Modeling Process

DATA 202 21FA

Q&A

> Is there a case where false positive can cause more harm than false negative?

Machine Bias

There's software used across the country to predict future criminals. And it's biased against blacks.

by Julia Angwin, Jeff Larson, Surya Mattu and Lauren Kirchner, ProPublica
May 23, 2016

ON A SPRING AFTERNOON IN 2014, Brisha Borden was running late to pick up her god-sister from school when she spotted an unlocked kid's blue Huffy bicycle and a silver Razor scooter. Borden and a friend grabbed the bike and scooter and tried to ride them down the street in the Fort Lauderdale suburb of Coral Springs.

Just as the 18-year-old girls were realizing they were too big for the tiny conveyances — which belonged to a 6-year-old boy — a woman came running after them saying, "That's my kid's stuff." Borden and her friend immediately dropped the bike and scooter and walked away.

https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

Q&A

> Is regression or classification more common?

Depends on the application. But classification seems more fundamental.

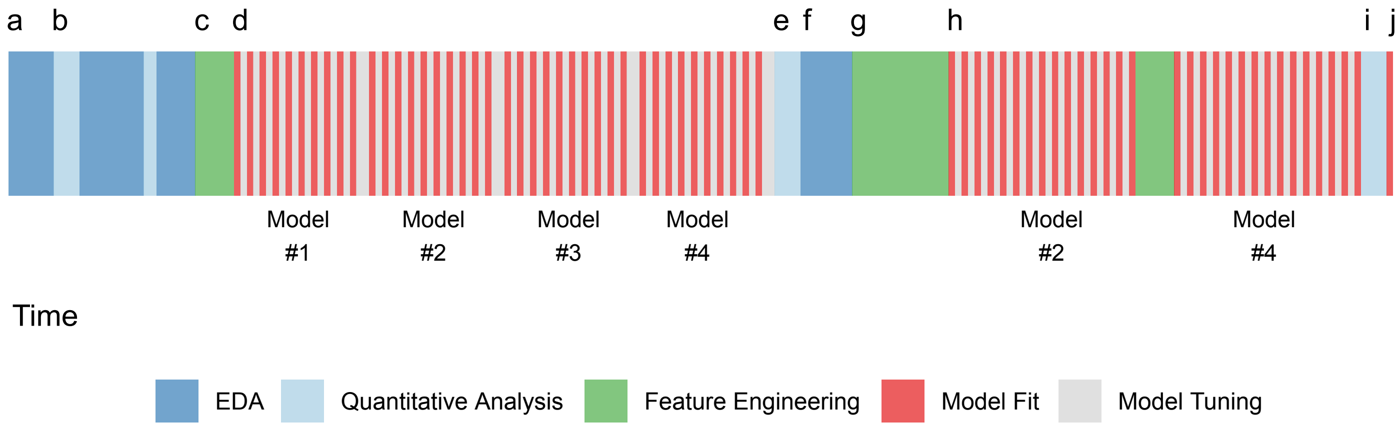
Think about how you'd do regression if all you had was a classification model. (Hint: histograms.)

> Confusion matrices are confusing. Can we practice?

In lab Friday!

Objectives

- What are the basic steps in training and validating any predictive model?
- Why is each step important?
- How can we use the `tidymodels` ecosystem to train and validate a linear model?



Source: Feature Engineering and Selection ch1

Predictive Modeling Workflow

Preliminaries:

1. **Define the problem**: predict *what*, based on *what*? What *metrics* will indicate success? (Measure success in multiple ways!)
2. **Explore your data** (EDA): understand its structure, make lots of plots

Predictive Modeling Workflow

Preliminaries:

1. **Define the problem**: predict *what*, based on *what*? What *metrics* will indicate success? (Measure success in multiple ways!)
2. **Explore your data** (EDA): understand its structure, make lots of plots

1. **Pick a model**: Which type(s) of models are appropriate for task and data?
2. **Transform the data** as needed by the model ("feature engineering", preprocessing", "recipe")
3. **Split the data** to allow for validation.
4. **Fit and evaluate the model**
5. **Tune**: adjust model hyperparameters
6. **Analyze model errors** and refine all earlier steps

```r
library(tidymodels)
```

Packages:

- `parsnip`: **Specify** and **train** the model you want
- `recipes`: **Prepare** the data
- `rsample`: **Split** data into training and validation
- `yardstick`: Compute **metrics** for performance
- `tune`: Helps you set the dials.

Where to find documentation

Theory

- An Introduction to Statistical Learning
- Feature Engineering and Selection

Practice

- TidyModels website: Getting Started, vignettes
- Tidy Modeling with R book (work in progress)

Some others:

- https://rviews.rstudio.com/2019/06/19/a-gentle-intro-to-tidymodels/
- https://juliasilge.com/blog/intro-tidymodels/

Example data: Ames home sales

Like before, but we subset the data as De Cock suggests. Again, see Data dictionary

```
data(ames, package = "modeldata")
ames <- AmesHousing::make_ames() %>%
  mutate(Sale_Price = Sale_Price / 1000) %>%
  filter(Gr_Liv_Area < 4000, Sale_Condition == "Normal")
nrow(ames)
```

```
[1] 2412
```

```
ames %>% head(5)
```

```
# A tibble: 5 × 81
  MS_SubClass      MS_Zoning    Lot_Frontage Lot_Area Street Alley
  <fct>            <fct>               <dbl>    <int> <fct>  <fct>
1 One_Story_1946… Residentia…           141    31770 Pave   No_Al…
2 One_Story_1946… Residentia…            80    11622 Pave   No_Al…
3 One_Story_1946… Residentia…            81    14267 Pave   No_Al…
4 One_Story_1946… Residentia…            93    11160 Pave   No_Al…
```

Defining the problem

- Predict *what*? `Sale_Price`
- How to measure success?

```
metrics <- yardstick::metric_set(mae, mape, rsq_trad)
```

Exploratory Analysis (EDA)

```
skimr::skim(ames)
```

Table: Data summary

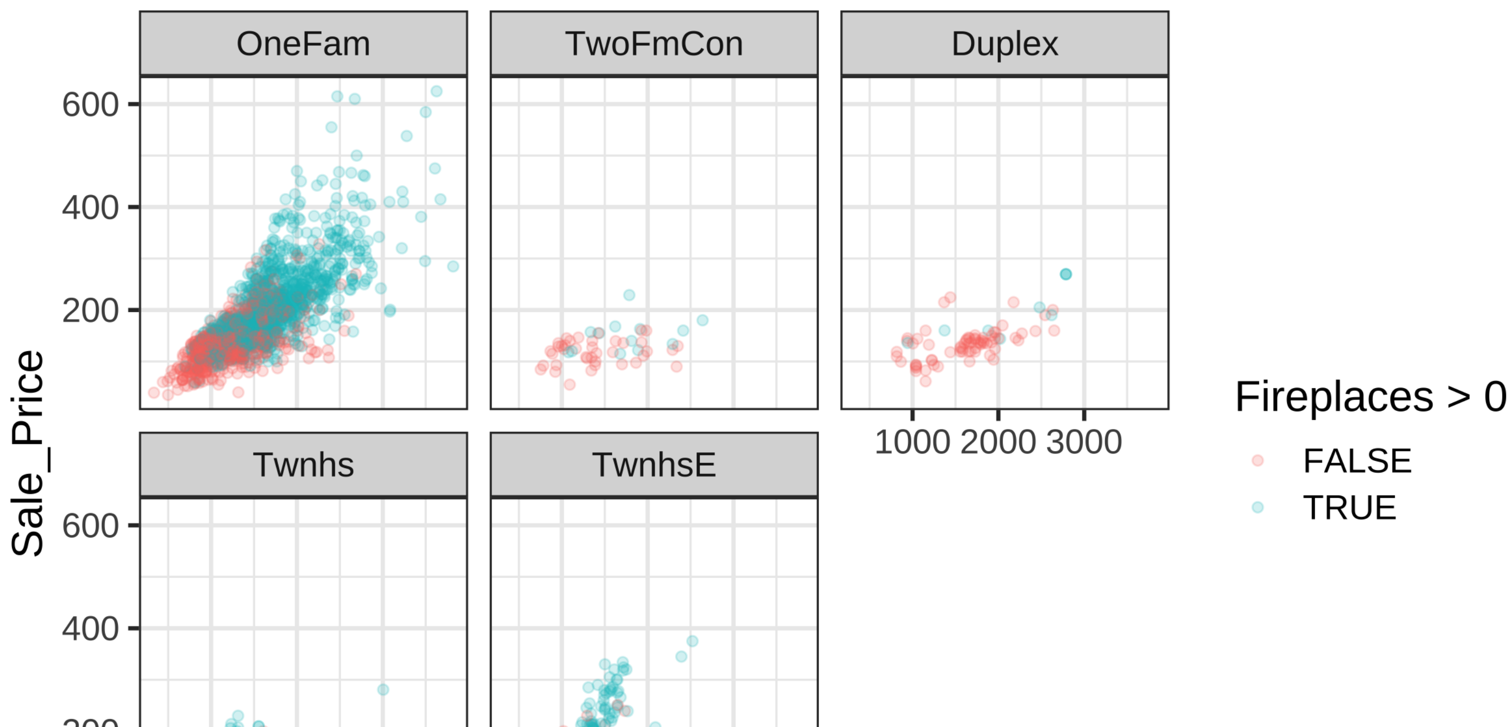
Name	ames
Number of rows	2412
Number of columns	81
–	
Column type frequency:	
factor	46
numeric	35
—	
Group variables	None

Variable type: factor

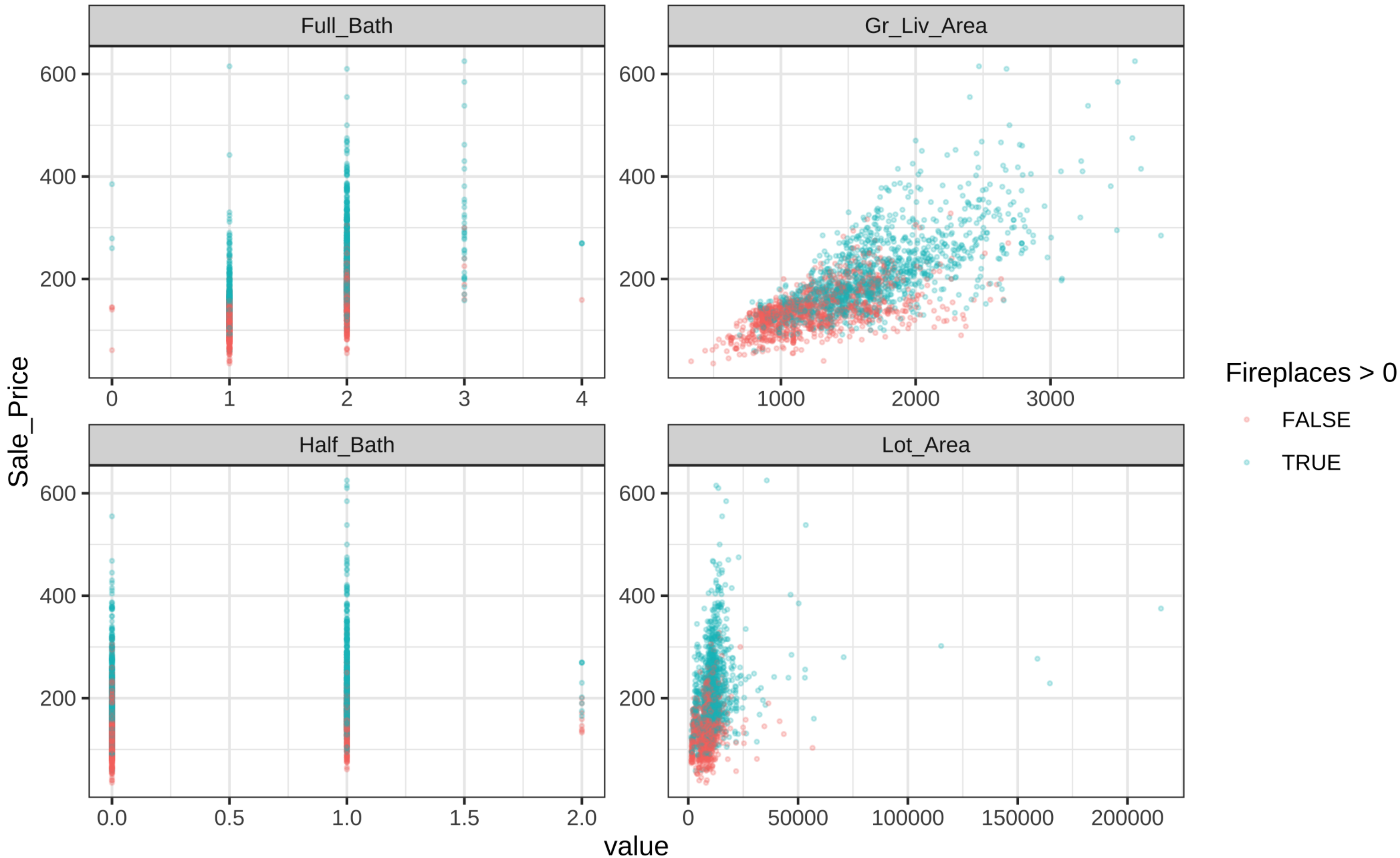
skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
MS_SubClass	0	1	FALSE	16	One: 866, Two: 464, One: 247, One: 155
MS_Zoning	0	1	FALSE	7	Res: 1890, Res: 394, Flo: 92, Res: 20
Street	0	1	FALSE	2	Pav: 2403, Grv: 9
Alley	0	1	FALSE	3	No_: 2258, Gra: 100, Pav: 54
Lot_Shape	0	1	FALSE	4	Reg: 1533, Sli: 802, Mod: 65, Irr: 12
Land_Contour	0	1	FALSE	4	Lvl: 2184, Bnk: 91, HLS: 86, Low: 51
Condition_1	0	1	FALSE	9	Nor: 2083, Fee: 130, Art: 77, Pos: 35
Condition_2	0	1	FALSE	8	Nor: 2390, Fee: 12, Art: 2, Pos: 2
Bldg_Type	0	1	FALSE	5	One: 2001, Twn: 188, Twn: 93, Dup: 78
House_Style	0	1	FALSE	8	One: 1189, Two: 725, One: 270, SLv: 115
Overall_Qual	0	1	FALSE	10	Ave: 715, Abo: 640, Goo: 493, Ver: 256
Overall_Cond	0	1	FALSE	9	Ave: 1282, Abo: 473, Goo: 352, Ver: 139
Roof_Style	0	1	FALSE	6	Gab: 1936, Hip: 432, Gam: 18, Fla: 13
Roof_Matl	0	1	FALSE	7	Com: 2381, Tar: 15, WdS: 7, WdS: 6
Exterior_1st	0	1	FALSE	15	Vin: 781, HdB: 396, Met: 390, Wd : 371
Exterior_2nd	0	1	FALSE	16	Vin: 770, Met: 388, HdB: 363, Wd : 357
Mas_Vnr_Type	0	1	FALSE	4	Non: 1492, Brk: 749, Sto: 153, Brk: 18
Exter_Qual	0	1	FALSE	4	Typ: 1562, Goo: 770, Exc: 53, Fai: 27
Exter_Cond	0	1	FALSE	5	Typ: 2080, Goo: 266, Fai: 53, Exc: 11
Foundation	0	1	FALSE	6	CBl: 1090, PCo: 1002, Brk: 264, Sla: 41
Bsmt_Qual	0	1	FALSE	6	Typ: 1117, Goo: 1002, Exc: 152, Fai: 72
Bsmt_Cond	0	1	FALSE	6	Typ: 2169, Goo: 86, Fai: 84, No_: 67
Bsmt_Exposure	0	1	FALSE	5	No: 1611, Av: 315, Gd: 218, Mn: 199
BsmtFin_Type_1	0	1	FALSE	7	GLQ: 694, Unf: 644, ALQ: 378, Rec: 250
BsmtFin_Type_2	0	1	FALSE	7	Unf: 2026, Rec: 96, LwQ: 78, No_: 68
Heating	0	1	FALSE	6	Gas: 2374, Gas: 24, Gra: 6, Wal: 5
Heating_QC	0	1	FALSE	5	Exc: 1183, Typ: 726, Goo: 422, Fai: 80
Central_Air	0	1	FALSE	2	Y: 2259, N: 153
Electrical	0	1	FALSE	5	SBr: 2207, Fus: 158, Fus: 39, Fus: 7
Kitchen_Qual	0	1	FALSE	5	Typ: 1294, Goo: 941, Exc: 115, Fai: 61
Functional	0	1	FALSE	6	Typ: 2239, Min: 63, Min: 55, Mod: 30
Fireplace_Qu	0	1	FALSE	6	No_: 1164, Goo: 555, Typ: 554, Fai: 68
Garage_Type	0	1	FALSE	7	Att: 1420, Det: 684, Bui: 140, No_: 116
Garage_Finish	0	1	FALSE	4	Unf: 1066, RFn: 671, Fin: 558, No_: 117
Garage_Qual	0	1	FALSE	6	Typ: 2163, No_: 117, Fai: 105, Goo: 20
Garage_Cond	0	1	FALSE	6	Typ: 2204, No_: 117, Fai: 65, Goo: 13

Exploratory Analysis (EDA): Make lots of plots.

```
ames %>%
  ggplot(aes(x = Gr_Liv_Area, y = Sale_Price, color = Fireplaces
  geom_point(alpha = .2) +
  facet_wrap(vars(Bldg_Type))
```



```r
ames %>% select(Sale_Price, Gr_Liv_Area, Lot_Area, Full_Bath, Ha
  pivot_longer(-c(Sale_Price, Fireplaces)) %>%
  ggplot(aes(x = value, y = Sale_Price, color = Fireplaces > 0))
  facet_wrap(vars(name), scales = "free") + theme_bw()
```



Specifying a Model

Example (without validation)

Specify the model:

```
my_model_spec <-  parsnip::decision_tree(mode = "regression")
```

Train it ("fit") on data:

```
my_trained_model <- my_model_spec %>%
  fit(Sale_Price ~ Gr_Liv_Area, data = ames)
```

Example (without validation)

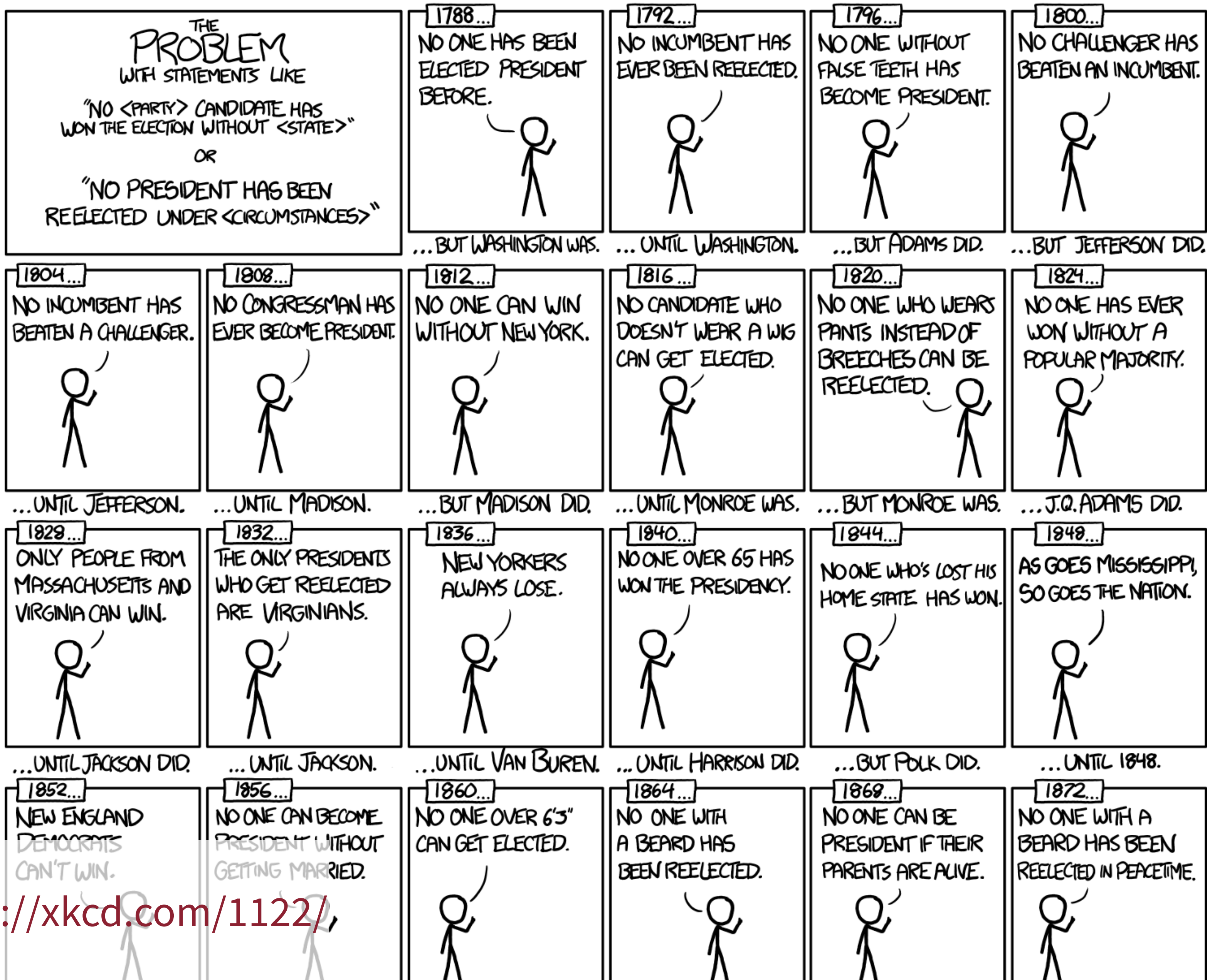
Predict on new data:

```
new_data <- tibble(Gr_Liv_Area = c(1000, 2000), Sale_Price = c(2
my_trained_model %>%
  predict(new_data) %>%
  bind_cols(new_data) # Put back the original data
```

```
# A tibble: 2 × 3
  .pred Gr_Liv_Area Sale_Price
  <dbl>       <dbl>      <dbl>
1  119.        1000     239000
2  228.        2000     185000
```

Evaluate on all data:

```
predictions <- my_trained_model %>%
  predict(ames) %>%
  bind_cols(ames)
predictions %>%
  metrics(truth = Sale_Price, estimate = .pred)
```



https://xkcd.com/1122/

Example, with validation

1. Hold out some data to use for validation:

```
set.seed(10)
ames_split <- initial_split(ames, prop = 3/4)
ames_train <- training(ames_split)
ames_test <- testing(ames_split)
glue("Using {nrow(ames_train)} sales to train, {nrow(ames_test)}
```

```
Using 1809 sales to train, 603 to test
```

1. Hold out some data to use for validation.
2. Specify the model to use.
3. Train the model **on the training set**:

```
my_trained_model <- my_model_spec %>%
  fit(Sale_Price ~ Gr_Liv_Area,
      data = ames_train)
```

1. Hold out some data to use for validation.
2. Specify the model to use.
3. Train the model **on the training set**
4. Evaluate on training set (optional):

```r
train_predictions <-
  my_trained_model %>%
    predict(ames_train) %>%
    bind_cols(ames_train) # Pu

train_predictions
```

```
# A tibble: 1,809 × 82
  .pred MS_SubClass  MS_Zoning
  <dbl> <fct>        <fct>
1  118. One_Story_1… Resident…
2  155. One_Story_1… Resident…
3  194. One_Story_P… Resident…
4  118. One_Story_1… Resident…
5  297. Two_Story_1… Resident…
6  118. One_Story_1… Resident…
# … with 1,803 more rows, and 7
```

```r
train_predictions %>%
  metrics(truth = Sale_Price,
```

```
# A tibble: 3 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 mae      standard      35.1
2 mape     standard      22.0
3 rsq_trad standard       0.513
```

1. Hold out some data to use for validation
2. Specify the model to use.
3. Train the model **on the training set**
4. Evaluate on training set (optional)
5. Evaluate on **test set**:

```
my_trained_model %>%
  predict(ames_test) %>%
  bind_cols(ames_test) %>%
  metrics(truth = Sale_Price, estimate = .pred)
```

```
# A tibble: 3 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 mae      standard      33.2
2 mape     standard      19.6
3 rsq_trad standard       0.563
```

> What's the optimal ratio of train to test?

What's the trade-off? What happens if train is too small? If test is too small?

Many models, same interface

```
trained_linear_model <-
  parsnip::linear_reg() %>%
  fit(Sale_Price ~ Gr_Liv_Area,
      data = ames_train)

trained_linear_model %>%
  predict(ames_test) %>%
  bind_cols(ames_test) %>%
  metrics(truth = Sale_Price, estimate = .pred)
```

```
# A tibble: 3 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 mae      standard      31.0
2 mape     standard      18.0
3 rsq_trad standard       0.586
```

Types of models

- Linear models
  - ordinary least-squares (OLS)
  - Lasso, Ridge, etc.: penalize large coefficients
  - Generalized Linear Models: outputs get transformed
  - Logistic Regression (also Support Vector Machine): transform output to *score* for each class
- Decision Lists and Trees
  - extension: Random Forests
- Neural Networks: layered combinations of the above
- many, many more

Which variables mean what?

The *formula interface*:

- `y ~ x`
  - predict *y* using *x*. `Sale_Price ~ Gr_Liv_Area`
- `y ~ x1 + x2 + x3`
  - predict *y* using *x1* and *x2* and *x3*
  - `Sale_Price ~ Gr_Liv_Area + Lot_Area + Full_Bath`

Don't get confused: they "forgot" the coefficients! A fitted linear model will actually look like:

Sale_Price = c1 * Gr_Liv_Area + c2 * Lot_Area + c3 * Full_Bath + intercept