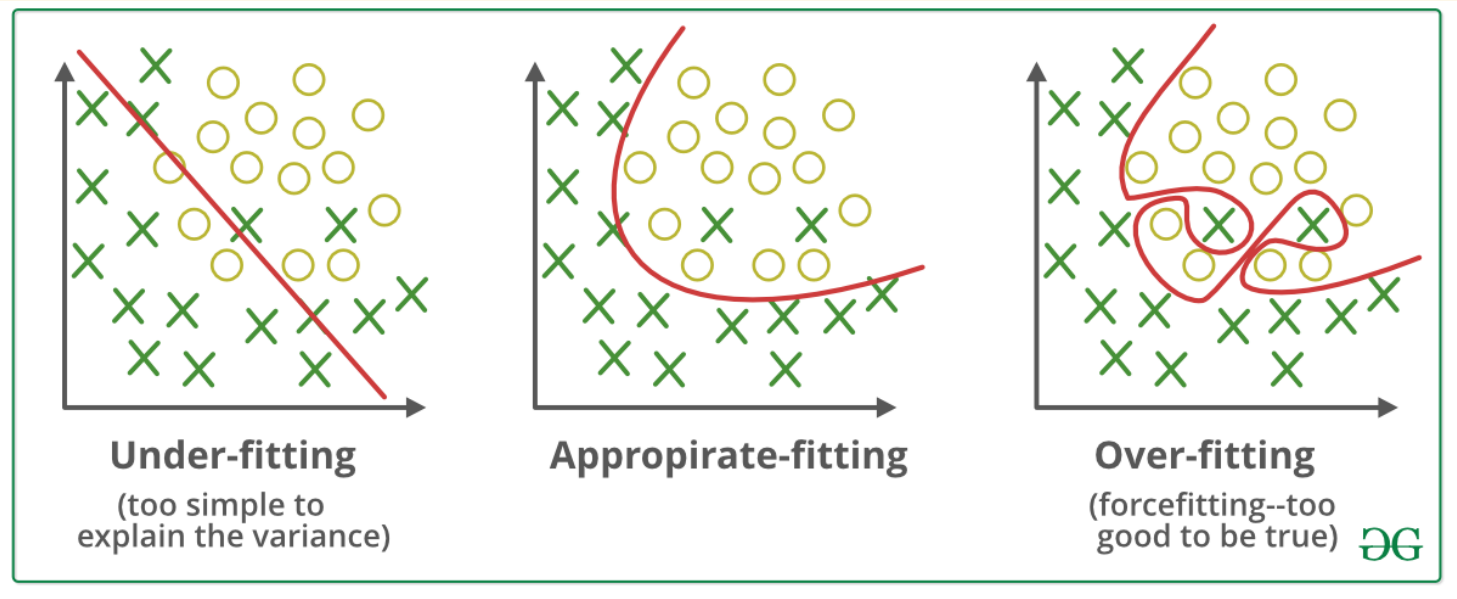
Cross Validation and Logistic Regression

Fall 2022

November 07 2022

Overfitting and underfitting



- Overfitting: Good performance on the training data, poor generalization to other data.
- Underfitting: Poor performance on the training data and poor generalization to other data

Tuning

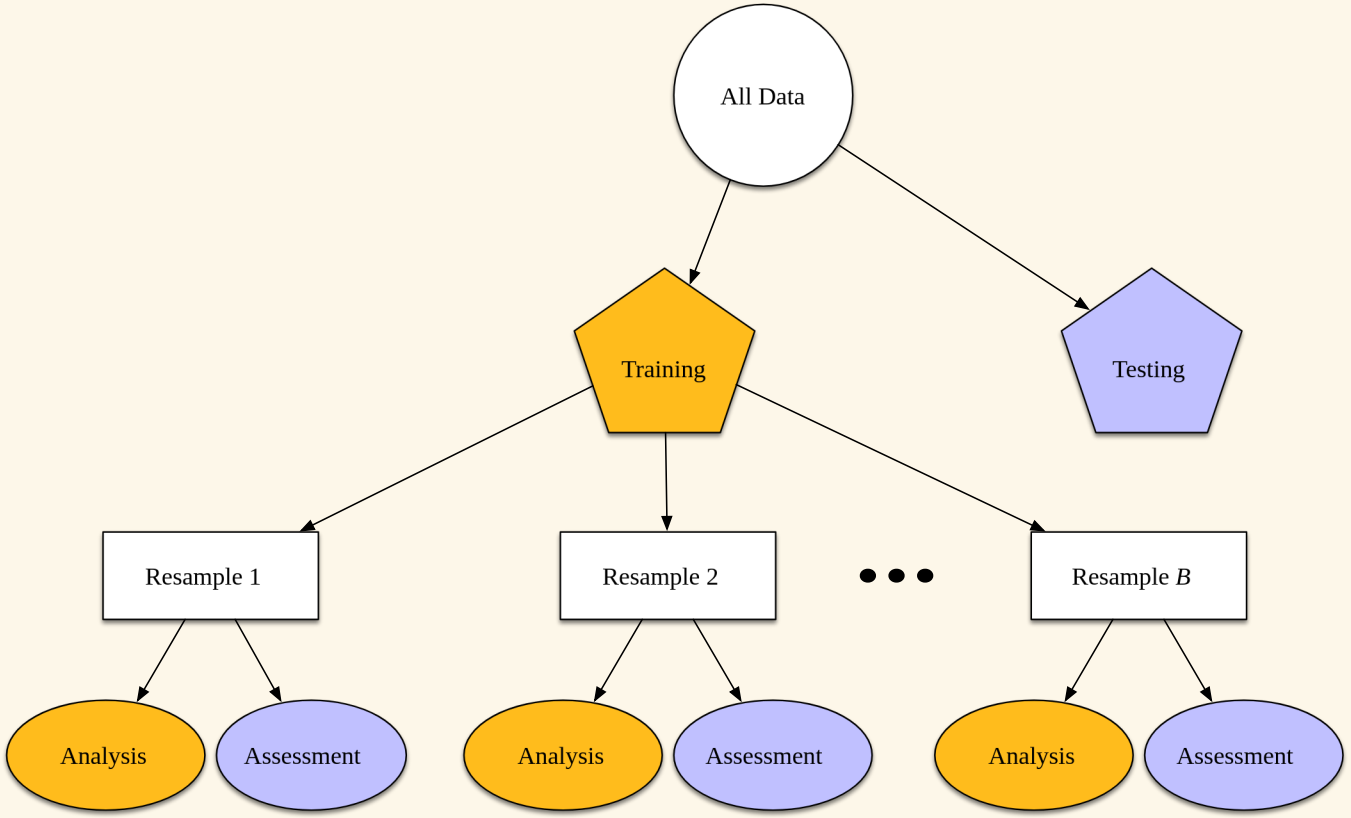
Usually a trial-and-error process by which you

- change some model parameters,
- train the model/algorithm on the data again, then
- compare its performance on a validation set to determine which set of hyper parameters results in the most accurate model.

KNN tuning: find the value of K that creates the best classifier

Don't touch the test data set during model tuning!

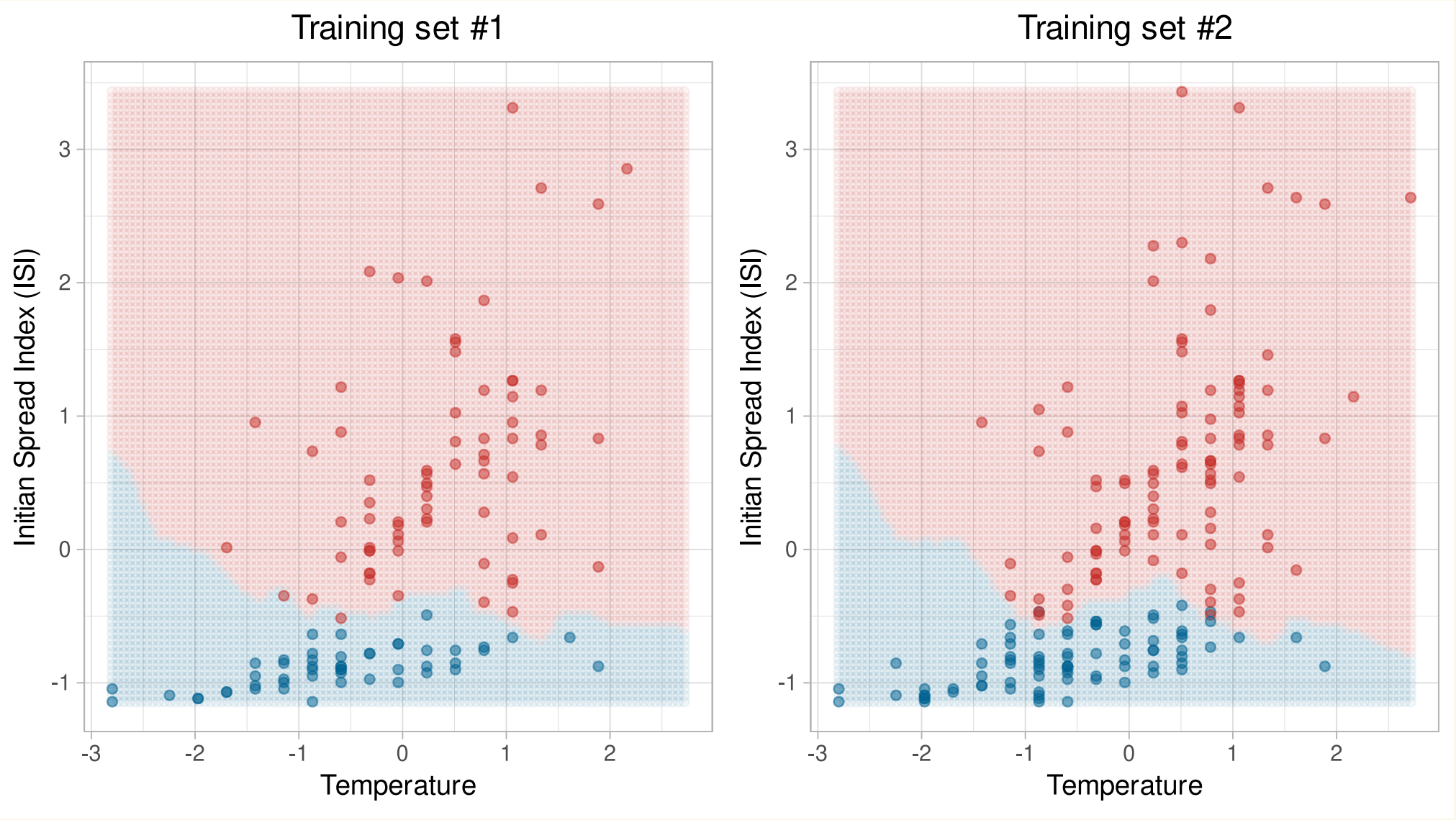
Resampling methods



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

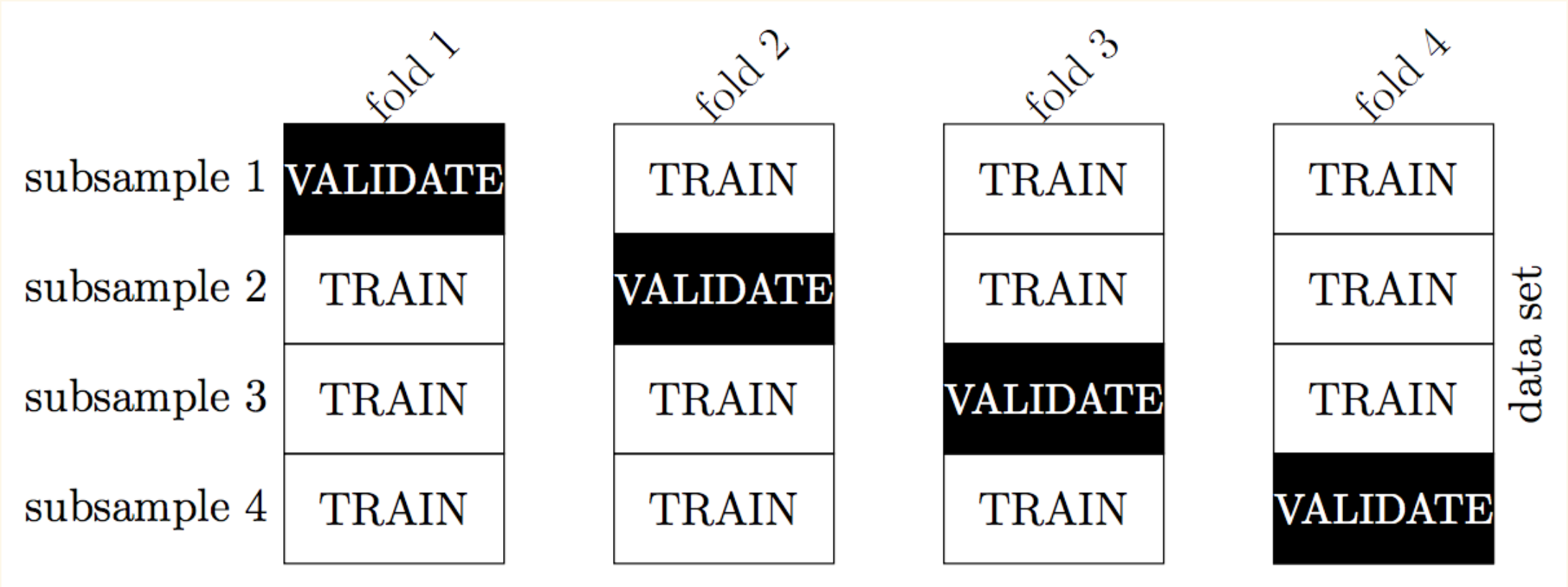
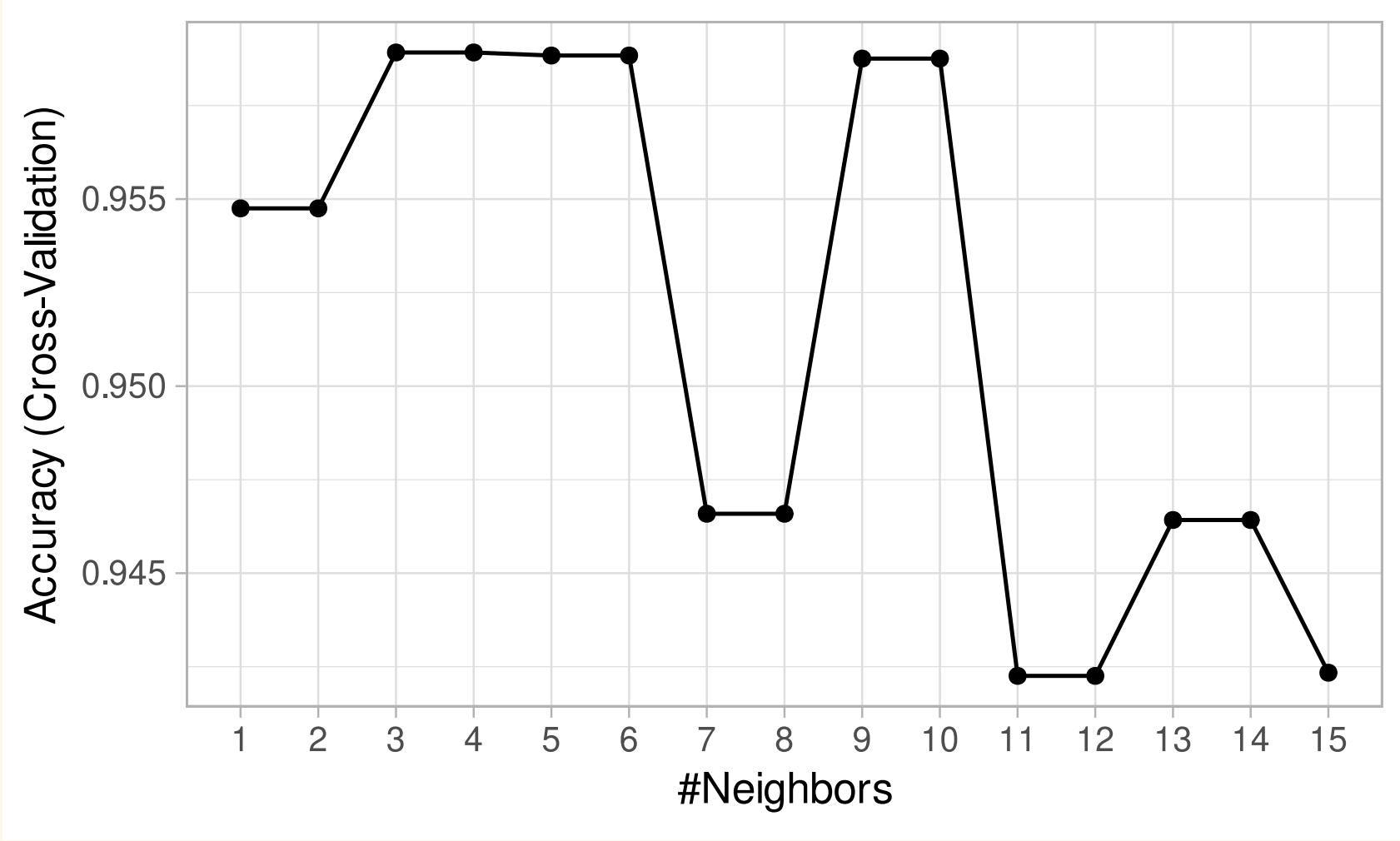


Image courtesy of Dennis Sun

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



- Can look at other metrics
- 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 = 40, by = 2))
```

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),
            control = control_resamples(save_pred = TRUE))
```

Choosing K

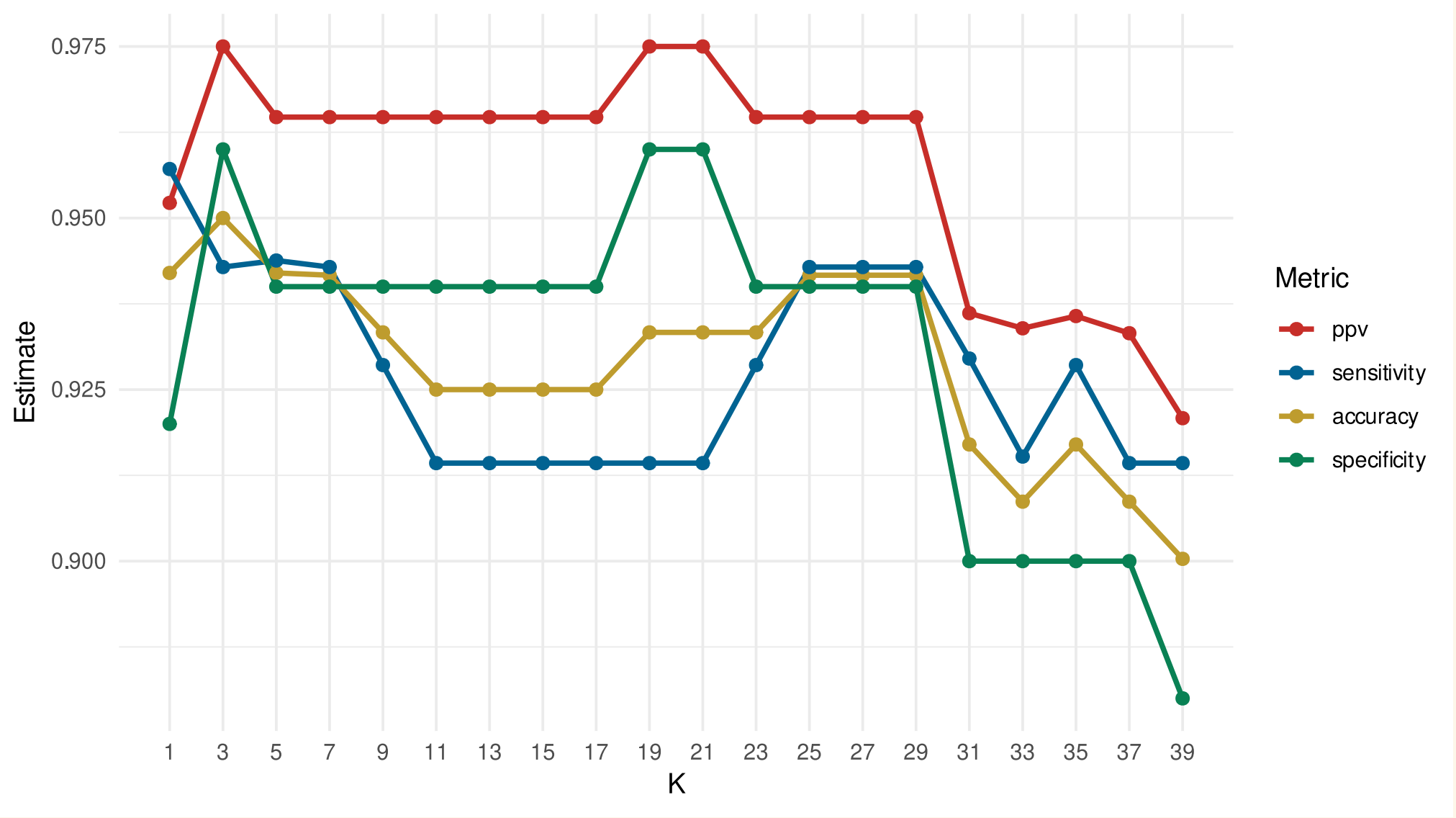
Collect the performance metrics and find the best model

```
cv_metrics <- collect_metrics(knn_fit)
cv_metrics %>% head(10)
# A tibble: 10 × 7
   neighbors .metric     .estimator  mean     n std_err .config
       <dbl> <chr>       <chr>      <dbl> <int>   <dbl> <chr>
 1         1 accuracy    binary     0.942     5  0.0283 Preprocessor1_Model01
 2         1 ppv         binary     0.952     5  0.0344 Preprocessor1_Model01
 3         1 sensitivity binary     0.957     5  0.0429 Preprocessor1_Model01
 4         1 specificity binary     0.92      5  0.0583 Preprocessor1_Model01
 5         3 accuracy    binary     0.95      5  0.0243 Preprocessor1_Model02
 6         3 ppv         binary     0.975     5  0.0250 Preprocessor1_Model02
 7         3 sensitivity binary     0.943     5  0.0416 Preprocessor1_Model02
 8         3 specificity binary     0.96      5  0.04   Preprocessor1_Model02
 9         5 accuracy    binary     0.942     5  0.0213 Preprocessor1_Model03
10         5 ppv         binary     0.965     5  0.0353 Preprocessor1_Model03
```

Choosing K

```
cv_metrics %>% group_by(.metric) %>% slice_max(mean)
# A tibble: 8 × 7
# Groups:   .metric [4]
  neighbors .metric     .estimator  mean     n std_err .config
      <dbl> <chr>       <chr>      <dbl> <int>   <dbl> <chr>
1         3 accuracy    binary     0.95      5  0.0243 Preprocessor1_Model02
2         3 ppv         binary     0.975     5  0.0250 Preprocessor1_Model02
3        19 ppv         binary     0.975     5  0.0250 Preprocessor1_Model10
4        21 ppv         binary     0.975     5  0.0250 Preprocessor1_Model11
5         1 sensitivity binary     0.957     5  0.0429 Preprocessor1_Model01
6         3 specificity binary     0.96      5  0.04   Preprocessor1_Model02
7        19 specificity binary     0.96      5  0.04   Preprocessor1_Model10
8        21 specificity binary     0.96      5  0.04   Preprocessor1_Model11
```

Choosing K


0.975
0.950
0.925
0.900
Estimate
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
K
Metric
ppv
sensitivity
accuracy
specificity

The full process

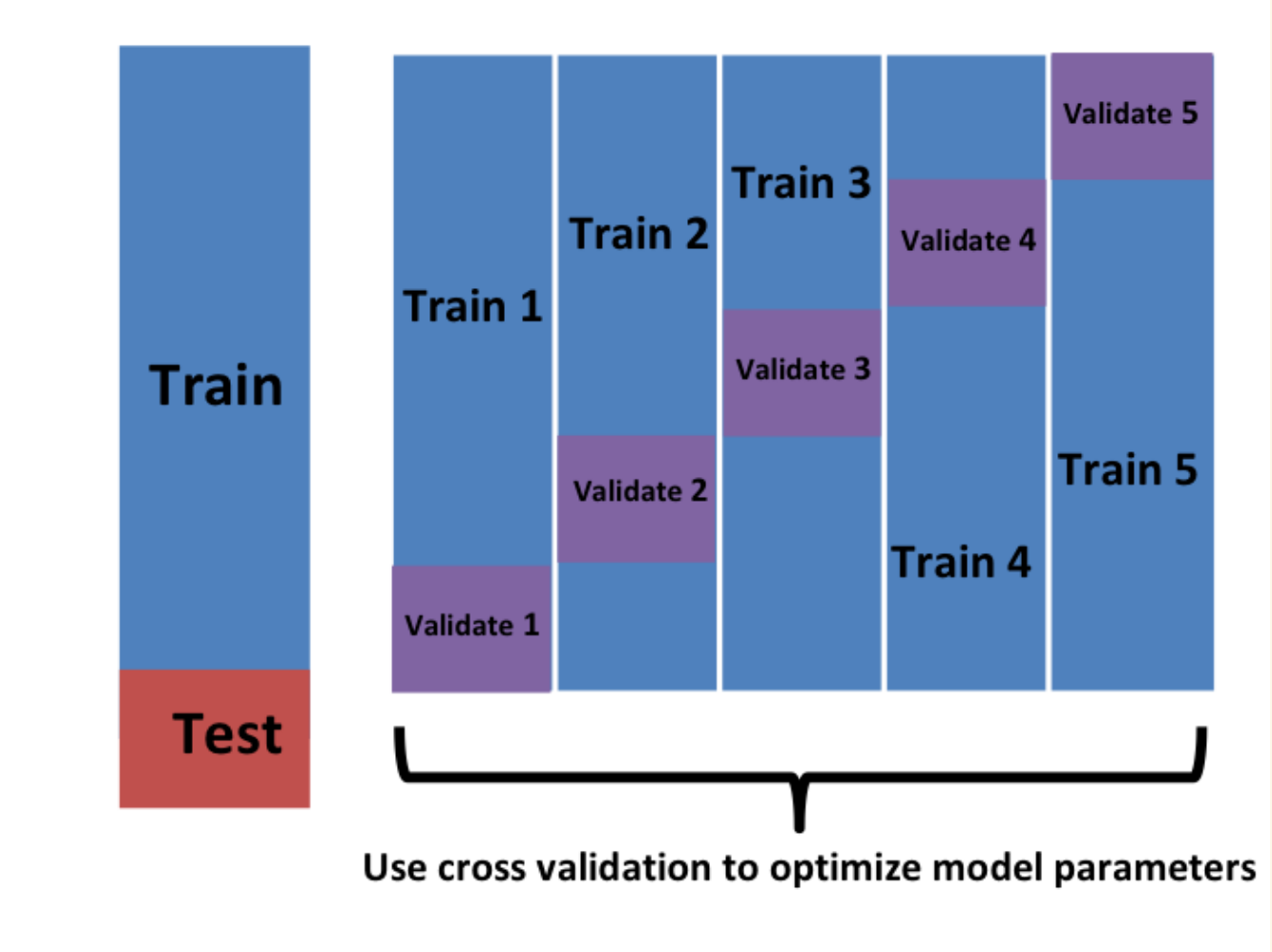
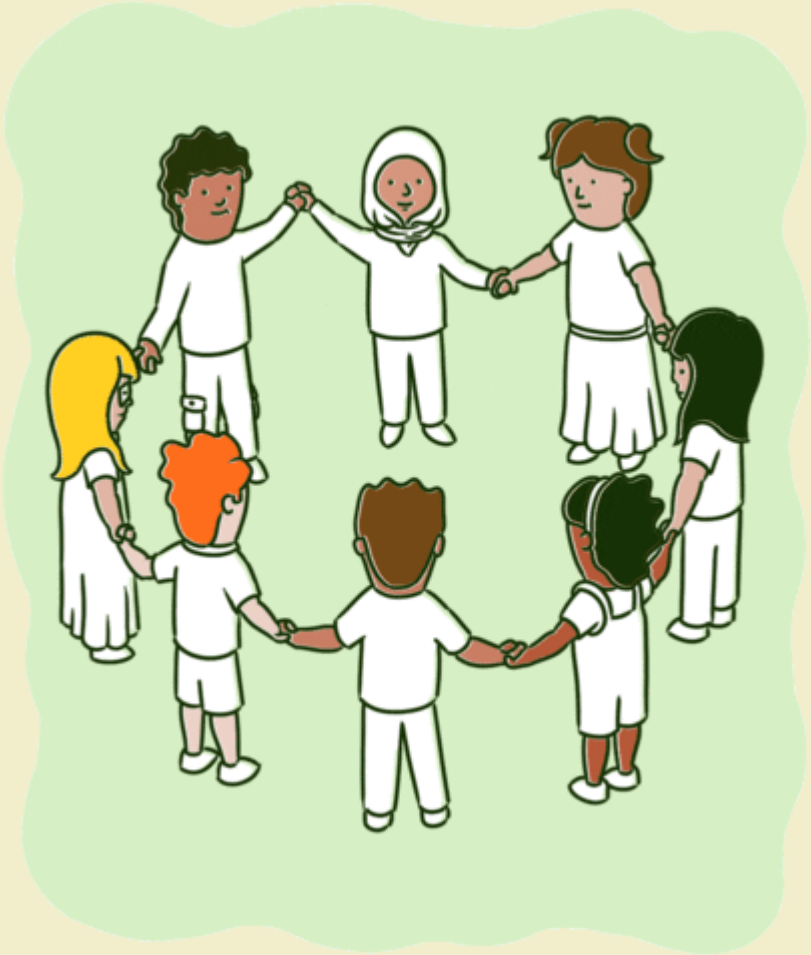


Image source: rafalab.github.io/dsbook/

Group Activity 1

05:00



- Get the class activity 24.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

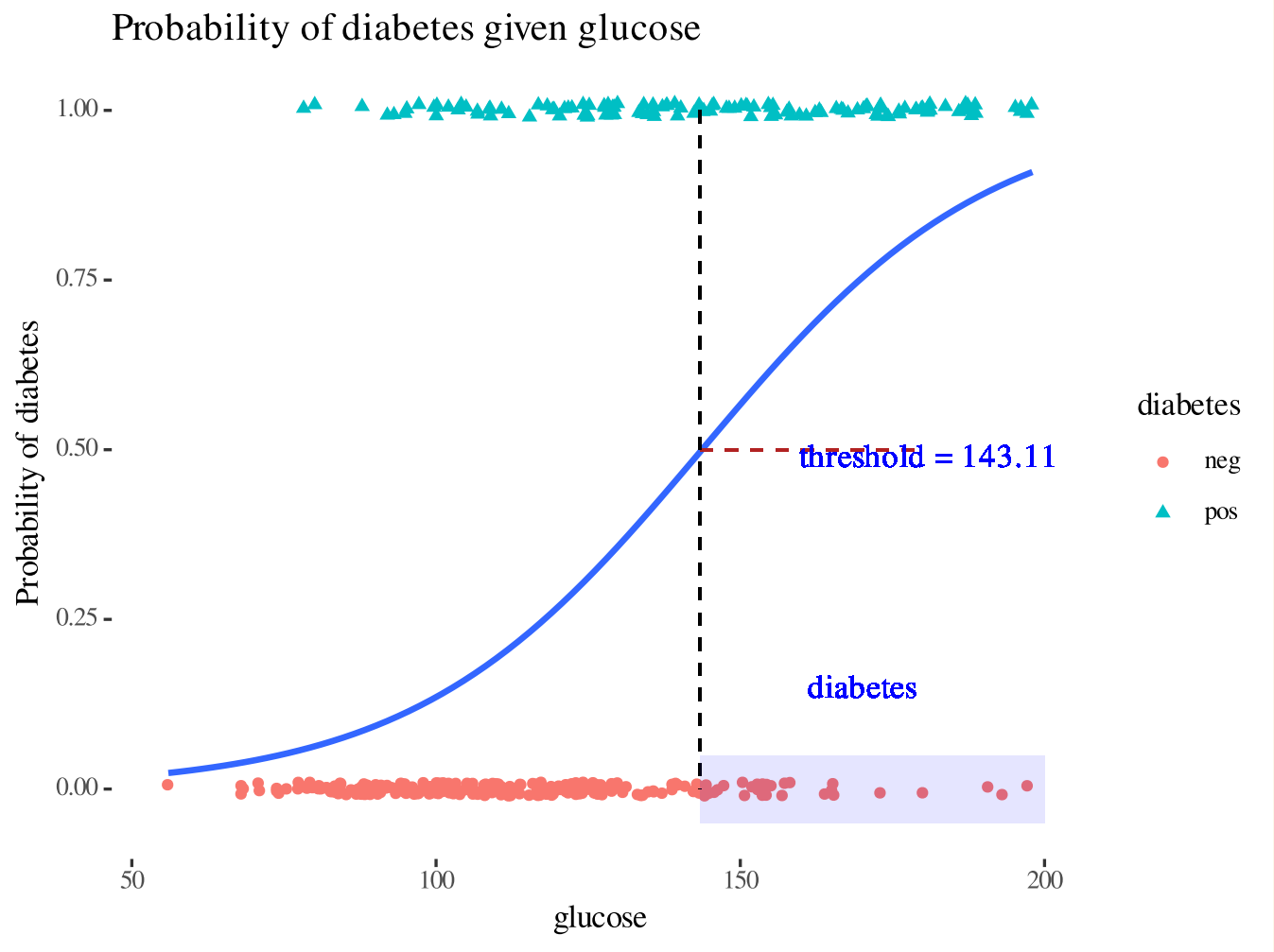
```
tidy(fitted_logistic_model)
# A tibble: 2 × 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)  -5.61     0.678       -8.28 1.20e-16
2 glucose       0.0392   0.00514      7.62 2.55e-14
```

Odds Ratio

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

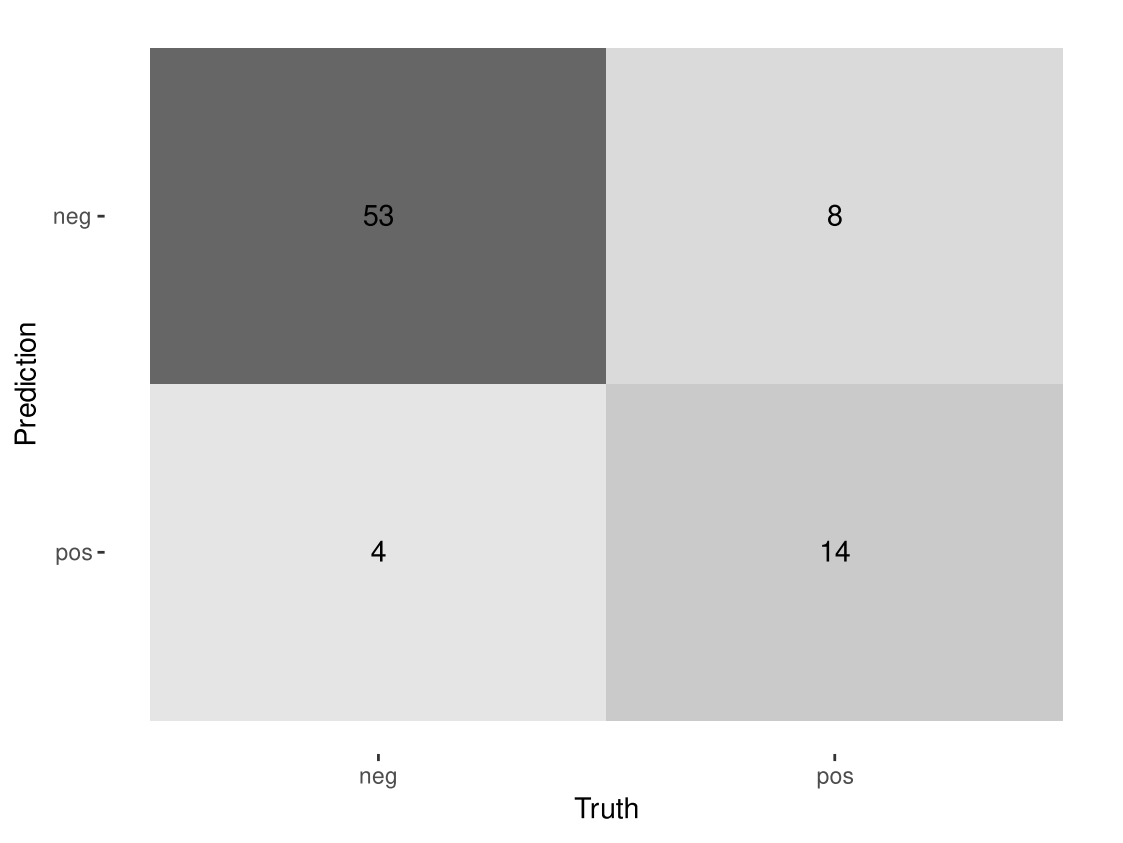
```
tidy(fitted_logistic_model, exponentiate = TRUE)
# A tibble: 2 × 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)  0.00364   0.678        -8.28 1.20e-16
2 glucose      1.04      0.00514       7.62 2.55e-14
```

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
```



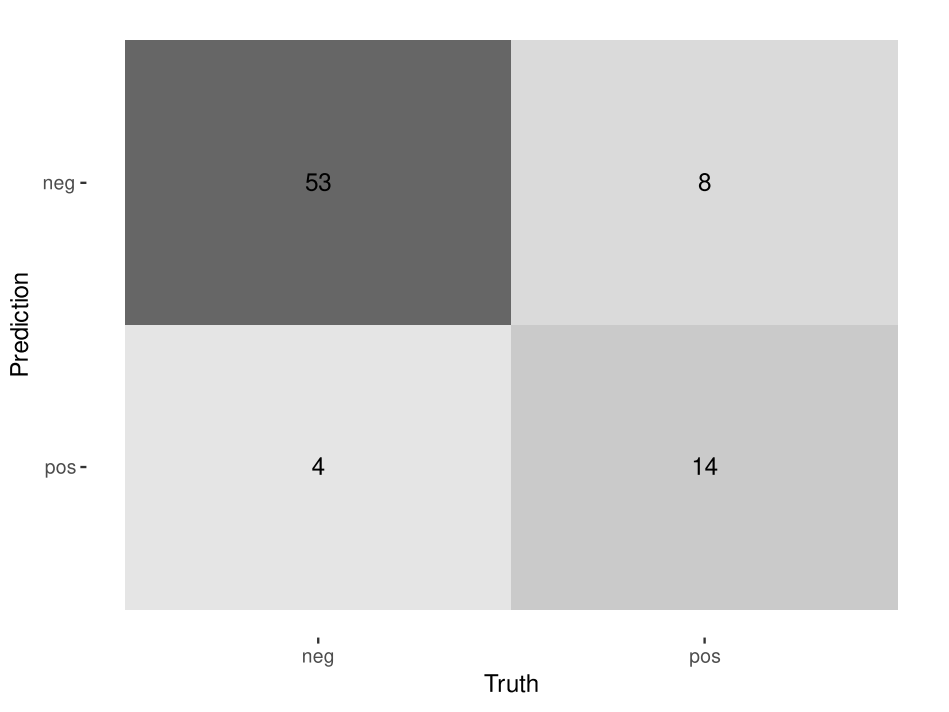
Class Probabilities with `threshold = 0.50`

```
# 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
```

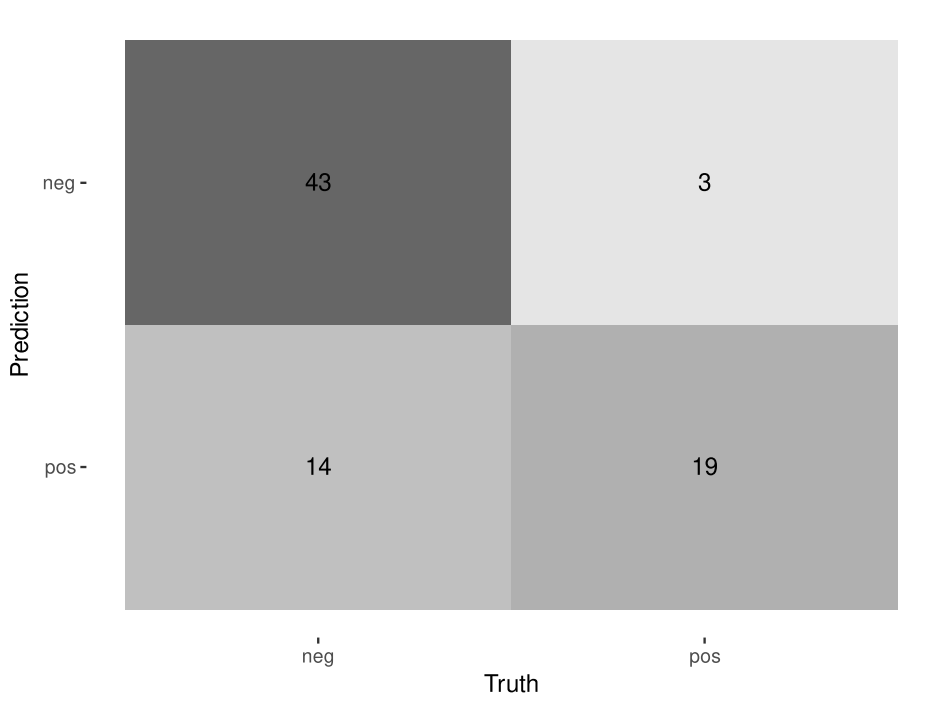
Class Probabilities with `threshold = 0.70`

```
# 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
```

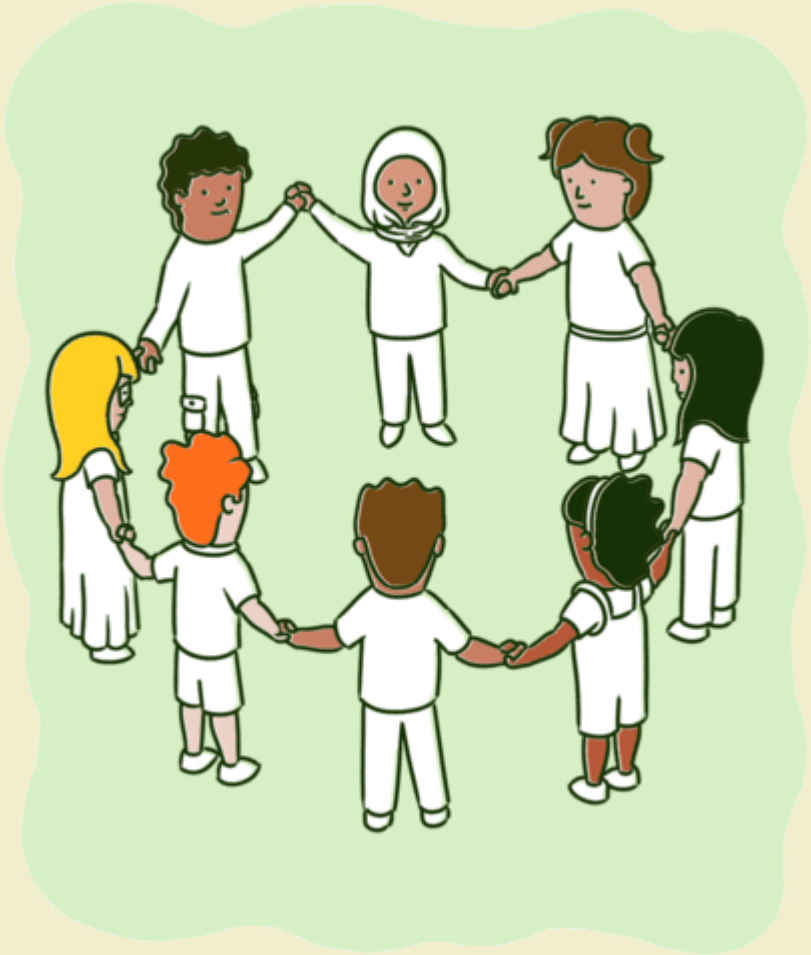
Custom Metrics




```
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