# Modeling Concepts: Inference vs Prediction

Justin Post

# What do we want to be able to do?

Data Science!

- Read in raw data and manipulate it
- Combine data sources

tidyverse

- Summarize data to glean insights — EDA
- Apply common analysis methods — where we are
- Communicate Effectively

qmd, dashboards

# Modeling Ideas

What is a (statistical) model?

- A mathematical representation of some phenomenon on which you've observed data
- Form of the model can vary greatly!

# Modeling (log) Selling Price

target or response

- First a visual on motorcycle sales data

- numeric variable ⇒ regression task

```
bike_data <- read_csv("https://www4.stat.ncsu.edu/~online/datasets/bikeDetails.csv")
bike_data <- bike_data |>
  mutate(log_selling_price = log(selling_price),
         log_km_driven = log(km_driven)) |>
  select(log_km_driven, log_selling_price, everything())
bike_data
```

```
## # A tibble: 1,061 × 9
##   log_km_driven log_selling_price name     selling_price  year seller_type owner
##           <dbl>             <dbl> <chr>            <dbl> <dbl> <chr>       <chr>
## 1          5.86              12.1 Royal E…        175000  2019 Individual  1st …
## 2          8.64              10.7 Honda D…         45000  2017 Individual  1st …
## 3          9.39              11.9 Royal E…        150000  2018 Individual  1st …
## 4         10.0               11.1 Yamaha …         65000  2015 Individual  1st …
## 5          9.95               9.90 Yamaha …         20000  2011 Individual  2nd …
## # ℹ 1,056 more rows
## # ℹ 2 more variables: km_driven <dbl>, ex_showroom_price <dbl>
```

X explanatory variable or predictor

Y response

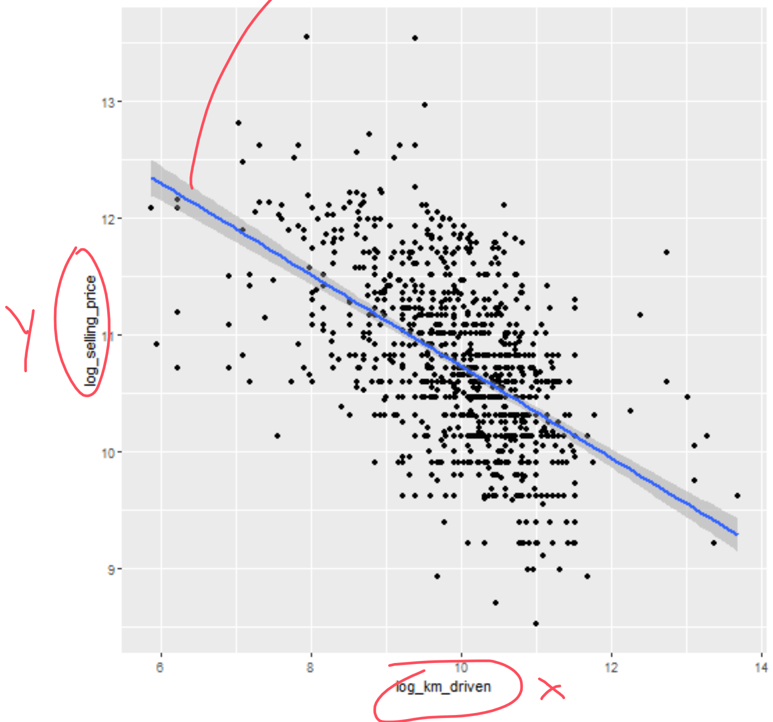
# Modeling (log) Selling Price

- First a visual on motorcycle sales data

```
ggplot(bike_data, aes(x = log_km_driven, y = log_selling_price)) +
  geom_point() +
  geom_smooth(method = "lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

SLR line

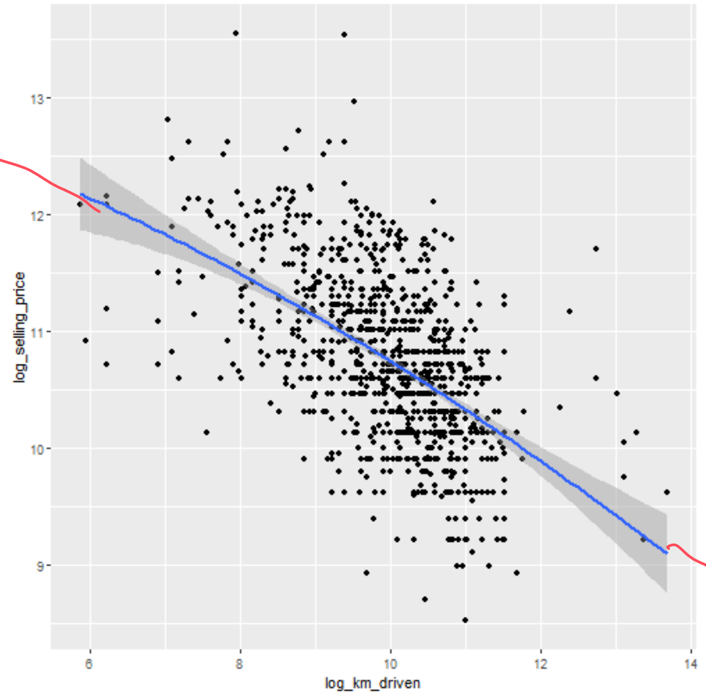


# Modeling (log) Selling Price

- First a visual on motorcycle sales data

```
ggplot(bike_data, aes(x = log_km_driven, y = log_selling_price)) +
  geom_point() +
  stat_smooth(method = "lm", formula = y ~ x + I(x^2))
```

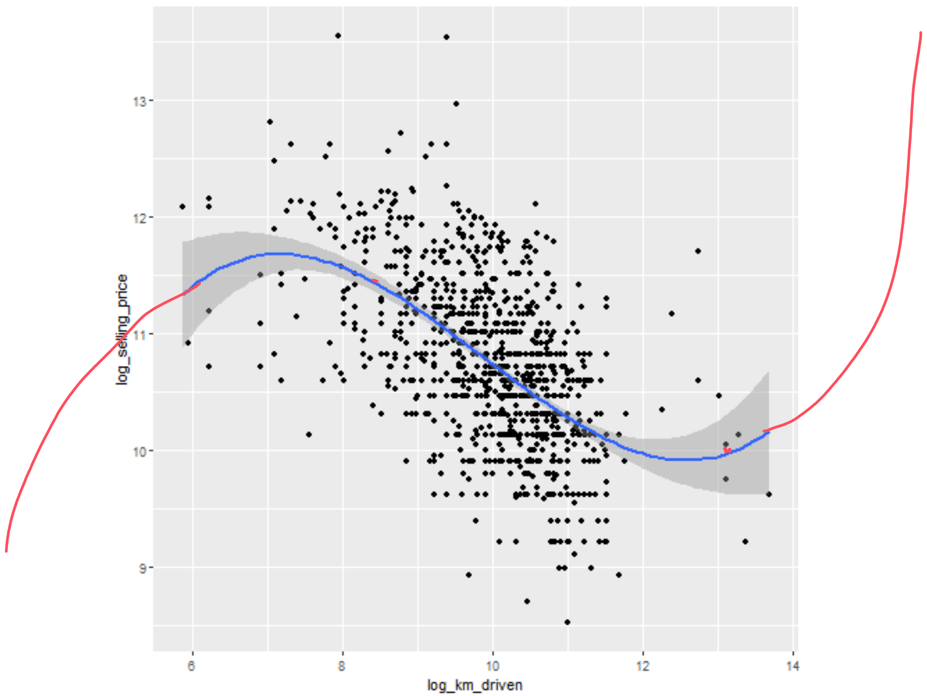
Formula for $Y = \beta_0 + \beta_1 x + \beta_2 x^2 + E$



# Modeling (log) Selling Price

- First a visual on motorcycle sales data

```
ggplot(bike_data, aes(x = log_km_driven, y = log_selling_price)) +
  geom_point() +
  stat_smooth(method = "lm", formula = y ~ x + I(x^2) + I(x^3))
```

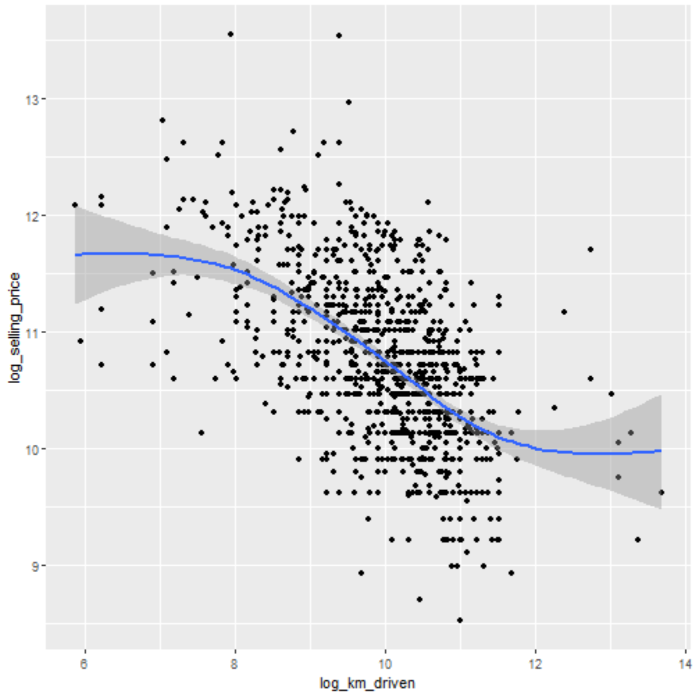


# Modeling (log) Selling Price

- First a visual on motorcycle sales data

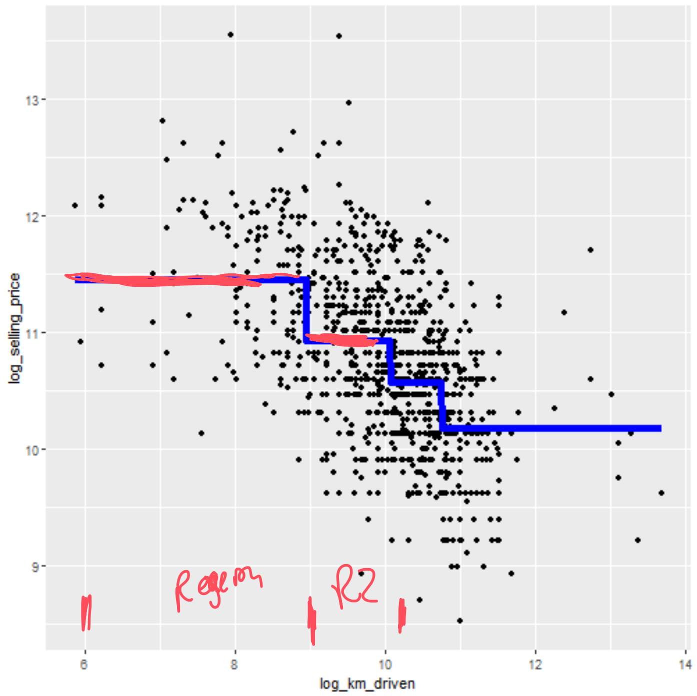
```
ggplot(bike_data, aes(x = log_km_driven, y = log_selling_price)) +
  geom_point() +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



# Modeling (log) Selling Price

```
ggplot(bike_data, aes(x = log_km_driven, y = log_selling_price)) +
  geom_point() +
  geom_line(data = preds, aes(x = log_km_driven, y = Tree_Predictions), color = "Blue", linewidth = 2)
```




Regression Tree

'non-linear model'

# Modeling Ideas

What is a (statistical) model?

- A mathematical representation of some phenomenon on which you've observed data
- Form of the model can vary greatly!

**Statistical learning** - Inference, prediction/classification, and pattern finding

- Supervised learning - a variable (or variables) represents an **output** or **response** of interest
    - May model response and
        - Make **inference** on the model parameters
        - **predict** a value or **classify** an observation

Our Goal: Understand what it means to be a good predictive model (not make inference)

# Training a Model

SLR models
or MLR models

- Once a class of models is chosen, we must define some criteria to **fit** (or train) the model

## Simple Linear Regression (SLR) Model

'given'

a linear function of x

estimate parameters

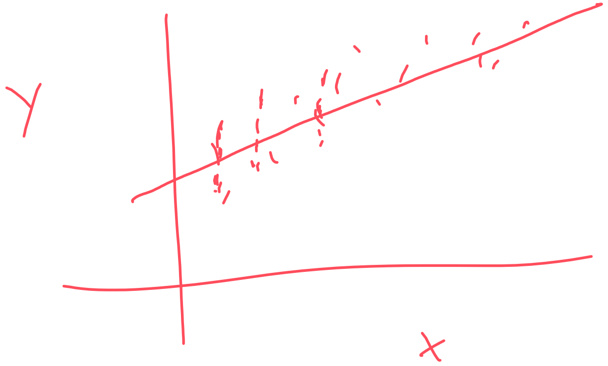
(fit model so it can find predictions)

$$E(Y_i|x_i) = \boxed{\beta_0 + \beta_1 x_i}$$

expected value (avg value)

Response

predictor



# Training a Model

- Once a class of models is chosen, we must define some criteria to **fit** (or train) the model

**Simple Linear Regression (SLR) Model**

$$E(Y_i|x_i) = \beta_0 + \beta_1 x_i$$

- **Loss function** - Criteria used to fit or train a model
  - For a given **numeric** response value, $y_i$ and prediction, $\hat{y}_i$

$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$

estimates or fitted Betas

observed

$$y_i - \hat{y}_i, \underbrace{(y_i - \hat{y}_i)^2}_{\text{Squared error loss}}, \underbrace{|y_i - \hat{y}_i|}_{\text{absolute deviations}}$$

# Training a Model

- Once a class of models is chosen, we must define some criteria to **fit** (or train) the model

**Simple Linear Regression (SLR) Model**

$$E(Y_i|x_i) = \beta_0 + \beta_1 x_i$$

- **Loss function** - Criteria used to fit or train a model
  - For a given **numeric** response value, $y_i$ and prediction, $\hat{y}_i$

$$y_i - \hat{y}_i, (y_i - \hat{y}_i)^2, |y_i - \hat{y}_i|$$

- We try to optimize the loss over all the observations used for training

$$\min_{\hat{\beta}_0, \hat{\beta}_1} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \qquad \min_{\hat{\beta}_0, \hat{\beta}_1} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

# Training (Fitting) the SLR Model

- Often use squared error loss (least squares regression)
- Nice solutions for our estimates exist!

$$\hat{\beta}_0 = \bar{y} - \bar{x}\hat{\beta}_1$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

# Training (Fitting) the SLR Model

- Often use squared error loss (least squares regression)

- Nice solutions for our estimates exist!

$$\hat{\beta}_0 = \bar{y} - \bar{x}\hat{\beta}_1$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

```
y <- bike_data$log_selling_price
x <- bike_data$log_km_driven
b1_hat <- sum((x-mean(x))*(y-mean(y)))/sum((x-mean(x))^2)
b0_hat <- mean(y)-mean(x)*b1_hat
c(round(b0_hat, 4), round(b1_hat, 4))
```

```
## [1] 14.6356 -0.3911
```

$\hat{\beta}_0$ $\hat{\beta}_1$

- Now we can find a prediction! Denoted as $\hat{y}$ = $14.6356 - 0.3911x$

# Training (Fitting) the SLR Model in R

- Use `lm()` function to fit in R
- Utilizes `formula` notation: `y ~ x` -> `response ~ model terms`

separate by + signs

```
slr_fit <- lm(log_selling_price ~ log_km_driven, data = bike_data)
slr_fit
```

$Y \sim X_1 + X_2 + X_1 : X_2 + I(X_1^2)$

Y X

```
##
## Call:
## lm(formula = log_selling_price ~ log_km_driven, data = bike_data)
##
## Coefficients:
##   (Intercept)  log_km_driven
##       14.6356        -0.3911
```

# Inference Using the SLR Model in R

- If we assume iid errors that are Normally distributed with the same variance, we can conduct inference!
    - Confidence intervals and hypothesis tests around the slope parameter
    - Use `summary()` (generic function!) on our fitted model

```
summary(slr_fit)
```

```
##
## Call:
## lm(formula = log_selling_price ~ log_km_driven, data = bike_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9271 -0.3822 -0.0337  0.3794  2.5656
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    14.63557    0.18455   79.31   <2e-16 ***
## log_km_driven  -0.39109    0.01837  -21.29   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5953 on 1059 degrees of freedom
## Multiple R-squared:  0.2997,	Adjusted R-squared:  0.299
```

# Inference Using the SLR Model in R

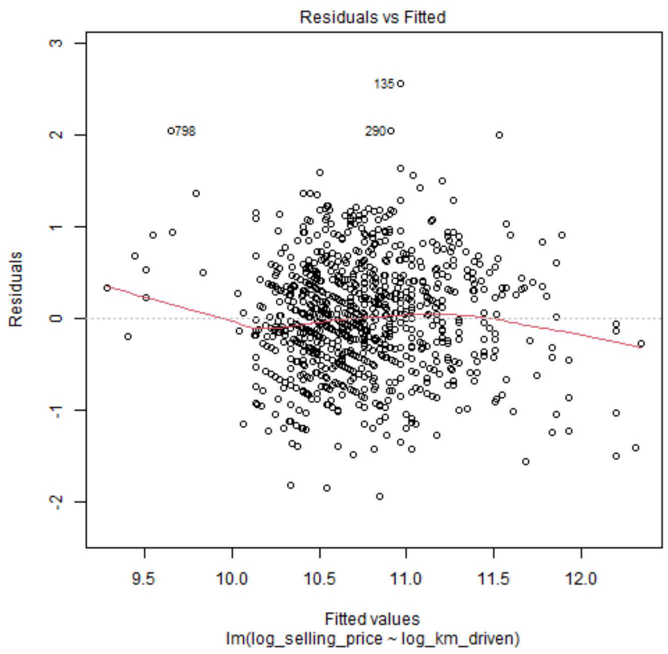
- If we assume iid errors that are Normally distributed with the same variance, we can conduct inference!
    - Can use `anova()` to get Analysis of Variance information

```
anova(slr_fit)
```

```
## Analysis of Variance Table
##
## Response: log_selling_price
##                 Df Sum Sq Mean Sq F value    Pr(>F)
## log_km_driven    1 160.58 160.583  453.19 < 2.2e-16 ***
## Residuals     1059 375.24   0.354
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

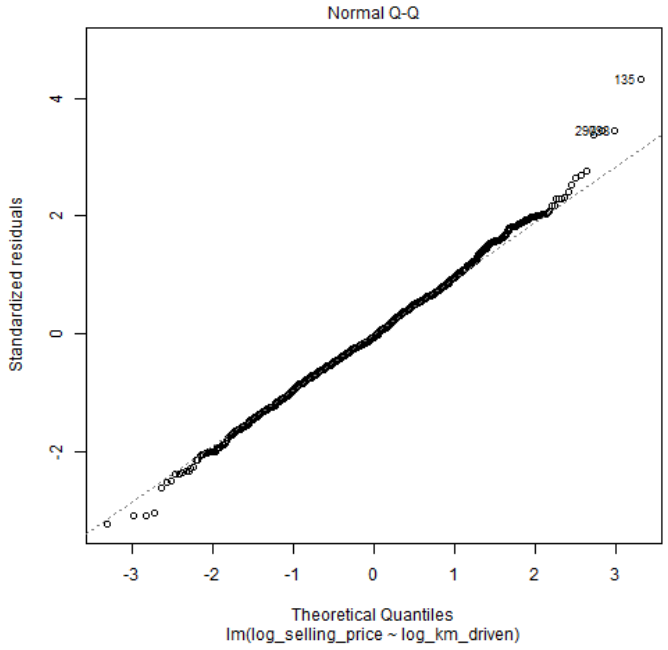
# Inference Using the SLR Model in R

- If we assume iid errors that are Normally distributed with the same variance, we can conduct inference!
  - Residual diagnostics can be found via `plot()` on the fitted model



# Inference Using the SLR Model in R

- If we assume iid errors that are Normally distributed with the same variance, we can conduct inference!
  - Residual diagnostics can be found via `plot()` on the fitted model



# Prediction Using the SLR Model in R

- Can use the line for prediction with `predict()`!
    - Another generic function in R

```
predict
```

```
## function (object, ...)
## UseMethod("predict")
## <bytecode: 0x00000132166ac2d0>
## <environment: namespace:stats>
```

data frame w/ values of our predictor(s) to predict for

```
predict.lm
```

fit

```
## function (object, newdata, se.fit = FALSE, scale = NULL, df = Inf,
##     interval = c("none", "confidence", "prediction"), level = 0.95,
##     type = c("response", "terms"), terms = NULL, na.action = na.pass,
##     pred.var = res.var/weights, weights = 1, ...)
## {
##     tt <- terms(object)
##     if (!inherits(object, "lm"))
##         warning("calling predict.lm(<fake-lm-object>) ...")
##     if (missing(newdata) || is.null(newdata)) {
##         mm <- X <- model.matrix(object)
##         mmDone <- TRUE
##         offset <- object$offset
##     }
##     else {
##         Terms <- delete.response(tt)
```

# Prediction Using the SLR Model in R

- Can use the line for prediction with `predict()`!
  - Should supply fitted object and `newdata`
    - An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.

```
predict(slr_fit, newdata = data.frame(log_km_driven = c(log(1000), log(10000), log(100000))))
```

Same name as predictor in model

```
##        1        2        3
## 11.93404 11.03353 10.13302
```

```
exp(predict(slr_fit, newdata = data.frame(log_km_driven = c(log(1000), log(10000), log(100000)))))
```

```
##         1         2         3
## 152365.60  61915.64  25160.19
```

# Quantifying How Well the Model Predicts

We use a **loss** function to fit the model. We use a **metric** to evaluate the model!

- Often use the same loss function for fitting and as the metric
- For a given **numeric** response value, $y_i$ and prediction, $\hat{y}_i$

$$(y_i - \hat{y}_i)^2, |y_i - \hat{y}_i|$$

- Incorporate all points via

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

MSE MAD

MAE

# Metric Function

- For a numeric response, we commonly use squared error loss as our metric to evaluate a prediction

$$L(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$$

- Use Root Mean Square Error as a **metric** across all observations

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{y}_i)} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

# Commonly Used Metrics

For prediction (numeric response)

- Mean Squared Error (MSE) or Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE or MAD - deviation)

$$L(y_i, \hat{y}_i) = |y_i - \hat{y}_i|$$

- Huber Loss
  - Doesn't penalize large mistakes as much as MSE

# Commonly Used Metrics

For prediction (numeric response)

- Mean Squared Error (MSE) or Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE or MAD - deviation)

$$L(y_i, \hat{y}_i) = |y_i - \hat{y}_i|$$

- Huber Loss
  - Doesn't penalize large mistakes as much as MSE

For classification (categorical response)

- Accuracy
- log-loss
- AUC
- F1 Score

# Evaluating our SLR Model

- We could find our metric for our SLR model using the training data
    - Called **training error**

```
head(predict(slr_fit))
```

```
##        1        2        3        4        5        6
## 12.34461 11.25681 10.96222 10.70779 10.74337 10.33280
```

predictions for x's the model was trained on

```
mean((bike_data$log_selling_price-predict(slr_fit))^2)
```

$Y$ $\hat{Y}$

MSE

```
## [1] 0.3536708
```

```
sqrt(mean((bike_data$log_selling_price-predict(slr_fit))^2))
```

RMSE

```
## [1] 0.5947023
```

- Doesn't tell us how well we do on data we haven't seen!

# Training vs Test Sets

Ideally we want our model to predict well for observations **it has yet to see**!

- For *multiple* linear regression models, our training MSE will always decrease as we add more variables to the model...

- We'll need an independent **test** set to predict on (more on this shortly!)

# Big Picture Modeling

Supervised Learning methods try to relate predictors to a response variable through a model

- Lots of common models
  - Regression models
  - Tree based methods
  - Naive Bayes
  - k Nearest Neighbors
  - ...
- For a set of predictor values, each will produce some prediction we can call $\hat{y}$
- Evaluate model via a metric
- Will use an independent test set or cross-validation to more accurately judge our model