Probability from Scratch

Thomas Nield
O'Reilly Media

About the Speaker

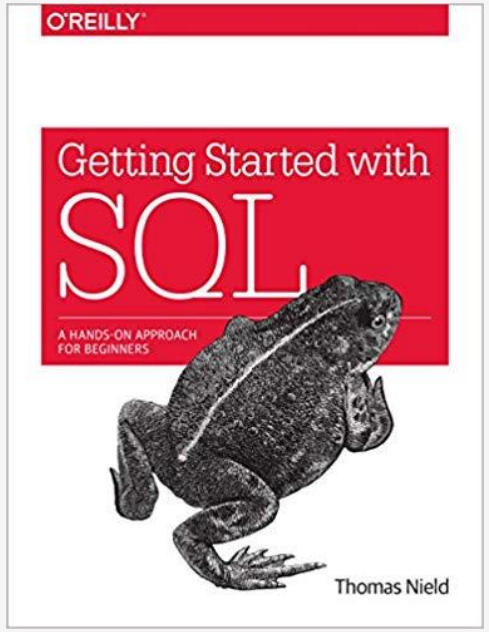
Thomas Nield

Nield Consulting Group, LLC

Author of *Getting Started with SQL* by O'Reilly and *Learning RxJava* by Packt

My other online trainings at O'Reilly:

- *SQL Fundamentals for Data*
- *Intermediate SQL for Data Analytics*
- *Programming with SQL*
- *Intro to Mathematical Optimization*
- *Machine Learning from Scratch*



thomasnield9727

https://github.com/thomasnield

What to Expect

1. Introduction
2. Probability Math
3. Bayes Theorem
4. Binomial and Beta Distribution
5. Normal Distribution
6. Logistic Regression (if we have time)

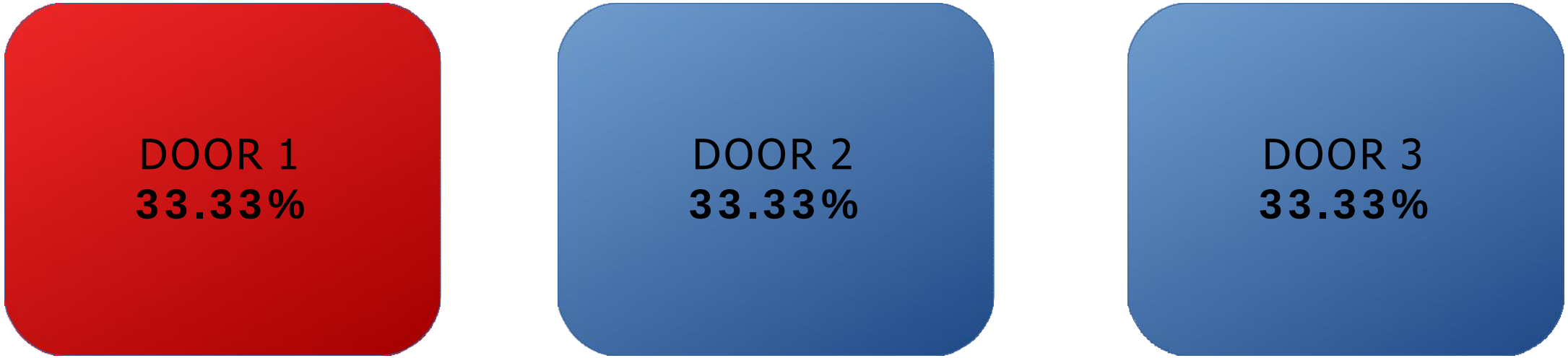
Section I

Why Learn Probability?

The Monty Hall Problem

Choose a door, one has a prize.
Two others have goats.

You choose **Door 1.**
What is the probability it has the prize?



You choose **Door 1**.
What is the probability it has the prize? **33%**

DOOR 1

DOOR 2

Twist! **Door 3** was just opened. It’s a goat.
Did you want to switch from **Door 1** to **Door 2**?

DOOR 1
?%

DOOR 2
?%

What is the prize probability of each door now?

DOOR 1
?%

DOOR 2
?%

HINT: The prize probability of either door is not 50%

DOOR 1
33.33%

DOOR 2
66.66%

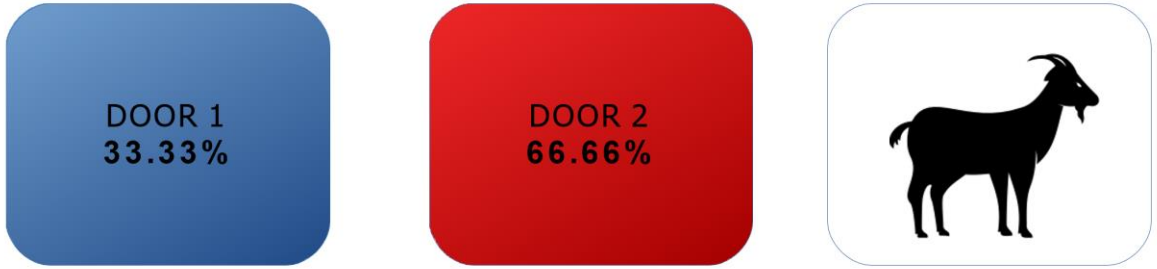
The probability of **Door 1** is **33.33%** while **Door 2** is now **66.66%**.
You should switch! But why?

33.33%

66.66%

33.33%

According to Bayes Theorem...



$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$P(P_1|D_2) = \frac{P(D_2|P_1)P(P_1)}{P(D_2)} = \frac{(.5)(.33)}{(.5)} = .33$$

$$P(P_2|D_2) = \frac{P(D_2|P_2)P(P_2)}{P(D_2)} = \frac{(1)(.33)}{(.5)} = .66$$

The Monty Hall Problem

$D_2 =$ Probability of door 2 being left $= .5$

$P_1 =$ Probability door 1 contains prize $= .33$

$P_2 =$ Probability door 2 contains prize $= .5$

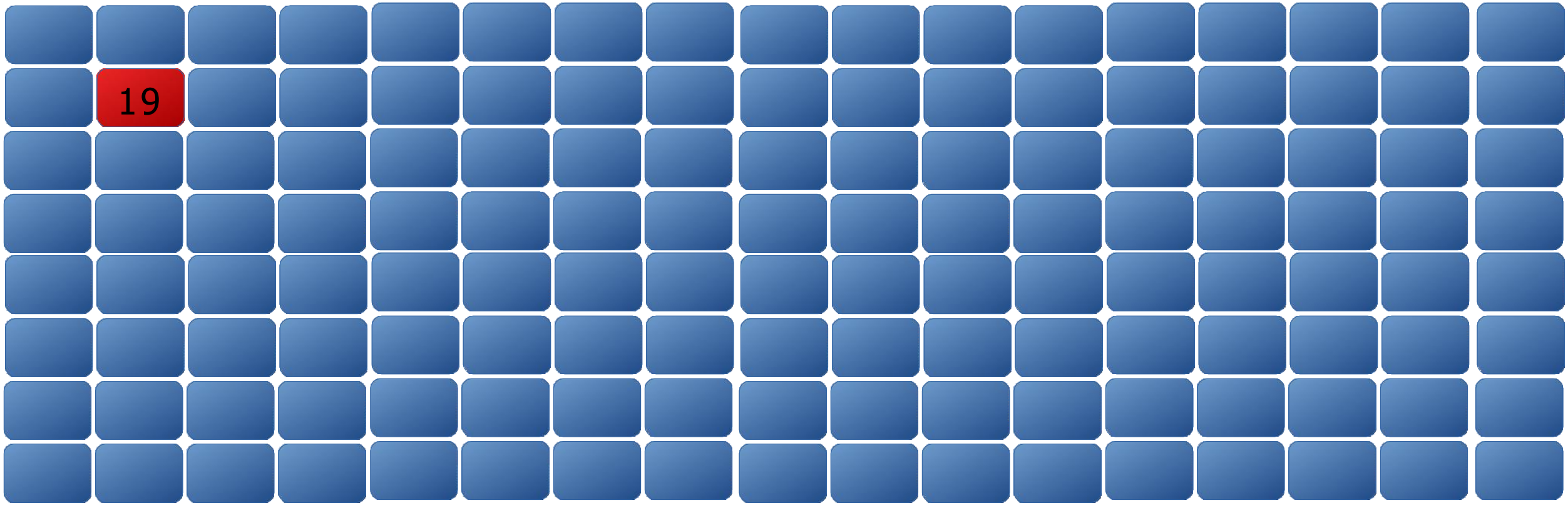
$P(P_1|D_2) =$ Probability door 1 contains prize given door 2 is left $= .33$

$P(P_2|D_2) =$ Probability door 2 contains prize given door 2 is left $= .66$

$P(D_2|P_1) =$ Probability door 2 is left given door 1 has prize $= .5$

$P(D_2|P_2) =$ Probability door 2 is left given door 2 has prize $= 1.0$

Still confused? Hyperbolize! Imagine you had 1000 doors, and you chose **Door #19.**



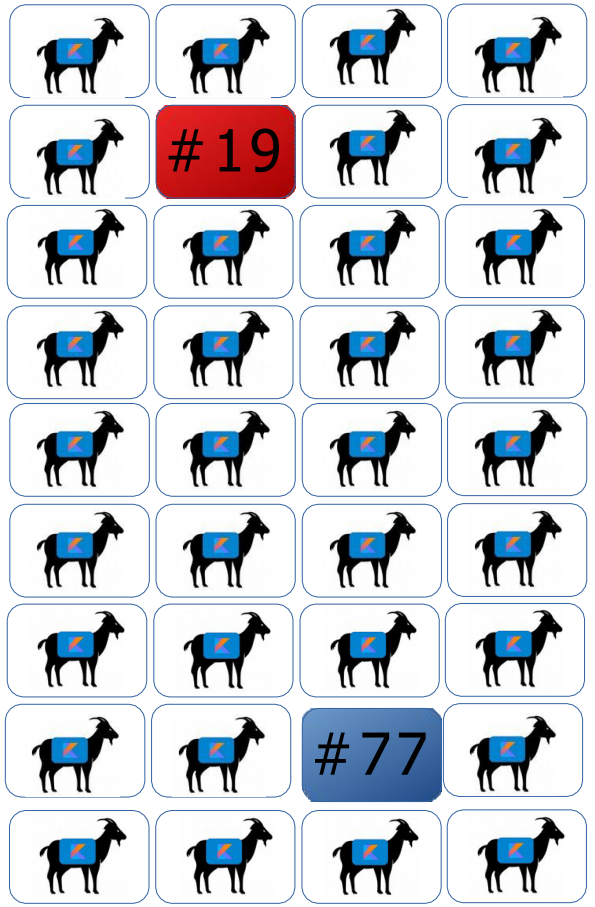
All other doors are opened but yours and **Door #77.** Inclined to switch now?

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$P(P_{19}|D_{77}) = \frac{P(D_{77}|P_{19})P(P_{19})}{P(D_{77})} = \frac{\frac{1}{999} * \frac{1}{1000}}{\frac{1}{1000}} = \frac{1}{999}$$

$$P(P_{77}|D_{77}) = \frac{P(D_{77}|P_{77})P(P_{77})}{P(D_{77})} = \frac{\frac{999}{1000} * \frac{1}{1000}}{\frac{1}{1000}} = \boxed{\frac{999}{1000}}$$

Yes, you should switch!



Monte Carlo Simulation

```python
from random import randint


def random_door():
    return randint(1, 3)


trial_count = 10000

stay_wins = 0
switch_wins = 0

for i in range(0, trial_count):
    prize_door = random_door()
    selected_door = random_door()
    opened_door = list(d for d in range(1, 4) if d != selected_door and d != prize_door)[0]
    switch_door = list(d for d in range(1, 4) if d != selected_door and d != opened_door)[0]

    if selected_door == prize_door:
        stay_wins += 1

    if switch_door == prize_door:
        switch_wins += 1

print("STAY WIN RATE: {}, SWITCH WIN RATE: {}".format(float(stay_wins) / float(trial_count),
                                                      float(switch_wins) / float(trial_count)))
```

OUTPUT:

STAY WIN RATE: 0.3373, SWITCH WIN RATE: 0.6627

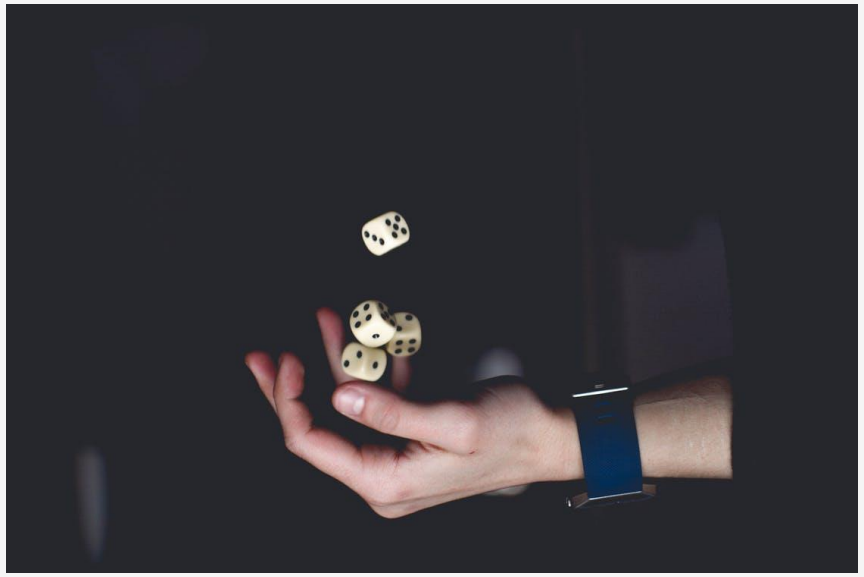
Monte Carlo simulation of the Monty Hall Problem
https://gist.github.com/thomasnield/95443be4d5255929a0d716933df29de7

Why Learn Probability?

Probability is the building block to statistics, machine learning, data science, engineering, and several other disciplines.

It is common to have to make decisions with limited information; as a matter of act, this is the majority of decision we make.

Probability enables us to measure how certain we are in an outcome occurring and while this is an imperfect art, it can be immensely valuable.



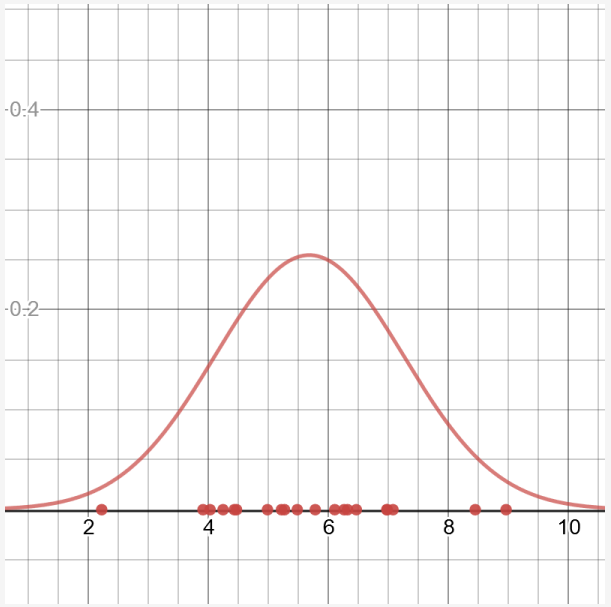
Probability versus Statistics

Probability and statistics often get confused and said interchangeably, but there is a distinction.

- **Probability** is solely about studying likelihood.
- **Statistics** utilizes data to discover likelihood.

In practicality, these two things are going to be tightly tied together, as one can argue it is hard to have probability without data.

We will put more emphasis on probability in this class, but we will work with data and statistics where it makes sense.



A normal distribution is a probability tool, but it becomes statistical when it is fit to data points.

What is Probability?

Probability is how likely an event will happen, based on observations or belief.

- How likely is it I will get 7 heads in 10 fair coin flips?
- What are my chances in winning an election?
- What is the likelihood my flight will be late?
- How certain am I a product is defective?

Probability is often expressed in two ways:

- As a percentage: 60% chance my flight will be late
- As an odds ratio: 3:2 odds my flight will be late

However, there are two philosophies on probability.

Frequentist Probability

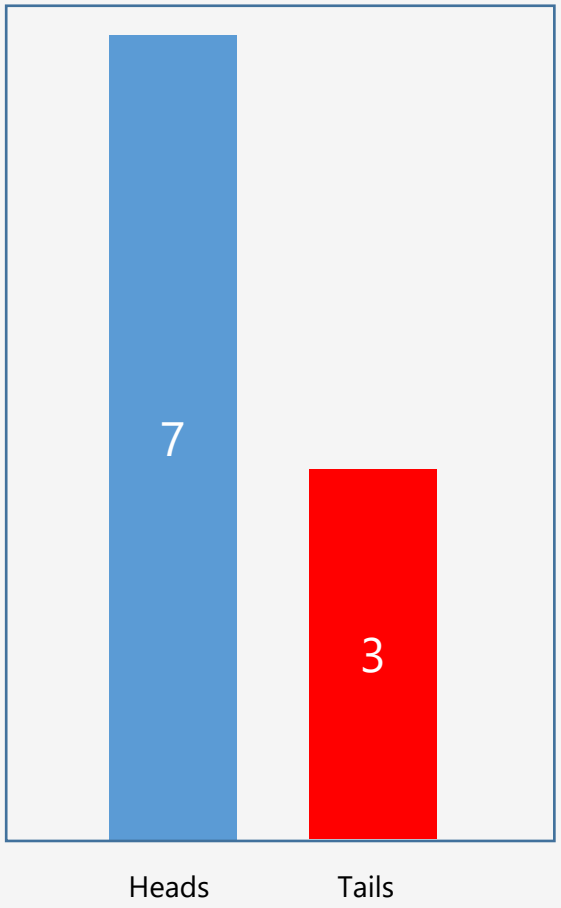
Frequentist probability is the most popularly understood approach to probability, believing that frequency of an event provides hard evidence of the probability.

EXAMPLE: I flip a coin 10 times and I get 7 heads, so I suspect the probability of heads is 70%.

This definition might be a little simplistic, as frequentists believe gathering more data will increase confidence in the probability.

Frequentism tends to work best when a lot of data is available, reliable, and complete.

Tools frequentists use include p-values and confidence intervals.



Bernoulli distribution of coin flip outcomes

Bayesian Probability

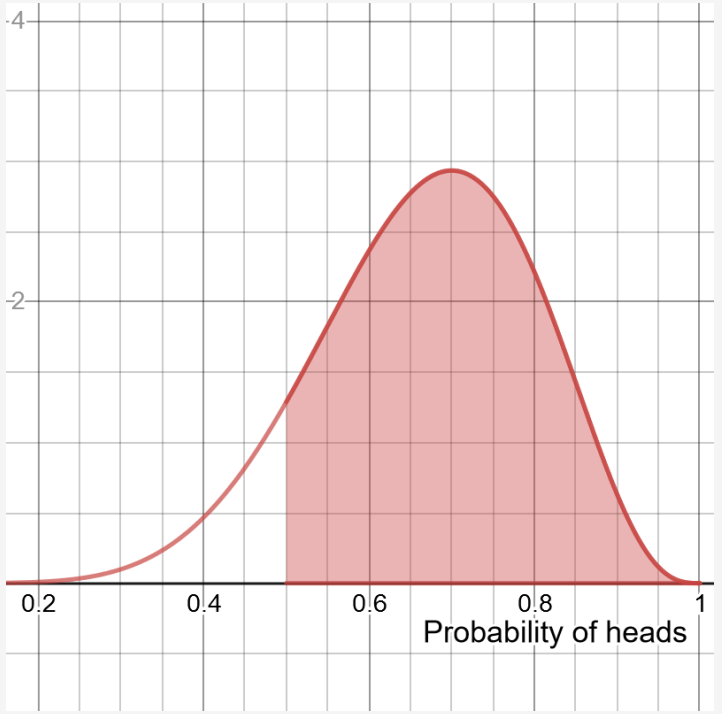
Bayesian probability is much more abstract in that it assigns subjective beliefs in a probability and not just data.

An arbitrary probability can be assigned based on subjective beliefs, and then data can be used to gradually update that belief.

EXAMPLE: I believe a coin has 50% probability of heads. I flip it 10 times and I get 7 heads. I update a beta distribution (to the right) and see there are greater likelihoods of heads being more than 50%.

Bayesian methods tend to work well when data is limited, a large amount of domain knowledge is present, or uncertainty is hard to eliminate.

Bayesian tools include the Bayes factor and credible intervals.



Beta distribution showing the probability of probabilities of heads, given 7 heads and 3 tails.

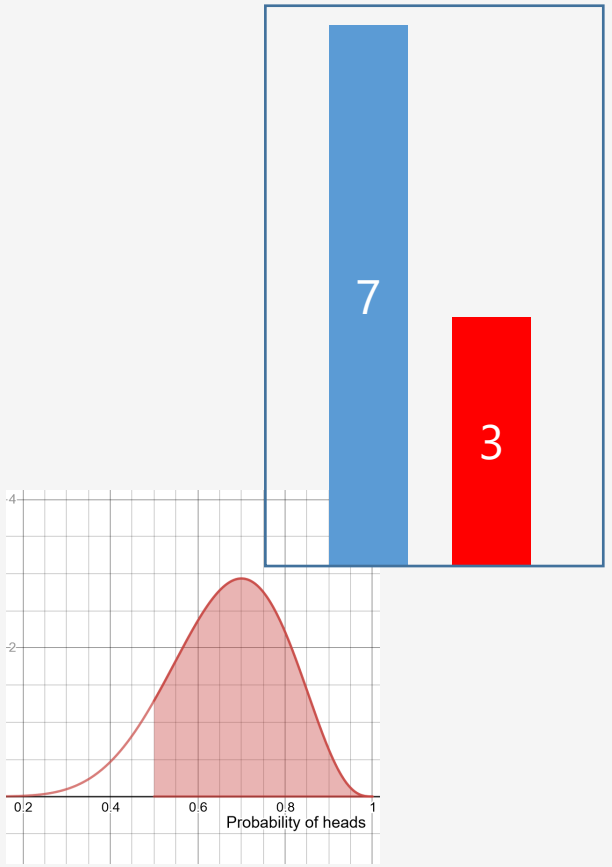
Bayes versus Frequentist? Which is right?

Frequentists believe probability is absolute, and Bayesians believe probability is a fuzzy construct.

Which of these philosophies are right? Both!

- Some situations warrant Frequentism, while others warrant Bayes.
- Frequentism is better if you can converge on a single probability based on enough data, but Bayes is more suited if you want to entertain larger ranges of possibility for a limited amount of data.

Keep in mind that all models are wrong but some are useful, and you should always seek the right tool for the job.



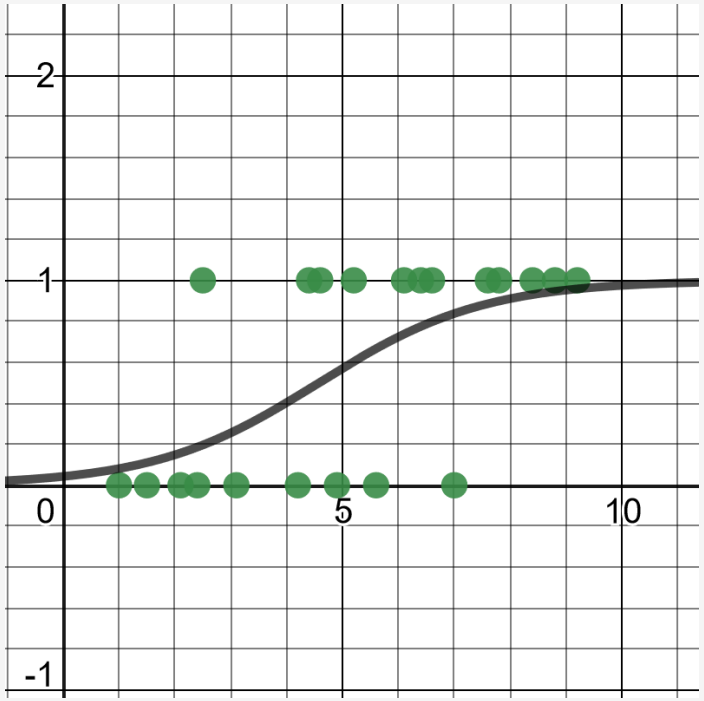
Probability and Machine Learning

While machine learning is not the focus of this class, know that probability is the building block of machine learning!

Machine learning processes data and infers and combines probabilities based on the data.

Probability is used not just for data preparation and analysis but also feature selection, optimization, and prediction.

We will learn how to perform maximum likelihood estimation later to fit a normal distribution to observed data, as well as a logistic regression.

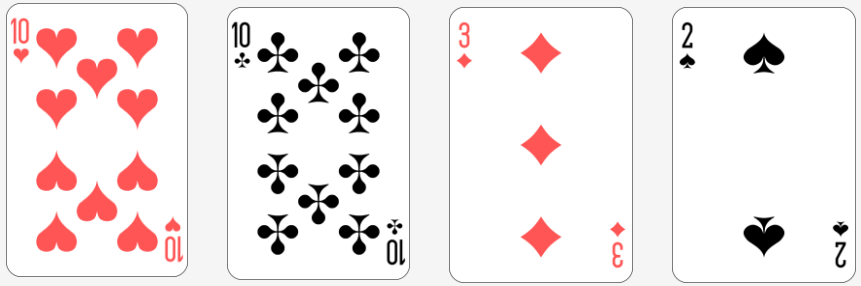


Logistic regression predicts a probability an event will happen given one or more variables and uses maximum likelihood estimation to fit a curve to the data.

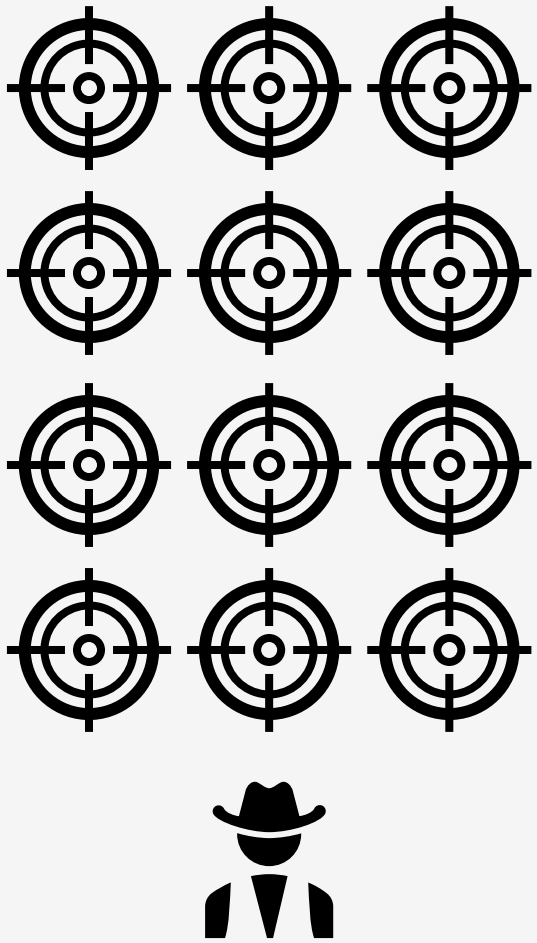
Texas Sharpshooter Fallacy

As you start to work with probability, statistics, and machine learning, one thing to be careful about is to not fall victim to finding patterns in randomness.

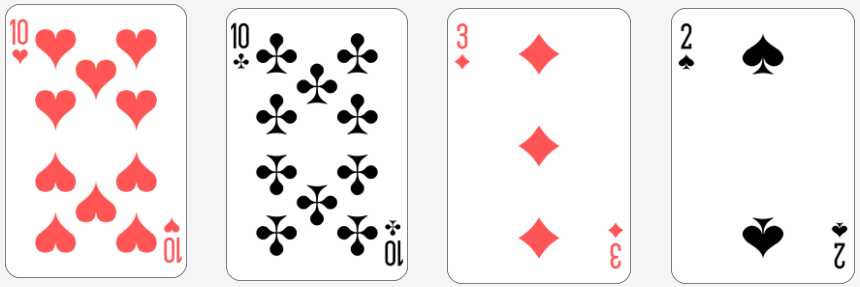
Say I drew four cards from a deck:

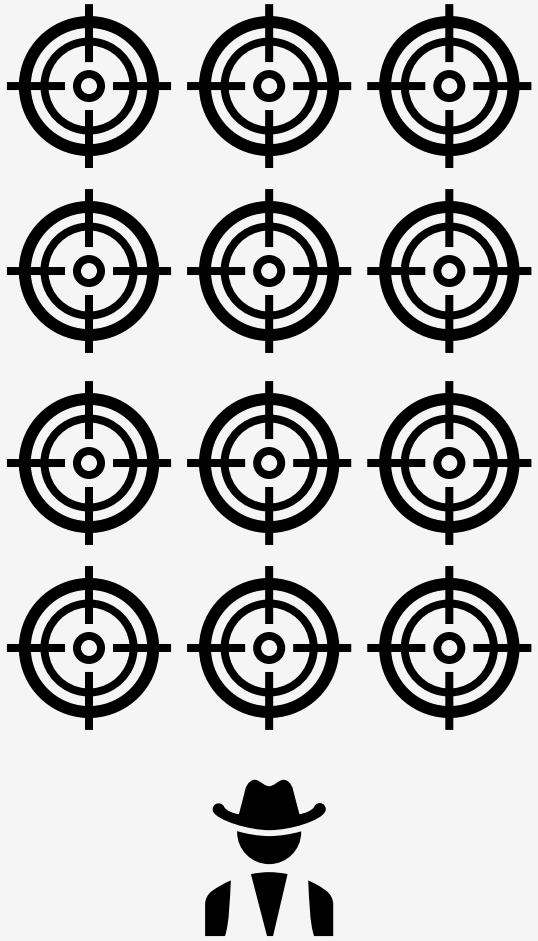


- Is this not fascinating? I have two 10's and a consecutive 3 and 2! What does it mean?
- Should I predict the next four cards will also have two 10's and two consecutive numbers?
- As I start diving into analyzing this outcome, note I never made it my objective to achieve this outcome; I observed it ***after*** it occurred.



Texas Sharpshooter Fallacy





- Unless I predicted this specific outcome beforehand, there is nothing meaningful I have accomplished nor did I create a predictive model.
- This is known as the **Texas Sharpshooter Fallacy**, where I have collected data without an objective and then analyzed it afterward, believing there to be meaningful patterns rather than consider it happened by random chance.
- This is one of the problems with **data mining**, a practice of analyzing data that was never collected with a specific objective.
- As you work with probability, statistics, and machine learning be careful to not fall into this trap of finding meaning in random patterns.

Section II

Probability Fundamentals

Probability Basics

Hopefully the concept of a **probability** is familiar, which measures how likely an outcome x is and typically is represented as a number $P(x)$ between 0.0 and 1.0.

A probability is typically represented as a decimal between 0.0 and 1.0 but is also represented as a percentage between 0% and 100%.

The probability of an event $P(x)$ ***NOT*** occurring can be calculated by $1.0 - P(x)$, which indicates both outcomes must add to 1.0.

When we work with a single simple probability, it is known as a **marginal probability**.

Where Does Probability Come From?

Probability can be based off data, a belief, or both!

Probability based on data: If we sample 10 products from a factory line and find 4 items are defective, that would be a 40% defective rate.

Probability based on belief: An engineer realizes an inferior material was used and guesses the defective rate for the product will be 50%.

Probability based on data + belief: we can quantify the engineer's belief and the data, merge them together, and find a 44.44% probability is most likely.

Expressing Probability as Odds

You may sometimes see probability expressed as an **odds ratio**, which expresses how many times we believe in something being true versus not being true.

If we believe out of 10 products, 6 will be defective and 4 will not, the odds ratio would be $6:4$ or $\frac{6}{4}$

This would reduce to 1.5, which means we believe a product is 1.5 times more likely to be defective than not defective.

Quantifying Belief with Odds

Odds ratios are also a helpful way to quantify subjective beliefs by means of "betting."

If my friend is willing to pay me $200 if the Dallas Cowboys win the Superbowl, but I must pay him $50 if they do not, that means he believes the Dallas Cowboys are 4x more likely to lose rather than win ($\frac{200}{50} = 4.0$).

To hyperbolize, if he pays $200 for them winning but I must pay him $1 if they lose, that means he *REALLY* believes the Dallas Cowboys are going to lose: 200x more likely ($\frac{200}{1} = 200$) and feels no need to hedge his bet.

Putting money on something (horse races, sports teams, stock markets) forces you to quantify how strongly you believe in an event, and can be effective measure to turn subjective beliefs into something quantitative.

Turning Odds into Probabilities

If you need to turn an odds ratio $O(X)$ into a probability, you can do so with this formula:

$$P(X) = \frac{O(X)}{1 + O(X)}$$

So if you believe something is 3x likely to happen versus not happen (3:1), the probability for that event is 75%:

$$P(X) = \frac{3}{1+3} = .75$$

Joint Probabilities

Probability gets interesting when we think about how multiple probabilities interact with each other.

Let's look at the probability of flipping a coin and rolling a die and getting a *heads* (Event A) and a *six* (Event B).

This is known as a **joint probability**, the probability of two events *A* and *B* occurring simultaneously.

Since *A* and B are **independent** in this case, meaning they do not affect each others' outcomes, their joint probability is as simple as multiplying them together:

$$P(A \cap B)$$

$$\mathrm{P}(A \text{ and } B)$$

$$\mathrm{P}(A, B)$$

$$P(A \text{ and } B) = P(A) * P(B)$$

$$P(\text{Heads and } 6) = \frac{1}{2} * \frac{1}{6} = \frac{1}{12} = .0833$$

and

Joint Probabilities

Why does joint probability work like this for independent events? Let's consider all possible outcomes in the tree to the right:

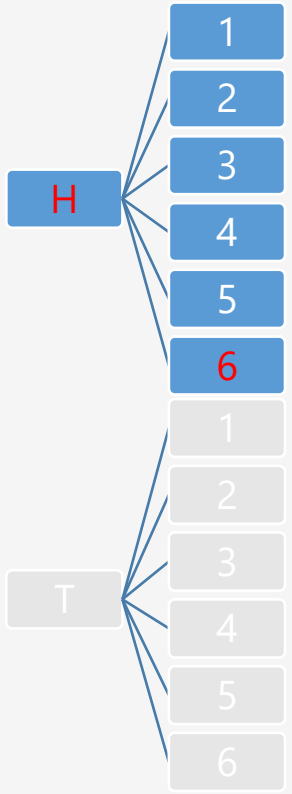
Notice that "heads" has a 50% probability, and "six" has a 16.66% probability. We have 12 possible outcomes:

H1, H2, H3, H4, H5, H6, T1, T2, T3, T4, T5, T6

Only one of these 12 possible outcomes meets our criteria, so $\frac{1}{12} = .08333$

or 8.333%

The multiplication allows us to avoid generating and counting all these combinations and is known as the **product rule**.

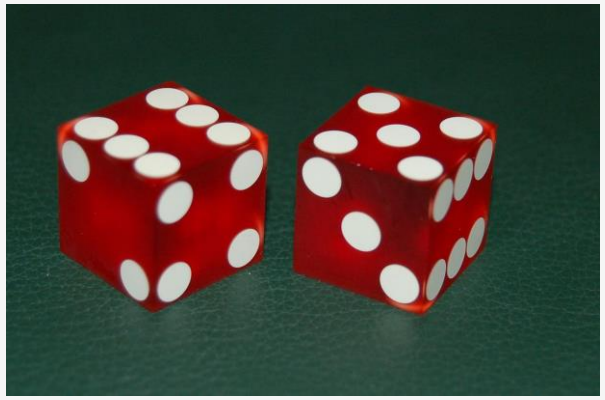


$$P(Heads\ and\ 6) = \frac{1}{2} * \frac{1}{6} = \frac{1}{12}$$

Joint Probabilities

You can use the product rule to combine as many probabilities as you want.

For example, the probability of rolling a "six" 10 times in a row:



$$\frac{1}{6} * \frac{1}{6} * \frac{1}{6} * \frac{1}{6} * \frac{1}{6} * \frac{1}{6} * \frac{1}{6} * \frac{1}{6} * \frac{1}{6} * \frac{1}{6} = \frac{1}{60466176} = 0.0000000165382$$

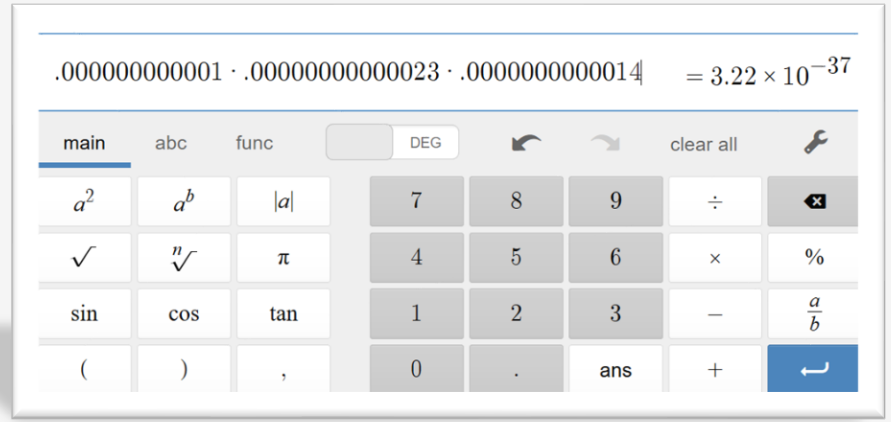
Of course, since this multiplication is repetitive, we can use an exponent:

$$(\frac{1}{6})^{10} = \frac{1}{60466176} = 0.0000000165382$$

Floating Point Underflow

When multiplying decimals you need to be careful about **floating point underflow**, which causes the decimal to become so small the computer can no longer keep track of it.

- We can remedy the issue here by using logarithmic addition rather than multiplication.
- Transform each probability with a $\log()$ or $\ln()$ function and then sum them, then call $\exp()$ to convert the result back.



For example, if we want calculate the probability of getting heads 5 times in a row, we might be better off using a ln() and exp() function to avoid floating point underflow:

$$\exp(\ln(.5) + \ln(.5) + \ln(.5) + \ln(.5) + \ln(.5)) = .03125$$

Union Probability – Mutually Exclusive

Things get a little more nuanced when we work with **union probabilities**, or the probability that at least one of multiple events will occur.

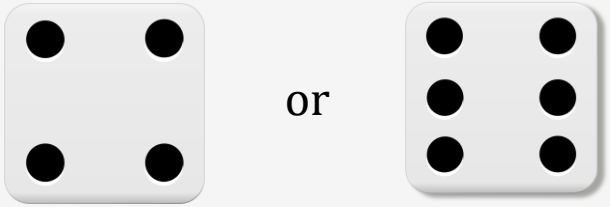
When two events are **mutually exclusive**, meaning that only one of the events can occur but not both, then it is as easy as adding their probabilities together.

For example, what is the probability of getting a "4" or "6" on a die roll? Since we cannot get "4" and a "6" simultaneously, we just add these probabilities together.

$$\frac{1}{6} + \frac{1}{6} = \frac{1}{12} = .08333$$

$$P(A \cup B)$$

$$\mathrm{P}(A \text{ or } B)$$



Union Probability – Non-Mutually Exclusive

When two events are **non-mutually exclusive**, meaning that two events can occur simultaneously, their union probability gets tricky.

$$P(A \cup B)$$
$$\mathrm{P}(A \text{ or } B)$$

What is the probability of getting a "heads" ***OR*** a "six" with a coin flip and a die roll? Before we try anything let's look at the possible outcomes:

H1, H2, H3, H4, H5, H6, T1, T2, T3, T4, T5, T6

or

The outcomes in red are the ones that meet our condition, and counting them would lead us to a correct answer:

$$\frac{7}{12} = .5833$$

But if we added those two probabilities together, we would get a wrong answer! So why does this not work?

$$\frac{1}{2} + \frac{1}{6} = \frac{2}{3} = .6666$$

Union Probability – Non-Mutually Exclusive

H1, H2, H3, H4, H5, H6, T1, T2, T3, T4, T5, T6

The problem with adding non-mutually exclusive events is it causes double-counting of joint probability events as highlighted above.

It's subtle but notice above that if we add the probability of "heads" (1/2) with the probability of getting a "six" (1/6), we have counted that "six" probability of 1/6 twice!

We can remedy this by subtracting the joint probability for non-mutually exclusive events:

$$P(A\ or\ B) = P(A) + P(B) - P(A\ and\ B)$$

$$P(heads\ or\ six) = \frac{1}{2} + \frac{1}{6} - \left(\frac{1}{2} * \frac{1}{6}\right) = \frac{7}{12}$$

This is known as the **sum rule**, where we can find the OR probability between two or more events by summing them and then subtracting their joint probability.

$$P(A \cup B)$$

$$\mathrm{P}(A \text{ or } B)$$

or

The Sum Rule

$$P(A\ or\ B) = P(A) + P(B) - P(A) * P(B)$$

$$P(A \cup B)$$

$$\mathrm{P}(A\ \mathrm{or}\ B)$$

Notice that the sum rule applies to both mutually exclusive and non-mutually exclusive probabilities.

When events A and B are mutually exclusive, the joint probability is 0 since they both cannot occur simultaneously.

But when events A and B are not mutually exclusive, the joint probability plays a role in subtracting the double-counting caused by both events occurring.

or

Conditional Probability

Another nuanced idea in probability is **conditional probability**, or the probability of *A* given *B* has occurred.

If *B* has no impact on whether *A* occurs, then $P(A) = P(A\ given\ B)$.

But if *B* does increase or decrease the probability of *A* occurring, then the $P(A)$ is going to be different than $P(A\ given\ B)$.

The probability of someone being colorblind is 4.5%, but the probability of a male being colorblind is 8%.

$$P(colorblind) = .045$$

$$P(colorblind\ given\ male) = .08$$

Does this mean any colorblind person is 8% likely to be male? Or that any male is 8% likely to be colorblind?

$P(A|B)$

P(A given B)

Conditional Probability

Direction of the conditional probability matters! The *probability of a male given they are colorblind* is not the same as the *probability of being colorblind given they are male*!

We will learn how to flip this probability with Bayes Theorem shortly, but first let's see why conditional probability plays a critical role in **AND** and **OR** probabilities.

Let's say we draw a random person from the population, and the probability of them being male $P(male)$ is 50%.

If we want to calculate the probability they are male and colorblind, do we multiply $P(male)$ with $P(colorblind)$ or $P(colorblind\ given\ male)$?

$P(male) = .5$
$P(colorblind) = .045$
$P(colorblind\ given\ male) = .08$

$P(male\ and\ colorblind) = P(male) \quad * \quad ?$

$P(A|B)$

P(A given B)

Conditional Probability

$P(male) = .5$

~~$P(colorblind) = .045$~~

$P(colorblind\ given\ male) = .08$

$P(male\ and\ colorblind) = P(male) * P(colorblind\ given\ male)$

$P(male\ and\ colorblind) = .5 * .08$

$P(male\ and\ colorblind) = .0036$

With joint probability we use the conditional probability when it is available! If we want to calculate the probability of someone being male and colorblind, use the conditional probability that accounts for them being male!

If we do not use the conditional probability, then the probability of them being male and colorblind will be no different than being female and colorblind!

$P(A|B)$

P(A given B)

Conditional Probability

We can update our joint probability and union probability formulas to account for conditional probability:

$$\boldsymbol{P(A\ and\ B) = P(A) * P(B\ given\ A)}$$

$$\boldsymbol{P(A\ or\ B) = P(A) + P(B) - P(A) * P(B\ given\ A)}$$

Remember that if A and B are independent and have no impact on each other, then these formulas still work because $P(B) = P(B\ given\ A)$.

It can be hard to determine if two events are conditional and related, and therefore it is common to assume in statistics they are independent.

We will see this assumption in machine learning applications like Naïve Bayes.

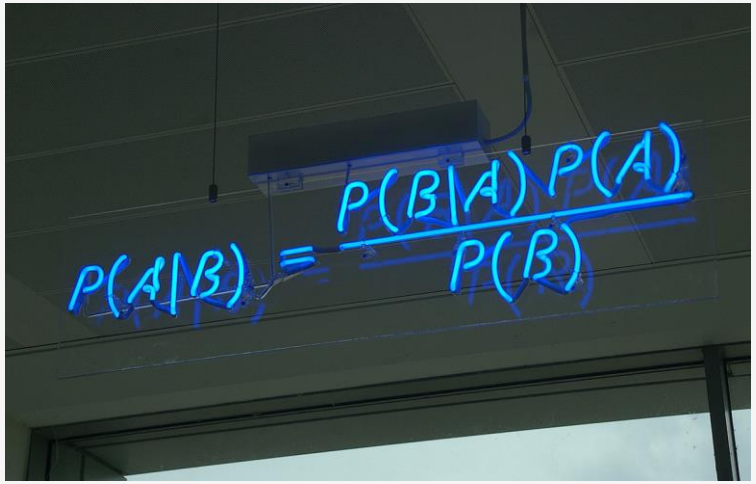
$P(A|B)$

P(A given B)

Bayes Theorem

We already saw Bayes Theorem in the Monty Hall Problem, and we will explore it in more depth in the next section.

But it is worth bringing up it allows us to flip a conditional probability, turning $P(A\ given\ B)$ into $P(B\ given\ A)$.

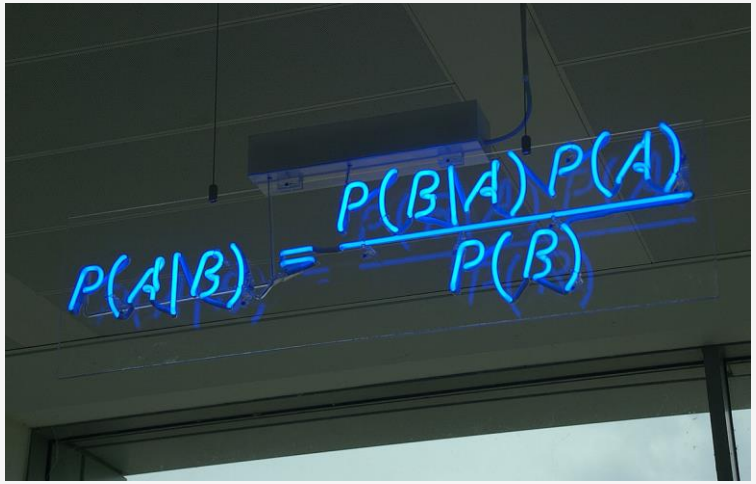


$$P(A\ given\ B) = \frac{P(B\ given\ A) * P(A)}{P(B)}$$

Bayes Theorem

Let's look at the color blindness problem again. We have these data points:

$$P(color\ blind) = .045$$
$$P(male) = .50$$
$$P(color\ blind\ given\ male) = .08$$



What is the probability of a *person being male given they are color blind*?

$$P(A\ given\ B) = \frac{P(B\ given\ A) * P(A)}{P(B)}$$

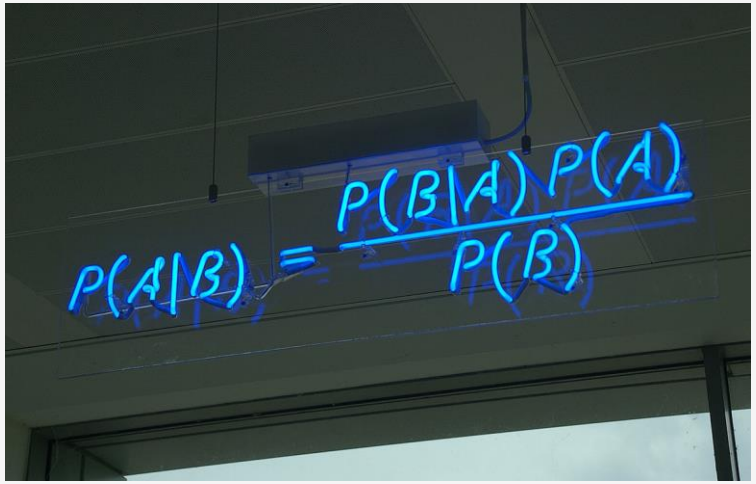
$$P(male\ given\ color\ blind) = \frac{.08*.5}{.45} = .8888$$

Bayes Theorem

The probability of a person *being male given they are color blind* is 88.88%, which is much different than the probability of *being color blind given they are a male*, which is 8%.

Bayes Theorem is the core of conditional probability, as it allows us to flip inferences based on conditions.

We will learn more about it in the next section.



Exercise 1A

There is a 30% chance of rain today, and a 40% chance your umbrella order will arrive on time. You are eager to walk in the rain today and cannot do so without either!

What is the probability it will rain ***AND*** your umbrella will arrive?

Exercise 1A

There is a 30% chance of rain today, and a 40% chance your umbrella order will arrive on time. You are eager to walk in the rain today and cannot do so without either!

What is the probability it will rain ***AND*** your umbrella will arrive?

$$P(A \text{ and } B) = P(A) * P(B)$$

$$P(\text{Rain and umbrella arrives}) = .30 * 40 = .12$$

12% chance it will rain and your umbrella will arrive.

Exercise 1B

There is a 30% chance of rain today, and a 40% chance your umbrella order will arrive on time.

You will only be able to run errands if it does not rain or your umbrella arrives.

What is the probability it will not rain ***OR*** your umbrella arrives?

Exercise 1B

There is a 30% chance of rain today, and a 40% chance your umbrella order will arrive on time.

You will only be able to run errands if it does not rain or your umbrella arrives.

What is the probability it will not rain ***OR*** your umbrella arrives?

$P(rain) = .30$
$P(no\ rain) = 1.0 - P(rain) = 1.0 - .30 = .70$
$P(umbrella\ arrives) = .40$

$P(A\ or\ B) = P(A) + P(B) - P(A\ and\ B)$
$P(no\ rain\ or\ umbrella\ arrives) = .70 * .40 - (.70 * .40) = .82$

82% chance it will not rain or your umbrella arrives.

Exercise 1C

There is a 30% chance of rain today, and a 40% chance your umbrella order will arrive on time.

However, you found out if it rains there is only 20% chance your umbrella will arrive on time.

What is the probability it will rain and your umbrella will arrive on time?

Exercise 1C

There is a 30% chance of rain today, and a 40% chance your umbrella order will arrive on time.

However, you found out if it rains there is only 20% chance your umbrella will arrive on time.

What is the probability it will rain and your umbrella will arrive on time?

$$P(A \text{ and } B) = P(A) * P(B \text{ given } A)$$
$$P(\text{rain and umbrella arrives given rain}) = .30 * .20 = .06$$

6% chance it will rain and your umbrella will arrive.

Section III

Bayes Theorem

Violence and Video Games

Let's say a study has been released that claims 85% of homicidal criminals in the United States have played violent video games.

What does this mean? Should we be alarmed and blame the video game industry?

Keep in mind that a conditional probability $P(A|B)$ is not the same as its reverse conditional probability $P(B|A)$.

$$P(gamer|homicidal) = .85$$
$$P(homicidal|gamer) = ?$$

Just because most homicidal people are gamers (according to this study), does that mean most gamers are homicidal?

The answer we will discover is "NO!"

Violence and Video Games

We need to use Bayes Theorem to flip the condition:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

$$P(homicidial|gamer) = \frac{P(gamer|homicidal) * P(homicidal)}{P(gamer)}$$

But we only have $P(gamer|homicidal)$ which is .85.

We also need $P(homicidal)$ and $P(gamer)$ if we are going to flip the direction of the probability.

Violence and Video Games

Sometimes it can be hard to get these other probabilities, but with population studies a little bit of research might give us some numbers.

According to the FBI, there are 17,251 known homicidal offenders in 2017 and according to Wolfram Alpha there are 324 million people in the United States.

$$P(homicial) = \frac{17251}{324000000} = .00005$$

Gathering data from video game industry market research, you estimate that 19% of the population plays violent video games.

$$P(gamer) = .19$$

Discovering Bayes Theorem

Unlike the media and politicians, we continued doing the math and research and found 19% of the population plays violent video games, and .005% of the population is homicidal.

We can use Bayes Theorem to merge these probabilities together.

*P(Homicidal if Gamer) = P(Gamer if Homicidal) * P(Homicidal) / P(Gamer)*

*P(Homicidal if Gamer) = (.85 * .00005) / .19*

P(Homicidal if Gamer) = 0.0002

Wow! So if you take any gamer there is only a .02% chance they are homicidal.

This number is much different than the 85% probability a homicidal criminal is a gamer.

Discovering Bayes Theorem

So according to Bayes Theorem, a person who plays violent video games is only .02% likely to be violent compared to a non-gamer who is .005% likely.

This number is much less significant than the 85% statistic, and shows how subtle but drastic the difference is between "*What is the probability of A given B?*" versus "*What is the probability of B given A?*"

But the media can spin that saying "gamers are 4x more likely to be homicidal" when in actuality, we are comparing small numbers to small numbers.

In other words, we should not take a common trait (playing video games, liking ice cream) and associate it with an uncommon one (being homicidal).

Deriving Bayes Theorem

So why does Bayes Theorem work? Let's look at why this formula makes sense.

Say we have a population of 100,000 people. Here are the probabilities we started with in our problem:

$P(homicial) = .00005$

$P(gamer) = .12$

$P(gamer|homicidal) = .85$

This means we would have the following numbers of people for each category:

Homicidal Criminals: 5

Gamers: 19,000

Gamers who are also criminals: 4.25

What do you observe with these numbers?

Deriving Bayes Theorem

Homicidal Criminals: 5

Gamers: 19,000

Gamers who are also criminals: 4.25

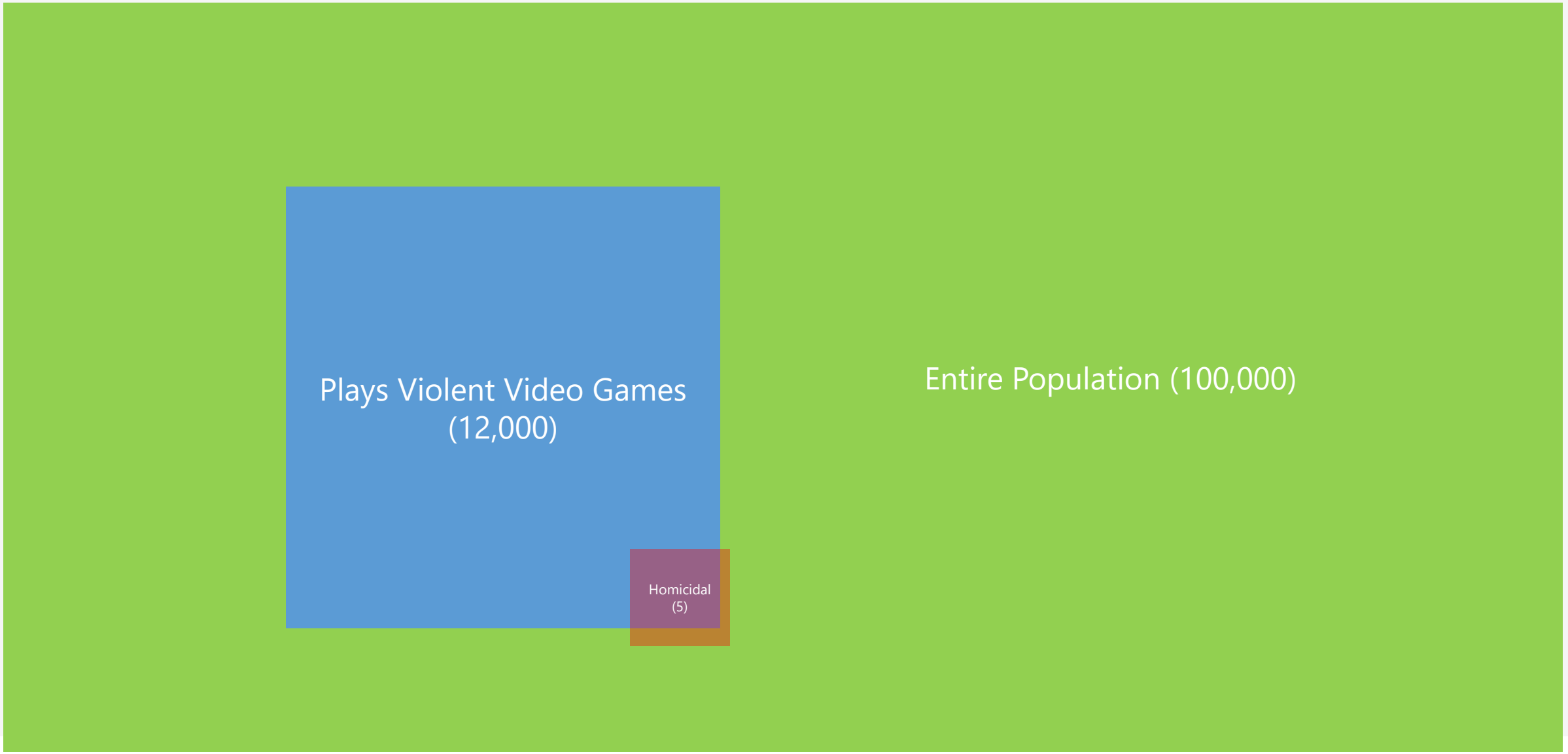
Are you really going to go after those 19,000 gamers even though only 4.25 (4 or 5) of them are homicidal criminals?

Now you can see why discrimination happens so easily based on a common attribute being tied to an uncommon attribute.

This is known as the prosecutor's fallacy.

This is also why machine learning algorithms can discriminate so easily when training datasets are imbalanced or poorly sampled.

Visualizing Bayes Theorem (Not to Scale)



Deriving Bayes Theorem

We can reasonably derive Bayes Theorem by working with an actual population number like 100,000.

Remember we derive any subset of this population by multiplying with its respective probability.

The number of homicidal people that are gamers is the probability of being a gamer given they homicidal multiplied by the homicidal population:

$$\#homicidal\ gamers\ = P(gamer\ given\ homicidal) * P(homicidal)\ *\ 100{,}000$$

The probability of being homicidal given they are gamer is the number of homicidal gamers divided by the gaming population.

$$P(homicidal\ given\ gamer) = \frac{P(gamer\ given\ homicidal) * P(homicidal)\ *\ 100{,}000}{P(gamer) * 100{,}000}$$

Deriving Bayes Theorem

$$P(homicidal\ given\ gamer) = \frac{P(gamer\ given\ homicidal) * P(homicidal)\ \cancel{* 100{,}000}}{P(gamer)\ \cancel{* 100{,}000}}$$

Notice we can cancel out the population size 100,000 since it exists in both the numerator and denominator.

$$P(homicidal\ given\ gamer) = \frac{P(gamer\ given\ homicidal) * P(homicidal)}{P(gamer)}$$

Look! Does this look familiar? We now have derived Bayes Theorem!

$$P(A\ given\ B) = \frac{P(B\ given\ A) * P(A)}{P(B)}$$

If you are struggling to see how we got here, think about the different population subsets we just calculated, the proportions of that population, and the Python code we will play with next.

Bayes and the Confusion Matrix

A medical technology vendor approaches your organization and says they have an “AI system” that takes a small blood sample, assays several variables, ensembles machine learning algorithms, and predicts a health risk (simply categorized as “AT RISK” or “NOT AT RISK”).

The vendor says that 99% of patients who are at risk will test positive.

Is this significant and credible? Why or why not? What should our next questions be?

Bayes and the Confusion Matrix

There is still some missing information you will have to solicit from the vendor.

We are told that for patients that have health risk, 99% will be identified successfully (*sensitivity*).

But what if we flip the question? What percentage of those who tested positive have the health risk (*precision*)?

The vendor goes back to their research team and returns with a confusion matrix of 1000 patients they tested.

They give you an answer: 198 / (198 + 50) = 79.8%

	TESTS POSITIVE	TESTS NEGATIVE
AT RISK	198	2
NOT AT RISK	50	750

Bayes and the Confusion Matrix

With this confusion matrix, we indeed see 99% of those who are at risk test positive, and that 79.8% of those tested positive are at risk. We found the conditional probability in both directions.

Despite these numbers, you start to feel uncomfortable that you are not using outside data beyond the vendor's test.

What else can we do to verify the vendor's claims and testing results?

	TESTS POSITIVE	TESTS NEGATIVE
AT RISK	198	2
NOT AT RISK	50	750

Bayes and the Confusion Matrix

After some Wikipedia research, you discover something pivotal…

Only 1% of the population is at risk.

This means if you planned on deploying this test to 100K patients, only 1000 are likely at risk.

The vendor advertised a 99% true positive rate. You now wonder: will the vendor's product single out these 1000 patients? Out of 100K? Without error?

	TESTS POSITIVE	TESTS NEGATIVE
AT RISK	198	2
NOT AT RISK	50	750

Bayes and the Confusion Matrix

If you feel like something is off because only 1% of the population has the health risk, but 20% of the test patients do, you would be right.

Utilizing Bayes Theorem, we can incorporate this 1% statistic to find the true probability of being at risk if tested positive.

*P(At Risk if Positive) = P(Positive if at risk) * P(At risk) / P(Positive)*

*P(At Risk if Positive) = (.99 * .01) / .248*

P(At Risk if Positive) = .0339

3.39% is now the probability of risk given a positive test. How did it drop from 99% to 3.39%?

	TESTS POSITIVE	TESTS NEGATIVE
AT RISK	198	2
NOT AT RISK	50	750

Bayes and the Confusion Matrix

3.39% is now the probability of risk given a positive test. How did it drop from 99% to 3.39%?

This just shows how easily we can get duped by probabilities that are only high in a specific population like the vendor's 1000 test patients (remember the homicides and video games example)?

So if this test only has a 3.39% probability of successfully identifying a true positive, we probably should not use it.

	TESTS POSITIVE	TESTS NEGATIVE
AT RISK	198	2
NOT AT RISK	50	750

Normalization Constant

A helpful tool in conditional probabilities is the **normalizing constant**, which can give us information we may be missing to use in Bayesian problems.

$$P(A) = P(A|B) * P(B) + P(A|not\ B) * P(not\ B)$$

This makes sense because if we have an event B that may conditionally affect event A, should not the true and false cases of B occurring add up to the total probability of A?

Take our umbrella problem from earlier:

There is a 30% chance of rain today, and a 40% chance your umbrella order will arrive on time.

However if it rains there is only 20% chance your order will arrive on time, and to run errands there needs to be NO RAIN or your umbrella arrives.

What is the probability it will ***NOT*** rain ***OR*** your umbrella arrives?

We do not know the conditional probability when there is no rain, but the formula above can help us algebraically solve for it.

Normalization Constant

There is a 30% chance of rain today, and a 40% chance your umbrella order will arrive on time.

However if it rains there is only 20% chance your order will arrive on time, and to run errands there needs to be NO RAIN or your umbrella arrives.

What is the probability it will ***NOT*** rain ***OR*** your umbrella arrives?

We can solve for $P(on\ time|no\ rain)$ algebraically:

$$P(A) = P(A|B) * P(B) + P(A|not\ B) * P(not\ B)$$

$$P(on\ time) = P(on\ time|rain) * P(rain) + P(on\ time|no\ rain) * P(no\ rain)$$

$$.40 = .20 * .3 + P(on\ time|no\ rain) * (1.0 - .3)$$

$$P(on\ time|no\ rain) = .486$$

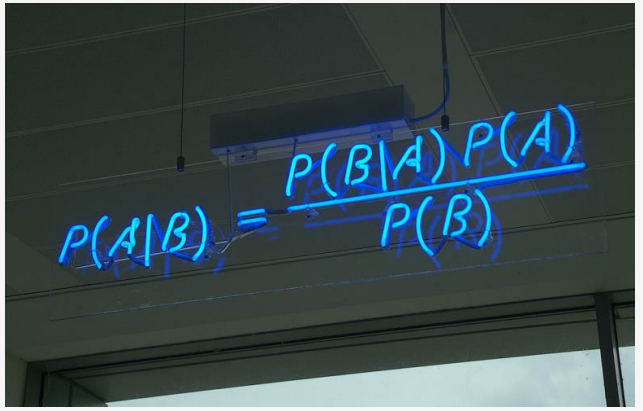
$$P(no\ rain\ or\ on\ time) = P(no\ rain) + P(on\ time) - P(no\ rain) * P(on\ time|no\ rain)$$

$$P(no\ rain\ or\ on\ time) = .7 + .40 - (.7 * .486) = .7598$$

So there is a 75.98% chance it will not rain or your umbrella arrives on time.

Chaining Conditional Probabilities

One of the cool things about Bayes is you can keep chaining several conditional probabilities together that affect an event of interest, assuming each condition is independent of the other conditions.



If I have an event A, and I want to calculate the probability it will occur given event B and event C have occurred, and I have the necessary conditional probabilities:

$$P(A|B \text{ and } C) = \frac{P(B|A) * P(C|A) * P(A)}{P(B|A) * P(C|A) * P(A) + P(B|not\ A) + P(C|not\ A) + P(not\ A)}$$

In the interest of time, I won't show the derivation of this formula and all the joint probability work but know this allows us to keep updating our beliefs with new evidence.

This is the foundational idea behind the Naïve Bayes machine learning algorithm.

Great Article: https://towardsdatascience.com/bayes-rule-applied-75965e4482ff

What is Naïve Bayes?

Naïve Bayes is a machine learning application of Bayes Theorem that merges probabilities of multiple features to predict a given category.

It is often used to classify text (e.g. email spam/not spam), which is the exercise the we will be doing today.

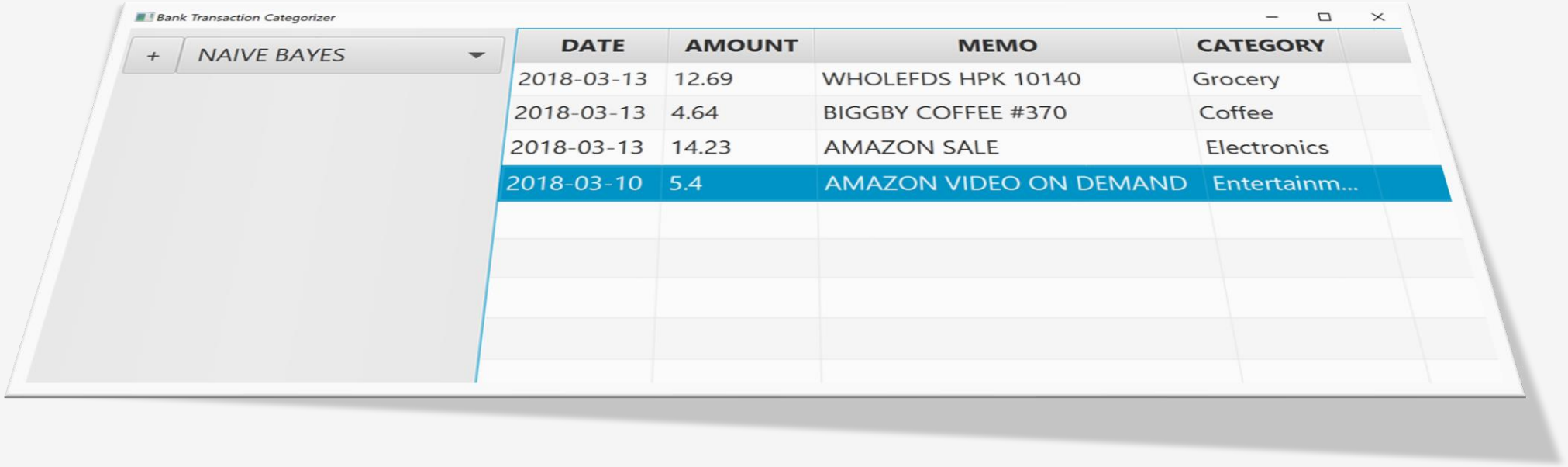
Naïve Bayes is effective because it learns quickly, even with little data.

Naive Bayes works by mapping probabilities of each individual feature occurring/not occurring for a given category (e.g. a word occurring in spam/not spam).

While commonly used for discrete variables like words, it can also be used with continuous variables using statistical distributions (we will not cover this today).

Thomas Bayes, the inventor of Bayes Theorem

Demo: Categorizing Bank Transactions



How to Build a Naïve Bayes Text Classifier

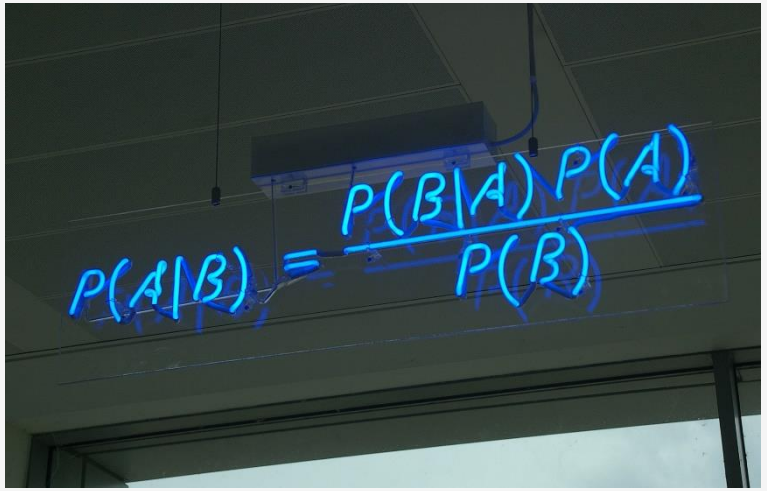
To predict a category for a new set of features:

1) For a given category (like ***spam, grocery,*** etc), combine the probabilities of each feature ***occurring*** and ***not occurring*** by multiplying them.

$$Occur\ Product = P_{f1} * P_{f2} * P_{f3}\ * \ldots *\ P_{fn}$$
$$Not\ Occur\ Product = (1\ - P_{f1}) * (1\ - P_{f2}) * (1\ - P_{f3}) *\ \ldots * (1\ - P_{fn})$$

2) Combine the probabilities above with this division operation:

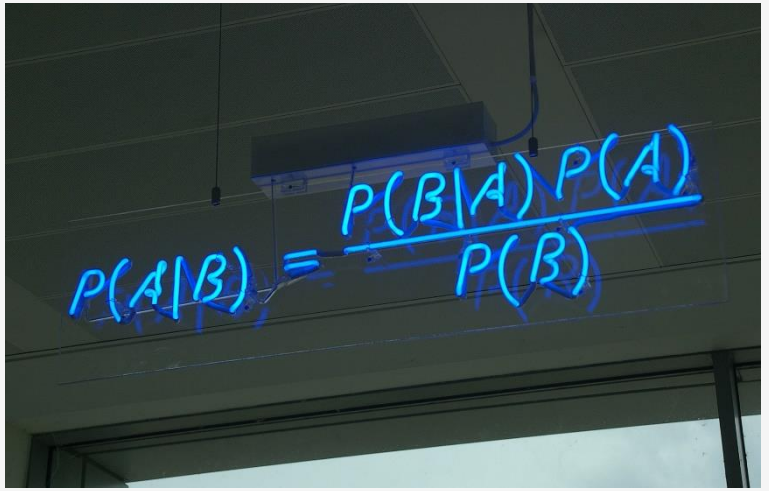
$$Combined\ Probability\ = \frac{(Occur\ Product)}{(Occur\ Product) + (Not\ Occur\ Product)}$$



How to Build a Naïve Bayes Text Classifier

3) Calculate the combined probability for every category, and the one that is the highest is the category you predict.

Easy right? But there is one complication…

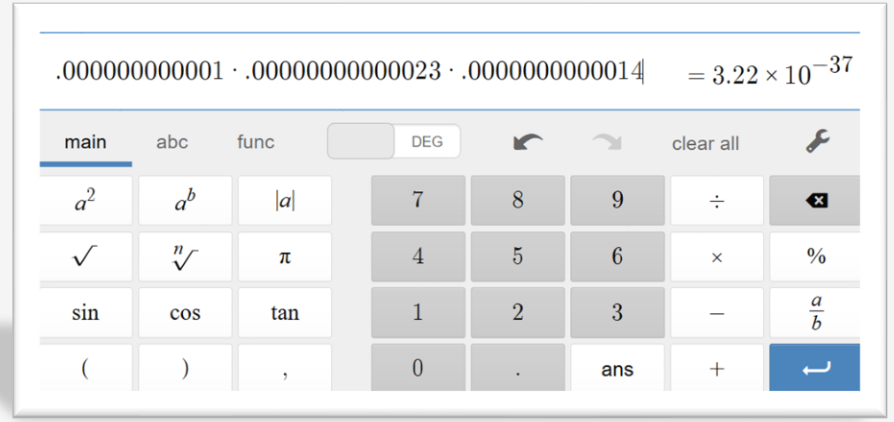


Dealing with Floating Point Underflow

Remember the floating point underflow issue, where multiplying small decimals gets so small the computer cannot handle them?

We can remedy the issue here by using logarithmic addition rather than multiplication.

Transform each probability with a $\log()$ or $\ln()$ function and then sum them, then call $\exp()$ to convert the result back!



Dealing with Floating Point Underflow

$$P_{fx} = Probability\ of\ feature\ \mathrm{X}$$

$$Occur\ Product = exp(log(P_{f1}) + log(P_{f2}) + log(P_{f3}) + log(P_{fn}))$$

$$Not\ Occur\ Product = \exp(log(1 - P_{f1}) + log(1 - P_{f2}) + log(1 - P_{f3}) + \ldots + log(1 - P_{fn}))$$

$$Combined\ Probability = \frac{(Occur\ Product)}{(Occur\ Product) + (Not\ Occur\ Product)}$$

One More Thing…

Never let a feature have a zero probability, so add some small constants to the numerator and denominator like 0.1 and 0.2 to make a default probability.

"Fudging" the numbers in machine learning can be acceptable since machine learning deals with estimates anyway.

You can make these numbers smaller if you like.

$$Feature\ Probability\ = \frac{0.1 + (Occur\ Probability)}{0.2 + \ (Occur\ Probability) + (Not\ Occur\ Probability)}$$

Final Naïve Bayes Formulation

$P_{fx} = Probability\ of\ feature$ x, *with a fudged constant in numerator and denominator*

$$Occur\ Product = exp(log(P_{f1}) + log(P_{f2}) + log(P_{f3}) + log(P_{fn}))$$

$$Not\ Occur\ Product = \exp(log(1 - P_{f1}) + log(1 - P_{f2}) + log(1 - P_{f3}) + \ldots + log(1 - P_{fn}))$$

$$Combined\ Probability = \frac{(Occur\ Product)}{(Occur\ Product) + (Not\ Occur\ Product)}$$

Exercise 2A

An auto manufacturing company Edison develops slick, cutting edge electric cars.

The founder Elom Nusk revealed an ”AI” diagnostics system that pre-emptively detects worn tires with 95% accuracy, so the driver is notified to replace them.

However at a given time only 5% of tires on the road need to be replaced, and the system is flagging positives 7% of the time.

What is the probability of the system correctly identifying a tire as worn?

Exercise 2A

An auto manufacturing company Edison develops slick, cutting edge electric cars.

The founder Elom Nusk revealed an "AI" diagnostics system that pre-emptively detects worn tires with 95% accuracy, so the driver is notified to replace them.

However at a given time only 5% of tires on the road are worn out and need to be replaced, and the system is flagging positives 7% of the time.

What is the probability of the system correctly identifying a tire as "worn"?

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

$$P(worn|positive) = \frac{P(positive|worn) * P(worn)}{P(positive)}$$

$$\mathbf{.6786} = \frac{.95 * .05}{.07}$$

67.86% chance the system correctly identifies a worn tire

Exercise 2B

An auto manufacturing company Edison develops slick, cutting edge electric cars.

The founder Elom Nusk revealed an “AI” diagnostics system that pre-emptively detects worn tires with 95% accuracy, so the driver is notified to replace them.

However at a given time only 5% of tires on the road need to be replaced, and the system is flagging positives 7% of the time.

What is the probability of the system falsely identifying a not worn tire as “worn”?

Exercise 2B

An auto manufacturing company Edison develops slick, cutting edge electric cars.

The founder Elom Nusk revealed an "AI" diagnostics system that pre-emptively detects worn tires with 95% accuracy, so the driver is notified to replace them.

However at a given time only 5% of tires on the road are worn out and need to be replaced, and the system is flagging positives 7% of the time.

What is the probability of the system falsely identifying a not worn tire as "worn"?

Find missing variable $P(positive|not\ worn)$ with Normalization Constant Formula

$P(A) = P(A|B) * P(B) + P(A|not\ B) * P(not\ B)$

$P(positive) = P(positive|worn) * P(worn) + P(positive|not\ worn) * P(not\ worn)$

$.07 = .95 * .05 + P(positive|not\ worn) * (1.0 - .05)$

$P(positive|not\ worn) = .0237$

Apply Bayes Theorem

$P(A|B) = P(B|A) * P(A) / P(B)$

$P(not\ worn|positive) = .3216 P(not\ worn|positive)$

$= (.0237 * .95) / .07 P(not\ worn\ |positive)$

$= P(positive|not\ worn) * P(not\ worn) / P(positive)$

32.16% chance a positively tested tire is not worn!

Section IV

Discovering the Binomial and Beta Distribution

Discussion: Measuring Uncertainty

You and an engineer colleague are testing a system to figure out its rate of failure, and you want at least 90% success rate.

Each test is expensive and time-consuming, but you were able to complete 10 tests.

8/10 of the tests succeeded but you are dissatisfied that just 10 tests already yielded two failures, and thus conclude an 80% failure rate.

TEST OUTCOME:

Discussion: Measuring Uncertainty

You propose going back to the drawing board, but the engineer wants to run more tests arguing this could be a fluke. The only way we will know for sure is to run more tests.

He argues “what if more tests might yield 90% or greater success? Consider the possible scenario below:”

POSSIBLE FUTURE TEST OUTCOME:

Discussion: Measuring Uncertainty

You *really* do not want to do more tests, but you consider the engineer's argument. After all, if you flip a coin 10 times and get 8 heads, it does not mean the coin is fixed at 80%.

DISCUSSION

Is the success rate really 80%?

How do we determine whether 10 tests are enough to discover the underlying probability of success?

If the true success rate is 90% or more, could we see 8 out of 10 successes by chance?

Why or why not?

Discussion: Measuring Uncertainty

Getting more data is always helpful, but sometimes expense and availability prevents us from doing so.

In situations where data is limited, a Bayesian approach can be advantageous over a Frequentist one.

Think about the probabilities of probabilities… if an event inherently has a 90% probability of occurring, what is the likelihood I would get 8/10 successes?

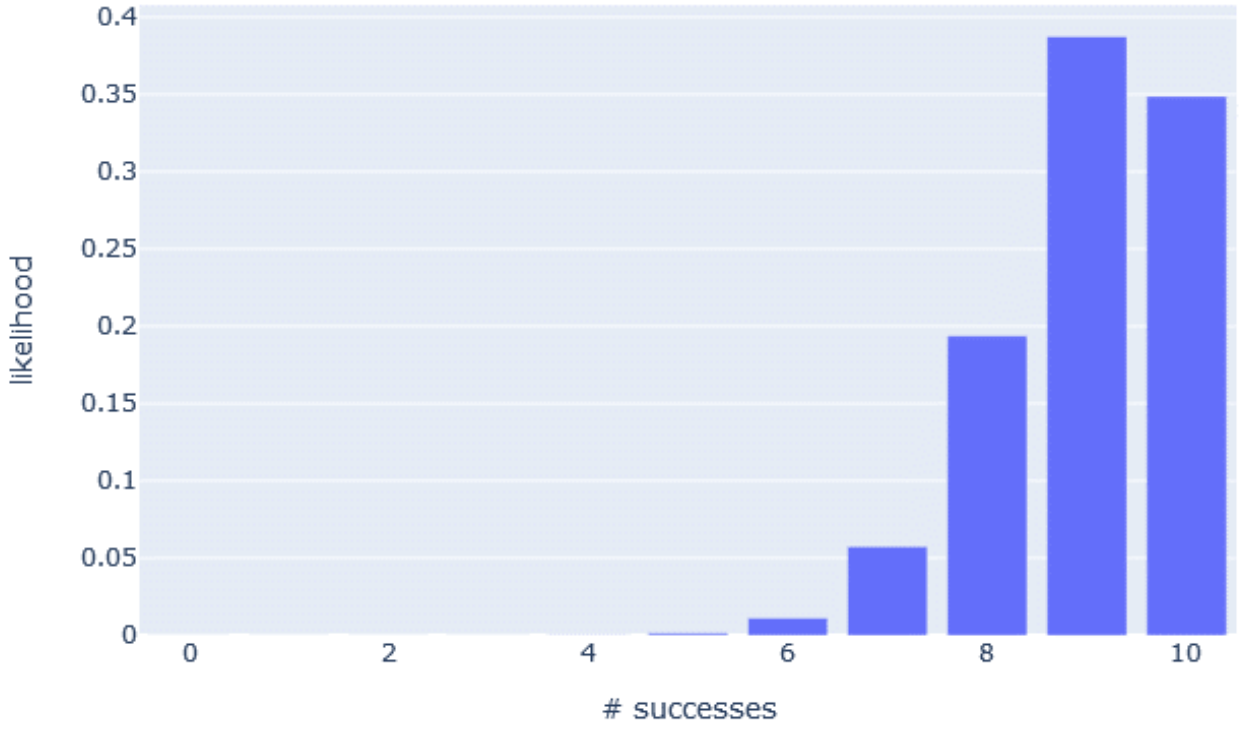
The Binomial Distribution

Let's first consider the **binomial distribution** to the right, which shows the *y likelihood* of *x successes* when the probability of success is 90% and there are 10 trials.

Each bar represents the probability of getting x successes.

If the probability of success is 90%, there is a 19.37% probability I will get 8 successes out of 10 trials.

Notice that all the outcomes add up to 1.0, or 100%, so this is a **probability distribution**.

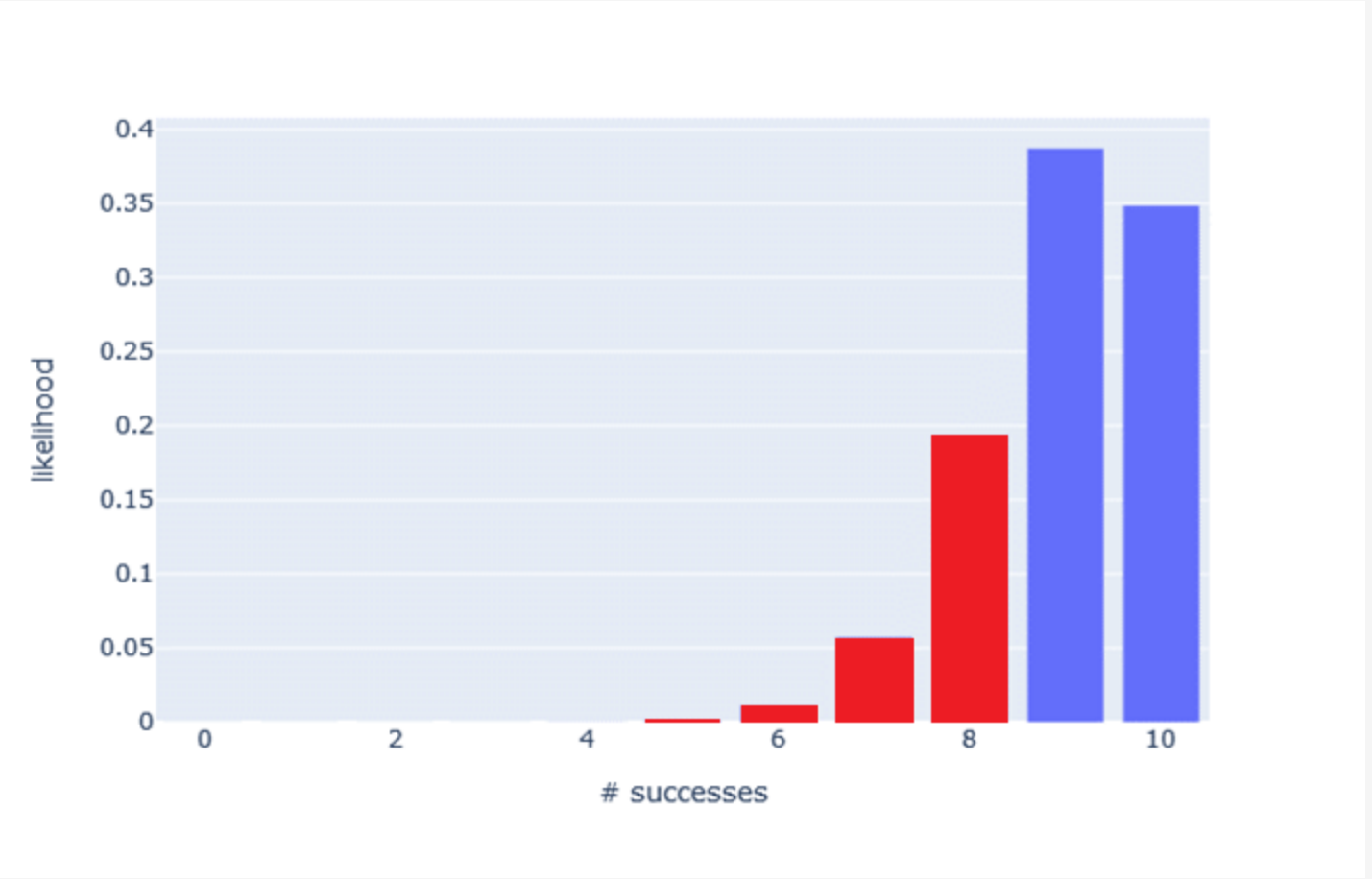


The Binomial Distribution

How would I calculate the probability of 8 or less successes?

Sum the probabilities (the bars in red) and you will find the probability of 8 or less successes to be 26.39%

If we are interested in having 90% or greater success, then the probability of seeing less than 90% success is 26.39% **EVEN IF** each outcome has a 90% success rate.



Binomial Distribution from Scratch in Python

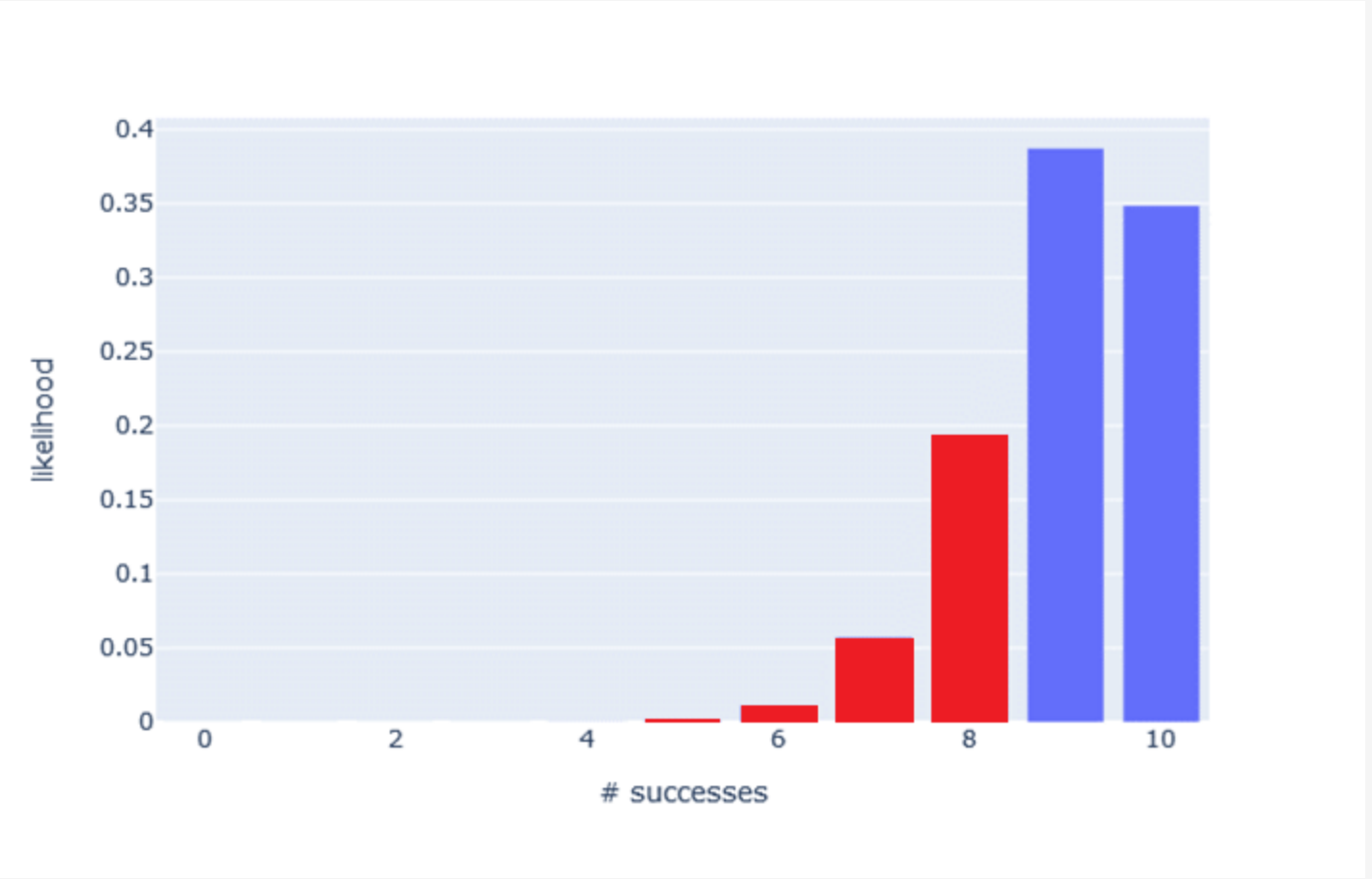
```python
# Factorials multiply consecutive descending integers down to 1
# EXAMPLE: 5! = 5 * 4 * 3 * 2 * 1
def factorial(n: int):
    f = 1
    for i in range(n):
        f *= (i + 1)
    return f

# Generates the coefficient needed for the binomial distribution
def binomial_coefficient(n: int, k: int):
    return factorial(n) / (factorial(k) * factorial(n - k))


# Binomial distribution calculates the probability of k events out of n trials
# given the p probability of k occurring
def binomial_distribution(n: int, k: int, p: float):
    return binomial_coefficient(n, k) * (p ** k) * (1.0 - p) ** (n - k)



p = .9
n = 10

prob_eight_or_less_successes = 0.0

for k in range(0,9):
    prob_of_k_successes = binomial_distribution(n, k, p)
    print("PROB {} SUCCESSES: {}".format(k, prob_of_k_successes))
    prob_eight_or_less_successes += prob_of_k_successes

print("PROB 8 OR LESS SUCCESSES: {}".format(prob_eight_or_less_successes))
```

The Binomial Distribution

So there is a 26.39% chance the engineer is correct, ***ASSUMING*** each success has a 90% success rate.

This is informative showing it's 26.39% possible to see 8 or less successes with an underlying 90% probability.

But notice that our model ***ASSUMED*** the probability of success is 90%. What if we flipped the question and explored other probabilities besides 90%, and how likely each would produce 8/10 successes?



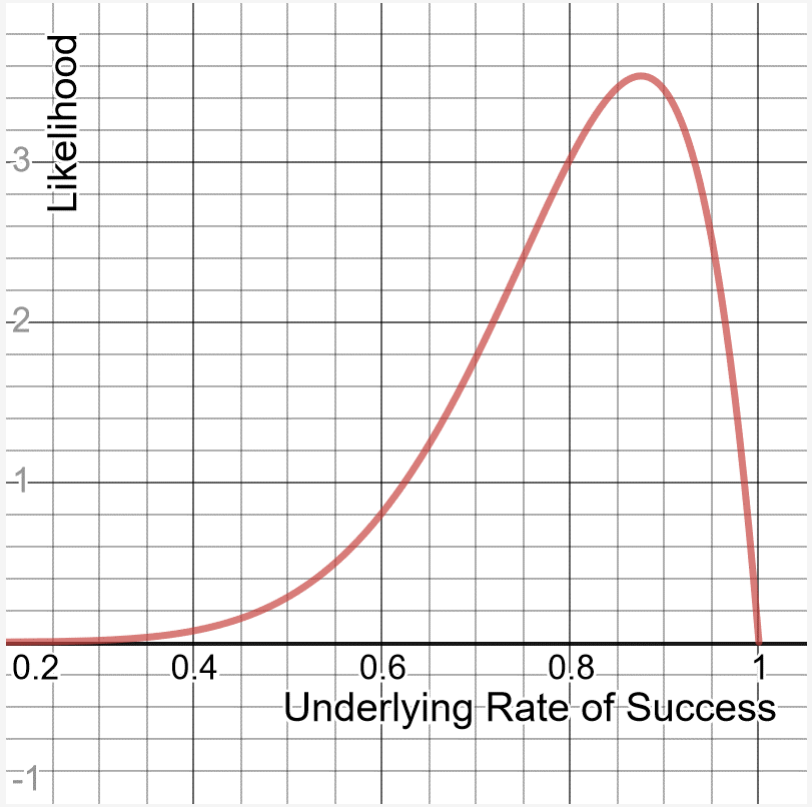
The Beta Distribution

Here's another way to look at this: consider all underlying rates of success and how likely each is to produce 8/10 successes.

This is the **beta distribution**, which allows us to see the probabilities of probabilities given so many successes and failures.

To the right shows the range of likelihoods for each actual rate of success, when 10 trials yields 8 successes.

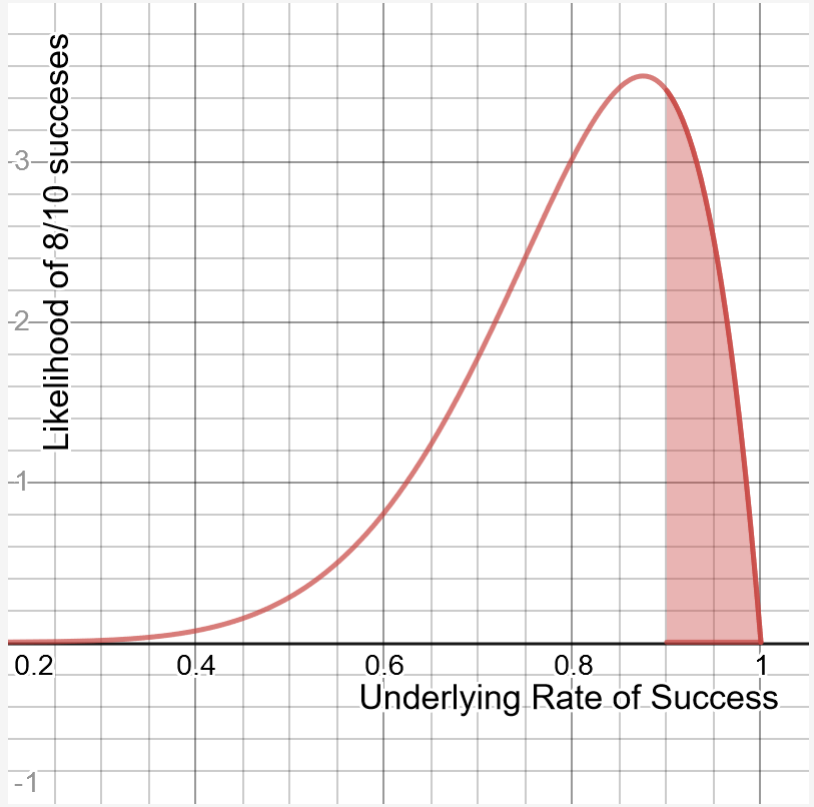
The beta distribution accepts two arguments, the **alpha** which is the number of successes, and the **beta** which is the number of failures.



https://www.desmos.com/calculator/zyranflxo4

The Beta Distribution

Given 8/10 successes, the probability that the underlying rate of success is 90% or higher is 22.5%.

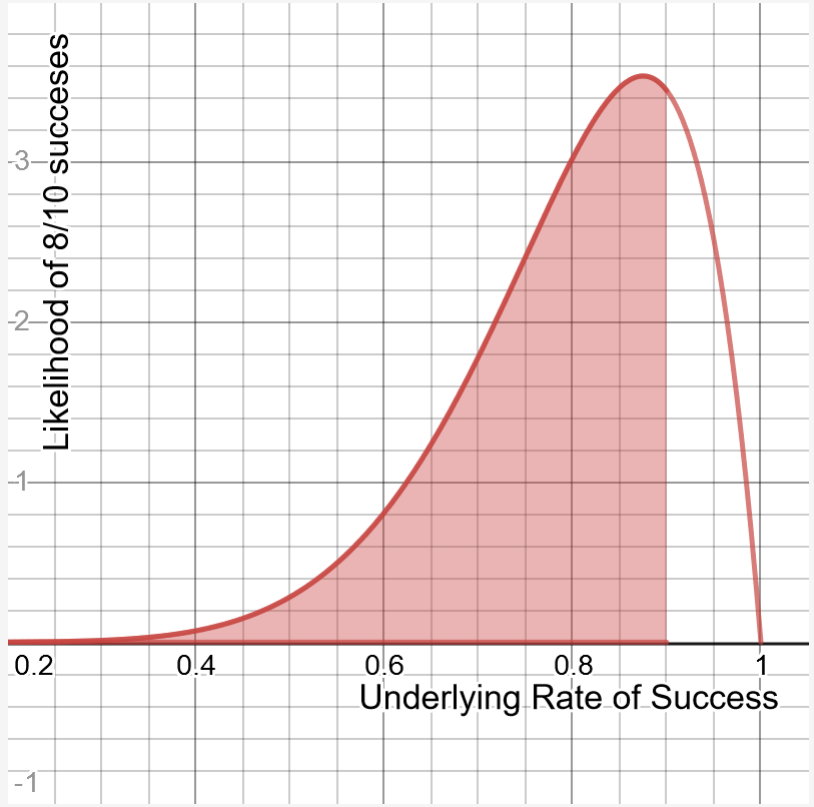


https://www.desmos.com/calculator/cakj42wlie

The Beta Distribution

Given 8/10 successes, the probability that the underlying rate of success is 90% or higher is 22.5%.

But the probability of the underlying rate being less than 90% is 77.5%.



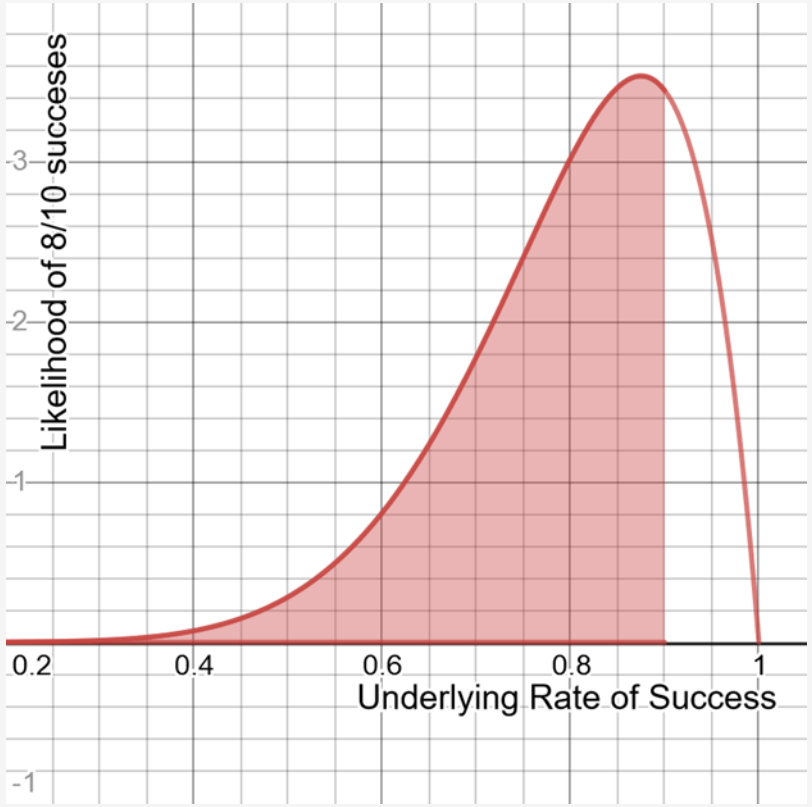
https://www.desmos.com/calculator/cakj42wlie

The Beta Distribution

Given 8/10 successes, the probability that the underlying rate of success is 90% or higher is 22.5%.

But the probability of the underlying rate being less than 90% is 77.5%.

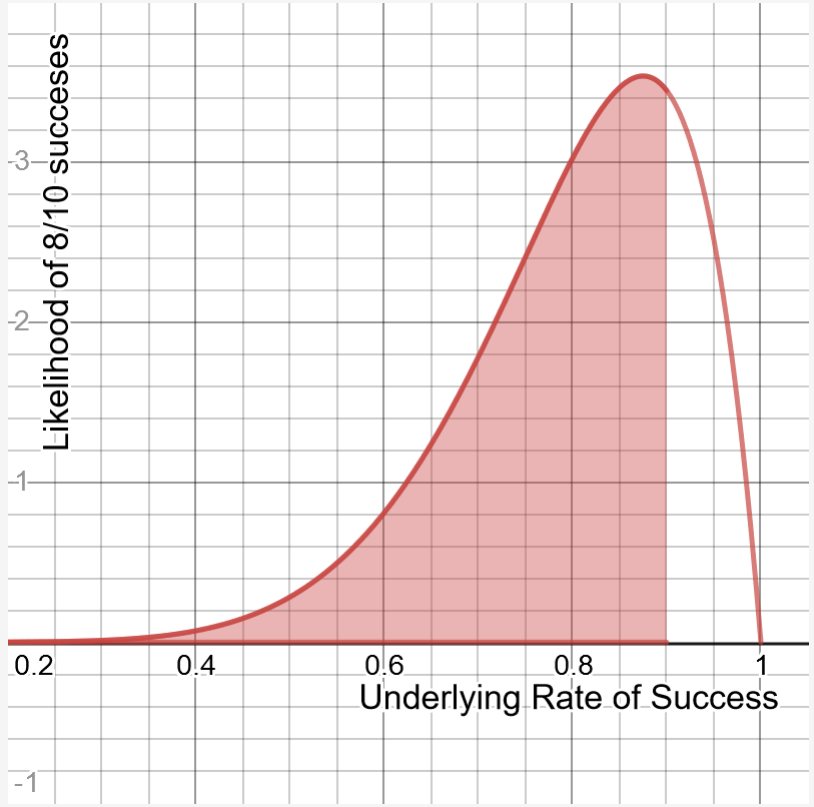
Given these numbers, is it worth doing more tests?



https://www.desmos.com/calculator/cakj42wlie

The Beta Distribution

What we essentially have found is there is a 77.5% chance the underlying success rate is below the required threshold of 90%.

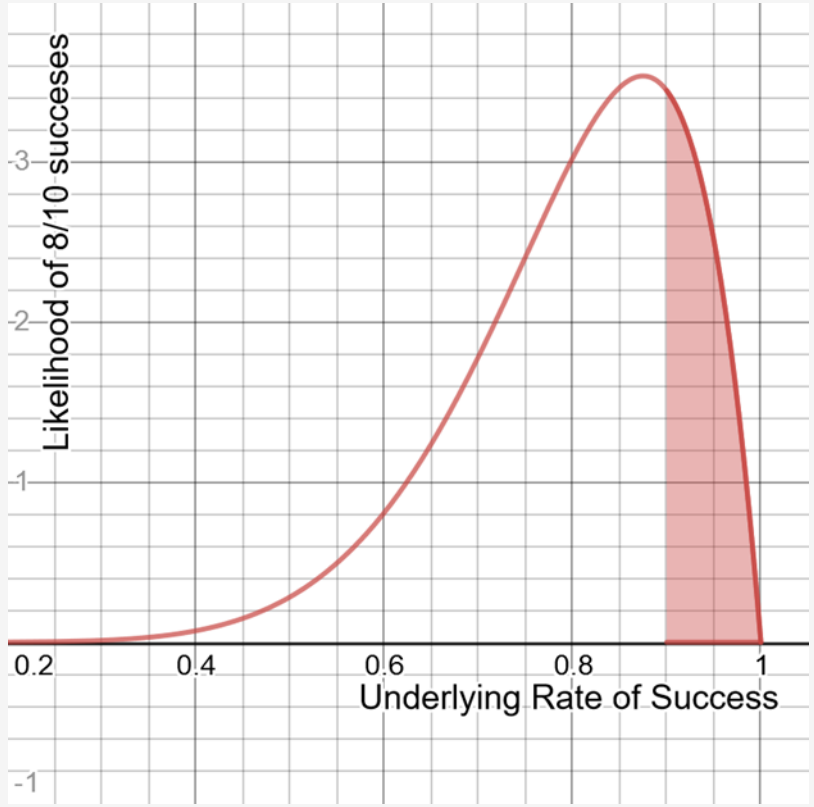


https://www.desmos.com/calculator/cakj42wlie

The Beta Distribution

What we essentially have found is there is a 77.5% chance the underlying success rate is below the required threshold of 90%.

But if we want to gamble on that other 22.5% and order a few more tests to confirm, we can make that call too.



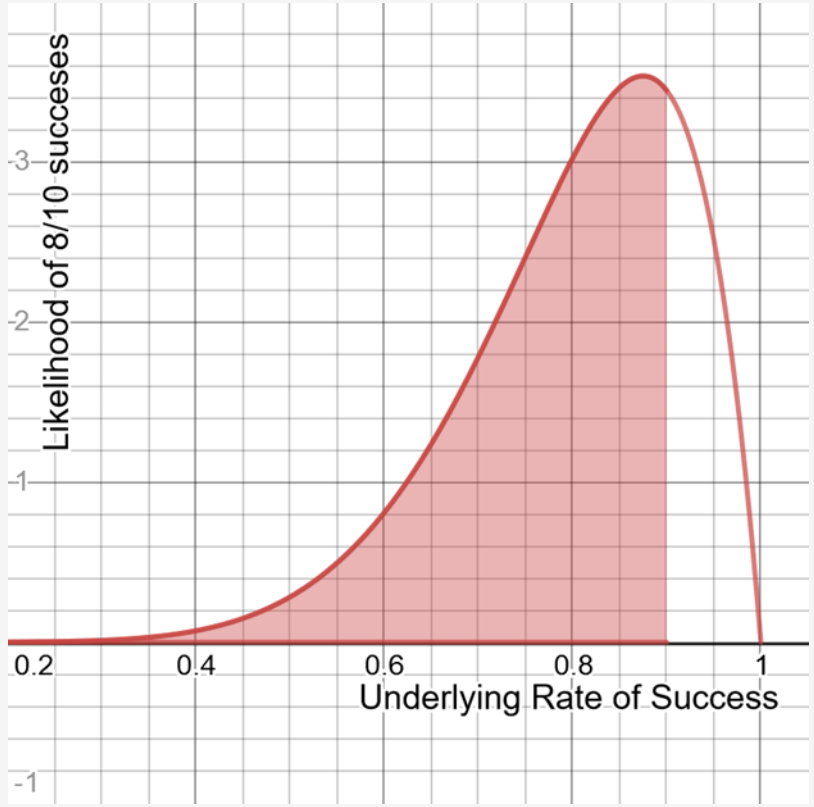
https://www.desmos.com/calculator/cakj42wlie

The Beta Distribution

What we essentially have found is there is a 77.5% chance the underlying success rate is below the required threshold of 90%.

But if we want to gamble on that other 22.5% and order a few more tests to confirm, we can make that call too.

But it may be a risk! Judging just by the numbers, we may just want to abandon testing because there's a 77.5% chance the test truly failed to meet the 90% success rate.



https://www.desmos.com/calculator/cakj42wlie

Using Beta Distribution to Update Predictions

Let's say our CFO granted funding for 26 more tests and wanted to take that risk.

Accumulating on top of the 8 successes and 2 failures, we now have 30 successes and 6 failures.

Notice our beta distribution got a lot narrower, giving us a stronger **credible interval**.

The probability of the underlying success rate being less than 90% is now 86.8%, leaving only 14.2% possibility that our system meets requirements.

At this point, it may be time to abandon testing and go back to the drawing board, unless you want to continue gambling on that 14.2%!

https://www.desmos.com/calculator/xnpjwqclnc

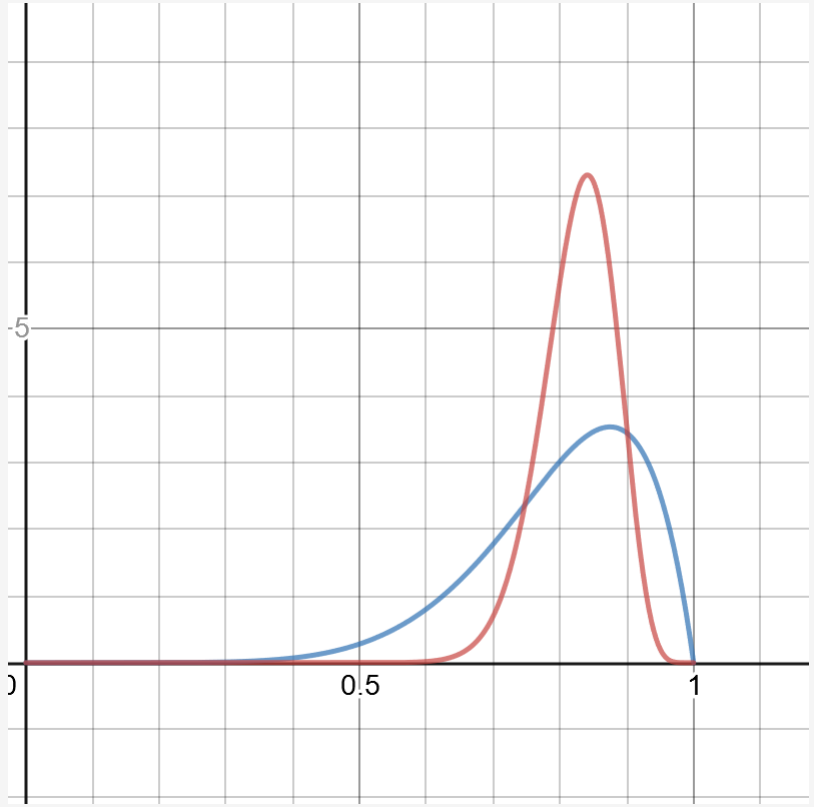
Using Beta Distribution to Update Predictions

This shows how versatile the Beta distribution is in updating predictions based on new data.

When we get more trials and get more successes/failures, the beta distribution will start to narrow more confidently on a smaller range.

You can even use the beta distribution to quantify subjective beliefs (using the gambling odds exercise we did earlier) and then merge predictions with different beliefs and data.

Starting the beta distribution without any prior data or beliefs is controversial, but you can start with an alpha=1, beta=1 initialization which has a neutral flat line that is easily overridden by new data.



https://www.desmos.com/calculator/iji1abdcov

Integrating Functions

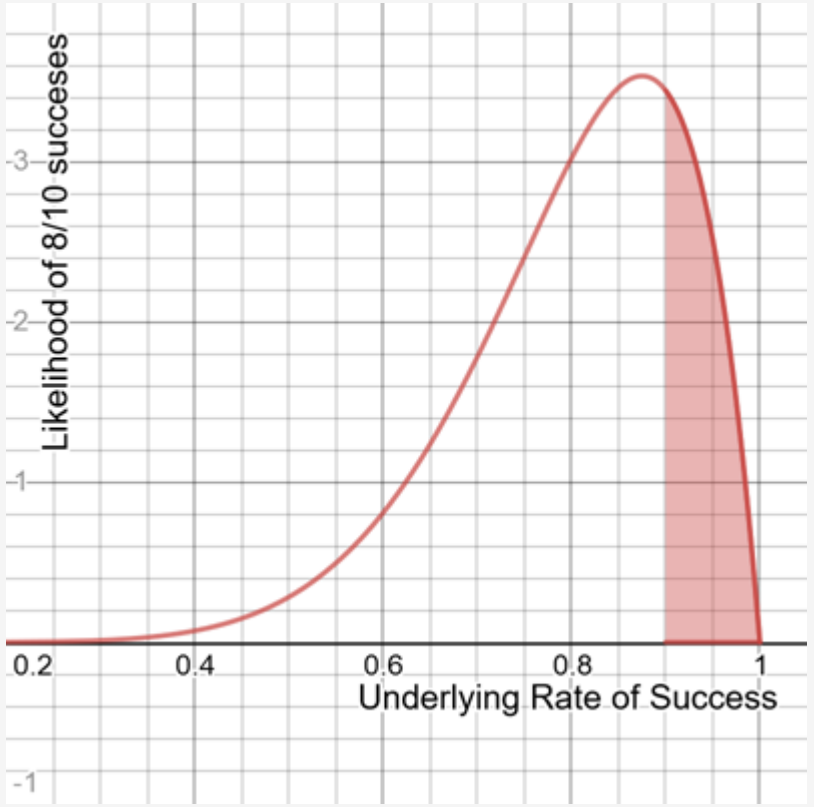
Before we implement the beta distribution from scratch in Python, let's first talk about **integration**, or finding the areas under a curve.

The binomial distribution was easy to integrate, as each probability was on a discrete value and we can just sum them.

But continuous distributions are a different animal, as they are not discrete but rather one continuous, curvy line with infinitely many values.

The x-axis and y-axis are both decimals, and the y-axis does not represent probabilities but rather "density" of likelihood.

To find a probability, we use the area under the curve for a range of x-values.



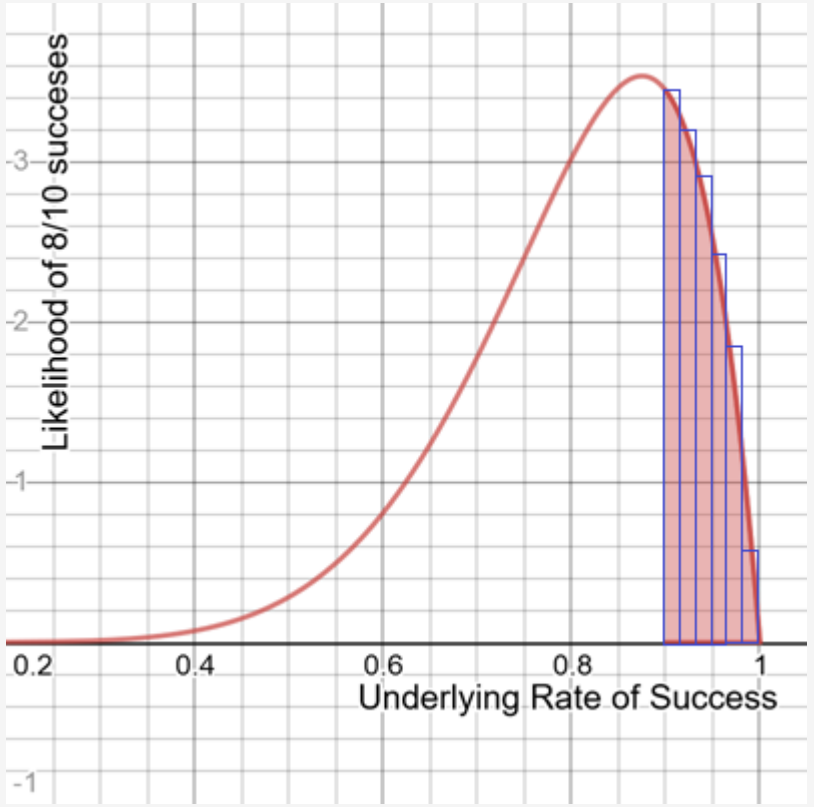
Integrating Functions

Imagine we packed 6 rectangles under the curve area we are interested in, where the midpoint of each rectangle touches the curve.

Finding the area of a rectangle is easy (area = length * width), and we can sum the areas of these rectangles to give an approximation of the area under the curve.

If we packed 1000 smaller rectangles we would get an even more precise approximation.

An approximation is not a bad thing since computers have to approximate with limited memory and computation anyway.



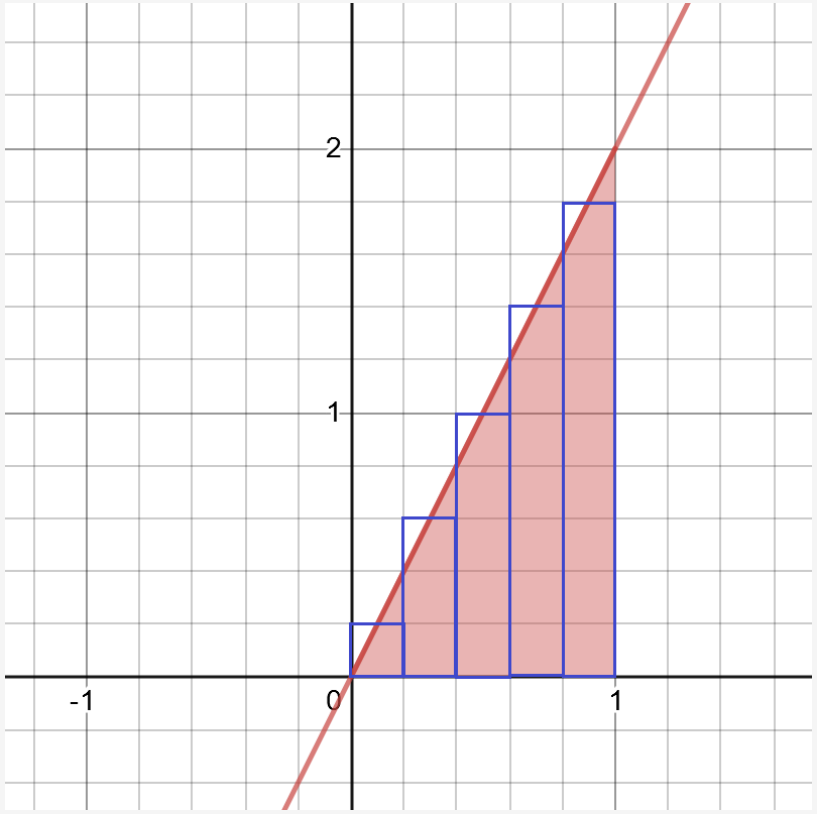
Integrating Functions in Python

Here is a simple function in Python that will find the area under any function curve **f** for a range between **a** and **b**, with a number of rectangles **n**.

```
def approximate_integral(a, b, n, f):
    delta_x = (b - a) / n
    total_sum = 0

    for i in range(1, n + 1):
        midpoint = 0.5 * (2 * a + delta_x * (2 * i - 1))
        total_sum += f(midpoint)

    return total_sum * delta_x
```

It works by packing the range with rectangles that extend their midpoint to touch the function curve, and the more rectangles you use the more precise it will be (with a diminishing return).

You can then find the area for ANY Python function that returns a number.




```
def my_function(x):
    return 2 * x

area = approximate_integral(a=0, b=1, n=5, f=my_function)

print(area)
```

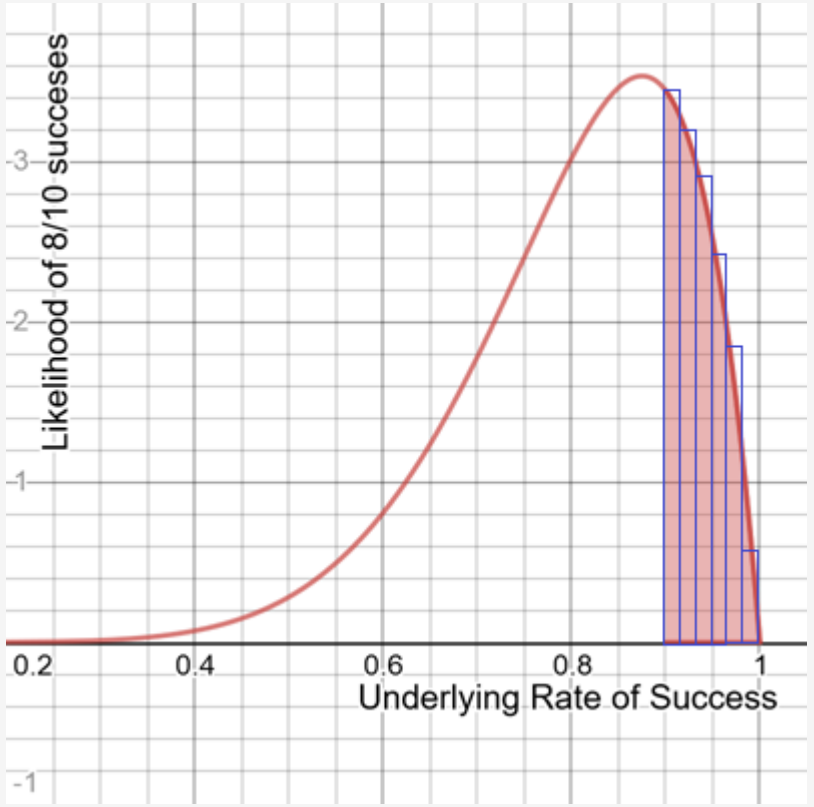
Integrating the Beta Distribution

We can use the integration function to find the area under our Beta distribution curve for a given range.

When we create a function for a probability distribution that integrates a given range of values, it is known as a **quantile function.**

To the right shows an integration with 6 rectangles to calculate the probability our underlying rate of success is between 90% and 100%.

Of course, we should probably use 1000 rectangles or more to have adequate precision.



Beta Distribution from Scratch in Python

```python
# Factorials multiply consecutive descending integers down to 1
# EXAMPLE: 5! = 5 * 4 * 3 * 2 * 1
def factorial(n: int):
    f = 1
    for i in range(n):
        f *= (i + 1)
    return f

def approximate_integral(a, b, n, f):
    delta_x = (b - a) / n
    total_sum = 0

    for i in range(1, n + 1):
        midpoint = 0.5 * (2 * a + delta_x * (2 * i - 1))
        total_sum += f(midpoint)

    return total_sum * delta_x


def beta_distribution(x: float, alpha: float, beta: float) -> float:
    if x < 0.0 or x > 1.0:
        raise ValueError("x must be between 0.0 and 1.0")

    numerator = x ** (alpha - 1.0) * (1.0 - x) ** (beta - 1.0)
    denominator = (1.0 * factorial(alpha - 1) * factorial(beta - 1)) / (1.0 * factorial(alpha + beta - 1))

    return numerator / denominator


def beta_quantile(lower: float, upper: float, alpha: float, beta: float):
    return approximate_integral(lower, upper, 1000, lambda x: beta_distribution(x, alpha, beta))

greater_than_90 = beta_quantile(.90, 1.0, 8, 2)
less_than_90 = 1.0 - greater_than_90

print("GREATER THAN 90%: {}, LESS THAN 90%: {}".format(greater_than_90, less_than_90))
```

Exercise 3A

You just flipped a coin 5 times and got 3 heads and 2 tails.

What is the probability the coin is rigged to get heads 80% or more?

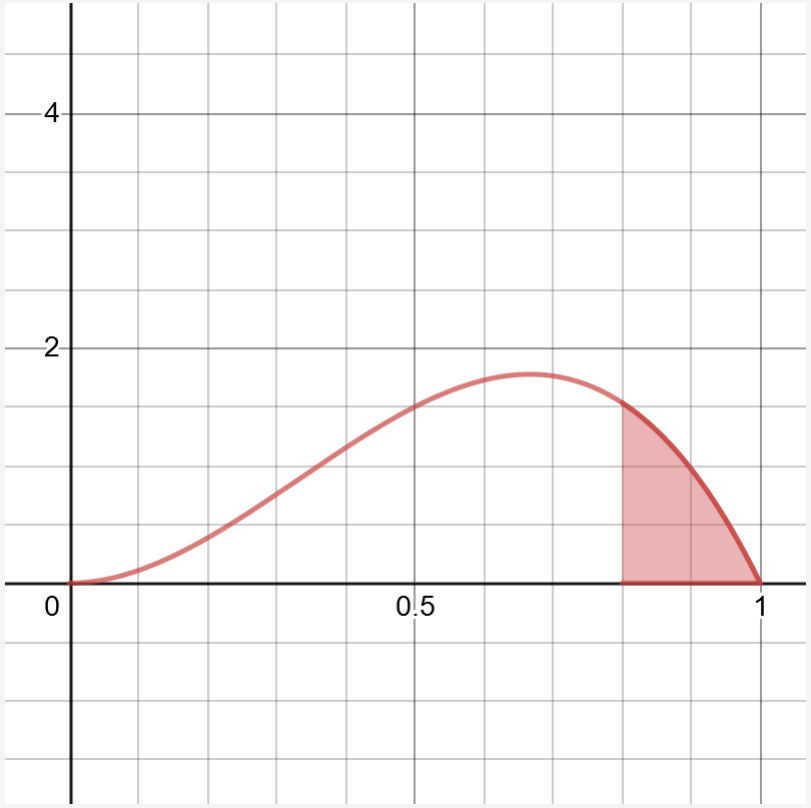
Exercise 3A

You just flipped a coin 5 times and got 3 heads and 2 tails.

What is the probability the coin is rigged to get heads 80% or more?

There is an 18.08% probability that the coin is rigged to get heads at least 80% of the time.

```
greater_than_80 = beta_quantile(.80, 1.0, 3, 2)

print("GREATER THAN 80%: {}".format(greater_than_80))
```



https://www.desmos.com/calculator/q92prchh36

Exercise 3B

You flipped the coin more, and now have gotten heads a total of 15 times and tails 4 times.

What is now the probability the coin is rigged to get heads at least 80% of the time?

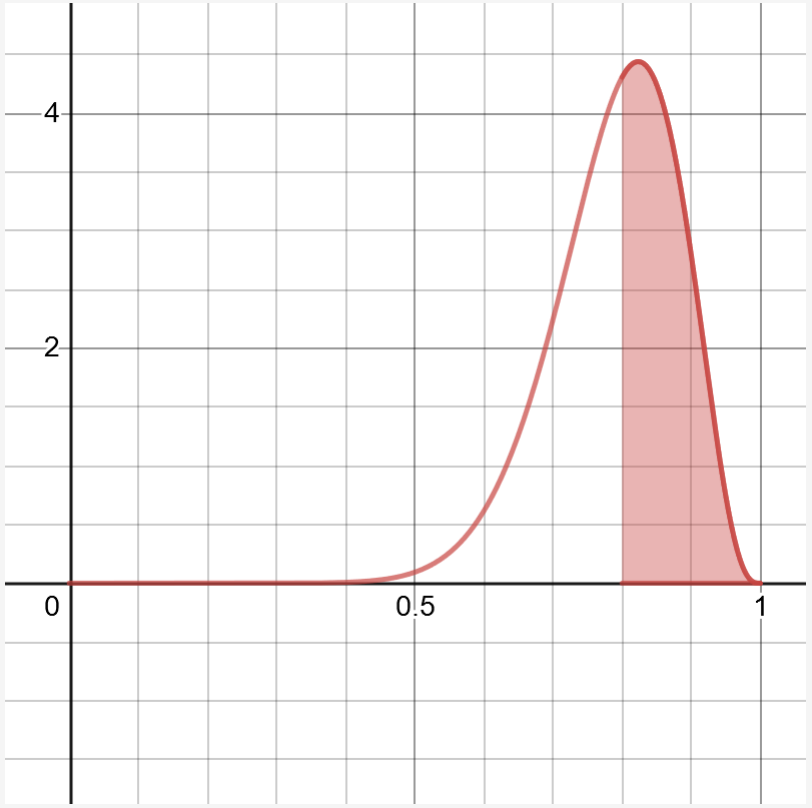
Exercise 3B

You flipped the coin more, and now have gotten heads a total of 15 times and tails 4 times.

What is now the probability the coin is rigged to get heads 80% or more?

There is now a 49.897% probability that the coin is rigged to get heads at least 80% of the time.

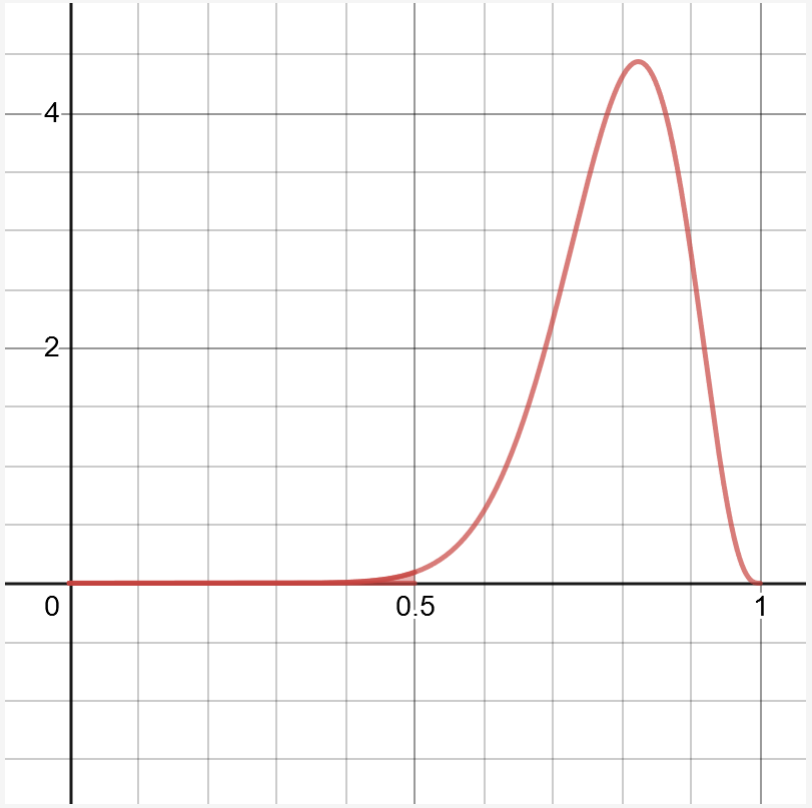
```
greater_than_80 = beta_quantile(.80, 1.0, 15, 4)

print("GREATER THAN 80%: {}".format(greater_than_80))
```



https://www.desmos.com/calculator/tpwvwasjfi

Exercise 3C

Do you think this coin has any good probability of being fair?

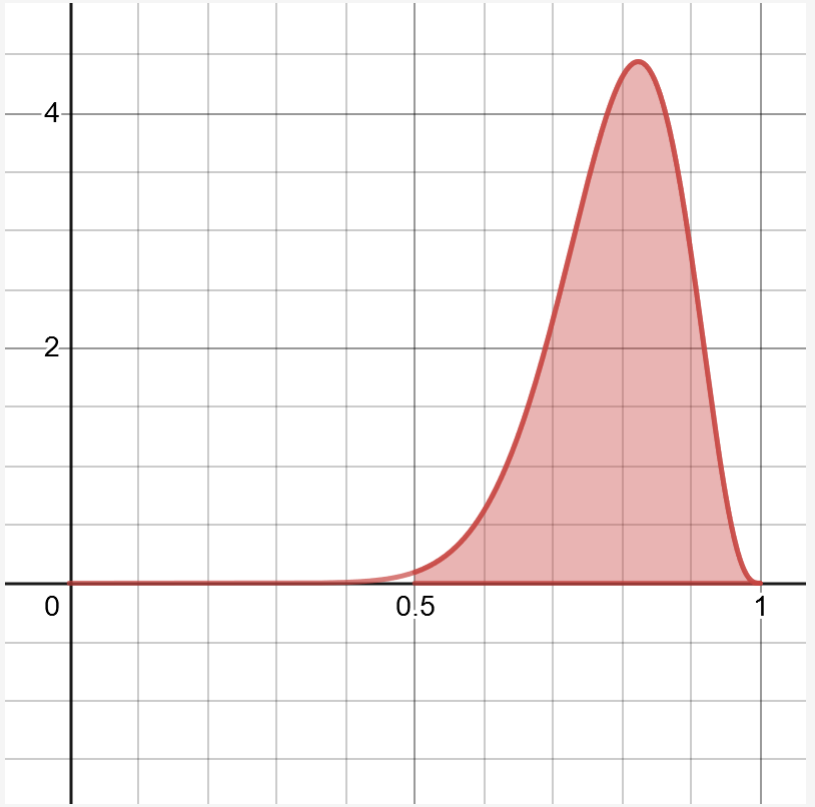


Exercise 3C

Do you think this coin has any good probability of being fair?

Considering 99.62% of the density is above 50%, this coin is highly unlikely to be fair.

```
greater_than_50 = beta_quantile(.50, 1.0, 15, 4)

print("GREATER THAN 50%: {}".format(greater_than_50))
```



Exercise 3D

You have 137 passengers booked on a flight from Las Vegas to Dallas.

However it is Las Vegas on a Sunday morning and you estimate each passenger is 40% likely to not show up.

You are trying to figure out how many seats to overbook so the plane does not fly empty.

What is the most likely number of passengers to not show up?

Exercise 3D

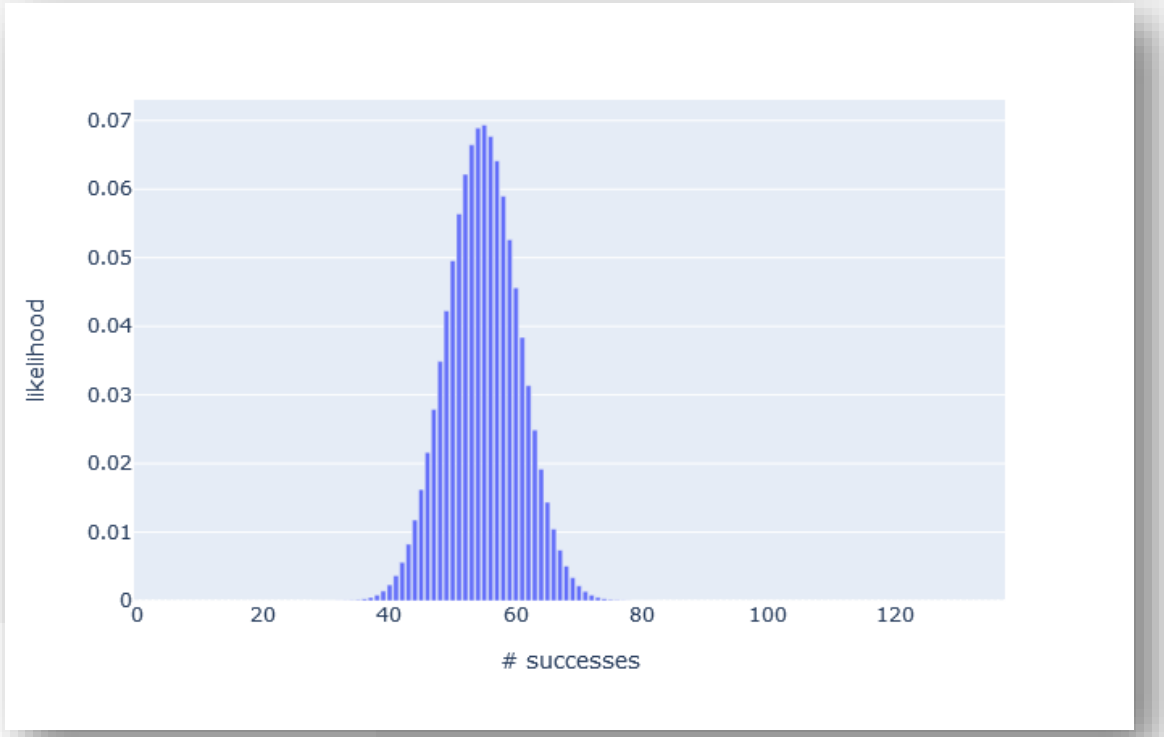
You have 137 passengers booked on a flight from Las Vegas to Dallas.

However, it is Las Vegas on a Sunday morning and you estimate each passenger is 40% likely to not show up.

You are trying to figure out how many seats to overbook so the plane does not fly empty.

What is the most likely number of passengers to not show up?

Most likely to be missing 55 passengers with 6.93% probability.




```
p = .4
n = 137

greatest_p = 0
greatest_k = 0

for k in range(1, 138):
    p_k_successes = binomial_distribution(n, k, p)
    print("PROB {} NO SHOWS: {}".format(k, p_k_successes))

    if greatest_p < p_k_successes:
        greatest_p = p_k_successes
        greatest_k = k

print("BEST OVERBOOK BET: {0}% CHANCE MISSING {1} PASSENGERS, OVERBOOK {1} SEATS".format(greatest_p, greatest_k))
```

Exercise 3D

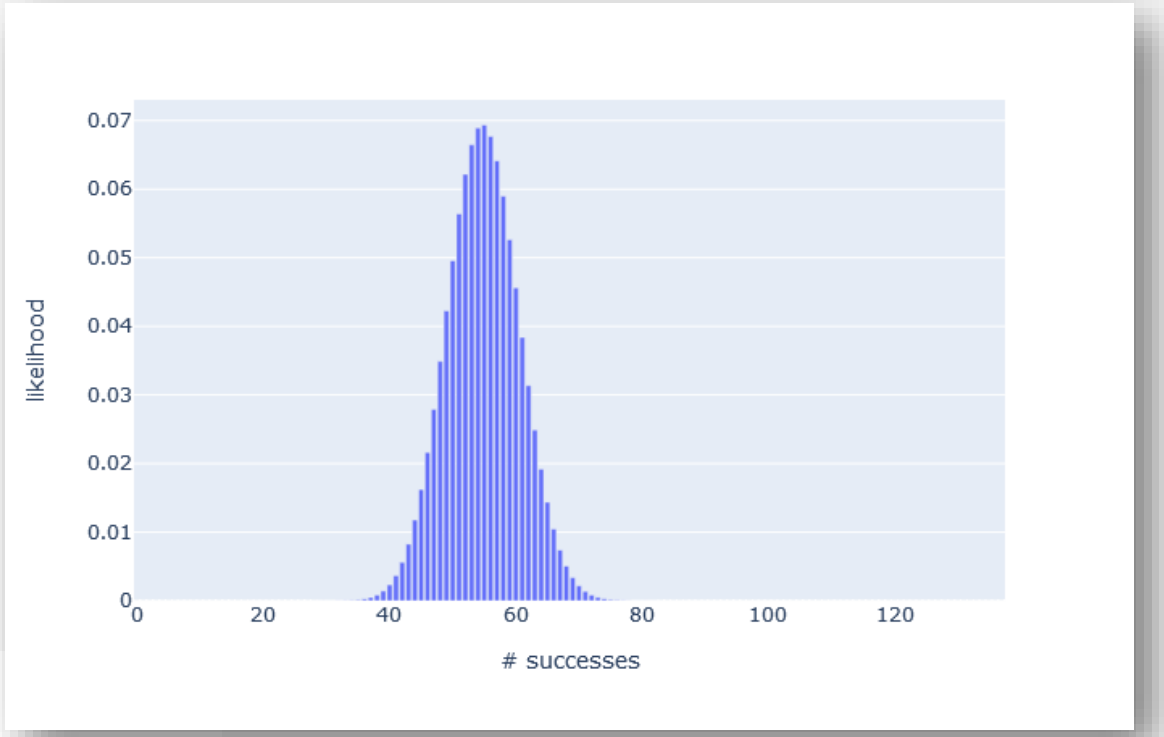
You have 137 passengers booked on a flight from Las Vegas to Dallas.

However, it is Las Vegas on a Sunday morning and you estimate each passenger is 40% likely to not show up.

You are trying to figure out how many seats to overbook so the plane does not fly empty.

What is the most likely number of passengers to not show up?

If you want to use SciPy and Plotly to generate chart to the right:




```
import numpy as np
import pandas as pd
import plotly.express as px
from scipy.stats import binom, beta

df = pd.DataFrame({"# successes" : np.linspace(0,137,138),
                   "likelihood" : np.array([binom.pmf(x, 137, .40) for x in np.linspace(0,137,138)])}
                  )

fig = px.bar(df, x="# successes", y="likelihood")

fig.show()
```

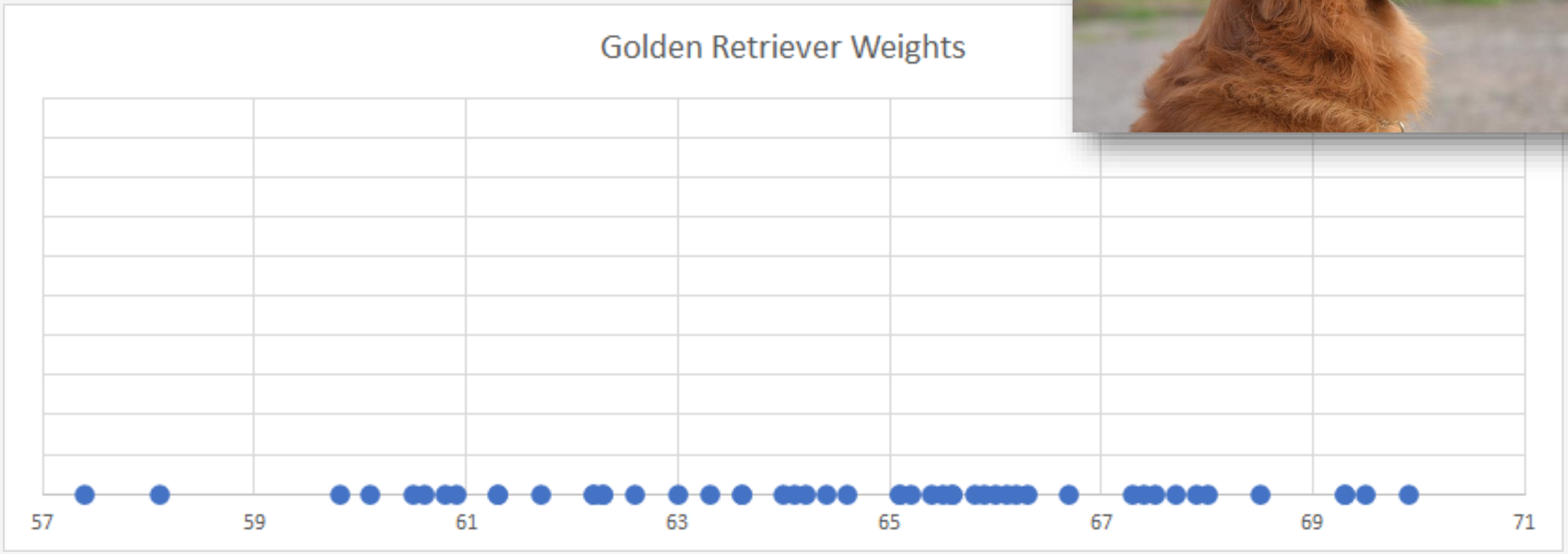
Section V

Discovering the Normal Distribution

Discovering the Normal Distribution

Below is a number line showing recorded weights (in pounds) for 50 adult golden retrievers.

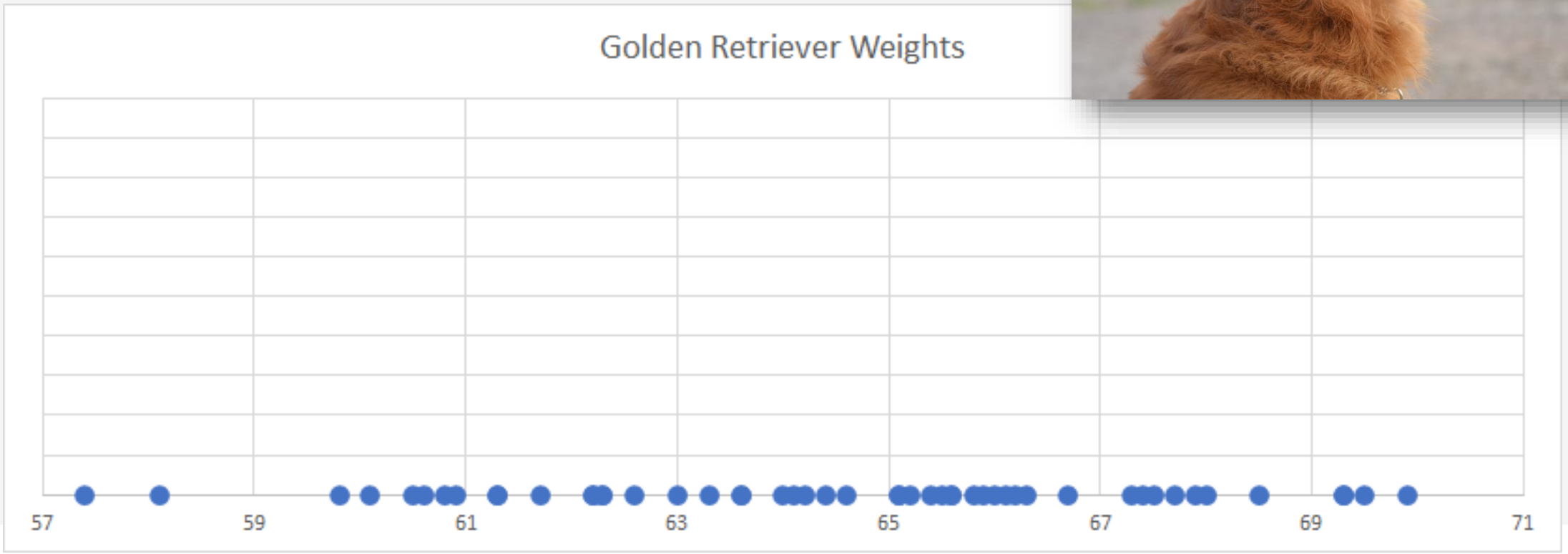
What do you notice about this data?



Discovering the Normal Distribution

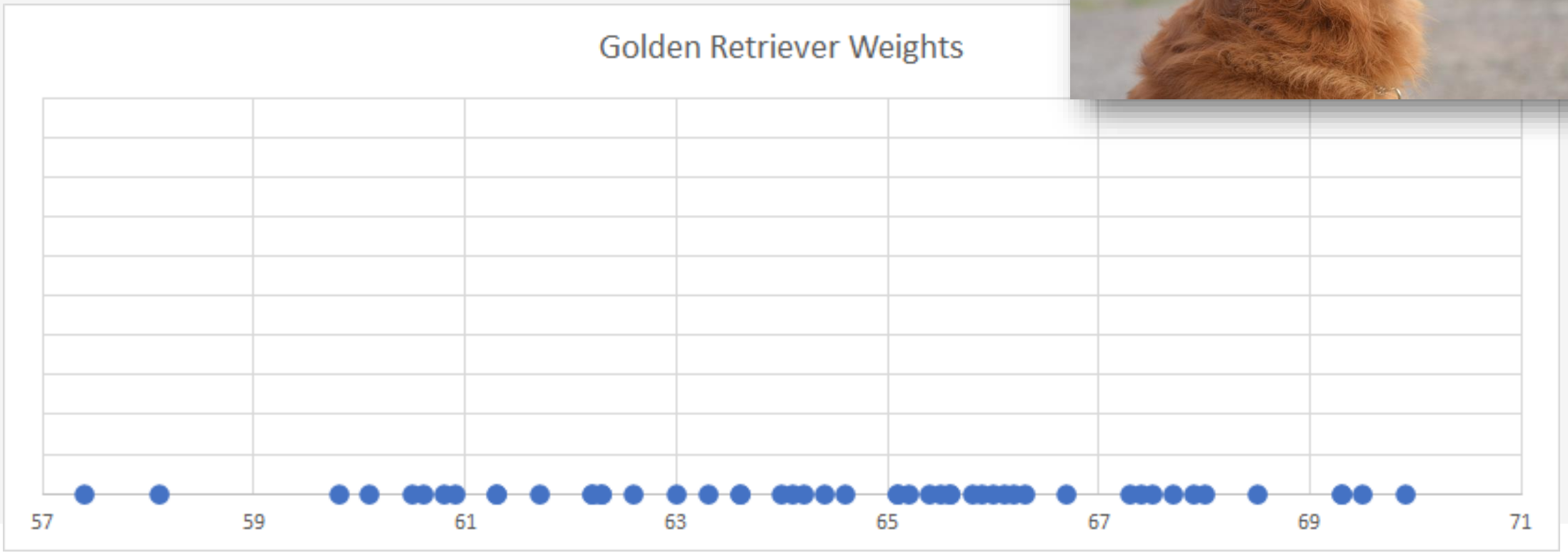
Notice that towards the “center” there are more points clustered together, while the “tails” on either end have less points.

We can infer that we are more likely to see more golden retrievers in the 61-67 lb range as opposed to lower or higher weights like 57 and 71.



Discovering the Normal Distribution

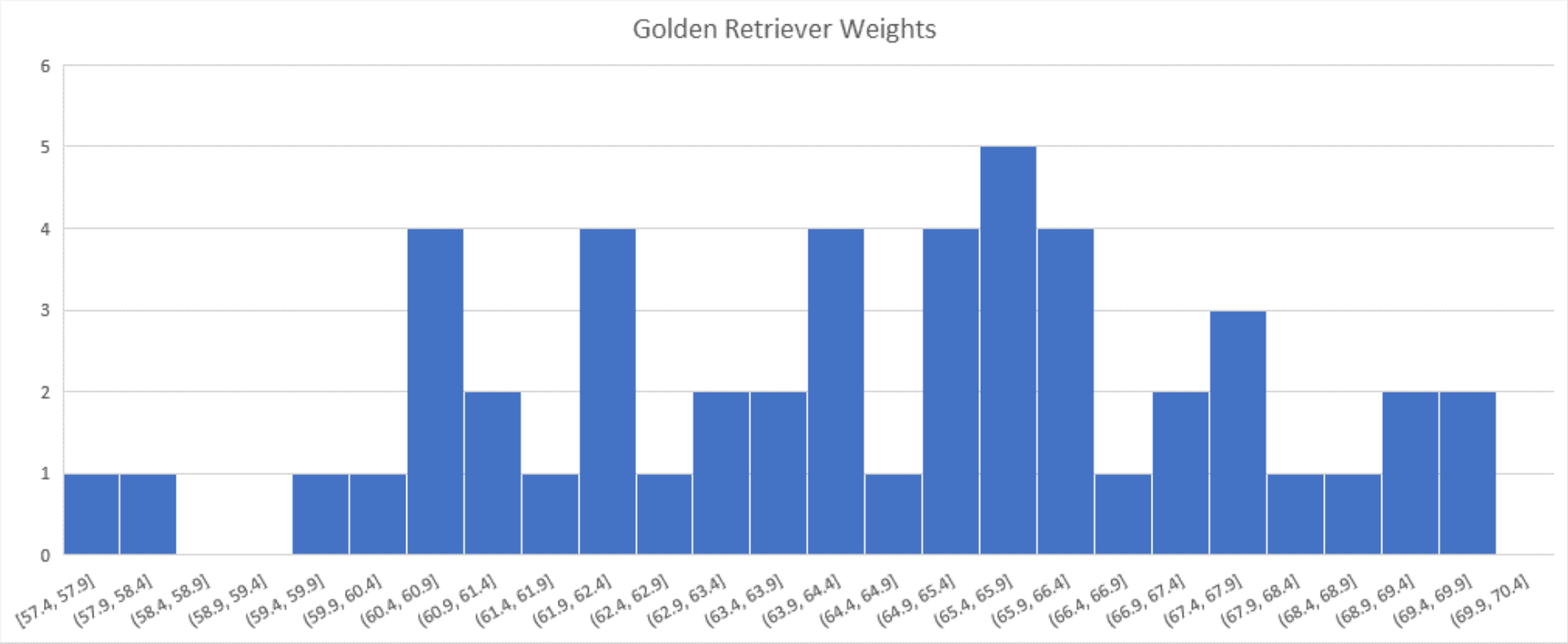
Is there a better way we can visualize this data? And get an idea where we would see more golden retrievers in terms of their weight?



Discovering the Normal Distribution

We can try to **bin** up points on equally-sized ranges, and then count the number of instances as a bar chart to create a **histogram**.

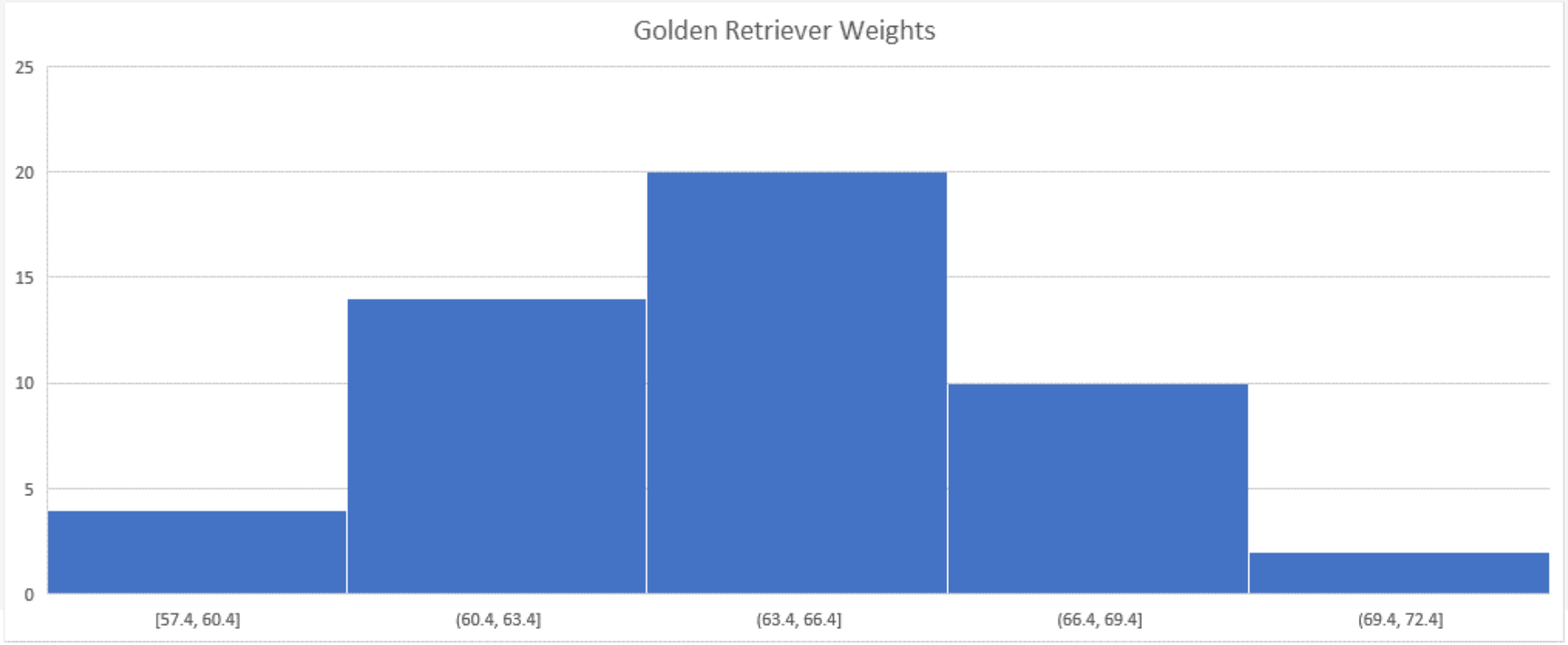
Unfortunately, we don't have an infinite amount of data so making the bin ranges too small will not reveal much about the underlying distribution.



Discovering the Normal Distribution

But if we make the bin sizes just right, we might be able to see a shape resembling a probability distribution.

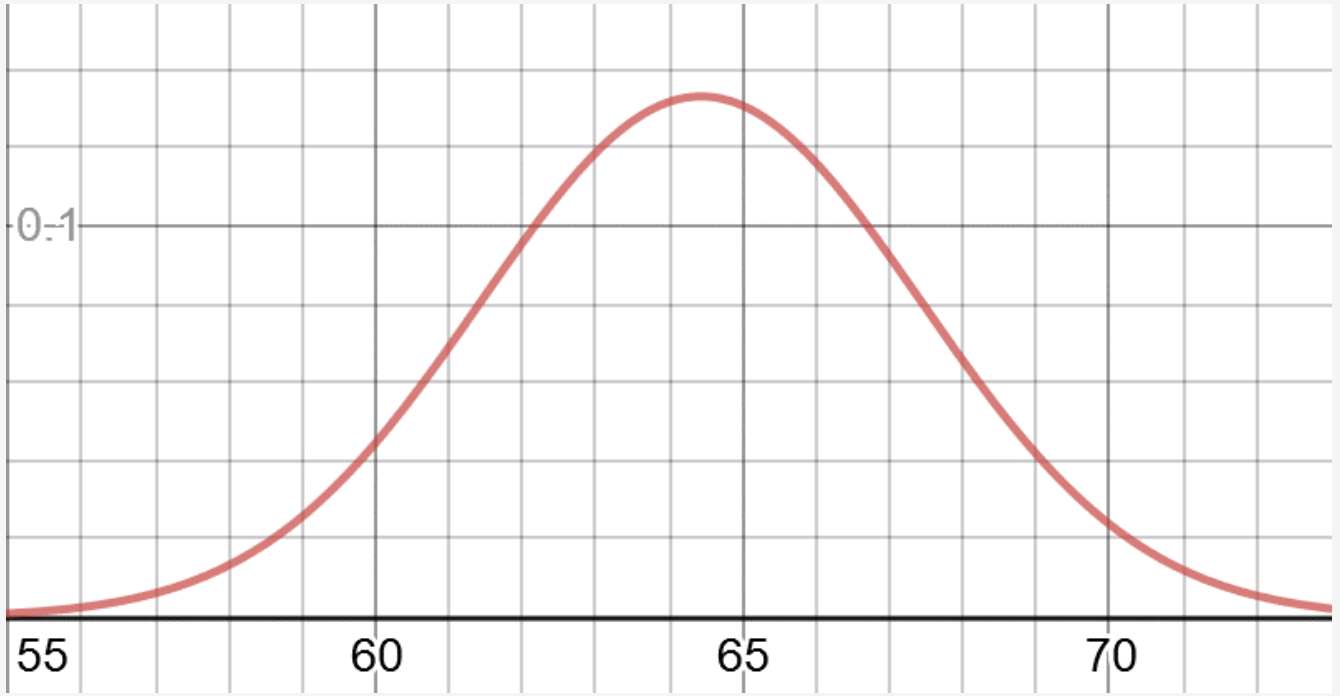
There are many types of distributions, but in this case we can see a nice bell curve known as the **normal distribution**, also called the **Gaussian Distribution**.



Discovering the Normal Distribution

We can fit this curve onto a histogram to make inferences about the entire population, as it's unlikely I will be able to weigh every golden retriever in existence.

There are ways to measure uncertainty whether a bell curve resembles the population's bell curve, but we will steer clear of statistics in this course.



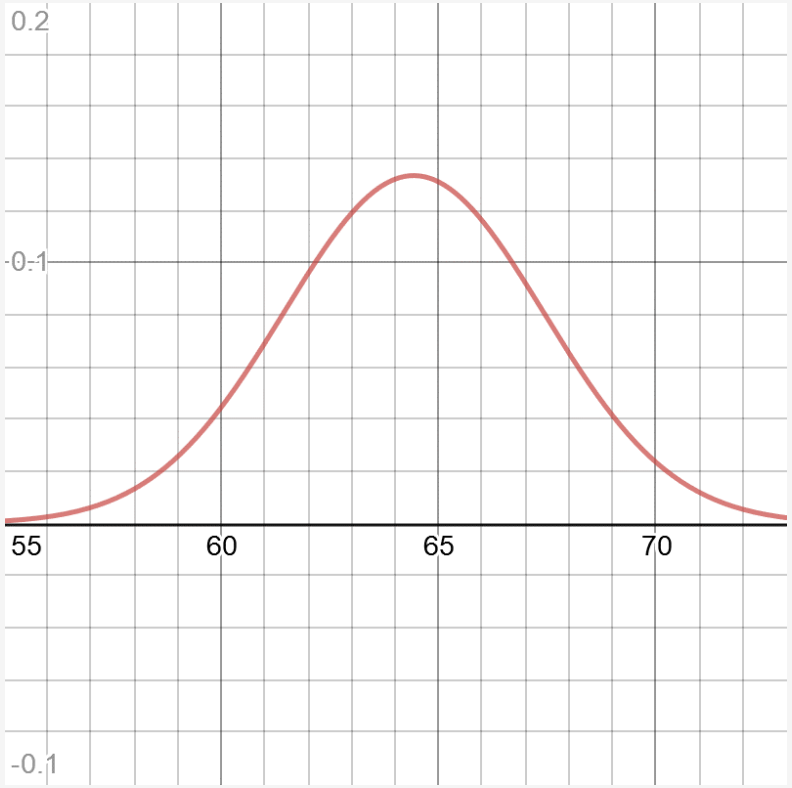
https://www.desmos.com/calculator/yftpag0tse

Discovering the Normal Distribution

The normal distribution has several important properties that make it useful:

- Symmetrical
- Most mass is at the center around the **mean**
- Has a spread (being narrow or wide) that is specified by **standard deviation**.
- The "tails" are the least likely outcomes, and approach zero infinitely but never touch zero.
- It resembles a lot of phenomena in nature and daily life, and even generalizes non-normal problems because of the central limit theorem.

We can expect any golden retriever to have a weight most likely around 64.43 (the mean), but highly unlikely around 55 or 73.



https://www.desmos.com/calculator/yftpag0tse

The Normal Distribution Formula

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

$$\mu = mean$$
$$\sigma = standard\ deviation$$
$$x = observed\ value$$

$$f(x) = probability\ density\ function$$

Python function:

```
# normal distribution, returns likelihood
def normal_pdf(x: float, mean: float, std_dev: float) -> float:
    return (1.0 / (2.0 * math.pi * std_dev ** 2) ** 0.5) * math.exp(-1.0 * ((x - mean) ** 2 / (2.0 * std_dev ** 2)))
```

Discovering the Normal Distribution

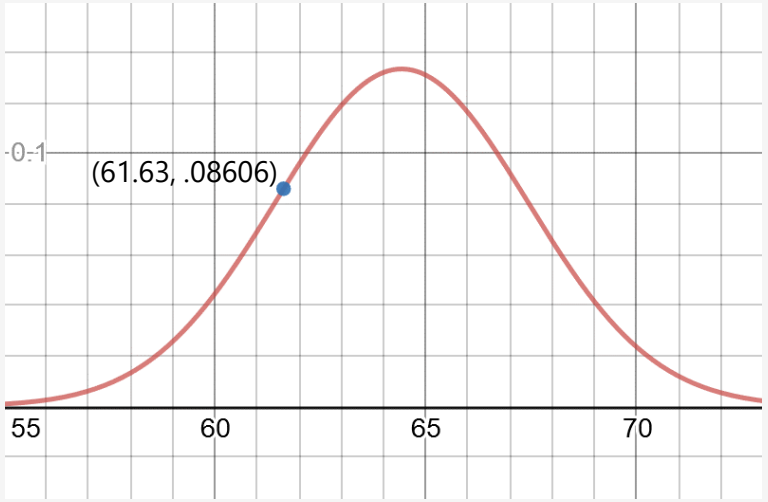
Trick question: What is the probability any golden retriever chosen at random will have a weight of exactly 61.63 lbs?

The answer may surprise you: It is 0%!

This is one of the great paradoxes of continuous distributions like the normal distribution and the beta distribution, as decimals can become so infinitely small there is an infinite number of possibilities on the curve.

The probability of getting a specific value is virtually impossible ***unless*** it is already observed.

The y-axis represents probability density, and to find a probability you need to use the area under a curve for a range of values.



Calculating Probabilities

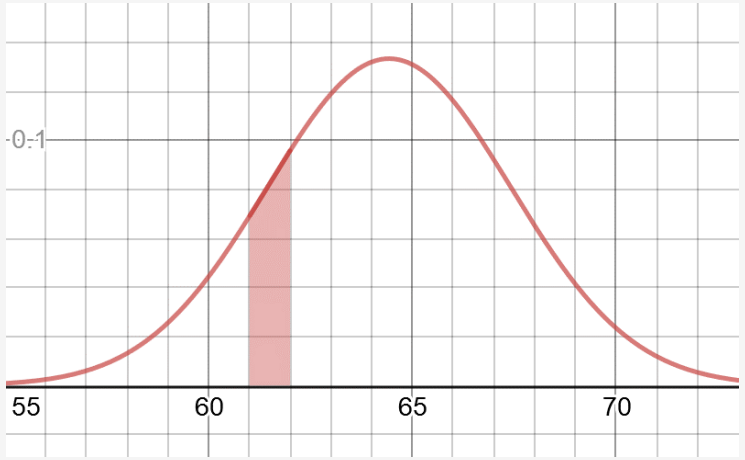
Rather than ask:

“What is the probability a golden retriever will be exactly 61.63 pounds?”

A more productive question would be:

“What is the probability a golden retriever will be ***between*** 61 and 62 pounds?”

The answer is 8.25%, and we will learn how to calculate this using integration just like we did with the beta distribution.

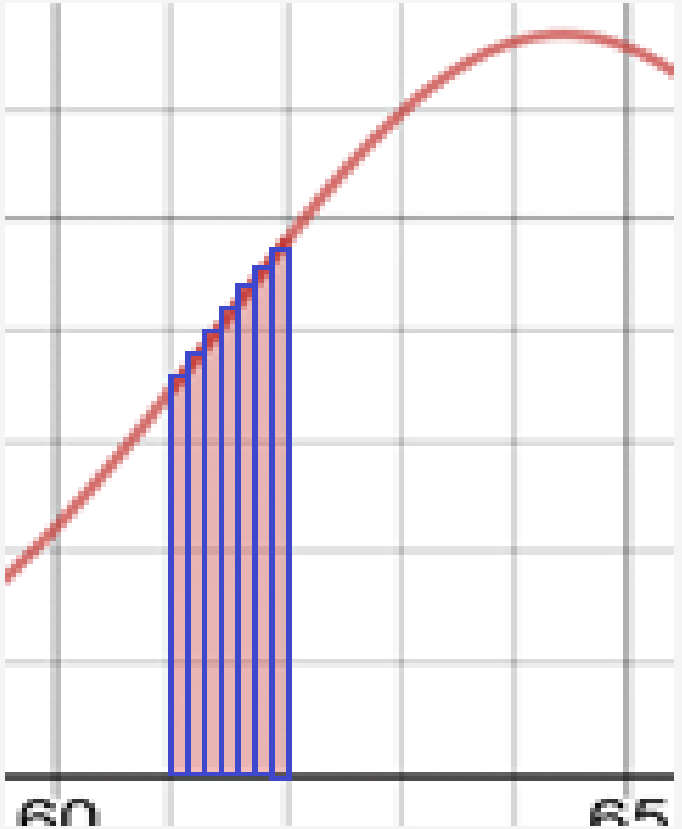


Calculating Probabilities

Just like we did with the beta distribution, we can find the probability in a range by finding the area under the curve.

A good enough way to approximate this area is to pack many equal-width rectangles under the curve, and then sum their areas.

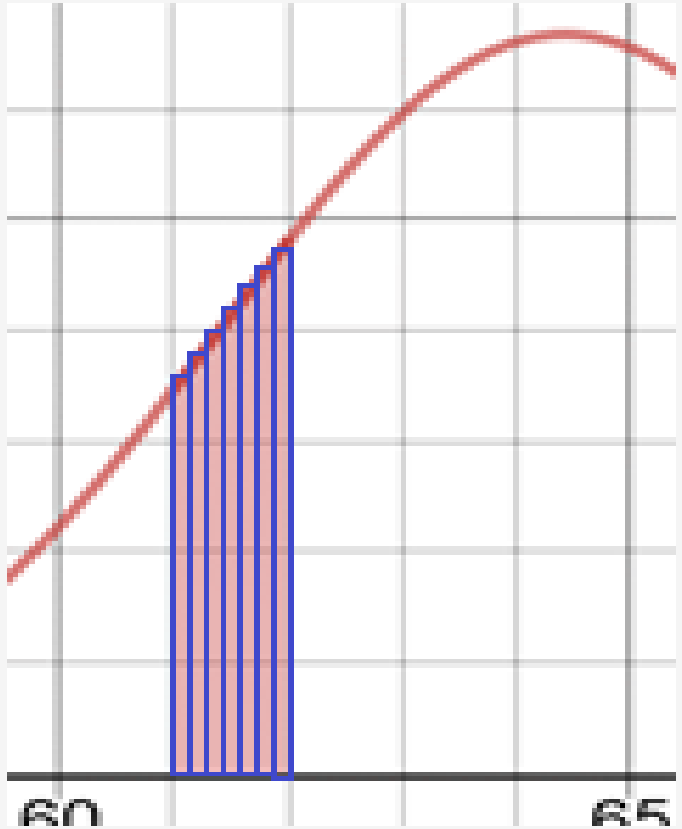
To the right we use 7 rectangles packed into the area of interest, and we can sum the area using this Python integral function :



```python
def approximate_integral(a, b, n, f):
    delta_x = (b - a) / n
    total_sum = 0

    for i in range(1, n + 1):
        midpoint = 0.5 * (2 * a + delta_x * (2 * i - 1))
        total_sum += f(midpoint)

    return total_sum * delta_x
```

Calculating Probabilities

```python
import math


def normal_pdf(x: float, mean: float, std_dev: float) -> float:
    return (1.0 / (2.0 * math.pi * std_dev ** 2) ** 0.5) * math.exp(-1.0 * ((x - mean) ** 2 / (2.0 * std_dev ** 2)))


def approximate_integral(a, b, n, f):
    delta_x = (b - a) / n
    total_sum = 0

    for i in range(1, n + 1):
        midpoint = 0.5 * (2 * a + delta_x * (2 * i - 1))
        total_sum += f(midpoint)

    return total_sum * delta_x


p_between_61_and_62 = approximate_integral(a=61, b=62, n=7, f= lambda x: normal_pdf(x,64.43,2.99))

print(p_between_61_and_62)
```



Running the code above, we get a probability of 8.25% that any given golden retriever's weight is between 61 and 62 pounds.

So whenever you need to calculate the probability of an observed value, it should actually be a range of observed values!

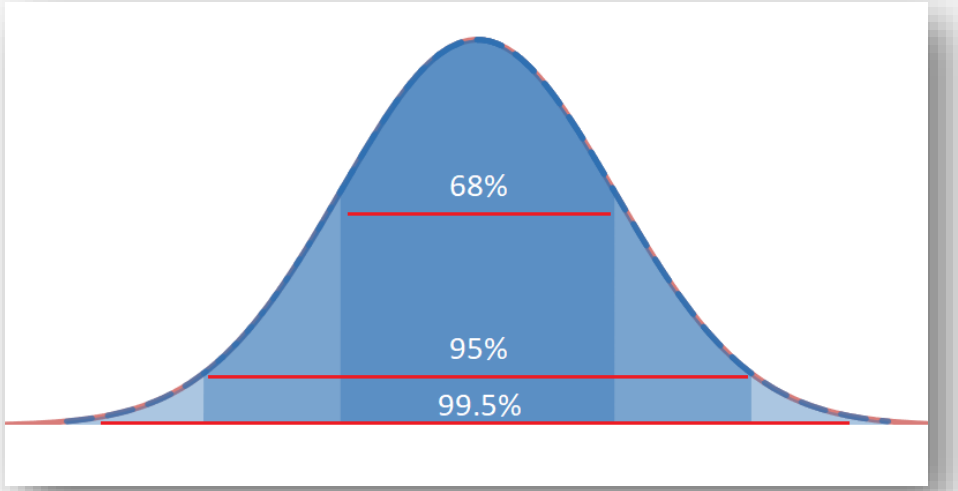
68-95-99.5 Rule

One of the nice features of the normal distribution is the **68-95-99.5** rule, which states the area/probability within 1, 2, and 3 standard deviations from the mean respectively.

This means that if I have a mean of 10 and standard deviation of 3, then…

- 68% of the probability will be between 7 and 13
- 95% will be between 4 and 16
- 99.5% will be between 1 and 19

You can use this rule to quickly calculate probabilities when using standard deviations.



Cumulative Density Function (CDF)

The way we used integration to find the probability between a range is perfectly valid and intuitive.

Conventionally however, the textbook approach is to use a **cumulative density function** when available, which is a function that returns the area up to a value "x".

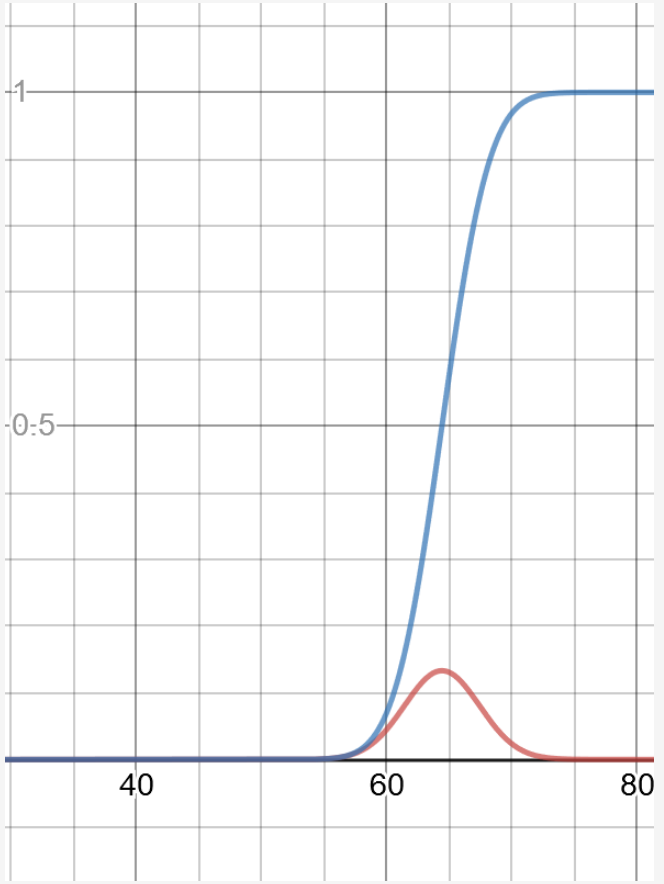
$$\frac{1}{2}\left[1+\operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right]$$

The normal distribution's probability density function (PDF) is shown alongside its cumulative density function (CDF), and the CDF is projecting the area captured up to that x-value on the PDF.

Here is the CDF expressed in Python:

```
def normal_cdf(x:float,mean:float=0.0,std_dev:float=1.0):
    return 0.5*(1+math.erf((x-mean)/((std_dev*2)**0.5)))
```

We do cheat a little here and use the `erf` function, and we will leave it a black box as it is beyond the scope of this class.

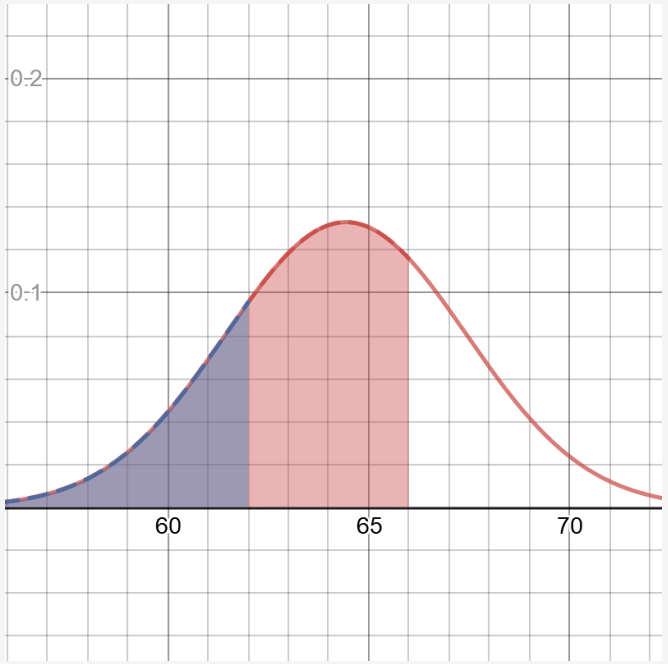


Cumulative Density Function (CDF)

I can use the CDF to find the probability of a golden retriever having a weight between 62 and 66 pounds, without integrating several rectangles and summing their areas.

I first find the area up to 66 (highlighted in red to the right) and then subtract the area up to 62 (highlighted in blue), and we should get 49.2%.

```
mean = 64.43
std_dev = 2.99

prob_between_62_and_66 = normal_cdf(66.0, mean, std_dev) - normal_cdf(62.0, mean, std_dev)

print("PROB BETWEEN 62_and_66: {}".format(prob_between_62_and_66))
```



This can be less work and I do not have to integrate approximations with rectangles, because the CDF function is algebraically integrated.

The rectangle approach is more intuitive, but the CDF function is more effective and precise.

It will also help us with generating random numbers from the normal distribution.

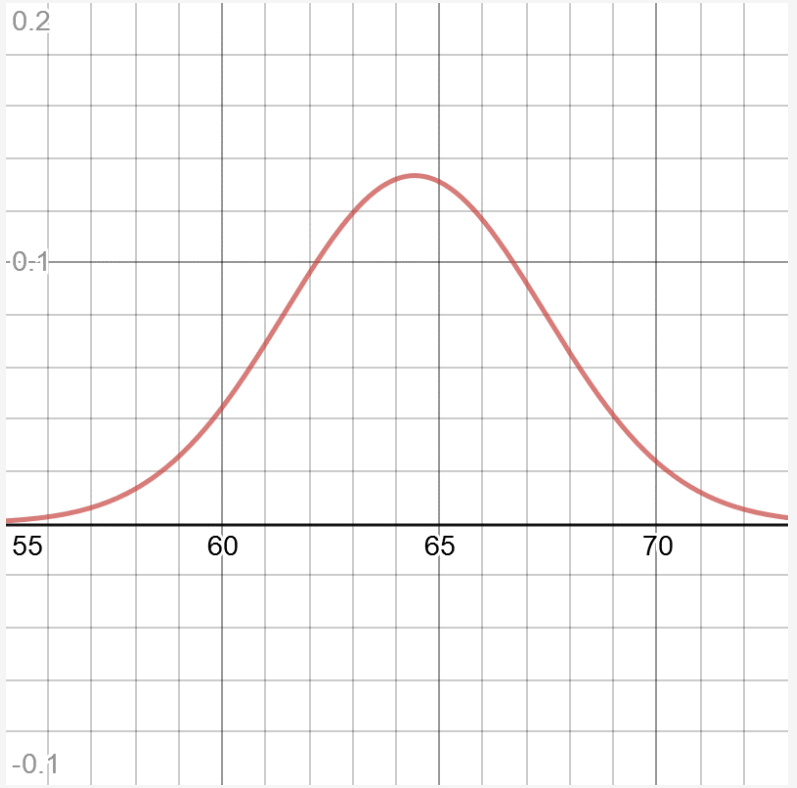
Quantile Function

You are probably familiar with the **uniform distribution**, where any number in a range (e.g. between 0.0 and 1.0) is equally likely and often used in random number generators.

But suppose we wanted to generate random numbers from the normal distribution, or any distribution that is not uniform?

For example, let's say I want a simulation that produced thousands of golden retrievers and I wanted them to have realistic but varied weights.

How do I take this normal distribution (*mean*=64.43, *std dev*=2.99) to the right and generate random numbers from it?

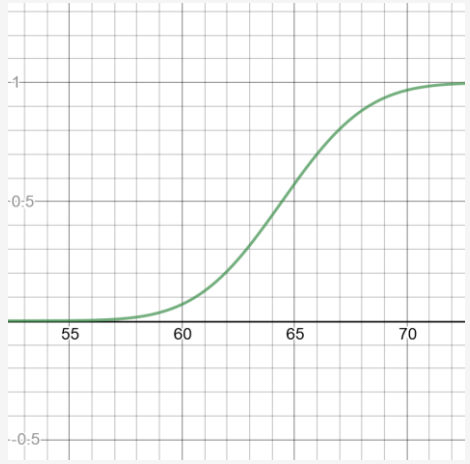


Quantile Function

There are several ways to generate random numbers off a normal distribution, but the most efficient way would be to use a **quantile function** which is the inverse of the CDF function.

With the quantile function, you can input a probability between 0.0 and 1.0 and it will return the respective "x" value.

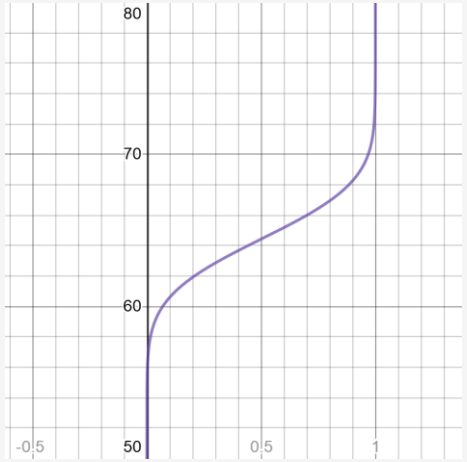
$$\mu + \sigma\sqrt{2}\,\mathrm{erf}^{-1}(2p-1)$$



Normal CDF

Again the expression above is very mathy so here it is in Python:

```python
from scipy.special import erfinv

def inv_normal_cdf(p: float, mean: float, std_dev: float):
    return mean + (std_dev * (2.0 ** 0.5) * erfinv((2.0 * p) - 1.0))
```



Normal Quantile

Think of it as an inverted lookup, where we look up the likelihood value on the y-axis, and then return the corresponding value from the x-axis.

We cheat a little bit here and use SciPy, as the `erfinv` function is a rabbit hole of a topic.

Generating Random Numbers from a Normal Distribution

If we generate a random probability between 0.0 and 1.0 from a uniform distribution (where any number is equally likely), we can pass it to the normal distribution's quantile function to return random "x" values!

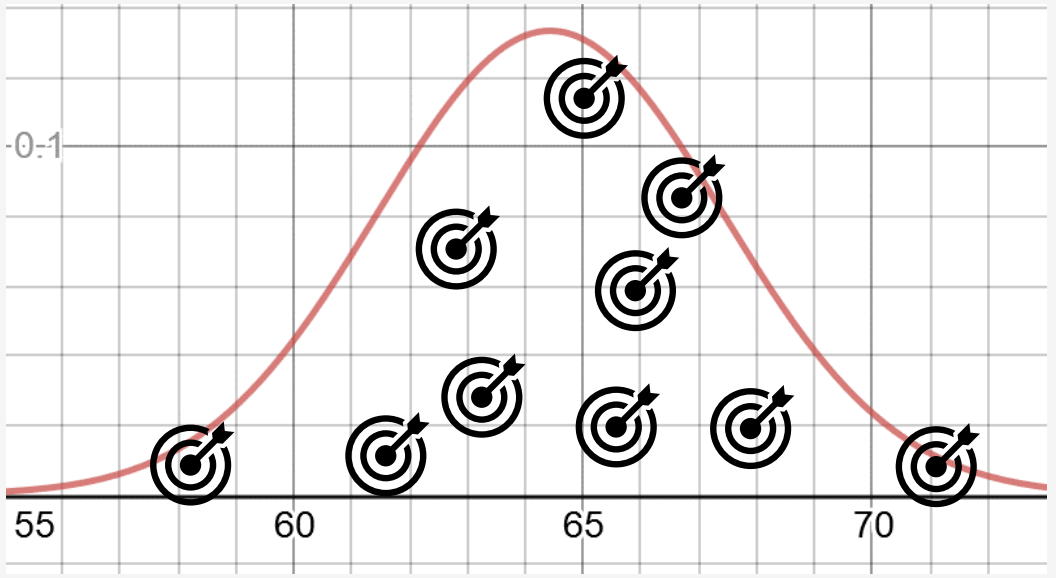
You will find more values will be around the mean, and less will be at the tails.

The code to the right should generate 10 random values for our golden retriever normal distribution (with mean of 64.43 and std dev of 2.99).

Weird, right? Think of the curve as a dartboard.

We are more likely to hit darts at the center of curve and less likely at the tails, and this is what the quantile function allows us to do.

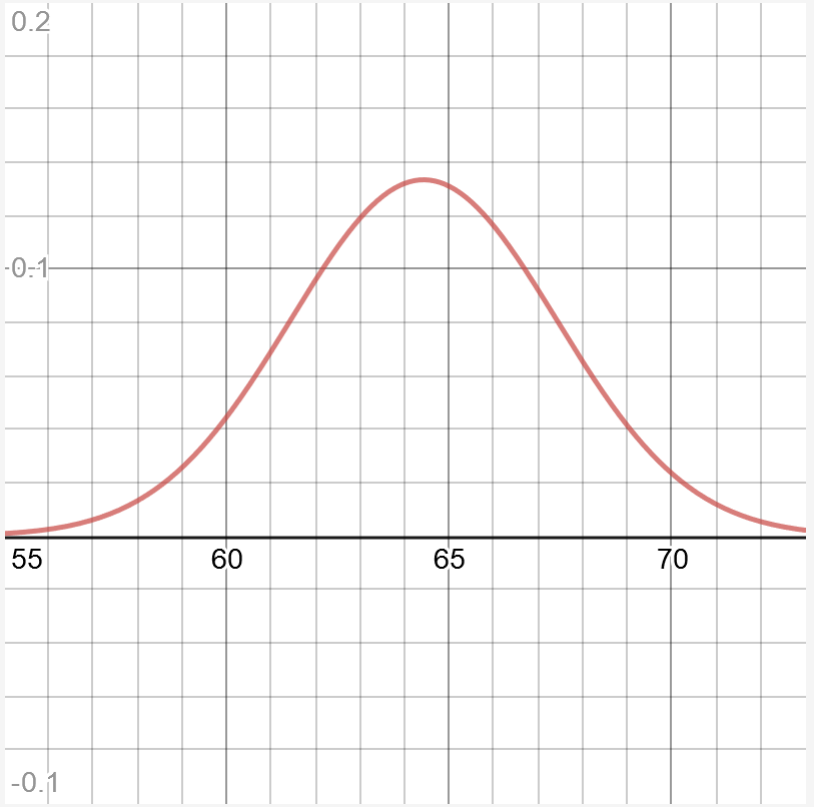
```python
import random
from scipy.special import erfinv


def inv_normal_cdf(p: float, mean: float, std_dev: float):
    return mean + (std_dev * (2.0 ** 0.5) * erfinv((2.0 * p) - 1.0))


mean = 64.43
std_dev = 2.99

for i in range(0, 10):
    random_p = random.uniform(0.0, 1.0)
    print(inv_normal_cdf(random_p, mean, std_dev))
```



Generating Random Numbers from a Normal Distribution

Still confused? Let's break it down.

1) We generate a random number between 0.0 and 1.0, and say we get .74

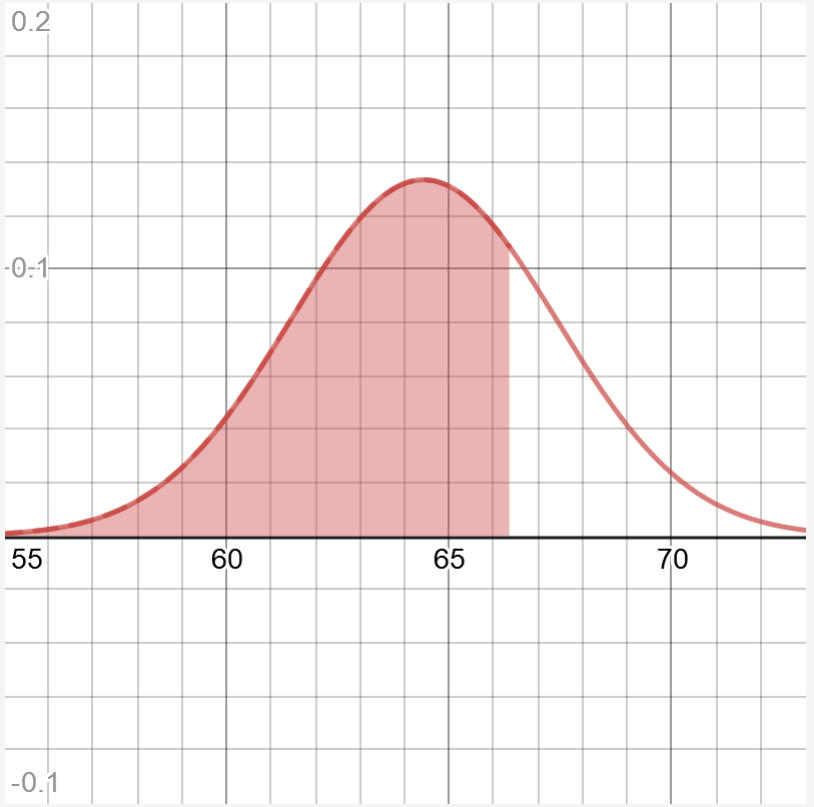


Generating Random Numbers from a Normal Distribution

Still confused? Let's break it down.

1) We generate a random number between 0.0 and 1.0, and say we get .74

2) We use the quantile function to look up the x-value where the area is *exactly* .74 (shaded in red to the right), and this x-value is approximately 66.3536.



Generating Random Numbers from a Normal Distribution

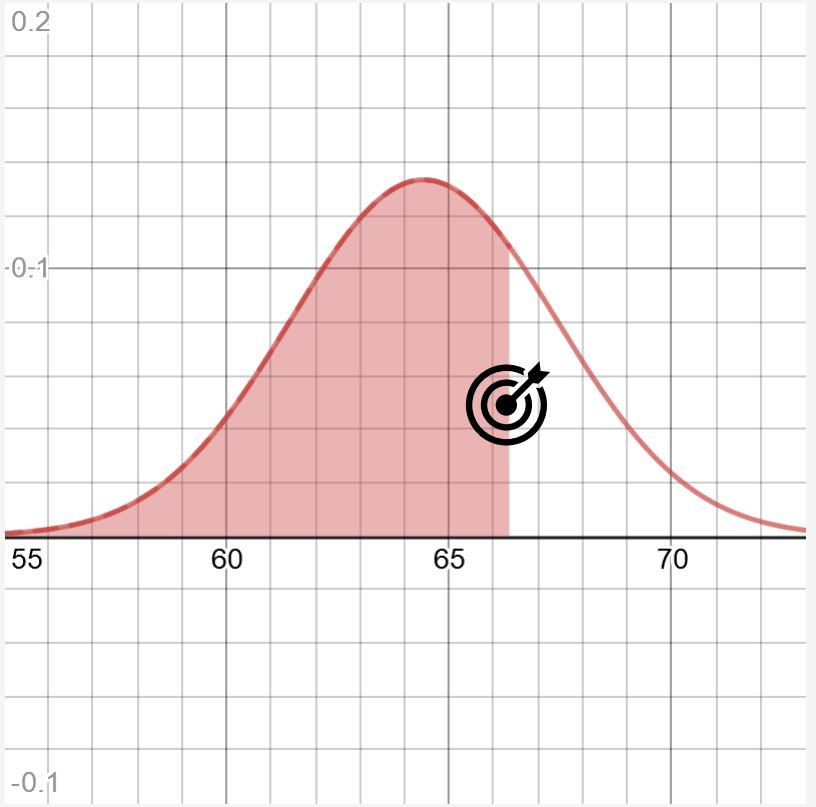
Still confused? Let's break it down.

1) We generate a random number between 0.0 and 1.0, and say we get .74

2) We use the quantile function to look up the x-value where the area is *exactly* .74 (shaded in red to the right), and this x-value is approximately 66.3536.

3) We now have a randomly-generated number 66.3536 that fits our normal distribution.

Again, think of it as throwing a "dart" at the curve and we are more likely to get values where density is highest at the center and less on the tails



Generating Random Numbers from a Normal Distribution

Still confused? Let's break it down.

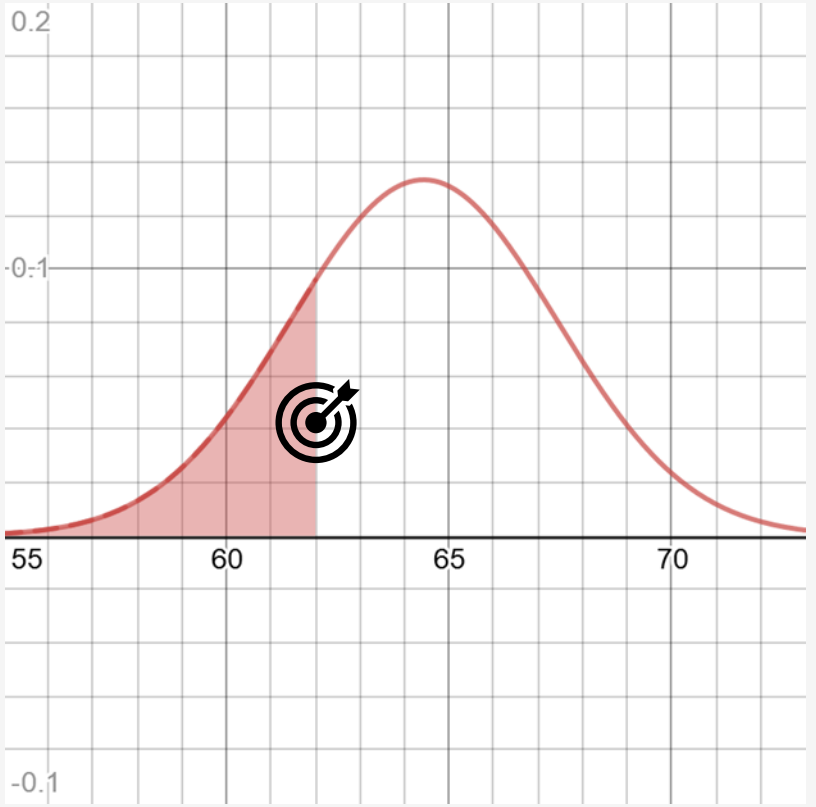
1) We generate a random number between 0.0 and 1.0, and say we get .74

2) We use the quantile function to look up the x-value where the area is *exactly* .74 (shaded in red to the right), and this x-value is approximately 66.3536

3) We now have a randomly-generated number 66.3536 that fits our normal distribution.

Again, think of it as throwing a "dart" at the curve and we are more likely to get values where density is highest at the center and less on the tails

Repeat the above until you have all the random values you need!



Generating Random Numbers from a Normal Distribution

Still confused? Let's break it down.

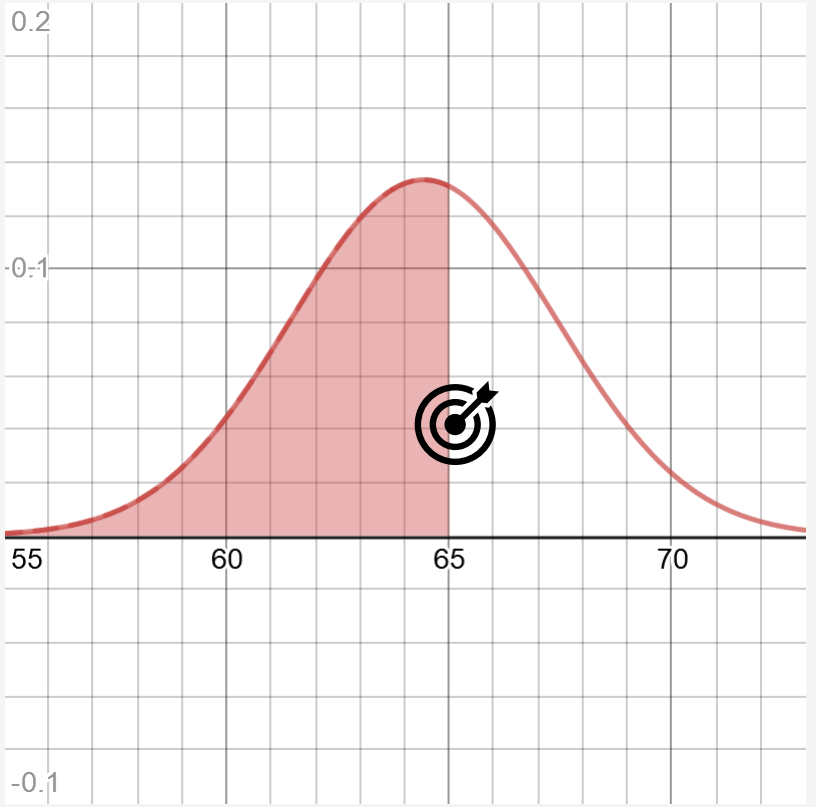
1) We generate a random number between 0.0 and 1.0, and say we get .74

2) We use the quantile function to look up the x-value where the area is *exactly* .74 (shaded in red to the right), and this x-value is approximately 66.3536

3) We now have a randomly-generated number 66.3536 that fits our normal distribution.

Again, think of it as throwing a "dart" at the curve and we are more likely to get values where density is highest at the center and less on the tails

Repeat the above until you have all the random values you need!



Generating Random Numbers from a Normal Distribution

Still confused? Let's break it down.

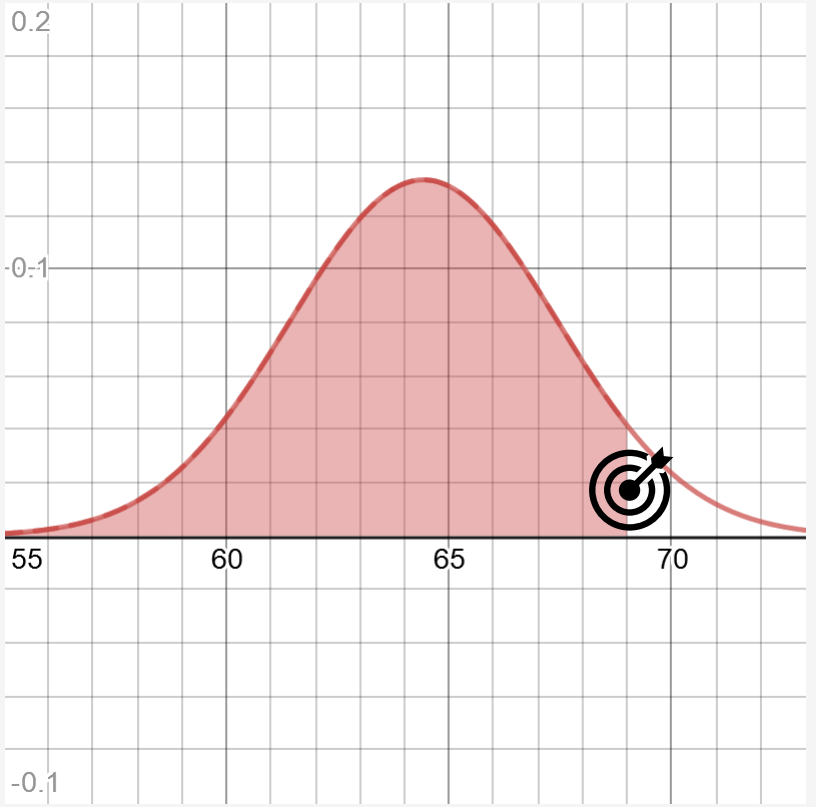
1) We generate a random number between 0.0 and 1.0, and say we get .74

2) We use the quantile function to look up the x-value where the area is *exactly* .74 (shaded in red to the right), and this x-value is approximately 66.3536

3) We now have a randomly-generated number 66.3536 that fits our normal distribution.

Again, think of it as throwing a "dart" at the curve and we are more likely to get values where density is highest at the center and less on the tails

Repeat the above until you have all the random values you need!



Generating Random Numbers from a Normal Distribution

Question: why are we less likely to get values on the tails?

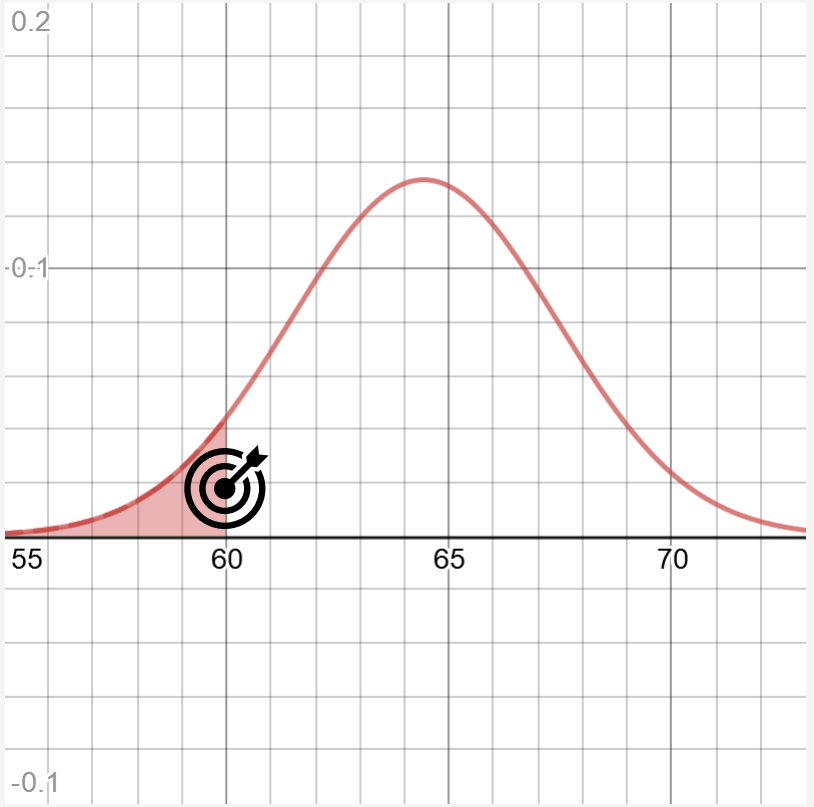
Is not a random value between 0.0 and 1.0 coming from a uniform distribution? Where any value is equally likely?

Think of it this way: the tails have a much smaller density/probability meaning they have a smaller range to “hit”.

We are less likely get a value between 0.0 and 0.05 as opposed to a value between 0.0 and 0.50.

We are also less likely to get a value between .99 and 1.0.

The smaller the range in the probability, the less likely we are to hit it.

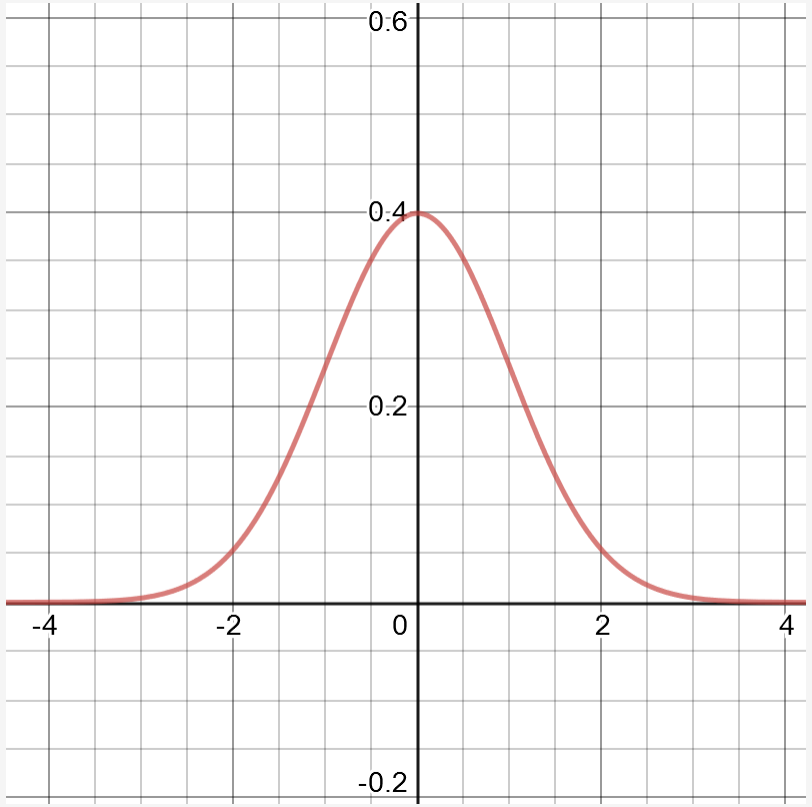


Standard Normal Distribution

A special type of normal distribution is the **standard normal distribution**, which has a mean of 0 and standard deviation of 1.

Sometimes a normal distribution will be converted into a standard normal distribution.

- This creates a convenient way to express all values in terms of the standard deviation, known as **z scores**.
- Turning several normal distributions into standard normal distributions also makes them easier to compare, as comparisons can be made relative to their means rather than absolute values.



Central Limit Theorem

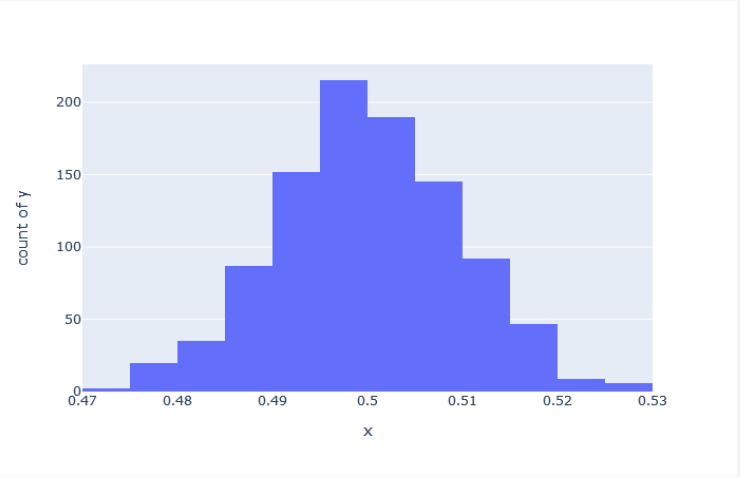
The normal distribution shows up in an important phenomena called the **central limit theorem**, which states that the means of different samples will form a normal distribution no matter what the underlying distribution is.

Say I generated 1000 samples, each containing 1000 random numbers coming from a uniform distribution between 0.0 and 1.0.

If I take the average of each sample and plot them in a histogram, we will see they form a normal distribution!

Even if the data is completely random or has a distribution different from a normal distribution, the averages of the samples will form a normal distribution.

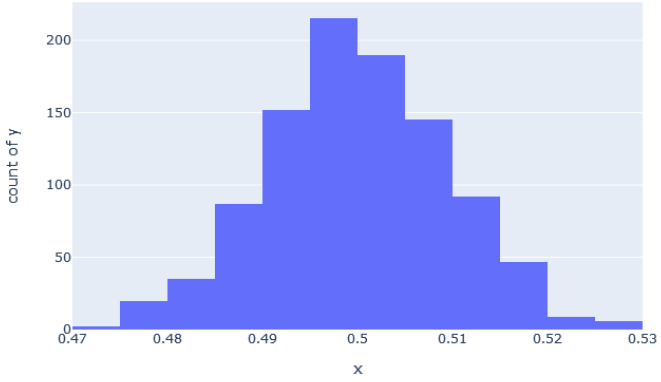
This is another reason why the normal distribution is useful, as it shows up even in data that is not normally distributed.



The above data came from a uniform distribution, and yet the mean of each sample of 1000 data points shows a normal distribution!

Central Limit Theorem

```
# Samples of the uniform distribution will average out to a normal distribution.

import random

import plotly.express as px

# Central limit theorem, 1000 samples each with 1000 random numbers between 0.0 and 1.0
x_values = [(sum([random.uniform(0.0, 1.0) for i in range(1000)]) / 1000.0) for _ in range(1000)]
y_values = [1 for _ in range(1000)]

px.histogram(x=x_values, y = y_values, nbins=20).show()
```



The above data came from a uniform distribution, and yet the mean of each sample of 1000 data points shows a normal distribution!

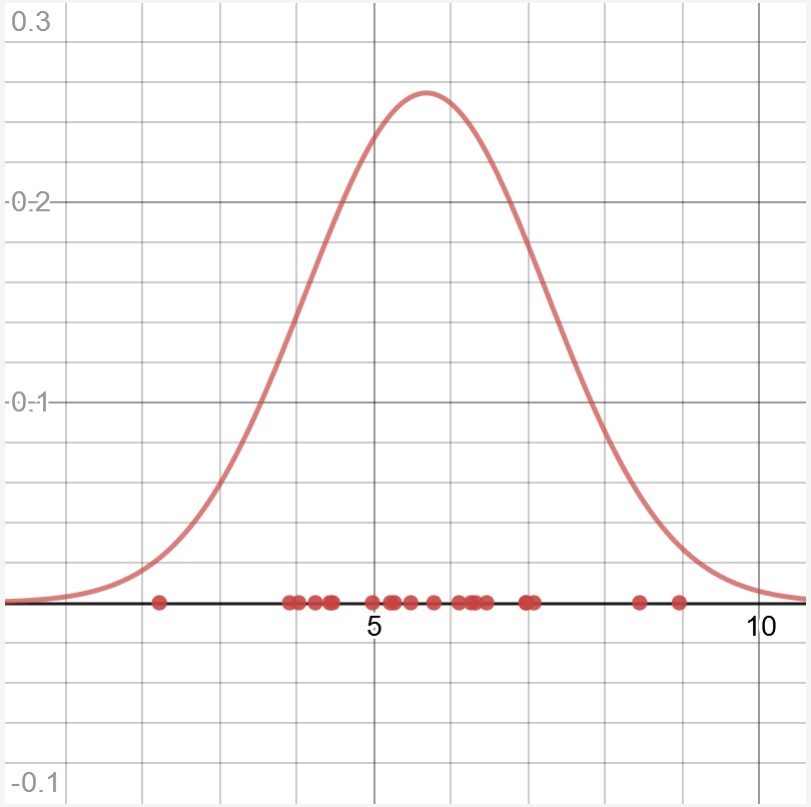
Fitting a Probability Distribution

If we feel our data can be appropriately fit to a normal distribution, we could use the means and standard deviation summaries of the data to fit a normal distribution.

However this does not work for every situation, and the maximum likelihood estimation algorithm is another way to fit a model to the data.

- MLE works by "randomly" adjusting parameters (in this case the *mean* and *standard deviation*) repeatedly until it no longer "improves" the model.
- The "improvement" we are trying to make is maximizing the joint probability of observing all the points for a given set of mean/standard deviation parameters.
- We can make "random" adjustments to the parameters by sampling from a standard normal distribution, where smaller values around 0 are most common but occasional large values are found in the tail.

It also is used for logistic regression and other probability-based models, so it is worth learning!



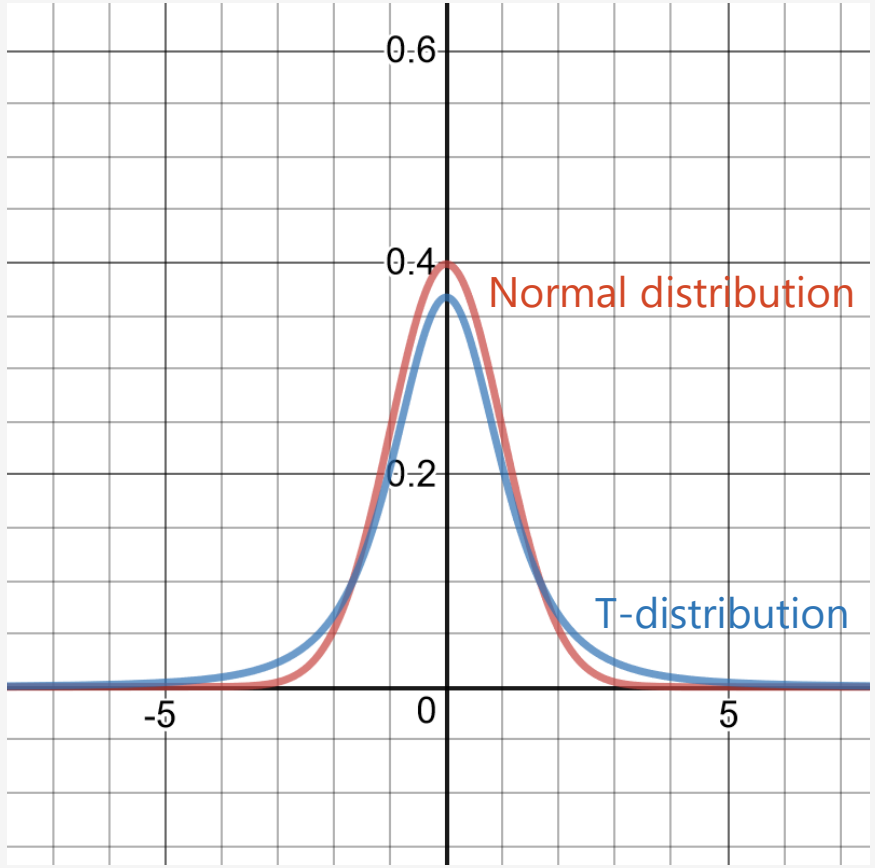
Solving for a Fit

If we are trying to maximize the joint probability of all the points being observed, how do we find the **mean** and **standard deviation** values to maximize it?

We randomly increase/decrease the **mean** and **standard deviation** with random values from a standard normal distribution, or even better a T-Distribution which has fatter tails.

Fatter tails = more smaller/larger values = more diverse moves.

We can use these random adjustments to the **mean** and **standard deviation** as our moves in **hill climbing**.



https://www.desmos.com/calculator/xm56tvvalh

Fitting a Probability Distribution

Again, what we are doing with the maximum likelihood estimation is finding the curve that will maximize the probability of seeing these points.

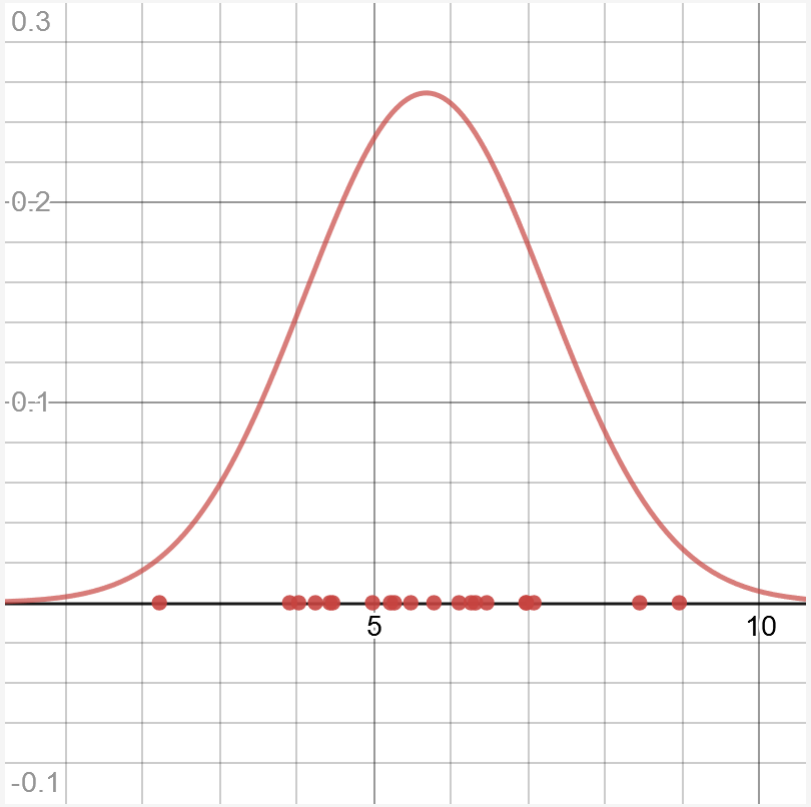
You can fit a normal distribution to some observations by performing the following steps:

1) Start the mean and standard deviation at an initial value, as well as a joint probability of 0.0.

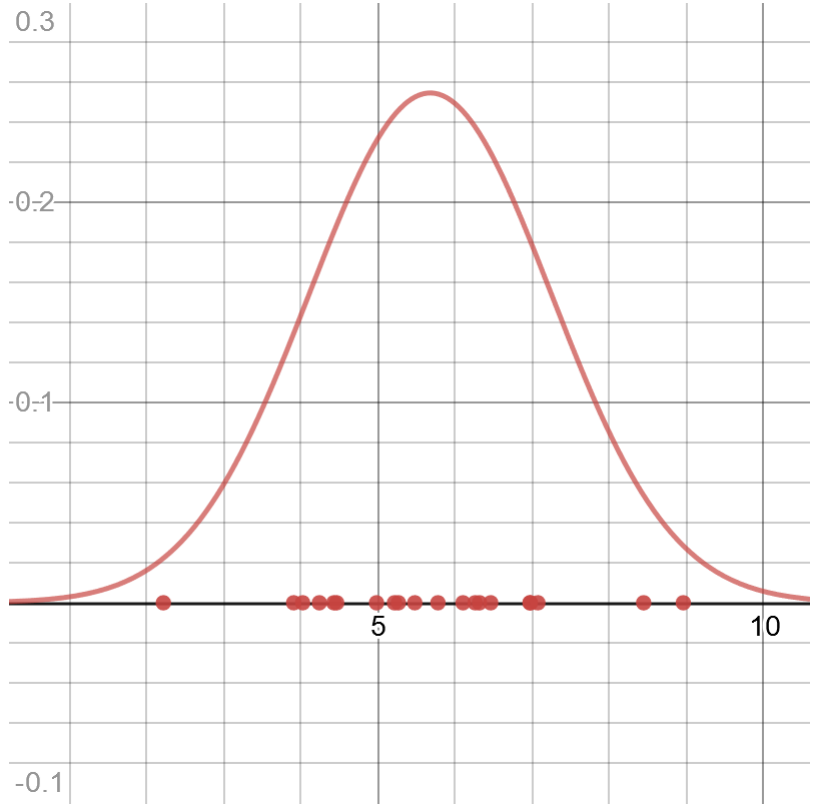
2) Select either the mean or standard deviation randomly, and use a standard normal distribution to generate a random amount to adjust it.

3) Calculate the joint probability across all the points, and if that increases the joint probability, keep it! Otherwise revert the adjustment back.

4) Repeat the steps above for thousands of iterations or until it cannot improve anymore.



```python
import math
import random
from scipy.special import erfinv


# Graph: https://www.desmos.com/calculator/82ww7obt3c

# normal distribution, returns likelihood
def normal_pdf(x: float, mean: float, std_dev: float) -> float:
    return (1.0 / (2.0 * math.pi * std_dev ** 2) ** 0.5) * math.exp(-1.0 * ((x - mean) ** 2 / (2.0 * std_dev ** 2)))

# quantile function for normal distribution
def inv_normal_cdf(p: float, mean: float, std_dev: float):
    return mean + (std_dev * (2.0 ** 0.5) * erfinv((2.0 * p) - 1.0))


# input data
from urllib.request import urlopen

observations = [5.47877451179531, 4.24197544484815, 6.10392629484813, 6.97602731373887, 6.31728101561784,
                2.21980728359946, 7.07446151527471, 4.02577075431478, 8.95948027942515, 5.21534108058981,
                8.44542048558608, 4.98281649234432, 6.26026913091136, 4.46656269404734, 6.46309961771669,
                5.77865756989296, 4.42979304141431, 5.26677750975235, 6.97225406575382, 3.90849578983089]


best_likelihood = 0.0
std_dev = 1.0
mean = sum(observations) / len(observations)

# randomly adjust mean and standard deviation using hill climbing
for i in range(100_000):

    # Select a random probability between 0 and 1
    random_p = random.uniform(0, 1)

    # Get a random adjustment amount based on standard normal distribution
    adjust = inv_normal_cdf(random_p, 0, 1)

    # Select the mean or average randomly and adjust it
    selected_variable = random.randint(0, 1)

    if selected_variable == 0:
        mean += adjust
    elif selected_variable == 1:
        std_dev += adjust

    # calculate the new likelihood
    likelihood = 0
    for x in observations:
        likelihood += math.log(.000000001 + normal_pdf(x, mean, std_dev))

    likelihood = math.exp(sum([math.log(.000000001 + normal_pdf(x, mean, std_dev)) for x in observations]))

    # if likelihood improves, keep it
    if likelihood > best_likelihood:
        best_likelihood = likelihood
    elif selected_variable == 0:
        mean -= adjust
    elif selected_variable == 1:
        std_dev -= adjust

print("mean={0}, std_dev={1}".format(mean, std_dev))
```



Other Distribution Types

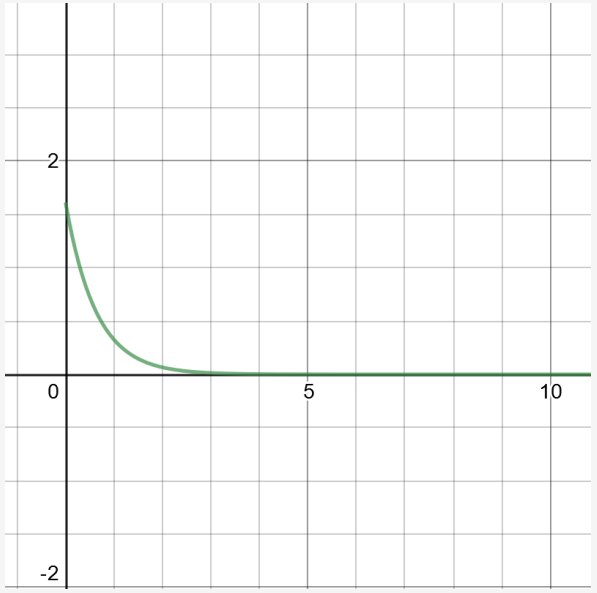
There are many types of probability distributions, some much more niche than others, but here are a few other notable ones:

Poisson – Discrete distribution of probabilities for **x** number of events in a fixed amount of time.

Exponential – Distribution of time between events in a Poisson process

Gamma – A generalization of a two-parameter distribution that includes exponential, Erlang, and chi-squared distributions

Weibull – Often used for survival analysis, failure analysis, and reliability engineering.

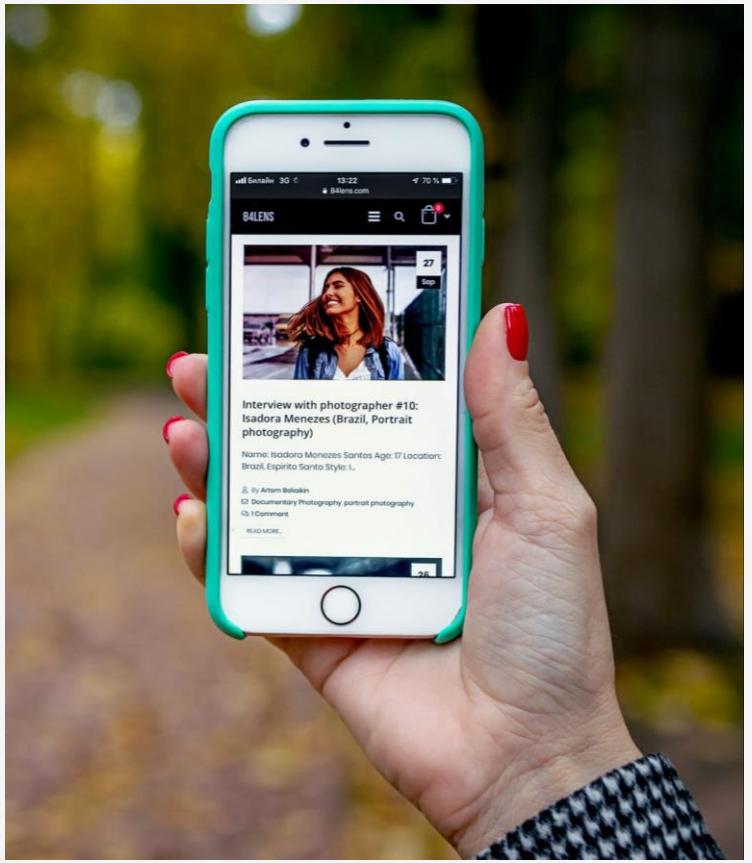


Exponential Distribution

Exercise 4A

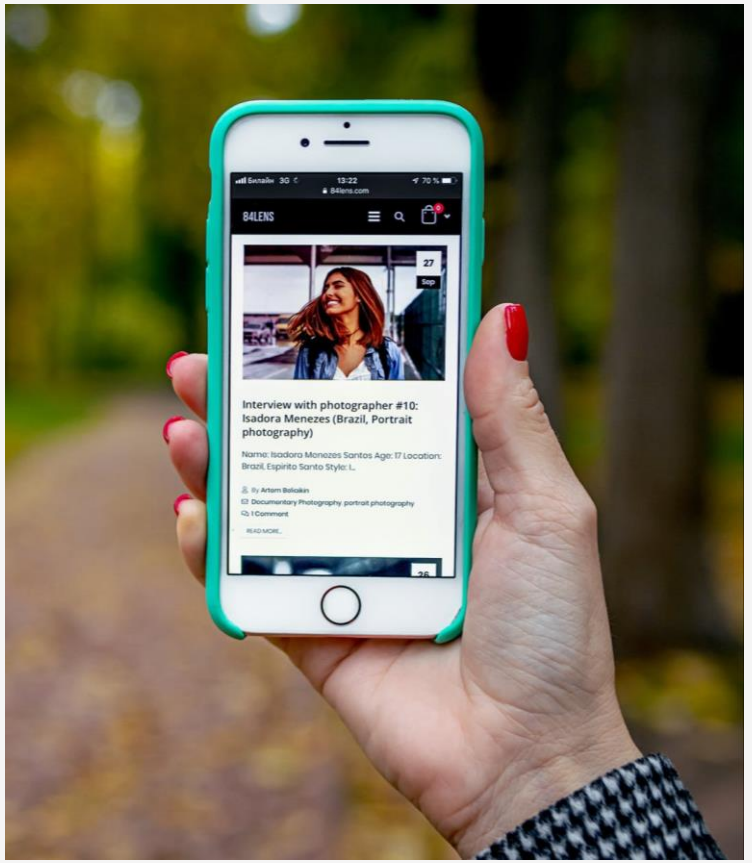
A market researcher estimates that the Z-Phone smart phone has a mean consumer life of 42 months with a standard deviation of 8 months.

Assuming a normal distribution, what is the probability a given random Z-Phone will last between 20 and 30 months?



Exercise 4A

A market researcher estimates that the Z-Phone smart phone has a mean consumer life of 42 months with a standard deviation of 8 months.



Assuming a normal distribution, what is the probability a given random Z-Phone will last between 20 and 30 months?

```
from normal_distribution import  normal_cdf

prob_between_20_and_30 = normal_cdf(30.0, 42.0, 8.0) - normal_cdf(20.0, 42.0, 8.0)

print("PROB BETWEEN 20 and 30: {}".format(prob_between_20_and_30))
```

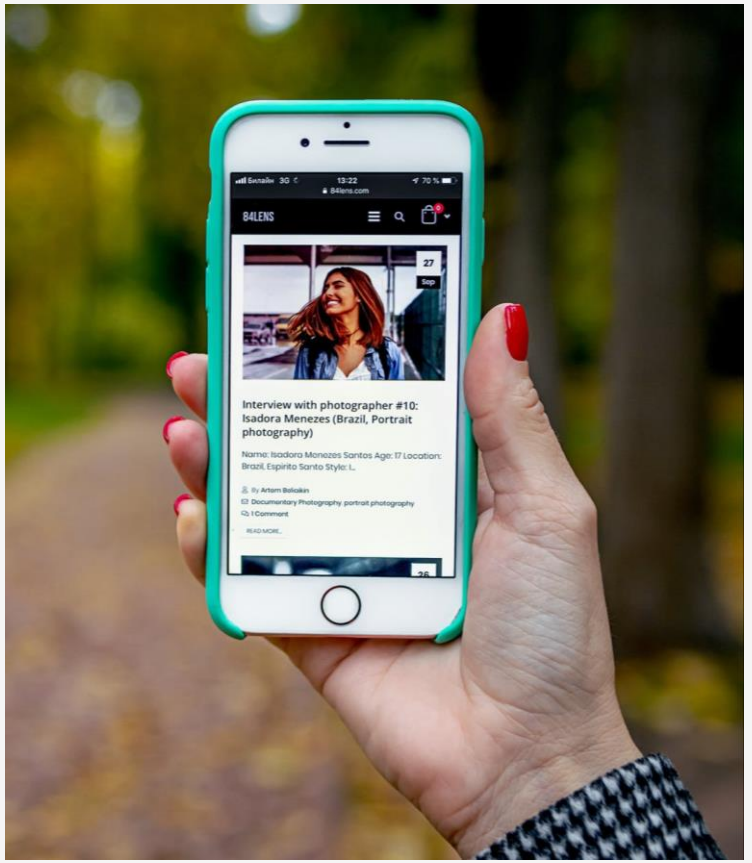
```
from normal_distribution import approximate_integral, normal_pdf

def phone_life_distribution(x):
    mean = 42.0
    std_dev = 8.0
    return normal_pdf(x, mean, std_dev)

prob_between_20_and_30 = approximate_integral(a=20.0, b=30.0, n=1000, f=phone_life_distribution)

print("PROB BETWEEN 20 and 30: {}".format(prob_between_20_and_30))
```

Both solutions above are valid. There is a 6.3827% probability the Z-Phone will last between 20 and 30 months.

Exercise 4B

A market researcher estimates that the Z-Phone smart phone has a mean consumer life of 42 months with a standard deviation of 8 months.

Assuming a normal distribution, what is the probability a phone will last exactly 32.25 months?

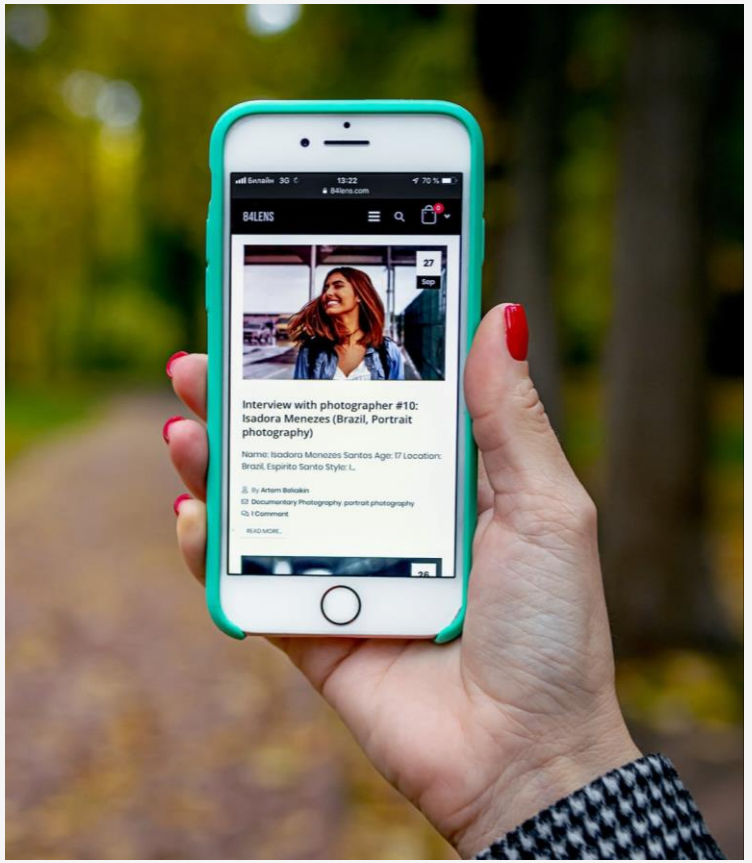


Exercise 4B

A market researcher estimates that the Z-Phone smart phone has a mean consumer life of 42 months with a standard deviation of 8 months.

Assuming a normal distribution, what is the probability a phone will last exactly 32.25 months?

Trick question! The probability of a single value in a continuous model of infinite values has a probability of 0%. The question should ask for a range (e.g. “between 32 and 33 months”) rather than seek the probability for a single value to happen.



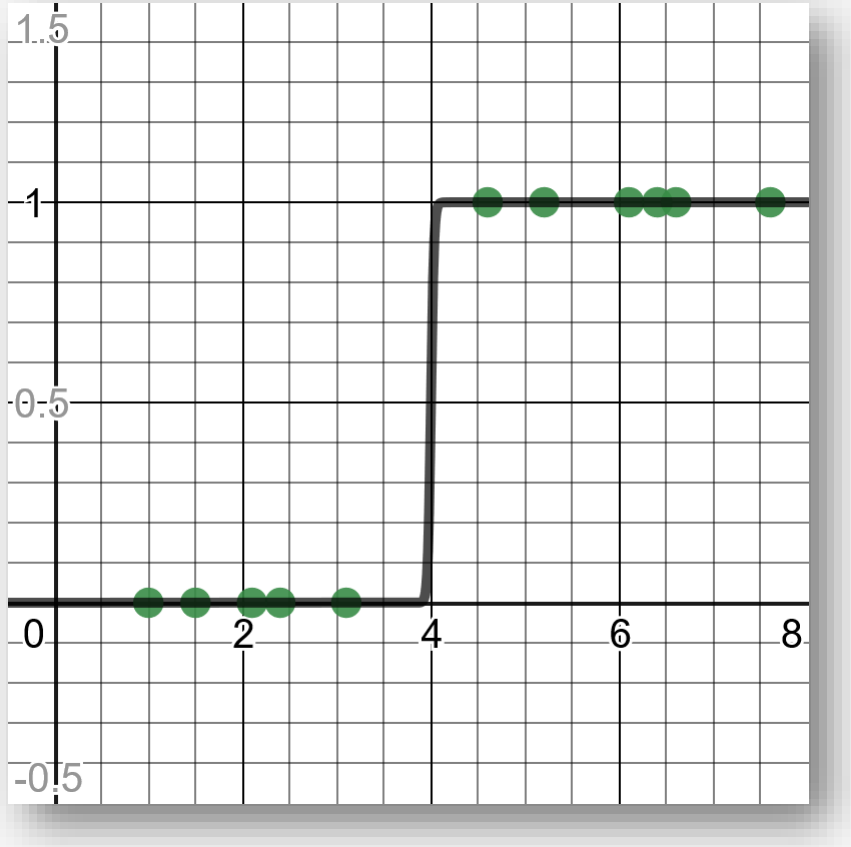
Section VI

The Logistic Function and Logistic Regression

Logistic Regression Intuition

Imagine you have 11 patients exposed to a chemical for x hours, and you plot whether they exhibited symptoms (1) or not (0).

Plotting our patient data (right), we can easily eyeball a clear cutoff at 4 hours where patients transition from ***not showing symptoms (0)*** to ***showing symptoms (1)***.



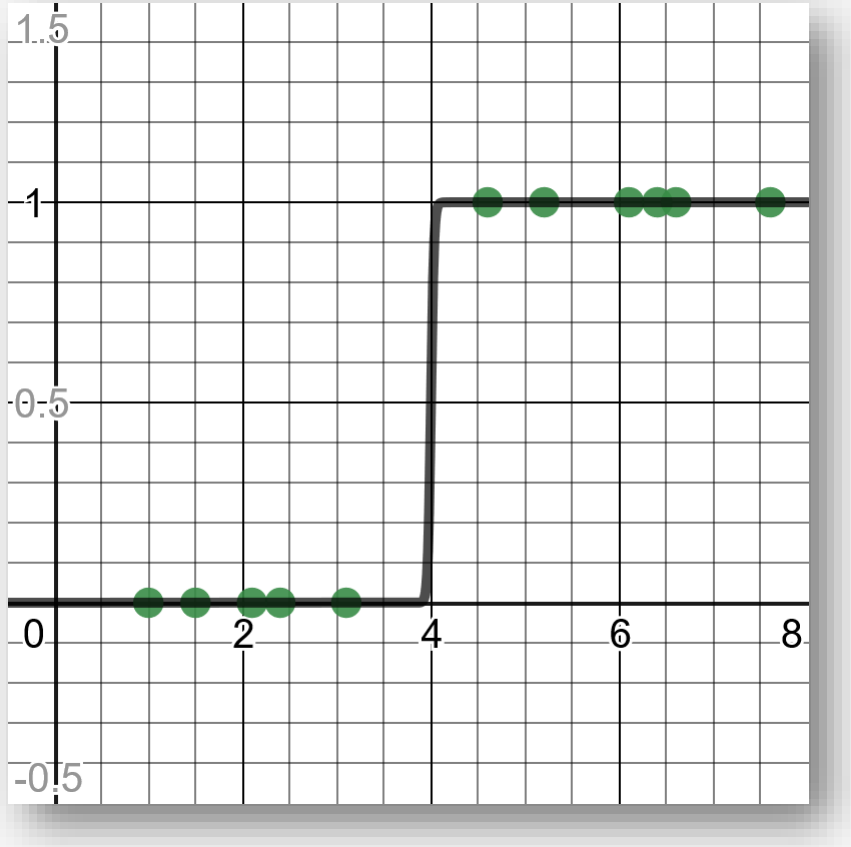
https://www.desmos.com/calculator/prs2p0sofc

Logistic Regression Intuition

This indicates any patient exposed for less than 4 hours will have a 0% chance of showing symptoms, but greater than 4 will have a 100% chance of showing symptoms.

Because there is a distinct separation at 4 hours, a logistic regression is going to “jump” from 0% to 100% at that boundary.

Of course, real life rarely works out this way…



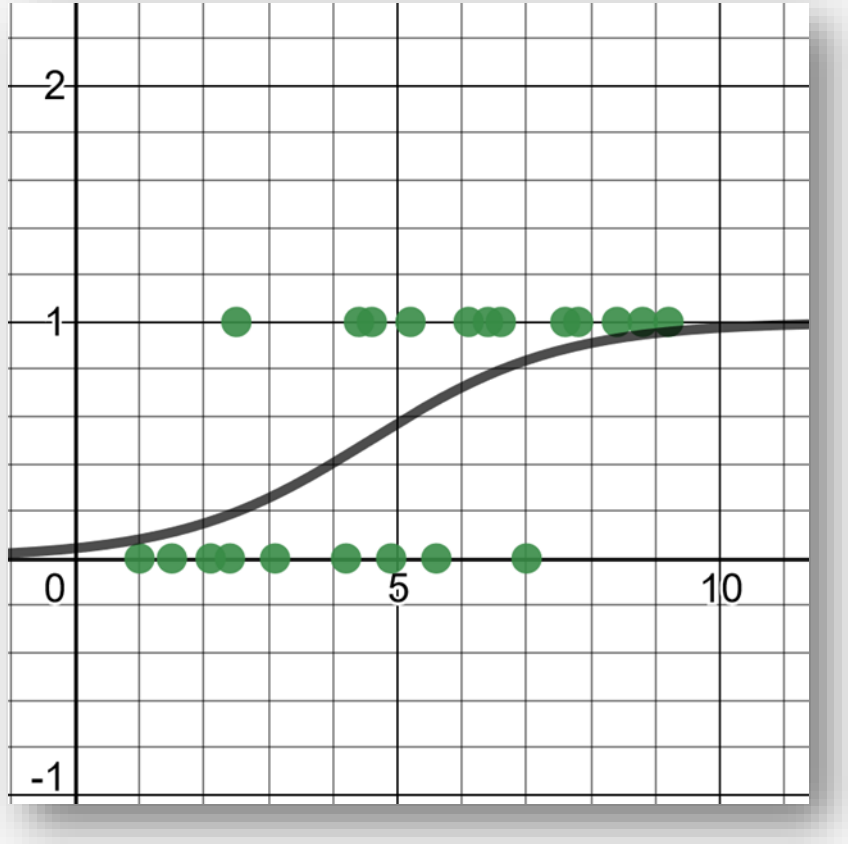
https://www.desmos.com/calculator/prs2p0sofc

Logistic Regression Intuition

Now let's say you gathered more data and got a realistic picture, where the middle of the range has a mix of patients showing symptoms and not showing symptoms.

The way to interpret this is the probability of patients showing symptoms gradually increases with each hour of exposure.

Because of this overlap of points in the middle, there is no distinct cutoff when patients show symptoms, but rather a gradual transition from 0% probability to 100% probability ("0" and "1").

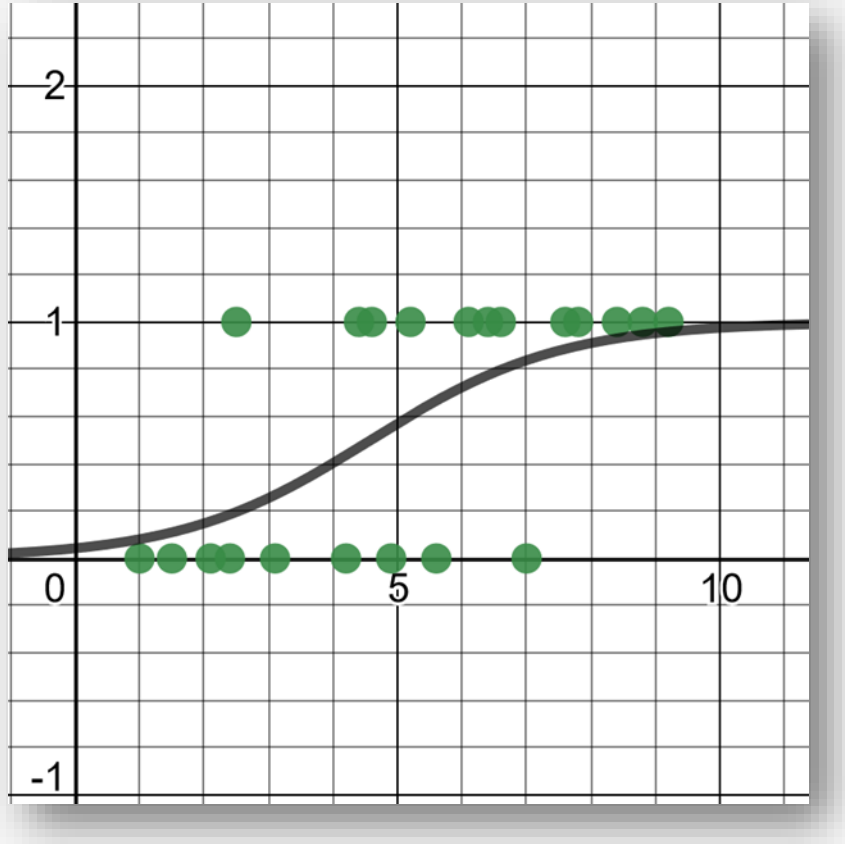


https://www.desmos.com/calculator/bsqtqfians

Logistic Regression Intuition

More technically, a logistic regression results in a curve indicating a probability of belonging to the **true** (1) category, which in this case means ***a patient showed symptoms***.

As the hours of chemical exposure increases, the number of patients showing symptoms increases, and thus the probability of showing symptoms increases.



https://www.desmos.com/calculator/bsqtqfians

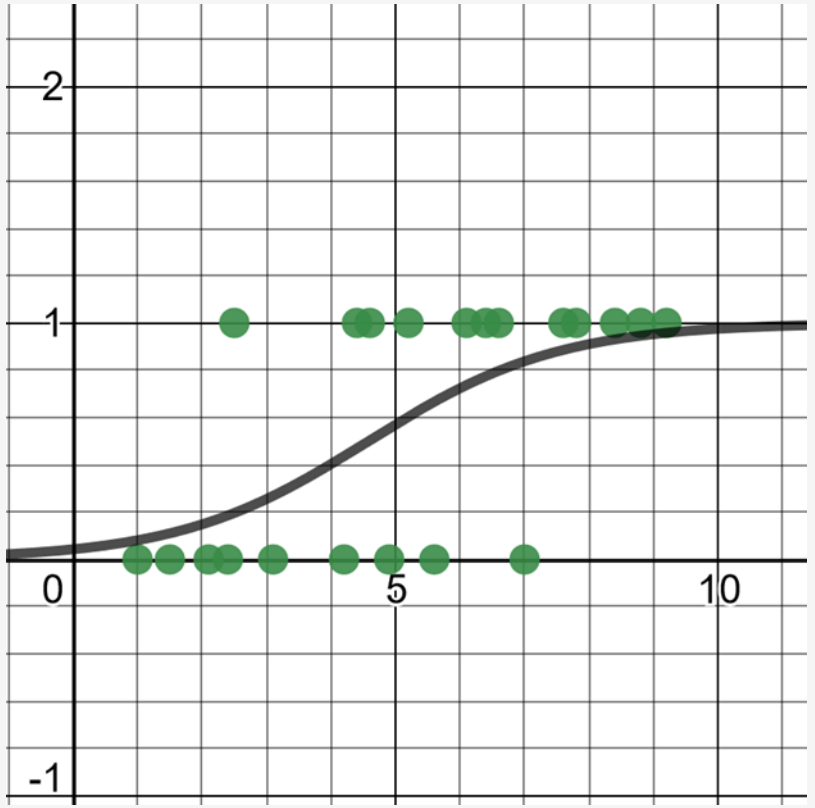
Logistic Regression

Logistic Regression is a probability tool that predicts the probability an event will happen given one or more variables.

Training data must have outcomes of 0 (false) or 1 (true), but the regression will output a probability value between 0 and 1.

An S-shaped curve (a **logistic function**) is fit to the points and then used to predict probability.

If a predicted value (the y-axis) is less than .5 it is typically categorized as false (0), and if the predicted value is greater than/equal to .5 it is typically categorized as true (1).



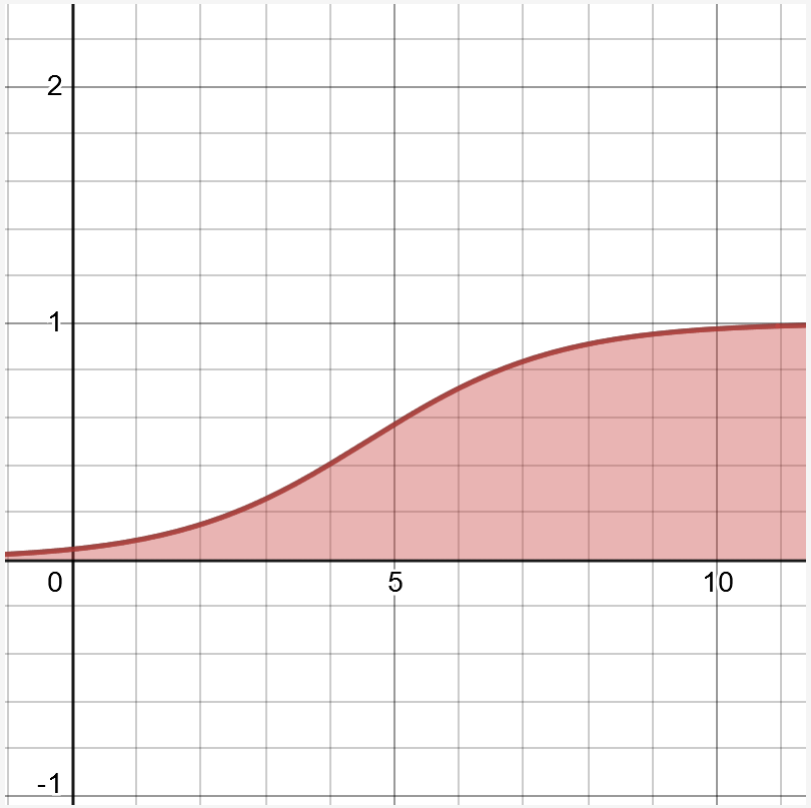
https://www.desmos.com/calculator/bsqtqfians

Is a Logistic Function a Probability Distribution?

Discussion: Is a Logistic Function a Probability Distribution? Why or why not?

It is not! It is a different probability tool.

- The area under the entire logistic function curve is not 1.0, but infinite, so it cannot be a probability distribution.
- It is used to predict a binary event occurring or not occurring by outputting a single probability value.
- It behaves more like a cumulative distribution function, showing the accumulating probability across a range of values (it actually is a CDF for the logistic distribution, which is similar to the normal distribution).



The logistic function is worth learning as it pops up a lot in machine learning and statistics.

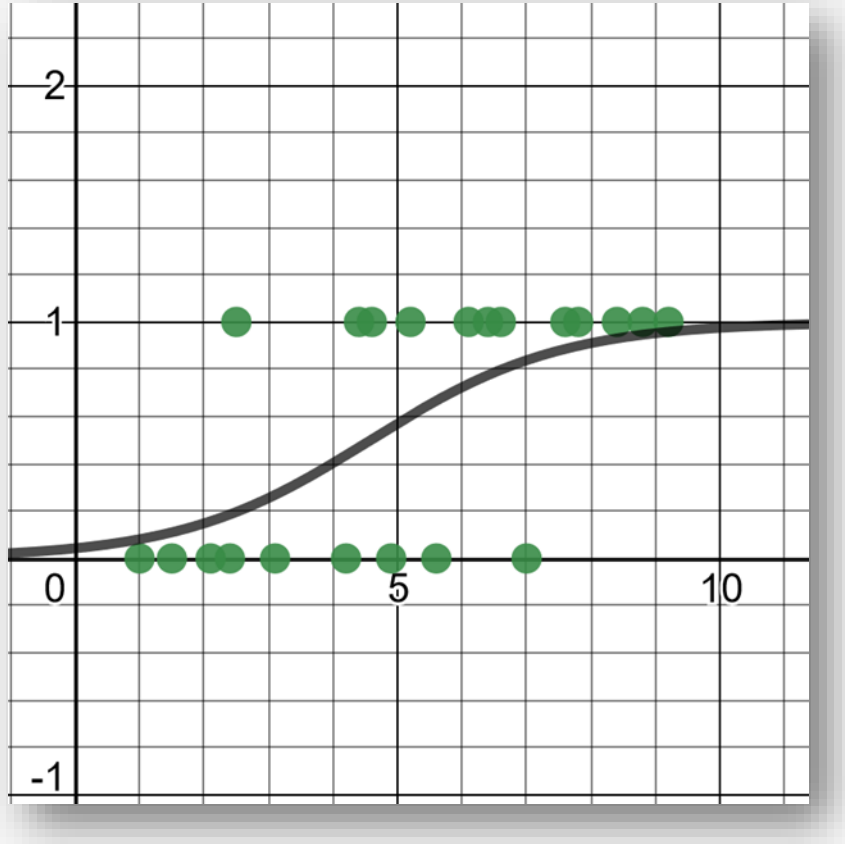
Just Show Me the Math!

For a single independent variable x to predict a dependent probability variable y, you need to fit the data to the logistic function:

$$y = \frac{1.0}{1.0 + e^{-(\beta_0 + \beta_1 x)}}$$

You can express this in Python as:

```
def predict_probability(x):
    p = 1.0 / (1.0 + math.exp(-(b0 + b1 * x)))
    return p
```



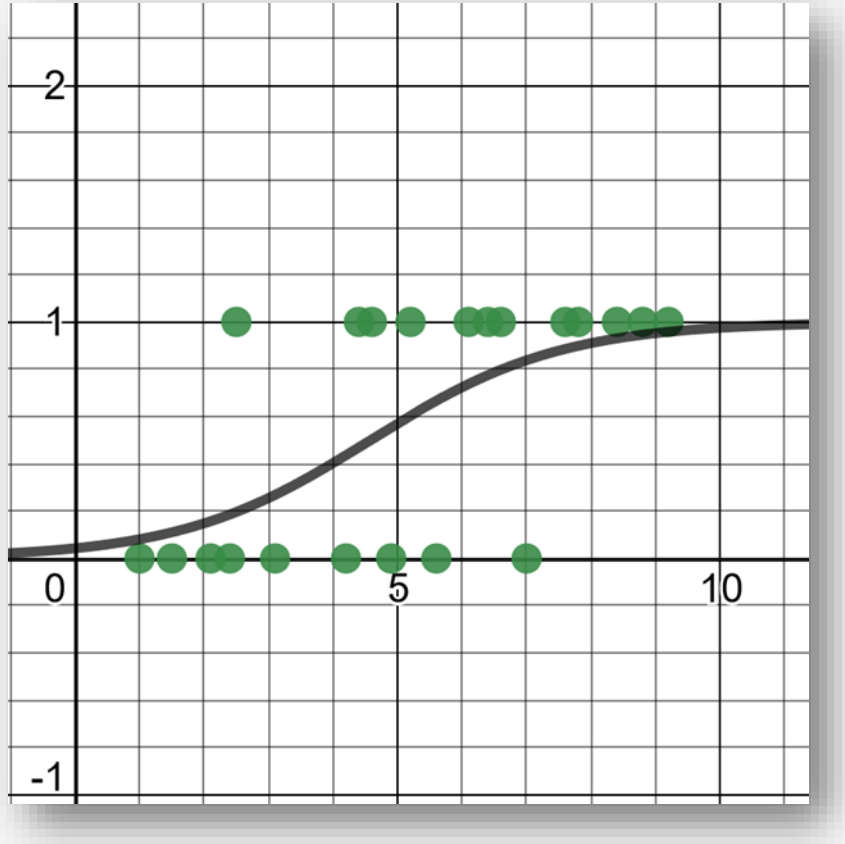
https://www.desmos.com/calculator/blfcwrlnuw

Just Show Me the Math!

You may see also see this logistic function expressed as:

$$y = \frac{e^{\beta_0+\beta_1 x}}{1.0 + e^{\beta_0+\beta_1 x}}$$

However it is the same and just algebraically expressed differently.



https://www.desmos.com/calculator/blfcwrlnuw

Just Show Me the Math!

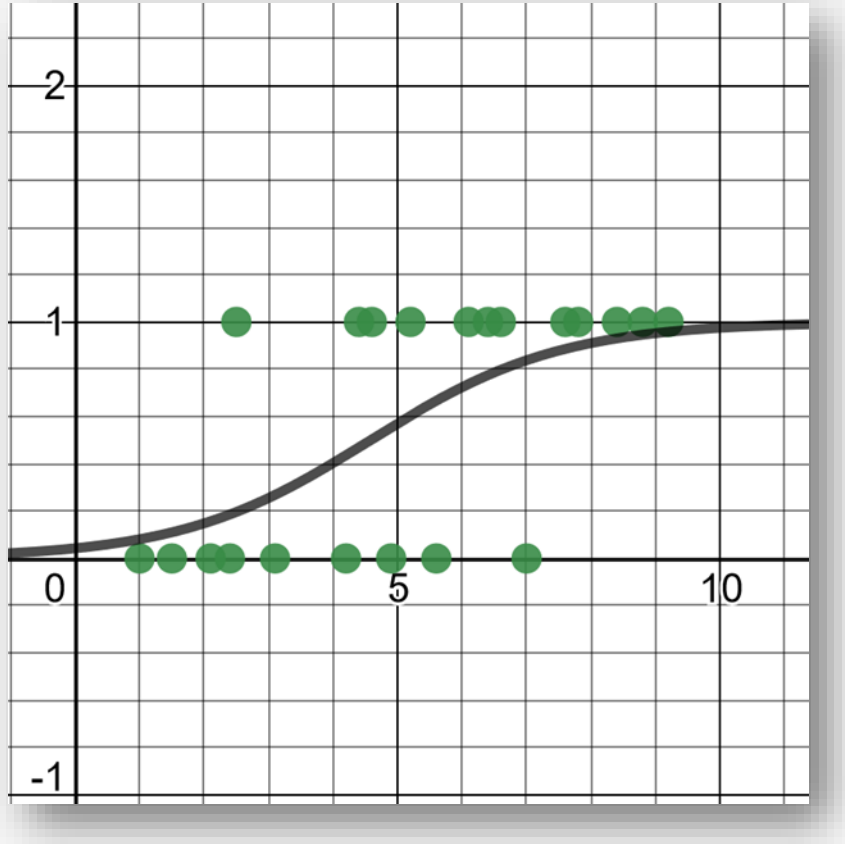
$$y = \frac{1.0}{1.0 + e^{-(\beta_0 + \beta_1 x)}}$$

```
def predict_probability(x):
    p = 1.0 / (1.0 + math.exp(-(b0 + b1 * x)))
    return p
```

Notice the expression $\beta_0 + \beta_1 x$ is linear, and this known as the **log odds function** which is translated logarithmically into a probability.

In the interest of time, we will avoid going into proofs and mathematical details about how this function works.

Just know it produces this S-shaped curve we need to output a probability between 0 and 1.



https://www.desmos.com/calculator/blfcwrlnuw

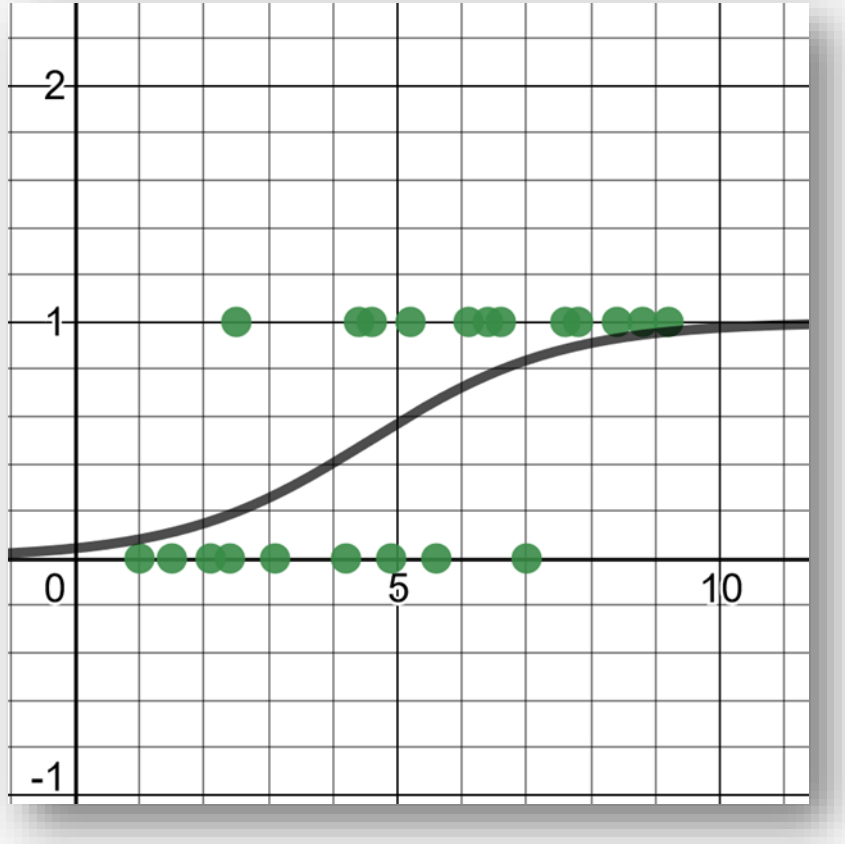
Just Show Me the Math!

$$y = \frac{1.0}{1.0 + e^{-(\beta_0 + \beta_1 x)}}$$

```
def predict_probability(x):
    p = 1.0 / (1.0 + math.exp(-(b0 + b1 * x)))
    return p
```

We need to solve for β_0 and β_1, but we cannot use least squares like in linear regression.

- We are trying to maximize probability of the curve predicting correctly, not finding the best fit.
- Using hill climbing, we need to find β_0 and β_1 that produces the maximum likelihood.



https://www.desmos.com/calculator/blfcwrlnuw

Maximum Likelihood

Maximum likelihood is a technique to estimate parameters that have the highest probability of outputting the observed data.

In our case, we need to find values for β_0 and β_1 that will yield the highest likelihood of outputting the correct true/false values.

Remember that our logistic function outputs a probability y for a given value x.

$$y = \frac{1.0}{1.0 + e^{-(\beta_0 + \beta_1 x)}}$$

Maximum Likelihood

Let's say during our hill-climbing, we test parameters $\beta_0 = -.3.17$ and $\beta_1 = 0.69$

$$y = \frac{1.0}{1.0 + e^{-(-3.17+0.69x)}}$$

To calculate the total likelihood for these parameters…

1. Get the true (1) data, calculate the probability $\boldsymbol{y}$ for each $\boldsymbol{x}$ value, and multiply them together.

2. Get the false (0) data, calculate the probability $(\boldsymbol{1.0 - y})$ for each $\boldsymbol{x}$ value, and multiply them together.

3. Multiply the two products above together, and that is your total likelihood.

Maximum Likelihood – Avoiding Floating Point Underflow

However, multiplying this many decimals together can cause floating point underflow.

With a clever mathematical hack, we can remedy this by using logarithmic addition instead of multiplication.

$$y = \log\left(\frac{1.0}{1.0 + e^{-(-3.17+0.69x)}}\right)$$

If you need to learn about logarithms, YouTube is the best place to get crash coursed.

PatrickJMT: https://youtu.be/AAW7WRFBKdw

Don't Memorize: https://youtu.be/4UNkQcBrLaQ

Maximum Likelihood – Avoiding Floating Point Underflow

$$y = \log\left(\frac{1.0}{1.0 + e^{-(-3.17+0.69x)}}\right)$$

To calculate the total likelihood for these parameters, but avoid floating point underflow…

1. Get the true (1) data, calculate the probability $\boldsymbol{y}$ for each $\boldsymbol{x}$ value, pass it to a $\boldsymbol{log}()$ function, then sum the values.

2. Get the false (0) data, calculate the probability $(\boldsymbol{1.0 - y})$ for each $\boldsymbol{x}$ value, pass it to a $\boldsymbol{log}()$ function, then sum the values.

3. Sum the two values above together, pass it to the $\exp()$ function to undo the logarithm, and that is your total likelihood.

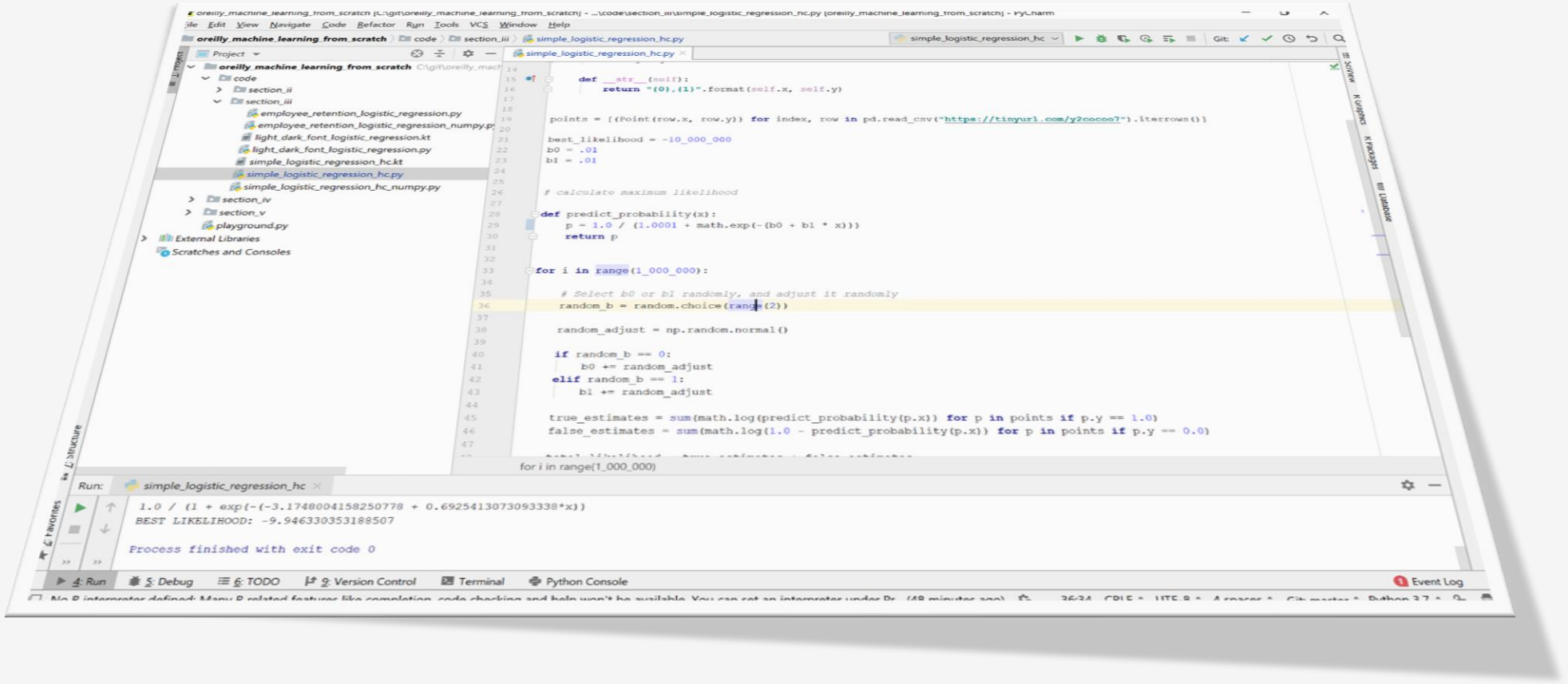
Maximum Likelihood

We now know how to calculate the likelihood for a given set of parameters.

To calculate the maximum likelihood…

1. Randomly adjust the β_0 and β_1 values (using hill-climbing, gradient descent, or other optimization)
2. Calculate the likelihood (as shown in the previous slide)
3. If the likelihood improves, keep the changes to β_0 and β_1 , otherwise revert.
4. Do this for as many iterations as necessary, until the likelihood stops improving.

Hands-On: Logistic Regression



Multivariable Logistic Regression

We can easily extend logistic regression to handle multiple independent variables, simply by adding more β_x variables for each additional variable.

We then solve for those β_x variables the same way as before.

$$y = \frac{1.0}{1.0 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots + \beta_n x_n)}}$$

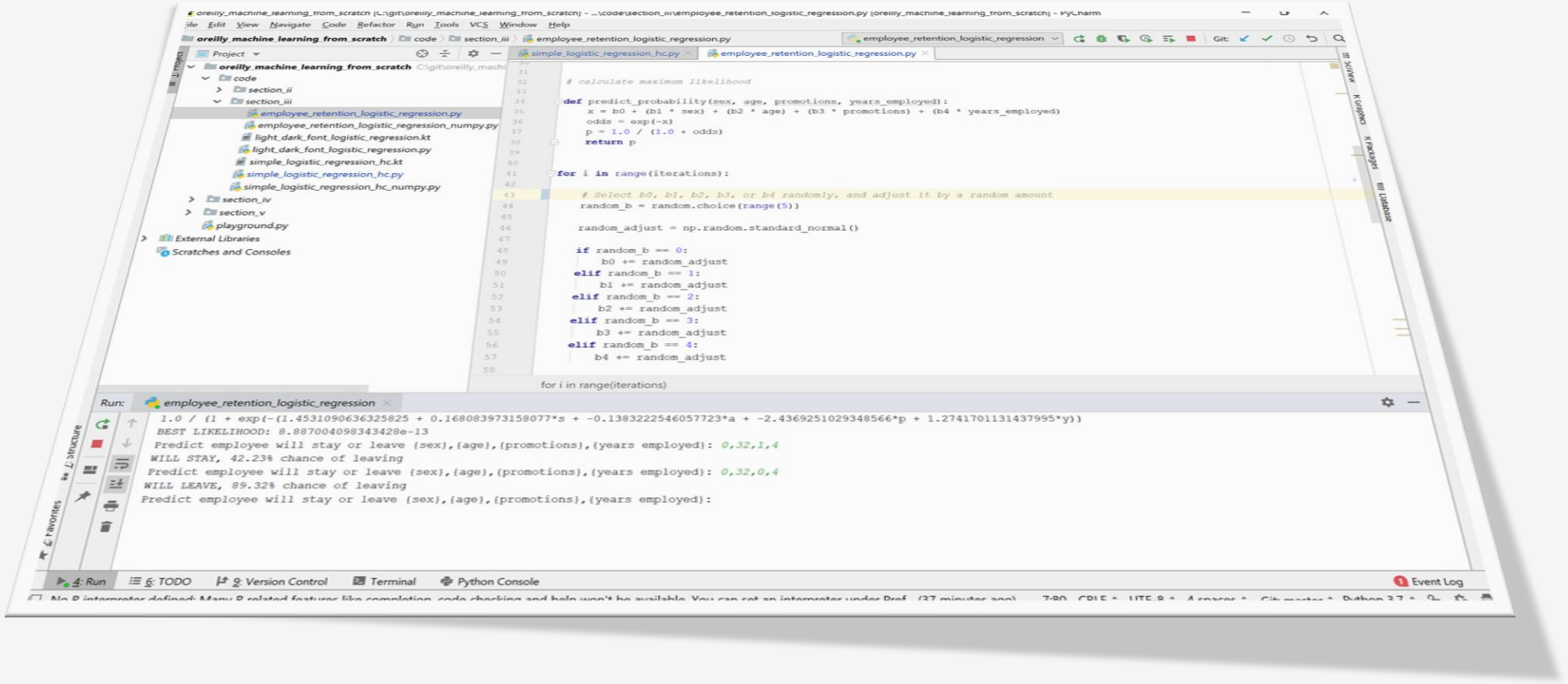
Multivariable Logistic Regression

We have some historical employee data (https://tinyurl.com/y6r7qjrp) and want to use it to predict whether an employee will quit or not.

SEX, AGE, $PROMOTIONS$, and $YEARS\_EMPLOYED$ are the predictor variables, and DID_QUIT is the outcome variable where 1 = true and 0 = false.

SEX	AGE	PROMOTIONS	YEARS_EMPLOYED	DID_QUIT
1	43	4	10	0
1	38	3	8	0
1	44	4	11	1
0	41	2	6	0
1	45	2	6	1
0	36	3	9	0
1	33	1	3	0
1	44	3	10	0
…				

Hands-On: Multivariable Logistic Regression



Using Logistic Regression for Classification

Logistic regression may seem limited in that it only supports two categories: true (1) or false (0).

But you can make it support any number of categories!

To use logistic regression for more than two categories:

1. Build a separate logistic regression for each category, where a given item belongs to that category (1) or doesn't belong to that category (0).

2. To predict which category an item belongs to, pass it through each category's logistic regression, and choose the one with the highest probability.

Quiz Time!

Logistic regression will only output a probability between 0 and 1 that an event will happen.

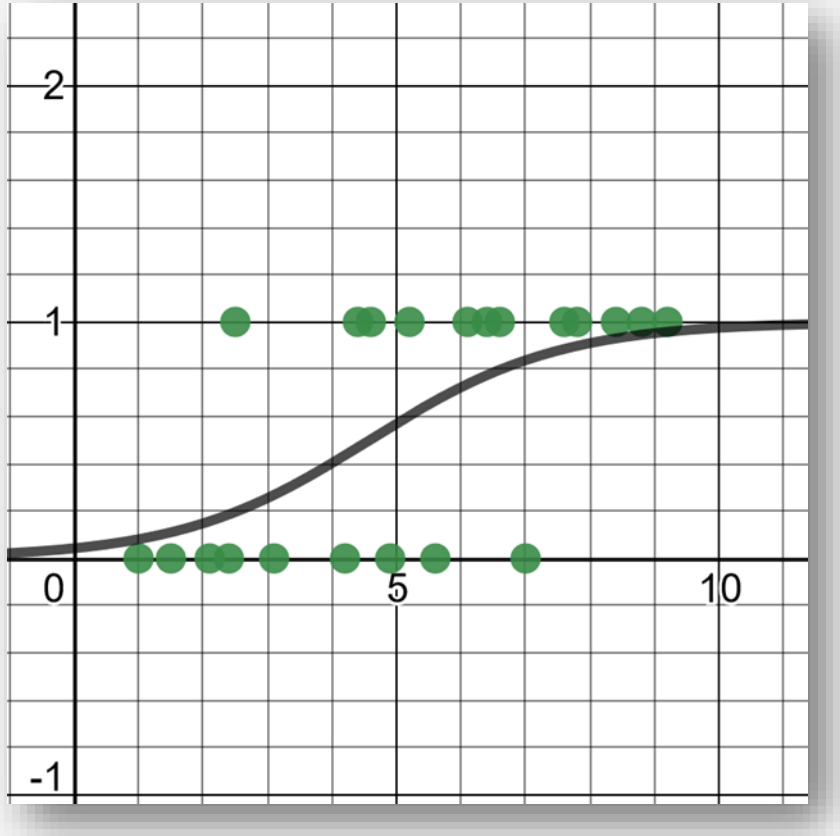
1) True
2) False

Quiz Time!

Logistic regression will only output a probability between 0 and 1 that an event will happen.

1) True
2) False

Like any probability model, logistic regression will output a percentage between 0 and 1



Quiz Time!

Logistic regression cannot be used for more than two categories.

A) True

B) False

Quiz Time!

Logistic regression cannot be used for more than two categories.

A) True

B) False

Logistic regression can support more than one category by doing a separate logistic regression for each category, and predicting the one that yields the highest probability.

Resources

3Blue1Brown:
https://www.youtube.com/c/3blue1brown

Bayesian Statistics the Fun Way (No Starch):
https://learning.oreilly.com/library/view/bayesian-statistics-the/9781098122492/

Data Science from Scratch (O'Reilly):
https://learning.oreilly.com/library/view/data-science-from/9781492041122/

Think Bayes (O'Reilly):
https://learning.oreilly.com/library/view/think-bayes/9781491945407/

Think Stats (O'Reilly):
https://learning.oreilly.com/library/view/think-stats-2nd/9781491907344/