

ARTIFICIAL INTELLIGENCE 501

Lesson 4

Supervised Learning

Legal Disclaimers

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at [intel.com](https://www.intel.com).

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others

Copyright © 2018 Intel Corporation. All rights reserved.

Learning Objectives

You will be able to:

- Name the steps in the data science workflow
- Explain how to formulate a supervised learning problem
- Compare “Training” and “Inference”
- Describe the dangers of overfitting, and training vs. testing data
- Identify the advantages of the PyData* ecosystem



REVIEW

Machine Learning

The study and construction of programs that learn from repeatedly seeing data, rather than being explicitly programmed by humans.

<u>Type</u>	<u>Dataset</u>
Supervised Learning	Data points have known outcome
Unsupervised Learning	Data points have unknown outcome

Target vs. Features

Target: Column to predict

Features: Properties of the data used for prediction (non-target columns)



sepal length	sepal width	petal length	petal width	species
6.7	3.0	5.2	2.3	virginica
6.4	2.8	5.6	2.1	virginica
4.6	3.4	1.4	0.3	setosa
6.9	3.1	4.9	1.5	versicolor
4.4	2.9	1.4	0.2	setosa
4.8	3.0	1.4	0.1	setosa
5.9	3.0	5.1	1.8	virginica
5.4	3.9	1.3	0.4	setosa
4.9	3.0	1.4	0.2	setosa
5.4	3.4	1.7	0.2	setosa

Example: Supervised Learning Problem

Goal: Predict if an email is spam or not spam.

Data: Historical emails labeled as spam or not spam.

Target: Spam or not spam

Features: Email text, subject, time sent, etc.



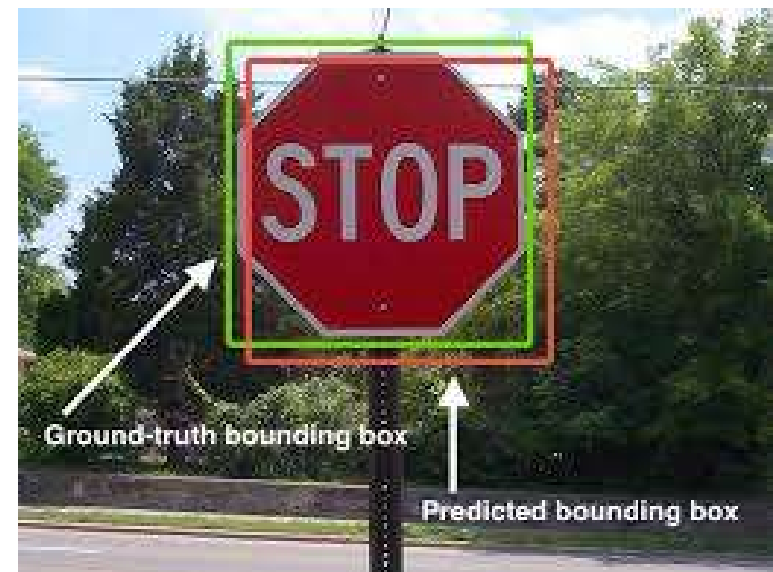
Example: Supervised Learning Problem

Goal: Predict location of bounding box around an object.

Data: Images with bounding box locations.

Target: Corners of bounding box

Features: Image pixels





DATA SCIENCE WORKFLOW

Data Science Workflow

Problem Statement

What problem are you trying to solve?

Data Collection

What data do you need to solve it?

Data Exploration & Preprocessing

How should you clean your data so your model can use it?

Modeling

Build a model to solve your problem?

Validation

Did I solve the problem?

Decision Making & Deployment

Communicate to stakeholders or put into production?



This Lesson's Focus: Modeling and Validation

Problem Statement

What problem are you trying to solve?

Data Collection

What data do you need to solve it?

Data Exploration & Preprocessing

How should you clean your data so your model can use it?

Modeling

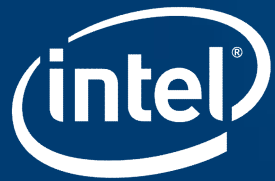
Build a model to solve your problem?

Validation

Did I solve the problem?

Decision Making & Deployment

Communicate to stakeholders or put into production?



SUPERVISED LEARNING

Formulating a Supervised Learning Problem

For a Supervised Learning Problem:

- Collect a labeled dataset (features and target labels).
- Choose the model.
- Choose an evaluation metric:
“What to use to measure performance.”
- Choose an optimization method:¹
“How to find the model configuration that gives the best performance.”

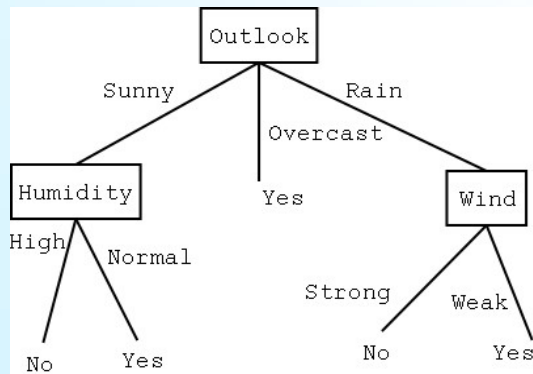
¹*There are standard methods to use for different models and metrics.*

Which Model?

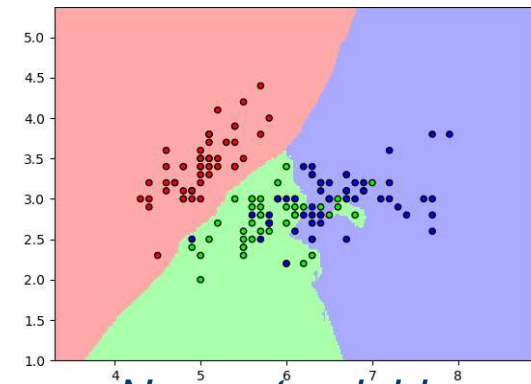
There are many models that represent the problem and make decisions in different ways each with their own advantages and disadvantages.

A **decision tree** makes predictions by asking a series of yes/no questions.

Nearest neighbor makes predictions by having the most similar examples vote.



Decision tree



Nearest neighbors

Which Model?

Some considerations when choosing are:

- Time needed for training
- Speed in making predictions
- Amount of data needed
- Type of data
- Problem complexity
- Ability to solve a complex problem
- Tendency to overcomplicate a simple one

Evaluation Metric

There are many metrics available¹ to measure performance, such as:

- **Accuracy:** how well predictions match true values.
- **Mean Squared Error:** average square distance between prediction and true value.

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

Mean square error formula



¹The wrong metric can be misleading or not capture the real problem.

Evaluation Metric

The wrong metric can be misleading or not capture the real problem.

For example: consider using **accuracy** for spam/not spam.

- If 99 out of 100 emails are actually spam, then a model that is predicting spam every time will have *99% accuracy*.
- This may force an important *real* email into spam, even though it has a high accuracy metric.



Email

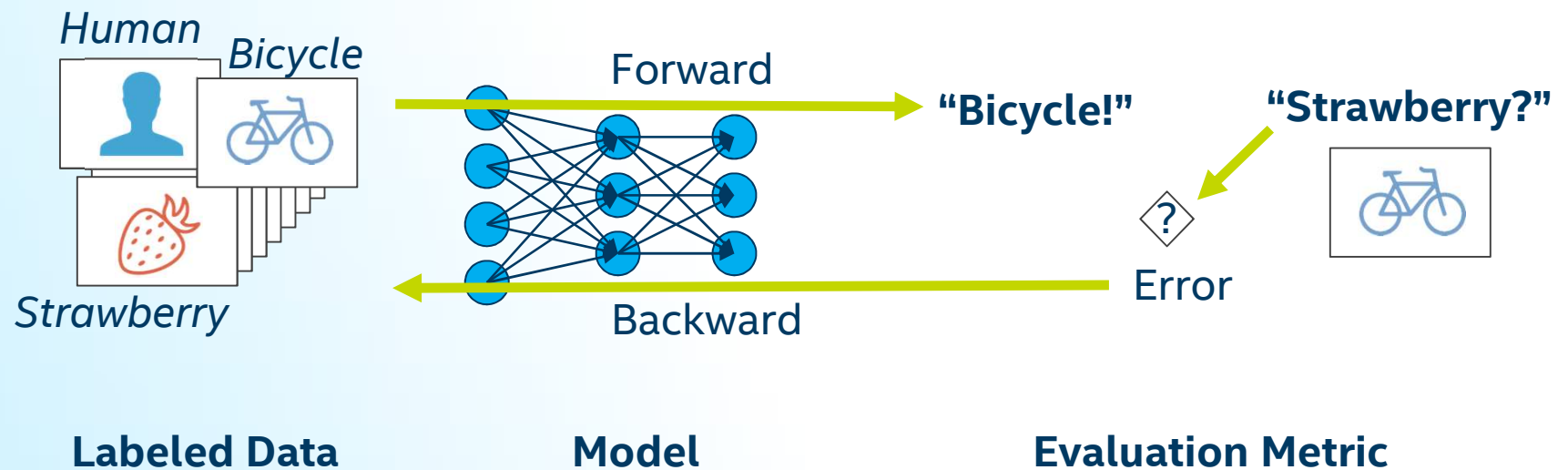
Training

Training Data: The dataset used to train the model.

Optimization: Configures the model for best performance.

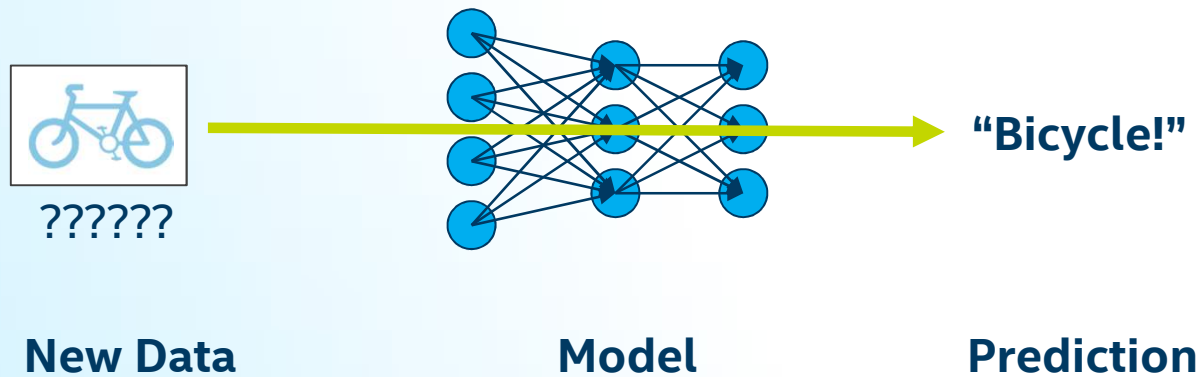
Training

With these pieces, a model can now be trained to find the best configuration.



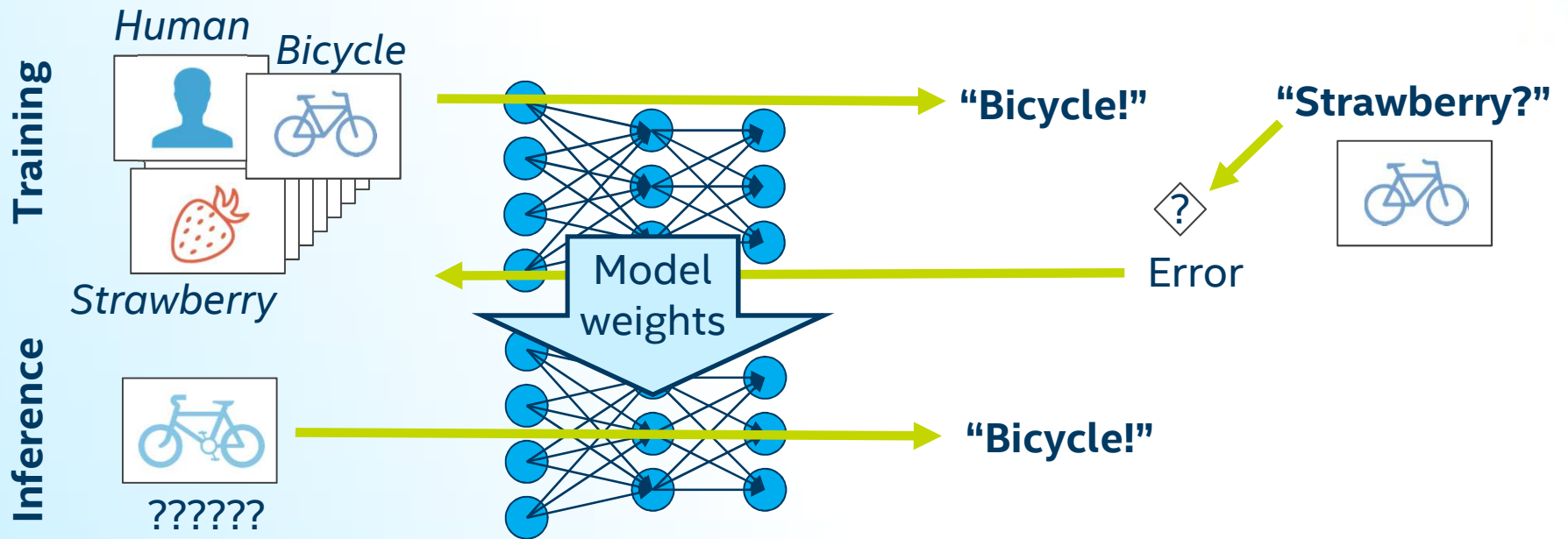
Inference

Once the model is trained, we can provide new examples for predictions.



Training vs. Inference

Goal: Perform well on unseen data during inference.



Supervised Learning Overview

Training: Train a model with known data.

**Data with answers
(features and labels)** + **Model**

Fit

**Trained
Model**

Inference: Feed unseen data into trained model to make predictions.

**Data without answers
(features only)** + **Trained
Model**

Predict

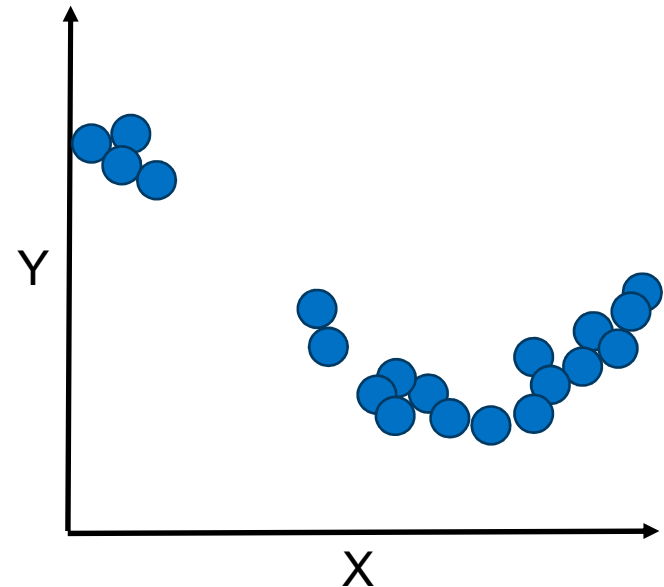
**Predicted
Answer**



OVERFITTING, TRAINING, AND TESTING DATA

Curve Fitting: Overfitting vs. Underfitting Example

Goal: Fit a curve to the data.

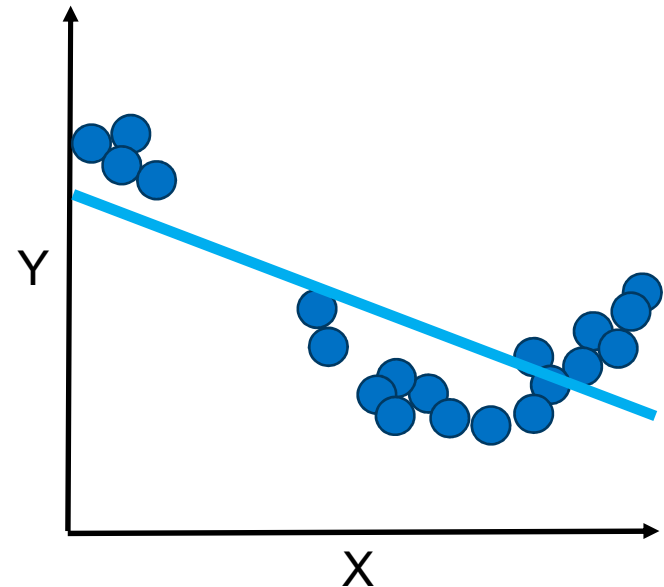


Curve Fitting: Underfitting Example

The curve can be too simple.

- This is called “underfitting”
- Poor fit on training data
- Poor fit on unseen data

Underfitting: Model is missing systematic trends in data.

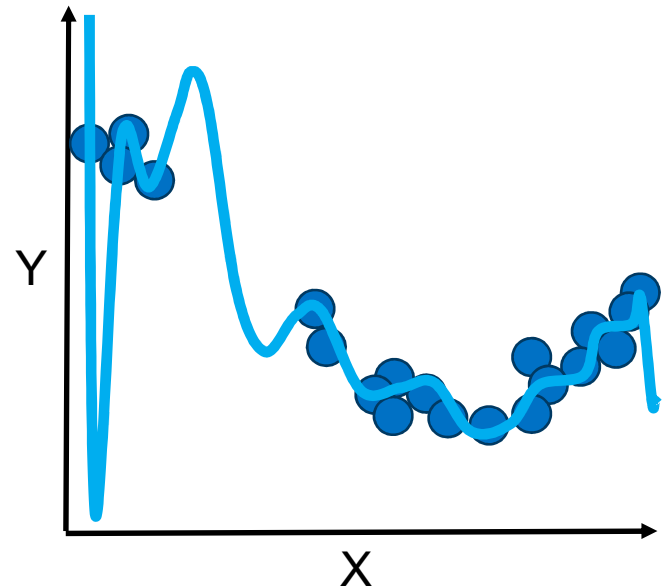


Curve Fitting: Overfitting Example

The curve can be too complex.

- This is called “overfitting”
- Good fit on training data
- Poor fit on unseen data

Overfitting: Model is too sensitive and fits the “noise” in the training data.

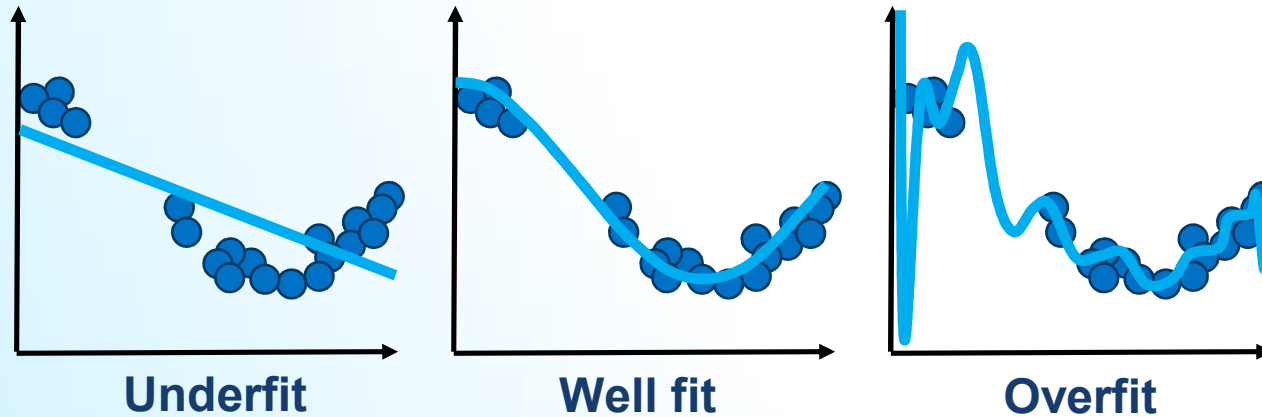


Curve Fitting Problem

Problem: Unseen data isn't available during training.

- How can performance be estimated?

When measuring performance on the training data, there is a tendency to overfit.



Solution: Split Data Into Two Sets

Training Set: Data used during the training process.

Test Set: Data used to measure performance, simulating unseen data¹.

sepal length	sepal width	petal length	petal width	species
6.7	3.0	5.2	2.3	virginica
6.4	2.8	5.6	2.1	virginica
4.6	3.4	1.4	0.3	setosa
6.9	3.1	4.9	1.5	versicolor
4.4	2.9	1.4	0.2	setosa
4.8	3.0	1.4	0.1	setosa
5.9	3.0	5.1	1.8	virginica
5.4	3.9	1.3	0.4	setosa
4.9	3.0	1.4	0.2	setosa
5.4	3.4	1.7	0.2	setosa

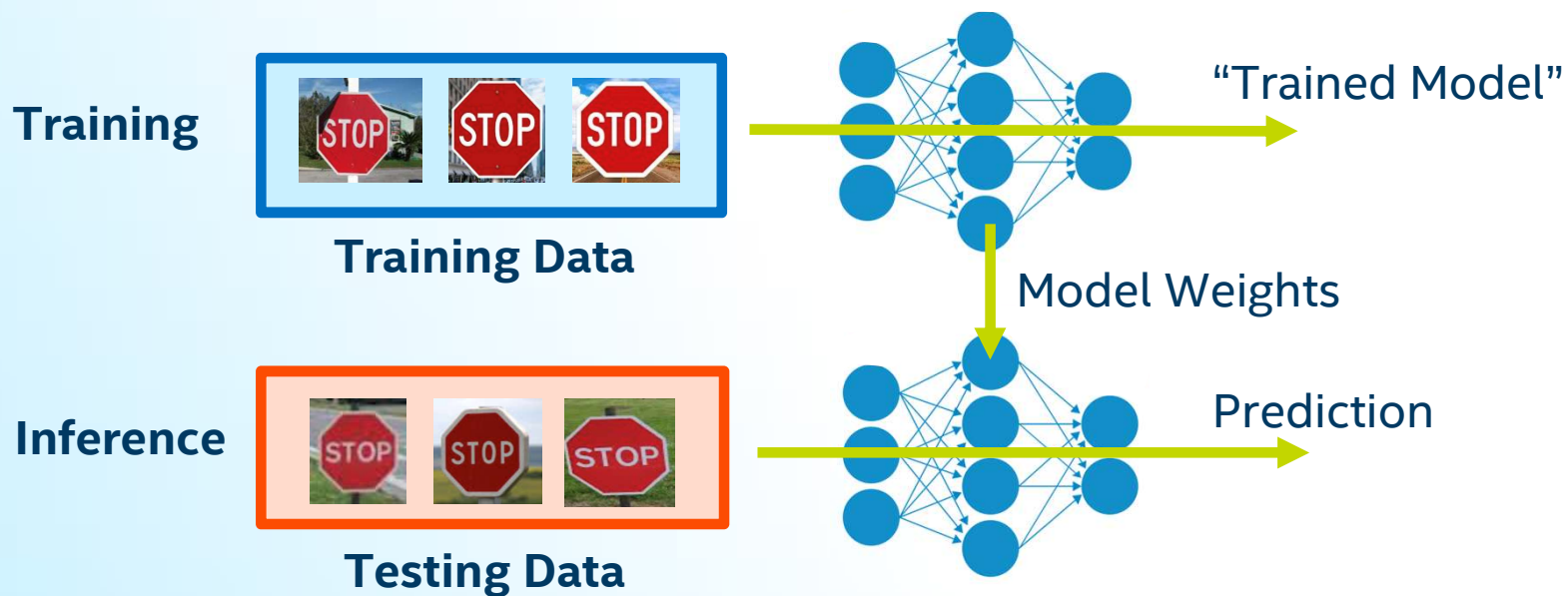
Training Set

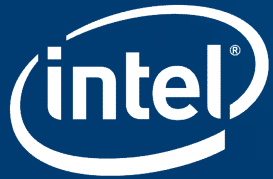
Testing Set

¹ Not used during the training process.

Train-Test Split

Evaluate trained model on data it hasn't "seen" before to simulate real-world inference.





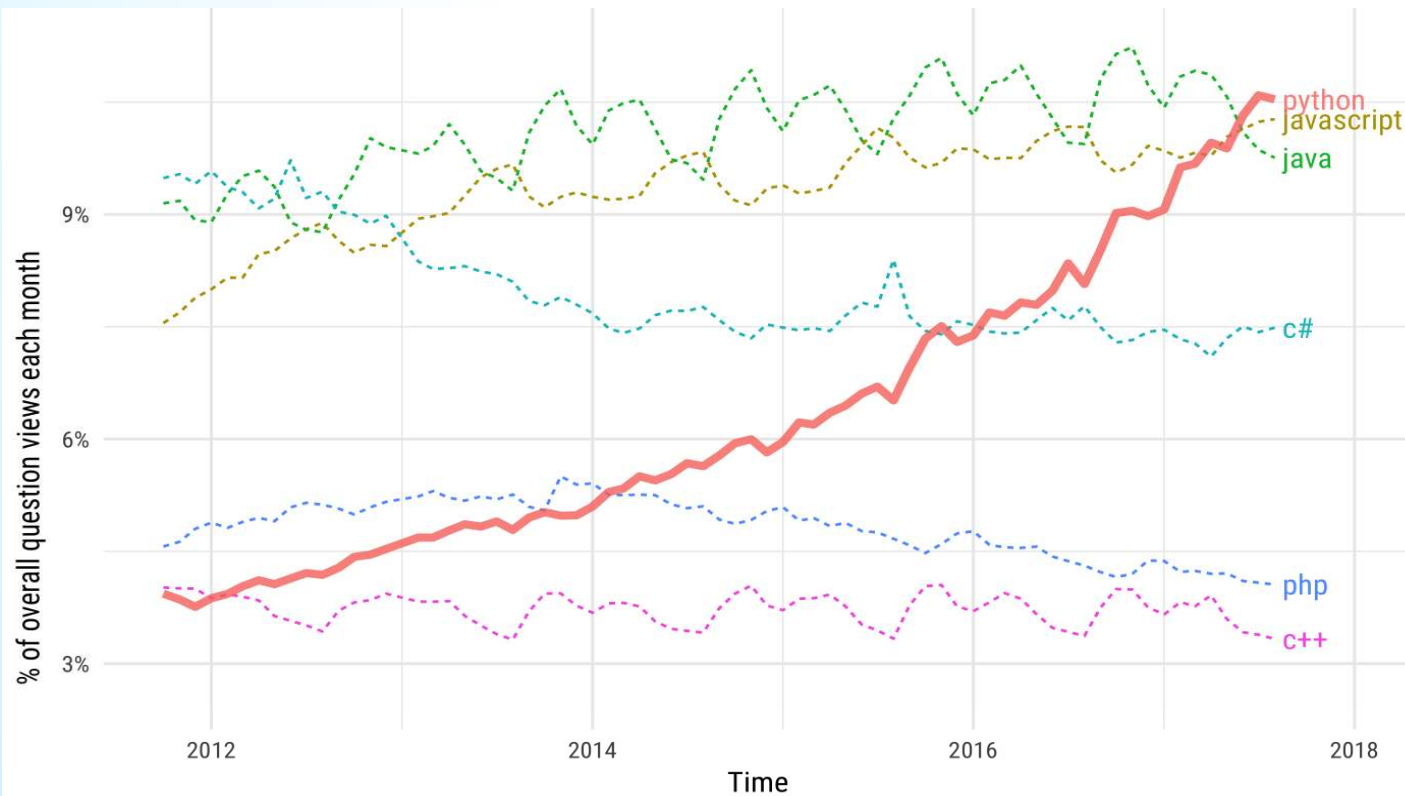
PYTHON* ECOSYSTEM

*Other names and brands may be claimed as the property of others.

Why Python*?

- General purpose language.
- Simple, readable syntax relative to other languages, such as Java* or C++.
- Has a good REPL.
- Can facilitate applications written in other languages, C++ and Fortran.
- Active community.
- Extensive libraries.

Python*: Fastest Growing Programming Language¹











¹Source:
Stack Overflow

*Other names and brands may be claimed as the property of others.



Python*: Highest Ranked Language¹

Language Rank	Types	Spectrum Ranking
1. Python	 	100.0
2. C	  	99.7
3. Java	  	99.5
4. C++	  	97.1
5. C#	  	87.7
6. R		87.7
7. JavaScript	 	85.6
8. PHP		81.2
9. Go	 	75.1
10. Swift	 	73.7

¹Source:
IEEE Spectrum

*Other names and brands may be claimed as the property of others.



Python* Libraries for Data Science



Fast numerical computing



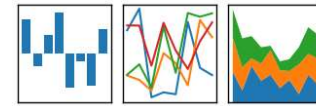
Machine learning



Deep Learning

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Data analysis and manipulation

matplotlib

Visualization

*Other names and brands may be claimed as the property of others.



Intel® Distribution for Python*

Easy, Out-of-the-box Access to High Performance Python*

- Prebuilt, optimized for numerical computing, data analytics, High Performance Computing (HPC).
- Drop in replacement for your existing Python (no code changes required).
- Download from <https://software.intel.com/en-us/distribution-for-python>

*Other names and brands may be claimed as the property of others.



Intel® Distribution for Python*

Drive Performance with Multiple Optimization Techniques

- Accelerated NumPy/ SciPy / Scikit-Learn with the Intel® Math Kernel Library (Intel® MKL).
- Data analytics with pyDAAL, enhanced thread scheduling with Intel® Thread Building Blocks (Intel® TBB), Jupyter* Notebook interface, Numba, Cython*.
- Scale easily with optimized MPI4Py* and Jupyter* notebooks.

*Other names and brands may be claimed as the property of others.



Intel® Distribution for Python*

Faster Access to Latest Optimizations for Intel® Architecture

- Distribution and individual optimized packages available through Conda* and Anaconda Cloud*.
- Optimizations upstreamed back to main Python* trunk.

*Other names and brands may be claimed as the property of others.



Train and Test Splitting: Syntax

Import the train and test split function.

```
from sklearn.model_selection import train_test_split
```

Split the data and put 30% into the test set.

```
train, test = train_test_split(data, test_size=0.3)
```

Other method for splitting data.

```
from sklearn.model_selection import ShuffleSplit
```

K Nearest Neighbors: The Syntax

Import the class containing the classification method.

```
from sklearn.neighbors import KNeighborsClassifier
```

Create an instance of the class.

```
KNN = KNeighborsClassifier(n_neighbors=3)
```

Fit the instance on the data and then predict the expected value.

```
KNN = KNN fit(X_data, y_data)  
y_predict = KNN predict(X_data)
```

Learning Objectives Recap

In this lesson, we worked to:

- Name the steps in the data science workflow
- Explain how to formulate a supervised learning problem
- Compare “Training” and “Inference”
- Describe the dangers of overfitting, and training vs. testing data
- Identify the advantages of the PyData* ecosystem

