Concepts in Machine Learning
Winter Institute in Data Science

Ryan T. Moore

2026-01-08

Building Models

Modeling Helper Functions

Example: `mtcars`

Example: Social Pressure Experiment (`recipes`)

Regularization Methods: LASSO, ridge regression, elastic nets

Building Models

How do we build models?

How do we build models?

What are our goals?

How do we build models?

What are our goals?

1. Generative modeling
2. Predictive modeling

How do we build models?

What are our goals?

1. Generative modeling
2. Predictive modeling

Breiman (2001b)

How do we build models?

How do we build models?

- Theory
  (novel theory, prior theory, prior findings)

How do we build models?

- Theory
  (novel theory, prior theory, prior findings)
- Raw data
  ("data look nonlinear, so $\ldots + \beta x^2 + \ldots$")

How do we build models?

- Theory
  (novel theory, prior theory, prior findings)
- Raw data
  ("data look nonlinear, so $\ldots + \beta x^2 + \ldots$")
- Specification searching
  (repeat modeling with same data)

How do we build models?

- Theory
  (novel theory, prior theory, prior findings)
- Raw data
  ("data look nonlinear, so $\ldots + \beta x^2 + \ldots$")
- Specification searching
  (repeat modeling with same data)
- Training and testing
  (repeat modeling, different data)

Which predictors should I include?

Which predictors should I include?

- All the important ones
  (No omitted variable bias)

Which predictors should I include?

- All the important ones
  (No omitted variable bias)
- No irrelevant ones
  (No included variable bias)

Which predictors should I include?

- All the important ones
  (No omitted variable bias)
- No irrelevant ones
  (No included variable bias)

Which predictors should I include?

- All the important ones
  (No omitted variable bias)
- No irrelevant ones
  (No included variable bias)

Helpful?

Which predictors should I include?

- All the important ones
  (No omitted variable bias)
- No irrelevant ones
  (No included variable bias)

Helpful?

- Affect outcome
- Confounders
- Pre-treatment only
- Avoid post-treatment
- “In-horizon”
- Test something “out-of-horizon”

Which predictors should I include?

- All the important ones
  (No omitted variable bias)
- No irrelevant ones
  (No included variable bias)

Helpful?

- Affect outcome
- Confounders
- Pre-treatment only
- Avoid post-treatment
- "In-horizon"
- Test something "out-of-horizon"

(Sometimes it will depend on goals.)

What to include, when thousands of predictors?

What to include, when thousands of predictors?

"Machine learning"

What to include, when thousands of predictors?

“Machine learning”

(but “machine learning” can mean different things.)

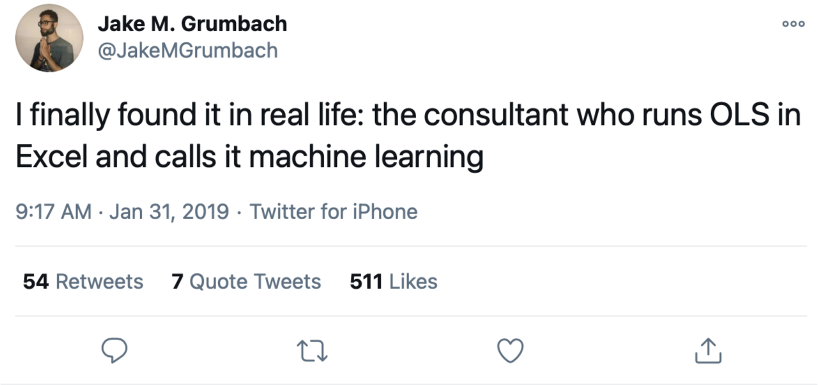


Figure 1: Don't do this.

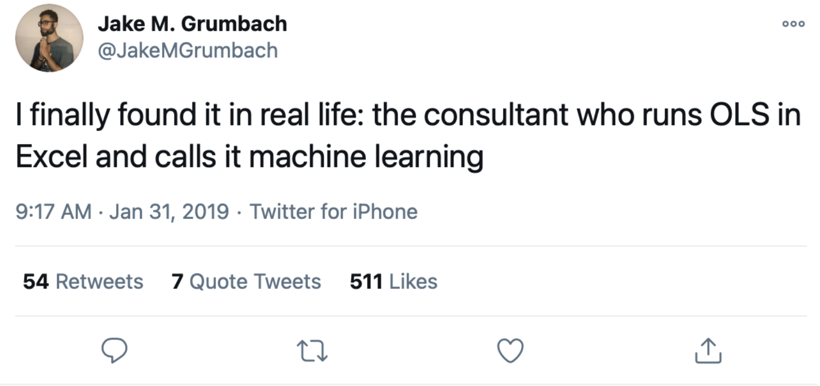


Figure 1: Don't do this.

If you can't describe the procedure's "learning", it may not be "machine learning".

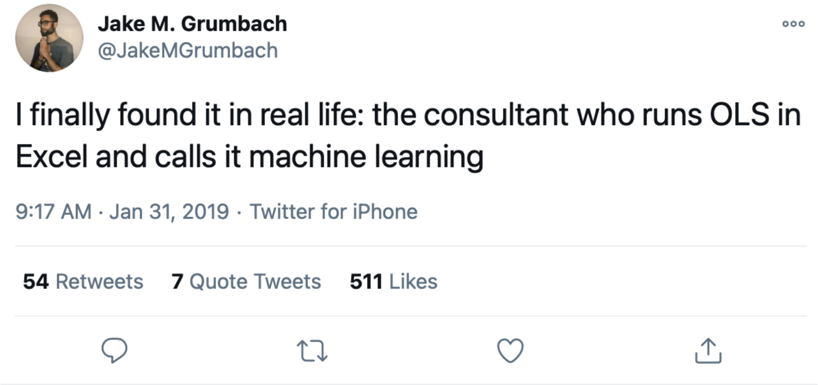


Figure 1: Don't do this.

If you can't describe the procedure's "learning", it may not be "machine learning".

There should probably be some testing/training, regularization, ...

Modeling Helper Functions

modelr Helper Functions

```r
data(sim1)

lm_out <- lm(y ~ x, data = sim1)

tidy(lm_out)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)     4.22     0.869      4.86 4.09e- 5
## 2 x               2.05     0.140     14.7  1.17e-14
```

modelr Helper Functions

```
glance(lm_out) |> select(1:5)
```

```
## # A tibble: 1 x 5
##   r.squared adj.r.squared sigma statistic  p.value
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl>
## 1     0.885         0.880  2.20      215. 1.17e-14
```

```
glance(lm_out) |> select(6:12)
```

```
## # A tibble: 1 x 7
##      df logLik   AIC   BIC deviance df.residual  nobs
##   <dbl>  <dbl> <dbl> <dbl>    <dbl>       <int> <int>
## 1     1  -65.2  136.  141.     136.          28    30
```

`modelr` Helper Functions

Special `mutate()` functions:

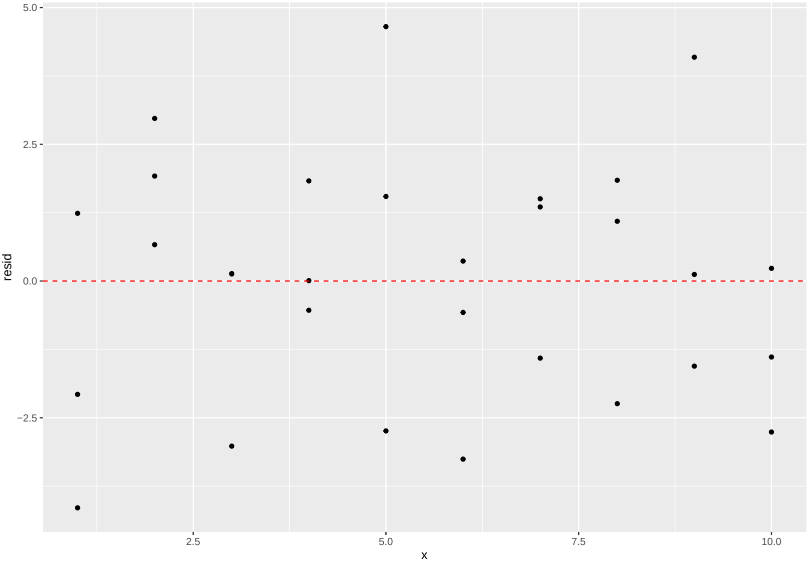
modelr Helper Functions

Special `mutate()` functions:

```
(sim1 <- sim1 |> add_residuals(lm_out))
```

```
## # A tibble: 30 x 3
##         x     y    resid
##     <int> <dbl>    <dbl>
##  1      1  4.20 -2.07
##  2      1  7.51  1.24
##  3      1  2.13 -4.15
##  4      2  8.99  0.665
##  5      2 10.2   1.92
##  6      2 11.3   2.97
##  7      3  7.36 -3.02
##  8      3 10.5   0.130
##  9      3 10.5   0.136
## 10      4 12.4   0.00763
## # i 20 more rows
```

```
ggplot(sim1, aes(x, resid)) + geom_point() +
  geom_hline(yintercept = 0, linetype = 2, color = "re
```



modelr Helper Functions

Special mutate() functions:

```
(sim1 <- sim1 |> add_predictions(lm_out))
```

```
## # A tibble: 30 x 4
##        x     y    resid  pred
##    <int> <dbl>    <dbl> <dbl>
##  1     1  4.20 -2.07     6.27
##  2     1  7.51  1.24     6.27
##  3     1  2.13 -4.15     6.27
##  4     2  8.99  0.665    8.32
##  5     2 10.2   1.92     8.32
##  6     2 11.3   2.97     8.32
##  7     3  7.36 -3.02    10.4
##  8     3 10.5   0.130   10.4
##  9     3 10.5   0.136   10.4
## 10     4 12.4   0.00763 12.4
## # i 20 more rows
```

modelr Helper Functions

```r
lm_out2 <- lm(y ~ x - 1, data = sim1)
```

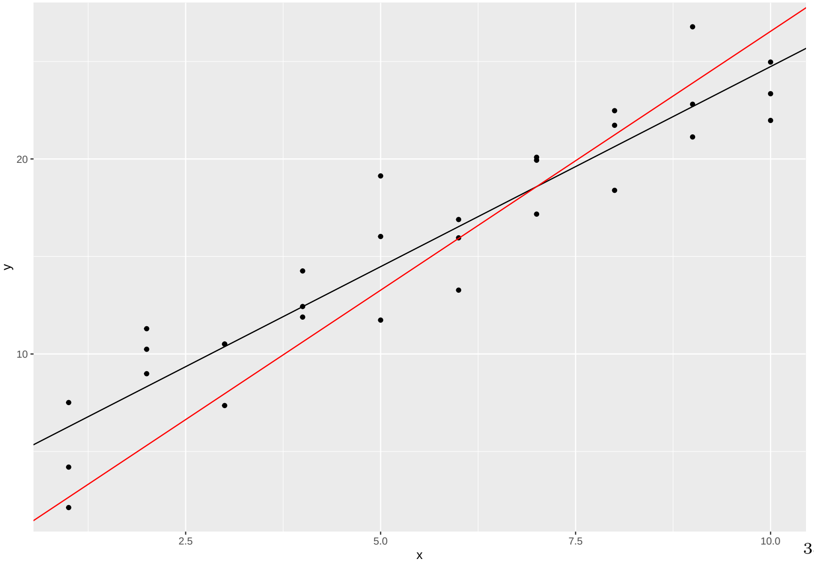
modelr Helper Functions

```r
lm_out2 <- lm(y ~ x - 1, data = sim1)
```

```r
coef(lm_out2)
```

```
##        x
## 2.654508
```

```
ggplot(sim1, aes(x, y)) + geom_point() +
  geom_abline(intercept = coef(lm_out)[1], slope = coef(lm_
  geom_abline(intercept = 0, slope = coef(lm_out2)["x"], co
```



```r
glance(lm_out)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic  p.value    df
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>
## 1     0.885         0.880  2.20      215. 1.17e-14     1
## # i 3 more variables: deviance <dbl>, df.residual <int>,
```

```r
glance(lm_out2)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic  p.value    df
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>
## 1     0.970         0.969  2.94      943. 1.15e-23     1
## # i 3 more variables: deviance <dbl>, df.residual <int>,
```

```r
glance(lm_out)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic  p.value    df
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>
## 1     0.885         0.880  2.20      215. 1.17e-14     1
## # i 3 more variables: deviance <dbl>, df.residual <int>,
```

```r
glance(lm_out2)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic  p.value    df
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>
## 1     0.970         0.969  2.94      943. 1.15e-23     1
## # i 3 more variables: deviance <dbl>, df.residual <int>,
```

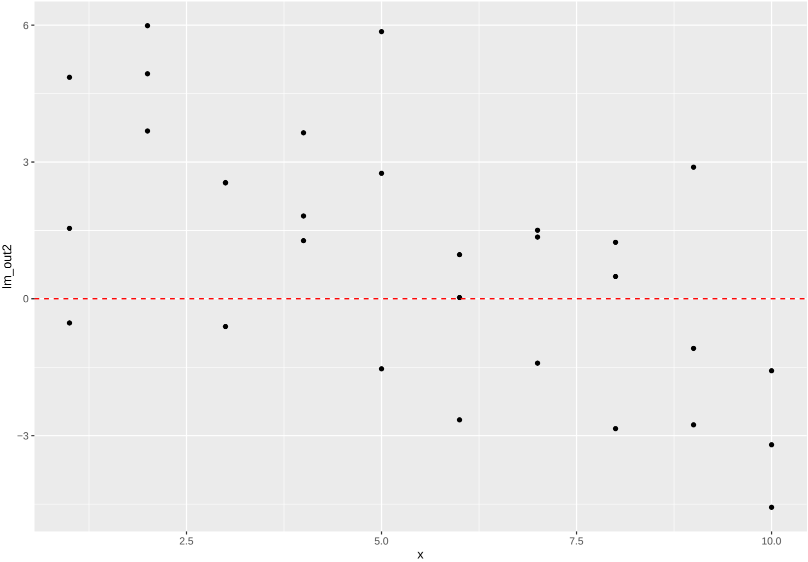
(R^2 and predictive quality are not the same thing ...)

modelr Helper Functions

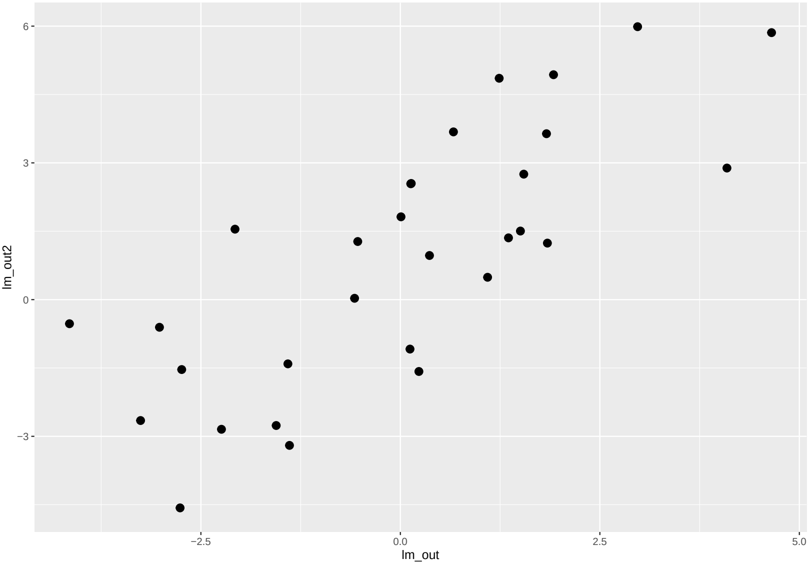
```
( sim1 <- sim1 |> spread_residuals(lm_out, lm_out2) )
```

```
## # A tibble: 30 x 6
##        x     y   resid  pred   lm_out lm_out2
##    <int> <dbl>   <dbl> <dbl>    <dbl>   <dbl>
##  1     1  4.20 -2.07    6.27 -2.07      1.55
##  2     1  7.51  1.24    6.27  1.24      4.86
##  3     1  2.13 -4.15    6.27 -4.15     -0.529
##  4     2  8.99  0.665   8.32  0.665     3.68
##  5     2 10.2   1.92    8.32  1.92      4.93
##  6     2 11.3   2.97    8.32  2.97      5.99
##  7     3  7.36 -3.02   10.4  -3.02     -0.607
##  8     3 10.5   0.130  10.4   0.130     2.54
##  9     3 10.5   0.136  10.4   0.136     2.55
## 10     4 12.4   0.00763 12.4  0.00763   1.82
## # i 20 more rows
```

```r
ggplot(sim1, aes(x, lm_out2)) + geom_point() +
  geom_hline(yintercept = 0, linetype = 2, color = "red")
```



```r
ggplot(sim1, aes(lm_out, lm_out2)) + geom_point(size = 3)
```

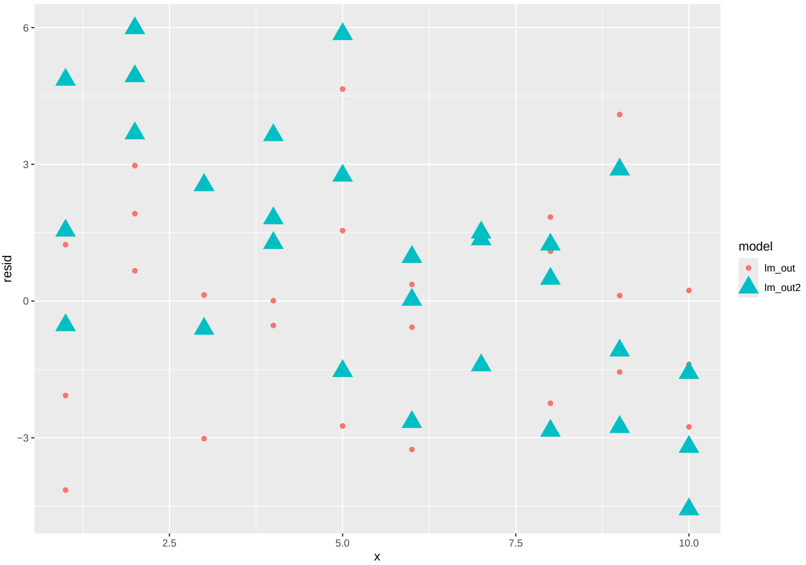


modelr Helper Functions

```
data(sim1)
( sim1 <- sim1 |> gather_residuals(lm_out, lm_out2) )
```

```
## # A tibble: 60 x 4
##    model       x     y    resid
##    <chr>   <int> <dbl>    <dbl>
##  1 lm_out      1  4.20 -2.07
##  2 lm_out      1  7.51  1.24
##  3 lm_out      1  2.13 -4.15
##  4 lm_out      2  8.99  0.665
##  5 lm_out      2 10.2   1.92
##  6 lm_out      2 11.3   2.97
##  7 lm_out      3  7.36 -3.02
##  8 lm_out      3 10.5   0.130
##  9 lm_out      3 10.5   0.136
## 10 lm_out      4 12.4   0.00763
## # i 50 more rows
```

```
ggplot(sim1, aes(x, resid)) +
  geom_point(aes(color = model, size = model, shape = model
```



`modelr` Helper Functions

- `add_residuals()`
- `spread_residuals()`
- `gather_residuals()`
- `add_predictions()`
- `spread_predictions()`
- `gather_predictions()`

Other Helpers for Many Models: `tidy()`

```
ll <- list(lm_out, lm_out2)

ll |> map_df(tidy)
```

```
## # A tibble: 3 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)     4.22    0.869       4.86 4.09e- 5
## 2 x               2.05    0.140      14.7  1.17e-14
## 3 x               2.65    0.0865     30.7  1.15e-23
```

Many Models: glance()

```r
ll |> map_df(glance) |> select(1:5)
```

```
## # A tibble: 2 x 5
##   r.squared adj.r.squared sigma statistic  p.value
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl>
## 1     0.885         0.880  2.20      215. 1.17e-14
## 2     0.970         0.969  2.94      943. 1.15e-23
```

Many Models: glance()

```
ll |> map_df(glance) |> select(1:5)
```

```
## # A tibble: 2 x 5
##   r.squared adj.r.squared sigma statistic  p.value
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl>
## 1     0.885         0.880  2.20      215. 1.17e-14
## 2     0.970         0.969  2.94      943. 1.15e-23
```

```
ll |> map_df(glance) |> select(6:12)
```

```
## # A tibble: 2 x 7
##      df logLik   AIC   BIC deviance df.residual  nobs
##   <dbl>  <dbl> <dbl> <dbl>    <dbl>       <int> <int>
## 1     1  -65.2  136.  141.     136.          28    30
## 2     1  -74.4  153.  156.     250.          29    30
```

Example: `mtcars`

Machine Learning Steps

1. Feature engineering

Machine Learning Steps

1. Feature engineering: collect/create the data

Machine Learning Steps

1. Feature engineering: collect/create the data
2. Data splitting

Machine Learning Steps

1. Feature engineering: collect/create the data
2. Data splitting: split the data

Machine Learning Steps

1. Feature engineering: collect/create the data
2. Data splitting: split the data

- Training. (80%? further split ("cross-validation")?)
- Validation. (for hyperparams; can be small (?))
- Testing. (20%?)

Machine Learning Steps

1. Feature engineering: collect/create the data
2. Data splitting: split the data

- Training. (80%? further split ("cross-validation")?)
- Validation. (for hyperparams; can be small (?))
- Testing. (20%?)

3. Feature selection

Machine Learning Steps

1. Feature engineering: collect/create the data
2. Data splitting: split the data

- Training. (80%? further split ("cross-validation")?)
- Validation. (for hyperparams; can be small (?))
- Testing. (20%?)

3. Feature selection: algorithms decide predictors to include

Machine Learning Steps

1. Feature engineering: collect/create the data
2. Data splitting: split the data

- Training. (80%? further split ("cross-validation")?)
- Validation. (for hyperparams; can be small (?))
- Testing. (20%?)

3. Feature selection: algorithms decide predictors to include
4. Model estimation

Machine Learning Steps

1. Feature engineering: collect/create the data
2. Data splitting: split the data

- Training. (80%? further split ("cross-validation")?)
- Validation. (for hyperparams; can be small (?))
- Testing. (20%?)

3. Feature selection: algorithms decide predictors to include
4. Model estimation: find the slopes (e.g.)

Machine Learning Steps

1. Feature engineering: collect/create the data
2. Data splitting: split the data

- Training. (80%? further split ("cross-validation")?)
- Validation. (for hyperparams; can be small (?))
- Testing. (20%?)

3. Feature selection: algorithms decide predictors to include
4. Model estimation: find the slopes (e.g.)
5. Validation + testing

Machine Learning Steps

1. Feature engineering: collect/create the data
2. Data splitting: split the data

- Training. (80%? further split ("cross-validation")?)
- Validation. (for hyperparams; can be small (?))
- Testing. (20%?)

3. Feature selection: algorithms decide predictors to include
4. Model estimation: find the slopes (e.g.)
5. Validation + testing: evaluate preds from trained models using new data

tidymodels Example

```r
library(tidymodels)
data_split <- initial_split(mtcars, prop = 2 / 3)

df_train <- training(data_split)
df_test <- testing(data_split)
```

tidymodels Example

```r
library(tidymodels)
data_split <- initial_split(mtcars, prop = 2 / 3)

df_train <- training(data_split)
df_test <- testing(data_split)
```

```r
dim(df_train)
```

```
## [1] 21 11
```

```r
dim(df_test)
```

```
## [1] 11 11
```

tidymodels Example

```
lm_fit <- linear_reg() |> fit(mpg ~ ., data = df_train)
lm_fit
```

```
## parsnip model object
##
##
## Call:
## stats::lm(formula = mpg ~ ., data = data)
##
## Coefficients:
## (Intercept)          cyl         disp           hp
##    -3.07537     -0.19100      0.02830     -0.02431
##        qsec           vs           am         gear
##     0.53145     -0.51353      1.54036      3.84550
```

tidymodels Example

```
out_preds <- bind_cols(df_test |> select(mpg),
                       predict(lm_fit, new_data = df_test)
                         rename(lm = .pred))
out_preds
```

```
##                      mpg       lm
## Mazda RX4 Wag       21.0 21.66514
## Hornet 4 Drive      21.4 20.20164
## Hornet Sportabout   18.7 18.80151
## Merc 450SE          16.4 13.48582
## Cadillac Fleetwood  10.4 14.11564
## Lincoln Continental 10.4 13.17200
## Toyota Corolla      33.9 28.64901
## Fiat X1-9           27.3 27.68602
## Porsche 914-2       26.0 30.67169
## Ford Pantera L      15.8 25.50938
## Ferrari Dino        19.7 19.95970
```

`tidymodels` Example

Next, predict with *random forest* algorithm.

`tidymodels` Example

Next, predict with *random forest* algorithm.

Ensemble learning algorithms:

- Boosting: models build on prior models

Next, predict with *random forest* algorithm.

Ensemble learning algorithms:

- Boosting: models build on prior models

`tidymodels` Example

Next, predict with *random forest* algorithm.

Ensemble learning algorithms:

- Boosting: models build on prior models $\rightsquigarrow$ pick feature, predict, upweight mispredicted data, ... Do several times and combine.
- Bagging: (random select units, model) $\rightarrow$ many times. No building.

`tidymodels` Example

Next, predict with *random forest* algorithm.

Ensemble learning algorithms:

- Boosting: models build on prior models $\rightsquigarrow$ pick feature, predict, upweight mispredicted data, ... Do several times and combine.
- Bagging: (random select units, model) $\rightarrow$ many times. No building.

`tidymodels` Example

Next, predict with *random forest* algorithm.

Ensemble learning algorithms:

- Boosting: models build on prior models $\rightsquigarrow$ pick feature, predict, upweight mispredicted data, ... Do several times and combine.
- Bagging: (random select units, model) $\rightarrow$ many times. No building.

Random Forests are bagging algorithms.

Breiman (2001a)

tidymodels Example

```r
rf_fit <- rand_forest(mode = "regression") |>
  fit(mpg ~ ., data = df_train)
rf_fit
```

```
## parsnip model object
##
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, num.thre
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      21
## Number of independent variables:  10
## Mtry:                             3
## Target node size:                 5
## Variable importance mode:         none
```

tidymodels Example

parsnip::rand_forest() uses ranger engine

`tidymodels` Example

`parsnip::rand_forest()` uses `ranger` engine

There is also "Spark".

tidymodels Example

```r
out_preds <- bind_cols(out_preds,
                       predict(rf_fit, new_data = df_test) 
                         rename(rf = .pred))
out_preds
```

```
##                      mpg       lm       rf
## Mazda RX4 Wag       21.0 21.66514 20.36497
## Hornet 4 Drive      21.4 20.20164 19.52452
## Hornet Sportabout   18.7 18.80151 16.45822
## Merc 450SE          16.4 13.48582 16.42280
## Cadillac Fleetwood  10.4 14.11564 16.26746
## Lincoln Continental 10.4 13.17200 15.78517
## Toyota Corolla      33.9 28.64901 29.41605
## Fiat X1-9           27.3 27.68602 29.41278
## Porsche 914-2       26.0 30.67169 25.27119
## Ford Pantera L      15.8 25.50938 17.23360
## Ferrari Dino        19.7 19.95970 19.91055
```

tidymodels Example

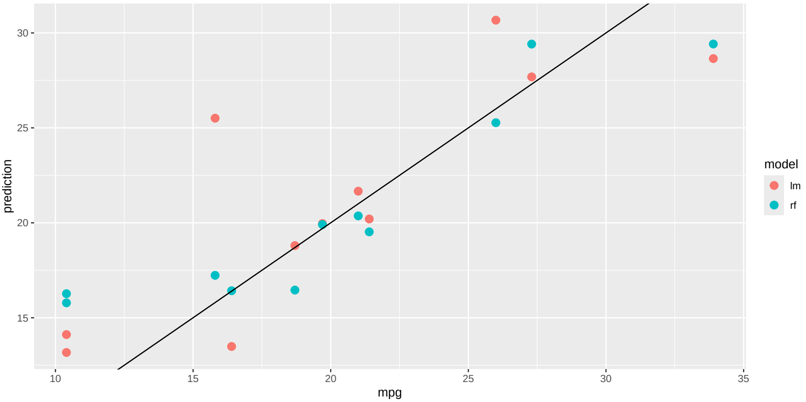
```
out_preds_long <- out_preds |>
  pivot_longer(cols = c(lm, rf),
               names_to = "model",
               values_to = "prediction")

out_preds_long
```

```
## # A tibble: 22 x 3
##      mpg model prediction
##    <dbl> <chr>      <dbl>
##  1  21   lm          21.7
##  2  21   rf          20.4
##  3  21.4 lm          20.2
##  4  21.4 rf          19.5
##  5  18.7 lm          18.8
##  6  18.7 rf          16.5
##  7  16.4 lm          13.5
##  8  16.4 rf          16.4
```

tidymodels Example

```
ggplot(out_preds_long, aes(mpg, prediction)) +
  geom_point(aes(color = model), size = 3) +
  geom_abline(slope = 1, intercept = 0)
```



tidymodels Example

Evaluate:

```
out_preds |> metrics(truth = mpg, estimate = lm) |>
  rename(lm = .estimate) |>
  left_join(out_preds |>
              metrics(truth = mpg, estimate = rf) |>
              rename(rf = .estimate))
```

```
## Joining with `by = join_by(.metric, .estimator)`
```

```
## # A tibble: 3 x 4
##   .metric .estimator    lm    rf
##   <chr>   <chr>      <dbl> <dbl>
## 1 rmse    standard   4.00  3.01
## 2 rsq     standard   0.683 0.842
## 3 mae     standard   2.88  2.27
```

Example: Social Pressure Experiment (`recipes`)

Data Splitting

```
social <- read_csv("https://raw.githubusercontent.com/I

soc_split <- initial_split(social)
soc_train <- training(soc_split)
soc_test <- testing(soc_split)
```

Data Splitting

```
social <- read_csv("https://raw.githubusercontent.com/I

soc_split <- initial_split(social)
soc_train <- training(soc_split)
soc_test <- testing(soc_split)
```

```
dim(soc_train)
```

```
## [1] 229399      6
```

```
dim(soc_test)
```

```
## [1] 76467     6
```

Feature Engineering

```r
social_recip <- recipe(primary2006 ~ ., data = soc_train)

social_recip
```

Feature Engineering

```r
summary(social_recip)
```

```
## # A tibble: 6 x 4
##   variable    type       role      source
##   <chr>       <list>     <chr>     <chr>
## 1 sex         <chr [3]> predictor original
## 2 yearofbirth <chr [2]> predictor original
## 3 primary2004 <chr [2]> predictor original
## 4 messages    <chr [3]> predictor original
## 5 hhsize      <chr [2]> predictor original
## 6 primary2006 <chr [2]> outcome   original
```

Feature Engineering

```
social_recip <- social_recip |>
  step_mutate(age = 2006 - yearofbirth) |>
  step_dummy(all_nominal(), -all_outcomes())
```

```
social_recip
```

```
##
## -- Recipe ----------------------------------------------
##
## -- Inputs
## Number of variables by role
## outcome:   1
## predictor: 5
##
## -- Operations
## * Variable mutation for: 2006 - yearofbirth
## * Dummy variables from: all_nominal() -all_outcomes()
```

Feature Engineering

```r
social_recip <- social_recip |>
  step_zv(all_predictors())
```

```
social_recip
```

```
##
## -- Recipe ------------------------------------------------------------------
##
## -- Inputs
## Number of variables by role
## outcome:   1
## predictor: 5
##
## -- Operations
## * Variable mutation for: 2006 - yearofbirth
## * Dummy variables from: all_nominal() -all_outcomes()
## * Zero variance filter on: all_predictors()
```

Feature Engineering

```r
social_recip <- social_recip |>
  step_center(all_predictors(), -primary2004)
```

```r
social_recip
```

```
##
## -- Recipe ----------------------------------------------------
##
## -- Inputs
## Number of variables by role
## outcome:   1
## predictor: 5
##
## -- Operations
## * Variable mutation for: 2006 - yearofbirth
## * Dummy variables from: all_nominal() -all_outcomes()
## * Zero variance filter on: all_predictors()
## * Centering for: all_predictors() -primary2004
```

Feature Engineering

```r
social_recip <- social_recip |>
  step_interact(terms = ~
                  age:all_predictors() +
                  primary2004:all_predictors()
                  )
```

Feature Engineering

Recipe complete. Time to prep and bake.

Feature Engineering

Recipe complete. Time to prep and bake.

```r
social_recip |>
  prep()
```

```
##

## -- Recipe -----------------------------------------

##

## -- Inputs

## Number of variables by role

## outcome:   1
## predictor: 5

##

## -- Training information
```

```r
soc_train_processed <- social_recip |>
  prep() |>
  bake(new_data = NULL)

soc_train_processed
```

```
## # A tibble: 229,399 x 22
##    yearofbirth primary2004 hhsize primary2006    age sex
##          <dbl>       <dbl>  <dbl>       <dbl>  <dbl>
##  1     -13.2             1 -0.185           0 13.2     -
##  2       0.779           1 -0.185           0 -0.779   -
##  3      -6.22            1 -0.185           0  6.22
##  4      -4.22            0  0.815           0  4.22    -
##  5      -5.22            1 -1.18            1  5.22    -
##  6     -19.2             0  0.815           0 19.2
##  7     -25.2             0 -0.185           1 25.2
##  8     -29.2             0 -0.185           1 29.2     -
##  9     -12.2             0  0.815           0 12.2
## 10      -4.22            0  0.815           0  4.22    -
## # i 229,389 more rows
```

```r
names(soc_train_processed)
```

```
##  [1] "yearofbirth"                         "primary2004
##  [3] "hhsize"                              "primary2006
##  [5] "age"                                 "sex_male"
##  [7] "messages_Control"                    "messages_Ha
##  [9] "messages_Neighbors"                  "age_x_yearo
## [11] "age_x_primary2004"                   "age_x_hhsiz
## [13] "age_x_sex_male"                      "age_x_messa
## [15] "age_x_messages_Hawthorne"            "age_x_messa
## [17] "yearofbirth_x_primary2004"           "primary2004
## [19] "primary2004_x_sex_male"              "primary2004
## [21] "primary2004_x_messages_Hawthorne" "primary2004
```

```r
soc_test_processed <- social_recip |>
  prep() |>
  bake(new_data = soc_test)

soc_test_processed
```

```
## # A tibble: 76,467 x 22
##    yearofbirth primary2004 hhsize primary2006    age sex
##          <dbl>       <dbl>  <dbl>       <dbl>  <dbl>
##  1      -15.2            0 -0.185           0  15.2
##  2       -6.22           0  0.815           1   6.22   -
##  3       10.8            0 -0.185           0 -10.8
##  4      -15.2            1 -1.18            1  15.2
##  5       12.8            1 -1.18            0 -12.8    -
##  6       10.8            1 -0.185           1 -10.8    -
##  7       26.8            0  1.82            0 -26.8    -
##  8        8.78           0 -0.185           0  -8.78
##  9      -24.2            0 -0.185           0  24.2    -
## 10        2.78           0 -0.185           0  -2.78   -
## # i 76,457 more rows
```

Now, with train and test data ready, add *model specification*, *fitting*, *evaluation*, *deployment* to workflow.

Regularization Methods: LASSO, ridge regression, elastic nets

Feature Selection

- Wrappers: pick subset of covars, train on data (estimate model), test on hold-out, score predictions. Keep best-scoring subset.

Feature Selection

- Wrappers: pick subset of covars, train on data (estimate model), test on hold-out, score predictions. Keep best-scoring subset.
- Filters: correlate covars with outcome. Keep strongest.

Feature Selection

- Wrappers: pick subset of covars, train on data (estimate model), test on hold-out, score predictions. Keep best-scoring subset.
- Filters: correlate covars with outcome. Keep strongest.
- Embeds: select features and estimate model at same time. Penalize using more predictors.

Embedded Regularization Methods

OLS reminder

Minimize SSR:

$$\arg\min_{\beta} \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2$$

$$\arg\min_{\beta} \sum_{i=1}^{n} \left(\mathbf{y} - \mathbf{X}\hat{\beta}\right)^2$$

Embedded Regularization Methods

L1 regularization: the LASSO (Least Absolute Shrinkage and Selection Operator)

$$\arg\min_{\beta}\left[\sum_{i=1}^{n}\left(y_i-\mathbf{X}\hat{\beta}\right)^2+\lambda\sum_{j=1}^{k}|\hat{\beta}_j|\right]$$

Embedded Regularization Methods

L1 regularization: the LASSO (Least Absolute Shrinkage and Selection Operator)

$$\arg\min_{\beta} \left[\sum_{i=1}^{n} \left(y_i - \mathbf{X}\hat{\beta} \right)^2 + \lambda \sum_{j=1}^{k} |\hat{\beta}_j| \right]$$

L2 regularization: Ridge regression

$$\arg\min_{\beta} \left[\sum_{i=1}^{n} \left(y_i - \mathbf{X}\hat{\beta} \right)^2 + \lambda \sum_{j=1}^{k} \hat{\beta}_j^2 \right]$$

Embedded Regularization Methods

Mix L1 and L2: Elastic net

$$\arg\min_{\beta} \left(\frac{\sum_{i=1}^{n} \left(y_i - \mathbf{X}\hat{\beta} \right)^2}{2n} + \lambda \left[\alpha \sum_{j=1}^{k} |\hat{\beta}_j| + \frac{1-\alpha}{2} \sum_{j=1}^{k} \hat{\beta}_j^2 \right] \right)$$

Embedded Regularization Methods

Mix L1 and L2: Elastic net

$$\arg\min_{\beta} \left(\frac{\sum_{i=1}^{n} \left(y_i - \mathbf{X}\hat{\beta} \right)^2}{2n} + \lambda \left[\alpha \sum_{j=1}^{k} |\hat{\beta}_j| + \frac{1-\alpha}{2} \sum_{j=1}^{k} \hat{\beta}_j^2 \right] \right)$$

Regularized trees, ...

R packages for Regularization, etc.

- `glmnet`
- `caret`

See also `tidymodels`, `parsnip`, ...

References

Breiman, Leo. 2001a. "Random Forests." *Machine Learning* 45: 5–32.

———. 2001b. "Statistical Modeling: The Two Cultures." *Statistical Science* 16 (3): 199–215. http://www.jstor.org/stable/2676681.