Predictive Modeling
Week 14

PH 700A, Spring 2025

Rick Calvo

Table of contents

1 Introduction to Predictive Modeling

1.1 Course Evaluations

ATTENTION STUDENTS:

The SDSU School of Public Health needs your feedback!

April 21-May 2 are the School of Public Health's (SPH) Assessment Weeks

For accreditation and strategic planning, SPH is required to survey its students on an annual basis. We need to hear from all students about ways to continue to improve the school.

All responses will be confidential.

- **Students will be e-mailed 1-2 brief surveys between April 21-May 2**, with 2 completion reminders
  - All students will receive the SPH Student Survey (hint: search the word "survey" in your email to find it)
  - Graduating students will also receive the Graduating Student Survey

- **Done with your surveys? Take a moment to create or update your LinkedIn profile! Graduating? Please also join our Alumni email list.**
  - Join our LinkedIn SPH Alumni group by searching "SDSU SPH Alumni" in LinkedIn or scanning the QR Code below
  - SIgn up for our alumni email list using ***your personal email address***: https://publichealth.sdsu.edu/alumni/mailing-list or with QR code

Join SPH Alumni LinkedIn Group --->

Join SPH Alumni Email List --->

SDSU

Questions?
eschmied@sdsu.edu

1.2 Packages

Classification and Regression Training

- `library(caret)`

Tidymodels Framework for Rapid Model Development

- `library(tidymodels)`
  - `library(rsample)`
  - `library(parsnip)`
  - `library(yardstick)`
  - `library(recipes)`

Tidymodels Parameter Tuning

- `library(tune)`

Post-processing of Model Predictions

- `library(probably)`

R Package Development Tools

- `library(devtools)`

User-Written Package for Supervised Variable Selection

- `library(colino)`
  - Install with: `devtools::install_github("stevenpawley/colino")`

1.3 Background

- *Predictions* are statements about the future
- **Predictive modeling** is the process of using input data to calculate probabilities for future events
- **Predictive analytics** is a framework that involves the perpetual engineering of data, evaluation of significance, and generation of predictions across a variety of disciplines
- Types of predictive models:
  - Forecasting
  - Classification

1.4 Ubiquity of Predictive Models

- Framingham Risk Score for CHD
- APACHE II Score for ICU Mortality
- 5- and 10-minute APGAR Scores and Cerebal Palsy
- Bone Mineral Density Test for 10-year Fracture Risk
- Breast Cancer Risk Assessment Tool

And pretty much any other clinical outcome…

Visit https://www.mdcalc.com/ for other examples.

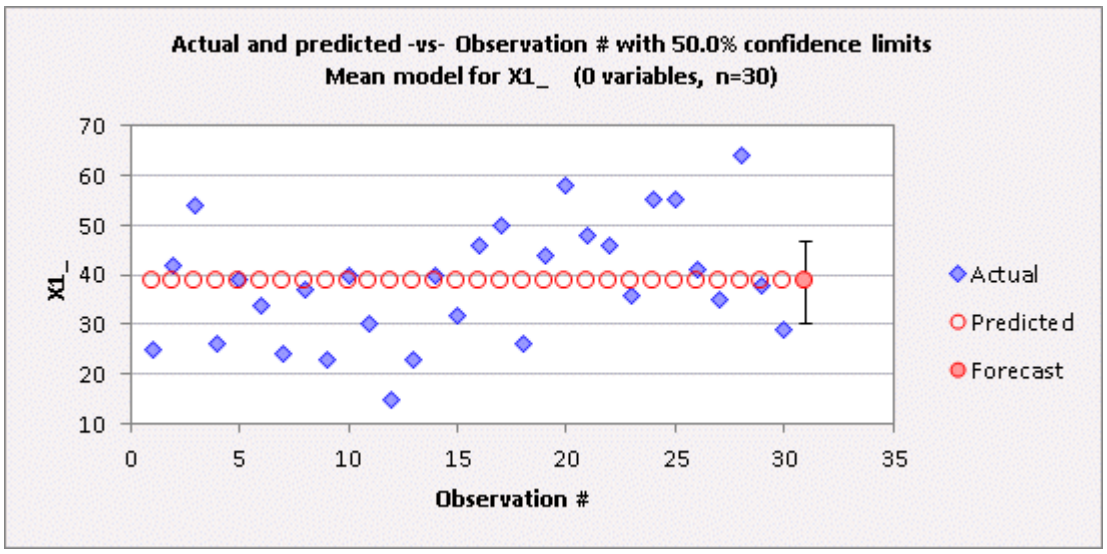
1.5 Method Categories

- Any statistical process can be used to make a *predictions*!
- True *predictive models* have unique methods added

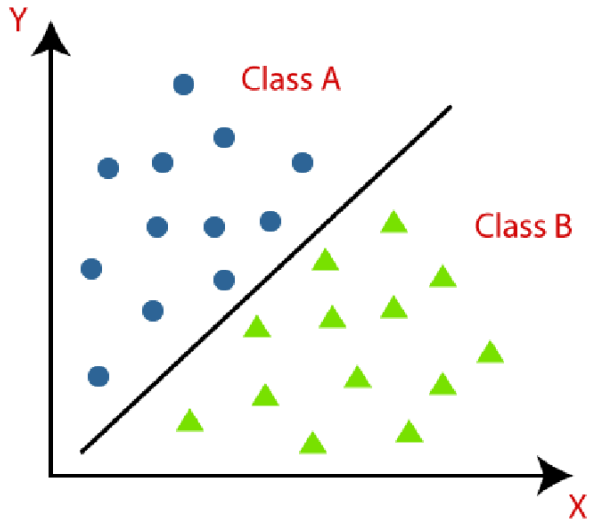
Method	Purpose	Type
Linear Regression	Regression	Parametric
Logistic Regression	Classification	Parametric
Decision Tree	Both	Non-Parametric
Random Forest	Both	Non-Parametric
Support Vector Machine (SVM)	Both	Non-Parametric
k-Nearest Neighbors (k-NN)	Both	Non-Parametric
Naive Bayes	Classification	Parametric
Neural Networks	Both	Parametric
Gradient Boosting Machines	Both	Non-Parametric
Gaussian Processes	Both	Non-Parametric
Lasso / Ridge Regression	Regression	Parametric
Cox Proportional Hazards Model	Classification	Semi-Parametric
Kaplan-Meier Estimator	Both	Non-Parametric
Accelerated Failure Time (AFT)	Both	Parametric
Random Survival Forest	Classification	Non-Parametric
DeepSurv (Neural Net Survival)	Both	Parametric

1.6 Regression vs. Classification

Regression is **forecasting**.



Classification is **labeling**.



1.7 Focus on Classification

Logistic Regression

$$ln(\frac{p_{event}}{p_{1-event}}) = \beta_0 + \beta_1(x_1) + \beta_2(x_2) + ... + \beta_i(x_i)$$

Cox Regression

$$h(t) = h_0(t) * e^{\beta_1(x_1)+\beta_2(x_2)+...+\beta_i(x_i)}$$

Given a set of inputs (i.e. the x_i variables), calculate the relative contributions of each to determine their event status.

In real terms:

Given a *set of clinical factors*, generate a *prediction for a disease* that you can apply to real patients.

1.8 Practical Assumptions

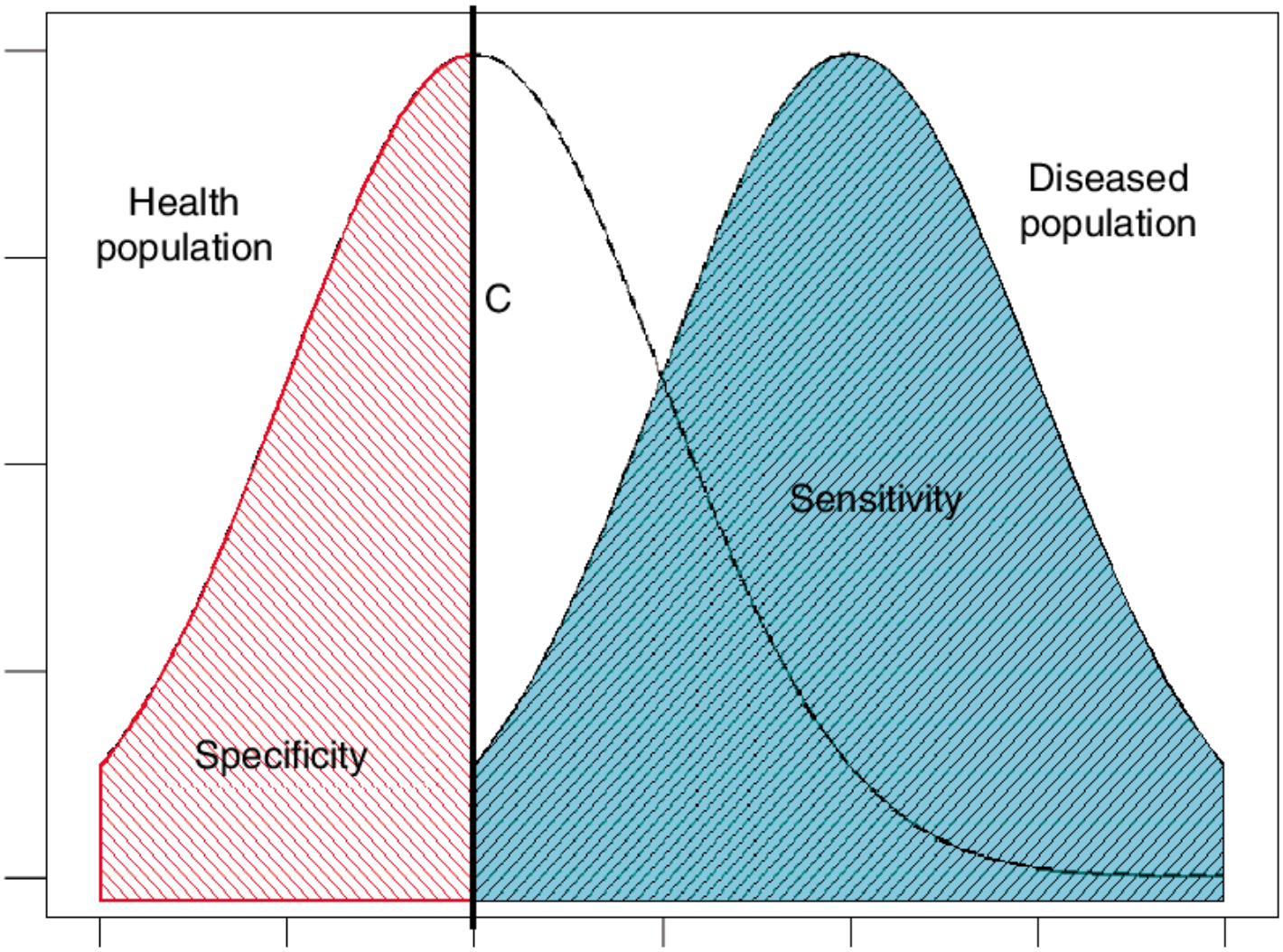
- The *outcome* represents a true public health threat
- *Advance warning* can mitigate health effects and consequences
- There is benefit in isolating a "best set" of risk factors
- Well-defined "at-risk" study population
- The model is feasible and easy to apply

1.9 Methodological Concerns

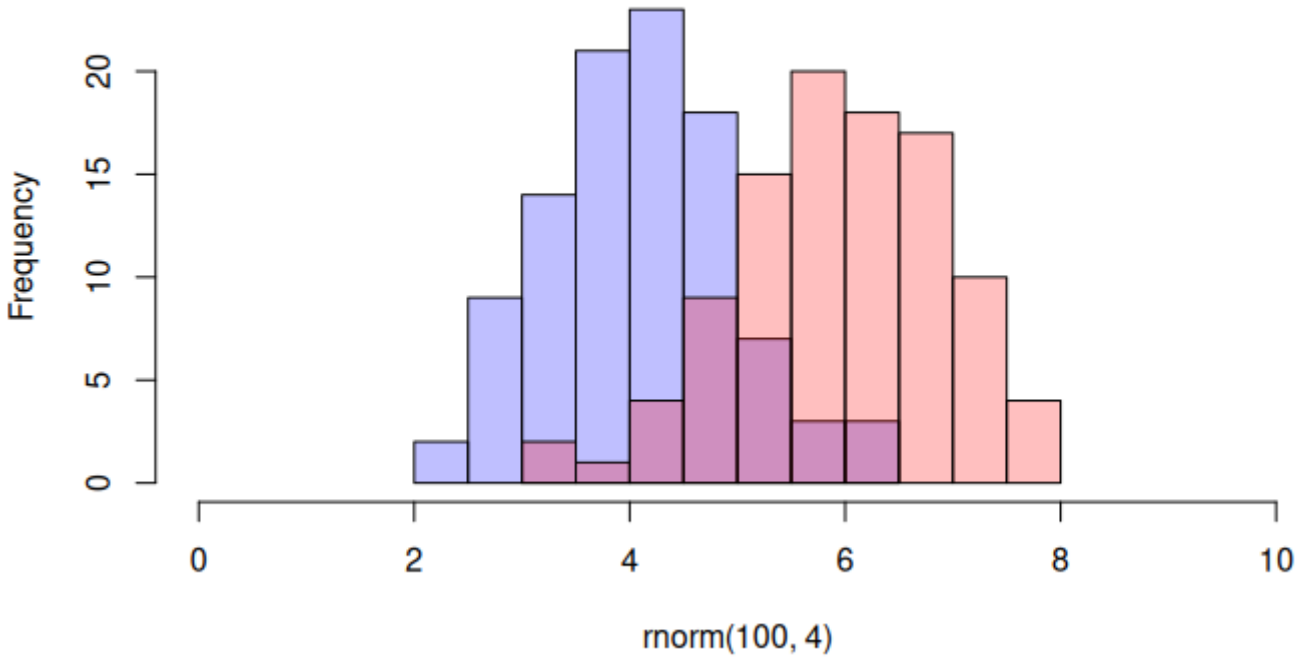
- *Predicted outcome probabilities* can only be between 0% and 100%
- Predictions must be *valid* and *generalizable*!
- Changes in data source or sample should still yield reasonable results
- There is no "perfect" model; errors and accuracy will vary
- Risk factors have been thoroughly researched
- Risk factors can be *accurately measured*

1.10 Measures of Interest

- Accuracy/Performance
  - Sensitivity
  - Specificity
- Calibration
  - Model Fitness
  - Tuning


Health
population
C
Diseased
population
Sensitivity
Specificity

1.11 Predicted Value Thresholds



1.12 On Accuracy and Calibration

Avoid data-driven *circular reasoning*!


It is time you go to bed
Because it is your bedtime
Why?

- Models that are created with a dataset usually *perform* well in that same dataset
- Similarly, variables selected for a model in a dataset already fit the data
- Independent samples for model development and *evaluation* avoids the circular logic

1.13 Option 1: Creating Data Subsamples

Before model development:

- Split source data into subsets
- At least two subsets are needed:

1. **Training Set** (aka **Development Set**)
2. **Validation Set**

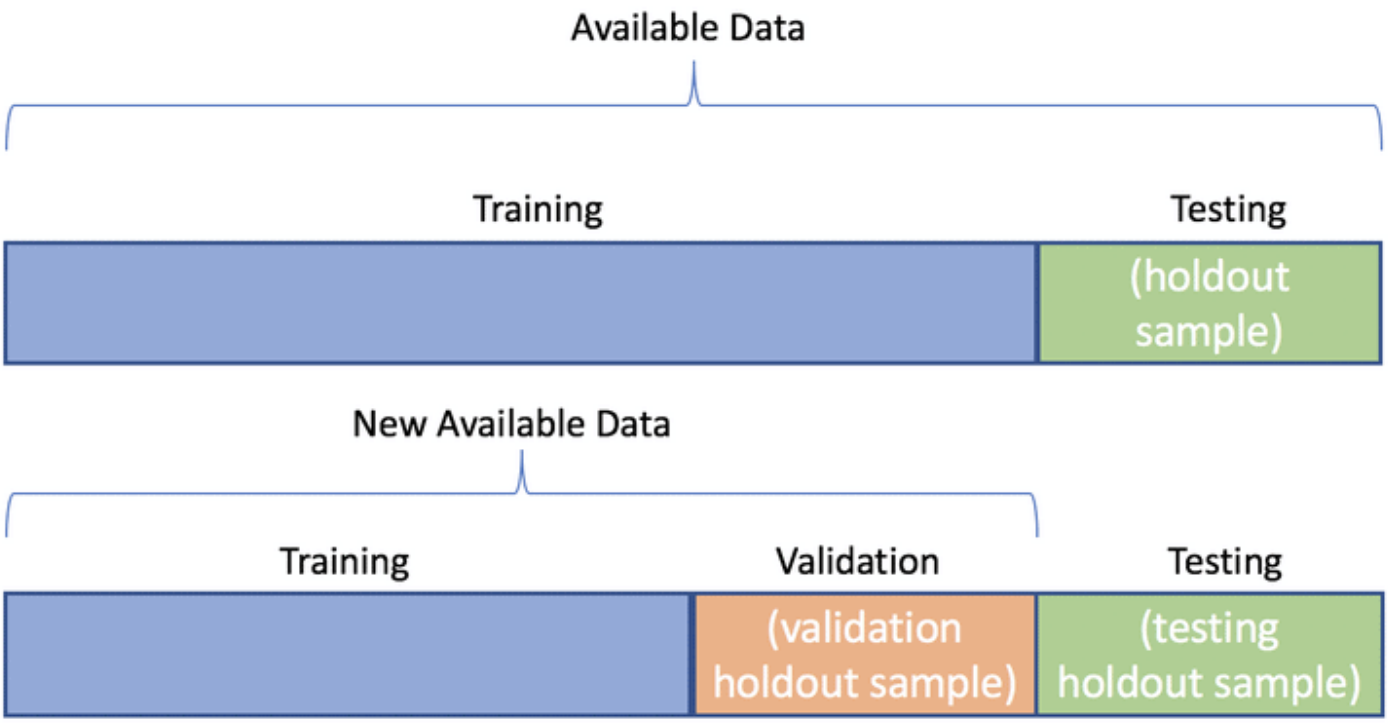
A separate, isolated, or independent dataset can also be retained to further test the compiled model.

3. **Test Set**

The test set usually contains a more heterogeneous sample than the development/validation sets.

1.14 Subsample Purposes

Training/Development	Validation	Test
Sample characterization	Parameter tuning	Alternate sample statistics
Variable selection	*Model performance*	*Final performance*
Model development	*Internal validity*	*External validity*
Model fitness	Scoring fitness evaluation	
Score creation		



1.15 Option 2: Cross-validation

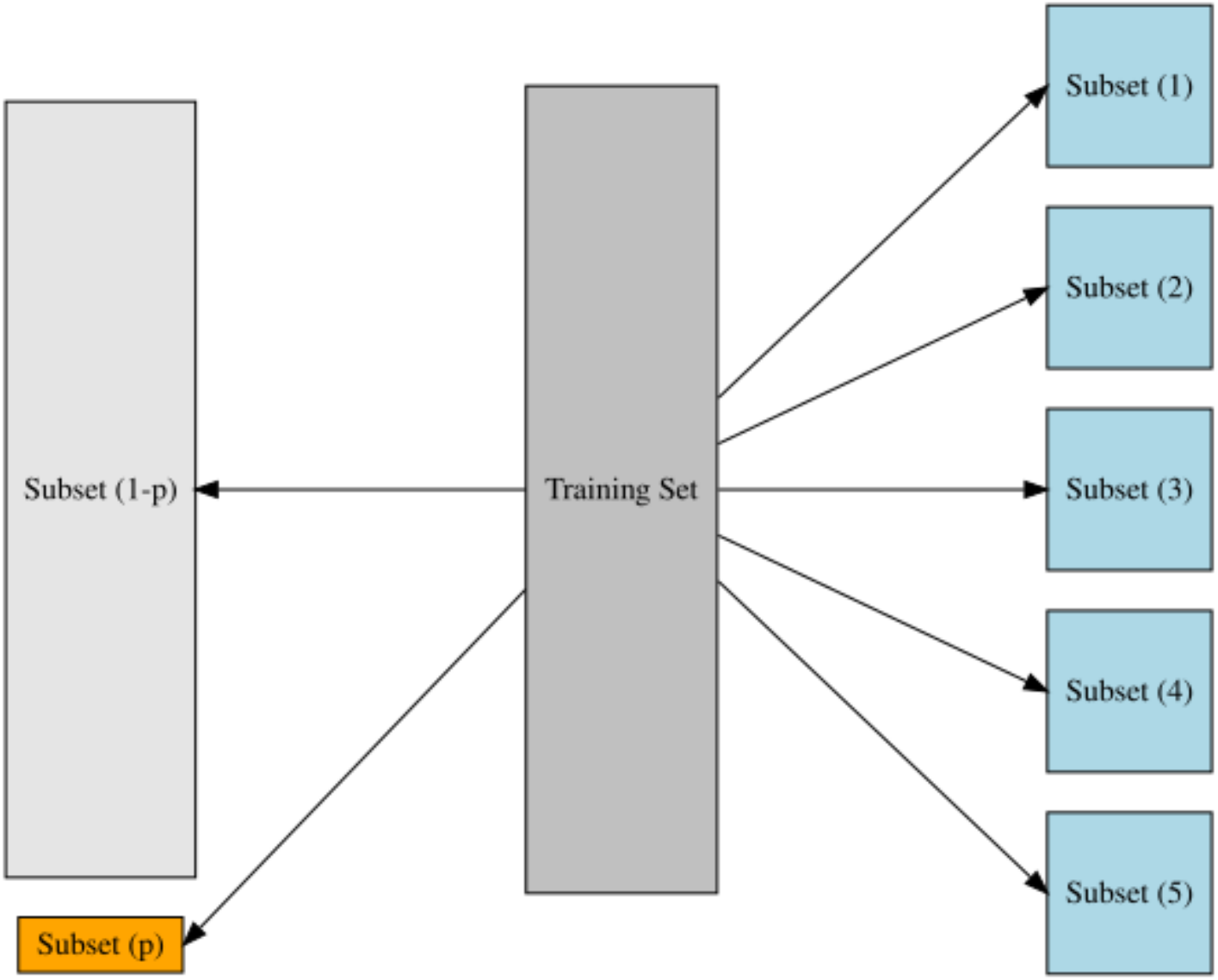
An alternative to manual creation of training and validation sets.

Cross-validation evaluates the average *performance* of a model when different subsets are hidden

Different variable combinations can be evaluated to maximize average performance.

Leave-p-*out Cross-validation*

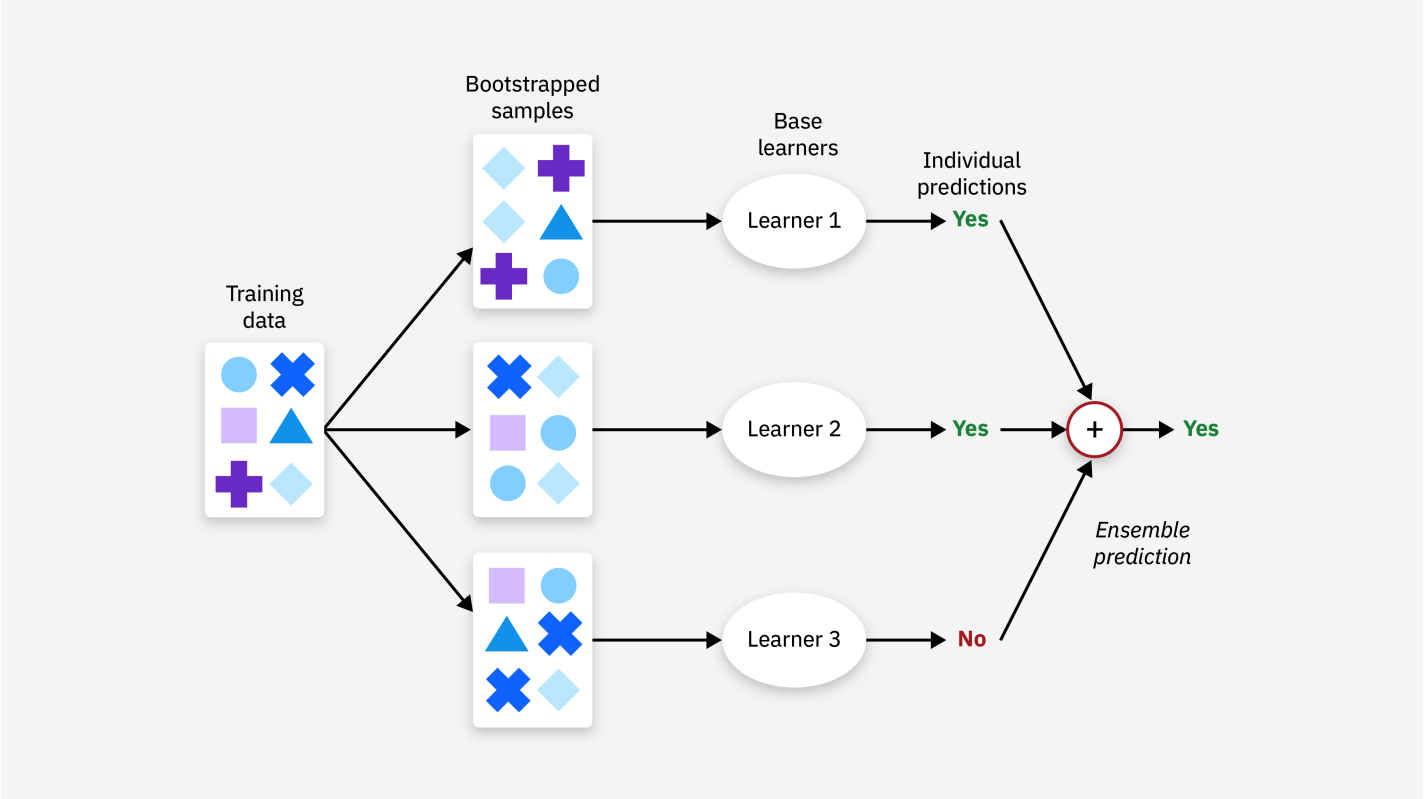
k-fold Cross-validation



Bagging

- *Bagging* stands for "Bootstrap-Aggregation"
- Resembles *Leave*-p-*out* cross-validation in subsample generation ONLY
- Can be manually performed but typically executed as a *machine learning algorithm*
- *Ensemble learning* strategy for model development
  - Combines different model types to generate predictions (i.e. Cox Model and Logistic Model)
  - Variable coefficients and predictions are averaged across all models

- Focuses on an averaged model that minimizes variance in predictions



1.16 Measuring (Classifier) Performance

- Area Under the Receiver Operating Characteristic (AUROC) curve
- Harrell's Concordance Statistic

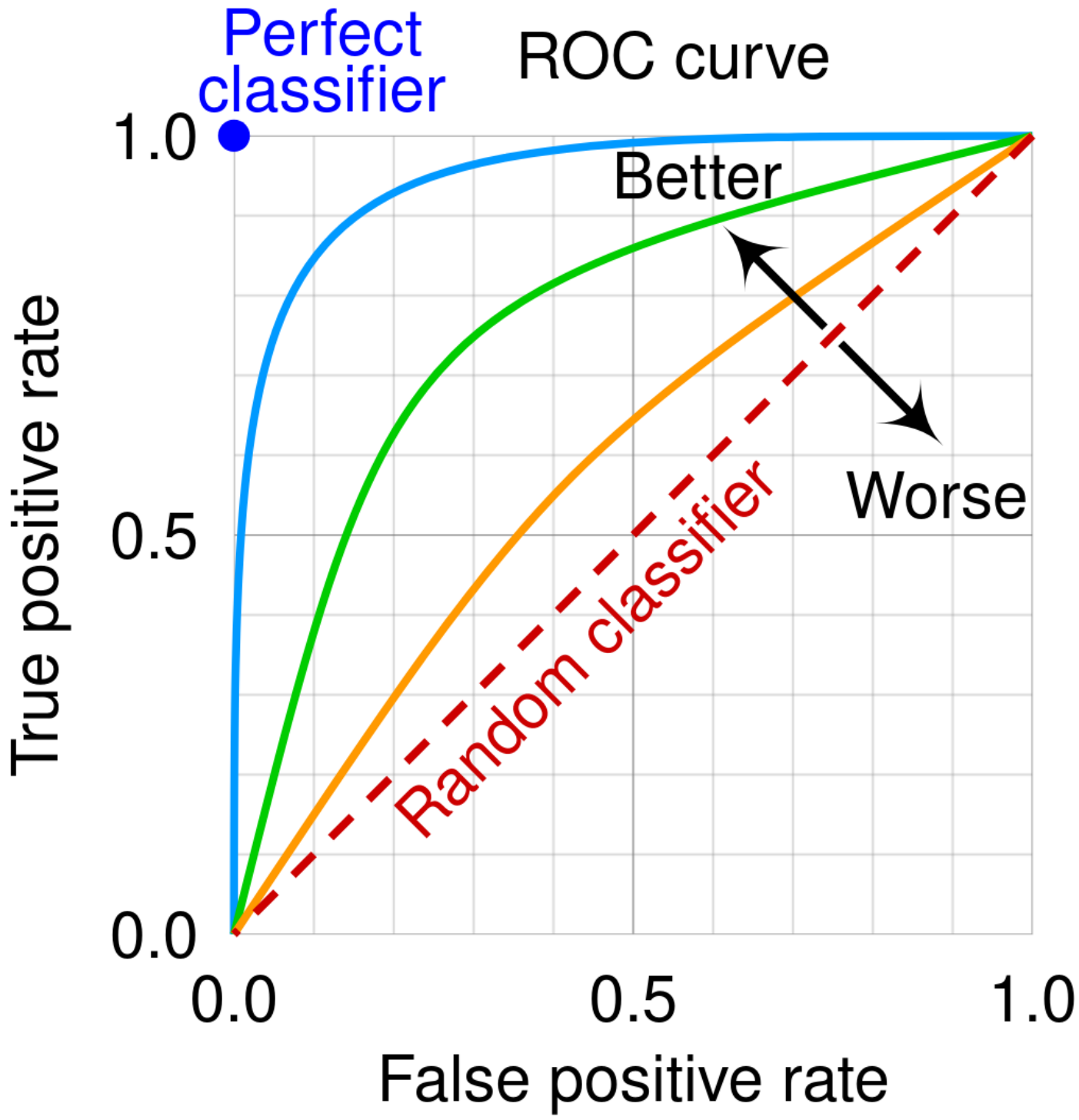


Figure 1: Y-axis is Sensitivity; X-axis is 1-Specificity

Ranges from 0-100%, but:

- 0-50% is negative classification
- 50-100% is positive classification

- 50% represents a random “coin flip” probability

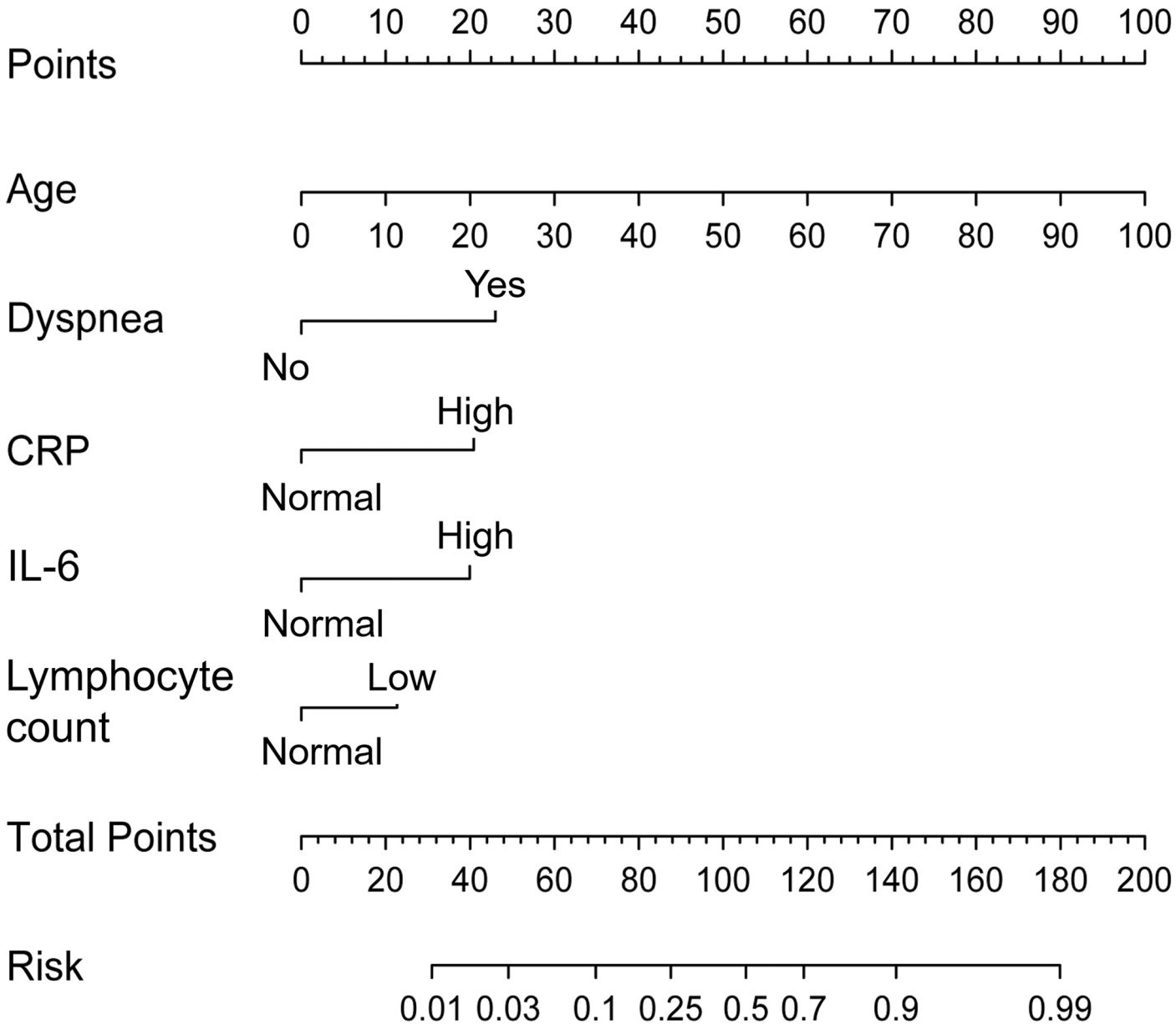
1.17 Translation

Conversion of the model results into a practical system.

Point-based Scoring

Variable	β Estimate	Score
diabetes	0.5	1
depression	1.0	2
anxiety	1.8	3.6
smoking	2.5	5

Nomogram



1.18 Calibration

Calibration is fitness!

Most common method: Hosmer-Lemeshow Goodness-of-Fit

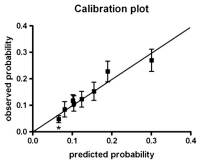


Table 4: Assumes 500 events in study sample

Risk Quantile	Observed	Expected
0 - 10%	4	5
11 - 20%	20	15
21 - 30%	29	25
31 - 40%	32	35
41 - 50%	41	45
51 - 60%	52	55
61 - 70%	63	65
71 - 80%	74	75
81 - 90%	86	85
91 - 100%	99	95

1.19 Caveats & Considerations

Monotonicity - Risk factors must affect risk in only one direction.

Independence - Observations (and their measurements) must not be associated with each other.

Orthogonality - Risk factors must not be significantly correlated to each other.

Parsimony - Predictive models should not be overly complex and cumbersome to use.

Main considerations before using R:

- What is your outcome of interest?
- Do you have time-to-event?
- What type of independent variables do you have access to?
- Variable selection strategy?
- How will you calculate accuracy?

1.20 Subsetting Data Commands

```
library(rsample)

set.seed(20250430)

var_subselection <- df %>%
  select(stay_id, gender, disposition, alzheimer, trauma, visual, edloshrs, temperature:acuity, card

subsets.tagged <- initial_validation_split(var_subselection, prop = c(0.5, 0.25))

train <- training(subsets.tagged)
```

```
valid <- validation(subsets.tagged)
test <- testing(subsets.tagged)
```

`set.seed()` establishes the psuedo-random number generation when allocating participants to data subsets

`initial_validation_split()` tags observations to be distributed to each of the subsets:

- `training()` allocates tagged observations for the *Training Set* (50% of `df`)
- `validation()` allocates tagged observations for the *Validation Set* (25% of `df`)
- `testing()` allocates the remaining observations for the *Testing Set* (100% - 50% - 25% = 25% of `df`)

1.21 Using Tidymodels

```
library(tidymodels)
show_engines(x)
```

`tidymodels` framework contains several additional subpackages

`show_engines()` provides you with a list of model types and categories from the `parsnip` subpackage

Options for `x` include:

Model Function	Mode(s)
`linear_reg`	regression
`logistic_reg`	classification
`decision_tree`	classification, regression
`rand_forest`	classification, regression
`boost_tree`	classification, regression
`svm_linear`	classification, regression
`svm_rbf`	classification, regression
`svm_poly`	classification, regression
`nearest_neighbor`	classification, regression
`mlp`	classification, regression
`mars`	regression
`naive_Bayes`	classification
`multinom_reg`	classification
`poisson_reg`	regression
`proportional_hazards`	censored regression
`surv_reg`	censored regression

1.22 Model Selection

```
library(tidymodels)
```

```
-- Attaching packages ------------------------------------- tidymodels 1.3.0 --

v broom        1.0.8     v recipes      1.3.0
v dials        1.4.0     v tune         1.3.0
v infer        1.0.8     v workflows    1.2.0
v modeldata    1.4.0     v workflowsets 1.1.0
v parsnip      1.3.1     v yardstick    1.3.2

-- Conflicts ---------------------------------------- tidymodels_conflicts() --
x dplyr::between()   masks data.table::between()
x scales::discard()  masks purrr::discard()
x dplyr::filter()    masks stats::filter()
x dplyr::first()     masks data.table::first()
x recipes::fixed()   masks stringr::fixed()
x dplyr::lag()       masks stats::lag()
x dplyr::last()      masks data.table::last()
x yardstick::spec()  masks readr::spec()
x recipes::step()    masks stats::step()
x purrr::transpose() masks data.table::transpose()
```

```
show_engines("logistic_reg")
```

```
# A tibble: 7 x 2
  engine    mode
  <chr>     <chr>
1 glm       classification
2 glmnet    classification
3 LiblineaR classification
4 spark     classification
5 keras     classification
6 stan      classification
7 brulee    classification
```

1.23 Outcome Assessment

Model development must be performed in the training set.

Important to check outcome frequency to verify statistical power.

```
train %>%
  count(disposition) %>%
    mutate(dispPerc = n/sum(n))
```

```
   disposition     n  dispPerc
        <fctr> <int>     <num>
1:    ADMITTED 35484 0.5417735
2:        HOME 30012 0.4582265
```

In the training set, 54.1% were admitted and 45.8% were sent home.

1.24 Specifying Models

```
logit_model <- logistic_reg(mode = "classification") %>% set_engine("glm")

model.null <- logit_model %>% fit(disposition ~ 1, data = train)
```

`logit_model` is an object that holds the parameters of the model you want to use

- `logistic_reg()` is the selected model

- `mode = "classification"` sets the modeling type

- `set_engine("glm")` tells R to use the `glm()` function to calculate the model

`model.null` stores the model output based on the parameters provided

- `logit_model` is the previously defined object

- `fit()` directs R to the variables and data frame used for model calculation

- `disposition ~ 1` is the standard formula notation for R, with a constant of 1 instead of covariat

1.25 Null Model Classification Accuracy

The `yardstick` package contains functions to measure accuracy.

Null model accuracy establishes a baseline for prediction.

1.25.1 Storing Predictions

```
library(yardstick)

pred.null <- train %>%
  select(disposition) %>%
  bind_cols(predict(model.null, new_data = train, type = "class") %>%
  rename(disposition_null = .pred_class)
  )
```

`pred.null` is a mini-data frame that will store the predicted probabilities from `model.null` and compare them to the observed events

`bind_cols()` adds columns from one object to another

- predict() calculates predicted probabilities based on parameters provided: a model, a data frame,

`rename()` is the standard `dplyr` command that changes the default variable name stub `.pred_class` to `disposition_null`

- `disposition_null` is the column name containing the predicted outcome from the predicted probabil

`accuracy()` is a function from `yardstick` that compares predictions to observations

- `data = pred.null` is the mini-data frame with predictions

- `truth =` tells R which column holds the "true" observed outcome

- `estimate =` tells R which column holds the predicted outcome

1.25.2 Confusion Matrix

```
library(yardstick)

accuracy(data = pred.null, truth = disposition, estimate = disposition_null)
```

```
# A tibble: 1 x 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.542
```

```
pred.null %>%
  conf_mat(truth = disposition, estimate = disposition_null)
```

```
          Truth
Prediction ADMITTED  HOME
  ADMITTED    35484 30012
  HOME            0     0
```

- `conf_mat()` displays the confusion matrix between truth and predicted

The null model does not contain any variables and predicts everyone will be "ADMITTED".

Per the accuracy measure, it is correct 53.9% of the time because 53.9% of the training data is "ADMITTED".

1.26 Model Development

Development can be manual or automated.

```
logit.m1 <- logit_model %>% fit(disposition ~ acuity, data = train)

summary(logit.m1$fit)
```

```
Call:
stats::glm(formula = disposition ~ acuity, family = stats::binomial,
    data = data)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.18235    0.04067 -29.073  < 2e-16 ***
acuity2      0.53913    0.04241  12.713  < 2e-16 ***
acuity3      1.50355    0.04230  35.542  < 2e-16 ***
acuity4      3.70261    0.11803  31.370  < 2e-16 ***
acuity5      4.54964    1.01710   4.473 7.71e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 90339  on 65495  degrees of freedom
Residual deviance: 84967  on 65491  degrees of freedom
AIC: 84977

Number of Fisher Scoring iterations: 5
```

```
logit.m2 <- logit_model %>% fit(disposition ~ acuity + gender, data = train)

summary(logit.m2$fit)
```

```
Call:
stats::glm(formula = disposition ~ acuity + gender, family = stats::binomial,
    data = data)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.19412    0.04147 -28.797  < 2e-16 ***
acuity2      0.53904    0.04241  12.711  < 2e-16 ***
acuity3      1.50484    0.04231  35.564  < 2e-16 ***
acuity4      3.70327    0.11803  31.375  < 2e-16 ***
```

```
acuity5      4.54873    1.01710   4.472 7.74e-06 ***
genderM      0.02391    0.01640   1.458    0.145
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 90339  on 65495  degrees of freedom
Residual deviance: 84965  on 65490  degrees of freedom
AIC: 84977

Number of Fisher Scoring iterations: 5
```

1.27 Full Model Assessment

```
train$stay_id <- NULL

logit.full <- logit_model %>% fit(disposition ~ ., data = train)
```

```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(logit.full$fit)
```

```
Call:
stats::glm(formula = disposition ~ ., family = stats::binomial,
    data = data)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.136e+00  4.974e-01 -12.337  < 2e-16 ***
genderM      2.248e-02  1.685e-02   1.335 0.181991
alzheimer1  -5.862e-01  5.987e-02  -9.791  < 2e-16 ***
traumayes    1.697e+00  1.110e-01  15.281  < 2e-16 ***
visualyes    2.376e-01  1.203e-01   1.976 0.048209 *
edloshrs     5.000e-02  1.416e-03  35.302  < 2e-16 ***
temperature -9.906e-03  2.675e-03  -3.704 0.000212 ***
heartrate   -2.599e-03  4.971e-04  -5.229 1.71e-07 ***
resprate    -8.949e-02  3.781e-03 -23.670  < 2e-16 ***
o2sat        7.445e-02  4.177e-03  17.822  < 2e-16 ***
sbp          1.293e-03  3.519e-04   3.675 0.000237 ***
dbp          2.335e-04  8.458e-05   2.761 0.005761 **
acuity2      3.400e-01  4.417e-02   7.696 1.40e-14 ***
acuity3      1.293e+00  4.420e-02  29.248  < 2e-16 ***
```

```
acuity4      3.553e+00  1.197e-01  29.681  < 2e-16 ***
acuity5      4.367e+00  1.021e+00   4.278 1.89e-05 ***
cardiacyes   2.187e-01  4.783e-02   4.572 4.83e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 90339  on 65495  degrees of freedom
Residual deviance: 81646  on 65479  degrees of freedom
AIC: 81680

Number of Fisher Scoring iterations: 6
```

```
pred.full <- train %>%
  select(disposition) %>%
  bind_cols(predict(logit.full, new_data = train, type = "class") %>%
  rename(disposition_full = .pred_class)
  )

accuracy(data = pred.full, truth = disposition, estimate = disposition_full)
```

```
# A tibble: 1 x 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.652
```

```
pred.full %>%
  conf_mat(truth = disposition, estimate = disposition_full)
```

```
          Truth
Prediction ADMITTED  HOME
  ADMITTED    24597 11890
  HOME        10887 18122
```

Please note that `type = class` generates a binary predicted outcome at the 50% threshold.

1.28 ROC Curves

1.28.1 Training Set Predicted Probabilities

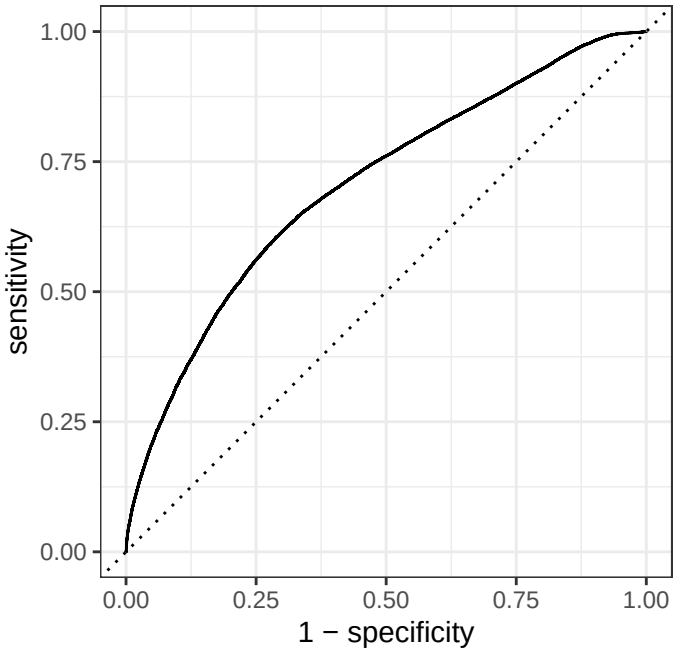
```
library(ggplot2)

# create another mini df to hold predictions
pred.fin <- logit.full %>%
    predict(new_data = train, type = "prob") %>%
      bind_cols(train)

pred.fin %>%
  roc_auc(truth = disposition, .pred_ADMITTED, event_level = "first")
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.704
```

```
pred.fin %>%
  roc_curve(disposition, .pred_ADMITTED, event_level = "first") %>%
    autoplot()
```



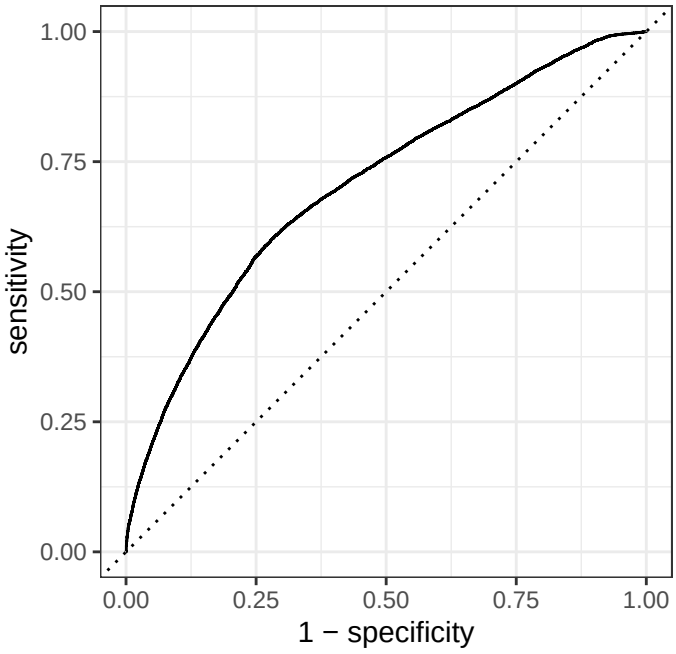
1.28.2 Validation Set Predictions

```
validation1 <- logit.full %>%
    predict(new_data = valid, type = "prob") %>%
      bind_cols(valid)
```

```
validation1 %>%
  roc_auc(disposition, .pred_ADMITTED, event_level = "first")
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.704
```

```
validation1 %>%
  roc_curve(disposition, .pred_ADMITTED, event_level = "first") %>%
    autoplot()
```



1.29 Scoring

```
summary(logit.full$fit)
```

```
Call:
stats::glm(formula = disposition ~ ., family = stats::binomial,
    data = data)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
```

```
(Intercept) -6.136e+00  4.974e-01 -12.337  < 2e-16 ***
genderM      2.248e-02  1.685e-02   1.335 0.181991
alzheimer1  -5.862e-01  5.987e-02  -9.791  < 2e-16 ***
traumayes    1.697e+00  1.110e-01  15.281  < 2e-16 ***
visualyes    2.376e-01  1.203e-01   1.976 0.048209 *
edloshrs     5.000e-02  1.416e-03  35.302  < 2e-16 ***
temperature -9.906e-03  2.675e-03  -3.704 0.000212 ***
heartrate   -2.599e-03  4.971e-04  -5.229 1.71e-07 ***
resprate    -8.949e-02  3.781e-03 -23.670  < 2e-16 ***
o2sat        7.445e-02  4.177e-03  17.822  < 2e-16 ***
sbp          1.293e-03  3.519e-04   3.675 0.000237 ***
dbp          2.335e-04  8.458e-05   2.761 0.005761 **
acuity2      3.400e-01  4.417e-02   7.696 1.40e-14 ***
acuity3      1.293e+00  4.420e-02  29.248  < 2e-16 ***
acuity4      3.553e+00  1.197e-01  29.681  < 2e-16 ***
acuity5      4.367e+00  1.021e+00   4.278 1.89e-05 ***
cardiacyes   2.187e-01  4.783e-02   4.572 4.83e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 90339  on 65495  degrees of freedom
Residual deviance: 81646  on 65479  degrees of freedom
AIC: 81680

Number of Fisher Scoring iterations: 6
```

```
library(gtsummary)
tbl_regression(logit.full, exponentiate = TRUE)
```

```
Extracting {parsnip} model fit with `tbl_regression(x = x$fit, ...)`

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

! `broom::tidy()` failed to tidy the model.

Profiled confidence intervals may take longer time to compute.
  Use `ci_method="wald"` for faster computation of CIs.

v `tidy_parameters()` used instead.

i Add `tidy_fun = broom.helpers::tidy_parameters` to quiet these messages.
```

Characteristic	OR	95% CI	p-value
gender			
F	—	—	
M	1.02	0.99, 1.06	0.2
alzheimer			
0	—	—	
1	0.56	0.49, 0.63	<0.001
trauma			
no	—	—	
yes	5.46	4.39, 6.78	<0.001
visual			
no	—	—	
yes	1.27	1.00, 1.61	0.048
edloshrs	1.05	1.05, 1.05	<0.001
temperature	0.99	0.98, 1.00	<0.001
heartrate	1.00	1.00, 1.00	<0.001
resprate	0.91	0.91, 0.92	<0.001
o2sat	1.08	1.07, 1.09	<0.001
sbp	1.00	1.00, 1.00	<0.001
dbp	1.00	1.00, 1.00	0.006
acuity			
1	—	—	
2	1.40	1.29, 1.53	<0.001
3	3.64	3.34, 3.97	<0.001
4	34.9	27.6, 44.1	<0.001
5	78.8	10.7, 583	<0.001
cardiac			
no	—	—	
yes	1.24	1.13, 1.37	<0.001

Abbreviations: CI = Confidence Interval, OR = Odds Ratio

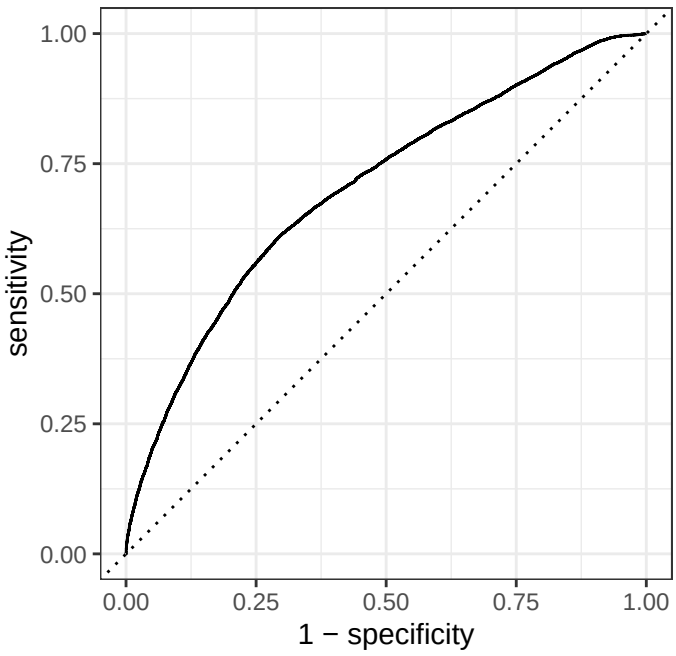
1.30 Apply to Test Set

```
test1 <- logit.full %>%
    predict(new_data = test, type = "prob") %>%
      bind_cols(test)

test1 %>%
  roc_auc(disposition, .pred_ADMITTED, event_level = "first")
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.702
```

```
test1 %>%
  roc_curve(disposition, .pred_ADMITTED, event_level = "first") %>%
    autoplot()
```



1.31 Evaluate Calibration

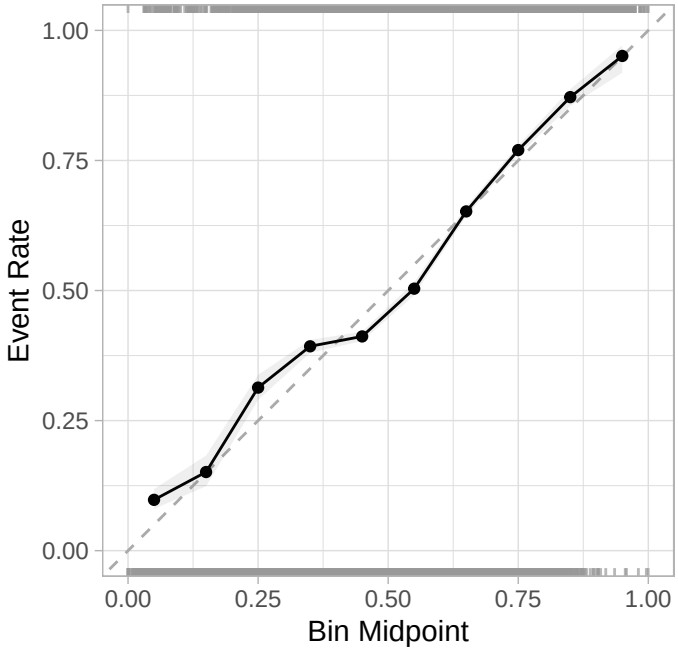
```
library(probably)
```

```
Attaching package: 'probably'
```

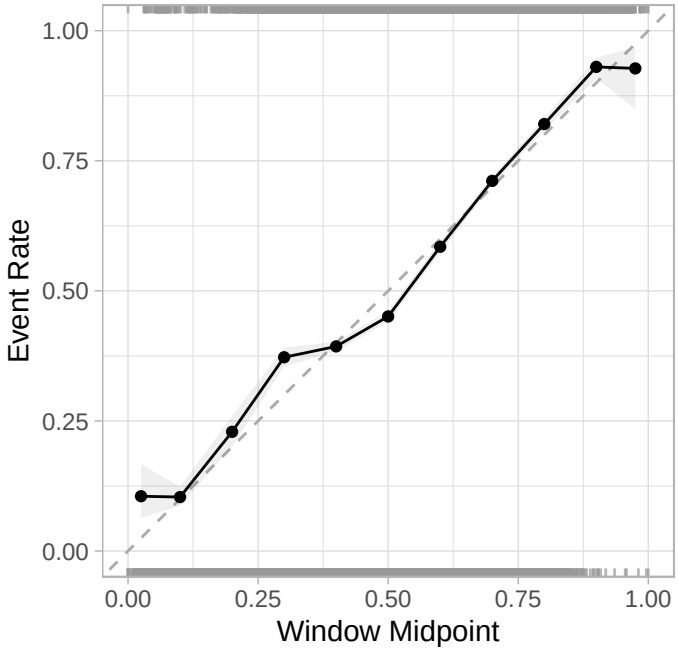
```
The following objects are masked from 'package:base':

    as.factor, as.ordered
```

```
test1 %>% cal_plot_breaks(truth = disposition, estimate = .pred_ADMITTED)
```




```
test1 %>% cal_plot_windowed(truth = disposition, estimate = .pred_ADMITTED, step_size = 0.1)
```



2 Appendix

2.1 References

- https://academic.oup.com/jamia/article/27/4/621/5762806
- https://www.tidyverse.org/blog/2022/11/model-calibration/
- https://www.tidymodels.org

3 Supplementary Functions

3.1 Data Pre-processing

With `tidymodels`, variables can be pre-processed using `recipes` to tell R which variables it should and should not use.

```
train <- training(subsets.tagged)
head(train)
```

```
    stay_id gender disposition alzheimer trauma visual  edloshrs temperature
     <fctr> <fctr>      <fctr>    <fctr> <fctr> <fctr>     <num>       <num>
1: 33258284      F    ADMITTED         0     no     no  4.216667        98.4
2: 32952584      F        HOME         0     no     no 13.500000        97.8
3: 39399961      F    ADMITTED         0     no     no  8.100000        98.7
4: 31021946      M        HOME         0     no     no  3.233333        97.4
5: 35420907      M    ADMITTED         0     no     no  4.033056        98.9
6: 35197384      F    ADMITTED         0     no     no  3.691111        97.8
   heartrate resprate o2sat   sbp   dbp acuity cardiac
       <num>    <int> <num> <int> <num> <fctr>  <fctr>
1:        70       16    97   106    63      3      no
2:        87       14    97    71    43      2      no
3:        77       16    98    96    50      2      no
4:        81       18   100   159    98      3      no
5:        92       18   100   140    77      2      no
6:       120       20    98   121    77      3      no
```

`stay_id` is a row identifier and R should know not to use it as a predictor.

```
library(recipes)

full_recipe <- recipe(disposition ~ .,
                      data = train) %>%
               update_role(stay_id, new_role = "ID")
```

- `recipe()` allows you to specify the data and variables in standard formula notation

  - The `.` is a wildcard that tells R to use all the variables in the data frame
- `update_role()` specifies the "role" of the `stay_id` variable as a non-predictor "ID"
- the updated role is only stored in the object `full_recipe` along with the formula

3.2 Stepwise Selection

Stepwise selection requires models to be pre-specified as `recipes` that R will iterate through.

Custom package `colino` contains many auxiliary functions for automated variable selection with `tidymodels`.

```
#| label: stepwise variable selection
#| include: true

library(colino)

# Recipe Creation
full.recipe <- train %>%
  recipe(disposition ~ .) %>%
  step_select_vip(all_predictors(),
                  threshold = 0.95,
                  outcome = "disposition",
                  model = logit_model)

# Prepared Variable List
full.recipe.vars <- prep(full.recipe)

stepwise.m1 <- logit_model %>% fit(disposition ~ ., juice(full.recipe.vars))

summary(stepwise.m1$fit)

pull_importances(stepwise.m1) %>% arrange(-importance)
```

- `step_select_vip()` from `colino` specifies the parameters for the stepwise selection process
- `all_predictors()` from `recipes` applies any *role* definitions before selecting candidate variables
- `threshold =` is the confidence interval-based selection process
  - alternatively, `top_p =` can be set to capture variables at a specific p threshold
- `prep()` applies the recipe with variable roles into an object
- `juice()` takes the prepared variables and iterates them through the specified model
- `pull_importances()` extracts the beta estimates from model objects
  - Shorthand to translate beta estimates to scores

3.3 Tuning and Cross-validation

We do not perform model tuning here because we did not assess any *hyperparameters*.

Tuning would be performed with the `library(tune)` package and cross-validation.

```
#| label: creating subsets
library(tune)

folds.defn <- vfold_cv(train, v = 5, repeats = 10)
```

- `vfold_cv` can be used to establish `v` number of cross-validation subsets and `repeats` number of iterations.

Average performance is generated across all the iterations and combinations of subsets.

- The model recipe must include the new folds definition object when fitting