Homework Unit 8: Advanced Performance Metrics

Nicole Friedl

2025-08-07

Introduction

To begin, download the following from the course web book (Unit 8):

- `hw_unit_8_performance.qmd` (notebook for this assignment)
- `breast_cancer.csv` (data for this assignment)

The data for this week's assignment has information about breast cancer diagnoses. It contains characteristics of different breast cancer tumors and classifies the tumor as benign or malignant. Your goal is to choose among two candidate statistical algorithms (general GLM vs a tuned KNN model) to identify and evaluate the best performing model for diagnosis.

You can imagine that the consequences of missing cancerous tumors are not equal to the consequences of misdiagnosing benign tumors. In this assignment, we will explore how the performance metric and balance of diagnoses affect our evaluation of best performing model in this data.

NOTE: Fitting models in this assignment will generate some warnings having to do with `glm.fit`. This is to be expected, and we are going to review these warnings and some related issues in our next lab.

Let's get started!

Setup

Set up your notebook in this section. You will want to set your path and initiate parallel processing here!

```
options(conflicts.policy = "depends.ok")
devtools::source_url("https://github.com/jjcurtin/lab_support/blob/main/fun_ml.R?raw=true")
```

```
i SHA-1 hash of file is "32a0bc8ced92c79756b56ddcdc9a06e639795da6"
```

```
tidymodels_conflictRules()

library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.0.4
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
library(tidymodels)
```

```
-- Attaching packages --------------------------------------- tidymodels 1.3.0 --
v broom        1.0.7     v rsample      1.2.1
v dials        1.4.0     v tune         1.3.0
v infer        1.0.7     v workflows    1.2.0
v modeldata    1.4.0     v workflowsets 1.1.0
v parsnip      1.3.0     v yardstick    1.3.2
v recipes      1.2.1
-- Conflicts ------------------------------------------ tidymodels_conflicts() --
x scales::discard() masks purrr::discard()
x dplyr::filter()   masks stats::filter()
x recipes::fixed()  masks stringr::fixed()
x dplyr::lag()      masks stats::lag()
x yardstick::spec() masks readr::spec()
x recipes::step()   masks stats::step()
```

```
library(discrim, exclude = "smoothness")
library(skimr)
library(future)
library(rsample)
library(cowplot, include.only = c("plot_grid", "theme_half_open"))
library(corrplot, include.only = "corrplot.mixed")
```

```
corrplot 0.95 loaded
```

```
library(ggplot2)
library(xfun, include.only = "cache_rds")

devtools::source_url("https://github.com/jjcurtin/lab_support/blob/main/fun_plots.R?raw=true")
```

```
i SHA-1 hash of file is "def6ce26ed7b2493931fde811adff9287ee8d874"
```

```
devtools::source_url("https://github.com/jjcurtin/lab_support/blob/main/fun_eda.R?raw=true")
```

```
i SHA-1 hash of file is "c045eee2655a18dc85e715b78182f176327358a7"
```

```
options(tibble.width = Inf, dplyr.print_max=Inf)
rerun_setting <- FALSE

# cl <- parallel::makePSOCKcluster(parallel::detectCores(logical = FALSE))
# doParallel::registerDoParallel(cl)

plan(multisession, workers = parallel::detectCores(logical = FALSE))
#I was getting warning messages so I went back into John's lecture about parallel processing

path_data <- "homework/data"
```

Read in your data

Read in the `breast_cancer.csv` data file and save as an object called `data_all`, perform any checks needed on the data (i.e., light *cleaning* EDA) and set the outcome `diagnosis` as a factor with malignant as the first level.

```r
data_all <- read_csv(here::here(path_data, "breast_cancer.csv"),
                     show_col_types = FALSE) |>
  glimpse()
```

```
Rows: 423
Columns: 31
$ diagnosis               <chr> "malignant", "benign", "benign", "malignant", ~
$ perimeter_se            <dbl> 1.9740, 1.6670, 1.4890, 2.9890, 2.6840, 1.4450~
$ fractal_dimension_mean  <dbl> 0.05986, 0.06320, 0.05828, 0.06768, 0.05934, 0~
$ concave_points_worst    <dbl> 0.12520, 0.11050, 0.03002, 0.20270, 0.06544, 0~
$ symmetry_mean           <dbl> 0.1594, 0.1886, 0.1845, 0.2157, 0.1834, 0.1514~
$ texture_se              <dbl> 0.3621, 0.7339, 1.6470, 0.9489, 0.8429, 1.0660~
$ concave_points_se       <dbl> 0.008260, 0.013040, 0.003419, 0.012710, 0.0091~
$ concavity_mean          <dbl> 0.075500, 0.070970, 0.004967, 0.169000, 0.0263~
$ fractal_dimension_se    <dbl> 0.002881, 0.001982, 0.002534, 0.003884, 0.0014~
$ radius_worst            <dbl> 17.770, 12.640, 12.360, 18.810, 12.970, 14.200~
$ concave_points_mean     <dbl> 0.040790, 0.044970, 0.006434, 0.089230, 0.0206~
$ radius_mean             <dbl> 15.120, 11.610, 11.220, 14.870, 11.500, 12.620~
$ smoothness_se           <dbl> 0.005472, 0.005884, 0.004359, 0.006985, 0.0063~
$ smoothness_worst        <dbl> 0.14910, 0.14150, 0.09994, 0.18780, 0.11830, 0~
$ symmetry_se             <dbl> 0.01523, 0.01848, 0.01916, 0.01602, 0.02292, 0~
$ radius_se               <dbl> 0.2711, 0.2456, 0.2239, 0.4266, 0.3927, 0.2449~
$ concavity_worst         <dbl> 0.33270, 0.23020, 0.02318, 0.47040, 0.08105, 0~
$ concavity_se            <dbl> 0.020390, 0.026310, 0.003223, 0.030110, 0.0124~
$ compactness_se          <dbl> 0.019190, 0.020050, 0.006813, 0.025630, 0.0106~
$ smoothness_mean         <dbl> 0.08876, 0.10880, 0.07780, 0.11620, 0.09345, 0~
$ area_se                 <dbl> 26.440, 15.890, 15.460, 41.180, 26.990, 18.510~
$ area_worst              <dbl> 989.5, 475.7, 470.9, 1095.0, 508.9, 624.0, 544~
$ perimeter_mean          <dbl> 98.78, 75.46, 70.79, 98.64, 73.28, 81.35, 79.0~
$ compactness_mean        <dbl> 0.09588, 0.11680, 0.03574, 0.16490, 0.05991, 0~
$ area_mean               <dbl> 716.6, 408.2, 386.8, 682.5, 407.4, 496.4, 466.~
$ fractal_dimension_worst <dbl> 0.09740, 0.07427, 0.07307, 0.10650, 0.06487, 0~
$ texture_mean            <dbl> 16.68, 16.02, 33.81, 16.67, 18.45, 23.97, 18.5~
$ perimeter_worst         <dbl> 117.70, 81.93, 78.44, 127.10, 83.12, 90.67, 85~
$ symmetry_worst          <dbl> 0.3415, 0.2787, 0.2911, 0.3585, 0.2740, 0.2826~
$ texture_worst           <dbl> 20.24, 19.67, 41.78, 27.37, 22.46, 31.31, 27.4~
$ compactness_worst       <dbl> 0.33310, 0.21700, 0.06885, 0.44800, 0.10490, 0~
```

```r
data_all <- data_all |>
  mutate(diagnosis = factor(diagnosis, levels = c("malignant", "benign")))

str(data_all$diagnosis)
```

```
 Factor w/ 2 levels "malignant","benign": 1 2 2 1 2 2 2 2 1 1 ...
```

```
sum(is.na(data_all))
```

```
[1] 0
```

```
summary(data_all)
```

```
    diagnosis    perimeter_se    fractal_dimension_mean concave_points_worst
 malignant: 66   Min.   : 0.757   Min.   :0.04996        Min.   :0.00000
 benign   :357   1st Qu.: 1.500   1st Qu.:0.05850        1st Qu.:0.05602
                 Median : 2.028   Median :0.06154        Median :0.08216
                 Mean   : 2.412   Mean   :0.06293        Mean   :0.09196
                 3rd Qu.: 2.679   3rd Qu.:0.06600        3rd Qu.:0.11550
                 Max.   :21.980   Max.   :0.09744        Max.   :0.29100
 symmetry_mean      texture_se     concave_points_se  concavity_mean
 Min.   :0.1060   Min.   :0.3602   Min.   :0.000000   Min.   :0.00000
 1st Qu.:0.1595   1st Qu.:0.8185   1st Qu.:0.006856   1st Qu.:0.02362
 Median :0.1742   Median :1.0950   Median :0.009883   Median :0.04249
 Mean   :0.1778   Mean   :1.2119   Mean   :0.010608   Mean   :0.06579
 3rd Qu.:0.1935   3rd Qu.:1.4825   3rd Qu.:0.013180   3rd Qu.:0.08322
 Max.   :0.2906   Max.   :4.8850   Max.   :0.052790   Max.   :0.42640
 fractal_dimension_se  radius_worst   concave_points_mean  radius_mean
 Min.   :0.0008948    Min.   : 7.93   Min.   :0.00000     Min.   : 6.981
 1st Qu.:0.0021550    1st Qu.:12.38   1st Qu.:0.01750     1st Qu.:11.305
 Median :0.0029680    Median :13.75   Median :0.02653     Median :12.540
 Mean   :0.0036915    Mean   :14.66   Mean   :0.03591     Mean   :13.006
 3rd Qu.:0.0044085    3rd Qu.:15.64   3rd Qu.:0.04424     3rd Qu.:14.155
 Max.   :0.0298400    Max.   :33.13   Max.   :0.19130     Max.   :28.110
 smoothness_se      smoothness_worst   symmetry_se         radius_se
 Min.   :0.001713   Min.   :0.07117   Min.   :0.007882   Min.   :0.1115
 1st Qu.:0.005215   1st Qu.:0.11270   1st Qu.:0.015360   1st Qu.:0.2183
 Median :0.006380   Median :0.12780   Median :0.018970   Median :0.2810
 Mean   :0.007054   Mean   :0.12806   Mean   :0.020793   Mean   :0.3408
 3rd Qu.:0.008199   3rd Qu.:0.14130   3rd Qu.:0.023835   3rd Qu.:0.3815
 Max.   :0.021770   Max.   :0.20980   Max.   :0.078950   Max.   :2.8730
 concavity_worst    concavity_se     compactness_se     smoothness_mean
 Min.   :0.00000   Min.   :0.00000   Min.   :0.002252   Min.   :0.05263
 1st Qu.:0.08625   1st Qu.:0.01269   1st Qu.:0.011885   1st Qu.:0.08448
```

```
 Median :0.16480   Median :0.02045   Median :0.017960   Median :0.09357
 Mean   :0.21574   Mean   :0.02878   Mean   :0.023152   Mean   :0.09421
 3rd Qu.:0.29770   3rd Qu.:0.03689   3rd Qu.:0.030050   3rd Qu.:0.10300
 Max.   :1.25200   Max.   :0.39600   Max.   :0.106400   Max.   :0.16340
    area_se          area_worst     perimeter_mean   compactness_mean
 Min.   :  6.802   Min.   : 185.2   Min.   : 43.79   Min.   :0.01938
 1st Qu.: 16.390   1st Qu.: 471.1   1st Qu.: 72.48   1st Qu.:0.05890
 Median : 20.950   Median : 582.6   Median : 80.98   Median :0.07943
 Mean   : 30.050   Mean   : 701.1   Mean   : 84.12   Mean   :0.09097
 3rd Qu.: 29.485   3rd Qu.: 749.9   3rd Qu.: 90.77   3rd Qu.:0.11255
 Max.   :525.600   Max.   :3432.0   Max.   :188.50   Max.   :0.34540
   area_mean      fractal_dimension_worst  texture_mean   perimeter_worst
 Min.   : 143.5   Min.   :0.05504         Min.   : 9.71   Min.   : 50.41
 1st Qu.: 391.6   1st Qu.:0.07050         1st Qu.:15.51   1st Qu.: 79.93
 Median : 481.6   Median :0.07842         Median :17.93   Median : 89.02
 Mean   : 546.2   Mean   :0.08156         Mean   :18.48   Mean   : 96.05
 3rd Qu.: 617.5   3rd Qu.:0.08922         3rd Qu.:20.52   3rd Qu.:102.30
 Max.   :2499.0   Max.   :0.17300         Max.   :39.28   Max.   :229.30
 symmetry_worst   texture_worst   compactness_worst
 Min.   :0.1565   Min.   :12.02   Min.   :0.02729
 1st Qu.:0.2445   1st Qu.:19.91   1st Qu.:0.12385
 Median :0.2744   Median :23.58   Median :0.18240
 Mean   :0.2808   Mean   :24.39   Mean   :0.21517
 3rd Qu.:0.3088   3rd Qu.:27.75   3rd Qu.:0.26535
 Max.   :0.6638   Max.   :49.54   Max.   :0.86810
```

```
table(data_all$diagnosis)
```

```
malignant    benign
       66       357
```

```
prop.table(table(data_all$diagnosis))
```

```
malignant    benign
0.1560284 0.8439716
```

Print a table to see the balance of positive and negative diagnosis cases.

```
data_all |> tab(diagnosis)
```

```
# A tibble: 2 x 3
  diagnosis     n  prop
  <fct>     <int> <dbl>
1 malignant    66 0.156
2 benign      357 0.844
```

What do you notice about the distribution of your outcome variable? Do you have any concerns?

We have more benign than malignant. It is concerning since it is unbalanced and could be misleading.

Split data into train and test

Hold out 1/3 of the data as a test set for evaluation using the `initial_split()` function. Use the provided seed. Stratify on diagnosis.

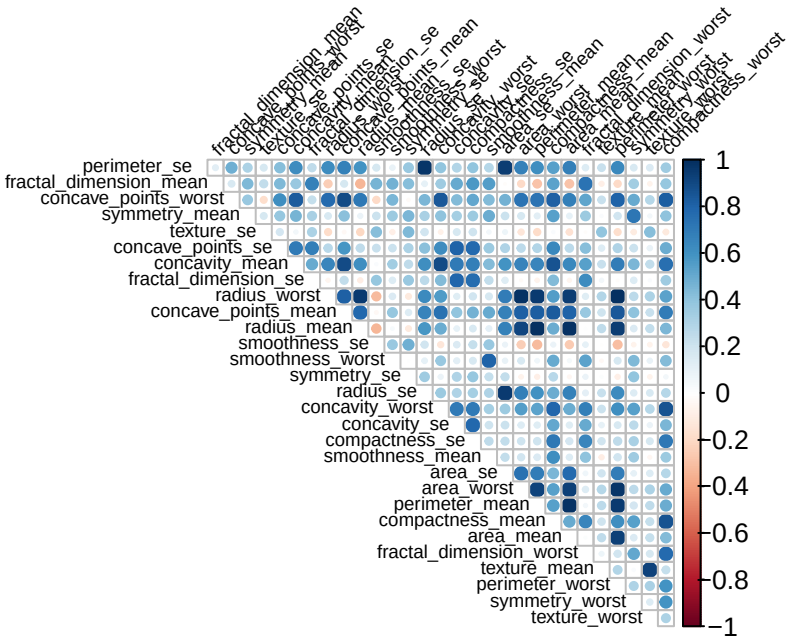
```
set.seed(12345)

splits_test <- data_all |>
  initial_split(prop = 2/3, strata = "diagnosis")

data_trn <- splits_test |>
  analysis()

data_test <- splits_test |>
  assessment()
```

Light Modeling EDA

Look at correlations between predictors in `data_train`.

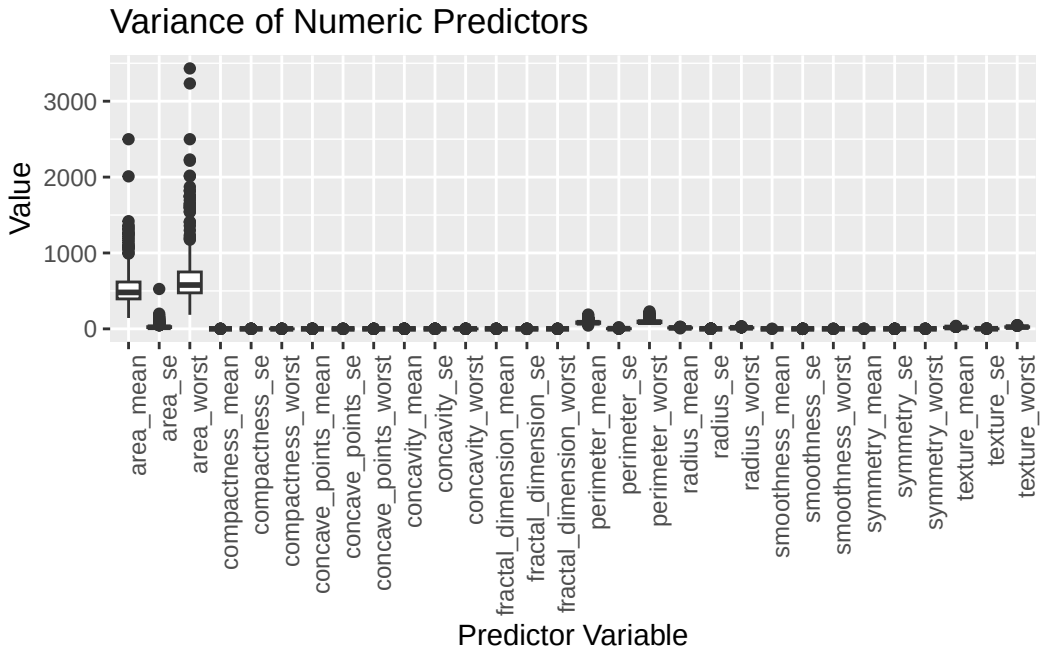
```
cor_matrix <- data_trn |>
  select(where(is.numeric)) |>
  cor(use = "pairwise.complete.obs")

corrplot::corrplot(cor_matrix,
                   method = "circle",
```

```
                    type = "upper",
                    tl.col = "black",
                    tl.srt = 45,
                    tl.cex = 0.6,
                    diag = FALSE,
                    mar = c(0,0,1,0))
```



Visualize the variance of your predictors in `data_train`.

```
data_trn |>
  select(where(is.numeric)) |>
  gather(key = "variable", value = "value") |>
  ggplot(aes(x = variable, y = value)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Variance of Numeric Predictors",
       x = "Predictor Variable",
       y = "Value")
```



Now, answer the following questions:

Why should you be looking at variance of your predictors?

If a predictor has very low variance, it means that the values of that predictor do not change much across the dataset. This could make it a poor feature for a model since it won't provide much information to distinguish between different outcomes. In such cases, the predictor could be dropped.

If you had concerns about the variance of your predictors, what would you do?

Extreme values or outliers in a predictor can greatly influence certain models, especially those sensitive to outliers (e.g., linear regression). By visualizing the variance and distribution of predictors (e.g., using boxplots), I can identify outliers and take appropriate action, such as transforming the variable or removing extreme values.

Do you have concerns about the variance of your predictors in these data?

High variance in a predictor often indicates that the feature provides significant information to the model. On the other hand, predictors with little variance may not contribute much to the predictive power of the model.

GLM vs KNN

In this part of this assignment, you will compare the performance of a standard GLM model vs a KNN model (tuned for neighbors) for predicting breast cancer diagnoses from all available

predictors. You will choose between these models using bootstrapped resampling, and evaluate the final performance of your model in the held out test set created earlier in this script. You will now select and evaluate models using ROC AUC instead of accuracy.

Bootstrap splits

Split `data_train` into 100 bootstrap samples stratified on diagnosis. Use the provided seed.

```
set.seed(12345)

splits_boot <- data_trn |>
  bootstraps(times = 100, strata = "diagnosis")

grid_glmnet <- expand_grid(penalty = exp(seq(-8, 3, length.out = 200)),
                           mixture = seq(0, 1, length.out = 6))
```

Build recipes

Write 2 recipes (one for GLM, one for KNN) to predict breast cancer diagnosis from all predictors in `data_train`. Include the minimal necessary steps for each algorithm, including what you learned from your light EDA above.

```
library(themis)

rec_glm <- recipe(diagnosis ~ ., data = data_trn) |>
  step_impute_median(all_numeric_predictors()) |>
  step_impute_mode(all_nominal_predictors()) |>
  step_dummy(all_nominal_predictors()) |>
  step_normalize(all_predictors())

rec_knn <- recipe(diagnosis ~ ., data = data_trn) |>
  step_impute_median(all_numeric_predictors()) |>
  step_impute_mode(all_nominal_predictors()) |>
  step_normalize(all_numeric_predictors())
```

Fit GLM

Fit a logistic regression classifier using the recipe you created and your bootstrap splits. Use ROC AUC (`roc_auc`) as your performance metric.

```r
fits_glmnet <- cache_rds(
  expr = {
    logistic_reg(penalty = tune(),
                 mixture = tune()) |>
      set_engine("glmnet") |>
      set_mode("classification") |>
      tune_grid(preprocessor = rec_glm,
                resamples = splits_boot, grid = grid_glmnet,
                metrics = metric_set(roc_auc))

  },
  rerun = rerun_setting,
  dir = "cache/008/",
  file = "fits_glmnet_auc")
```

Print the average ROC AUC of your logistic regression model.

```r
fits_glmnet %>%
  collect_metrics() %>%
  filter(.metric == "roc_auc") %>%
  summarize(mean_roc_auc = mean(mean))
```

```
# A tibble: 1 x 1
  mean_roc_auc
         <dbl>
1        0.850
```

Fit KNN

Set up a hyperparameter grid to consider a range of values for `neighbors` in your KNN models.

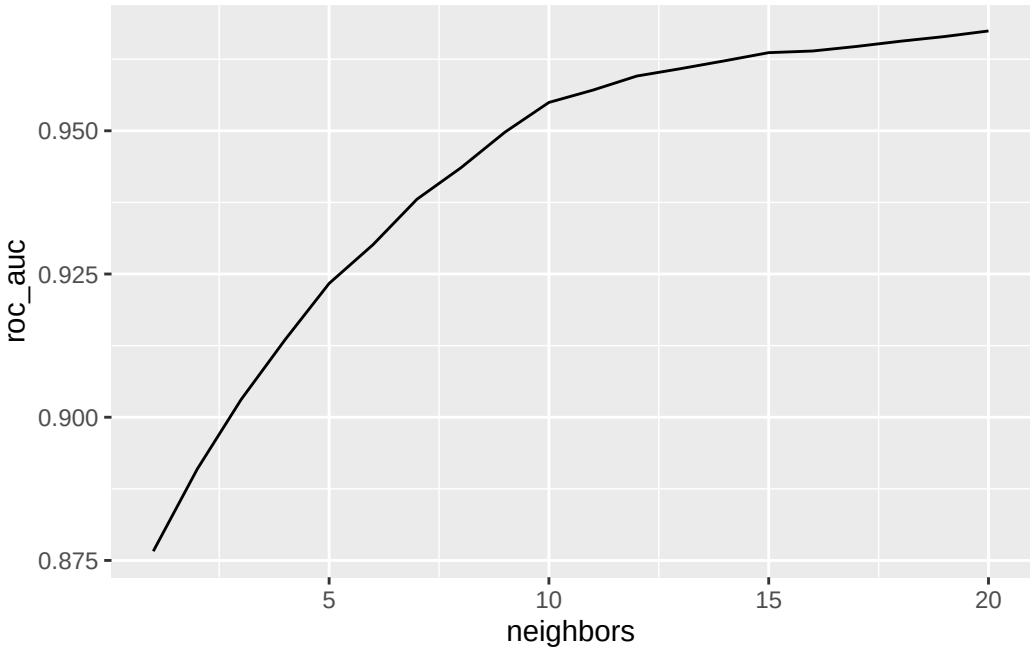
```r
grid_knn <- expand_grid(neighbors = seq(1, 20, by = 1))
```

Fit a KNN model using the recipe you created and your bootstrap splits. Use ROC AUC (`roc_auc`) as your performance metric.

```
fits_knn <- cache_rds(
  expr = {
    nearest_neighbor(neighbors = tune()) |>
      set_engine("kknn") |>
      set_mode("classification") |>
      tune_grid(preprocessor = rec_knn,
                resamples = splits_boot, grid = grid_knn,
                metrics = metric_set(roc_auc))
  },
  dir = "cache/008/",
  file = "fits_knn_auc",
  rerun = rerun_setting)
```

Generate a plot to help you determine if you considered a wide enough range of values for `neighbors`.

```
plot_hyperparameters(tune_fit = fits_knn,
                     hp1 = "neighbors",
                     metric = "roc_auc",
                     log_hp1 = FALSE)
```



Print the best value for the `neighbors` hyperparameter across resamples based on model ROC AUC.

```r
best_knn <- fits_knn %>%
  collect_metrics() %>%
  filter(.metric == "roc_auc") %>%
  group_by(neighbors) %>%
  summarize(mean_roc_auc = mean(mean)) %>%
  arrange(desc(mean_roc_auc)) %>%
  slice(1)
```

Print the average ROC AUC of your best KNN regression model

```r
best_knn_m <- best_knn %>%
  summarize(mean_roc_auc = mean(mean_roc_auc))

best_knn_m
```

```
# A tibble: 1 x 1
  mean_roc_auc
         <dbl>
1        0.967
```

Select and fit best model

Now you will select your best model configuration among the various KNN and GLM models based on overall ROC AUC and train it on your full training sample.

Create training (`feat_train`) and test (`feat_test`) feature matrices using your best recipe (GLM or KNN)

```r
feat_train <- rec_knn %>%
  prep() %>%
  bake(new_data = data_trn)

feat_test <- rec_knn %>%
  prep() %>%
  bake(new_data = data_test)
```

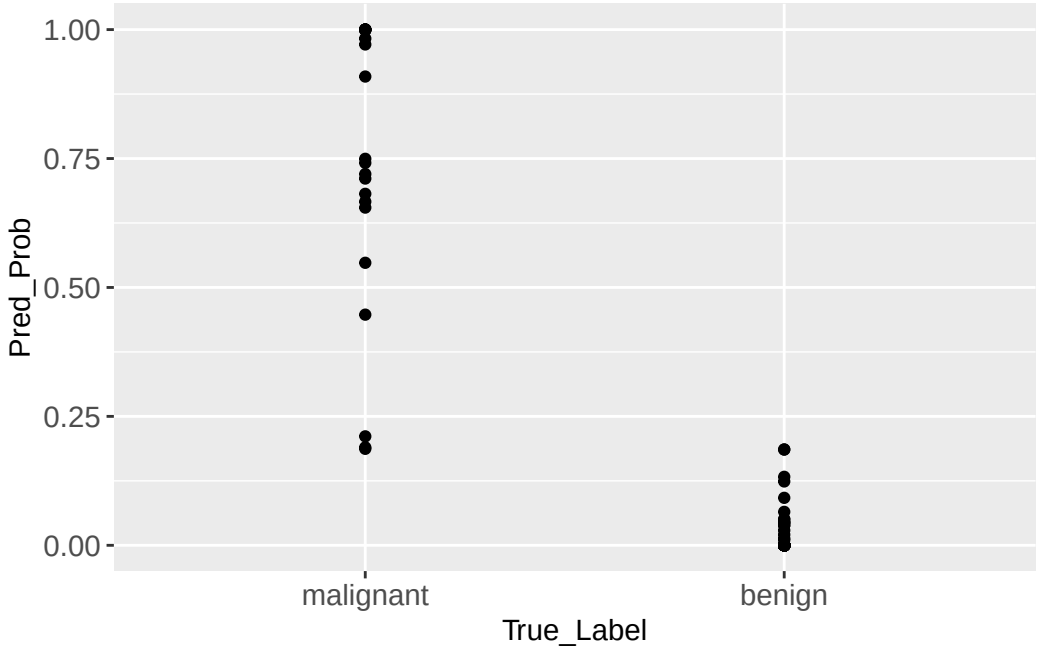
Fit your best performing model on the full training sample (`feat_train`).

```r
best_knn_model <- nearest_neighbor(neighbors = best_knn$neighbors) %>%
  set_engine("kknn") %>%
  set_mode("classification") %>%
  fit(diagnosis ~ ., data = feat_train)
```

Evaluate the best model

Make a figure to plot the ROC of your best model in the test set.

```
knn_preds <- predict(best_knn_model, new_data = feat_test, type = "prob")

results <- data.frame(
  True_Label = feat_test$diagnosis,
  Pred_Prob = knn_preds$.pred_malignant
)

plot_scatter(results, "True_Label", "Pred_Prob")
```



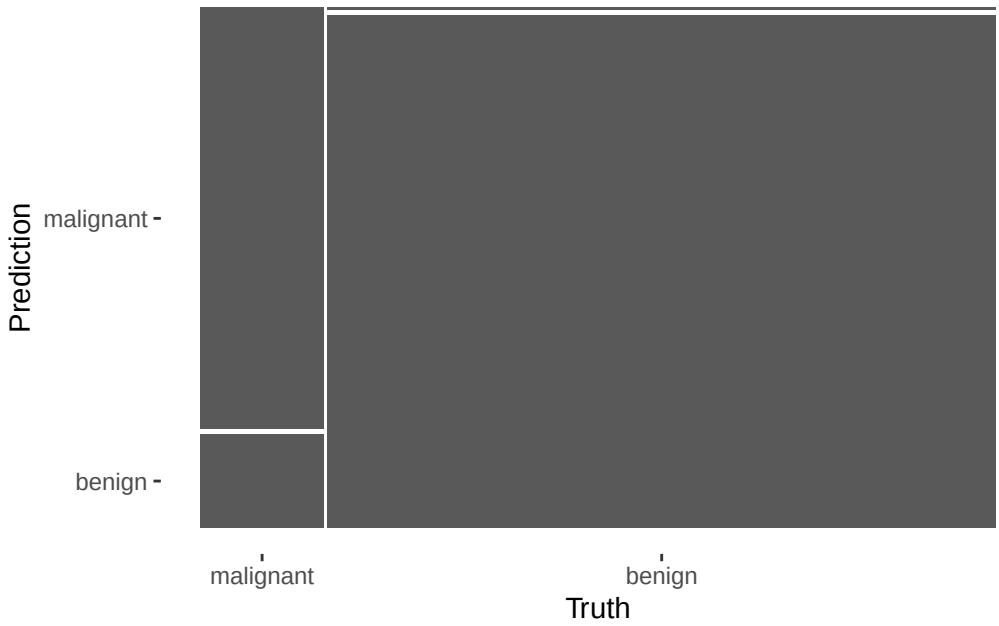
Generate a confusion matrix depicting your model's performance in test.

```
knn_preds_class <- predict(best_knn_model, feat_test)$.pred_class

cm_knn <- tibble(
  truth = feat_test$diagnosis,
  estimate = knn_preds_class
) %>%
  conf_mat(truth, estimate)

cm_knn
```

```
           Truth
Prediction  malignant benign
  malignant        18      0
  benign            4    119
```

Make a plot of your confusion matrix.

```
autoplot(cm_knn)
```



Report the ROC AUC, accuracy, sensitivity, specificity, PPV, and NPV of your best model in the held out test set.

```
cm_knn |>
  summary() |>
  filter(.metric == "ppv" | .metric == "npv") |>
  select(-.estimator)
```

```
# A tibble: 2 x 2
  .metric .estimate
  <chr>       <dbl>
1 ppv         1
2 npv         0.967
```

```r
cm_knn |>
  summary() |>
  filter(.metric == "sens" | .metric == "spec" | .metric == "bal_accuracy") |>
  select(-.estimator)
```

```
# A tibble: 3 x 2
  .metric      .estimate
  <chr>            <dbl>
1 sens             0.818
2 spec             1
3 bal_accuracy     0.909
```

Part 2: Addressing class imbalance

Since only 15% of our cases our malignant, let's see if we can achieve higher sensitivity by up-sampling our data with SMOTE. We will again select between a standard GLM vs tuned KNN using bootstrapped CV and evaluate our best model in test.

Build recipes

Update your previous recipes to up-sample the minority class (malignant) in diagnosis using `step_smote()`. Remember to make 2 recipes (one for GLM, one for KNN).

I read ahead in lecture and had already applied this to my old recipe, but I went back and applied it here instead. No wonder my model was already performing so well.

```r
rec_knn_1 <- recipe(diagnosis ~ ., data = data_trn) |>
  step_impute_median(all_numeric_predictors()) |>
  step_impute_mode(all_nominal_predictors()) |>
  step_normalize(all_numeric_predictors()) |>
  step_smote(diagnosis)

rec_glm_2 <- recipe(diagnosis ~ ., data = data_trn) |>
  step_impute_median(all_numeric_predictors()) |>
  step_impute_mode(all_nominal_predictors()) |>
  step_dummy(all_nominal_predictors()) |>
  step_normalize(all_predictors()) |>
  step_smote(diagnosis)
```

Fit GLM

Fit an up-sampled logistic regression classifier using the new GLM recipe you created and your bootstrap splits. Use ROC AUC as your performance metric.

```
fits_glmnet_up <- cache_rds(
  expr = {
    logistic_reg(penalty = tune(),
                 mixture = tune()) |>
      set_engine("glmnet") |>
      set_mode("classification") |>
      tune_grid(
        preprocessor = rec_glm_2,
        resamples = splits_boot,
        grid = grid_glmnet,
        metrics = metric_set(roc_auc)
      )
  },
  rerun = rerun_setting,
  dir = "cache/008/",
  file = "fit_glmnet_up"
)
```

Print the average ROC AUC of your logistic regression model

```
fits_glmnet_up %>%
  collect_metrics() %>%
  filter(.metric == "roc_auc") %>%
  summarize(mean_roc_auc = mean(mean))
```

```
# A tibble: 1 x 1
  mean_roc_auc
         <dbl>
1        0.862
```

Fit KNN

Set up a hyperparameter grid to consider a range of values for `neighbors` in your KNN models.

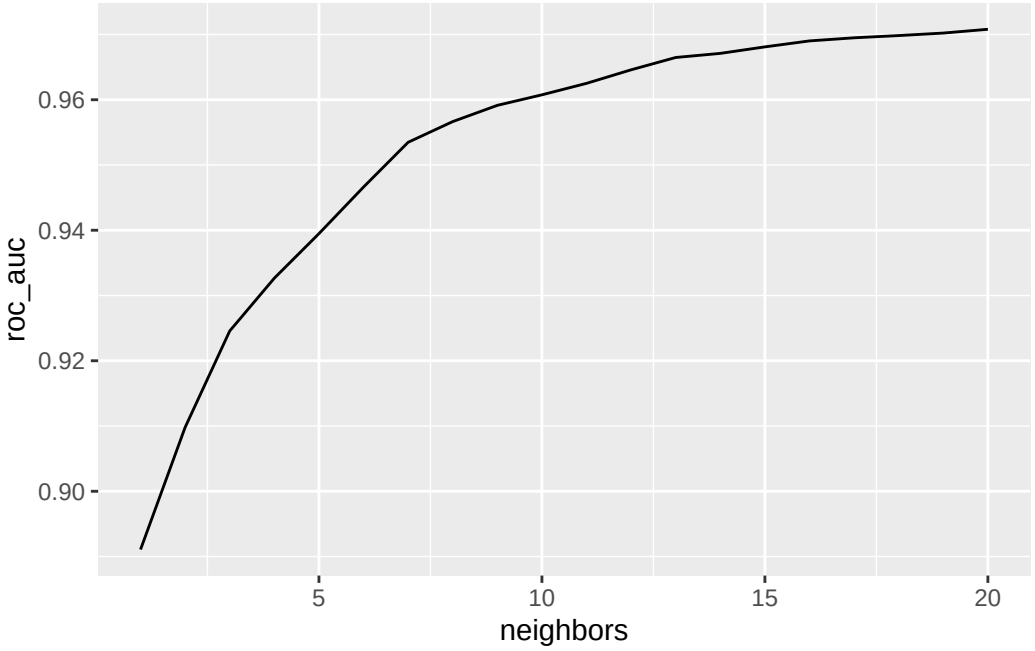
```
grid_knn_up <- expand_grid(neighbors = seq(1, 20, by = 1))
```

Fit an up-sampled KNN using the new KNN recipe you created and your bootstrap splits. Use ROC AUC as your performance metric.

```
fits_knn_up <- cache_rds(
  expr = {
    nearest_neighbor(neighbors = tune()) |>
      set_engine("kknn") |>
      set_mode("classification") |>
      tune_grid(
        preprocessor = rec_knn_1,
        resamples = splits_boot,
        grid = grid_knn_up,
        metrics = metric_set(roc_auc)
      )
  },
  dir = "cache/008/",
  file = "fits_knn_auc",
  rerun = rerun_setting
)
```

Generate a plot to help you determine if you considered a wide enough range of values for `neighbors`.

```
plot_hyperparameters(tune_fit = fits_knn_up,
                     hp1 = "neighbors",
                     metric = "roc_auc",
                     log_hp1 = FALSE)
```



Print the best value for the `neighbors` hyperparameter across resamples based on model ROC AUC.

```
best_knn_2 <- fits_knn_up %>%
  collect_metrics() %>%
  filter(.metric == "roc_auc") %>%
  group_by(neighbors) %>%
  summarize(mean_roc_auc = mean(mean)) %>%
  arrange(desc(mean_roc_auc)) %>%
  slice(1)
```

Print the average ROC AUC of your best KNN regression model

```
best_knn_up_1 <- best_knn_2 %>%
  pull(mean_roc_auc)
```

Select and fit the best model

Create the up-sampled training feature matrix using your best recipe (GLM or KNN). *Remember, do not upsample your test data!*

```
feat_train_up <- rec_knn_1 %>%
  prep(training = data_trn) %>%
  bake(new_data = NULL)
```

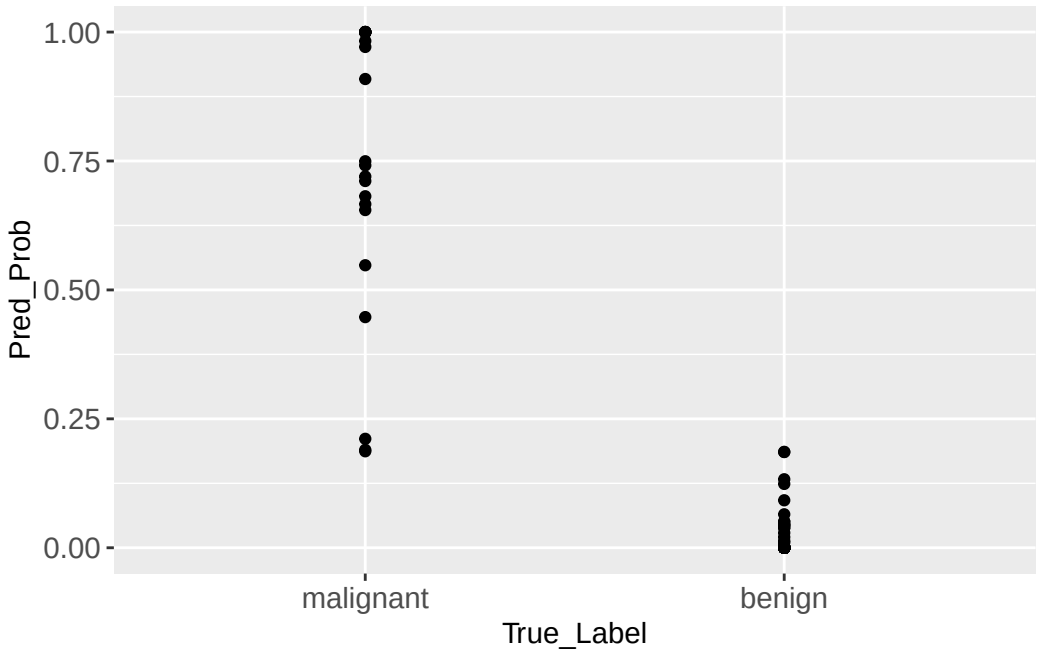
Fit your best performing up-sampled model on the full training sample.

```
fits_knn_up_2 <- cache_rds(
  expr = {
    nearest_neighbor(neighbors = tune()) |>
      set_engine("kknn") |>
      set_mode("classification") |>
      tune_grid(
        preprocessor = rec_knn_1,
        resamples = splits_boot,
        grid = grid_knn_up,
        metrics = metric_set(roc_auc)
      )
  },
  dir = "cache/008/",
  file = "fits_knn_auc",
  rerun = rerun_setting
)
```

Evaluate the best model

Make a figure to plot the ROC of your best ups-ampled model in the test set.

```
results_2 <- tibble(
  True_Label = feat_test$diagnosis,
  Pred_Prob = knn_preds$.pred_malignant
)

plot_scatter(results_2, "True_Label", "Pred_Prob")
```



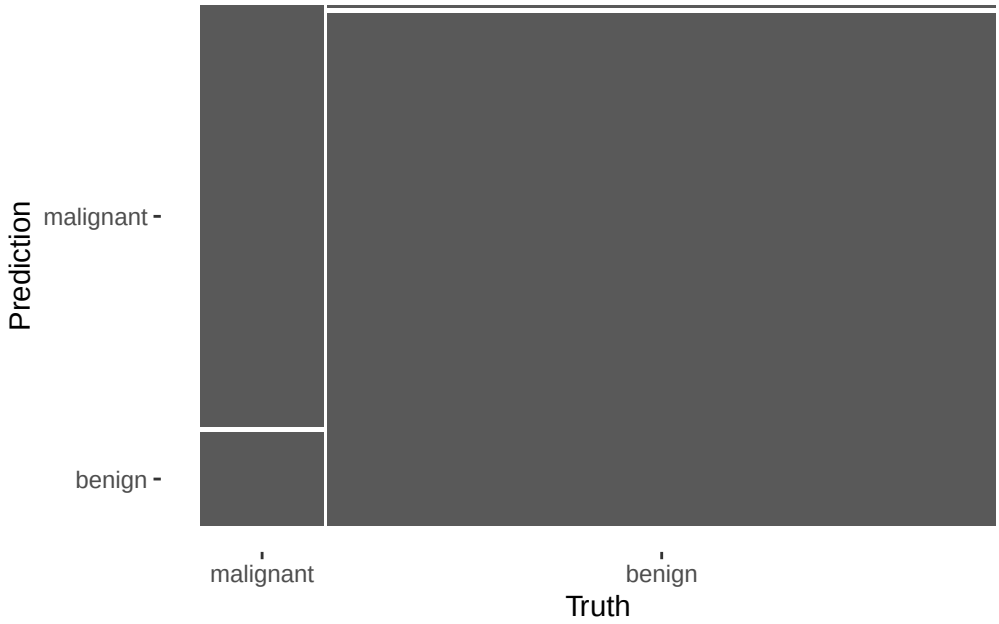
Generate a confusion matrix depicting your up-sampled model's performance in test.

```
knn_preds_class_2 <- predict(best_knn_model, feat_test)$.pred_class

cm_knn_2 <- tibble(
  truth = feat_test$diagnosis,
  estimate = knn_preds_class_2
) %>%
  conf_mat(truth, estimate)

cm_knn_2
```

```
           Truth
Prediction  malignant benign
  malignant        18      0
  benign            4    119
```

Make a plot of your confusion matrix.

```
autoplot(cm_knn_2)
```



Report the ROC AUC, accuracy, sensitivity, specificity, PPV, and NPV of your best up-sampled model in the held out test set.

```
cm_knn_2 |>
  summary() |>
  filter(.metric %in% c("ppv", "npv")) |>
  select(-.estimator)
```

```
# A tibble: 2 x 2
  .metric .estimate
  <chr>       <dbl>
1 ppv         1
2 npv         0.967
```

```
cm_knn_2 |>
  summary() |>
  filter(.metric %in% c("sens", "spec", "bal_accuracy")) |>
  select(-.estimator)
```

```
# A tibble: 3 x 2
  .metric      .estimate
  <chr>            <dbl>
1 sens             0.818
2 spec             1
3 bal_accuracy     0.909
```

Part 3: New Classification Threshold

Now you want to check if there may be an additional benefit for your model's performance if you adjust the classification threshold from its default 50% to a threshold of 40%

1) Adjust classification threshold to 40%

Make a tibble containing the following variables -

- truth: The true values of diagnosis in your test set
- prob: The predicted probabilities made by your best up-sampled model above in the test set
- estimate_40: Binary predictions of `diagnosis` (benign vs malignant) created by applying a threshold of 40% to your best model's predicted probabilities.

```
knn_preds_prob <- predict(best_knn_model, feat_test, type = "prob")

threshold_40_tibble <- tibble(
  truth = feat_test$diagnosis,
  prob = knn_preds_prob$.pred_malignant,
  estimate_40 = if_else(knn_preds_prob$.pred_malignant >= 0.40, "malignant", "benign")
)

threshold_40_tibble
```

```
# A tibble: 141 x 3
    truth        prob estimate_40
    <fct>       <dbl> <chr>
  1 benign    0       benign
  2 benign    0       benign
  3 malignant 1       malignant
  4 benign    0       benign
  5 malignant 0.655   malignant
  6 benign    0       benign
  7 benign    0       benign
  8 benign    0       benign
  9 benign    0       benign
 10 benign    0       benign
 11 benign    0.0507  benign
 12 benign    0       benign
 13 benign    0.00135 benign
 14 benign    0       benign
```

```
15 benign    0       benign
16 benign    0.0444  benign
17 benign    0       benign
18 benign    0.186   benign
19 benign    0.0507  benign
20 benign    0       benign
21 benign    0       benign
22 malignant 0.190   benign
23 benign    0.133   benign
24 benign    0       benign
25 benign    0       benign
26 benign    0       benign
27 malignant 1       malignant
28 benign    0       benign
29 benign    0.0649  benign
30 benign    0       benign
31 benign    0       benign
32 benign    0       benign
33 benign    0       benign
34 benign    0       benign
35 benign    0       benign
36 benign    0       benign
37 benign    0       benign
38 benign    0       benign
39 malignant 0.971   malignant
40 benign    0.0102  benign
41 benign    0       benign
42 benign    0       benign
43 benign    0       benign
44 benign    0       benign
45 benign    0       benign
46 malignant 0.548   malignant
47 benign    0       benign
48 malignant 0.666   malignant
49 benign    0.0444  benign
50 benign    0       benign
51 malignant 1       malignant
52 benign    0.0444  benign
53 benign    0       benign
54 benign    0       benign
55 benign    0       benign
56 benign    0       benign
57 benign    0       benign
```

```
 58 benign    0       benign
 59 malignant 0.711   malignant
 60 benign    0       benign
 61 benign    0       benign
 62 benign    0       benign
 63 benign    0.124   benign
 64 benign    0       benign
 65 malignant 0.999   malignant
 66 benign    0       benign
 67 benign    0       benign
 68 benign    0       benign
 69 benign    0       benign
 70 benign    0.0290  benign
 71 benign    0       benign
 72 benign    0       benign
 73 benign    0       benign
 74 malignant 0.909   malignant
 75 malignant 0.187   benign
 76 benign    0       benign
 77 benign    0       benign
 78 benign    0       benign
 79 benign    0       benign
 80 benign    0       benign
 81 benign    0       benign
 82 malignant 0.749   malignant
 83 benign    0.186   benign
 84 malignant 1       malignant
 85 malignant 0.742   malignant
 86 benign    0       benign
 87 benign    0       benign
 88 benign    0       benign
 89 benign    0       benign
 90 benign    0       benign
 91 benign    0       benign
 92 malignant 0.682   malignant
 93 benign    0       benign
 94 benign    0       benign
 95 benign    0       benign
 96 benign    0       benign
 97 benign    0       benign
 98 benign    0       benign
 99 benign    0       benign
100 benign    0.00135 benign
```

```
101 benign    0       benign
102 malignant 0.211   benign
103 benign    0.0135  benign
104 benign    0       benign
105 benign    0.0207  benign
106 benign    0       benign
107 benign    0       benign
108 malignant 0.983   malignant
109 benign    0       benign
110 benign    0.0388  benign
111 benign    0       benign
112 malignant 0.720   malignant
113 benign    0       benign
114 benign    0       benign
115 benign    0       benign
116 benign    0       benign
117 benign    0       benign
118 benign    0       benign
119 benign    0       benign
120 benign    0       benign
121 benign    0       benign
122 benign    0       benign
123 benign    0       benign
124 benign    0       benign
125 benign    0       benign
126 malignant 1       malignant
127 benign    0       benign
128 benign    0       benign
129 benign    0.0921  benign
130 malignant 1       malignant
131 benign    0       benign
132 benign    0       benign
133 benign    0.0388  benign
134 benign    0       benign
135 benign    0       benign
136 benign    0       benign
137 benign    0       benign
138 malignant 0.447   malignant
139 benign    0       benign
140 benign    0       benign
141 benign    0       benign
```

2) Evaluate model at new threshold

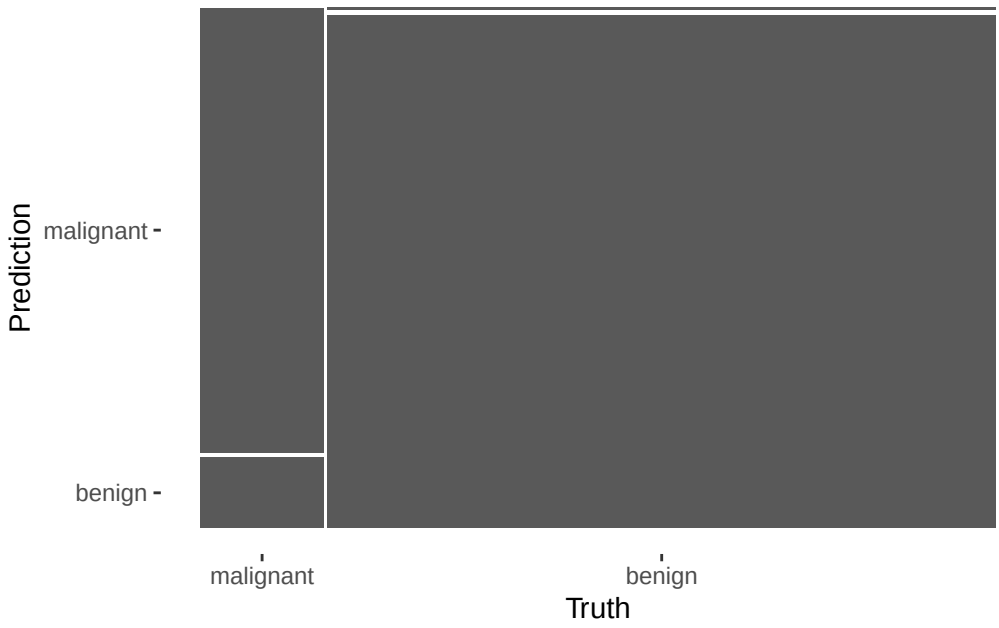
Generate a confusion matrix depicting your up-sampled model's performance in test at your new threshold.

```
cm_knn_40 <- tibble(
  truth = factor(threshold_40_tibble$truth, levels = c("malignant", "benign")),
  estimate = factor(threshold_40_tibble$estimate_40, levels = c("malignant", "benign"))
) %>%
  conf_mat(truth, estimate)

cm_knn_40
```

```
           Truth
Prediction  malignant benign
  malignant        19      0
  benign            3    119
```

Make a plot of your confusion matrix.

```
autoplot(cm_knn_40)
```



Report the ROC AUC, accuracy, sensitivity, specificity, PPV, and NPV of your best up-sampled model in the held-out test set.

```r
cm_knn_40 |>
  summary() |>
  filter(.metric %in% c("ppv", "npv")) |>
  select(-.estimator)
```

```
# A tibble: 2 x 2
  .metric .estimate
  <chr>       <dbl>
1 ppv         1
2 npv         0.975
```

```r
cm_knn_40 |>
  summary() |>
  filter(.metric %in% c("sens", "spec", "bal_accuracy")) |>
  select(-.estimator)
```

```
# A tibble: 3 x 2
  .metric      .estimate
  <chr>            <dbl>
1 sens             0.864
2 spec             1
3 bal_accuracy     0.932
```

You are a machine learning superstar