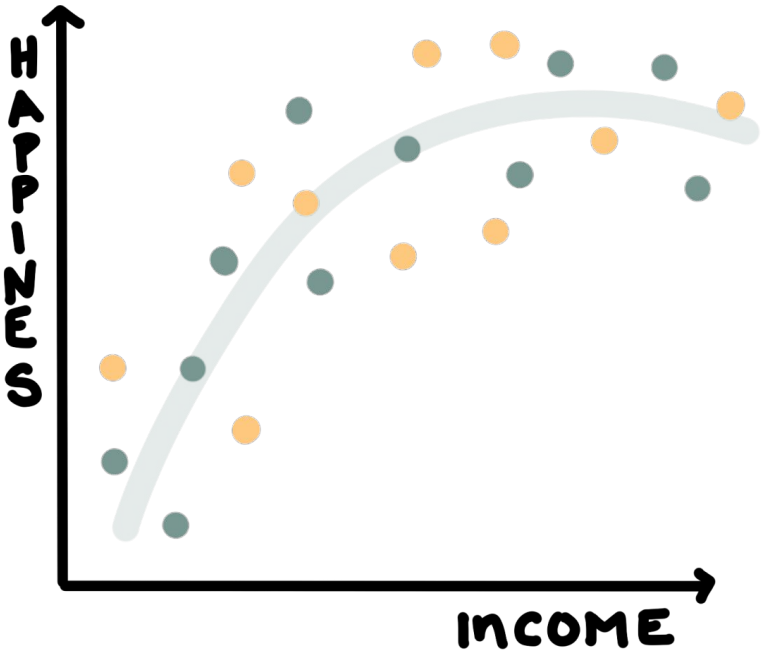
Data Science Review

Dr. Chelsea Parlett-Pelleriti

Outline

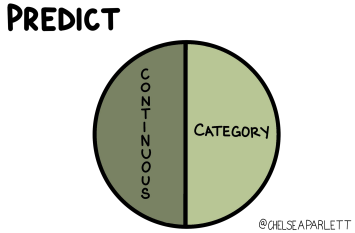
- The Bias-Variance Tradeoff
- Model Validation
- Loss Functions
- Hyperparameter Tuning
- Regression vs. Classification
- Supervised vs. Unsupervised
- Gradient Descent

Bias and Variance



data we have now

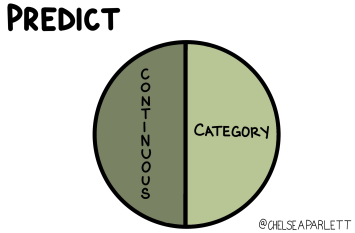
future data



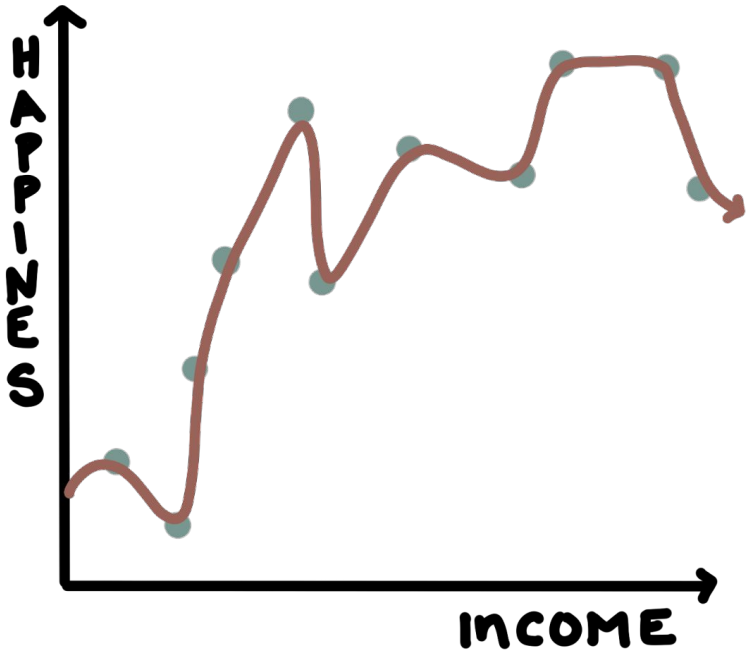
Bias and Variance

data we have now

future data

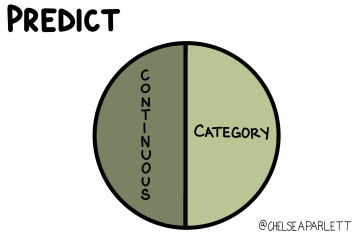


Bias and Variance

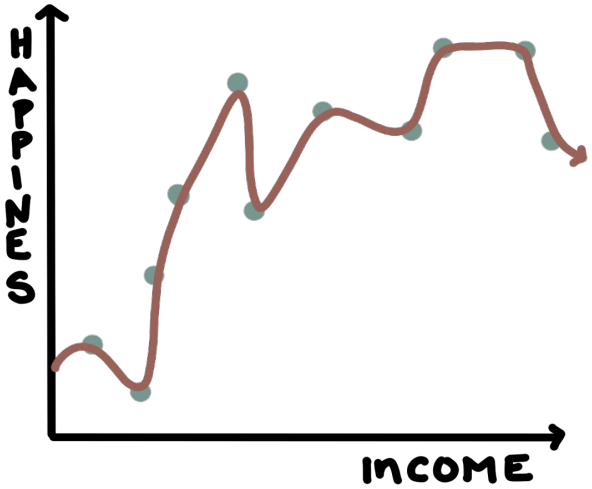


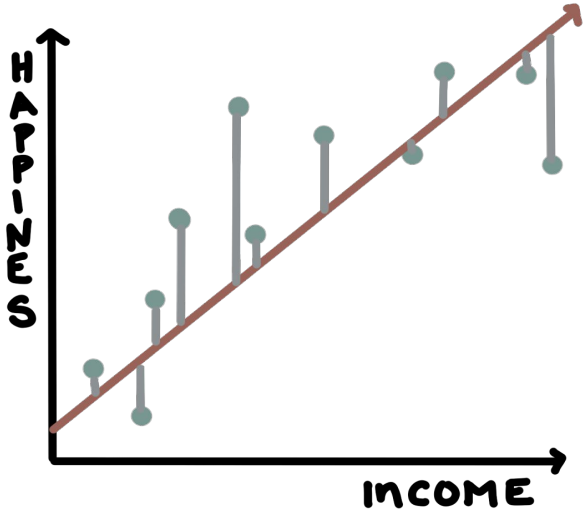
data we have now

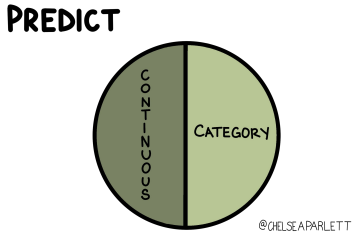
future data



Bias and Variance



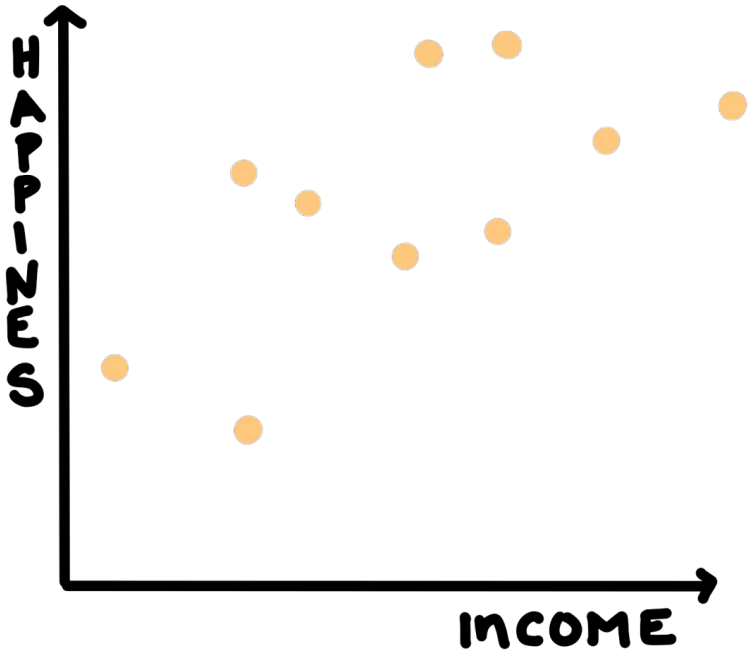




data we have now

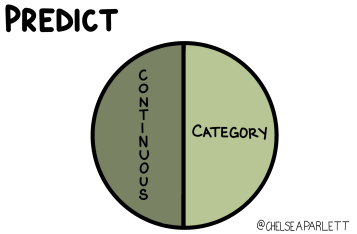
future data

Bias and Variance

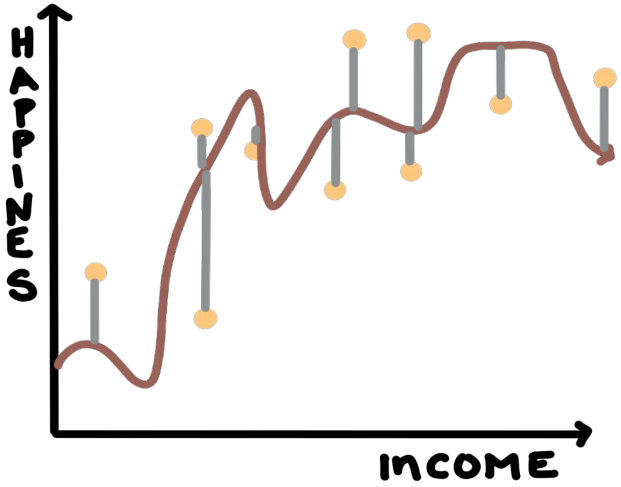


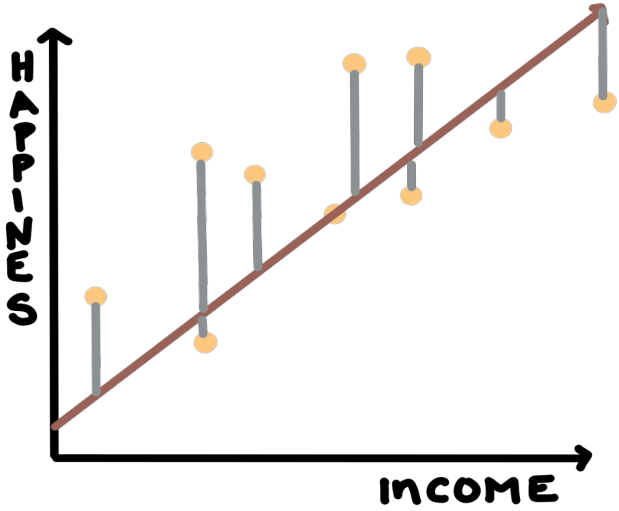
data we have now

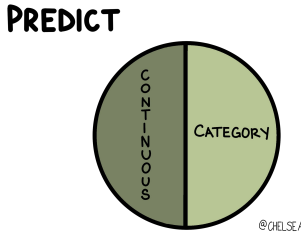
future data



Bias and Variance



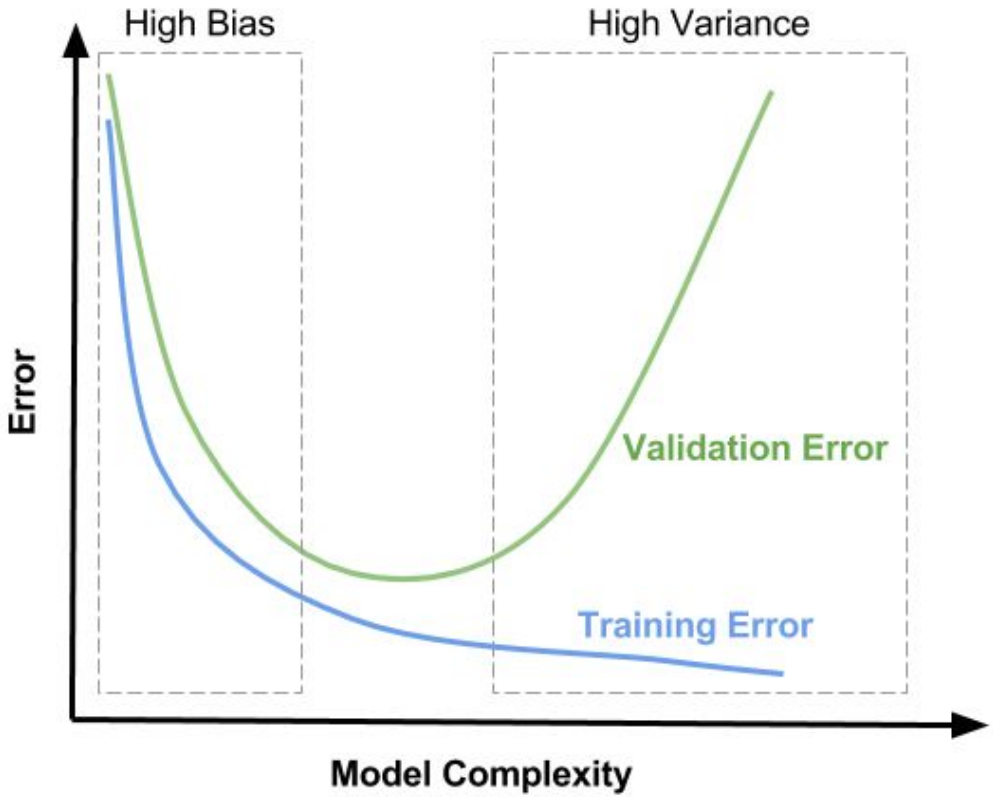




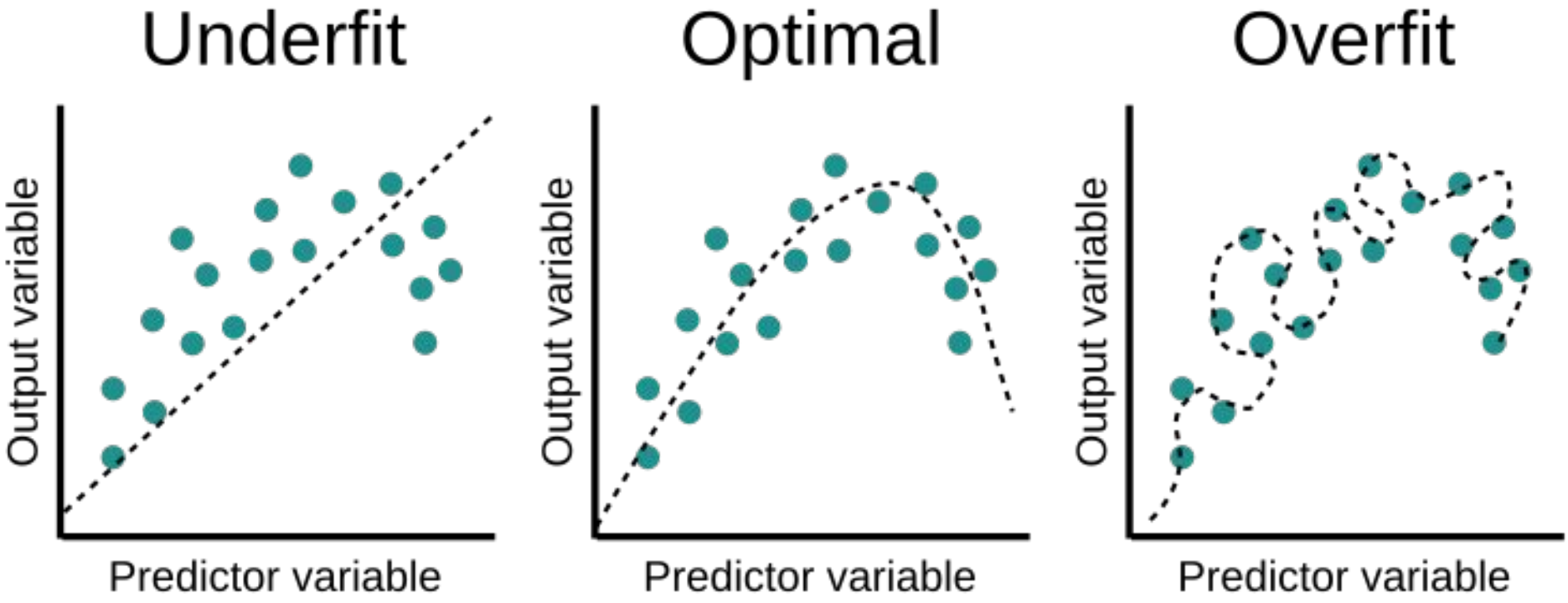
data we have now

future data

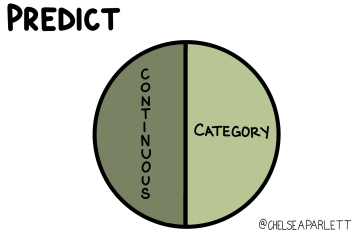
Bias and Variance



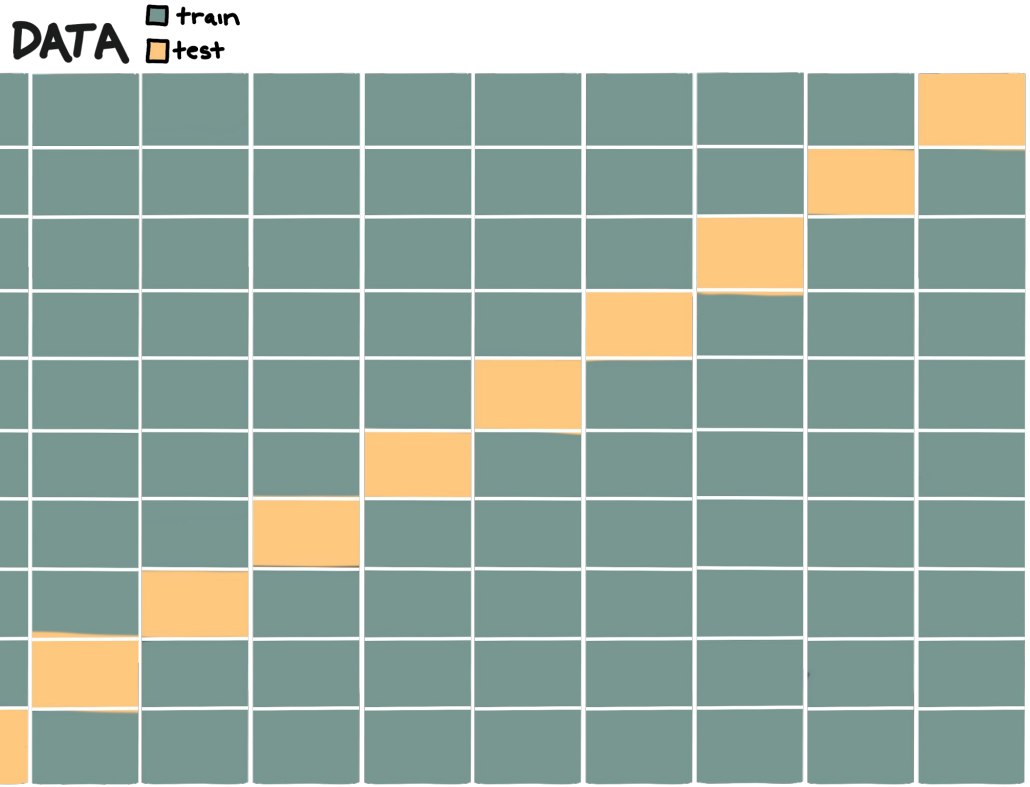
The Bias Variance Tradeoff

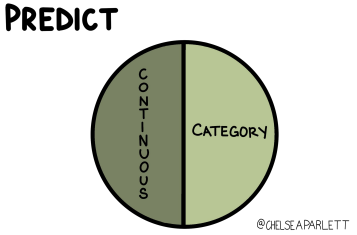


Train Test Split

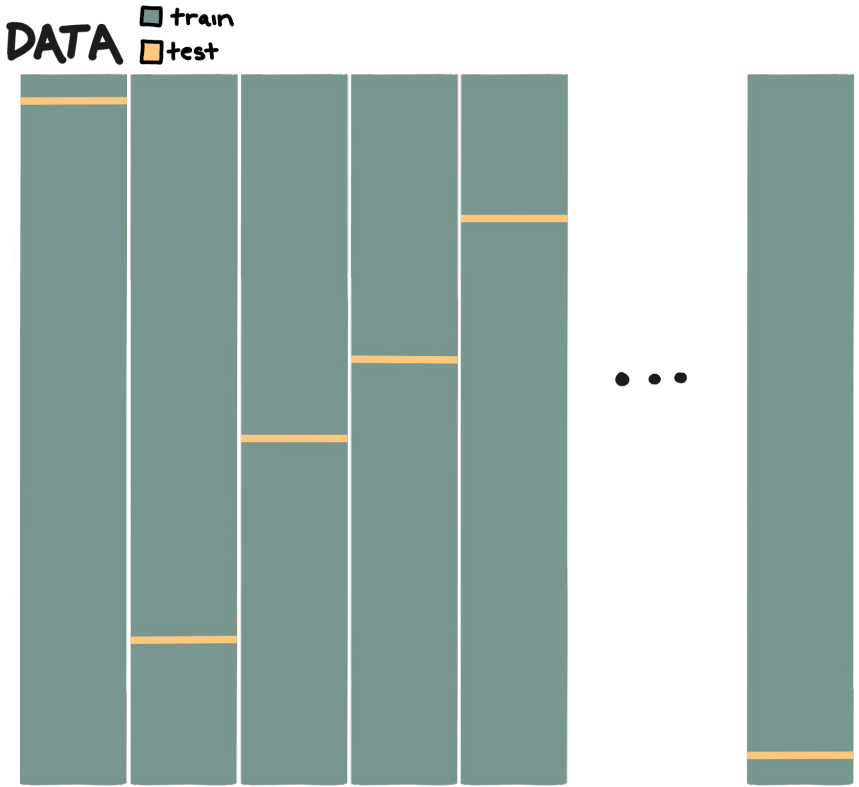


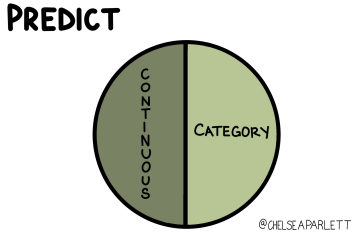
KFold Cross Validation





LOO Cross Validation





Loss Functions

A **metric** that measures the **performance** of your model where **lower** is better

Common Loss Functions (continuous)

MSE

$$\frac{1}{N}\sum_{i=1}^{N}(\text{actual} - \text{predicted})^2$$

MAE

$$\frac{1}{N}\sum_{i=1}^{N}|\text{actual} - \text{predicted}|$$

Common Loss Functions (categorical)

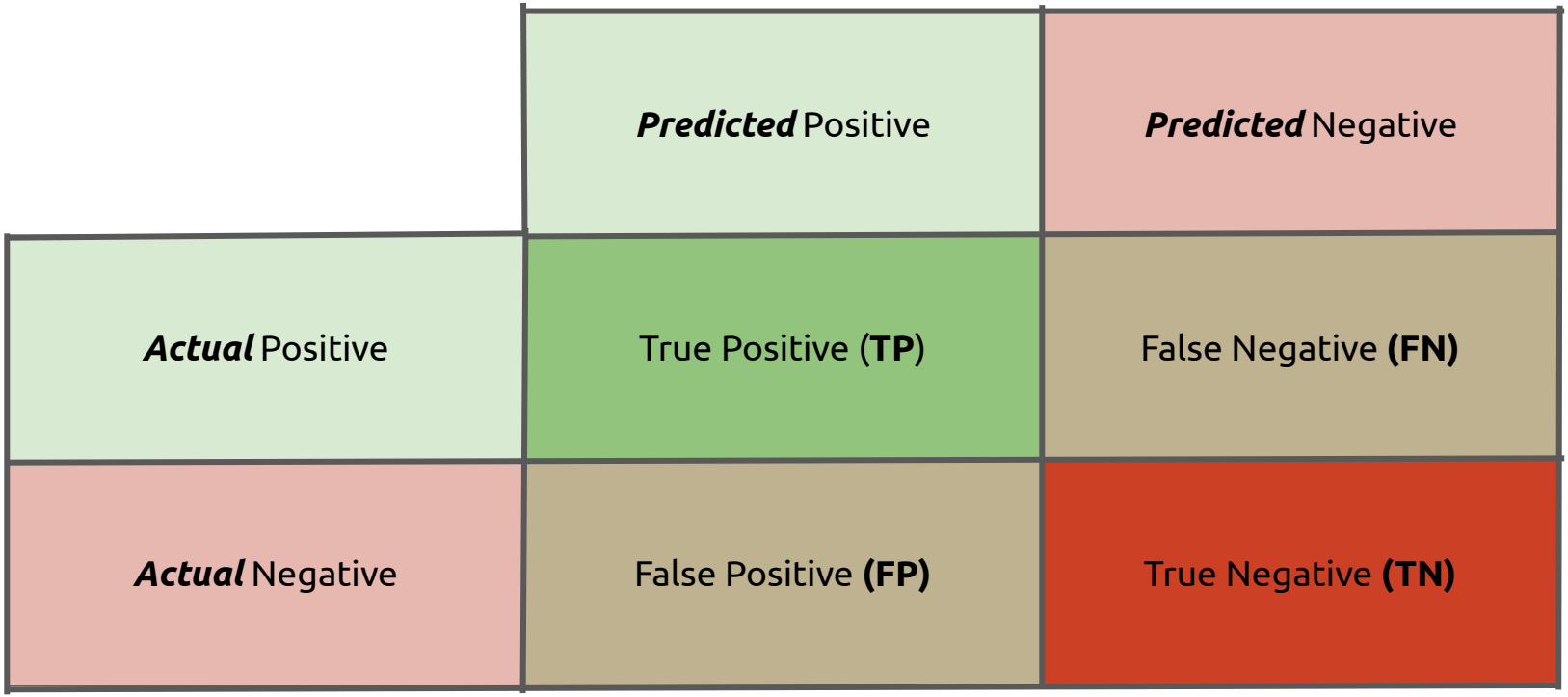
Log Loss/ Binary Cross Entropy

$$-\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p_i) + (1 - y_i) \cdot log(1 - p_i)$$

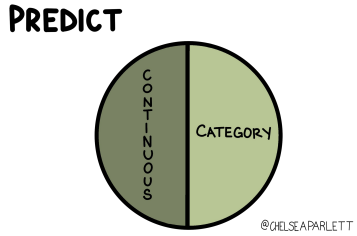
Hinge Loss

$$\sum_{i=1}^{N} max(0, 1 - t_i \cdot y_i)$$

Classification Metrics



Accuracy



Correct Predictions

True Positive (**TP**) + True Negative **(TN)**

All Predictions

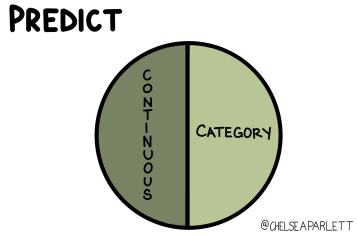
False Negative **(FN)** + False Positive **(FP)** +

True Positive (**TP**) + True Negative **(TN)**

"How often is the model correct?"

	Predicted Positive	***Predicted*** Negative
Actual Positive	True Positive **(TP)**	False Negative **(FN)**
Actual Negative	False Positive **(FP)**	True Negative **(TN)**

Sensitivity/Recall



Correctly Predicted Positives

True Positive (**TP**)

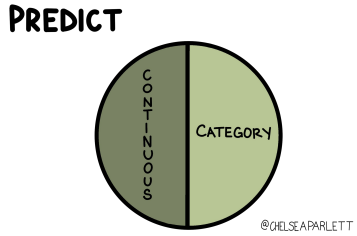
Actual Positives

False Negative **(FN)** + True Positive (**TP**)

"How often is the model correct for Positive Cases?"

	Predicted Positive	***Predicted*** Negative
Actual Positive	True Positive **(TP)**	False Negative **(FN)**
Actual Negative	False Positive **(FP)**	True Negative **(TN)**

Specificity



Correctly Predicted Negatives

True Negative **(TN)**

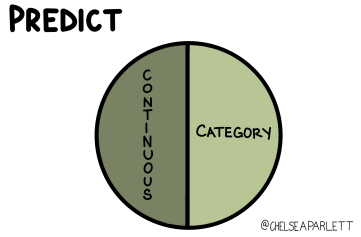
Actual Negatives

True Negative **(TN)** + False Positive **(FP)**

"How often is the model correct for Negative Cases?"

	Predicted Positive	***Predicted*** Negative
Actual Positive	True Positive **(TP)**	False Negative **(FN)**
Actual Negative	False Positive **(FP)**	True Negative **(TN)**

Precision



Correctly Predicted Positives

True Positive (**TP**)

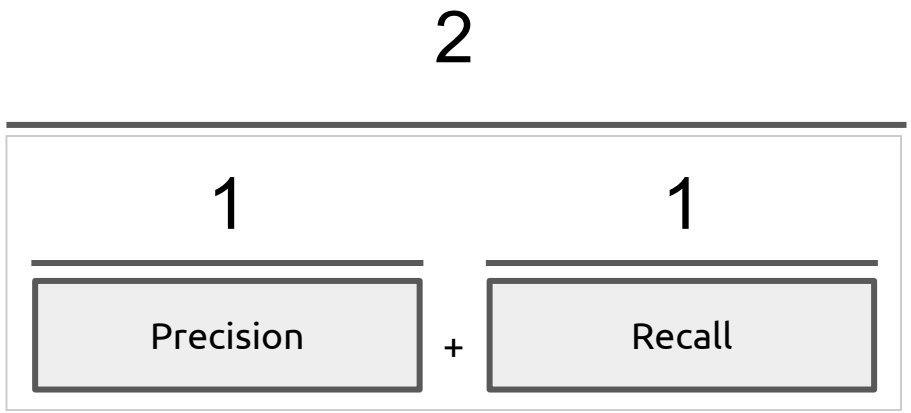
Actual Positives

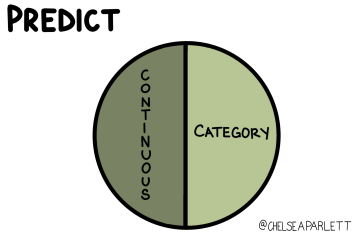
False Positive **(FP)** + True Positive (**TP**)

"How many of the predicted Positives are correct?"

	Predicted Positive	***Predicted*** Negative
Actual Positive	True Positive **(TP)**	False Negative **(FN)**
Actual Negative	False Positive **(FP)**	True Negative **(TN)**

F1 Score

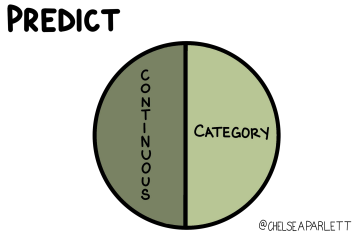




Combination of Precision (how often predicted positives ARE positive) and Recall (how often we correctly predict actual positives)

	Predicted Positive	***Predicted*** Negative
Actual Positive	True Positive **(TP)**	False Negative **(FN)**
Actual Negative	False Positive **(FP)**	True Negative **(TN)**

ROC AUC



Correctly Predicted Positives

True Positive (**TP**)

Actual Positives

False Negative (**FN**) + True Positive (**TP**)

	Predicted Positive	***Predicted*** Negative
Actual Positive	True Positive (**TP**)	False Negative **(FN)**
Actual Negative	False Positive **(FP)**	True Negative **(TN)**

1 -

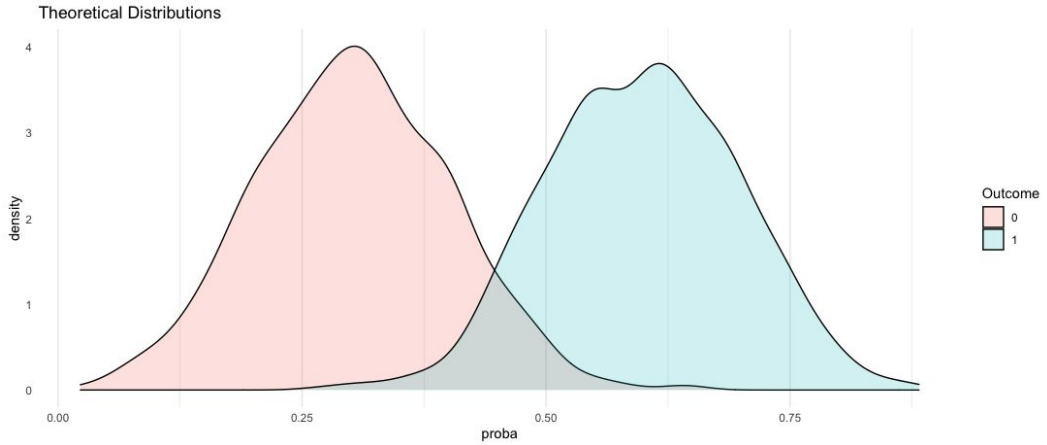
Correctly Predicted Negatives

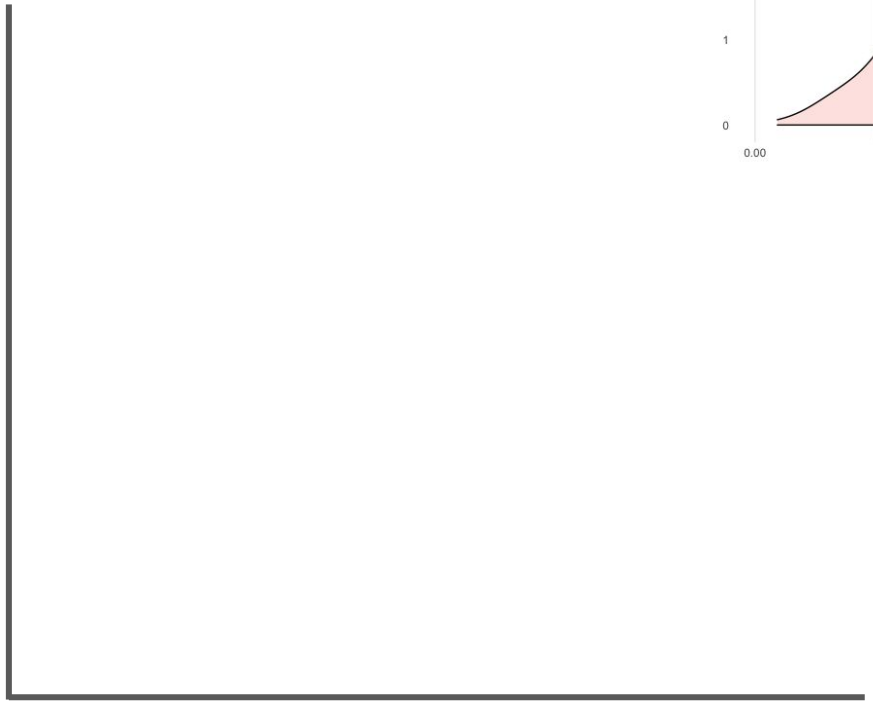
True Negative (**TN**)

Actual Negatives

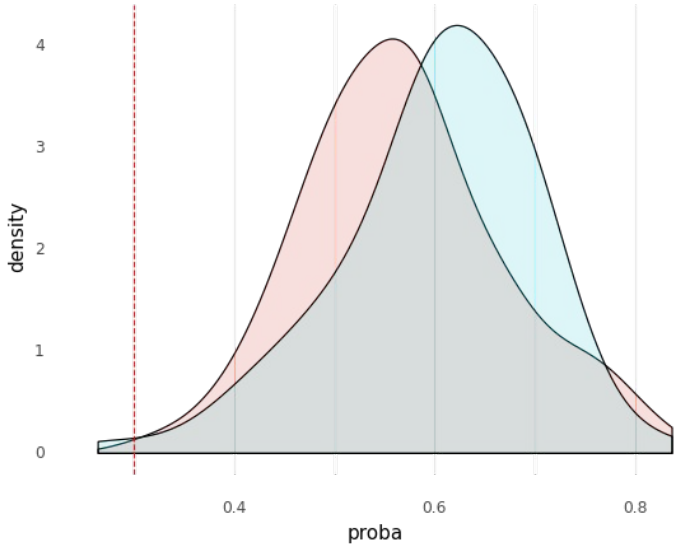
True Negative (**TN**) + False Positive (**FP**)

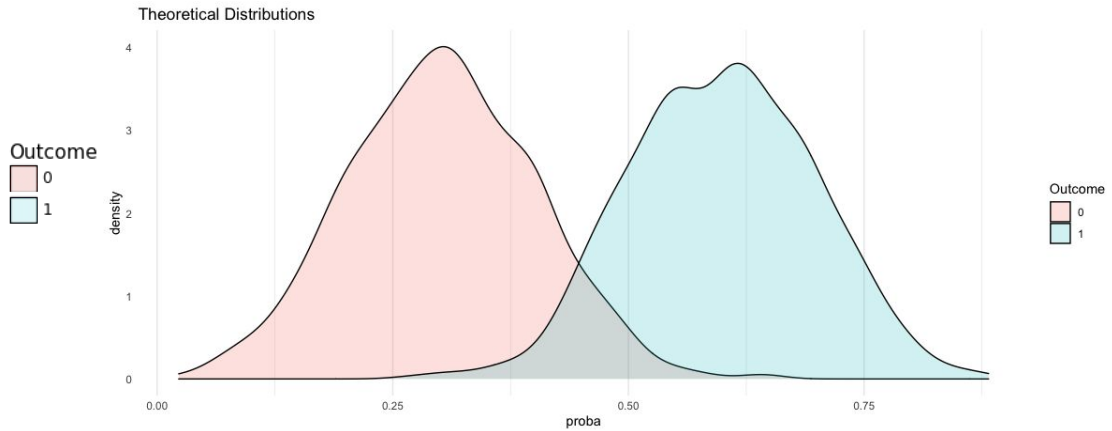
ROC AUC


Theoretical Distributions
density
proba
0.00
0.25
0.50
0.75
0
1
2
3
4
Outcome
0
1



ROC AUC


density
proba
0
1
2
3
4
0.4
0.6
0.8


Theoretical Distributions
Outcome
0
1
density
proba
0.00
0.25
0.50
0.75

Hyperparameter Tuning

Parameters: values in your model that your model chooses (e.g. coefficients)

Hyperparameters: values in your model that your model does NOT choose (e.g. K in KNN, max_depth in DTs)

Hyperparameter Tuning

Two Options

- Choose the hyperparameter based on *domain knowledge*
  - “Because there are no more than 420 pitchers in the MLB, I will set K to be 300”
- Choose the hyperparameter using *hyperparameter tuning*
  - Train-Test-**Validation**: The validation set serves as a pretend-test-set for us to see how well different hyperparameter values do on unseen data *without touching our actual test set*

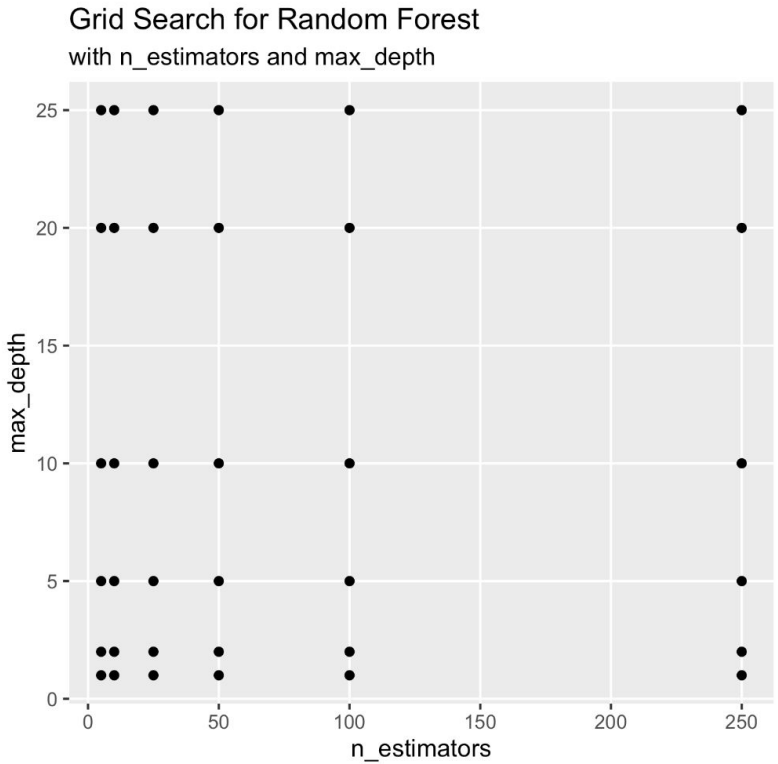
No Hyperparameters:

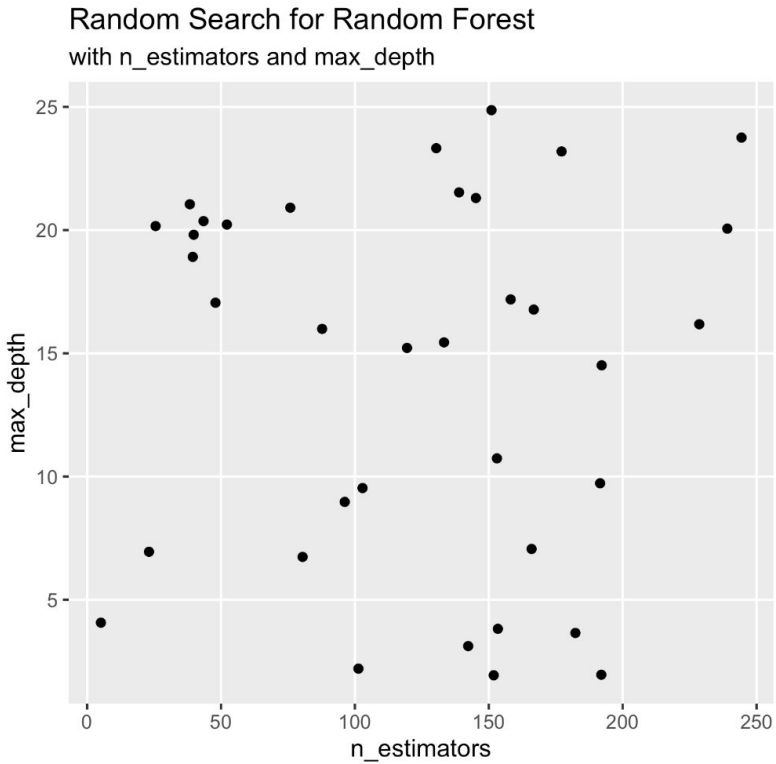
With Hyperparameters:

Hyperparameter Tuning

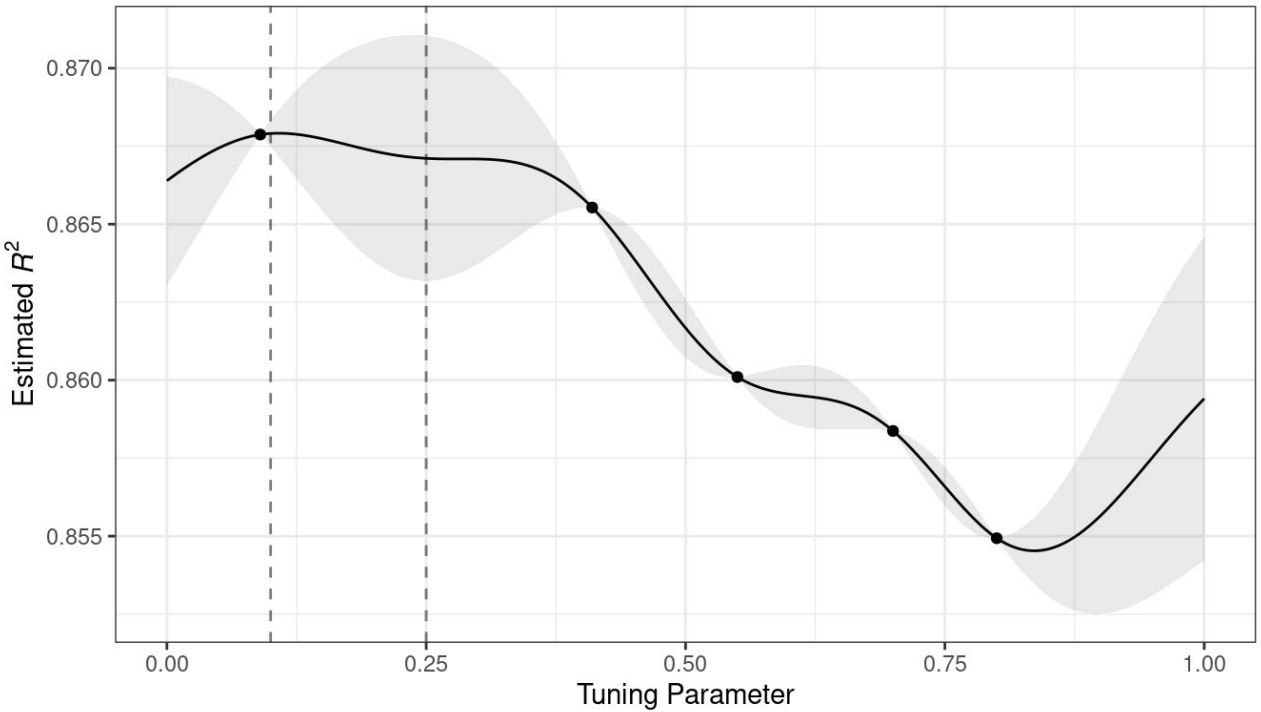
- **Grid Search**
- Random Search
- Bayesian Optimization
- Simulated Annealing

Hyperparameter Tuning



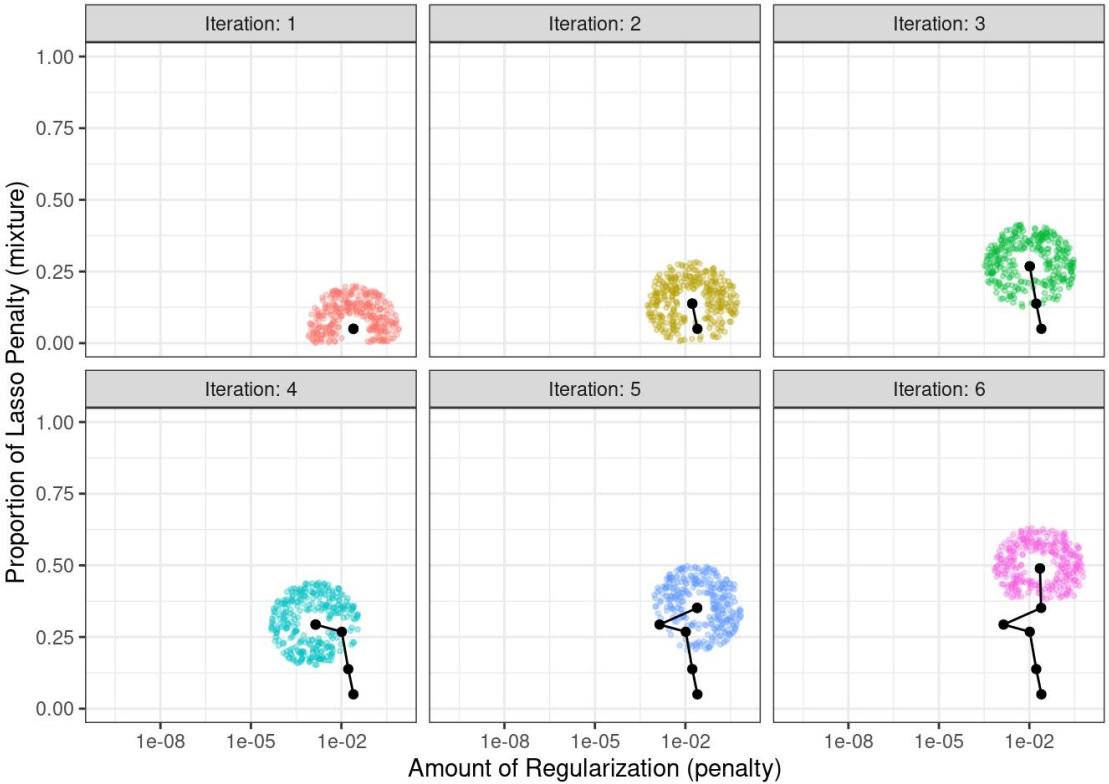


Hyperparameter Tuning



From: https://www.tmwr.org/iterative-search.html

Hyperparameter Tuning

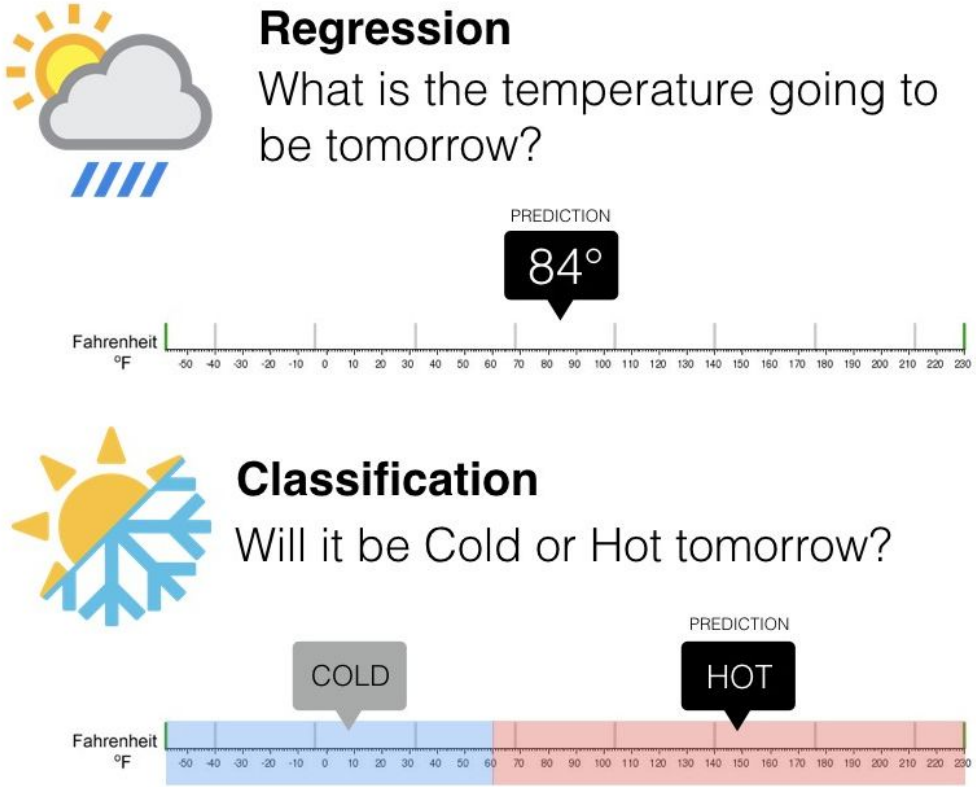


From: https://www.tmwr.org/iterative-search.html

Regression vs. Classification

Common Regression Models:

- Linear Regression
- Polynomial Regression
- Regression Trees/Random Forests
- Support Vector Regression
- XGBoost
- Neural Nets
- …



Common Classification Models:

- Logistic Regression
- Decision Trees
- K-Nearest Neighbors
- Support Vector Machines
- Naive Bayes
- XGBoost
- Neural Nets
- …

Supervised vs. Unsupervised ML

Supervised

Has **labeled data** (we know the correct answers and use them to train the model)

Goal: to accurately predict our target value

e.g. classification or regression

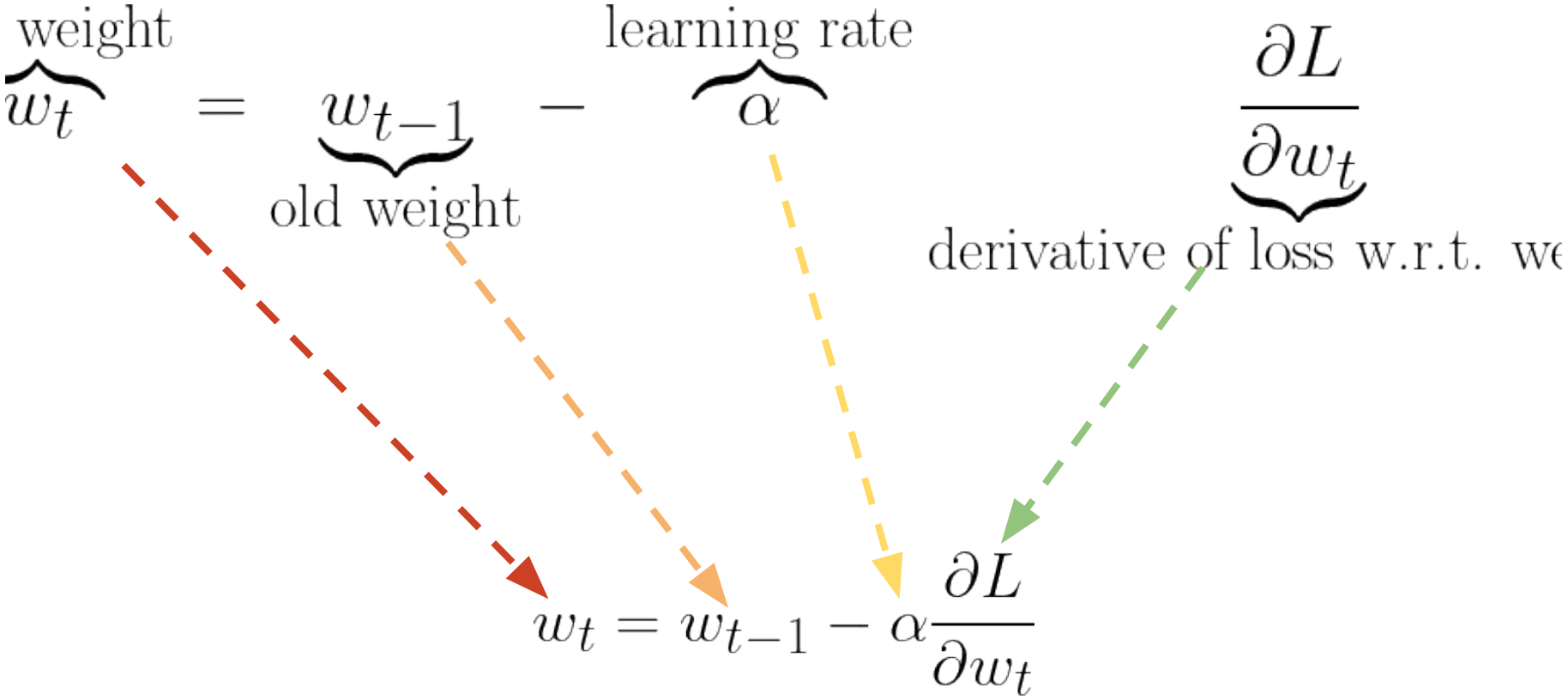
Unsupervised

Does **not** have labeled data (there are no correct answers)

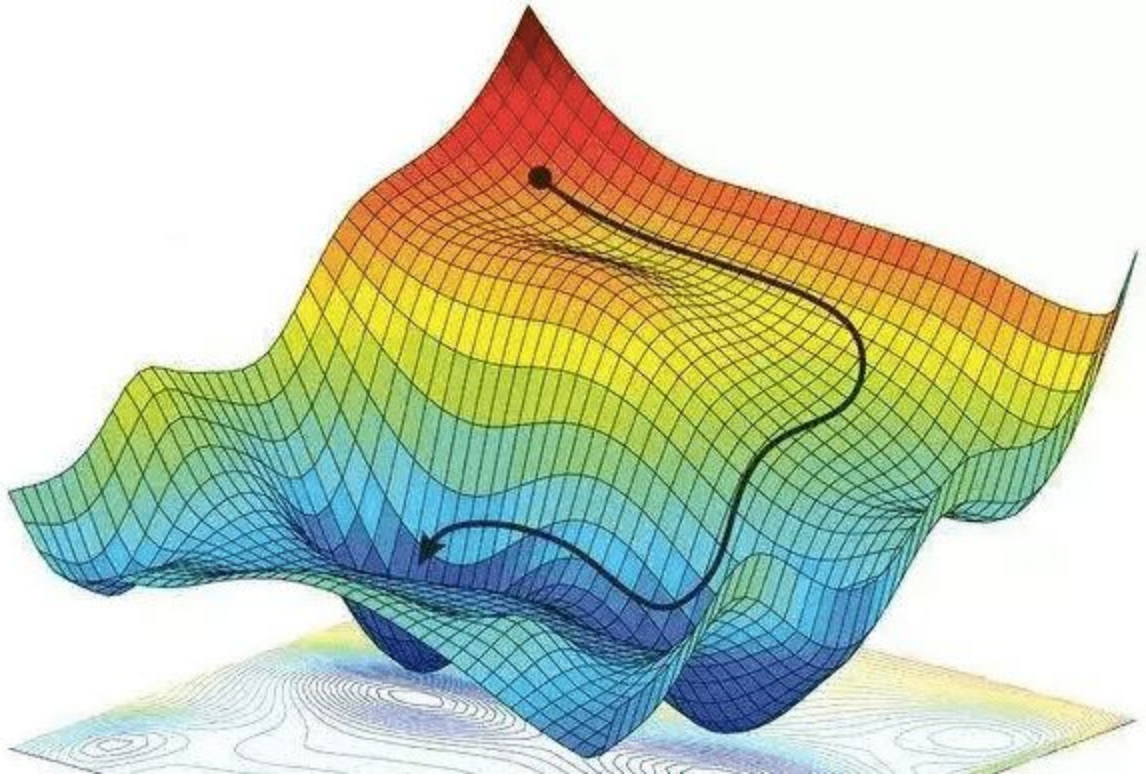
Goal: to create/recognize latent structure in the data

e.g. PCA, clustering

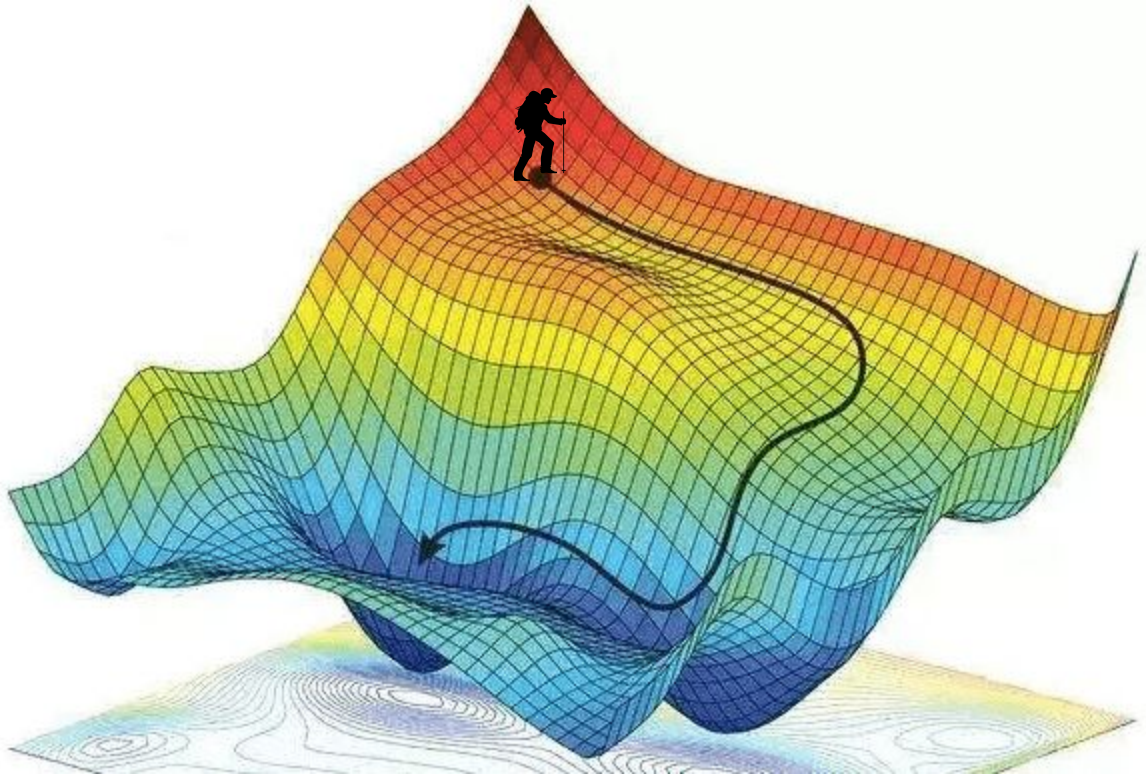
Gradient Descent



Gradient Descent



Gradient Descent



Gradient Descent Ideas

1. The goal is to minimize the loss function

Partial Derivatives

$$f = x^2 + xy + y^2$$

How does **f(x,y)** change when **x** changes? What about when **y** changes?

Partial Derivatives

$$f = x^2 + xy + y^2$$

How does **f(x,y)** change when **x** changes? What about when **y** changes?

$$\frac{\partial f}{\partial x} = 2x + y \qquad \frac{\partial f}{\partial y} = x + 2y$$

Gradient

$$f = x^2 + xy + y^2$$

How does **f(x,y)** change when **x** changes? What about when **y** changes?

$$\frac{\partial f}{\partial x} = 2x + y \qquad \frac{\partial f}{\partial y} = x + 2y$$

$$\begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x + y \\ x + 2y \end{bmatrix}$$

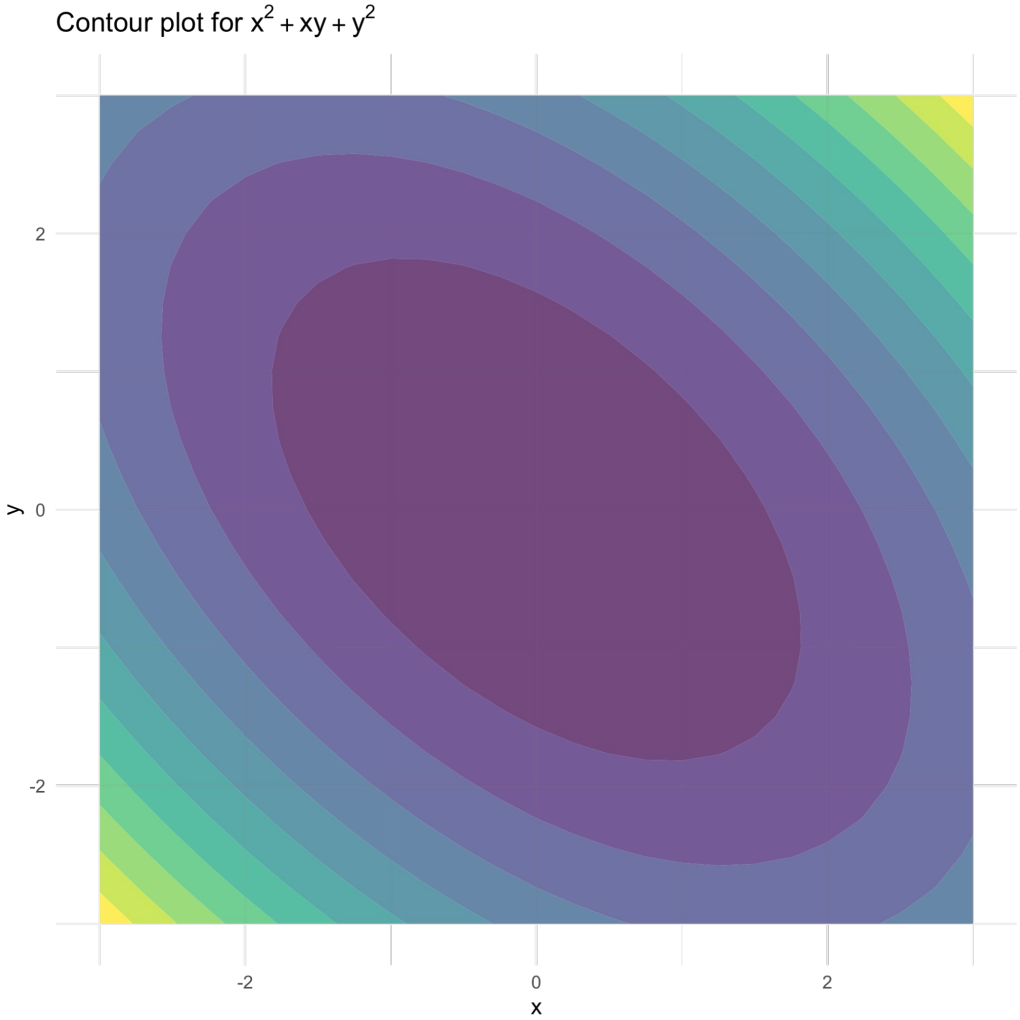
Gradient

$$f = x^2 + xy + y^2$$

How does **f(x,y)** change when **x** changes? What about when **y** changes?

$$\frac{\partial f}{\partial x} = 2x + y \qquad \frac{\partial f}{\partial y} = x + 2y$$

$$\begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x + y \\ x + 2y \end{bmatrix}$$



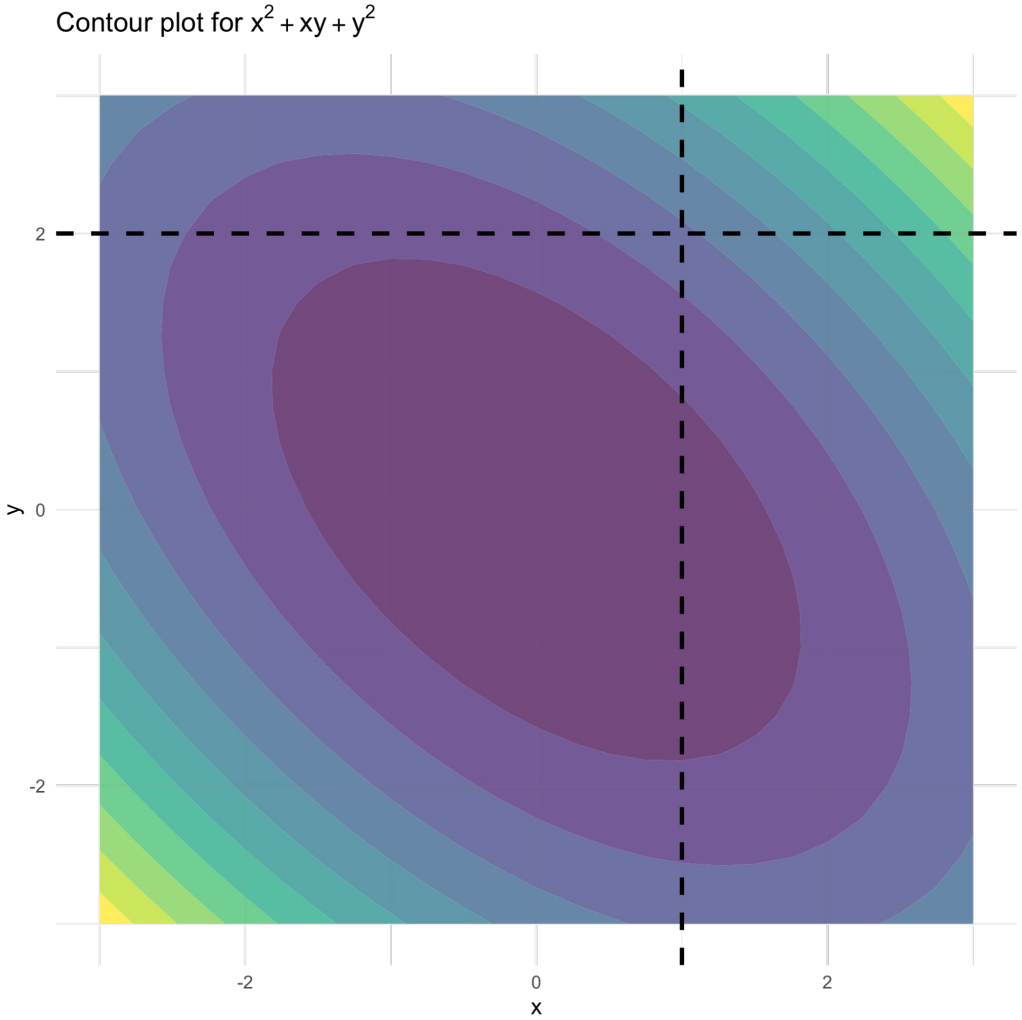
Gradient

$$f = x^2 + xy + y^2$$

How does **f(x,y)** change when **x** changes? What about when **y** changes?

$$\frac{\partial f}{\partial x} = 2x + y \qquad \frac{\partial f}{\partial y} = x + 2y$$

$$\begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x + y \\ x + 2y \end{bmatrix}$$



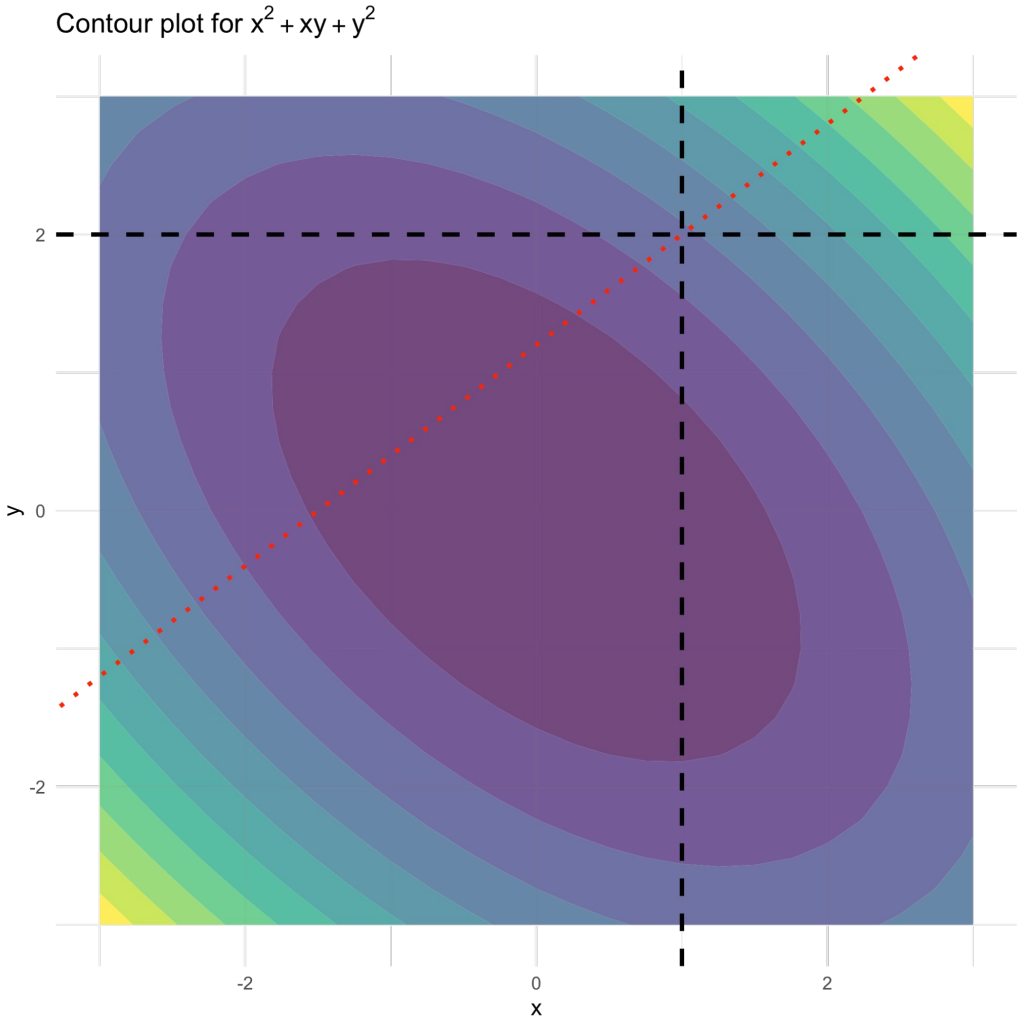
Gradient

$$f = x^2 + xy + y^2$$

How does **f(x,y)** change when **x** changes? What about when **y** changes?

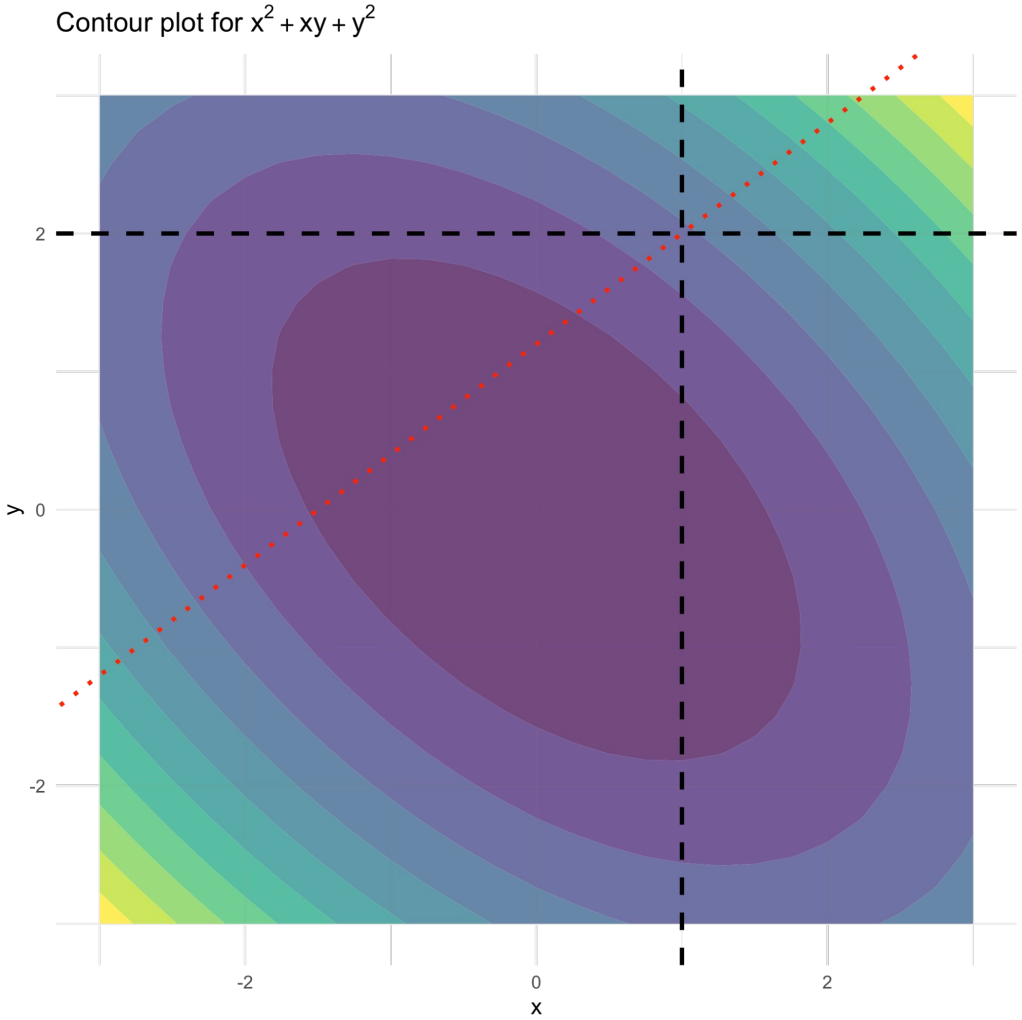
$$\frac{\partial f}{\partial x} = 2x + y \qquad \frac{\partial f}{\partial y} = x + 2y$$

$$\begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x + y \\ x + 2y \end{bmatrix}$$



Learning Rate

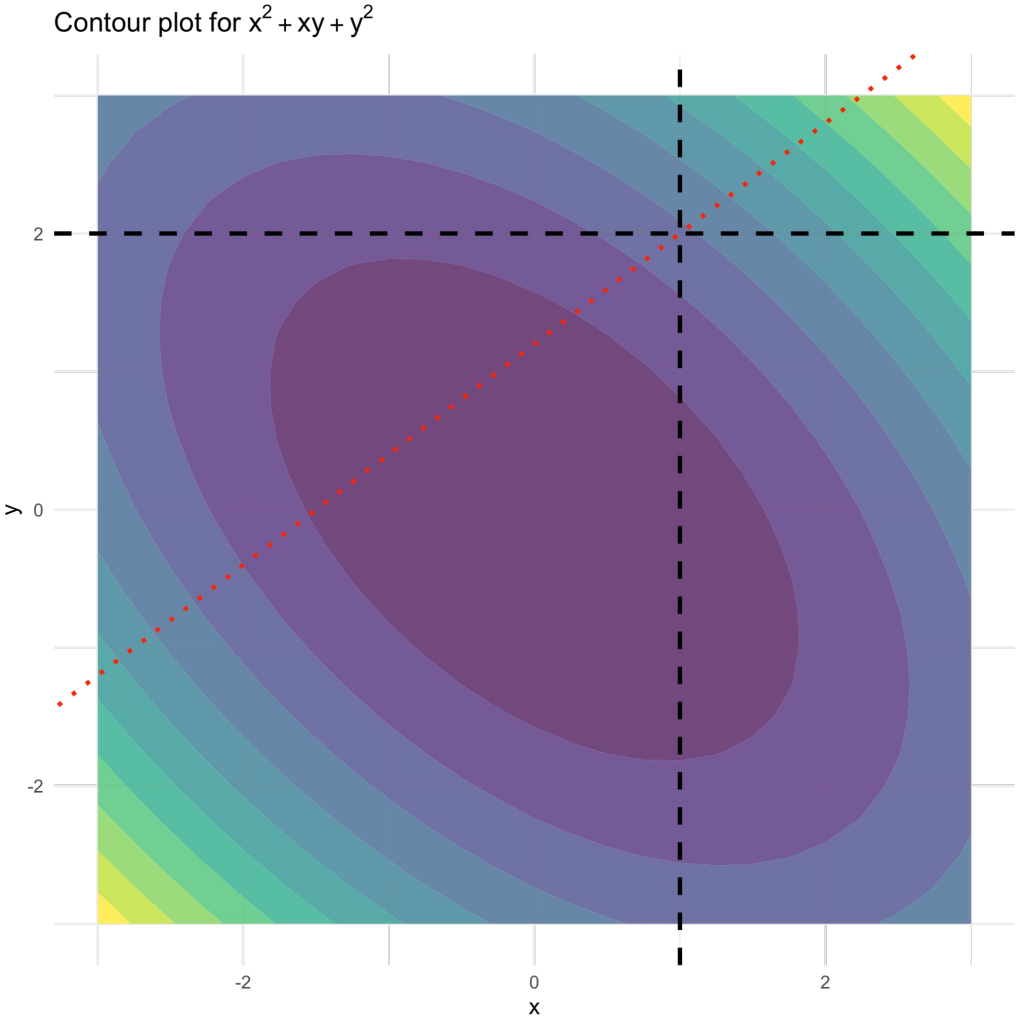
$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} - \alpha \begin{bmatrix} \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$



Learning Rate

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} - \alpha \begin{bmatrix} \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

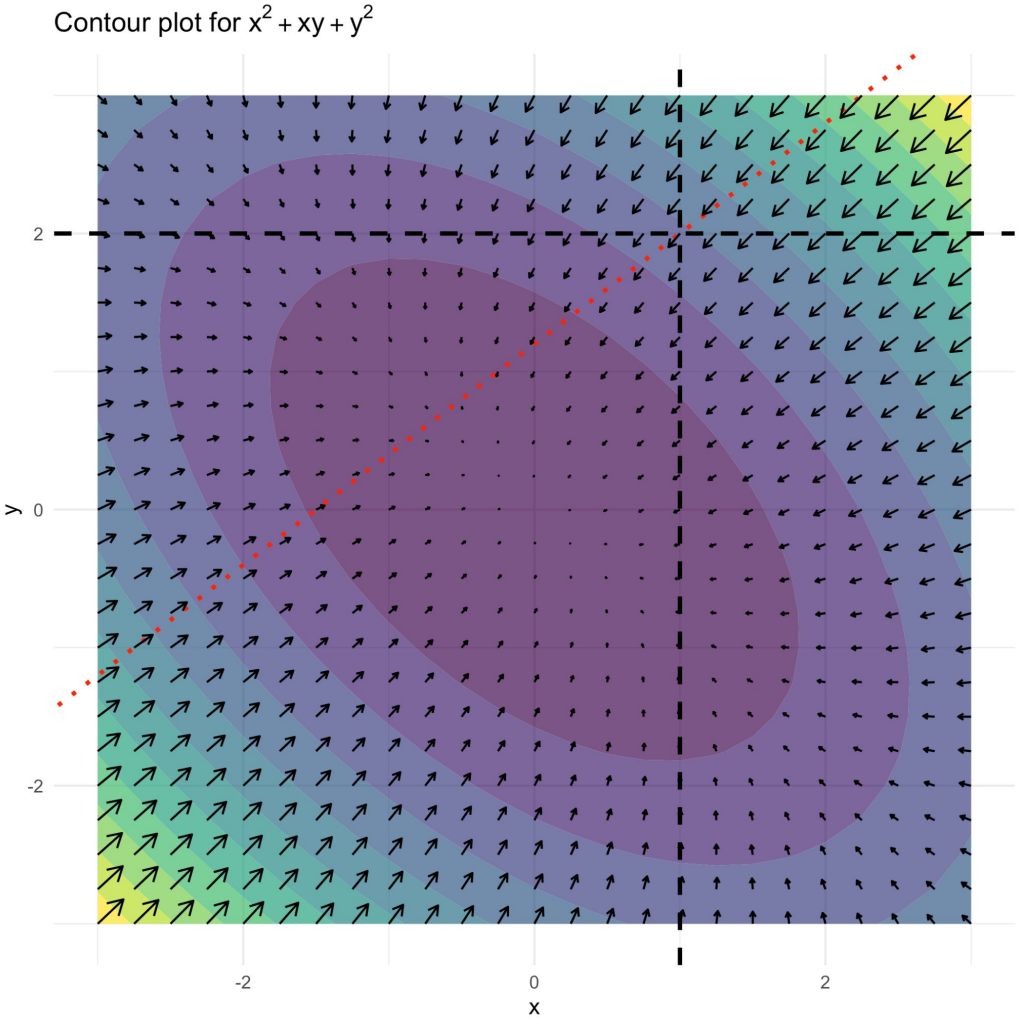
Gradients are COOL…but you can just think of it as what adjustments we should make to each of our parameters



Learning Rate

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} - \alpha \begin{bmatrix} \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Gradients are COOL…but you can just think of it as what adjustments we should make to each of our parameters



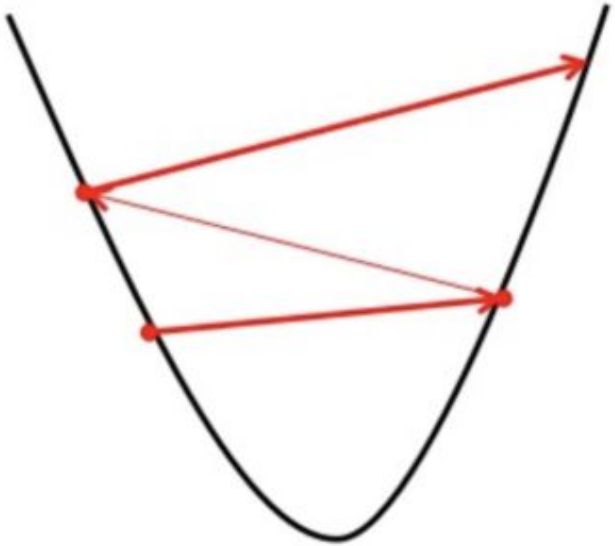
Gradient Descent Ideas

1. The goal is to minimize the loss function
2. The gradient tells us which direction to move in
    - In other words, it tells us what adjustments to make to each parameter

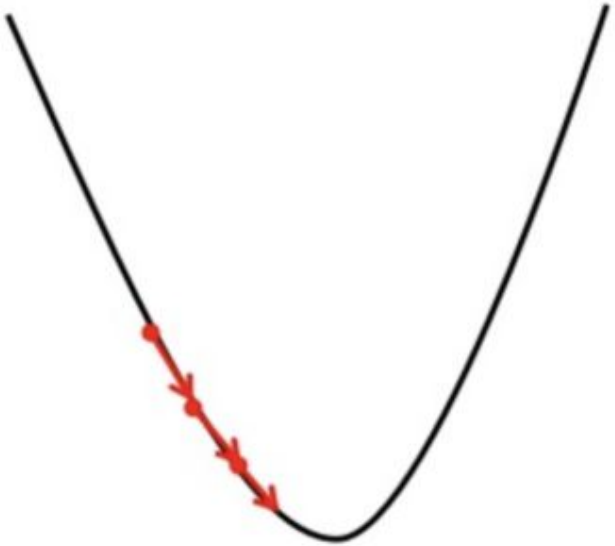
Learning Rate

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} - \alpha \begin{bmatrix} \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Big learning rate



Small learning rate



Gradient Descent Ideas

1. The goal is to minimize the loss function
2. The gradient tells us which direction to move in
    - In other words, it tells us what adjustments to make to each parameter
3. The Learning Rate controls how big of a step we take
    - Small steps mean slower convergence
    - Large steps mean we might step over minima

Simple Example

$$f = x^2 + xy + y^2$$

How does **f(x,y)** change when **x** changes? What about when **y** changes?

$$\frac{\partial f}{\partial x} = 2x + y \qquad \frac{\partial f}{\partial y} = x + 2y$$

$$\begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x + y \\ x + 2y \end{bmatrix}$$

Sanity Check

$$f = x^2 + xy + y^2$$

How does **f(x,y)** change when **x** changes? What about when **y** changes?

$$\frac{\partial f}{\partial x} = 2x + y \qquad \frac{\partial f}{\partial y} = x + 2y$$

$$\begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x + y \\ x + 2y \end{bmatrix}$$

Linear Regression Example

$$SSE = \sum_{i}^{N} (y_i - \hat{y})^2$$

$$\hat{y} = b_0 + b_1 x$$

$$SSE = \sum_{i}^{N} (y_i - (b_0 + b_1 x))^2 = \sum_{i}^{N} (y_i - b_0 - b_1 x)^2$$

Linear Regression Example

$$\nabla_\theta = \begin{bmatrix} \frac{\partial SSE}{\partial b_0} \\ \frac{\partial SSE}{\partial b_1} \end{bmatrix} = \begin{bmatrix} -2\sum_i^N (y_i - (b_0 + b_1 x_i)) \\ -2\sum_i^N x_i (y_i - (b_0 + b_1 x_i)) \end{bmatrix}$$

Linear Regression Example

Let's initialize our gradient descent algorithm to start at (0,0)*

$$\begin{bmatrix} -2\sum_i^N (y_i - (0 + 0x_i)) \\ -2\sum_i^N x_i(y_i - (0 + 0x_i)) \end{bmatrix} = \begin{bmatrix} -2\sum_i^N (y_i) \\ -2\sum_i^N x_i(y_i) \end{bmatrix}$$

This makes sense, as we're guessing 0 for every point when b0 and b1 are both 0

$$\hat{y} = 0x_i + 0 = 0$$

* just because it makes our math easier

Linear Regression Example

Notice that our gradient overall is the sum of the gradients at each of our data points

Let's pretend we have two points: (1,1) and (2,3)

$$\begin{bmatrix} -2\sum_i^N (y_i) \\ -2\sum_i^N x_i(y_i) \end{bmatrix} = \begin{bmatrix} -2(1+3) \\ -2\sum_i^N (1*1+2*3) \end{bmatrix} = \begin{bmatrix} -8 \\ -14 \end{bmatrix}$$

This shows us what changes to make to b0 and b1 in order to reduce our loss function. (Notice we should change b1 MORE than b0.

Linear Regression Example

So let's make those changes!

$$\begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - 0.01 \begin{bmatrix} -8 \\ -14 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 0.01 \begin{bmatrix} 8 \\ 14 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} + 0.01 \begin{bmatrix} 8 \\ 14 \end{bmatrix} = \begin{bmatrix} 0.08 \\ 0.14 \end{bmatrix}$$

Sanity Check

$$SSE = (1 - (0 + 0 * 1))^2 + (3 - 0 - 0 * 2)^2 = 1 + 9 = 10$$

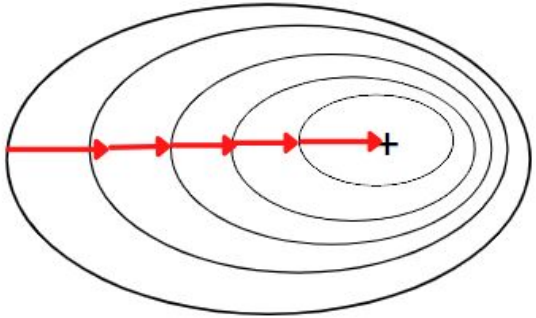
$$SSE = (1 - (0.08 + 0.14 * 1))^2 + (3 - 0.8 - 0.14 * 2)^2 = 0.6084 + 3.6864 = 4.2948$$

Linear Regression Example

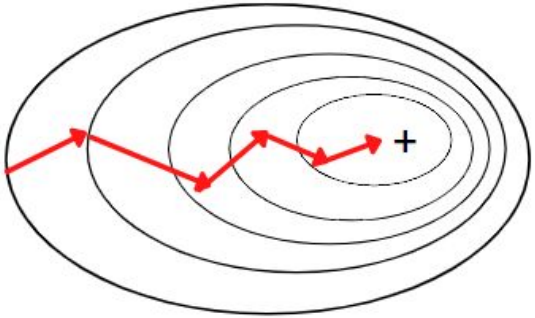
Code Example

Flavors of GD

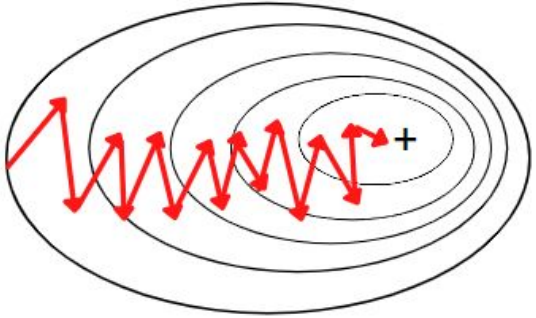
Batch Gradient Descent



Mini-Batch Gradient Descent



Stochastic Gradient Descent



Gradient Descent Ideas

- The Gradient uses partial derivatives to tell us how **sensitive our loss function is to changes** in each of our model's parameters
- To **update our parameters**, we take our old parameters and subtract (learning rate * gradient) $w_t = w_{t-1} - \alpha * g_t$
- **Large Learning Rates** cause us to bounce around and skip minima; **Small Learning Rates** make us SLOW
- **Stochastic** (1-data point) and **Mini-Batch** (a few data points) Gradient Descent allows us to calculate our gradient using fewer points, which is **faster**!

Pandas and sklearn