Training, test and validation splits

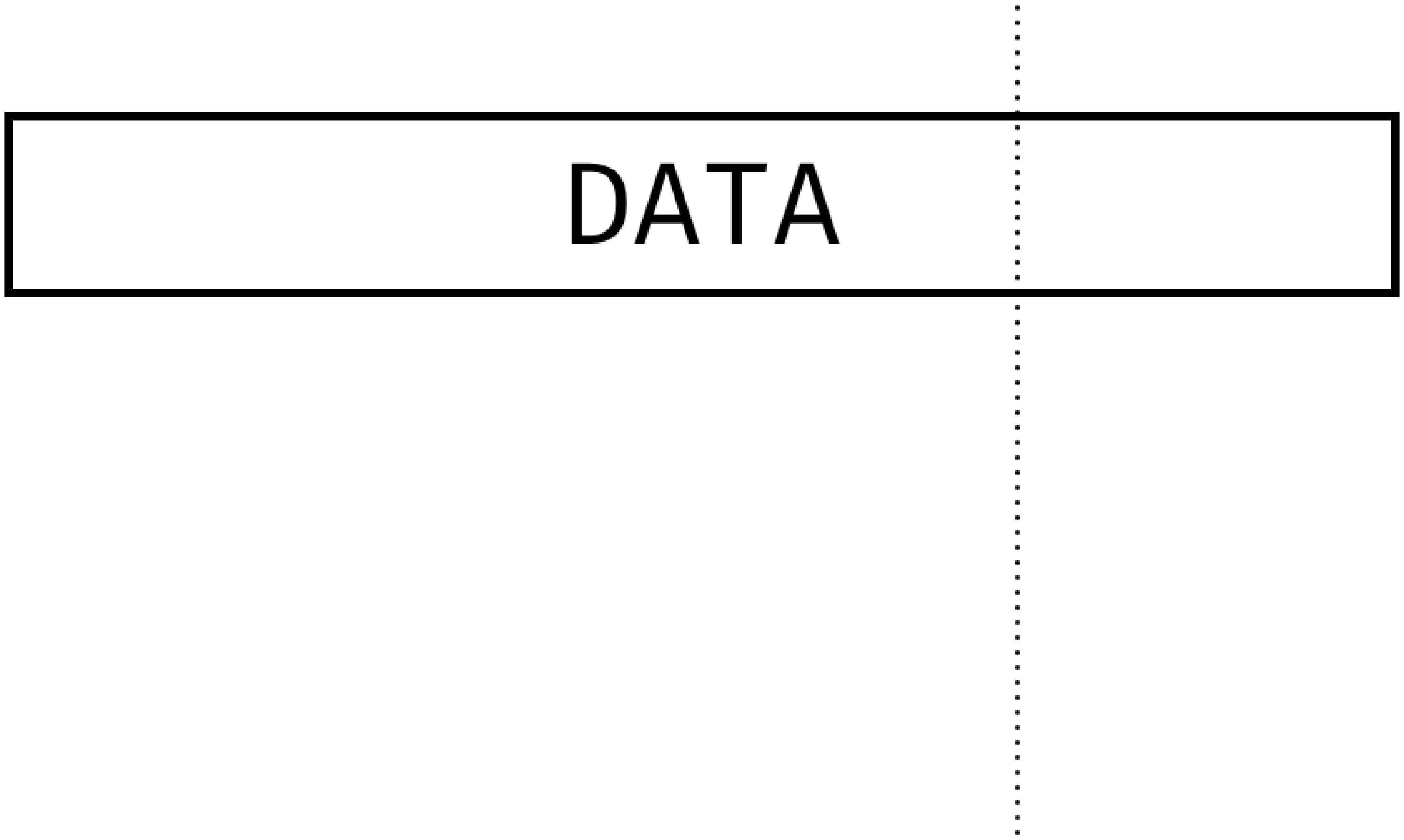
MACHINE LEARNING IN THE TIDYVERSE

Dmitriy (Dima) Gorenshteyn
Lead Data Scientist, Memorial Sloan Kettering Cancer Center

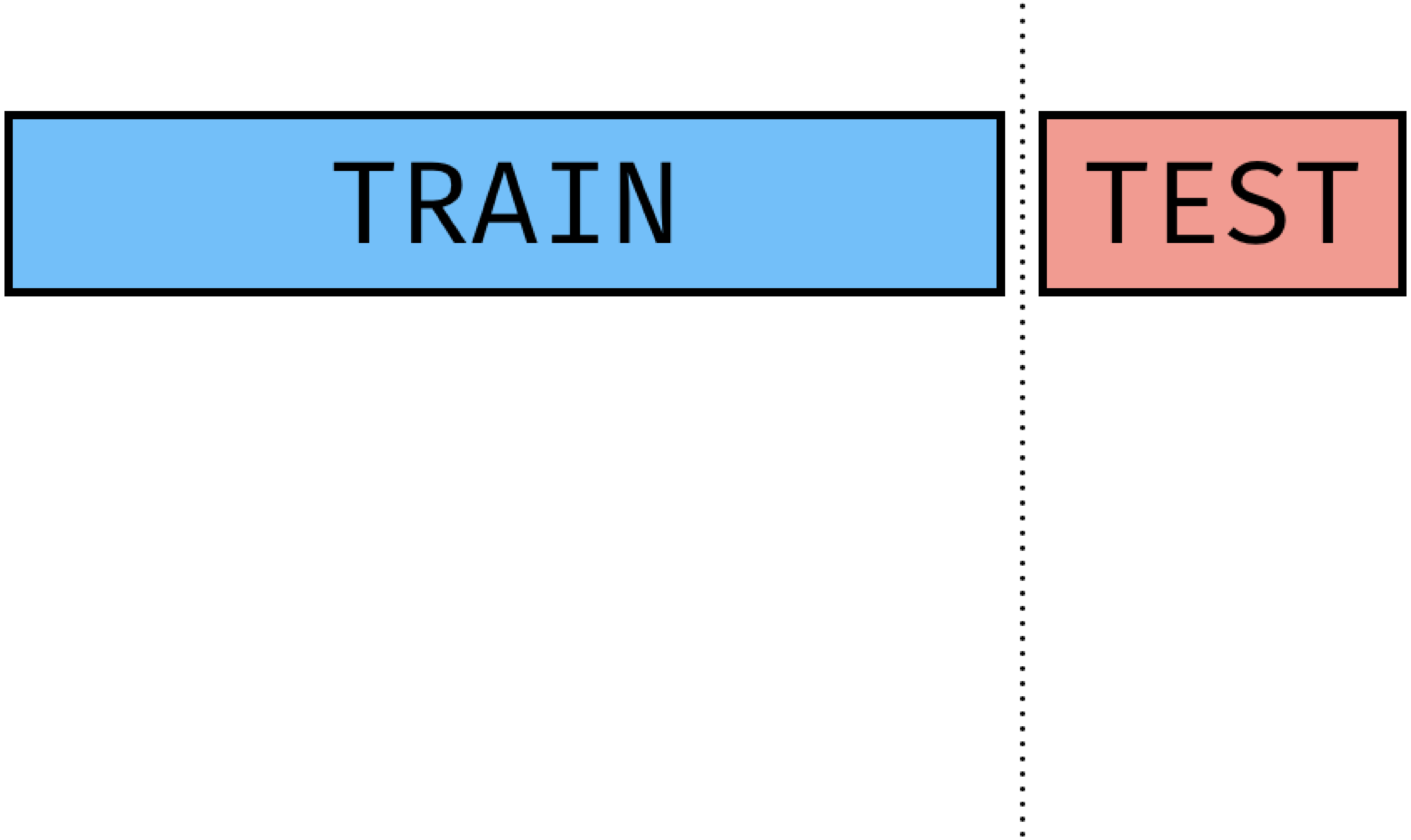
Train-Test Split

DATA

Train-Test Split



Train-Test Split



initial_split()

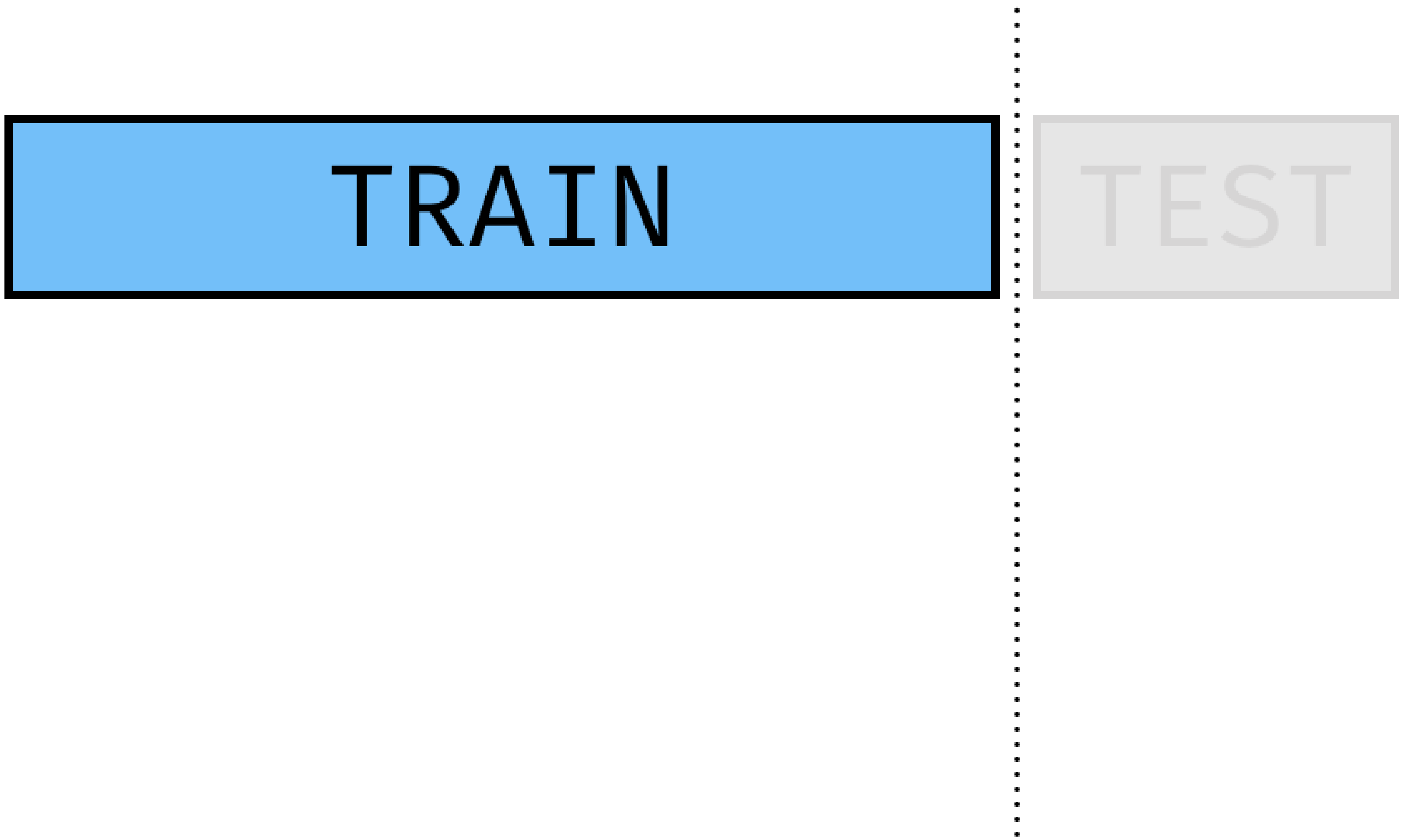
```
library(rsample)
gap_split <- initial_split(gapminder, prop = 0.75)
training_data <- training(gap_split)
testing_data <- testing(gap_split)
nrow(training_data)
```

```
3003
```

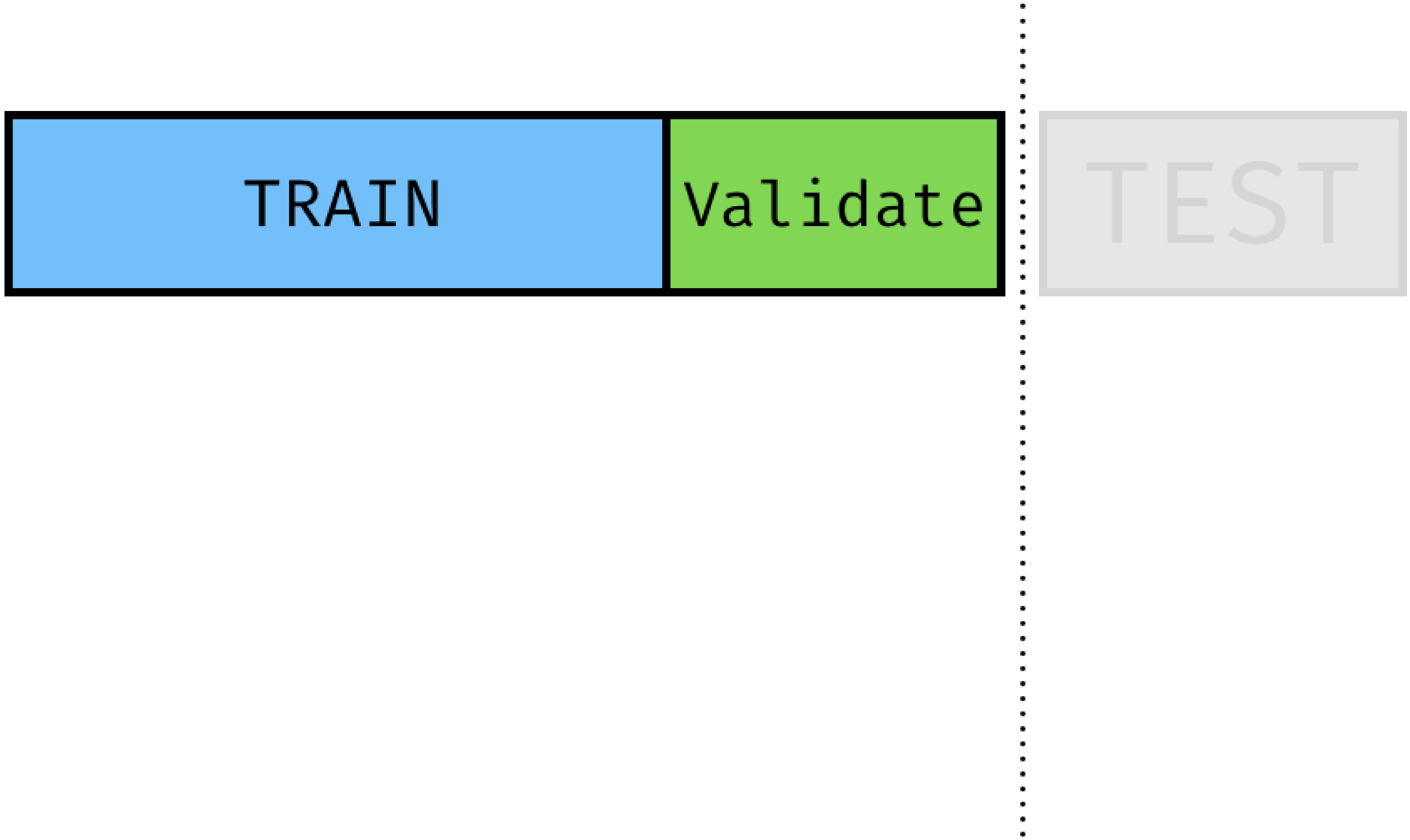
```
nrow(testing_data)
```

```
1001
```

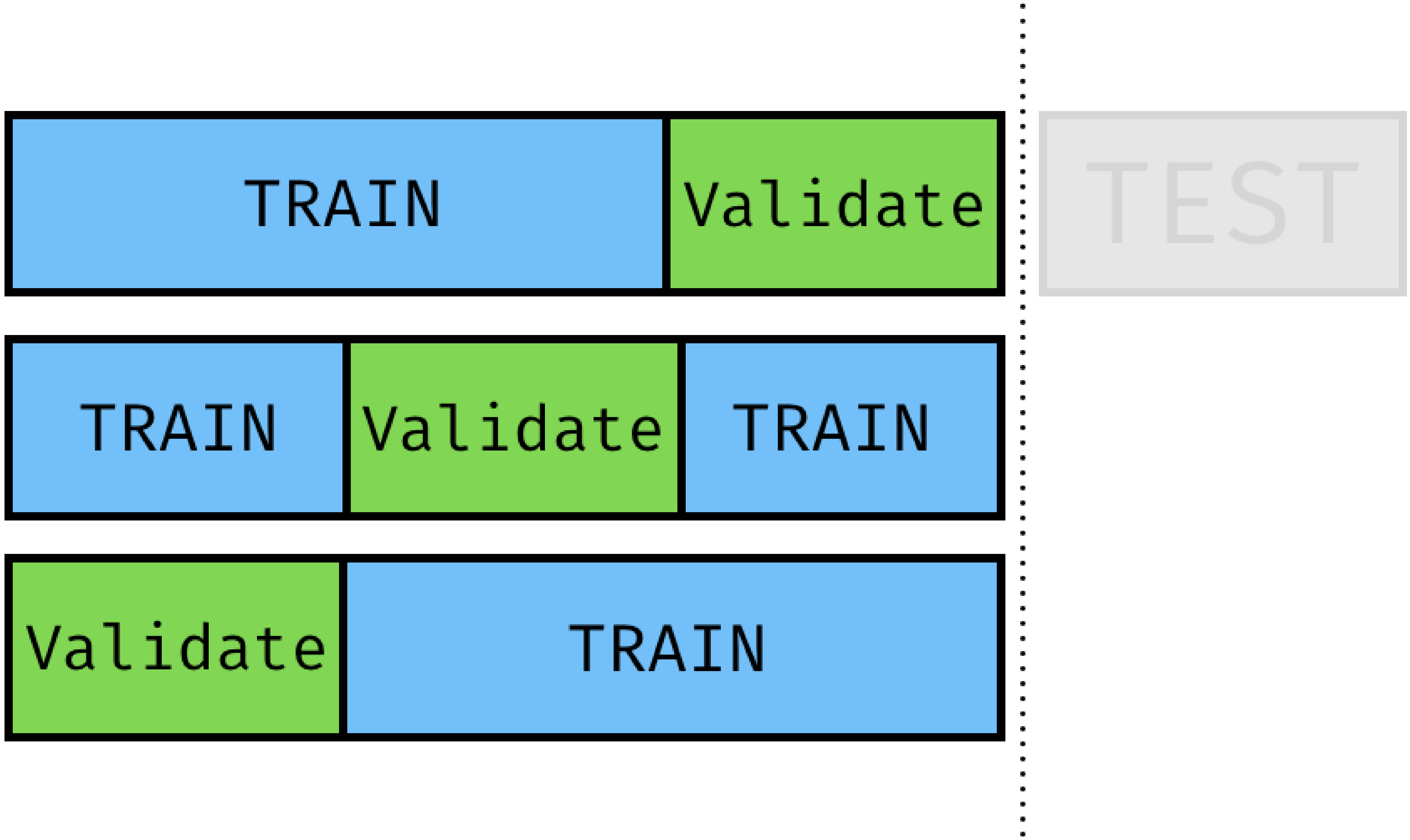
Train-Validate Split



Train-Validate Split



Cross Validation



vfold_cv()

```
library(rsample)
cv_split <- vfold_cv(training_data, v = 3)
cv_split
```

```
#  3-fold cross-validation
# A tibble: 3 x 2
  splits         id
  <list>         <chr>
1 <S3: rsplit>  Fold1
2 <S3: rsplit>  Fold2
3 <S3: rsplit>  Fold3
```

Mapping train & validate

```
cv_data <- cv_split %>%
  mutate(train = map(splits, ~training(.x)),
         validate = map(splits, ~testing(.x)))
```

Cross Validated Models

```
head(cv_data)
```

```
# A tibble: 3 x 4
  splits       id    train                validate
* <list>       <chr> <list>               <list>
1 <S3: rsplit> Fold1 <tibble [2,002 × 7]> <tibble [1,001 × 7]>
2 <S3: rsplit> Fold2 <tibble [2,002 × 7]> <tibble [1,001 × 7]>
3 <S3: rsplit> Fold3 <tibble [2,002 × 7]> <tibble [1,001 × 7]>
```

```
cv_models_lm <- cv_data %>%
  mutate(model = map(train, ~lm(formula = life_expectancy~., data = .x)))
```

Let's practice!

MACHINE LEARNING IN THE TIDYVERSE

Measuring cross-validation performance

MACHINE LEARNING IN THE TIDYVERSE

Dmitriy (Dima) Gorenshteyn
Lead Data Scientist, Memorial Sloan Kettering Cancer Center

Measuring Performance

life_expectancy	country	year	infant_mortality	fertility	population	gdpPercap
66.4	Peru	1986	67.6	4.25	19996250	2185
48.4	Senegal	1979	94.3	7.42	5424299	511
74	Paraguay	2006	23.1	3.19	5882797	1423
77.7	France	1993	6.3	1.72	57749881	19251
75.2	Netherlands	1977	9.7	1.58	13827329	15174
66.2	Panama	1969	53.2	5.28	1476478	2628

Measuring Performance - Truth

life_expectancy	country	year	infant_mortality	fertility	population	gdpPercap
66.4	Peru	1986	67.6	4.25	19996250	2185
48.4	Senegal	1979	94.3	7.42	5424299	511
74	Paraguay	2006	23.1	3.19	5882797	1423
77.7	France	1993	6.3	1.72	57749881	19251
75.2	Netherlands	1977	9.7	1.58	13827329	15174
66.2	Panama	1969	53.2	5.28	1476478	2628

Measuring Performance - Truth

life_expectancy	country	year	infant_mortality	fertility	population	gdpPercap
66.4	Peru	1986	67.6	4.25	19996250	2185
48.4	Senegal	1979	94.3	7.42	5424299	511
74	Paraguay	2006	23.1	3.19	5882797	1423
77.7	France	1993	6.3	1.72	57749881	19251
75.2	Netherlands	1977	9.7	1.58	13827329	15174
66.2	Panama	1969	53.2	5.28	1476478	2628

Measuring Performance - Truth

life_expectancy	country	year	infant_mortality	fertility	population	gdpPercap
66.4	Peru	1986	67.6	4.25	19996250	2185
48.4	Senegal	1979	94.3	7.42	5424299	511
74	Paraguay	2006	23.1	3.19	5882797	1423
77.7	France	1993	6.3	1.72	57749881	19251
75.2	Netherlands	1977	9.7	1.58	13827329	15174
66.2	Panama	1969	53.2	5.28	1476478	2628

Actual
66.4
48.4
74
77.7
75.2
66.2

Measuring Performance - Prediction

life_expectancy
66.4
48.4
74
77.7
75.2
66.2

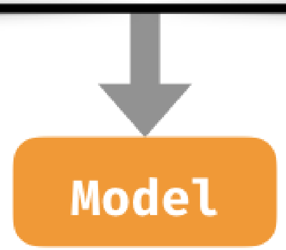
country	year	infant_mortality	fertility	population	gdpPercap
Peru	1986	67.6	4.25	19996250	2185
Senegal	1979	94.3	7.42	5424299	511
Paraguay	2006	23.1	3.19	5882797	1423
France	1993	6.3	1.72	57749881	19251
Netherlands	1977	9.7	1.58	13827329	15174
Panama	1969	53.2	5.28	1476478	2628

Actual
66.4
48.4
74
77.7
75.2
66.2

Measuring Performance - Prediction

life_expectancy
66.4
48.4
74
77.7
75.2
66.2

country	year	infant_mortality	fertility	population	gdpPercap
Peru	1986	67.6	4.25	19996250	2185
Senegal	1979	94.3	7.42	5424299	511
Paraguay	2006	23.1	3.19	5882797	1423
France	1993	6.3	1.72	57749881	19251
Netherlands	1977	9.7	1.58	13827329	15174
Panama	1969	53.2	5.28	1476478	2628



Actual
66.4
48.4
74
77.7
75.2
66.2

Measuring Performance - Prediction

life_expectancy	country	year	infant_mortality	fertility	population	gdpPercap
66.4	Peru	1986	67.6	4.25	19996250	2185
48.4	Senegal	1979	94.3	7.42	5424299	511
74	Paraguay	2006	23.1	3.19	5882797	1423
77.7	France	1993	6.3	1.72	57749881	19251
75.2	Netherlands	1977	9.7	1.58	13827329	15174
66.2	Panama	1969	53.2	5.28	1476478	2628

Model

Actual
66.4
48.4
74
77.7
75.2
66.2

Predicted
70.0
47
74
80.0
75.0
58.0

Measuring Performance

Actual
66.4
48.4
74
77.7
75.2
66.2

COMPARE

Predicted
70.0
47
74
80.0
75.0
58.0

Mean Absolute Error

Actual
66.4
48.4
74
77.7
75.2
66.2

Predicted
70.0
47
74
80.0
75.0
58.0

$$MAE = \frac{\sum_{i=1}^{n} \left| Actual_i - Predicted_i \right|}{n}$$

Ingredients for Performance Measurement

1) Actual `life_expectancy` values
2) Predicted `life_expectancy` values
3) A metric to compare 1) & 2)

1) Extract the actual values

```
cv_prep_lm <- cv_models_lm %>%
  mutate(validate_actual = map(validate, ~.x$life_expectancy))
```

The predict() & map2() functions

```
predict(model, data)
```

```
map2(.x = model, .y = data, .f = ~predict(.x, .y))
```

2) Prepare the predicted values

```
cv_prep_lm <- cv_eval_lm %>%
  mutate(validate_actual = map(validate, ~.x$life_expectancy),
         validate_predicted = map2(model, validate, ~predict(.x, .y)))
```

3) Calculate MAE

```
library(Metrics)
cv_eval_lm <- cv_prep_lm %>%
  mutate(validate_mae = map2_dbl(validate_actual, validate_predicted,
                                 ~mae(actual = .x, predicted = .y)))
cv_eval_lm
```

```
#  5-fold cross-validation
# A tibble: 5 x 8
splits        id    train validate model validate_a. validate_p validate_mae
<S3: rsplit> Fold1 <tib. <tib.   <S3.   <dbl.       <dbl.       1.47
<S3: rsplit> Fold2 <tib. <tib.   <S3.   <dbl.       <dbl.       1.51
<S3: rsplit> Fold3 <tib. <tib.   <S3.   <dbl.       <dbl.       1.44
<S3: rsplit> Fold4 <tib. <tib.   <S3.   <dbl.       <dbl.       1.48
<S3: rsplit> Fold5 <tib. <tib.   <S3.   <dbl.       <dbl.       1.68
```

Let's practice!

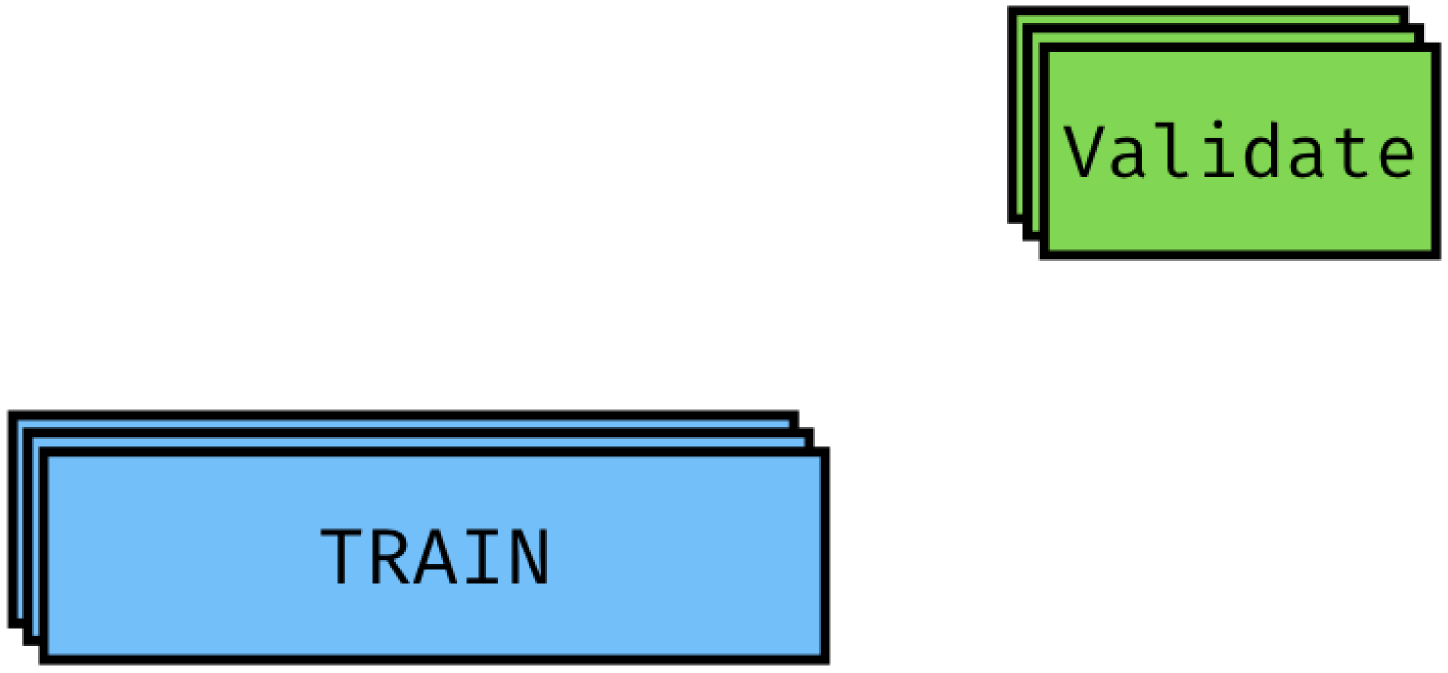
MACHINE LEARNING IN THE TIDYVERSE

Building and tuning a random forest model

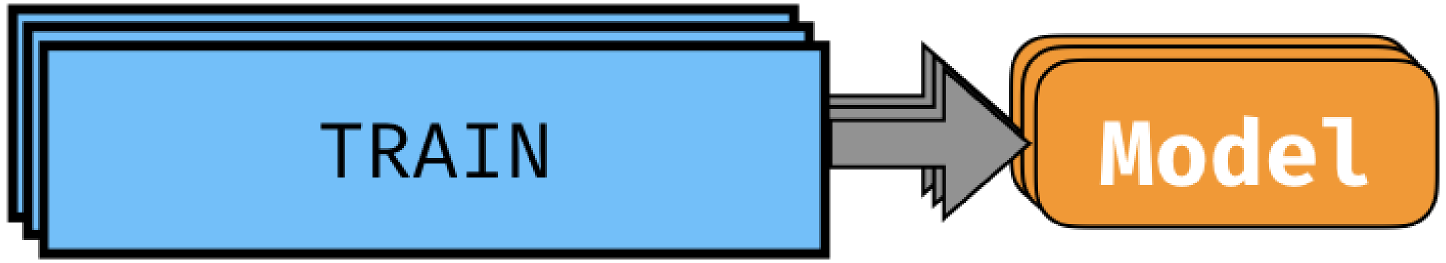
MACHINE LEARNING IN THE TIDYVERSE

Dmitriy (Dima) Gorenshteyn
Lead Data Scientist, Memorial Sloan Kettering Cancer Center

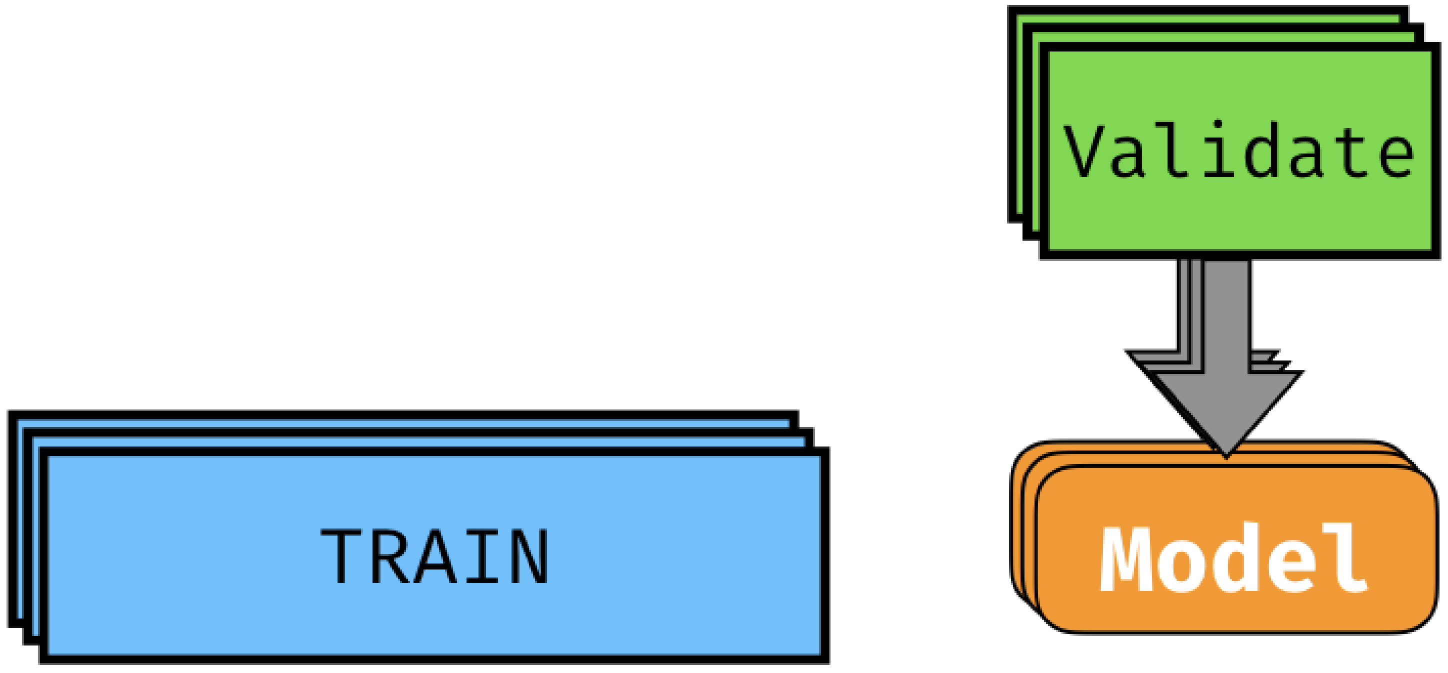
Cross Validation Performance



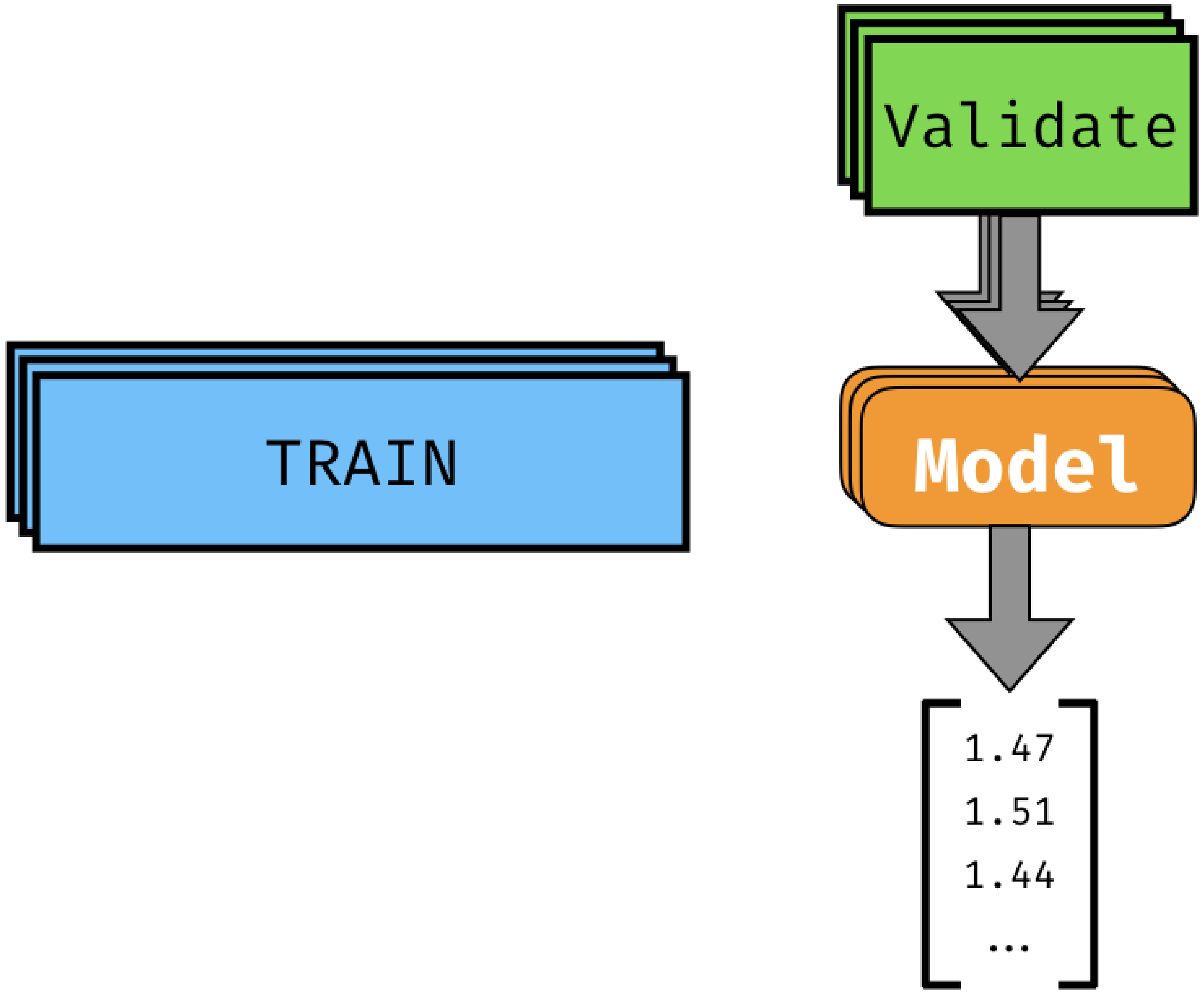
Cross Validation Performance



Cross Validation Performance



Cross Validation Performance

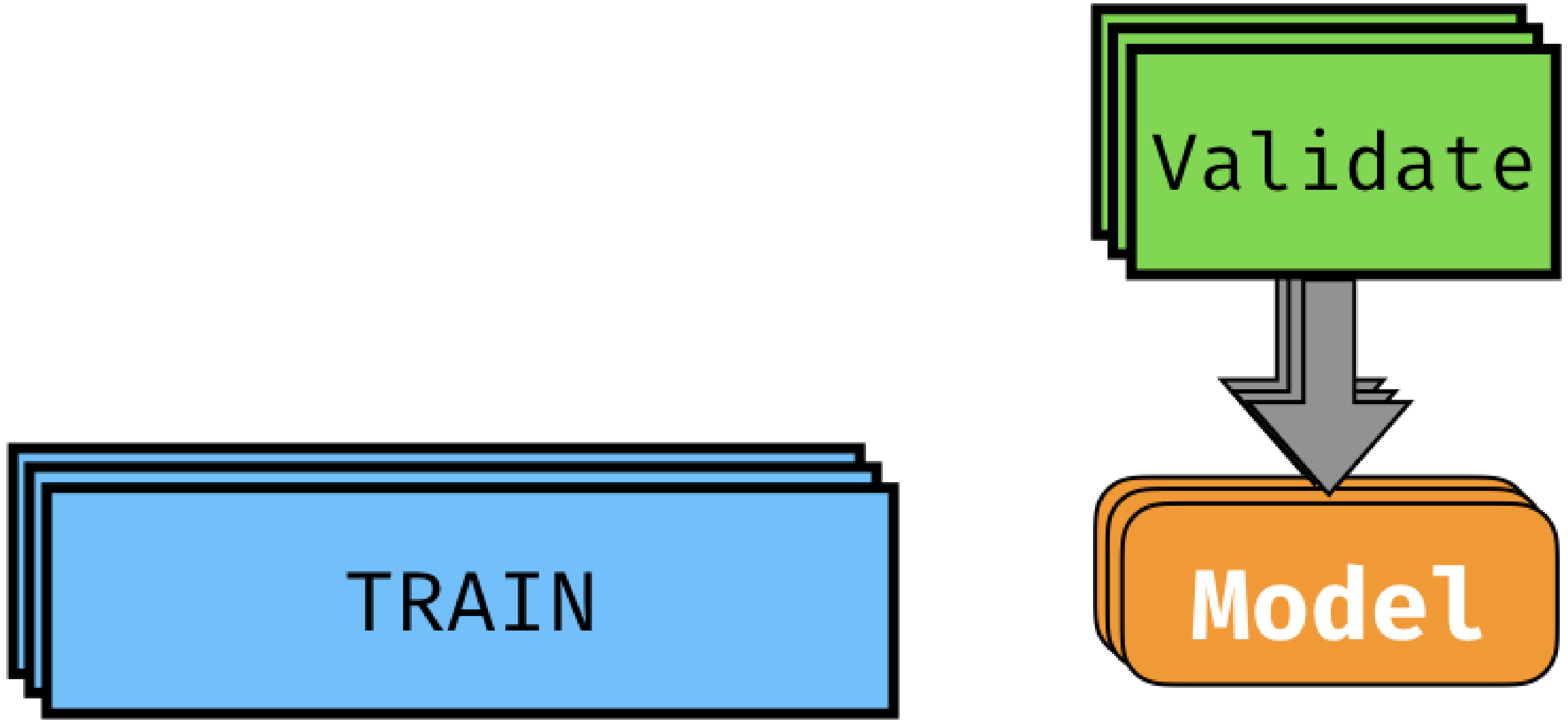


Linear Regression Model

Validate Mean Absolute Error:

1.5 Years

Another Model



Random Forest Benefits

- Can handle non-linear relationships
- Can handle interactions

Basic Random Forest Tools

Model

```
rf_model <- ranger(formula = ___, data = ___, seed = ___)
```

Prediction

```
prediction <- predict(rf_model, new_data)$predictions
```

Build Basic Random Forest Models

```
library(ranger)
cv_models_rf <- cv_data %>%
 mutate(model = map(train, ~ranger(formula = life_expectancy~.,
                                    data = .x, seed = 42)))
```

```
cv_prep_rf <- cv_models_rf %>%
 mutate(validate_predicted = map2(model, validate,
                                  ~predict(.x, .y)$predictions))
```

ranger Hyper-Parameters

Model

```
rf_model <- ranger(formula, data, seed, mtry, num.trees)
```

Hyper-Parameters

name	*range*	*default*
mtry	$1 : number\ of\ features$	$\sqrt{number\ of\ feat}$
num.trees	$1 : \infty$	500

Tune The Hyper-Parameters

```
cv_tune <- cv_data %>%
  crossing(mtry = 1:5)
cv_tune
```

```
# A tibble: 25 x 5
   splits         id    train                   validate              mtry
   <list>         <chr> <list>                  <list>               <int>
 1 <S3: rsplit> Fold1 <tibble [2,402 × 7]> <tibble [601 × 7]>      1
 2 <S3: rsplit> Fold1 <tibble [2,402 × 7]> <tibble [601 × 7]>      2
 3 <S3: rsplit> Fold1 <tibble [2,402 × 7]> <tibble [601 × 7]>      3
 4 <S3: rsplit> Fold1 <tibble [2,402 × 7]> <tibble [601 × 7]>      4
 5 <S3: rsplit> Fold1 <tibble [2,402 × 7]> <tibble [601 × 7]>      5
 6 <S3: rsplit> Fold2 <tibble [2,402 × 7]> <tibble [601 × 7]>      1
 7 <S3: rsplit> Fold2 <tibble [2,402 × 7]> <tibble [601 × 7]>      2
```

Tune The Hyper-Parameters

```
cv_model_tunerf <- cv_tune %>%
  mutate(model = map2(train, mtry, ~ranger(formula = life_expectancy~.,
                                           data = .x, mtry = .y)))
cv_model_tunerf
```

```
# A tibble: 25 x 6
   splits       id    train                  validate       mtry  model
 * <list>       <chr> <list>                 <list>         <int> <list>
 1 <S3: rsplit> Fold1 <tibble [2,402 × 7]> <tibble [60…   1     <S3: ranger>
 2 <S3: rsplit> Fold1 <tibble [2,402 × 7]> <tibble [60…   2     <S3: ranger>
 3 <S3: rsplit> Fold1 <tibble [2,402 × 7]> <tibble [60…   3     <S3: ranger>
 4 <S3: rsplit> Fold1 <tibble [2,402 × 7]> <tibble [60…   4     <S3: ranger>
 5 <S3: rsplit> Fold1 <tibble [2,402 × 7]> <tibble [60…   5     <S3: ranger>
 6 <S3: rsplit> Fold2 <tibble [2,402 × 7]> <tibble [60…   1     <S3: ranger>
 7 <S3: rsplit> Fold2 <tibble [2,402 × 7]> <tibble [60…   2     <S3: ranger>
```

Let's practice!

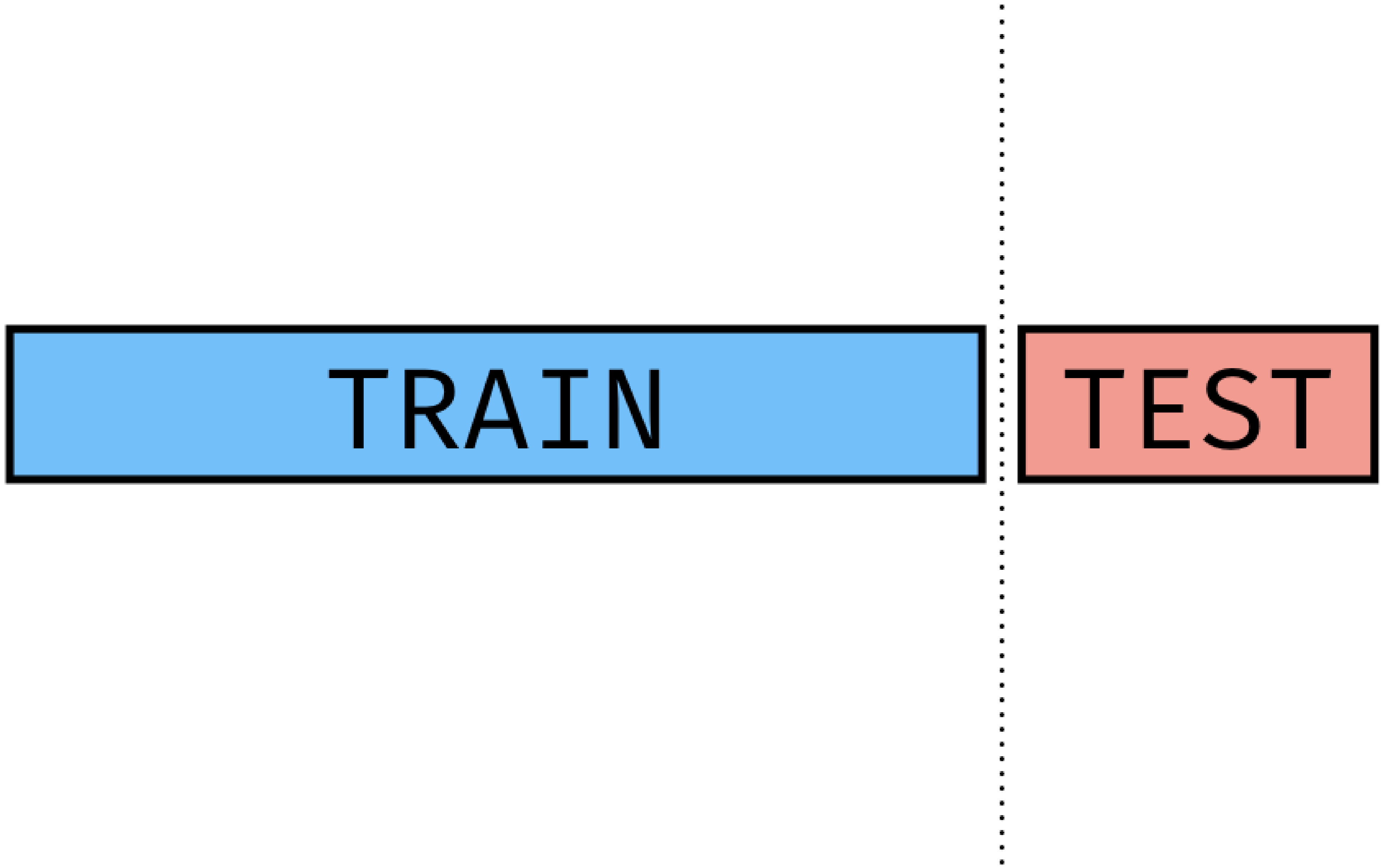
MACHINE LEARNING IN THE TIDYVERSE

Measuring the Test Performance

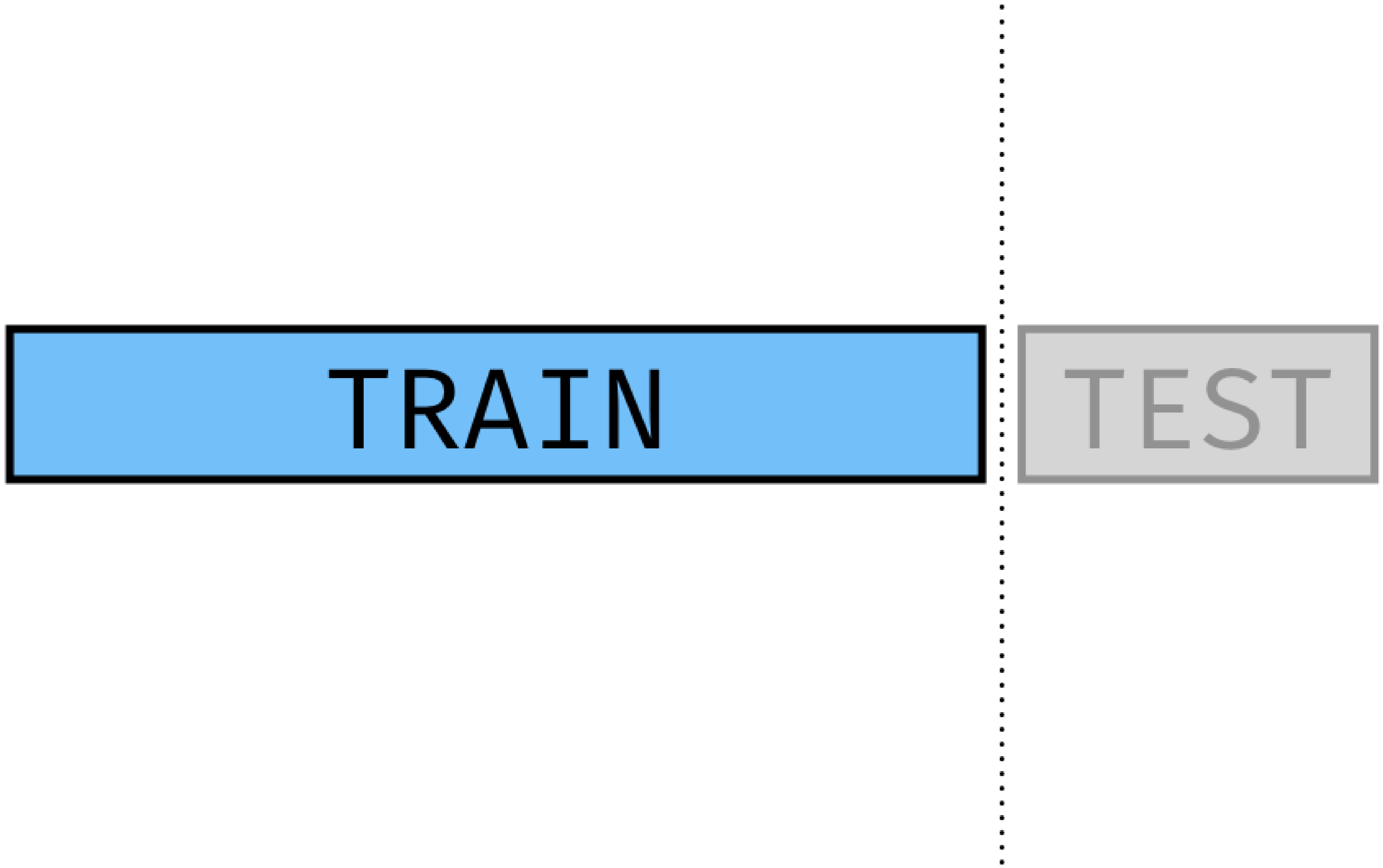
MACHINE LEARNING IN THE TIDYVERSE

Dmitriy (Dima) Gorenshteyn
Lead Data Scientist, Memorial Sloan Kettering Cancer Center

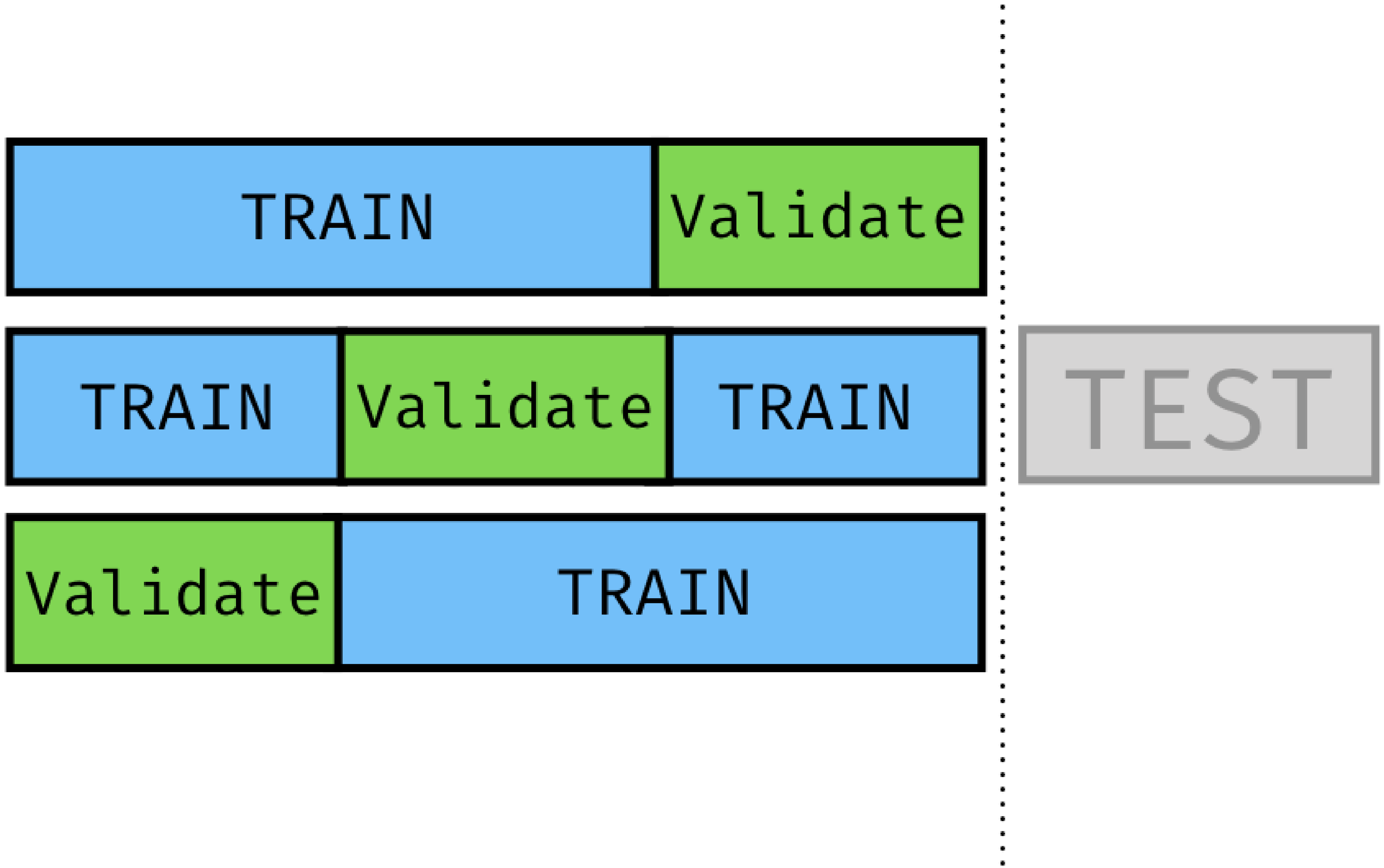
Machine Learning Workflow



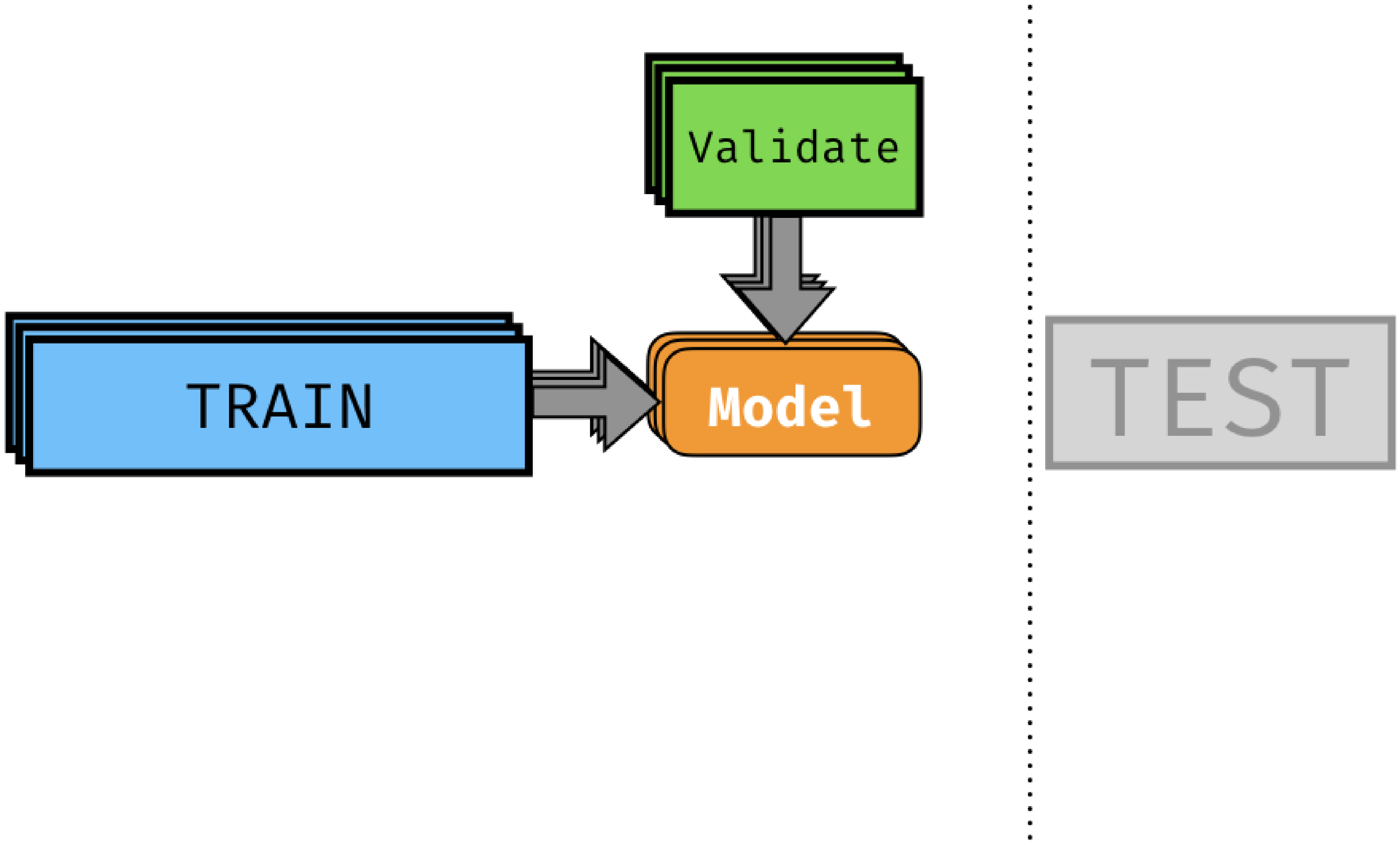
Machine Learning Workflow



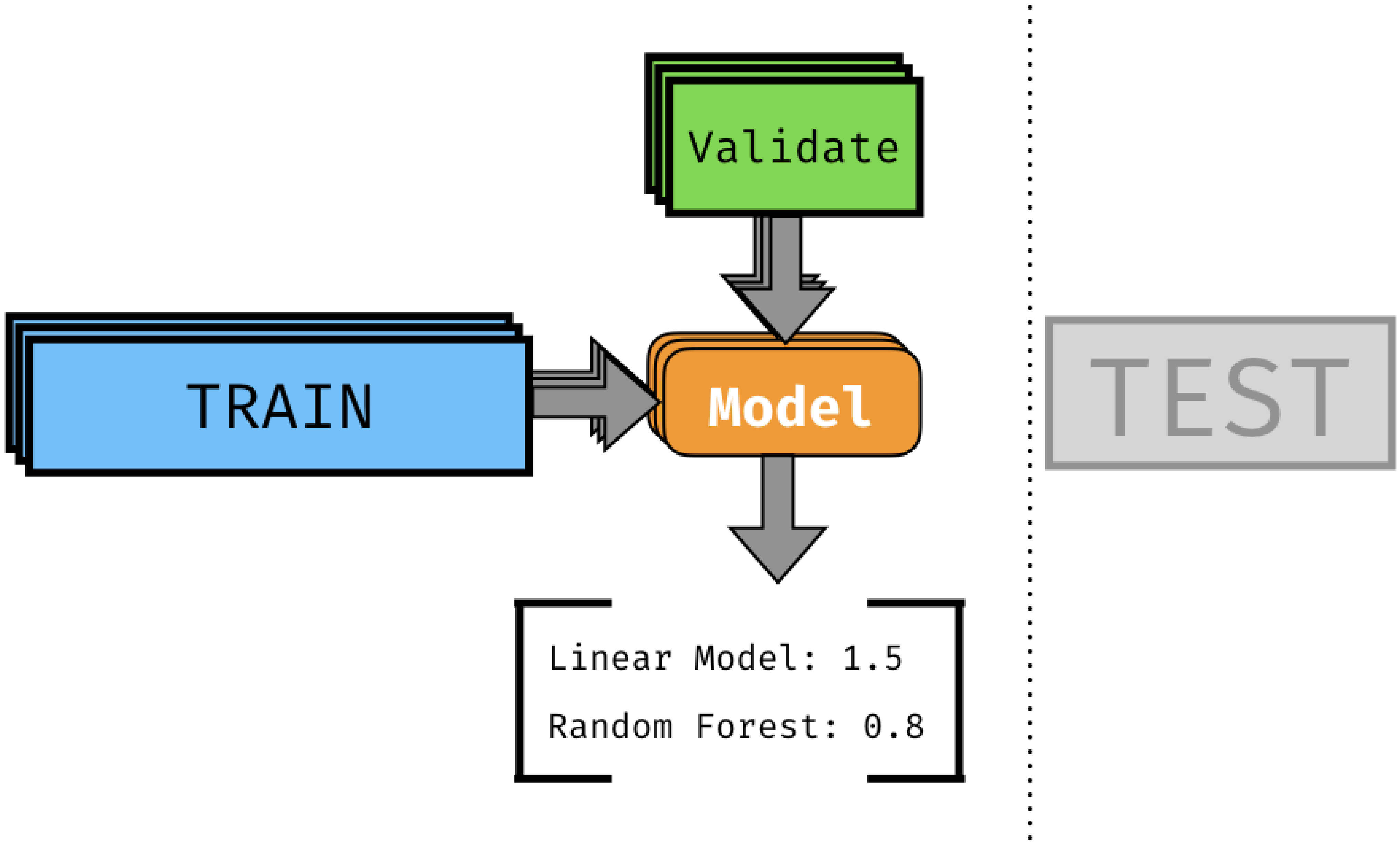
Machine Learning Workflow



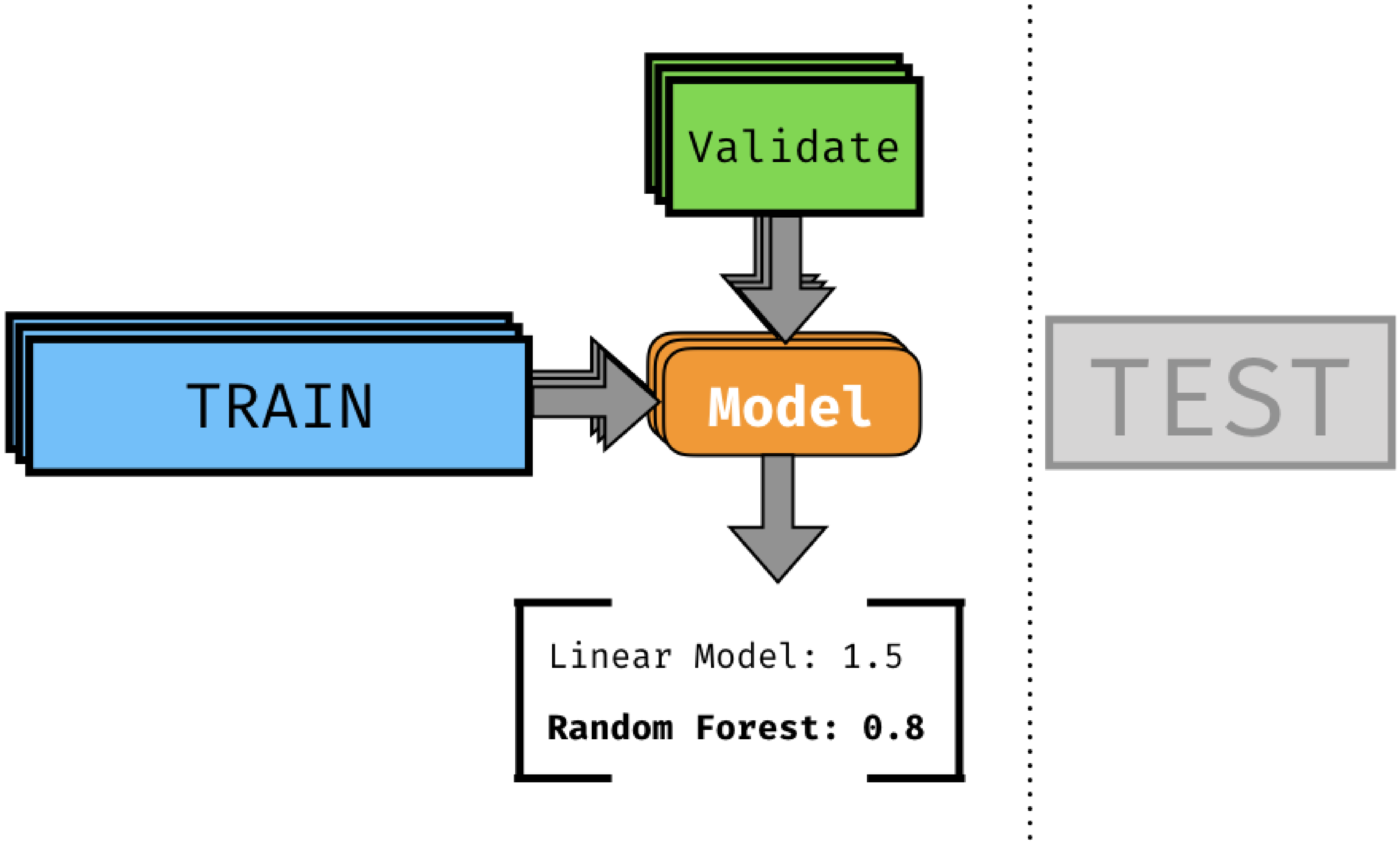
Machine Learning Workflow



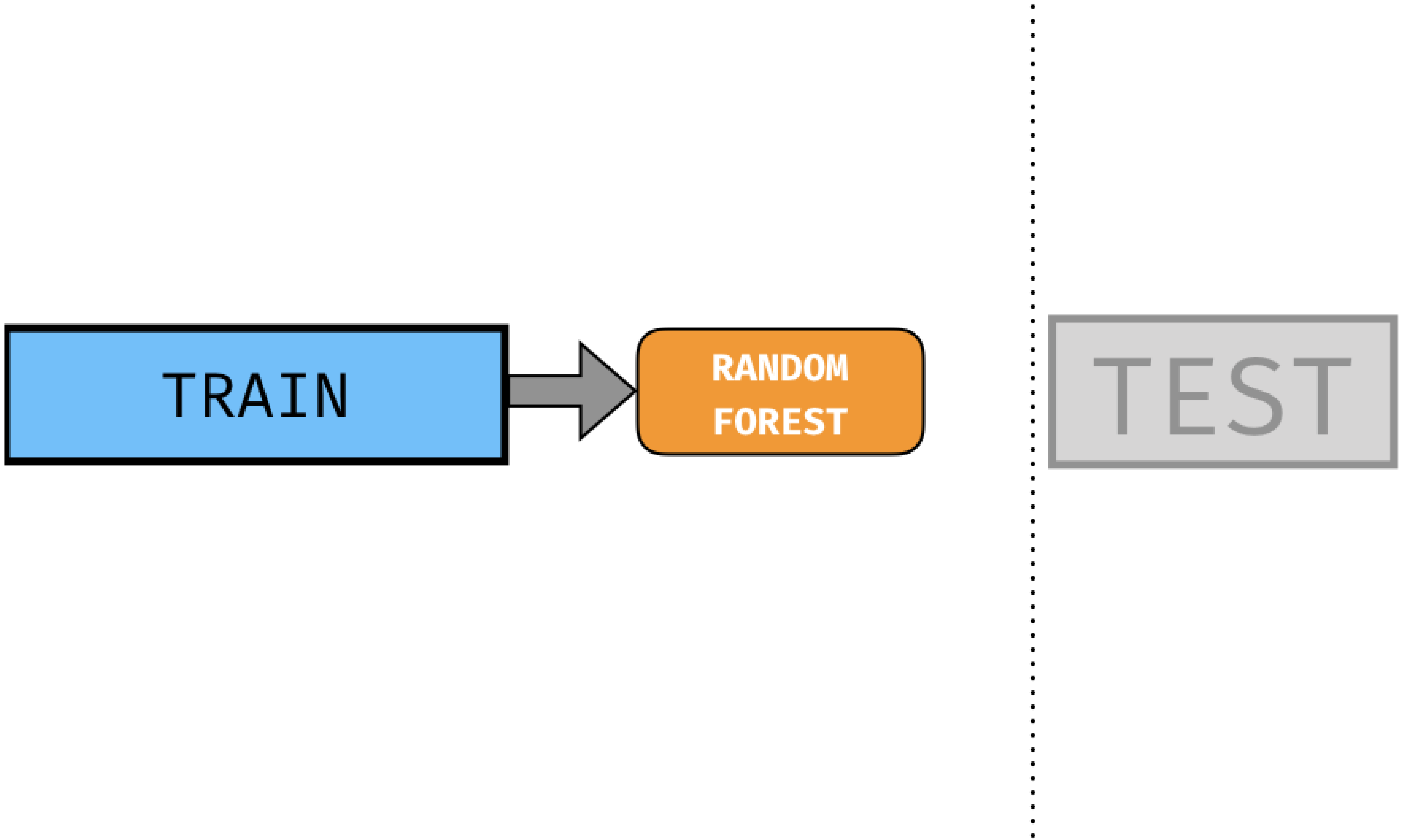
Machine Learning Workflow



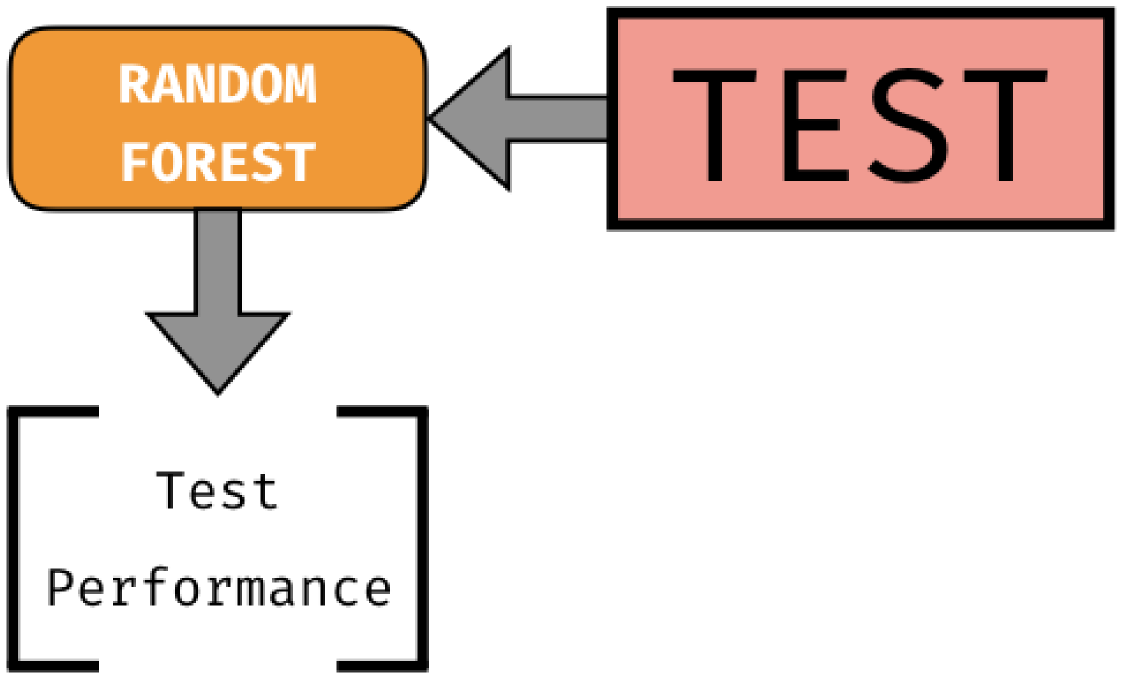
Machine Learning Workflow



Machine Learning Workflow



Machine Learning Workflow



Measuring the Test Performance

```
best_model <- ranger(formula = life_expectancy~.,
                     data = training_data,
                     mtry = 4, num.trees = 100, seed = 42)
```

```
test_actual <- testing_data$life_expectancy
test_predict <- predict(best_model, testing_data)$predictions
```

```
mae(test_actual, test_predict)
```

Let's practice!

MACHINE LEARNING IN THE TIDYVERSE