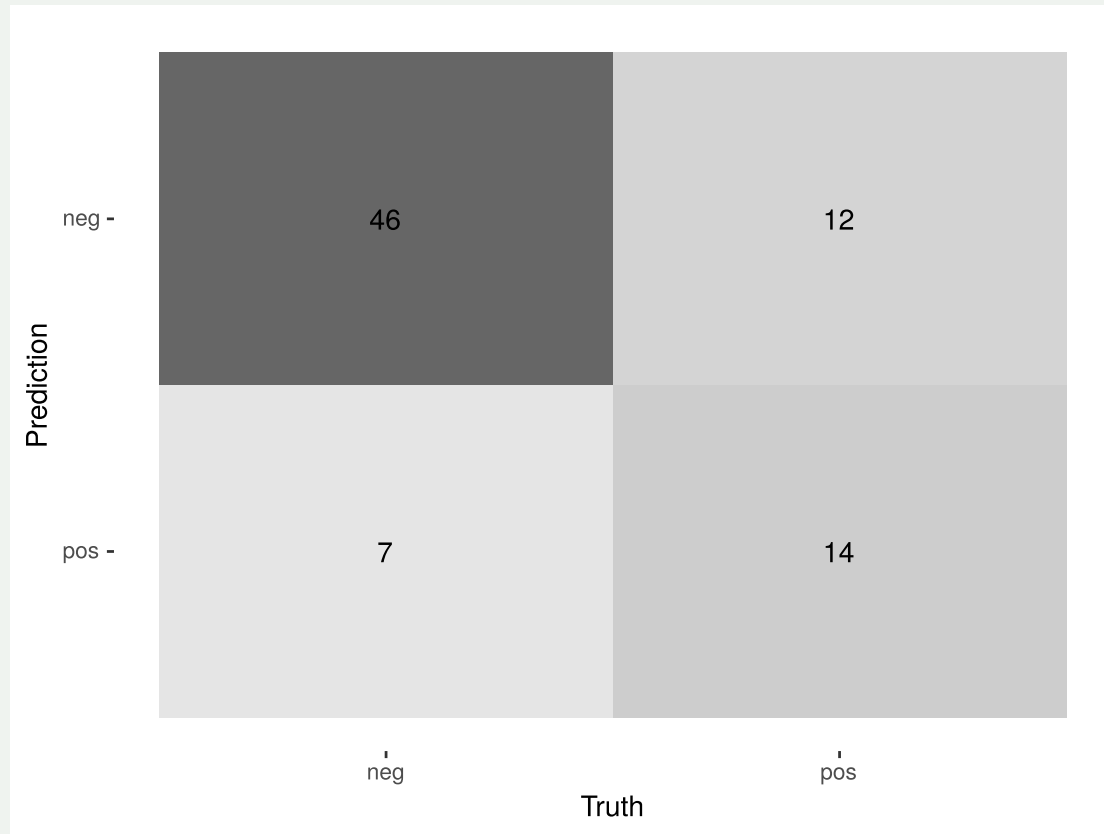
- Verify that the glucose value of 142.31 gives the probability of having diabetes as 1/2.
- What value of glucose gives us a probability threshold (of having diabetes) of 0.75?

# Class Prediction

Use the predict function and supply the trained model object, test dataset and the type of variable to predict

```
# Class prediction
pred_class <- predict(fitted_logistic_model, new_data = db_test,
                      type = "class") # default 0.5 probability threshold
```

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



# Class Probabilities

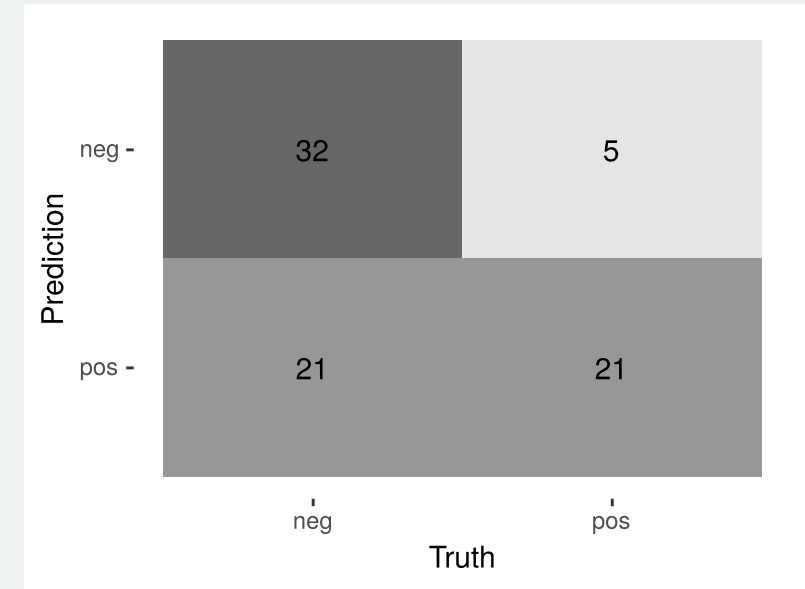
```
# Prediction Probabilities
pred_prob <- predict(fitted_logistic_model,
                    new_data = db_test,
                    type = "prob")
```

```
db_results <- db_test %>%
  bind_cols(pred_prob) %>%
  mutate(.pred_class = make_two_class_pred(.pred_neg, levels(diabetes),
                                          threshold = .75)) %>%

  select(diabetes, glucose, contains(".pred"))
```

# Results

```
head(db_results,12)
  diabetes glucose .pred_neg .pred_pos .pred_class
4      neg     89 0.91349756 0.08650244      neg
5      pos    137 0.56272912 0.43727088      pos
9      pos    197 0.08480112 0.91519888      pos
20     pos    115 0.77153063 0.22846937      neg
25     pos    143 0.49728445 0.50271555      pos
26     pos    125 0.68534836 0.31465164      pos
29     neg    145 0.47537870 0.52462130      pos
40     pos    111 0.80097183 0.19902817      neg
44     pos    171 0.22466256 0.77533744      pos
58     neg    100 0.86700622 0.13299378      neg
69     neg     95 0.89031867 0.10968133      neg
89     pos    136 0.57348808 0.42651192      pos
```



# Custom Metrics

```
custom_metrics <- metric_set(accuracy, sens, spec, ppv)
```

```
custom_metrics(db_results,  
               truth = diabetes,  
               estimate = .pred_class)
```

```
# A tibble: 4 × 3
```

	.metric	.estimator	.estimate
	<chr>	<chr>	<dbl>
1	accuracy	binary	0.671
2	sens	binary	0.604
3	spec	binary	0.808
4	ppv	binary	0.865

# ROC-AUC (Receiver Operator Characteristic- Area Under Curve)

Uses the class probability estimates to give us a sense of performance across the entire set of potential probability cutoffs

```
db_results %>% roc_auc(truth = diabetes, .pred_neg)
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>    <chr>      <dbl>
1 roc_auc binary      0.802
```

- ROC\_AUC tells how much the model is capable of distinguishing between classes.



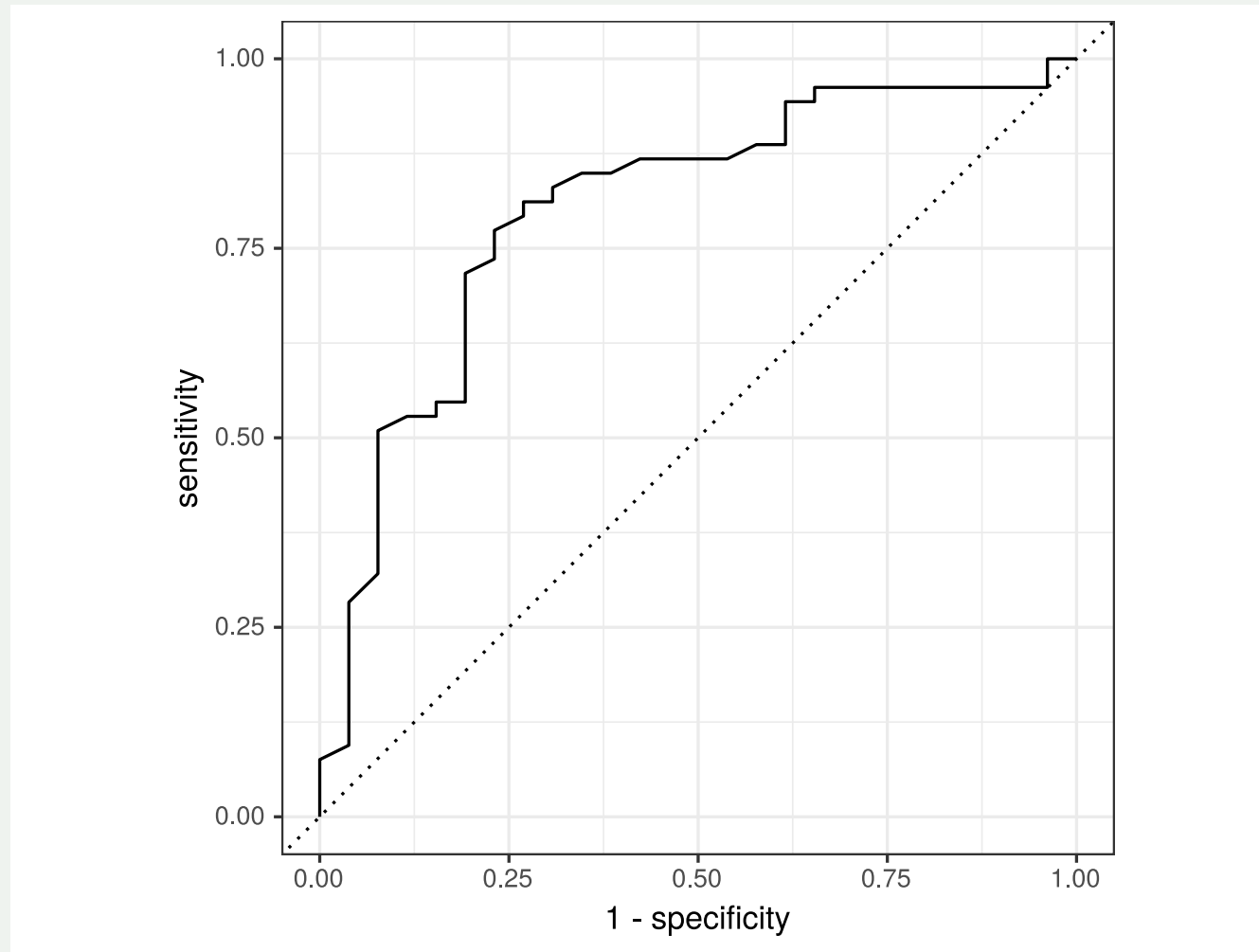
# ROC-AUC

plotted with TPR/Recall/Sensitivity against the FPR/ (1- Specificity), where TPR is on the y-axis and FPR is on the x-axis

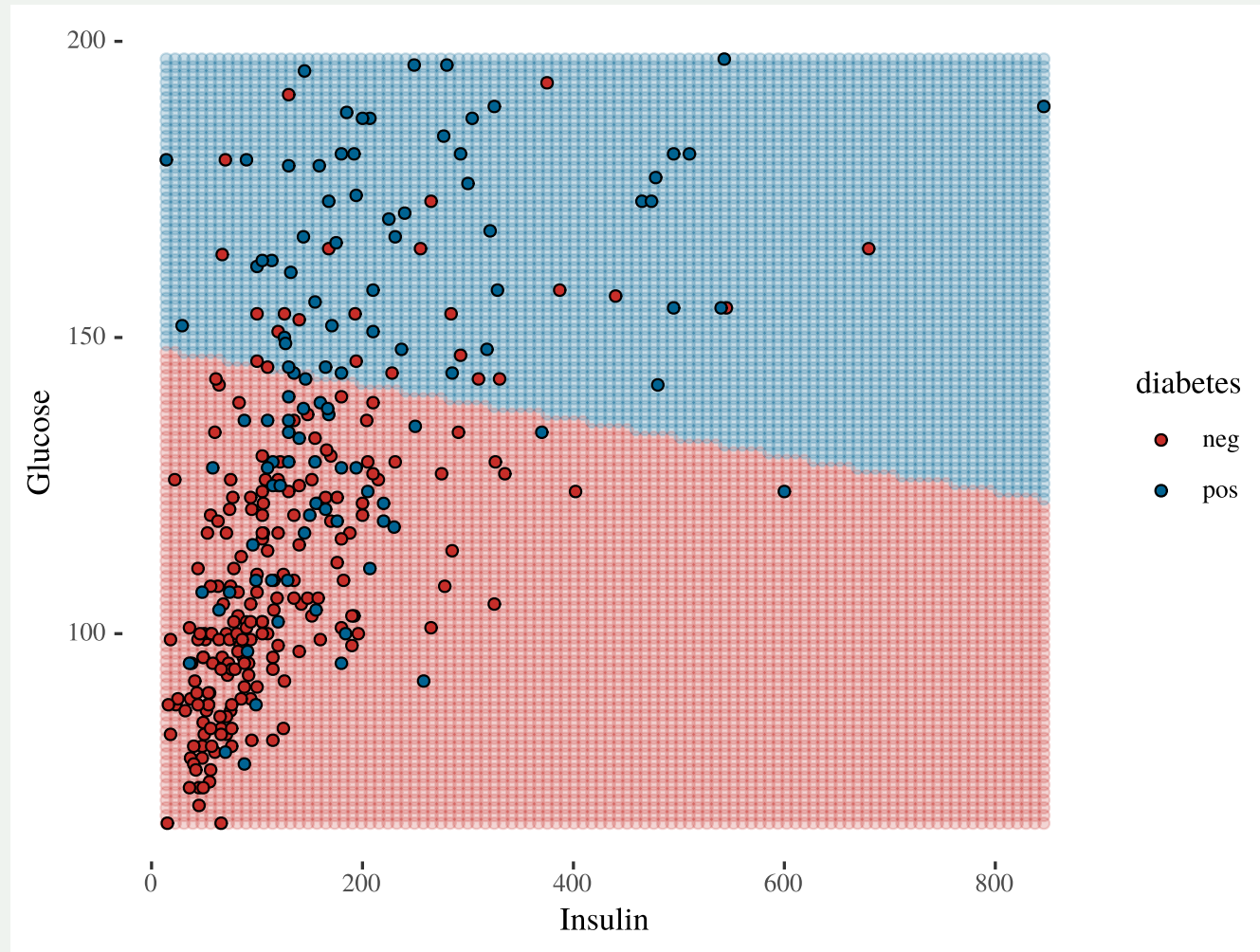
- ROC curves with area = 1 under the curve (AUC) are perfect classifiers
- ROC curves with area = 0.5 AUC are just as good as random guesses

```
db_results %>%  
  roc_curve(truth = diabetes, .pred_neg) %>%  
  autoplot()
```

# ROC Curve



# Decision boundary



# Let's look at the full model

```
# Create data split for train and test
set.seed(1234)
db_split <- initial_split(db, prop = 0.80, strata = diabetes)
```

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

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

# Model Tuning with a Cross Validation

```
set.seed(100)

cv_folds <-
  vfold_cv(db_train,
            v = 5,
            strata = diabetes)
```

# Recipe

```
db_recipe <- recipe(diabetes ~ ., data = db_train) %>%  
  step_scale(all_predictors()) %>%  
  step_center(all_predictors()) %>% prep()
```

## Specify the model

```
log_spec <- # your model specification  
  logistic_reg() %>% # model type  
  set_engine(engine = "glm") %>% # model engine  
  set_mode("classification") # model mode
```

## Workflow

```
log_wflow <- # new workflow object  
workflow() %>% # use workflow function  
add_recipe(db_recipe) %>% # use the new recipe  
add_model(log_spec) # add your model spec
```

## Fit, tune, and evaluate

```
log_res_2 <-  
  log_wflow %>%  
  fit_resamples(  
    resamples = cv_folds,  
    metrics = metric_set(  
      recall, precision,  
      accuracy, kap,  
      roc_auc, sens, spec),  
    control = control_resamples(  
      save_pred = TRUE,  
      extract = get_model) # use extract function as before  
  )
```



# Extract the model

```
log_res_2$.extracts[[1]][[1]]  
[[1]]
```

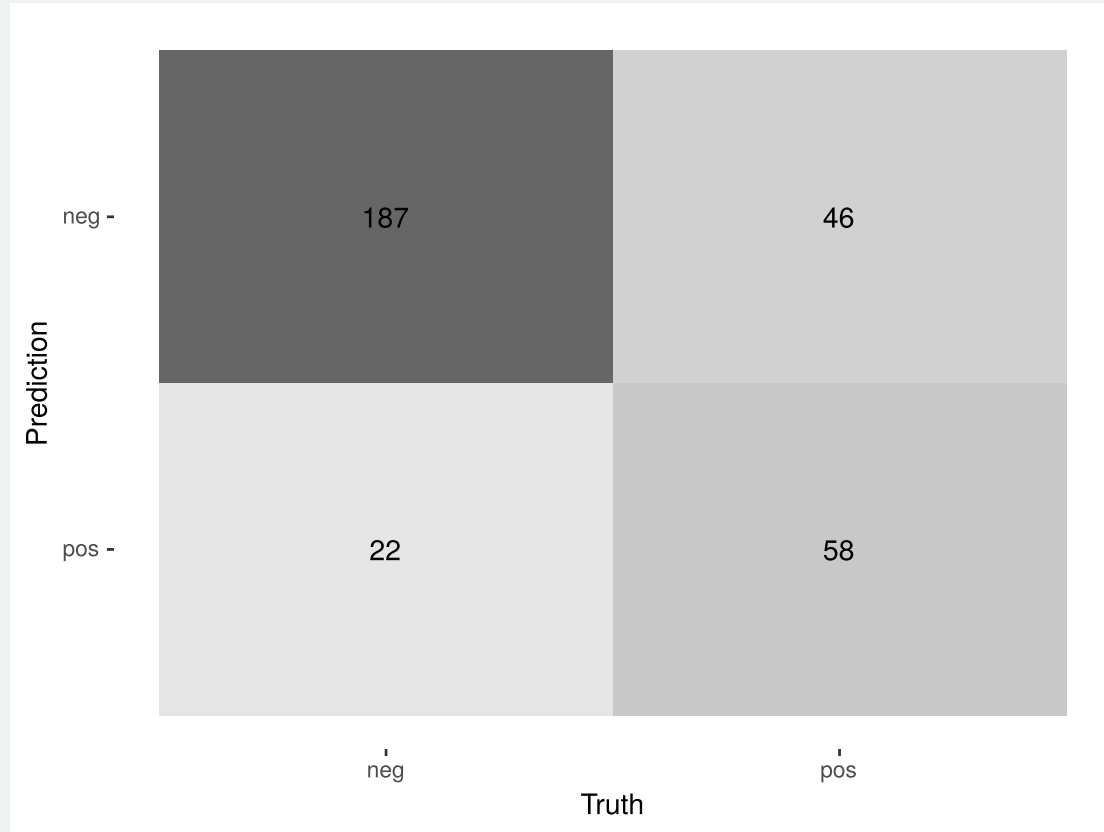
```
# A tibble: 9 × 5
```

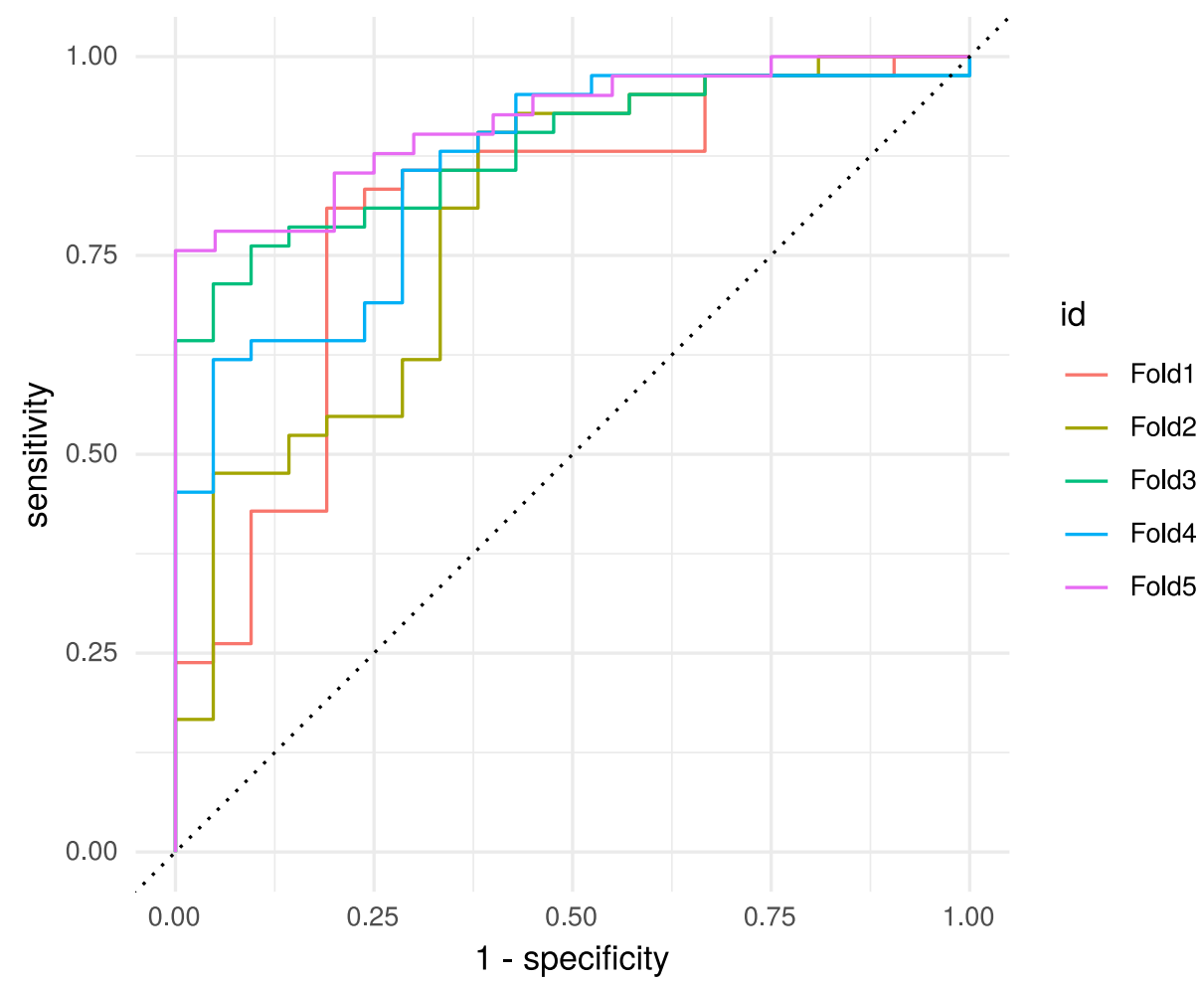
	term <chr>	estimate <dbl>	std.error <dbl>	statistic <dbl>	p.value <dbl>
1	(Intercept)	-1.05	0.188	-5.58	0.00000000242
2	pregnant	0.379	0.237	1.60	0.110
3	glucose	1.30	0.239	5.42	0.00000000580
4	pressure	-0.102	0.182	-0.560	0.576
5	triceps	0.326	0.225	1.45	0.147
6	insulin	-0.225	0.198	-1.14	0.255
7	mass	0.461	0.238	1.94	0.0530
8	pedigree	0.422	0.184	2.29	0.0219
9	age	0.367	0.241	1.52	0.128

## Collect the metrics

```
log_res_2 %>% collect_metrics(summarize = TRUE)
# A tibble: 7 × 6
  .metric      .estimator  mean      n std_err .config
  <chr>      <chr>      <dbl> <int>   <dbl> <chr>
1 accuracy  binary      0.783     5  0.0159 Preprocessor1_Model1
2 kap       binary      0.480     5  0.0405 Preprocessor1_Model1
3 precision binary      0.803     5  0.0139 Preprocessor1_Model1
4 recall    binary      0.895     5  0.0141 Preprocessor1_Model1
5 roc_auc   binary      0.851     5  0.0234 Preprocessor1_Model1
6 sens      binary      0.895     5  0.0141 Preprocessor1_Model1
7 spec      binary      0.558     5  0.0381 Preprocessor1_Model1
```

```
log_pred <- log_res_2 %>%  
  collect_predictions()
```





# Optimal cut-off

