Data Science

Lecture 2-2: Data Science Fundamentals (Modeling)

UNIVERSITY OF AMSTERDAM

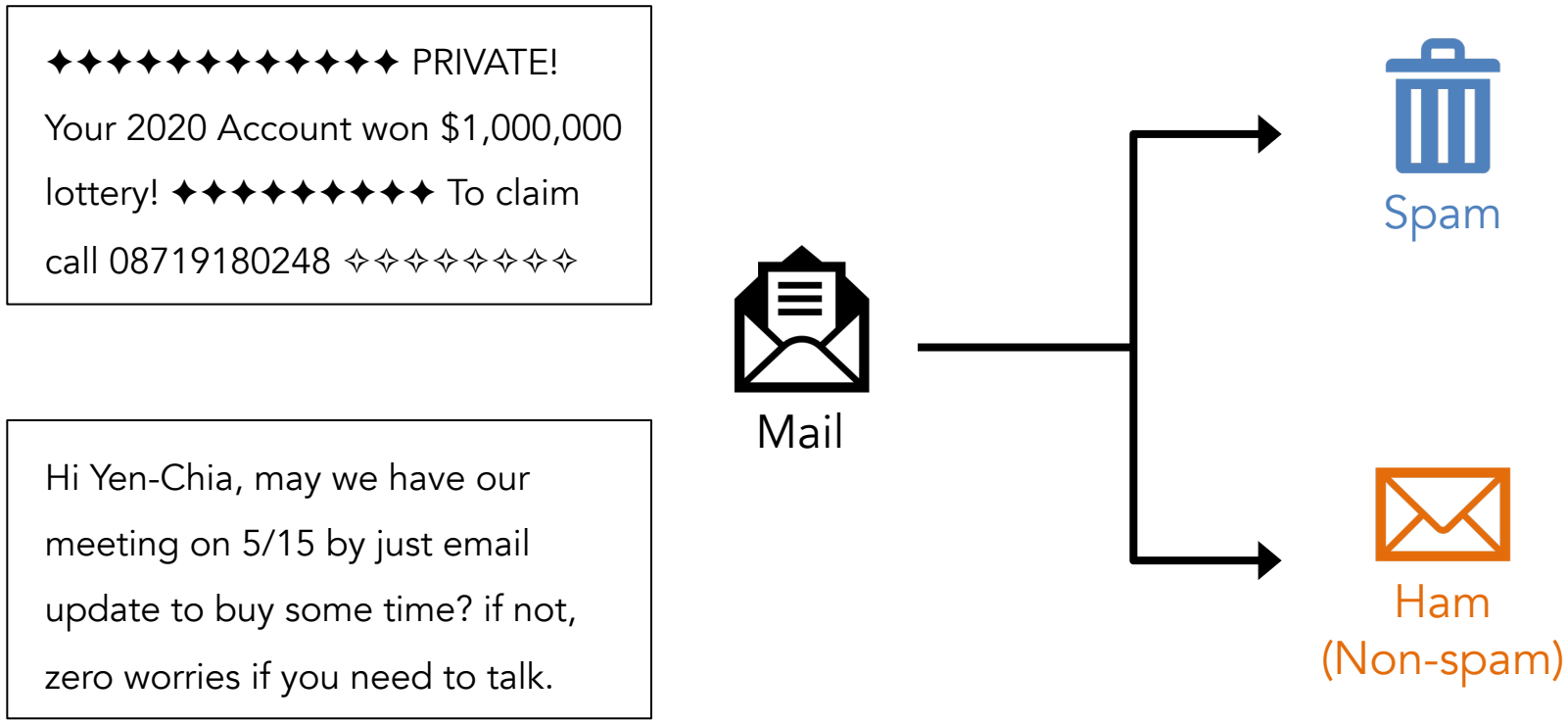
Lecturer: Yen-Chia Hsu

Date: Feb 2023

This lecture recaps classification and regression techniques for modeling data.

Classification

For this lecture, let us now use the following text classification task as an example: identifying whether a text message is spam or ham (non-spam).



Classification

To classify spam messages, we need examples: a dataset with observations (messages) and labels (spam or non-spam).

Observations	Labels
✦✦✦✦✦✦✦✦✦✦✦✦ PRIVATE! Your 2020 Account won $1,000,000 lottery! ✦✦✦✦✦✦✦✦✦ To claim call 08719180248 ✧✧✧✧✧✧✧✧✧	 Spam
Hi Yen-Chia, may we have our meeting on 5/15 by just email update to buy some time? if not, zero worries if you need to talk.	 Ham
Would you be willing to meet with me on 3/26 Thursday when I was in TU Delft after (or before) giving the guest lecture (10:35am-11:50am)?	 Ham
⋮	⋮

Classification

We can extract features (information) using human knowledge, which can help distinguish spam and ham messages.

✦✦✦✦✦✦✦✦✦✦✦✦ PRIVATE! Your 2020 Account won $1,000,000 lottery! ✦✦✦✦✦✦✦✦✦ To claim call 08719180248 ✧✧✧✧✧✧✧✧✧

Number of special characters = 34

Number of digits = 22

Spam

Hi Yen-Chia, may we have our meeting on 5/15 by just email update to buy some time? if not, zero worries if you need to talk.

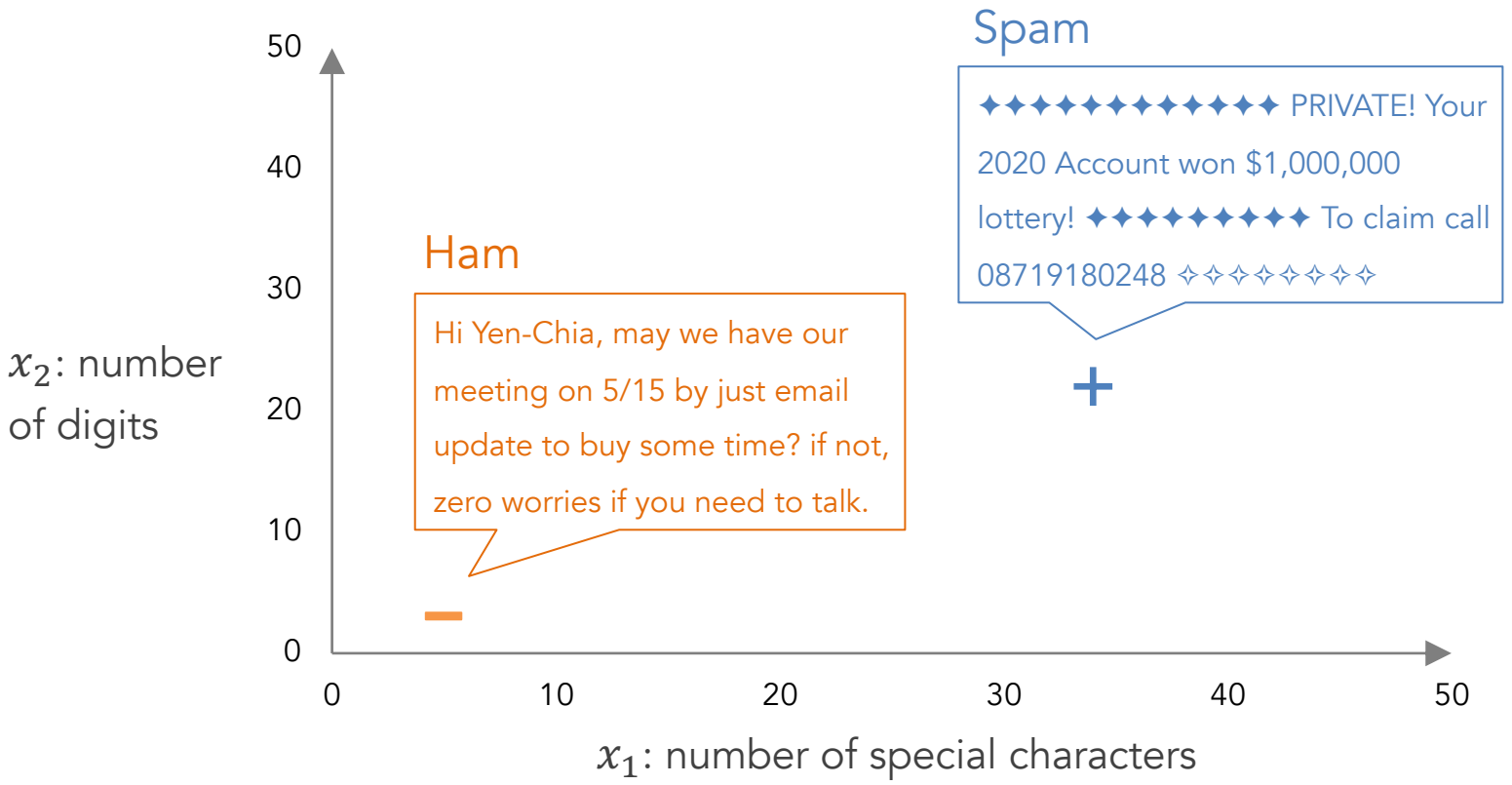
Number of special characters = 5

Number of digits = 3

Ham

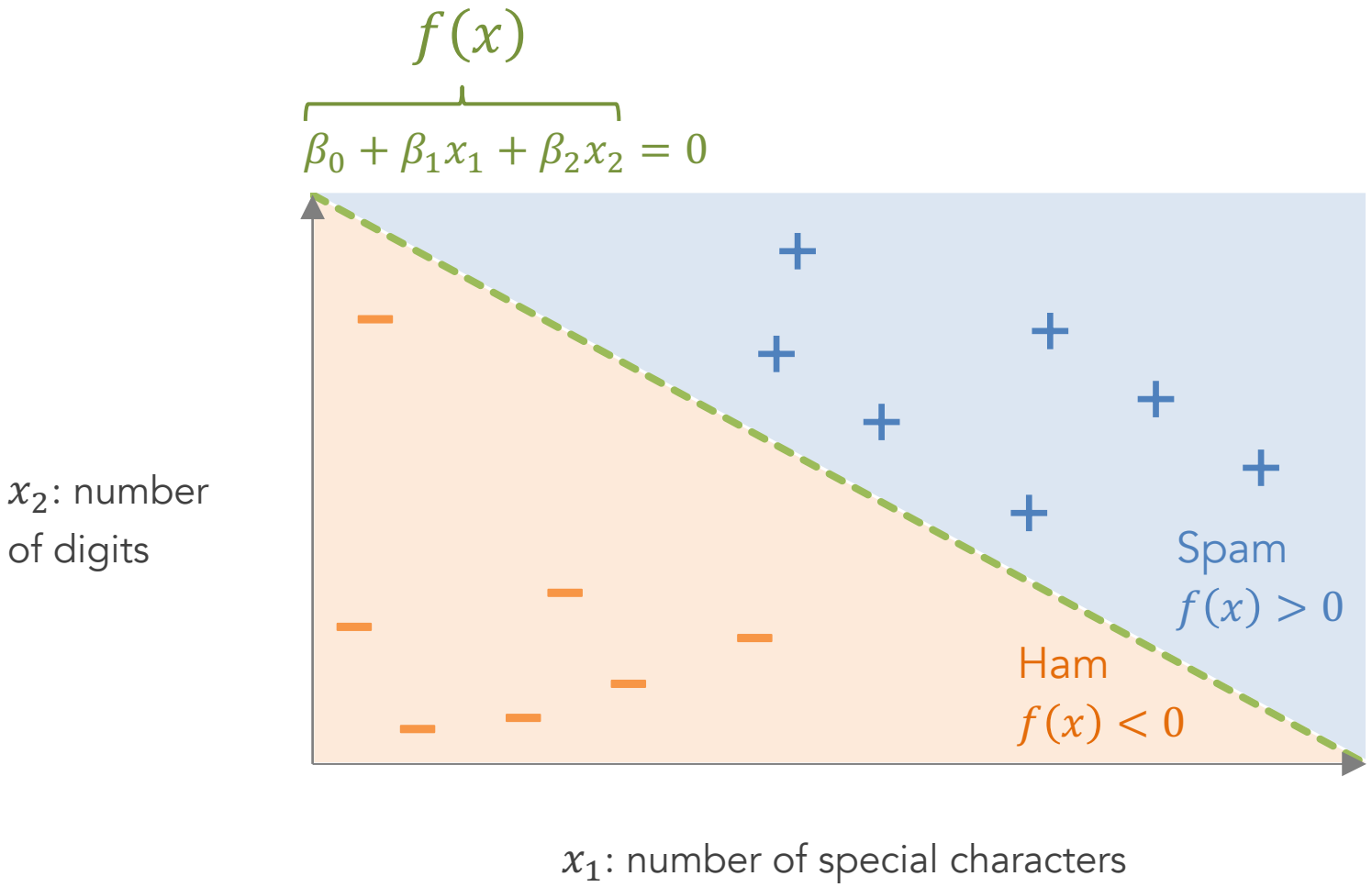
Classification

Using features x (which contains x_1 and x_2), we can represent each message as one data point on an p-dimensional space ($p = 2$ in this case).



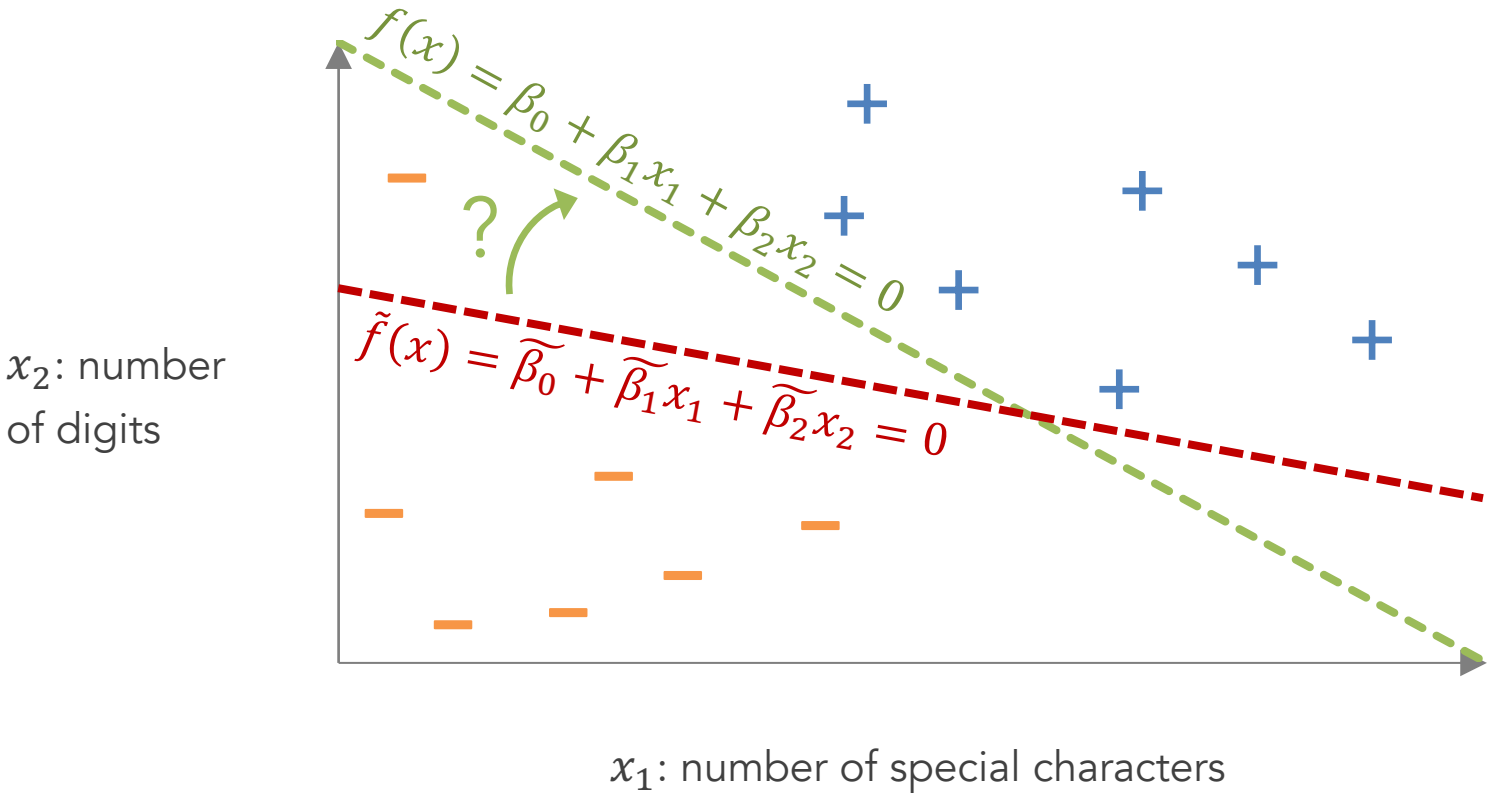
Classification

We can think of the model as a function f that can separate the observations into groups (i.e., class labels y) according to their features $x = \{x_1, x_2\}$.



Classification

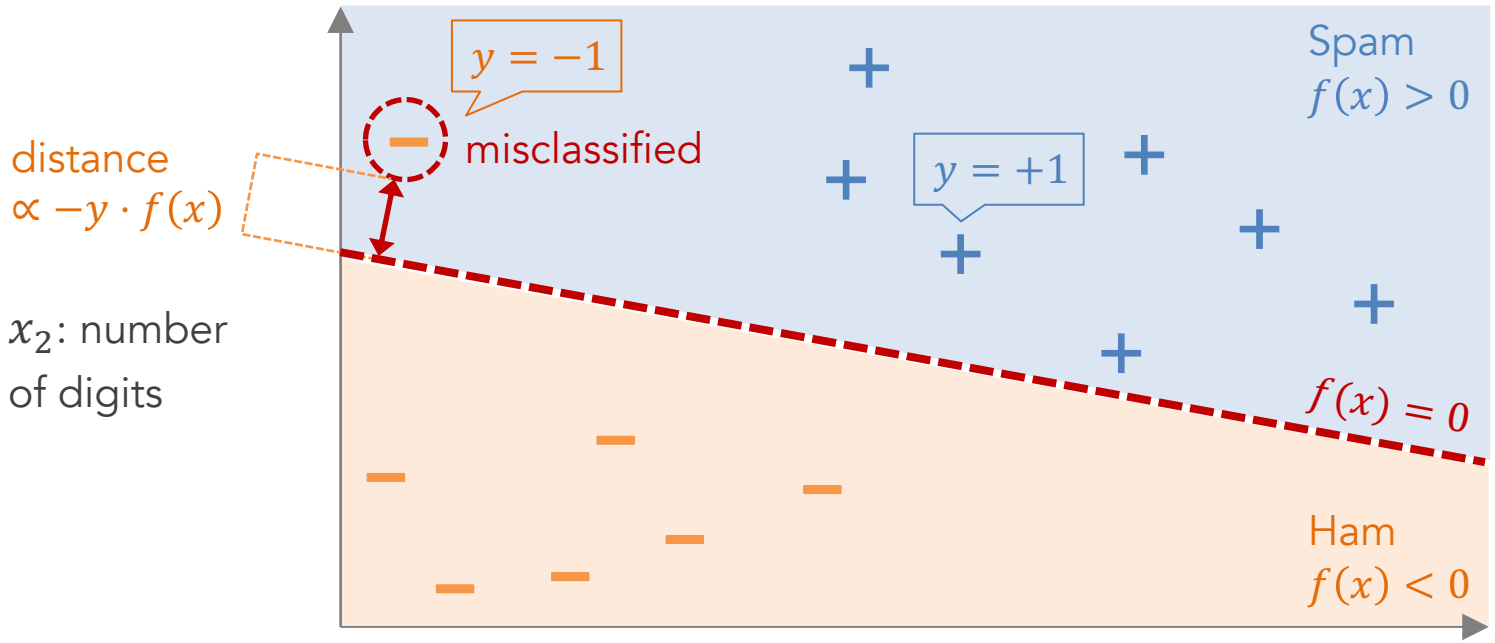
To find a good function f, we start from some f and train it until satisfied. We need something to tell us which direction and magnitude to update.



Classification

First, we need an error metric (i.e., loss function). For example, we can use the sum of distances between the misclassified points and line f.

$$\text{error} = \sum -y \cdot f(x) \quad \text{for each misclassified point } x = \{x_1, x_2\}$$

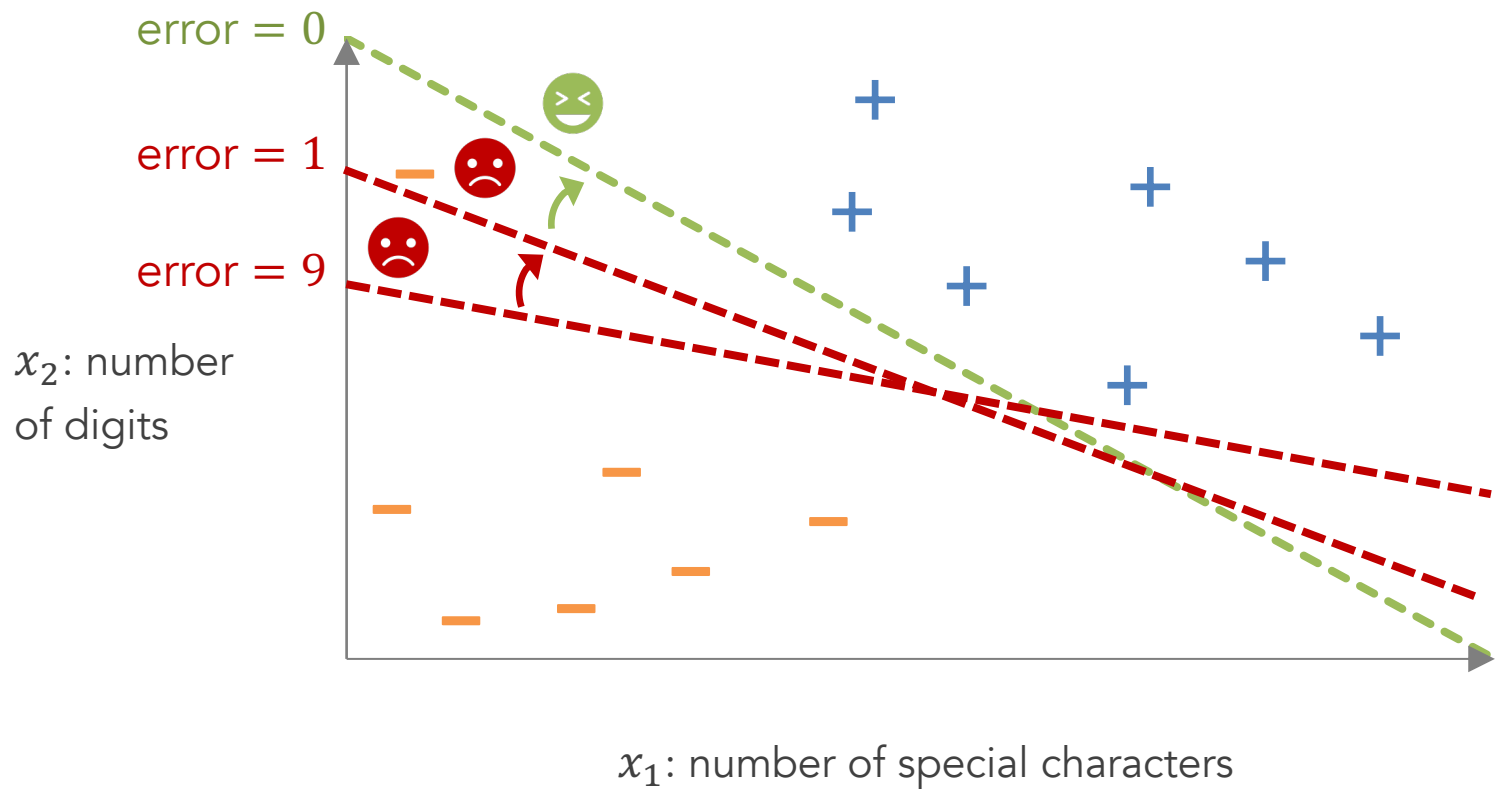


Distance from point to plane: https://mathinsight.org/distance_point_plane

Classification

We can use gradient descent (an optimization algorithm) to minimize the error to train the model f iteratively. This example is the Perceptron algorithm.

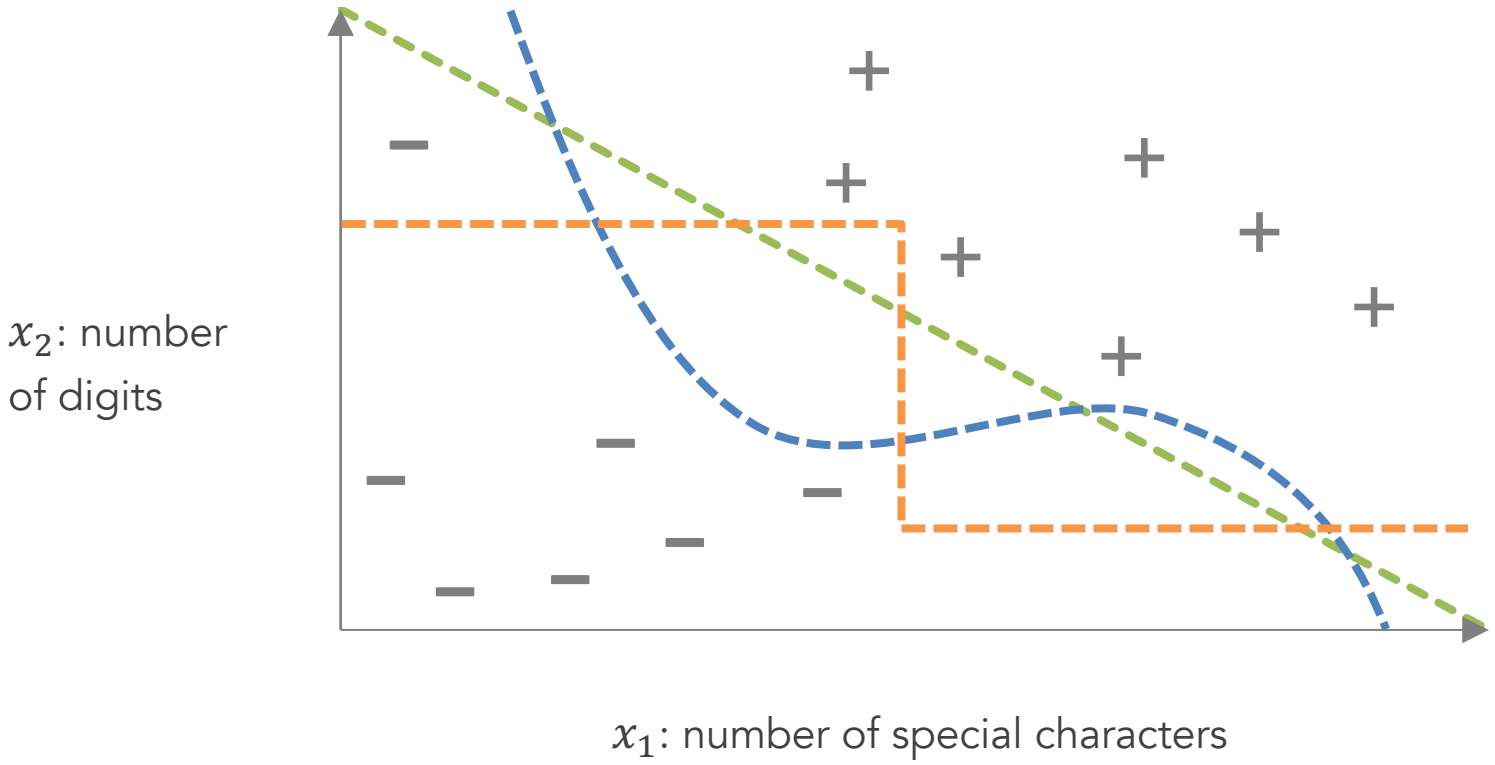
minimize error = $\sum -y \cdot f(x)$ for each misclassified point $x = \{x_1, x_2\}$



Rosenblatt's Perceptron Learning Algorithm: section 4.5.1 in book https://hastie.su.domains/ElemStatLearn/

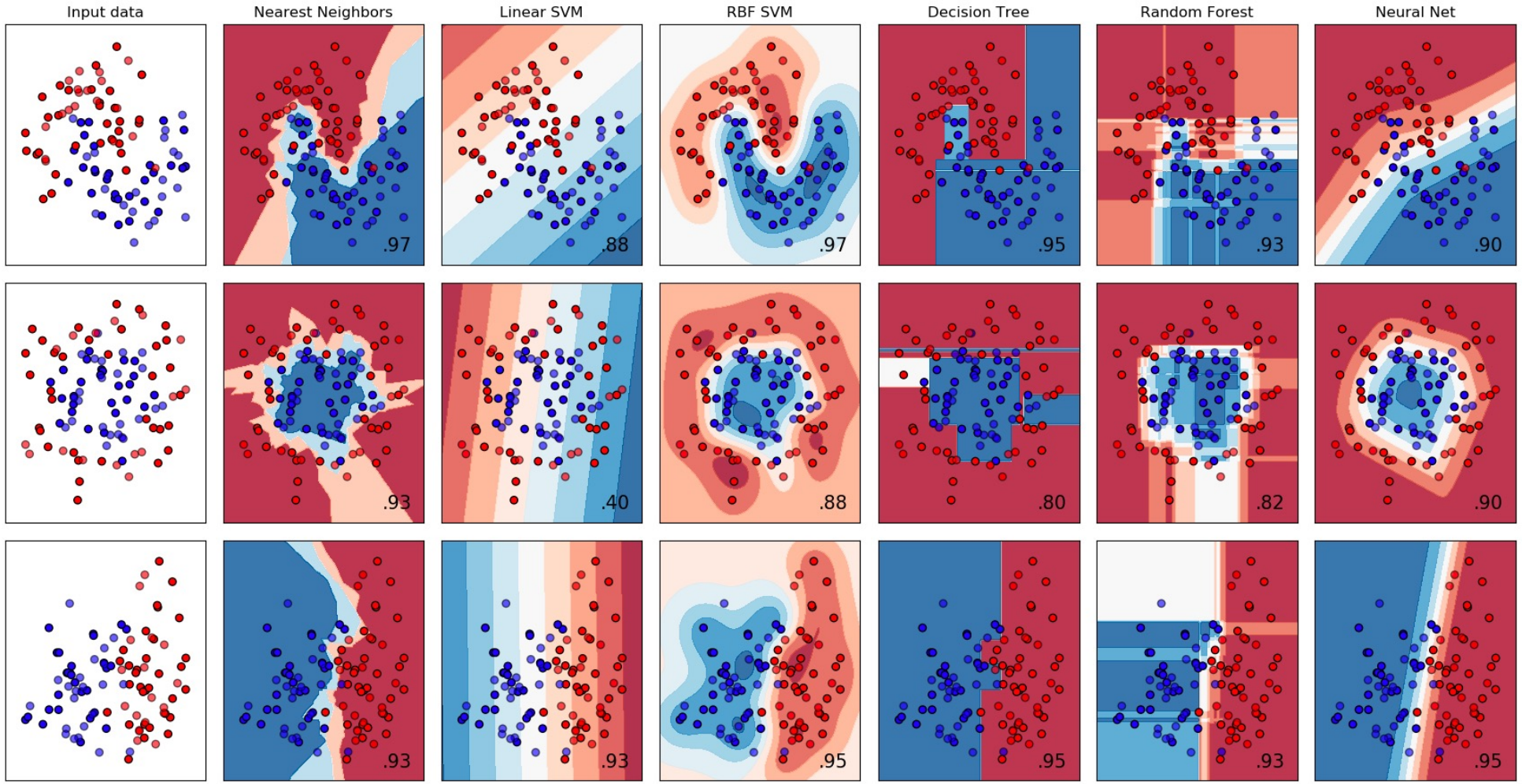
Classification

Depending on the needs, we can train different models (using different loss functions) with various shapes of decision boundaries.



Classification

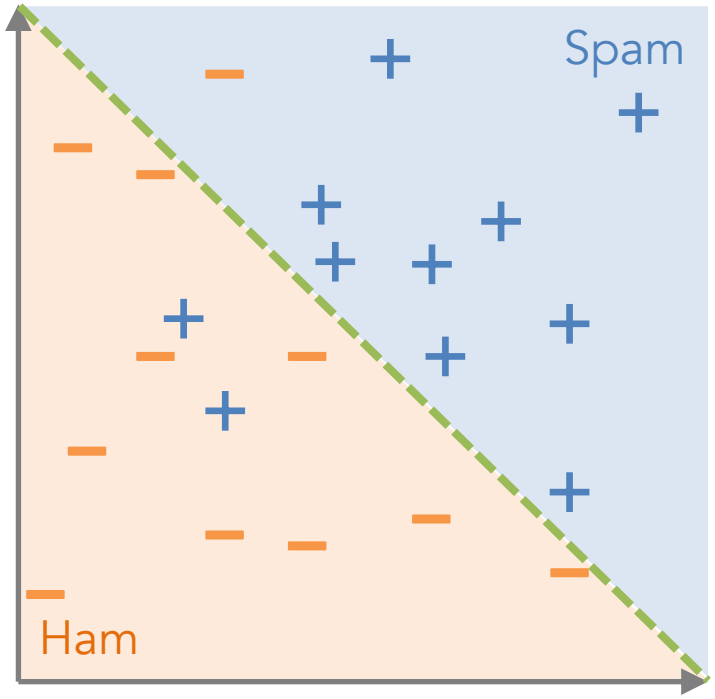
Depending on the needs, we can train different models (using different loss functions) with various shapes of decision boundaries.



Retrieved from https://scikit-learn.org/stable/auto_examples/#classification

Classification

To evaluate our classification model, we need to compute evaluation metrics to measure and quantify model performance, such as the accuracy of all data.



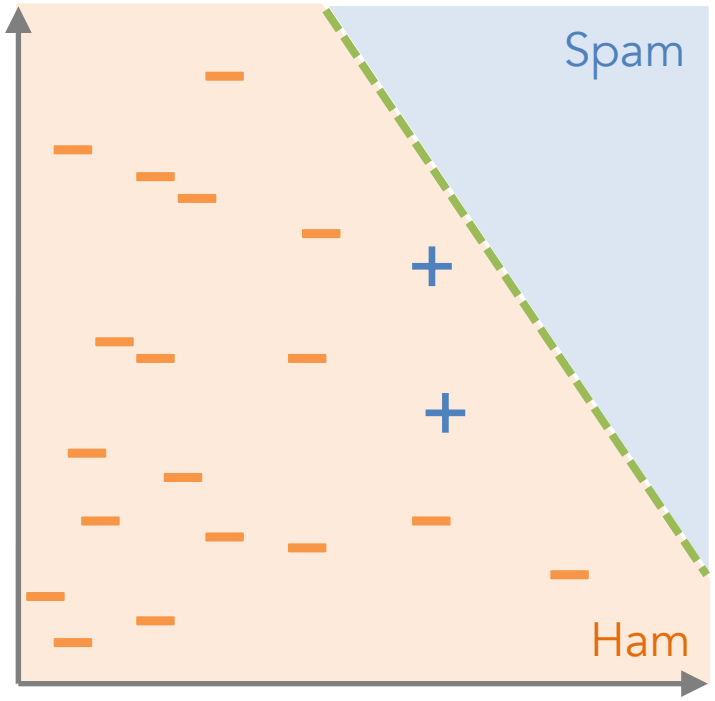
Accuracy for all data

$$= \frac{\#\text{ of correctly classified points}}{\#\text{ of all points}}$$

$$= \frac{19}{22} = 0.86$$

Classification

But what if the dataset is imbalanced (i.e., some classes have far less data)? In this case, the accuracy of all data is a bad evaluation metric.



Classify all data as non-spam

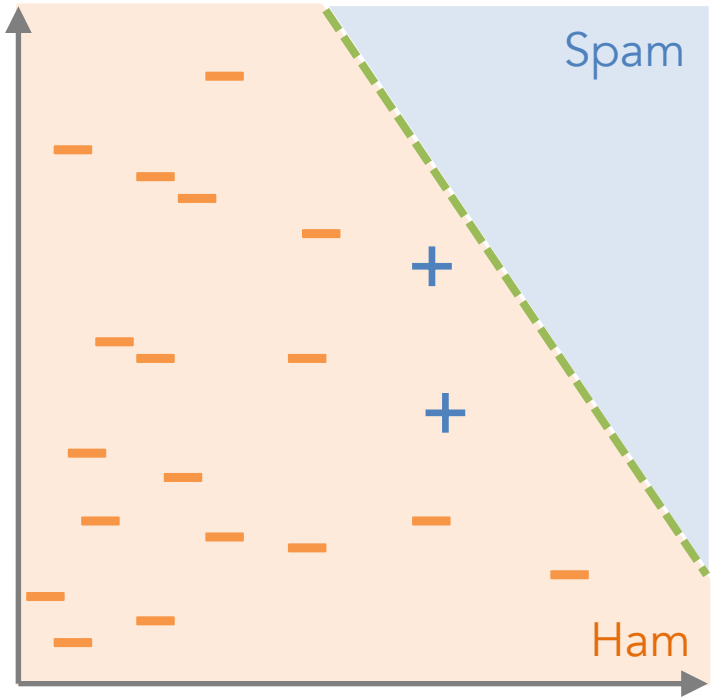
$$\text{Accuracy for all data}$$

$$= \frac{\#\text{ of correctly classified points}}{\#\text{ of all points}}$$

$$= \frac{18}{20} = 0.9$$

Classification

Instead of computing the accuracy for all the data, we can compute accuracy for each class, which allows us to see the performance of different labels.



Classify all data as non-spam

$$\text{Accuracy for spam} = \frac{0}{2} = 0$$

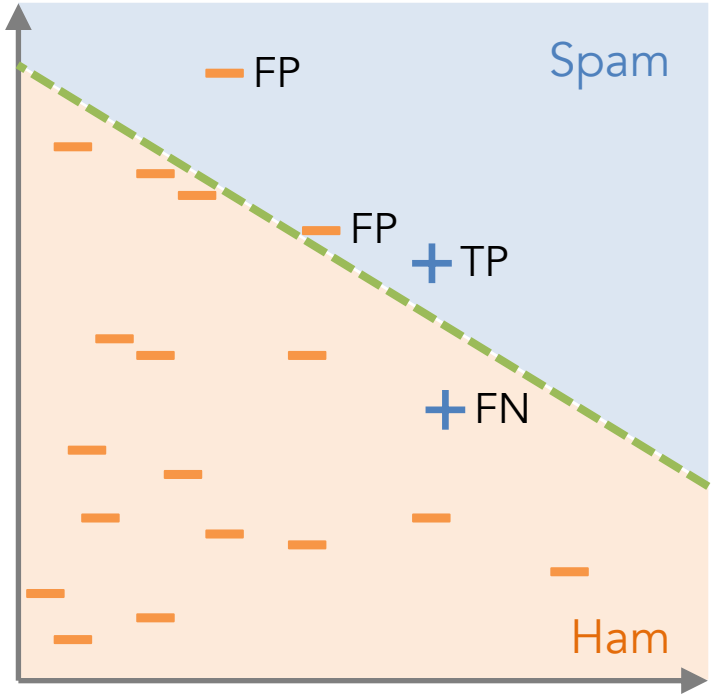
(true positive rate, recall, sensitivity)

$$\text{Accuracy for ham} = \frac{18}{18} = 1$$

(true negative rate, specificity)

Classification

If we care more about the positive class (e.g., spam), we can use precision and recall, with its best value at 1 and the worst value at 0.



$\mathrm{TP} = 1$ (True Positive)

$\mathrm{FP} = 2$ (False Positive)

$\mathrm{FN} = 1$ (False Negative)

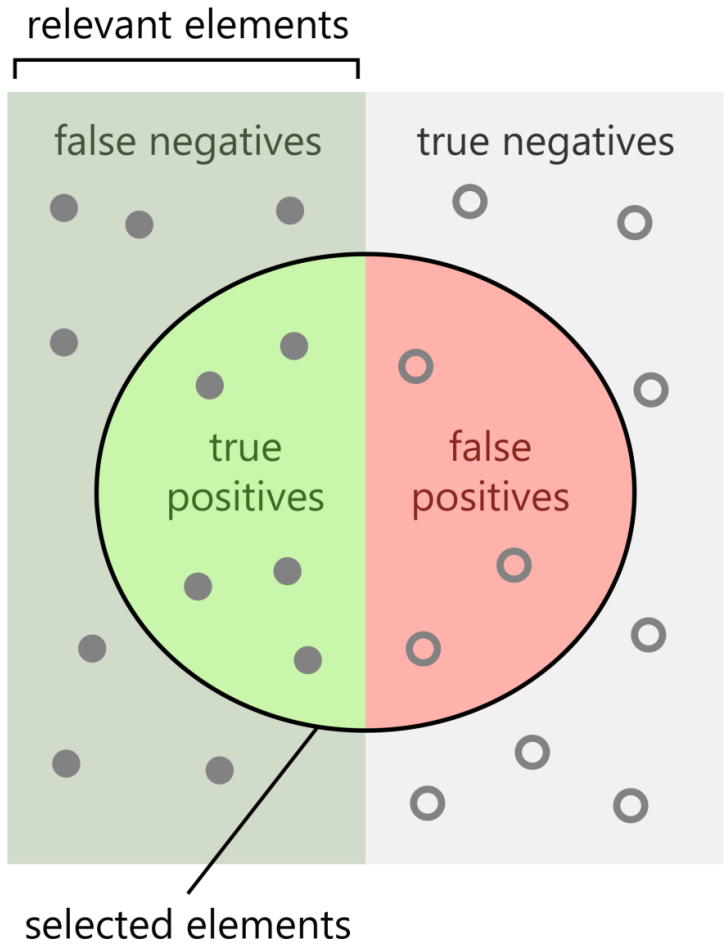
$$\text{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = 0.33$$

$$\text{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = 0.5$$

He and Garcia. 2009. Learning from imbalanced data. IEEE TKDE.

Classification

Precision and recall can be aggregated into F-score as a general model performance, with its best value at 1 and worst value at 0.



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

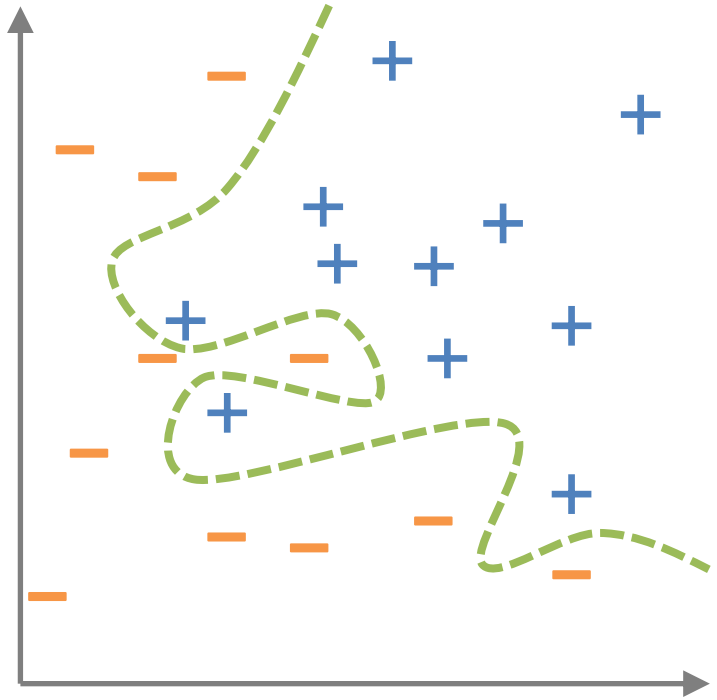
$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

$$\text{F–score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

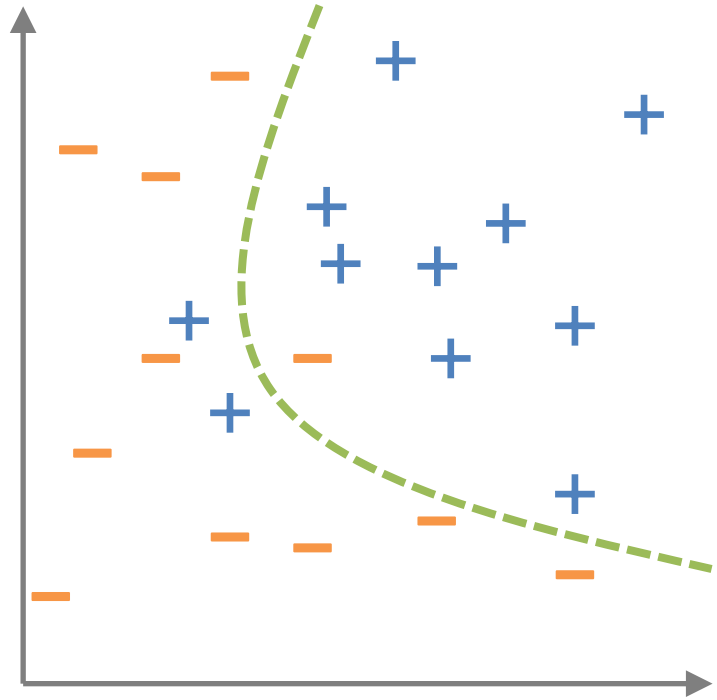
Retrieved from https://en.wikipedia.org/wiki/Precision_and_recall

Classification

We can train different types of models. But how do we know which one is better? Can we just pick an evaluation metric to determine which model is good?



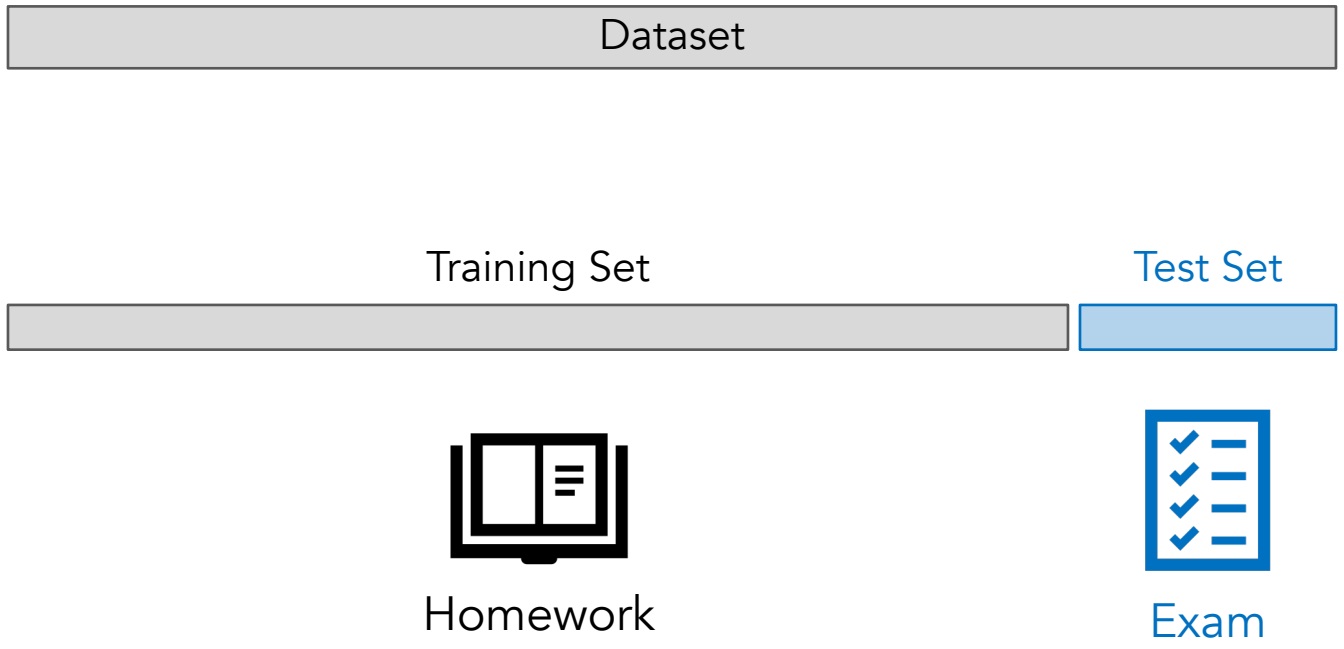
Model A



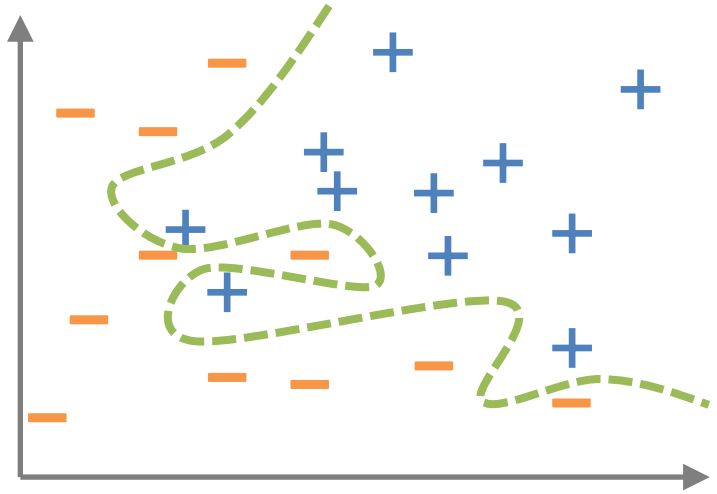
Model B

Classification

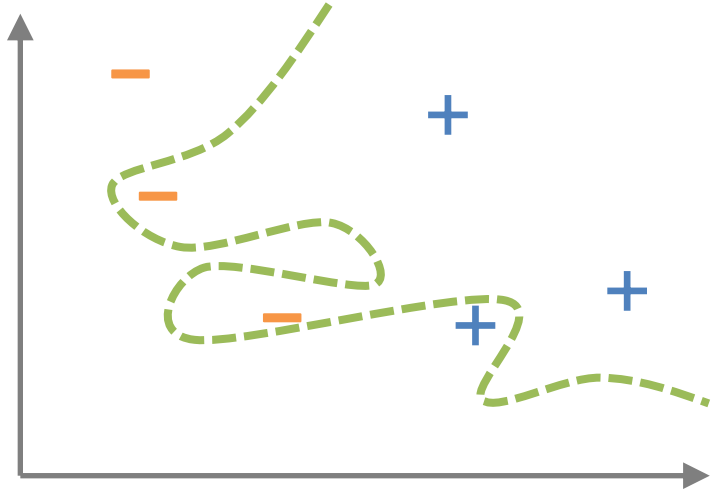
To choose models, we need a test set, which contains data that the models have not yet seen before during the training phase.



Model A

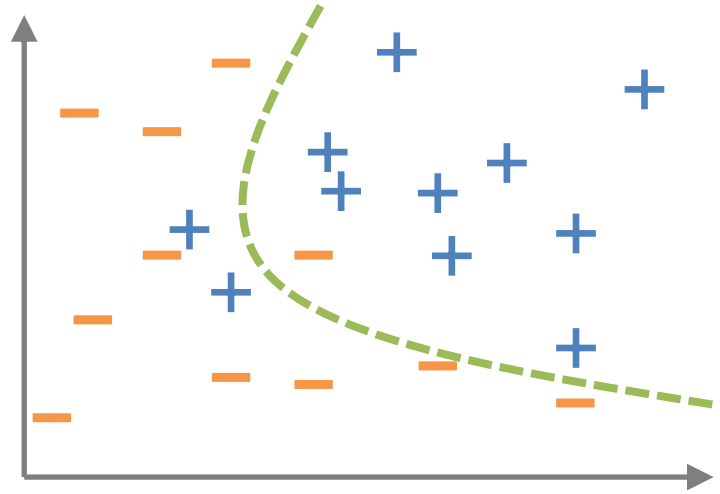


Training Accuracy = 1

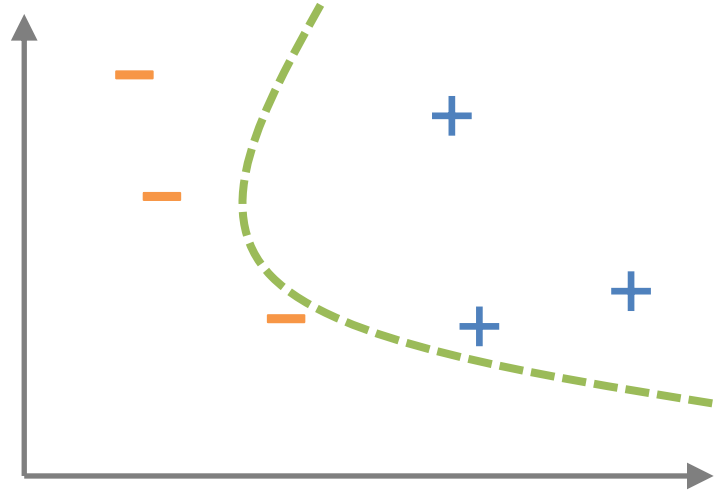


Testing Accuracy = 0.5

Model B



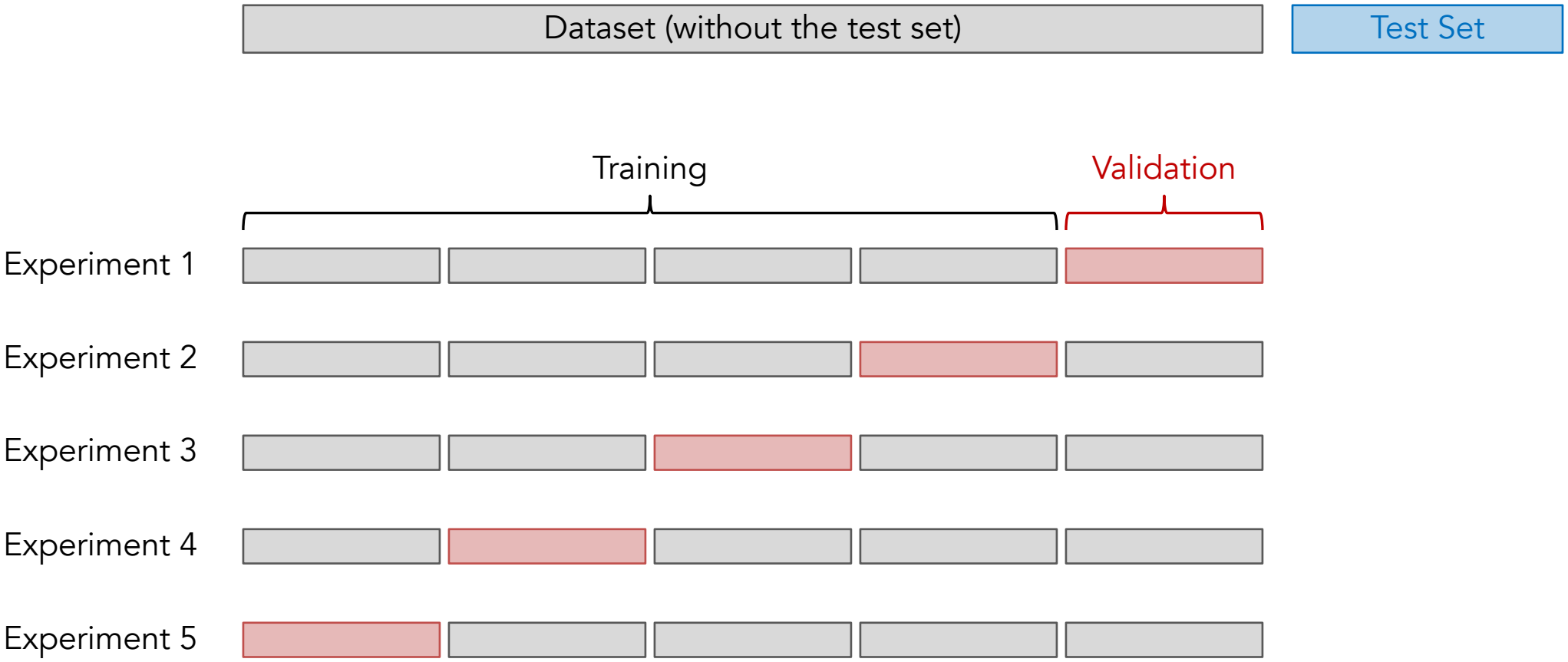
Training Accuracy = 0.86



Testing Accuracy = 1

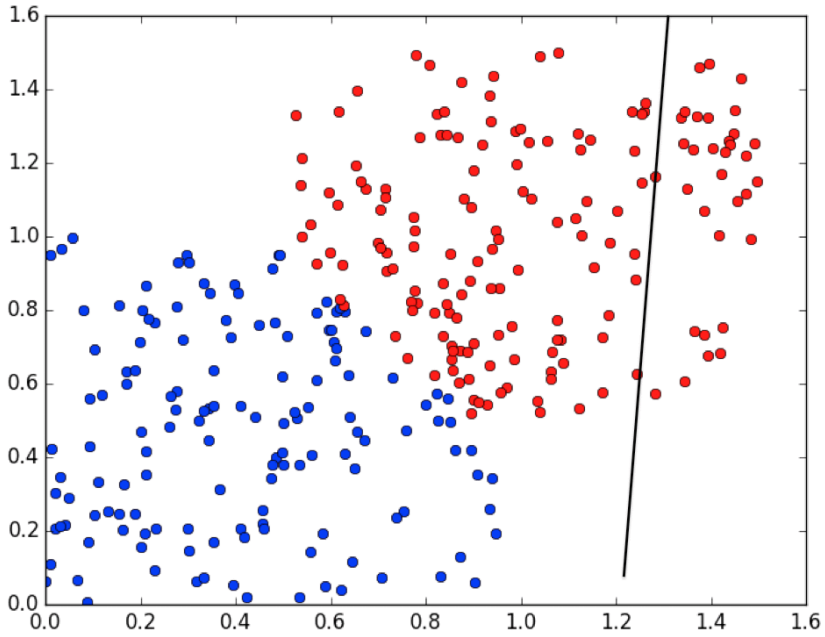
Classification

To tune hyper-parameters for a model, we use cross-validation to divide the dataset into folds and use each fold for validation.

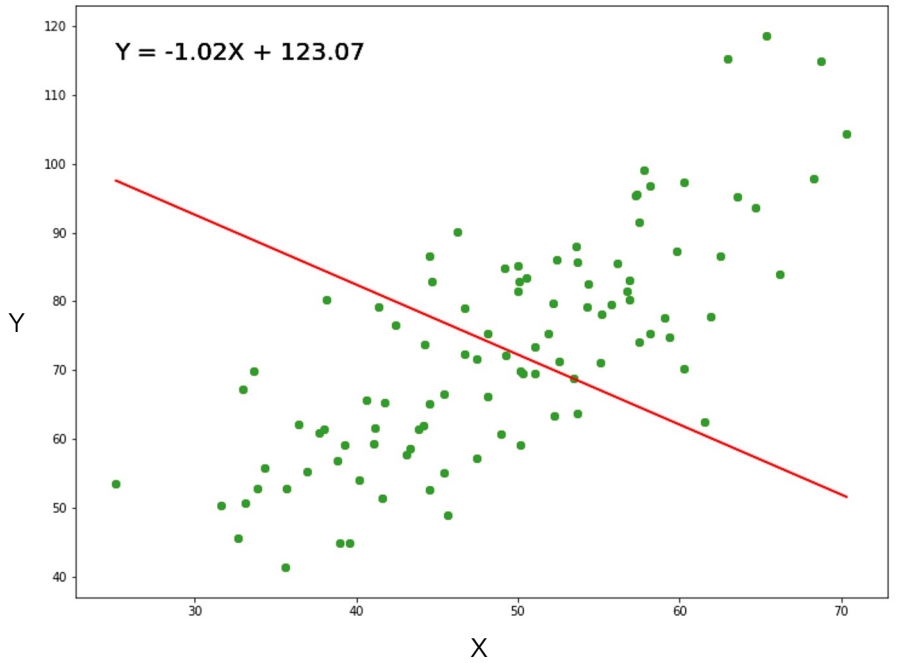


Regression

Unlike classification (which separates data into categories), regression fits a function that maps features x to a continuous variable y (i.e., the response).



- [Classification] How can we fit a function that separates data points into different groups?

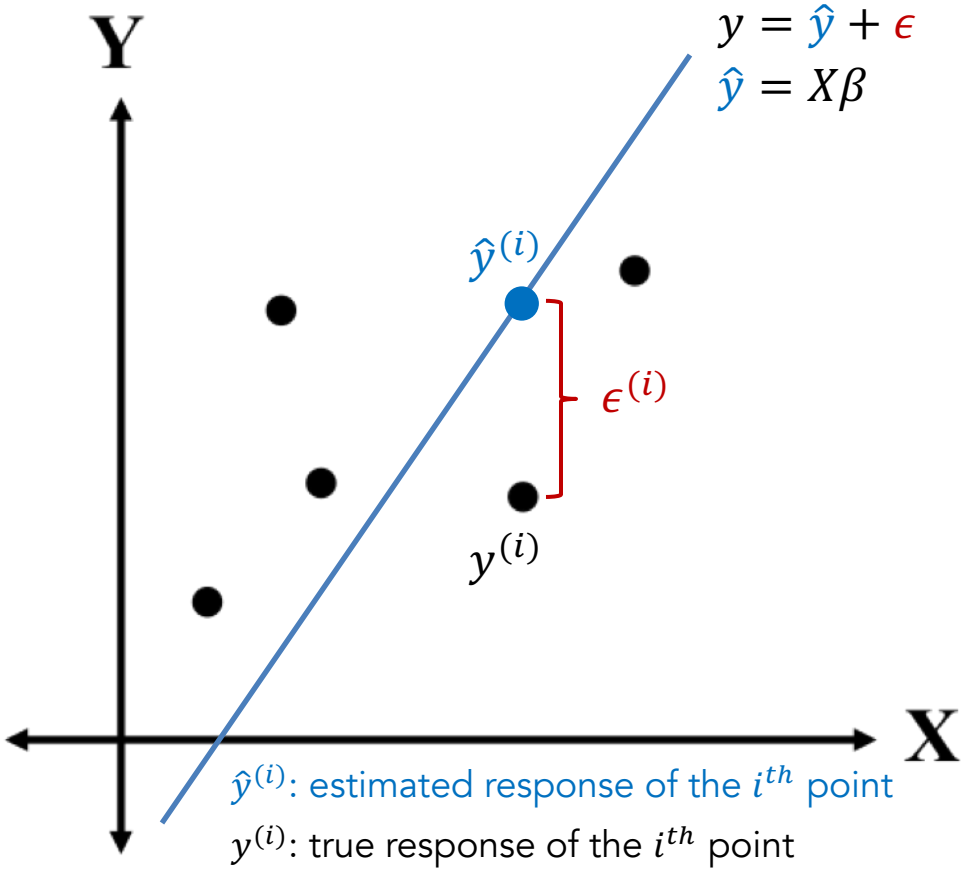


- [Regression] How can we fit a function that maps features (input) to a continuous variable (output)?

Source of Figures -- https://saichandra1199.medium.com/classification-vs-regression-622b83ff8e90

Regression

Linear regression fits a linear function f that maps x to y using some error metric, which can best describe the linear relationship between variables x and y.



$\hat{y}^{(i)}$: estimated response of the i^{th} point

$y^{(i)}$: true response of the i^{th} point

$x^{(i)}$: features of the i^{th} point

y: true response (in vector form)

$\hat{y}$: estimated response (in vector form)

X: predictors/features (in matrix form)

β: coefficient (in vector form)

ϵ: error/noise/residual (in vector form)

$$X = [\mathbf{1} \quad x_1] = \begin{bmatrix} 1 & x_1^{(1)} \\ \vdots & \vdots \\ 1 & x_1^{(n)} \end{bmatrix}$$

β_0: intercept

β_1: slope

x_1: first predictor

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

$$\hat{y} = X\beta = \beta_0 + \beta_1 x_1 = \begin{bmatrix} \hat{y}^{(1)} \\ \vdots \\ \hat{y}^{(n)} \end{bmatrix}$$

We use the vector and matrix forms to simplify equations.

Vector

Matrix

Scalar

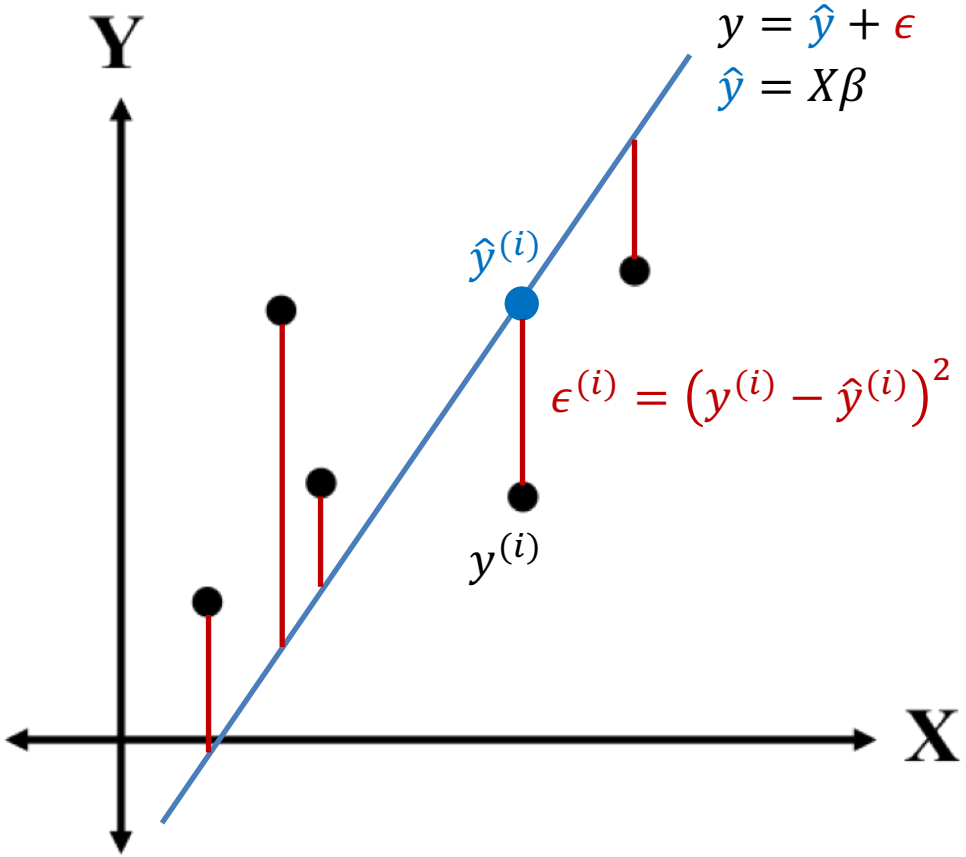
$$X = [\mathbf{1} \quad x_1 \quad \cdots \quad x_p] = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_p^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{(n)} & \cdots & x_p^{(n)} \end{bmatrix} \qquad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$$

Vector

$$\hat{y} = f(x) = X\beta = [\mathbf{1} \quad x_1 \quad \cdots \quad x_p] \cdot \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix} = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p = \begin{bmatrix} \hat{y}^{(1)} \\ \vdots \\ \hat{y}^{(n)} \end{bmatrix}$$

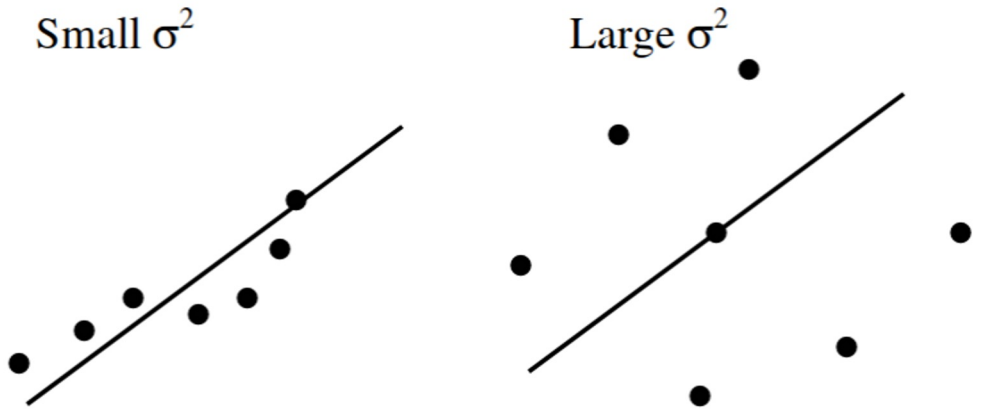
Regression

Usually, we assume that the error ϵ is IID (independent and identically distributed) and follows a normal distribution with zero mean and some variance σ^2.



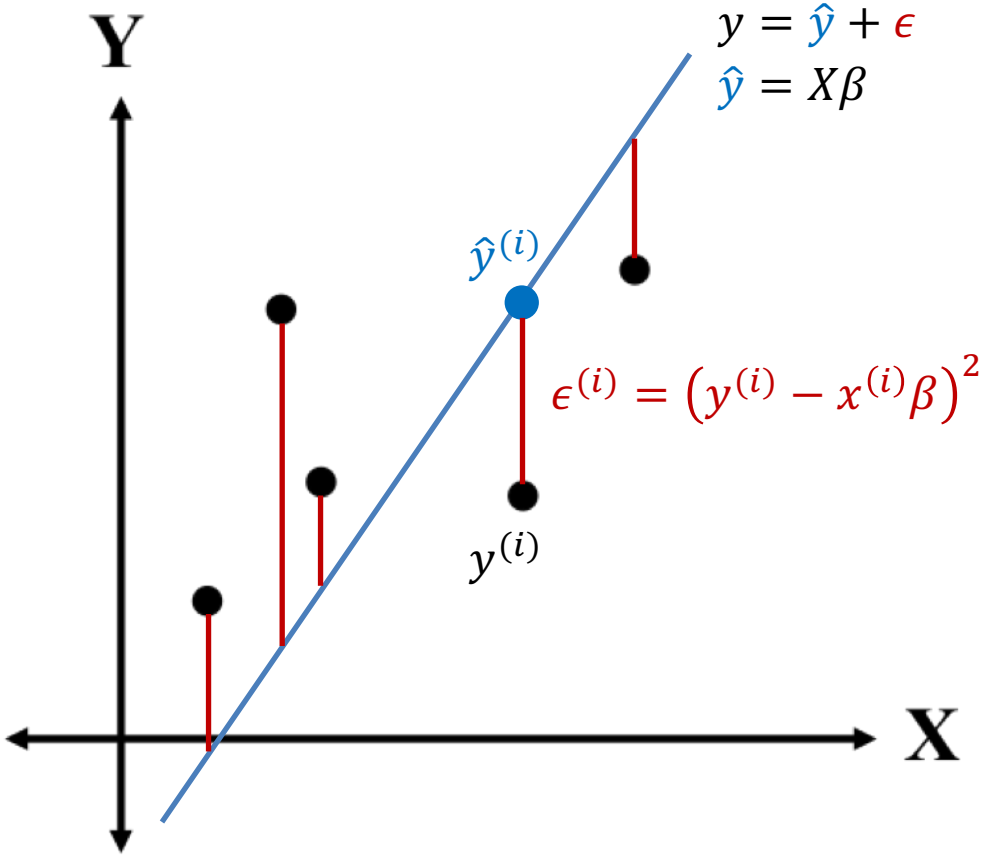
$$\epsilon \sim^{iid} N(0, \sigma^2)$$

$$\epsilon = \|y - \hat{y}\|_2^2 = \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)}\right)^2$$



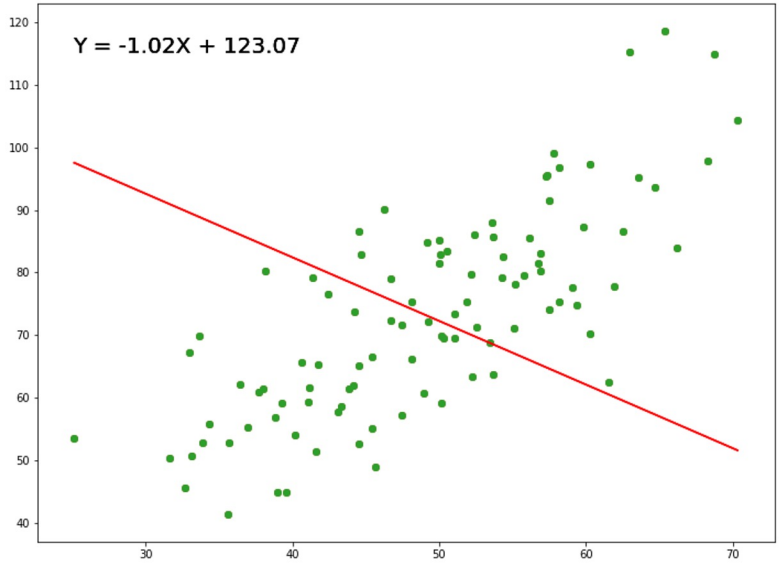
Regression

To find the optimal coefficient β, we need to minimize the error (the sum of squared error) using gradient descent or taking the derivative of its matrix form.



$$\min_{\beta} \epsilon = \min_{\beta} \sum_{i=1}^{n} \left(y^{(i)} - x^{(i)}\beta\right)^2$$

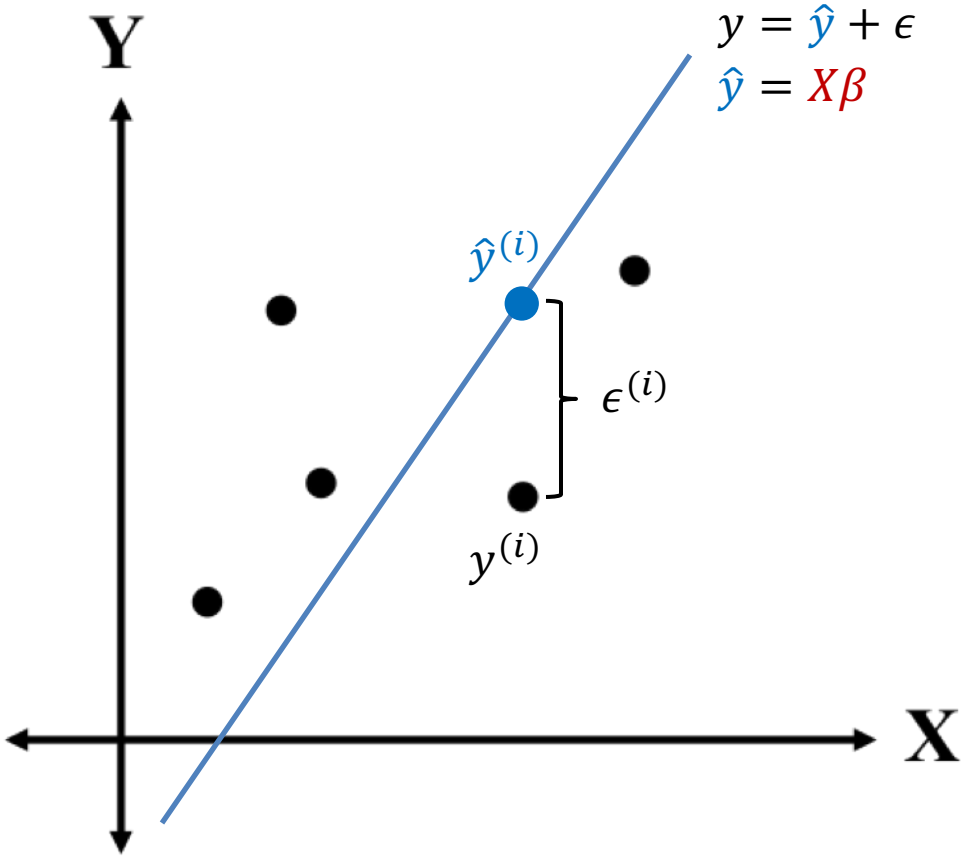
$$= \min_{\beta} (y - X\beta)^T (y - X\beta)$$



More about linear regression -- https://www.stat.cmu.edu/~cshalizi/mreg/15/lectures/13/lecture-13.pdf

Regression

We can generalize linear regression to have multiple predictors (i.e., multiple linear regression) and keep the original mathematical representation.



$x_j^{(i)}$: the j^{th} predictor of the i^{th} data point

$y^{(i)}$: response of the i^{th} data point

$\hat{y}^{(i)}$: estimated response of the i^{th} data point

$$X = [\mathbf{1} \quad x_1 \quad \cdots \quad x_p] = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_p^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{(n)} & \cdots & x_p^{(n)} \end{bmatrix}$$

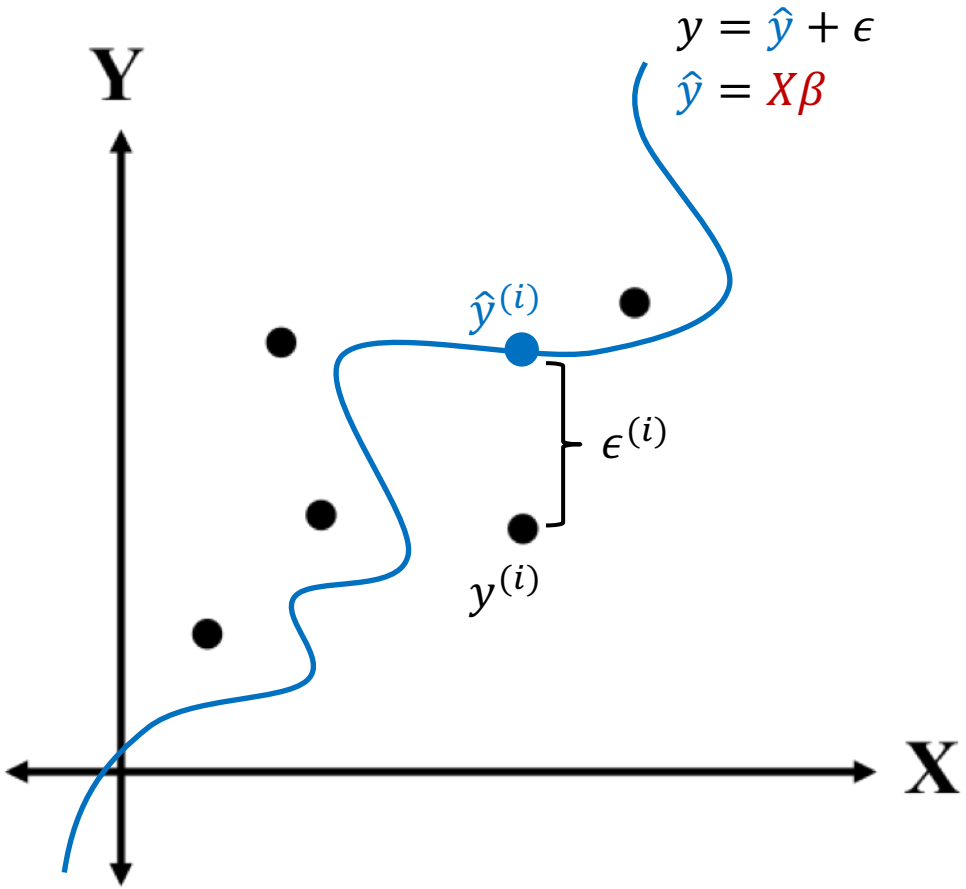
$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$$

$$\hat{y} = X\beta = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p = \begin{bmatrix} \hat{y}^{(1)} \\ \vdots \\ \hat{y}^{(n)} \end{bmatrix}$$

More about multiple linear regression -- https://www.stat.cmu.edu/~cshalizi/mreg/15/lectures/14/lecture-14.pdf

Regression

We can model a non-linear relationship using polynomial functions with degree k. The example below uses one predictor x_1.



y: true response (in vector form)

$\hat{y}$: estimated response (in vector form)

X: predictors/features (in matrix form)

β: coefficient (in vector form)

$$X = [\mathbf{1} \quad x_1 \quad (x_1)^2 \quad \cdots \quad (x_1)^k]$$

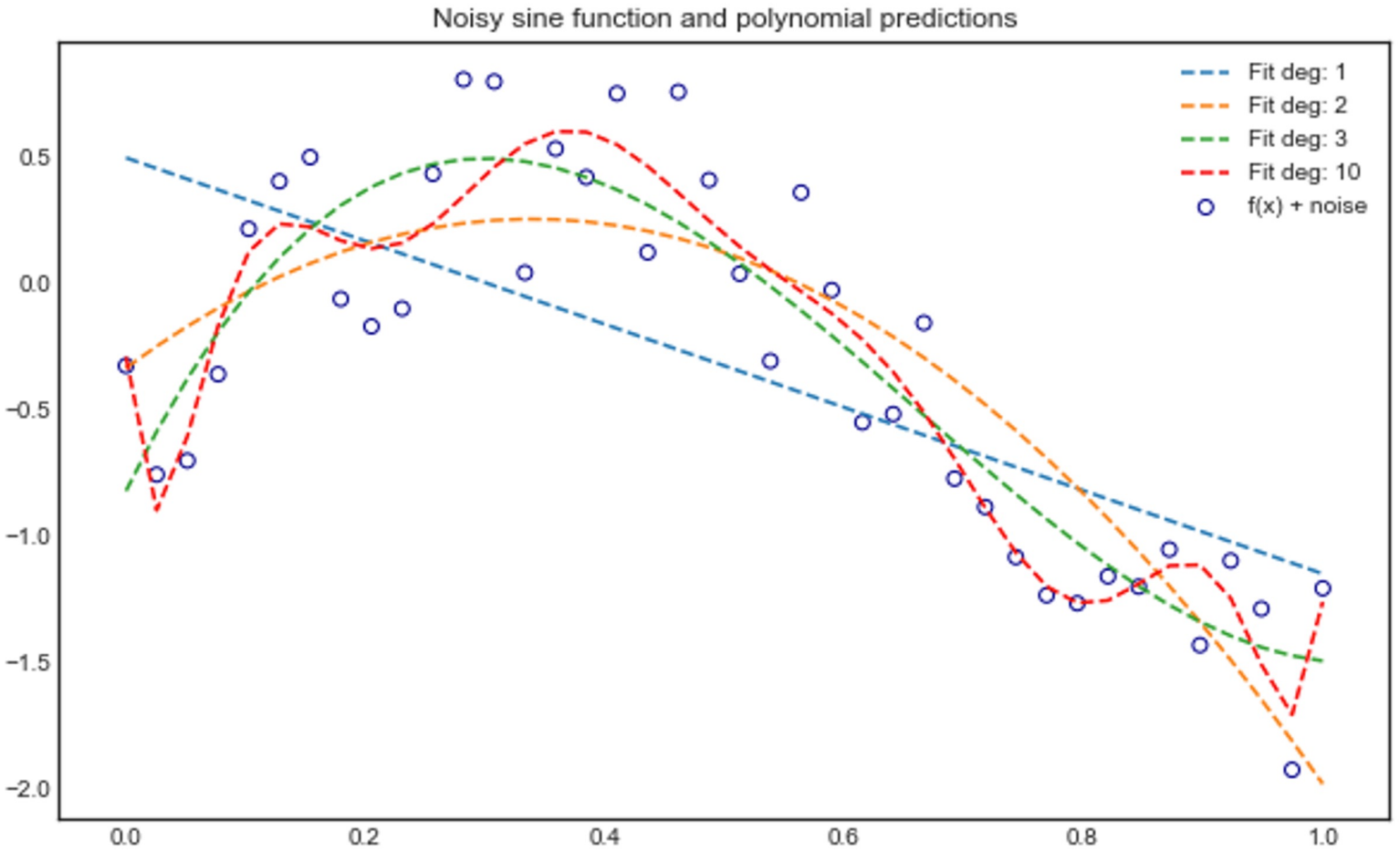
$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}$$

$$\hat{y} = X\beta = \beta_0 + \beta_1 x_1 + \beta_2 (x_1)^2 + \cdots + \beta_k (x_1)^k$$

More about polynomial regression -- https://www.stat.cmu.edu/~cshalizi/mreg/15/lectures/lecture-16.pdf

Regression

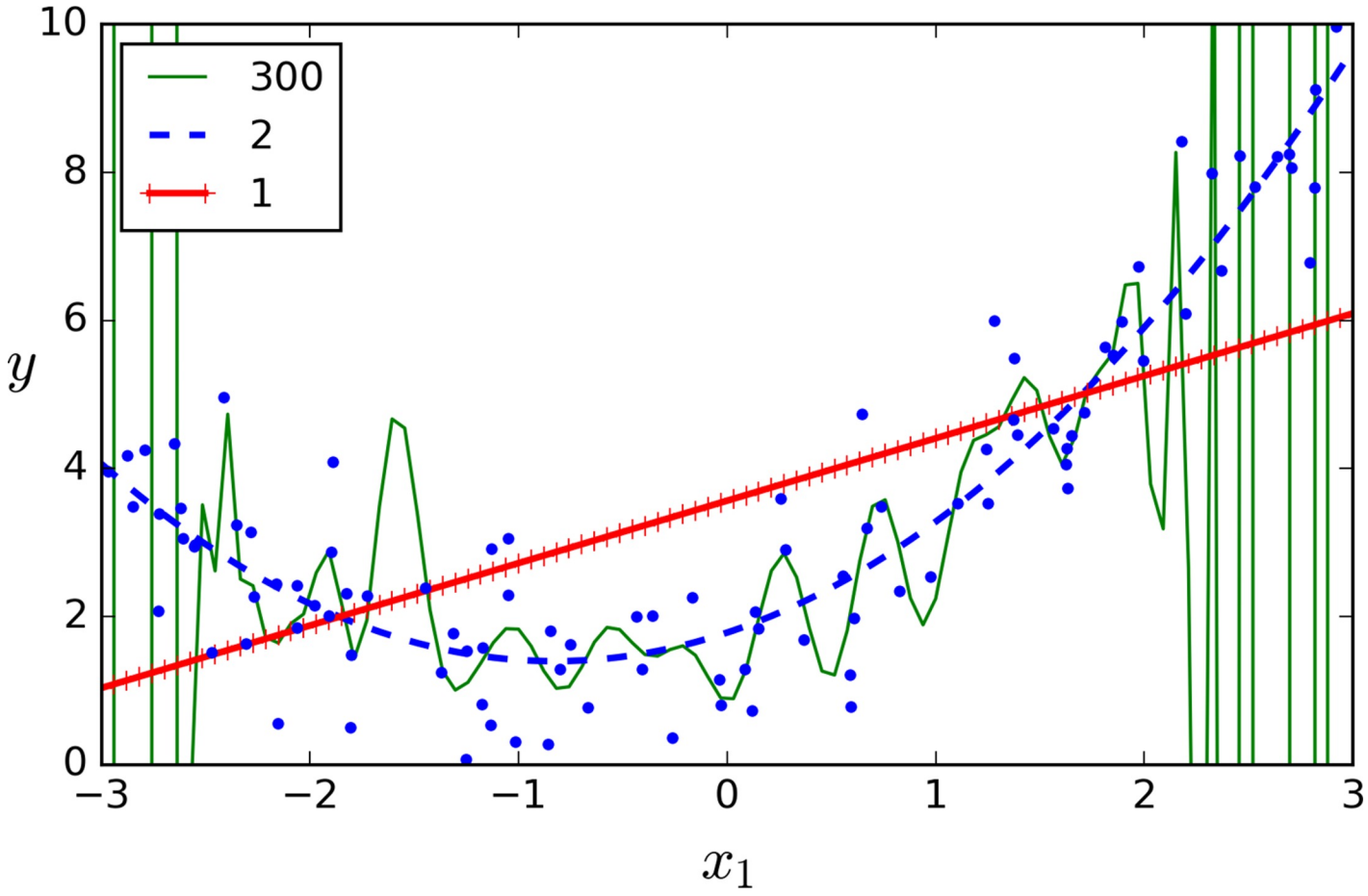
Here is an example of applying linear and polynomial regression to the data that is created using a sine function with some random noise.

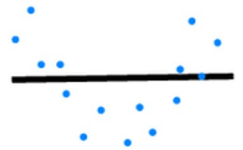


Source -- https://stats.stackexchange.com/questions/350130/why-is-gradient-descent-so-bad-at-optimizing-polynomial-regression

Regression

Using too complex/simple models can lead to overfitting/underfitting, which means the model fits the training set well but generalizes poorly on the test set.

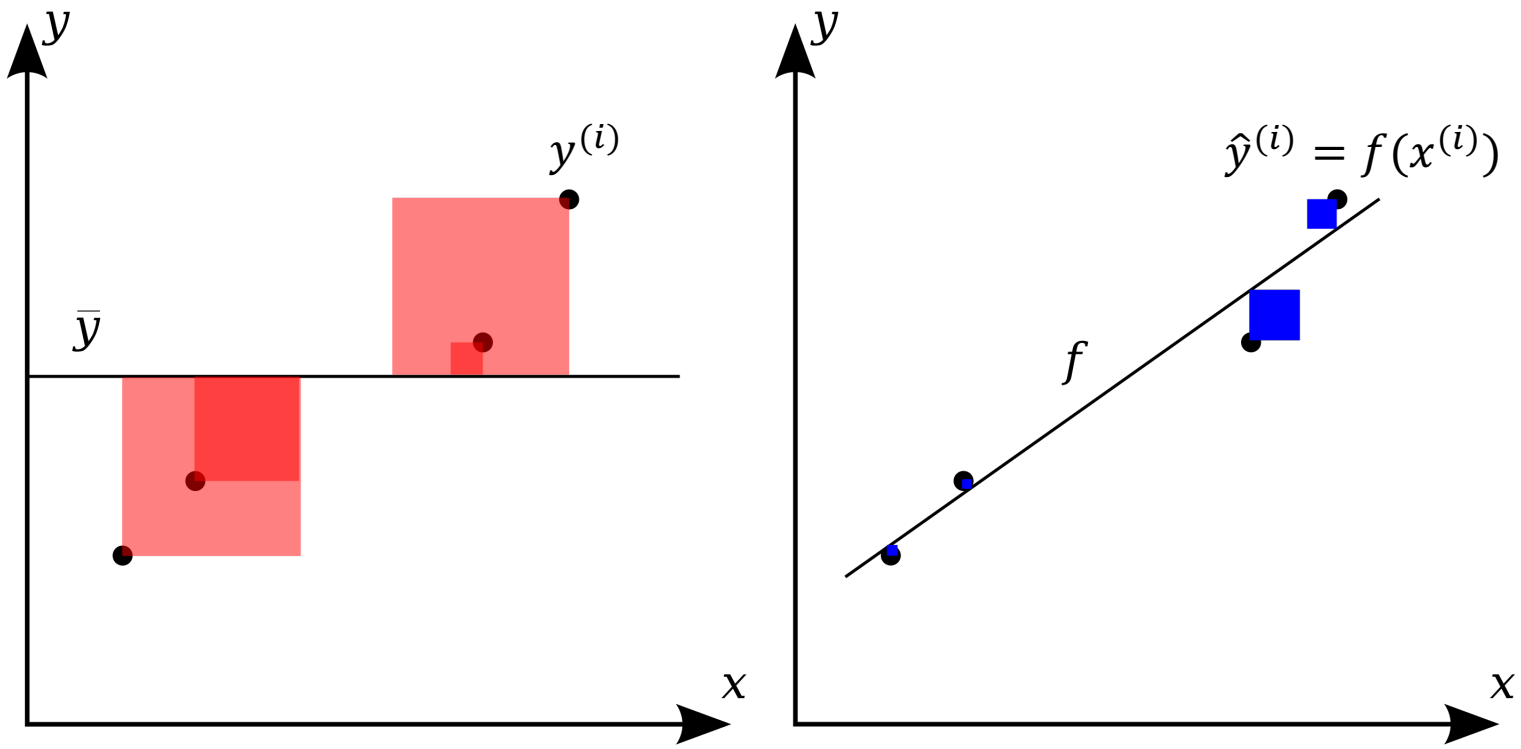




Source -- https://www.oreilly.com/library/view/hands-on-machine-learning/9781491962282/ch04.html

Regression

To evaluate regression models, one common metric is the coefficient of determination (R-squared, R^2). There exist other metrics such as AIC (Akaike's Information Criterion) that is based on likelihood, which is not covered in this lecture.



Total sum of squares (SS_{tot})

Residual sum of squares (SS_{res})

Unexplained Variation

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

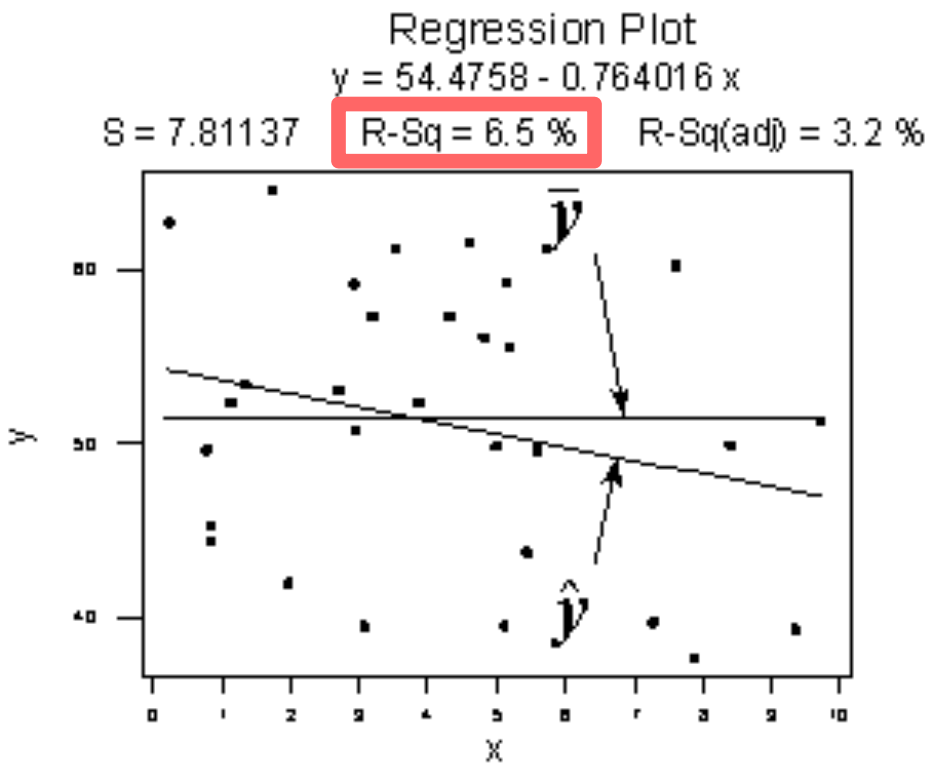
$$SS_{res} = \sum_i \left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

$$SS_{tot} = \sum_i \left(y^{(i)} - \bar{y}\right)^2$$

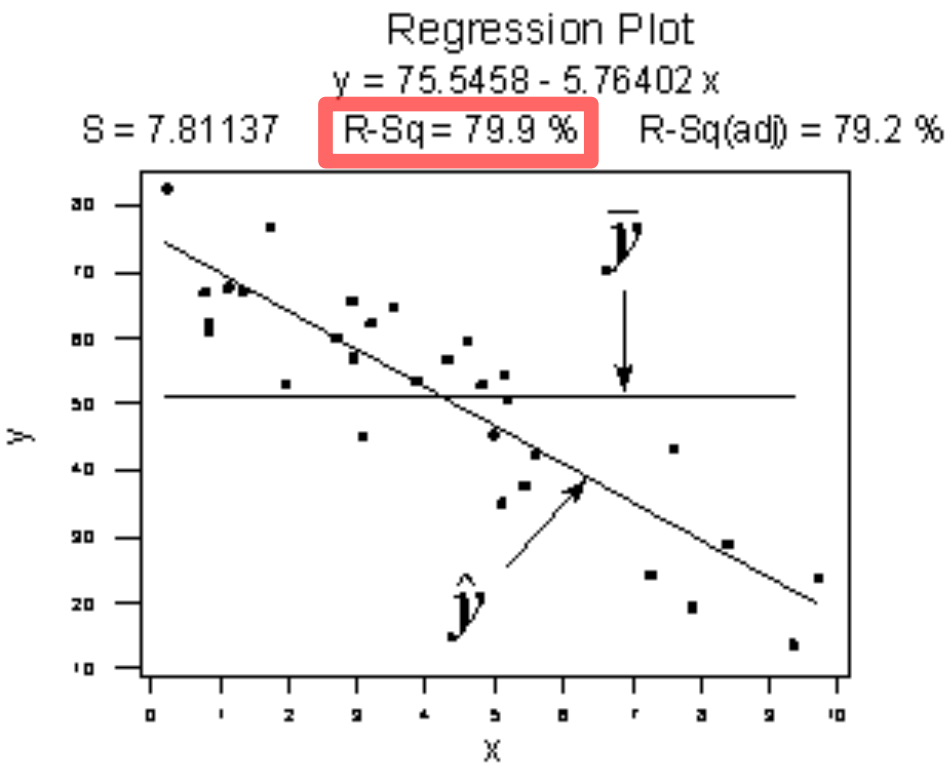
Figure Source -- https://en.wikipedia.org/wiki/Coefficient_of_determination

Regression

For simple/multiple linear regression, R^2 equals the square of Pearson correlation coefficient r between the true y and the estimated $\hat{y} = f(X)$.



Lower Correlation



Higher Correlation

More about the Coefficient of Determination -- https://online.stat.psu.edu/stat501/lesson/1/1.5

Regression

R^2 increases as we add more predictors (because the optimization always want to decrease the residual sum of squares) and thus is not a good metric for model selection. We need the adjusted R^2, which considers the number of predictors.

$$R^2_{adj} = 1 - \frac{SS_{res}/df_{res}}{SS_{tot}/df_{tot}}$$

$df_{res} = n - p - 1$

$df_{tot} = n - 1$

$$SS_{res} = \sum_i \left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

$$SS_{tot} = \sum_i \left(y^{(i)} - \bar{y}\right)^2$$

p: number of features/predictors

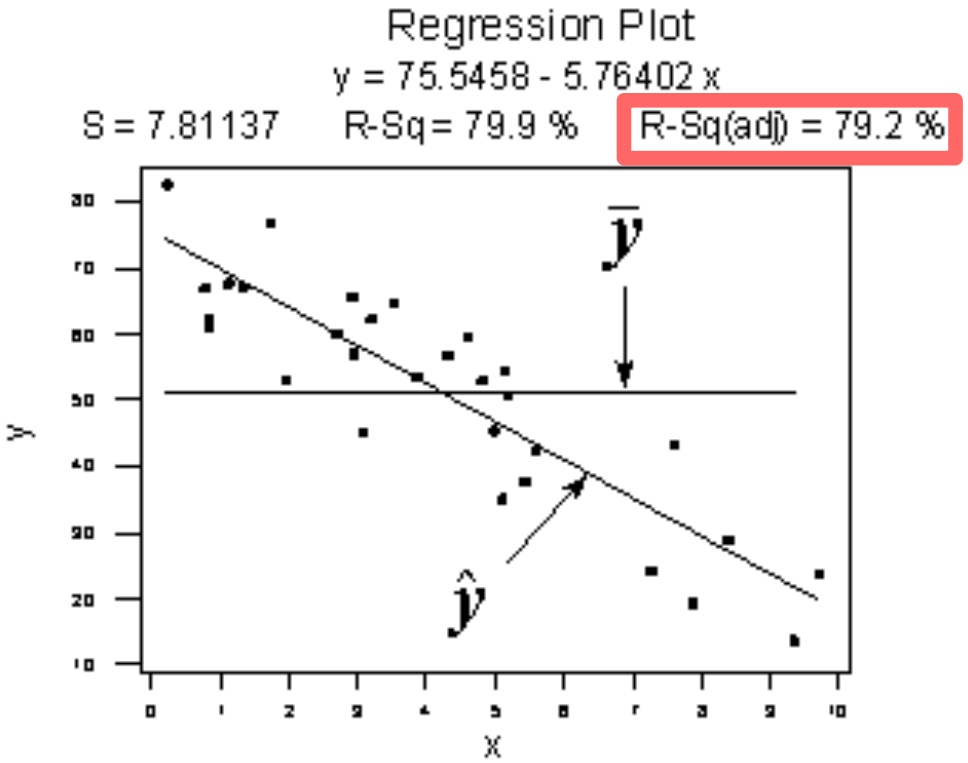
R^2_{adj}: adjusted value of R^2

df_{res}: residual degree of freedom

df_{tot}: total degree of freedom

SS_{res}: residual sum of squares

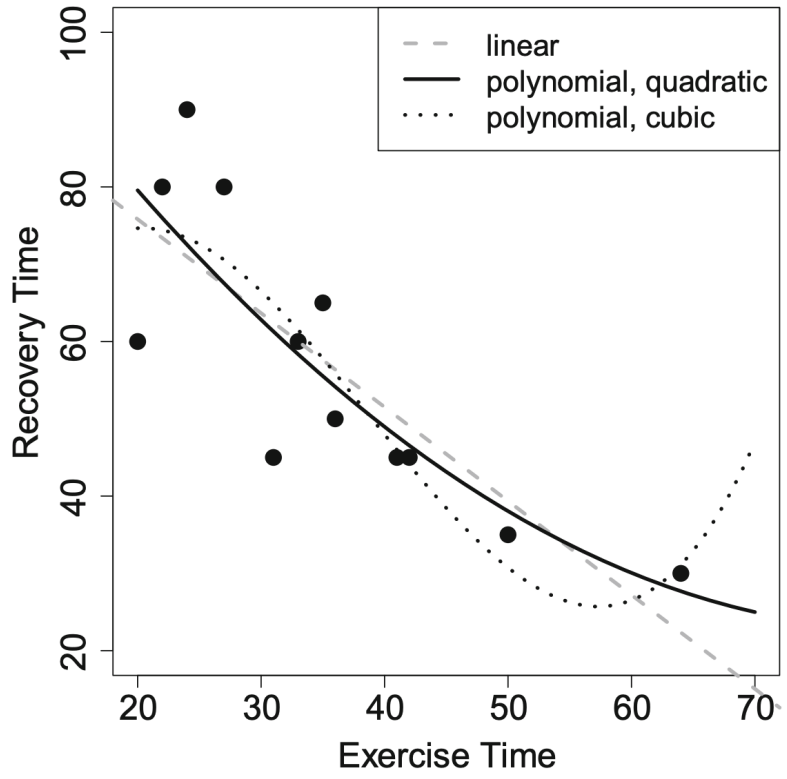
SS_{tot}: total sum of squares

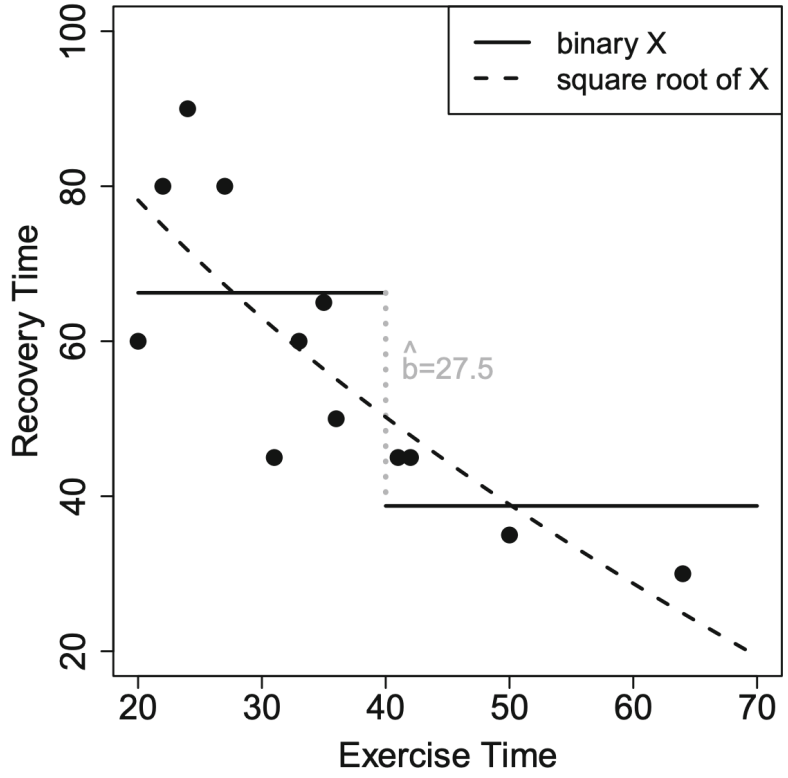


More about the adjusted Coefficient of Determination -- https://online.stat.psu.edu/stat501/lesson/10/10.3

Regression

In the example below, R^2 is larger for the model with more predictors (i.e., the cubic model that has three predictors). The adjusted R^2, which considers the number of predictors (model complexity), favors the the square-root model.

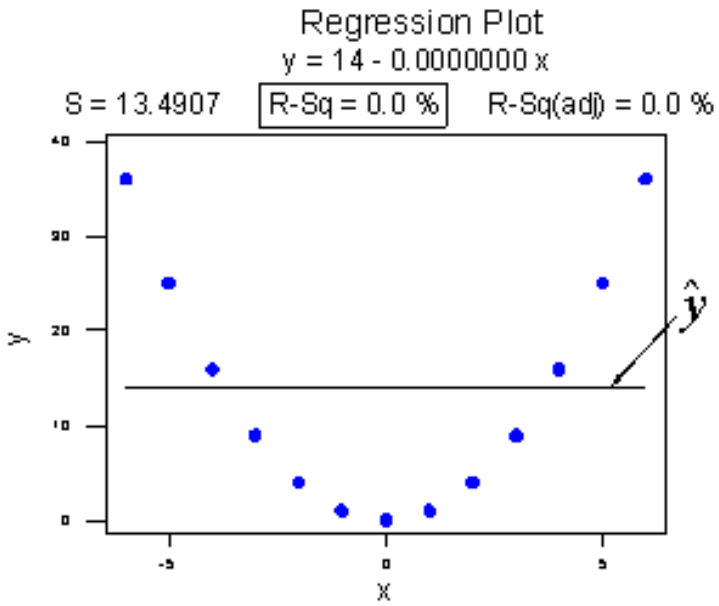


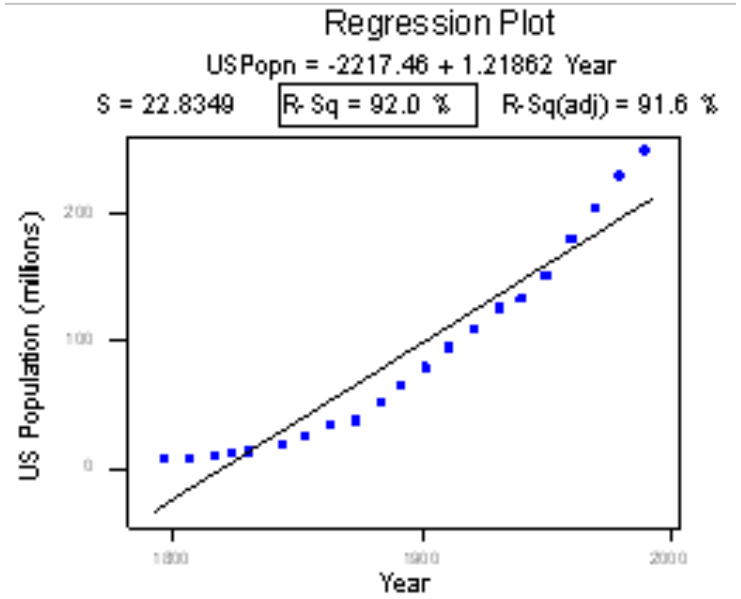


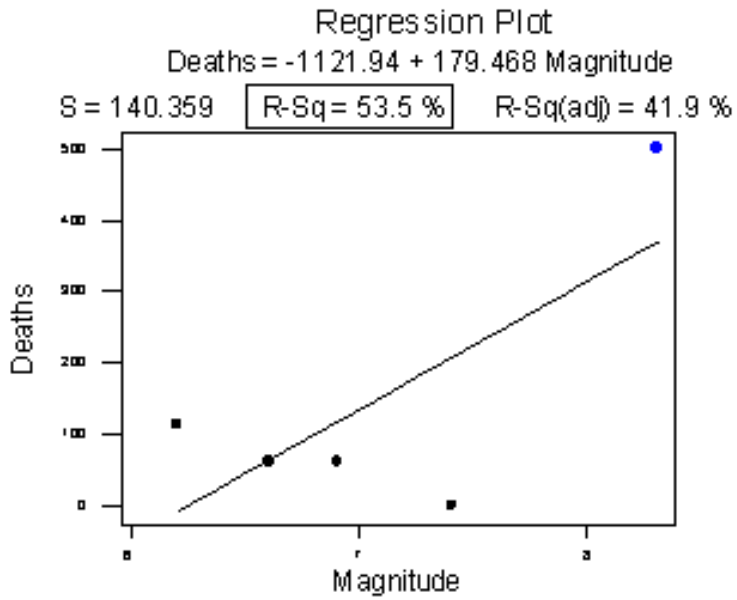
	R^2	R^2_{adj}
Linear	0.6584	0.6243
Quadratic	0.6787	0.6074
Cubic	0.7151	0.6083
Square root	0.6694	0.6363

This example is from section 11.8 Comparing Different Models in book: Introduction to Statistics and Data Analysis

Be careful when using and explaining R^2 in your findings. A bad R^2 does not always mean no pattern in the data. A good R^2 does not always mean that the function fits the data well. And R^2 can be greatly affected by outliers.







Cautions of using Coefficient of Determination -- https://online.stat.psu.edu/stat501/lesson/1/1.8

Take-Away Messages

- Classification outputs discrete labels, while regression outputs continuous values.
- Precision, recall, and F-score are common metrics for evaluating classification models.
- R-squared is a common evaluation metric for regression models.
- Feature engineering is an important step for models that do not use deep learning techniques.
- To train and update a model iteratively, you need a loss function to measure errors.
- Generally, it is a good practice to divide datasets into different parts for model training and testing.
- A model can perform extremely well on the training set but badly on the test set (i.e., overfitting).
- Cross-validation is a good technique to prevent overfitting.

Questions?