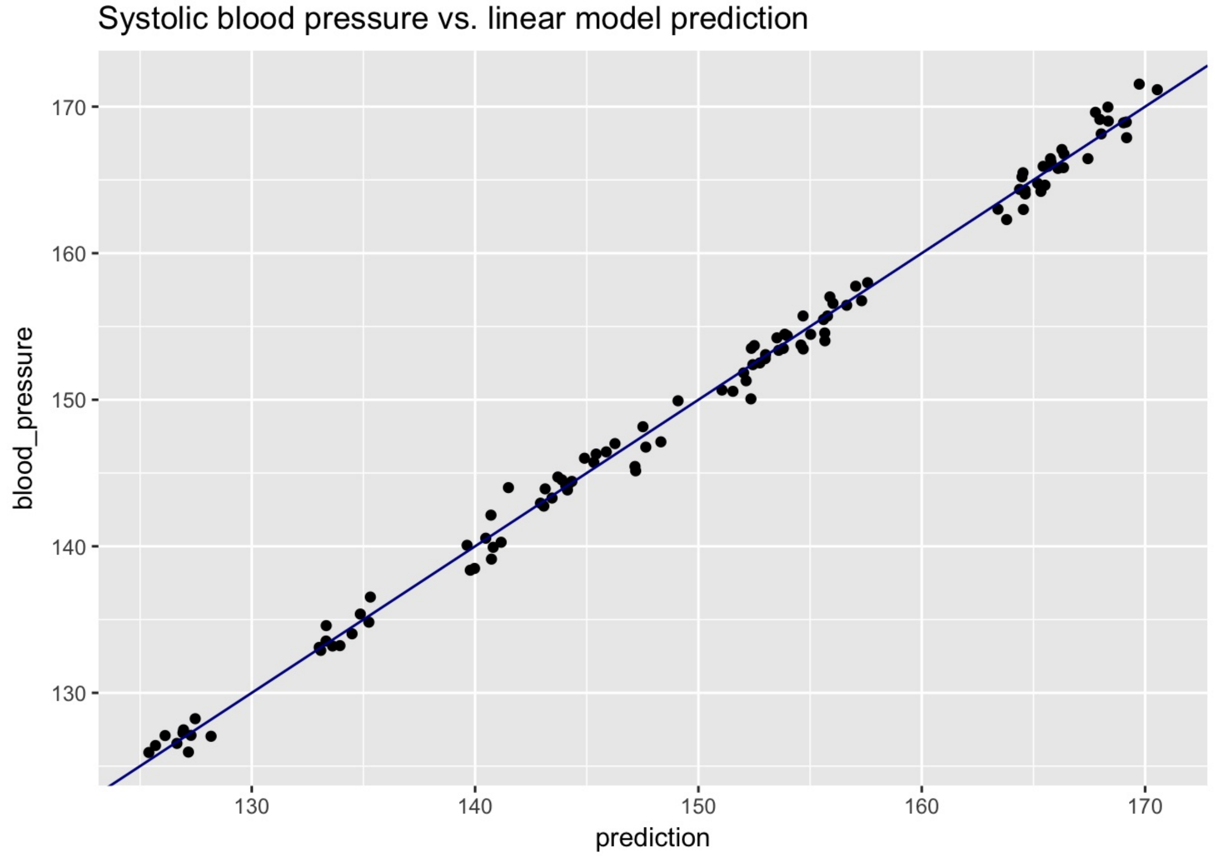
SUPERVISED LEARNING IN R: REGRESSION

Evaluating a model graphically

Nina Zumel and John Mount
Win-Vector LLC

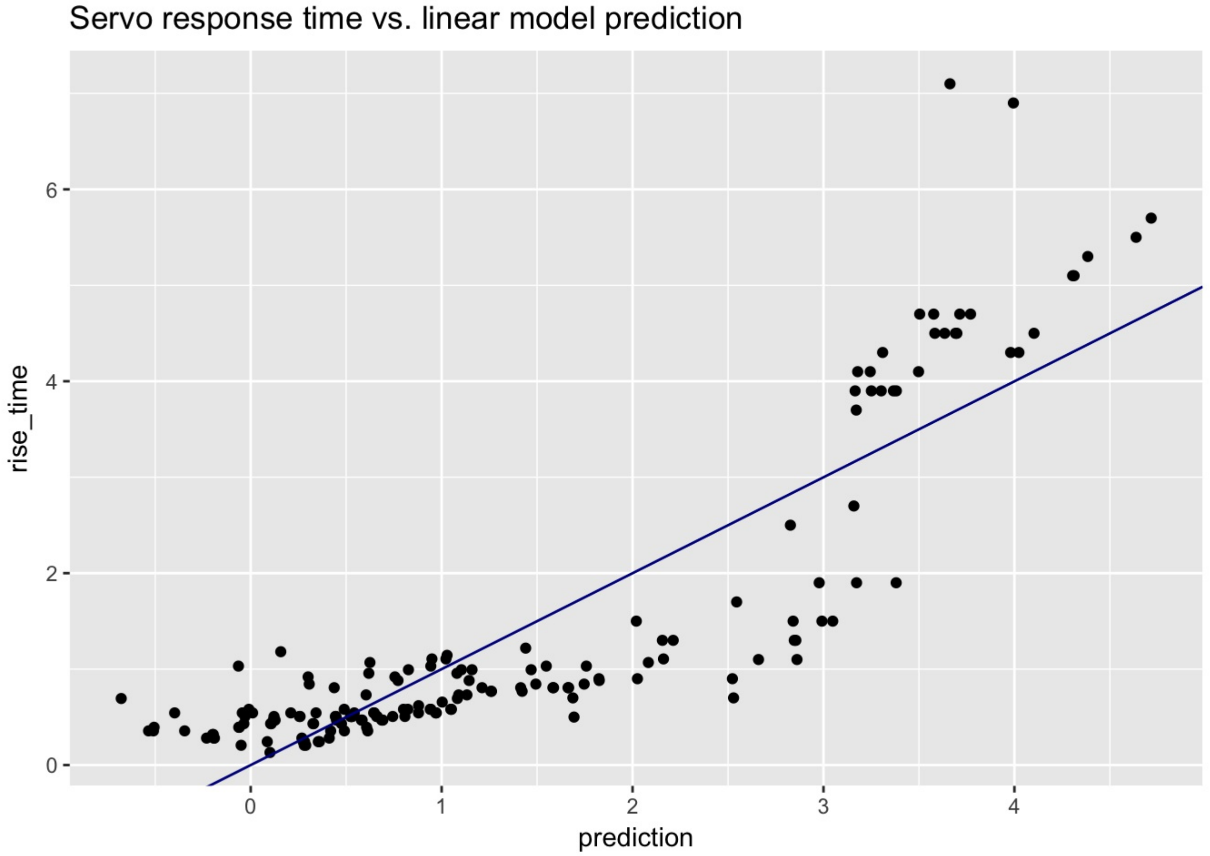
Plotting Ground Truth vs. Predictions

A well fitting model



- $x = y$ line runs through center of points
- "line of perfect prediction"

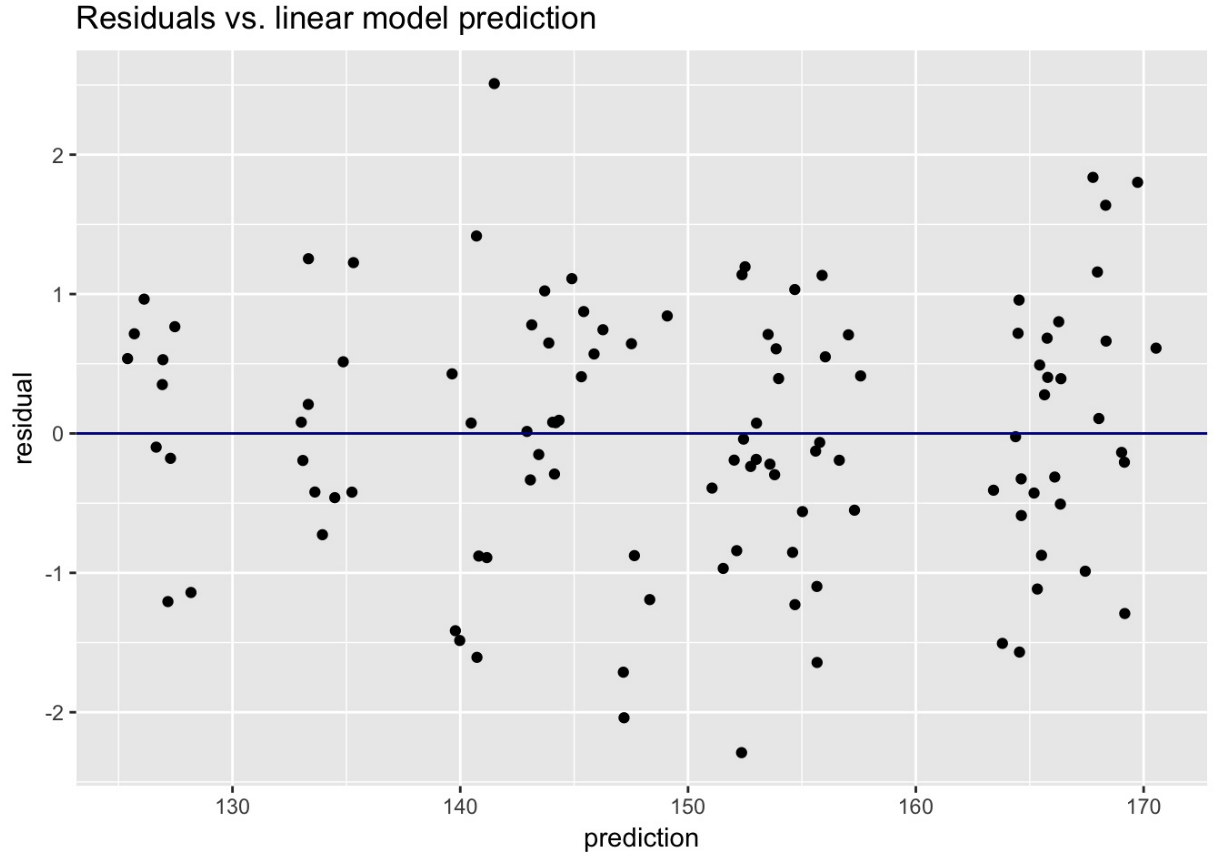
A poorly fitting model



- Points are all on one side of $x = y$ line
- Systematic errors

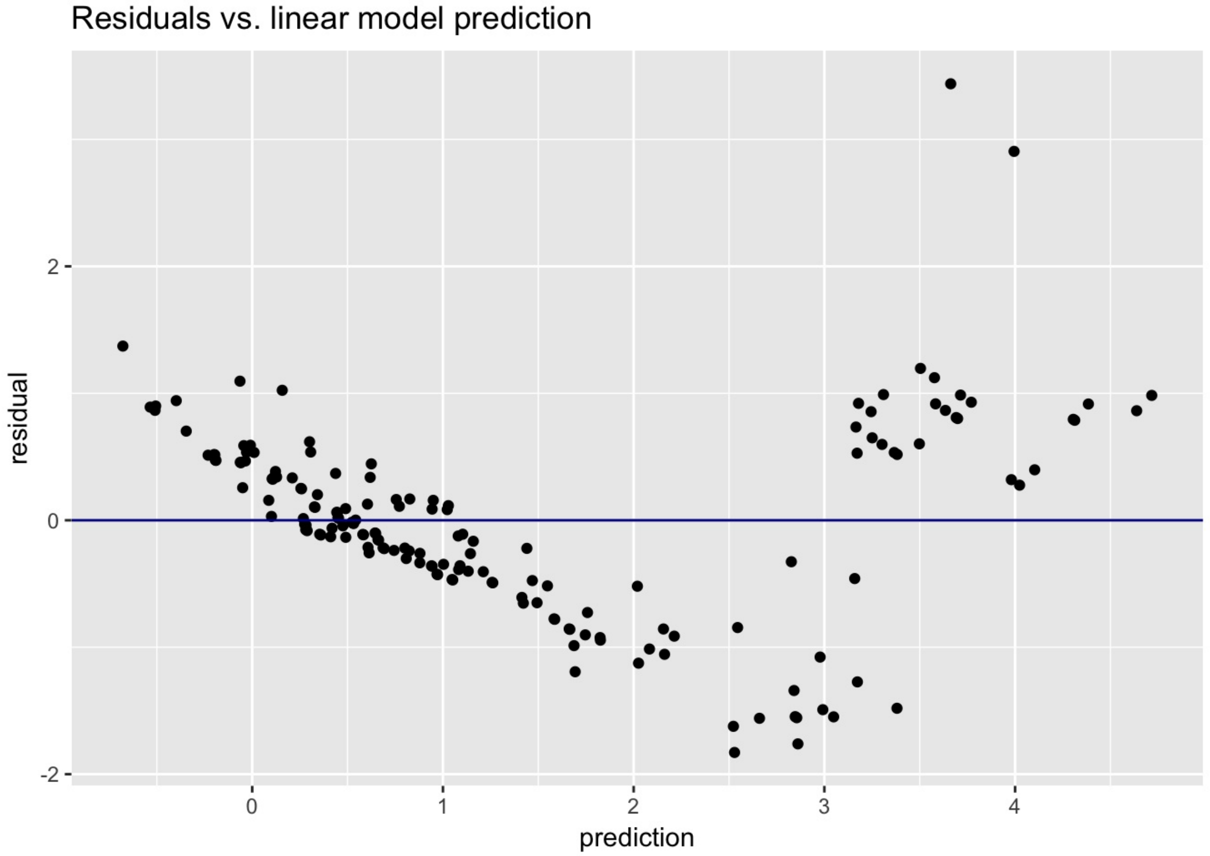
The Residual Plot

A well fitting model



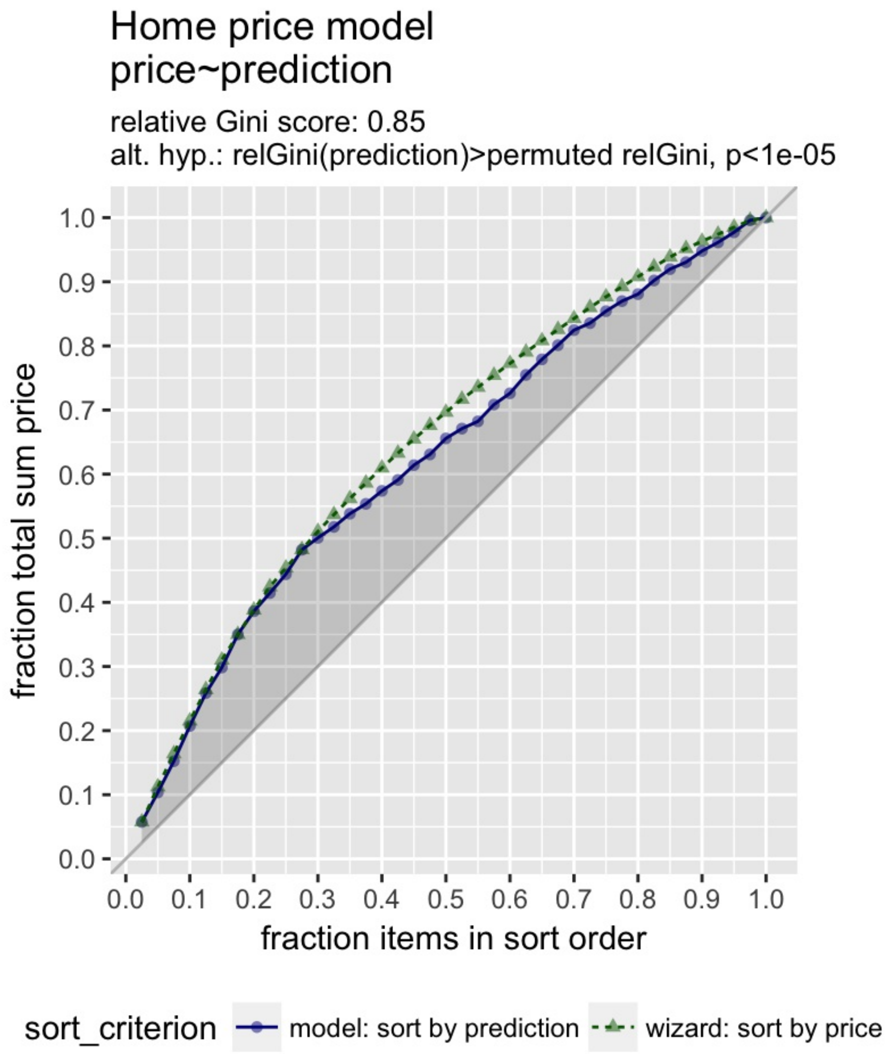
- Residual: actual outcome - prediction
- Good fit: no systematic errors

A poorly fitting model



- Systematic errors

The Gain Curve

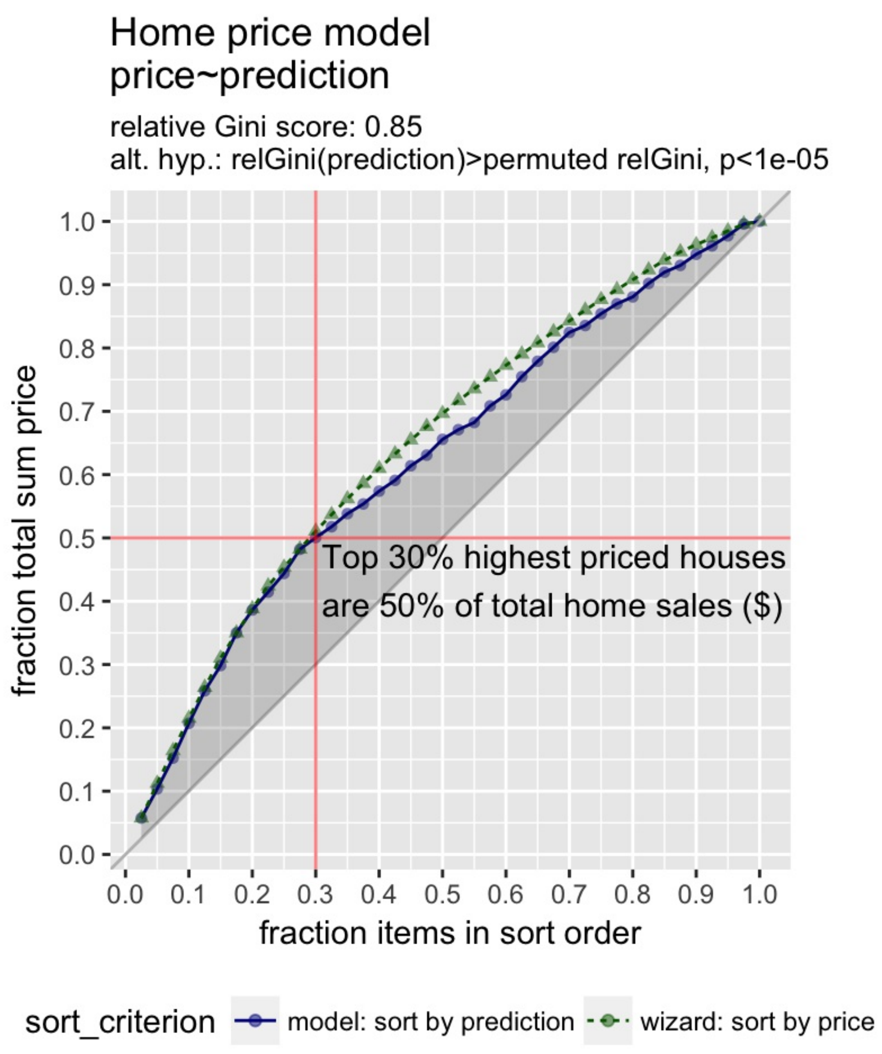


Measures how well model sorts the outcome

- **x-axis**: houses in model-sorted order (decreasing)
- **y-axis**: fraction of total accumulated home sales

Wizard curve: perfect model

Reading the Gain Curve



- GainCurvePlot(houseprices, "prediction", "price", "Home price model")

SUPERVISED LEARNING IN R: REGRESSION

Let's practice!

SUPERVISED LEARNING IN R: REGRESSION

Root Mean Squared Error (RMSE)

Nina Zumel and John Mount
Win-Vector LLC

What is Root Mean Squared Error (RMSE)?

$$RMSE = \sqrt{\overline{(pred - y)^2}}$$

where

- $pred - y$: the error, or residuals vector
- $\overline{(pred - y)^2}$: mean value of $(pred - y)^2$

RMSE of the Home Sales Price Model

```
# Calculate error
> err <- houseprices$prediction - houseprices$price
```

- price: column of actual sale prices (in thousands)
- prediction: column of predicted sale prices (in thousands)

RMSE of the Home Sales Price Model

```
# Calculate error
> err <- houseprices$prediction - houseprices$price

# Square the error vector
> err2 <- err^2
```

RMSE of the Home Sales Price Model

```
# Calculate error
> err <- houseprices$prediction - houseprices$price

# Square the error vector
> err2 <- err^2

# Take the mean, and sqrt it
> (rmse <- sqrt(mean(err2)))
[1] 58.33908
```

- $RMSE \approx 58.3$

Is the RMSE Large or Small?

```
# Take the mean, and sqrt it
> (rmse <- sqrt(mean(err2)))
[1] 58.33908

# The standard deviation of the outcome
> (sdtemp <- sd(houseprices$price))
[1] 135.2694
```

- $RMSE \approx 58.3$
- $sd(price) \approx 135$

SUPERVISED LEARNING IN R: REGRESSION

Let's practice!

SUPERVISED LEARNING IN R: REGRESSION

R-Squared (R^2)

Nina Zumel and John Mount
Win-Vector LLC

What is R^2?

A measure of how well the model fits or explains the data

- A value between 0-1
  - near 1: model fits well
  - near 0: no better than guessing the average value

Calculating R^2

R^2 is the *variance explained by the model.*

$$R^2 = 1 - \frac{RSS}{SS_{Tot}}$$

where

- $RSS = \sum (y - prediction)^2$
  - Residual sum of squares (variance from model)
- $SS_{Tot} = \sum (y - \overline{y})^2$
  - Total sum of squares (variance of data)

Calculate R^2 of the House Price Model: RSS

- Calculate error

```
> err <- houseprices$prediction - houseprices$price
```

- Square it and take the sum

```
> rss <- sum(err^2)
```

- price: column of actual sale prices (in thousands)
- pred: column of predicted sale prices (in thousands)
- $RSS \approx$ 136138

Calculate R^2 of the House Price Model: SS_{Tot}

- Take the difference of prices from the mean price

```
> toterr <- houseprices$price - mean(houseprices$price)
```

- Square it and take the sum

```
> sstot <- sum(toterr^2)
```

- $RSS \approx$ 136138
- $SS_{Tot} \approx$ 713615

Calculate R^2 of the House Price Model

```
> (r_squared <- 1 - (rss/sstot) )
[1] 0.8092278
```

- $RSS \approx 136138$
- $SS_{Tot} \approx 713615$
- $R^2 \approx 0.809$

Reading R^2 from the Model

For lm() models:

- From summary():

```
> summary(hmodel)
## ...
## Residual standard error: 60.66 on 37 degrees of freedom
## Multiple R-squared:  0.8092, Adjusted R-squared:  0.7989
## F-statistic: 78.47 on 2 and 37 DF,  p-value: 4.893e-14

> summary(hmodel)$r.squared
[1] 0.8092278
```

- From glance():

```
> glance(hmodel)$r.squared
[1] 0.8092278
```

Correlation and R^2

```
> rho <- cor(houseprices$prediction, houseprices$price)
[1] 0.8995709

> rho^2
[1] 0.8092278
```

- ρ = cor(prediction, price) = 0.8995709
- ρ^2 = 0.8092278 = R^2
- True for models that minimize squared error:
    - Linear regression
    - GAM regression
    - Tree-based algorithms that minimize squared error
- True for training data; **NOT** true for future application data

SUPERVISED LEARNING IN R: REGRESSION

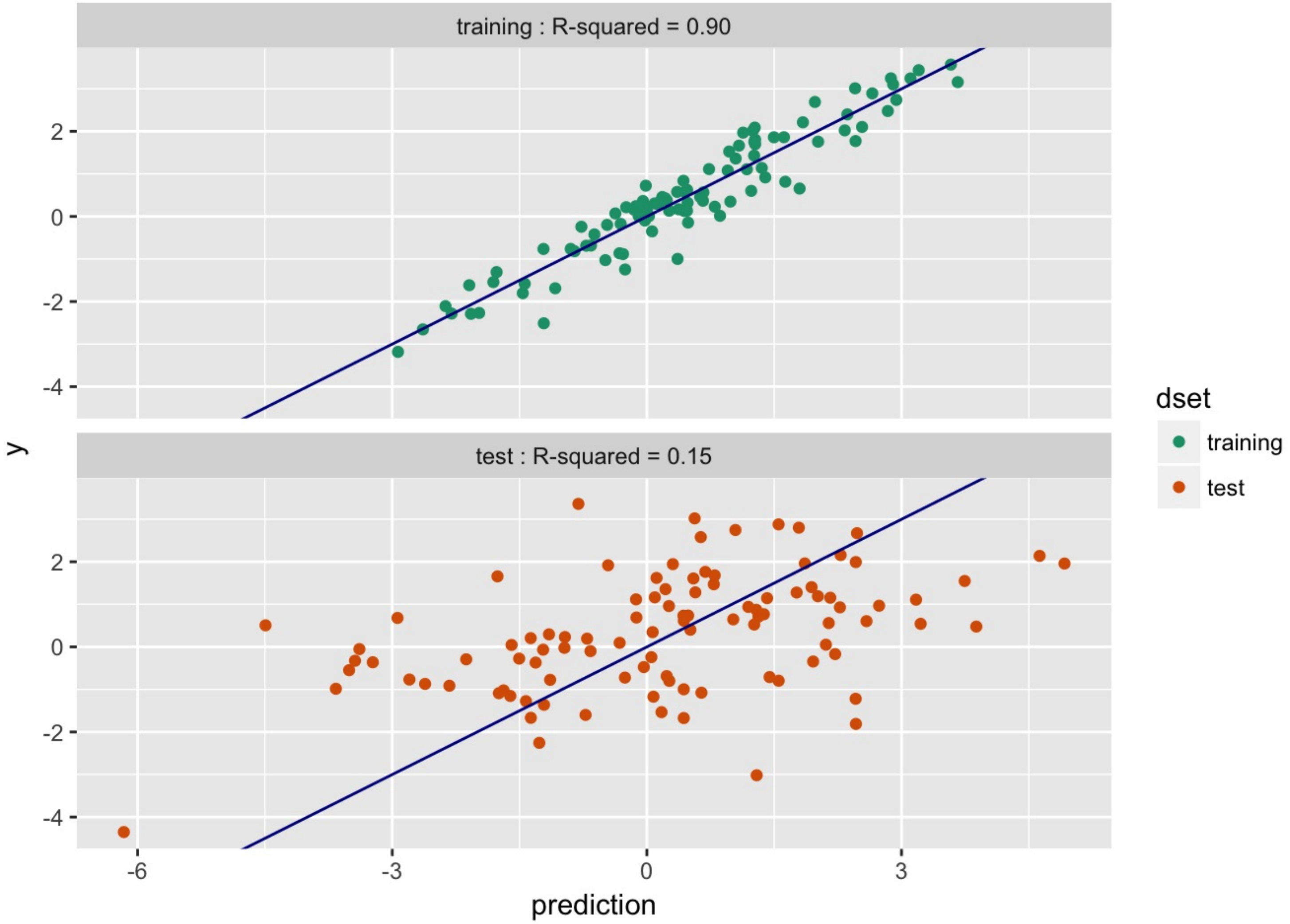
Let's practice!

SUPERVISED LEARNING IN R: REGRESSION

Properly Training a Model

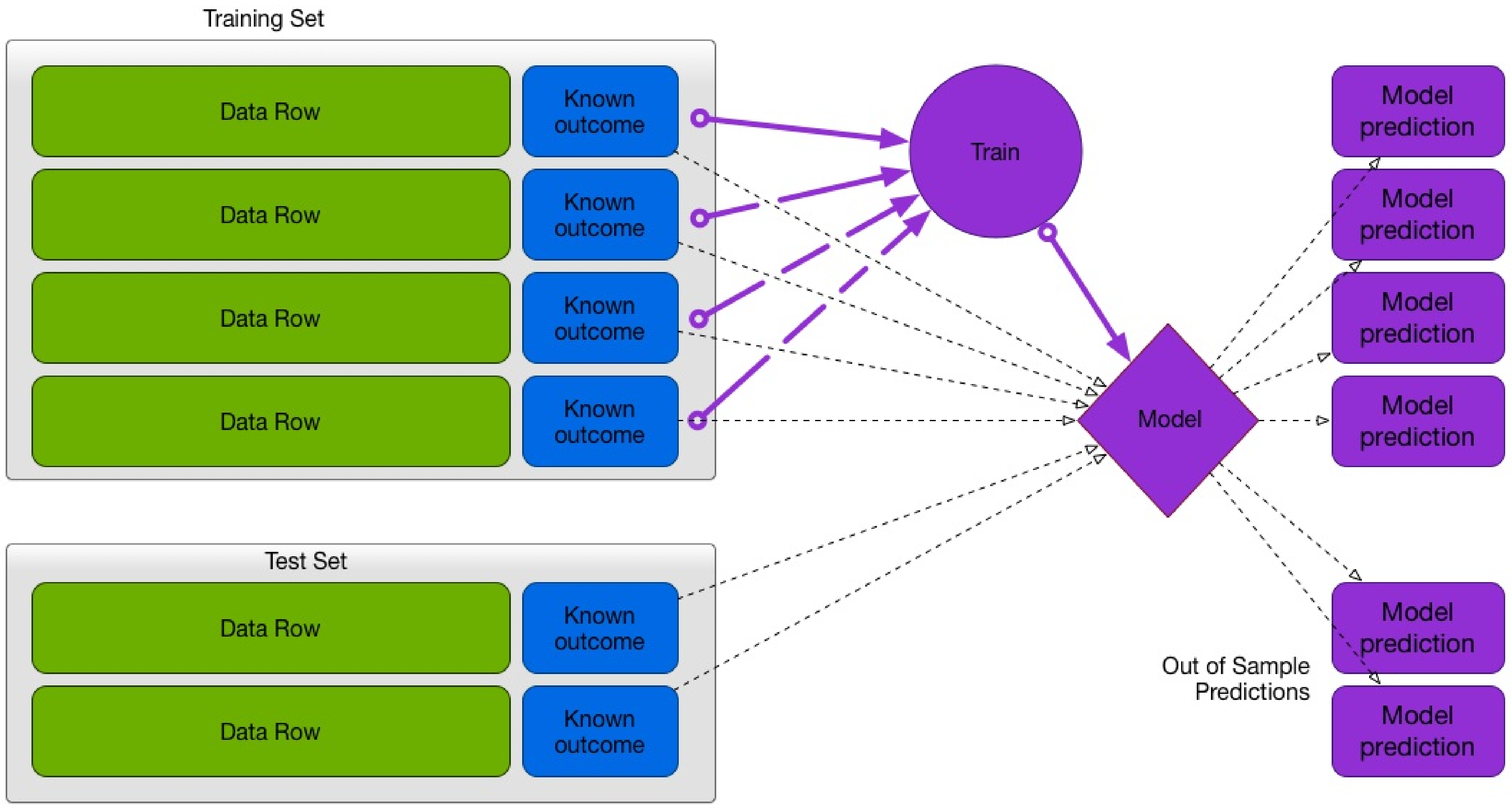
Nina Zumel and John Mount
Win-Vector, LLC

Models can perform much better on training than they do on future data.



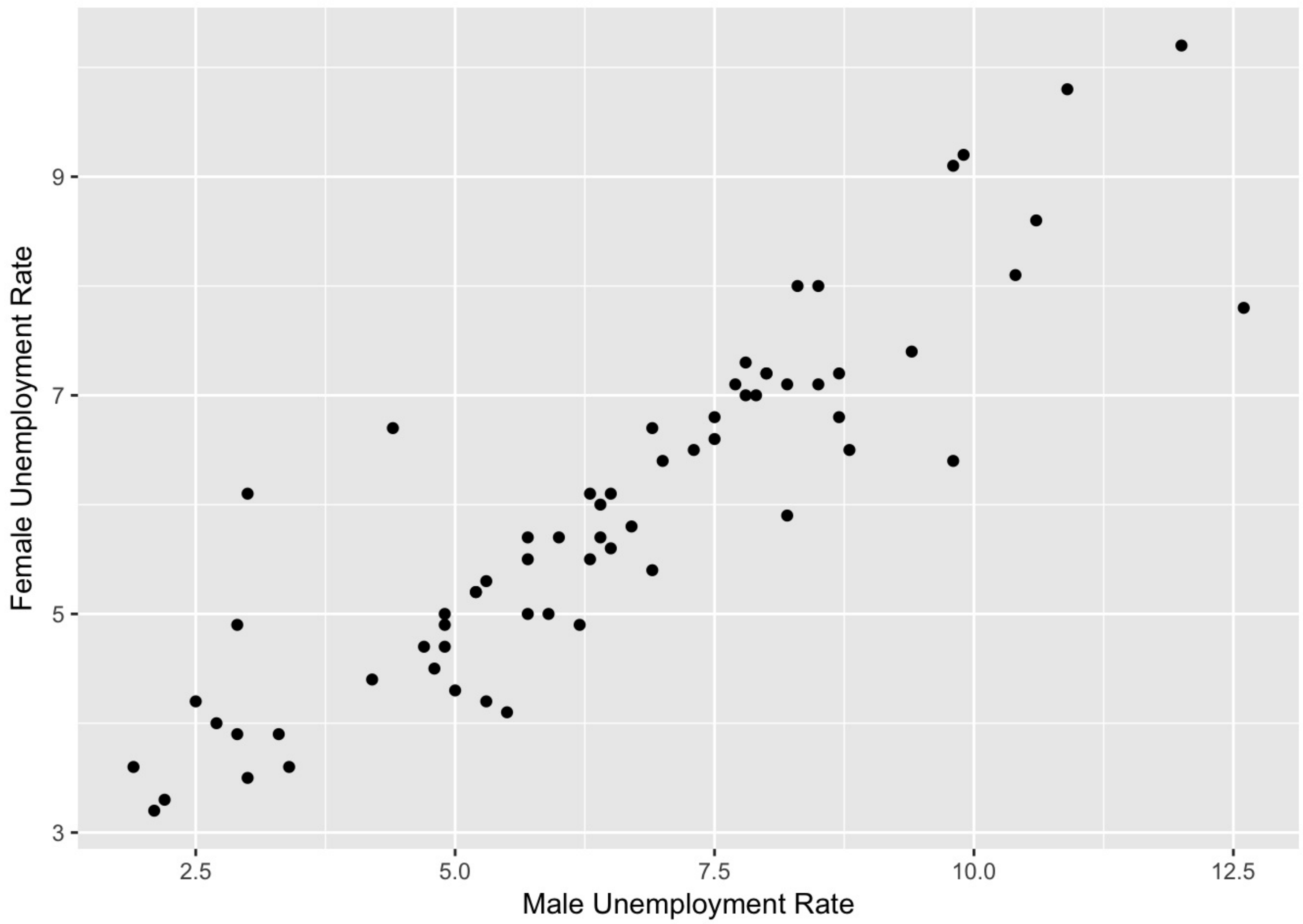
- Training R^2: 0.9; Test R^2: 0.15 -- **Overfit**

Test/Train Split



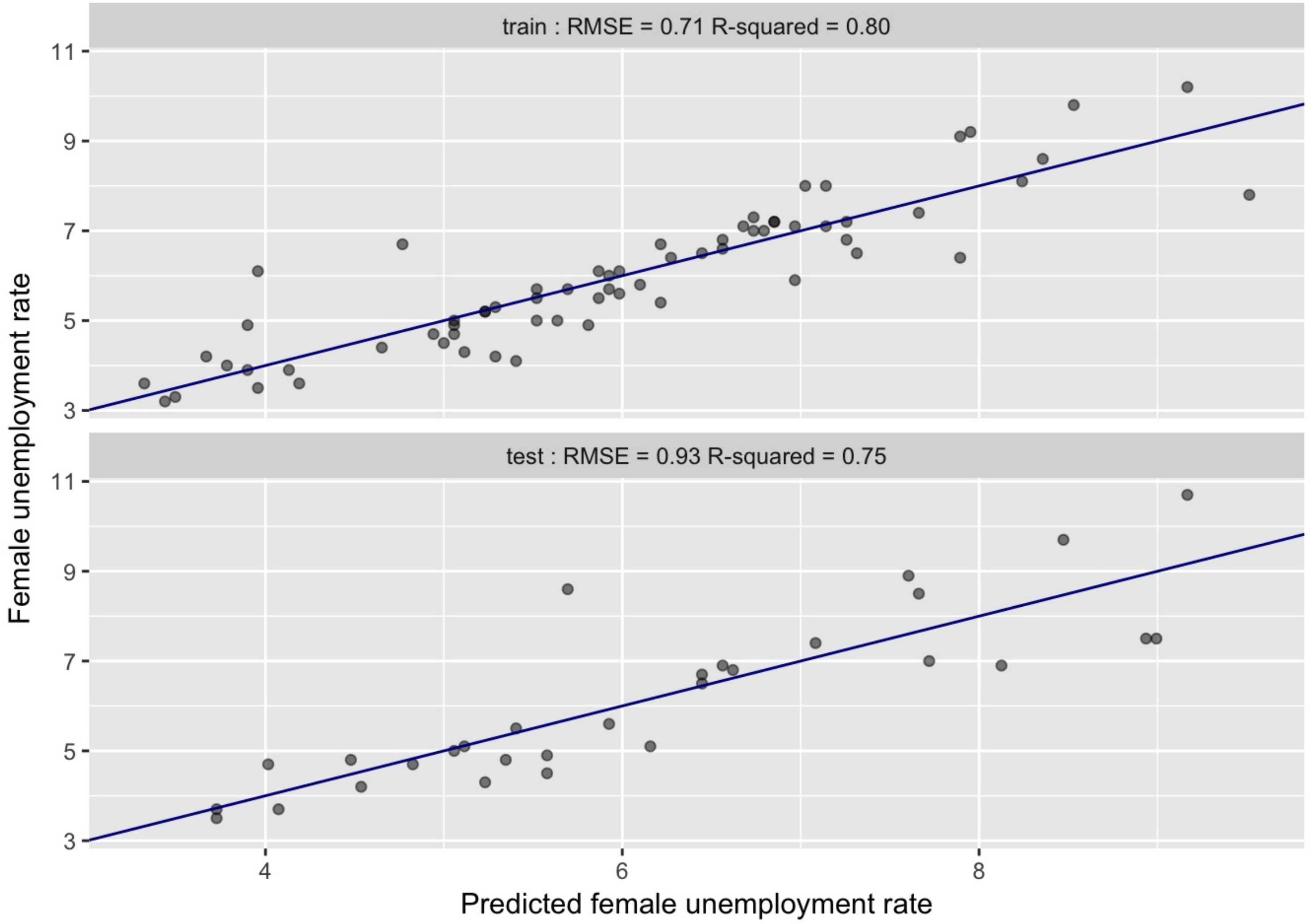
Recommended method when data is plentiful

Example: Model Female Unemployment



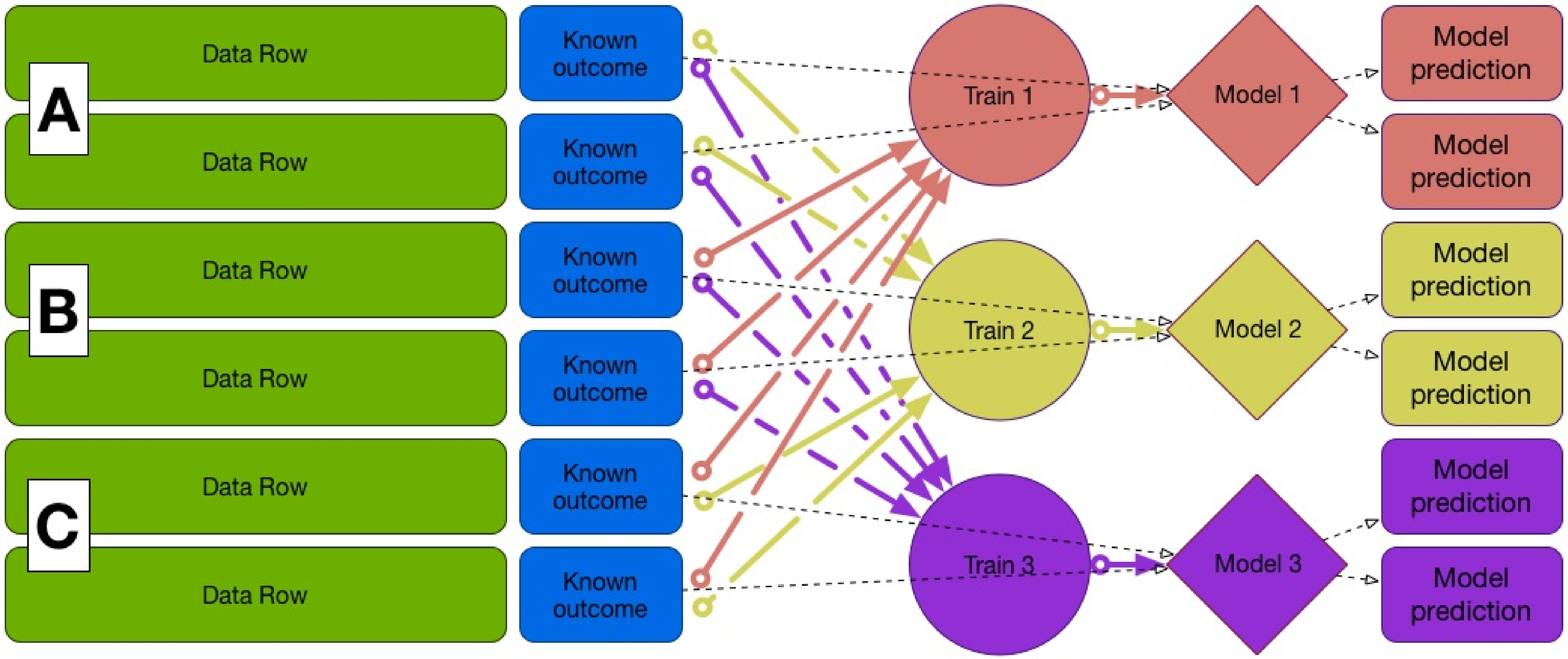
- Train on 66 rows, test on 30 rows

Model Performance: Train vs. Test



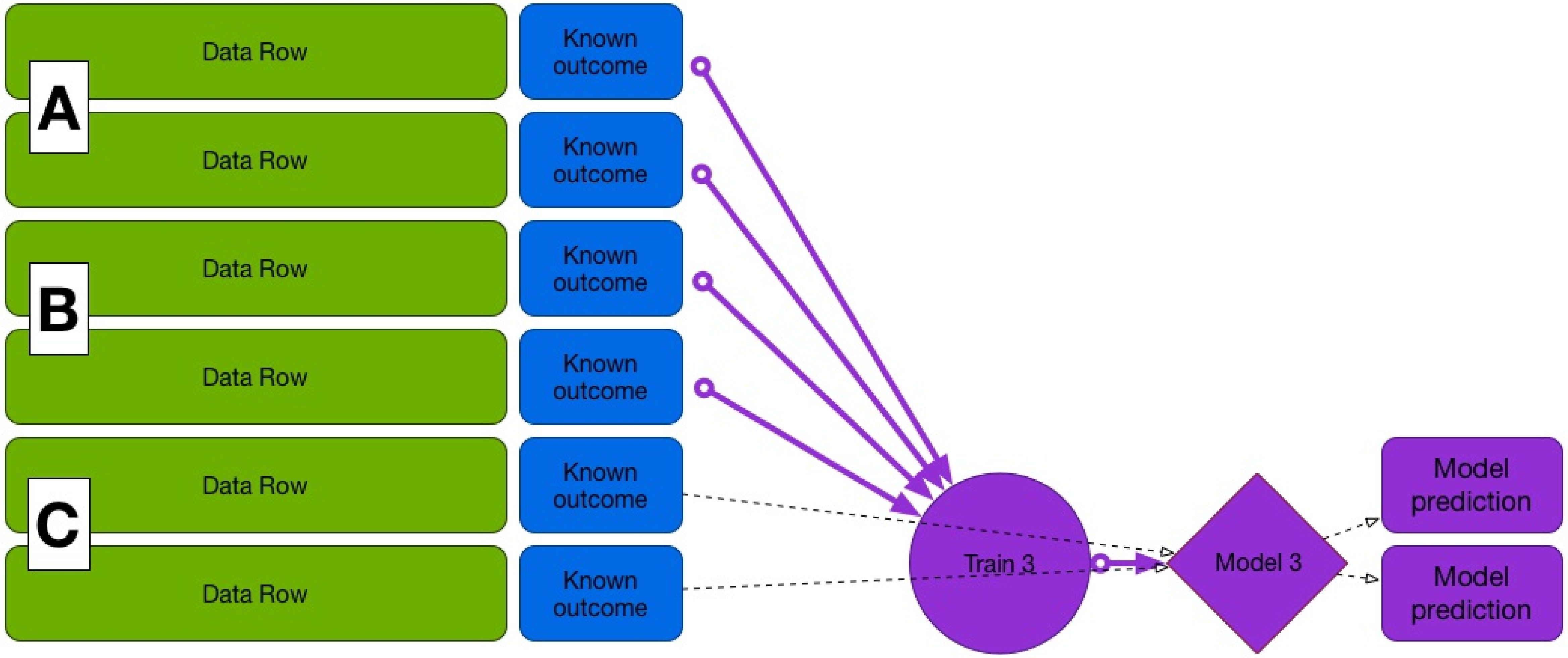
- Training: RMSE 0.71, R^2 0.8
- Test: RMSE 0.93, R^2 0.75

Cross-Validation

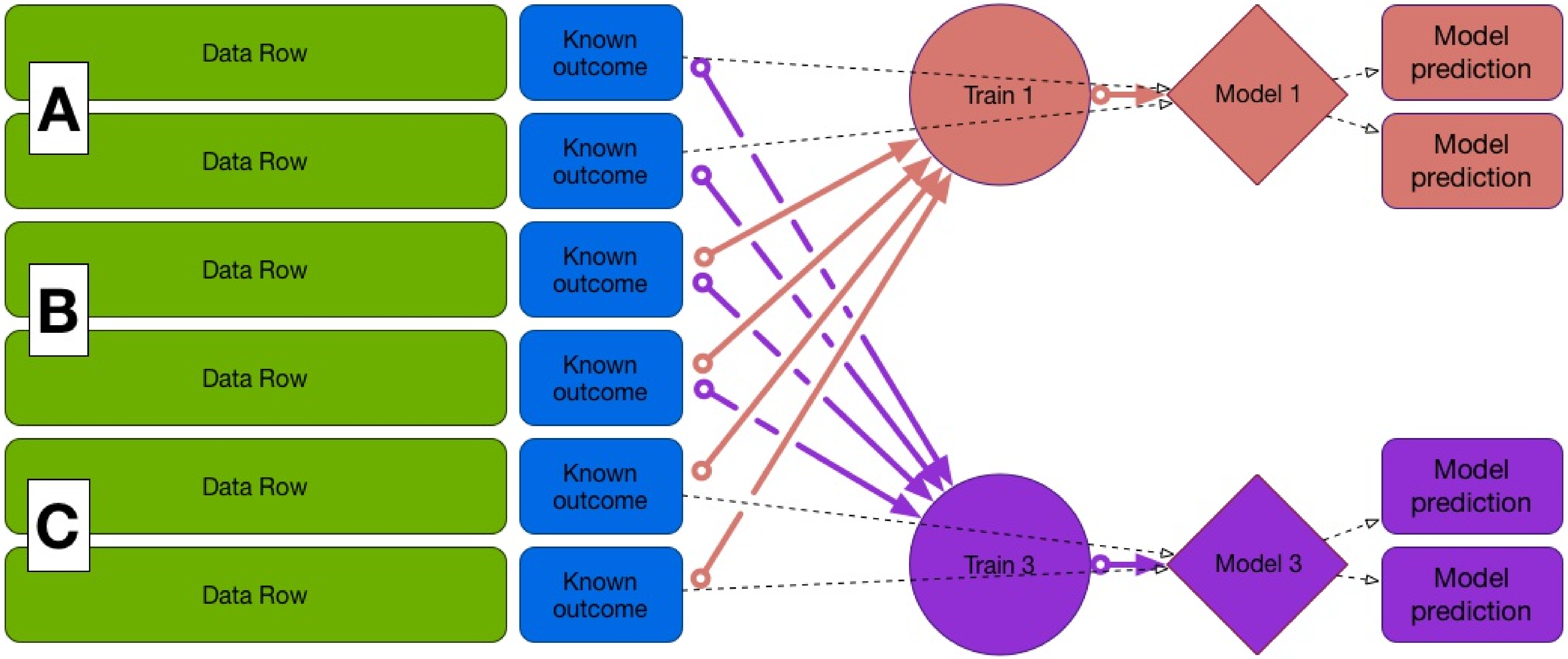


Preferred when data is not large enough to split off a test set

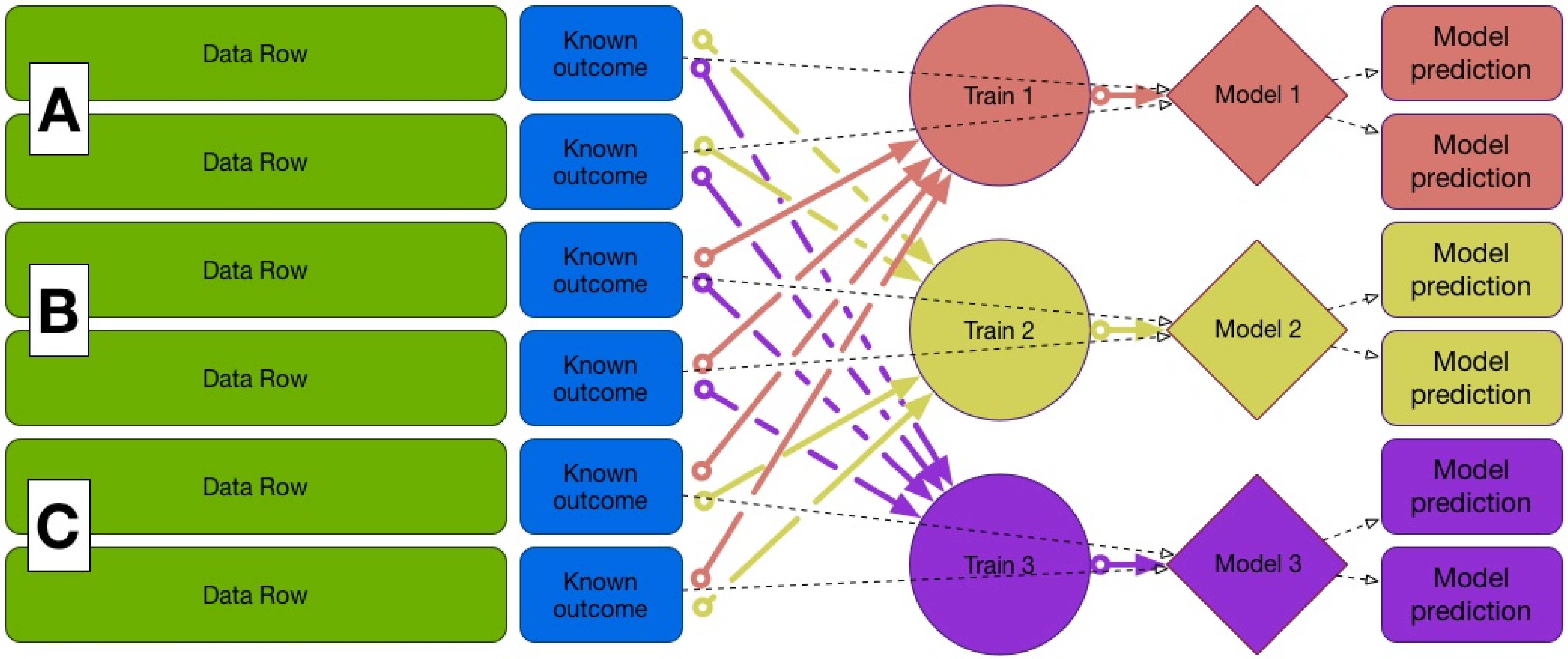
Cross-Validation



Cross-Validation



Cross-Validation



Create a cross-validation plan

```
> library(vtreat)
> splitPlan <- kWayCrossValidation(nRows, nSplits, NULL, NULL)
```

- nRows: number of rows in the training data
- nSplits: number folds (partitions) in the cross-validation
    - e.g, nfolds = 3 for 3-way cross-validation
- remaining 2 arguments not needed here

Create a cross-validation plan

```
> library(vtreat)
> splitPlan <- kWayCrossValidation(10, 3, NULL, NULL)
```

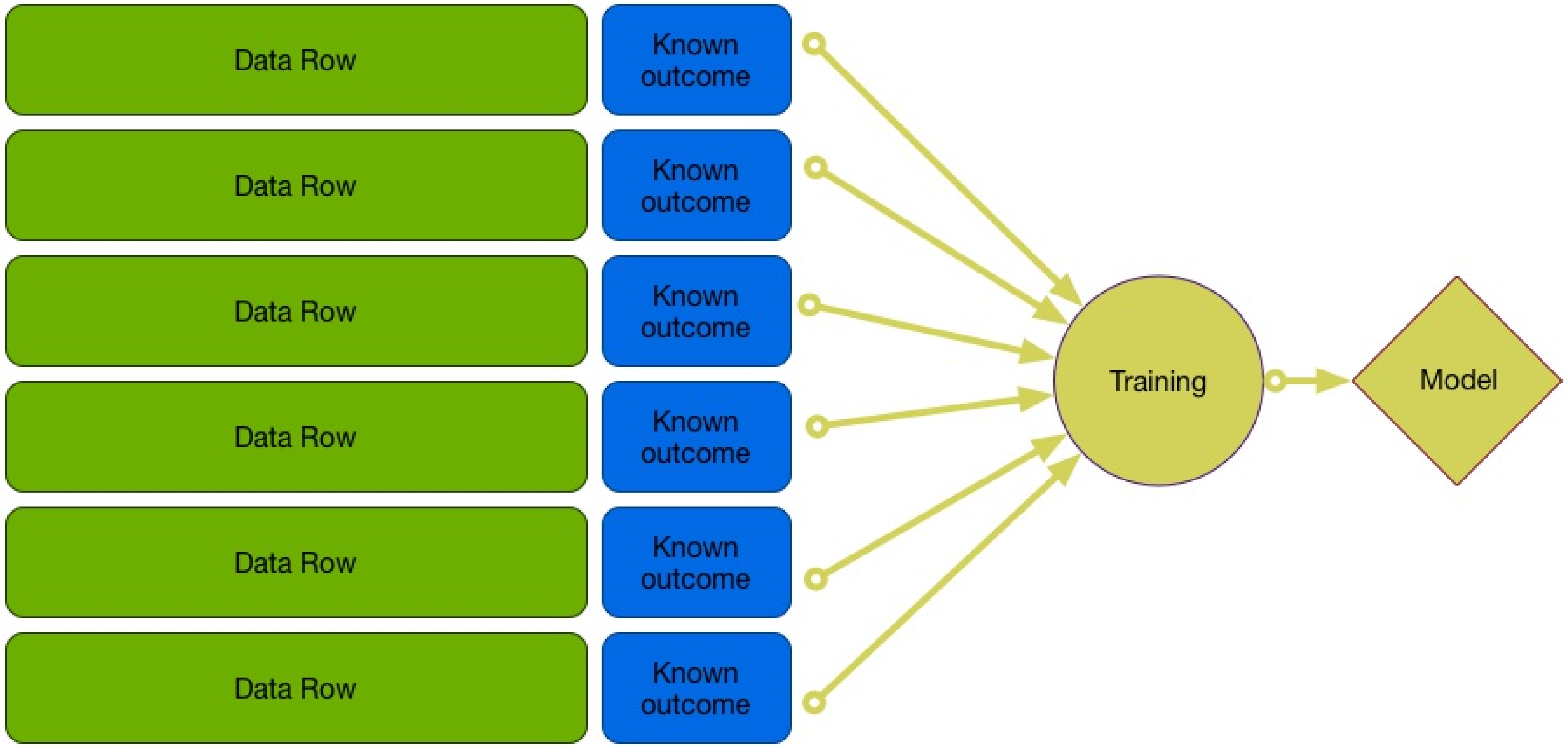
First fold (A and B to train, C to test)

```
> splitPlan[[1]]
## $train
## [1]  1  2  4  5  7  9 10
##
## $app
## [1] 3 6 8
```

Train on A and B, test on C, etc...

```
> split <- splitPlan[[1]]
> model <- lm(fmla, data = df[split$train,])
> df$pred.cv[split$app] <- predict(model, newdata = df[split$app,])
```

Final Model


Data Row
Known outcome
Data Row
Known outcome
Data Row
Known outcome
Data Row
Known outcome
Data Row
Known outcome
Data Row
Known outcome
Training
Model

Example: Unemployment Model

Measure type	RMSE	R^2
train	0.7082675	0.8029275
test	0.9349416	0.7451896
cross-validation	0.8175714	0.7635331

SUPERVISED LEARNING IN R: REGRESSION

Let's practice!