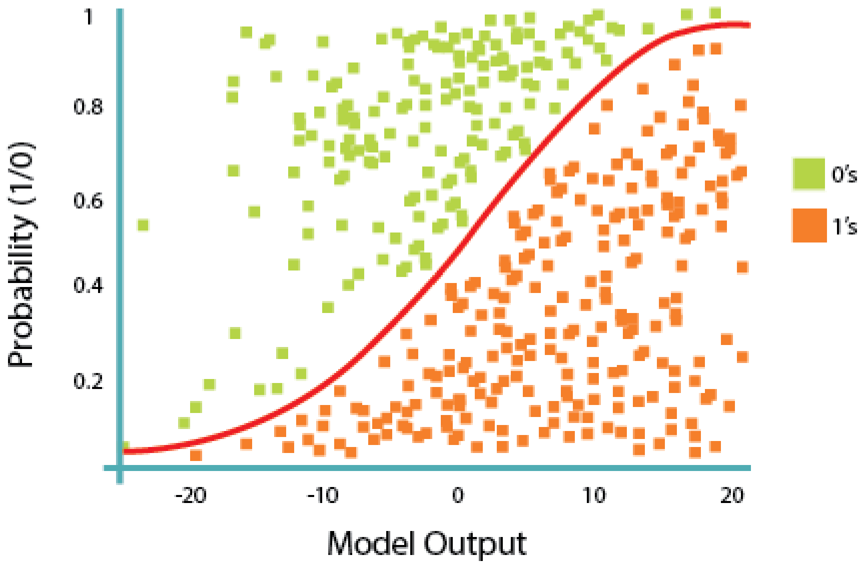
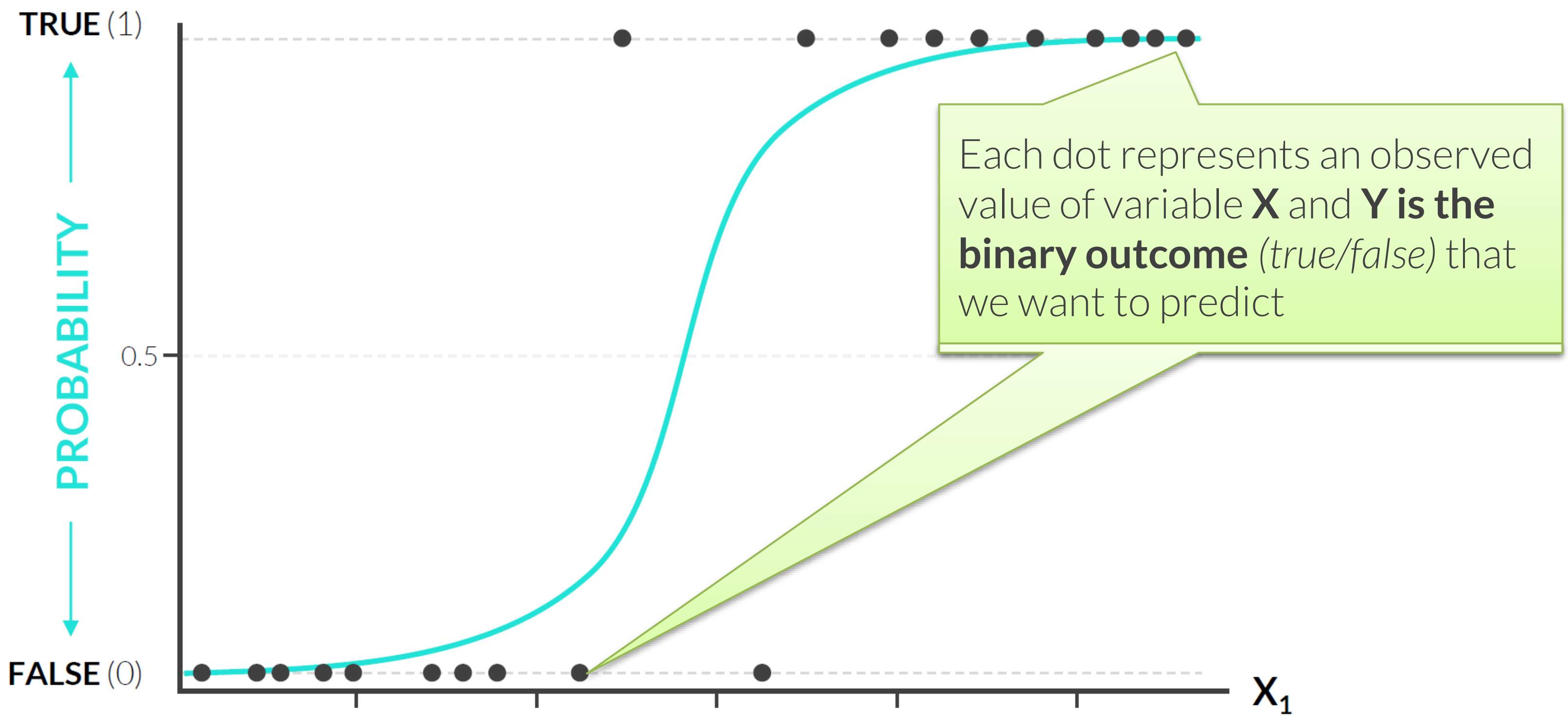
Logistic Regression



Logistic Regression

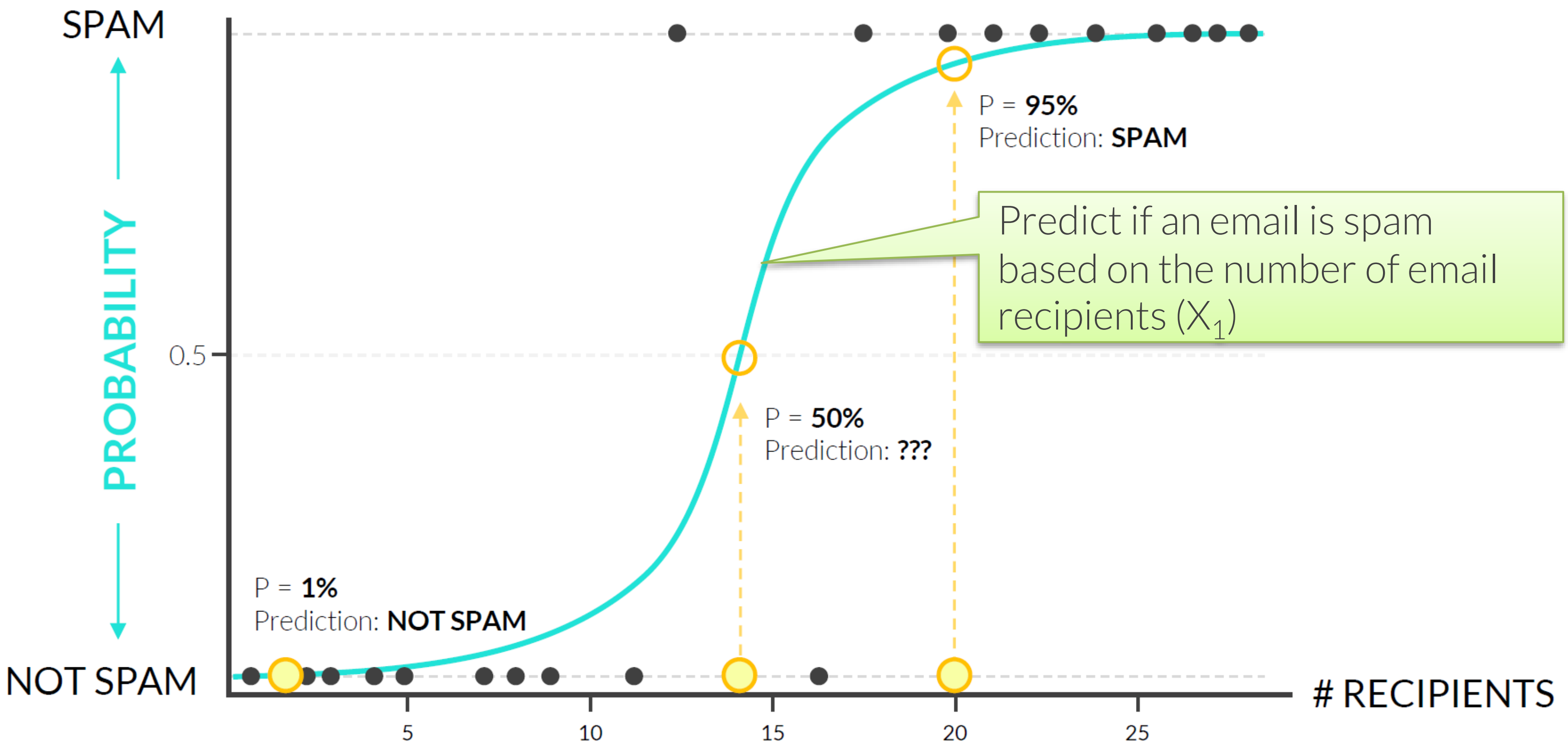
- Logistic Regression is a **classification** technique used to predict the probability of a binary (true/false) outcome
  - Although it has the word "regression" in its name, logistic regression is not used for predicting numeric variables
- Example use cases:
  - Classifying spam emails or fraudulent credit card transactions
  - Determining whether to serve a particular ad to a website visitor
- Logistic regression forms an S-shaped curve between 0 and 1 which represents the probability of a TRUE outcome for any given value of X
- The **cost function** is used to measure how accurately a model predicts outcomes, and is used to optimize the "shape" of the curve

Logistic Regression



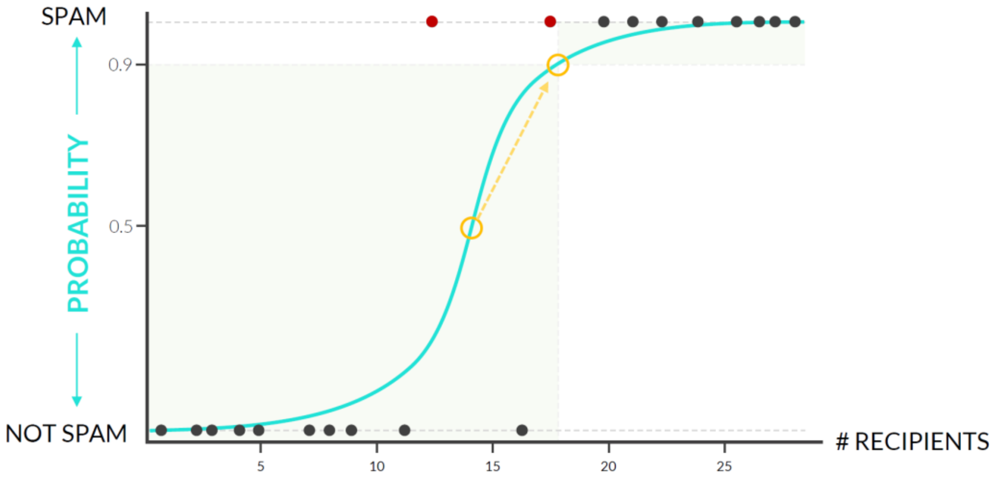
- Logistic regression plots the **best-fitting curve between 0 and 1**, which tells us the probability of Y being TRUE for any given value of X_1

Logistic Regression - Example



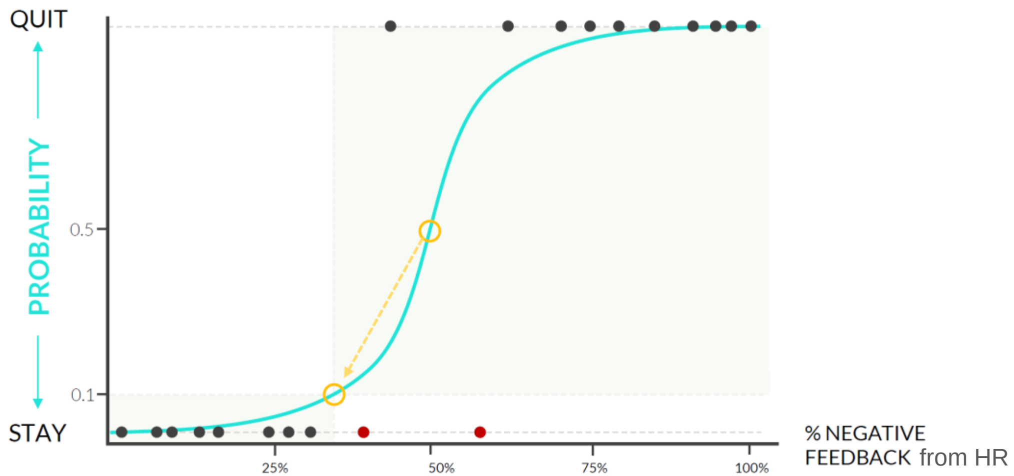
- Using this model, we can classify unobserved values of $\mathbf{X_1}$ (number of recipients) to predict the probability that **Y** is true or false *(i.e., the probability that an email is spam)*
- *In practice a threshold of 0.5 is a common a decision point for logistic models (P > 0.5 means Y is predicted to be True)*

Is 50% always the right decision point for logistic regression models?



- When the cost of a **false positive** (incorrectly predicting a TRUE outcome) is high we may increase the threshold (e.g., 90%), to avoid classifying legit emails as span

(Incorrectly mark few spam emails as "not spam" is not a big deal)



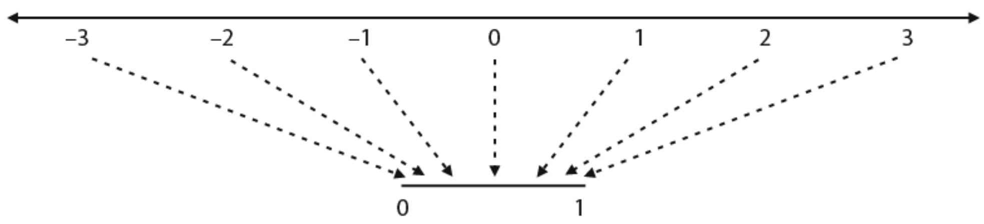
- When the risk of a false negative (incorrectly predicting an employee will stay) is high we may decrease the threshold (e.g., 10%), to correctly predict more cases where someone is likely to quit

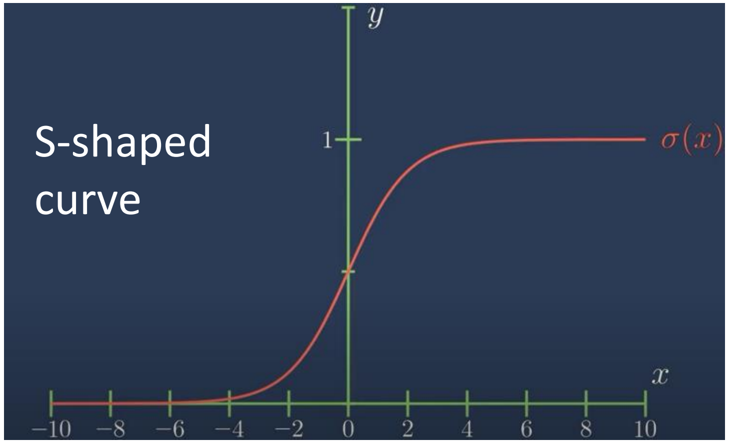
(It's easier to train and retain an employee than hire a new one, so the cost of a false negative -incorrectly predicting an employee will stay- is high)

sigmoid function

- The sigmoid function, also known as the logistic function, is a function that maps any real number into a range between 0 and 1
- The sigmoid function, denoted with the Greek letter sigma (σ), is defined as:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$





Logistic Regression - Model Representation

- Given a training set, learn a function **f** so that **f(x)** is a "good" predictor for the corresponding value of y
  - Learn the weights (w) and bias (b) given inputs (x)

$$z = \left(\sum_{i=1}^{n} w_i x_i \right) + b \quad \xrightarrow{\text{Vector Notation}} \quad z = \mathbf{w} \cdot \mathbf{x} + b$$

W and X are vectors

- Furthermore, to get your prediction, you must apply the **sigmoid** function

$$\boldsymbol{f}(\boldsymbol{x}) = \hat{y} = \sigma(z) = \frac{1}{1+e^{-z}}$$

Learning a Logistic Regression Model

- Learn $\boldsymbol{w} = [\mathrm{w}_{\mathbf{1}}, \dots, \mathrm{w}_{\boldsymbol{m}}]$ and $\boldsymbol{b}$ by minimizing the following cost function:

$$J = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)})]$$

  - $y^{(i)}$ is the actual label (0 or 1) for the ith training example.
  - $\hat{y}^{(i)}$ is the predicted probability that the ith example belongs to class 1, given input $x^{(i)}$.

- The cost function (also known as a loss function) used is Binary cross entropy
  - It measures the difference between the predicted probabilities $\hat{\mathbf{y}}$ and the actual binary labels $\mathbf{y}$
  - This cost function essentially penalizes the model for predicting probabilities far from the actual labels
  - If the actual label is 1, it penalizes the model more for predicting a probability close to 0 (as given by $\log(1 - \hat{y}^{(i)})$ term), and vice versa

Gradient descent algorithm

Want to find $\mathbf{w}$ and b that minimize the cost function $\mathbf{J}$ $\quad \underset{w,b}{\text{minimize}}\, \boldsymbol{J}(\boldsymbol{w}, \boldsymbol{b})$

1. Initialize the values of $\mathbf{w}$ and b to some arbitrary values (say 0, 0)
2. Calculate the predicted values of ***y*** using the current values of $\mathbf{w}$ and b
3. Calculate the gradients of the cost function with respect to $\mathbf{w}$ and b
4. Update the values of $\mathbf{w}$ and b using the gradients and a learning rate

Learning Rate

Derivative of the Cost Function w.r.t $\mathbf{w}$

$$w = w - \alpha \boxed{\frac{\partial}{\partial w} J(w,b)}$$

$$b = b - \alpha \frac{\partial}{\partial b} J(w,b)$$

5. Repeat steps 2-4 until convergence (i.e., until the cost function converges to a minimum)

Gradient descent algorithm

Gradient descent utilizes the partial derivative of the cost function with respect to **w** and $\boldsymbol{b}$ to update **w** and $\boldsymbol{b}$ parameters

Repeat until convergence {

$$\boldsymbol{w} = \boldsymbol{w} - \boldsymbol{\alpha} \underbrace{\frac{\mathbf{1}}{\boldsymbol{m}} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} \left(\hat{\mathbf{y}}^{(\boldsymbol{i})} - \boldsymbol{y}^{(\boldsymbol{i})}\right) . \, \boldsymbol{x}^{(\boldsymbol{i})}}_{\frac{\partial J}{\partial w}}$$

Learing rate α*, controls how big a step we take when we update* $\boldsymbol{w}$ *and* $\boldsymbol{b}$

$$\boldsymbol{b} = \boldsymbol{b} - \boldsymbol{\alpha} \underbrace{\frac{\mathbf{1}}{\boldsymbol{m}} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} \left(\hat{\mathbf{y}}^{(\boldsymbol{i})} - \boldsymbol{y}^{(\boldsymbol{i})}\right)}_{\frac{\partial J}{\partial b}}$$

}

(simultaneously update $\boldsymbol{w}$ and $\boldsymbol{b}$)

`09.logistic-regression\`
`2_logistic-regression.ipynb`