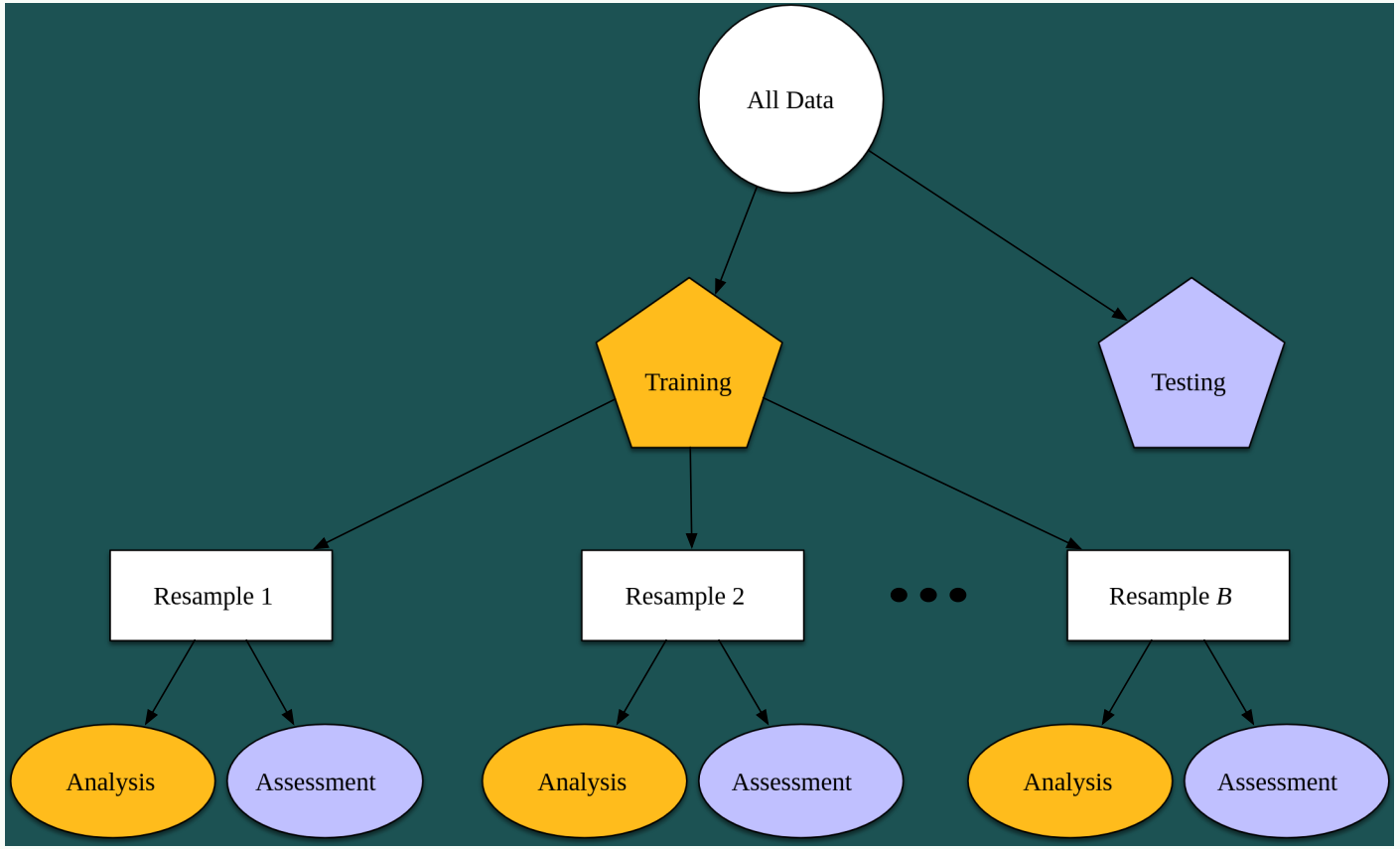
Cross Validation and Logistic Regression

Spring 2023

May 19 2023

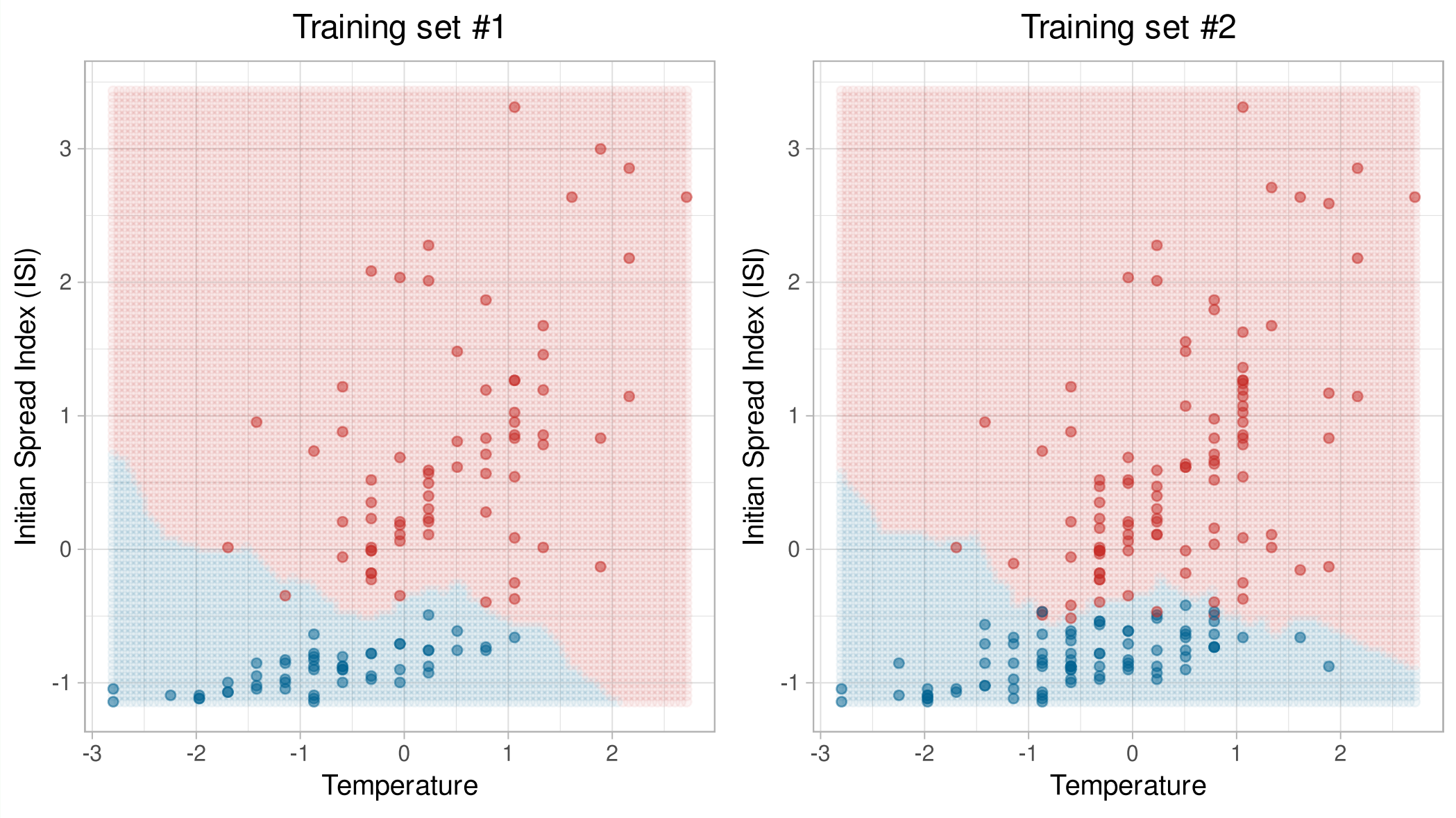
Resampling methods



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

Kuhn and Johnson (2019)

Why not to use single (training) test set



Cross validation

Idea: Split the training data up into multiple training-validation pairs, evaluate the classifier on each split and average the performance metrics

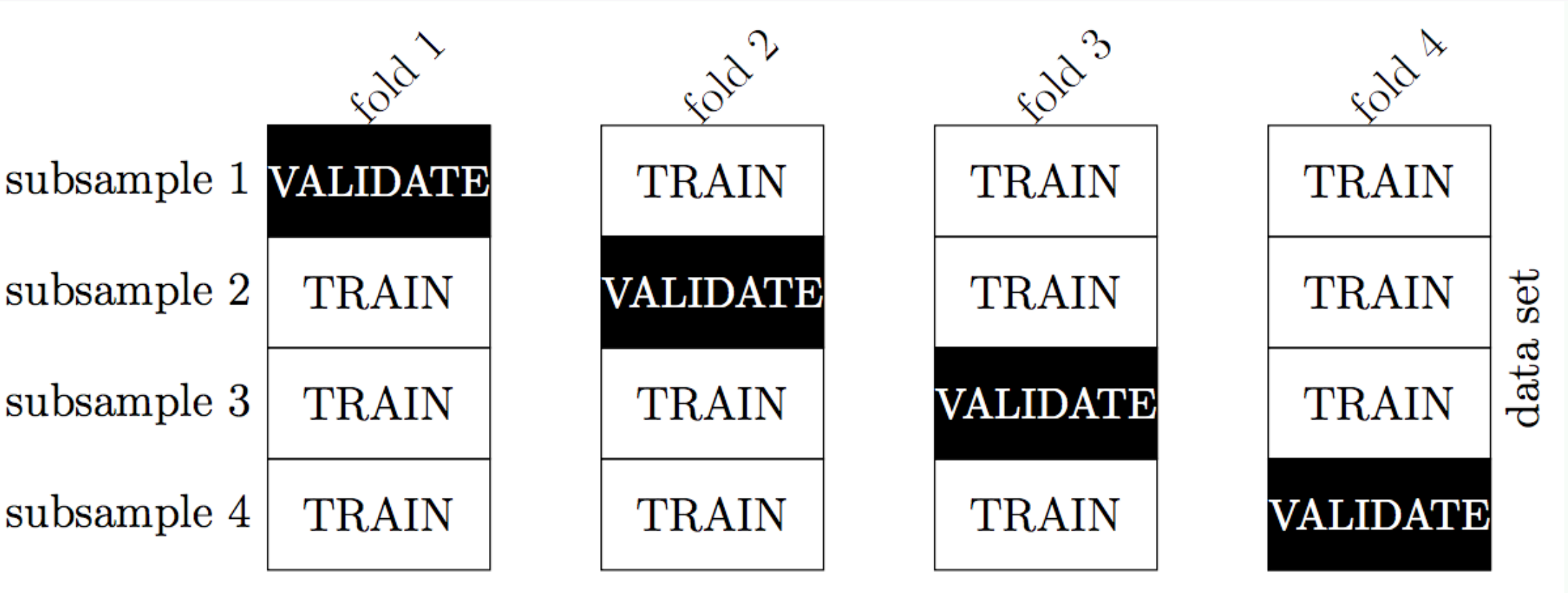
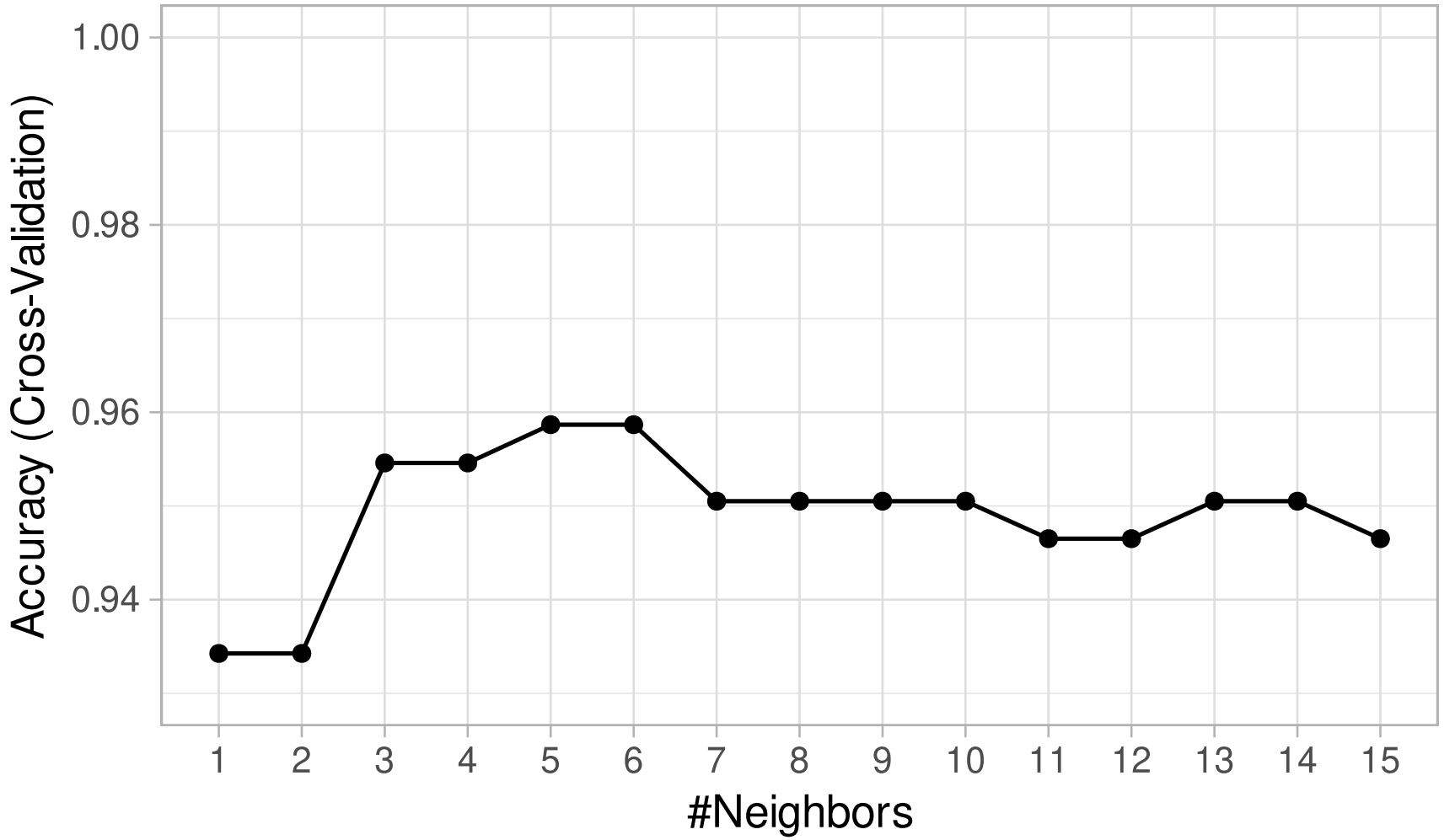


Image courtesy of Dennis Sun

k-fold cross validation

1. split the data into k subsets
2. combine the first $k-1$ subsets into a training set and train the classifier
3. evaluate the model predictions on the last (i.e. kth) held-out subset
4. repeat steps 2-3 k times (i.e. k "folds"), each time holding out a different one of the k subsets
5. calculate performance metrics from each validation set
6. average each metric over the k folds to come up with a single estimate of that metric



- Based on accuracy, $k = 7$ appears best, We C=can look at other metrics. Notice that accuracy doesn't always decrease with k

5-fold cross validation

Creating the recipe

```
fire_recipe <- recipe(classes ~  temperature + isi, data = fire_train) %>%
  step_scale(all_predictors()) %>%
  step_center(all_predictors()) %>%
  prep()
```

5-fold cross validation

Create your model specification and use `tune()` as a placeholder for the number of neighbors

```
knn_spec <- nearest_neighbor(mode = "classification",
                             engine = "kknn",
                             weight_func = "rectangular",
                             neighbors = tune())
```

Split the `fire_train` data set into `v = 5` folds, stratified by `classes`

```
fire_vfold <- vfold_cv(fire_train, v = 5, strata = classes)
```

5-fold cross validation

Create a grid of K values, the number of neighbors

```
k_vals <- tibble(neighbors = seq(from = 1, to = 15, by = 1))
```

Run 5-fold CV on the k_vals grid, storing four performance metrics

```
knn_fit <- workflow() %>%
  add_recipe(fire_recipe) %>%
  add_model(knn_spec) %>%
  tune_grid(resamples = fire_vfold,
            grid = k_vals,
            metrics = metric_set(accuracy, sensitivity, specificity, ppv))
```

Choosing K

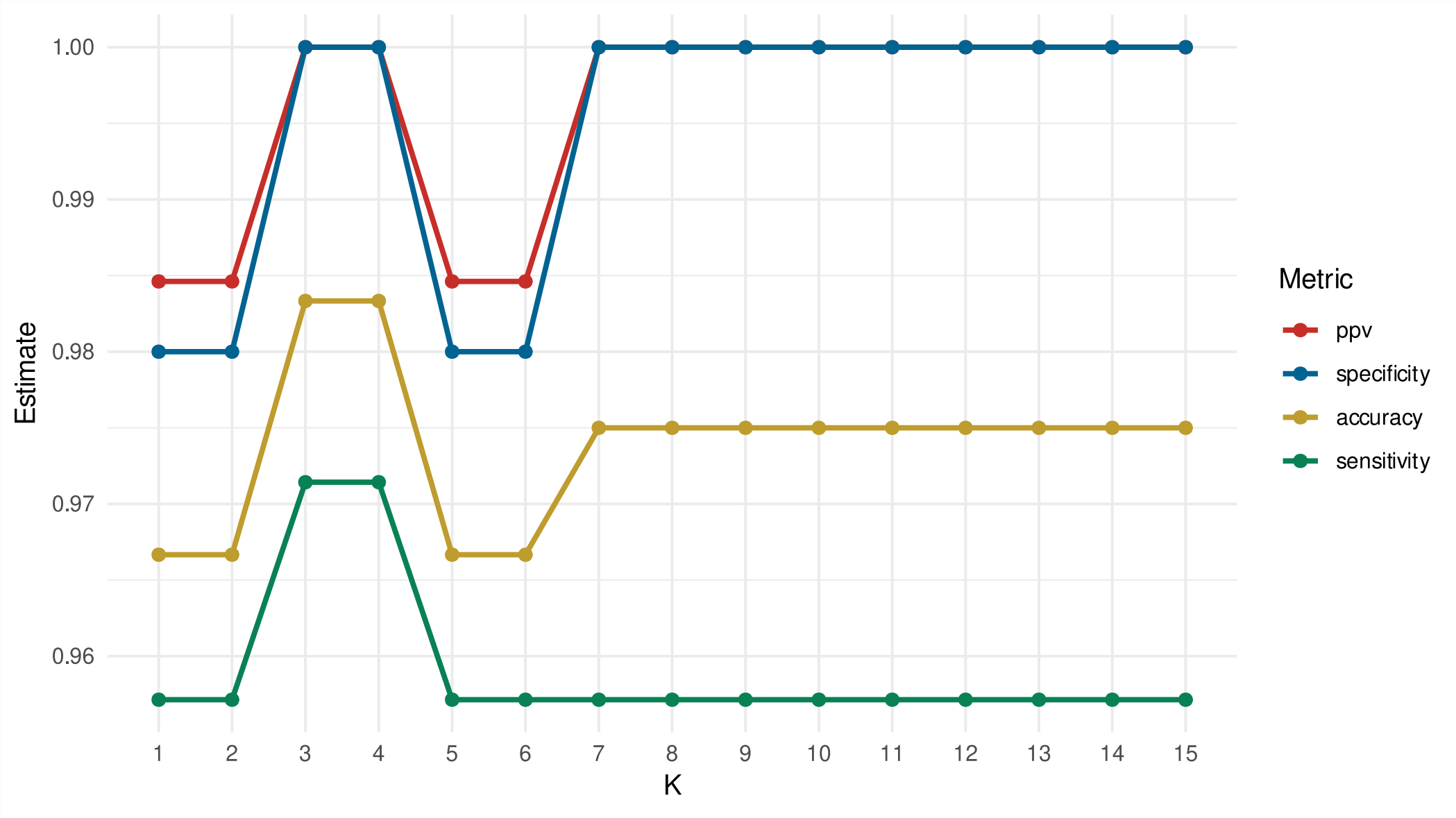
Collect the performance metrics and find the best model

```
cv_metrics <- collect_metrics(knn_fit)
cv_metrics %>% head(6)
# A tibble: 6 × 7
  neighbors .metric     .estimator  mean     n std_err .config
      <dbl> <chr>       <chr>      <dbl> <int>   <dbl> <chr>
1         1 accuracy    binary     0.967     5  0.0243 Preprocessor1_Model01
2         1 ppv         binary     0.985     5  0.0154 Preprocessor1_Model01
3         1 sensitivity binary     0.957     5  0.0286 Preprocessor1_Model01
4         1 specificity binary     0.98      5  0.02   Preprocessor1_Model01
5         2 accuracy    binary     0.967     5  0.0243 Preprocessor1_Model02
6         2 ppv         binary     0.985     5  0.0154 Preprocessor1_Model02
```

Choosing K

```
cv_metrics %>%
  group_by(.metric) %>%
  slice_max(mean)
# A tibble: 26 × 7
# Groups:   .metric [4]
   neighbors .metric  .estimator  mean     n std_err .config
       <dbl> <chr>    <chr>       <dbl> <int>   <dbl> <chr>
 1         3 accuracy binary      0.983     5  0.0167 Preprocessor1_Model03
 2         4 accuracy binary      0.983     5  0.0167 Preprocessor1_Model04
 3         3 ppv      binary      1         5  0      Preprocessor1_Model03
 4         4 ppv      binary      1         5  0      Preprocessor1_Model04
 5         7 ppv      binary      1         5  0      Preprocessor1_Model07
 6         8 ppv      binary      1         5  0      Preprocessor1_Model08
 7         9 ppv      binary      1         5  0      Preprocessor1_Model09
 8        10 ppv      binary      1         5  0      Preprocessor1_Model10
 9        11 ppv      binary      1         5  0      Preprocessor1_Model11
10        12 ppv      binary      1         5  0      Preprocessor1_Model12
# … with 16 more rows
```

Choosing K


1.00
0.99
0.98
0.97
0.96
Estimate
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
K
Metric
ppv
specificity
accuracy
sensitivity

The full process

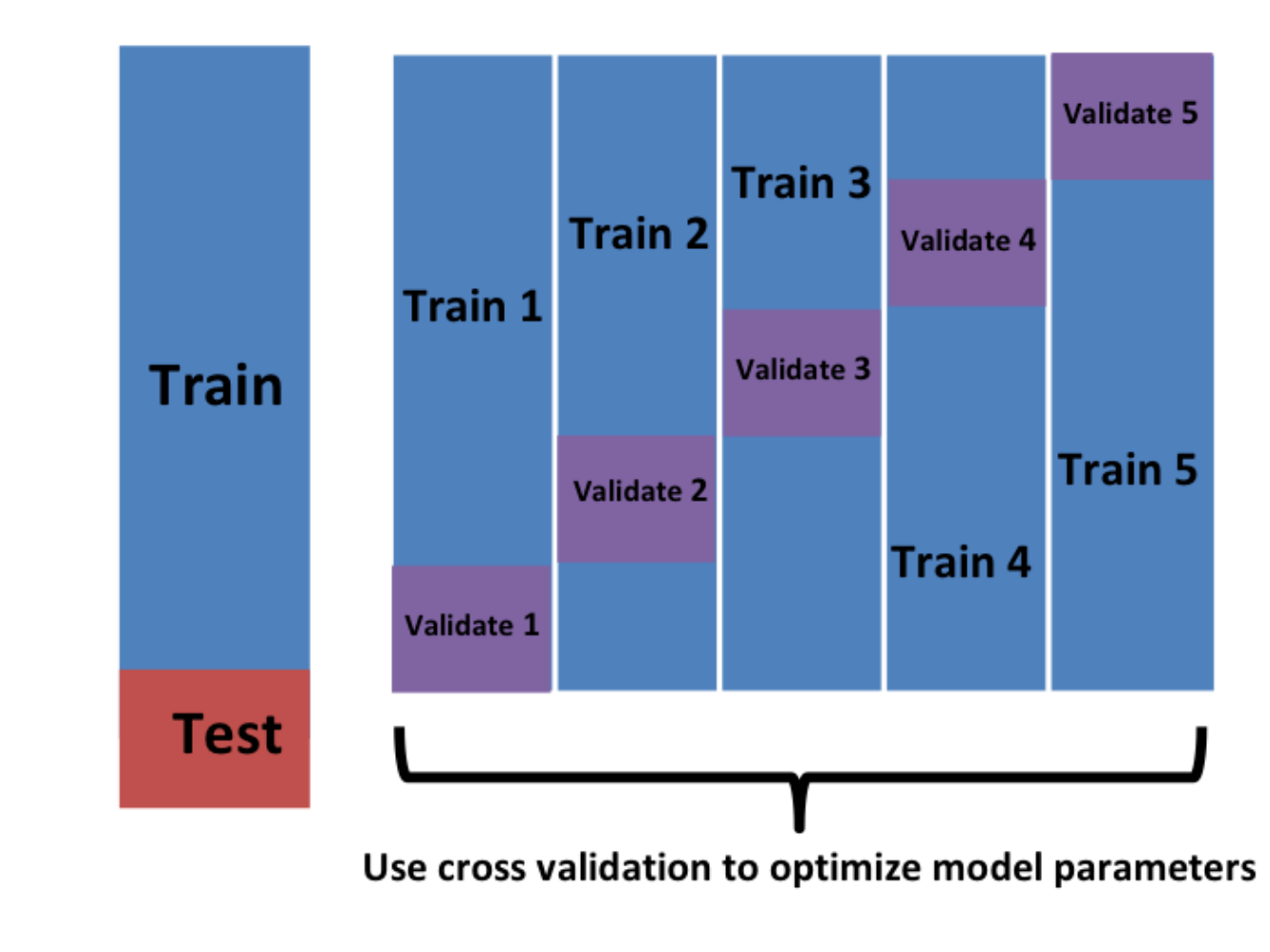
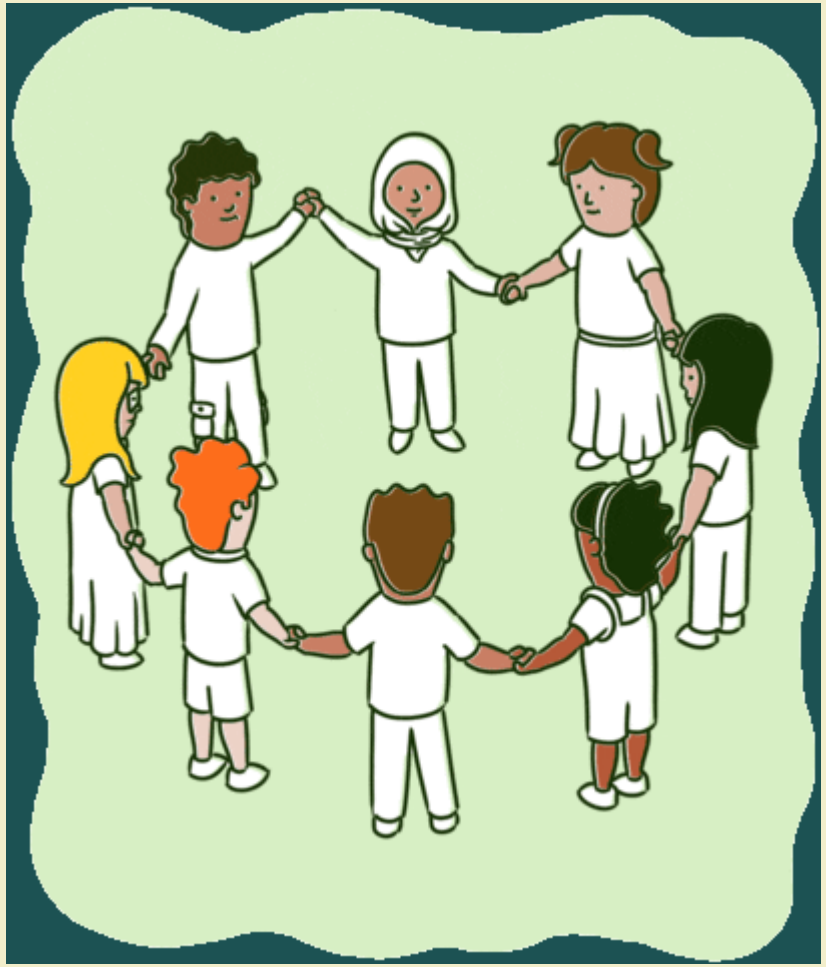


Image source: rafalab.github.io/dsbook/

GROUP ACTIVITY 1

05:00



- Get the class activity 23.Rmd file from moodle
- Let's work on group activity 1 together

Let's see further example of classification using simple logistic regression!

- **Binary response, Y, with an explanatory (predictor, features) variables, X_1.**
- **We model the probability that Y belongs to a particular category.**

$$P(Y = 1) = \frac{e^{\beta_0 + \beta_1 X_1}}{1 + e^{\beta_0 + \beta_1 X_1}}$$

$$\text{Odds} = \frac{P(Y = 1)}{1 - P(Y = 1)} = e^{\beta_0 + \beta_1 X_1}$$

$$\text{Log Odds} = \beta_0 + \beta_1 X_1$$

Tidy the Summary

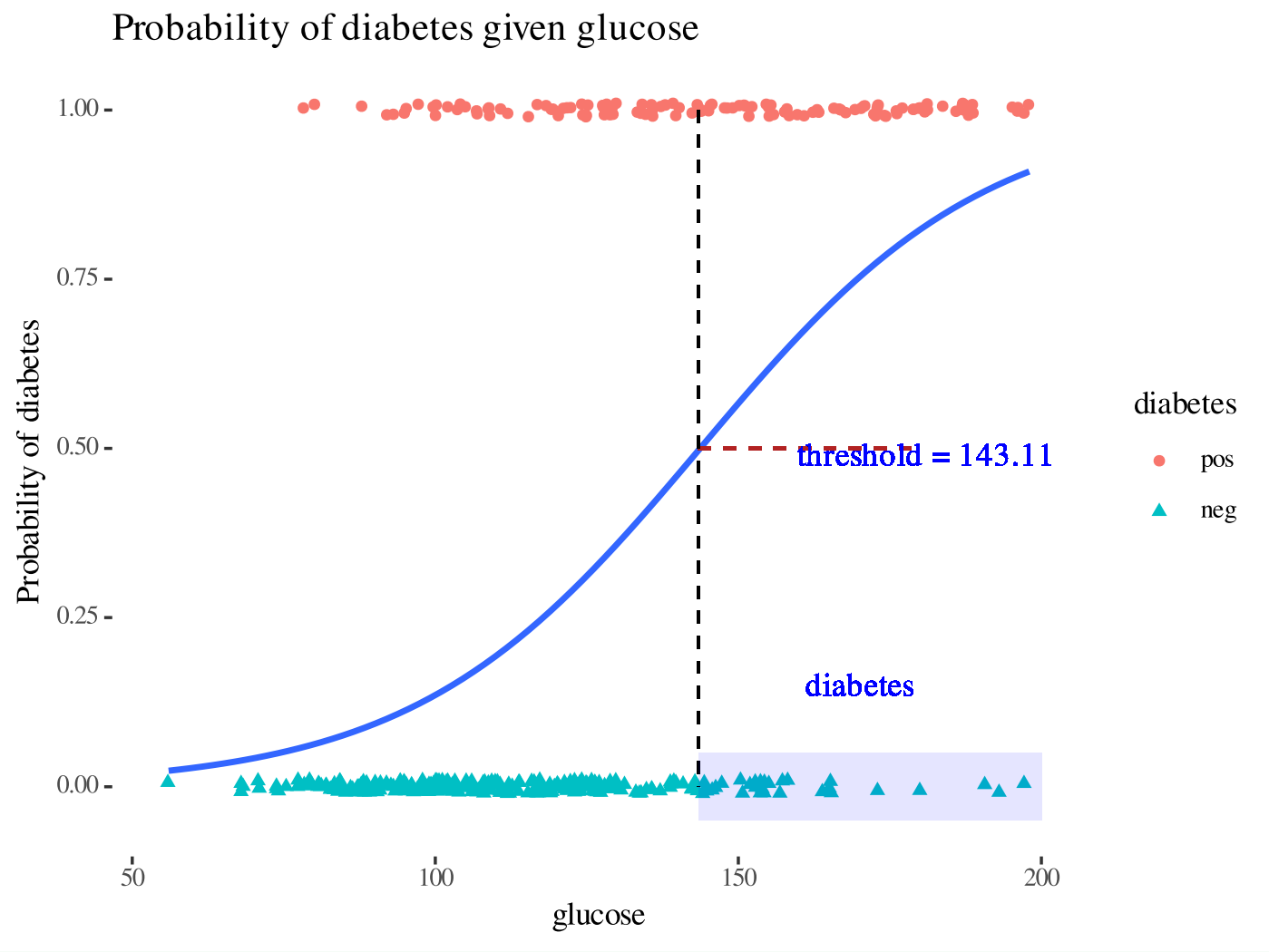
```
broom::tidy(fitted_logistic_model)
# A tibble: 2 × 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)  -5.61     0.678       -8.28 1.20e-16
2 glucose       0.0392   0.00514      7.62 2.55e-14
```

Odds Ratio

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

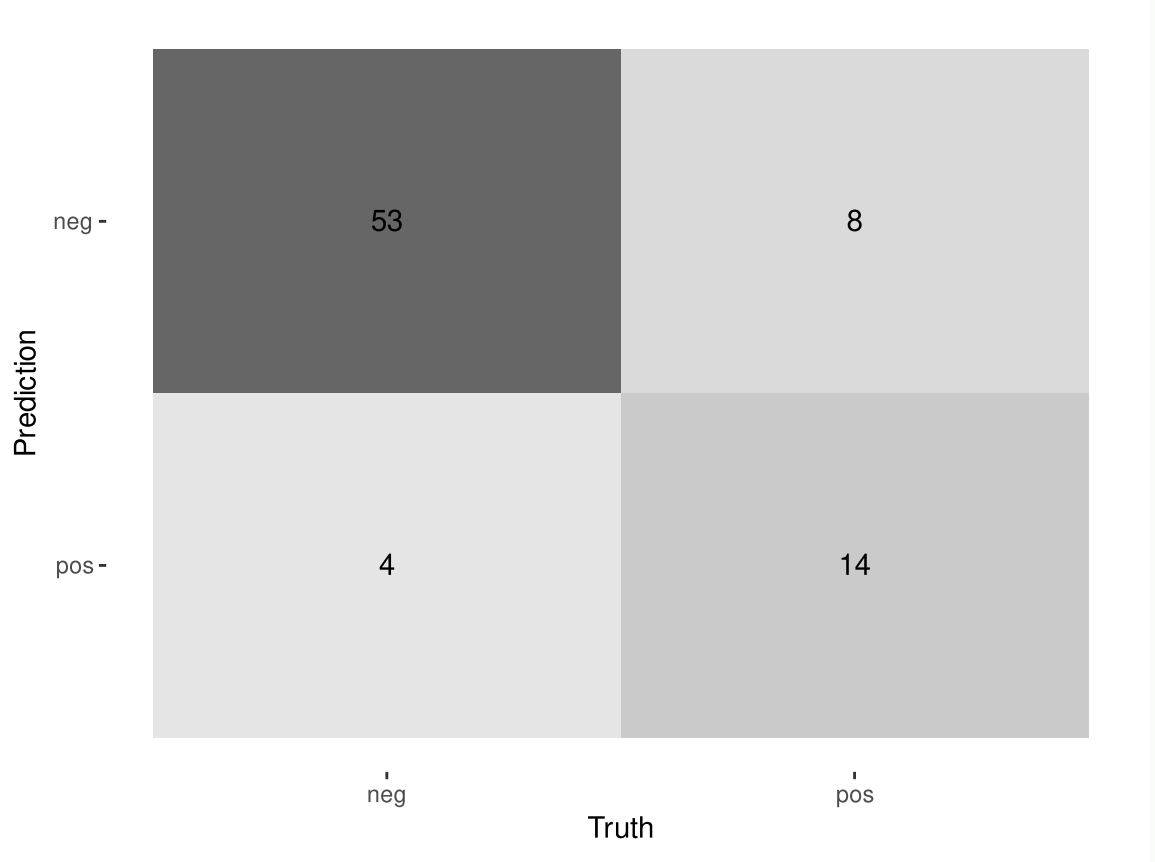
```
broom::tidy(fitted_logistic_model, exponentiate = TRUE)
# A tibble: 2 × 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)  0.00364   0.678       -8.28 1.20e-16
2 glucose      1.04      0.00514      7.62 2.55e-14
```

Threshold for classification



Class Prediction

```
set.seed(12345)
pred_class <- predict(fitted_logistic_model,  new_data = db_test)
bind_cols(db_test %>% select(diabetes), pred_class) %>%
  conf_mat(diabetes, .pred_class) %>% # confusion matrix
  autoplot(type = "heatmap") # with graphics
```



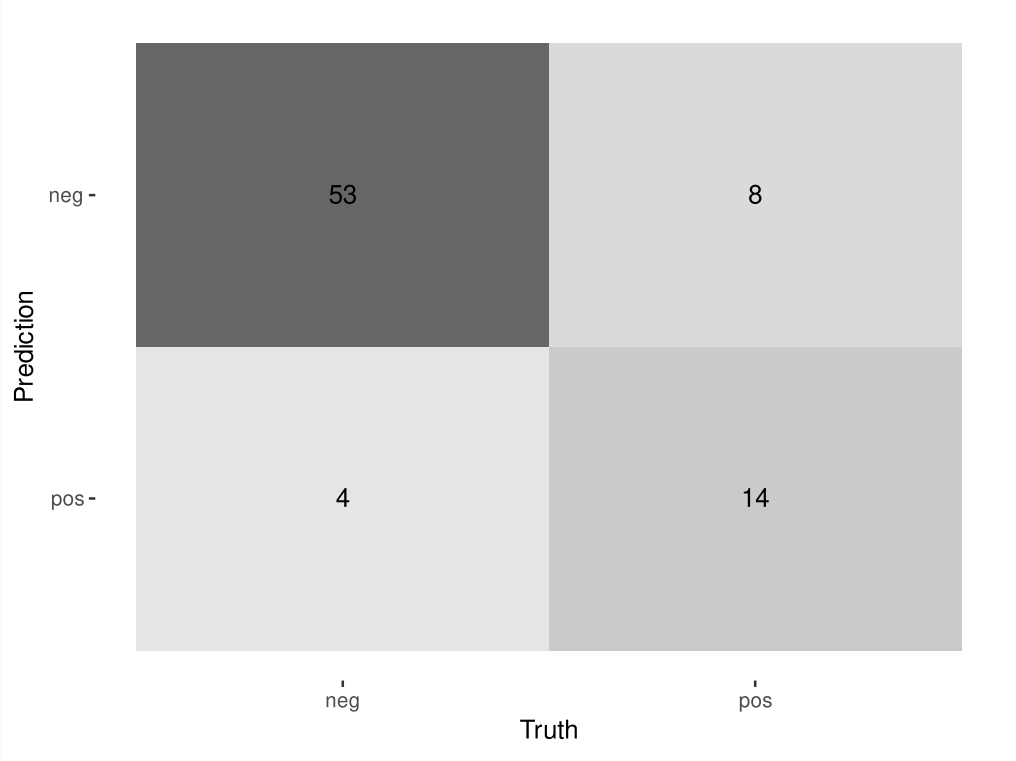
Class Probabilities with `threshold = 0.50`

```
# 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 = .5)) %>%
  select(diabetes, glucose, contains(".pred"))
```

```
    diabetes glucose .pred_neg  .pred_pos .pred_class
25       pos     143 0.5040090 0.49599103          neg
52       neg     101 0.8402912 0.15970881          neg
60       neg     105 0.8181391 0.18186086          neg
64       neg     141 0.5235673 0.47643271          neg
69       neg      95 0.8693592 0.13064080          neg
83       neg      83 0.9141287 0.08587128          neg
98       neg      71 0.9445349 0.05546507          neg
110      pos      95 0.8693592 0.13064080          neg
135      neg      96 0.8648480 0.13515203          neg
143      neg     108 0.8000067 0.19999330          neg
```

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         0.848
2 sens     binary         0.930
3 spec     binary         0.636
4 ppv      binary         0.869
```

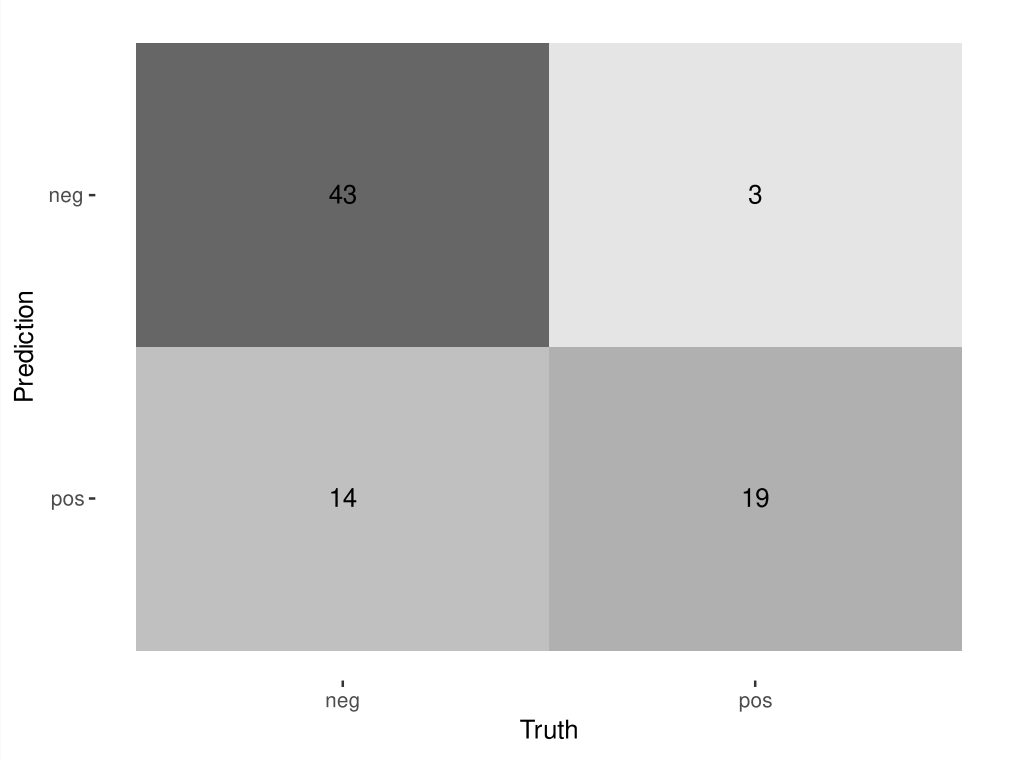
Class Probabilities with `threshold = 0.70`

```
# 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 = .70)) %>%
  select(diabetes, glucose, contains(".pred"))
```

```
    diabetes glucose .pred_neg  .pred_pos .pred_class
25       pos     143 0.5040090 0.49599103          pos
52       neg     101 0.8402912 0.15970881          neg
60       neg     105 0.8181391 0.18186086          neg
64       neg     141 0.5235673 0.47643271          pos
69       neg      95 0.8693592 0.13064080          neg
83       neg      83 0.9141287 0.08587128          neg
98       neg      71 0.9445349 0.05546507          neg
110      pos      95 0.8693592 0.13064080          neg
135      neg      96 0.8648480 0.13515203          neg
143      neg     108 0.8000067 0.19999330          neg
```

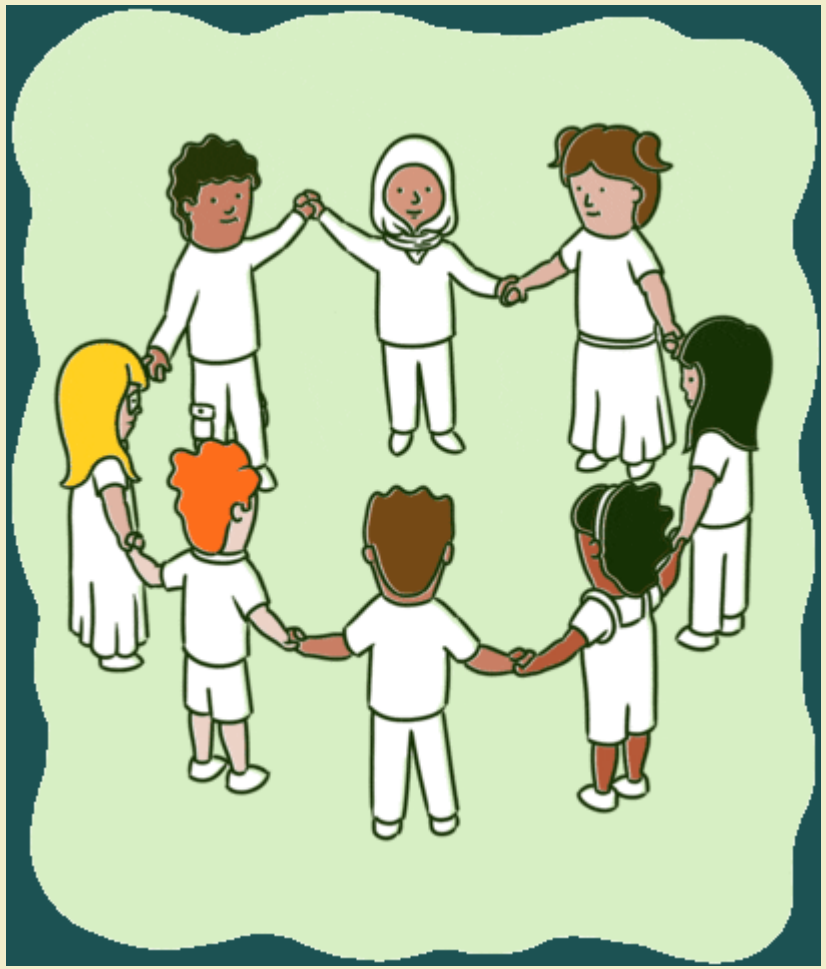
Custom Metrics




```
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         0.785
2 sens     binary         0.754
3 spec     binary         0.864
4 ppv      binary         0.935
```

GROUP ACTIVITY 2

10:00



- Please continue working on group activity 2