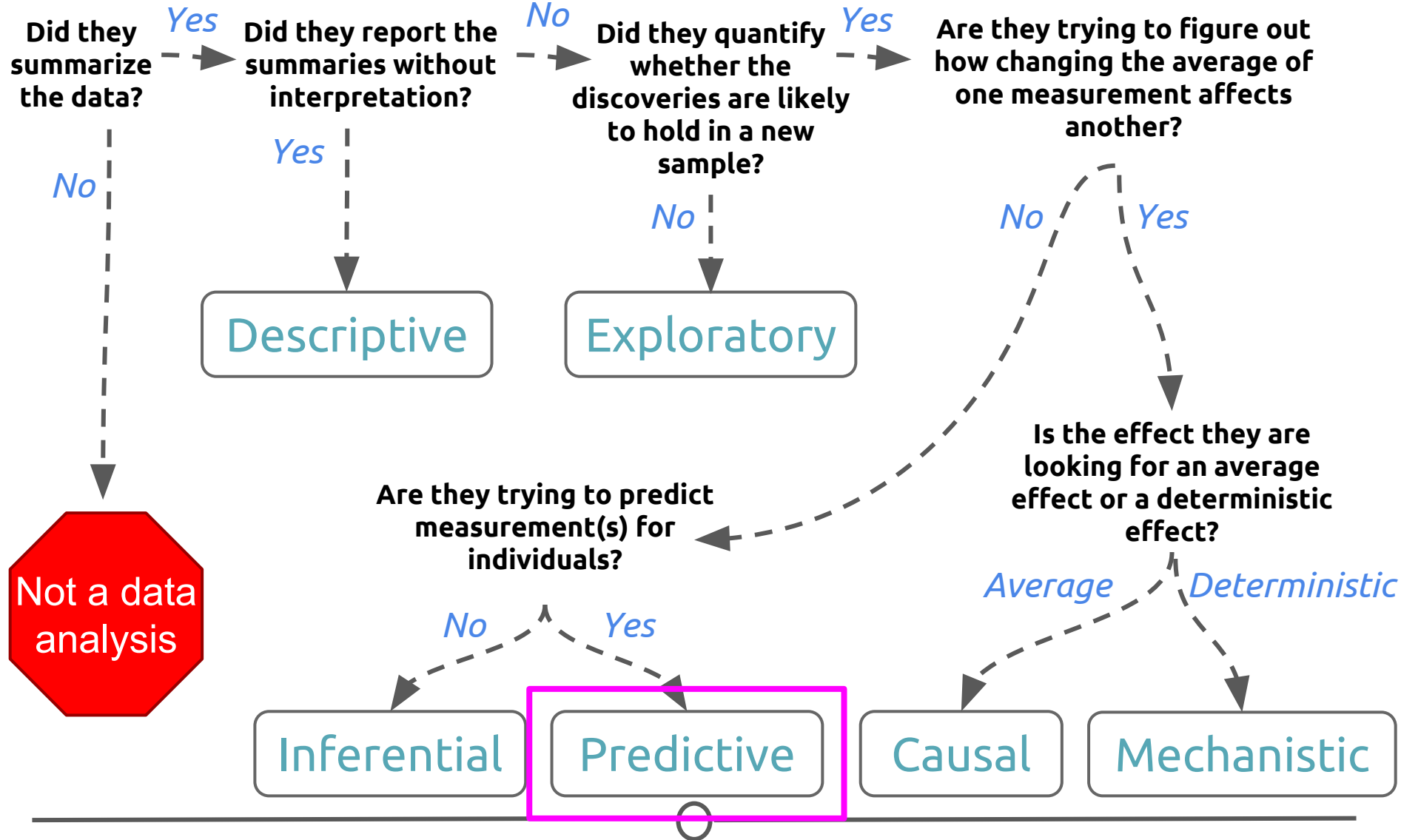
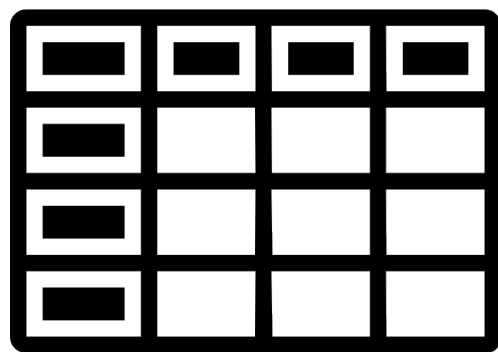


# Prediction & Machine Learning



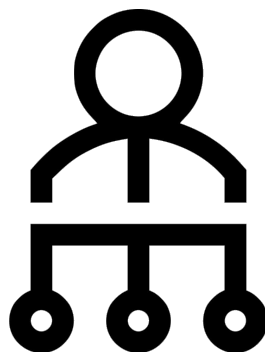
Data Analysis





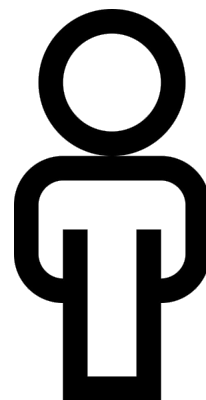
data

train →



model

→ predict





## THE LATEST

8:57 AM

**The Rockets Are  
Melo's Best, Last  
Hope**

7:20 AM

**Significant Digits For  
Tuesday, July 24,  
2018**

6:00 AM

**What The Rise Of  
Kamala Harris Tells  
Us About The  
Democratic Party**

## INTERACTIVES

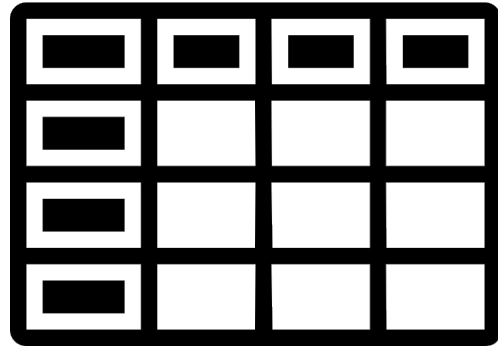
**How Popular Is Donald Trump?**

UPDATED 15 HOURS AGO



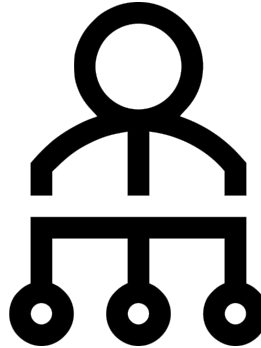
See all approval polls

**predictive analysis** uses  
data you have now to make  
predictions in the future



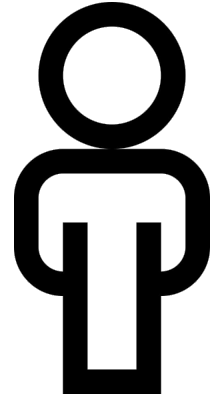
data

train →



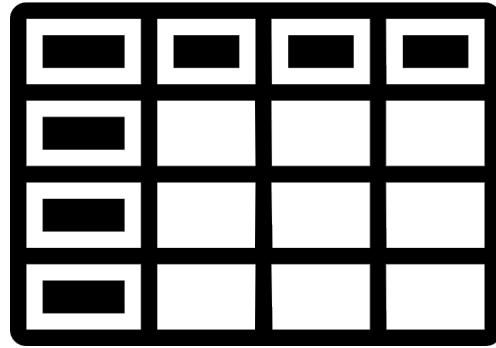
model

→ predict



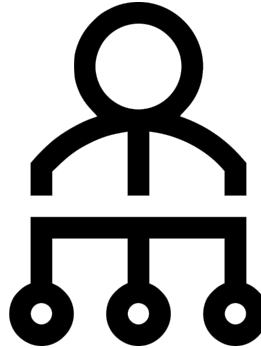
**predictive analysis** uses  
data you have now to make  
predictions in the future

**machine learning** is another  
term for this!



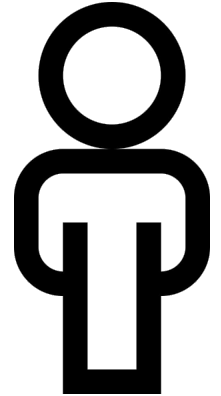
data

train →

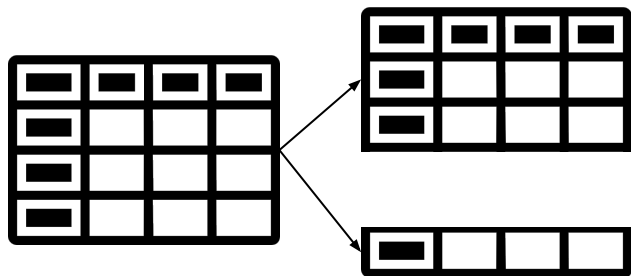


model

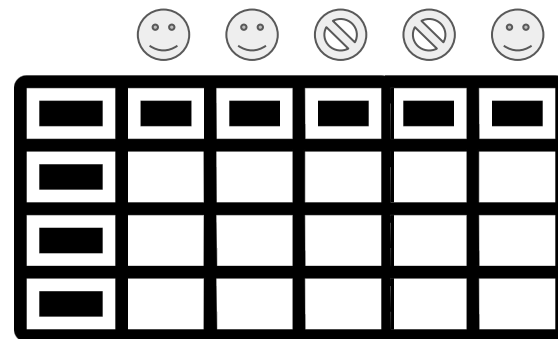
→ predict



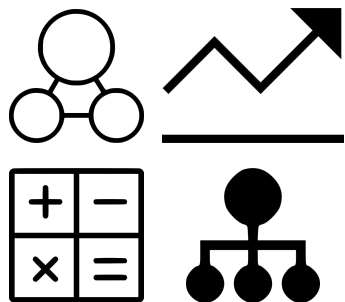
# Basic Steps to Prediction



data splitting



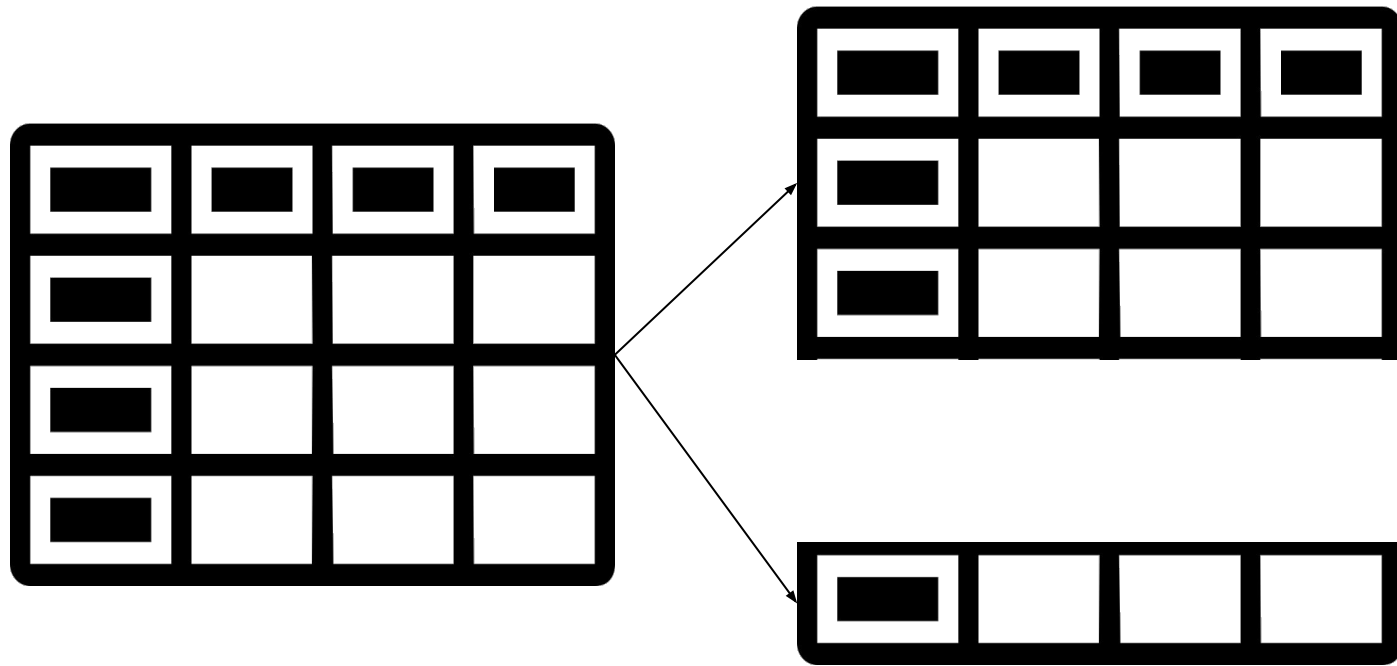
variable selection



model selection



accuracy assessment



data splitting







**Michael Hoffman**

@michaelhoffman

Following



What do you call the datasets you use for (1) training, (2) feature selection and hyperparameter tuning, and (3) quantifying performance. Please share your reasoning and disciplinary background too.

32% train, test, validate

37% train, validate, test

3% other (specify)

28% Just show me the results

pretty even split  
between the two  
options...

244 votes • Final results

5:15 PM - 25 Apr 2018

<https://twitter.com/michaelhoffman/status/989251677646704641>



**Michael Hoffman** @michaelhoffman · Apr 27

@CarldeBoerPhD suggests train, tune, test as an alternative. I think I like that. "Tune" is a better description for what goes on in the second part and it avoids the contentious and multisyllabic "validate".

**Carl de Boer** @CarldeBoerPhD

Let's all switch to **train, tune, test**. Alliterative and descriptive.  
Alliterscriptive.

we'll go with **train,**  
**tune, test** instead



2



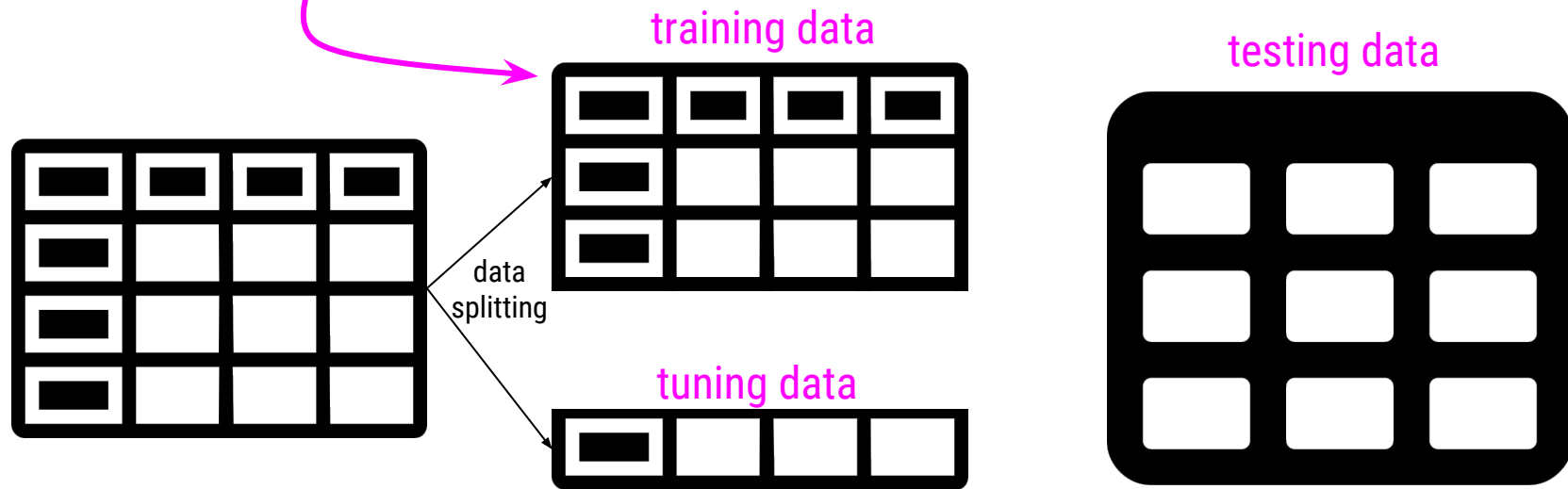
4

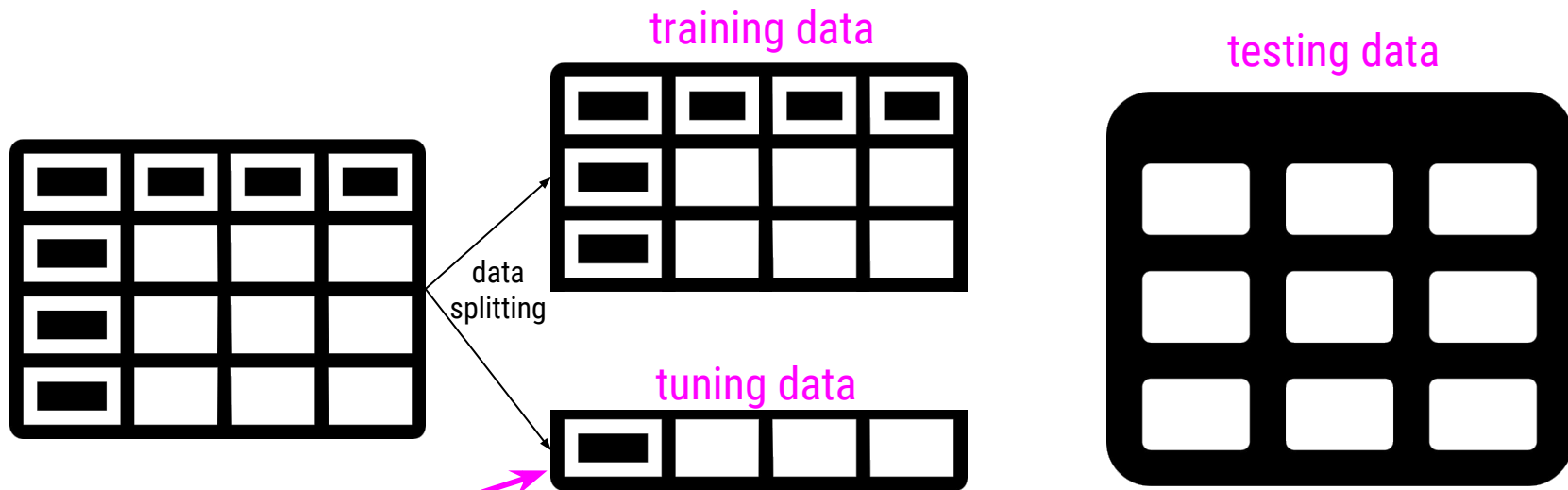


14



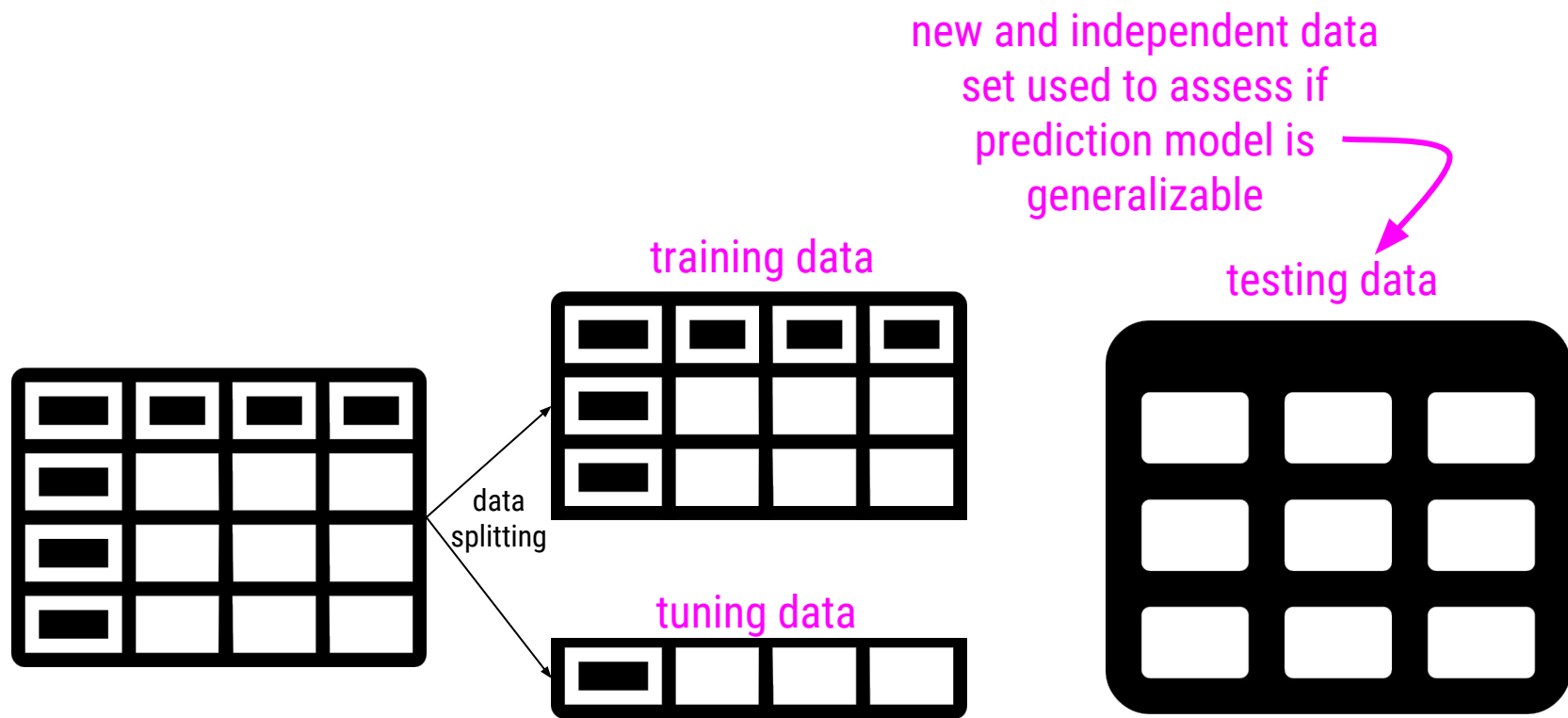
the data used to  
build your predictive  
model

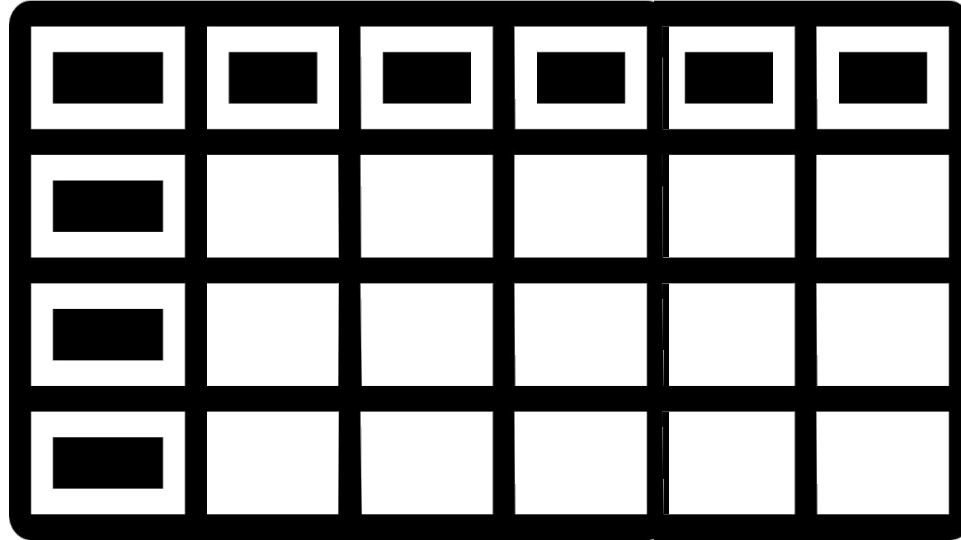




Data from original dataset that was held out and not used in training the model ; helpful in fine-tuning prediction accuracy



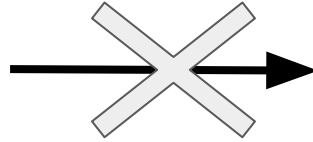
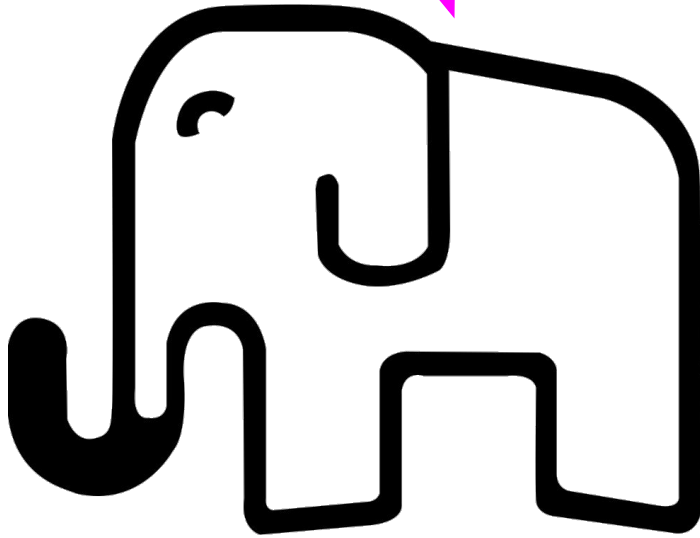


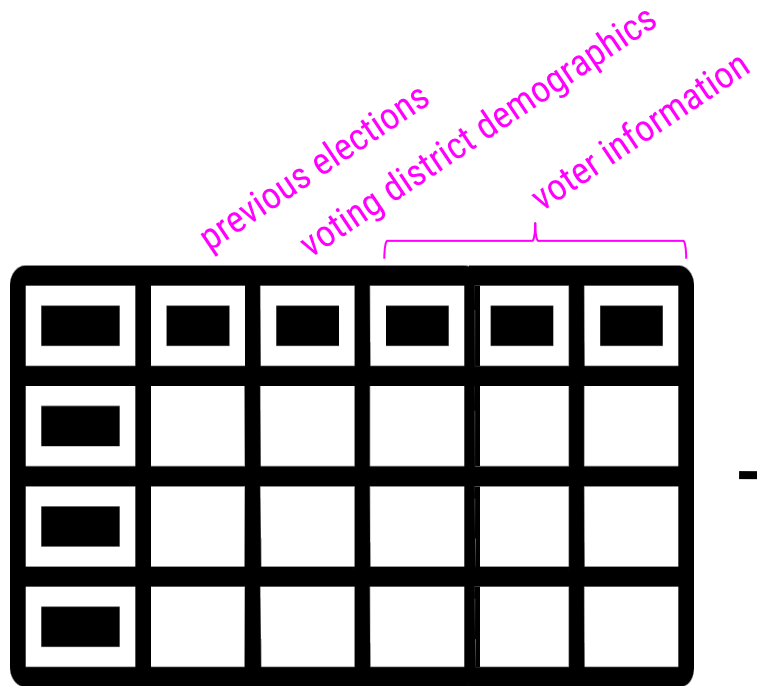


variable selection



elephant height data are  
likely not predictive of US  
elections



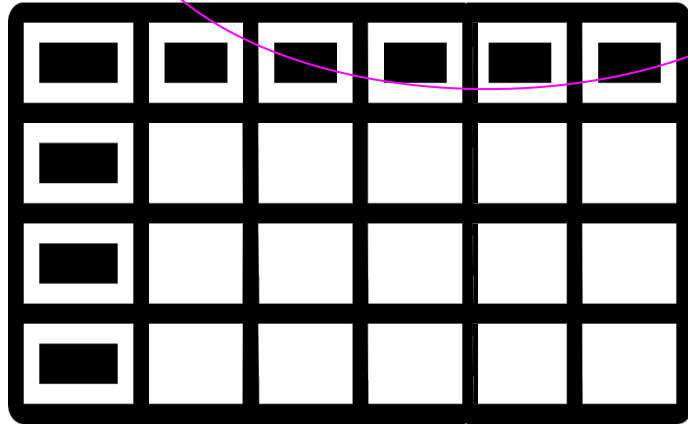











these data are likely  
predictive of US election  
outcomes





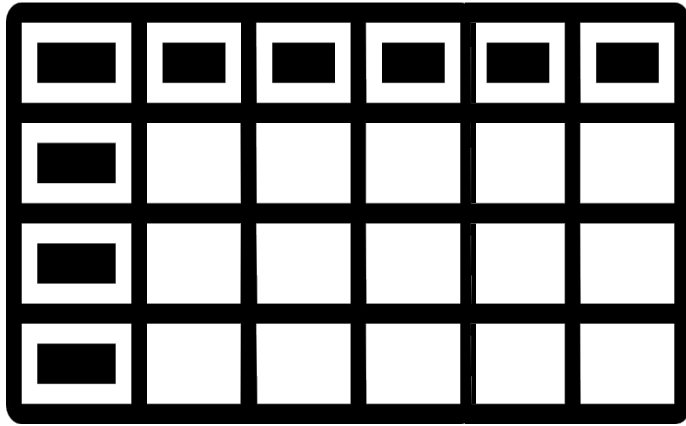
**variable selection** determines  
which variables are most predictive  
and includes them in the model



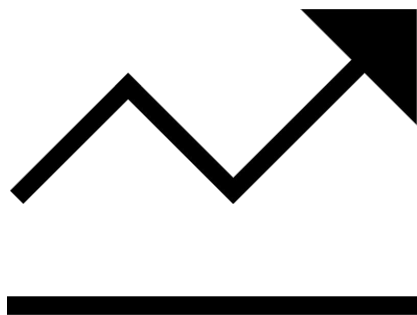
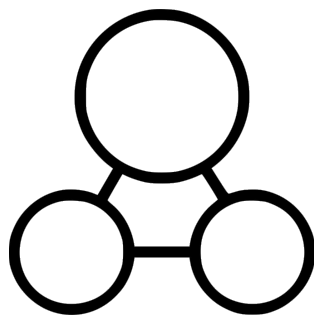
these data are likely  
predictive of US election  
outcomes



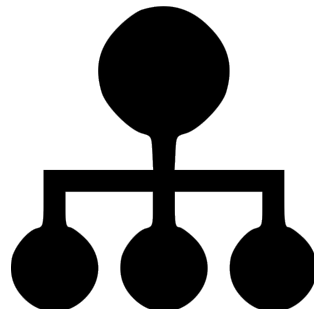


variables that can be used for accurate  
prediction exploit the relationship  
between the variables but do NOT mean  
that one causes the other

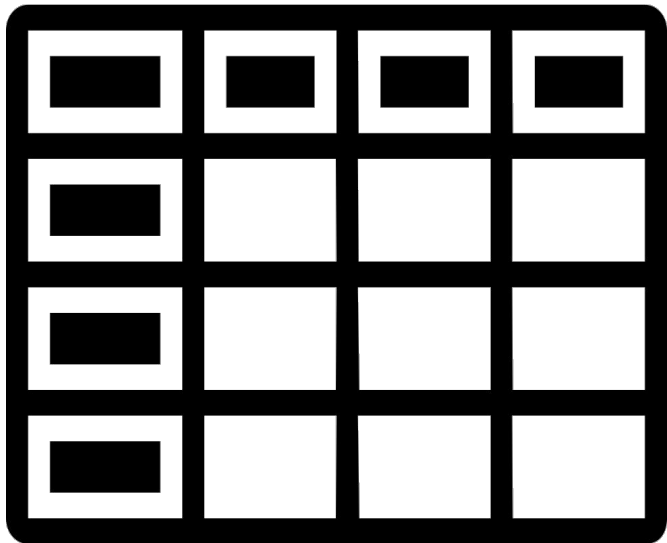




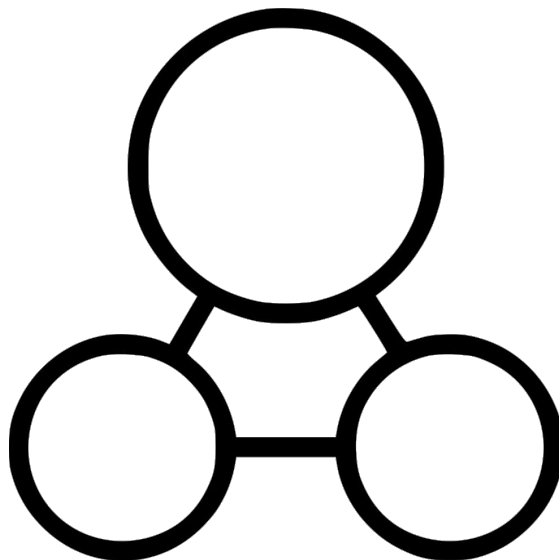
+	-
×	=



model selection

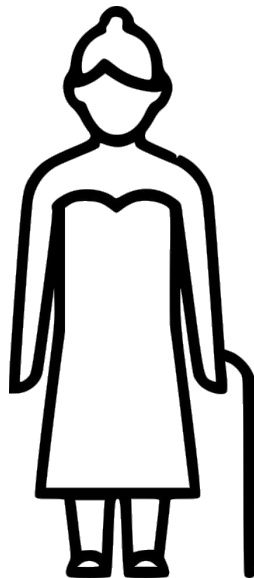


big  
datasets



simple  
models



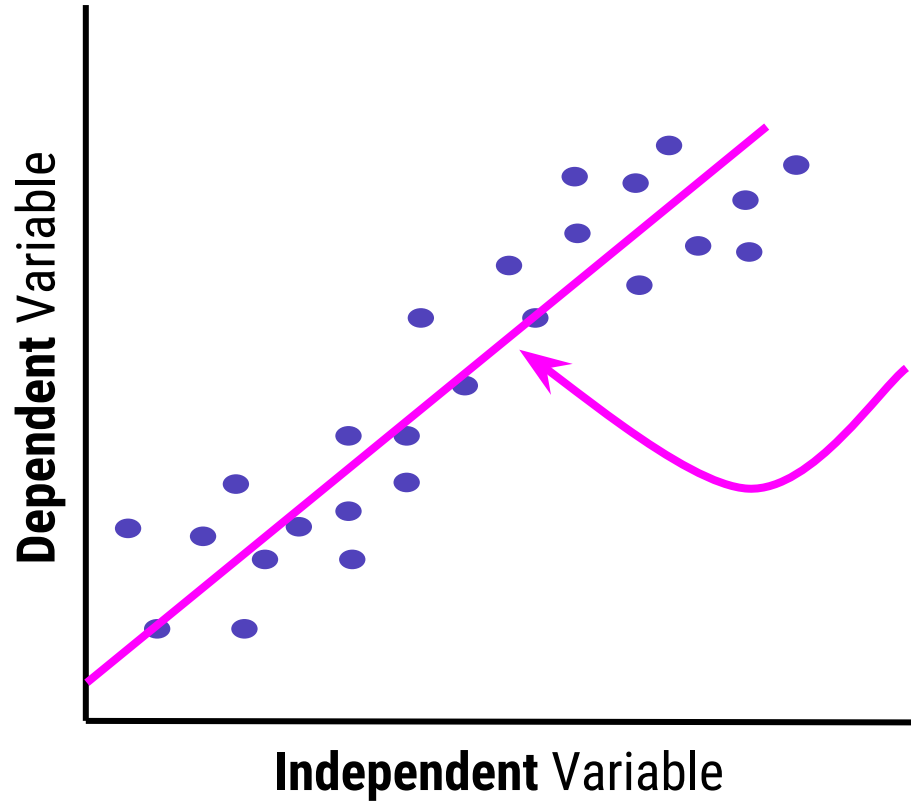


**Regression:**  
predicting continuous  
variables  
(i.e. Age)



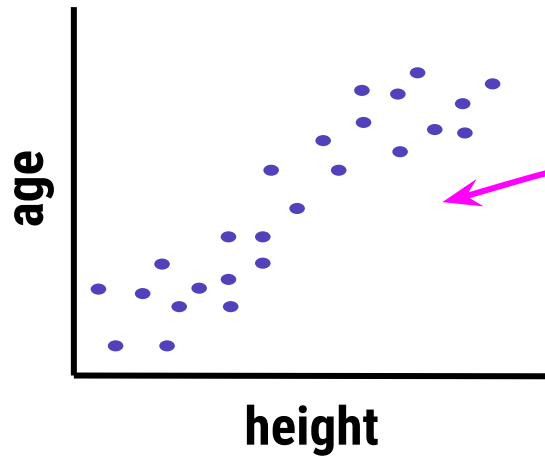
**Classification:**  
predicting categorical  
variables  
(i.e. education level)





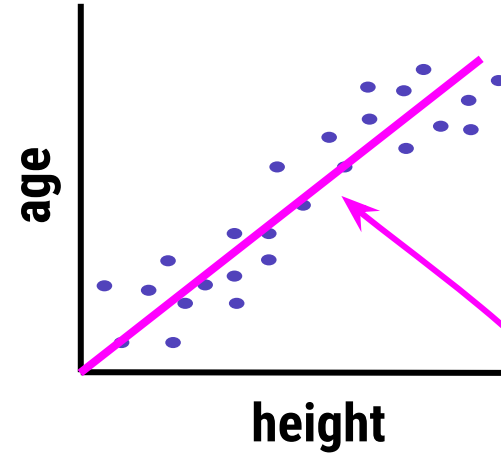
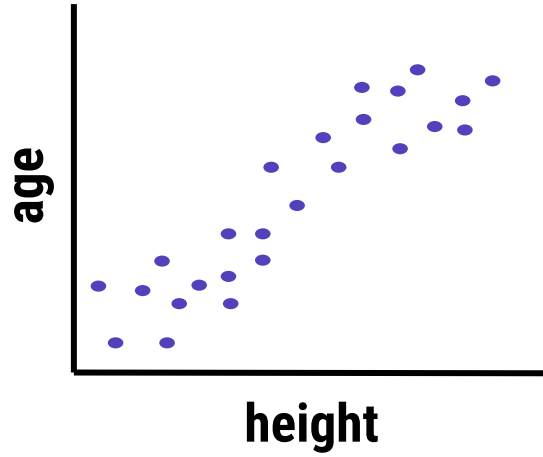
We'll use the  
linear relationship  
between variables  
to generate a  
**predictive model!**





the training data  
will be used to  
build the  
predictive model

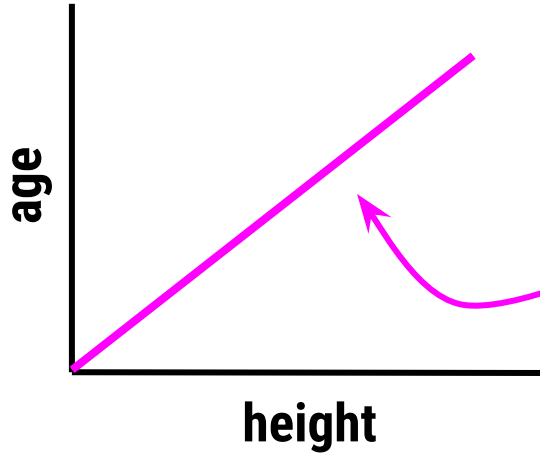
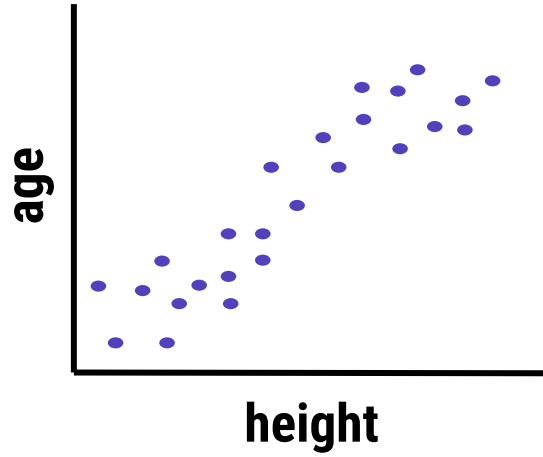




use **linear regression** to model the relationship

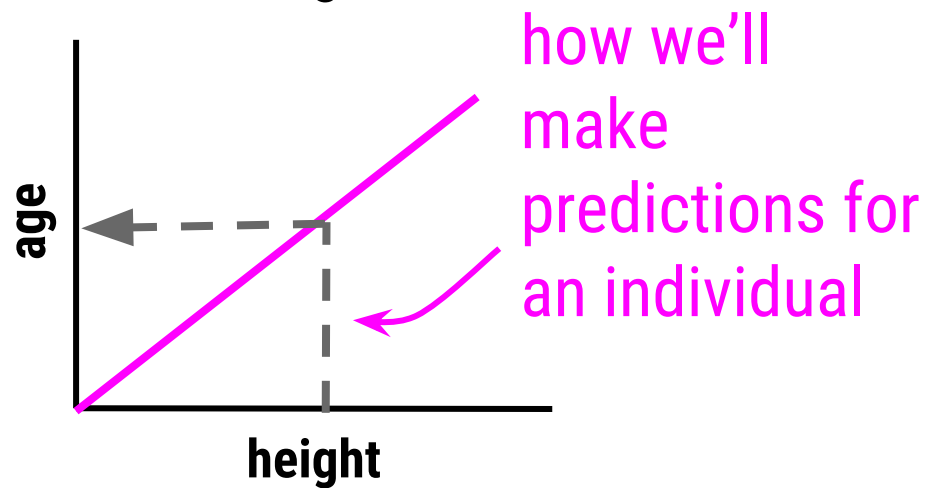
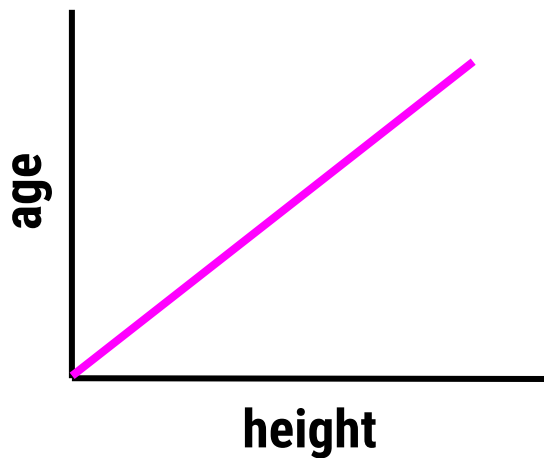
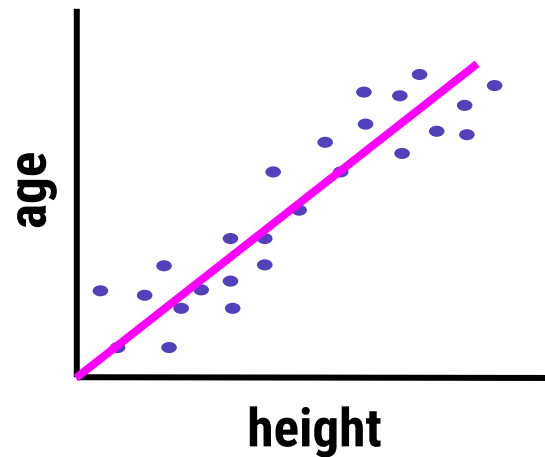
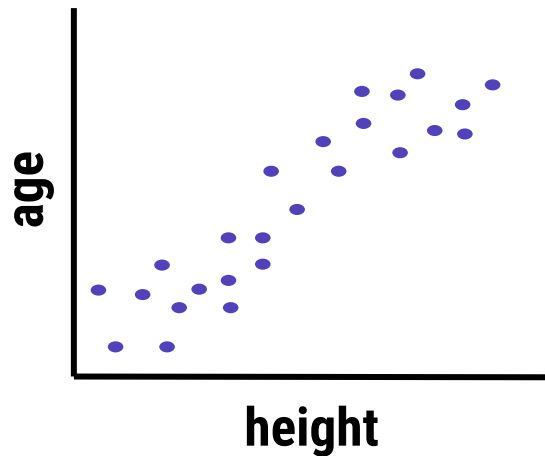


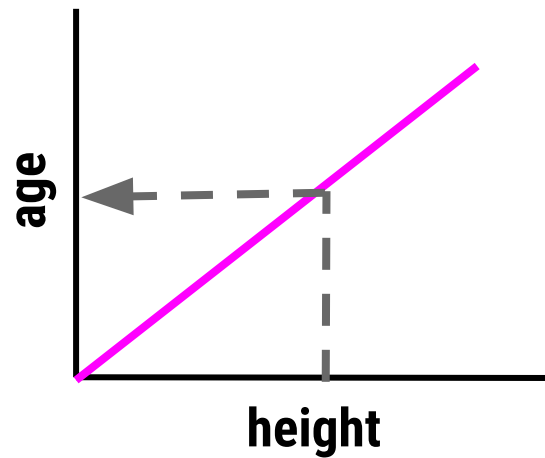
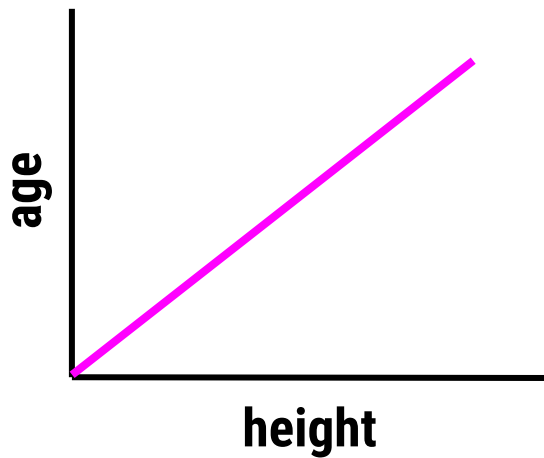
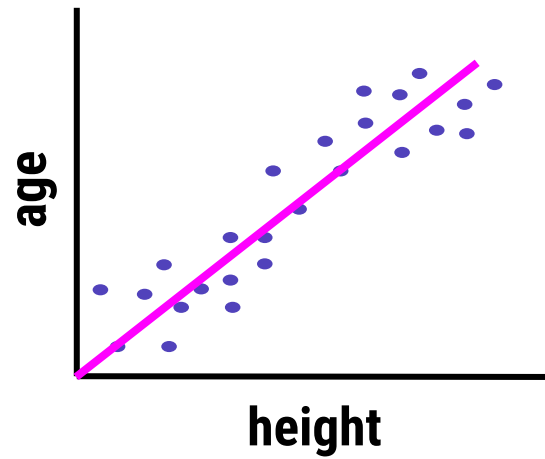
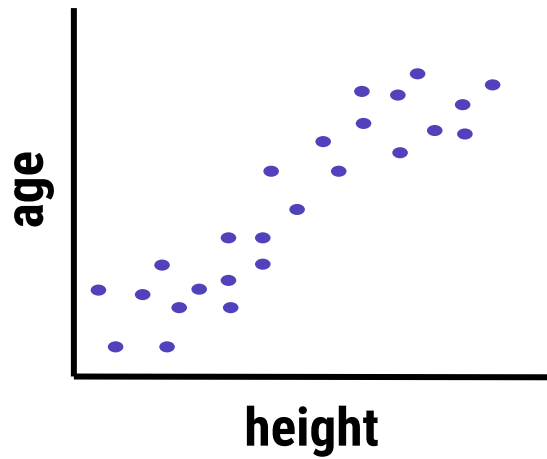




For prediction, the individual values in the training data are *not* important. We only need the model.







training data

Y

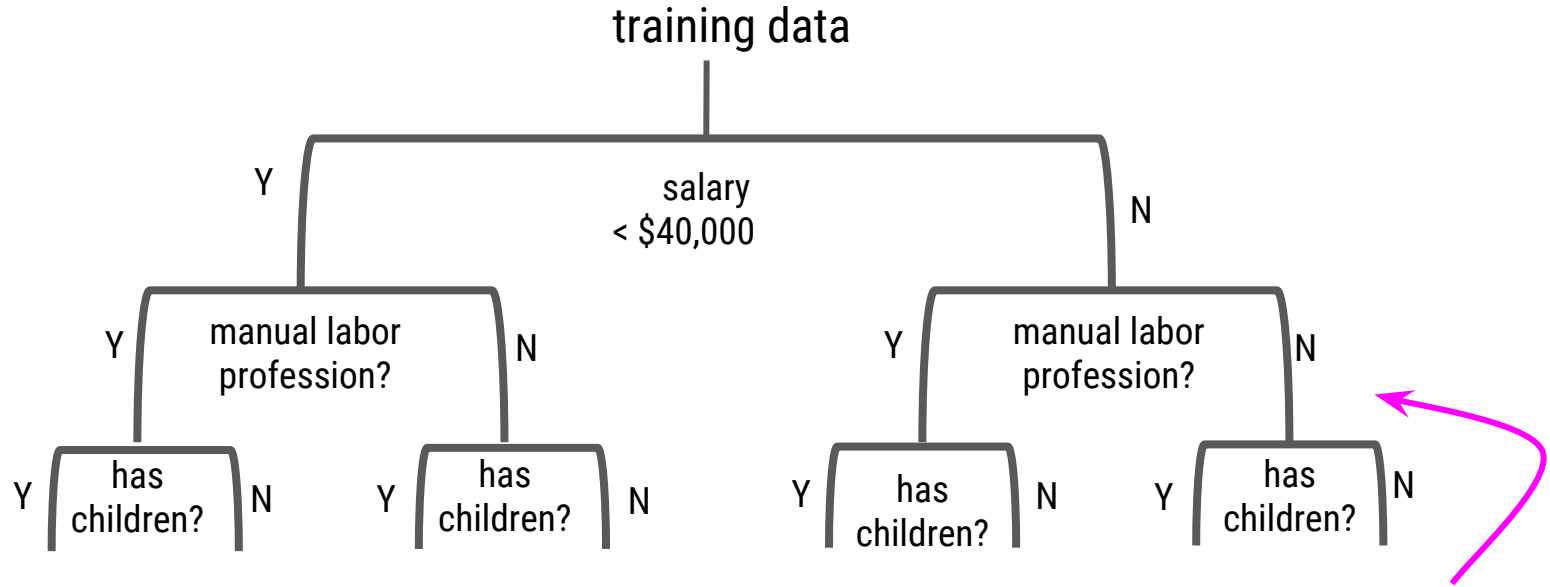
salary  
< \$40,000

N

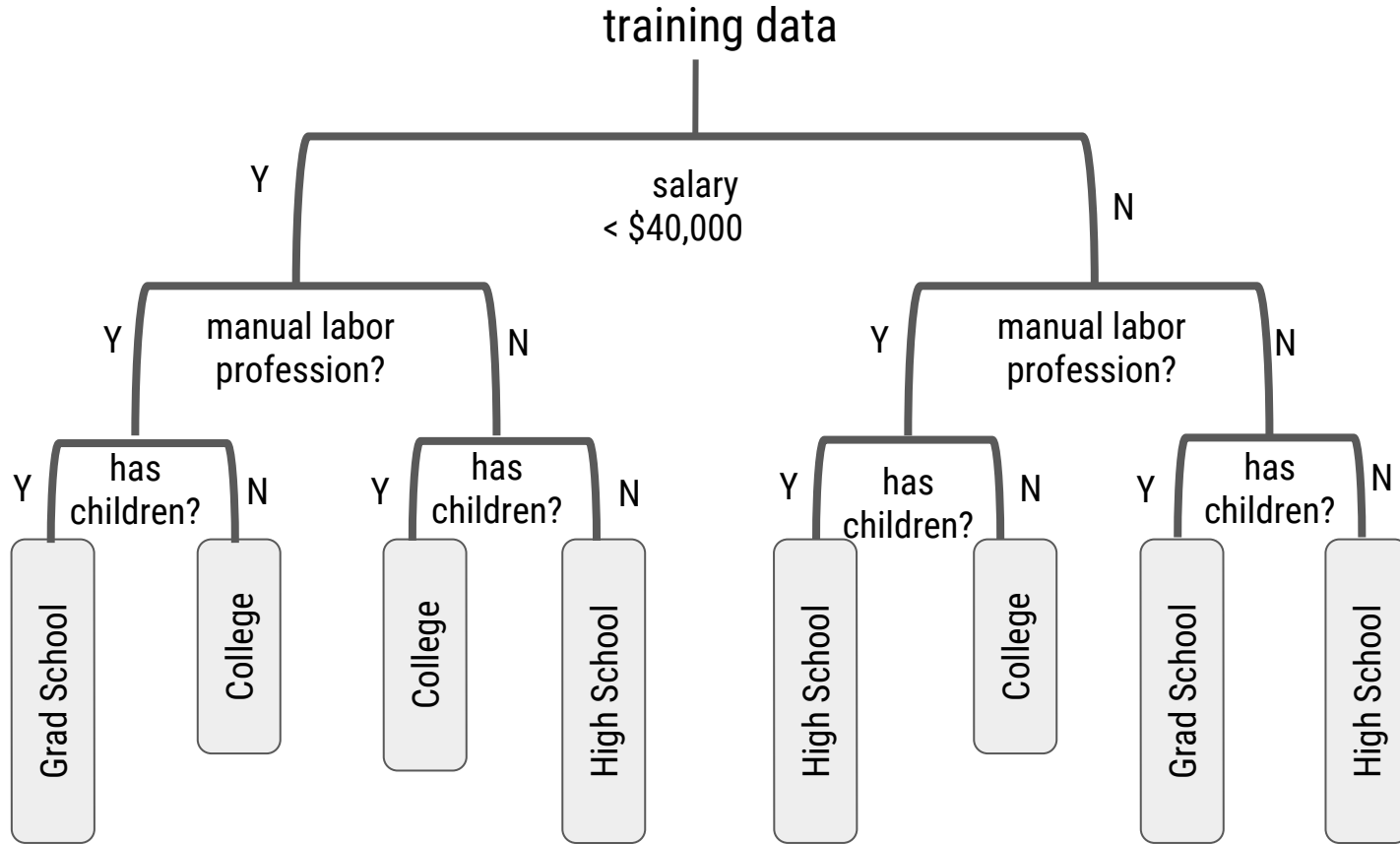
All the people  
who make  
*less* than 40K  
over here

All the people  
who make  
*more* than 40K  
over here

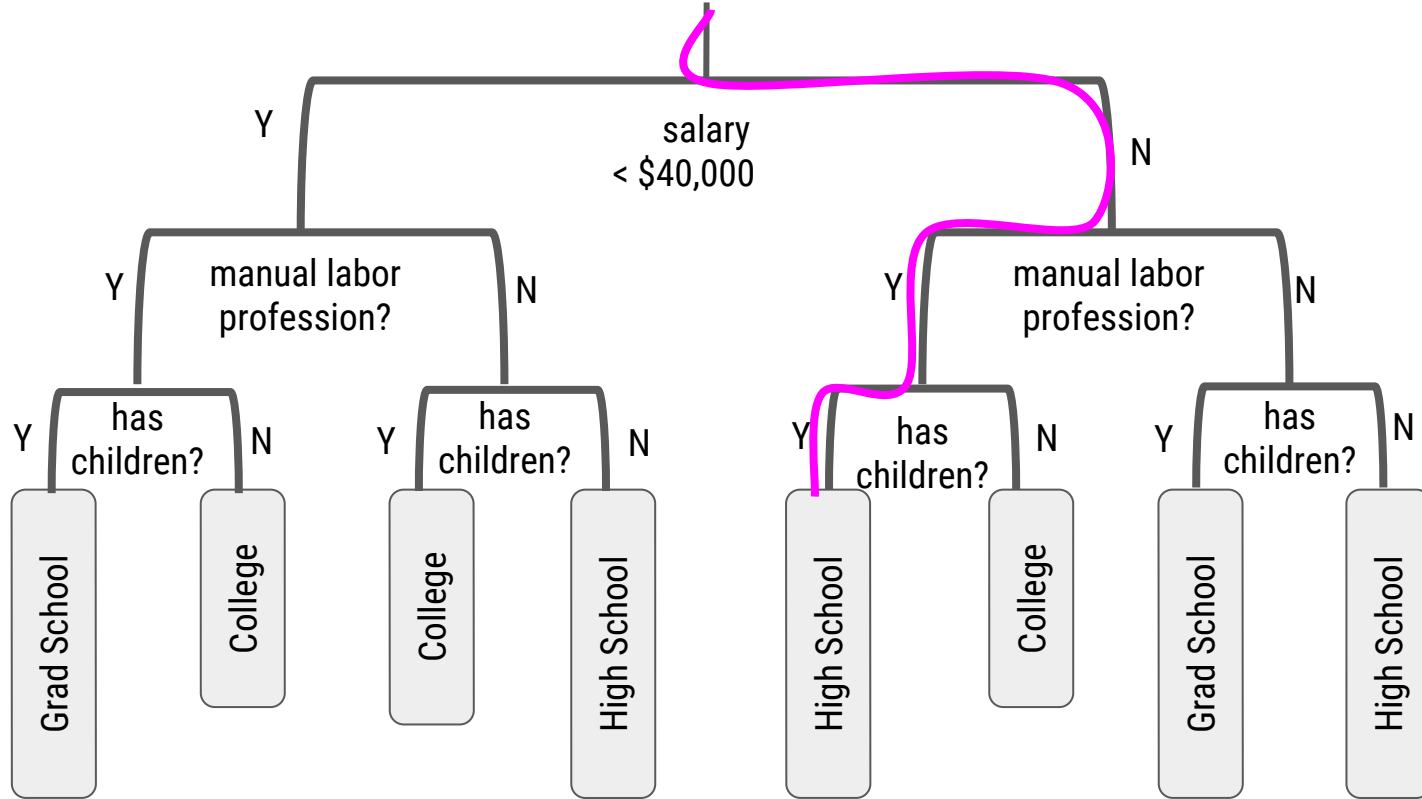


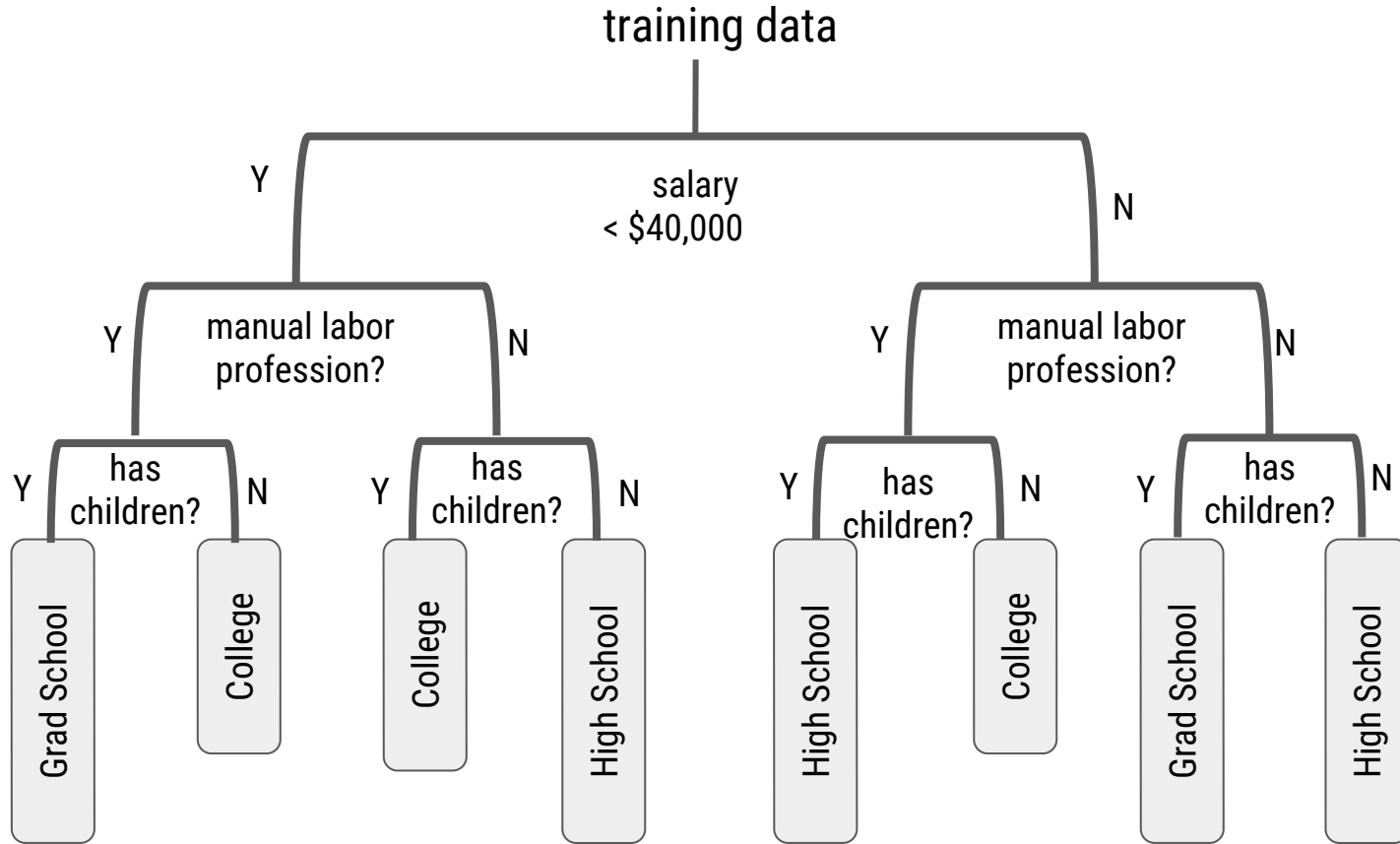


Continue building the decision tree where the variables and information in the training data decide who goes down which branch



training data









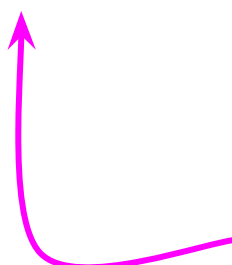
accuracy assessment



$$RMSE = \sqrt{\frac{\sum_{i=1}^N (\text{Predicted}_i - \text{Actual}_i)^2}{N}}$$



$$RMSE = \sqrt{\frac{\sum_{i=1}^N (\text{Predicted}_i - \text{Actual}_i)^2}{N}}$$



A few outliers can lead to a big increase in RMSE, even if all the other predictions are pretty good



$$\text{Accuracy} = \frac{\# \text{ of samples predicted correctly}}{\# \text{ of samples predicted}} * 100$$



1 Introduction

2 Visualizations

3 Pre-Processing

3.1 Creating Dummy Variables

3.2 Zero- and Near Zero-Variance Pr...

3.3 Identifying Correlated Predictors

3.4 Linear Dependencies

3.5 The preProcess Function

3.6 Centering and Scaling

3.7 Imputation

3.8 Transforming Predictors

3.9 Putting It All Together

3.10 Class Distance Calculations

4 Data Splitting

4.1 Simple Splitting Based on the O...

4.2 Splitting Based on the Predictors

4.3 Data Splitting for Time Series

4.4 Simple Splitting with Important ...

5 Model Training and Tuning

5.1 Model Training and Parameter T...

# The caret Package

Max Kuhn

2018-05-26

## 1 Introduction

The `caret` package (short for `_C_lassification_A_nd_RE_gression_T_raining`) is a set of functions that attempt to streamline the process for creating predictive models. The package contains tools for:

- data splitting
- pre-processing
- feature selection
- model tuning using resampling
- variable importance estimation

as well as other functionality.

There are many different modeling functions in R. Some have different syntax for model training and/or prediction. The package started off as a way to provide a uniform interface the functions themselves, as well as a way to standardize common tasks (such parameter tuning and variable importance).

# Edgar Anderson's Iris Data

## Description

This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are *Iris setosa*, *versicolor*, and *virginica*.

## Usage

```
iris  
iris3
```

## Format

`iris` is a data frame with 150 cases (rows) and 5 variables (columns) named `Sepal.Length`, `Sepal.Width`, `Petal.Length`, `Petal.Width`, and `Species`.

`iris3` gives the same data arranged as a 3-dimensional array of size 50 by 4 by 3, as represented by S-PLUS. The first dimension gives the case number within the species subsample, the second the measurements with names `Sepal L.`, `Sepal W.`, `Petal L.`, and `Petal W.`, and the third the species.

## Source

Fisher, R. A. (1936) The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, **7**, Part II, 179–188.

The data were collected by Anderson, Edgar (1935). The irises of the Gaspe Peninsula, *Bulletin of the American Iris Society*, **59**, 2–5.

## References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole. (has `iris3` as `iris`.)



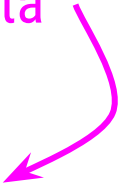
```
## install and load packages
```

```
install.packages("caret")
```

```
library(caret)
```

```
library(dplyr)
```

Specify to include 70%  
of the observations in  
the training data



```
## get Index for training set
```

```
set.seed(123)
```

```
trainIndex <- createDataPartition(iris$Species, p = .7,  
                                   list = FALSE,  
                                   times = 1)
```

```
## split into training and tuning set
```

```
iris_train <- iris %>% slice(trainIndex)
```

```
iris_tune <- iris %>% slice(-trainIndex)
```

```
## take a look
```

```
str(iris_train)
```

```
str(iris_tune)
```



70% of the observations  
are in the *training* data

```
> str(iris_train)
```

```
'data.frame':  105 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 4.6 4.4 5.4 4.8 4.3 5.8 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.4 2.9 3.7 3 3 4 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.4 1.5 1.4 1.1 1.2 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.3 0.2 0.2 0.1 0.1 0.2 ...
 $ Species      : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
```

```
> str(iris_tune)
```

```
'data.frame':  45 obs. of  5 variables:
 $ Sepal.Length: num  5 5.4 5 4.9 4.8 5.7 5.4 5.2 5.2 5.5 ...
 $ Sepal.Width : num  3.6 3.9 3.4 3.1 3.4 4.4 3.9 3.4 4.1 4.2 ...
 $ Petal.Length: num  1.4 1.7 1.5 1.5 1.6 1.5 1.3 1.4 1.5 1.4 ...
 $ Petal.Width : num  0.2 0.4 0.2 0.1 0.2 0.4 0.4 0.2 0.1 0.2 ...
 $ Species      : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
```

30% of the observations  
are in the *tuning* data





```
## train regression model
set.seed(123)
fit.lm <- train(Sepal.Length ~
Sepal.Width,
                data = iris,
                method = "lm",
                metric = "RMSE")
```



```
> ## look at RMSE
```

```
> fit.lm$results
```

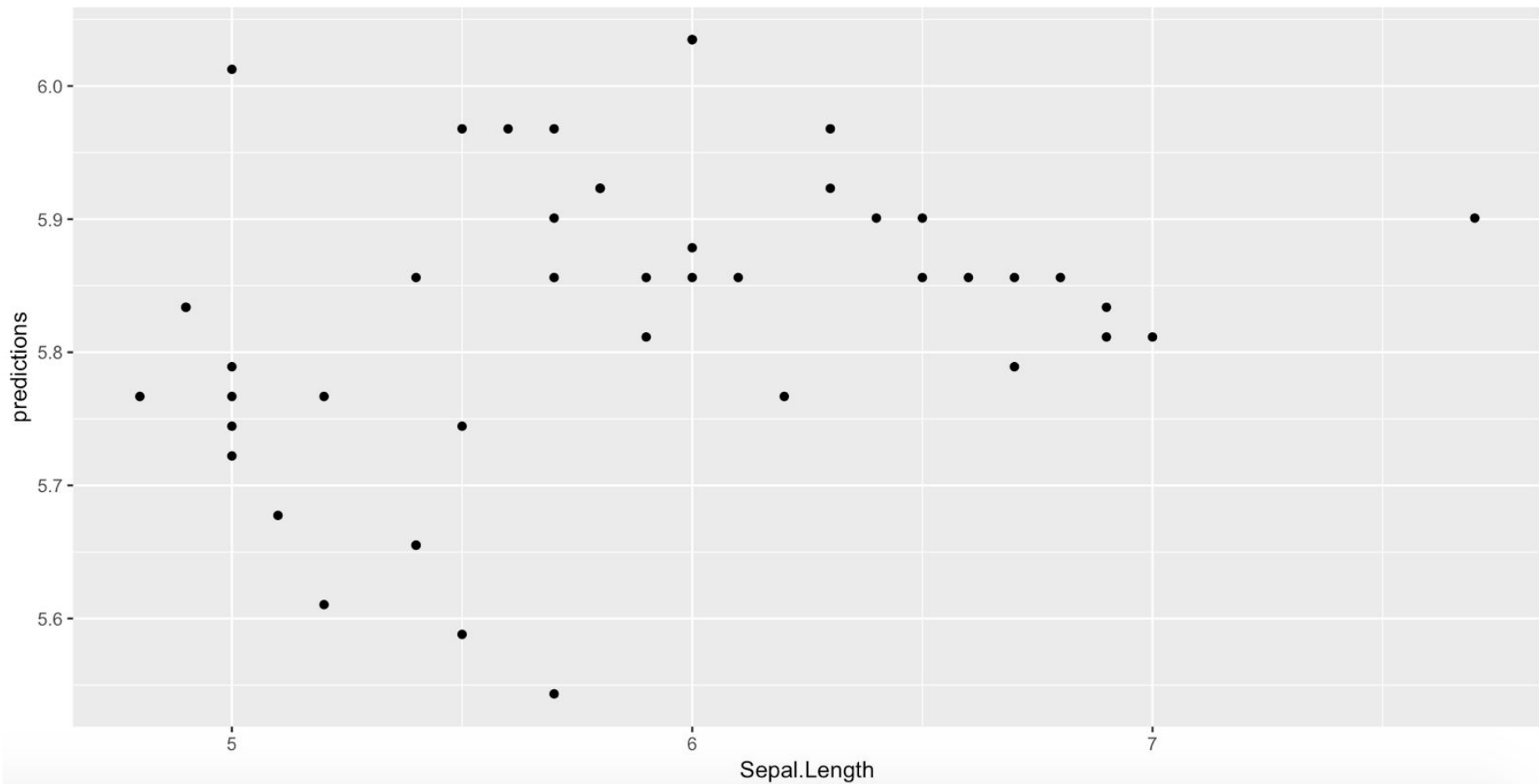
	intercept	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
1	TRUE	0.8206736	0.02384196	0.6755489	0.04703881	0.0306969	0.04749583



```
## make predictions in tuning data set
predictions <- predict(fit.lm, iris_tune)

## visualize results
iris_tune %>%
  mutate(predictions = predictions) %>%
  ggplot() +
  geom_point(aes(Sepal.Length, predictions))
```





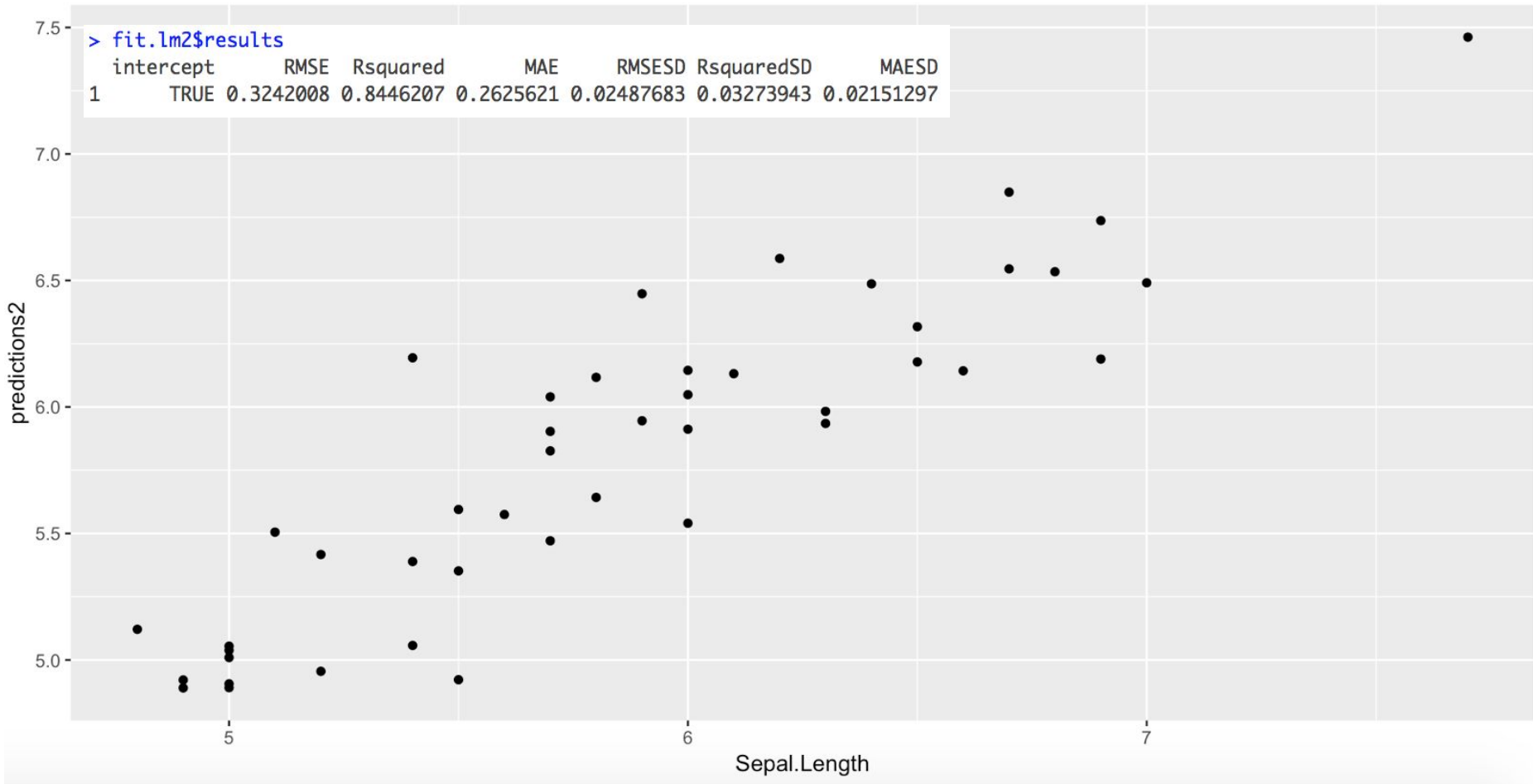
```
## train regression model
set.seed(123)
fit.lm <- train(Sepal.Length ~ .,
                data = iris,
                method = "lm",
                metric = "RMSE")

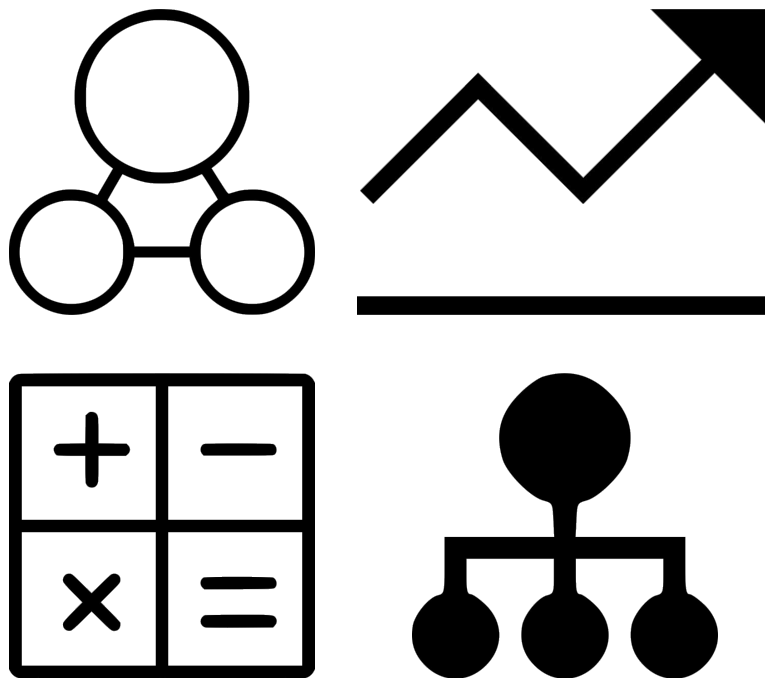
## look at RMSE
fit.lm2$results

## make predictions in tuning data set
predictions2 <- predict(fit.lm2, iris_tune)

## visualize results
iris_tune %>%
  mutate(predictions2 = predictions2) %>%
  ggplot() +
  geom_point(aes(Sepal.Length, predictions2))
```







model selection

```
## Random Forest
```

```
set.seed(7)
```

```
fit.rf <- train(Species~.,  
               data = iris,  
               method = "rf",  
               metric = "Accuracy")
```

rf specifies to run  
a random forest  
model



```
## look at Accuracy
```

```
fit.rf$results
```

```
## make predictions in tuning data set
```

```
predictions_rf <- predict(fit.rf, iris_tune)
```





predictions

```
> table(iris_tune$Species, predictions_rf)
```

predictions\_rf

setosa versicolor virginica

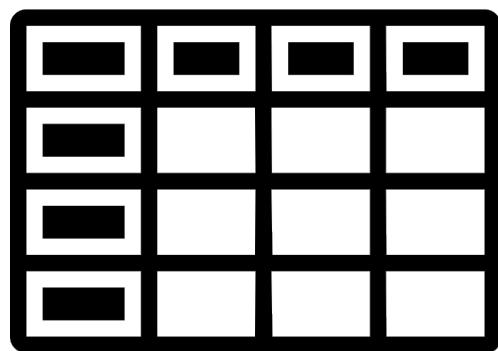
setosa 15 0 0

versicolor 0 15 0

virginica 0 0 15

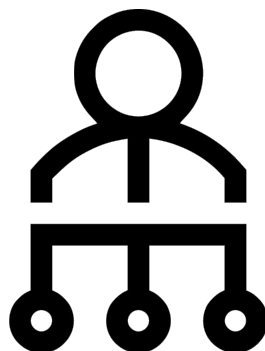
actual





data

train →



model

→ predict

