AASD 4000
Machine Learning - I

Applied AI Solutions Developer Program

Module 03
Machine Learning

Vejey Gandyer

Agenda

- What is ML?
- Traditional vs ML Approach
- ML Intuition
- Task: ML Problems Identification
- Types of Machine Learning
- Task: ML Problem Types
- ML Concepts
- Evaluation Metrics

ML

What is it?

What is ML?

Machines Learn

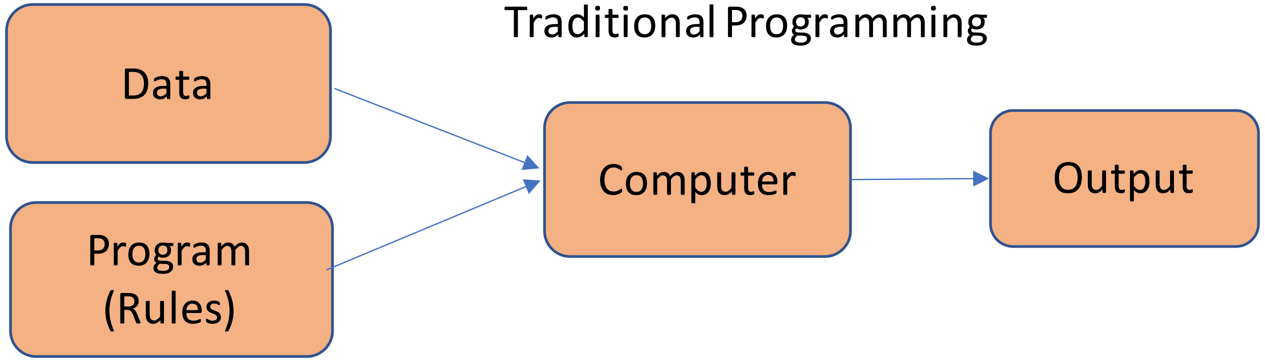
How to combine input

To produce predictions

On unseen data

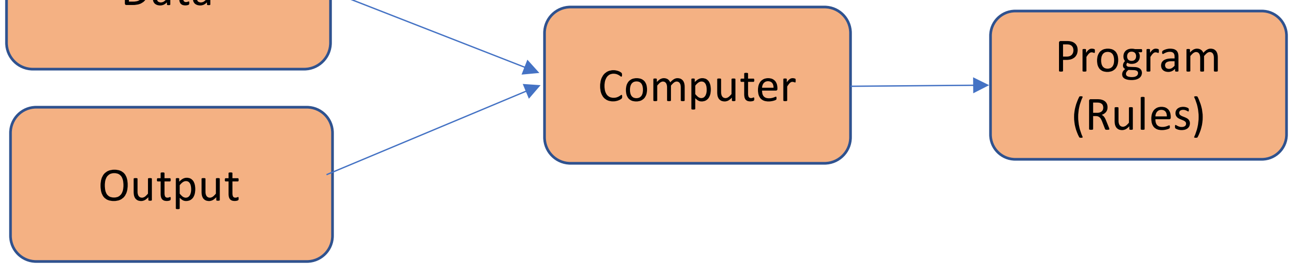
ML vs Traditional Programming

Traditional Programming



Learn output from data given rules

Machine Learning



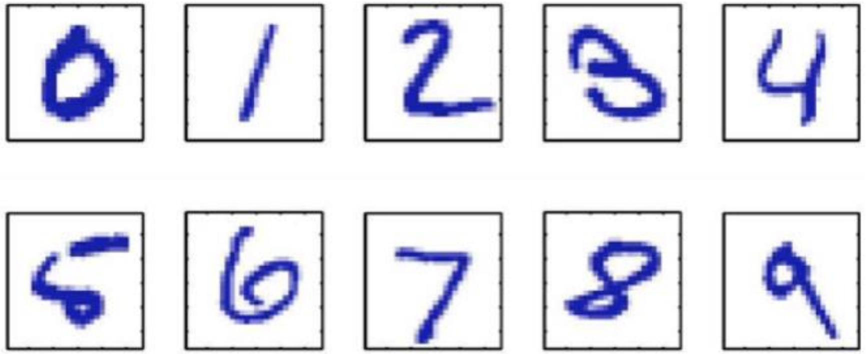
Learn rules from data given desired output

ML Intuition

Digit Recognition

ML Intuition

Task: Digit Recognition

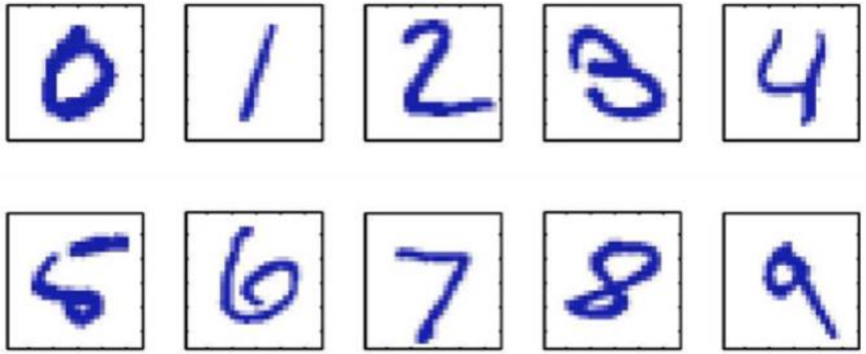


Build an expert algorithm that identifies these digits from your own expertise of digits identification
Assumptions: All given images are 16*16 pixels
Can you construct simple rules and code them?

```
def count_vert_lines(image):
    ...
def count_horiz_lines(image):
    ...

def classify(image):
    ...
    nv = count_vert_lines(image)
    nh = count_horiz_lines(image)
    ...

    if (nv == 1) and (nh == 1):
        digit = 7
    ...

    return digit
```

ML Intuition

Task: Digit Recognition



Build an expert algorithm that identifies these digits from your own expertise of digits identification

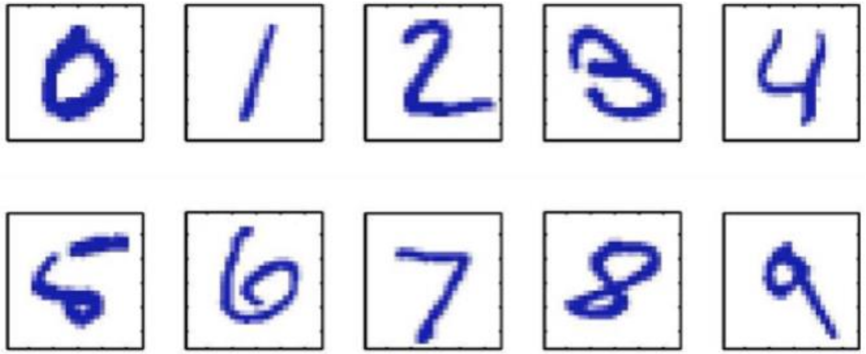
```
def count_vert_lines(image):
    ...
def count_horiz_lines(image):
    ...

def classify(image):
    ...
    nv = count_vert_lines(image)
    nh = count_horiz_lines(image)
    ...

    if (nv == 1) and (nh == 1):
        digit = 7
    ...

    return digit
```

Too many rules

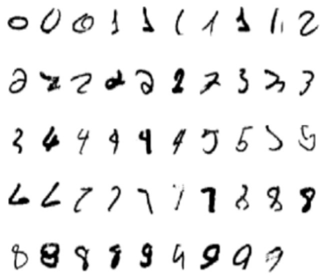
Simple to start, but very complex as you start adding different kinds of data

ML Intuition

Task: Digit Recognition



Training inputs images $\boldsymbol{x}_i$ (ex. 5000 ex per class)



?

Learned classifier
$f(\boldsymbol{x})$

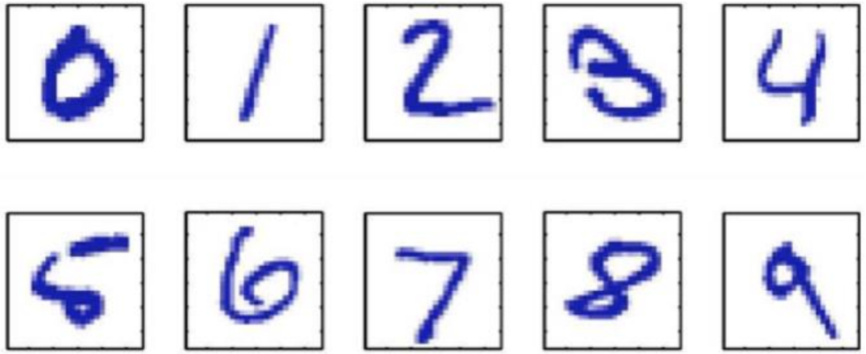
Training output labels $y_i \in \{0,1,\ldots,9\}$

```
def count_vert_lines(image):
    ...
def count_horiz_lines(image):
    ...

def classify(image):
    ...
    nv = count_vert_lines(image)
    nh = count_horiz_lines(image)
    ...

    if (nv == 1) and (nh == 1):
        digit = 7
    ...

    return digit
```

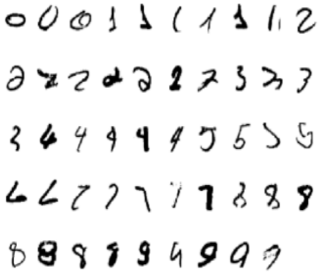
Learn the function (?) from the data

ML Intuition

Task: Digit Recognition



Training inputs images x_i (ex. 5000 ex per class)



?

Learned classifier $f(x)$

Training output labels $y_i \in \{0,1,\ldots,9\}$

```
def count_vert_lines(image):
    ...
def count_horiz_lines(image):
    ...

def classify(image):
    ...
    nv = count_vert_lines(image)
    nh = count_horiz_lines(image)
    ...

    if (nv == 1) and (nh == 1):
        digit = 7
    ...

    return digit
```

Challenges:

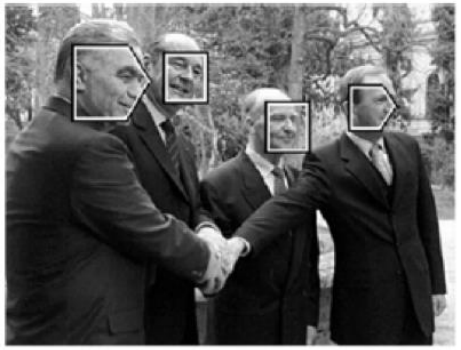
- How do we acquire data? Someone has to manually label examples.
- How do we parametrize a set of functions to search?
- How do we fit the function to data?
- If a function works on training example, will it generalize on new data?

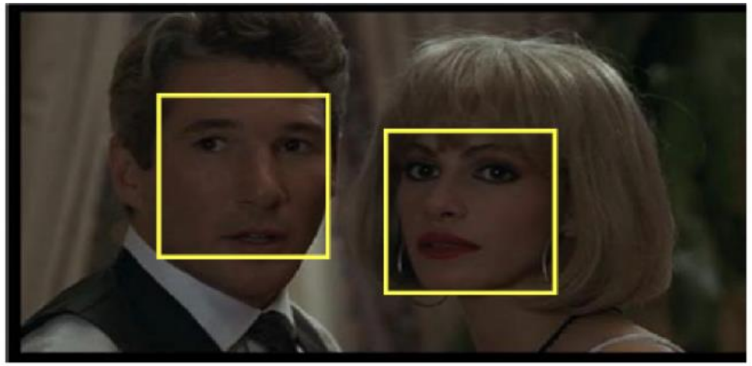
ML Intuition

Face Detection

ML Intuition

Task: Face Detection

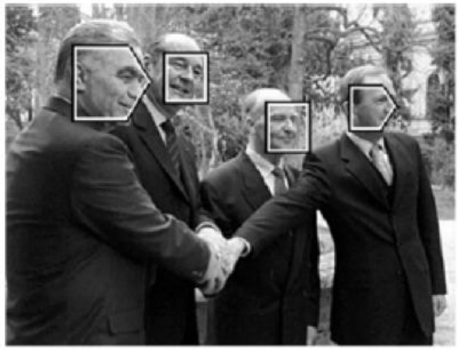


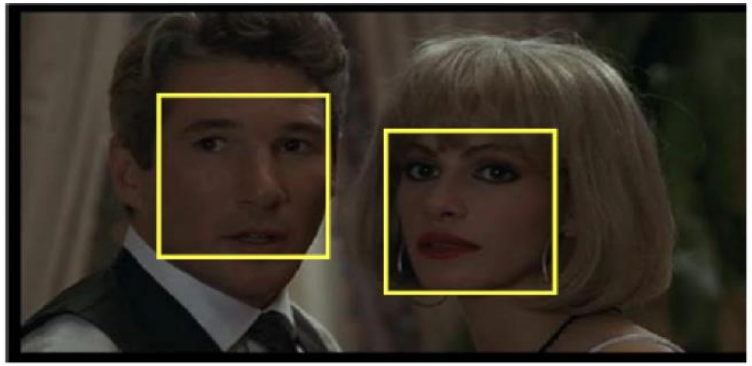


Build a face detector: for every region in image, determine if face is present or not

Harder problem: Harder to describe a face than a digit

ML Intuition

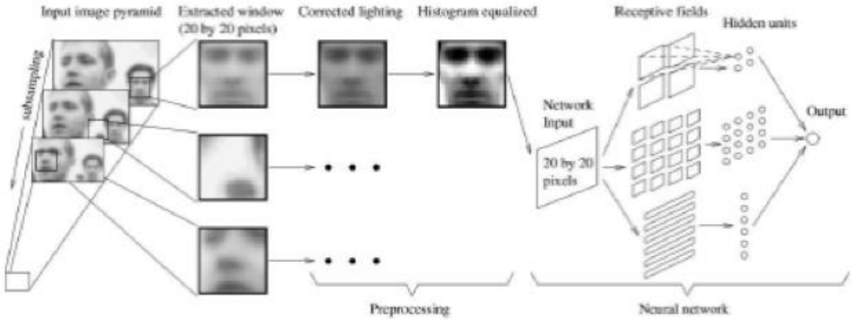




Task: Face Detection

Build a face detector: for every region in image, determine if face is present or not

Harder problem: Harder to describe a face than a digit



ML Intuition

Spam Detection

ML Intuition

Task: Spam Detection

Build a spam detector: for every email, determine if spam or not

Harder problem: Harder to describe a text

<u>Representing Text</u>
Common model: bag of words
Enumerate all words,
Represent email via word count num instances of word
<u>Challenges</u>
Very high-dimensional vector
System must continue to adapt (keep up with spammers)

ML Applications

ML Problems Identification

Task 3

ML Problems Identification

Task3: List a set of problems that could be solved by ML in the following domains. Also specify their level of difficulty in solving.

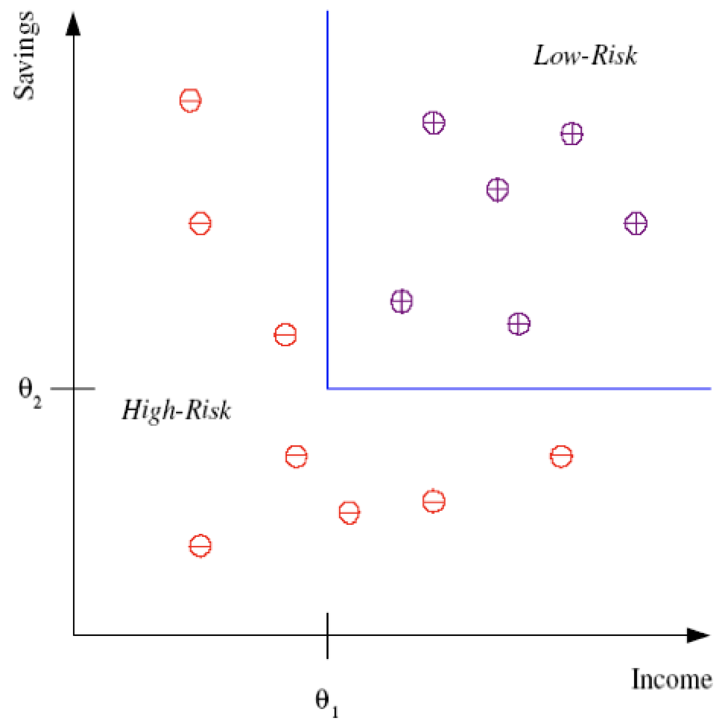
- Healthcare
- Education
- Banking
- E-Commerce
- Gaming
- Any other problems around you

Types of Machine Learning

Classification

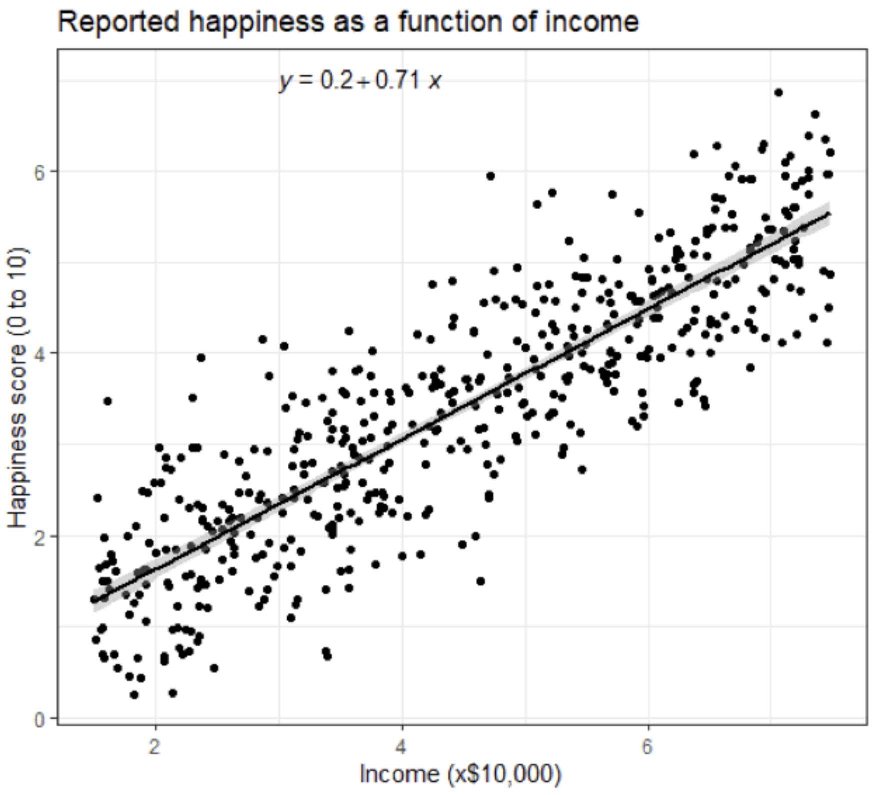
<u>Supervised learning</u>
Learn mapping from features x to target y
<u>Classification</u>:
Target is discrete. One of a finite number of values
Example: Credit assessment
Target: customer is high-risk or low-risk
Features: income & saving
Learn a function from features to target
- Use past training data
- Need to get this data

The function on the right is an example of a decision tree.



Regression

Also supervised learning
Predicting a continuous-valued target
Example:
Predict y=happiness score (e.g. from surveys)
From x=income, country, age, …
Can use multiple predictors
Assume some form of the mapping
Find parameters from data



Unsupervised Learning

Learning "what normally happens"

No output

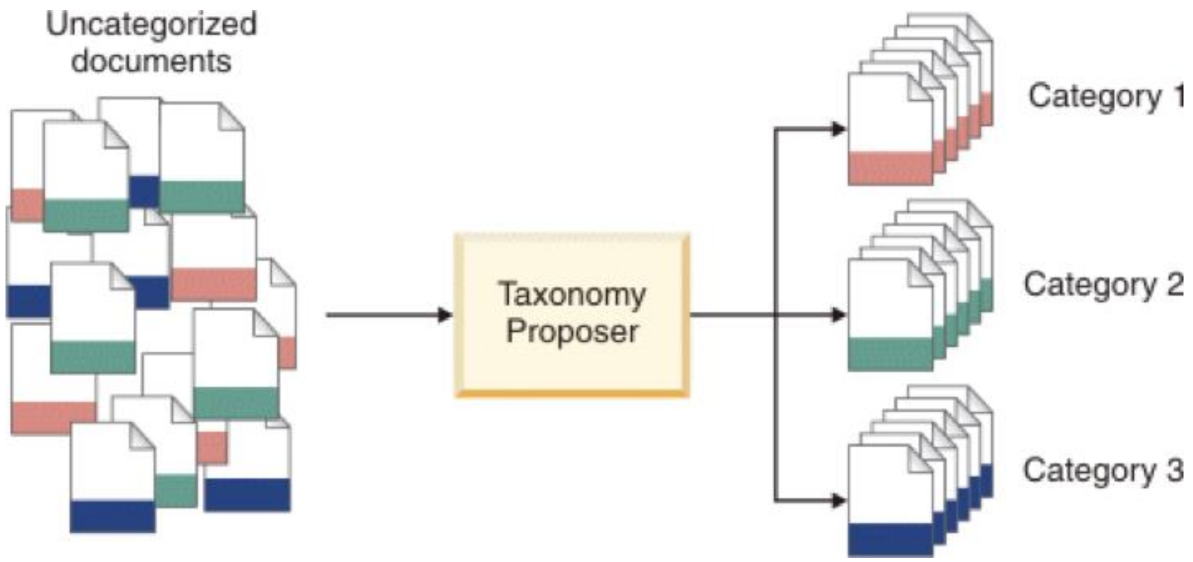
Just values x. No target y

<u>Clustering</u>: Grouping similar instances

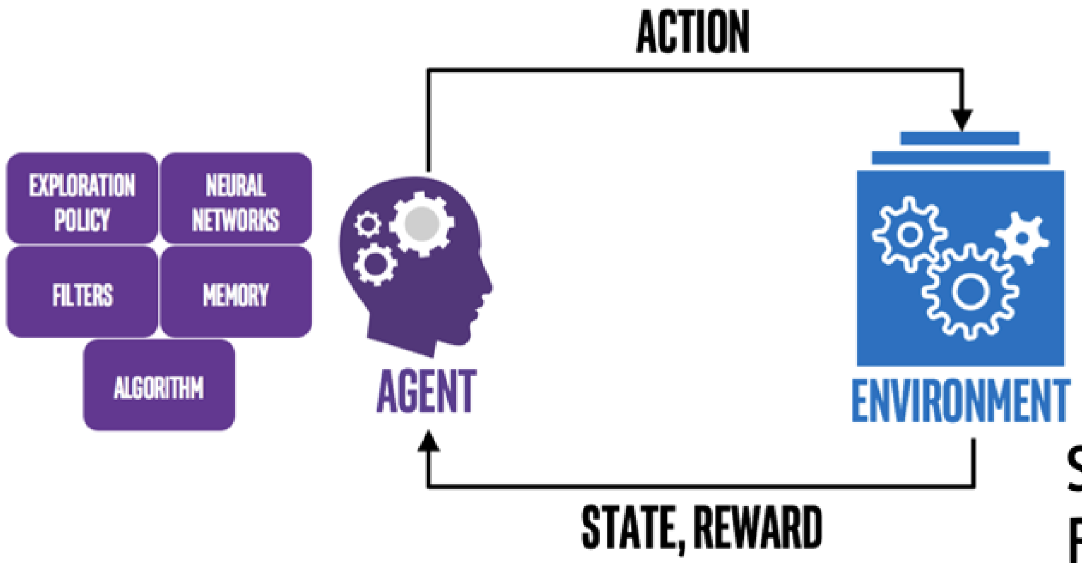
<u>Example applications</u>

Customer Segmentation

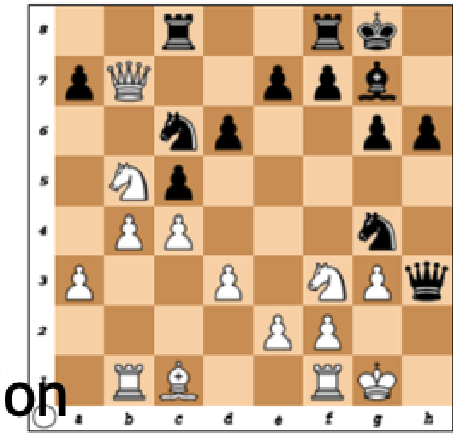
Image Compression: Color Quantization



Reinforcement Learning



Move: ...Bd4



State: Current position

Reward: Win or Lose

Agent learns to make **actions** that interact with an **environment** to maximize a **reward**

Exploitation (Learn from past actions) Vs Exploration (Try new choices)

Credit assignment: Which actions in the past led to the current reward?

ML Problem Types

Task 4

ML Problem Types

Task4: For the set of problems identified in Task3, specify their ML problem type (classification, regression, clustering, association, reinforcement) for the following domains.

- Healthcare
- Education
- Banking
- E-Commerce
- Gaming
- Any other problems around you

ML Concepts

ML Concepts

Label
Features
Example
Labeled Example
Unlabeled Example
Algorithm
Model
Classification
Regression
Unsupervised
Reinforcement

Linear Regression
Training
Loss
Gradient Descent
Learning Rate
Hyperparameter
Generalization
Training set
Test set
Validation set

Regularization
L_1
L_2
Accuracy
Precision
Recall
F1 Score
ROC
AUC

Labels

A label is the thing we're predicting—the ***y*** variable in simple linear regression.

- future price of wheat
- kind of animal shown in a picture
- meaning of an audio clip

Features

A Feature is an input variable —the ***x*** variable in simple linear regression.

A simple machine learning project might use only a single feature or a set of features

$$x_1, x_2, \ldots x_N$$

Spam detection project features include:

- words in the email text
- sender's address
- time of day the email was sent
- email contains the phrase "one weird trick."

Examples

An Example is a particular instance of data ***x***

Two types of Examples

- Labeled examples - {features, label}: (x, y)

- Unlabeled examples - {features, ?}: (x, ?)

Model

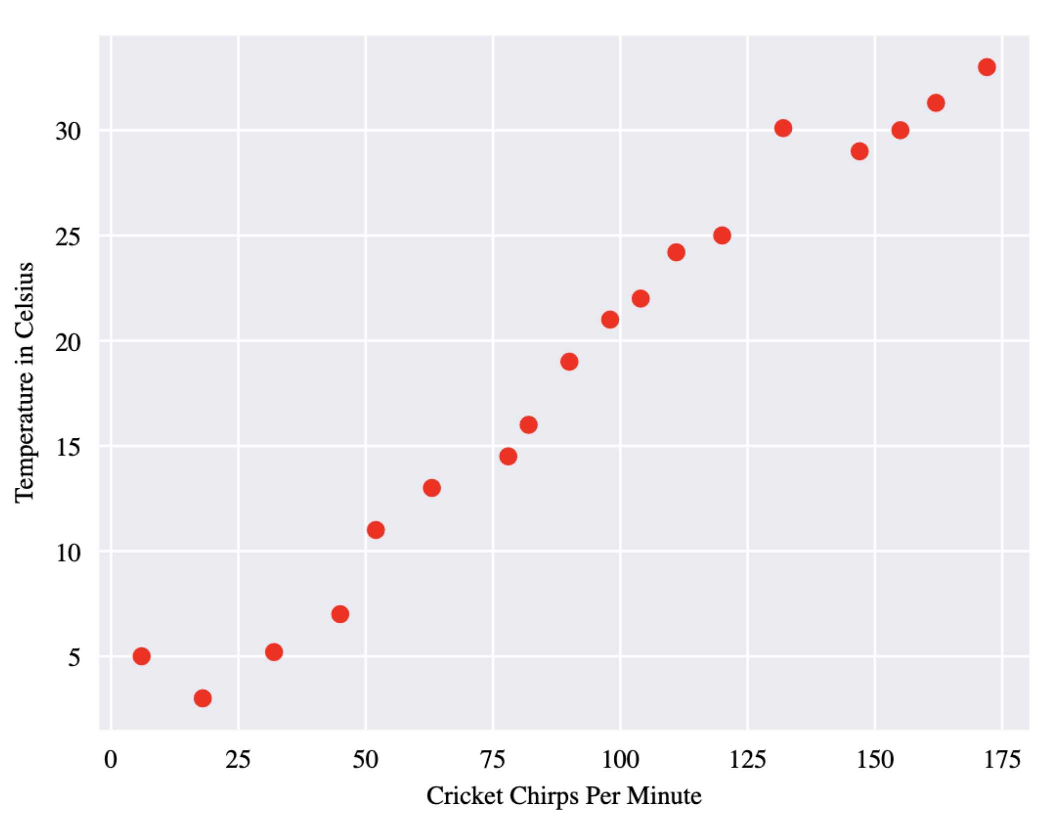
A model defines the relationship between features and label.

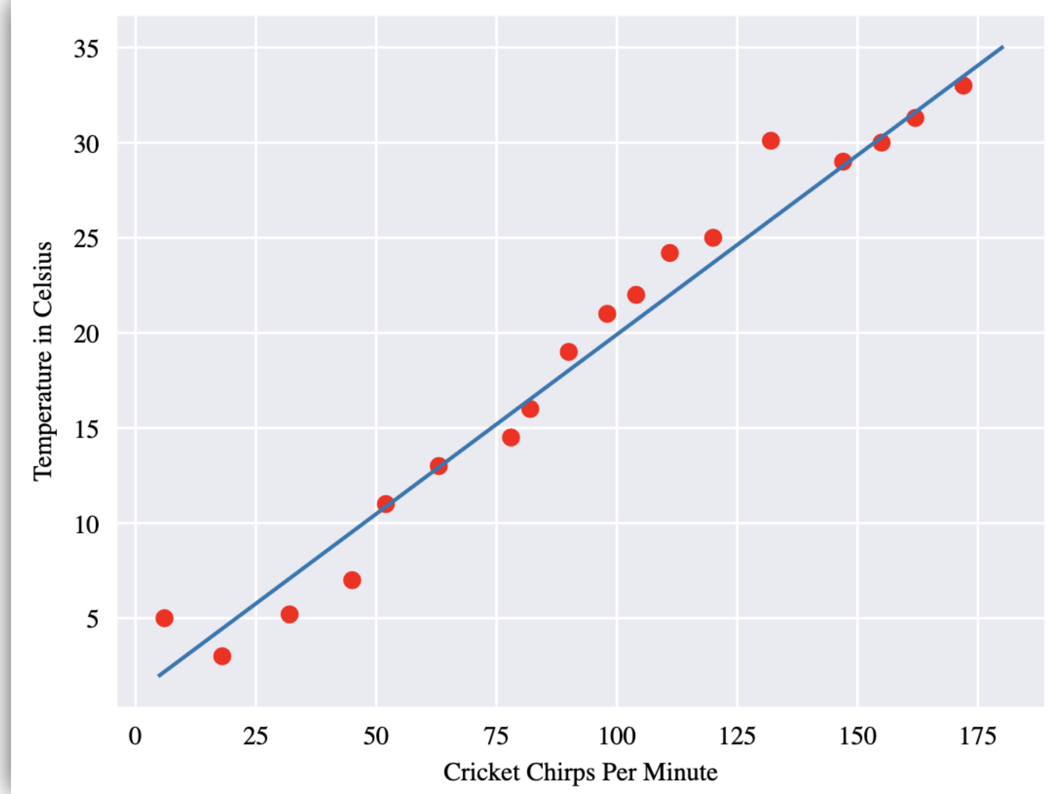
Two phases of a model's lifecycle:

- **Training** means creating or learning the model. Show the model labeled examples and enable the model to gradually learn the relationships between features and label
- **Inference** means applying the trained model to unlabeled examples. Use the trained model to make useful predictions (y').

Ex: Predict ***price*** for new unlabeled and unseen examples

Linear Regression





Linear Regression – High school Math

$$y = mx + b$$

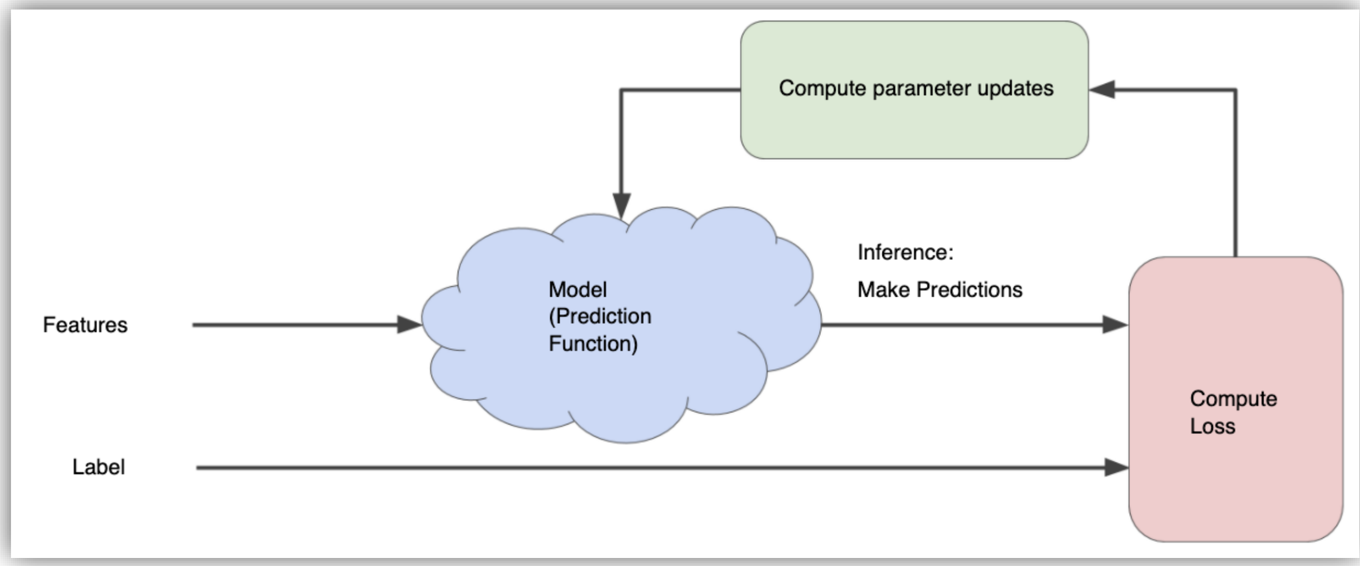
$$y' = b + w_1 x_1$$

$$y' = b + w_1 x_1 + w_2 x_2 + w_3 x_3$$

Model Training

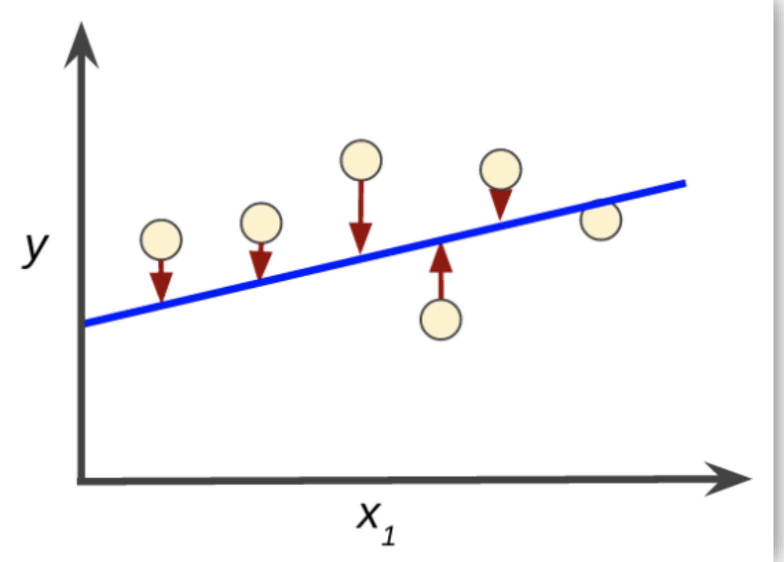
Training a model simply means learning (determining) good values for all the weights and the bias from labeled examples.

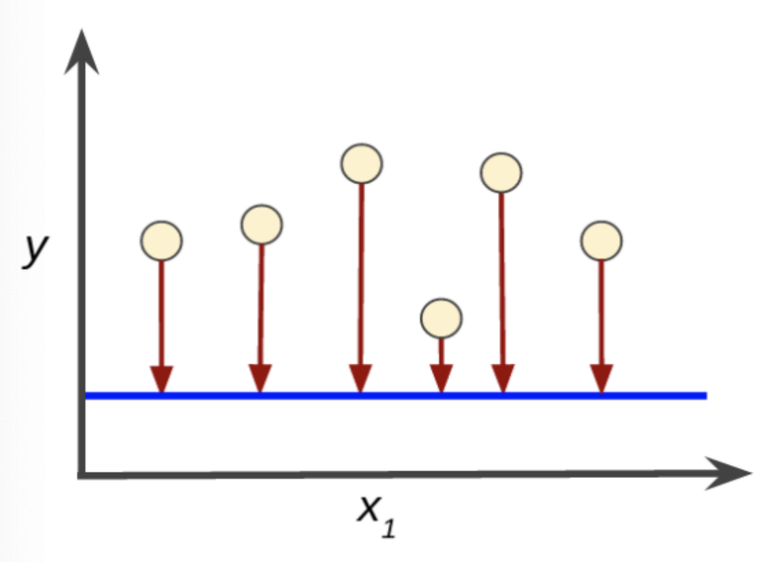
In supervised learning, a machine learning algorithm builds a model by examining many examples and attempting to find a model that minimizes loss; this process is called **empirical risk minimization**



Loss

Loss - Penalty for a bad prediction





Goal: Find a set of weights and biases that have the **least loss**

A Machine Learning model is trained by starting with an initial guess for the weights and bias and iteratively adjusting those guesses until learning the weights and bias with the lowest possible loss.

Loss

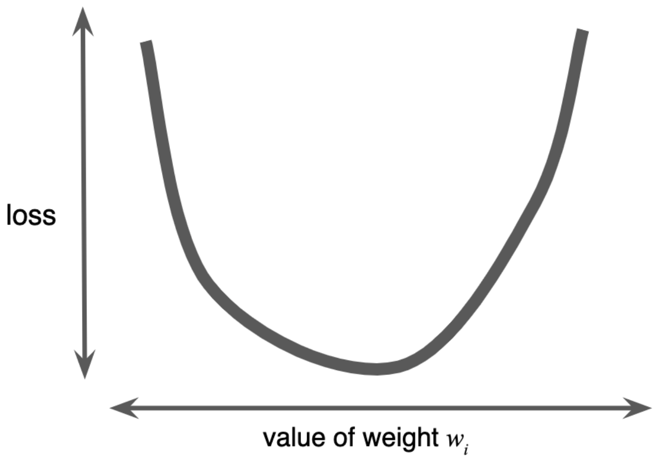
Mean Square Error (MSE) - Average squared loss per example over the whole dataset

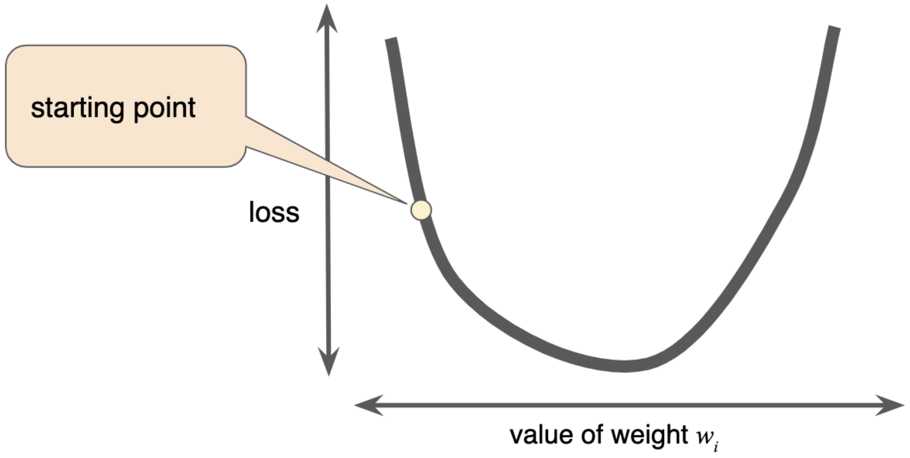
$$MSE = \frac{1}{N} \sum_{(x,y)\in D} (y - prediction(x))^2$$

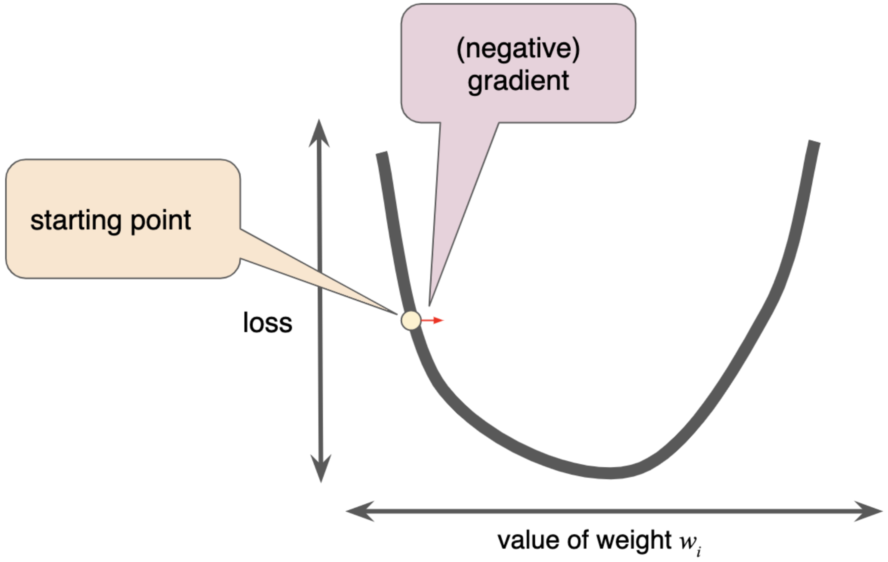
Learning continues iterating until the algorithm discovers the model parameters with the lowest possible loss. Usually, iterate until overall loss stops changing or at least changes extremely slowly. When that happens, the model has **converged**

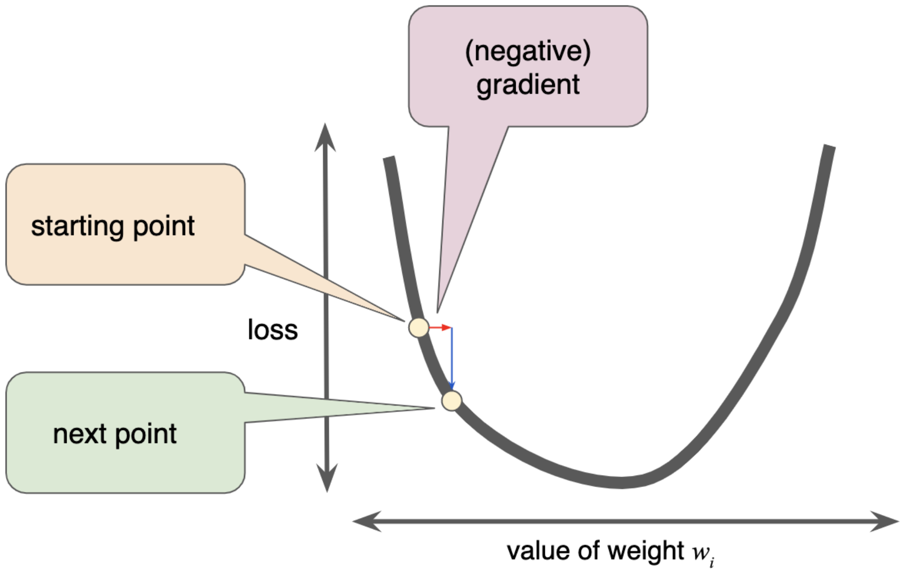
Gradient Descent

~~Gradients, Partial Derivates, Calculus~~



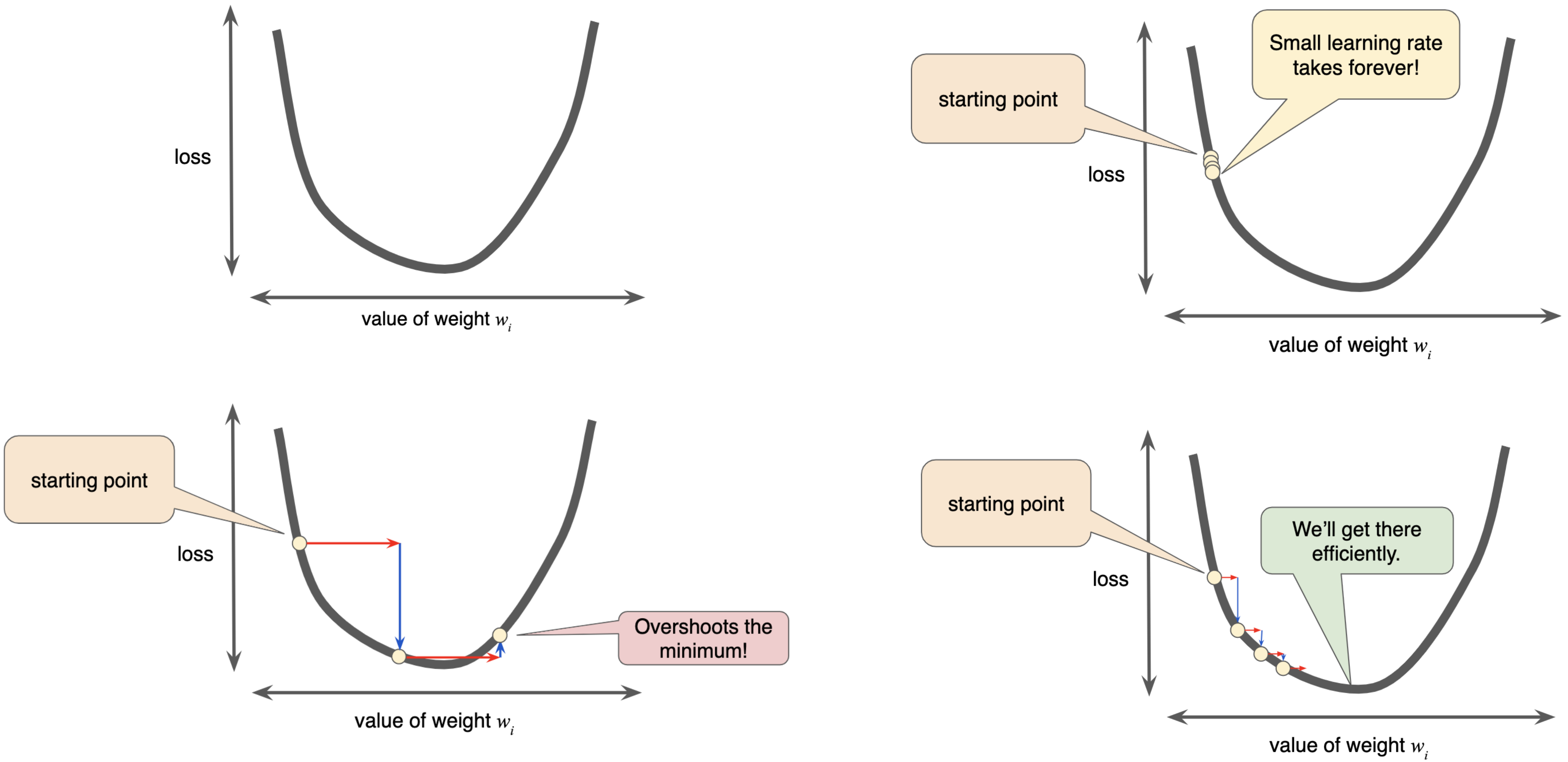




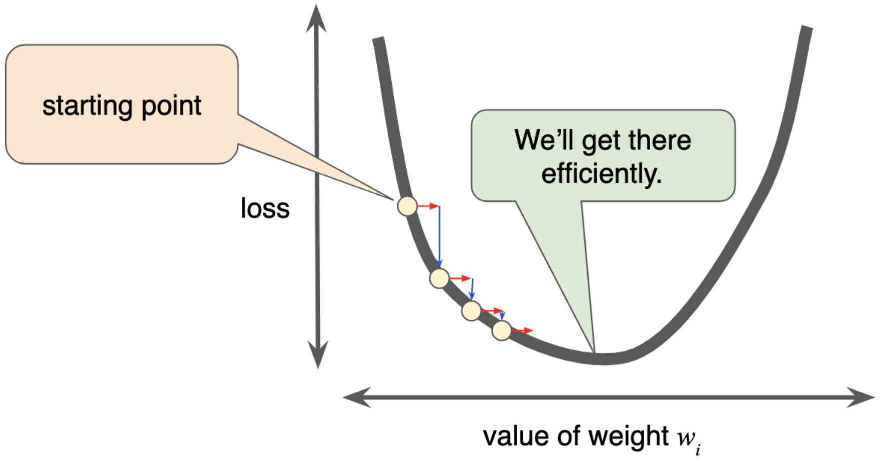


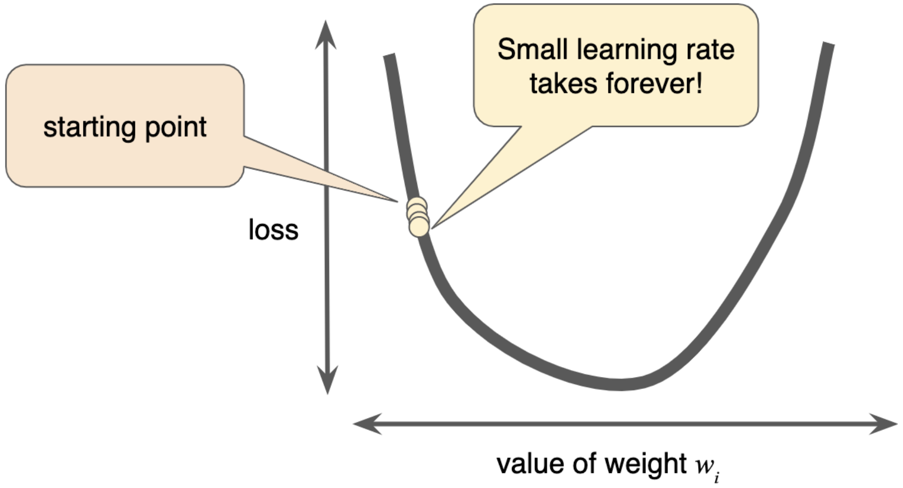
Learning Rate

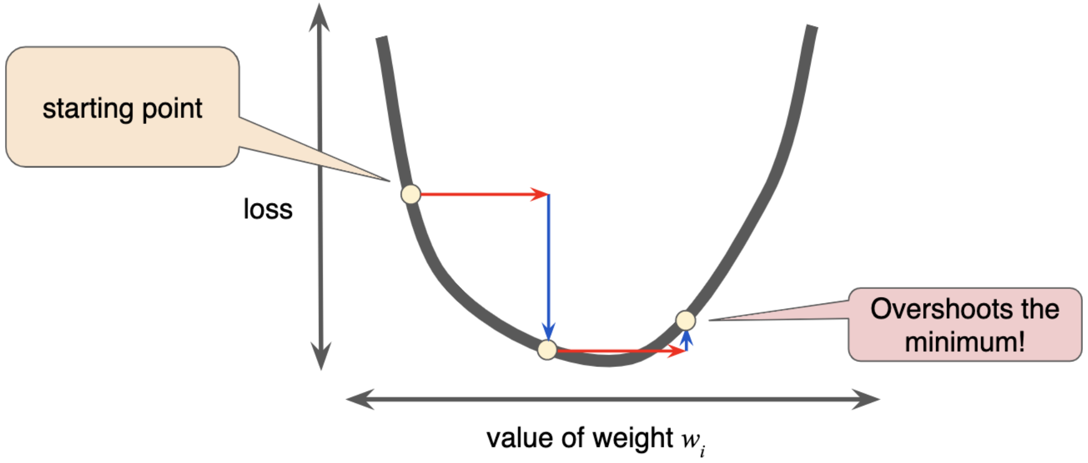
Gradient descent algorithms multiply the gradient by a scalar known as the **learning rate** (also sometimes called **step size**) to determine the next point.



Hyperparameters







Hyperparameters are the knobs that programmers tweak in machine learning algorithms.

Goldilocks principle is used to find learning rate.

The goal is to find a learning rate **large enough** that gradient descent converges efficiently, but not so large that it never converges.

Types of Gradient Descent

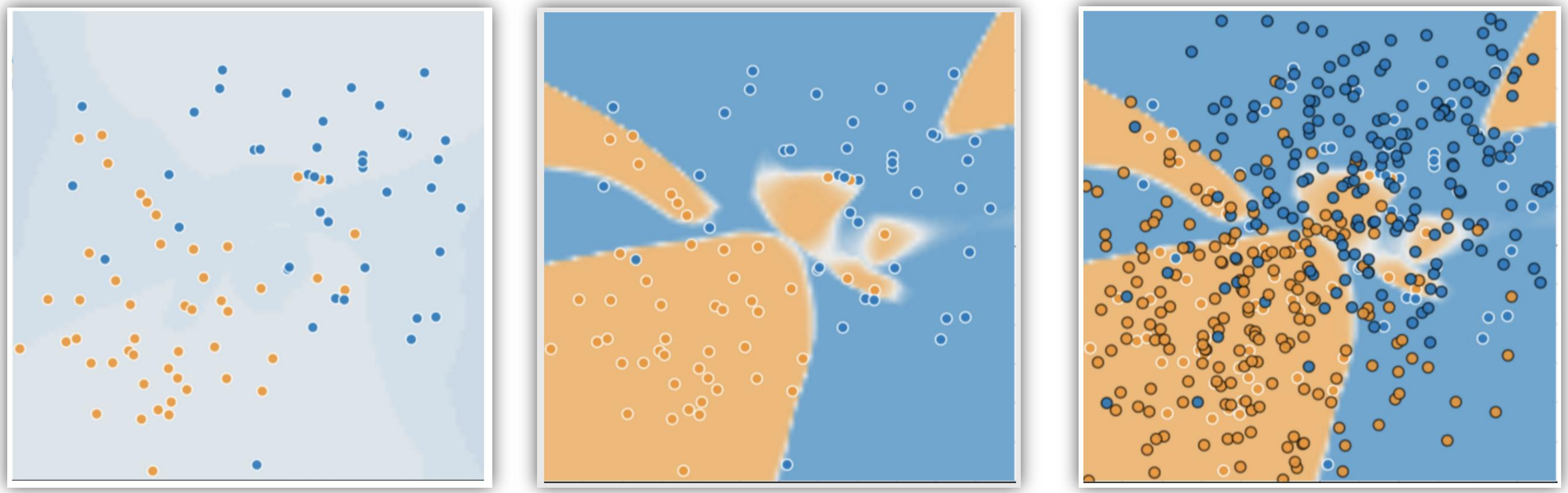
Batch – Examples used to calculate gradient in a single iteration

Batch Gradient Descent - entire dataset (size=N examples) per iteration
Long time to compute gradients of the entire batch

Stochastic Gradient Descent (SGD) - 1 example (size=1) per iteration
Works fine but it is very noisy

Mini-Batch Gradient Descent (mini-batch SGD)
10-1000 examples (batch size = 10 or 1000 examples) per iteration
Reduces noise that occurs in SGD but efficient than Batch SGD

Generalization

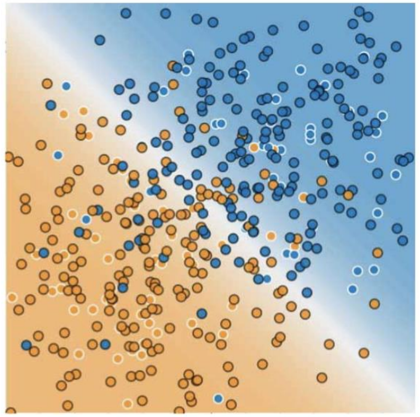


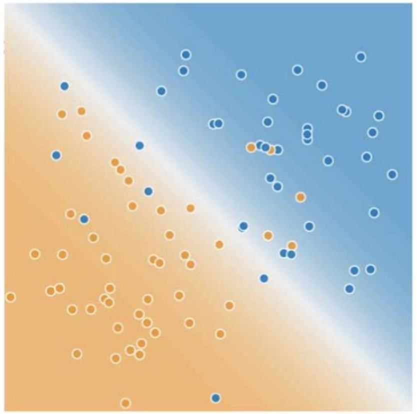
Ockham's Razor
The less complex a ML model, more likely that a good empirical result is not only due to pecularities of sample

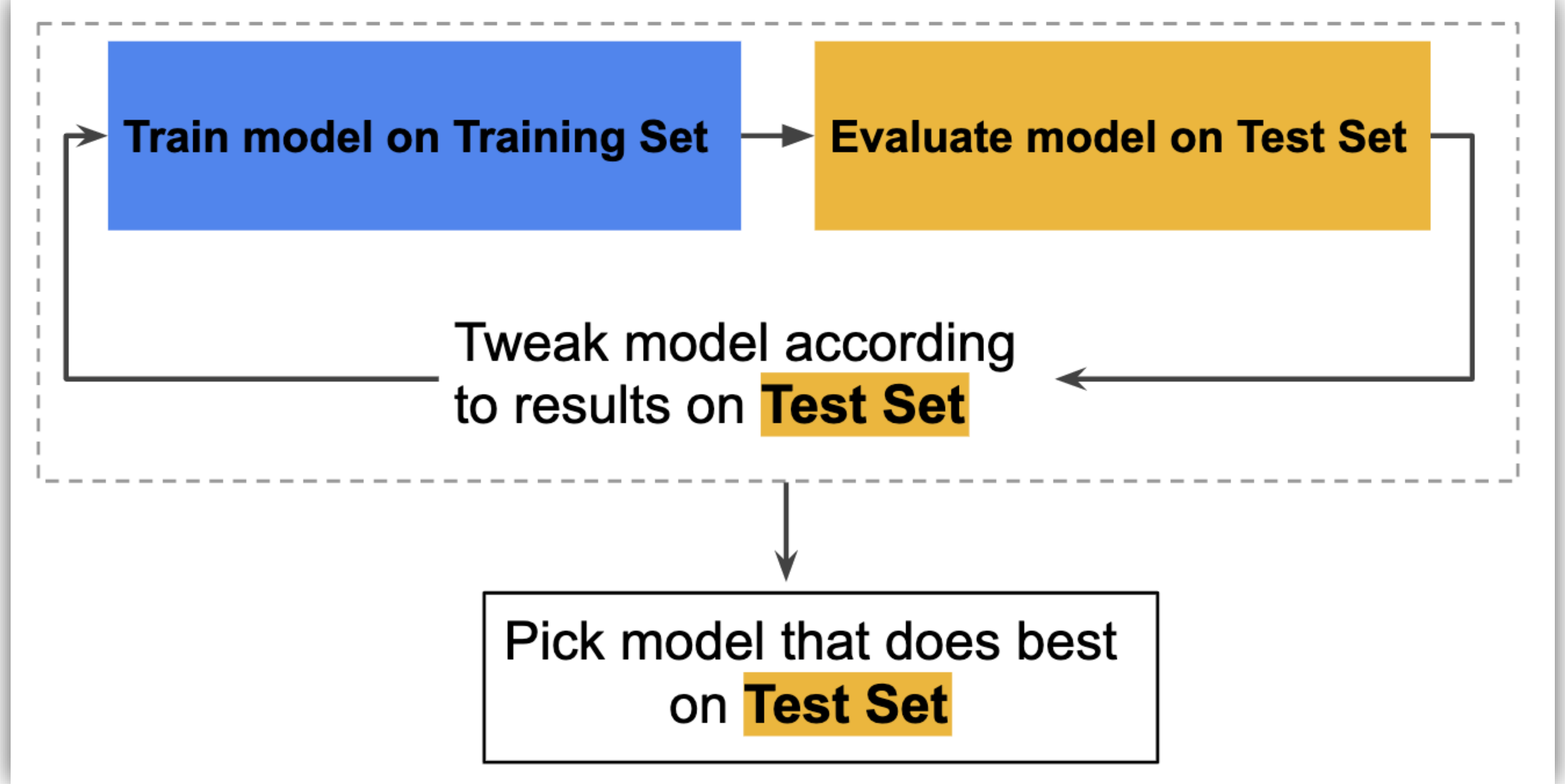
Overfitting occurs when a model tries to fit training data so closely that it does not generalize well to new data

Dataset Splitting

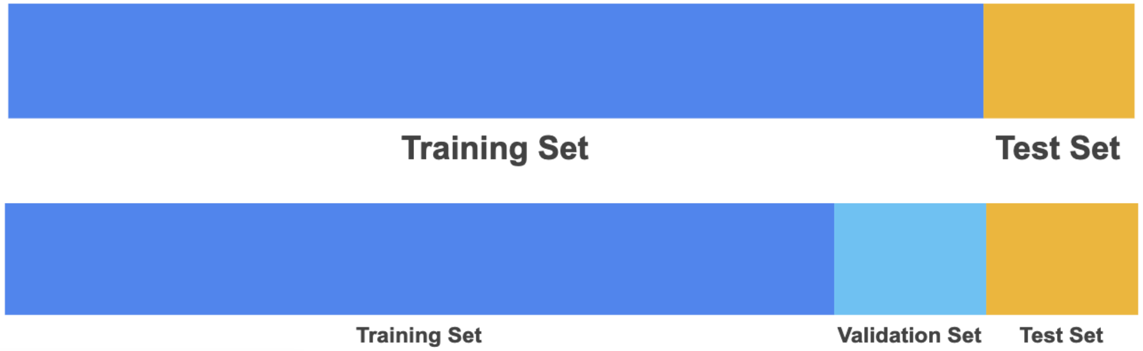
Training set - a subset to train a model
Testing set - a subset to test the trained model

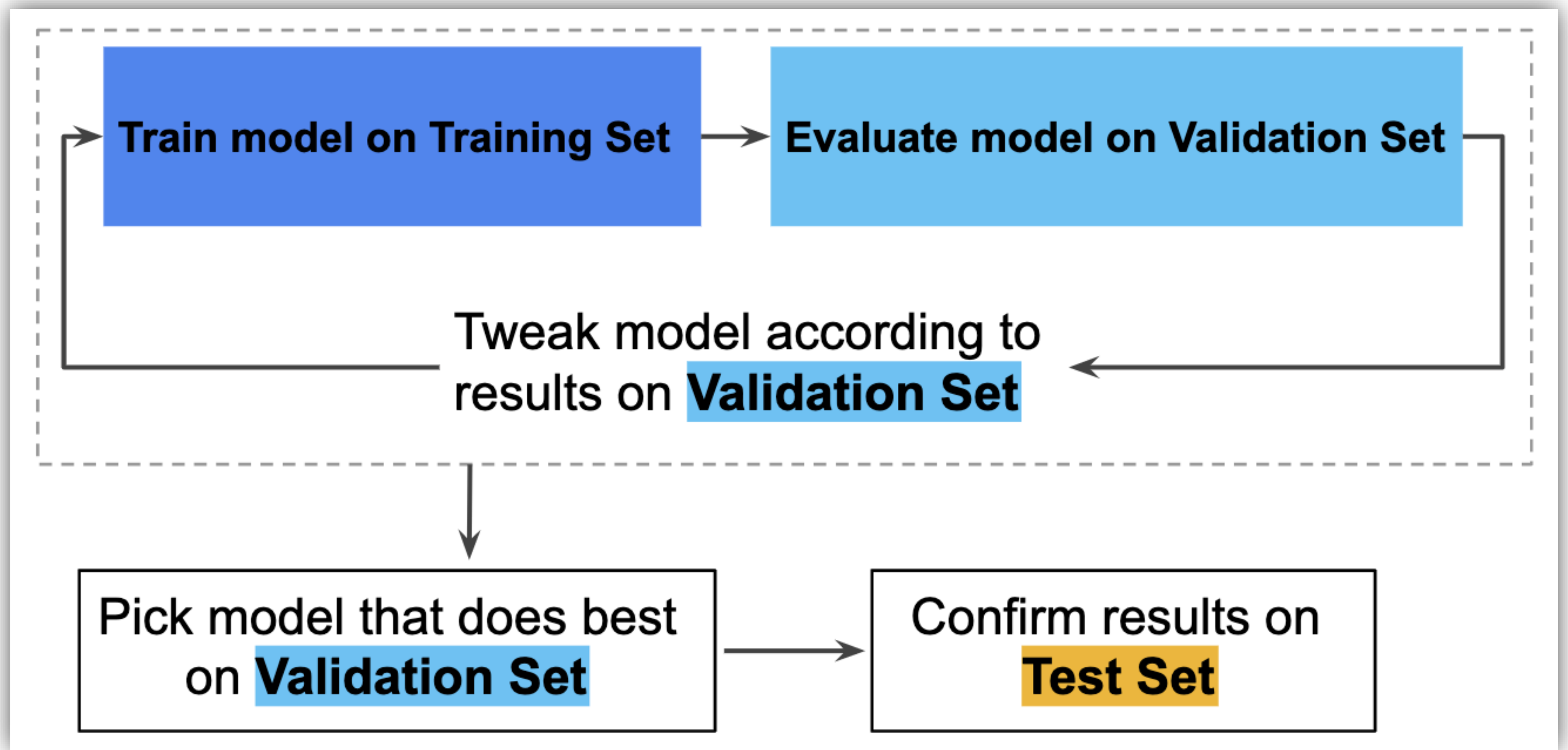

Training Data


Test Data



Training workflow





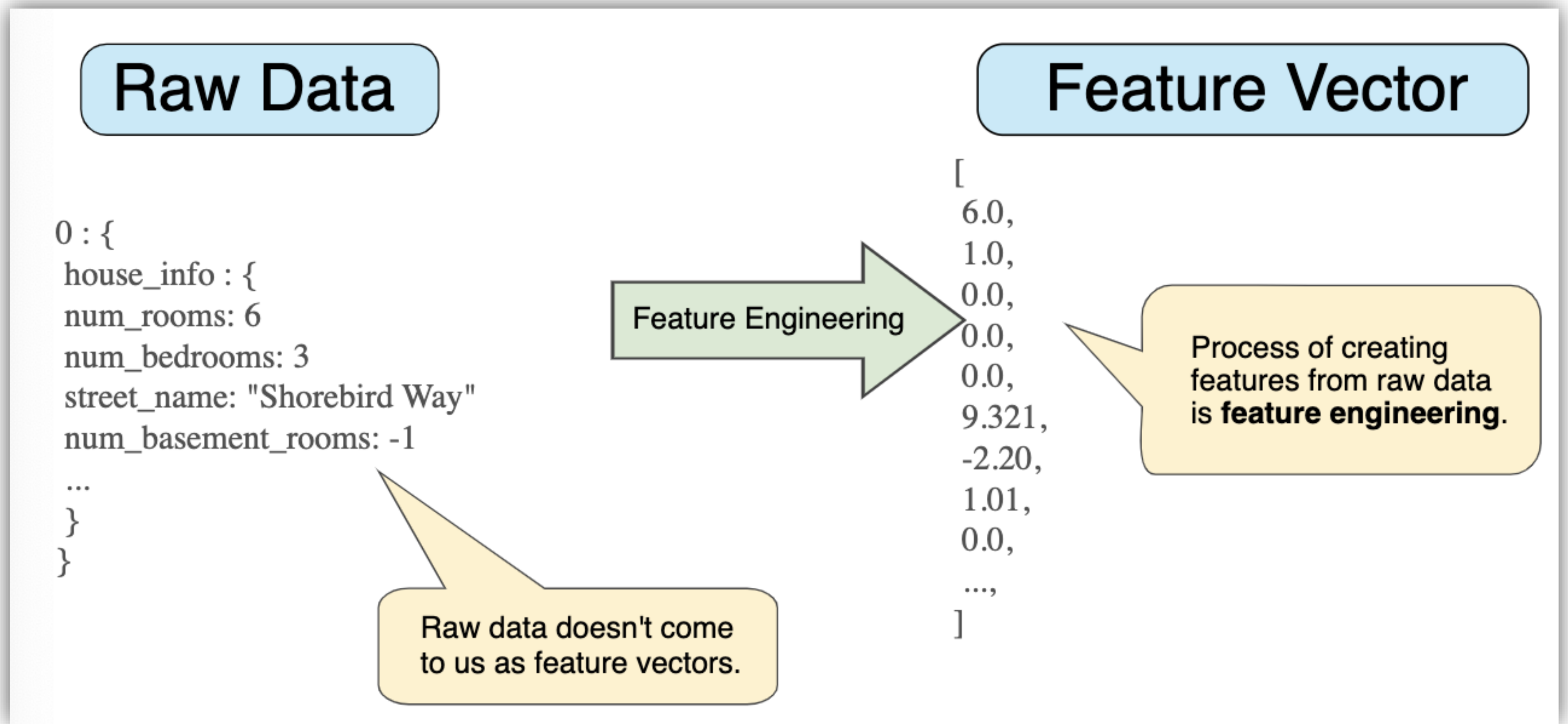
Training workflow

1. Pick the model that does best on the validation set.
2. Double-check that model against the test set.

Feature Engineering

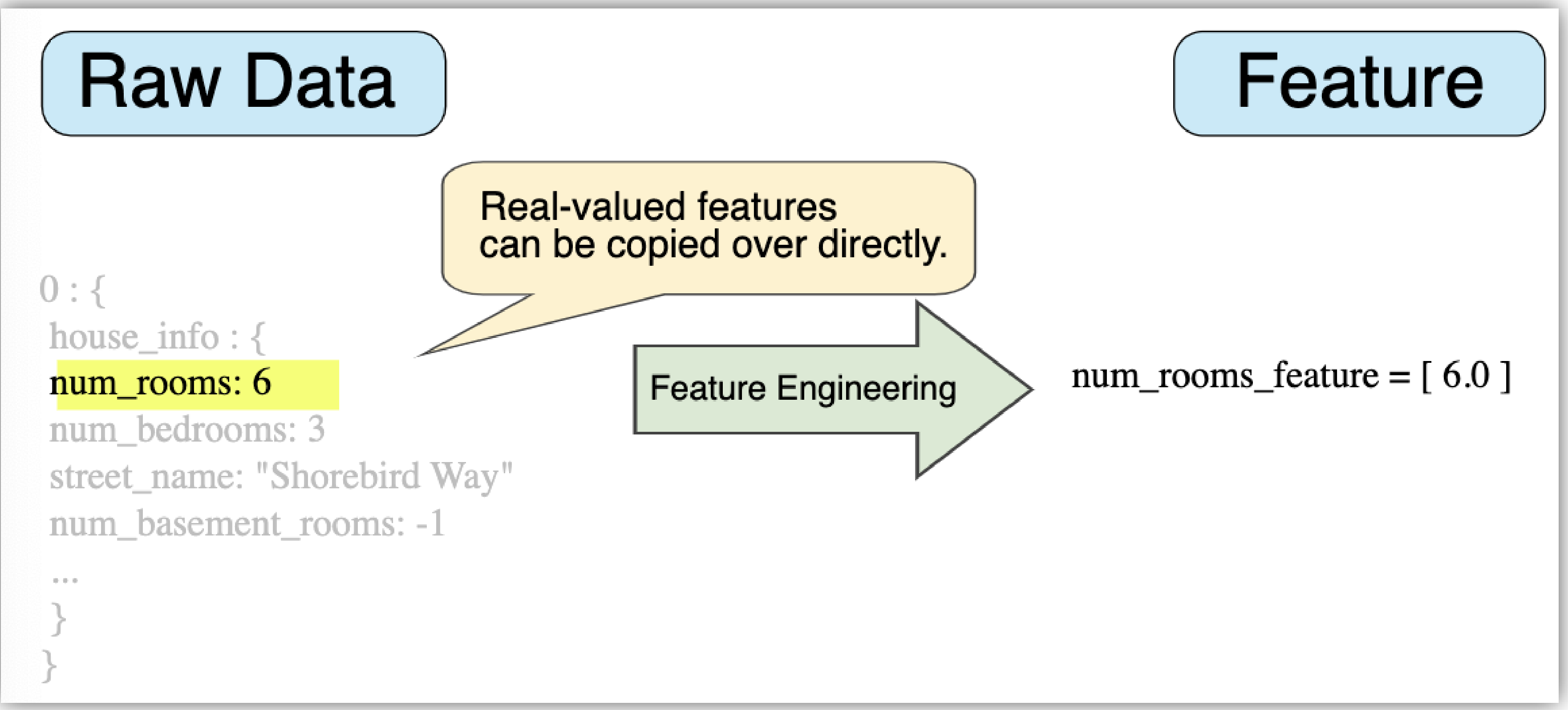
Feature engineering means transforming raw data into a feature vector

Mapping raw data into feature vectors



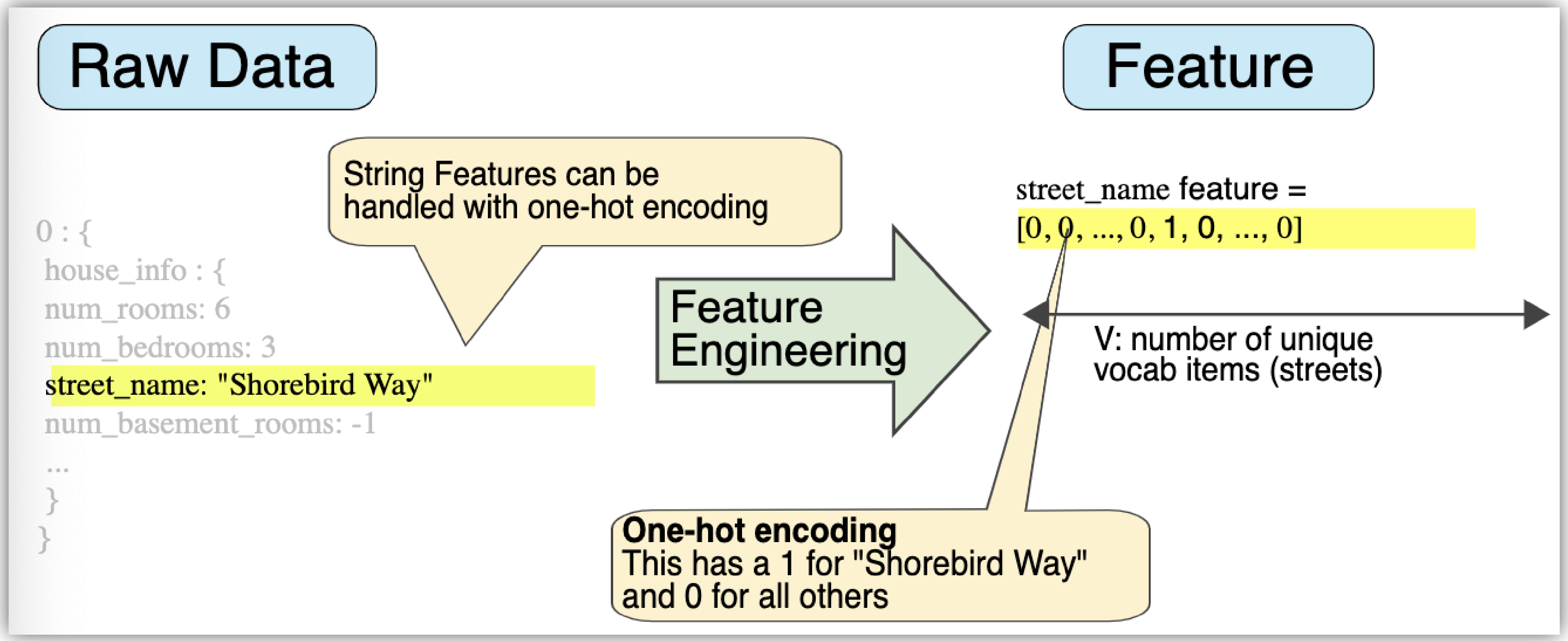
Mapping Numeric Values

Mapping integer and floating-point data don't need a special encoding



Mapping Categorical Values

Mapping categorical data need a **one-hot encoding**



Regularization

Penalize complex models to control and avoid overfitting

Empirical Risk Minimization

$$\text{minimize}(\text{Loss}(\text{Data}|\text{Model}))$$

Structural Risk Minimization

$$\text{minimize}(\text{Loss}(\text{Data}|\text{Model}) + \text{complexity}(\text{Model}))$$

Model complexity

Function of the ***weights*** of all the features in the model

Function of ***total number of features*** with nonzero weights

Regularization

Penalize complex models to control and avoid overfitting

$$L_2 \text{ regularization term} = ||\boldsymbol{w}||_2^2 = w_1^2 + w_2^2 + \ldots + w_n^2$$

$$\{w_1 = 0.2, w_2 = 0.5, w_3 = 5, w_4 = 1, w_5 = 0.25, w_6 = 0.75\}$$

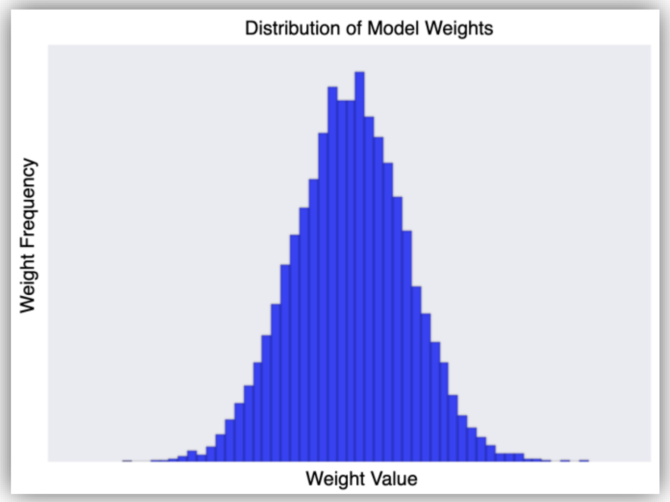
Regularization Lambda

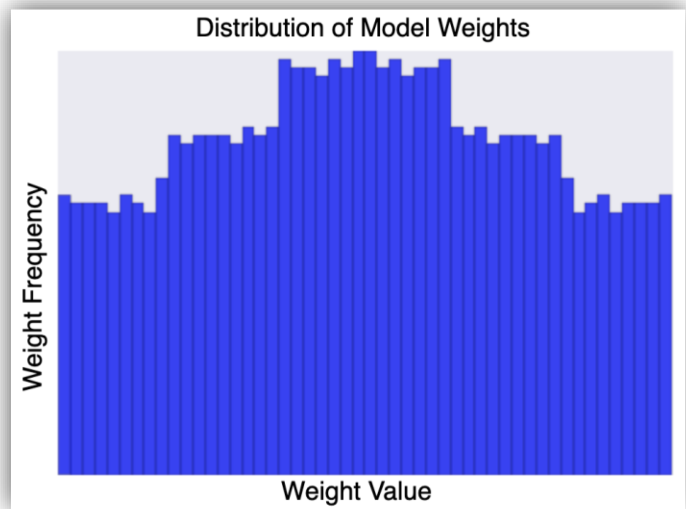
Penalize complex models to control and avoid overfitting

$$\text{minimize}(\text{Loss}(\text{Data}|\text{Model}) + \lambda\ \text{complexity}(\text{Model}))$$

If lambda value is too high, your model will be simple, but you run the risk of ***underfitting*** your data.

If your lambda value is too low, your model will be more complex, and you run the risk of ***overfitting*** your data.





Logistic Regression

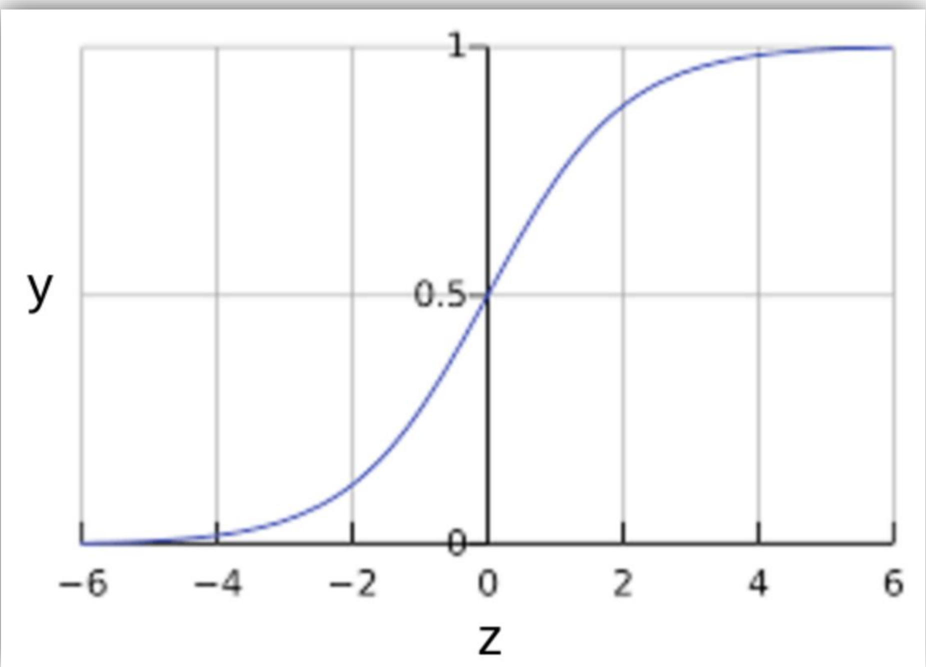
Calculating Probability using Sigmoid function

$$y = \frac{1}{1 + e^{-z}}$$

$$z = (b + w_1x_1 + w_2x_2 + \dots w_Nx_N)$$

bias and weights: $b = 1$ $w_1 = 2$ $w_2 = -1$ $w_3 = 5$

feature values: $x_1 = 0$ $x_2 = 10$ $x_3 = 2$



Loss function

Log loss function

$$\text{Log Loss} = \sum_{(x,y)\in D} -y\log(y') - (1-y)\log(1-y')$$

Two strategies to dampen model complexity:

- **L_2 regularization**
- **Early stopping** (limiting the number of training steps or the learning rate)

Evaluation Metrics

Classification Terminologies

An Aesop's Fable: The Boy Who Cried Wolf

A shepherd boy gets bored tending the town's flock. To have some fun, he cries out, "Wolf!" even though no wolf is in sight. The villagers run to protect the flock, but then get really mad when they realize the boy was playing a joke on them.

[Iterate previous paragraph *N* times.]

One night, the shepherd boy sees a real wolf approaching the flock and calls out, "Wolf!" The villagers refuse to be fooled again and stay in their houses. The hungry wolf turns the flock into lamb chops. The town goes hungry. Panic ensues.

True Positive (TP): • Reality: A wolf threatened. • Shepherd said: "Wolf." • Outcome: Shepherd is a hero.	**False Positive (FP):** • Reality: No wolf threatened. • Shepherd said: "Wolf." • Outcome: Villagers are angry at shepherd for waking them up.
False Negative (FN): • Reality: A wolf threatened. • Shepherd said: "No wolf." • Outcome: The wolf ate all the sheep.	**True Negative (TN):** • Reality: No wolf threatened. • Shepherd said: "No wolf." • Outcome: Everyone is fine.

Classification Terminologies

Accuracy

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

True Positive (TP): • Reality: Malignant • ML model predicted: Malignant • **Number of TP results: 1**	**False Positive (FP):** • Reality: Benign • ML model predicted: Malignant • **Number of FP results: 1**
False Negative (FN): • Reality: Malignant • ML model predicted: Benign • **Number of FN results: 8**	**True Negative (TN):** • Reality: Benign • ML model predicted: Benign • **Number of TN results: 90**

Classification Terminologies

True Positives (TPs): 1	False Positives (FPs): 1
False Negatives (FNs): 8	True Negatives (TNs): 90

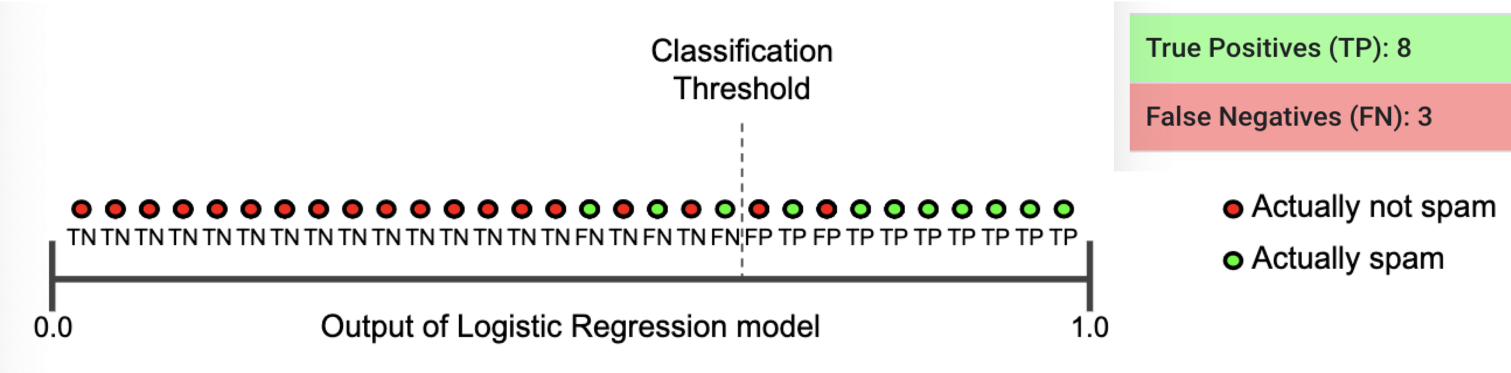
Precision - What proportion of positive identifications was actually correct?

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

Recall - What proportion of actual positives was identified correctly?

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

Precision and Recall: Tug of war



True Positives (TP): 8	False Positives (FP): 2
False Negatives (FN): 3	True Negatives (TN): 17

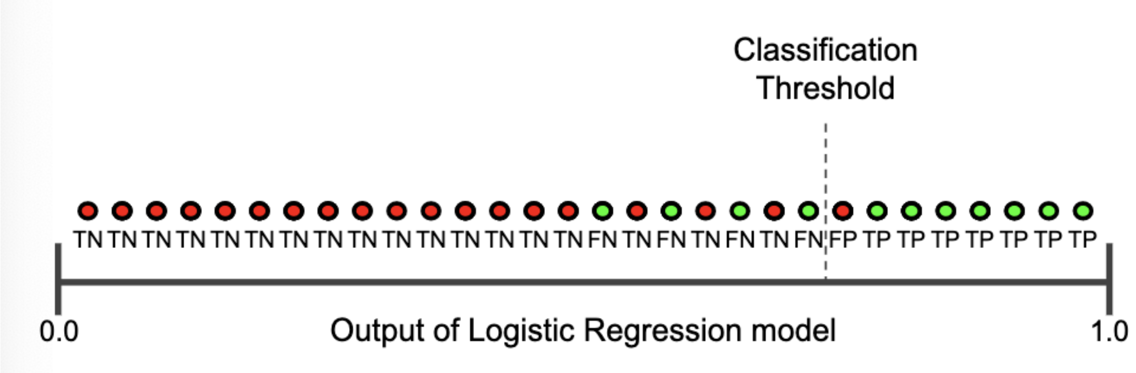
Precision - What proportion of positive identifications was actually correct?

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

Recall - What proportion of actual positives was identified correctly?

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

Precision and Recall: Tug of war



True Positives (TP): 7	False Positives (FP): 1
False Negatives (FN): 4	True Negatives (TN): 18

Actually not spam
Actually spam

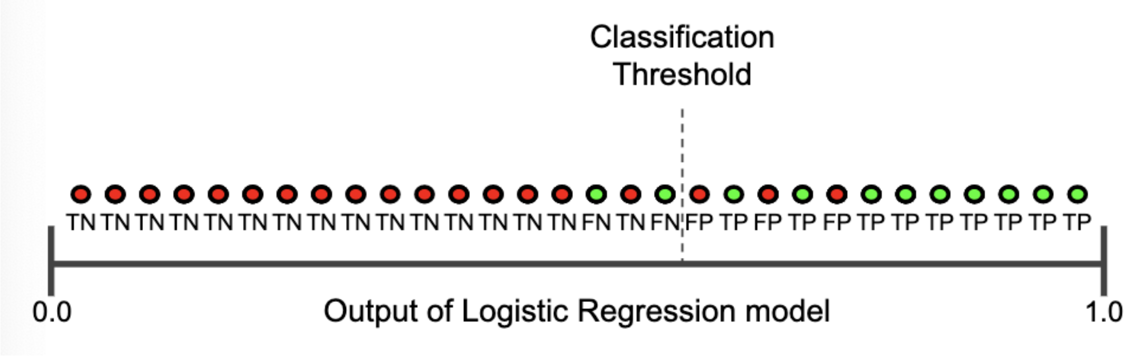
Precision - What proportion of positive identifications was actually correct?

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

Recall - What proportion of actual positives was identified correctly?

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

Precision and Recall: Tug of war



True Positives (TP): 9	False Positives (FP): 3
False Negatives (FN): 2	True Negatives (TN): 16

- Actually not spam
- Actually spam

Precision - What proportion of positive identifications was actually correct?

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

Recall - What proportion of actual positives was identified correctly?

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

ROC Curve

ROC Curve plots TPR vs FPR at different classification thresholds

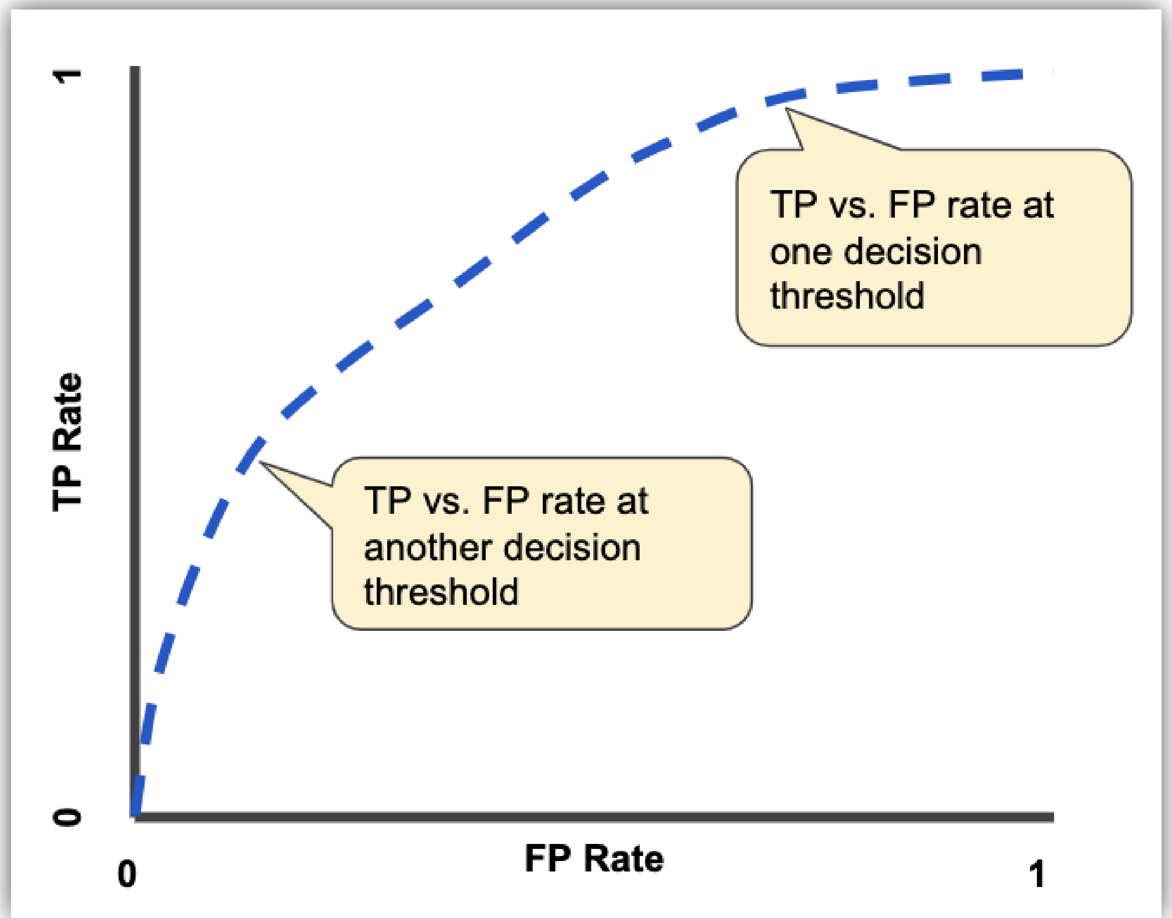
True Positive Rate (TPR) = Recall

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

$$TPR = \frac{TP}{TP + FN}$$

False Positive Rate (FPR)

$$FPR = \frac{FP}{FP + TN}$$



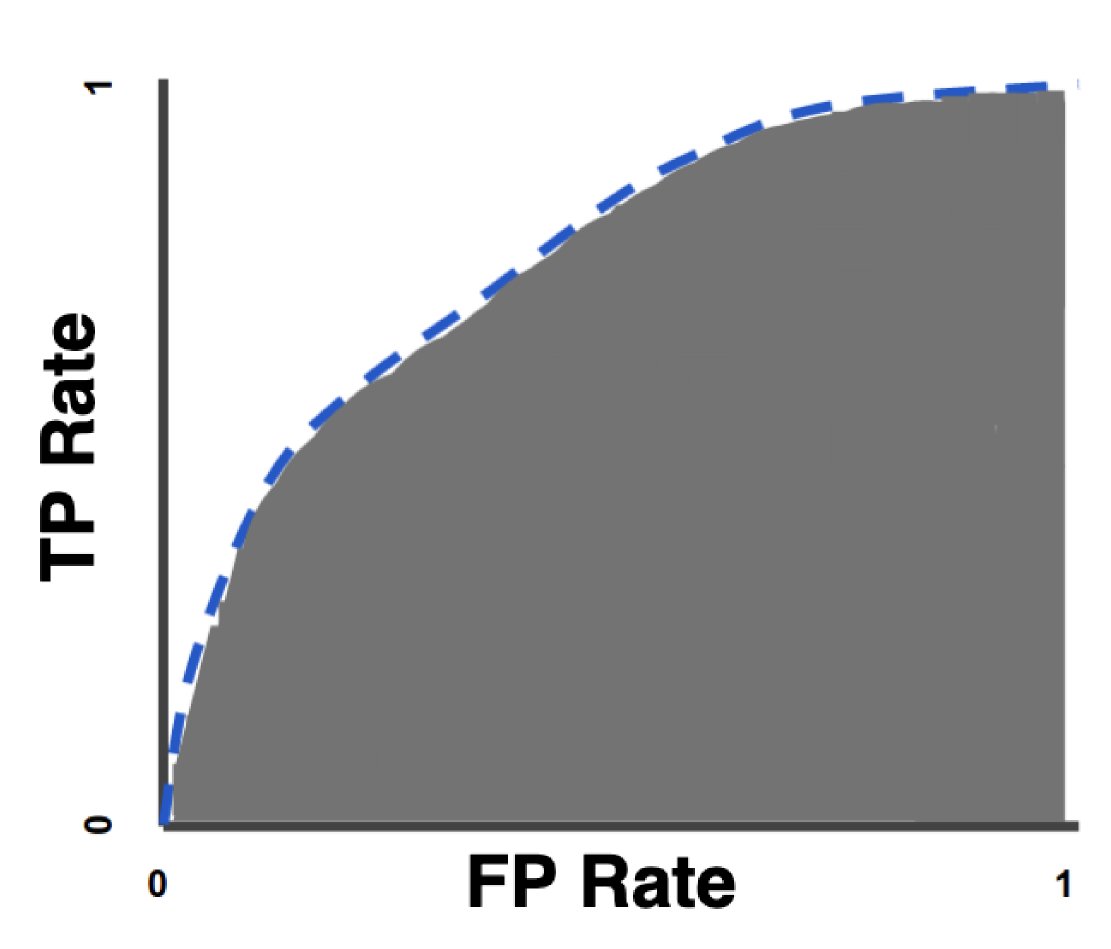
AUC: Area Under the ROC Curve

AUC measures the area underneath the entire ROC curve

AUC represents the probability that a random positive (green) example is positioned to the right of a random negative (red) example.
AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

AUC is desirable for the following two reasons:

- AUC is **scale-invariant**. It measures how well predictions are ranked, rather than their absolute values.
- AUC is **classification-threshold-invariant**. It measures the quality of the model's predictions irrespective of what classification threshold is chosen.



References

Images and concepts in the slides are used from the following resources

- Introduction to Machine Learning with Python *Andreas Mueller*
- Pattern Recognition and Machine Learning *Christopher Bishop*
- Machine Learning Course *Google*