ML, Data Analysis, Simulation, Large Code Projects

Machine Learning

Overview

- In data analysis and simulation, we designed algorithms and models to explore specific datasets.
- Instead of designing a model for a dataset, we apply a general machine learning algorithm to the data. This identifies patterns in the data and generates the model automatically.
- Three ways: supervised learning, unsupervised learning, and reinforcement learning.

Machine Learning

Supervised learning

- **Supervised learning** algorithms train models to predict outputs based on inputs. A data entry in the dataset is the input, and a label or a score is the output.
- Once the model is fully trained, it can be used to make predictions about unlabeled data.
- E.g. You're trying to predict a correct answer (label) which is known by the teacher. This means the teacher (ML algorithm) can check your work. Later, you'll apply your knowledge to new problems where the answer is unknown.

Machine Learning

Unsupervised learning

- **Unsupervised learning** algorithms train models that infer the natural structure of a dataset.
- Unsupervised learning is often used when the data is not labeled – **no true answer is known**. That means that a human being will need to look over the model's results to see if they make sense or seem random.
- E.g. identify common categories of favorite ice cream flavors by grouping together people who have similar desert preferences.

Machine Learning

Reinforcement learning

- **Reinforcement learning** algorithms train models that help an artificial intelligence agent approach an overall goal. Usually this involves taking a sequence of actions to bring the agent closer to the endpoint.
- E.g. A guessing game where the person with the secret word tells you if your guesses are hot or cold. You adjust your guesses and questions based on the person's feedback until you've gotten to the correct word.

Machine Learning

Reasoning Algorithms

- For supervised learning, common reasoning approaches include classification and regression algorithms.
- **Classification**: Categorical result (discrete)
- **Regression**: Numerical result (continuous)

- For unsupervised learning, common reasoning approaches include clustering.
- **Clustering**: A clustering algorithm takes a set of data and produces a model that breaks the data into some number of clusters by identifying data entries that are similar to each other. (We don't know the actual result.)

Machine Learning

Reasoning Algorithms

- ML Process: Decide, Train, then Test
- To apply machine learning to a supervised task, follow a simple process
- First: **decide** which learning algorithm you'll use and which features you'll train on. Make sure the algorithm matches the feature types!
- Second: **train** a model created by the algorithm by providing it with the data. The algorithm will 'learn' from the data the same way a student learns by going over worked examples.
- Third: **test** the model on a different set of data. This helps determine how accurate the model actually is.

Machine Learning

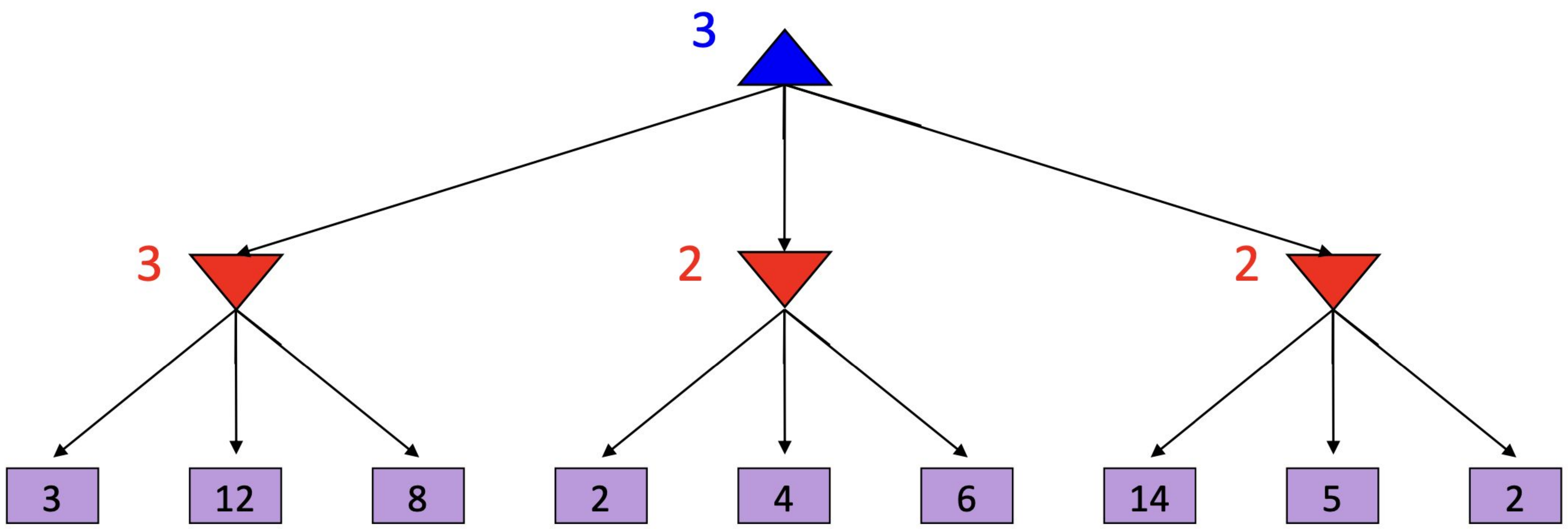
Testing Machine Learning Models

- The **training data** is normally composed of the majority (maybe 70%) of the available dataset. This data is run through the machine learning algorithm to produce the model. The more training data there is, the more accurate the algorithm's model becomes.
- To detect and remove parameters in the model that cause overfitting, you can use **validation data**. This is a subset of the data (maybe 15%) that is not used when training the model. Instead, it will be used to validate the model during training.
- When the programmer thinks they've achieved an optimal model, the **testing data** is used to determine how accurate that model actually is. This is a portion of the data (maybe 15%) that was set aside at the beginning and never used during the training or validation process. Unlike the validation data, which is evaluated multiple times, the model is run on the test data only once. We measure how close the predicted results are to the actual results. That score is the accuracy of the model.

Machine Learning

AI agents

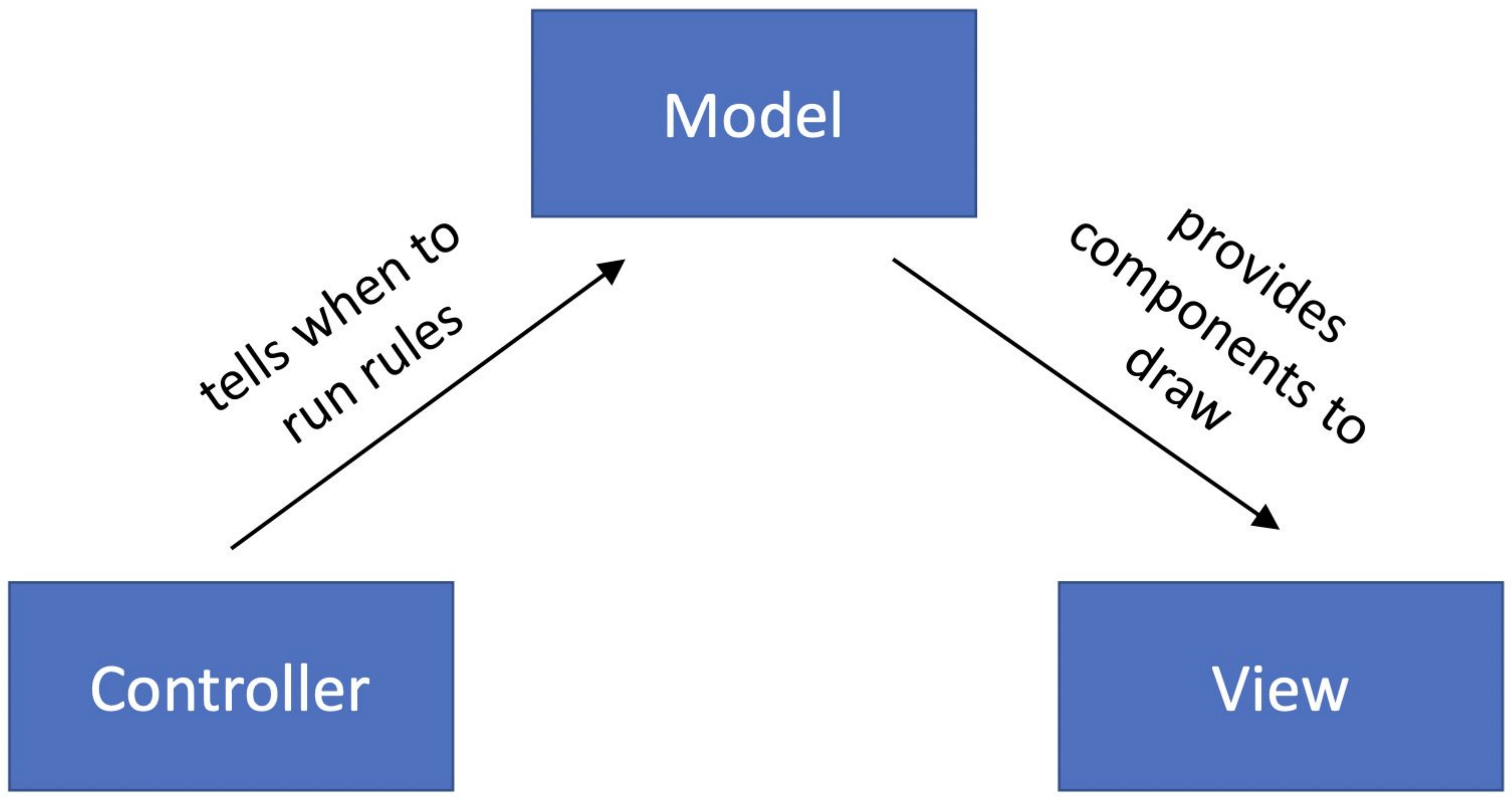
- An AI agent attempts to reach its goal by cycling through three steps: **perceive** information, **reason** about it, then **act** on it.
- Minimax:



Simulation

MVC overview

- MVC — Model, View, Controller



Simulation

MVC — Model

- **Model**: manages the components and rules of the thing we're simulating

- We can represent the model's components in code in a dictionary called data.
- E.g. Information about a circle:
- data["x"] = 200
- data["y"] = 200
- data["r"] = 50

Simulation

MVC — View

- **View**: displays the data in the model so that the user can look at it

- E.g.
- canvas.create_oval(data["x"] - data["r"], data["y"] - data["r"], data["x"] + data["r"], data["y"] + data["r"])

Simulation

MVC — Controller

- **Controller:**
  - **Time controller & Event controller**
  - **Updates the model**

- E.g. in timeLoop, we set

  data[“cx”] = data[“cx”] + 5

Simulation

Interaction events

- def mousePressed(data, event):
- … (How to get the coordinate?)

- def keyPressed(data, event):
- … (How to get the pressed key?)

Simulation

Experiments and trials

- import random

- random.random()   # pick a random float between 0-1
- random.randint(x, y) # pick a random number in a range
- random.choice(lst)   # chooses an element randomly
- random.shuffle(lst)   # destructively shuffles the list

Simulation

Experiments and trials

- Monte Carlo methods

- E.g. Expected value of the sum of rolling three dices
- Is this correct?

```python
import random

def runTrial():
    return 3 * random.randint(1, 6)

def getExpectedValue(numTrials):
    sum = 0
    for trial in range(numTrials):
        sum += runTrial()
    return sum / numTrials
```

Simulation

Experiments and trials

- Monte Carlo methods

- E.g. Expected value of the sum of rolling three dices

```
import random

def runTrial():
    return random.randint(1, 6) + random.randint(1, 6) + random.randint(1, 6)

def getExpectedValue(numTrials):
    sum = 0
    for trial in range(numTrials):
        sum += runTrial()
    return sum / numTrials
```

Simulation

Experiments and trials

- Monte Carlo methods

```
import random

def runTrial():
    return random.randint(1, 6) + random.randint(1, 6) + random.randint(1, 6)

def getExpectedValue(numTrials):
    sum = 0
    for trial in range(numTrials):
        sum += runTrial()
    return sum / numTrials
```

- print(getExpectedValue(10000000)) — Law of Large Numbers

Data Analysis

Modeling and Parsing

Data Analysis is the process of using computational or statistical methods to **gain insights** about data.

Data is complicated and messy.

We'll focus mainly on three steps: Data Cleaning, Exploration & Visualization, and Statistics & Analysis

Data Analysis

Three Types of Data:

Categorical: Data fall into one of several categories. Those categories are separate and cannot be compared. Example: style of house (ranch, split-level, two-story, duplex, Victorian, etc.)

Ordinal: Data fall into separate categories, but those categories can be compared – they have a specific order. Example: what is the condition of the house? (poor, fair, good, excellent, new)

Numerical: Data are numbers. We can perform mathematical operations on it and compare it to other data. Example: how large is the house in square feet?

Data Analysis

Data Cleaning

Data Cleaning is the process of taking raw data and smoothing out all these differences. It can be partially automated but usually requires some level of human intervention.

Irregularities:

E.g.

the header line

the blank lines

capitalized & uncapitalized

punctuation marks

misspelled words

……

	Flavor 1	Flavor 2	Flavor 3
1			
2	green tea	strawberry	cookies and cream
3	Jasmine Milk Tea	Vietnamese Coffee	Thai Tea
4	Mint Chocolate Chip	Rocky Road	Chocolate
5	Vanilla	Strawberry	Cookies and Cream
6	Vanilla	Coffee	Pistachio
7	Coffee!	Mint chip	birthday cake BATTER (try th
8			
9	grapenut	Peppermint stick	Chocolate
10	Chunky Monkey	Mint Chocolate Chip	Coffee
11	Yam	Vanilla	Oreo

Data Analysis

When reading data from a file, you need to determine what the **structure** of that data is.
That will inform how you store the data in Python.

E.g. CSV, JSON and plaintext structures

Data Analysis

CSV

Comma-Separated Values (CSV)

```
,Flavor 1,Flavor 2,Flavor 3¶
1,,,¶
2,green tea,strawberry,cookies and cream¶
3,Jasmine Milk Tea,Vietnamese Coffee,Thai Tea¶
4,Mint Chocolate Chip,Rocky Road,Chocolate¶
5,Vanilla,Strawberry,Cookies and Cream¶
6,Vanilla,Coffee,Pistachio¶
7,Coffee!,Mint chip,birthday cake BATTER (try t
8,,,¶
9,grapenut,Peppermint stick,Chocolate¶
10,Chunky Monkey,Mint Chocolate Chip,Coffee¶
11,Yam,Vanilla,Oreo¶
12,cherry,Matcha,Chocolate¶
13,Strawberry,Vanilla,chocolate chip¶
14,dulce de leche,Vanilla,Coffee¶
15,Vanilla,Banana,Strawberry¶
16,Cookie Dough,Cookies and Cream,Triple Fudge
17,Vanilla,Mocha,Strawberry¶
18,Butter Pecan,Cotton Candy,Mango¶
19,Turtle,Cookies and Cream,Vanilla¶
```

These files don't always have to use commas as separators, but they do need a delimiter to separate values (maybe spaces or tabs).

import the csv library and read:

```
import csv

f = open("icecream.csv", "r")
reader = csv.reader(f)

data = list(reader)
print(data)

f.close()
```

Data Analysis

CSV

You can also make(write) data into CSV format:

```
import csv

data = [[ "chocolate", "mint chocolate",
           "peppermint" ],
        [ "vanilla", "matcha", "coffee" ],
        [ "strawberry", "mango", "cherry" ]]

f = open("results.csv", "w", newline="")
writer = csv.writer(f)

writer.writerows(data)

f.close()
```

Data Analysis

JSON

JavaScript Object Notation (JSON) files store data that is nested, like **trees**.

```
{
  "vanilla" : 10,
  "chocolate" : {
                  "chocolate" : 15,
                  "chocolate chip" : 7,
                  "mint chocolate chip" : 5
                },
  "other" : [ "strawberry", "matcha", "coffee" ]
}
```

Data Analysis

JSON

JavaScript Object Notation (JSON) files store data that is nested, like **trees**.

We can read and write after importing json library.

```
import json

f = open("icecream.json", "r")
j = json.load(f)

print(j)

f.close()
```

```
import json

d = { "vanilla" : 10,
      "chocolate" : 27,
      "other" : [ "strawberry", "matcha", "coffee" ]
    }

f = open("results.json", "w")
json.dump(d, f)
f.close()
```

Data Analysis

Plaintext Data

What about plaintext data?

Identify the **patterns** and parse!

We can use the string library functions.

Data Analysis

Plaintext Data

Strings:

s[start:end:step]

s.split("\n") s.splitlines() s.index("j")   s.strip()   s.lower()   s.islower() …

Lists:

l.pop(2)   l.append("hahah") l.insert(2, "final")   l.remove("hahah") …

Data Analysis

Analysis and Visualization

```
import statistics
data = [41, 65, 64, 50, 45, 13, 29, 14, 7, 14]

mean = statistics.mean(data) # 34.2
median = statistics.median(data) # 35.0
mode = statistics.mode(data) # 14
```

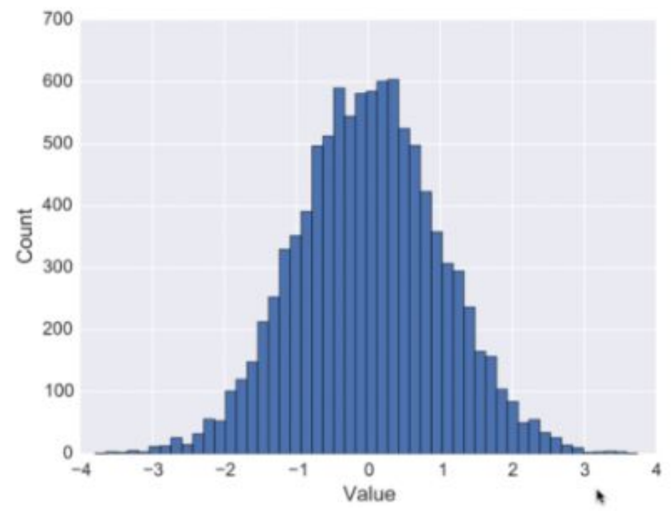
Data Analysis

Analysis and Visualization

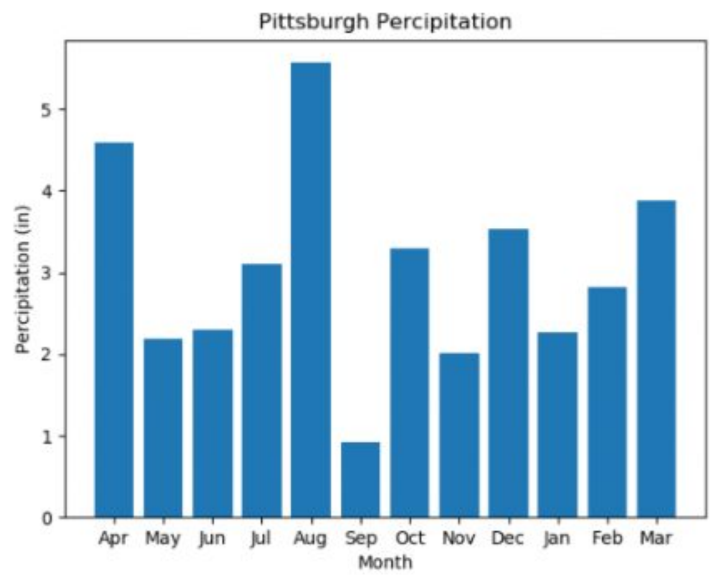
In order to choose the best visualization for the job, consider the **type** of the data you're presenting (categorical, ordinal, or numerical), and how many **dimensions** of data you need to visualize.

One-dimensional:

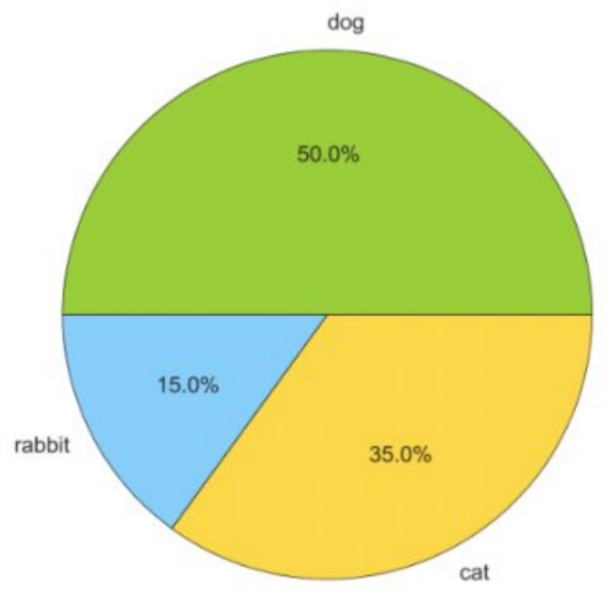
To visualize **numerical** data, use a **histogram**.



To visualize **ordinal** data, use a **bar chart**.



To visualize **categorical** data, use a **pie chart**.



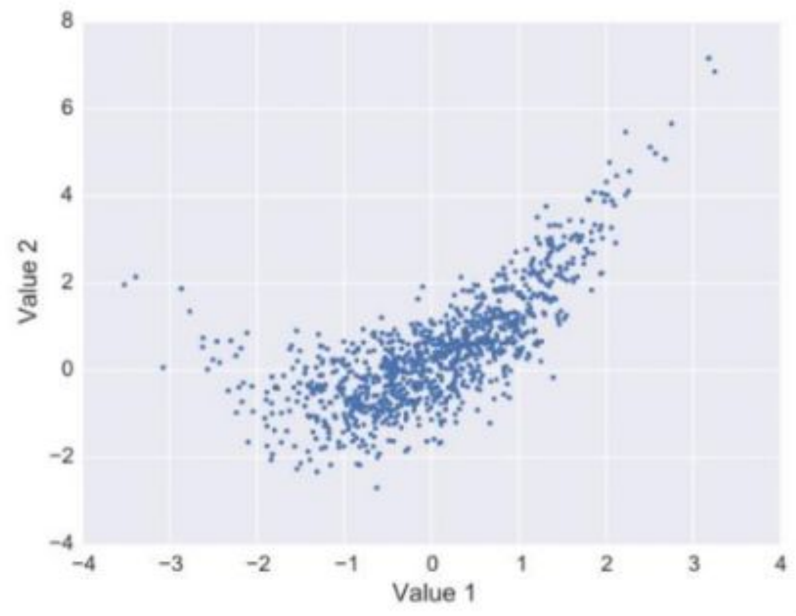
Data Analysis

Analysis and Visualization

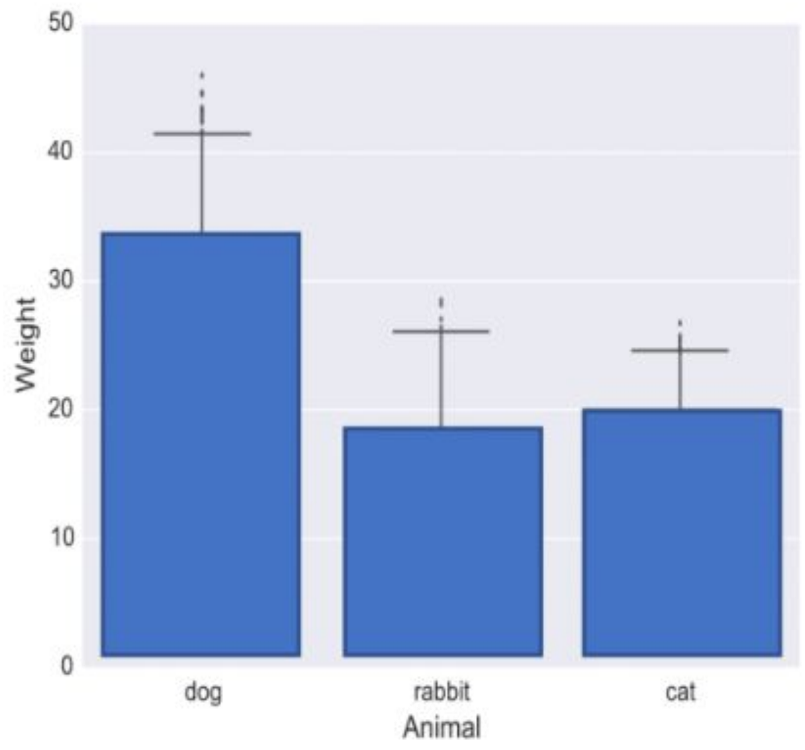
In order to choose the best visualization for the job, consider the **type** of the data you're presenting (categorical, ordinal, or numerical), and how many **dimensions** of data you need to visualize.

Two-dimensional:

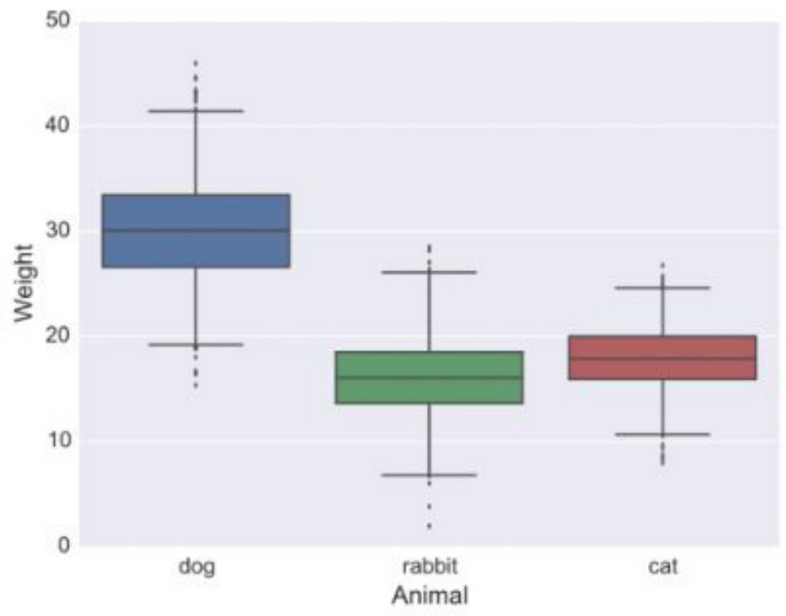
To analyze **numerical x numerical** data, use a **scatter plot**.



To analyze **numerical x ordinal/categorical** data, use a **bar chart** for averages or a **box-and-whiskers plot** for ranges.



It is difficult to analyze **ordinal/categorical x ordinal/categorical** data visually; use a table instead.



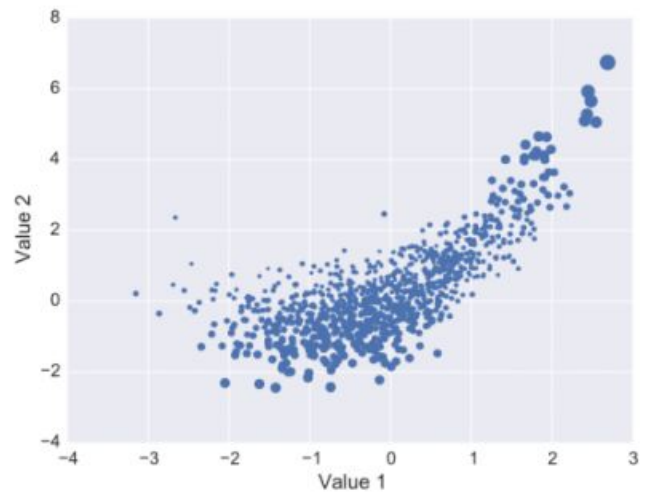
Data Analysis

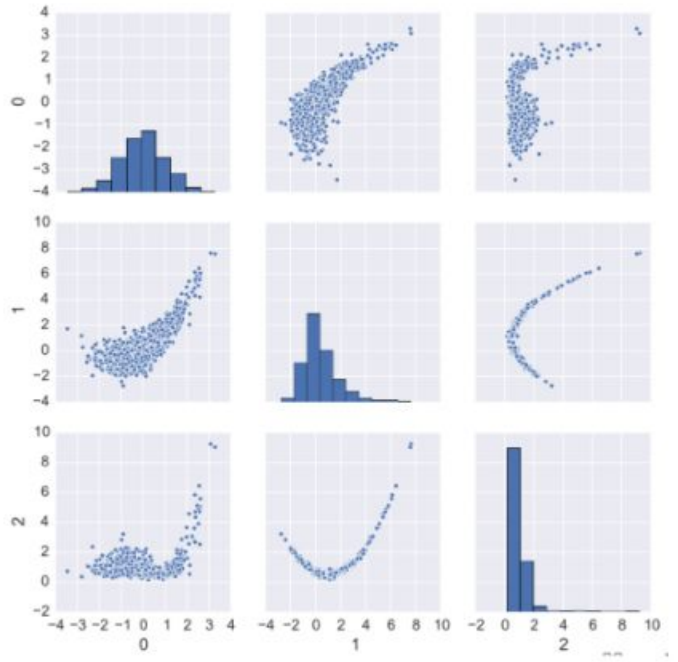
Analysis and Visualization

In order to choose the best visualization for the job, consider the **type** of the data you're presenting (categorical, ordinal, or numerical), and how many **dimensions** of data you need to visualize.

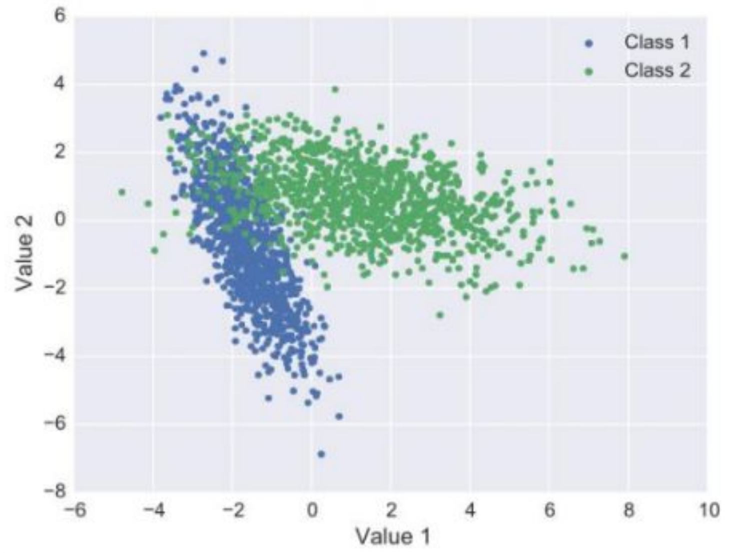
Three-dimensional:

To analyze **numerical x numerical x numerical** data, use a **bubble plot** to compare all three together or a **scatter plot matrix** to compare all the pairs.





To analyze **numerical x numerical x ordinal/categorical** data, use a **colored scatter plot**.



Data Analysis

Analysis and Visualization

Reformat the data (sometimes we don't have to) and draw!

import matplotlib.pyplot as plt

…

e.g.

```
labels = [ "A", "B", "C", "D", "E" ]
yValues = [ 10, 40, 36, 46, 21 ]
colors = [ "red", "yellow", "green",
           "blue", "purple" ]
plt.bar(labels, yValues, color=colors)

plt.xlabel("Product Categories")
plt.ylabel("# Purchased")

plt.show()
```