FAQ/Review

FAQ

Do we have to cross-validate to get metrics on the final model?

Do we have to cross-validate to get metrics on the final model?

YES!!!!

Remember the cautionary tale of overfitting.

Once we select our final hyperparameters and final predictors, we fit our final model on the full data. (No data should be wasted!)

But we want to report the quality of this model in terms of how we think it will perform on future data.

To get a fair estimate of the metrics, we need to cross-validate.

Why do we bother fitting a final model?

Why do we bother fitting a final model?

If we are doing **inference:**

Because we would like to use **all** the data on hand to get our final statistical estimates (e.g., the **coefficients**)

If we are doing **prediction:**

Because we want to prepare to predict on **future observations**.

We would like to use **all** the data on hand to **train** the model that we will use in the future

Which metric is the best one?

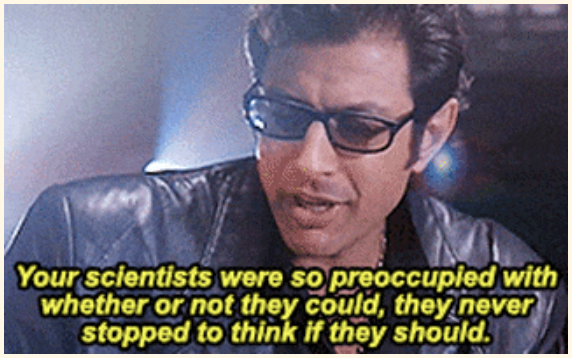
Which metric is the best one?

Regression: While there is no "objective" right answer, the **MSE** and the **R-squared** are popular choices.

Classification: This depends very much on **context**.

- How bad is it if we predict A, and the truth is B?
- How bad is it if we predict B, and the truth is A?

How do I compute individual metrics?


Your scientists were so preoccupied with whether or not they could, they never stopped to think if they should.

How do I compute individual metrics?

The only time it makes sense to report non-cross-validated metrics is if you have new data (or a "validation set").

Each metric has its own function, e.g. `roc_auc()`.

This function requires:

1. The dataset
2. The true classes
3. Either the predicted classes or the predicted probabilities of Class 1

(What is Class 1? The "first" one in the factor, usually alphabetical.)

How do I compute individual metrics on a validation set?

```
logit_mod <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

ins_recipe <- recipe(smoker ~ charges,
                     data = ins)

logit_wflow <- workflow() %>%
  add_recipe(ins_recipe) %>%
  add_model(logit_mod)

logit_fit <- logit_wflow %>%
  fit(ins_new)
```

How do I compute individual metrics on a validation set?

```
predict_classes <- predict(logit_fit, ins_new)
predict_probs <- predict(logit_fit, ins_new,
                         type = "prob")

ins_new <- ins_new %>%
  mutate(
    smoker_predicted = predict_classes$.pred_class,
    smoker_prob_no = predict_probs$.pred_no,
    smoker_prob_yes = predict_probs$.pred_yes
  )
```

```
## # A tibble: 143 x 4
##    smoker smoker_predicted smoker_prob_no smoker_prob_yes
##    <fct>  <fct>                     <dbl>           <dbl>
##  1 no     no                        0.996         0.00362
##  2 no     no                        0.956         0.0443
##  3 no     no                        0.997         0.00296
##  4 no     no                        0.876         0.124
##  5 no     no                        0.896         0.104
##  6 no     no                        0.997         0.00341
##  7 no     no                        0.923         0.0770
##  8 no     no                        0.996         0.00404
##  9 no     no                        0.988         0.0118
## 10 no     no                        0.979         0.0215
## # ... with 133 more rows
```

How do I compute individual metrics on a validation set?

```
ins_new %>%
  accuracy(truth = smoker,
           estimate = smoker_predicted)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.902
```

```
ins_new %>%
  roc_auc(truth = smoker,
          smoker_prob_no)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.977
```

```
ins_new %>%
  roc_auc(truth = smoker,
```

What's the deal with ROC anyways?

What's the deal with ROC anyways?

Our model predictions give us the probability of each observation belonging to each category.

By default, we'll predict that each observation belongs to the category with the highest probability.

(i.e., in this case, the cutoff is 0.5)

What's the deal with ROC anyways?

```
ins_new <- ins_new %>%
  mutate(
    cutoff_50 = ifelse(smoker_prob_yes > 0.5, "yes", "no"),
    cutoff_50 = factor(cutoff_50)
  )
```

```
## # A tibble: 143 x 5
##    smoker smoker_predicted cutoff_50 smoker_prob_no smoker_prob_yes
##    <fct>  <fct>            <fct>               <dbl>           <dbl>
##  1 no     no               no                  0.996         0.00362
##  2 no     no               no                  0.956         0.0443
##  3 no     no               no                  0.997         0.00296
##  4 no     no               no                  0.876         0.124
##  5 no     no               no                  0.896         0.104
##  6 no     no               no                  0.997         0.00341
##  7 no     no               no                  0.923         0.0770
##  8 no     no               no                  0.996         0.00404
##  9 no     no               no                  0.988         0.0118
## 10 no     no               no                  0.979         0.0215
## # ... with 133 more rows
```

What's the deal with ROC anyways?

```
ins_new %>%
  sensitivity(truth = smoker,
              estimate = smoker_predicted)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    binary         0.948
```

```
ins_new %>%
  specificity(truth = smoker,
              estimate = smoker_predicted)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 spec    binary         0.714
```

```
ins_new %>%
  sensitivity(truth = smoker,
              estimate = cutoff_50)
```

What's the deal with ROC anyways?

But what if we used a different number as our cutoff?

For example, maybe we really want to prioritize specificity: that is, we want to make sure that if we say someone is a smoker, they really are.

```
ins_new <- ins_new %>%
  mutate(
    cutoff_75 = ifelse(smoker_prob_yes > 0.75, "yes", "no"),
    cutoff_75 = factor(cutoff_75)
  )
```

```
## # A tibble: 143 x 5
##    smoker smoker_predicted cutoff_50 smoker_prob_no smoker_prob_yes
##    <fct>  <fct>            <fct>               <dbl>           <dbl>
##  1 no     no               no                  0.996         0.00362
##  2 no     no               no                  0.956         0.0443
##  3 no     no               no                  0.997         0.00296
##  4 no     no               no                  0.876         0.124
##  5 no     no               no                  0.896         0.104
##  6 no     no               no                  0.997         0.00341
##  7 no     no               no                  0.923         0.0770
##  8 no     no               no                  0.996         0.00404
##  9 no     no               no                  0.988         0.0118
## 10 no     no               no                  0.979         0.0215
## # ... with 133 more rows
```

What's the deal with ROC anyways?

```
ins_new %>%
  sensitivity(truth = smoker,
              estimate = smoker_predicted)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    binary         0.948
```

```
ins_new %>%
  specificity(truth = smoker,
              estimate = smoker_predicted)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 spec    binary         0.714
```

```
ins_new %>%
  sensitivity(truth = smoker,
              estimate = cutoff_75)
```

What's the deal with ROC anyways?

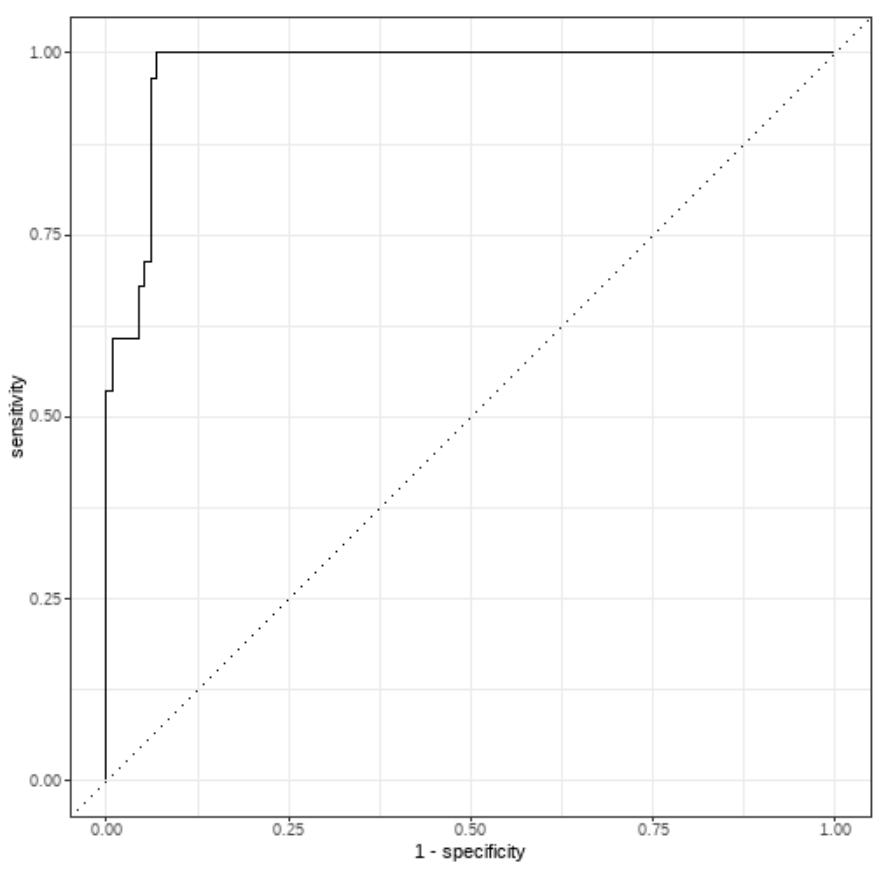
Now, imagine if we did that for all possible cutoffs between 0 and 1.

For each cutoff, we'd have a sensitivity and specificity pair.

Let's plot those pairs!

```
ins_new %>%
  roc_curve(truth = smoker,
            smoker_prob_yes,
            event_level = "second")
```

What's the deal with ROC anyways?



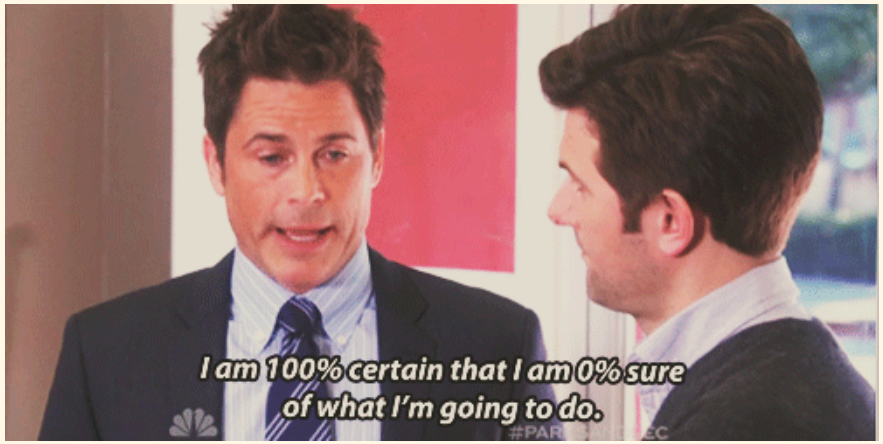
What's the deal with ROC anyways?

A good classifie *can achieve high specificity* and **high sensitivity*.

A good classifier doesn't change its answers too much when the cutoff changes.

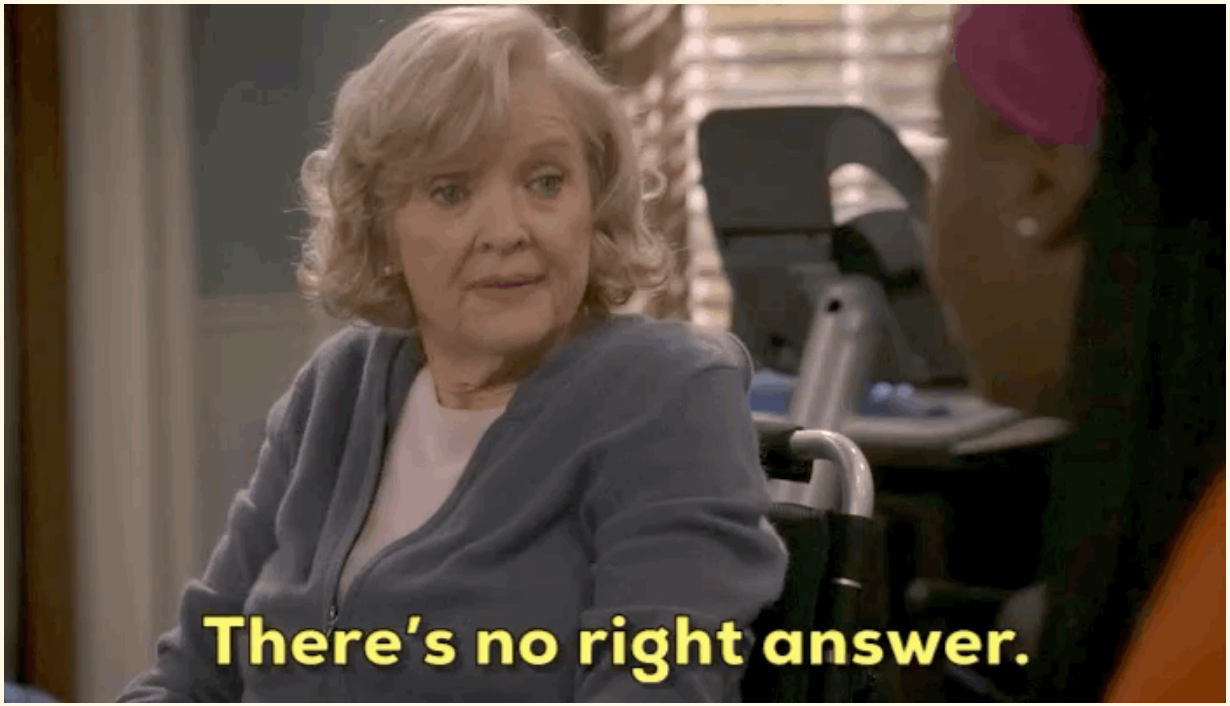
If your probabilities are close to 0 or 1, that's good.

If your probabilities are close to 0.5, you're just guessing.


I am 100% certain that I am 0% sure
of what I'm going to do.

How the heck do we know which models to try?


THIS IS TOO MUCH


There's no right answer.

Some ideas...

- Plot your potential predictors against your response variable. Start by using just the ones that appear to have some association.
- Backwards selection: Begin with all your predictors in the model, and drop one at a time to see if the model improves.
- Forwards selection: Begin with just one predictor in the model, and add in one at a time to see if the model improves. (*I like this one, personally*)
- Every subset selection: Try every possible combination of predictors that exists. Yikes.

Regarding Pre-Processing

You *can* just guess-and-check to see which transformations help...

... but ideally, you'd choose your pre-processing for a reason.

In KNN, we usually normalize everything, so that the predictors are on the same scale.

In regression, we often do log transformations or square root transformation of data that is skewed, to match the model assumptions.

Maybe you have some domain knowledge of the data, that leads you to a certain choice of pre-processing.

From now on...

From now on:

- If I say fit a classification model, it is up to you to decide which model types (KNN, Logistic, etc) to try.
- If I say choose the best model, it is up to you to decide which metric you are using and justify it.
- If I do not explicitly say to consider variable transformations, you should still consider variable transformations.
- If I say report your final model, you should also report relevant, cross-validated metrics.

Final Advice:

(skip the math!)