Model Accuracy and Evaluation

Fall 2022

November 03 2022

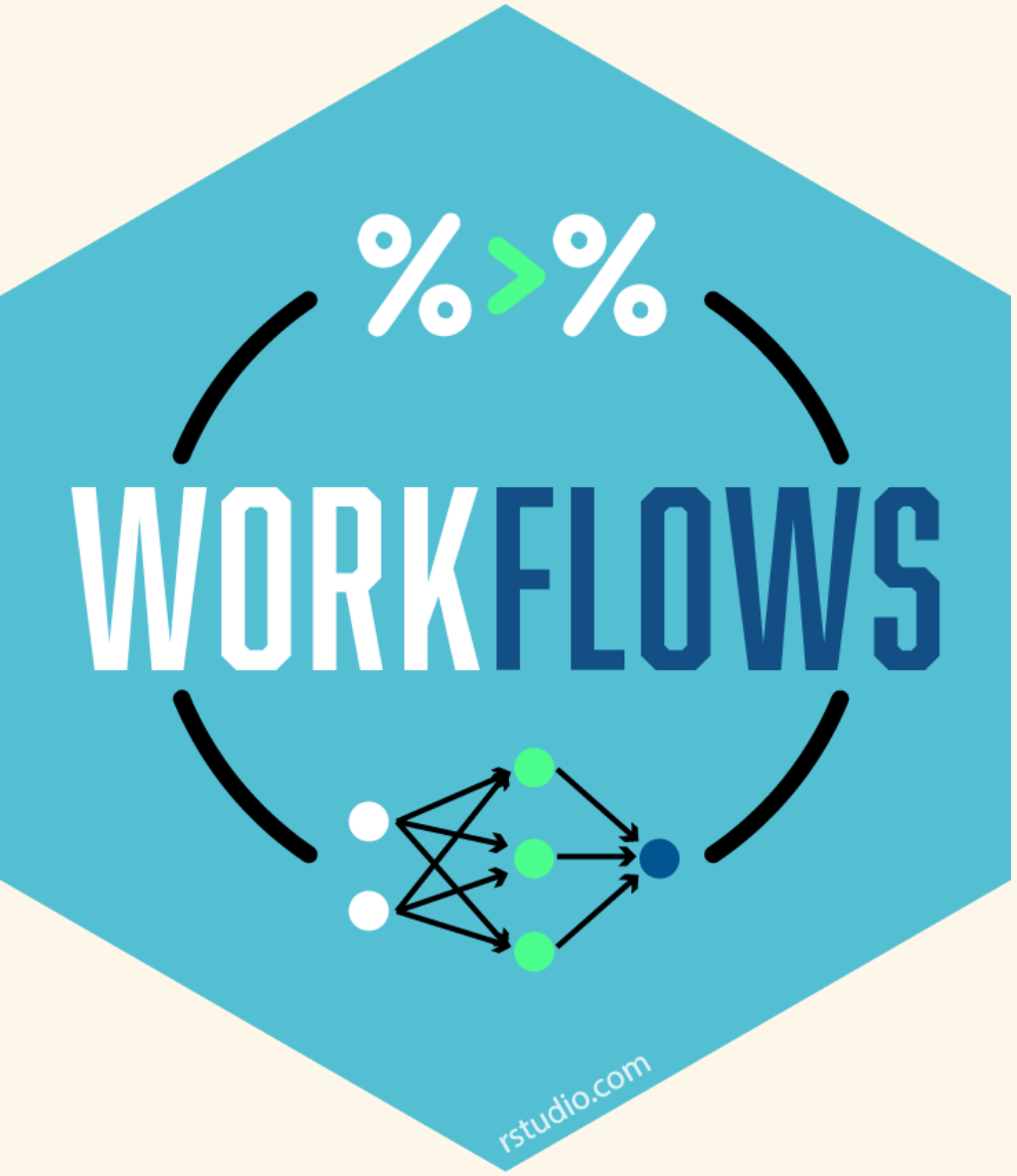
KNN (K- Nearest Neighbor)

- Supervised machine learning algorithm i.e., it requires labeled data for training
- Need to tell the algorithm the exact number of neighbors (K) we want to consider

Training and Testing

Training: Fitting a model with certain hyper-parameters on a particular subset of the dataset

Testing: Test the model on a different subset of the dataset to get an estimate of a final, unbiased assessment of the model's performance



Workflows

A machine learning workflow (the “black box”) containing model specification and preprocessing recipe/formula

1. Create a workflow: Split the raw data

```
set.seed(123) # set seed for reproducibility

fire_split <- initial_split(fire_raw, prop = 0.75)

# Create training data
fire_train <- fire_split %>% training()


# Create testing data
fire_test <- fire_split %>% testing()
```

2. Make a recipe

```
fire_recipe <- recipe(classes ~ ., data = fire_raw) %>%
  step_scale(all_predictors()) %>%
  step_center(all_predictors()) %>%
  prep()
```

3. Specify the model

```
fire_knn_spec <- nearest_neighbor(mode = "classification",
                                  engine = "kknn",
                                  weight_func = "rectangular",
                                  neighbors = 5)
```

4. Define the workflow object

```
fire_workflow <- workflow() %>%
  add_recipe(fire_recipe) %>%
  add_model(fire_knn_spec)
```

5. Fit the model

```
fire_fit <- fit(fire_workflow, data = fire_train)
```

```
══ Workflow [trained] ══════════════════════════════════════════════════════
Preprocessor: Recipe
Model: nearest_neighbor()

── Preprocessor ────────────────────────────────────────────────────────────
2 Recipe Steps

• step_scale()
• step_center()

── Model ───────────────────────────────────────────────────────────────────

Call:
kknn::train.kknn(formula = ..y ~ ., data = data, ks = min_rows(5,

Type of response variable: nominal
Minimal misclassification: 0.03296703
Best kernel: rectangular
Best k: 5
```

6. Evaluate the model on test dataset

```
test_features <- fire_test %>% select(temperature, isi) %>%
  data.frame()
fire_pred <- predict(fire_fit, test_features, type = "raw")
fire_results <- fire_test %>%
  select(classes) %>%
  bind_cols(predicted = fire_pred)
```

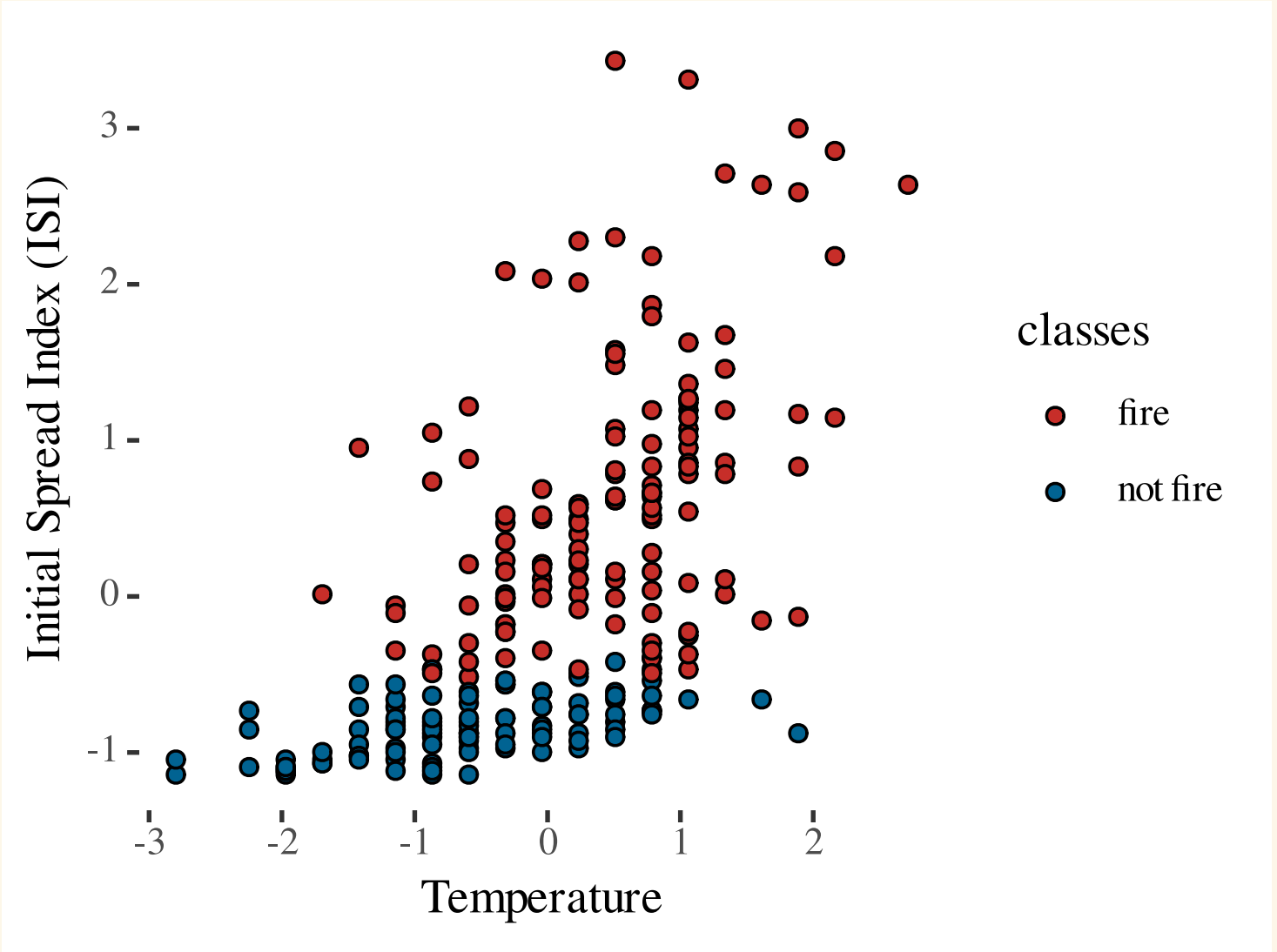
7. Compare the known labels and predicted labels

```
knitr::kable(fire_results)
```

classes	predicted
not fire	not fire
not fire	not fire
fire	fire
not fire	not fire
not fire	not fire
not fire	not fire
fire	fire
fire	fire
not fire	not fire
not fire	not fire
not fire	fire
fire	fire
fire	fire

How do we choose the number of neighbors in a principled way?

Scatter-plot Code Take Away

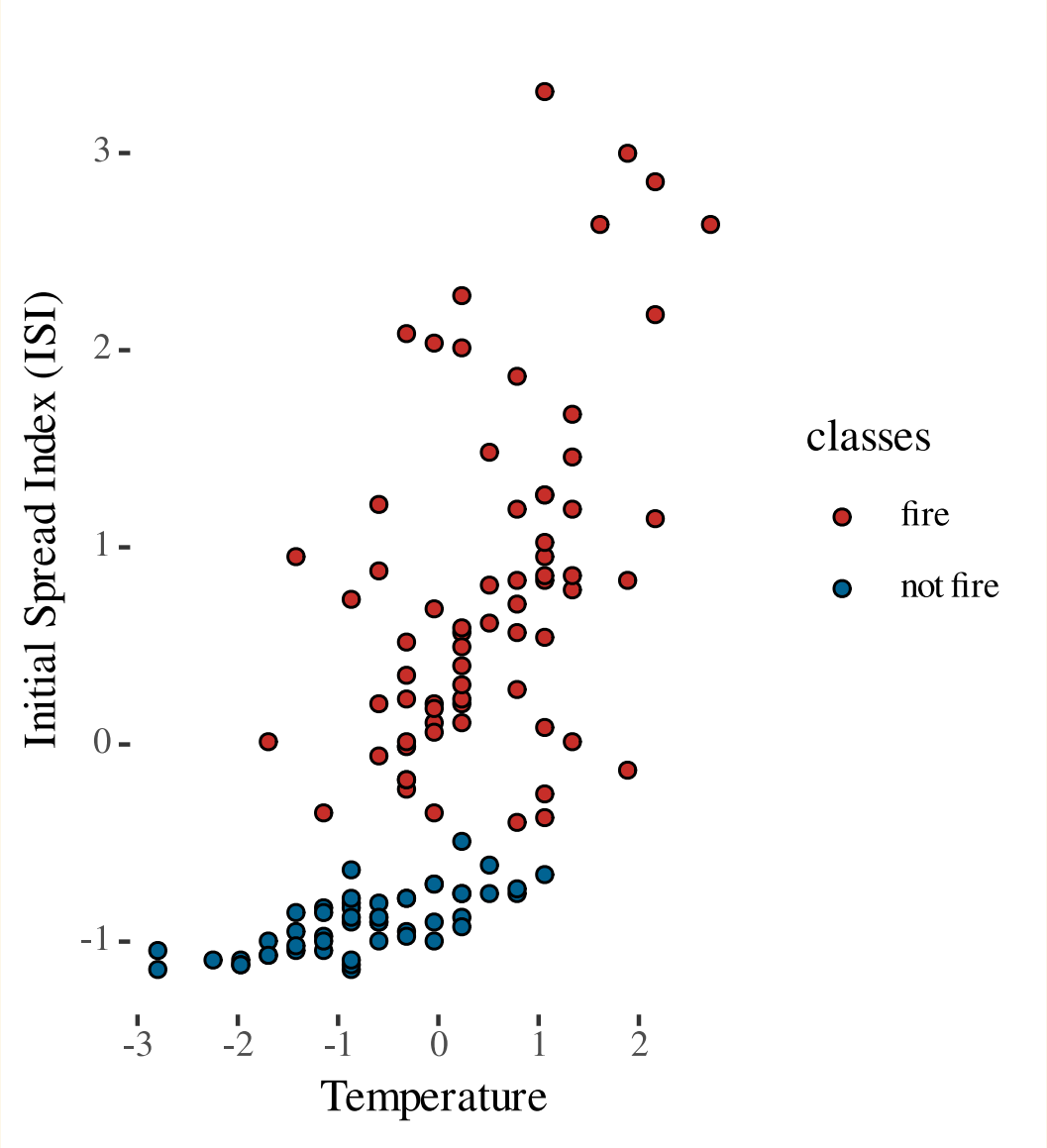


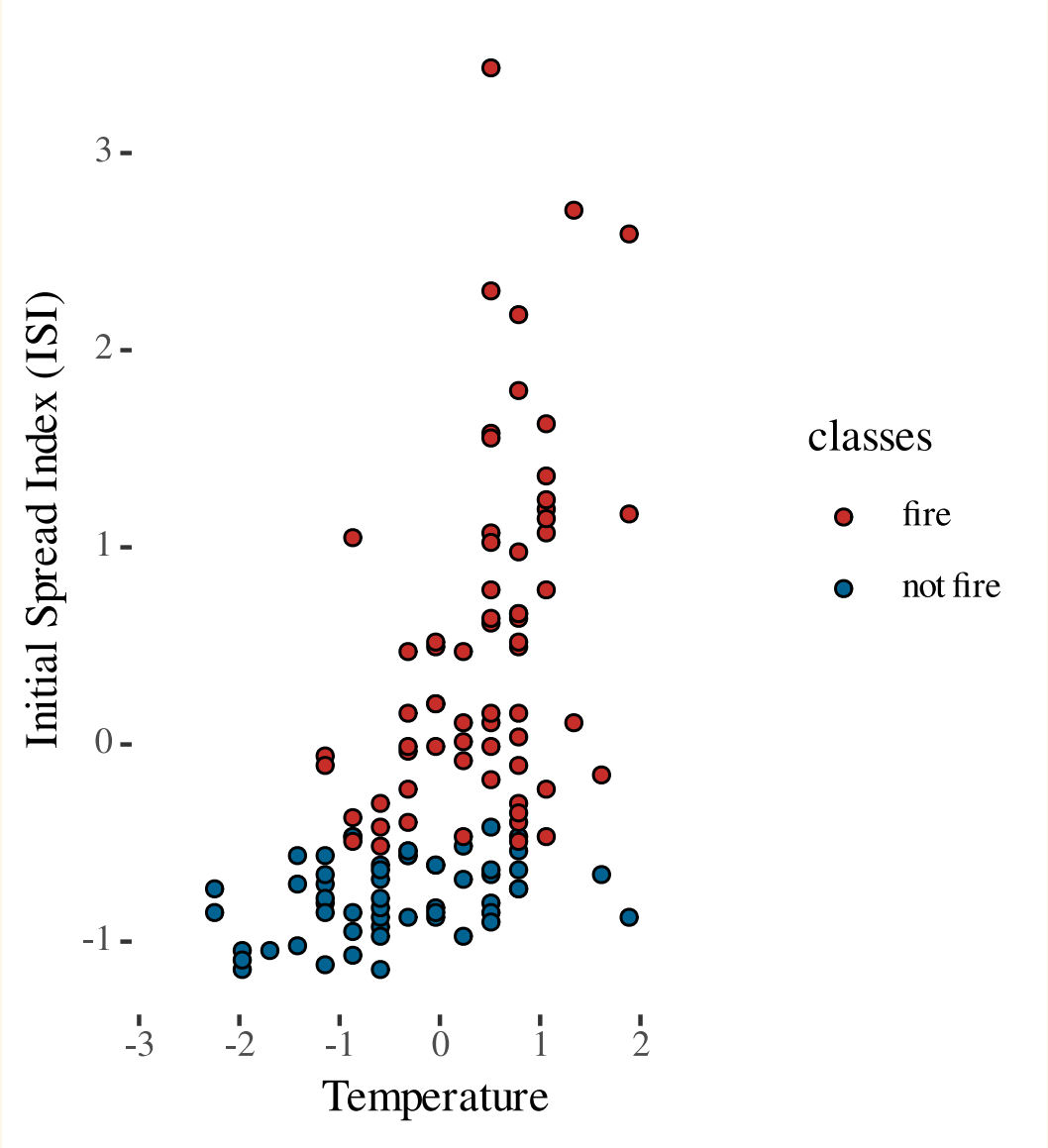
Evaluating accuracy

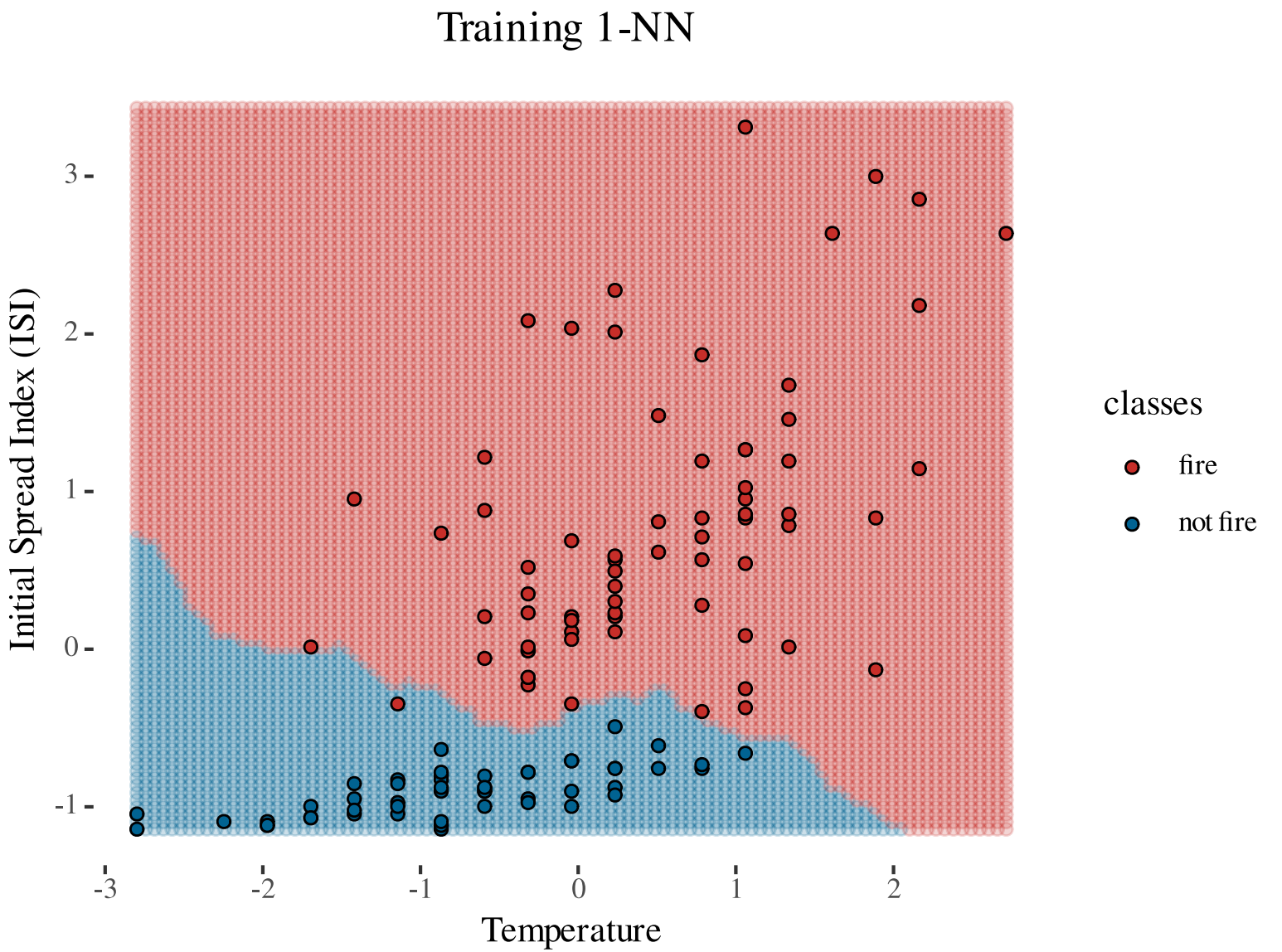
We want to evaluate classifiers based on some accuracy metrics.

- Randomly split data set into two pieces: training set and test set
- Train (i.e. fit) KNN on the training set
- Make predictions on the test set
- See how good those predictions are

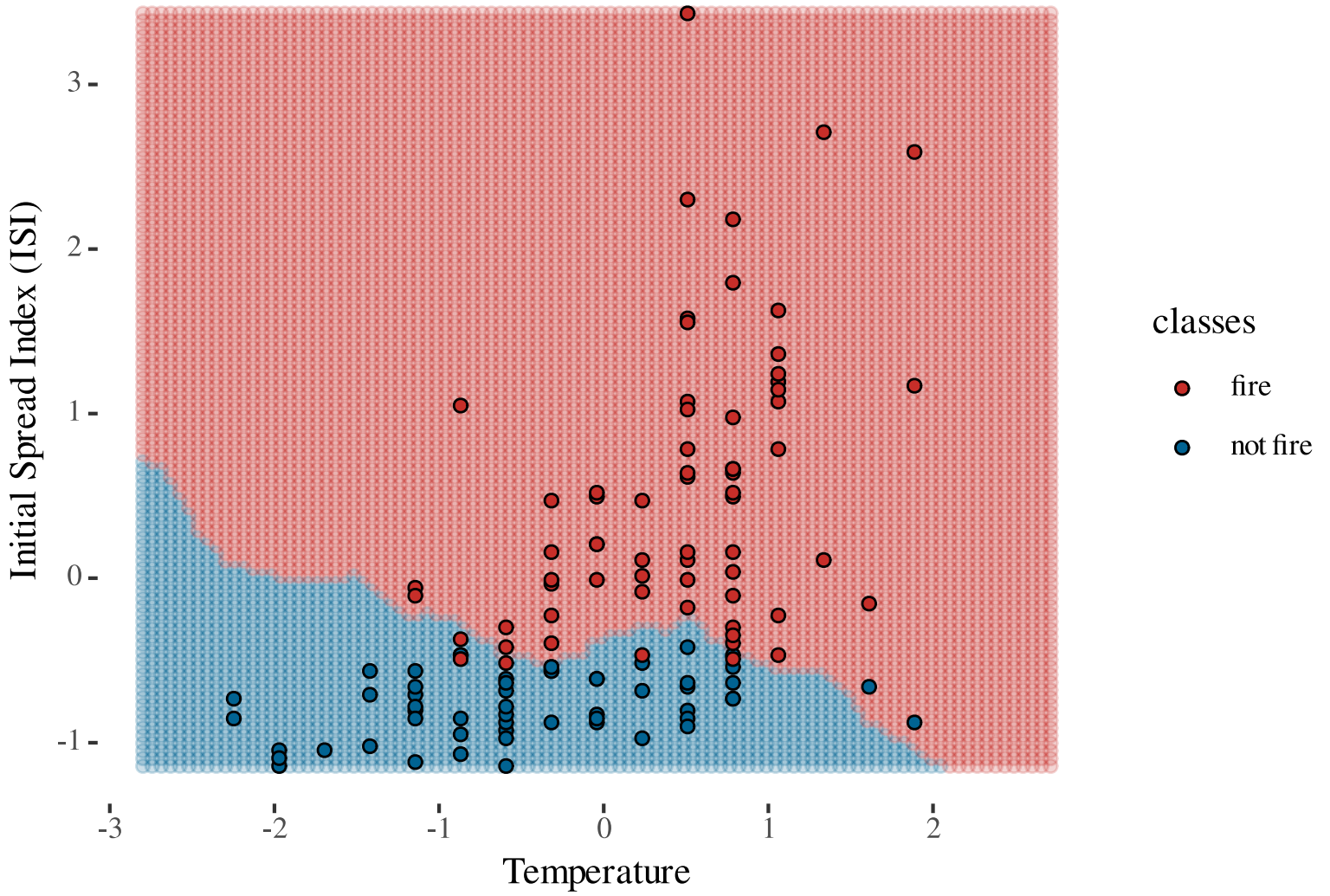
Train (left) and test (right) dataset (50-50)

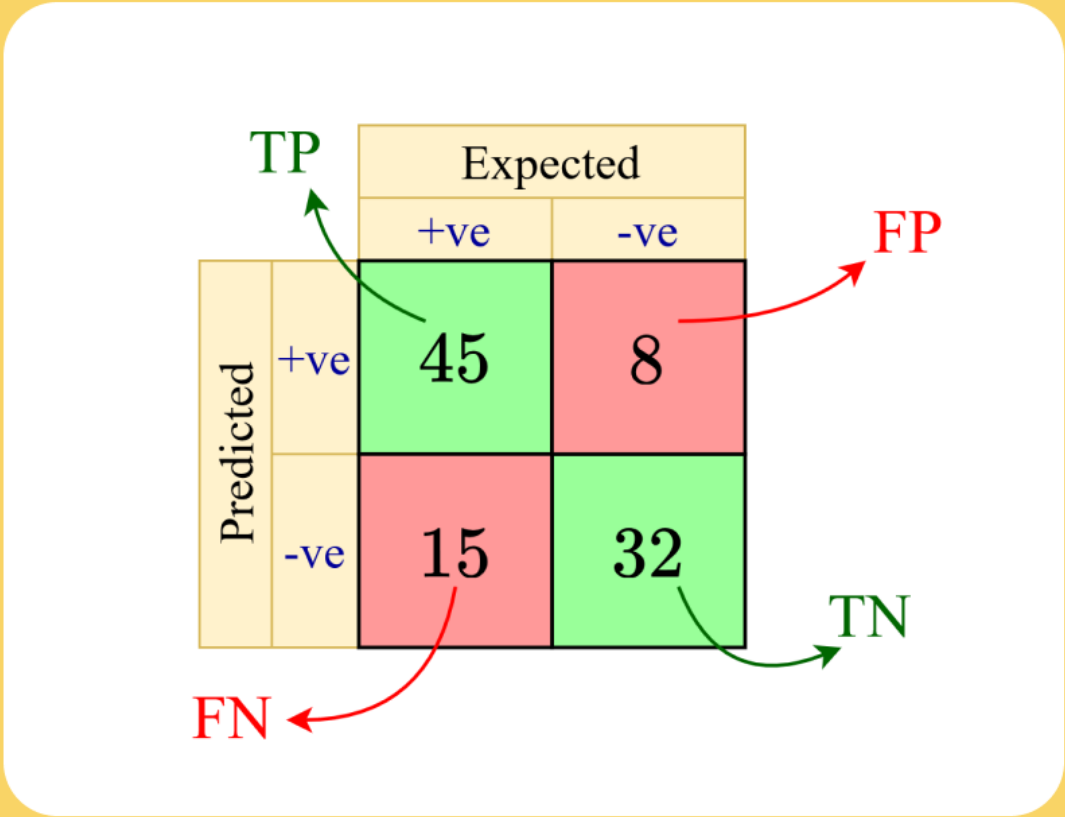




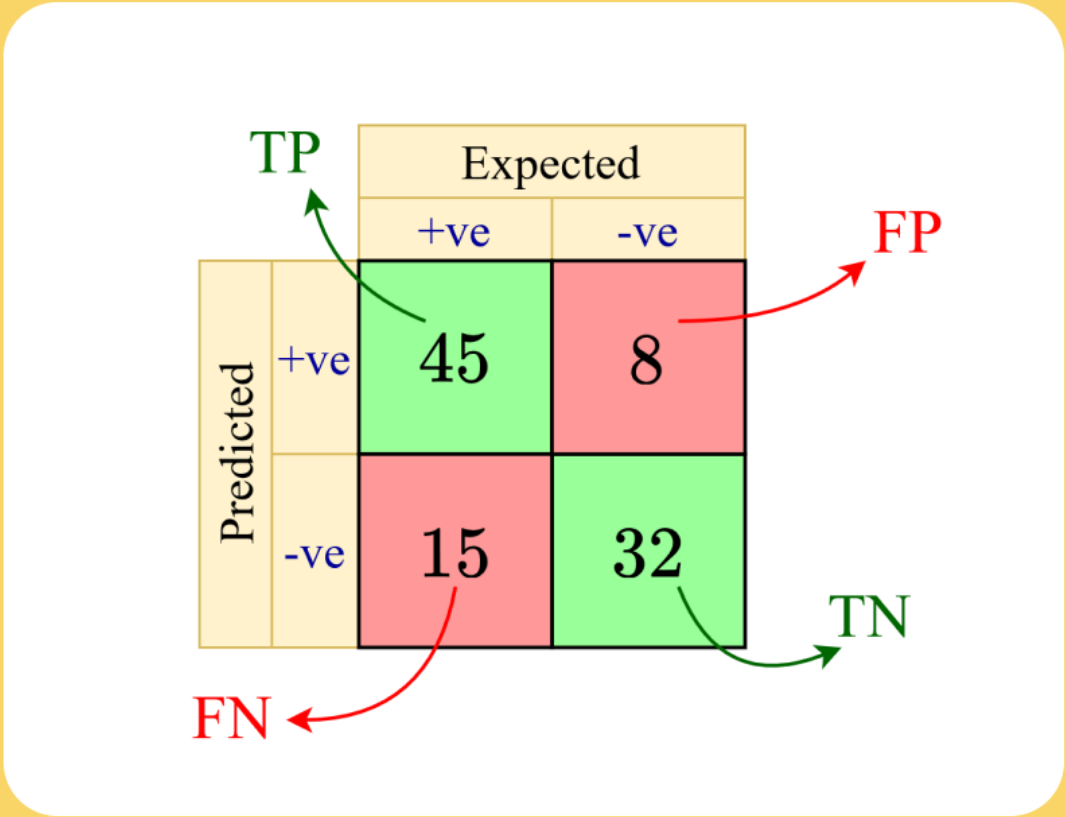

Training 1-NN
Initial Spread Index (ISI)
3
2
1
0
-1
-3
-2
-1
0
1
2
Temperature
classes
fire
not fire

Evaluating performance




TP
Expected
+ve
-ve
FP
Predicted
+ve
45
8
-ve
15
32
TN
FN

Confusion matrix: tabulation of true (i.e. expected) and predicted class labels



Performance metrics

Common metrics include:

- accuracy
- sensitivity
- specificity
- positive predictive value (PPV)

```
conf_mat(fire_results, truth = classes, estimate = predicted)
          Truth
Prediction fire not fire
  fire       61        2
  not fire    6       53
```

Accuracy

Proportion of correctly classified cases

$$\text{Accuracy} = \frac{\text{true positives} + \text{true negatives}}{n}$$

```
          Truth
Prediction fire not fire
  fire       61        2
  not fire    6       53
```

```
accuracy(fire_results, truth = classes,
         estimate = predicted)
# A tibble: 1 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.934
```

Sensitivity

Proportion of positive cases that are predicted to be positive

$$\text{Sensitivity} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Also called... true positive rate or recall

```
            Truth
Prediction fire not fire
  fire       61        2
  not fire    6       53
```

```
sens(fire_results, truth = classes,
          estimate = predicted)
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 sens    binary         0.910
```

Specificity

Proportion of negative cases that are predicted to be negative

$$\text{Specificity} = \frac{\text{true negatives}}{\text{false positives} + \text{true negatives}}$$

Also called... true negative rate

```
           Truth
Prediction fire not fire
  fire       61        2
  not fire    6       53
```

```
spec(fire_results, truth = classes,
          estimate = predicted)
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 spec    binary         0.964
```

Positive predictive value (PPV)

Proportion of cases that are predicted to be positives that are truly positives

$$\text{PPV} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

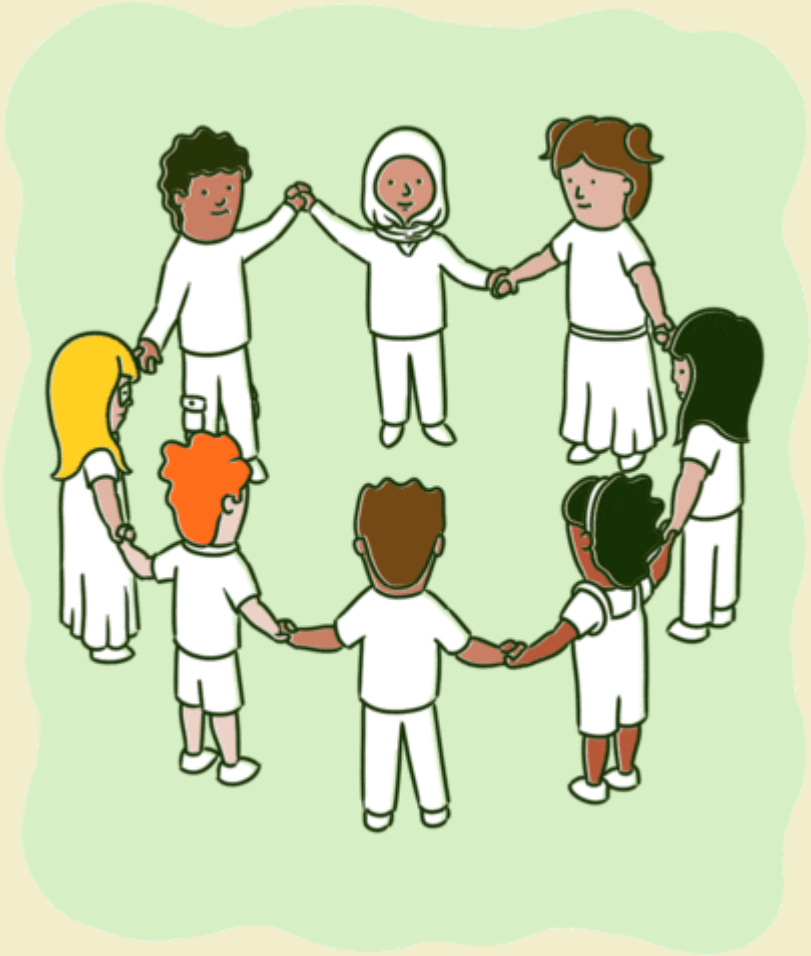
Also called... precision

```
          Truth
Prediction fire not fire
  fire       61        2
  not fire    6       53
```

```
ppv(fire_results, truth = classes,
          estimate = predicted)
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 ppv     binary         0.968
```

Group Activity 1

15:00



- Get the class activity 20.Rmd file from moodle and work on activity 1

Here is the confusion matrix for a hypothetical two-class 7-NN penguin classifier

	Truth	
Prediction	fire	not fire
fire	51	2
not fire	1	44

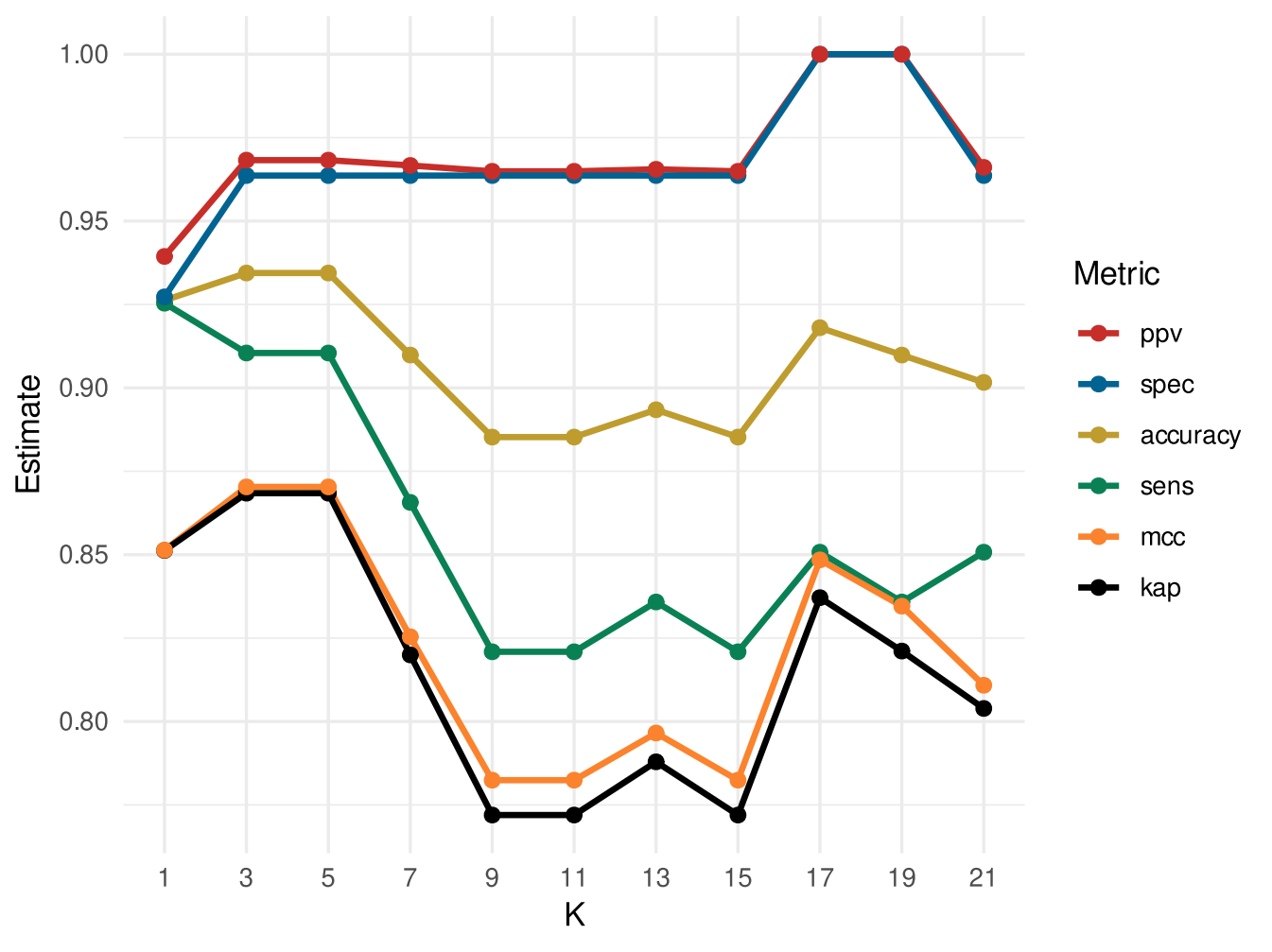
Calculate the accuracy, sensitivity, specificity, and PPV of this classifier.

Choice of metrics !!

```
custom_metrics <- metric_set(accuracy, sens, spec, ppv)
metrics <- custom_metrics(fire_results, truth = classes, estimate = predicted)
```

```
metrics %>% select(-.estimator)
# A tibble: 4 × 2
  .metric  .estimate
  <chr>        <dbl>
1 accuracy     0.934
2 sens         0.910
3 spec         0.964
4 ppv          0.968
```

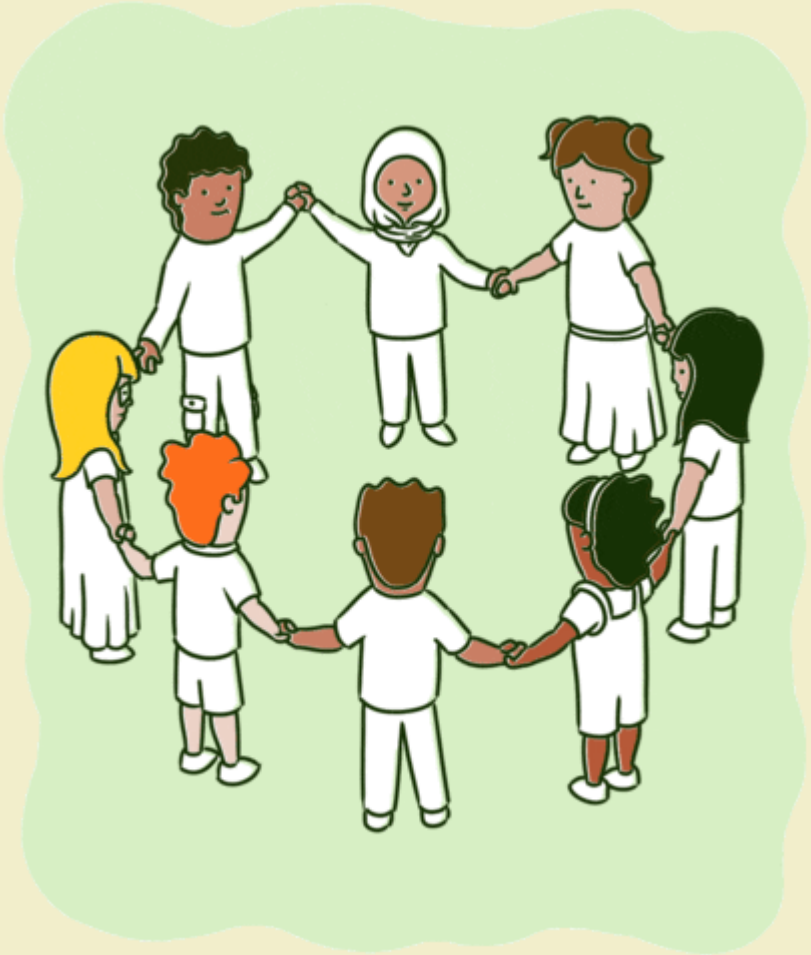
Plot them over the hyperparamter, K



Choose the optimal K based on majority of the metrics!

Group Activity 2

15:00



- Get the class activity 20.Rmd file from moodle and work on activity 1